

DCWriter 结构化文本编辑器

第五代 Web 版编程参考手册

南京都昌信息科技有限公司

2024-03-01

接口更新记录

修改时间	外发版本验证	修改接口	维护接口者
2024-09-05	2024-09-05	InsertSubDocuments 新增参数	李应祥
2024-09-02	2024-09-06	FocusElementById 新增参数	李应祥
2024-06-24	2024-06-24	文档选项 BehaviorOptions.RemoveLastParagraphFlagWhenInsertDocument	李应祥
2024-06-14	2024-06-14	文档选项： BehaviorOptions.AutoTranslateSourceString	李应祥
2024-06-14	2024-06-14	文档选项： BehaviorOptions.AutoTranslatedDescString	李应祥
2024-06-13	2024-06-13	文档选项： BehaviorOptions	李应祥

		.EnableExpression	
2024-06-13	2024-06-13	文档选项: BehaviorOptions .OutputFieldBorderTextToContentText	李应祥
2024-06-05	2024-06-13	命令: HeaderBottomLineVisible	李应祥

1. 引言

1.1. 目的

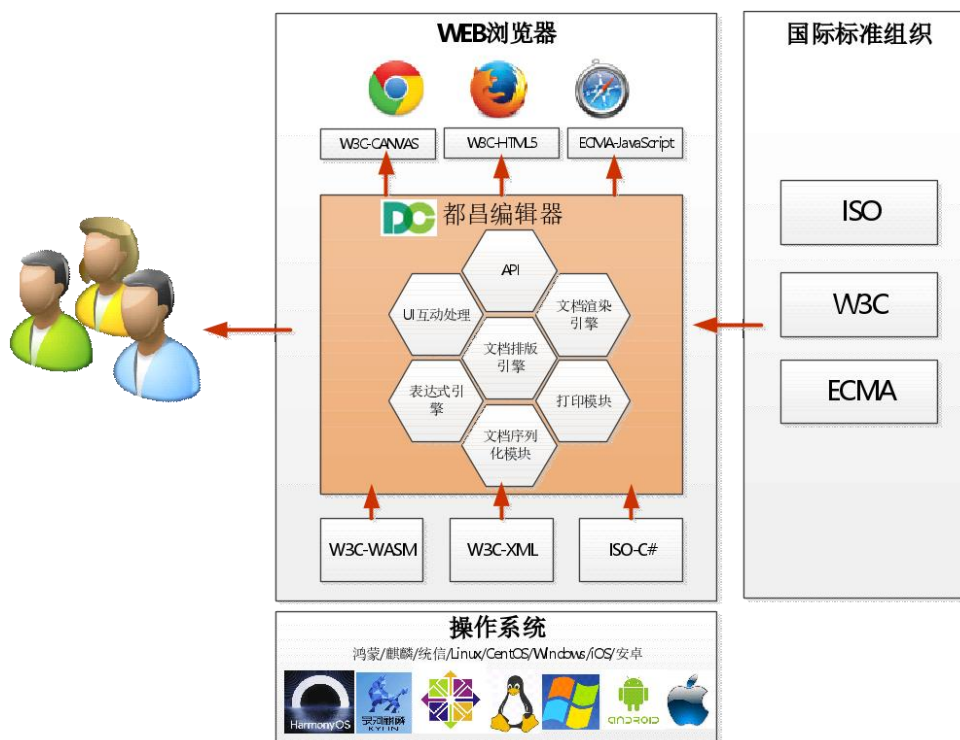
本文档说明了 DCWriter 结构化文本编辑器第五代 Web 版本的接口声明、参数及使用方法、使用方法的，主要提供有软件编程能力且对各行业软件系统功能熟悉的开发人员进行扩展功能开发使用。

1.2. 系统概述

DCWriter 结构化文本编辑器软件是南京都昌信息科技有限公司自主研发的新一代电子病历编辑器产品，不依赖其他任何第三方组件，不与数据库进行直接关联，对软件开发商的系统结构和数据库没有要求，不依赖 RichTextBox、MicroSoft Word、WPS 等软件，拥有完全自主知识产权。

它是一款基于 WASM-HTML5 的软件产品，是一种跨平台的 WEB 客户端 GUI 应用软件，经过仔细测试，它既可以运行在 Windows、Linux、IOS、安卓等传统操作系统中，也可以运行在统信、麒麟、鸿蒙、欧拉等多种国产操作系统中，符合当前的信创要求。

以下是它的技术架构图：



对这个软件技术架构图有以下说明：

第一，选择国际标准组织。

我们首先选择依赖于 ISO、W3C、ECMA 等中立的国际标准组织，这些国际标准组织包含了很多中国实体。比如：中国政府是 ISO 的常任理事国；W3C 的 4 个联合总部中有一个在北京航空航天大学，负责人是中国工业和信息化部副部长；ECMA 就包括 360、紫晶存储等众多中国企业。中方在这些国际标准组织中具有很大的投票权，领导着不少细分领域的发展。

第二，遴选底层技术平台。

我们从这些国际标准组织产生的立场中立的国际技术标准中进行遴选，选取了 W3C-WASM、W3C-HTML5、W3C-CANVAS、ECMA-JavaScript、ISO-C#、W3C-XML 作为我们的底层技术平台。

W3C-WASM 技术全称 W3C 制定的 WebAssembly 技术标准。它是一种基于堆栈虚拟机的二进制指令格式，是一种比较新的原生云技术，它运行效率高，跨平台。Docker 联合创始人 Solomon Hykes 曾表示：如果 WASM 在 2008 年就存在了，就没有必要发明 Docker 了。编辑器中存在复杂的文档排版算法，因此我们选择 W3C-WASM 在浏览器前端执行这些复杂运算。

ISO-C# 技术虽然是微软发明的，但已经提交到 ISO 成为国际标准，C# 使用的 .NET 在 2014 年就开源了，并由非盈利机构《.NET 基金会》管理。因此使用 ISO-C# 及 .NET 技术不存在任何法律、经济和技术风险。

另外陈述相关事实：JAVA 不是国际标准。它是美国 SUN 公司发明的，后来 ORACLE 公司

收购 SUN 公司而获得 JAVA 的版权,然后起诉谷歌公司侵犯了 JAVA 的版权而索赔 90 亿美元,这拉锯战的官司一打就是十年。

第三,指定目标操作系统。

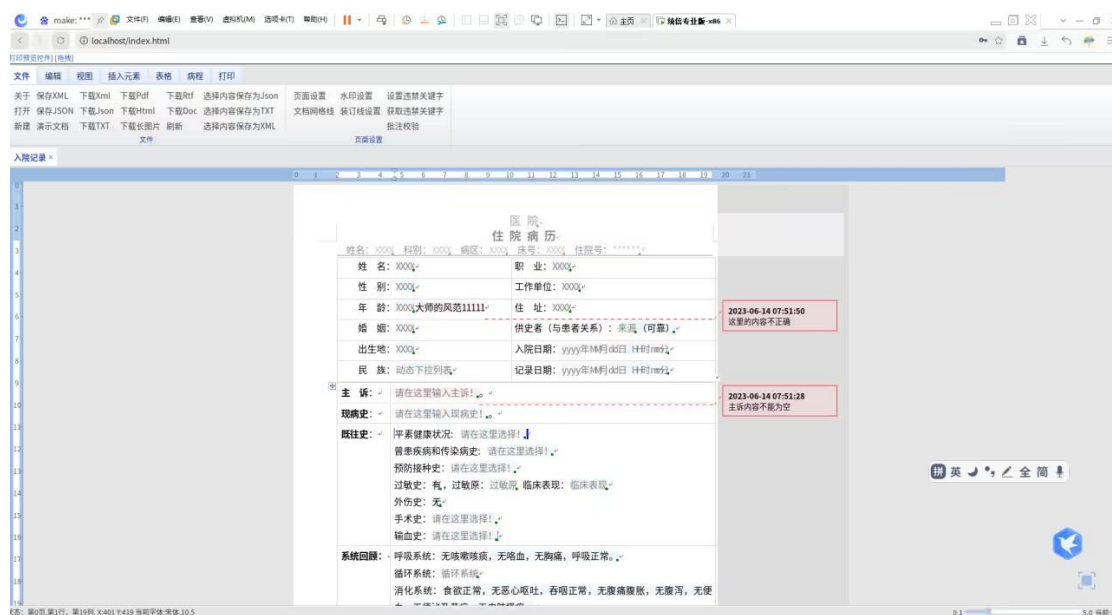
在目标运行操作系统方面,我们自己内部已经完成了对鸿蒙、银河麒麟、统信、Linux、CentOS、Windows、安卓、IOS 等多种操作系统的运行测试。并在客户的生产环境中实际使用的多种操作系统中运行稳定。

第四,跨平台开发。

在开发过程中,我们坚持跨平台的底线,不调用特定平台提供的 API,使得软件可以在各种运行环境中保持一致软件行为。

本产品是一个运行在 WEB 浏览器中的纯前端的软件组件,对此我们进行了很细致的浏览器兼容性测试。确保软件在各种主流浏览器中保持一致的用户体验,所有输出结果都符合预期。

例如下图就是软件在统信操作系统中的运行效果:



第五,硬件兼容性。

在医院应用中,运行软件的计算机硬件多种多样。我们针对常见的计算机屏幕和打印机进行兼容性测试和调整。例如针对普通屏幕、高分屏幕、超宽屏幕进行了优化处理。在保持一致的用户体验时,尽量充分的利用新硬件的能力,提高软件解决行业问题的能力。

本软件对软件开发商的系统结构和数据库没有要求,可运行在各类浏览器中,包含谷歌、火狐、IE、MS Edge 等。客户端跨平台运行,可运行在各类操作系统中,包含 Windows、Android、IOS、Linux 等。服务器支持 Windows 系统、Linux 系统及衍生系统。

文档采用 XML 技术存储数据，支持 W3C XML 标准，和 DCWriter 电子病历编辑器 Winform 版无缝衔接；本软件符合卫生部《电子病历系统功能规范（试行）》中对病历编辑器的 41 个必需功能和推荐功能，提供输入域、单选框、复选框、医学表达式、表格等，实现结构化要求。

软件服务端部署和更新简便快捷。客户端支持各种框架开发，包括 ASP.NET、CORE、MVC、JAVA、Vue、Angular、React 等。

软件既可用于医疗软件中的电子病历开发，又可应用于普通的办公软件应用程序中；提供多语言应用，包含繁体版、维文版、藏文版，并提供其他语言的定制开发。

DCWriter 可将病历中的内容或部分内容保存为 XML 文件，也能导出 RTF、PDF、HTML、JSON 等各种开放文档格式。支持开发者自定义对数据进行加密保存。

DCWriter 提供给医护人员与微软 Word 相似的操作体验，降低电子病历开发商的实施成本，提高医护人员的学习接受度。

1.3. 约定

在每个“代码示例”中，“ctl”代表的 DCWriter 控件对象的默认实例名称。

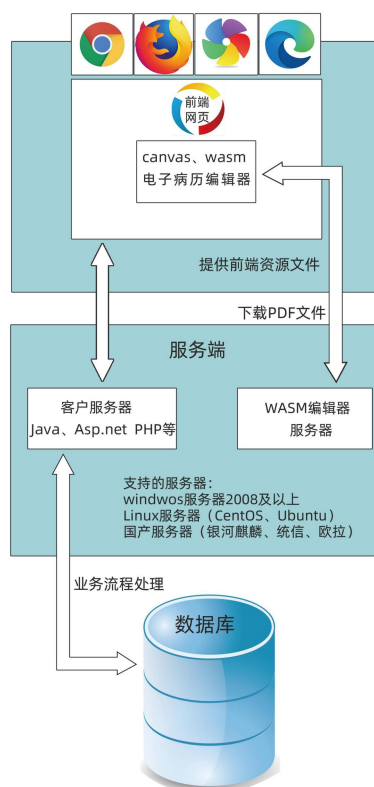
如无特殊说明，以下代码使用 js 代码做参考实例。

NativeHandle 对象：结构化元素的唯一标识，当文档中出现多个 ID 相同的元素，请用该属性来进行功能实现。

内部引用对象：暂缺

1.4. 环境说明

使用 DCWriter 第五代进行开发时，软件的架构如下：



项目主服务器是用户的项目本身需要的 WEB 服务器，可以是 IIS、Tomcat、Apache 等各种服务器及架构，由客户的业务系统自主决定。

DCWriter 服务器为 DCWriter 服务程序，主要提供控件初始化资源文件、并发处理、PDF 的处理等。它支持部署在 Windows 服务器上，还支持在 .Net Core 环境下部署在 Linux 及其衍生服务器、及国产服务器上。

项目主服务器和 DCWriter 服务器既可以部署在一个服务器上，也可以分别部署在不同的服务器上。

说明：服务端部署在 Linux 下，或国产操作系统上时，需要在服务端安装 .Net Core。

如果不需要 PDF 处理的功能，也可以不需要安装 .Net Core，通过 JAVA 开发服务端。

2. 入门指南

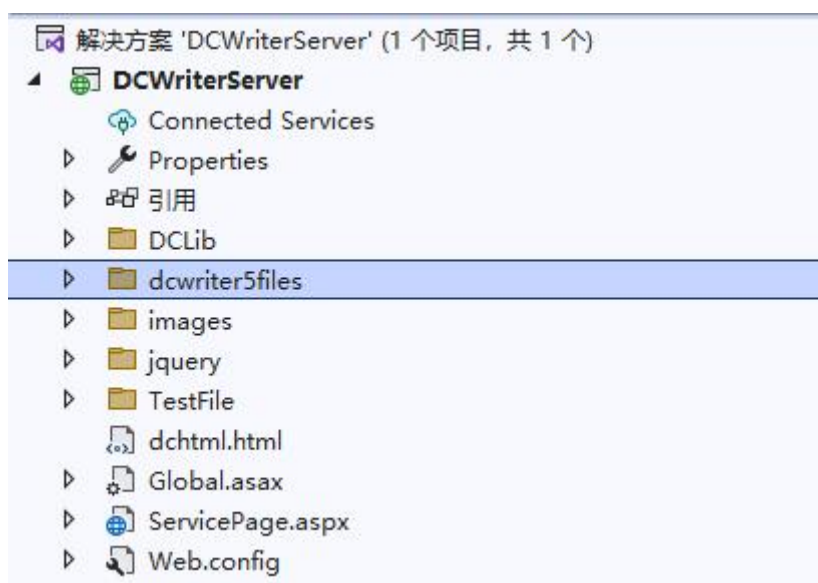
2.1. 服务端开发

服务端开发支持 ASP.NET 服务端、ASP.NET MVC 服务端、ASP.NET CORE 服务端、JAVA


```
        return;  
    }  
}  
}
```

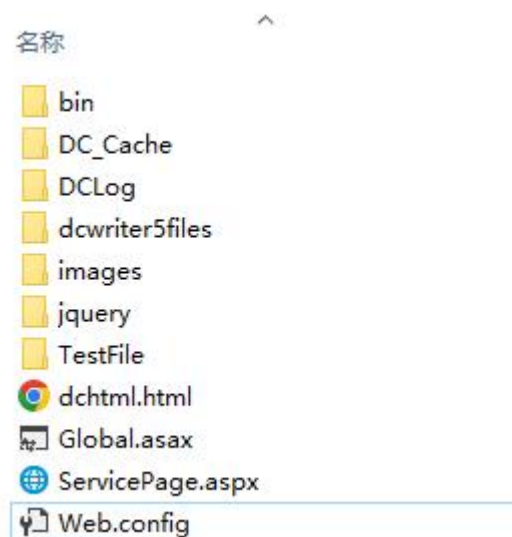
其中，“WebWriterControlServicePageExecuter”是编辑器控件的服务器页面功能执行器，它是前端和服务器数据和接口交换的核心，这是内部处理，客户无需关注内部实现。

4. 在项目下新建文件夹“dcwriter5files”，把从都昌公司获取的资源包放在该文件夹中。此时项目的结构如下：



调试运行该项目，在浏览器中访问该路径，如：
<https://myserver/ServicePage.aspx?wasmres=dcwriter5.js>，可以下载JS文件，表示部署完成。

5. 发布该项目，在 IIS 上部署即可。此时发布并部署的文件夹结构如下：



注意：在 IIS 上部署时，由于资源包的文件类型，此时需要设置 IIS 站点的 MIME 类型，需要支持 “.br”、“.dat”、“.js”、“.json”、“.wasm”、“.gz”、“.dll”。

2. 1. 2. ASP.NET MVC 服务端

1. 新建 ASP.NET MVC 项目，在项目中添加引用 “DCSoft.Writer.ForASPNET.dll”。
2. 在 MVC 项目中添加类，如 “DCWriterServer”，它的代码如下图所示

```
public class DCWriterServer{  
    private WebWriterControlEngine dcWControl;  
    public WebWriterControlEngine DCWControl{  
        get { return dcWControl; }  
        set { dcWControl = value; }  
    }  
    public DCWriterServer() {  
    }  
    public WebWriterControlEngine Init() {  
        DCWControl = new WebWriterControlEngine();  
        DCWControl.SetRegisterCode("注册码");  
        DCWControl.Options.ControlName = "myWriterControl";//生成前端  
        标签时的 ID  
        DCWControl.Options.ServicePageURL =  
        "/控制器/MoreHandleDCWriterServicePage";  
    }  
}
```

```
        return DCWControl;
    }
}
```

其中，“Options.ControlName”是生成前端标签时的 ID，前端 JS 通过该 ID 可以找到控件；“Options.ServicePageURL”是服务路径的地址，有对应的控制器方法。

3. 根据上一步的“Options.ServicePageURL”添加对应的控制器方法，如下所示：

```
public class MyServerController : Controller{
    [ValidateInput(false)]
    public ActionResult MoreHandleDCWriterServicePage() {
        return new MoreActionResult();
    }
    private class MoreActionResult : ActionResult{
        public MoreActionResult() {
        }
        public override void ExecuteResult(ControllerContext context){
            WebWriterControlServicePageExecuter executer =
new WebWriterControlServicePageExecuter();
            executer.CrossDomainSettings.Enabled = true;//是否跨域
            if (executer.HandleService()){
                return;
            }
        }
    }
}
```

4. 在项目下新建文件夹“dcwriter5files”，把从都昌公司获取的资源包放在该文件夹中。调试运行该项目，在浏览器中访问该路径，如：<https://MyServer/MoreHandleDCWriterServicePage?wasmres=dcwriter5.js>，可以下载 JS 文件，表示部署完成。

5. 发布该项目，在 IIS 上部署即可。

2.1.3. ASP.NET CORE 服务端

1. 新建 ASP.NET CORE MVC 项目，在项目中添加引用“DCSoft.Writer.ForASPNETCore.dll”。

2. 在 MVC 项目中添加类，如“DCwriterServer”，它的代码如下图所示：

```
public class DCwriterServer{
    private WebWriterControlEngine dcWControl;
```

```

        public WebWriterControlEngine DCWControl{
            get { return dcWControl; }
            set { dcWControl = value; }
        }
//此处 ASP.NET MVC 有区别
        private Microsoft.AspNetCore.Http.IHttpContextAccessor _accessor;
        public Microsoft.AspNetCore.Http.IHttpContextAccessor Accessor{
            get { return _accessor; }
            set { _accessor = value; }
        }
        public DCWriterServer() {

        }

        public WebWriterControlEngine Init() {
            DCWControl = new WebWriterControlEngine(Accessor.HttpContext); //此处
//处 ASP.NET MVC 有区别
            DCWControl.SetRegisterCode("注册码");
            DCWControl.Options.ControlName = "myWriterControl"; //生成前端标签时的 ID
            DCWControl.Options.ContentRenderMode =
WebWriterControlRenderMode.NormalHtmlEditable;
            DCWControl.Options.ServicePageURL =
"/控制器/MoreHandleDCWriterServicePage";
            return DCWControl;
        }
    }
}

```

其中，“Options.ControlName”是生成前端标签时的 ID，前端 JS 通过该 ID 可以找到控件；“Options.ServicePageURL”是服务路径的地址，有对应的控制器方法。与 ASP.NET MVC 的区别是需要添加 IHttpContextAccessor 的接口。

3. 根据上一步的“Options.ServicePageURL”添加对应的控制器方法，如下所示：

```

public class MyServerController : Controller{
    private Microsoft.AspNetCore.Http.IHttpContextAccessor _accessor;
    private Microsoft.AspNetCore.Http.HttpContext httpcontext;
    private readonly Microsoft.AspNetCore.Hosting.IWebHostEnvironment
_webHostEnvironment;
    public
MyServerController(Microsoft.AspNetCore.Http.IHttpContextAccessor accessor,
Microsoft.AspNetCore.Hosting.IWebHostEnvironment webHostEnvironment) {
        _accessor = accessor;
        _webHostEnvironment = webHostEnvironment;
    }
    public ActionResult MoreHandleDCWriterServicePage() { //服务路径

```

```

        return new MoreActionResult();
    }
    private class MoreActionResult : ActionResult{
        public MoreActionResult() {
        }
        public override void ExecuteResult(ActionContext context){
            DCSoft.Writer.Controls.WebWriterControlServicePageExecuter
executer = new DCSoft.Writer.Controls.WebWriterControlServicePageExecuter();
            executer.CrossDomainSettings.Enabled = true;
            executer.EnabledCrossDomain = true;
            if (executer.HandleService(context.HttpContext)){
                return;
            }
        }
    }
}
}
}

```

4. 在项目下新建文件夹“dcwriter5files”，把从都昌公司获取的资源包放在该文件夹中。调试运行该项目，在浏览器中访问该路径，如：<https://MyServer/MoreHandleDCWriterServicePage?wasmres=dcwriter5.js>，可以下载 JS 文件，表示部署完成。

5. 发布该项目，在 IIS 上部署即可。

2.1.4. JAVA 服务端

如果服务端不处理 PDF 和并发，此时的服务端只负责前端资源的下载，用户可使用 java 自行开发服务端下载资源包。可参考以下代码进行：

```

public static class DCWriter5FileDownloadForASPNET
{
    /// <summary>
    /// 第五代编辑器程序文件所在目录名,若未指定,则默认为发布目录下的
dcwriter5files 子目录下。
    /// </summary>
    public static string WorkDirectory
    {
        get
        {
            return _WorkDirectory;
        }
        set
        {
            _WorkDirectory = value;
        }
    }
}

```

```
    }
}
/// <summary>
/// WASM 资源文件所在目录
/// </summary>
private static string _WorkDirectory = null;
@"D:\DCSoft.WASM.Publish\Publish\wwwroot\_framework";
public static string MainFileName = "dcwriter5.js";
#if DCWriterForASPNET
private static bool _UseAddResponseHeader = false;
private static void SetResponseHeader(System.Web.HttpResponse response,
string name, string v)
{
    if (_UseAddResponseHeader)
    {
        response.AddHeader(name, v);
    }
    else
    {
        try
        {
            response.Headers.Set(name, v);
        }
        catch (System.PlatformNotSupportedException
ext)//System.PlatformNotSupportedException: “此操作要求使用 IIS 集成管线模式。”
        {
            System.Diagnostics.Debug.WriteLine(ext.Message);
            _UseAddResponseHeader = true;
            response.AddHeader(name, v);
        }
    }
}

private static void WriteText(System.Web.HttpResponse response, string
txt, bool bolIsJavaScript = true)
{
    if (bolIsJavaScript)
    {
        response.ContentType = "text/javascript;charset=utf-8";
        txt = "window.alert(" +
DCSoft.Design.DesignUtils.StringToCSharpValueString(txt) + ")";
    }
    else
    {
        response.ContentType = "text/plain;charset=utf-8";
    }
}
```

```

        //SetResponseHeader(response, "Content-Type",
"text/plain;charset=utf-8");
    }
    response.Write(txt);
}
#elif DCWriterForASPNETCore
    private static async void WriteText(
        Microsoft.AspNetCore.Http.HttpResponse response,
        string txt,
        bool bolIsJavaScript = true )
    {
#if DCWriterForASPNET

response.Cache.SetCacheability(System.Web.HttpCacheability.NoCache);
#elif DCWriterForASPNETCore
        SetResponseHeader(response, "Cache-Control", "no-cache");
#endif

        if (bolIsJavaScript)
        {
            SetResponseHeader(response, "Content-Type",
"text/javascript;charset=utf-8");
            txt = "window.alert(" +
DCSoft.Design.DesignUtils.StringToCSharpValueString(txt) + ")";
        }
        else
        {
            SetResponseHeader(response, "Content-Type",
"text/plain;charset=utf-8");
        }
        var bs9 = System.Text.Encoding.UTF8.GetBytes(txt);
        await response.Body.WriteAsync(bs9, 0, bs9.Length);
    }
    private static bool _UseAddResponseHeader = false;
    private static void
SetResponseHeader(Microsoft.AspNetCore.Http.HttpResponse response, string name,
string v)
    {
        if (response.HasStarted == false)
        {
            response.Headers[name] = v;
        }
    }
#endif
    private static string _WASMApVersion = null;

```

```
public static string GetWASMApVersion()
{
    if (_WASMApVersion == null)
    {
        // 根据主文件的修改时间来设置程序版本号
        lock (_CacheFileInfos)
        {
            var fn9 = Path.Combine(_WorkDirectory, MainFileName);
            if (File.Exists(fn9))
            {
                var bs2 = File.ReadAllBytes(fn9);
                var bsKey = 0;
                for (var iCount = 0; iCount < bs2.Length; iCount++)
                {
                    bsKey += bs2[iCount];
                }
                bsKey = Math.Abs(bsKey % 10000);
                _WASMApVersion =
File.GetLastWriteTime(fn9).ToString("yyyyMMddHHmmss") + "_" + bsKey.ToString();
            }
        }
    }
    return _WASMApVersion;
}

private static readonly string _EmptyString =
DateTime.Now.Ticks.ToString();
/// <summary>
/// 执行操作
/// </summary>
/// <returns>true:执行成功, 无需后续处理。false:执行不成功, 继续执行后
续处理。</returns>
#if DCWriterForASPNET
    public static bool Execute(System.Web.HttpRequest request,
System.Web.HttpResponse response)
#elif DCWriterForASPNETCore
    public static async System.Threading.Tasks.Task<bool> Execute(
        Microsoft.AspNetCore.Http.HttpRequest request,
        Microsoft.AspNetCore.Http.HttpResponse response,
        Microsoft.AspNetCore.Http.HttpContext context)
#endif
{
    if (request == null)
    {
        throw new ArgumentNullException("request");
    }
}
```



```
    }
    if (response == null)
    {
        throw new ArgumentNullException("response");
    }
#endif DCWriterForASPNET
    var strResName = request.QueryString["wasmres"];
    var strQueryVer = request.QueryString["ver"];
    var strFlag = request.QueryString["flag"];
    var strWASMRootPath = request.QueryString["wasmrootpath"];
#elif DCWriterForASPNETCore
    string strResName = null;
    string strQueryVer = null;
    string strFlag = null;
    string strWASMRootPath = null;
    foreach (var item in request.Query)
    {
        if (item.Key == "wasmres")
        {
            strResName = item.Value;
        }
        else if (item.Key == "ver")
        {
            strQueryVer = item.Value;
        }
        else if (item.Key == "flag")
        {
            strFlag = item.Value;
        }
        else if (item.Key == "wasmrootpath")
        {
            strWASMRootPath = item.Value;
        }
    }
#endif
    if (strResName == null || strResName.Length == 0)
    {
        return false;
    }
    if (strFlag == null || strFlag.Length == 0)
    {
        strFlag = "dcsoft";
    }
#endif DCWriterForASPNET
```

```
        response.BufferOutput = false;
        response.Buffer = false;
#endif

        if (_WorkDirectory == null)
        {
            // 自动判断 WASM 资源文件所在目录
            string strDefaultPath = null;
            lock (_CacheFileInfos)
            {
                const string SubDirName = "dcwriter5files";
            }
#if DCWriterForASPNET
            strDefaultPath =
System.Web.HttpContext.Current.Server.MapPath(SubDirName);
            if (IsDirecotryOK(strDefaultPath))
            {
                _WorkDirectory = strDefaultPath;
            }
#elif DCWriterForASPNETCore
            if (context != null)
            {
                var e =
(Microsoft.AspNetCore.Hosting.IHostingEnvironment)context.RequestServices.GetService(
typeof(Microsoft.AspNetCore.Hosting.IHostingEnvironment));
                if (e != null)
                {
                    var rootPath = e.WebRootPath;
                    if (rootPath != null && Directory.Exists(rootPath))
                    {
                        strDefaultPath = Path.Combine(rootPath,
SubDirName);

                        if (IsDirecotryOK(strDefaultPath))
                        {
                            _WorkDirectory = strDefaultPath;
                        }
                    }
                }
            }
#endif

            if (_WorkDirectory == null)
            {
                var path =
typeof(DCWriter5FileDownloadForASPNET).Assembly.CodeBase;
                if (path != null && path.Length > 0)
                {

```

```

        var url2 = new System.Uri(path);
        var path4 =
Path.Combine(Path.GetDirectoryName(url2.LocalPath), SubDirName);
        if (IsDirecotryOK(path4))
        {
            _WorkDirectory = path4;
        }
    }
    if (_WorkDirectory == null)
    {
        var path6 =
typeof(DCWriter5FileDownloadForASPNET).Assembly.Location;
        if (path6 != null && path6.Length > 0)
        {
            path6 =
Path.Combine(System.IO.Path.GetDirectoryName(path6), SubDirName);
            if (IsDirecotryOK(path6))
            {
                _WorkDirectory = path6;
            }
        }
    }
    if (_WorkDirectory == null)
    {
        WriteText(response, "未配置 DCWriter5 程序文件所在目录。请检
查目录["
            + strDefaultPath + "], 或者设置
WebWriterControlServicePageExecuter.DCWriter5FileDirectory 的属性值。");
        return true;
    }
}
Console.WriteLine("设置了第 5 代编辑器程序文件目录为: " +
_WorkDirectory);
bool bolCheckModified = true;
RestartFlag;;
GetWASMApVersion();
var cacheVersionBack = __CacheFileInfosVersion;
#if DCWriterForASPNET
    var strAbsoluteUri = request.Url.AbsoluteUri;
#elif DCWriterForASPNETCore
    var strAbsoluteUri = request.Scheme + "://" + request.Host +
request.Path.Value + request.QueryString;

```

```
#endif

    if( strWASMRootPath != null && strWASMRootPath.Length > 0 )
    {
        var index33 = strAbsoluteUri.IndexOf('?');
        if( index33 > 0 )
        {
            strAbsoluteUri = strWASMRootPath + "?" +
strAbsoluteUri.Substring(index33 + 1);
        }
    }
    var strKey = strResName + "#" + strAbsoluteUri;
    ResFileInfo file = null;
    lock ( _CacheFileInfos )
    {
        if ( _CacheFileInfos.TryGetValue(strKey, out file) == false )
        {
            var totalSize = 0;
            foreach (var item in _CacheFileInfos.Values)
            {
                if (item.Content != null)
                {
                    totalSize += item.Content.Length;
                }
            }
            if (totalSize > 100 * 1024 * 1024)
            {
                // 缓存的数据太大了，清空
                ClearCachedData();
            }
            file = new ResFileInfo(strResName, strAbsoluteUri);
            _CacheFileInfos[strKey] = file;
        }
    }
    if (file.FileExisted == false)
    {
        //SetResponseStatusCode(404);
        file.UpdateLastWriteTime();
        if (file.FileExisted)
        {
            // 之前文件不存在，现在又存在了。
            lock ( _CacheFileInfos )
            {
                file = new ResFileInfo(strResName, strAbsoluteUri);
                _CacheFileInfos[strKey] = file;
            }
        }
    }
}
```

```
        }
    }
    else
    {
        AddCORSHeader(request, response);
        WriteText(response, "DCWriter 未找到文件:" +
file.LocalFileName);
        return true;
    }
}
if (string.Equals(strResName, MainFileName,
StringComparison.OrdinalIgnoreCase))
{
    // 遇到主文件, 则进行文件版本检查
    var oldTime = file.LastWriteTime;
    lock (_CacheFileInfos)
    {
        file.UpdateLastWriteTime();
    }
    if (oldTime != file.LastWriteTime)
    {
        // 文件内容发生改变, 认为版本改变了
        if (file.FileExists == false)
        {
            AddCORSHeader(request, response);
            WriteText(response, "DCWriter 未找到文件:" +
file.LocalFileName);
            return true;
        }
        lock (_CacheFileInfos)
        {
            // 清空缓存的数据
            ClearCachedData();
            _WASMApVersion = null;
            goto RestartFlag;
        }
    }
}

#if DCWriterForASPNET
    if (strQueryVer != null && strQueryVer.Length > 10)
    {
        // 对于带有版本号的文件, 永久缓存
        response.CacheControl = "public";
    }
}
```

```
        response.Cookies.Clear();

response.Cache.SetCacheability(System.Web.HttpCacheability.Public);
        response.Cache.SetMaxAge(TimeSpan.MaxValue);
        response.Cache.SetExpires(DateTime.Now.AddYears(10));
    }
    else
    {
        // 没有版本号的文件，不缓存

response.Cache.SetCacheability(System.Web.HttpCacheability.NoCache);
    }
#elif DCWriterForASPNETCore
    if (strQueryVer != null && strQueryVer.Length > 10)
    {
        // 启用永久缓存
        SetResponseHeader(response, "Cache-Control",
"public,max-age=31536000");
    }
    else
    {
        // 没有版本号的文件，不缓存
        SetResponseHeader(response, "Cache-Control", "no-cache");
    }
#endif

    // 添加跨域使用的标记头
    AddCORSHeader(request, response);
    if (file.Header_ContentEncoding != null)
    {
        SetResponseHeader(response, "Content-Encoding",
file.Header_ContentEncoding);
    }
    //SetResponseHeader("Last-Modified22", file.LastWriteTimeString);
    if (cacheVersionBack != __CacheFileInfosVersion)
    {
        // 缓存的信息发生改变，这次不使用缓存数据，使用最新数据。
        file = new ResFileInfo(strResName, strAbsoluteUri);
        if (file.FileExisted == false)
        {
            WriteText(response, "DCWriter 未找到文件:" +
file.LocalFileName);
            return true;
        }
    }
}
```

```
#if DCWriterForASPNET
    if (file.Header_ContentType != null)
    {
        response.ContentType = file.Header_ContentType;
    }
    if (file.Content == null)
    {
        response.TransmitFile(file.LocalFileName);
    }
    else
    {
        response.BinaryWrite(file.Content); // .OutputStream.Write(file.Content, 0,
        file.Content.Length);
    }
    response.Flush();
    //response.Close();
#elif DCWriterForASPNETCore
    SetResponseHeader(response, "Content-Type",
file.Header_ContentType);
    if (file.Content == null)
    {
        using (var stream = new System.IO.FileStream(file.LocalFileName,
        FileMode.Open, FileAccess.Read))
        {
            await stream.CopyToAsync(response.Body);
        }
    }
    else
    {
        await response.Body.WriteAsync(file.Content, 0,
file.Content.Length);
    }
#endif
    return true;
}
private static bool IsDirecotryOK(string dir)
{
    return File.Exists(Path.Combine(dir, MainFileName));
}
/// <summary>
/// 添加跨域相关的头
/// </summary>
#endif DCWriterForASPNET
```

```
        private static void AddCORSHeader(System.Web.HttpRequest request,
System.Web.HttpResponse response)
#elif DCWriterForASPNETCore
        private static void AddCORSHeader(Microsoft.AspNetCore.Http.HttpRequest
request, Microsoft.AspNetCore.Http.HttpResponse response)
#endif
    {
        var strClientOrigin = (string)request.Headers["origin"];
        SetResponseHeader(response, "Access-Control-Allow-Credentials",
"true");
        SetResponseHeader(response, "Access-Control-Allow-Private-Network",
"true");
        //SetResponseHeader("Access-Control-Expose-Headers", "dcversion");
        SetResponseHeader(response, "P3P", "CP=CAO PSA OUR");
        if (strClientOrigin != null && strClientOrigin.Length > 0)
        {
            SetResponseHeader(response, "Access-Control-Allow-Origin",
strClientOrigin);
        }
        else
        {
            SetResponseHeader(response, "Access-Control-Allow-Origin",
"*");
        }
        SetResponseHeader(response, "Access-Control-Allow-Methods",
"POST, GET, PUT, DELETE, OPTIONS");
        SetResponseHeader(response, "Access-Control-Allow-Headers",
"Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");
        SetResponseHeader(response, "Access-Control-Max-Age", "0");
        SetResponseHeader(response, "Access-Control-Allow-Headers",
"Origin, No-Cache, X-Requested-With, If-Modified-Since, Pragma, Last-Modified, Cache
-Control, Expires, Content-Type, X-E4M-With, userId, token");
        SetResponseHeader(response, "XDomainRequestAllowed", "1");
    }
    private static void ClearCachedData()
    {
        lock (_CacheFileInfos)
        {
            foreach (var item in _CacheFileInfos.Values)
            {
                item.Content = null;
                //item.LastWriteTimeString = null;
            }
            _CacheFileInfos.Clear();
        }
    }
```



```
        __CacheFileInfosVersion++;
    }
}

private static volatile int __CacheFileInfosVersion = 0;
private static readonly Dictionary<string, ResFileInfo> _CacheFileInfos
= new Dictionary<string, ResFileInfo>();
private class ResFileInfo
{
    public ResFileInfo(string resName, string strAbsoluteUrl)
    {
        this.LocalFileName = null;
        this.LastWriteTime = DateTime.MinValue;
        //this.LastWriteTimeString = null;
        this.Content = null;

        var fullName = Path.Combine(_WorkDirectory, resName);
        this.FileExisted = System.IO.File.Exists(fullName);
        this.LocalFileName = fullName;
        if (this.FileExisted == false)
        {
            // 文件不存在
            return;
        }
        this.LastWriteTime =
System.IO.File.GetLastWriteTime(fullName).ToUniversalTime();
        if (fullName.EndsWith(".js",
StringComparison.OrdinalIgnoreCase)
            && resName.StartsWith("blazor.",
StringComparison.OrdinalIgnoreCase) == false
            && resName.StartsWith("dotnet.",
StringComparison.OrdinalIgnoreCase) == false)
        {
            // 对于 JS 文件更新其中的 import 指令
            this.LocalFileName = fullName;
            this.Header_ContentType = "text/javascript;charset=utf-8";
            var strResult = new StringBuilder();
            if (strAbsoluteUrl != null && strAbsoluteUrl.Length > 0)
            {
                // 更新版本号
                strAbsoluteUrl = strAbsoluteUrl.Replace("$$version$$",
GetWASMApVersion());
                // 抠出文件名
                string strHeader = string.Empty;
                string strEnd = string.Empty;
```

```
var index5 = strAbsoluteUrl.IndexOf("wasmres=");
if (index5 > 0)
{
    strHeader = strAbsoluteUrl.Substring(0, index5 + 8);
    var index6 = strAbsoluteUrl.IndexOf(".js", index5 +
8);

    if (index6 > 0)
    {
        strEnd = strAbsoluteUrl.Substring(index6 + 3);
    }
}
var bolHasChanged = false;
using (var reader = new
System.IO.StreamReader(this.LocalFileName, System.Text.Encoding.UTF8, true))
{
    for (var strLine = reader.ReadLine(); strLine != null;
strLine = reader.ReadLine())
    {
        if (strLine.StartsWith("import ",
StringComparison.OrdinalIgnoreCase))
        {
            // 创建新的 JS 路径
            var index = strLine.IndexOf("\\". "/",
StringComparison.Ordinal);

            if (index > 0)
            {
                var index3 = strLine.IndexOf("'", index
+ 3);

                string strFileName = null;
                if (index3 > 0)
                {
                    strFileName =
strLine.Substring(index + 3, index3 - index - 3);
                }
                else
                {
                    strFileName =
strLine.Substring(index + 3);
                }
                var strNewLine = strLine.Substring(0,
index) + "\\\" + strHeader + strFileName + strEnd + "\\\";";
                strResult.AppendLine(strNewLine);
                bolHasChanged = true;
                continue;
            }
        }
    }
}
```

```

        }
    }
    else if (strLine.Contains("\$$version$$\"))
    {
        // 更新版本号
        strLine = strLine.Replace("\$$version$$\\"",
"\" + GetWASMApVersion() + "\"");
        strResult.AppendLine(strLine);
        bolHasChanged = true;
        continue;
    }
    strResult.AppendLine(strLine);
} //for
}
if (bolHasChanged)
{
    this.Content =
System.Text.Encoding.UTF8.GetBytes(strResult.ToString());
}
}
}
else if (resName.EndsWith(".json",
StringComparison.OrdinalIgnoreCase))
{
    this.Header_ContentType = "application/json";
}
else
{
    if (File.Exists(fullName + ".gz"))
    {
        this.LocalFileName = fullName + ".gz";
        this.Header_ContentEncoding = "gzip";
    }
    if (resName.EndsWith(".wasm",
StringComparison.OrdinalIgnoreCase))
    {
        this.Header_ContentType = "application/wasm";
    }
    else if (resName.EndsWith(".js",
StringComparison.OrdinalIgnoreCase))
    {
        this.Header_ContentType =
"text/javascript;charset=utf-8";
    }
}

```

```

        else
        {
            this.Header_ContentType = "application/octet-stream";
        }
    }
}

public void UpdateLastWriteTime()
{
    if (this.LocalFileName != null && this.LocalFileName.Length > 0)
    {
        this.FileExisted = File.Exists(this.LocalFileName);
        if (this.FileExisted)
        {
            this.LastWriteTime =
File.GetLastWriteTime(this.LocalFileName).ToUniversalTime();
            //this.LastWriteTimeString =
this.LastWriteTime.ToString("r");
        }
        else
        {
            this.LastWriteTime = DateTime.MinValue;
            //this.LastWriteTimeString = null;
        }
    }
}

public bool FileExisted = false;
public string LocalFileName = null;
public byte[] Content = null;
//public string LastWriteTimeString = null;
public DateTime LastWriteTime = DateTime.MinValue;
public string Header_ContentType = null;
public string Header_ContentEncoding = null;
}

/// <summary>
/// 留给未来扩展用，压缩 JS 代码
/// </summary>
/// <param name="strJS"></param>
/// <returns></returns>
public static string MiniJavaScriptCode(string strJS)
{
    return strJS;
}
}

```

它的主要功能就是在服务端上下载 dcwriter5files 文件夹的资源文件。

2.2. 前端开发

2.2.1. html 开发

在前端使用 html、Asp.Net、MVC、Core、Java 开发时可参考以下步骤。

1. 在网页中添加添加 JQuery、DCWriter 资源的引用，如下：

```
<script src="http://server:8099/Jquery/jquery-1.7.2.min.js"></script>
<script
src="http://server:8099/ServicePage.aspx?wasmres=dcwriter5.js"></scri
pt>
```

2. 在页面中创建控件容器，并设置初始化属性：

```
<div id="myWriterControl" dctype="WriterControlForWASM"
style="height: 1200px; border-bottom: 0px solid black;
background-attachment: fixed; background-size: cover;
background-position: center top; " RegisterCode="050307D547168..."
RuleVisible="true" RuleBackColor="rgb(155, 187, 227)"
CaretCss="5,Blue"
DocumentOptions.BehaviorOptions.ParagraphFlagFollowTableOrSection="true" >
正在加载...
</div>
```

其中各个属性的含义可参考本文的属性介绍。

设置完以上步骤后，运行页面即可显示控件。

还可以通过动态代码的方式创建控件。

```
var ctl = document.getElementById("dcwriter");
window.CreateWriterControlForWASM(ctl);
```

2.2.2. Vue2 开发

暂缺

2. 2. 3. Vue3 开发

暂缺

2. 2. 4. Angular 开发

暂缺

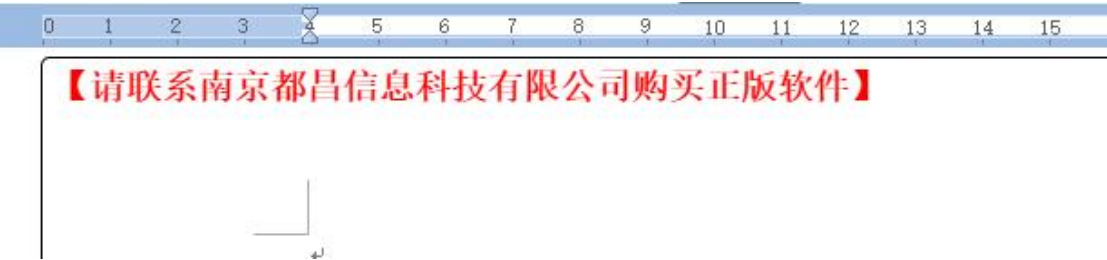
2. 2. 5. React 开发

暂缺

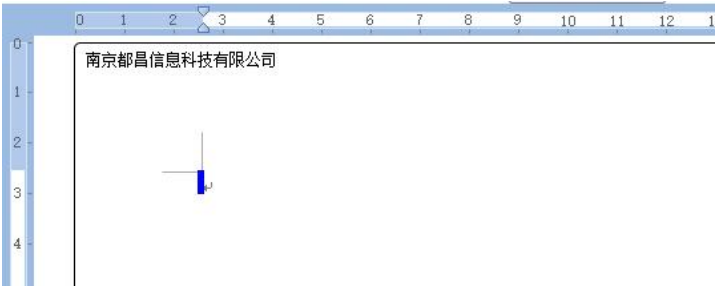
2. 3. 授权注册

DCWriter 产品经过南京都昌信息科技有限公司授权使用，会获得一个字符串的注册码字符串，将该注册码设置到控件的 RegisterCode 属性值上即可。

未注册时，会在界面上显示一个红色的提示。如下图：



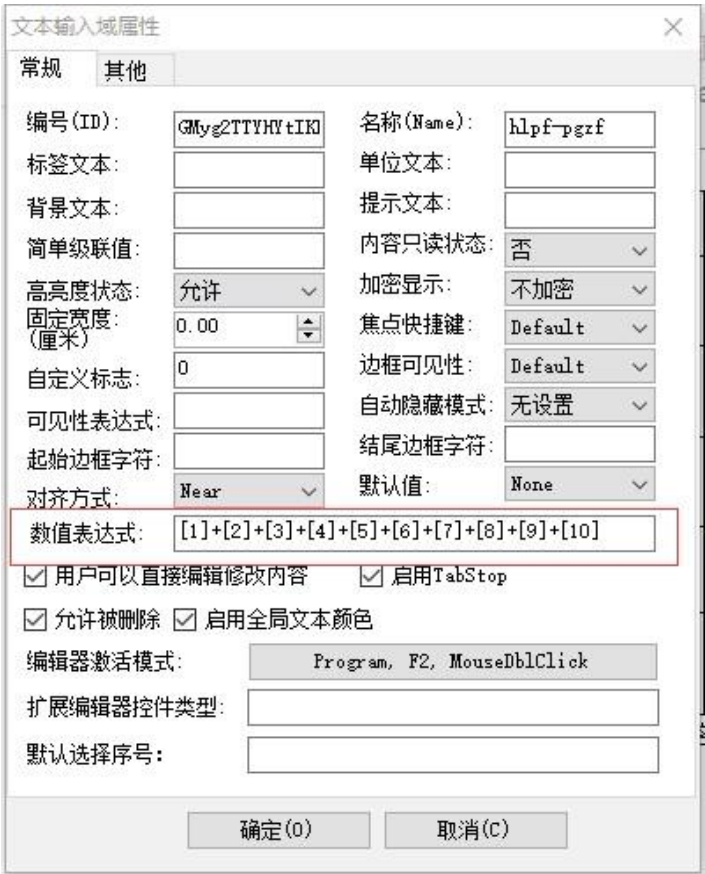
注册成功后，默认会在页面左上角显示注册码的文字。如下图：



该注册标记在打印时是不显示的。可参考本文的前端属性调整注册打印的位置。该注册标记在打印时时是不显示的。

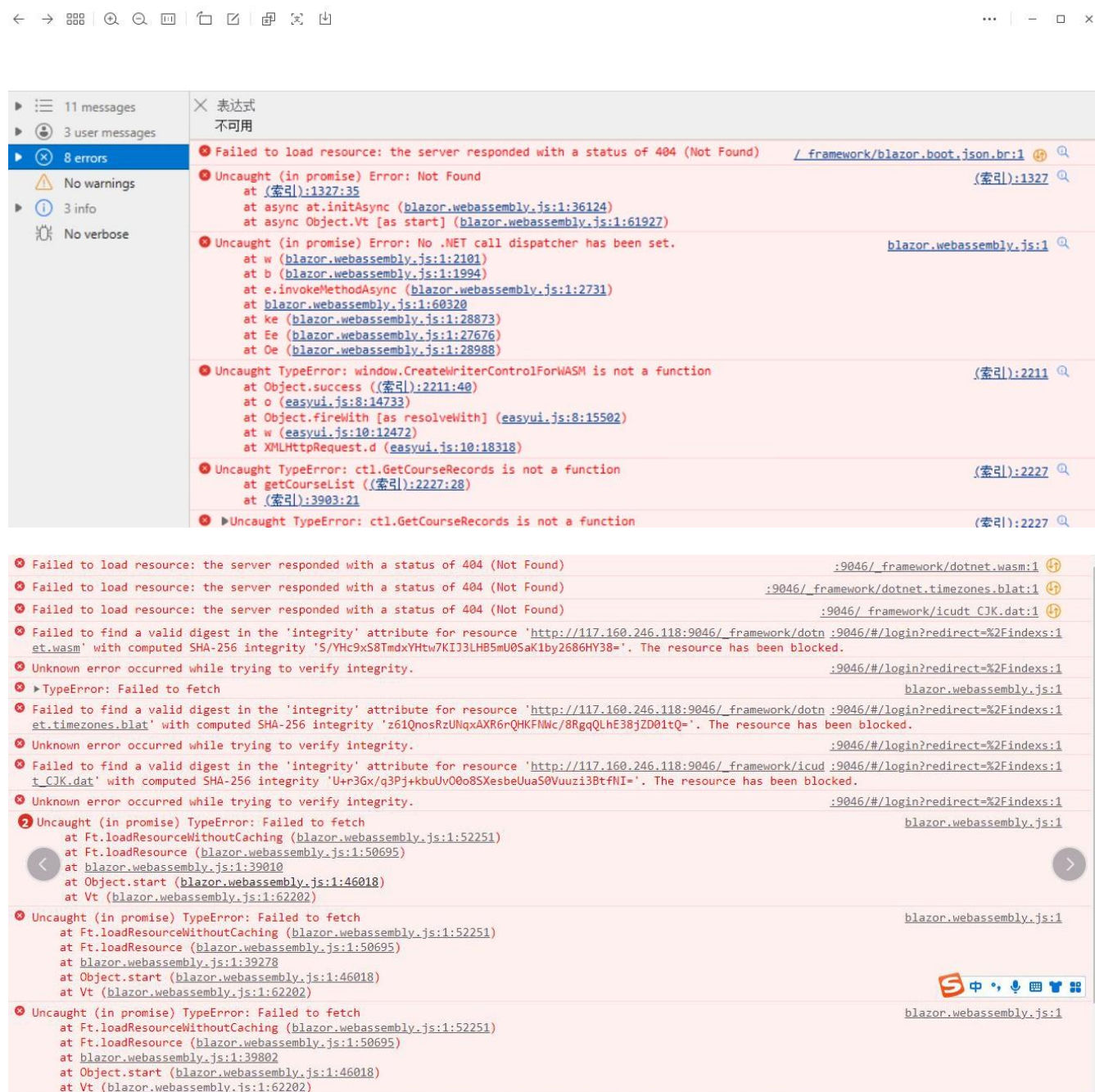
3. 使用规范

3.1. 在使用数值表达式的时候，输入域元素编号的命名需要符合标识符的规范，不要用纯数字或者带小数点进行命名。



答：如果用纯数字，数值表达式计算是直接取数字进行计算。

3.2. 第五代编辑器部署在 IIS 上进行访问，编辑器初始化的时候报以下错误的时候



HTTP 错误 404.3 - Not Found

由于扩展配置问题而无法提供所请求的页面。如果该页面是脚本，请添加处理程序。如果应下载文件，请添加 MIME 映射。

最可能的原因:

- 可能是缺少处理程序映射。默认情况下，静态文件处理程序将处理所有内容。
- 您要使用的功能可能尚未安装。
- 这可能是因为应用程序未使用适当的 MIME 映射。(警告: 请不要为用户不应的下载的 .ASPX 页或 .config 文件等内容创建 MIME 映射。)
- 如果未安装 ASP.NET。

可尝试的操作:

- 在 system.webServer/handlers 中:
 - 确保映射了当前页所需的处理程序。
 - 请特别检查前缀(例如, runtimeVersion, pipelineMode, bitness)并将这些前缀与应用程序池的设置进行比较。
 - 请特别检查所设处理程序行中的书写错误。
- 请确认要使用的功能已经安装。
- 确认启用了 MIME 映射, 或使用命令行工具 appcmd.exe 为网站添加 MIME 映射。
 - 要设置 MIME 类型, 请使用以下语法: %SystemRoot%\windows\system32\inetsrv\appcmd set config /section:staticContent /+[fileExtension="string",mimeType="string"]
 - 变量 fileExtension 字符串是文件扩展名, 变量 mimeType 字符串是文件类型说明。
 - 例如, 要为扩展名为 ".xyz" 的文件添加 MIME 映射: appcmd set config /section:staticContent /+[fileExtension=".xyz",mimeType="text/plain"]

警告: 请在确认您的 Web 服务器需要此 MIME 映射之后再将其添加到列表中。配置文件(如 .CONFIG)或动态脚本页(如 .ASP 或 .ASPX)不应直接下载。应始终通过处理程序来处理。有时会使用其他文件(如数据库文件或 .XHL 或 .MDF 等用于存储配置的文件)来存储配置信息。请先确认客户端能否下载这些类型的文件, 然后再添加这些文件。
- 安装 ASP.NET。
- 创建跟踪规则以跟踪此 HTTP 状态代码的失败请求。有关为失败的请求创建跟踪规则的详细信息, 请单击[此处](#)。

详细错误信息:

模块	StaticFileModule	请求的 URL	http://localhost:80/_framework/blazor.boot.json.br
通知	ExecuteRequestHandler	物理路径	C:\inetpub\wwwroot_framework\blazor.boot.json.br
处理程序	StaticFile	登录方法	匿名
错误代码	0x80070032	登录用户	匿名

详细信息:

当所请求 URL 的文件扩展名是针对未在服务器上配置的 MIME 类型时, 便会发生此错误。您可以为动态脚本页、数据库或配置文件以外的文件的文件扩展名添加 MIME 类型。这些文件类型使用一个处理程序来处理。您不应允许直接下载动态脚本页、数据库或配置文件。

[查看详细错误信息 >](#)

答：需要在当前网站下 MIME 下添加以下类型。

分稳定版

icode

icode2.0

ipNet

eCeshi

MIME 类型

使用此功能管理被 Web 服务器用作静态文件的文件扩展名和关联的内容类型的列表。

分组依据: 不进行分类

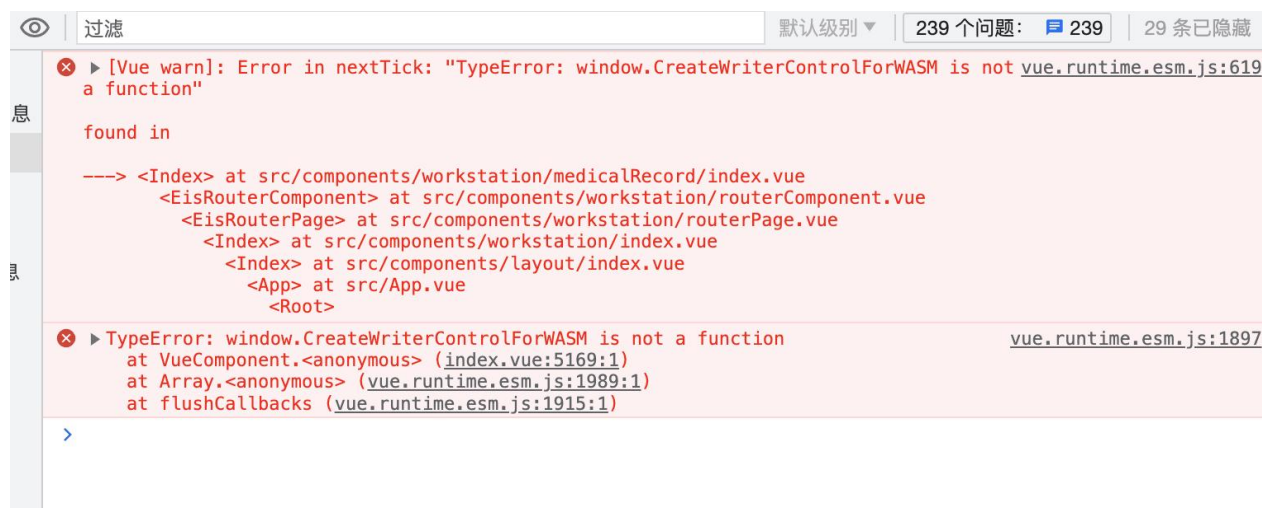
扩展名	MIME 类型	条目类型
.wasm	application/octet-stream	本地
.dat	application/octet-stream	本地
.blat	application/octet-stream	本地
.br	application/octet-stream	本地

3.3. vue 引用第五代编辑器文件的时候，启动 vue，浏览器控制台报：SyntaxError:Unexpected end of JSON input。



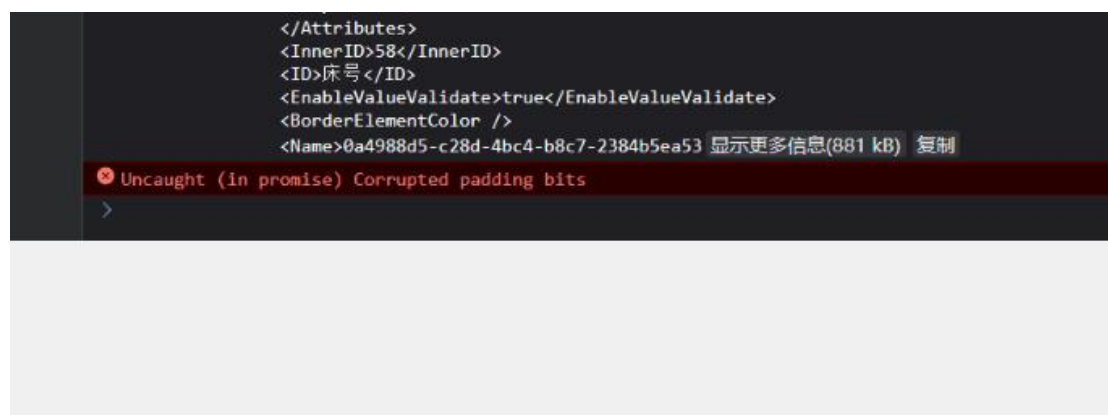
答：找到_framework 文件夹下面的 blazor.webassembly.js 文件，把_framework 全局替换下/_framework 即可

3.4. vue3 编辑器初始化浏览器报以下错误: "TypeError:window.CreateWriterControlForWASM is not a function"



答: 把_framework 测试包放入到 vue 下 public 文件夹下即可

3.5. vue 运行编辑器初始化的时候报错: "Corrupted padding bits"



答: 找到编辑器初始化的代码, 如下:

```
//注释掉这个js代码, 替换成下面的引用js
<script type="module">
  import { BrotliDecode } from './_framework/decode.min.js';
  Blazor.start({
    loadBootResource: function (type, name, defaultUri, integrity) {
      if (type !== 'dotnetjs') {
```

```
        return (async function () {
            const response = await fetch(defaultUri + '.br', { cache:
'no-cache' });

            if (!response.ok) {
                throw new Error(response.statusText);
            }

            const originalResponseBuffer = await response.arrayBuffer();
            const originalResponseArray = new
Int8Array(originalResponseBuffer);

            const decompressedResponseArray =
BrotliDecode(originalResponseArray);

            const contentType = type ===
                'dotnetwasm' ? 'application/wasm' :
'application/octet-stream';

            return new Response(decompressedResponseArray,
                { headers: { 'content-type': contentType } });
        })();
    }
});
</script>

//换成这个写法
<script src="_framework/blazor.webassembly.js"></script>
```

3.6. 不能在病程记录对象里面插入病程记录

病程记录中不支持插入病程记录。

3.7. 用户执行登录接口，在有的容器元素里面输入内容不会产生痕迹。

答：是因为容器元素都有 EnablePermission 属性设置后只对自己有影响，在这个容器内不开启痕迹。

比如表格：



3.8. 设置文档选项的时候，不能同步调用编辑器的命令。

答：例如：

```
//调用的命令会无效
ctl. DocumentOptions.ViewOptions.ShowHeaderBottomLine= false
ctl.ExecuteCommand( "FormViewMode" , false , "Normal" );
ctl.ApplyDocumentOptions()

//正常用法
ctl. DocumentOptions.ViewOptions.ShowHeaderBottomLine= false
ctl.ApplyDocumentOptions()
ctl.ExecuteCommand( "FormViewMode" , false , "Normal" );
```

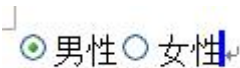
4. 结构化元素

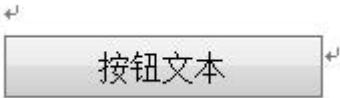
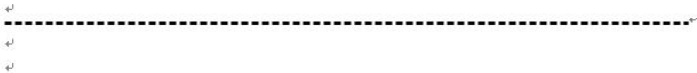
DCWriter 第五代 Web 版是结构化文本编辑器，包含的常用结构化元素包含：

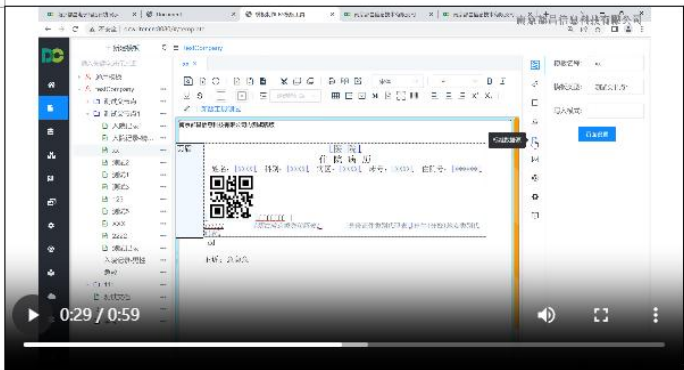
序号	类型	说明
1	文本输入域，类型名称 XTextInputFieldElement	元素属性 InnerEditStyle 为 Text 即为输入普通文本。如下图所示： 
2	静态下拉列表输入域，类型名称 XTextInputFieldElement	元素属性 InnerEditStyle 为 DropDownList 即为静态下拉输入域。用户单击、双击或通过其他方式激活时，会展示下拉列表，下拉列表的内容存储在 XML 中。下拉列表还可以设置为多选。如下图所示： 
3	动态下拉列表输入域，类型名称 XTextInputFieldElement	元素属性 InnerEditStyle 为 DropDownList 且 DynamicListItems 为 true 即为动态下拉输入域用户单击、双击或通过其他方式激活时，会展示下拉列表，下拉列表的内容需要通过代码设置。下拉列表还可以设置为多选。如下图所示： 

4	时间输入域, 类型名称 XTextInputField Element	元素属性 InnerEditStyle 为 Time 即为时间输入域。用户单击、双击或通过其他方式激活时, 会显示时间选择框。如下图所示: 
5	日期输入域, 类型名称 XTextInputField Element	元素属性 InnerEditStyle 为 Date 即为日期输入域。用户单击、双击或通过其他方式激活时, 会显示日期选择框。如下图所示: 
6	日期时间输入域, 类型名称 XTextInputField Element	元素属性 InnerEditStyle 为 DateTime 即为日期时间输入域。用户单击、双击或通过其他方式激活时, 会显示日期时间选择框。如下图所示:

		入院日期: yyyy年MM月dd日 HH时mm分 ◀ 2024 年 1 月 ▶ 日 一 二 三 四 五 六 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1 2 3 选择时间 13 : 48 : 11 现在 清除 确定
	不带秒的日期时间输入域，类型名称 XTextInputField Element	元素属性 InnerEditStyle 为 DateTimeWithoutSecond 即为不带秒日期时间输入域。用户单击、双击或通过其他方式激活时，会显示日期时间选择框。如下图所示： 入院日期: yyyy年MM月dd日 HH时mm分 记录日期: ◀ 2024 年 6 月 ▶ 日 一 二 三 四 五 六 26 27 28 29 30 31 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 1 2 3 4 5 6 选择时间 09 : 26 现在 清除 确定

	XTextInputField Element	示： 
8	单选框，类型名称 XTextRadioBoxElement	对于单选框，当鼠标点击设置了勾选状态后，同组的其他单选框就会去掉勾选状态。如下图所示： 
9	复选框，类型名称 XTextCheckBoxElement	复选框表示这一组复选框中，点击勾选一个，不会影响其他的选择框的选择。如下图所示： 
10	标签文本，类型名称 XTextLabelElement	表示一组文本是一个整体。删除时他们会被一起删除。双击可修改内容。如下图所示： 
11	条形码，类型名称 XTextNewBarcodeElement	表示各种编码的一维条码。如下图所示： 
12	二维码，类型名称 XTextTDBarcodeElement	表示 QR 编码的二维条码。如下图所示：

		
13	分页符，类型名称 XTextPageBreakElement	默认文档是自动分页的，当内容较多时，超出纸张高度时，它会自动分页。但是当插入分页符时，此时会强制分页。分页符是不显示和打印的。
14	图片，类型名称 XTextImageElement	支持 base64 的图片数据，也支持网络图片。网络图片的跨域需要开发者自行处理。
15	按钮，类型名称 XTextButtonElement	以按钮的方式显示，可通过事件实现按钮的点击功能。如下图所示： 
16	水平线，类型名称 XTextHorizontalLineElement	以水平线的方式，对上下文内容进行分隔区域显示。如下图所示： 
17	页码，类型名称 XTextPageInfoElement	能显示当前页码和文档总页数。可以设置页码的格式。如下图所示： 
18	多媒体，类型名称 XTextMediaElement	在文档中插入音频视频，支持在文档中播放。如下图所示：

		
19	月经史, 类型名称 XTextNewMedicalExpressionElement	支持四种医学月经史公式。
20	瞳孔图, 类型名称 XTextNewMedicalExpressionElement	支持瞳孔图
21	光定位图, 类型名称 XTextNewMedicalExpressionElement	支持光定位图
22	眼球突出度图, 类型名称 XTextNewMedicalExpressionElement	支持眼球突出度图
23	斜视符号图, 类型名称 XTextNewMedicalExpressionElement	支持斜视符号图
24	胎心图, 类型名称 XTextNewMedicalExpressionElement	支持胎心图
25	疼痛指数图, 类型	支持疼痛指数图

	名 称 XTextNewMedicalExpressionElement	
26	分数公式, 类型名称 XTextNewMedicalExpressionElement	支持分数公式
27	恒牙牙位图, 类型名称 XTextNewMedicalExpressionElement	支持恒牙牙位图
28	乳牙牙位图, 类型名称 XTextNewMedicalExpressionElement	支持乳牙牙位图
29	PD 牙位图, 类型名称 XTextNewMedicalExpressionElement	支持 PD 牙位图
30	病变上牙牙位图, 类型名称 XTextNewMedicalExpressionElement	支持病变上牙牙位图
31	病变下牙牙位图, 类型名称 XTextNewMedicalExpressionElement	支持病变下牙牙位图
32	电话力测试牙位图, 类型名称	支持电话力测试牙位图

	XTextNewMedicalExpressionElement	
33	固定桥牙位图，类型名称 XTextNewMedicalExpressionElement	支持固定桥牙位图
34	复合牙位图/替牙列，类型名称 XTextNewMedicalExpressionElement	支持复合牙位图/替牙列

这些结构化元素都可以通过代码操作。

5. 前端属性

在本文前端开发中设置了很多前端属性，这些属性可以直接写在控件容器的属性中。如下：

```
<div id="myWriterControl" dctype="WriterControlForWASM"
    style="height: 700px;width: 100%; border-bottom: 0px solid black;
background-attachment: fixed; background-size: cover;
background-position: center top; "
    RuleVisible="true" RuleBackColor="rgb(155, 187, 227)"
    CaretCss="5,Blue" RegisterCode="123"
    PageCssString="box-shadow:0px 0px 0px grey" onload="this.load()">
    正在加载...
</div>
```

控件支持的前端属性和说明如下：

序号	属性	说明
1	id*	前端控件的编号，表示控件在网页内的唯一标识符 js 通过 <code>var ctl=document.getElementById("myWriterControl")</code> 可以定位到该控件。非必须属性，但建议设置。

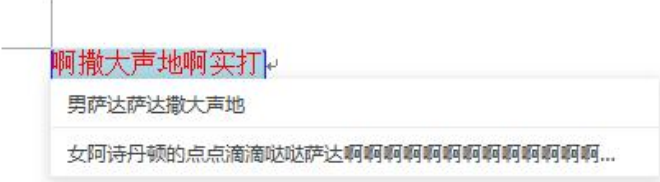
2	dctype*	表示前端 DCWriter 第五代控件的标识，它的值只能是“WriterControlForWASM”。这是必须设置的属性。
3	style*	表示用于在 HTML 文档中定义元素的 CSS 代码。非必须属性，但建议设置，比如 height、width。
4	RegisterCode*	此处设置编辑器的注册码。DCWriter 产品经过南京都昌信息科技有限公司授权使用，会获得一个字符串的注册码字符串，将该注册码设置到控件的 RegisterCode 属性值上即可。不设置，此时文档默认左上角会显示红色的未注册提示文字
5	CaretCss*	表示光标的样式，它的格式如下：CaretCss="5, Blue"，其中第一个参数表示光标的宽度，默认是 1；第二个参数表示光标的颜色，默认是 Black。
6	RuleVisible*	表示是否显示标尺。默认不显示标尺。显示标尺时，会在页面的上部、左侧各显示一个标尺公式。 当光标定位在文档中，此时拖拽左侧标尺，可以设置文档的上下页边距的高度。 当光标定位在文档中，此时上部的标尺会由两个滑块，拖动上部的滑块，可以调整当前段落的首行缩进量，拖动下部的滑块，可以调整当前段落的左侧缩进量。在两个滑块之间拖动，可以调整页面的左边距。拖动上部标尺右侧，可以调整页面右边距。非必须属性，可不设置。
7	RuleBackColor*	当 RuleVisible 为 true 时设置标尺的背景色。参考的格式是“rgb(230, 230, 230)”。非必须属性，可不设置。
8	DocumentOptions	DocumentOptions 表示设置文档的选项。通过设置选项可以实现客户的差异化需求，如代码： <pre>ctl.DocumentOptions.ViewOptions.ShowParagraphFlag = false;</pre> 上面的代码，设置为 true，表示在文档中段落的最后面显示符号标记；设置为 false 表示不在文档中显示段落符号的标记。 具体的选项可参考本文的前端选项。非必须属性，可不设置。
9	Doublebuffer	表示绘图时绘图时是否使用双缓冲功能。非必须属性，可不设置。 双缓冲即在内存中创建一个与绘图区域一致的对象，先将图形绘制到

		内存中的这个对象上，再一次性将这个对象上的图形拷贝到屏幕上，这样能大大加快绘图的速度，也可以在一定程度上避免刷新闪烁问题。
10	PageTitlePosition*	注册后，注册标记文字的位置。它包含：TopLeft 左上角、TopCenter 顶部中间、TopRight 右上角、BottomLeft 左下角、BottomCenter 底部中间、BottomRight 右下角。默认是左上角，PageTitlePosition=“TopLeft”。非必须属性，可不设置使用默认值。
11	PageBorderColor*	页面边框的颜色，默认是黑色。参考设置 PageBorderColor=“rgb(255, 0, 0)”，可调整边框颜色，如下图所示：  非必须属性，可不设置。
12	servicepageurl*	服务端路径。在服务端开发中，可以在服务端属性中设置注册码。此时再设置 servicepageurl 属性指向服务端地址时，就可以实现在服务端进行注册的功能。 非必须属性，可不设置。谨慎使用这个属性，还没有完全测试。
13	debugmode	是否启用调试模式，默认是 false。若为 true，则系统处于调试模式，系统会输出一些调试文本信息。调试模式下会帮助开发者了解编辑器内部运行的一些情况，不过这会影响一些性能。
14	AutoZoom*	表示控件是否使用自动缩放视图模式，当设置为 true 时，页面尺寸会根据控件的大小自动缩放填满整个控件区域。默认是 false，以页面设置的纸张尺寸显示。
15	AutoMaticCorrectionRepeatID*	表示是否自动校正重复的 ID，当插入元素时设置元素 ID，当文档中已经存在对应的 ID 时，自动在 ID 后面添加序号，格式如下“field_1”
16	DropDownList_Two	表示下拉列表是否已两栏显示

	ColumnDisplay	
17	DropDownListIsShowUsingSearchContent	当该属性设置成 true 的时候，下拉输入域带有检索框时，在检索框里面输入的内容，会出现‘使用此文本’按钮，把检索框里面的内容回填到输入域中 
18	printimagescale	在调用 PrintAsImage 接口，为提供清晰度，数字越大清晰度越高。
19	pagelayoutmode*	视图显示的模式，分为 singlecolumn 单栏、multicolumn 多栏、Horizontal 水平平铺。
20	pagecssstring	设置页面的样式信息，和 style 属性相同
21	readonlyautofocus	表示在移动端表单模式下，点击非输入域时是否会自动聚焦到最近的输入域中，防止弹出软键盘。
22	mobileisableautokeyboard	表示在移动端，是否除了手动点击其他操作都不会弹出软键盘。
23	notscrollwithcaret*	表示页面是否随着光标进行跳转
24	clickcommentareafocuscaret	表示点击批注位置时是否会丢失光标
25	TTFPath	表示初始化 TTF 文件系统的路径，默认是发布包的 _framework/Fonts/。
26	enabledlogapi*	表示是否记录 API 日志
27	IsDialogReadOnly	表示属性对话框是否只读。默认属性对话框是允许修改的。当设置 IsDialogReadOnly 为 true 时表示属性对话框只能查看，不允许修改。
28	ShowElementViewHandle*	默认时光标定位在表格时，表格左上角会出现一个全选的图标，如下图所示：

		 可通过 ShowElementViewHandle 属性控制该图标是否显示。当设置为 false 时，该图标不显示。默认值是 true，显示图标。
29	DropDownListItem DeselectButton*	控制在下拉选项中是否存在删除按钮的功能。 当参数设为 true 时，此时下拉项出现删除按钮，如下图所示：  当设置为 false 时则不显示删除按钮。
30	isPasteAsText	粘贴时，是否为纯文本粘贴。设置成 true，粘贴的内容都是纯文本
31	MobileDisableAutoSoftKeyboard	移动端会在表单模式下除了文本输入域外会全都直接禁止掉软键盘。设置成 true
32	IsUsePasteDialog	类似 word 粘贴 
33	FullWidthSpace	用户输入空格时，使用全角空格，默认是半角空格
34	HighlightInsertedString	调用命令 InsertString 插入文本内容时，插入内容是否需要选中，true 选中，false：不选中。默认值为 true
35	OutputSVGWhitespace	打印预览时，是否输出空格，默认为 false，不输出空格字符，复制时，粘贴出来的内容空格丢失。设置成 true，粘贴不丢失空格
36	moveToButtonWhenClick	点击按钮时，是否移动输入焦点到按钮元素旁，默认值为 true，移动输入焦点到按钮元素旁边，false，不修改输入焦点位置
37	PrintRuleOccupies	打印预览时，标尺是否占位，true 值时，隐藏且占位，false 值时，隐藏不占位

	ng	
38	InvalidateTextCommentBackgroundColor	自定义批注校验背景色颜色  The screenshot shows a medical form with fields for '发病节气' (Onset of disease), '医师签名' (Physician signature), and '发病除还有' (Onset of disease still has). A red error message box is displayed on the right side, stating: '数据校验错误 发病节气,文本过长,不得超过 2 个字符。' (Data validation error: Onset of disease, text is too long, it must not exceed 2 characters).
39	ShowMediaOnMask	点击播放视频文件时，是否在编辑器页面全屏播放。  The screenshot shows a video player interface. At the top, there is a red banner with the text: '【请联系南京都昌信息科技有限公司购买正版软件】' (Please contact Nanjing Dunchang Information Technology Co., Ltd. to purchase genuine software). Below the banner, there is a form for a '诊断证明书' (Diagnosis Certificate) with fields for '姓名' (Name), '性别' (Gender), '年龄' (Age), '科室' (Department), '类别' (Category), '身份证号' (ID Card Number), and '住院号' (Inpatient Number). The video player itself is in the center, showing a '开发者中心' (Developer Center) page with various links and information.
40	BoundsSelectionModeUseNewLogic	区域选择打印是否支持跨页选择打印内容。true 值时支持跨页，编辑器默认值为 false。
41	DropDownListWidth	用户自定义下拉弹框的宽度。单位为 px。

		
42	DCDorpDownAllowKeyboardEntry	下拉输入域弹出下拉项后，用户是否还允许在输入域里面输入内容，默认是禁用医生输入（数字、字母，中文还允许输入）。默认值为 false，当设置成 true 值时，医生允许输入。
43	ExecuteExpressionAfterLoadingDocument	编辑器初始化加载文档是否需自动触发表达式。默认值为 true, 触发表达式。false 值不会自动触发文档表达式功能。
44	ManuallyLoadingfonts()	编辑器初始化下载 pdf 所需字体资源，无须等需要下载 pdf 时，在开始下载字体资源
45	DownLoadFontsForLocalPDF()	优先下载资源，无需当需要的时候在开始下载

6. 前端选项

DCWriter 支持很多个文档选项，开发者通过编辑器控件的 DocumentOptions 属性就能访问文档选项，保存文档选项数据的对象类型为 DCSoft.Writer.DocumentOptions，该类型的主要分为四个部分：SecurityOptions 安全和权限相关的选项、ViewOptions 文档视图选项、BehaviorOptions 系统行为选项、EditOptions 编辑选项。

通过设置选项可以实现客户的差异化需求，如代码：

```
myEditControl.DocumentOptions.ViewOptions.ShowParagraphFlag = false;
```

上面的代码，设置为 true，表示在文档中段落的最后面显示符号标记；设置为 false 表示不在文档中显示段落符号的标记。

设置文档选项的方法有两种：

第一种，在创建控件时，在控件容器的属性中添加，如下：

```
<div id="myWriterControl" dctype="WriterControlForWASM" style="height: 1200px; "
RegisterCode="050307D547168..."
DocumentOptions.ViewOptions.ShowParagraphFlag="true" >正在加载...
</div>
```

可设置多个。

第二种，在创建控件后，调用接口实现如下：

```
ctl.DocumentOptions.ViewOptions.ShowHeaderBottomLine= false;
ctl.DocumentOptions.ViewOptions.ShowParagraphFlag = false;
ctl.ApplyDocumentOptions()
```

可设置多个，再调用 `ApplyDocumentOptions` 接口。

注意这些文档选项是编辑器下的接口，也就是说跟文档无关，设置的值不会保存到文档中。

注意当文档选项值为枚举值时，需要知道各个枚举值的含义，请看对应的示例，里面有详细的介绍

6.1. 安全和权限相关选项

类型 `DCSoft.Writer.Security.DocumentSecurityOptions` 用于存储安全和权限相关的选项，主要属性有：

*6.1.1. AutoEnablePermissionWhenUserLogin

描述：在用户登录时自动启用授权控制。默认为 `true`。

参数：`bool` 值。

代码示例：

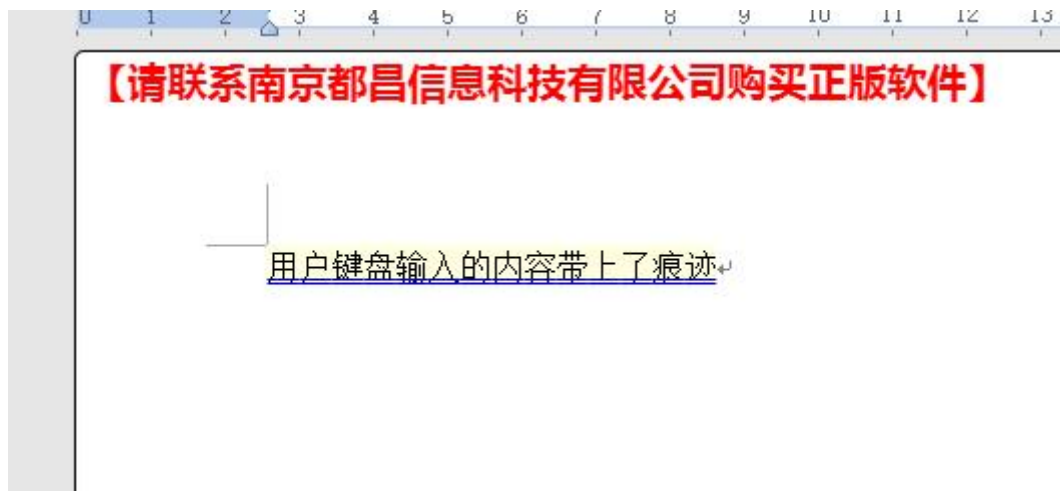
```
ctl.DocumentOptions.SecurityOptions.AutoEnablePermissionWhenUserLogin
=true;

ctl.ApplyDocumentOptions();
```

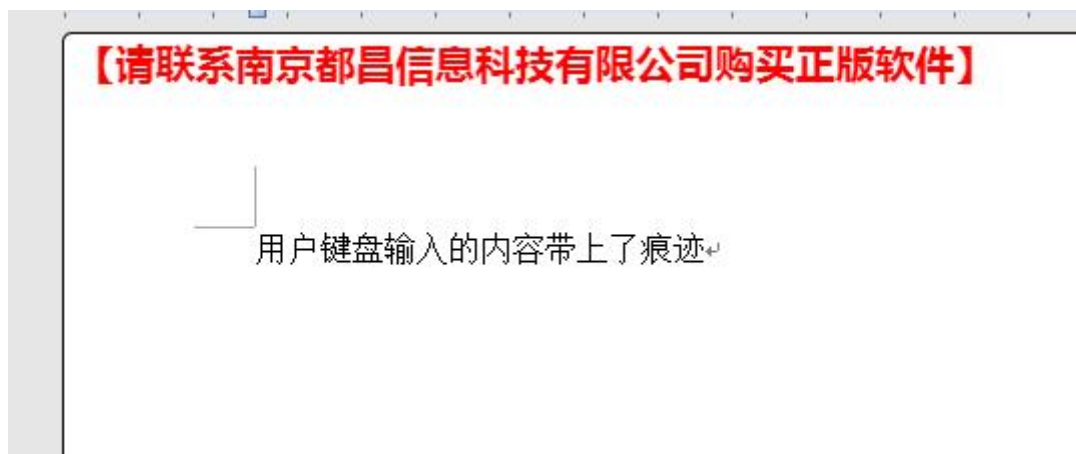
备注：当该属性为 `false` 的时候，用户执行了 `UserLoginByUserLoginInfo` 等登录接口，通过键盘在输入域里面输入内容，不会带上痕迹，也就是没有权限

控制功能。

设置成 true 的界面效果：



设置成 false 的界面效果：



6.1.2. CAMode

描述：CA 运行模式。默认为 Software。

参数：枚举值。

代码示例：

```
ctl.DocumentOptions.SecurityOptions.CAMode="Disabled";  
ctl.ApplyDocumentOptions();
```

备注：当该文档选项为 Disabled 值的时候，调用方法 CASignature 无效，不支持软件 CA 功能。

默认为 Software 值时：



为 Disabled 时，没有对应效果

6.1.3. CanModifyDeleteSameLevelContent②

描述：能否修改或删除同授权等级的内容。默认为 true。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.SecurityOptions.CanModifyDeleteSameLevelContent=false;
```

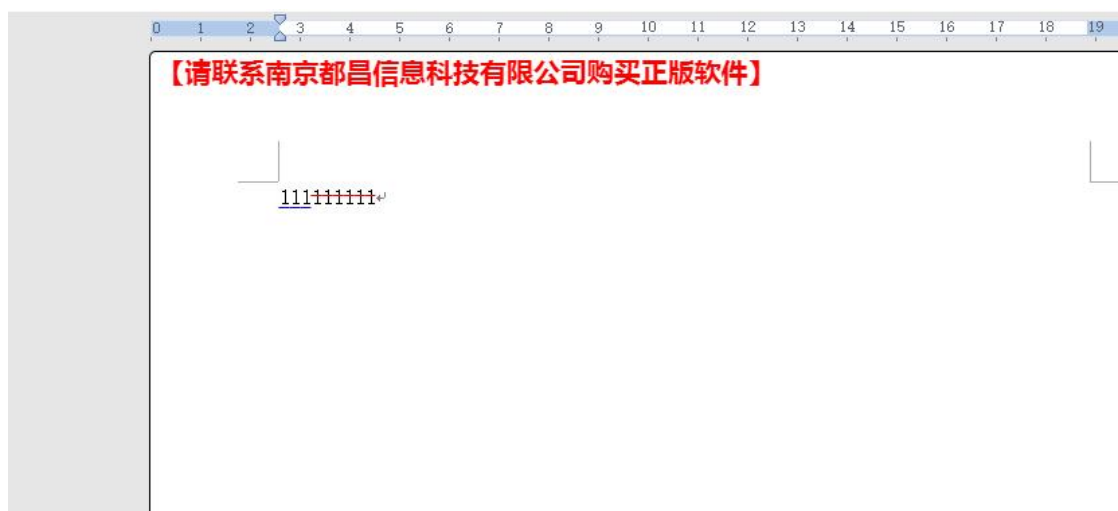
```
ctl.ApplyDocumentOptions();
```

备注：无。

设置成 false 的效果：



设置成 true 的效果:



6.1.4. CreatorTipFormatString

描述：文档内容创建信息格式化字符串

参数：字符串。

代码示例：

```
ctl.DocumentOptions.SecurityOptions.CreatorTipFormatString="创建痕迹展示";
```

备注：用户执行了登录接口输入内容之后光标放在带有痕迹的内容上面，留痕创建提示信息可以自定义。

设置了自定义信息值效果:



默认值的效果:



6.1.5. DeleterTipFormatString

描述: 文档内容删除信息格式化字符串

参数: 字符串。

代码示例:

```
ctl.DocumentOptions.SecurityOptions.DeleterTipFormatString="删除痕迹展示";
```

备注: 用户执行了登录接口输入内容之后光标放在带有痕迹的内容上面, 留痕删除提示信息可以自定义。效果跟 CreatorTipFormatString 一致

6.1.6. EnableLogicDelete①

描述: 执行逻辑删除。默认为 false。

参数: bool 值。

代码示例：

```
ctl.DocumentOptions.SecurityOptions.EnableLogicDelete = true;
```

备注：需结合使用，如用户登录后设置以下代码，表示复杂留痕视图模式：

```
ctl.DocumentOptions.SecurityOptions.EnablePermission = true;
```

```
ctl.DocumentOptions.SecurityOptions.EnableLogicDelete = true;
```

```
ctl.DocumentOptions.SecurityOptions.ShowLogicDeletedContent =  
true;
```

```
ctl.DocumentOptions.SecurityOptions.ShowPermissionMark = true;
```

```
ctl.DocumentOptions.SecurityOptions.ShowPermissionTip = true;
```

效果图：



6.1.7.EnablePermission①

描述：启用文档内容安全和权限控制。若为 false 则不启用，文档内容可任意编辑。默认为 false。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.SecurityOptions.EnablePermission = true;
```

备注：需结合使用，如用户登录后设置以下代码，表示复杂留痕视图模式：

```
ctl.DocumentOptions.SecurityOptions.EnablePermission = true;
```

```
ctl.DocumentOptions.SecurityOptions.EnableLogicDelete = true;
```

```
ctl.DocumentOptions.SecurityOptions.ShowLogicDeletedContent =  
true;
```



```
ctl.DocumentOptions.SecurityOptions.ShowPermissionMark = true;
```

```
ctl.DocumentOptions.SecurityOptions.ShowPermissionTip = true;
```

效果图：



6.1.8. ShowLogicDeletedContent①

描述：显示被逻辑删除的元素。默认为 false。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.SecurityOptions.ShowLogicDeletedContent=true;
```

备注：可结合使用，如用户登录后设置以下代码，表示复杂留痕视图模式：

```
ctl.DocumentOptions.SecurityOptions.EnablePermission = true;
```

```
ctl.DocumentOptions.SecurityOptions.EnableLogicDelete = true;
```

```
ctl.DocumentOptions.SecurityOptions.ShowLogicDeletedContent      =  
true;
```

```
ctl.DocumentOptions.SecurityOptions.ShowPermissionMark = true;
```

```
ctl.DocumentOptions.SecurityOptions.ShowPermissionTip = true;
```

界面效果：



6.1.9. ShowPermissionMark①

描述：是否显示授权标记。若为 true，则用蓝色下划线标记出高权限用户输入的内容，使用删除线标记出被逻辑删除的内容。默认为 false。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.SecurityOptions.ShowPermissionMark= true;
```

备注：需结合使用，如用户登录后设置以下代码，表示复杂留痕视图模式：

```
ctl.DocumentOptions.SecurityOptions.EnablePermission = true;
```

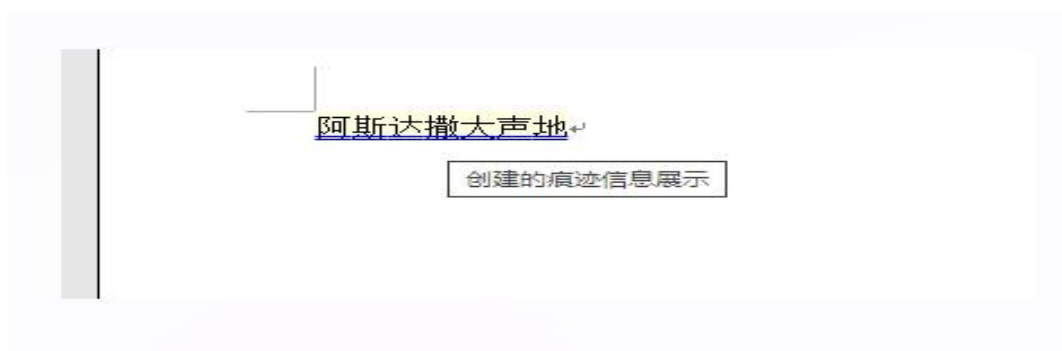
```
ctl.DocumentOptions.SecurityOptions.EnableLogicDelete = true;
```

```
ctl.DocumentOptions.SecurityOptions.ShowLogicDeletedContent = true;
```

```
ctl.DocumentOptions.SecurityOptions.ShowPermissionMark = true;
```

```
ctl.DocumentOptions.SecurityOptions.ShowPermissionTip = true;
```

界面效果：



6.1.10. ShowPermissionTip①

描述：是否显示授权相关的提示信息，若为 true，则在编辑器中当鼠标移动到 文档内容时，会以提示文本的方式显示文档内容权限和痕迹信息。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.SecurityOptions.ShowPermissionTip= true;
```

备注：需结合使用，如用户登录后设置以下代码，表示复杂留痕视图模式：

```
ctl.DocumentOptions.SecurityOptions.EnablePermission = true;
```

```
ctl.DocumentOptions.SecurityOptions.EnableLogicDelete = true;
```

```
ctl.DocumentOptions.SecurityOptions.ShowLogicDeletedContent = true;
```

```
ctl.DocumentOptions.SecurityOptions.ShowPermissionMark = true;
```

```
ctl.DocumentOptions.SecurityOptions.ShowPermissionTip = true;
```

界面效果：



6.1.11. TrackVisibleLevel0②

描述：等级 0 的用户痕迹可视化选项

参数：TrackVisibleConfig 用户痕迹可视化设置对象。TrackVisibleConfig。

代码示例：

```
ctl.DocumentOptions.SecurityOptions.TrackVisibleLevel0.DeleteLineNum = 5;
```

备注：默认无背景色，下划线为一条浅蓝色线，删除线为一条淡红色线。应用程序可以修改这些用户痕迹可视化设置，使得用户界面上能显示各种风格的用户痕迹可视化效果。

6.1.12. TrackVisibleLevel1②

描述：等级 1 的用户痕迹可视化选项

参数：TrackVisibleConfig 用户痕迹可视化设置对象。[参考附录](#) TrackVisibleConfig。

代码示例：

```
ctl.DocumentOptions.SecurityOptions.TrackVisibleLevel1.DeleteLineNum = 5;
```

备注：默认为无背景色，下划线为 1 条深蓝色线，删除线为 1 条红色线。应用程序可以修改这些用户痕迹可视化设置，使得用户界面上能显示各种风格的用户痕迹可视化效果。

6.1.13. TrackVisibleLevel2②

描述：等级 2 的用户痕迹可视化选项

参数：TrackVisibleConfig 用户痕迹可视化设置对象。[参考附录](#) TrackVisibleConfig。

代码示例：

```
ctl.DocumentOptions.SecurityOptions.TrackVisibleLevel2.DeleteLineNum = 5;
```

备注：默认背景色为淡黄色，下划线为双蓝色线，删除线为双红色线。应用程序可以修改这些用户痕迹可视化设置，使得用户界面上能显示各种风格的用户痕迹可视化效果。

6.1.14. RealDeleteOwnerContent②

描述：用户对自己输入的内容删除，是否为逻辑删除还是物理删除。默认值为 false

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.SecurityOptions.RealDeleteOwnerContent =true ;
```

备注：用户客户自己来控制自己输入的内容，自己删除是否需要带上痕迹。

设置为 true 的效果：



设置成 false 值效果:



6.2. 文档视图选项

类型 `DCSoft.Writer.DocumentViewOptions` 用于存储文档视图选项数据, 这些数据影响文档对于用户的显示结果, 该类型主要属性有:

6.2.1. AdornTextBackColor

描述: 扩展文字背景色, 默认为#64808080。

参数：Color 值。

代码示例：需结合使用

```
ctl.DocumentOptions.ViewOptions.AdornTextBackColor= "Red";  
ctl.DocumentOptions.ViewOptions.AdornTextVisibility= "Activated";  
ctl.DocumentOptions.ViewOptions.AdornTextBackColor= "ID";  
ctl.ApplyDocumentOptions();
```

备注：无。

效果图：



6.2.2. AdornTextVisibility

描述：文档元素扩展文本显示方式，默认为 Hidden。

参数：枚举值。

代码示例：需结合使用

```
ctl.DocumentOptions.ViewOptions.AdornTextVisibility= "Activated";  
ctl.DocumentOptions.ViewOptions.AdornTextBackColor= "Red";  
ctl.DocumentOptions.ViewOptions.AdornTextBackColor= "ID";  
ctl.ApplyDocumentOptions();
```

备注：无。

效果图：

机构名称

科别: 科别 床号: 床号床 姓名: 姓名 住院号: 住院号

入院记录

姓名: 姓名 11	出生地: 出生地
性别: 性别	职业: 职业
年龄: 年龄岁	入院时间: 入院时间
民族: 民族	记录时间: 记录时间
婚姻: 婚姻	病史陈述者: 病史陈述者
发病节气: 发病节气	

医师签名: 医师签名

6.2.3. BackgroundTextColor①

描述：字段域的背景文本颜色，默认为灰色。

参数：Color 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.BackgroundColor= "Red";  
ctl.ApplyDocumentOptions();
```

备注：无。

6.2.4. BothBlackWhenPrint①

描述：打印的时候前景色全部强制为黑色

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.BothBlackWhenPrint= true;  
ctl.ApplyDocumentOptions();
```

备注：无。

6.2.5. CommentDateFormatString②

描述：文档批注时间格式化字符串

参数：字符串值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.CommentDateFormatString='yyyy-MM-dd';  
ctl.ApplyDocumentOptions();
```

备注：无。

6.2.6. CommentFontName②

描述：文档批注字体名称。

参数：string 字符串。

代码示例：

```
ctl.DocumentOptions.ViewOptions.CommentFontName="宋体";  
ctl.ApplyDocumentOptions();
```

备注：无。

6.2.7. CommentFontSize②

描述：文档批注字体大小，默认为 10。

参数：float 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.CommentFontSize=10;  
ctl.ApplyDocumentOptions();
```

备注：无。

6.2.8. DefaultAdornTextType

描述：默认的输入域的扩展标记文字类型。默认值 DataSource

参数：枚举值。

代码示例：需结合使用

```
ctl.DocumentOptions.ViewOptions.AdornTextBackColor= "ID";  
ctl.DocumentOptions.ViewOptions.AdornTextVisibility= "Activated";  
ctl.DocumentOptions.ViewOptions.AdornTextBackColor= "Red";
```

备注：无。

6.2.9. DefaultInputFieldHighlight①

描述：默认的输入域高亮度显示模式。默认值 Enabled

参数：枚举值。

代码示例：需结合使用

```
ctl.DocumentOptions.ViewOptions.DefaultInputFieldHighlight="Disabled";
```

备注：只有当输入域的高亮度设置为默认状态，才会受到该文档选项的影响。

6.2.10. DropDownListFontName①

描述：下拉列表字体名称，默认为空。

参数：string 字符串。

代码示例：

```
ctl.DocumentOptions.ViewOptions.DropDownListFontName= "宋体";
```

备注：无。

6.2.11. DropDownListFontSize①

描述：下拉列表字体大小，小于等于 0 则不设置，默认为 0。

参数：float 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.DropDownListFontSize= 16;
```

备注：无。

6.2.12. EmphasisMarkSize②

描述：着重号的直径。

参数：float 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.EmphasisMarkSize=30;
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 13. EnableEncryptView①

描述：是否允许视图加密显示。默认值为 true

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.EnableEncryptView=true;
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 14. EnableFieldTextColor①

描述：是否启用输入域文字颜色，默认为 false。和 FieldTextColor 命令搭配使用。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.EnableFieldTextColor= "true";
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 15. FieldBackColor①

描述：文本输入域的默认背景颜色，默认为浅蓝色。

参数：color 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.FieldBackColor=' #8C9EA6' ;
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 16. FieldBorderColor①

描述：输入域两侧边框元素颜色，默认为#00000000。

参数：color 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.FieldBorderColor=' red' ;
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6.2.17. FieldBorderElementPixelWidth②

描述：输入域边框元素像素宽度

参数：float 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.FieldBorderElementPixelWidth=20;
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6.2.18. FieldBorderPrintVisibility①

描述：打印时输入域边界元素的可见性设置，默认为 Default。也就是不显示不占排版位置

参数：枚举值。隐藏边框且占位（Hidden）、隐藏边框不占位（None）

代码示例：

```
ctl.DocumentOptions.ViewOptions.FieldBorderPrintVisibility="None";
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6.2.19. FieldFocusedBackColor①

描述：文本输入域获得输入焦点时的高亮度背景颜色,默认为淡蓝色

参数：color 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.FieldFocusedBackColor='yellow';
```

```
ctl.ApplyDocumentOptions();
```

备注：DocumentOptions.ViewOptions.DefaultInputFieldHighlight="Disabled";//

输入域的高亮度不可用之后该文档选项无效。

6. 2. 20. FieldHoverBackColor①

描述：鼠标悬浮在文本输入域时文本输入域的高亮度背景颜色，默认为淡蓝色

参数：color 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.FieldHoverBackColor='red';
```

```
ctl.ApplyDocumentOptions();
```

备注：DocumentOptions.ViewOptions.DefaultInputFieldHighlight="Disabled";//

输入域的高亮度不可用之后该文档选项无效。

6. 2. 21. FieldInvalidateValueBackColor①

描述：文本输入域数据异常时的高亮度背景色，默认为全透明

参数：color 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.FieldInvalidateValueBackColor='black';
```

```
ctl.ApplyDocumentOptions();
```

备注：DocumentOptions.ViewOptions.DefaultInputFieldHighlight="Disabled";//

输入域的高亮度不可用之后该文档选项无效。

6. 2. 22. FieldInvalidateValueForeColor①

描述：文本输入域数据异常时的高亮度文本色，默认为淡红色。

参数：color 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.FieldInvalidateValueForeColor='red';
```

```
ctl.ApplyDocumentOptions();
```

备注：DocumentOptions.ViewOptions.DefaultInputFieldHighlight="Disabled";//

输入域的高亮度不可用之后该文档选项无效。

6. 2. 23. FieldTextColor①

描述：输入域文字颜色，和 EnableFieldTextColor 搭配使用，默认为黑色。

参数：Color 值。

代码示例：需搭配使用该选项才会生效

```
ctl.DocumentOptions.ViewOptions.FieldTextColor="black";  
ctl.DocumentOptions.ViewOptions.EnableFieldTextColor=true;  
ctl.ApplyDocumentOptions();
```

备注：无。

6.2.24.FieldTextPrintColor①

描述：文档域打印时文字颜色值。默认颜色为#00FFFFFF

参数：string 字符串，ARGB 颜色格式。

代码示例：

```
ctl.DocumentOptions.ViewOptions.FieldTextPrintColor="red";
```

备注：无。

6.2.25.HeaderBottomLineWidth②

描述：页眉下边框线的宽度，默认值是 1。

参数：数值型。

代码示例：

```
ctl.DocumentOptions.ViewOptions.HeaderBottomLineWidth=1;
```

备注：无。

6.2.26.HiddenFieldBorderWhenLostFocus②

描述：输入域失去焦点时隐藏边框元素

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.HiddenFieldBorderWhenLostFocus= true;  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 27. HiddenTableBorderJumpPrintPage②

描述：在打印发生续打时的页面时隐藏表格边框线，但对于其他页面则正常打印。默认 false。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.HiddenTableBorderJumpPrintPage=true;  
ctl.ApplyDocumentOptions();
```

备注：旗舰版编辑器该文档选项才有效果。

6. 2. 28. HighlightProtectedContent②

描述：高亮度显示受保护的内容。默认值 false

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.HighlightProtectedContent=true;  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 29. IgnoreFieldBorderWhenPrint①

描述：打印的时候忽略掉输入域边界元素, 是其不占据位置, 默认为 true。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.IgnoreFieldBorderWhenPrint= true;  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 30. MaskColorForJumpPrint①

描述：续打时使用的遮盖色，可以是半透明颜色。

参数：color 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.MaskColorForJumpPrint='Yellow';  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 31. MinTableColumnWidth①

描述：表格列的最小宽度，单位为 1/300 英寸，默认为 50。

参数：int 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.MinTableColumnWidth=500;  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 32. NewInputContentUnderlineColor②

描述：表示新输入的内容的下划线颜色。

参数：color 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.NewInputContentUnderlineColor='red';  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 33. NoneBorderColor①

描述：绘制隐藏的边框线（虚边框）使用的颜色。默认淡灰色。

参数：Color 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.NoneBorderColor= "LightGray";  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 34. NormalFieldBorderColor①

描述：常规的输入域的边界元素颜色，用户可以在常规的输入域中直接输入内容。该属

性值默认为蓝色

参数: Color 值。

代码示例:

```
ctl.DocumentOptions.ViewOptions.NormalFieldBorderColor= "Blue";  
ctl.ApplyDocumentOptions();
```

备注: 无。

6.2.35. PageMarginLineLength②

描述: 表示页面边界的线条长度

参数: int 值。

代码示例:

```
ctl.DocumentOptions.ViewOptions.PageMarginLineLength=100;  
ctl.ApplyDocumentOptions();
```

备注: 无。

6.2.36. PreserveBackgroundTextWhenPrint①

描述: 在打印的时候保留背景文字的位置, 但不打印背景文字。

参数: bool 值。

代码示例:

```
ctl.DocumentOptions.ViewOptions.PreserveBackgroundTextWhenPrint= true;
```

备注: 无。

6.2.37. PrintBackgroundText①

描述: 打印时是否显示输入域的背景文字, 默认为 false。

参数: bool 值。

代码示例:

```
ctl.DocumentOptions.ViewOptions.PrintBackgroundText= true;  
ctl.ApplyDocumentOptions();
```

备注: 无。

6. 2. 38. ReadonlyFieldBackColor①

描述：只读输入域的默认背景颜色，默认为浅灰色

参数：Color 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.ReadonlyFieldBackColor='red';
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 39. ReadonlyFieldBorderColor①

描述：只读输入域的边界元素颜色，默认为灰色

参数：Color 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.ReadonlyFieldBorderColor="Gray";
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 40. SectionBorderVisibility②

描述：文档节边框线可见性，默认为 All

参数：枚举值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.SectionBorderVisibility = "Hidden";
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 41. SelectionHightlightMaskColor②

描述：选择区域高亮度遮盖色,默认为半透明淡蓝色

参数：color 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.SelectionHightlightMaskColor='red';
```

```
ctl.ApplyDocumentOptions();
```

备注：传的是 ARGB 格式的颜色，需要把后面的两位数据，移到最前面，编辑器处理透明度是在前两位数。

6. 2. 42. ShowBackgroundCellID①

描述：作为背景显示单元格编号，默认为 false。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.ShowBackgroundCellID= true;  
  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 43. ShowCellNoneBorder①

描述：是否显示表格单元格的隐藏的边框线。当为 true 时，若表格没有边框线，则在编辑器中也会使用 NoneBorderColor 选项指定的颜色显示出边框线。该设置不影响打印结果。该选项默认值为 true。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.ShowCellNoneBorder=false;  
  
ctl.ApplyDocumentOptions();
```

备注：当表格设置边框线为虚边框的时候，设置这个文档选项，虚边框就不会显示出来。

6. 2. 44. ShowExpressionFlag②

描述：当单元格设置了表达式，则在右下角显示表达式标记，默认为 true。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.ShowExpressionFlag= true;
```

备注：无。

6. 2. 45. ShowFieldBorderElement①

描述：是否显示输入域边框元素, 默认为 true。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.ShowFieldBorderElement= true;  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 46. ShowFieldBorderTextInTheMiddle②

描述：新的输入域显示方式-前缀文本-左边框-背景文字-表单按钮-右边框-后缀文本。默认为 false，需要设置下编辑器 div 上才有效果

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.ShowFieldBorderTextInTheMiddle=true;  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 47. ShowFormButton①

描述：是否显示表单按钮。默认值 false

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.ShowFormButton=true;  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 48. ShowHeaderBottomLine①

描述：当页眉有内容时显示页眉下边框线。若为 false，则页眉和正文之间就没有分隔横线。默认值为 true。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.ShowHeaderBottomLine= true;  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 49. ShowInputFieldStateTag①

描述：是否显示输入域状态标签。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.ShowInputFieldStateTag= true;
```

备注：无。

6. 2. 50. ShowLineNumber①

描述：是否显示行号

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.ShowLineNumber= true;  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 51. ShowPageBreak①

描述：是否显示分页符

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.ViewOptions.ShowPageBreak=false;  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 2. 52. ShowParagraphFlag①

描述：显示段落标记。不影响打印，默认为 true。

参数: bool 值。

代码示例:

```
ctl.DocumentOptions.ViewOptions.ShowParagraphFlag= true;
```

```
ctl.ApplyDocumentOptions();
```

备注: 无。

6.2.53. TableCellBorderVisibility②

描述: 表格单元格边框线可见性, 默认为 All。

参数: 枚举值。

代码示例:

```
ctl.DocumentOptions.ViewOptions.TableCellBorderVisibility = "Hidden";
```

```
ctl.ApplyDocumentOptions();
```

备注: 无。

6.2.54. TagColorForModifiedField①

描述: 内容修改的输入域的标记点颜色, 默认为蓝色。

参数: Color 值。

代码示例:

```
ctl.DocumentOptions.ViewOptions.TagColorForModifiedField= "Blue";
```

备注: 无。

6.2.55. TagColorForNormalField①

描述: 普通输入域的标记点颜色, 默认为蓝色。

参数: Color 值。

代码示例:

```
ctl.DocumentOptions.ViewOptions.TagColorForNormalField= "Blue";
```

备注: 无。

6.2.56. TagColorForReadonlyField①

描述: 只读的输入域的标记点颜色, 默认为透明色。

参数: Color 值。

代码示例:

```
ctl.DocumentOptions.ViewOptions.TagColorForReadonlyField=" Gray";
```

备注: 无。

6. 2. 57. TagColorForValueInvalidateField①

描述: 数据不正确的输入域的标记点颜色, 默认为蓝色。

参数: Color 值。

代码示例:

```
ctl.DocumentOptions.ViewOptions.TagColorForValueInvalidateField=" Red";
```

备注: 无。

6. 2. 58. UnEditableFieldBorderColor①

描述: 内容不能被用户直接录入修改的输入域的边界元素颜色, 默认为红色

参数: Color 值。

代码示例:

```
ctl.DocumentOptions.ViewOptions.UnEditableFieldBorderColor="Red";
```

```
ctl.ApplyDocumentOptions();
```

备注: 无。

6. 2. 59. AutoZoomDropDownListFontSize①

描述: 自动缩放下拉列表字体大小、设置该文档选项, 当编辑器视图放大缩小的时候, 输入域的下拉列表下拉项的字号会随着缩放比率进行自适应

参数: bool 值。

代码示例:

```
ctl.DocumentOptions.ViewOptions.AutoZoomDropDownListFontSize=true;
```

```
ctl.ApplyDocumentOptions();
```

备注: 无。

6.3. 编辑选项

类型 `DocumentEditOptions` 存储了编辑器选项数据，其主要属性有：

6.3.1. `AutoInsertTableRowWhenMoveToNextCell`①

描述：在移动到下一个单元格时根据需要自动插入表格行。默认为 `true`

参数：`bool` 值。

代码示例：

```
ctl.DocumentOptions.EditOptions.KeepTableWidthWhenInsertDeleteColumn=true;

ctl.ApplyDocumentOptions();
```

备注：光标在表格最后一个单元格按下 `tab` 键自动新增表格行。

6.3.2. `ClearFieldValueWhenCopy`①

描述：在复制内容时清空输入域的内容，默认为 `false`

参数：`bool` 值。

代码示例：

```
ctl.DocumentOptions.EditOptions.ClearFieldValueWhenCopy = true;

ctl.ApplyDocumentOptions();
```

备注：无。

6.3.3. `CopyInTextFormatOnly`①

描述：仅仅采用纯文本格式复制内容,若为 `true`，则只复制纯文本内容，而且忽略掉控件的 `CreationDataFormats` 属性设置

参数：`bool` 值。

代码示例：

```
ctl.DocumentOptions.EditOptions.CopyInTextFormatOnly = true;

ctl.ApplyDocumentOptions();
```

备注：无。

6.3.4. CopyWithoutInputFieldStructure①

描述：复制时去掉输入域层次结构。默认值：false

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.EditOptions.CopyWithoutInputFieldStructure = true;  
ctl.ApplyDocumentOptions();
```

备注：全文也就输入域为纯文本复制。

6.3.5. FixSizeWhenInsertImage①

描述：在插入图片时自动修正图片的宽度，使得图片元素的宽度不会超过容器客户区宽度，而且高度不超过标准页高。默认为 true。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.EditOptions.FixSizeWhenInsertImage=true;  
ctl.ApplyDocumentOptions();
```

备注：无。

6.3.6. KeepTableWidthWhenInsertDeleteColumn①

描述：在插入和删除表格列时是否保持表格的总宽度不变。默认 true。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.EditOptions.KeepTableWidthWhenInsertDeleteColumn=true;  
ctl.ApplyDocumentOptions();
```

备注：无。

6.3.7. TabKeyToFirstLineIndent①

描述：是否使用 Tab 键来设置段落首行缩进，默认为 true。

参数：bool 值。

代码示例：


```
ctl.DocumentOptions.EditOptions.KeepTableWidthWhenInsertDeleteColumn=true;  
ctl.ApplyDocumentOptions();
```

备注：无。

6.3.8. ValueValidateMode②

描述：文档数据校验模式，默认为 LostFocus。

参数：枚举值：

None：禁止数据校验

Dynamic：实时的数据校验。需要再编辑器初始化的时候执行下校验方法（且元素必填不参与实时校验）

LostFocus：当文本输入域失去输入焦点校验

Program：编辑器不自动进行数据校验

代码示例：

```
ctl.DocumentOptions.EditOptions.ValueValidateMode="None";
```

备注：无。

6.4. 文档行为选项

存储了系统行为选项数据，这些选项影响了 DCWriter 软件的一些行为

6.4.1. AcceptDataFormats①

描述：编辑器控件能接受的数据格式，包括粘贴操作和 OLE 拖拽操作获得的数据。

参数：枚举值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.AcceptDataFormats='Text';  
ctl.ApplyDocumentOptions();
```

备注：无。

6.4.2. AllowDeleteJumpOutOfField①

描述：允许跳出输入域来删除内容。

参数: bool 值。

代码示例:

```
ctl.DocumentOptions.BehaviorOptions.AllowDeleteJumpOutOfField = true;  
ctl.ApplyDocumentOptions();
```

备注: 无。

6. 4. 3. AutoAssistInsertString①

描述: 是否启用快捷辅助录入功能。

参数: bool 值。

代码示例:

```
ctl.DocumentOptions.BehaviorOptions.AutoAssistInsertString = true;  
ctl.ApplyDocumentOptions();
```

备注: 无。

6. 4. 4. AutoAssistInsertStringDetectTextLength①

描述: 用自动辅助输入文本时的检测长度。

参数: int 值。

代码示例:

```
ctl.DocumentOptions.BehaviorOptions.AutoAssistInsertStringDetectTextLength  
=2;  
ctl.ApplyDocumentOptions();
```

备注: 无。

6. 4. 5. AutoTranslateDescString

描述: 自动转换功能中的目标字符组成的字符串

参数: string 值。

代码示例: 需搭配使用

```
ctl.DocumentOptions.BehaviorOptions.AutoTranslateDescString = "A B C D E  
F G H I J K L M N O P Q R S T U V W X Y Z 0 1 2 3 4 5 6 7 8 9。+-";
```

```
ctl.DocumentOptions.BehaviorOptions.AutoTranslateSourceString=  
"abcdefghijklmnopqrstuvwxyz0123456789.+-";  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 6. AutoTranslateSourceString

描述：自动转换功能中的原始字符组成的字符串

参数：string 值。

代码示例：需搭配使用

```
ctl.DocumentOptions.BehaviorOptions.AutoTranslateDescString = "A B C D E  
F G H I J K L M N O P Q R S T U V W X Y Z 0 1 2 3 4 5 6 7 8 9.+-";  
ctl.DocumentOptions.BehaviorOptions.AutoTranslateSourceString=  
"abcdefghijklmnopqrstuvwxyz0123456789.+-";  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 7. AutoUppercaseFirstCharInFirstLine②

描述：段落中第一个字母大写。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.AutoUppercaseFirstCharInFirstLine= false;  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 8. CommentEditableWhenReadOnly①

描述：即使在只读状态下是否能编辑文档批注。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.CommentEditableWhenReadOnly = true;  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 9. CommentVisibility①

描述：是否显示文档批注

参数：枚举值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.CommentVisibility = "Hide";  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 10. CompressXMLContent②

描述：压缩 XML 内容

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.CompressXMLContent = true;  
ctl.ApplyDocumentOptions();
```

备注：文档选项：BehaviorOptions.OutputFormattedXMLSource=false 效果一致。

6. 4. 11. DataObjectRange①

描述：数据对象应用范围

参数：枚举值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.DataObjectRange="SingleWriterControl";  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 12. DefaultEditorActiveMode①

描述：默认的数值编辑器激活模式

参数：枚举值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.DefaultEditorActiveMode="GotFocus, Mouse  
DblClick";
```

```
ctl.ApplyDocumentOptions();
```

备注：只有输入域属性下的激活模式设置成了默认值，该文档选项才有效果。

6. 4. 13. DesignMode①

描述：编辑器是否处于设计模式。

参数：枚举值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.DesignMode = true;
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 14. DoubleClickToEditComment①

描述：鼠标双击来编辑文档批注。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.DoubleClickToEditComment=false;
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 15. DoubleClickToSelectWord①

描述：双击选中文字。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.DoubleClickToSelectWord = false;
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 16. EnableCalculateControl

描述：是否允许使用计算器控件(数字小键盘)--数值输入域的数字小键盘是否可用。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.EnableCalculateControl=false;
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 17. EnableContentLock^①

描述：是否启用文档内容锁定机制。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.EnableContentLock=false;
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 18. EnableCopySource^①

描述：允许执行内容复制。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.EnableCopySource=false;
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 19. EnableDataBinding^①

描述：是否启用数据源绑定，如果为 false，则编辑器不执行数据源绑定。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.EnableDataBinding = false;
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 20. EnableLogUndo①

描述：是否记录撤销操作信息,如果为 `false`，则系统不记录文档操作的信息，用户操作就不能撤销或重做。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.EnableLogUndo = false;
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 21. ExcludeKeywords①

描述：全文违禁关键字列表，可包含多个违禁关键字，各个关键字之间用半角逗号分开。

参数：string 字符串。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.ExcludeKeywords = "子宫, 卵巢";
```

```
ctl.ApplyDocumentOptions();
```

备注：当该选项不为空时，应用程序调用 `DocumentValueValidate` 命令校验文档内容时会进行文档正文的违禁关键字检查，若包含了违禁关键字则认为文档内容出现校验错误。

编辑器控件 `WriterControl` 或 `WriterControlExt` 的 `ExcludeKeywords` 属性值就等价于该选项值。

6. 4. 22. FastInputMode①

描述：快速录入模式。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.FastInputMode = true;
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 23. FormView①

描述：表单视图模式。

参数：枚举值：

Strict：严格表单模式

Normal：普通表单模式

Disable：退出表单模式

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.FormView= "Normal";  
  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 24. MaxTextLengthForPaste②

描述：粘贴操作中的最大文本长度的限制。若大于 0 则要粘贴的文本长度超过该值则禁止操作；为负数表示部分粘贴；等于 0 表示无限制。

参数：int 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.MaxTextLengthForPaste = -10;  
  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 25. MinCountForDropDownListSpellCodeArea②

描述：显示下拉列表中拼音码区域的最小项目个数

参数：int 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.MinCountForDropDownListSpellCodeArea=4;  
  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 26. MoveFocusHotKey①

描述：移动焦点快捷键

参数：枚举值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.MoveFocusHotKey = "Enter";  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 27. OutputBackgroundTextToRTF②

描述：在导出 RTF 文档时是否导出文本输入域的背景文本。默认为 true。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.OutputBackgroundTextToRTF = false;  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 28. OutputFormattedXMLSource①

描述：是否输出格式化的 XML 文本，默认为 true。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.OutputFormattedXMLSource = true;  
ctl.ApplyDocumentOptions();
```

备注：格式化的 XML 文本带有缩进控制，阅读方便，但文档比较大。

6. 4. 29. ParagraphFlagFollowTableOrSection①

描述：排版时段落符号紧跟在表格或文档节后面。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.ParagraphFlagFollowTableOrSection=
```

```
true;
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 30. PromptJumpBackForSearch②

描述：是否提示跳回去继续搜索文本

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.PromptJumpBackForSearch = false;
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 31. PromptProtectedContent①

描述：当视图删除无法删除的内容时的提示方式，默认为 Details。

参数：枚举值：

None：不提示

Simple：简单的提示

Details：显示详细信息的提示

Flash：内容闪烁提升

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.PromptProtectedContent="None";
```

```
ctl.ApplyDocumentOptions();
```

备注：文档内容受保护时的提示方式。当用户进行删除内容的操作时，比如按下 Delete/Backspace 键删除内容，或者删除表格行、表格列、表格时会遇到受保护的文档内容而导致操作受到限制或者无法执行。此时编辑器可以显示对话框用于提示用户。而本选项用于控制编辑器如何显示提示信息。

6. 4. 32. SaveBodyTextToXml①

描述：保存文档对象的 BodyText 到 XML 文件中。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.SaveBodyTextToXml = true;  
  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 33. ShowTooltip①

描述：是否显示提示文本。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.ShowTooltip = true;  
  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 34. ThreeClickToSelectParagraph①

描述：鼠标连续三击选中段落，默认为 true。

参数：枚举值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.ThreeClickToSelectParagraph = false;  
  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 35. WordBreak①

描述：.单词断开模式。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.WordBreak="BreakAll";  
  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 36. SelectionTextIncludeBackgroundText②

描述：获取选择区域的文本是否包含输入域背景文本。默认为 true，包含背景文字。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.SelectionTextIncludeBackgroundText=false;
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 37. PageLineUnderPageBreak②

描述：分页线在分页符的下面,如果为 true，则分页线在分页符的上面,也就是分页符在上一页的最下面。本属性默认为 false。

参数：bool 值。

代码示例：可以结合使用。

```
ctl.DocumentOptions.BehaviorOptions.PageLineUnderPageBreak=false;
```

```
ctl.DocumentOptions.ViewOptions.ShowPageBreak=true;
```

```
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 38. OutputFieldBorderTextToContentText①

描述：是否将输入域的边界文本输出到内容文本中。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.OutputFieldBorderTextToContentText=false;
```

```
ctl.ApplyDocumentOptions();
```

备注：该文档选项影响 SaveDocumentToString('xml') 中的 BodyText 属性,当值为 false 的时候, BodyText 内容中不会带上输入域的边框。

6. 4. 39. EnableExpression①

描述：是否允许执行表达式功能（数值表达式、可见性表达式）。

参数：bool 值。

代码示例：

```
ctl.DocumentOptions.BehaviorOptions.EnableExpression=false;  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 40. AutoTranslateDescString

描述：英文字符标点符号下的全角字符自动转成半角。

参数：bool 值。

代码示例：需结合使用。

```
ctl.DocumentOptions.BehaviorOptions.AutoTranslateDescString=" A B C D E F G H  
I J K L M N O P Q R S T U V W X Y Z 0 1 2 3 4 5 6 7 8 9 。 + -";  
ctl.DocumentOptions.BehaviorOptions.AutoTranslateSourceString = "abcdefgh  
ijklmnopqrstuvwxyz0123456789.+-";  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 41. AutoTranslateSourceString

描述：英文字符标点符号下的半角字符自动转成全角。

参数：bool 值。

代码示例：需结合使用。

```
ctl.DocumentOptions.BehaviorOptions.AutoTranslateDescString=" A B C D E F G H  
I J K L M N O P Q R S T U V W X Y Z 0 1 2 3 4 5 6 7 8 9 。 + -";  
ctl.DocumentOptions.BehaviorOptions.AutoTranslateSourceString = "abcdefgh  
ijklmnopqrstuvwxyz0123456789.+-";  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 42. RemoveLastParagraphFlagWhenInsertDocument①

描述：在插入文档时删除要插入的文档的最后一个段落符号。默认 false。

参数：bool 值。

代码示例：需结合 InsertXmlString 方法使用。

```
ctl.DocumentOptions.BehaviorOptions.RemoveLastParagraphFlagWhenInsertDocument=true;  
  
ctl.ApplyDocumentOptions();
```

备注：无。

6. 4. 43. DocumentTextOutputMode①

描述：文档正文输出选项,本选项可以叠加使用，TXT 格式保存的文件内容。

参数：枚举值。（设置枚举值，就是根据导出正文文本+设置枚举值的设置）

Normal = 0,（导出正文，带输入域背景文本）

IncludeHeader = 1,（导出文本+页眉）

IncludeFooter = 2,（导出文本+页脚）

IncludeBody = 4,（导出纯文本，方法不带参数时，输入域背景文字也会导出）

IncludeBorderText = 10,（导出纯文本，带上输入域边框文本-标签文本-单位文本）

IncludeHiddenText = 20

```
ctl.DocumentOptions.BehaviorOptions.DocumentTextOutputMode="3";
```

```
ctl.ApplyDocumentOptions();
```

备注：可以设置多个枚举值，通过枚举值相加即可。比如设置 3，就是页眉+页脚效果

7. 前端事件

在控件加载和文档加载后会触发各类事件（回调函数）。包含

7.1. WriterControl_OnLoad 控件加载完成事件

在控件加载后触发。示例如下：

```
function WriterControl_OnLoad(rootElement) {  
    console.log(' 控件加载完毕')  
};
```

7.2. WriterControl_DocumentContentChanged 内容改变事件

在文档中内容改变时触发。示例如下：

```
function WriterControl_DocumentContentChanged(rootElement) {  
    console.log(rootElement.id + " 内容发生改变");  
};
```

7.3. DocumentLoad 文档加载事件

在文档加载后触发。该事件需要写在控件加载事件 WriterControl_OnLoad 中。

示例如下：

```
function WriterControl_OnLoad(rootElement) {  
    rootElement.DocumentLoad = function (eventSender, args) {  
        console.log("文件加载完毕");  
    };  
};
```

7.4. SelectionChanged 选择内容改变事件

在文档中选择内容时触发。该事件需要写在控件加载事件 WriterControl_OnLoad 中。示例如下：

```
function WriterControl_OnLoad(rootElement) {  
    rootElement.SelectionChanged = function (eventSender, args) {  
        var txt = "第" + (this.FocusedPageIndexBase0 + 1)  
            + "页, 第"  
            + this.CurrentLineIndexInPage + "行, 第"  
            + this.CurrentColumnIndex + "列.";  
        if (this.SelectionLength != 0) {  
            txt = txt + "选择了" +
```

```
        Math.abs(this.SelectionLength) + "个元素";
    }
    var p = this.SelectionStartPosition;
    txt = txt + " X:" + p.x + " Y:" + p.y;
    var selectMes =
        txt + " 当前字体:" + this.CurrentFontName + ", " + this.CurrentFontSize;
    var fontfamilySelect = document.getElementById("fontfamilySelect");
    fontfamilySelect.value = this.CurrentFontName;
    var fontSizeSelect = document.getElementById("fontSizeSelect");
    fontSizeSelect.value = this.CurrentFontSize;
    };
};
```

7.5. EventCanInsertObject 是否可以插入内容

判断当前位置是否可以插入内容，一般需要结合 rootElement.EventInsertObject 事件，在外部向编辑器内部拖拽数据时使用。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {
    rootElement.EventCanInsertObject = function (eventSender, eventArgs) {
        eventArgs.Result = true;
    };
};
```

7.6. EventInsertObject 插入内容事件

在向编辑器文档中插入内容或拖拽到编辑器内时触发。一般需要结合 rootElement.EventCanInsertObject 事件。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {
    rootElement.EventInsertObject = function (eventSender, eventArgs) {
        if (eventArgs.GetDataPresent("Text")) {
            var txt = eventArgs.GetData("Text");
            rootElement.DCExecuteCommand("InsertString", false, new
            Date().toLocaleDateString() + " 插入字符串 " + txt);
            eventArgs.Handled = true; // 设置Handle=true,表示事件已经处理了,
            不需要后续的默认处理。
        }
    };
};
```


7.7. EventShowContextMenu 右键菜单事件

通过该事件可以在编辑器文档中添加右键菜单。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {
    rootElement.EventShowContextMenu= function (eventSender, args) {
        let myWriterControl = GetCurrentWriterControl();
        var typename = args.ElementType;//当前的元素
        if (typename != null && typename != "") {
            typename = typename.toLowerCase();
            var options = [
                {
                    label: '撤销',
                    exec: () => {
                        myWriterControl.DCExecuteCommand ('Undo', false, null);
                    },
                },
                '-',
            ];
            if (typename == "xtexttablecellelement") { //单元格
                var options2 = [
                    '-',
                    {
                        label: '删除表格行',
                        exec: () => {
                            myWriterControl.DCExecuteCommand ("Table_DeleteRow", false,
null);
                        },
                    },
                ];
                options = options.concat(options2);
            }
            else {
            }
            ContextMenu(options, args, eventSender);
        };
    };
}

function ContextMenu(options, menuObj, rootElement) {
    if (options != null && Array.isArray(options)) {
        if (menuObj) {
            //获取到包裹元素
            var pageContainer =
rootElement.querySelector('[dctype=page-container]');
            //判断是否存在dcContextMenu的元素
```

```
var hasContextMenu =
pageContainer.querySelector('#dcContextMenu');
if (!hasContextMenu) {
    var meunDiv = document.createElement('div');
    meunDiv.setAttribute('id', 'dcContextMenu')
    pageContainer.appendChild(meunDiv);
    meunDiv.innerHTML = `<div class="dcMenu-line"></div>`;
    hasContextMenu = meunDiv;
    //判断是否有css
    var dcHead = document.head;
    //判断是否存在对应的css
    var hasContextMenuCss =
dcHead.querySelector('#ContextMenuCss');
    if (!hasContextMenuCss) {
        var newCssString = `
#dcContextMenu{
    position: absolute;
    margin: 0;
    padding: 2px;
    border-width: 1px;
    border-style: solid;
    background-color: #fafafa;
    border-color: #ddd;
    color: #444;
    box-shadow: rgb(204, 204, 204) 2px 2px 3px;
    width: 144px;
    overflow: hidden;
    /* left: 8px;
    top: 481.672px; */
    z-index: 110008;
    display: none;
}
#dcContextMenu > div:hover{
    color: rgb(0, 0, 0);
    border-color: rgb(183,210,255);
    background: rgb(234,242,255);
}
#dcContextMenu .dcMenu-line{
    position: absolute;
    left: 26px;
    top: 0;
    height: 100%;
    font-size: 1px;
    border-left: 1px solid #ccc;
```

```
        border-right: 1px solid #fff;
    }
    #dcContextMenu .dcMenu-item{
        position: relative;
        white-space: nowrap;
        cursor: pointer;
        margin: 0px;
        padding: 0px;
        overflow: hidden;
        border-width: 1px;
        border-style: solid;
        border-color: transparent;
    }
    #dcContextMenu .dcMenu-item .dcMenu-text{
        float: left;
        padding-left: 28px;
        font-size: 12px;
    }
    #dcContextMenu .dcMenu-icon{
        position: absolute;
        width: 16px;
        height: 16px;
        left: 2px;
        top: 50%;
        margin-top: -8px;
    }
    #dcContextMenu .dcMenu-sep{
        margin: 3px 0px 3px 25px;
        font-size: 1px;
        border-top: 1px solid #ccc;
        border-bottom: 1px solid #fff;
    }
    var ContextMenuCss = document.createElement('style');
    ContextMenuCss.setAttribute('id', 'ContextMenuCss');
    dcHead.appendChild(ContextMenuCss);
    ContextMenuCss.innerHTML = newCssString;
    //页面滚动
    pageContainer.addEventListener('scroll', function () {
        var hasContextMenu =
rootElement.querySelector('#dcContextMenu');
        if (hasContextMenu) {
            hasContextMenu.remove();
        }
    })
})
```

```
//鼠标按下
document.body.addEventListener('mousedown', function ()
{
    var hasContextMenu =
rootElement.querySelector('#dcContextMenu');
    if (hasContextMenu) {
        hasContextMenu.remove();
    }
})
//丢失焦点
window.addEventListener('blur', function () {
    var hasContextMenu =
rootElement.querySelector('#dcContextMenu');
    if (hasContextMenu) {
        hasContextMenu.remove();
    }
})
}
} else {
    //存在时，清空所有item和sep的元素
    var allItem =
hasContextMenu.querySelectorAll('.dcMenu-item, .dcMenu-sep');
    for (var i = 0; i < allItem.length; i++) {
        allItem[i].remove();
    }
}
var containerHeight = 0;
if (Array.isArray(options) && options.length > 0) {
    //根据options显示元素
    for (var option = 0; option < options.length; option++) {
        if (typeof options[option] == 'object') {
            var itemEle = document.createElement('div');
            itemEle.setAttribute('class', 'dcMenu-item');
            itemEle.style.cssText = 'height: 30px;';
            hasContextMenu.appendChild(itemEle);
            itemEle.innerHTML = `
                <div class="dcMenu-text" style="height:
30px; line-height: 30px;">${options[option].label}</div>
                <div class="dcMenu-icon"></div>
            `;
            itemEle.setAttribute('menuIndex', option)
            itemEle.addEventListener('mousedown', function (e) {
                e.stopPropagation();
                e.preventDefault();
            });
        }
    }
}
```

```

        })
        itemEle.addEventListener('click', function () {
options[this.getAttribute('menuIndex')].exec();
        //itemEle.getAttribute()
        //将hasContextMenu元素隐藏
        hasContextMenu.remove();
        })
        containerHeight += 30;
    } else if (typeof options[option] == 'string' &&
options[option] == '-') {
        var sepEle = document.createElement('div');
        sepEle.setAttribute('class', 'dcMenu-sep');
        hasContextMenu.appendChild(sepEle);
        containerHeight += 8;
    }
}
var pageElement = menuObj.PageElement;
containerHeight = containerHeight ? containerHeight + 6 : 0
hasContextMenu.style.height = containerHeight + 'px';
hasContextMenu.style.left = pageElement.offsetLeft +
menuObj.X + 'px';
hasContextMenu.style.top = pageElement.offsetTop + menuObj.Y
+ 'px';
hasContextMenu.style.display = 'block';
    }
}
}
}

```

通过以上代码可以在编辑器中添加右键菜单，并且可以自定义实现右键菜单的样式。

7.8. EventBeforeCut 剪切内容前事件

在编辑器内剪切内容之前触发该事件。示例代码如下：

```

function WriterControl_OnLoad(rootElement) {
    rootElement.EventBeforeCut = function (e, data) {
        console.log(e + " 复制前事件EventBeforeCut " + data)
        //return false;//拦截复制
    }
};

```

7.9. EventBeforeCopy 复制内容前事件

在编辑器内复制内容之前触发该事件。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {
    rootElement.EventBeforeCopy = function (e, copydata) {
        console.log(e + " 复制前事件EventBeforeCopy " + copydata)
        //return false;//拦截复制
    }
};
```

7.10. EventBeforePaste 粘贴内容前事件

在编辑器内粘贴内容之前触发该事件。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {
    rootElement.EventBeforePaste = function (copydata) {
        console.log(" 粘贴前事件EventBeforePaste " + copydata)
        //return false;//拦截粘贴
    }
};
```

7.11. EventBeforePrintPreview 打印预览前事件

在编辑器内打印预览之前触发该事件。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {
    rootElement.EventBeforePrintPreview = function (PrintPreviewDiv) {
        console.log(rootElement.id + " 打印预览前事件EventBeforePrintPreview " + args.Element);
    }
};
```

7.12. EventAfterPrintPreview 打印预览后事件

在编辑器内打印预览之后触发该事件。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {
    rootElement.EventAfterPrintPreview = function (PrintPreviewDiv) {
        console.log(rootElement.id + " 打印预览前事件 " + args.Element);
    }
};
```

7.13. EventBeforePrint 打印前事件

在编辑器内打印文档之前触发该事件。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {  
    rootElement.EventBeforePrint = function (printDoc) {  
        console.log(rootElement.id + " 打印前事件EventBeforePrint " +  
            args.Element);  
    }  
};
```

7.14. EventBeforePrintToGetHtml 在获取打印的 Html 前触发

在编辑器内获取打印的 Html 前触发该事件。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {  
    rootElement.EventBeforePrintToGetHtml = function (printDoc) {  
        console.log(rootElement.id + " 打印前事件EventBeforePrintToGetHtml"  
            + args.Element);  
    }  
};
```

7.15. ondocumentclick 鼠标单击事件

在编辑器内鼠标单击事件。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {  
    rootElement.ondocumentclick = function (e) {  
        console.log("点击事件");  
    }  
};
```

7.16. ondocumentdblclick 鼠标双击事件

在编辑器内鼠标双击事件。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {  
    rootElement.ondocumentdblclick = function (e) {  
        console.log("双击事件");  
    }  
};
```

7.17. QueryListItems 动态下拉列表事件

当输入域元素设置为动态下拉列表时，用户激活（一般使用单击或双击）该输入域时，会触发该事件。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {
    rootElement.QueryListItems = function (sender, eventObject) {
        console.log("QueryListItems获取的编号： "
            + eventObject.ElementID
            + "，获取的数据来源名称： "
            + eventObject.ListSourceName);
        eventObject.AddResultItemByTextValue("刘一", "liuyi");
        eventObject.AddResultItemByTextValue("陈二", "chener");
        eventObject.AddResultItemByTextValue("张三", "zhangsan");
        eventObject.AddResultItemByTextValue("李四", "lisi");
        eventObject.AddResultItemByTextValue("王五", "wangwu");
        eventObject.AddResultItemByTextValue("赵六", "zhaoliu");
        eventObject.AddResultItemByTextValue("孙七", "sunqi");
        eventObject.AddResultItemByTextValue("周八", "zhouba");
        eventObject.AddResultItemByTextValue("吴九", "wujiu");
        eventObject.AddResultItemByTextValue("郑十", "zhengshi");
    }
};
```

其中 AddResultItemByTextValue 接口，表示对该输入域的下拉列表赋值，第一个参数为文本、第二个参数为值。

此时会出现搜索框，用户输入内容后会在下拉列表中筛选数据。如果用户需要再次向数据库或缓存中查询数据，可以添加 EventChangeSearchInputSpellCode 事件。

7.18. EventChangeSearchInputSpellCode 下拉列表搜索事件

在触发 QueryListItems 事件后，用户在搜索框中输入内容会触发该事件，用户可以再次向数据库或缓存中查询数据。示例如下：

```
function WriterControl_OnLoad(rootElement) {
    rootElement.EventChangeSearchInputSpellCode= function (eventObject) {
        console.log("EventChangeSearchInputSpellCode获取的编号： "
            + eventObject.ElementID
```



```
        + "，获取的数据来源名称："  
        + eventObject.ListSourceName  
        + "，用户输入信息：" + eventObj);  
    for (var i = 0; i < 10; i++) {  
        eventObj.AddResultItemByTextValue(eventObj.Value + i, i);  
    }  
    eventObj.ChangeSpellCode();  
}  
};
```

7.19. EventAfterInsertSubDocuments 插入病程后事件

在编辑器内插入病程后触发该事件。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {  
    rootElement.EventAfterInsertSubDocuments = function (arr) {  
        console.log("添加病程回调函数EventAfterInsertSubDocuments:", arr);  
    }  
};
```

7.20. ondocumentmousedown 鼠标按下事件

在编辑器内鼠标按下触发该事件。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {  
    rootElement. ondocumentmousedown = function (arr) {  
        console.log("ondocumentmousedown:", arr);  
    }  
};
```

7.21. ondocumentmouseup 鼠标弹起事件

在编辑器内鼠标弹起触发该事件。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {  
    rootElement. ondocumentmouseup = function (arr) {  
        console.log("ondocumentmousedown:", arr);  
    }  
};
```

7.22. WriterControl_EventElementGotFocus 元素获取事件

在编辑器内元素获取焦点触发该事件。示例代码如下：

```
function WriterControl_EventElementGotFocus(rootElement, args) {  
    console.log(rootElement.id + " 聚 焦 事 件  
WriterControl_EventElementGotFocus " + args.Element);  
};
```

7.23. WriterControl_EventElementLostFocus 元素失去焦点事件

在编辑器内元素失去焦点触发该事件。示例代码如下：

```
function WriterControl_EventElementLostFocus( rootElement,args) {  
    console.log(rootElement.id + " 失 去 焦 点 事 件  
WriterControl_EventElementLostFocus " + args.TargetElement); //返回失去焦点元素  
对象  
};
```

7.24. WriterControl_EventElementMouseClicked 元素点击事件

在编辑器内点击元素触发该事件。示例代码如下：

```
function WriterControl_EventElementMouseClicked(rootElement, args) {  
    console.log(rootElement.id + " 元 素 点 击 事 件  
EventElementMouseClicked " + args.ElementID)  
}
```

7.25. WriterControl_EventElementMouseDb1Click 元素双击事件

在编辑器内双击元素触发该事件。示例代码如下：

```
function WriterControl_EventElementMouseDb1Click(rootElement, args) {  
    console.log(rootElement.id + " 元 素 双 击 事 件  
EventElementMouseDb1Click " + args.ElementID)  
}
```

7.26. ondocumentkeydown 键盘按下事件

在编辑器内通过按下键盘来触发该事件。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {
    rootElement.onkeydown = function (eventObject) {
        console.log(eventObject);
        if(eventObject.ctrlKey==true&& eventObject.key=="C"){
            console.log(“监控到用户按下了Ctrl和C键的组合键”)
        }
    }
};
```

7.27. EventDialogChangeProperties 元素属性对话框点击确定按钮回调函数

在元素属性对话框中点击确定按钮来触发该事件。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {
    rootElement.EventDialogChangeProperties = function (ele) {
        console.log(ele,“元素属性”);
    }
};
```

7.28. EventAfterPageCountChange 文档换页触发事件

在编辑器初始化的时候或者手动新增页，删页触发事件。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {
    rootElement.EventAfterPageCountChange = function (ele) {
        console.log(“文档进行了换页”);
    }
};
```

7.29. EventCanvasCountChange 新增删除页触发事件

在编辑器加载文档成功后，用户手动新增页、删除页触发该事件。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {
    rootElement.EventCanvasCountChange = function (ele) {
        console.log("文档进行了换页");
    }
};
```

7. 30. EventPromptProtectedContent 提示内容受保护的事件

在编辑器加载文档成功后，用户操作出现提示内容受保护会触发事件，在事件中可以自定义提示内容。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {
    rootElement.EventPromptProtectedContent = function (ele) {
        //用户自定义提示内容
        args.Handled = true;//阻止编辑器自带提示弹出
    }
};
```

7. 31. EventDropdownEditor 用户自定义弹窗事件

在编辑器加载成功后，用户操作下拉输入域，在事件中可以自定义弹窗内容。示例代码如下：

```
function WriterControl_OnLoad(rootElement) {
    rootElement.EventDropdownEditor = function () {
        console.log(111111111111);
    };
};
```

8. 前端接口

8. 1. 文档

8. 1. 1. *生成预览 html——不推荐使用

接口：

```
string getHtmlByXMLString(string parameter)
```

描述:

将制定的 XML 转换为 html 字符串。转换后的 html 只具备阅读功能，不能呈现结构化内容。该接口和 getHtmlByXMLBase64String 具有相同的功能，只是参数不同。

此时生成的 html 不具有分页、且 xml 中段落设置的对齐方式（居右、居中）失效。

参数:

参数名	是否必填	数据类型	说明
parameter	必填	字符串	需要转换的 XML 字符串，该字符串通过编辑器可以加载。如果是当前文档的 XML，一般通过 <code>ctl.SaveDocumentToString('xml')</code> 返回，再使用。

返回值:

数据类型	说明	备注
字符串	返回生成的 html 内容	

代码示例:

```
let str = ctl.SaveDocumentToString('xml');
var result = ctl.getHtmlByXMLString(str);
var oPrntWin = window.open("", "",
    "width=450,height=470,left=400,top=100,menubar=yes,
    toolbar=no,location=no,scrollbars=yes,resizable=yes");
oPrntWin.document.write(result);
```

结合使用:

无

接口类型:

纯前端实现, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用

兼容接口:

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化:

无任何改变

推荐接口:

用例地址:

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=getHtmlByXMLString>

8. 1. 2. *加载指定类型的病历模板文档

接口:

```
Bool LoadDocumentFromString(string strData, string strFormat,
string specifyLoadPart,function errorCallback)
```

描述:

加载指定格式的文档内容。目前支持的文档格式包含 XML、json、RTF、html、Text 格式。

参数:

参数名	是 否 必 填	数据类型	说明
strData	必填	字符串	需要加载的文档的字符串内容
strFormat	非必填	字符串	需要加载的文档的字符串格式
specifyLoadPart	非必填	字符串	在控件中当前文档的指定位置加载对应的内容，它包含 header 页眉、footer 页脚、headerfooter 页眉页脚、body 正文。可以为空。当需使用该参数时，null 表示全文加载，不允许使用空字符串""
errorCallback	非必填	回调函数	当 json 数据格式异常触发的回调函数，且只对 json 格式数据才会进行校验

返回值:

数据类型	说明	备注
	无返回值	

代码示例:

```
ctl.LoadDocumentFromString(strxmldate, 'xml');
ctl.LoadDocumentFromString(strxmldate, 'xml', 'body');
ctl.LoadDocumentFromString(strxmldate, 'xml', 'body',function);
```

```
ctl.LoadDocumentFromString(strxmldata, 'xml', '', function);
```

结合使用：

DocumentLoad 事件（文档加载完成事件）-当编辑器把文档渲染成功之后触发的事件，当编辑器初始化的时候需要加载文档并同步进行数据源绑定，得在 WriterControl_OnLoad 事件中先定义加载完成事件（DocumentLoad），然后再调用加载文档的接口，否则无效。第二个、第三个参数不管什么值，都触发该事件

接口类型：

纯前端实现，不经过服务端

有无限制：

编辑器初始化成功之后进行调用（除打印预览、续打、区域选择模式外），阅读模式、编辑器只读模式下允许重复加载文档（2024-06 月后续版本支持）。

兼容接口：

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化：

五代下该方法相比四代来说，新增支持加载 json 格式的文档和新增一个参数，用来校验 json 数据是否正常的回调函数

用例地址：

<http://www.dcwriter.cn:8022/apiDocument/edit/index.html?apiname=LoadDocumentFromString>

8.1.3. *加载 base64 格式病历模板文档

接口：

Bool LoadDocumentFromBase64String(string strData, string strFormat, string specifyLoadPart)

描述：

加载指定格式的文档内容。目前支持的文档格式 XML-base64 数据格式。

参数：

参数名	是否必填	数据类型	说明
strData	必填	字符串	需要加载的文档的 Base64 格式字符串内容
strFormat	必填	字符串	需要加载的文档的 Base64 格式字符串的文档

			格式
specifyLoadPart	非必填	字符串	在控件中当前文档的指定位置加载对应的内容，它包含 header 页眉、footer 页脚、headerfooter 页眉页脚、body 正文。可以为空，或者空字符串表示全文加载。

返回值：

数据类型	说明	备注
	无返回值	

代码示例：

<pre>ctl.LoadDocumentFromBase64String(strxmldata, 'xml'); ctl.LoadDocumentFromBase64String(strxmldata, 'xml', 'body');</pre>

结合使用：

DocumentLoad 事件（文档加载完成事件）-当编辑器把文档渲染成功之后触发的事件，当编辑器初始化的时候需要加载文档并同步进行数据源绑定，得在 WriterControl_OnLoad 事件中先定义加载完成事件（DocumentLoad），然后再调用加载文档的接口，否则无效。第二个、第三个参数不管什么值，都触发该事件

接口类型：

纯前端实现，不经过服务端

有无限制：

编辑器初始化成功之后进行调用（除打印预览模式外）

兼容接口：

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化：

接口完全一致

用例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=LoadDocumentFromBase64String>

8.1.4. *加载文档中不同部位组成新的病历模板

接口：

Bool LoadDocumentFromMixString(json parameter)

描述:

分别提供页眉，文档体，页脚的 XML 来组合成一个文档进行加载。

参数（不区分大小写）:

参数名	是否必填	数据类型	说明
parameter	必填	Json	以下参数是 json 的内容
+ fileformat	必填	字符串	需要加载的文档内容字符串的文档格式，支持 xml、json
+ usebase64string	必填	bool	当前加载的文档内容是否是 base64 格式。为 base64 格式，文档数据不能为 json
+ headercontent	必填	字符串	传的病历数据只提取页眉数据加载到组合的页眉上，为 null 表示清空页眉区域
+bodycontent	必填	字符串	传的病历数据全部解析，然后根据 headercontent、footercontent 传的值对 bodycontent 先加载出来的文档重新处理页眉页脚处。
+footercontent	必填	字符串	传的病历数据只提取页脚数据加载到组合的页脚上，为 null 表示清空页脚区域

返回值:

数据类型	说明	备注
	无返回值	

代码示例:

```
var options = {  
    FileFormat: "xml",  
    UseBase64String: "false",  
    HeaderContent: header,  
    BodyContent: body,  
    FooterContent: footer  
}  
ctl.LoadDocumentFromMixString(options);
```

结合使用:

DocumentLoad 事件（文档加载完成事件）-当编辑器把文档渲染成功之后触发的事件，当编辑器初始化的时候需要加载文档并同步进行数据源绑定，得在 WriterControl_OnLoad 事件中先定义加载完成事件（DocumentLoad），然后再调用加载文档的接口。

使用场景：

可以只维护一个只有页眉正文是空白的主模板，然后只制作只有正文的模板，然后客户根据需要自由来拼接页眉，或者替换掉原文档的页眉页脚

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器初始化成功之后进行调用（除打印预览模式外）

兼容接口：

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化：

参数还是一样，新增了支持 json 格式的文档数据

用例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=LoadDocumentFromMixString>

8.1.5. *保存返回指定类型的字符串病历文档

接口：

String SaveDocumentToString(json parameter)

描述：

保存文档内容。

参数（不区分大小写）：

参数名	是否必填	数据类型	说明
parameter	必填	Json	以下参数是 json 的内容
+ fileformat	必填	字符串	需要保存的文档内容字符串的

			文档格式。支持保存的格式包含 XML、json、html、rtf、text 等。
+ commitusertrace	非必填	Bool、字符串	是否清除文档痕迹。默认 false。
+ outputformatxml	非必填	bool、字符串	是否输出格式化的内容，如果保存为 XML、json 时，格式化的内容占用的内存会比较大，但有利于人工阅读；非格式化的内容，占用的内存会比较小，但不利于人工阅读。默认 true。
+savebase64string	非必填	Bool、字符串	是否保存为 base64 的内容但是 fileformat 参数不能为 json。默认 false。
+specifysavepart	非必填	字符串	保存文档的部分内容，包含 header 页眉、footer 页脚、headerfooter 页眉页脚、body 正文。为空或者空字符串时，全文保存。
+cleardatabindingcontent	非必填	Bool、字符串	清空病历文档中输入域设置了带有数据源或者绑定路径的内容，该输入域不管是数据源绑定的值还是手动输入的内容都会全部清空。
+insertlasttablerowtopagebottom	非必填	Bool、字符串	当文档正文处最后面是表格时，保存出来的数据是把正文最后面的空白区域新增行铺满表格。默认 false。

+attachedcustomattributes	非 必 填	Json 格 式	保存的同时设置文档自定义属性信息

返回值:

数据类型	说明	备注
String	需要保存的字符串格式的内容	

代码示例:

```
var options = {  
    FileFormat: "xml",  
    commitusertrace: "true"  
}  
let res=ctl.SaveDocumentToString(options);
```

该接口支持减省写法, 如下:

```
let res=ctl.SaveDocumentToString("xml");  
let res=ctl.SaveDocumentToString("json");  
let res=ctl.SaveDocumentToString("rtf");  
let res=ctl.SaveDocumentToString("text",{  
    IncludeBackgroundText: false,//是否包含背景文字---默认值 false  
    IncludeBorderText: true,//是否包含前后边框---默认值 false  
    IncludeHiddenText: false,//是否包含隐藏文本---默认值 false  
    IncludeLabelUnitText: false,//是否包含标签、单位文本---不传参数默认值为 true  
    IncludeLogicDeletedContent: false//是否包含逻辑删除的文本内容---默认值 false  
});  
let res=ctl.SaveDocumentToString("html");  
let res=ctl.SaveDocumentToString();默认保存为XML
```

效果影响:

ctl.DocumentOptions.BehaviorOptions.OutputFormattedXMLSource=false, 返回的 xml 数据不带格式化

ctl.DocumentOptions.BehaviorOptions.DocumentTextOutputMode 不同枚举值。会影响返回纯文本内容。详细枚举值请看对应文档选项节点

相同效果:

与 ctl.GetXmlContent() 方法返回的数据一样

SaveDocumentToStringAsync 该方法没有了异步操作, 因为五代不需要进行前后端交互, 只有同步执行。所以 SaveDocumentToStringAsync 方法走的就是 SaveDocumentToString() 方法

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器初始化成功之后进行调用都行

兼容接口：

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化：

该方法在五代编辑器下第二个参数去掉了，且新增了保存为 json 格式的数据

用例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=SaveDocumentToString>

注意：

当需要通过该方法获取文档中纯文本内容时，需要注意方法中第二个参数和文档选项 DocumentTextOutputMode 结合使用的关联（单位文本、标签文本）处理会比较特殊，需要经过大量测试才能决定是否满足业务需求。

8.1.6. *保存返回指定类型的 base64 字符串病历文档

接口：

String SaveDocumentToBase64String(string fileformat)

描述：

保存文档内容指定格式的 base64 内容。

参数：

参数名	是否必填	数据类型	说明
fileformat	必填	字符串	需要保存的文档的 base64 内容字符串的文档格式。支持保存的格式包含 XML、pdf、html、rtf、text 等。

返回值：

数据类型	说明	备注

String	需要保存的字符串格式的内容	

代码示例:

```
let res=ctl.SaveDocumentToBase64String("xml");
```

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后进行调用都行

兼容接口:

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化:

无任何改变

用例地址:

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=SaveDocumentToBase64String>

注意:

java 作服务端也能通过该方法生成 rtf 的 base 数据

8.1.7. *保存返回只包含文档正文指定类型的字符串数据

接口:

String SaveBodyDocumentToString(string fileformat)

描述:

按照指定的格式只保存正文内容。

参数:

参数名	是否必填	数据类型	说明
fileformat	必填	字符串	需要保存的文档的 base64 内容字符串的文档格式。支持保存的格式包含 XML、json、html、rtf、text 等。

返回值:

数据类型	说明	备注
String	需要保存的字符串格式的内容	

代码示例：

```
let res=ctl.SaveBodyDocumentToString(“xml”);  
let res=ctl.SaveBodyDocumentToString(“json”);
```

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器初始化成功之后进行调用都行

兼容接口：

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化：

新增支持保存为 json 数据

用例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=SaveBodyDocumentToString>

8.1.8. *下载本地指定类型文件

接口：

```
Bool DownloadFile(string strFormat, string strFileName)
```

描述：

把当前文档下载为指定的格式。

参数：

参数名	是否必填	数据类型	说明
strFormat	非必填	字符串	需要下载的文档格式。支持保存的格式包含 XML、json、html、rtf、(text,txt)、PDF、(longimg,longimage)、local.pdf,ofd 等。默认是 XML 格式。

strFileName	非必填	字符串	下载后的文件名称

返回值:

数据类型	说明	备注
Bool	需要保存的字符串格式的内容	

代码示例:

```
ctl.DownloadFile("xml", "123.xml");
ctl.DownloadFile("json");
ctl.DownloadFile("ofd");
//生成ofd二进制数据
ctl.DownloadFile('ofd','',function (params) {
    console.log(params)
})
```

该接口支持简省写法,

```
let res=ctl.DownloadFile();默认xml格式
```

解释说明:

pdf 参数是会经过服务端然后才下载成 pdf，也就是说必须服务器必须部署后台服务才能使用该功能

local.pdf 是 2024-02 新增功能，纯前端生成 pdf 文件，服务器不需要部署后台服务。可以使用 java 当服务端使用

功能受限:

longimage、longimg 生成长图片是旗舰版的功能，标准版不支持

结合使用:

无

接口类型:

当参数为 rtf、pdf 为非纯前端接口，需要经过服务端转换才行，也就是需要.net core 部署的服务

有无限制:

编辑器初始化成功之后进行调用都行

兼容接口:

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化:

看解释说明，该接口新增 local.pdf 参数

用例地址:

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=DownLoadFile>

8.1.9. *当前光标处插入 xml 内容片段

接口:

```
Bool InsertXmlString(string content)
```

描述:

在当前位置插入 XML 文字中正文的内容。

参数:

参数名	是否必填	数据类型	说明
content	必填	字符串	Xml 格式的文档内容

返回值:

数据类型	说明	备注
Bool	插入是否成功	

代码示例:

```
let res=ctl.InsertXmlString(strXML);
```

相同效果:

与 InsertXmlBase64String 方法一致, 只不过传入的数据格式不同

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器初始化成功之后当编辑器存在焦点调用该接口才有效果。(普通表单模式在非输入域内调用该接口也无效)

兼容接口:

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化:

五代编辑器只保留了该方法的第一个参数。有使用其它参数的请联系南京都昌

用例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=InsertXmlString>

8.1.9. *当前光标处插入 xml 内容片段并设置内容样式

接口:

Bool InsertXmlStringByStyle(string content,json style)

描述:

在当前位置插入 XML 文本中正文的内容。

参数:

参数名	是否必填	数据类型	说明
content	必填	字符串	Xml 格式的文档内容
style	非必填	json	插入内容的样式

返回值:

数据类型	说明	备注
Bool	插入是否成功	

代码示例:

```
let Style = {
    "Bold": true,
    "ColorString": "red",
    "FontName": "楷体",
    "FontSize": 18
};
let res=ctl.InsertXmlStringByStyle(strXML,Style);
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后当编辑器存在焦点调用该接口才有效果。（普通表单模式在非输入域内调用该接口也无效）

兼容接口:

五代独有接口。

接口优化:

无

用例地址:

无

8.1.10. *当前光标处插入 base64 内容片段

接口:

Bool InsertXmlBase64String(string content)

描述:

在当前位置插入 base64 格式的 XML 内容。

参数:

参数名	是否必填	数据类型	说明
content	必填	字符串	Base64 格式的 Xml 格式的文档内容

返回值:

数据类型	说明	备注
Bool	插入是否成功	

代码示例:

```
let res=ctl.InsertXmlBase64String(strXML);
```

相同效果:

与 InsertXmlString 方法一致，只不过传入的数据格式不同

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后当编辑器存在焦点调用该接口才有效果。（普通表单模式在非输入域内调用该接口也无效）

兼容接口:

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化:

五代编辑器只保留了该方法的第一个参数。有使用其它参数的请联系南京都昌

用例地址:

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=InsertXmlBase64String>

8. 1. 11. *在指定输入域里面插入 xml 内容片段

接口:

Bool InsertXmlById (string content, string id)

描述:

在指定的输入域内插入文档。

参数 (不区分大小写):

参数名	是 否 必填	数据类 型	说明
parameter	必填	Json	以下参数是 json 的内容
+ file	必填	字符串	病历 xml 数据、病历 json 数据
+ format	非 必 填	字符串	文档格式: xml
+ base64	非 必 填	bool	是否为 base64 数据格式
+position	非 必 填	字符串	插入在指定元素位置: start 为插入点在当前元素最前方、 End 为插入点在当前元素的最 后方
+logundo	非 必 填	Bool	是否记录到撤销操作
ele	必填	objetc	元素编号或者是元素对象或者 是元素的 NativeHandle 值

返回值:

数据类型	说明	备注
Bool	插入是否成功	

代码示例:

```
var obj = {
    "file": xml, //xml 文档
    "format": "xml", //xml 文档格式
    "base64": "false", //是否是 base64 字符串
    "position": "start", //插入到元素最前方（输入域起始边框后面）
    "logundo": true //记录到撤销操作上
}
console.log(obj, '=====obj')
ctl.InsertXmlById(obj, 'field1');//给输入域ID为field1插入xml片
段
```

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后当编辑器存在该元素接口才有效果（预览模式下无效）

兼容接口:

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化:

五代编辑器只方法的第三个参数进行了优化，且把第一个参数中 object 里面的 type、ClearFormat 属性去掉，新增了 logundo 属性

用例地址:

<http://www.dcwriter.cn:8022/apiDocument/edit/index.html?apiname=InsertXmlById>

8. 1. 11. *在指定输入域里面插入 xml 内容片段并集成当前元素段落样式

接口:

Bool InsertXmlByIdUseCurrentStyle(string content, string id)

描述:

在指定的输入域内插入文档并且插入的内容会集成当前光标处的段落样式

参数（不区分大小写）:

参数名	是否必填	数据类型	说明
parameter	必填	Json	以下参数是 json 的内容
+ file	必填	字符串	病历 xml 数据、病历 json 数据
+ format	非必填	字符串	文档格式: xml
+ base64	非必填	bool	是否为 base64 数据格式
+position	非必填	字符串	插入在指定元素位置: start 为插入点在当前元素最前方、End 为插入点在当前元素的最后方
+logundo	非必填	Bool	是否记录到撤销操作
ele	必填	objetc	元素编号或者是元素对象或者是元素的 NativeHandle 值

返回值:

数据类型	说明	备注
Bool	插入是否成功	

代码示例:

```
var obj = {
    "file": xml,//xml 文档
    "format": "xml",//xml 文档格式
    "base64": "false",//是否是 base64 字符串
    "position": "start",//插入到元素最前方（输入域起始边框后面）
    "logundo": true //记录到撤销操作上
}
console.log(obj, '=====obj')
```

```
ctl.InsertXmlByIdUseCurrentStyle(obj, 'field1');//给输入域ID为field1插入xml片段
```

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后当编辑器存在该元素接口才有效果（预览模式下无效）

兼容接口:

五代独有接口。

接口优化:

无

用例地址:

无

8. 1. 12. *动态修改文档状态

接口:

Bool resetModified(bool parameter)

描述:

重置文档的修改状态为指定的状态。

参数:

参数名	是否必填	数据类型	说明
parameter	必填	Bool	重置为指定的状态，如果参数为 false，更新为未修改状态，会把所有已修改的元素的状态重置为未修改状态。如果参数是 true，把编辑器状态重置为未修改状态。

返回值:

数据类型	说明	备注
Bool	重置是否成功	

代码示例:

```
let res=ctl.resetModified(false);  
let res=ctl.resetModified(true);
```

结合使用:

无

相同效果:

ctl.Modified 属性设置的值效果相同

接口类型:

纯前端方法，不经过服务端

有无限制:

编辑器初始化成功之后进行调用都行（预览模式除外）

兼容接口:

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化:

无任何区别

用例地址:

<http://www.dewriter.cn:8022/apiDocument/edit/index.html?apiname=resetModified>

8.1.13. *已修改文档修改成未修改

接口:

Bool ResetDocumentModified()

描述:

重置文档的修改状态为未修改状态的状态。

参数:

无参数。

返回值:

数据类型	说明	备注
Bool	重置是否成功	

代码示例:

```
let res=ctl.ResetDocumentModified();//把编辑器的状态和文档元素的状态重置为未修改  
该接口等价于ctl.resetModified(false);
```


结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器初始化成功之后进行调用都行（预览模式除外）

兼容接口：

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化：

无任何区别

用例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=ResetDocumentModified>

8.1.14. *获取文档状态

接口：

Bool getModified()

描述：

获取当前文档的修改状态。

参数：

无参数。

返回值：

数据类型	说明	备注
Bool	获取当前文档的修改状态	

代码示例：

```
let res=ctl.getModified();  
//等价于  
ctl.Modified属性获取的值
```

使用场景：

当编辑器加载的文档产生了变化的时候，该接口的返回值是 true

相同效果：

与ctl.Modified属性获取的值相同含义

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器初始化成功之后进行调用都行（预览模式除外）

兼容接口：

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化：

无任何区别

用例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=getModified>

8. 1. 15. *清空正文内容

接口：

Void ClearDocumentBody()

描述：

清空文档的正文内容。

参数：

无参数。

返回值：

无返回值。

代码示例：

```
ctl.ClearDocumentBody();
```

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器初始化成功之后进行调用都行，但阅读视图模式、预览模式、续打模式、表单模式、只读模式下不能调用该接口

兼容接口：

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化：

无任何区别

用例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=ClearDocumentBody>

8.1.16. *动态控制编辑器只读

接口：

Bool SetControlReadonly(bool parameter)

描述：

设置控件是否只读, 只读状态下即使病历里面的元素设置了不只读也不起效果，且病历上面不会显示光标。

参数：

参数名	是否必填	数据类型	说明
parameter	必填	Bool	传 true 值，表示对编辑器设置只读，传 false，对编辑器取消只读。

返回值：

数据类型	说明	备注
Bool	设置是否成功	

代码示例：

```
let res=ctl.SetControlReadonly(false);
```

相同效果：

通过编辑器.Readonly 属性也能对编辑器进行只读切换

方法扩展：

通过接口对元素赋值(SetElementTextByID)（通过 ID 赋值、数据源绑定）、删除

(DeleteElementByID)没有影响

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器初始化成功之后进行调用都行，但预览模式调用该接口无效

兼容接口：

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化：

无任何区别

示例地址：

<http://www.dcwriter.cn:8022/apiDocument/edit/index.html?apiname=SetControlReadonly>

8.1.17. *获取文档总页数

接口：

Int GetDocumentPageNum()

描述：

获取当前文档的总页数。

参数：

无参数。

返回值：

数据类型	说明	备注
Int	当前文档的总页数	

代码示例：

```
let res=ctl.GetDocumentPageNum();
```

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器初始化成功之后进行调用都行

兼容接口：

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化：

无任何区别

示例地址：

<http://www.dcwriter.cn:8022/apiDocument/edit/index.html?apiname=GetDocumentPageNum>

8. 1. 18. *多文档合并生成 base64 格式的 pdf

接口：

String GetPDFByFiles(json parameter)

描述：

获取文档的 PDF 格式的 base64 字符串，用来供医生预览或者打印使用。

参数（不区分大小写）：

参数名	是否必填	数据类型	说明
parameter	必填	Json	以下参数是 json 的内容
+files	必填	字符串	xml 文档内容字符串数组。支持 xml 数据和 json 数据
+base64	非必填	Bool、字符串	是否是 base64 字符串，默认 false。
+megedoc	非必填	Bool	是否通过病程的方式组合成一个新文档（该模式下组合的新文档是使用的相同的页眉页脚），否则使用传的文档组中各个的文档的页眉页脚（splitmode 为 pagebreak 模式时）
+modefile	非必填	字符串	当 megedoc 设置为 true 时，modefile 是必

		串	填，表示加载的 XML 文档数组都使用这个 modefile 文档的页眉页脚。
+ splitmode	非必填	字符串	表示各个文件合并时中间的分隔模式，“none”不分隔，“pagebreak”换新页，“linebreak”，用换行符分隔，“line”，用水平线分隔。默认是 none 不分隔。
+ filename	非必填	字符串	如果是下载 PDF，表示下载的文件名称，默认是空字符串，使用随机值。
+ resulttype	非必填	字符串	返回类型：“DownloadFile”表示直接下载文件；“Base64String”表示返回 pdf 二进制的 BASE64 字符串，默认值是 Base64String。
+forceblacktextcolor	非必填	字符串	设置导出的 PDF 强制使用黑色字体，默认 false。

返回值：

数据类型	说明
String	当 resulttype 属性为 DownloadFile 时，此时会直接下载 PDF 文件，不返回任何内容。如果 resulttype 属性为 Base64String 时，此时返回 PDF 文件的 base64 内容。

代码示例：

```
var options={
    "files": arr,//数组对象，存的是xml文档
    "base64": "false",//是否是base64字符串
    "megedoc": "false"//是否合并
    "modefile": "xmlstring"//病程合并模式下提供页眉页脚的主文档XML字符串
    "splitmode": "none"//各个文件合并时中间的分隔模式，“none”不分隔，“pagebreak”
    换新页，“linebreak”，用换行符分隔，“line”，用水平线分隔
    "filename": ""//导出PDF的文件名
    "resulttype": ""//返回类型：“DownloadFile”表示直接下载文件；“Base64String”表示
    返回pdf二进制的BASE64字符串
    "forceblacktextcolor": "false"//设置导出的PDF强制使用黑色字体
}
let res=ctl.GetPDFByFiles(options);
```

注意：

编辑器新增 GetPDFByFiles2 方法，该方法跟 GetPDFByFiles 方法使用一致，但是

GetPDFByFiles2 方法新增一个回调函数，在回调函数中获取返回合并之后的 base64-pdf 数据。该方法是 java 部署使用的接口，java 部署不能调用 GetPDFByFiles 方法

```
var ctl = document.getElementById("myWriterControl");

ctl.

(options,function(a){

    console.log(a,"pdf 方法返回值")

});
```

结合使用：

无

接口类型：

后台接口，会经过服务端，只能使用.net 方式部署服务才能调用

有无限制：

编辑器初始化成功之后进行调用都行

示例地址：

<http://www.dcwriter.cn:8022/apiDocument/edit/index.html?apiname=GetPDFByFiles>

8.1.19. *多文档合并生成预览下的 html

接口：

String getHtmlByFiles(json parameter,callback)

描述：

给定若干文档作为子文档合并后转换成预览下的 HTML，用来供医生预览或者打印

参数：

参数名	是否必填	数据类型	说明
parameter	必填	Json	以下参数是 json 的内容
+files	必填	字符串	文档内容字符串数组。支持 xml 数据和 json 数据

+format	非必填	字符串	文档内容格式，包含 json 和 xml。默认 xml
+base64	非必填	字符串	文档内容是否是 base64 格式。默认 false。
+magedoc	非必填	bool	是否通过病程的方式组合成一个新文档（该模式下组合的新文档是使用的相同的页眉页脚），否则使用传的文件组中各个的文档的页眉页脚（splitmode 为 pagebreak 模式时）
+modefile	非必填	字符串	表示加载的 XML 文档数组都使用这个 modefile 文档的页眉页脚，以病程记录的方式加载文档数组。
+ splitmode	非必填	字符串	表示各个文件合并时中间的分隔模式，“none”不分隔，“pagebreak”换新页，“linebreak”，用换行符分隔，“line”，用水平线分隔。默认是 none 不分隔。
callBack	非必填	函数	

返回值：

数据类型	说明
String	返回 html 内容字符串。

代码示例：

```
var options={
    "files": arr,//数组对象，存的是xml文档
    "base64": "false",//是否是base64字符串
    "modefile": "xmlstring"//病程合并模式下提供页眉页脚的主文档XML字符串
    "splitmode": "none"//各个文件合并时中间的分隔模式，“none”不分隔，“pagebreak”
    换新页，“linebreak”，用换行符分隔，“line”，用水平线分隔
}
let res=ctl.getHtmlByFiles(options);
```

接口扩展：

当 magedoc 和 modefile 参数没有定义的时候，组合生成的预览 html 是不带上页眉页脚数据的，只有传的文件组中各自的正文，参数正常定义，组合生成的预览 html 页眉页脚统一使用 modefile 提供的数据

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器初始化成功之后进行调用都行

兼容接口：

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化：

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=getHtmlByFiles>

8.1.20. *生成预览 base64 格式 html---不推荐使用

接口：

```
String getHtmlByXMLBase64String(string parameter)
```

描述：

根据 base64 字符串获取到 HTML 字符串。转换后的 html 只具备阅读功能，不能呈现结构化内容。该接口和 getHtmlByXMLString 具有相同的功能，只是参数不同。

此时生成的 html 不具有分页功能、且 xml 中段落设置的对齐方式（居右、居中）失效。

参数：

参数名	是否必填	数据类型	说明
parameter	必填	字符串	需要转换 html 的 xml 文档的 base64 内容字符串

返回值：

数据类型	说明	备注
String	转换后的 html 字符串	

代码示例：

```
let res=ctl.getHtmlByXMLBase64String(strXML);
```

结合使用：

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后进行调用都行

兼容接口:

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=getHtmlByXMLBase64String>

8. 1. 21. *两文档对比差异

接口:

String MergeDocumentByFileContent(json parameter,function errorCallback)

描述:

合并对比文档内容。该接口需要访问服务端，通过.net 或者.net core 部署的服务加载五代编辑器，就能正常调用该方法。

参数:

参数名	是否必填	数据类型	说明
parameter	必填	Json	以下参数是 json 的内容
+OldUserID	必填	字符串	第一个文档的文档用户编号
+OldUserName	必填	字符串	第一个文档的文档用户名称
+OldSaveTime	必填	字符串	第一个文档的文档保存时间
+OldPermissionLevel	必填	整数型	第一个文档的文档用户登记，整数型
+OldFileContent	必填	字符串	第一个文档的文档内容
+OldClientName	必填	字符串	第个文档的文档客户端电脑名称

+NewUserID	必填	字符串	第二个文档的文档用户编号
+NewUserName	必填	字符串	第二个文档的文档用户名称
+NewSaveTime	必填	字符串	第二个文档的文档保存时间
+NewPermissionLevel	必填	整数型	第二个文档的文档用户登记，整数型
+NewFileContent	必填	字符串	第二个文档的文档内容
+NewClientName	必填	字符串	第二个文档的文档客户端电脑名称
+FileFormat	必填	字符串	文档的格式是 xml 还是 json，默认是 xml。
+ReturnHTML	必填	字符串	若为 false 则返回合并文档的 XML 字符串，否则返回预览 HTML。
errorCallback	非必填	回调函数	加载完成后的回调函数

返回值：

数据类型	说明	备注
Bool	参数 ReturnHTML 若为 false 则返回合并文档的 XML 字符串，否则返回预览 HTML	

代码示例：

```

var args = {
    OldUserID: "zhangsan", // 保存旧文件的用户名
    OldUserName: "张三主治医师", // 保存旧文件的用户姓名
    OldSaveTime: "2018-3-4 12:30:02", // 保存旧文件的时间
    OldPermissionLevel: 0, // 旧用户授权等级
    OldFileContent: strDemoFileXml, // 旧文件内容
    OldClientName: "192.168.0.4", // 保存旧文件的客户端名称
    NewUserID: "lisi", // 保存新文件的用户名
    NewUserName: "李四主任医师", // 保存新文件的用户姓名
    NewSaveTime: "2018-3-8 9:30:14", // 新文件保存时间
    NewPermissionLevel: 2, // 新用户授权等级
    NewFileContent:
        document.WriterControl.SaveDocumentToString(), // fileContent2, // 新文件内容
    NewClientName: "192.168.0.99", // 新用的客户端名称
    FileFormat: "xml", // 文件格式
    ReturnHTML: false // false 返回结果对比文档 XML，设为 true 则返回预览的 HTML
};
ctl.MergeDocumentByFileContent(args, function (a) {
    ctl.DocumentOptions.SecurityOptions.EnablePermission = "true"
    ctl.DocumentOptions.SecurityOptions.EnableLogicDelete = "true"
});

```

```
ctl.DocumentOptions.SecurityOptions.ShowLogicDeletedContent = "true"
ctl.DocumentOptions.SecurityOptions.ShowPermissionTip = "true"
ctl.DocumentOptions.SecurityOptions.ShowPermissionMark = "true"
ctl.ApplyDocumentOptions();
ctl.LoadDocumentFromString(a, "xml");
});
```

注意：

编辑器新增方法 MergeDocumentByFileContent2 方法，该方法跟 MergeDocumentByFileContent 方法使用一致，但是 MergeDocumentByFileContent2 方法新增一个回调函数，在回调函数中获取返回合并之后的 xml 数据。该方法是 java 部署使用的接口，java 部署不能调用 MergeDocumentByFileContent 方法

```
let ctl = document.getElementById("myWriterControl");
ctl.MergeDocumentByFileContent(args, function (a) {
    ctl.setAttribute("servicepageurl", back);
    ctl.DocumentOptions.SecurityOptions.EnablePermission = "true"
    ctl.DocumentOptions.SecurityOptions.EnableLogicDelete = "true"
    ctl.DocumentOptions.SecurityOptions.ShowLogicDeletedContent = "true"
    ctl.DocumentOptions.SecurityOptions.ShowPermissionTip = "true"
    ctl.DocumentOptions.SecurityOptions.ShowPermissionMark = "true"
    ctl.ApplyDocumentOptions();
    ctl.LoadDocumentFromString(a, "xml");
});
```

结合使用：

无

接口类型：

后台接口，经过服务端，只有.net core 部署服务才能调用

有无限制：

编辑器初始化成功之后进行调用都行

兼容接口：

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化：

方法新增回调函数，在回调函数里面获取合并之后的 xml 文档

示例地址：

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=MergeDocumentByFileContent>

8. 1. 22. *获取文档设置的自定义属性

接口:

String GetCustomAttribute(string parameter)

描述:

获取文档中的指定的自定义属性的值。自定义属性在 XML 中存储在根节点 Attributes 下，以键值对的方式存储。如下图所示：

```
<XTextDocument xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" >
  <Attributes>
    <Attribute>
      <Name>姓名1</Name>
      <Value>值1</Value>
    </Attribute>
    <Attribute>
      <Name>姓名2</Name>
      <Value>值2</Value>
    </Attribute>
  </Attributes>
</XTextDocument>
```

参数:

参数名	是否必填	数据类型	说明
parameter	必填	字符串	指定的自定义属性的名称

返回值:

数据类型	说明	备注
String	指定的自定义属性的值。如果文档中没有自定义属性，或自定义属性没有对应的名称，则返回空字符串。	

代码示例:

```
let res=ctl.GetCustomAttribute(strXML);
```

相同效果:

与 GetDocumentCustomAttributes 方法（全部获取）类似，该方法是获取文档中指定的某条自定义属性。

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制（暂定）:

编辑器加载成功之后在任何情况下都能调用（预览模式除外）

兼容接口:

五代独有函数。

接口优化:

无

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=GetCustomAttribute>

8. 1. 23. *设置文档的自定义属性

接口:

Bool SetCustomAttribute(string name, string Value)

描述:

设置文档中的指定的自定义属性的值。当指定名称的自定义属性在文档中不存在时，则添加自定义属性；当指定名称的自定义属性在文档中存在时，则替换该自定义属性的值。

自定义属性在 XML 中存储在根节点 Attributes 下，以键值对的方式存储。如下图所示：

```
<XTextDocument xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema-instance">
  <Attributes>
    <Attribute>
      <Name>姓名1</Name>
      <Value>值1</Value>
    </Attribute>
    <Attribute>
      <Name>姓名2</Name>
      <Value>值2</Value>
    </Attribute>
  </Attributes>
</XTextDocument>
```

参数:

参数名	是否必填	数据类型	说明
Name	必填	字符串	指定的自定义属性的名称
Value	必填	字符串	指定的自定义属性的值

返回值:

数据类型	说明	备注
Bool	设置自定义属性是否为空	

代码示例:

```
let res=ctl.SetCustomAttribute('科室','精神科');
```

相同效果:

与 SetDocumentCustomAttributes 方法类似，该方法是一条一条自定义属性进行添加到文档下。

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制 (暂定):

编辑器加载成功之后在任何情况下都能调用 (预览模式除外)

兼容接口:

五代独有函数。

接口优化:

无

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=SetCustomAttribute>

8.7.24. *对文档产生批注式校验**接口:**

Json DocumentValueValidateWithCreateDocumentComments ()

描述:

对编辑器加载的文档进行数据校验, 当存在校验失败的元素时, 返回校验失败的元素

json 集合和在校验失败的元素上添加批注式校验提示

参数:

参数名	是否必填	数据类型	说明
无			

返回值:

数据类型	说明	备注
json	ElementID: 校验失败的元素 ID ExcludeKeywordText: 违禁关键字 Level: 校验等级。Error: 错误、Warning: 警告、Info: 提示 Title: Type: 类型 Message: 错误提示文本	

代码示例:

```
var vaulevalidate=ctl.DocumentValueValidateWithCreateDocumentComments ()
```

使用场景:

可以当医生保存书写好的病历时先调用下该方法进行全文校验,校验输入的内容是否满足设置的规则,然后提醒医生修改之后再进行保存

相同效果:

```
ctl.DCExecuteCommand('DocumentValueValidateWithCreateDocumentComments',
false, null);
ctl.DocumentValueValidate()
ctl.DCExecuteCommand('DocumentValueValidate', false, null)
```

前两个接口也是对文档进行校验,但是当校验失败的时候会在编辑器界面上进行批注式的提示,而本方法和最后一个方法只会返回校验失败的元素集(上面的两个方法也有返回值)

结合使用:

无

接口类型:

纯前端接口,不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用(预览模式除外)

兼容接口:

兼容 CS-CS、五代编辑器调用方式一致。

接口优化:

无

示例地址:

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=DocumentValueValidate>

8. 1. 25. *获取文档结构化数据

接口:

```
Json DCFormTransmission(json parameter)
```

描述:

获取文档中全部带有层次关系的结构化数据列表。

参数:

参数名	是否必填	数据类型	说明
parameter	必填	Json	以下参数是 json 的内容
+filecontentxml	必填	字符串	需要解析的文档内容

+isusebase64	非必填	Bool	文档内容是否是 base64 格式。默认 false。
+isbindingdata	非必填	Bool	是否绑定数据。默认 false。
+datas	非必填	字符串	数据源绑定所需要的数据集
+isreturnfilecontent	非必填	Bool	是否返回文档的 XML 内容
+isreturnstruct	非必填	Bool	是否返回结构化数据
+outputelementinnerxml	非必填	Bool	是否返回元素的 XML (对应 OuterXml 值)
+nestedmode	非必填	Bool	是否是嵌套模式，默认为 false 时，所有元素平铺输出前端，没有嵌套关系。设为 true 后，按文档排列顺序和嵌套顺序，父容器的内容会输出到容器元素 childnodes 内，开关设为 false 为和老版本兼容
+ReturnMergedContent	非必填	bool	是否返回带痕迹的绑定结果文档
userinfo	非必填	json	用户登录信息，参考接口 UserLoginByUserLoginInfo 参数说明

返回值：

数据类型	说明
Json	返回 json 格式的内容。属性较多，暂不列举。

代码示例：

```
var options = {
    FileContentXML: loadstring,    //1. 需要加载处理的原文档XML字符串
    IsUseBase64: false,           //2. 指定加载和保存的字符串是否是BASE64字符串
    IsBindingData: true,          //3. 指定加载文档后是否要做将数据绑定到文档的处理
    Datas: data,                  //4. 若要绑定数据在这里指定绑定文档所需要的数据集
    IsReturnFileContent: true,     //5. 加载并绑定处理完文档后是否将文档保存成字符串返回前端
    IsReturnStruct: true,          //6. 加载并绑定处理完文档后是否返回加载文档的前端JSON结构
    OutputElementInnerXML: false, //是否输出元素自身的XML 20220620新增
    NestedMode: true              //是否输出元素嵌套关系
};
```

```
var obj = ctl.DCFormTransmission(options);
```

兼容接口：

兼容 CS-CS、五代编辑器调用方式一致。

接口优化：

无

8. 1. 26. *获取文档页码数组

接口：

Array GetAllPageIndexs()

描述：

获得所有的页码数组。从 0 开始计算

参数：

无。

返回值：

数据类型	说明	备注
Array	页码数组	

代码示例：

```
let res=ctl.GetAllPageIndexs();
```

使用场景：

当客户需要实现奇偶打印，或者通过页码进行打印时，可以调用该方法来实现自定义打印

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用

兼容接口：

五代独有函数。

接口优化：

无

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=GetAllPageIndexs>

8. 1. 27. *获取编辑器支持的字体列表

接口:

Array getSupportFontFamillys()

描述:

获得当前编辑器支持的所有字体名称。由于浏览器的权限,无法获取操作系统的所有字体,这里是通过先设置已知的各种常用字体名称,通过 canvas 绘图的方式来判断该字体是否支持。

该接口使用中需要注意:

1. 这里预设常用字体可能不全,导致未判断该字体是否支持;
2. 在使用中,需要对预设的所有字体循环判断,效率和性能较差。谨慎使用。

参数:

无。

返回值:

数据类型	说明	备注
Array	字体名称数组	

代码示例:

```
let res=ctl.getSupportFontFamillys();
```

结合使用:

无

接口类型:

纯前端接口,不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用

兼容接口:

五代独有函数。

接口优化:

无

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=getSupportFontFamiliys>

8. 1. 28. *对文档不产生批注式的文档校验

接口:

Json DocumentValueValidate()

描述:

返回当前文档校验失败的元素集合。

参数:

参数名	备注
无	

返回值:

数据类型	说明	备注
数组	返回文档中设置了校验格式但校验失败的数组	

代码示例:

```
let res=ctl.DocumentValueValidate();
```

使用场景:

可以当医生保存书写好的病历时先调用下该方法进行全文校验,校验输入的内容是否满足设置的规则,然后提醒医生修改之后再进行保存

相同效果:

```
ctl.DCExecuteCommand('DocumentValueValidateWithCreateDocumentComments',  
false, null);
```

```
ctl.DocumentValueValidateWithCreateDocumentComments()
```

```
ctl.DCExecuteCommand('DocumentValueValidate', false, null)
```

前两个接口也是对文档进行校验,但是当校验失败的时候会在编辑器界面上进行批注式的提示,而本方法和最后一个方法只会返回校验失败的元素集(上面的两个方法也有返回值)

结合使用:

无

接口类型:

纯前端接口,不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用（预览模式除外）

兼容接口：

兼容 CS-CS、五代编辑器调用方式一致。

接口优化：

无

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=DocumentValueValidate>

8.1.29. *全选正文内容

接口：

Void SelectAll()

描述：

选择全文内容。只会选择正文内容，不会选择页眉页脚内容。快捷键：ctrl+a

参数：

无

返回值：

无。

代码示例：

```
ctl.SelectAll();
```

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用（预览模式除外）

兼容接口：

兼容 CS-CS、五代编辑器调用方式一致。

接口优化：

无

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=SelectAll>

8. 1. 30. *设置编辑器违禁关键字

接口:

Void SetExcludeKeywords(string keys)

描述:

设置全文的违禁关键词。各个违禁关键词之间以英文逗号分隔。该接口的功能应该是和 BehaviorOptions.ExcludeKeywords 设置相同。

设置违禁关键词后,调用 DocumentValueValidate 命令校验文档内容时会进行文档正文的违禁关键字检查,若包含了违禁关键字则认为文档内容出现校验错误提示。

参数:

参数名	是否必填	数据类型	说明
Keys	必填	String	全文的违禁关键词。各个违禁关键词之间以英文逗号分隔。

返回值:

无。

代码示例:

```
ctl.SetExcludeKeywords('子宫,卵巢');
```

相同效果:

```
ctl.DocumentOptions.BehaviorOptions.ExcludeKeywords = "子宫,卵巢";
ctl.ApplyDocumentOptions()
跟上面的设置效果一致
```

结合使用:

跟 [8. 1. 28 接口结合使用](#)

接口类型:

纯前端接口,不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用

兼容接口:

五代独有函数。

接口优化:

无

示例地址:

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=SetExcludeKeywords>

8. 1. 31. *获取编辑器违禁关键字

接口:

String GetExcludeKeywords()

描述:

获取文本设置的设置的违禁关键词。该接口返回的就是 SetExcludeKeywords 接口设置的内容。

参数:

无

返回值:

数据类型	说明	备注
String	全文的违禁关键词。各个违禁关键词之间以英文逗号分隔。	

代码示例:

```
Let keys=ctl.GetExcludeKeywords();
```

相同效果:

var xx=ctl.DocumentOptions.BehaviorOptions.ExcludeKeywords;
跟上面的获取的值一样

结合使用:

跟 [8. 1. 31 接口结合使用](#)

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用（预览模式除外）

兼容接口:

五代独有函数。

接口优化:

无

示例地址:

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=GetExcludeKeywords>

8. 1. 32. *选中内容生成 html**接口:**

String SelectionHtml (bool isContentHeaderFooter)

描述:

获取选择内容的 HTML 字符串。

参数:

参数名	是否必填	数据类型	说明
isContentHeaderFooter	必填	Bool	生成的内容中是否包含页眉页脚。默认值是 false。

返回值:

数据类型	说明
String	返回内容字符串。

代码示例:

```
let res=ctl.SelectionHtml(true);
```

参数详细说明:

当没有选择内容时，调用该方法传不同参数分别含义：

1. 传 true 值时：返回一个只带页眉页脚的 html 内容
2. 传 false 值时：返回一个空的 html 内容（相当于一个空白文档返回的 html 效果）
3. 不传参数时：效果跟 false 一致

当选择内容时，调用该方法传不同参数的分别含义：

1. 传 true 值时：返回一个带有页眉页脚且选中内容的 html 数据
2. 传 false 值时：返回一个只带选中内容的 html 数据
3. 不传参数时：效果跟 false 一致

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用，（预览模式、表单模式除外，没有选中状态调用该方法无效）

兼容接口:

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=SelectionHtml>

8.1.33. *选中内容生成 Text

接口:

String SelectionText(bool isContentHeaderFooter)

描述:

获取选择内容的文本内容字符串。

参数:

参数名	是否必填	数据类型	说明
isContentHeaderFooter	必填	Bool	生成的内容中是否包含页眉页脚。默认值是false。

返回值:

数据类型	说明
String	返回内容字符串。

代码示例:

```
let res=ctl.SelectionText(true);
```

参数详细说明:

当没有选择内容时，调用该方法传不同参数分别含义：

- 1. 传 true 值时：返回一个只带页眉页脚的 text 内容
- 2. 传 false 值时：返回一个空的 text 内容（相当于一个空白文档返回的 text 效果）
- 3. 不传参数时：效果跟 false 一致

当选择内容时，调用该方法传不同参数的分别含义：

- 1. 传 true 值时：返回一个带有页眉页脚且选中内容的 text 数据
- 2. 传 false 值时：返回一个只带选中内容的 text 数据
- 3. 不传参数时：效果跟 false 一致

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用，（预览模式、表单模式除外，没有选中状态调用该方法无效）

兼容接口：

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化：

无任何区别

示例地址：

<http://www.dewriter.cn:8022/apiDocument/edit/index.html?apiname=SelectionText>

8. 1. 34. *选中内容设置 xml

接口：

String SelectionXml (bool isContentHeaderFooter)

描述：

获取选择内容的 XML 内容字符串。

参数：

参数名	是否必填	数据类型	说明

isContentHeaderFooter	必填	Bool	生成的内容中是否包含页眉页脚。默认值是 false。

返回值:

数据类型	说明
String	返回内容字符串。

代码示例:

```
let res=ctl.SelectionXml(true);
```

参数详细说明:

当没有选择内容时, 调用该方法传不同参数分别含义:

1. 传 true 值时: 返回一个只带页眉页脚的 xml 内容
2. 传 false 值时: 返回一个空的 xml 内容 (相当于一个空白文档返回的 xml 效果)
3. 不传参数时: 效果跟 false 一致

当选择内容时, 调用该方法传不同参数的分别含义:

1. 传 true 值时: 返回一个带有页眉页脚且选中内容的 xml 数据
2. 传 false 值时: 返回一个只带选中内容的 xml 数据
3. 不传参数时: 效果跟 false 一致

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用, (预览模式、表单模式除外, 没有选中状态调用该方法无效)

兼容接口:

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=SelectionXml>

8.1.35. *选中内容生成 Json

接口:

```
String SelectionJson(bool isContentHeaderFooter)
```

描述:

获取选择内容的 Json 内容字符串。

参数:

参数名	是否必填	数据类型	说明
isContentHeaderFooter	必填	Bool	生成的内容中是否包含页眉页脚。默认值是 false。

返回值:

数据类型	说明
String	返回内容字符串。

代码示例:

```
let res=ctl.SelectionJson(true);
```

参数详细说明:

当没有选择内容时，调用该方法传不同参数分别含义:

1. 传 true 值时: 返回一个只带页眉页脚的 json 内容
2. 传 false 值时: 返回一个空的 json 内容 (相当于一个空白文档返回的 json 效果)
3. 不传参数时: 效果跟 false 一致

当选择内容时，调用该方法传不同参数的分别含义:

1. 传 true 值时: 返回一个带有页眉页脚且选中内容的 json 数据
2. 传 false 值时: 返回一个只带选中内容的 json 数据
3. 不传参数时: 效果跟 false 一致

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用，（预览模式、表单模式除外，没有选中状态调用该方法无效）

兼容接口：

五代独有函数。

接口优化：

无

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=SelectionJson>

8. 1. 36. *获取当前病历 xml 数据--不推荐使用

接口：

String GetXmlContent()

描述：

获取当前文档 xml 内容。

参数：

无参数。

返回值：

数据类型	说明	备注
String	当前文档的 xml 内容	

代码示例：

```
let res=ctl.GetXmlContent();
```

相同效果：

与 `ctl.SaveDocumentToString('xml')` 返回值效果一致

效果影响：

与 `ctl.DocumentOptions.BehaviorOptions.OutputFormattedXMLSource=false` 设置的值不同返回的效果不同

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制（暂定）:

编辑器加载成功之后在任何情况下都能调用

兼容接口:

兼容四代-四代、五代编辑器调用方式一致。

接口优化:

五代编辑器把该函数的参数给去掉了

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=GetXmlContent>

8. 1. 37. *获取编辑器处于哪种类型的表单模式

接口:

String FormView()

描述:

获取当前的表单视图模式类型。表单模式包含 Disable、Normal、Strict。表单视图模式为 Disable 时，也就是禁用了表单模式，为普通编辑模式。对于“Strict”表示设置为严格表单模式，对于“Normal”表示设置为普通表单模式。

在普通表单模式下，插入点不受限制，但是用户只能在文本输入域中编辑内容，输入域之外的内容是只读的，可以鼠标选择输入域外面的内容。

在严格表单模式下，插入点受限制，光标只能定位在输入域内，输入域外不能定位光标，也不能在输入域外选择内容。

参数:

无参数。

返回值:

数据类型	说明	备注
String	表单模式名称	Disable: 不处于表单模式，Normal: 处于普通表单模式（光标能在输入域元素外，但是不能操作），Strict: 严格表单模式（光标只能在输入域里面操作）

代码示例:

```
let res=ctl.FormView();
```

相同效果:

与 `ctl.DocumentOptions.BehaviorOptions.FormView` 返回值一致

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制（暂定）:

编辑器加载成功之后在任何情况下都能调用（预览模式除外）

兼容接口:

五代独有函数。

接口优化:

无

示例地址:

<http://www.dcwriter.cn:8022/apiDocument/edit/index.html?apiname=FormView>

8. 1. 38. *获取选中内容处于哪种保护模式下

接口:

`String ProtectType()`

描述:

获取当前的选择内容的内容保护类型名称。内容保护类型包含 `None`、`Content`、`Range`。内容保护类型为 `None` 时，也就是没有设置内容保护，为普通编辑模式。对于“`Content`”表示对于当前选择内容可以在中间输入内容，但是不能删除内容。对于“`Range`”表示对于当前选择内容不允许删除内容，也不允许在中间输入内容。

参数:

无参数。

返回值:

数据类型	说明	备注
String	内容保护状态名称	None: 不保护内容, Content: 保护内容, 但可以在保护内容的中间插入内容, Range: 保护区域, 中间不能插入新的内容

代码示例:

```
let res=ctl.ProtectType();
```

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制 (暂定):

编辑器加载成功之后在任何情况下都能调用 (预览模式除外)

兼容接口:

五代独有函数。

接口优化:

无

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=ProtectType>

8.1.39. *清空全文内容

接口:

Void ClearContent()

描述:

清空全文内容, 相当于新建一个空白的文档。

参数:

无参数。

返回值:

无

代码示例:

```
ctl.ClearContent();
```

相同效果:

ctl.DCExecuteCommand('FileNew',false,true)

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制（暂定）:

编辑器加载成功之后在任何情况下都能调用。阅读、预览、续打、区域选择时接口无效。

兼容接口:

兼容 CS-CS、五代编辑器调用方式一致。

接口优化:

无

示例地址:

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=ClearContent>

8.1.40. *动态设置页脚水平线

接口:

Bool ShowFooterLine()

描述:

是否在页脚的开头显示水平线的功能。如果页脚的开头没有水平线，该接口将添加一个水平线；如果页脚的开头包含水平线，该接口将会删除该水平线。

参数:

无

返回值:

数据类型	说明
Bool	设置是否成功

代码示例:

```
let res=ctl.ShowFooterLine();
```

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制（暂定）:

编辑器加载成功之后在任何情况下都能调用

兼容接口：

五代独有函数。

接口优化：

无

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=ShowFooterLine>

8. 1. 41. *获取文档的长图片 base64 格式数据

接口：

Bool GetLongImageData(function callBack)

描述：

获取文档的长图片格式的 base64 数据。该接口通过回调的方式获取数据。

参数：

参数名	是否必填	数据类型	说明
callBack	必填	function	回调函数

返回值：

数据类型	说明	备注
操作是否成功	内容保护状态名称	

代码示例：

```
var result = ctl.GetLongImageData(function (params) {
    console.log(params, '=====params')
    var blob = params;
    let downloadElement = document.createElement("a");
    let href = window.URL.createObjectURL(blob); //创建下载的连接
    downloadElement.href = href;
    downloadElement.download = 'asd'; // file.name; //下载后文件名
    document.body.appendChild(downloadElement);
    downloadElement.click(); //点击下载
    document.body.removeChild(downloadElement); //下载完成移除元素
    window.URL.revokeObjectURL(href); //释放掉blob对象
})
```

相同效果：

ctl.DCExecuteCommand('longimage')效果一致,生成的图片没有分页效果

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制 (暂定):

编辑器加载成功之后在任何情况下都能调用

兼容接口:

五代独有函数。

接口优化:

无

示例地址:

<http://www.dcwriter.cn:8022/apiDocument/edit/index.html?apiname=GetLongImageData>

8.1.42. *当前位置插入 json 内容片段

接口:

Bool InsertJson(string content)

描述:

在当前位置插入 json 内容。

参数:

参数名	是否必填	数据类型	说明
content	必填	字符串	json 格式的文档内容

返回值:

数据类型	说明	备注
Bool	插入是否成功	

代码示例:

```
let res=ctl.InsertJson(strJson);
```

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制（暂定）：

编辑器加载成功之后在任何情况下都能调用。在阅读、预览、续打、区域选择时接口无效。

兼容接口：

五代独有函数。

接口优化：

无

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=InsertJson>

8.1.43. *在文档末尾处追加 xml 内容片段

接口：

Bool AppendBody(string content)

描述：

在文档的正文内容追加 XML 内容。

参数：

参数名	是否必填	数据类型	说明
content	必填	字符串	XML 格式的文档内容

返回值：

数据类型	说明	备注
Bool	操作是否成功	

代码示例：

```
ctl.AppendBody(strXML);
```

相同效果：

InsertXmlString、InsertXmlBase64String、InsertXmlById（在不处于限制光标的视图模式下先进行光标定位到文档最后面）具有相同效果

结合使用：

ctl.RefreshDocument();需要执行完 AppendBody 方法之后执行文档刷新方法，否则接口无效

接口类型:

纯前端接口，不经过服务端

有无限制（暂定）:

编辑器加载成功之后在任何情况下都能调用

兼容接口:

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=AppendBody>

8.1.44. *选中内容生成指定格式数据

接口:

String DocumentSelection(string format)

描述:

获取选择内容的指定格式内容字符串。如果传入参数为 html，和 SelectionHtml 类似；如果是 XML，和 SelectionXml 类似；如果是 json，和 SelectionJson 类似；如果是 text，和 SelectionText 类似

参数:

参数名	是否必填	数据类型	说明
format	必填	String	文档格式。

返回值:

数据类型	说明
String	返回内容字符串。

代码示例:

```
let res=ctl.DocumentSelection("XML");
```

相同效果:

SelectionXml、SelectionJson、SelectionHtml、SelectionTex 方法返回值效果一致

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制（暂定）:

编辑器加载成功之后在任何情况下都能调用

兼容接口:

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化:

新增 json 格式数据

示例地址:

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=DocumentSelection>

8. 1. 45. *LoadDocumentFromString2---不推荐使用

接口:

Bool LoadDocumentFromString2(json parameter)

描述:

特殊的加载文档的方式。

参数:

参数名	是否必填	数据类型	说明
parameter	必填	Json	以下参数是 json 的内容
+ fileformat	必填	字符串	需要加载的文档内容字符串的文档格式
+ usebase64string	必填	字符串	当前加载的文档内容是否是 base64 格式。
+FileContent	必填	字符串	需要加载的文档内容
+LoadPrintPreivew	必填	Bool	是否以预览的方式加载，默认是 false
+DataBindingXML	必填	字符串	需要绑定数据源的内容
+CommitUserTrace	必填	Bool	是否提交用户痕迹，默认是 false
+AppendSubDocuments	必填	Bool	在加载的主文档正文后面通过病程的方式追加，默认是 false

+loadfromurl	必填	字符串	以指定的 url 来获取文档内容再加载

返回值:

数据类型	说明	备注
Bool	返回是否加载完成	

代码示例:

```
var obj = {
    FileContent: data,           //加载的文档XML字符串
    FileFormat: "xml",          //加载的文档格式
    UseBase64String: "false",    //是否加载的是BASE64字符串
    LoadPrintPreivew: "true",   //是否直接进入打印预览的界面而不是编辑界面
    DataBindingXML: null,        //加载时附带的数据源XML, 后台会绑定好再展示到前端
    CommitUserTrace: "true",     //是否将文档在后台提交完痕迹后再展示到前端
    AppendSubDocuments:options  //在主文档正文后面追加病程文档
};
ctl.LoadDocumentFromString2(obj);
```

结合使用:

DocumentLoad 事件（文档加载完成事件）-当编辑器把文档渲染成功之后触发的事件，当编辑器初始化的时候需要加载文档并同步进行数据源绑定，得在 WriterControl_OnLoad 事件中先定义加载完成事件（DocumentLoad），然后再调用加载文档的接口，否则无效。

接口类型:

服务端接口

有无限制:

编辑器初始化成功之后进行调用

兼容接口:

兼容四代接口-五代、四代编辑器该接口调用方式一致。不推荐使用，该接口是在五代编辑器下对病程留痕问题无法彻底解决新增的接口，五代没有该问题，不推荐在使用该接口

接口优化:

无任何区别

用例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=LoadDocumentFromString2>

8.1.46. *查找

接口:

Int Search(json parameter)

描述:

查找内容。

参数:

参数名	是否必填	数据类型	说明
parameter	必填	Json	以下参数是 json 的内容
+ searchstring	必填	字符串	要查找的字符串
+ enablereplacestring	必填	Bool	是否启用替换。默认是 false
+ replacestring	必填	字符串	要 替 换 的 字 符 串 。 在 enablereplacestring为true时才有效。
+ backward	必填	Bool	是否向后替换，默认是 false
+ ignorecase	必填	Bool	是否区分大小写，默认是 true
+ logundo	必填	Bool	记录撤销操作信息，默认是 true
+ excludebackgroundtext	必填	Bool	是否忽略掉背景文字，默认是 true
+ SearchID	必填	Bool	是否限制为查询元素编号，默认是 true

返回值:

数据类型	说明	备注
Int	返回查找的字符在文档中内容索引值	

代码示例:

```
var options = {
    searchstring,
    enablereplacestring: "false",
    replacestring,
    backward,
    ignorecase,
    logundo,
    excludebackgroundtext,
    SearchID
}
ctl.Search(options)
```

相同效果:

ctrl+f

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制（暂定）:

编辑器加载成功之后在任何情况下都能调用。打印预览模式下无效，查找时，（只读、阅读模式）下编辑器视图不会进行跳转。

兼容接口:

五代编辑器独有接口。

接口优化:

新增阅读模式下也能查找

示例地址:

无

8.1.47. *Reaplace

接口:

Int Reaplace (json parameter)

描述:

替换内容。该接口的参数是 Search 查找接口的重新设置，默认设置了 enablereplacestring 为 true。

参数:

参数名	是否必填	数据类型	说明
parameter	必填	Json	以下参数是 json 的内容
+ searchstring	必填	字符串	要查找的字符串
+replacestring	必填	字符串	要替换的字符串。
+backward	必填	Bool	是否是否向后替换，默认是 false
+ignorecase	必填	Bool	是否区分大小写，默认是 true

+logundo	必填	Bool	记录撤销操作信息，默认是 true
+excludebackgroundtext	必填	Bool	是否忽略掉背景文字，默认是 true
+SearchID	必填	Bool	是否限制为查询元素编号，默认是 true

返回值：

数据类型	说明	备注
Int	返回查找到的次数	

代码示例：

```
var options = {
    searchstring,
    replacestring,
    backward,
    ignorecase,
    logundo,
    excludebackgroundtext,
    SearchID
}
ctl.Replace(options)
```

8.1.48. *ReaplaceAll

接口：

Int ReaplaceAll(json parameter)

描述：

替换内容。该接口的参数是 Search 查找接口的重新设置，默认设置了 enablereplacestring 为 true。暂未测试 Reaplace 和 ReplaceAll 的区别。

参数：

参数名	是否必填	数据类型	说明
parameter	必填	Json	以下参数是 json 的内容
+ searchstring	必填	字符串	要查找的字符串
+replacestring	必填	字符串	要替换的字符串。
+backward	必填	Bool	是否是否向后替换，默认是 false

+ignorecase	必填	Bool	是否区分大小写，默认是 true
+logundo	必填	Bool	记录撤销操作信息，默认是 true
+excludebackgroundtext	必填	Bool	是否忽略掉背景文字，默选中认是 true
+SearchID	必填	Bool	是否限制为查询元素编号，默认是 true

返回值：

数据类型	说明	备注
Int	返回查找到的次数	

代码示例：

```
var options = {
    searchstring,
    replacestring,
    backward,
    ignorecase,
    logundo,
    excludebackgroundtext,
    SearchID
}
ctl.Replace(options)
```

8. 1. 49. *查找、替换对话框

接口：

Int SearchAndReplaceDialog()

描述：

根据查找替换的参数，弹出内置的查找替换对话框。快捷键是 ctrl+f，弹出对话框如下：



参数：

无

返回值：

数据类型	说明	备注
Int	返回查找到的次数	

代码示例：

```
var ctl=document.getElementById("myWriterControl")
ctl.SearchAndReplaceDialog()
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。阅读模式、打印预览模式下无效。严格表单模式下，只有当查找的内容在输入域内才有效果。替换功能只有编辑器存在输入焦点，能编辑的情况下才有效果

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://www.dcwritr.cn:8765/apiDocument/edit/index.html?apiname=SearchAndReplaceDialog>

8.1.50. *页面刷新文档

接口：

Void RefreshDocument()

描述：

对全文进行刷新操作。在一些功能会对全文进行操作，更新了文档 DOM 结构，需要调用

编辑器控件的 RefreshDocument 方法更新文档视图。

参数:

无

返回值:

无

代码示例:

```
ctl.RefreshDocument();
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用

兼容接口:

兼容 CS-CS、五代编辑器调用方式一致。

接口优化:

无

示例地址:

<http://www.dcwriter.cn:8022/apiDocument/edit/index.html?apiname=RefreshDocument>

8.1.51. *内部刷新文档

接口:

```
Void RefreshInnerView()
```

描述:

刷新文档内部排版和分页。不更新用户界面。该接口一般是内部使用。当调用某个接口没有任何效果时，执行了 RefreshDocument 方法也是一样，就可以使用该方法刷新下

参数:

无

返回值:

无

代码示例:

```
ctl.RefreshInnerView();
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用

兼容接口:

兼容 CS-CS、五代编辑器调用方式一致。

接口优化:

无

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=RefreshInnerView>

8. 1. 52. *获得表单数据

接口:

string GeFormValue(string name)

描述:

获得文档中单复选框相同名称已经勾选的数值 (Value)，勾选多个用英文逗号分隔。

参数:

参数名	是否必填	数据类型	说明
Name	必填	String	表单元素的名称

返回值:

数据类型	说明

String	表单元素对应的值。

代码示例:

<pre>let res=ctl.GetFormValue('name');//返回单复选框中为name名称已经勾选的数值</pre>

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制（暂定）:

编辑器加载成功之后在任何情况下都能调用

兼容接口:

兼容 CS-CS、五代编辑器调用方式一致。

接口优化:

无

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=GetFormValue>

注意:

使用该接口，输入域的 name 值和单选框组的 name、复选框组的 name 值不能相同。

8. 1. 53. *动态读写违禁关键字

接口:

ctl.ExcludeKeywords

描述:

获取或设置文档的违禁关键词。该接口和 SetExcludeKeywords、GetExcludeKeywords 的功能关联。

参数:

字符串，违禁关键词之间以英文逗号分隔。

返回值:

返回已经设置的违禁关键词，违禁关键词之间以英文逗号分隔。

代码示例:

```
let res=ctl.ExcludeKeywords;  
ctl.ExcludeKeywords= “月经,子宫”
```

相同效果:

ExcludeKeywords()、BehaviorOptions.ExcludeKeywords 文档选项、编辑器 div 上设置属性一样

结合使用:

编辑器校验方法

接口类型:

纯前端接口，不经过服务端

有无限制（暂定）:

编辑器加载成功之后在任何情况下都能调用

兼容接口:

兼容 CS-CS、五代编辑器调用方式一致。

接口优化:

无

示例地址:

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=ExcludeKeywords>

8. 1. 54. *设置文档的自定义属性

接口:

Bool SetDocumentCustomAttributes(json options)

描述:

设置全文的自定义属性。如果自定义属性名称相同，会覆盖原因的属性。

参数:

参数名	是否必填	数据类型	说明
options	必填	Json	自定义属性 json 对象

返回值:

数据类型	说明	备注
Bool	操作是否成功	

代码示例:

```
var opts = {  
    d1: "DDDDDD",  
    d2: "valuevalue",  
    AA: "myAA"  
};  
var result = myWriterControl.SetDocumentCustomAttributes(opts);
```

相同效果:

SetCustomAttribute 方法效果类似

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用（预览模式下无效）

兼容接口:

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=SetDocumentCustomAttributes>

8.1.55. *获取文档设置的自定义属性

接口:

Json GetDocumentCustomAttributes()

描述:

获取文档的自定义属性。

参数:

无

返回值:

数据类型	说明	备注
Json	自定义属性 json 对象	

代码示例：

```
var result = myWriterControl.GetDocumentCustomAttributes();
```

相同效果：

GetCustomAttribute 方法效果类似

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用（预览模式下无效）

兼容接口：

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化：

无任何区别

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=GetDocumentCustomAttributes>

8.2. 页面设置

8.2.1. *获取文档页面设置

接口：

Json GetDocumentPageSettings()

描述：

获取当前文档的页面设置信息。

参数:

无。

返回值:

返回值可参考本文的附录: [页面设置](#)。

代码示例:

```
let result = ctl.GetDocumentPageSettings();
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用

兼容接口:

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=GetDocumentPageSettings>

8.2.2. *修改文档页面设置

接口:

```
Bool ChangeDocumentSettings(json parameter)
```

描述:

设置当前页面设置信息。

参数:

该参数可参考本文的附录: [页面设置](#)

返回值:

返回是否设置成功

代码示例:

```
ctl.ChangeDocumentSettings({
  "PaperKind": "A4",
  "Landscape": true,
  "LeftMargin": 77,
  "RightMargin": 77,
  "TopMargin": 100,
  "BottomMargin": 100,
  "HeaderDistance": 50,
  "FooterDistance": 50,
  "ShowGutterLine": false,
  "GutterPosition": 0,
  "SwapGutter": true,
  "GutterStyle": "Left",
  "LeftMarginInCM": 1.9557998,
  "RightMarginInCM": 1.9557998,
  "TopMarginInCM": 2.54,
  "BottomMarginInCM": 2.54,
  "HeaderFooterDifferentFirstPage": false
});
```

相同效果:

无

结合使用:

RefreshDocument(), 设置完页面设置需要执行刷新方法才有效果

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用

兼容接口:

兼容四代接口-五代、四代编辑器该接口调用方式一致。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=ChangeDocumentSettings>

8.2.3. *获取文档水印

接口:

Json GetDocumentWatermark()

描述:

获取当前文档的水印设置信息。

参数:

无。

返回值:

返回值可参考本文的附录: [水印](#)。

代码示例:

```
let result = ctl.GetDocumentWatermark();
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用

兼容接口:

五代编辑器独有接口。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=GetDocumentWatermark>

8.2.4. *设置文档水印

接口:

Bool SetDocumentWatermark(json parameter)

描述:

设置当前水印设置信息。

参数:

该参数可参考本文的附录：[水印](#)。

返回值:

返回是否设置成功

代码示例:

```
var options = {  
    "type": "Text",  
    "densityforrepeat": 0.5,  
    "repeat": true,  
    "alpha": 80,  
    "colorvalue": "#000000",  
    "angle": 45,  
    "imagedatabase64string": "",  
    "text": "asfasfaffaf",  
    "font": "宋体, 24"  
}  
ctl.SetDocumentWatermark(options)
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用(打印模式预览下设置水印，预览下没有效果，但是退出预览，编辑下存在水印)

兼容接口:

五代编辑器独有接口。

接口优化:

无任何区别

示例地址:

<http://www.dewriter.cn:8022/apiDocument/edit/index.html?apiname=SetDocumentWatermark>

[rmark](#)

注意:

图片水印不要设置重复和倾斜属性。

8.2.5. *获取文档装订线

接口:

Json GetDocumentGutter()

描述:

获取当前文档的装订线信息。

参数:

无。

返回值:

返回值可参考本文的附录: [装订线](#)。

代码示例:

```
let result = ctl.GetDocumentGutter();
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用

兼容接口:

五代编辑器独有接口。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=GetDocumentGutter>

8.2.6. *设置文档装订线

接口:

```
Bool SetDocumentGutter(json parameter)
```

描述:

设置当前文档的装订线信息。

参数:

该参数可参考本文的附录: [装订线](#)。

返回值:

返回是否设置成功

代码示例:

```
let gridLineInfo = {  
    "ShowGutterLine": true,  
    "GutterPosition": 100,  
    "SwapGutter": true,  
    "GutterStyle": "0"//0表示Left, 1表示Top, 2表示Right  
};  
ctl.SetDocumentGutter(gridLineInfo);
```

相同效果:

通过 `ChangeDocumentSettings()` 方法里面设置装订线效果一致

结合使用:

以下两个刷新方法都可以使用:

`RefreshDocument()`, 设置完装订线需要执行刷新方法才有效果, 会存在一个刷新页面的效果

`RefreshInnerView(true)`, 调用该方法也是刷新, 但是是编辑器内部刷新, 界面没有影响

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用(打印模式预览下设置装订线, 预览下没有效果, 但是退出预览, 编辑下存在装订线)

兼容接口:

五代编辑器独有接口。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=SetDocumentGutter>

8.2.7. *获取文档网格线

接口:

Json GetDocumentGridLine()

描述:

获取当前文档的网格线信息。

参数:

无。

返回值:

返回值可参考本文的附录: [文档网格线](#)。

代码示例:

```
let result = ctl.GetDocumentGridLine();
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用

兼容接口:

五代编辑器独有接口。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=GetDocumentGridLine>

8.2.8. *设置文档网格线

接口:

Bool SetDocumentGridLine(json parameter)

描述:

设置当前文档的网格线信息。

参数:

该参数可参考本文的附录: [文档网格线](#)。

返回值:

返回是否设置成功

代码示例:

```
let gridLineInfo = {
  Visible: true,
  LineStyle: "Dash",
  ColorValue: "red",
  GridNumInOnePage: 30,
  AlignToGridLine: true,
  Printable: true
};
ctl.SetDocumentGridLine(gridLineInfo);
ctl.RefreshDocument();
```

相同效果:

无

结合使用:

RefreshDocument(), 设置完文档网格线需要执行刷新方法才有效果

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用(打印模式预览下设置网格线, 预览下没有效果, 但是退出预览, 编辑下存在文档网格线)

兼容接口:

五代编辑器独有接口。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=SetDocumentGridLine>

8.3. 视图

8.3.1. *获取编辑器缩放比率

接口:

Float GetZoomRate()

描述:

获取当前视图的视图缩放比率

参数:

无。

返回值:

数据类型	说明	备注
Float	返回当前的缩放比率。返回的数值在 0.1 到 5 之间。	

代码示例:

```
let res=ctl.GetZoomRate();
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用

兼容接口:

五代编辑器独有接口。

接口优化：

无任何区别

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=GetZoomRate>

注意：

放大缩小比率在 0.1-5 倍之间才行

8.3.2. *设置编辑器缩放比率

接口：

Bool SetZoomRate(float Value)

描述：

设置缩放比率。如果调用这个接口，会把控件的 autozoom 属性设置为 false。

参数：

参数名	是否必填	数据类型	说明
Value	必填	Float	指定的自定义属性的值。参数的数值在 0.1 到 5 之间。不在这个区间设置的值无效

返回值：

数据类型	说明	备注
Bool	设置是否成功	当传的参数超过 5 或者小于 0.1 的时候返回值为 false

代码示例：

```
let res=ctl.SetZoomRate(2.5);
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用

兼容接口：

五代编辑器独有接口。

接口优化：

无任何区别

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=SetZoomRate>

8.3.3. *设置编辑器视图自动适应

接口：

Void SetAutoZoom()

描述：

默认控件是以页面设置的纸张尺寸显示。调用该接口后，设置页面为自动缩放视图模式，此时，此时页面尺寸会根据控件的大小自动缩放填满整个控件区域。该接口会影响控件的前端 AutoZoom 属性。

调用接口后，会重置控件。为避免在部分框架下重置无效，建议直接设置前端 AutoZoom 属性，或在加载文档之前调用 SetAutoZoom 接口。

参数：

无

返回值：

无

代码示例：

```
let res=ctl.SetAutoZoom();
```

相同效果：

编辑器 div 中可以设置 AutoZoom 属性为 true，跟该方法的效果一致

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用

兼容接口：

五代编辑器独有接口。

接口优化：

无任何区别

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=SetAutoZoom>

8.3.4. *移动输入焦点到指定位置

接口：

Void MoveTo(string Value)

描述：

移动光标到指定的位置。

参数：

参数名	是否必填	数据类型	说明
Value	必填	String	新的光标位置。可选择值包含： None 不进行任何操作、 DocumentHome 文档的开头、 CellHome 当前插入点所在表格单元格的开头、 ParagraphHome 当前插入点所在的段落的开头、 PreLine 上一行、 Home 行首、 End 行尾、 NextLine 下一行、 ParagraphEnd 当前段落的结尾、 CellEnd 当前单元格的结尾、 DocumentEnd 文档的结尾、

			BeforeTable 在表格前面、 AfterTable 在表格后面、 BeforeField 在文档域的前面、 AfterField 在文档域的后面、 PreCell 前一个表格单元格、 AfterCell 在后一个表格单元格。 默认是 None，不移动。注意参数的大小写，不允许写错。

返回值:

无

代码示例:

```
let res=ctl.MoveTo("End");
```

相同效果:

ctl.DCExecuteCommand("MoveTo",false,"End")效果一致。该命令的第三个参数跟上面方法的参数一致

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用(编辑器处于只读模式下无效、打印预览模式下无效)

兼容接口:

五代编辑器独有接口。

接口优化:

无任何区别

示例地址:

<http://www.dewriter.cn:8022/apiDocument/edit/index.html?apiname=MoveTo>

8.3.5. *编辑器页面排版模式

接口:

Void SetPageLayoutMode(string Value)

描述:

设置编辑器的页面布局展示方式。默认编辑器是按照垂直方向上按照页展示，在部分环境下屏幕较宽，可使用水平展示的方式（带鱼屏）。编辑器初始化默认排版模式为多栏展示

参数:

参数名	是否必填	数据类型	说明
Value	必填	String	新的光标位置。可选择的值包含： SingleColumn, MultiColumn, Horizontal。 SingleColumn 表示单栏展示，这是默认值； MultiColumn 表示根据当前控件容器的宽度，自动计算水平页面编辑器显示的数量。 Horizontal 表示横向展示，此时不展示纵向滚动条。如果容器的高度较小时会缩小页面展示。

返回值:

无

代码示例:

```
ctl.SetPageLayoutMode("MultiColumn");
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用

兼容接口:

五代编辑器独有接口。

接口优化:

无任何区别

示例地址:

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=SetPageLayoutMode>

8.3.6. *移动输入焦点到指定页

接口:

Void MoveToPage(string Value)

描述:

滚动视图到指定页。页数从 0 开始计算。

参数:

参数名	是否必填	数据类型	说明
Value	必填	String	页数，整数或整数字符串形式。

返回值:

当传负数页码时，返回值为 false，传正数返回值为 true，不管是否跳转成功

代码示例:

```
ctl.MoveToPage("4");
```

相同效果:

ctl.DCExecuteCommand('MoveToPage', false, 4)

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用（续打模式、区域选择模式、打印预览模式下该方法无效）

兼容接口:

五代编辑器独有接口。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=MoveToPage>

8.3.7. *移动输入焦点到指定容器元素内

接口:

Void ScrollIntoView(string id)

描述:

滚动视图到指定元素。

参数:

参数名	是否必填	数据类型	说明
Id	必填	String	结构化元素的编号。

返回值:

数据类型	说明	备注
Bool	设置是否成功	

代码示例:

```
Let res=ctl.ScrollIntoView ( “field4” );
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用（打印预览模式、阅读模式、只读模式下该方法无效）只有当编辑器存在焦点该方法才有效果

兼容接口:

五代编辑器独有接口。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=ScrollIntoView>

8.3.8. *把编辑器视图滚动到当前输入焦点处

接口:

Void ScrollToCaret()

描述:

滚动视图到光标处。在一些情况下,比如直接滚动滚动条,会出现光标和视图不在同一处的情况,调用该接口,可以把视图滚动到光标处。当前视图在输入焦点的上方调用该接口就会把编辑器视图定位到可视化界面的最下方。当前视图在输入焦点的下方调用该接口就会把编辑器视图定位到可视化界面的最上方。

参数:

无

返回值:

无

代码示例:

```
Let res=ctl.ScrollToCaret ();
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口,不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用(打印预览模式、阅读模式、只读模式下该方法无效)只有当编辑器存在焦点该方法才有效果

兼容接口:

五代编辑器独有接口。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=ScrollToCaret>

8.3.9. *编辑器所在的父容器大小进行缩放编辑器

接口:

```
void SetWriterAutoZoom()
```

描述:

根据当前编辑器所在的父容器大小来缩放编辑器。当容器较大时，会放大编辑器尺寸；当容器较小时会缩小编辑器尺寸。

参数:

无

返回值:

bool: true: 方法执行成功进行自适应、false: 方法执行失败

代码示例:

```
var result=ctl.SetWriterAutoZoom()
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用

兼容接口:

五代编辑器独有接口。

接口优化:

无任何区别

示例地址

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=SetWriterAutoZoom>

8.3.10. *设置编辑器为阅读模式

接口:

ctl.ReadViewMode

描述:

获取或设置文档是否是阅读视图模式。阅读模式下，其显示内容和打印结果高度一致，不显示段落符号，不显示虚拟的表格线，不显示输入域边界元素，没有各种高亮度的动态效果。文档内容只读，没有表单样式。用户可以选择和复制文档内容。

参数:

Bool 值

返回值:

Bool 值

代码示例:

```
ctl.ReadViewMode=true  
ctl.RefreshDocument()  
let res=ctl.ReadViewMode
```

相同效果:

```
DCExecuteCommand("ReadViewMode",false,true)
```

结合使用:

得结合文档刷新才有效果

```
ctl.RefreshDocument();//带界面刷新的效果
```

```
ctl.RefreshInnerView(true);//不带界面刷新的效果
```

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用(预览模式下能获取到编辑器处于阅读视图模式、预览模式下设置阅读，取消预览也是阅读模式)。

兼容接口:

五代编辑器独有接口。

接口优化:

无任何区别

示例地址:

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=ReadViewMode>

8.3.11. *按下鼠标左键上下移动编辑器视图位置

接口:

Void setMouseDragScrollMode(bool isMouseDrag)

描述:

是否开启鼠标拖拽选择。开启后，在文档中按住鼠标左键不松口，上下移动鼠标，可以移动文档中的滚动条进行上下查看，对编辑器设置平铺模式，不能通过鼠标滚轮上下查找内容，设置该方法能很好操作。

参数:

参数名	是否必填	数据类型	说明
isMouseDrag	必填	Bool	是否开启鼠标拖拽选择。

返回值:

无

代码示例:

```
ctl.setMouseDragScrollMode(true);
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始加载完成之后都能调用（编辑模式下调用该方法，用户在编辑器界面输入不了内容，需要输入内容时关闭该模式）

兼容接口:

兼容四代-该接口在四代编辑器中只有预览模式下才有效。

接口优化:

五代编辑器中该接口在编辑模式、打印预览模式下都有效果

示例地址:

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=setMouseDragScrollMode>

8.3.12. *获取输入焦点所在当前页的坐标位置

接口:

Json ReturnLastCaretPosition()

描述:

返回最近的一次光标展示位置。

参数:

无

返回值:

参数名	数据类型	说明
canvasElement	Object	光标所在的 canvas 元素。
intPageIndex	Int	光标所在的 canvas 在编辑器的页码，从 0 开始计算
intDX	Float	相对于 canvas 元素的 x 偏移量
intDY	Float	相对于 canvas 元素的 y 偏移量

代码示例:

```
Let res=ctl.ReturnLastCaretPosition();
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制（讨论是否有问题）:

编辑器初始化成功之后都能调用（阅读模式下有效果、预览模式下能获取到位置，但是点击不同地方获取的位置一致）

兼容接口:

五代编辑器独有接口。

接口优化:

无任何区别

示例地址:

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=ReturnLastCaretPosition>

8.4. 打印

8.4.1. *单文档生成打印所需 base64 的 pdf 数据

接口:

Bool PrintAsPDF(null, function errorCallback)

描述:

把文档打印为 pdf。该接口需要访问服务端，必须设置控件的 servicepageurl 属性。

参数:

errorCallback	非必填	回调函数	完成后的回调函数

返回值:

数据类型	说明	备注
Bool	返回是否执行完成。由于打印时无法调用打印机的返回数据，此时的打印完成，表示代码执行完成，不代表真正的打印机返回。	

代码示例:

```
ctl.PrintAsPDF(null, function (a) { console.log(a) });
```

注意:

调用该方法不传任何参数，会下载一个 pdf 到客户端

相同效果:

无

结合使用:

无

接口类型:

服务器接口，会跟服务器进行交互、需要部署.net 或者.net core 服务才能使用

有无限制（讨论是否有问题）：

编辑器初始化成功之后都能调用

兼容接口：

五代编辑器独有接口。

接口优化：

无任何区别

示例地址：

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=PrintAsPDF>

8.4.2. *单文档生成打印所需 html 数据

接口：

Bool PrintAsHtml(json parameter,function errorCallback)

描述：

把文档打印为 Html，.net 服务端和 java 服务端都能正常调用该方法（2025-01-04）测试验证通过。

参数：

参数名	是否必填	数据类型	说明
parameter	必填	Json	以下参数是 json 的内容。参数允许为 null。以下是参数的属性
+ pageindexfix	非必填	Int	页码修正值，默认是 0 不需要设置。
+ cleanmode	非必填	Bool	是否为整洁打印模式，默认是 false
+ printHeaderfooteratfirstpage	非必填	Bool	是否在首页打印页眉页脚，默认是 false
+printrange	非必填	字符串	打印范围，可选值包含：AllPages 全部页面，这是默认

			值、SomePages 部分页面、Selection 当前选择内容、CurrentPage 当前页面。
+printmode	非必填	字符串	打印模式，可选值包含：Normal、OddPage、EvenPage、ManualDuplex、CurrentPage。
+printtablecellborder	非必填	Bool	是的打印表格的单元格边框
+collate	非必填	Bool	是否逐份打印
+copies	非必填	Int	打印分数
+frompage	非必填	字符串	打印开始页码
+topage	非必填	字符串	打印结束页码
+specifypageindexs	非必填	字符串	指定要打印的页码
+bodylayoutoffset	非必填	Float	正文偏移打印的位置
+jumpprintstartelementid	非必填	字符串	续打开始元素编号，续打位置设置为该元素的上边缘
+jumpprintendelementid	非必填	字符串	续打结束元素编号，下端续打位置设置为该元素的下边缘
+jumpprint	非必填	Json	续打设置，是 json 的内容。参数允许为 null。以下是参数的属性
++pageindex	非必填	Int	续打发生的页码序号
++position	非必填	Float	续打的开始位置
++endpageindex	非必填	Int	续打结束的页码序号
++endposition	非必填	Float	续打的结束位置
++mode	非必填	字符串	续打模式，可选择值为Disable、Normal、Offset、Append。
+forceprintheadfooter	非必填	Bool	是否强制打印页眉页脚，即使处于打印被选中的内容模式。

errorCallback	非必填	回调函数	完成后的回调函数

返回值:

数据类型	说明	备注
Bool	该方法无返回值，通过回调函数获取到编辑器生成的html 数据来实现业务需求。	

代码示例:

```
ctl.PrintAsHtml(null, function (a) { console.log(a) });
```

注意:

通过html打印，建议先进入打印预览模式，然后再调用PrintAsHtm方法打印，因为编辑器下文档选项设置不同会影响编辑模式与预览模式下排版不同，会受输入域的左右边框、背景文字的影响，在预览模式下排版不一致问题。

当方法中第二个回调方法不设时，会在本地下载一个相同内容的html文件。

8.4.3. *单文档生成打印所需 html 数据

接口:

```
String GetPrintNowHTML(string parameter)
```

描述:

调整需要打印的 html，使其解决直接打印的结果。该接口一般不单独使用，生成的 html 会带上边框样式，如果不使用编辑器内部的方法打印这个 html，自己实现打印功能的话会出现边框。通过该方法进行 html 转换，就不会带上这个边框。



参数:

参数名	是否必填	数据类型	说明
parameter	必填	字符串	需要打印的 html 内容

返回值：

数据类型	说明	备注
Bool	表示设置行间距是否成功	

代码示例：

```
ctl.GetPrintNowHTML(content);
```

相同效果：

无

结合使用：

需要结合 getHtmlByFiles、PrintByHtml、
GetPrintPreviewHTMLByMixedFiles、getHtmlByFileWithSource、
GetPrintPreviewHTML2 使用

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器初始化成功之后都能调用

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=GetPrintNowHTML>

8.4.4. *浏览器打印使用编辑器生成的 html 数据

接口：

```
Void PrintByHtml(string parameter)
```

描述：

打印 html 内容。该接口一般不单独使用。也就是需要传一个打印的 html，然后通过该方法进行打印。

参数：

参数名	是否必填	数据类型	说明
parameter	必填	字符串	需要打印的 html 内容

返回值：

由于技术限制，无法获取打印的返回，该接口无返回值

代码示例：

```
ctl.PrintByHtml (content); //content--打印所需要的html
```

相同效果:

PrintDocument () 方法一致, 但是上面的方法需要格外注意是否所见即所得
结合使用:

需要结合 getHtmlByFiles、GetPrintNowHTML、GetPrintAsHtml 等所有能生成打印所需要的 html 接口结合使用

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器初始化成功之后都能调用

示例地址:

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=PrintByHtml>

8. 4. 5. *多文档合并生成 html 数据-服务端方法

接口:

String GetPrintPreviewHTML(json parameter, function)

描述:

获取文档的打印预览 HTML。第二个参数是回调函数。

参数:

参数名	是 否 必填	数 据 类型	说明
parameter	必填	Json	以下参数是 json 的内容
+files	必填	字 符 串	文档内容字符串数组, 如果为空 则使用当前文档
usebase	非 必 填	bool	true: 传的是 base64 病历数据 默认为 false, xml 病历数据
+InsertLastTableRowToPageBottom	非 必 填	Bool	预览时, 是否将表格插入到页底
+WaterMark	非 必	Json	附加的水印信息。参数可参考本

	填		文的水印接口设置。
+OutputMedicalUsingHtml	非 必 填	Bool	是否使用 HTML 标记输出医学表达式而不使用图片。默认 false，使用的绘图。
+ CommitUserTrace	非 必 填	Bool	是否获取提交痕迹后的 HTML
errorCallback	非 必 填	回 调 函数	完成后的回调函数

返回值:

数据类型	说明
String	返回 html 内容字符串。

代码示例:

```
let res=ctl.GetPrintPreviewHTML();
```

注:

接口效果跟 GetPrintPreviewHTML2 一致，只不过该接口是服务端接口, 必须得部署一个.net core 的环境才能使用，GetPrintPreviewHTML2 是纯前端接口，详细看 8.4.5.1

且部署的.net core 服务需要通过服务器机器码注册，预览 html 才不会出现未注册

注:
多文档合并 html 或者预览，当需要使用当前文档自带的页码时，需要设置页码元素的 ValueType 属性不为 PageIndex 即页码。设置该属性多文档合并时页码从 1 开始计算进行显示，所有跟多文档操作的功能该属性都生效。

8.4.5.1. *多文档合并生成 html 数据-纯前端方法

接口:

```
String GetPrintPreviewHTML2(json options, function)
```

描述:

获取文档的打印预览 HTML。第二个参数是回调函数，只能通过回调函数获取对应的 html 数据。

参数 1:

参数名	是否必填	数据类型	说明
options	必填	Json	参数信息
+UseBase64	非必填	Bool	是否使用 base64 字符串
+Format	非必填	String	加载文档的格式 xml、json
+watermark	非必填	Json	水印信息, 参考 SetDocumentWatermark 接口的参数
+BindingDatas	非必填	json	各个文档中对应的数据源绑定内容
+insertlasttable rowtopagebottom	非必填	Bool	在页面最底下, 是否以表格行的形式铺满整个页面
+commitusertrace	非必填	Bool	是否提交文档的痕迹信息。提交痕迹后, 将删除所有被逻辑删除的元素, 清空痕迹信息。
+files	必填	Array	需要转换的多个文档的数组

参数 2:

参数名	是否必填	数据类型	说明
回调函数	是	无	通过回调函数获取到合并之后的预览html 用来打印或者预览

返回值:

数据类型	说明	备注
Bool	操作是否成功	

代码示例:

```
var arr1 = new Array();
arr1.push(xml1);
arr1.push(xml2);
let obj = {
  files: arr1, //文档数组
  format: 'xml', //文档格式、支持json格式
  base64: 'false', //是否是base64
  BindingDatas: [
```

```
        {
            EMR_Patients: {
                DropDownValue: "V5,V6",
                NAME: "王五",
                sex: "未知"
            }
        },
        {
            EMR_Patients: {
                DropDownValue: "V3,V4",
                NAME: "李四",
                sex: "女"
            }
        }
    ]//非必填
};

let printHtml = ctl.GetPrintPreviewHTML2(obj, function (file) {
//预览 file:html
var oPrntWin =
window.open("", "",
"width=450,height=470,left=400,top=100,menubar=yes,toolbar=no,location=no,scrollbars=yes,resizable=yes");
oPrntWin.document.write(file);
});
```

相同效果:

跟 GetPrintPreviewHTMLByMixedFiles、GetPrintPreviewHTML 效果一样, 这个不需要经过服务端

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=GetPrintPreviewHTML2>

8.4.6. *把编辑器生成的 html 数据通过预览模式显示

接口:

```
Void PrintPreviewByHtml(string parameter)
```

描述:

以预览的方式显示 html。该接口一般不单独使用，把生成的 html 数据通过该方法进行预览显示。

参数:

参数名	是否必填	数据类型	说明
parameter	必填	字符串	需要打印的 html 内容

返回值:

无

代码示例:

```
ctl.PrintPreviewByHtml(content);
```

相同效果:

跟 LoadPrintPreview() 方法效果一致

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=PrintPreviewByHtml>

8.4.7. *多文档合并生成 html 数据

接口:

```
String GetPrintPreviewHTMLByMixedFiles (json parameter, function errorCallback)
```

描述:

给定若干文档作为子文档合并后转换成预览下的 HTML。

参数：

参数名	是 否 必填	数 据 类型	说明
parameter	必填	Json	以下参数是 json 的内容
+files	必填	字 符 串	文档内容字符串数组, 如果为空 则使用当前文档
+InsertLastTableRowToPageBottom	非 必 填	Bool	预览时, 是否将表格插入到页底
+WaterMark	非 必 填	Json	附加的水印信息。参数可参考本 文的水印接口设置。
+OutputMedicalUsingHtml	非 必 填	Bool	是否使用 HTML 标记输出医学表 达式而不使用图片。默认 false, 使用的绘图。
+CommitUserTrace	非 必 填	Bool	是否获取提交痕迹后的 HTML
errorCallback	非 必 填	回 调 函数	完成后的回调函数

返回值：

数据类型	说明
String	返回 html 内容字符串。

代码示例：

```
let res=ctl.GetPrintPreviewHTMLByMixedFiles(options);
```

相同效果：

跟 GetPrintPreviewHTML、GetPrintPreviewHTML2 效果一样, 这个不需要经过服务端

结合使用：

通过需要传 html 打印的接口 (PrintByHtml) 或者通过 html 预览等

接口类型：

服务端接口, 需要部署 .net core 服务并且通过机器码来进行注册

有无限制：

编辑器加载成功之后在任何情况下都能调用

示例地址:

[http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=GetPrintPreviewHTMLByM
ixedFiles](http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=GetPrintPreviewHTMLByM
ixedFiles)

8. 4. 9. *编辑器打印模式-可读可写

接口:

String ExtViewMode

描述:

获取或设置当前的打印模式。打印模式包含普通模式 Normal、续打 JumpPrint、偏移续打 OffsetJumpPrint、区域选择打印 BoundsSelection。

参数:

编辑器属性、可读可写

Normal: 不设置打印模式

JumpPrint: 设置续打

OffsetJumpPrint: 设置偏移续打

BoundsSelection: 设置区域选择打印

返回值:

数据类型	说明	备注
String	模式名称	

代码示例:

```
let res=ctl.ExtViewMode;//获取编辑器处于打印的哪种模式
ctl.ExtViewMode= "JumpPrint" ;//设置续打
```

相同效果:

跟 DCExecuteCommand("JumpPrintMode", false, true)//设置续打

DCExecuteCommand("OffsetJumpPrintMode", false, true)//设置偏移续打

SetBoundsSelectionViewMode(true)//设置区域选择打印

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用

示例地址：

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=ExtViewMode>

8. 4. 10. *浏览器打印

接口：

Bool PrintDocument(json options)

描述：

打印文档。

参数：

参数名	是否必填	数据类型	说明
options	必填	Json	参数信息，包含
+ CompletedCallback	非必填	函数	方法的回调函数
+ ForcePrintHeaderFooter	非必填	bool	是否强制打印页眉页脚，即使处于打印被选中的内容模式。
+ PrintTableCellBorder	必填	Bool	是否打印表格单元格边框，默认为 true。可选值为打印表格单元格边框 true、不打印 false、采用文档中的设置 null。
+ CleanMode	必填	Bool	默认为空，是否为整洁打印模式，可选值为 true:整洁打印, false:留痕打印, null:采用编辑器当前的留痕显示设置。
+ PrintRange	必填	String	默认 AllPages,表示打印范围，为一个字符串，可以为 AllPages 所有页面,Selection 选择内容,SomePages 多个页面,Current Page 当前页面
+ PrintMode	必填	String	默认 Normal,表示打印模式，为一个字符串，可以为 Normal 全部打印,OddPage 只

			打印奇数,EvenPage 只打印偶数。这里的页码是从 0 开始计算的。
+ Collate	必填	Bool	默认 false,是否为逐份打印, 为一个布尔值。
+ Copies	必填	Int	默认 1,打印份数, 为一个整数。
+ FromPage	必填	Int	默认 0, 从 0 开始计算的打印开始页码, 只有 PrintRange=SomePages 时本设置才有效。
+ ToPage	必填	Int	默认为总页数, 从 0 开始计算的打印结束开始页码, 只有 PrintRange=SomePages 时本设置才有效
+ SpecifyPageIndexs	必填	String	默认空, 打印指定页码列表, 页码从 1 开始计算, 各个项目之间用逗号分开, 如果项目中间有个横杠, 表示一个页码范围。可使用"1,3,6-11,12"这种格式。
+ BodyLayoutOffset	必填	Number	默认为 0, 正文偏移续打的纵向偏移量。当该值大于 0, 则续打设置无效。
+ PageIndexFix	必填	Int	默认为 0, 打印出来的页码值的修正量。
+ JumpPrintStartElementID	非必填	String	续打开始处的文档元素编号, 若该属性值有效, 则续打开始位置为该指定 ID 的要素的上边缘。
+ JumpPrintEndElementID	非必填	String	续打结束处元素编号, 若该属性值有效, 则结束处续打的开始位置为指定 ID 的要素的下边缘。
+ JumpPrint	非必填	Json	续打信息, 参数如下:
++ PageIndex	必填	Number	从 0 开始计算的续打开始的页码
++ Position	必填	Number	开始续打的位置
++ EndPageIndex	必填	Number	从 0 开始计算的下端续打的页码
++ EndPosition	必填	Number	下端续打的位置

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
let CreatePrintOptions =
{
    CleanMode: true,
    PrintRange: "CurrentPage",
    PrintMode: "Normal",
    Collate: true,
    Copies: 1,
    FromPage: 0,
    ToPage: 1,
    SpecifyPageIndexs: "1"
};
ctl.PrintDocument(CreatePrintOptions);
```

8.4.11. *通过元素 ID 来设置续打位置

接口:

Void SetJumpPrintByElementID(string startElementID, string endElementID)

描述:

根据加载的文档中结构化元素编号来设置续打位置

参数:

参数名	是否必填	数据类型	说明
startElementID	必填	字符串	开始元素编号
endElementID	必填	字符串	结束元素编号

返回值:

该接口无返回值

代码示例:

```
ctl.SetJumpPrintByElementID(start, end);
```

相同效果:

先开启续打功能，然后通过鼠标先选择一个续打起始位置，然后按住键盘的 ctrl 键在使

用鼠标选中一个结束位置

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制（讨论）:

编辑器初始化成功之后都能调用

示例地址:

<http://www.dcwriter.cn:8022/apiDocument/edit/index.html?apiname=SetJumpPrintByElementID>

8.4.12. *续打信息

接口:

String JumpPrint

描述:

获取或设置当前的续打模式。

参数:

无参数。

返回值:

数据类型	说明	备注
Json	<pre>{ "HasValidateInfo": false, //是否存在有效的信息 "Mode": "Normal", //续打模式，包含 Disable 禁止续打、Normal 常规续打模式，鼠标定位选择续打位置、Offset 整体偏移续打模式，鼠标定位偏移续打位置、Append 增量续打模式，程序设置续打的页数和位置，第一页不忽略标题行 "Enabled": true, //是否允许续打 "Page": null, //发生续打的页面对象 "PageIndex": 1, //发生续打的页面号 "NativePosition": 2751, //原始续打位置 "Position": 0, //实际使用的续打位置离续打页面顶端的距离 "StartPositionMode": // "Position", 续打位置模式，包含 Position 由 Position 属性明确指定续打位置、</pre>	

	ElementTop 指定文档元素的顶端位置、ElementBottom 指定文档元素的低端位置 “StartPosition”: null, //确定续打位置使用的文档元素对象。当 StartPositionMode 属性为 ElementTop 或 ElementBottom 时该属性才有效。 “EndPositionMode”: “Position”, //续打位置模式, 包含 Position 由 Position 属性明确指定续打位置、ElementTop 指定文档元素的顶端位置、ElementBottom 指定文档元素的低端位置 “EndElement”: null, //确定续打位置使用的文档元素对象。当 EndPositionMode 属性为 ElementTop 或 ElementBottom 时该属性才有效。 “AbsPosition”: 0, //续打的绝对位置 “NativeEndPosition”: 0, //原始续打结束位置, 如果小于等于 Position 值就无效 “EndPosition”: 0, //实际使用的续打结束位置离续打页面顶端的距离, 如果小于等于 Position 值就无效 “EndPageIndex”: -1 //续打结束位置处的页码 }	

代码示例:

```
let res=ctl.JumpPrint;  
ctl.JumpPrint={Mode:"Normal"}; // 开启续打模式
```

相同效果:

也可以通过编辑器命令 DCExecuteCommand("JumpPrintMode", false, true) 或者通过编辑器方法 SetJumpPrintMode(true) 方法来开启续打模式

结合使用:

无

接口类型:

纯前端属性, 不经过服务端

有无限制:

编辑器初始化成功之后都能调用。除了打印预览模式无效、但是打印预览模式下也能获取到续打信息

兼容接口:

五代编辑器独有接口。

接口优化:

无任何区别

示例地址:

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=JumpPrint>

8. 4. 13. *续打结束位置-可读可写

接口:

String JumpPrintEndPosition

描述:

获取或设置当前的续打结束位置。

参数:

编辑器属性、可读可写。

返回值:

数据类型	说明	备注
Number	续打结束位置	

代码示例:

```
let res=ctl.JumpPrintEndPosition;//获取续打结束位置
ctl.JumpPrintEndPosition =1000;//设置续打结束位置，该属性有效果必须在续打功能开启才行
```

相同效果:

无

结合使用:

设置续打结束位置必须要先开启续打功能

接口类型:

纯前端属性，不经过服务端

有无限制:

编辑器初始化成功之后都能调用

兼容接口:

五代编辑器独有接口。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=JumpPrintEndPosition>

8. 4. 14. *续打开始位置-可读可写

接口:

String JumpPrintPosition

描述:

获取或设置当前的续打开始位置。

参数:

编辑器属性、可读可写。

返回值

数据类型	说明	备注
Number	续打开始位置	

代码示例:

```
let res=ctl.JumpPrintPosition;//获取续打开始位置
ctl.JumpPrintPosition=1000;//设置续打开始位置，该属性起效果的前提是必须开启续打功能
```

相同效果:

无

结合使用:

设置续打位置必须要先开启续打功能

接口类型:

纯前端属性，不经过服务端

有无限制:

编辑器初始化成功之后都能调用

兼容接口:

五代编辑器独有接口。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=JumpPrintPosition>

8.4.15. *打印当前输入焦点所在的页

接口:

Void PrintCurrentPage()

描述:

打印当前页。判断编辑模式下光标位置所在页进行打印当前页。最好是处于编辑模式下调用该方法

参数:

无

返回值:

无

代码示例:

```
ctl.PrintCurrentPage();
```

相同效果:

通过 PrintDocument 里面的参数来控制打印当前页

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后都能调用（续打模式、区域选中模式、打印预览模式调用该接口无效）

兼容接口:

五代编辑器独有接口。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=PrintCurrentPage>

8.4.16. *打印预览

接口:

Bool LoadPrintPreview(json options)

描述:

打印预览文档。参数和 PrintDocument 相同，参数允许为空。

参数:

参数名	是否必填	数据类型	说明
options	必填	Json	参数信息，包含
+ PrintTableCellBorder	必填	Bool	是否打印表格单元格边框，默认为 true。 可选值为打印表格单元格边框 true、不打印 false、采用文档中的设置 null。
+ CleanMode	必填	Bool	默认为空，是否为整洁打印模式，可选值为 true:整洁打印, false:留痕打印, null:采用编辑器当前的留痕显示设置。
+ PrintRange	必填	String	默认 AllPages,表示打印范围，为一个字符串，可以为 AllPages 所有页面,Selection 选择内容,SomePages 多个页面,Current Page 当前页面
+ PrintMode	必填	String	默认 Normal,表示打印模式，为一个字符串，可以为 Normal 全部打印,OddPage 只打印奇数,EvenPage 只打印偶数。这里的页码是从 0 开始计算的。
+ Collate	必填	Bool	默认 false,是否为逐份打印，为一个布尔值。
+ Copies	必填	Int	默认 1,打印份数，为一个整数。
+ FromPage	必填	Int	默认 0，从 0 开始计算的打印开始页码，只有 PrintRange=SomePages 时本设置才有效。

+ ToPage	必填	Int	默认为总页数, 从 0 开始计算的打印结束开始页码, 只有 PrintRange=SomePages 时本设置才有效
+ SpecifyPageIndexs	必填	String	默认空, 打印指定页码列表, 页码从 0 开始计算, 各个项目之间用逗号分开, 如果项目中间有个横杠, 表示一个页码范围。可使用"1,3,6-11,12"这种格式。
+ BodyLayoutOffset	必填	Number	默认为 0, 正文偏移续打的纵向偏移量。当该值大于 0, 则续打设置无效。
+ PageIndexFix	必填	Int	默认为 0, 打印出来的页码值的修正量。
+ JumpPrintStartElementID	非必填	String	续打开始处的文档元素编号, 若该属性值有效, 则续打开始位置为该指定 ID 的要素的上边缘。
+ JumpPrintEndElementID	非必填	String	续打结束处元素编号, 若该属性值有效, 则结束处续打的开始位置为指定 ID 的要素的下边缘。
+ JumpPrint	非必填	Json	续打信息, 参数如下:
++ PageIndex	必填	Number	从 0 开始计算的续打开始的页码
++ Position	必填	Number	开始续打的位置
++ EndPageIndex	必填	Number	从 0 开始计算的下端续打的页码
++ EndPosition	必填	Number	下端续打的位置

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
let CreatePrintOptions =
{
    CleanMode: true,
    PrintRange: "CurrentPage",
    PrintMode: "Normal",
    Collate: true,
```

```
Copies: 1,  
FromPage: 0,  
ToPage: 1,  
SpecifyPageIndexes: "1"  
};  
ctl. LoadPrintPreview(CreatePrintOptions);  
//或者直接不传参数进入打印模式。没有特殊情况一般都是直接使用下面的代码  
ctl. LoadPrintPreview();
```

相同效果:

通过方法 PrintPreviewByHtml 预览的效果一致、通过这个方法客户可以自己实现打印预览的效果

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后都能调用。

兼容接口:

兼容接口-跟四代编辑器使用效果一致。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=LoadPrintPreview>

8.4.17. *关闭打印预览模式

接口:

```
Void ClosePrintPreview()
```

描述:

关闭打印预览。

参数:

无

返回值:

无返回值

代码示例:

```
ctl.ClosePrintPreview();
```

相同效果:

无

结合使用:

LoadPrintPreview()方法使用之后，调用上面的方法关闭预览模式

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后都能调用、只不过在打印预览模式下调用该方法才能起对应的效果

兼容接口:

兼容接口-跟四代编辑器使用效果一致。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=ClosePrintPreview>

8. 4. 18. *留痕模式下的打印预览-不推荐使用

接口:

Bool LoadPrintPreviewWithPermissionMark(json options)

描述:

以留痕的方式打印预览文档。参数和 PrintDocument 相同，参数允许为空。该接口参数强制设置 CleanMode 为 false，表示留痕模式。

参数:

参数名	是否必填	数据类型	说明
options	必填	Json	参数信息，包含

+ PrintTableCellBorder	必填	Bool	是否打印表格单元格边框，默认为 true。 可选值为打印表格单元格边框 true、不打印 false、采用文档中的设置 null。
+ CleanMode	必填	Bool	默认为空，是否为整洁打印模式，可选值为 true:整洁打印, false:留痕打印, null:采用编辑器当前的留痕显示设置。
+ PrintRange	必填	String	默认 AllPages,表示打印范围，为一个字符串，可以为 AllPages 所有页面,Selection 选择内容,SomePages 多个页面,Current Page 当前页面
+ PrintMode	必填	String	默认 Normal,表示打印模式，为一个字符串，可以为 Normal 全部打印,OddPage 只打印奇数,EvenPage 只打印偶数。这里的页码是从 0 开始计算的。
+ Collate	必填	Bool	默认 false,是否为逐份打印，为一个布尔值。
+ Copies	必填	Int	默认 1,打印份数，为一个整数。
+ FromPage	必填	Int	默认 0，从 0 开始计算的打印开始页码，只有 PrintRange=SomePages 时本设置才有效。
+ ToPage	必填	Int	默认为总页数,从 0 开始计算的打印结束开始页码，只有 PrintRange=SomePages 时本设置才有效
+ SpecifyPageIndexs	必填	String	默认空，打印指定页码列表，页码从 0 开始计算，各个项目之间用逗号分开，如果项目中间有个横杠，表示一个页码范围。可使用"1,3,6-11,12"这种格式。
+ BodyLayoutOffset	必填	Number	默认为 0，正文偏移续打的纵向偏移量。当该值大于 0，则续打设置无效。

+ PageIndexFix	必填	Int	默认为 0，打印出来的页码值的修正量。
+ JumpPrintStartElementID	非必填	String	续打开始处的文档元素编号，若该属性值有效，则续打开始位置为该指定 ID 的要素的上边缘。
+ JumpPrintEndElementID	非必填	String	续打结束处元素编号，若该属性值有效，则结束处续打的开始位置为指定 ID 的要素的下边缘。
+ JumpPrint	非必填	Json	续打信息，参数如下：
++ PageIndex	必填	Number	从 0 开始计算的续打开始的页码
++ Position	必填	Number	开始续打的位置
++ EndPageIndex	必填	Number	从 0 开始计算的下端续打的页码
++ EndPosition	必填	Number	下端续打的位置

返回值：

数据类型	说明
Bool	操作是否成功

代码示例：

```
let CreatePrintOptions =
{
  CleanMode: true,
  PrintRange: "CurrentPage",
  PrintMode: "Normal",
  Collate: true,
  Copies: 1,
  FromPage: 0,
  ToPage: 1,
  SpecifyPageIndexes: "1"
};
ctl.LoadPrintPreviewWithPermissionMark(CreatePrintOptions);
```

相同效果：

当需要打印预览的时候显示痕迹，有两个方法，第一个方法就是编辑模式下直接开启留痕视图模式，这个时候在调用打印预览模式就能看到痕迹了。第二个方法就是当编辑模式下不需要出来痕迹，那就调用 `ctl.LoadPrintPreview({"CleanMode":false})` 打印预览下也会显示痕迹。只是为了兼容四代编辑器才加上这个方法。但是都昌不推荐使用。

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后都能调用

兼容接口:

兼容接口-跟四代编辑器使用效果一致。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=LoadPrintPreviewWithPermissionMark>

8. 4. 19. *判断是否处于打印预览模式

接口:

Bool IsPrintPreview()

描述:

判断当前是否处于打印预览。

参数:

无

返回值:

数据类型	说明	备注
Bool	返回是否处于打印预览模式	

代码示例:

```
Let res=ctl.IsPrintPreview();
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器初始化成功之后都能调用、只不过在打印预览模式下调用该方法才能返回 true 值

兼容接口:

兼容接口-跟四代编辑器使用效果一致。

接口优化:

无任何区别

示例地址:

<http://www.dcwriter.cn:8022/apiDocument/edit/index.html?apiname=IsPrintPreview>

8. 4. 20. *开启续打模式

接口:

Bool SetJumpPrintMode()

描述:

调用一次开启续打功能, 在次调用关闭续打功能

参数--非必填:

bool 值: true--开启续打模式、false--关闭续打模式。当不传参数时, 第一次调用开启续打模式、第二次调用关闭续打模式。

返回值 (讨论是否需要返回值):

无

代码示例:

```
Let res=ctl. SetJumpPrintMode();
```

相同效果:

ctl.DCExecuteCommand("JumpPrintMode",false,true)//兼容 CS 写法

ctl.ExtViewMode="JumpPrint"//兼容 CS 写法

ctl.SetJumpPrintMode(true)//兼容四代写法

ctl.JumpMode={Mode:"Normal"};

都是开启续打功能

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后都能调用。

兼容接口:

兼容接口-跟四代编辑器使用效果一致。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=SetJumpPrintMode>

8. 4. 21. *设置结构化元素打印是否可见

接口:

Bool SetElementPrintVisibility(object id, bool visible)

描述:

设置元素是否打印时可见。打印预览、打印该元素就会隐藏不可见。通过编辑器的方法生成打印的 html 或者 base64-pdf 数据都是隐藏的

参数:

参数名	是否必填	数据类型	说明
Id	必填	Object	字符串类型的元素 ID, 为空时表示设置当前元素。
visible	必填	String	Visible: 打印时可见 Hidden: 打印时隐藏

返回值:

数据类型	说明	备注
Bool	设置是否成功	

代码示例:

```
let res=ctl.SetElementPrintVisibility(“field1”, “Hidden”);
```

相同效果:

模板中设置元素的 PrintVisibility 属性跟调用该方法的效果一致

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后都能调用。(预览模式下无效)

兼容接口:

兼容接口-跟四代编辑器使用效果一致。

接口优化:

无任何区别

示例地址:

<http://www.dewriter.cn:8022/apiDocument/edit/index.html?apiname=SetElementPrintVisibility>

8.4.22. *LastPrintPosition---弃用

接口:

Int LastPrintPosition

描述:

获取或设置最后一次的打印位置。一般本属性和控件的 JumpPrintPosition 属性搭配使用。比如在打印后存储该属性值,下次打开文档后,再设置 JumpPrintPosition 属性值。能设置上次打印结束的位置为续打起始位置。

参数:

无参数。

返回值:

数据类型	说明	备注
Number	打印位置	

代码示例:

```
let res=ctl. LastPrintPosition;  
ctl. LastPrintPosition =1000;
```

8. 4. 23. *获取正文最后一行到首页的位置

接口:

String LastPrintResult

描述:

获取或设置当前的续打模式。

参数:

无参数。

返回值:

数据类型	说明	备注
Json	最后一次打印位置信息 PageIndexFix:0//页面偏移位置 PageCount:1//总页数，从 1 开始计算 LastPageHeight:1686//最后一页正文内容到当前页的高度 BodyOffsetY:0//正文偏移高度 BodyHeight:1386//需要设置续打的位置、使用病历的追打，就是获取这个值来实现	

代码示例:

```
let res=ctl.LastPrintResult;//获取文档正文最后一个元素距离文档开头处的问题。
```

相同效果:

ctl.GetPrintResult()方法的效果一模一样。二者选其一即可

结合使用:

```
//客户需要实现病历先打印一部分，然后医生继续在原来基础上书写的时候，需要再  
原有纸张上进行追加打印。在调用打印方法的时候可以使用下面的代码获取到最后一次  
打印位置  
  
var result=ctl.LastPrintResult;
```

```
var lsatprintposition=result.BodyHeight;

//然后需要二次打印的时候，在调用打印方法前。调用以下代码：

ctl.SetJumpPrintMode(true);//开启续打模式

ctl.JumpPrintPosition=lsatprintposition;//使用上一次的打印位置
```

接口类型：

纯前端属性，不经过服务端

有无限制：

编辑器初始化成功之后都能调用

兼容接口：

五代编辑器独有接口。

接口优化：

无任何区别

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=LastPrintResult>

8. 4. 24. *获取正文最后一行到首页的位置

接口：

Json GetPrintResult()

描述：

获取加载文档第一次打印之后的打印参数，可以用来当医生在打印之后的文档基础上重新编辑需要再打印纸张上继续打印，就可以用该方法返回值中的 BodyHeight 属性设置续打

参数：

无。

返回值：

参数名	数据类型	说明
PageIndexFix	Int	页码修正量
PageCount	Int	总页数

LastPageHeight	Float	最后一页的正文最后一个元素到当前页正文第一个元素的位置
BodyOffsetY	Float	正文偏移量
BodyHeight	Float	文档正文最后一个元素的位置（距离正文第一个元素的位置）

代码示例：

```
var result = ctl.GetPrintResult();
alert("PageIndexFix: " + result.PageIndexFix +
      ", PageCount: " + result.PageCount +
      ", LastPageHeight: " + result.LastPageHeight +
      ", BodyOffsetY: " + result.BodyOffsetY +
      ", BodyHeight: " + result.BodyHeight
);
```

相同效果：

ctl.LastPrintResult 属性的效果一模一样，二者选取一即可。

结合使用：

```
//客户需要实现病历先打印一部分，然后医生继续在原来基础上书写的时候，需要再
原有纸张上进行追加打印。在调用打印方法的时候可以使用下面的代码获取到最后一次
打印位置

var result=ctl.GetPrintResult();
var lsatprintposition=result.BodyHeight;

//然后需要二次打印的时候，在调用打印方法前。调用以下代码：

ctl.SetJumpPrintMode(true);//开启续打模式

ctl.JumpPrintPosition=lsatprintposition;//使用上一次的打印位置
```

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器初始化成功之后都能调用。

兼容接口：

五代编辑器独有接口。

接口优化:

无任何区别

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=GetPrintResult>

8. 4. 25. *本地打印-静默打印**接口:**

Void LocalPrintDocuments (json options, function)

描述:

执行本地插件进行打印功能。第二个参数是回调函数

参数:

参数名	是否必填	数据类型	说明
options	必填	Json	参数信息
+ShowUI	必填	Bool	是否弹出打印对话框
+PrinterName	必填	String	打印机名称
+PrintPageRange	必填	String	打印范围。默认 All, 表示所有页面。 可以单独设置页码, 如“2, 5-7, 9”等混合格式。
+OffsetY	必填	Float	垂直偏移量。有的打印机需要单独设置, 否则打印显示不全。
+AutoFitPageSize	必填	Bool	是否整体缩放, 适应纸张大小
+CleanMode	必填	Array	是否整洁模式打印。
+ManualDuplex	必填	Bool	是否手动双面打印
+Files	必填	Array	需要打印的文档数组
+JumpPrintPosition	必填	Float	需要开始位置
+JumpPrintEndPosition	必填	Float	续打结束位置

返回值:

数据类型	说明	备注
Bool	操作是否成功	当前端向打印小程序中发送数据成功,

		且内部调用打印接口后，就会存在返回值，但是这个返回值不能用来判断打印机打印是否成功，打印机脱机或者不存在也是会存在返回值的。

代码示例：

```
var options = {
  ShowUI: false, //是否弹出打印对话框
  PrinterName: null, //自定义打印机名称，若为空则选择默认打印机
  PrintPageRange: "all", //支持格式"2, 5-7, 9"等混合格式
  OffsetY: 0,
  AutoFitPageSize: true,
  CleanMode: true,
  ManualDuplex: false,
  Files: null, //为空则打印当前文档，可设为二维数组支持混合合并文档打印
  JumpPrintPosition: null,
  JumpPrintEndPosition: null
};

ctl.LocalPrintDocuments(options, function (args) {
  console.log(args);
});
```

8. 4. 26. *单元格套打

接口：

Void TemplatePrintingWithCells()

描述：

单元格套打功能，比如一个表格第一次正常打印，但是当医生需要再此基础上补充内容时，需要进行单元格套打功能，第二次打印需要隐藏第一次已经打印的数据，只打印第二次录入数据。

参数：

参数名	是否必填	数据类型	说明

返回值:

数据类型	说明	备注
Bool		

代码示例:

无

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后都能调用。

兼容接口:

兼容四代。

接口优化:

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=index&pageUrl=TemplatePrintingWithCells.html>

8. 4. 27. *区域选择打印

接口:

Void SetBoundsSelectionMode(bool isenable,bool reverse)

描述:

用户开启区域选择打印，通过鼠标滑动选择需要打印的区域进行打印。

参数:

参数名	是否必填	数据类型	说明
isenable	必填	boolean	true: 开启区域选择功能

			false: 关闭区域选择功能
reverse	非必填	boolean	默认值: false, 正常区域选择打印 true 值: 反向打印, 打印未选择区域部分

返回值:

数据类型	说明	备注
无		

代码示例:

```
var ctl=document.getElementById("myWriterControl")  
ctl.SetBoundsSelectionMode(true);
```

相同效果:

无

结合使用:

ctl.PrintDocument();//打印区域选择后的内容

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器初始化成功之后都能调用。

兼容接口:

兼容四代。

接口优化:

新增第二个参数, 反向打印

示例地址:

<http://117.89.85.111:8086/apiDocument/edit/index.html?apiname=SetBoundsSelectionMode>

注意:

区域选择功能开启, 执行打印方法时 PrintDocument 方法不允许存在参数。当有其它特殊打印功能时, 可单独判断编辑器是否处于区域选择打印。是的话单独处理。ctl.RectInfo 等于 null 时, 表示不处于区域选择模式下。

8. 4. 28. *编辑模式下获取续打信息

接口:

json GetJumpPrintInfo()

描述:

客户需判断是否不处于打印预览模式然后调用该方法获取续打信息。

参数:

参数名	是否必填	数据类型	说明
无			

返回值 (Json):

数据类型	说明
Enabled	是否开启续打。true 处于续打模式、false 不处于
EndPageIndex	续打结束位置页码，页码从 0 开始计算
EndPosition	续打结束位置
Mode	续打类型：Normal：普通续打。Offset：偏移续打
PageIndex	续打开始页码，页码从 0 开始计算
Position	续打开始位置

代码示例:

<pre>var ctl=document.getElementById("myWriterControl") ctl.GetJumpPrintInfo();</pre>

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后都能调用。编辑模式下可任意调用

兼容接口:

五代编辑器独有接口。

接口优化:

无

示例地址:

<http://117.89.85.111:8086/apiDocument/edit/index.html?apiname=GetJumpPrintInfo>

注意:

无

8.4.29. *编辑模式下设置续打信息**接口:**

Void SetJumpPrintInfo(json jumpinfo)

描述:

客户需判断是否不处于打印预览模式然后调用该方法设置续打信息。

参数 (Json):

数据类型	类型	说明
Enabled	bool	是否开启续打。true 开启续打模式、false 不开启
EndPageIndex	int	续打结束位置页码，页码从 0 开始计算
EndPosition	float	续打结束位置
Mode	string	续打类型：Normal：普通续打。Offset：偏移续打
PageIndex	int	续打开始页码，页码从 0 开始计算
Position	float	续打开始位置

返回值:

参数名	说明
bool	true: 方法执行成功 false: 方法执行失败

代码示例:

```
var ctl=document.getElementById("myWriterControl")

var jumpinfo={

Enabled:true,

Mode:"Normal",
```

```
Position:790

"PageIndex": 0//注意设置续打信息必须设置续打页码该方法才有效果
}

ctl.SetJumpPrintInfo(jumpinfo);
```

相同效果:

结合使用:

 需要从获取打印信息最后一次打印位置和最后打印页码值相互配合使用

接口类型:

 纯前端接口，不经过服务端

有无限制:

 编辑器初始化成功之后都能调用。编辑模式下可任意调用

兼容接口:

 五代编辑器独有接口。

接口优化:

 无

示例地址:

<http://117.89.85.111:8086/apiDocument/edit/index.html?apiname=SetJumpPrintInfo>

注意:

 无

8. 4. 30. *打印预览模式下获取续打信息

接口:

 Void GetJumpPrintInfoForPrintPreview()

描述:

 客户需判断是否处于打印预览模式然后调用该方法获取续打信息。

参数:

参数名	是否必填	数据类型	说明
无			

返回值 (Json):

数据类型	说明
Enabled	是否开启续打。true 处于续打模式、false 不处于
EndPageIndex	续打结束位置页码，页码从 0 开始计算
EndPosition	续打结束位置
Mode	续打类型：Normal：普通续打。Offset：偏移续打
PageIndex	续打开始页码，页码从 0 开始计算
Position	续打开始位置

代码示例:

```
var ctl=document.getElementById("myWriterControl")

if(ctl.IsPrintPreview()==true){

ctl.GetJumpPrintInfoForPrintPreview();

}
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后都能调用。打印预览模式下可任意调用

兼容接口:

五代编辑器独有接口。

接口优化:

无

示例地址:

<http://117.89.85.111:8086/apiDocument/edit/index.html?apiname=GetJumpPrintInfoForPrintPreview>

注意:

无

8. 4. 31. *打印预览模式下设置续打信息

接口:

```
Void SetJumpPrintInfoForPrintPreview(json jumpinfo)
```

描述:

客户需判断是否处于打印预览模式然后调用该方法设置续打信息。

参数 (Json):

数据类型	类型	说明
Enabled	bool	是否开启续打。true 开启续打模式、false 不开启
EndPageIndex	int	续打结束位置页码, 页码从 0 开始计算
EndPosition	float	续打结束位置
Mode	string	续打类型: Normal: 普通续打。Offset: 偏移续打
PageIndex	int	续打开始页码, 页码从 0 开始计算
Position	float	续打开始位置

返回值:

参数名	说明
bool	true: 方法执行成功 false: 方法执行失败

代码示例:

```
var ctl=document.getElementById("myWriterControl")

var jumpinfo={

Enabled:true,

Mode:"Normal",

Position:790

"PageIndex": 0//注意设置续打信息必须设置续打页码该方法才有效果

}

ctl.SetJumpPrintInfoForPrintPreview(jumpinfo);
```

相同效果:

结合使用:

 需要从获取打印信息最后一次打印位置和最后打印页码值相互配合使用

接口类型:

 纯前端接口，不经过服务端

有无限制:

 编辑器初始化成功之后都能调用。编辑模式下可任意调用

兼容接口:

 五代编辑器独有接口。

接口优化:

 无

示例地址:

<http://117.89.85.111:8086/apiDocument/edit/index.html?apiname=SetJumpPrintInfoForPrintPreview>

注意:

 无

8.5. 格式

8.5.1. *获取当前光标处前面内容样式

接口:

 Json getObject()

描述:

 获取当前光标处的字体内容。返回 json 格式的字体内容信息。

参数:

 无。

返回值:

参数名	数据类型	说明

FontFamily	字符串	字体名称
FontSize	Float	字体大小
Bold	Bool	是否粗体
Italic	Bool	是否斜体
Underline	Bool	是否有下划线
Strikeout	Bool	是否有删除线
BorderBottom	Bool	是否有下边框（2024-4-15 版本弃用）
BorderTop	Bool	是否有上边框（2024-4-15 版本弃用）
BorderRight	Bool	是否有右边框（2024-4-15 版本弃用）
BorderLeft	Bool	是否有左边框（2024-4-15 版本弃用）
BorderWidth	Float	边框线的宽度（2024-4-15 版本弃用）

代码示例：

```
var result = ctl.getFontObject();
alert("Bold: " + result.Bold +
      ", FontFamily: " + result.FontFamily +
      ", FontSize: " + result.FontSize +
      ", Italic: " + result.Italic +
      ", Strikeout: " + result.Strikeout +
      ", Underline: " + result.Underline);
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器初始化成功之后都能调用。只有当编辑器存在选中状态的时候该接口才有效果（打印预览模式下该接口无效（能获取到默认样式，该数据不准））

兼容接口：

兼容四代-与四代编辑器使用效果一致。

接口优化：

新增几个样式、加粗、倾斜、下划线、删除线样式

示例地址:

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=getFontObject>

8.5.2. *设置光标选中内容样式

接口:

Bool setFontObject(json parameter)

描述:

设置当前字体格式。

参数:

参数名	是否必填	数 据 类 型	说明
parameter	非必填	Json	以下参数是 json 的内容
+FontFamily	非必填	字符串	字体名称
+FontSize	非必填	Float	字体大小
+Bold	非必填	Bool	是否粗体
+Italic	非必填	Bool	是否斜体
+Underline	非必填	Bool	是否有下划线
+Strikeout	非必填	Bool	是否有删除线
+BorderBottom	非必填	Bool	是否有下边框（2024-4-15 版本弃用）
+BorderTop	非必填	Bool	是否有上边框（2024-4-15 版本弃用）
+BorderRight	非必填	Bool	是否有右边框（2024-4-15 版本弃用）
+BorderLeft	非必填	Bool	是否有左边框（2024-4-15 版本弃用）
+BorderWidth	非必填	Float	边框线的宽度（2024-4-15 版本弃用）

返回值:

返回是否设置字体成功

代码示例:

```
var obj = {
    "fontFamily": "楷体",
    "fontSize": 21,
```

```
    "Bold": true
}
ctl.setFontObject(obj);
```

相同效果:

通过命令分别设置: **FontName:** 设置字体。**FontSize:** 设置字号。**Bold:** 设置粗体。**Italic:** 设置斜体。**Strikeout:** 设置删除线。**Underline:** 设置下划线。

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器初始化成功之后都能调用。只有当编辑器存在输入焦点时该接口才有效果(普通表单模式下也无效)

兼容接口:

兼容四代-与四代编辑器使用效果一致。

接口优化:

新增几个样式、加粗、倾斜、下划线、删除线样式

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=setFontObject>

8.5.3. *全局修改行间距

接口:

Bool AllLineHeight(string parameter)

描述:

表示设置全文的行间距。

参数:

参数名	是否必填	数据类型	说明
parameter	必填	字符串	需要设置行间距的类型。它有多种写法, 可设置标准行间距的倍数, 如参数 150%, 或者参数 1.5em; 也可以设置指定的行高, 如参数 10px 或 10。

返回值:

数据类型	说明	备注
无		

代码示例:

```
ctl.AllLineHeight('150%');//设置行间距为多倍行间距--1.5倍  
ctl.AllLineHeight('1.5em');//设置行间距为多倍行间距--1.5倍  
ctl.AllLineHeight('10px');//设置行间距为固定值--单位磅值  
ctl.AllLineHeight('1.5');//设置行间距为多倍行间距--1.5倍
```

相同效果:

通过快捷键 **ctrl+a** (全选正文) 然后调用命令设置行间距效果一致。命令
`ctl.DCExecuteCommand('alllineheight', true, "1.5");`

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器初始化成功之后都能调用。(只设置正文, 页眉页脚该方法不起效果)

兼容接口:

五代独立接口。

接口优化:

无。

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=AllLineHeight>

8.5.4. *剪切

接口:

`Void Cut()`

描述:

剪切选中的文档内容, 无用户参数, 无返回值, 快捷键为 **Ctl+X**。

参数:

无

返回值:

无

代码示例:

```
Let res=ctl.Cut();
```

相同效果:

通过快捷键 ctrl+x、命令 DCExecuteCommand("Cut", false, true) 效果一致

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后都能调用。编辑器设置只读、阅读模式下、内容保护、打印预览模式下、结构化元素设置了内容只读、不允许编辑、不允许删除下该功能是无效的。

兼容接口:

兼容四代-与四代编辑器使用效果一致。

接口优化:

无。

示例地址:

<http://www.dewriter.cn:8022/apiDocument/edit/index.html?apiname=Cut>

8.5.5. *复制

接口:

```
Void Copy()
```

描述:

复制被选中的文档内容。

参数:

无

返回值:

无

代码示例:

```
Let res=ctl.Copy();
```

相同效果:

通过快捷键 ctrl+c、命令 DCExecuteCommand("copy", false, true) 效果一致

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后都能调用。编辑器存在选中状态该方法才有效果（打印预览模式下调用该方法无效。）

兼容接口:

兼容四代-与四代编辑器使用效果一致。

接口优化:

无。

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=Copy>

8.5.6. *获取当前光标处段落格式

接口:

Json GetCurrentParagraphStyle()

描述:

获取当前光标处的段落格式。返回 json 格式的段落格式信息。（修改过默认段落样式后）在返回信息中，字体信息是不准确的，因为段落中可能存在多种字体格式。只有关于段落的属性才有效果。比如：行间距、段落前后间距、段落缩进量、悬挂缩进、段落列表方式等有效果

参数:

无。

返回值:

返回的是段落内容的 json 格式，包含：

参数名	数据类型	说明
TitleLevel	Int	标题层次级别
FontName	String	字体名称
FontHeight	Float	字体高度
FontSize	Float	字体大小
FontStyle	String	字体样式
Bold	Bool	是否粗体
BorderBottom	Bool	是否有下边框
BorderTop	Bool	是否有上边框
BorderRight	Bool	是否有右边框
BorderLeft	Bool	是否有左边框
BorderBottomColorString	String	下边框的颜色
BorderRightColorString	String	右边框线的颜色
BorderLeftColorString	String	左边框线的颜色
BorderTopColorString	String	上边框线的颜色
BorderStyle	String	边框线样式
BorderWidth	Float	边框线的宽度
BorderSpacing		边框线间距
ColorString	String	前景色
BackgroundColorString	String	背景色
VerticalAlign	String	文本垂直对齐方式--枚举值： Top：对齐到顶端 Middle：垂直居中对齐 Bottom：对齐到底端
Align	String	文本水平对齐方式--枚举值 Left：左对齐 Center：居中对齐 Right：右对齐 Justify：两边对齐

		Distribute: 分散对齐
PaddingBottom	Float	下内边距
PaddingLeft	Float	左内边距
PaddingRight	Float	右内边距
PaddingTop	Float	上内边距
LineSpacing	Float	行间距值, 若采用固定值则为行间距的值; 若为多倍行距则为行间距的倍数。
LeftIndent	Float	整个段落的左缩进量。
LineSpacingStyle	String	行间距样式, 可选值有 SpaceSingle=单倍行距。Space1pt5=1.5 倍行距。SpaceDouble=2 倍行距。SpaceExactly=最小值。SpaceSpecify=固定值。SpaceMultiple=多倍行距。
SpacingAfterParagraph	Float	段落后间距。
SpacingBeforeParagraph	Float	段落前间距。
FirstLineIndent	Float	段落第一行的缩进量, 即首行缩进量。
ParagraphOutlineLevel	int	段落大纲层次
ParagraphMultiLevel	bool	多层次段落列表
ParagraphListStyle	String	列表方式
LayoutAlign	String	内容布局对齐方式

代码示例:

```
var result = ctl.GetCurrentParagraphStyle();
```

相同效果:

与编辑器下的属性 CurrentStyle 效果一致

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器初始化成功之后都能调用。打印预览模式下获取无效

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=GetCurrentParagraphStyle>

8.5.7. *设置当前光标处段落格式

接口：

Bool SetCurrentParagraphStyle(json parameter)

描述：

设置当前段落的格式。

参数：

参数名	是否必填	数据类型	说明
parameter	非必填	Json	以下参数是 json 的内容
+FontFamily	非必填	字符串	字体名称
+FontSize	非必填	Float	字体大小
+Bold	非必填	Bool	是否粗体
+Italic	非必填	Bool	是否斜体
+Underline	非必填	Bool	是否有下划线
+Strikeout	非必填	Bool	是否有删除线
+BorderBottom	非必填	Bool	是否有下边框
+BorderTop	非必填	Bool	是否有上边框
+BorderRight	非必填	Bool	是否有右边框
+BorderLeft	非必填	Bool	是否有左边框
+BorderWidth	非必填	Float	边框线的宽度

返回值：

返回是否设置成功。没有对应的效果该放也返回 true，返回值没有任何意义

代码示例:

```
var result = ctl.SetCurrentParagraphStyle({
    "TitleLevel": -1,
    "Bold": false,
    "ColorString": null,
    "VerticalAlign": "Top",
    "Align": "Justify",
    "BorderTop": false,
    "BorderRight": false,
    "BorderLeft": false,
    "BorderBottom": false,
    "BorderBottomColorString": null,
    "BorderRightColorString": null,
    "BorderLeftColorString": null,
    "BorderTopColorString": null,
    "BorderStyle": "Solid",
    "BorderWidth": 0,
    "BorderSpacing": 0,
    "PaddingBottom": 0,
    "PaddingLeft": 0,
    "PaddingRight": 0,
    "PaddingTop": 0,
    "BackgroundColorString": null,
    "LineSpacing": 0,
    "LeftIndent": 100,
    "LineSpacingStyle": "SpaceDouble",
    "SpacingAfterParagraph": 200,
    "SpacingBeforeParagraph": 200,
    "FirstLineIndent": 100,
    "ParagraphOutlineLevel": -1,
    "ParagraphMultiLevel": false,
    "ParagraphListStyle": "ListNumberStyleArabic2",
    "FontStyle": "Regular",
    "FontHeight": 0,
    "FontName": "宋体",
    "FontSize": 10.5,
    "LayoutAlign": "EmbedInText"
});
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器初始化成功之后都能调用。编辑器存在输入焦点该方法才有效果。

注意:

不推荐使用该方法来设置段落样式。建议使用方法 `ParagraphFormat` 来设置段落样式

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=SetCurrentParagraphStyle>

8.5.8. *清空编辑器撤销记录

接口:

`Void ClearUndo()`

描述:

编辑器会记录用户的撤销和重做信息, 在长时间的编辑过程中会积累大量的信息, 这些信息会占用系统内容, 调用该接口可清空这些信息。单独当编辑器调用了 `LoadDocument` 系列的加载方法会自动清空撤销和重做信息的

参数:

无

返回值:

无

代码示例:

```
ctl.ClearUndo();
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后都能调用。

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<http://www.dcwriter.cn:8022/apiDocument/edit/index.html?apiname=ClearUndo>

8.5.9. hasLocalPaste——讨论

接口:

Bool hasLocalPaste()

描述:

获取浏览器的粘贴权限。由于浏览器一般可自定义设置粘贴权限，为正常使用粘贴功能，需要先进行权限的校验，判断当前是否可以使用浏览器的粘贴权限。该接口通过 navigator.clipboard 和 navigator.clipboard.write 来判断权限。

参数:

无

返回值:

数据类型	说明	备注
Bool	返回当前的粘贴权限	

代码示例:

```
ctl.hasLocalPaste ();
```

8.5.10. *删除选择区域内容

接口:

Void DeleteSelection(bool showUI)

描述：

删除选择内容。该接口会受到文档只读、元素只读、表单模式、权限校验、内容保护等设置的影响。

参数：

参数名	是否必填	数据类型	说明
showUI	必填	Bool	该值表示在删除时遇到不允许删除的内容是否通过对话框来进行提示。默认值：提示 不传参数：提示 true：提示。 false：不提示

返回值：

无

代码示例：

```
ctl.DeleteSelection(true);
```

相同效果：

backspace 键、delete 键删除效果一致

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器初始化成功之后都能调用。受权限、不允许编辑情况下方法无效

注意：

当文档选项:DocumentOptions.BehaviorOptions.PromptProtectedContent="None"时，即使该方法传 true，也没有提示。

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname>DeleteSelection>

8.5.11. *粘贴

接口:

Void Paste()

描述:

把从编辑器内部复制的内容粘贴到编辑器中。不允许从外部复制内容，然后调用该方法进行粘贴（需要通过 https 访问编辑器）才允许粘贴外部复制的内容。默认效果从编辑器内部复制是什么效果，调用该方法粘贴就是什么效果（复制结构化元素，粘贴之后元素的 ID 会自动修改）。

参数:

参数名	是否必填	数据类型	说明
无			

返回值:

数据类型	说明	备注
无		

代码示例:

```
Let result=ctl.Paste()
```

相同效果:

命令 DCExecuteCommand("Paste", false, true) 效果一致。

快捷键: ctrl+v。内部复制，快捷键粘贴一致。

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后都能调用。存在输入焦点才能调用该方法

兼容接口:

兼容四代-跟四代编辑器使用一致。

接口优化:

无。

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=Paste>

8.5.12. *重做

接口:

Void Redo()

描述:

重做。重复上一次操作，快捷键为 ctrl+y。

参数:

无

返回值:

无

代码示例:

```
ctl.Redo();
```

相同效果:

命令 DCExecuteCommand("Redo", false, true)、ctrl+y 效果一致。

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后都能调用。编辑器存在输入焦点才能重做。

兼容接口:

兼容四代-跟四代编辑器使用一致。

接口优化:

无。

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=Redo>

8.5.13. *撤销

接口:

```
Void Undo()
```

描述:

撤销上一次操作, 快捷键为 ctrl+z。

参数:

无

返回值:

无

代码示例:

```
ctl.Undo();
```

相同效果:

命令 DCExecuteCommand("Undo", false, true)、ctrl+z 效果一致。

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器初始化成功之后都能调用。编辑器存在输入焦点才能撤销。

兼容接口:

兼容四代-跟四代编辑器使用一致。

接口优化:

无。

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=Undo>

8.5.14. DCPaste--讨论

接口:

```
bool DCPaste()
```

描述:

执行粘贴操作。

参数:

无

返回值:

数据类型	说明	备注
Bool	操作是否成功	

代码示例:

```
Let res=ctl.DCPaste()
```

8.5.15. *格式刷

接口:

Void BeginFormatBrush()

描述:

开始执行格式刷功能。编辑器会记下插入点或选择内容所在的文档内容格式和段落格式，然后在进入格式刷激活状态，当用户使用鼠标选择一段文档内容，则会把未激活前选中的内容样式修改掉当前选中的内容格式和段落格式。格式刷激活模式下，鼠标点击任意地方，退出格式刷模式。

参数:

无

返回值:

无

代码示例:

```
ctl.BeginFormatBrush()
```

相同效果:

命令 DCExecuteCommand("FormatBrush", false, true) 效果一致。

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制：

编辑器初始化成功之后都能调用。只有当编辑器存在输入焦点的时候该方法才有效果（普通表单模式下也无效）。

兼容接口：

兼容四代-跟四代编辑器使用一致。

接口优化：

无。

示例地址：

<http://www.dcwriter.cn:8022/apiDocument/edit/index.html?apiname=BeginFormatBrush>

8.5.16. *设置当前段落格式

接口：

Bool ParagraphFormat(json parameter)

描述：

设置当前段落的格式。

参数：

参数名	是否必填	数据类型	说明
parameter	非必填	Json	以下参数是 json 的内容
+ LineSpacing	非必填	Float	行间距值，若采用固定值则为行间距的值；若为多倍行距则为行间距的倍数。
+ LeftIndent	非必填	Float	段落左侧缩进量。单位 1/300 英寸
+ LineSpacingStyle	非必填	String	行 间 距 样 式 ， 为 DCSoft.Drawing.LineSpacingStyle 类型，可选值有 SpaceSingle=单倍行距。 Space1pt5=1.5 倍行距。 SpaceDouble=2 倍行距。 SpaceExactly=最小值。

			SpaceSpecify=固定值。 SpaceMultiple=多倍行距
+ SpacingAfter	非必填	Float	段后间距
+ SpacingBefore	非必填	Float	段前间距
+ FirstLineIndent	非必填	Float	首行缩进量。单位 1/300 英寸
+ OutlineLevel	非必填	Bool	段落大纲层次
+ ParagraphMultiLevel	非必填	Bool	多层次段落列表
+ ParagraphListStyle	非必填	Bool	段落列表的样式，包含 None 、 ListNumberStyle 、 ListNumberStyleArabic1 、 ListNumberStyleArabic2 、 ListNumberStyleLowercaseLetter 、 ListNumberStyleLowercaseRoman 、 ListNumberStyleNumberInCircle 、 ListNumberStyleSimpChinNum1 、 ListNumberStyleSimpChinNum2 、 ListNumberStyleTradChinNum1 、 ListNumberStyleTradChinNum2 、 ListNumberStyleUppercaseLetter 、 ListNumberStyleUppercaseRoman 、 ListNumberStyleZodiac1 、 ListNumberStyleZodiac2 、 BulletedList 、 BulletedListBlock 、 BulletedListDiamond 、 BulletedListCheck 、 BulletedListRightArrow 、 BulletedListHollowStar 详细介绍到： 9.18 段落列表样式

返回值：

无

代码示例：

```
var result = ctl.ParagraphFormat({
    "LeftIndent": 100,
```

```
"LineSpacingStyle": "SpaceDouble",
"SpacingAfter": 200,
"SpacingBefore": 200,
"FirstLineIndent": 100,
"OutlineLevel": -1,
"ParagraphMultiLevel": false,
"ParagraphListStyle": "ListNumberStyleArabic2", });
```

相同效果:

SetCurrentParagraphStyle 方法类似

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后都能调用。只有当编辑器存在输入焦点的时候该方法才有效果。

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=ParagraphFormat>

8.5.17. *获取光标前面内容字体大小

接口:

Json getFontSize()

描述:

获取当前光标处字体大小

参数:

无。

返回值:

数据类型	说明
Float	字体大小信息

代码示例:

```
var result = ctl.getFontSize();
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后都能调用。预览模式下和阅读模式下该方法无效

兼容接口:

兼容四代-跟四代编辑器使用一致。

接口优化:

无。

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=getFontSize>

8.5.18. *获取光标前面内容是否加粗

接口:

ctl.CurrentBold

描述:

获取当前光标处前面的内容是否是粗体。

参数:

编辑器属性

返回值:

bool: true: 粗体。false: 不是粗体

代码示例:

```
Let res=ctl.CurrentBold ;
```

相同效果:

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器初始化成功之后都能调用。预览模式下和阅读模式下该方法无效

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=CurrentBold>

8.5.19. *获取光标前面内容段落对齐方式

接口：

ctl.CurrentParagraphAlign

描述：

获取当前光标处段落水平对齐方式。

参数：

编辑器属性

返回值：

String 值：

Left：段落居左对齐。

Center：段落居中对齐。

Right：段落居右对齐。

Justify：段落两端对齐。

Distribute：段落分散对齐。

代码示例：

```
Let res=ctl.CurrentParagraphAlign;
```

相同效果：

无

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器初始化成功之后都能调用。预览模式下和阅读模式下该方法无效

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=CurrentParagraphAlign>

8.5.20. *获取光标前面内容段落样式信息

接口:

ctl.CurrentStyle

描述:

获取当前光标处段落样式信息。

参数:

编辑器属性

返回值:

与 [8.5.6](#) 返回值一样

代码示例:

```
Let res=ctl.CurrentStyle;
```

相同效果:

与 GetCurrentParagraphStyle 方法效果一模一样。建议使用 GetCurrentParagraphStyle 方法获取当前段落的样式。不推荐使用 CurrentStyle 属性

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后都能调用。打印预览模式下获取无效

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=CurrentStyle>

8.6. 痕迹

痕迹保留，用户对文档中的内容的新增、修改和删除都能产生痕迹信息并保留在文档中，痕迹信息包括操作员的名称、时间、操作类型和操作的文档内容。痕迹信息能在用户界面上展现出来。

痕迹信息保存在 XML 的文档样式中。文档样式设置中包含了 CreateIndex 子元素，其数值表示文档内容的创建者历史信息编号；包含了 DeleterIndex 子元素，其数值表示文档内容的逻辑删除者历史信息编号。

例如对于 XML 片段 “<TextElement StyleIndex=“2”>腹痛。</TextElement>”，它的 StyleIndex 属性值为 2，据此可以在 ContentStyles/Styles 列表中找到 XML 片段 “<Style Index=“2”> <CreatorIndex>1</CreatorIndex> <Underline>true</Underline> </Style>”，可以获得 UnderLine 元素值为 true，说明在用户界面上显示或打印这段文本时是具有下划线的；CreatorIndex 元素值为 1，系统可以进而在 UserHistories 元素下查找，可以获得 XML 片段

```
<History>
  <ID>1</ID>
  <Name>李医师</Name>
  <SavedTime>2012-11-15 9:30:13</SavedTime>
  <PermissionLevel>1</PermissionLevel>
```

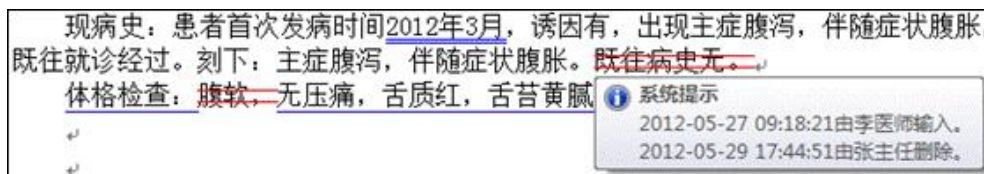
```
<Description>查房</Description>
</History>
```

这样就可以知道这段文本是李医师于 2012 年 11 月 15 日输入的，而且李医师的 PermissionLevel 为 1，一般解释为中级医生权限。

编辑器能设置可视化的显示用户操作痕迹，对于不同等级的用户留下的痕迹，其可视化样式是不一样的，而且这种样式是可以动态配置的。

在默认情况下，对于等级小于等于 0 的用户，无下划线无背景色，删除线为一条淡红色线。对于等级为 1 的用户，无背景色，下划线为 1 条蓝色线，删除线为 1 条红色线。对于等级为 2 的用户，背景色为淡黄色，下划线为双蓝色线，删除线为双红色线。对于等级大于等于 3 的用户，背景色为黄色，下划线为 2 条蓝色，删除线为 2 条红色。当鼠标停留在到文档内容上时，会显示一个提示标签，说明文档内容的创建信息和被逻辑删除的信息。

当鼠标移动到标记了 **痕迹** 信息的文档内容上面时会显示相关的提示文本进行详细说明。如下图所示，这段文档就留下了用户修改 **痕迹**：



在这里，被红色删除线标注的文档内容就是被逻辑删除的部分。当编辑器不显示逻辑删除的内容时，这段文档就显示如下：

现病史：患者首次发病时间2012年3月，诱因有，出现主症腹泻，伴随症状腹胀，既往就诊经过。刻下：主症腹泻，伴随症状腹胀。体格检查：无压痛，舌质红，舌苔黄腻，脉滑数。

当不显示修改 **痕迹** 时，也就是处于整洁显示模式下，此时文档内容显示如下：

现病史：患者首次发病时间2012年3月，诱因有，出现主症腹泻，伴随症状腹胀，既往就诊经过。刻下：主症腹泻，伴随症状腹胀。体格检查：无压痛，舌质红，舌苔黄腻，脉滑数。

这样文档的权限控制从传统的文件级别深入到文档内部，这成为电子病历应用系统实现三级权限控制和留痕技术的基础。

8.6.1. *根据参数进行用户登录

接口：

```
bool UserLoginByParameter(string userID, string userName, int
permissionLevel)
```

描述：

要实现痕迹保留和权限控制，首先需要进行用户登录，告知编辑器控件当前用户使用的用户名和权限等级。之后新增、修改或逻辑删除的内容都归于当前用户名下。

UserLogin、UserLoginByUserLoginInfo 接口和本接口具有相同的功能，参数和返回值相同。

参数：

参数名	是否必填	数据类型	说明
userID	必填	String	字符串类型的用户编号
userName	必填	String	字符串类型的用户名称
permissionLevel	必填	int	用户等级。数字越大，权限越大。默认情况下，高权限用户可以删除低权限用户输入的内容，而低权限的用户不允许删除高权限用户输入的内容。

返回值：

数据类型	说明	备注
Bool	操作是否成功	

代码示例：

```
var result = ctl.UserLoginByParameter('zhangsan', '张三', 1);
```

相同效果：

```
ctl.UserLogin();  
ctl.UserLoginByUserLoginInfo()
```

与上面两个方法效果一样、推荐用户使用 UserLoginByUserLoginInfo 方法。

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器初始化成功之后都能调用。但是需要该接口有对应的效果，一定是需要 LoadDocumentFromString 或者 LoadDocumentFromBase64String 接口执行完毕了之后再调用登录接口，用户对于病历的操作才会记录痕迹且执行接口默认就是处于留痕视图模式，也就

是痕迹样式会显示出来。

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=UserLoginByParameter>

8.6.2. *通过容器对象提交痕迹信息

接口：

bool CommitContentUserTrace (object element)

描述：

提交指定容器元素中的用户痕迹信息。提交是指删除所有被逻辑删除的元素，清除输入内容的痕迹。该接口需要在管理员模式下运行。

参数：

参数名	是否必填	数据类型	说明
element	必填	Object	容器文档元素对象, 支持元素 ID、NativeHandle 对象、引用对象。

返回值：

数据类型	说明	备注
Bool	操作是否成功	

代码示例：

<pre>var fieldlist=ctl.GetAllInputFields() var result= ctl.CommitContentUserTrace(fieldlist[0].NativeHandle);</pre>

相同效果：

ctl.CommitContentUserTraceById() 效果一致

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器初始化成功之后都能调用。阅读模式、只读模式无效。打印预览下有效。

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=CommitContentUserTrace>

8.6.3. *通过容器编号（ID）提交痕迹信息

接口：

```
bool CommitContentUserTraceById (string id)
```

描述：

提交指定容器元素中的用户痕迹信息。提交是指删除所有被逻辑删除的元素，清除输入内容的痕迹。该接口需要在管理员模式下运行。

参数：

参数名	是否必填	数据类型	说明
element	必填	Object	容器文档元素对象, 支持元素 ID

返回值：

数据类型	说明	备注
Bool	操作是否成功	

代码示例：

```
var result = ctl.CommitContentUserTraceById('field1');
```

相同效果：

ctl.CommitContentUserTrace() 效果一致

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器初始化成功之后都能调用。阅读模式、只读模式无效。打印预览下有效。

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=CommitContentUserTraceById>

8.6.4. *根据用户登录信息执行用户登录操作

接口：

Bool UserLoginByUserLoginInfo (json parameter, bool updateUI)

描述：

要实现痕迹保留和权限控制，首先需要进行用户登录，告知编辑器控件当前用户使用的用户名和权限等级。之后新增、修改或逻辑删除的内容都归于当前用户名下。

和 UserLoginByParameter 具有相同的功能，只是参数不同。

参数：

参数名	是否必填	数据类型	说明
parameter	必填	Json	用户信息，参数如下：
+ID	必填	String	字符串类型的用户编号
+Name	必填	String	字符串类型的用户名称
+permissionLevel	必填	int	用户等级。数字越大，权限越大。默认情况下，高权限用户可以删除低权限用户输入的内容，而低权限的用户不允许删除高权限用户输入的内容。
+ ClientName	非必填	String	字客户端电脑名称
+ Description	非必填	String	其他说明
updateUI	必填	Bool	是否刷新文档

返回值：

返回是否设置成功

代码示例:

```
ctl.UserLoginByUserLoginInfo(  
    {  
        ID: userid, // 用户编号  
        Name: username, // 用户名  
        ClientName: "KDD", // 客户端电脑名称  
        PermissionLevel: userlevel, // 用户授权等级  
        Description: description // 其他说明  
    }, true);
```

相同效果:

```
ctl.UserLogin();
```

```
ctl.UserLoginByParameter()
```

与上面两个方法效果一样、推荐用户使用 UserLoginByUserLoginInfo 方法。

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器初始化成功之后都能调用。但是需要该接口有对应的效果，一定是需要 LoadDocumentFromString 或者 LoadDocumentFromBase64String 接口执行完毕了之后再调用登录接口，用户对于病历的操作才会记录痕迹且执行接口默认就是处于留痕视图模式，也就是痕迹样式会显示出来。

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=UserLoginByParameter>

8.6.5. *整洁视图-清洁视图模式

接口:

```
Void CleanViewMode()
```


描述:

将编辑器控件切换到整洁视图模式。编辑器在整洁视图模式下不支持文档内容授权控制,不显示历史上保留的痕迹,显示的是文档内容的最终版本。功能和 ComplexViewMode 命令的作用是相反的。

参数:

无

返回值:

无

代码示例:

```
ctl.CleanViewMode();
```

相同效果:

```
ctl.DCExecuteCommand("CleanViewMode", false, true) 效果一致
```

结合使用:

无

接口类型:

纯前端接口,不经过服务端

有无限制:

编辑器初始化成功之后都能调用。在打印预览模式下能隐藏痕迹信息(阅读、只读模式下无效)

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=CleanViewMode>

8.6.6. *拒绝用户修订内容

接口:

```
viod RejectUserTrace()
```

描述:

把用户操作恢复成一开始编辑器加载文档的效果。把这个期间用户删除的内容进行回滚，输入的内容进行物理删除。

参数：

参数名	是否必填	数据类型	说明
无			

返回值：

数据类型	说明	备注
bool	true：已拒绝用户修订（处理了痕迹）、false：文档中无痕迹无须拒绝用户修订	

代码示例：

```
let ctl=document.getElementById("myWriterControl");
ctl.DCExecuteCommand("AdministratorViewMode",false,true);
var result= ctl.RejectUserTrace();
```

相同效果：

```
DCExecuteCommand("RejectUserTrace",false,true)
```

结合使用：

```
ctl.DCExecuteCommand("AdministratorViewMode",false,true);//必须在管理员模式下才生效
```

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。阅读模式、只读模式无效。打印预览下有效

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=RejectUserTrace>

8.6.7. *清除文档中所有用户痕迹

接口:

Void ClearAllUserTrace()

描述:

清除文档中所有的用户修改痕迹信息, 这个操作是不可恢复的, 不可 Undo/Redo。只有当编辑器控件处于管理员模式该命令才有效。

注意: 清除痕迹时会将被标记为逻辑删除的内容给去掉逻辑删除的标记, 使得这些内容“洗白”了, 从而在整洁视图模式下又能显示出来。

参数:

无

返回值:

无

代码示例:

```
ctl.ClearAllUserTrace();
```

相同效果:

```
ctl.DCExecuteCommand("ClearAllUserTrace", false, true)
```

结合使用:

```
ctl.DCExecuteCommand("AdministratorViewMode", false, true); // 必须在管理员模式下才生效
```

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。阅读模式、只读模式无效。打印预览下有效

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=ClearAllUserTrace>

8.6.8. *清除文档中选中内容的痕迹

接口:

Void ClearUserTrace()

描述:

清除文档中被选中部分的用户操作痕迹, 这个操作是可恢复的。只有当编辑器控件处于管理员模式该命令才有效。

参数:

无

返回值:

无

代码示例:

```
ctl.ClearUserTrace();
```

相同效果:

```
ctl.DCExecuteCommand("ClearUserTrace", false, true)
```

结合使用:

```
ctl.DCExecuteCommand("AdministratorViewMode", false, true); // 必须在管理员模式下才生效
```

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。阅读模式、只读模式无效。打印预览下有效

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=ClearUserTrace>

8.6.9. *提交文档中痕迹

接口:

Void CommitUserTrace()

描述:

提交所有的用户痕迹信息,删除所有被逻辑删除的元素,清空痕迹信息。只有在管理员模式下而且编辑器控件没有设置内容只读则该命令才有效。

注意提交痕迹和清除痕迹的区别。

提交痕迹是彻底删除被逻辑删除的元素,清空用户输入的痕迹信息;清除痕迹会将被标记为逻辑删除的内容给去掉逻辑删除的标记,清空用户输入的痕迹信息。

参数:

无

返回值:

无

代码示例:

```
ctl.CommitUserTrace();
```

相同效果:

```
ctl.DCExecuteCommand("CommitUserTrace", false, true)
```

结合使用:

```
ctl.DCExecuteCommand("AdministratorViewMode", false, true); // 必须在管理员模式下才生效
```

接口类型:

纯前端接口,不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。阅读模式、只读模式无效。打印预览下有效

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=ClearUserTrace>

8.6.10. *提交文档中痕迹

接口:

Void CommitDocumentUserTrace()

描述:

提交所有的用户痕迹信息,删除所有被逻辑删除的元素,清空痕迹信息。只有在管理员模式下而且编辑器控件没有设置内容只读则该命令才有效。

该接口和 CommitUserTrace 接口具有相同的功能,是为了兼容四代的接口。

参数:

无

返回值:

无

代码示例:

```
ctl.CommitDocumentUserTrace();
```

相同效果:

```
ctl.DCExecuteCommand("CommitUserTrace", false, true)

ctl.CommitUserTrace()
```

结合使用:

```
ctl.DCExecuteCommand("AdministratorViewMode", false, true); // 必须在管理员模式
下才生效
```

接口类型:

纯前端接口,不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。阅读模式、只读模式无效。打印预览下有效

兼容接口:

兼容四代-跟四代编辑器的用法一致。

接口优化:

无。

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=ClearUserTrace>

8.6.11. *获取文档的用户痕迹信息列表

接口：

Array GetDocumentUserTrackInfos (bool isClearMode)

描述：

获取文档的痕迹列表信息。

参数：

参数名	是否必填	数据类型	说明
isClearMode	非必填	Bool	不填参数：获取的全是创建的痕迹。 传 true：同时获取创建和逻辑删除的痕迹信息

返回值：

该接口返回痕迹的数组信息，单个数组项的信息如下：

参数名	数据类型	说明
CommentText	String	批注文本信息。BehaviorOptions.FillCommentToUserTrackList 在选项设置为 true 时，表示文档中的批注将作为痕迹信息显示的用户痕迹列表中。添加后获取痕迹信息时将获取到批注文本。默认该选项是 false。 前提：在不带痕迹的内容上插入批注才会记录痕迹信息
InfoType	String	痕迹类型，其中“Comment”表示批注类型的痕迹；“Delete”表示删除类型的痕迹；“Create”表示创建类型的痕迹。
NativeHandle	String	痕迹对应内容的第一个元素的 NativeHandle 对象信息
PermissionLevel	Int	对应用户的权限信息
SaveTime	String	保存的时间
Text	String	痕迹的内容信息

UserID	String	对应用户的用户编号信息
UserName	String	对应用户的用户姓名信息

代码示例:

```
var res=ctl.GetDocumentUserTrackInfos ()
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

兼容四代-跟四代编辑器的用法一致。

接口优化:

无。

示例地址:

<http://www.dewriter.cn:8022/apiDocument/edit/index.html?apiname=GetDocumentUserTrackInfos>

8.6.12. NavigateByUserTrackInfo--讨论

接口:

Bool NavigateByUserTrackInfo (string NativeHandle)

描述:

根据痕迹对应的 NativeHandle 信息，在文档中定位内容。

参数:

参数名	是否必填	数据类型	说明
NativeHandle	必填	String	痕迹对应内容的第一个元素的 NativeHandle 对象信息。该参数是 GetDocumentUserTrackInfos 返回值中的

			NativeHandle 信息。

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

Var res=ctl. NavigateByUserTrackInfo (“123”)

8. 6. 13. *编辑器是否处于留痕模式

接口:

Bool InComplexViewMode ()

描述:

判断是否处于留痕模式。该接口是根据三个选项值共同判断， DocumentOptions.SecurityOptions.ShowLogicDeletedContent、 DocumentOptions.SecurityOptions.ShowPermissionMark、 DocumentOptions.SecurityOptions.ShowPermissionTip， 当这个三个选项值都是返回 true 时，认为当前处于留痕模式。

参数:

无

返回值:

数据类型	说明
Bool	是否处于留痕模式 true: 当前为留痕视图模式 false: 当前不为留痕视图模式

代码示例:

var res=ctl. InComplexViewMode ()

相同效果:

同时判断
DocumentOptions.SecurityOptions.ShowLogicDeletedContent、
DocumentOptions.SecurityOptions.ShowPermissionMark、
DocumentOptions.SecurityOptions.ShowPermissionTip
都为 true 时。处于留痕模式

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。阅读、预览模式下无效。

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=InComplexViewMode>

8.6.14. *编辑器是否处于清洁模式

接口:

Bool InCleanViewMode()

描述:

判断是否处于清洁模式。该接口是根据三个选项值共同判断， DocumentOptions.SecurityOptions.ShowLogicDeletedContent、 DocumentOptions.SecurityOptions.ShowPermissionMark、 DocumentOptions.SecurityOptions.ShowPermissionTip， 当这个三个选项值都是返回 false 时，认为当前处于清洁模式。

参数:

无

返回值:

数据类型	说明
Bool	是否处于清洁模式 true: 处于清洁模式 false: 不处于清洁模式

代码示例:

```
var res=ctl.InCleanViewMode()
```

相同效果:

同时判断
DocumentOptions.SecurityOptions.ShowLogicDeletedContent、
DocumentOptions.SecurityOptions.ShowPermissionMark、
DocumentOptions.SecurityOptions.ShowPermissionTip
都为 false 时。处于清洁模式

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。阅读、预览模式下无效。

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=InCleanViewMode>

8.6.15. *根据参数进行用户登录

接口：

bool UserLogin(string userID, string userName, int permissionLevel)

描述：

要实现痕迹保留和权限控制，首先需要进行用户登录，告知编辑器控件当前用户使用的用户名和权限等级。之后新增、修改或逻辑删除的内容都归于当前用户名下。

UserLoginByParameter、UserLoginByUserLoginInfo 接口和本接口具有相同的功能，参数和返回值相同。

参数：

参数名	是否必填	数据类型	说明
userID	必填	String	字符串类型的用户编号
userName	必填	String	字符串类型的用户名称
permissionLevel	必填	int	用户等级。数字越大，权限越大。默认情况下，

			高权限用户可以删除低权限用户输入的内容，而低权限的用户不允许删除高权限用户输入的内容。

返回值：

数据类型	说明	备注
Bool	操作是否成功	

代码示例：

```
var result = ctl.UserLogin('zhangsan', '张三', 1);
```

相同效果：

```
ctl.UserLoginByParameter();
```

```
ctl.UserLoginByUserLoginInfo()
```

与上面两个方法效果一样、推荐用户使用 UserLoginByUserLoginInfo 方法。

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器初始化成功之后都能调用。但是需要该接口有对应的效果，一定是需要 LoadDocumentFromString 或者 LoadDocumentFromBase64String 接口执行完毕了之后再调用登录接口，用户对于病历的操作才会记录痕迹且执行接口默认就是处于留痕视图模式，也就是痕迹样式会显示出来。

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=UserLogin>

8.7. 病程记录

8.7.1. *通过病程 ID 定位到指定病程记录

接口:

```
Bool SelectSubDocumentByID(string id)
```

描述:

选择指定编号的病程。

参数:

参数名	是否必填	数据类型	说明
Id	必填	String	字符串类型的病程编号 ID

返回值:

数据类型	说明	备注
Bool	操作是否成功	

代码示例:

```
let res=ctl.SelectSubDocumentByID("sub1");
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。预览模式下无效。

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=SelectSubDocumentByID>

8.7.2. *获取所有病程记录对象

接口:

Array GetCourseRecords(string id)

描述:

选择指定编号的病程或者所有的病程。

参数:

参数名	是否必填	数据类型	说明
Id	非必填	String	字符串类型的病程编号 ID，允许为空 当传病程 ID 时、获取指定 ID 的病程记录，不传参数的时候，获取所有病程对象。

返回值:

数据类型	说明	备注
Array	获取的病程数组。	

代码示例:

<pre>let res=ctl.GetCourseRecords("sub1");//获取ID为sub1的病程记录 let res=ctl.GetCourseRecords();//获取所有病程记录</pre>

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。预览模式下无效。

兼容接口:

兼容四代-跟四代编辑器效果一致。

接口优化:

方法返回的数据不一致。

示例地址:

<http://www.dewriter.cn:8022/apiDocument/edit/index.html?apiname=GetCourseRecords>

8.7.3. *设置当前输入焦点所在的病程记录是否只读

接口：

```
bool SetCurrentSubDocumentReadOnly(bool isReadOnly, string  
backgroundColorValue)
```

描述：

设置当前病程是否只读。

参数：

参数名	是否必填	数据类型	说明
isReadOnly	必填	Bool	是否只读。true 只读、false 不只读
backgroundColorValue	非必填	字符串	病程记录背景颜色。 当参数不存在时，病程只读默认显示灰色

返回值：

数据类型	说明	备注
Bool	操作是否成功。	

代码示例：

```
ctl.SetCurrentSubDocumentReadOnly(true, "#FFFFCD");  
ctl.SetCurrentSubDocumentReadOnly(false);
```

相同效果：

在调用 InsertSubDocuments 或者 AppendSubDocuments 方法时,生成的 options 样式时, 设置属性 Readonly 为 true 即可为只读

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。

兼容接口：

五代独有接口。

接口优化：

无。

示例地址:

<http://www.dcwritr.cn:8022/apiDocument/edit/index.html?apiname=SetCurrentSubDocumentReadOnly>

8.7.4. *获取输入焦点所在病程记录的 ID

接口:

string CurrentSubDoc()

描述:

获取当前病程编号。

参数:

无

返回值:

数据类型	说明	备注
String	当前病程编号。	

代码示例:

```
ctl.CurrentSubDoc();
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。打印预览模式下无效

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=CurrentSubDoc>

8.7.5. *根据病程 ID 删除指定病程记录

接口:

```
bool DeleteSubDoc(string id)
```

描述:

删除指定编号的病程。当执行了用户登录接口，调用该接口，病程记录就是处于逻辑删除的状态，通过方法 GetCourseRecords 获取的所有病程记录集合，需要通过病程记录下的 IsLogicDeleted 属性判断病程记录是否已经被逻辑删除。

参数:

参数名	是否必填	数据类型	说明
Id	必填	String	字符串类型的病程编号 ID，不允许为空

返回值:

数据类型	说明	备注
Bool	操作是否成功。	

代码示例:

```
let res=ctl.DeleteSubDoc("sub1");
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式。打印预览模式下无法删除病程

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=DeleteSubDoc>

8.7.6. *删除输入焦点所在的病程记录

接口:

```
bool DeleteCurrentSubDoc()
```

描述:

删除当前输入焦点所在的病程记录。当执行了用户登录接口,调用该接口,病程记录就是处于逻辑删除的状态,通过方法 GetCourseRecords 获取的所有病程记录集合,需要通过病程记录下的 IsLogicDeleted 属性判断病程记录是否已经被逻辑删除。

参数:

无

返回值:

数据类型	说明	备注
Bool	操作是否成功。	

代码示例:

```
let res=ctl.DeleteCurrentSubDoc();
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口,不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式。打印预览模式下无法删除病程

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=DeleteCurrentSubDoc>

8.7.7. *根据病程 ID 设置病程只读

接口:

```
bool SetSubDocumentReadOnly(string subID, bool isReadOnly, string
backgroundColorValue)
```

描述:

设置指定编号的病程是否只读。

参数:

参数名	是否必填	数据类型	说明
subID	必填	字符串	字符串类型的病程编号
isReadOnly	必填	Bool	是否只读
backgroundColorValue	非必填	字符串	背景颜色名称

返回值:

数据类型	说明	备注
Bool	操作是否成功。	

代码示例:

```
var result=ctl.SetSubDocumentReadOnly("sub1",true,"#FFFFCD");
var result=ctl.SetSubDocumentReadOnly("sub1",false,"black");

//设置病程只读时，同步修改病程里面输入域背景颜色
ctl.SetSubDocumentReadOnly("subdoc1",true,'#E1F5ED');
ctl.DocumentOptions.ViewOptions.ReadOnlyFieldBackColor="#E1F5ED";
ctl.ApplyDocumentOptions();
ctl.RefreshInnerView(true)
```

相同效果:

```
ctl.SetSubDocumentState(true,'#FFFFCD')//根据当前光标所在的病程设置只读和只
读背景色需要配合刷新使用
```

```
ctl.SetCurrentSubDocumentReadOnly(true,"red");//根据当前光标所在的病程设置
只读和只读背景色需要配合刷新使用
```

结合使用:

```
ctl.RefreshInnerView(true)//刷新视图编辑器不会闪动
```

ctl.DocumentOptions.ViewOptions.ReadOnlyFieldBackColor="";同时设置这个文档选项可以把病程中输入域背景颜色同步成设置病程背景色效果一致

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetSubDocumentReadOnly>

8.7.8. *根据病程序号定位视图

接口:

Bool SelectSubDocument(int index)

描述:

定位视图到指定序号的病程。

参数:

参数名	是否必填	数据类型	说明
Index	必填	Int	病程在文档中的序号，从 1 开始计算

返回值:

数据类型	说明	备注
Bool	操作是否成功	

代码示例:

```
let res=ctl.SelectSubDocument(2);//序号从1开始
```

相同效果:

ctl.SelectSubDocumentByID("M5UCYj")//根据病程 ID 进行视图定位

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在除打印预览模式下都能调用。

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SelectSubDocument>

8.7.9. *设置当前病程是否只读并设置颜色。**接口:**

```
bool SetSubDocumentState(bool editable, string strS tyle)
```

描述:

设置当前病程是否只读并设置颜色。

参数:

参数名	是否必填	数据类型	说明
editable	必填	Bool	是否只读
strStyle	必填	字符串	背景颜色名称

返回值:

数据类型	说明	备注
Bool	操作是否成功。	

代码示例:

```
var result=ctl.SetSubDocumentState(true, "#FFFFCD");  
var result=ctl.SetSubDocumentState(false, null);
```

相同效果:

```
ctl.SetSubDocumentReadOnly("sub1", true, "#FFFFCD");//根据病程设 ID 设置只读和  
只读背景色需要配合刷新使用
```

```
ctl.SetCurrentSubDocumentReadOnly(true, "red");//根据当前光标所在的病程设置  
只读和只读背景色需要配合刷新使用
```

```
ctl.RefreshInnerView(true)//刷新视图编辑器不会闪动
```

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetSubDocumentState>

8. 7. 10. *设置指定病程是否跨页显示（不推荐）

接口:

bool SubDocCrossPage(string id, bool isCrossams)

描述:

设置指定编号的病程是否强制换页。

参数:

参数名	是否必填	数据类型	说明
Id	必填	字符串	病程编号
isCrossams	必填	Bool	是否强制换页

返回值:

数据类型	说明	备注
Bool	操作是否成功。	

代码示例:

```
var result=ctl.SubDocCrossPage("sub1", true);
ctl.RefreshInnerView(true)//需要配合刷新使用
```

相同效果:

先获取到指定病程对象,然后通过 SetElementProperties 方法修改病程对象下 NewPage 属性设置强制换页

结合使用:

ctl.RefreshInnerView(true)//需要配合刷新使用

接口类型:

纯前端接口，不经过服务端

有无限制:

无

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SubDocCrossPage>

8.7.11. *将输入焦点定位到指定病程中的输入域中

接口:

bool FocusSubElement(object sub, string id)

描述:

在病程记录中定位指定的输入域，然后进行赋值或者进行签名

参数:

参数名	是否必填	数据类型	说明
sub	必填	Object	病程引用对象,可使用 GetElementByIdExt 的返回值
id	必填	String	输入域编号

返回值:

数据类型	说明	备注
Bool	操作是否成功。	

代码示例:

<pre>var ctl = document.getElementById("myWriterControl"); var sub = ctl.GetElementByIdExt('subdoc2');</pre>

```
ctl.FocusSubElement(sub, "field1");
```

相同效果：

通过 GetAllInputFields 方法在第三个参数传入病程 id 获到指定病程的所有输入域进行循环找到对应的元素进行元素的取值、赋值、插入图片的操作

通过 GetElementByIdExt('输入域 ID','病程 ID')就能获取到病程中指定输入域的唯一值，可以通过这个唯一值进行元素的取值、赋值、插入图片的操作

通过上述另外两个方式都可以先获取到病程中的指定输入域，之后调用其它接口实现不同业务功能

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

无

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=FocusSubElement>

8.7.12. *设置指定病程的边框颜色，线形和边距大小

接口：

```
bool SetSubDocumentBorderByID(string id, JsonElement options, bool isSetNextTop, bool isExcludeOthersBorder)
```

描述：

设置指定病程的边框。第一个参数也可以传入 ctl.CurrentSubDoc() 设置的就是当前病程的边框

参数：

参数名	是否必填	数据类型	说明

Id	必填	String	病程的编号
options	必填	Json	边框的参数是 json 的内容。参数允许为 null，表示不设置。以下是参数的属性
+borderleft	非必填	Bool	是否显示左边框、默认为 false
+bordertop	非必填	Bool	是否显示上边框、默认为 false
+borderright	非必填	Bool	是否显示右边框、默认为 false
+borderbottom	非必填	Bool	是否显示下边框、默认为 false
+borderwidth	非必填	Float	边框线的宽度
+borderspacing	非必填	float	边框线透明度
+paddingleft	非必填	float	左内边距大小
+paddingtop	非必填	float	上内边距大小
+paddingright	非必填	float	右内边距大小
+paddingbottom	非必填	float	下内边距大小
+bordercolor	非必填	字符串	边框线的颜色
+borderstyle	非必填	字符串	边框线的线形，包含 Solid、Dash、Dot、DashDot、DashDotDot、Custom。
isSetNextTop	非必填	Bool	是否设置当前病程下一个病程的上边框颜色。推荐默认设置成 false
isExcludeOthersBorder	非必填	Bool	是否取消其他病程的边框。推荐默认设置成 false
isAllRefresh	非必填	Bool	默认值为 true，强制页面刷新,这个时候页面会出现闪烁。可以设置参数为 false,然后再这个方法后面设置 <code>ctl.RefreshInnerView(true)</code>

返回值:

数据类型	说明	备注
Bool	返回是否执行完成。	

代码示例:

```
var options = {
    borderleft: true,
    bordertop: true,
    borderright: true,
    borderbottom: true,
    borderwidth: 1,
    paddingleft: 10,
    paddingtop: 10,
    paddingright: 10,
    paddingbottom: 10,
    bordercolor: "red",
    borderstyle: "Solid"
};
let result = ctl.SetSubDocumentBorderByID("sub1", options, false, false, false);
ctl.RefreshInnerView(true);
//设置病程ID为Sub1的边框颜色为红色
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载完病程记录之后就能进行调用。

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetSubDocumentBorderByID>

8. 7. 13. *获取的病程编号以数组的形式返回（不推荐使用）

接口：

Array GetAllSubDocumentIDs()

描述：

获取所有病程的编号的数组。然后通过 SaveSubDocumentToString 方法进行批量病程保存。

参数：

无

返回值：

数据类型	说明	备注
Array	获取的病程编号的数组。	

代码示例：

```
var ctl = document.getElementById("myWriterControl");  
let res=ctl.GetAllSubDocumentIDs();
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

无

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=GetAllSubDocumentIDs>

8.7.14. *获取当前光标处病程的基本属性信息

接口：

Json GetCurrentSubDocumentInfo()

描述：

获取当前病程的信息。

参数：

无

返回值：

数据类型	说明	备注
Json	获取当前病程的信息。返回值参考本文的附录：病程	

代码示例：

```
let res=ctl.GetCurrentSubDocumentInfo();
```

相同效果：

通过 `ctl.GetCourseRecords("subdoc5")` 获取出来指定病程的属性信息效果一样，一个是通过获取当前光标处的，一个是获取指定病程的属性信息

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载完病程该接口才能进行调用，其中打印预览模式下通过鼠标给病程文档提供输入焦点该接口就无效。

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=GetCurrentSubDocumentInfo>

8.7.15. *获取所有病程的所有边框和背景颜色信息，并是否清除边框和背景色

接口：

```
Array GetAllSubdocumentsBorderAndBackgroundColor(bool isClear)
```

描述:

获取所有病程的所有边框和背景颜色信息，并是否清除边框和背景色。在书写时需要设置病程的边框和背景色，打印时一般不需要显示。此时可通过 `GetAllSubdocumentsBorderAndBackgroundColor(true)` 清除边框和背景色，并返回所有的边框和背景色信息，在打印完成后，再通过 `SetAllSubdocumentsBorderAndBackgroundColor` 重新设置。

参数:

参数名	是否必填	数据类型	说明
isClear	必填	Bool	在获取所有边框和背景颜色信息后，是否需要清除所有边框和背景颜色信息。

返回值:

以数组的形式返回所有边框和背景颜色信息。单向的信息如下：

数据类型	说明	备注
ID	病程的编号	
Bold	病程的样式的粗体	
LetterSpacing	病程的样式的字符间距	
BorderStyle	边框样式，包含 Solid、Dash、Dot、DashDot、DashDotDot、Custom。	
BorderWidth	边框宽度	
BackgroundColor	背景色	
BorderLeft	是否显示左边框	
BorderLeftColor	左边框颜色	
BorderRight	是否显示右边框	
BorderRightColor	右边框颜色	
BorderTop	是否显示上边框	
BorderTopColor	上边框颜色	
BorderBottom	是否显示下边框	
BorderBottomColor	下边框颜色	

代码示例:

```
let res=ctl.GetAllSubdocumentsBorderAndBackgroundColor(true)
```

相同效果：

无

结合使用：

当需要清除病程样式时，先通过这个方法进行清空然后记录病程边框样式，需要重新设置边框样式根据 GetAllSubdocumentsBorderAndBackgroundColor 方法把一开始存储的样式传入方法中即可

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载完病程文档即可调用。

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://www.dewriter.cn:8765/apiDocument/edit/index.html?apiname=GetAllSubdocumentsBorderAndBackgroundColor>

8.7.16. *设置所有病程的所有边框和背景颜色信息

接口：

Bool GetAllSubdocumentsBorderAndBackgroundColor (json options)

描述：

设置所有病程的所有边框和背景颜色信息。在书写时需要设置病程的边框和背景色，打印时一般不需要显示。此时可通过 GetAllSubdocumentsBorderAndBackgroundColor (true) 清除边框和背景色，并返回所有的边框和背景色信息，在打印完成后，再通过 GetAllSubdocumentsBorderAndBackgroundColor 重新设置。

参数：

参数名	是否必填	数据类型	说明
Options	必填	Json	该接口的参数一般是 GetAllSubdocumentsBorderAndBackgroundColor

			的返回值。

返回值：

数据类型	说明	备注
Bool	操作是否成功	

代码示例：

```
var json = [{
    "ID": "M5UCYj",
    "Bold": false,
    "LetterSpacing": 5,
    "BorderStyle": "Solid",
    "BorderWidth": 10,
    "BackgroundColor": "red",
    "BorderLeft": true,
    "BorderLeftColor": "red",
    "BorderRight": true,
    "BorderRightColor": "red",
    "BorderTop": true,
    "BorderTopColor": "red",
    "BorderBottom": true,
    "BorderBottomColor": "red"
}]

let res=ctl.SetAllSubdocumentsBorderAndBackgroundColor(json)
```

注意：设置病程的边框数据，要和病程的个数数量对应上，以及病程的 ID 对应上。才可设置成想要的病程样式。

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载完病程即可调用

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetAllSubdocumentsBorderAndBackgroundColor>

8. 7. 18. *在病程文档后批量追加病程记录文档

接口：

bool AppendSubDocuments(json options)

描述：

在正文中追加多个病程记录

参数：

参数名	是否必填	数据类型	说明
Options	必填	Json	该接口的参数如下：
+usebase64	非必填	Bool	是否加载 base64 格式的内容
+fileformat	非必填	Bool	需要加载的病程内容的格式
+files	必填	Array	需要加载的病程内容的数组
+Options	必填	Array	病程记录对象的数组信息，单项参数如下：
++id	必填	字符串	病程的编号
++tooltip	非必填	字符串	病程的提示文本
++title	非必填	字符串	病程的标题
++newpage	非必填	Bool	是否强制换页
++readonly	非必填	Bool	是否只读

++enablepermission	非必填	Bool	是否启用权限
++attributes	非必填	Json	自定义属性信息
++parameters	非必填	Json	数据源绑定对象
++databindingxml	非必填	字符串	绑定的 XML 信息
++titlelevel	非必填	Int	标题层级
++loadoriginheaderfooter	非必填	Bool	当前端设置保存病程页眉页脚的时候，将页眉页脚字符串存在病程的自定义属性内
++autofittablewidth	非必填	Float	设置子文档内的表格列宽进行缩放适应主文档宽度
++cssstyle	非必填	Json	自定义的样式。 比如： "border:1px solid yellow;background-color: gray;" --设置病程边框颜色和病程背景颜色

返回值：

数据类型	说明	备注
Bool	操作是否成功	

代码示例：

```
var xml1 = subXml1
file.push(xml1);
var xml2 = subXml2;
file.push(xml2);
var options = new Array();
var attr1 = {
    // "Attributes": { "姓名": "张三", "科室": "胸外科", "床号": "12345" }, // 自定义属性
    "NewPage": false, // 是否新起一页
    "ReadOnly": false, // 是否可编辑
    "Title": "张三的病程记录", // 病程收缩时显示的文档标题
    "ToolTip": "提示：张三的病程记录", // 病程鼠标移上去时的提示文本
}
```

```

        "ID": randomString(6), //
        "BackgroundColorValue": "#F0FFFF",
        "TitleLevel": 0, // 导航级别, 从0到7的整数
        //"CSSStyle": "border: 3px solid #FF0000; background-color: #FFFF00;", // 自定义的样式
        "Parameters": { "emr.DE02.01.052.00": "汪汪汪", "aaa": "王二麻子" }, // 绑定的数据格式
        //"DataBindingXML": " <DataSourceName><Name>张三</Name></DataSourceName><DataSourceName2><Name>李四</Name></DataSourceName2>",
        "EnablePermission": false
    };
    var attr2 = {
        //"Attributes": { "姓名": "李四", "科室": "太平间", "床号": "45678" }, // 自定义属性
        "NewPage": false, // 是否新起一页
        "ReadOnly": false, // 是否可编辑
        "Title": "李四的病程记录", // 文档标题
        "ToolTip": "提示: 李四的病程记录", // 文档标题
        "ID": randomString(6), //
        "BackgroundColorValue": "#C0C0C0",
        "TitleLevel": 0, // 导航级别, 从0到7的整数
        //"CSSStyle": "border: 1px solid #0000FF; background-color: #FFF0F5;", // 自定义的样式
        "Parameters": { "emr.DE02.01.052.00": "喵喵喵", "aaa": "刘三姐" }, // 绑定的数据格式
        //"DataBindingXML": " <DataSourceName><Name>李四</Name></DataSourceName>",
        "EnablePermission": false
    };
    options.push(attr1);
    options.push(attr2);
    var options = {
        Files: file,
        Options: options,
        Usebase64: "false"
    };
    ctl.AppendSubDocuments(options);
    ctl.RefreshInnerView(true);

```

相同效果:

InsertSubDocuments 方法可以根据指定 ID 的病程记录后批量插入病程

结合使用:

追加病程接口需要执行刷新方法才能起效果

调用接口后调用ctl.RefreshInnerView(true)

接口类型:

纯前端接口，不经过服务端

有无限制：

编辑器加载完病程之后，调用该接口就可以进行追加病程, 编辑器只读、阅读模式、打印预览下无效。

前端事件：

调该接口会自动触发前端事件 EventAfterInsertSubDocuments--病程插入成功后事件

兼容接口：

兼容四代接口-五代调用跟四代保持一致

接口优化一讨论：

无。

示例地址：

<https://www.dcwritr.cn:8765/apiDocument/edit/index.html?apiname=AppendSubDocuments>

8. 7. 19. *在指定病程后追加多个病程记录

接口：

bool InsertSubDocuments(json options, object afterElement,bool isafter)

描述：

在指定病程后面追加多个病程记录

参数：

参数名	是否必填	数据类型	说明
Options	必填	Json	该接口的参数如下：
+usebase64	非必填	Bool	是否加载 base64 格式的内容
+fileformat	非必填	Bool	需要加载的病程内容的格式
+files	必填	Array	需要加载的病程内容的数组
+Options	必填	Array	病程记录对象的数组信息，单项参数如下：
++id	必填	字符串	病程的编号

++tooltip	非必填	字符串	病程的提示文本
++title	非必填	字符串	病程的标题
++newpage	非必填	Bool	是否强制换页
++readonly	非必填	Bool	是否只读
++enablepermission	非必填	Bool	是否启用权限
++attributes	非必填	Json	自定义属性信息
++parameters	非必填	Json	传入键值对数据，然后通过 key 值对应病程中输入域的 ID，对应上之后，通过 value 进行赋值
++databindingxml	非必填	字符串	绑定的 XML 信息（病程里面的输入域需要设置数据源信息）
++titlelevel	非必填	Int	标题层级
++loadoriginheaderfooter	非必填	Bool	当前端设置保存病程页眉页脚的时候，将页眉页脚字符串存在病程的自定义属性内
++autofittablewidth	非必填	Float	设置子文档内的表格列宽进行缩放适应主文档宽度
++cssstyle	非必填	Json	自定义的样式
afterElement	非必填	Object	引用对象，为空表示在文档的最后追加，一般可使用 GetElementByIdExt 的返回值
*isafter (2024-09-05 新增参数)	非必填	bool	控制在指定病程前面还是后面进行插入病程，默认值为 true，或者不传第三个参数，在指定病程后面进行插入，该参数设置成 false, 在指定病程前面进行插入

返回值：

数据类型	说明	备注
Bool	操作是否成功	

代码示例:

```
var file = new Array();
file.push(xmlForSubBinding);
file.push(xmlForSubBinding);
file.push(xmlForSubBinding);
file.push(xmlForSubBinding);

//构造两个选项分别对应两个子文档
var options = new Array();
var attr1 = {
    "ID": "subdoc5",//
    "Parameters": { "rdogroup": "V3", "chkgroup": "V2,V3" },//绑定的数据格式
    "DataBindingXML": "<DataSourceName><Name> 王 五 </Name><Sex> 男  
</Sex></DataSourceName>"
    "Title": "首次病程"
};
var attr2 = {
    "ID": "subdoc6",//
    "Parameters": { "rdogroup": "V1", "chkgroup": "V1,V4" }, //绑定的数据格式
    "DataBindingXML": "<DataSourceName><Name> 刘 六 </Name><Sex> 女  
</Sex></DataSourceName>"
};
var attr3 = {
    "ID": "subdoc7",//
    "Parameters": { "rdogroup_2": true, "chkgroup_1": true, "chkgroup_2":  
true } ,//绑定的数据格式
    "DataBindingXML": "<DataSourceName><Name> 李 四 </Name><Sex> 女  
</Sex></DataSourceName>"
};
var attr4 = {
    "ID": "subdoc8",//
    "Parameters": { "rdogroup_0": true, "chkgroup_0": true, "chkgroup_3": true },  
//绑定的数据格式
    "DataBindingXML": "<DataSourceName><Name> 张 三 </Name><Sex> 男  
</Sex></DataSourceName>"
};
options.push(attr1);
options.push(attr2);
options.push(attr3);
options.push(attr4);
var options = {
    Files: file,
    Options: options,
    Usebase64: "false"
```

```
};  
var afterElement = ctl.GetElementByIdExt("subdoc1");  
let result = ctl.InsertSubDocuments(options, afterElement, false); //指定病程前面  
插入新的病程
```

相同效果:

InsertSubDocuentAtCurrentPosition 也是在指定病程处前面或者后面插入病程但是只插入一个空的病程对象,需要再结合接口 LoadSubDocumentFromString 在空病程添加病程内容

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

无

兼容接口:

兼容四代

接口优化:

无。

示例地址:

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=InsertSubDocuments>

8. 7. 20. *在指定病程处插入病程元素

接口:

InsertSubDocuentAtCurrentPosition

描述:

在指定病程处插入病程元素: (通过 insertUp 参数控制向上插入或者向下插入)。不建议使用该方法, 推荐使用 InsertSubDocuments 方法进行病程插入

参数:

参数名	是否必填	数据类型	说明
Options	必填	Json	该接口的参数如下:
++ID	必填	string	病程 id
++title	非必填	string	病程的标题

<code>insertUp</code>	必填	<code>boolean</code>	<code>true</code> :在上面插入; <code>false</code> :在下面插入

返回值：无

代码示例：

```
var ctl = document.getElementById("myWriterControl");
    ctl.InsertSubDocuentAtCurrentPosition({
        ID: ,
        Title: "日常病程记录",
    }, true);
```

相同效果：

无

结合使用：

先通过 InsertSubDocuentAtCurrentPosition 接口插入一个病程再使用 LoadSubDocumentFromString 接口对已有的病程填充 xml 内容

```
var ctl = document.getElementById("myWriterControl");
    ctl.InsertSubDocuentAtCurrentPosition({
        ID: Index.toString(),
        Title: "日常病程记录",
    }, true);
    var options = {
        FileContentXML: SUB9,
        ID: Index.toString(),//已经存在的病程 id
        Usebase64: false,
        ShowMaskUI: true
    };
    let result = ctl.LoadSubDocumentFromString(options);//给已经存在的病程
    填充数据
}
```

接口类型：

纯前端接口，不经过服务端

有无限制：

- 打印预览无法执行
- 阅读模式下无法执行
- 只读模式下无法执行

兼容接口：

五代独立接口

接口优化：

无。

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=InsertSubDocuentAtCurrentPosition>

8.7.21. *给已经存在的病程填充文档

接口:

```
bool LoadSubDocumentFromString(Json options)
```

描述:

填充一个空的病程文档

参数:

参数名	是否必填	数据类型	说明
Options	必填	Json	该接口的参数如下:
+usebase64	非必填	Bool	是否加载 base64 格式的 xml 内容
+filecontentxml	必填	字符串	需要加载的病程 XML 内容
+id	非必填	字符串	病程的编号

返回值: 无

数据类型	说明	备注
Bool	操作是否成功	

代码示例:

```
var options = {
    FileContentXML: xml2,
    ID: '123',
    Usebase64: false,
    ShowMaskUI: true
};
let result = ctl.LoadSubDocumentFromString(options);
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制：

打印预览无法执行

阅读模式下无法执行

兼容接口：

兼容四代

接口优化：

无。

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=LoadSubDocumentFromString>

8.7.22. *保存单个病程的信息**接口：**

Json SaveSubDocumentToString(Json options)

描述：

保存单个病程的信息

参数：

参数名	是否必填	数据类型	说明
Options	必填	Json	该接口的参数如下：
+usebase64	非必填	Bool	是否加载 base64 格式的 xml 内容
+subdocid	必填	字符串	需要保存的病程编号
+subdocids	非必填	数组	多个病程编号["subdoc1", "subdoc2"]
+fileformat	必填	字符串	需要保存的病程内容格式，支持 XML、json、pdf。默认是 XML
+commitusertrace	非必填	Bool	是否提交病程的痕迹信息，提交后逻辑删除的内容将不再显示。默认是 false。
+outputformatxml	非必填	Bool	是否输出格式化的内容。格式化的内容有利于人工阅读，非格式化的内容不利于人工阅读。默认是 true。

+saveheaderfooter	非必填	Bool	是否保存页眉页脚信息。默认是 false。
+LocalMode	非必填	bool	是否纯前端生成 base64 的 pdf 数据。默认值为 false，当为 true 值时，纯前端生成

返回值：

返回保存的病程信息，参数如下

数据类型	说明	备注

代码示例：

```
let id = ctl.CurrentSubDoc();
if (id != null && id != "") {
    var options = {
        FileFormat: 'xml',
        SubDocID: id,
        Usebase64: false,
        CommitUserTrace: true,
        OutputFormatXML: false,
        SaveHeaderFooter: true,
        SaveOriginHeaderFooter: false,
        LogUserEditTrack: false
    };
    contentSaveSubDocumentToString = ctl.SaveSubDocumentToString(options);
}
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

无

兼容接口：

兼容四代

接口优化：

五代保存病程时无法带上页眉页脚数据

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SaveSubDocumentToString>

注意:

当部署编辑器服务端为 java 时，病程保存成 pdf 需要在参数中加上 LocalMode 为 true，且在方法中第二个参数定义回调函数，函数方法中变量就是生成的 base64pdf 数据。

8.7.23. *保存指定病程为 base64 格式数据

接口:

string SaveSubDocumentToBase64String(string fileFormat, string id)

描述:

获取病程文档 BASE64 字符串

参数:

参数名	是否必填	数据类型	说明
fileFormat	必填	字符串	需要保存的病程内容格式，支持 XML、json。默认是 XML
id	必填	字符串	需要保存的病程编号

返回值:

数据类型	说明	备注
String	病程内容的 base64 内容	

代码示例:

```
var ctl = document.getElementById("myWriterControl");
let result = ctl.SaveSubDocumentToBase64String("xml", "subdoc2");
console.log(result)
```

相同效果:

```
var options = {
    FileFormat: "xml",//文档格式
    subDocIDs:"subdoc2",// id 集合的数组字符串
    Usebase64: true,
    CommitUserTrace: true,
    OutputFormatXML: false,
    SaveHeaderFooter: true,
```

```
        SaveOriginHeaderFooter: false,
        LogUserEditTrack: false
    }
    var xml = ctl.SaveSubDocumentToString(options);//与 Usebase64 设置为 true 返回数据一致
```

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

无

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<http://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SaveSubDocumentToBase64String>

8.7.24. *CA 签名(软签)

接口：

```
bool CASignature(JsonElement Options, function back)
```

描述：

CA 签名与四代兼容接口

参数：

参数名	是否必填	数据类型	说明
Options	必填	Json	该接口的参数如下：
+ id	必填	字符串	签名的编号
+ base64img	必填	字符串	签名图片的 base64 数据
+ scope	必填	字符串	签名范围，包含 None、Parent、InputField、TableCell、TableRow、Table、Section、Body、Document。

+ userid	必填	字符串	签名用户的编号
+ username	非必填	字符串	签名用户的名称
+ clientname	非必填	字符串	签名用户的客户端名称
+ role	非必填	字符串	自定义属性信息
+ width	非必填	Float	签名的宽度
+ height	非必填	Float	签名的高度
+ imageinfrontoftext	非必填	字符串	签名的层次，包含 Normal、UnderText、InFrontOfText。
+ providername	非必填	字符串	签名的厂商，默认是 Software。可选择 1 表示 Software、2 表示诺安。
+ bstrpfx	非必填	字符串	自定义属性 PER 对于的值
+ bstrpwd	非必填	字符串	自定义属性 PWD 对于的值
+ resign	非必填	Bool	是否重新签名。重签的话不插入元素，在下面直接找。默认是 false。
+ specifycontainer	非必填	Object	签名的元素，支持元素的 ID、元素的 NativeHandle 对象。
Back	非必填	Function	回调函数

返回值：

数据类型	说明	备注
bool	操作是否成功	

代码示例：

```
var obj = {
  "id": "img1",
  "base64Img": imgaebase64,
  "scope": "InputField",
  "userID": "zhangsan",
  "userName": "zhangsan",
  "imageInFrontOfText": "false",
  "clientName": "(local)",
  "width": "80",
  "height": "80",
  "providerName": "1", //1:软签 2: 诺安签名
  "specifycontainer": input//“医师签名”
```

```
};
var result = ctl.CASignature(obj, function () {
    console.log("签名成功")
    console.log(result, "result");
});
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑模式下才可签名

兼容接口:

非五代独有接口。

接口优化:

无。

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=CASignature>

8.7.25. *根据病程编号（ID）向上移动

接口:

bool SetSubDocumentUp(string id)

描述:

设置指定病程上移、当不传参数时，将输入焦点所在的病程向上移动

参数:

参数名	是否必填	数据类型	说明
Id	必填	String	病程的编号

返回值:

数据类型	说明	备注
Bool	返回是否执行完成。	

代码示例:

```
let result = ctl. SetSubDocumentUp("sub1",);
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

当编辑器加载完病程之后，

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetSubDocumentUp>

8.7.26. *根据病程编号（ID）向下移动

接口：

```
bool SetSubDocumentDown(string id)
```

描述：

设置指定病程下移、当不传参数时，将输入焦点所在的病程向下移动

参数：

参数名	是否必填	数据类型	说明
Id	必填	String	病程的编号

返回值：

数据类型	说明	备注
Bool	返回是否执行完成。	

代码示例：

```
let result = ctl. SetSubDocumentDown("sub1",);
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑模式下才可移动

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetSubDocumentDown>

8.7.27. *获得指定编号的子文档内容 XML 字符串

接口：

String GetSubDocumentContentXmlById(string id)

描述：

获得指定编号的子文档内容 XML 字符串。

参数：

参数名	是否必填	数据类型	说明
Id	必填	String	病程的编号

返回值：

数据类型	说明	备注
String	返回子文档的 XML 内容	

代码示例：

```
let result = rootElement. GetSubDocumentContentXmlById("sub1");
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

无

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetSubDocumentContentXmIByID>

8. 7. 28. *是否显示所有病程的边框

接口：

Bool ShowAllSubDocumentBorder (bool isShow)

描述：

是否显示所有病程的边框。

参数：

参数名	是否必填	数据类型	说明
isShow	必填	Bool	是否显示病程边框

返回值：

数据类型	说明	备注
Bool	操作是否成功	

代码示例：

```
let result = rootElement. ShowAllSubDocumentBorder ( false);
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

- 打印预览无法执行
- 阅读模式下无法执行
- 只读模式下无法执行

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=ShowAllSubDocumentBorder>

8. 7. 29. *修改指定病程记录属性内容值

接口：

Bool SetCourseRecordByID(string subdoceleid, json subdocele, bool isRefresh)

描述：

是否显示所有病程的边框。

参数：

参数名	是否必填	数据类型	说明
subdoceleid	必填	string	需要修改病程属性的病程编号
subdocele	必填	json	病程属性
+Attributes	非必填	json	病程自定义属性
+BackgroundColor	非必填	string	病程背景颜色
+EnableCollapse	非必填	bool	病程是否允许收缩
+NewPage	非必填	bool	病程是否强制换页
+ContentReadonly	非必填	bool	病程是否只读
+Title	非必填	string	病程标题
+ToolTip	非必填	string	标题提示文本

+IsLogicDeleted	非必填	bool	病程是否逻辑删除
+ID	非必填	string	病程编号
+TitleLevel	非必填	string	病程等级
+EnablePermission	非必填	string	病程权限控制
+BorderWidth	非必填	int	病程边框粗细值
+BorderTop	非必填	bool	病程上边框是否显示
+BorderTopColor	非必填	string	病程上边框线颜色
+BorderLeft	非必填	bool	病程左边框是否显示
+BorderLeftColor	非必填	string	病程左边框线颜色
+BorderRight	非必填	bool	病程右边框是否显示
+BorderRightColor	非必填	string	病程右边框线颜色
+BorderBottom	非必填	bool	病程下边框是否显示
+BorderBottomColor	非必填	string	病程下边框线颜色
isRefresh	必填	bool	是否全文刷新

返回值非必填：

数据类型	说明	备注
Bool	操作是否成功	

代码示例：

```
var ctl = document.getElementById("myWriterControl");  
let result = ctl.SetCourseRecordByID('subdoc1', { Title: '新的标题新的标题新的标题' }, true);
```

相同效果：

SetElementProperties 方法效果一致

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

兼容接口：

五代独有接口

接口优化:

无。

示例地址:

<http://117.89.85.111:8086/apiDocument/edit/index.html?apiname=SetCourseRecordByID>

8. 7. 30. *获取指定病程记录中结构化元素列表**接口:**

```
Json GetSubDocumentDomStructure(string subxmlstring, string  
format, string subid)
```

描述:

根据病程编号获取指定病程记录中结构化元素列表。

参数:

参数名	是否必填	数据类型	说明
subxmlstring	必填	string	不传为当前编辑器中加载的病程记录。不传需传 null 参数
format	必填	string	不传为当前病程文件格式。不传需传 null 参数
subid	必填	string	病程编号值

返回值非必填:

数据类型	说明	备注
json	结构化元素列表	

代码示例:

```
var ctl = document.getElementById("myWriterControl");  
let result = ctl.GetSubDocumentDomStructure(null, null, 'subdoc1');
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

无

兼容接口:

五代独有接口

接口优化:

无。

示例地址:

<http://117.89.85.111:8086/apiDocument/edit/index.html?apiname=GetSubDocumentDomStructure>

8. 7. 31. *获取指定病程记录中指定元素编号文本内容

接口:

string GetElementTextByIdInSub(string id, json options, string subid)

描述:

通过病程编号和病程里面元素编号通过条件获取元素文本内容。

参数:

数据类型	是否必填	说明	备注
id	必填	元素编号	病程中元素编号
options	非必填		通过条件输出内容
+IncludeBackgroundText	非必填	Bool	表示是否包含背景文字。 默认值为 false
+IncludeBorderText	非必填	Bool	是否保存边框文字 默认值为 false
+IncludeHiddenText	非必填	Bool	是否包含隐藏的文本 默认值为 false
+IncludeLabelUnitText	非必填	Bool	是否包含标签单位文本 默认值为 true
+IncludeLogicDeletedContent	非必填	Bool	是否包含逻辑删除内容 默认值为 false
subid	必填	string	病程编号

返回值:

数据类型	说明	备注
string		元素内容文本

代码示例:

```
var ctl = document.getElementById("myWriterControl");
ctl.GetElementTextByIdInSub("field1", {IncludeBackgroundText:false} , "
subdoc1");
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口

有无限制:

无

兼容接口:

不兼容四代

接口优化:

无。

测试地址:

无

8. 7. 32. *设置指定病程记录中指定元素编号文本内容

接口:

```
string SetElementTextByIdInSub(string id,string text,string
subid,bool withTrackInfos)
```

描述:

通过病程编号和病程里面元素编号设置文本内容并且通过条件判断是否留痕。

参数:

数据类型	是否必填	类型	备注
id	必填	string	病程中元素编号
text	必填	string	赋值文本
subid	必填	string	病程编号
withTrackInfos	必填	Bool	是否产生痕迹

返回值:

数据类型	说明	备注
bool		true: 表示接口调用成功 false:接口调用失败

代码示例:

```
var ctl = document.getElementById("myWriterControl");
ctl.SetElementTextByIdInSub("field1","张三", "subdoc1", true); //赋值并且留痕
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口

有无限制:

无

兼容接口:

不兼容四代

接口优化:

无。

测试地址:

无

8.8. 数据源绑定

为了批量修改文档中结构化元素的内容，引入了数据源绑定的功能。

在数据源绑定时，元素的属性 ValueBinding 标记了数据源绑定的信息，它包含的主要属性如下：

- 1. DataSource 标记了绑定的对象，一般理解为数据对象的实例名或别名。
- 2. BindingPath 标记了绑定的路径信息，该路径一般可解释成对象的属性名。

在使用数据源绑定时需要注意：

- 1. 元素存在嵌套时，不允许同时使用数据源绑定；
- 2. 数据源绑定是通过 DOM 修改 XML 节点，不会产生留痕记录而且会删除掉元素之前的痕迹；
- 3. 由于数据源绑定一般是在文档加载之后进行的，此时文档的只读或元素只读不影响数据源绑定的执行；
- 4. 如果有两个以上的元素的数据源绑定属性信息完全一致，通过数据源绑定取值时，后一个元素的值会覆盖前一个元素的值，即只能取到最后一个元素的值。
- 5. 如果用户调用接口给的参数是 null，则表示不进行赋值操作，保持原文档的数据不变。如果需要赋值空数据，需要使用空字符串。

8.8.1. *获取文档中的数据源绑定属性 DataSource 为指定的文本的信息

接口：

Json GetDocumentParameterValue(string name)

描述：

获取文档中的数据源绑定属性 DataSource 为指定的文本的信息。在执行该接口前需要先调用 WriteDataFromDocumentToDataSource。

参数：

参数名	是否必填	数据类型	说明
Name	必填	String	表示文档中的数据源绑定属性 DataSource 为指定的文本的信息。

返回值：

数据类型	说明	备注
Json	返回数据源绑定属性 DataSource 对应的 BindingPath 的结构。返回的结构根据数据绑定的信息不同结果也不同。 如返回 <pre>▼ {Name: '', Age: '', Address.Street: '', Sex: '', 过敏: '', ...} Address.Street: '' Age: '' ICD_Code: null ICD_Name: '' Name: '' Sex: '' 医保: '' 过敏: ''</pre> 表示该文档中所有的结构化元素的属性 ValueBinding 的子属性标记为 DataSource 相同的 BindingPath 的值。	

代码示例:

```
ctl. WriteDataFromDocumentToDataSource();
let res=ctl.GetDocumentParameterValue("EMR_Patients")
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

无

兼容接口:

兼容四代接口

接口优化:

无。

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetDocumentParameterValue>

8.8.2. *设置文档中的数据源绑定属性 DataSource 为指定的文本的元素
的文本

接口:

Bool SetDocumentParameterValue(string name, object Value)

描述:

设置文档中的数据源绑定属性 DataSource 为指定的文本的元素的文本。在执行该接口前后需要再次调用 WriteDataFromDataSourceToDocument。

参数:

参数名	是否必填	数据类型	说明
Name	必填	String	表示文档中的数据源绑定属性 DataSource 为指定的文本的信息。
Value	必填	Json	需要绑定的对象数据信息，该数据的信息一般和 GetDocumentParameterValue 的返回信息相同。

返回值:

数据类型	说明	备注
Bool	操作是否成功。	

代码示例:

```
var values = {
    Name: "张三",
    //病人id
    BirthPlace: "112233",
    sex: "女",
    Job: "护士",
    Age: "33",
    // 住院次数
    indate: "2019-3-27",
    //门诊号
    mingzu: "汉族",
    //一卡通号
    recorddate: "2019-3-28",
    //住院号
    marriage: "3333",
    //住院病区号（病室）
    patientname: "004",
```

```
}  
ctl.SetDocumentParameterValue("EMR_Patients", values);//patInfo EMR_Patients  
var result =ctl.WriteDataFromDataSourceToDocument();
```

相同效果:

效果跟 SetElementTextById 一致。编辑器的另外一种赋值方式。

结合使用:

```
var result = ctl.SetDocumentParameterValue("cornealTopography", values);  
ctl.WriteDataFromDataSourceToDocument();
```

调用完方法一定得调用-WriteDataFromDataSourceToDocument 把数据源中的内容回填到文档中

接口类型:

纯前端接口，不经过服务端

有无限制:

无

兼容接口:

兼容四代接口

接口优化:

无。

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetDocumentParameterValue>

注意:

当需要对输入域赋空、单、复选框取消勾选，前端拼接数据时，对应绑定路径值设置成空字符串即可，当需要使用当前元素内容，不进行覆盖，前端拼接数据对应绑定路径值设置成 null 即可。

8.8.3. *根据数据源名称来绑定对应的数据

接口:

```
Int WriteDataFromDataSourceToDocumentSpecifyParameterNames (string  
name)
```

描述:

将指定名称的文档参数值填充到文档中。指定的文档参数名称，各个名称之间用英文逗号分开。比如“姓名，性别，国籍”，如果为空则更新全部数据源。

参数：

参数名	是否必填	数据类型	说明
Name	必填	String	表示文档中的数据源绑定属性 DataSource 为指定的文本的信息。

返回值：

数据类型	说明	备注
Int	绑定成功的个数，为 0 的情况就是都没有绑定成功。	

代码示例：

```
ctl.WriteDataFromDataSourceToDocumentSpecifyParameterNames("姓名");
```

注意：

当文书中存在多个数据源信息，但编辑器初始化加载文档且只绑定一个数据源的时候，根据业务需求需要绑定另外数据源的时候，SetDocumentParameterValue 方法执行完毕之后不调用 WriteDataFromDataSourceToDocument 方法把绑定的数据附件到文档中，否则会影响第一次绑定的内容，需要调用 WriteDataFromDataSourceToDocumentSpecifyParameterNames 方法只绑定特定的数据源，就不会互相影响。

相同效果：

```
ctl.SetDocumentParameterValue("datasourceforbarcode", "555555555");  
ctl.WriteDataFromDataSourceToDocument();
```

结合使用：

```
先执行 ctl.SetDocumentParameterValue("姓名", "张三");
```

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑模式下才可使用

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=WriteDataFromDataSource>

ToDocumentSpecifyParameterNames

8.8.4. *判断指定的数据源是否设置了不允许绑定

接口:

bool GetDocumentParameterEnabled(string parameterName)

描述:

获取指定的数据源绑定参数是否有效。

参数:

参数名	是否必填	数据类型	说明
parameterName	必填	String	数据源绑定信息

返回值:

数据类型	说明	备注
Bool	参数是否有效	

代码示例:

```
var result = ctl.GetDocumentParameterEnabled("name");
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑模式下才可使用

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetDocumentParameterEnabled>

8.8.5. 获取某个数据源绑定名称的文本信息

接口:

```
String GetDocumentParameterValueXml(string name)
```

描述:

获取文档中的数据源绑定属性 DataSource 为指定的文本的信息。在执行该接口前需要先调用 WriteDataFromDocumentToDataSource。返回的参数是 XML 格式的字符串。该接口和 GetDocumentParameterValue 使用方法相同，只是返回的参数有区别。

参数:

参数名	是否必填	数据类型	说明
Name	必填	String	表示文档中的数据源绑定属性 DataSource 为指定的文本的信息。

返回值:

数据类型	说明	备注
String	返回数据源绑定属性 DataSource 对应的 BindingPath 的结构。返回的结构根据数据绑定的信息不同结果也不同。表示该文档中所有的结构化元素的属性 ValueBinding 的子属性标记为 DataSource 相同的 BindingPath 的值。	

代码示例:

```
let res=ctl.GetDocumentParameterValueXml("EMR_Patients")
```

相同效果:

```
ctl. WriteDataFromDocumentToDataSource();
```

```
let res=ctl.GetDocumentParameterValue("EMR_Patients")
```

返回的数据格式不同。对应的值是一样的。

结合使用:

```
先执行: ctl.WriteDataFromDocumentToDataSource()
```

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑模式下才可使用

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<https://www.dcwritr.cn:8765/apiDocument/edit/index.html?apiname=GetDocumentParameterValueXml>

8.8.6. 设置指定的数据源是否可以进行绑定操作

接口:

```
void SetDocumentParameterEnabled(string parameterName, bool enabled)
```

描述:

设置指定的数据源绑定的参数是否有效。当进行数据源绑定时，文档中存在多个数据源绑定，为了使部分数据源绑定不参与数据更新，可以设置指定的数据源无效。

参数:

参数名	是否必填	数据类型	说明
parameterName	必填	String	数据源绑定参数名称
enabled	必填	Bool	是否有效

返回值:

数据类型	说明	备注
undefined	undefined	

代码示例:

```
ctl.SetDocumentParameterEnabled("xxx", false); //对文档中设置了数据源值为xxx的输入域不允许数据源绑定
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑模式下才可使用

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SetDocumentParameterEnabled>

8.8.7. 通过生成 Xml 格式的数据进行数据源绑定

接口：

```
void SetDocumentParameterValueXml(string name, string xmlText)
```

描述：

通过 XML 格式的数据，来设置文档中的数据源绑定属性 DataSource 为指定的文本的元素的文本。在执行该接口前后需要再次调用 WriteDataFromDataSourceToDocument。

参数：

参数名	是否必填	数据类型	说明
Name	必填	String	表示文档中的数据源绑定属性 DataSource 为指定的文本的信息。
xmlText	必填	String	字符串类型的 XML 格式的数据。

返回值：

数据类型	说明	备注
Number	绑定成功的值	

代码示例：

```
var xml = "<xx><befK10d>41</befK10d><befK20d>43</befK20d></xx>";
var result=ctl.SetDocumentParameterValueXml("cornealTopography", xml);
var result = ctl.WriteDataFromDataSourceToDocument();
```

相同效果：

无

结合使用：

需要配合 WriteDataFromDataSourceToDocument 将数据传入到页面

接口类型:

纯前端接口

有无限制:

- 打印预览无法执行
- 阅读模式下无法执行
- 只读模式下无法执行

兼容接口:

五代独有接口

接口优化:

无。

测试地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetDocumentParameterValueXml>

8.8.8. *横向扩展列数据的表格数据源绑定

接口:

Bool BindingColumnExpandingTables(string bindname, array datas,bool newpage)

描述:

横向扩展列数据的表格数据源绑定客户定制专属接口。也就是对纵向表格进行数据源绑定

参数:

参数名	是否必填	数据类型	说明
bindname	必填	String	绑定的数据源名称,模板中必须有一个表格的数据源名称被设置成这个。
datas	必填	Array	需要绑定的数据信息。数据集传数组, 格式如下, 模板只需制作一个表格, 后台会根据数组大小自动扩展表格 * [

			<pre>* { * AAA: [* { * CCC: "YYY", * DDD: "ZZZ" * } *] * BBB: "XXX" * } *]</pre> * AAA: 表格绑定起始列设置的数据源绑定路径 * BBB: 表格绑定起始列之前的列中若有需要单独绑值的单元格设置的绑定路径 * CCC、DDD: 表格绑定起始列中单元格设置的绑定路径
newpage	非必填	bool	当值为 true 值时，绑定多页内容时，会另起新页创建相同的表格绑定内容。 当值为 false 时，绑定多页内容时，会在当前页创建相同的表格绑定内容，当当前页显示不下时，自动换页展示。

返回值:

数据类型	说明	备注
Bool	操作是否成功。	

代码示例:

<pre>var bindingdata = { StringTime: "2023-7-15", LIST: [{ REG_TIME_TYPE: null, REG_VALUE: "n2的值1", INSULIN: "n3的值1", SIGNATURE_NAME: "n4的值1"</pre>

```

        },
        {
            //REG_TIME_TYPE: "n1的值1",
            REG_VALUE: "n2的值1",
            INSULIN: "n3的值1",
            SIGNATURE_NAME: "n4的值1"
        },
        {
            REG_TIME_TYPE: "n1的值1",
            REG_VALUE: "n2的值1",
            INSULIN: "n3的值1",
            SIGNATURE_NAME: "n4的值1"
        },
        {
            REG_TIME_TYPE: "n1的值1",
            REG_VALUE: "n2的值1",
            INSULIN: "n3的值1",
            SIGNATURE_NAME: "n4的值1"
        }
    ]
};
var arr = new Array();
arr.push(bindingdata);
arr.push(bindingdata);
arr.push(bindingdata);
arr.push(bindingdata);
arr.push(bindingdata);
arr.push(bindingdata);
arr.push(bindingdata);
processArr(arr);
myWriterControl.BindingColumnExpandingTables("TDATA", arr);
}

function processArr(arr) {
    if (Array.isArray(arr) === false) {
        return;
    }
    for (var i = 0; i < arr.length; i++) {
        var item = arr[i];
        if (typeof (item) !== "object") {
            continue;
        }
        for (var key in item) {
            if (Array.isArray(item[key]) === true) {
                processArr(item[key]);
            }
        }
    }
}

```

```
        } else if (item[key] == null) {  
            item[key] = " ";  
        }  
    }  
}
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口

有无限制：

无

兼容接口：

五代独有接口

接口优化：

无。

测试地址：

http://117.89.85.111:8086/pingcode/DUWRITER5_0-3085.html

注意：

需要实现纵向表格数据源绑定时，在制作模板中，表格需要设置数据源名称、第一数据列（开始绑定内容的第一列）设置绑定路径对应前端拼接的 json 对象对应的 key 值，表格列中单元格设置绑定路径。

当纵向表格绑定成功时，本质是会新建页中创建上面也中相同的表格来进行数据赋值，所以不同页中表格属性对话框中数据源名称不同是正常现场。

8.8.9. 获取文档中的所有数据源值

接口：

```
String GetBindingDataSources ()
```

描述：

获得文档中绑定的数据源名称字符串列表。各个名称之间用逗号分开

参数:

无

返回值:

数据类型	说明	备注
String	数据源名称字符串列表。各个名称之间用逗号分开	

代码示例:

```
Var res=ctl. GetBindingDataSources ();
```

相同效果:

通过获取元素属性的的方法 `ctl.GetElementProperties(元素 id)`也可以找对对应的数据源名称

结合使用:

无

接口类型:

纯前端接口

有无限制:

无

兼容接口:

五代独有接口

接口优化:

无。

测试地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetBindingDataSources>

8.8.10. 以 json 的方式返回文档的数据源绑定的信息

接口:

Json GetDataSourceBindingDescriptions()

描述:

以 json 的方式返回文档的数据源绑定的信息。

参数:

无

返回值:

数据类型	说明	备注
Json	以 json 的方式返回文档的数据源绑定的信息	

代码示例:

```
Var res=ctl.GetDataSourceBindingDescriptions ();
```

相同效果:

ctl.GetDataSourceBindingDescriptionsJSON() 也会返回 json 数据

结合使用:

无

接口类型:

纯前端接口

有无限制:

无

兼容接口:

五代独有接口

接口优化:

无。

测试地址:

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetDataSourceBindingDescriptions>

8.8.11. 以 xml 的方式返回文档的数据源绑定的信息

接口:

String GetDataSourceBindingDescriptionsXML ()

描述:

以 xml 的方式返回文档的数据源绑定的信息。

参数:

无

返回值:

数据类型	说明	备注
String	以 xml 的方式返回文档的数据源绑定的信息	

代码示例:

```
Var res=ctl. GetDataSourceBindingDescriptionsXML ();
```

相同效果:

ctl.GetDataSourceBindingDescriptionsJSON() //返回 json 格式的数据源绑定信息

ctl.GetDataSourceBindingDescriptions() //返回 json 格式的数据源绑定信息

结合使用:

无

接口类型:

纯前端接口

有无限制:

无

兼容接口:

五代独有接口

接口优化:

无。

测试地址:

<https://www.dcwritr.cn:8765/apiDocument/edit/index.html?apiname=GetDataSourceBindingDescriptionsXML>

8.8.12. 检测数据源填充操作导致的修改文档元素的相关信息，但不真正填充数据源，不会修改文档内容。

接口:

Json GetDetectValueBindingModifiedMessage ()

描述:

检测数据源填充操作导致的修改文档元素的相关信息，但不真正填充数据源，不会修改文档内容。

参数:

无

返回值:

数据类型	说明	备注
Json	返回被修改的数据源绑定的对象信息	

代码示例:

```
Var res=ctl. GetDetectValueBindingModifiedMessage ();
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口

有无限制:

无

兼容接口:

五代独有接口

接口优化:

无。

测试地址:

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetDetectValueBindingModifiedMessage>

8.8.13. 将文档中的数据写入到数据源中

接口:

Bool UpdateDataSourceForView ()

描述:

将文档中的数据写入到数据源中。建议使用 WriteDataFromDataSourceToDocument。

参数:

无

返回值:

数据类型	说明	备注
无返回值	无返回值	

代码示例:

```
ctl.UpdateDataSourceForView();
```

相同效果:

```
var values = {
    Name: "张三",
    BirthPlace: "112233",
    patientname: "004",
}
ctl.SetDocumentParameterValue("EMR_Patients", values);
ctl.WriteDataFromDataSourceToDocument();
```

结合使用:

无

接口类型:

纯前端接口

有无限制:

无

兼容接口:

五代独有接口

接口优化:

无。

测试地址:

<https://www.dcwritr.cn:8765/apiDocument/edit/index.html?apiname=UpdateDataSourceForView>

8.8.14. 获取文档中的数据源绑定信息

接口:

```
Json UpdateViewForDataSource()
```

描述:

获取文档中的数据源绑定信息。建议使用 WriteDataFromDocumentToDataSource。

参数:

无

返回值:

数据类型	说明	备注
无返回值	无返回值	

代码示例:

```
ctl.UpdateViewForDataSource()
```

相同效果:

```
ctl.WriteDataFromDocumentToDataSource();
var data = ctl.GetDocumentParameterValue("EMR_Patients");//T_DA_0001
if (data != null) {
    var json = JSON.stringify(data);
}
```

结合使用:

无

接口类型:

纯前端接口

有无限制:

无

兼容接口:

五代独有接口

接口优化:

无。

测试地址:

<https://www.dcwritr.cn:8765/apiDocument/edit/index.html?apiname=UpdateViewForDataSource>

8.8.15. 将数据源中的数据写入到文档中

接口:

```
Bool WriteDataFromDataSourceToDocument ()
```

描述:

将数据源中的数据写入到文档中

参数:

无

返回值:

数据类型	说明	备注
Bool	操作是否成功	

代码示例:

```
var values = {
    Name: "张三",
    BirthPlace: "112233",
    patientname: "004",
}
ctl.SetDocumentParameterValue("EMR_Patients", values);
ctl.WriteDataFromDataSourceToDocument();
```

相同效果:

通过 SetElementTextById 方法也可以对元素赋值

结合使用:

无

接口类型:

纯前端接口

有无限制:

只有编辑状态下才会执行

兼容接口:

兼容四代

接口优化:

无。

测试地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=WriteDataFromDataSourceToDocument>

8.8.16. 获取文档中的数据源绑定信息

接口:

Json WriteDataFromDocumentToDataSource ()

描述:

获取文档中的数据源绑定信息。

参数:

无

返回值:

数据类型	说明	备注
Json	文档中的数据源绑定信息	

代码示例:

```
ctl.WriteDataFromDocumentToDataSource();
var data = ctl.GetDocumentParameterValue("EMR_Patients");//T_DA_0001
if (data != null) {
    var json = JSON.stringify(data);
}
```

相同效果:

无

结合使用:

需要先通过方法 WriteDataFromDocumentToDataSource 把文档中的数据返回到数据源中，在进行获取

接口类型:

纯前端接口

有无限制:

无

兼容接口:

兼容四代

接口优化:

无

测试地址:

[https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetDocumentParamet
erValue](https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetDocumentParameterValue)

8.8.17. 清空所有绑定数据源的输入域的值和文本、单选框的选择状态、复选框的选择状态

接口:

```
Void ClearAllParameterValue ()
```

描述:

清空所有绑定数据源的输入域的值和文本、单选框的选择状态、复选框的选择状态。

参数:

无

返回值:

无

代码示例:

```
ctl. ClearAllParameterValue ();
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口

有无限制:

只有编辑状态下才会执行

兼容接口:

兼容四代

接口优化:

无。

测试地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=ClearAllParameterValue>

8.8.18. 以 json 的方式返回文档的数据源绑定的信息

接口:

Json GetDataSourceBindingDescriptionsJSON()

描述:

以 json 的方式返回文档的数据源绑定的信息。

参数:

无

返回值:

数据类型	说明	备注
Json	以 json 的方式返回文档的数据源绑定的信息	

代码示例:

```
Var res=ctl. GetDataSourceBindingDescriptionsJSON ();
```

相同效果:

通过 GetDataSourceBindingDescriptions 方法也可以获取到 json 数据

结合使用:

无

接口类型:

纯前端接口

有无限制:

无

兼容接口:

兼容四代

接口优化:

无。

测试地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetDataSourceBindingDescriptionsJSON>

8.8.19. *新方式对结构化元素批量赋值

接口:

Json setDataWithDataSources()

描述:

以 json 的方式一次性绑定输入域、单、复选框、条形码、二维码、表格等结构化元素内容。

参数:

无

返回值:

数据类型	说明	备注
Json		

代码示例:

```
var ctl = document.getElementById("myWriterControl");
var values = {
    PatientName: "张入梅",
    Gender: "女",
    Age: "23",
    Ward: "内科",
    BedNO: "0001",
    EmpiID: "2024080923341",
    EhealthCardNO: "3242234",
    DischargeDiag: "无大碍，在家观察三天，如有不适在进行就医",
    DischargeDateTime: "2024-09-04 13:34:23",
    SpecialityGuide: "望闻问切都进行观察，没有问题，只是身体有点虚，回家炖老母鸡吃吃",
    Remark: "多补补",
    EducationalNurse: "小美",
    EducationalDatetime: "2024-09-04 13:34:23"
};
var values1 = {
    Badl: "完全自理",
    DischargeStatus: "未愈",
    DischargeMethod: "非医嘱离院",
    UsageGuide: "餐前服药",
    DietStatus: "较差",
    DietGuide: "软食",
    LivingGuides: "戒烟戒酒",
```

```
EducationalContents: "记住管床医生、护士",
SafetyNotices: "绝对卧床",
Time: "出院半年",
Place: "门诊",
Process: "预约挂号"
};
var dataobj = {
    "emr": values,
    "Patient": values1
};
ctl.setDataWithDataSources(null, dataobj);
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口

有无限制:

无

兼容接口:

不兼容四代

接口优化:

无。

测试地址:

<https://www.dcwritr.cn:8765/apiDocument/edit/index.html?apiname=index&pageUrl=newDataSourceBinding.html>

8.8.20. *新方式返回数据源绑定的结构化元素值

接口:

```
Json getDataWithDataSources(string parnetid, string datasourcename)
```

描述:

以 json 的方式返回指定数据源名称对应的绑定内容。

参数:

无

返回值:

数据类型	说明	备注
parnetid	父容器编号	获取指定容器元素里面数据源名称绑定的内容，为空时，返回全文
datasourcename	数据源名称	

代码示例:

```
var ctl = document.getElementById("myWriterControl");
ctl.setDataWithDataSources(null, "emr");
```

相同效果:

与方法 GetDocumentParameterValue 效应功能一致

结合使用:

无

接口类型:

纯前端接口

有无限制:

无

兼容接口:

不兼容四代

接口优化:

无。

测试地址:

无

8.9. 结构化元素

8.9.1. 删除指定元素

接口:

Bool DeleteElement(object id)

描述:

删除指定编号的元素。在参数是元素 ID 时, 和 DeleteElementByID 功能相同。

参数:

参数名	是否必填	数据类型	说明
Id	必填	Object	参数可以是字符串类型的元素编号, 也可以是编辑器特有的唯一对象 NativeHandle 对象。或者是当前元素对象

返回值:

数据类型	说明	备注
false	删除是否成功(目前返回都是 false)	

代码示例:

```
let res=ctl.DeleteElement (strID);  
或  
var cc=ctl.GetElementProperties("txtAge")  
let bb = cc.NativeHandle  
let res=ctl.DeleteElementByID (bb);  
或  
ctl.DeleteElement(ctl.CurrentElement())
```

相同效果:

```
var result = ctl.DeleteElementByID("sdad2");
```

结合使用:

无

接口类型:

纯前端接口

有无限制:

编辑模式下才可使用

兼容接口:

五代独有接口

接口优化:

无。

测试地址:

<http://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=DeleteElement>

8.9.2. 删除指定输入域

接口：

Bool DeleteElementByID (object id)

描述：

删除指定编号的元素。

参数：

参数名	是否必填	数据类型	说明
Id	必填	Object	参数可以是字符串类型的元素编号。也可以根据元素的唯一对象 NativeHandle 对象。

返回值：

数据类型	说明	备注
Bool	删除是否成功	

代码示例：

```
let res=ctl.DeleteElementByID(strID);
或
var cc=ctl.GetElementProperties("txtAge")
let bb = cc.NativeHandle
let res=ctl.DeleteElementByID(bb);
或
document.WriterControl.DeleteElementByID(document.WriterControl.CurrentElement(
))需待确定
```

相同效果：

```
var result = ctl.DeleteElement("sdad2");
```

结合使用：

无

接口类型：

纯前端接口

有无限制：

无

兼容接口：

五代独有接口

接口优化：

无。

测试地址:

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname>DeleteElementById>

8.9.3. 获取指定编号的元素的指定名称的自定义属性的值

接口:

String GetElementAttribute (string id, string attributeName)

描述:

获取指定编号的元素的指定名称的自定义属性的值。

参数:

参数名	是否必填	数据类型	说明
Id	必填	字符串	指定的元素编号。
attributeName	必填	字符串	指定的自定义属性名称

返回值:

数据类型	说明	备注
String	指定的自定义属性的值	

代码示例:

```
let res=ctl.GetElementAttribute(strID, strName);
```

相同效果:

var result = ctl.GetElementCustomAttributes("txtAge");//根据 id 获取元素的自定义属性列表

结合使用:

无

接口类型:

纯前端接口

有无限制:

无

兼容接口:

非五代独有接口

接口优化:

无。

测试地址：

<https://www.dcwritr.cn:8765/apiDocument/edit/index.html?apiname=GetElementAttribute>

8.9.4. 设置指定编号的元素的指定名称的自定义属性的值

接口：

```
Bool SetElementAttribute(string id, string attributeName, string
attributeValue)
```

描述：

设置指定编号的元素的指定名称的自定义属性的值。

参数：

参数名	是否必填	数据类型	说明
Id	必填	字符串	指定的元素编号。
attributeName	必填	字符串	指定的自定义属性名称
attributeValue	必填	字符串	指定的自定义属性的值

返回值：

数据类型	说明	备注
Bool	设置是否成功	

代码示例：

```
let res=ctl.SetElementAttribute (strID, strName);
```

相同效果：

```
var opts = {
    aa: "DDDDDD"
};

var result = myWriterControl.SetElementCustomAttributes("ID",opts) //根据设置元素的
自定义属性列表
```

结合使用：

无

接口类型：

纯前端接口

有无限制：

无

兼容接口:

非五代独有接口

接口优化:

无。

测试地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SetElementAttribute>

8.9.5. 获取当前输入域对象

接口:

Object CurrentInputField ()

描述:

获取当前输入域对象。

参数:

无

返回值:

数据类型	说明	备注
Object	返回当前输入域的引用对象	

代码示例:

```
let res=ctl.CurrentInputField()
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口

有无限制:

无

兼容接口:

非五代独有接口

接口优化:

无。

测试地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=CurrentInputField>

8.9.6. 在指定的容器下查找所有的输入域

接口:

Json GetAllInputFieles (bool excludeReadonly, bool
excludeHiddenElement, object specifyRootElement, bool nestNode)

描述:

在指定的容器下查找所有的输入域。

参数:

参数名	是否必填	数 据 类 型	说明
excludeReadonly	必填	Bool	是否包含只读的输入域。
excludeHiddenElement	必填	Bool	是否包含隐藏的输入域
specifyRootElement	必填	字符串	指定的容器的引用对象, 如果为 null 则 表示全文
nestNode	必填	Bool	兼容参数, 不做任何意义

返回值:

数据类型	说明	备注
数组	返回查找到的所有输入域数组	

代码示例:

<pre>let res=ctl. GetAllInputFieles (true, true, null, true);</pre>

相同效果:

无

结合使用:

无

接口类型:

纯前端接口

有无限制:

无

兼容接口:

兼容四代

接口优化:

无。

测试地址: 暂无

8.9.7. 获取指定元素的文本内容

接口:

String GetElementTextByID(object id, json parameter)

描述:

获取指定元素的文本内容

参数:

参数名	是否必填	数据类型	说明
Id	必填	Object	它可以是三种类型参数: 字符串类型的元素 ID、NativeHandle 对象、引用对象。
parameter	非必填	Json	必须得参数, 以下是参数的属性
+ IncludeBackgroundText	非必填	Bool	表示是否包含背景文字 默认值为 false
+ IncludeBorderText	非必填	Bool	是否保存边框文字 默认值为 false
+ IncludeHiddenText	非必填	Bool	是否包含隐藏的文本 默认值为 false
+ IncludeLabelUnitText	非必填	Bool	是否包含标签单位文本 默认值为 true
+	非必填	Bool	是否包含逻辑删除内容

IncludeLogicDeletedContent			默认值为 false

返回值：

数据类型	说明	备注
String	返回输入域内容	当处于特殊模式，返回值为 false

代码示例：

```
let res=ctl. GetElementTextByID ( “field1”, { IncludeBackgroundText: false, IncludeBorderText: false, IncludeHiddenText: false, IncludeLabelUnitText: false, IncludeLogicDeletedContent: false });
```

相同效果：

GetElementTextByIDPro 方法，传元素 ID 只获取到输入域的 Text 值，不带标签文本、边框、背景文本、单位文本（需要通过添加参数控制）。且当输入域隐藏，默认返回空值。

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只有编辑器处于可编辑模板下才能获取到内容。

兼容接口：

兼容四代-跟四代编辑器效果一致。

接口优化：

五代方法返回的数据跟四代（只返回 Text 值）不一致。

示例地址：

<http://www.dcwritter.cn:8022/apiDocument/edit/index.html?apiname=GetElementTextByID>

8.9.8. 设置指定编号元素的文本

接口：

```
Bool SetElementTextByID(object id, string newText, bool
isAddTraces, bool isFocusContainer)
```

描述：

设置指定编号输入域的文本。（插入的内容能被撤销掉）

参数：

参数名	是否必填	数据类型	说明
Id	必填	Object	它可以是三种类型参数：字符串类型的元素 ID、NativeHandle 对象、引用对象。
newText	必填	String	需要设置的文本
isAddTraces	非必填	bool	赋值时是否产生痕迹，默认值为 false, 不产生痕迹
isFocusContainer	非必填	bool	赋值时输入焦点是否存在当前元素，默认为 true, 进行聚焦，当第四个参数不传时也会进行聚焦，false 表示不聚焦

返回值：

数据类型	说明	备注
Bool	设置是否成功	

代码示例：

```
let res=ctl.SetElementTextByID("field1","newtext");
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

只有编辑模式下才能执行方法。但是只能对容器元素赋值，比如输入域元素、单元格元

素。

兼容接口：

兼容四代-跟四代编辑器效果一致。

接口优化：

无

示例地址：

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetElementTextById>

注意：

需要清空输入域里面内容的时候，当输入域里面出现图片时，第二个参数一定得传 null 才行

8.9.9. 获取指定编号输入域

接口：

Json GetElementById(object id)

描述：

获取指定编号输入域。

参数：

参数名	是否必填	数据类型	说明
Id	必填	Object	它可以是三种类型参数：字符串类型的元素 ID、NativeHandle 对象、引用对象。

返回值：

数据类型	说明	备注
Json	返回查找到的输入域属性信息	

代码示例：

```
let res=ctl.GetElementById("field1");
```

相同效果：

通过 GetElementProperties 方法特可以获取到元素属性

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

无

兼容接口：

五代独有接口，四代获取的是 dom 对象

接口优化：

无

示例地址：

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=GetElementById>

注意：

当元素名称值跟编号相同，通过该方法也能获取到 name 值跟编号值相同的元素对象。

8.9.10. 获取指定元素的属性

接口：

Json GetElementProperties (object id)

描述：

获取指定元素的属性。

参数：

参数名	是否必填	数据类型	说明
Id	必填	Object	它可以是三种类型参数：字符串类型的元素 ID、NativeHandle 对象、引用对象。

返回值：

数据类型	说明	备注
Json	返回查找到的输入域属性信息	

代码示例：

```
let res=ctl. GetElementProperties ( “field1” );
```

相同效果：

通过 GetElementById 方法特可以获取到元素属性

结合使用：

无

接口类型:

纯前端接口，不经过服务端

有无限制:

无

兼容接口:

兼容四代-跟四代编辑器效果一致。

接口优化:

无

示例地址:

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetElementProperties>

8.9.11.获取所有 name 属性相同的结构化元素数组。

接口:

Array GetElementsByName (string typename)

描述:

获取所有 name 属性相同的结构化元素数组。

参数:

参数名	是否必填	数据类型	说明
typename	必填	字符串	元素的 name 属性

返回值:

数据类型	说明	备注
数组	获取所有 name 属性相同的结构化元素数组。	

代码示例:

```
let res=ctl. GetElementsByName ("name");
```

相同效果:

可通过 GetAllInputFields 方法特可以获取到所有元素进行属性循环筛选。

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

无

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765//apiDocument/edit/index.html?apiname=GetElementsByName>

8.9.12. 获取相同编号的元素

接口:

Json GetElementsById (object id)

描述:

获取相同编号的元素。注意和 GetElementById 的区别。

参数:

参数名	是否必填	数据类型	说明
Id	必填	字符串	

返回值:

数据类型	说明	备注
数组	返回查找到的元素属性信息	

代码示例:

```
let res=ctl.GetElementsById ("field1");
```

相同效果:

可通过 GetAllInputFields 方法特可以获取到所有元素进行属性循环筛选。

结合使用:

ctl.GetElementProperties(res[i])获取对应元素的属性

接口类型:

纯前端接口，不经过服务端

有无限制:

无

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765//apiDocument/edit/index.html?apiname=GetElementsById>

8.9.13. *根据元素编号 (ID) 获取元素内容的 xml (默认不包含元素本身)

——参数修改日期

接口:

String GetElementInnerXmlById(object id, bool includeThis)

描述:

返回元素内容的文档 XML 字符串默认带上元素本身格式，比如输入域里面内容为张三，获取的 xml 数据重新插入到病历中还是输入域里面为张三，当第二个参数为 false，插入到病历中不带输入域只有内容张三。

参数:

参数名	是否必填	数据类型	说明
Id	必填	object	它可以是三种类型参数：字符串类型的元素 ID、NativeHandle 对象、引用对象。
includeThis	非必填	Bool	生成的 XML 中是否包含元素自身，默认是 true，包含元素本身结构。

返回值:

数据类型	说明	备注
String	返回元素外置内容的文档 XML 字符串	

代码示例:

```
let res=ctl.GetElementInnerXmlById(“field1”, false);
```

注意:

当需要实现元素数据同步功能，比如医生在入院记录中对主诉输入域中输入内容，然后书写首次病程记录的时候需要把入院记录中主诉内容自动同步，就可以先调用接口 GetElementInnerXmlById 保存不带元素格式的 xml，然后加载首次病程记录通过 InsertXmlById 方法在指定元素插入 xml 片段内容

相同效果:

当第二个参数为 true 时，与方法 `ctl.GetElementOuterXmlByID("field1")` 返回值效果一致

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

非五代独有接口。

接口优化:

无。

示例地址:

<https://www.dcwriter.cn:8765//apiDocument/edit/index.html?apiname=GetElementInnerXmlByID>

注意:

该接口名称跟四代有区别，后面的 D 是大写，四代的这个方法 d 是小写

8.9.14. 获取指定元素的值和文本

接口:

`String GetElementInnerValueStringByID (string id)`

描述:

获取元素 InnerValue 文本。

参数:

参数名	是否必填	数据类型	说明
Id	必填	String	字符串类型的元素 ID

返回值:

数据类型	说明	备注
String	返回元素 InnerValue 文本	

代码示例:


```
let res=ctl. GetElementInnerTextStringByID ( “field1” );
```

注意：

无

相同效果：

let res=ctl.GetElementProperties("field1")//与获取元素属性，找到 InnerValue 字段值一样。

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。

兼容接口：

非五代独有接口。

接口优化：

无。

示例地址：

<https://www.dcwriter.cn:8765//apiDocument/edit/index.html?apiname=GetElementInnerTextStringByID>

8.9.15. 设置指定元素的值和文本

接口：

```
Bool SetElementInnerTextStringByID(string id, string newValue, string newText)
```

描述：

兼容四代接口设置指定元素的值和文本。

参数：

参数名	是否必填	数据类型	说明
Id	必填	String	字符串类型的元素 ID、handle 对象、.NET 引用对象
newValue	必填	String	新的 value 文本

newText	必填	String	新的 Text 文本

返回值:

数据类型	说明	备注
Bool	返回设置是否成功	

代码示例:

```
let res=ctl. SetElementInnerValueStringByID ("field1", "newValue", "newText");
```

注意:

无

相同效果:

let res=ctl.SetElementProperties("field1",opts)//与修改元素属性接口执行效果一样。

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

非五代独有接口。

接口优化:

无。

示例地址:

<https://www.dcwriter.cn:8765//apiDocument/edit/index.html?apiname=SetElementInnerValueStringByID>

8.9.16. 设置元素是否只读

接口:

```
Bool SetElementContentReadonly(string id, bool breadonly)
```

描述:

设置元素是否只读。

参数:

参数名	是否必填	数据类型	说明
Id	必填	String	字符串类型的元素 ID, NativeHandle 属性也是可以以及当前元素对象也可以。
breadonly	必填	Bool	是否只读

返回值:

数据类型	说明	备注
Bool	返回设置是否成功	

代码示例:

```
let res=ctl.SetElementContentReadonly ("field1", false);
```

注意:

无

相同效果:

let res=ctl.SetElementProperties("field1",ops)// 与修改元素属性, 找到ContentReadonly 字段赋值为 true 效果一样。

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在编辑模式下能调用。

兼容接口:

非五代独有接口。

接口优化:

无。

示例地址:

<https://www.dcwritter.cn:8765//apiDocument/edit/index.html?apiname=SetElementContentReadonly>

8.9.17. 设置元素是否显示

接口:

```
Bool SetElementVisibility(string id, bool visibility)
```

描述:

设置元素是否显示。

参数:

参数名	是否必填	数据类型	说明
Id	必填	String	字符串类型的元素 ID
visibility	必填	Bool	是否显示

返回值:

数据类型	说明	备注
Bool	返回设置是否成功	

代码示例:

```
let res=ctl.SetElementVisibility("field1", false);
```

注意:

无

相同效果:

let res=ctl.SetElementProperties("field1",ops)//与修改元素属性，找到 Visible 字段赋值为 false 效果一样。

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在编辑模式下能调用。

兼容接口:

非五代独有接口。

接口优化:

无。

示例地址:

<https://www.dcwritter.cn:8765//apiDocument/edit/index.html?apiname=SetElementVisibility>

8.9.18. 获取指定容器下的指定 ID 的元素对象

接口:

Json GetElementByIdExt(object id, object specifyParent)

描述:

在指定的容器中获取指定编号元素。

参数:

参数名	是否必填	数据类型	说明
Id	必填	Object	它可以是三种类型参数：字符串类型的元素 ID、NativeHandle 对象、引用对象。
specifyParent	非必填	Object	它可以是三种类型参数：元素 ID、NativeHandle 对象、引用对象。

返回值:

数据类型	说明	备注
Json	返回查找到的元素属性信息	

代码示例:

```
let res=ctl. GetElementByIdExt ("field1",  ctl.CurrentTable());
```

注意:

无

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

无

兼容接口:

非五代独有接口。

接口优化:

无。

示例地址:

<https://www.dcwritter.cn:8765//apiDocument/edit/index.html?apiname=GetElementByIdExt>

8.9.19. 将光标定位在指定的元素中

接口：

Bool FocusElement(object item)

描述：

将光标定位在指定的元素中。

参数：

参数名	是否必填	数据类型	说明
item	必填	Object	它可以是三种类型参数：字符串类型的元素 ID、NativeHandle 对象、引用对象。

返回值：

数据类型	说明	备注
Bool	是否执行成功	

代码示例：

```
let res=ctl.FocusElement("field1");
```

注意：

无

相同效果：

```
let res=ctl.FocusAdjacent("afterBegin","field1")
let res=ctl.FocusElementById("field1");
```

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

预览下无法定位

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://www.dcwritr.cn:8765//apiDocument/edit/index.html?apiname=FocusElement>

8.9.20. 将光标定位在指定的元素中

接口:

Bool FocusElementById(string id,string scrollMode)

描述:

将光标定位在指定的元素中。

参数:

参数名	是否必填	数据类型	说明
Id	必填	String	字符串类型的元素 ID
scrollMode (2024-09-02 新增)	非必填	string	top: 元素定位之后元素在编辑器视图顶端 middle: 元素定位之后元素在编辑器视图居中 bottom: 元素定位之后元素在编辑器视图底端

返回值:

数据类型	说明	备注
Bool	是否执行成功	

代码示例:

```
let res=ctl.FocusElementById("field1");
或
ctl.FocusElementById("field1","middle");
```

注意:

无

相同效果:

```
let res=ctl.FocusAdjacent("afterBegin","field1")
let res=ctl.FocusElement("field1");
```

结合使用:

无

接口类型:

纯前端接口，不经过服务端。

有无限制:

打印预览模式下无法定位

兼容接口：

五代独有接口。

接口优化：

在 2024-09-02 新增第二个参数，可以自定义元素定位之后，元素在编辑器视图顶端、居中、底端位置显示。

示例地址：

<https://www.dcwritr.cn:8765//apiDocument/edit/index.html?apiname=FocusElementById>

8.9.21. *设置元素的属性

接口：

Bool SetElementProperties(object element, json options, bool issetText)

描述：

设置元素的属性。该接口通常情况下需要和其他接口结合使用，如 GetElementByIdExt、GetElementProperties。

参数：

参数名	是否必填	数据类型	说明
element	必填	Object	它可以是三种类型参数：字符串类型的元素 ID、NativeHandle 对象、引用对象。
options	必填	Json	Json 格式 的 元 素 属 性 。 可 使 用 GetElementProperties 接口的返回信息，详细的属性内容可参考本文的附录中的元素属性。
issettext	非必填	Bool	是否设置文本属性。默认是 false，不设置。

返回值：

数据类型	说明	备注
Bool	是否执行成功	

代码示例：

```
var normalfield = ctl.GetElementByIdExt("txtDate");
if (normalfield != null) {
    var options = ctl.GetElementProperties(normalfield);
}
```



```
console.log(options);
options.InnerEditStyle = "Numeric";
options.SpecifyWidth = 200;
options.Alignment = "Center";
options.DisplayFormat = {
    Style: "Numeric", //注意拼写
    Format: "0.00"
};
ctl.SetElementProperties(normalfield, options);
}

//对单、复选框设置属性表达式:
ctl.SetElementProperties("sex_0",{
    PropertyExpressions:{
        "Checked": "if([field1]=1,true,false)"
    }
})
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端**结合使用:**

无

有无限制（暂定）:

编辑器初始化成功之后进行调用都行（预览模式除外）

兼容接口:

兼容四代

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SetElementProperties>

注意:

能修改所有结构化元素属性，包含病程元素

8.9.22. *插入给定的元素列表到指定容器

接口:

Bool SetChildElements(object targetEle, object newEles, string options)

描述:

插入给定的元素列表到指定容器。

参数:

参数名	是否必填	数据类型	说明
targetEle	必填	Object	指定容器元素的后台.NET 引用对象或元素的 ID 或元素的 NativeHandle 值，一般可使用 GetElementByIdExt 的返回对象
newEles	必填	Object	指定插入的元素对象集合。支持插入输入域、单选框、复选框、图片、html、text 内容的组合或者单一插入。 输入域类型: input 图片类型: image 单选框类型: radio 复选框类型:checkbox 条形码类型: barcode 二维码类型: tdcode 页码类型: pageinfo 段落类型: paragraphflag 文本标签类型: label 水平线类型: horizontalline 按钮类型: button 医学表达式类型: medicalexpression 音视频类型: media 表格类型: table 纯文本类型: text

			html 类型:html
options	非必填	String	插入位置: afterbegin 结束位置 beforebegin 最后位置 (在空容器中段落符号之前插入新的内容, 单元格里插入元素, 不会出现换行) after: 元素并排后面插入内容 before: 元素并排前面插入内容 index 坐标序号。
isRefresh	非必填	bool	是否刷新 默认值为 true, 调用一次该方法就刷新一次, 当调用多次该方法, 会影响性能 false, 调用该方法不进行刷新, 当接口调用完毕可以在最后面调用刷新方法
isAddTrackIn fos	非必填	bool	插入内容是否带上痕迹 默认值为 false 当设置成 true, 用户登录成功之后, 调用该接口新插入的内容就会带上痕迹

返回值:

数据类型	说明	备注
Bool	是否执行成功	

代码示例:

```
var options = [
  {
    //图片
    'image':
    {
      id: "123", //图片id
      //src: redpicbase64data, //图片路径
      src: "http://localhost:5214/pingcode/webdemo.png", //图片路径
      width: "100", //图片宽度
      height: "100", //图片高度
      savecontentinfile: false
    }
  }
]
```

```
        }  
        'text': "文本1\r\n文本2"  
    }  
];  
var field1 = ctl.GetElementByIdExt("field1");  
ctl.SetChildElements(field1, options, 'afterEnd', true, false)
```

相同效果:

无

应用场景:

可以给签名输入域里面插入一张签名图片

接口扩展:

当方法中传的指定元素不存在的时候，调用方法无效。

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制（暂定）:

编辑器初始化成功之后进行调用都行（预览模式、阅读模式除外）

兼容接口:

兼容四代

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SetChildElements>

8.9.23. *获取当前单选框或复选框对象

接口:

Object CurrentCheckboxOrRadio()

描述:

获取当前单选框或复选框对象。

参数:

无

返回值:

数据类型	说明	备注
Object	返回当前对象的引用对象	返回值: {_id: 4}

代码示例:

```
let res=ctl.CurrentCheckboxOrRadio();
```

相同效果:

无

接口扩展:

如需获取到当前选中的单选框或者复选框, 可以搭配 `GetElementProperties` 获取元素属性或者通过 `SetElementProperties` 修改属性

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制 (暂定):

编辑器初始化成功之后进行调用都行 (预览模式除外)

兼容接口:

兼容四代

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=CurrentCheckboxOrRadio>

8.9.24. *设置指定编号的单选框或复选框是否选中。

接口:

`Bool SetElementCheckedByID (string id, bool isChecked)`

描述:

设置指定编号的单选框或复选框是否选中。

参数:

参数名	是否必填	数据类型	说明
Id	必填	字符串	指定的单选框、复选框元素编号。
isChecked	必填	Bool	是否勾选

返回值:

数据类型	说明	备注
Bool	设置是否成功	

代码示例:

```
let res=ctl.SetElementCheckedByID(strID, true);
```

相同效果:

无

接口扩展:

当对一组单选框组修改选中状态的时候,比如男、女,先勾选男,然后通过这个接口勾选女,界面上男还是勾选状态,不会产生互斥的效果

结合使用:

无

接口类型:

纯前端接口,不经过服务端

有无限制(暂定):

编辑器初始化成功之后进行调用都行(预览模式除外)

兼容接口:

兼容四代

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SetElementCheckedByID>

8.9.25. *获取全部或指定名称的单选框或复选框数组

接口:

Array GetAllCheckboxOrRadio (string type, string name)

描述:

获取全部或指定名称的单选框或复选框数组。

参数：

参数名	是否必填	数据类型	说明
type	必填	字符串	Radio 表示单选框。Checkbox 表示复选框。大小写不区分。
name	必填	Bool	指定的元素的 name 属性（可传空，表示获取所有）

返回值：

数据类型	说明	备注
Bool	设置是否成功	

代码示例：

```
let res=ctl. GetAllCheckboxOrRadio ( "radio" , "name1" );
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制（暂定）：

编辑器初始化成功之后进行调用都行（预览模式除外）

兼容接口：

兼容四代

接口优化：

无

示例地址：

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=GetAllCheckboxOrRadio>

8.9.27. 闪烁指定的元素，并移动滚动条使元素在页面上可见

接口：

Bool FlashElements(object element, bool autoScroll)

描述:

闪烁指定的元素，并移动滚动条使元素在页面上可见。

参数:

参数名	是否必填	数据类型	说明
element	必填	字符串	指定的容器的引用对象, 如果为 null 则表示全文
nestNode	必填	Bool	兼容参数, 不做任何意义

返回值:

数据类型	说明	备注
数组	操作是否成功	

代码示例:

```
let res=ctl.FlashElements ( "field1", true);
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制（暂定）:

编辑器初始化成功之后进行调用都行（预览模式除外）

兼容接口:

五代独有接口

接口优化:

无

示例地址:

8.9.28. 获取指定元素的 xml 字符串

接口:

String GetElementXMLFragmentByID(string id, bool includeThis)

描述:

返回包含数据的 XML 片段, 本函数返回的 XML 字符串可以作为编辑器控件或文档对象的函数 CreateElementByXMLFragment() 的参数。

参数:

参数名	是否必填	数据类型	说明
Id	必填	字符串	元素的 ID
includeThis	必填	Bool	生成的 XML 中是否包含元素自身

返回值:

数据类型	说明	备注
String	返回包含数据的 XML 片段	

代码示例:

```
let res=ctl.GetElementXMLFragmentByID("field1", true);
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器初始化成功之后进行调用都行。

兼容接口:

五代独有接口

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=GetElementXMLFragmentByID>

8.9.29. 获得文档的 Html 文本内容

接口:

String GetHtmlText(bool IncludeSelectionOnly)

描述:

获得文档的 Html 文本内容

参数:

参数名	是否必填	数据类型	说明
IncludeSelectionOnly	必填	Bool	是否只包含选择区域

返回值:

数据类型	说明	备注
String	返回包含数据的 html 内容字符串	

代码示例:

```
let res=ctl. GetHtmlText ( "field1", true);
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制 (暂定):

无

兼容接口:

五代都有接口

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=GetHtmlText>

8.9.30. 根据开始元素和结束元素选择

接口:

Bool SelectContentByStartEndElement (object startEle, object endEles)

描述:

根据开始元素和结束元素选择。

参数:

参数名	是否必填	数据类型	说明
startEle	必填	Object	指定元素的 id，或者后台.NET 引用对象、Handler 对象，一般可使用 GetElementByIdExt 的返回对象
endEles	必填	Object	指定元素的 id，或者后台.NET 引用对象、Handler 对象，一般可使用 GetElementByIdExt 的返回对象

返回值:

数据类型	说明	备注
Bool	是否执行成功	

代码示例:

```
ctl.SelectContentByStartEndElement("20150112104536","201407171240268434");
let startNodeElement = null;
let endNodeElement = null;
startNodeElement = ctl.GetElementProperties(ctl.CurrentElement())
endNodeElement = ctl.GetElementProperties(ctl.CurrentElement())
    ctl.SelectContentByStartEndElement(startNodeElement, endNodeElement)
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制（暂定）:

无

兼容接口:

五代都有接口

接口优化:

无

示例地址:

<https://www.dcwritr.cn:8765/apiDocument/edit/index.html?apiname=SelectContentByStartElement>

8.9.31. *根据元素编号（ID）来选中元素

接口:

Bool SelectElementById(string id)

描述:

选择指定编号的元素（输入域、单选框、复选框、表格、条形码、二维码等结构化元素都有效）。

参数:

参数名	是否必填	数据类型	说明
Id	必填	Strng	元素编号

返回值:

数据类型	说明	备注
Bool	是否执行成功	无实际意义

代码示例:

```
var result=ctl. SelectElementById("20150112104536");
```

注意:

当需要选中的元素在当前编辑器视图的下方位置，编辑器视图会自动跳转到该元素的下方位置，当需要选中的元素在当前编辑器视图的上方位置，编辑器视图会自动跳转到该元素的上方位置

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。打印预览模式下该接口无对应效果。（验证版本 2024-11-29）

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<http://www.dcwriter.cn:8022/apiDocument/edit/index.html?apiname=SelectElementById>

8.9.32. *根据表格编号（ID）使其从表格最后一行开始新增行扩充至当前页页底

接口:

bool InsertRowsToPageBottomByAfterRowHasNewPage(string id)

描述:

在指定表格下面插入空行，并延伸到页面底端。

参数:

参数名	是否必填	数据类型	说明
Id	必填	Strng	表格编号

返回值:

数据类型	说明	备注
Bool	是否执行成功	

代码示例:

```
var result=ctl.InsertRowsToPageBottomByAfterRowHasNewPage("table5");
```

注意:

只读模式、阅读模式、打印预览模式下能新增表格行，但是无法撤销

相同效果:

ctl.DCExecuteCommand("Table_InsertRowsToPageBottom",false,null)，光标放在表

格中就自动新增表格行至当前页底

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=InsertRowsToPageBottomByAfterRowHasNewPage>

8.9.33. *自定义单、复选框勾选图标

接口：

```
void SetDomImageByBase64String(string key, string base64String)
```

描述：

自定义单、复选框勾选和不勾选框的图标样式，图标数据为base64 格式且大小为 16*16。

第一个参数支持下面四种格式：

- 1. CheckBoxChecked, 表示复选框选中时的图标；
- 2. CheckBoxUnchecked, 表示复选框未选中时的图标；
- 3. RadioBoxChecked, 表示单选框选中时的图标；
- 4. RadioBoxUnchecked, 表示单选框未选中时的图标；

参数：

参数名	是否必填	数据类型	说明
Key	必填	字符串	修改单、复选框勾选或者不勾选图标枚举值。
base64String	必填	字符串	Base64 格式的图片数据（16*16）

返回值：

数据类型	说明	备注
Bool	是否执行成功	

代码示例:

```
var result=ctl.SetDomImageByBase64String(keyName, xmlText);
```

注意:

自定义图标格式必须小于等于 16*16 大小

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetDomImageByBase64String>

注意:

当用户需要重置单、复选框图标样式时，可以使用方法 `SetDomImageByBase64String("",null);` 重置指定样式的图标，但是需要结合方法 `RefreshInnerView(true)` 才能起效果。（2025-01-10 后版本支持重置图标功能）

8.9.34. *用户锁定指定输入域元素-只有特定用户才能编辑

接口:

```
Bool LockContentByElement(object element, string userID, Array  
authoriseUserIDList, bool logUndo)
```

描述:

用户登录后可以对指定输入域元素进行用户锁定，只有特定的用户才能进行编辑。

参数：

参数名	是否必填	数据类型	说明
element	必填	Object	可以传元素编号 元素.NET 对象 元素的 NativeHandle 值
userID	必填	string	由哪个用户锁定,传用户登录接口中的 UserID
authoriseUserIDList	必填	Array	数组对象
logUndo	非必填	bool	是否记录撤销操作信息

返回值：

数据类型	说明	备注
bool	是否执行成功	

代码示例：

```
let res=ctl.LockContentByElement("txtAge","lisi", null, true);
```

注意：

ctl.LockContentByElementID("txtAge",null,null, false);取消用户锁定功能

相同效果：

LockContentByElementID 方法类似，就第一个参数不同，该方法只能传输入域元素的编号为参数才行。

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。推荐是编辑器允许编辑的情况下进行调用

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=LockContentByElement>

8.9.35. *用户锁定指定输入域编号（ID）元素-只有特定用户才能编辑

接口：

```
bool LockContentByElementID(string elementID, string userID, string  
authoriseUserIDList, bool logUndo)
```

描述：

用户登录后可以对指定输入域元素进行用户锁定，只有特定的用户才能进行编辑。

参数：

参数名	是否必填	数据类型	说明
elementID	必填	字符串	元素编号，必须为一个容器类型的元素的编号
userID	必填	字符串	由哪个用户锁定，传用户登录接口中的 UserID
authoriseUserIDList	必填	字符串	允许用户编辑的账号 ID 列表集，以英文逗号分隔
logUndo	必填	Bool	是否记录撤销操作信息

返回值：

数据类型	说明	备注
Bool	操作是否成功	

代码示例：

```
let res=ctl.LockContentByElementID('性别', 'lisi', '', true);
```

相同效果：

LockContentByElement 方法类似，就第一个参数不同，该方法能传输入域元素的编号、唯一值（NativeHandle）、.NET 对象。

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。推荐是编辑器允许编辑的情况下进行调用

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=LockContentByElementID>

8.9.36. *用户锁定指定表格行-只有特定用户才能编辑

接口：

```
bool LockContentByTableRow(string tableID, int rowIndex, string
userID, string authoriseUserIDList, bool logUndo)
```

描述：

用户登录后可以对指定表格编号中锁定对应表格行索引值对应的表格行元素，使得表格行只能由指定的用户才能修改。

参数：

参数名	是否必填	数据类型	说明
tableID	必填	字符串	表格编号
rowIndex	必填	Int	表格行序号，从 0 开始计算的表格行在表格中的序号
userID	必填	字符串	锁定操作的用户 ID
authoriseUserIDList	必填	字符串	授权操作的用户 ID 列表，各个列表之间用英文逗号分开
logUndo	必填	Bool	是否记录撤销操作信息

返回值：

数据类型	说明	备注
Bool	操作是否成功	

代码示例：

```
let res=ctl.LockContentByTableRow("table1",1,"lisi","lisi,zhangsan",true);
```

相同效果:

无。

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。推荐是编辑器允许编辑的情况下进行调用

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=LockContentByTableRow>

8.9.37. *获取全文单、复选框被勾选的编号、名称值组合

接口:

```
string GetCheckedValueList(string splitter, bool includeCheckbox,
bool includeRadio, bool includeElementID, bool includeElementName)
```

描述:

获得文档中所有的勾选的单选框、复选框元素的值。例如调用
ctl.GetCheckedValueList(";", true, true, true, true) 返回类似字符串 “xbzw; 胸部
正位; gpzw; 骨盆正位; fbww; 腹部卧位”。

参数:

参数名	是否必填	数据类型	说明
splitter	必填	String	各个项目之间的分隔字符串
includeCheckbox	必填	Bool	返回值是否包含复选框勾选值
includeRadio	必填	Bool	返回值是否包含单选框勾选的值

includeElementID	必填	Bool	是否包含勾选的单、复选框元素 ID 值
includeElementName	必填	Bool	是否包含勾选的单、复选框 Name 属性值

返回值:

数据类型	说明	备注
String	返回字符串	

代码示例:

```
let res=ctl.GetCheckedValueList(";", true, true, true, true);
```

相同效果:

无。

结合使用:

客户获取到文档中所有勾选框的编号，解析后循环可以通过方法 GetElementProperties 获取到勾选框的文本和 Value 值

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetCheckedValueList>

8.9.38. *根据单、复选框元素编号（ID）获取勾选状态

接口:

```
bool GetElementChecked(string id)
```

描述:

获得单、复选框元素的勾选状态，如果没找到元素则返回 false。

参数:

参数名	是否必填	数据类型	说明
Id	必填	String	单复选框文档元素的编号

返回值:

数据类型	说明	备注
Bool	返回元素是否勾选	

代码示例:

```
let res=ctl.GetElementChecked("chk1");
```

相同效果:

无。

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetElementChecked>

8.9.39. *根据元素编号（ID）获取元素内容的 xml（包含元素本身）

接口:

```
String GetElementOuterXmlById(string id)
```

描述:

返回元素内容的文档 XML 字符串—带上元素结构，比如输入域里面内容为张三，获取的 xml 数据重新插入到病历中输入域里面为张三

参数:

参数名	是否必填	数据类型	说明

Id	必填	String	文档元素的编号

返回值:

数据类型	说明	备注
String	返回元素的属性构成的 XML	

代码示例:

```
let res=ctl.GetElementOuterXmlById ("field1");
```

相同效果:

与方法 `ctl.GetElementInnerXmlById(“field1”)` 返回值一致。

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

[https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetElementOuterXml
ById](https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetElementOuterXmlById)

8.9.40. *根据元素编号（ID）获取元素可见性

接口:

```
bool GetElementVisible(string id)
```

描述:

获得指定元素的可见性状态，当元素在界面显示调用接口返回 `true`，界面上隐藏调用接口返回 `false`，且文档中不存在该元素调用接口返回 `false`。

参数:

参数名	是否必填	数据类型	说明

Id	必填	String	文档元素的编号

返回值:

数据类型	说明	备注
Bool	返回元素是否的可见性属性的值	

代码示例:

```
let res=ctl.GetElementVisible ("field");
```

相同效果:

与 `var xx=ctl.GetElementProperties('field1');xx.Visible` 返回值结果一致, 推荐使用这个方式获取元素的可见性。

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetElementVisible>

8.9.41. *根据元素编号 (ID) 获取 InnerValue 值

接口:

```
String GetInputFieldInnerValue(string id)
```

描述:

获得指定编号的输入域的 InnerValue 属性值。比如性别下拉输入域, 男对应 0、女对应 1, 通过该接口返回性别实际的 Value 值存入数据库

参数:

参数名	是否必填	数据类型	说明

Id	必填	String	文档元素的编号

返回值:

数据类型	说明	备注
String	返回输入域的 InnerValue 属性值	

代码示例:

```
let res=ctl.GetInputFieldInnerValue ("field1");
```

相同效果:

无。

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

五代独有接口。

接口优化:

无。

示例地址: (需更新)

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetInputFieldInnerValue>

8.9.42. *根据单、复选框元素编号 (ID) 设置勾选状态

接口:

```
bool SetElementChecked(string id, bool newChecked)
```

描述:

设置单、复选框的勾选状态，如果没找到元素则返回 false。

参数:

参数名	是否必填	数据类型	说明
Id	必填	String	单复选框文档元素的编号

newChecked	必填	Bool	是否勾选

返回值：

数据类型	说明	备注
Bool	返回操作是否成功	

代码示例：

```
let res=ctl.SetElementChecked("chk1", true);
```

注意：

调用接口勾选或者取消勾选后，无法撤销操作，且当需要修改某一组中单选框勾选状态时无法实现互斥效果。比如：男、女单选框组，女性优先勾选，调用接口对男性进行勾选之后，女性的勾选状态不会自动取消。

相同效果：

无。

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下无效

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：（需更新）

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetElementChecked>

8.9.43. *根据元素对象给指定元素赋值文本内容

接口：

```
bool SetElementText(object Element, string text)
```

描述：

设置文档元素文本内容。

参数：

参数名	是否必填	数据类型	说明
Element	必填	Object	.NET 引用对象、元素编号（ID）、NativeHandle 值
text	必填	String	文本内容

返回值：

数据类型	说明	备注
Bool	返回操作是否成功	

代码示例：

```
let res=ctl.SetElementText("field", "123");
```

注意：

当执行了登录接口后、调用该方法赋值的内容也会带上痕迹。且赋值完毕能进行撤销

相同效果：

SetElementText、SetElementTextByIDPro 等接口都能对元素进行赋值，但是只能对容器元素赋值，比如输入域元素、单元格元素。

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。阅读模式、只读模式、打印预览模式下无效

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：（需更新）

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetElementText>

8.9.44. *根据元素编号（ID）获取元素可见性

接口：

```
bool SetElementVisible(string id, bool visible)
```

描述：

设置文档元素可见性。

参数：

参数名	是否必填	数据类型	说明
Id	必填	String	元素编号
visible	必填	Bool	是否可见、传 false 设置元素不可见

返回值：

数据类型	说明	备注
Bool	返回操作是否成功	

代码示例：

```
let res=ctl.SetElementVisible("field1", false);
```

注意：

调用完该接口设置元素隐藏，无法撤销操作

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。阅读模式、只读模式、打印预览模式下无效

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetElementVisible>

8.9.45. *根据元素编号（ID）设置 InnerValue 值

接口：

```
bool SetInputFieldInnerValue(string id, string newValue)
```

描述：

设置指定编号的输入域的 InnerValue 属性值。

参数：

参数名	是否必填	数据类型	说明
Id	必填	String	输入域编号
newValue	必填	String	文本

返回值：

数据类型	说明	备注
Bool	返回操作是否成功	

代码示例：

```
let res=ctl.SetInputFieldInnerValue("field1", "1");
```

相同效果：

ctl.SetElementInnerValueStringByID('field1','男','1'), 同时修改输入域元素的 Text 值和 Value 值，推荐使用该方法

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。阅读模式、只读模式、打印预览模式下该接口无效

兼容接口：

五代独有接口。

接口优化：

无。

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetInputFieldInnerValue>

8. 9. 46. *根据元素对象给指定元素插入片段内容

接口:

```
bool InsertContentByInputID(string content, string format, string elementID, bool clearold)
```

描述:

在指定编号的输入域内追加带有格式的片段内容。格式文档支持 XML 和 JSON、html、text。

参数:

参数名	是否必填	数据类型	说明
content	必填	String	格式文档内容
format	必填	String	文档的格式, 支持 XML 和 JSON、html、text
elementID	必填	String	输入域编号或者输入域 NativeHandle 值 (2024 年 5 月 16 日)
clearold	必填	Bool	是否清除元素原有内容
withTrackInfos	非必填	Bool	插入内容是否带痕迹, 默认值为 false, 赋值内容不带痕迹(2024-05-20 后版本支持)

返回值:

数据类型	说明	备注
Bool	操作是否成功	

代码示例:

```
var str="哈哈";
let result = ctl.InsertContentByInputID(str, "text", "txtAge", false);
```

注意:

插入内容成功后, 无法撤销

相同效果:

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=InsertContentByInputID>

8.9.47. *根据输入焦点位置批量插入图片

接口：

object InsertImageByJSONText(Array arr, function callback)

描述：

通过 JSON 参数插入图片。

参数：数组形式的 json 对象，它的属性包含

参数名	是否必填	数据类型	说明
ID	必填	string	图片编号
Width	必填	float	图片宽度
Height	必填	float	图片高度
Src	必填	string	图片来源，支持网络图片，但是网络图片地址需跟编辑器引用地址保持同源。支持 base64 图片数据
EnableEditImageAd ditionShape	非必填	Bool	图片是否允许编辑，默认双击图片会弹出图片编辑框，设置 false, 不触发

SaveContentInFile	非必填	Bool	图片数据是否保留文档中。默认 true 保存
callback	非必填	function	该方法的回调函数，执行完之后进行触发，实现另外的业务逻辑

返回值：

数据类型	说明	备注
Object	插入的图片的 object 对象	

代码示例：

```
var ctl = document.getElementById("myWriterControl");
    var arr = new Array();
    var options2 = {
        id: "123",
        src: "http://www.dcwritter.cn:8022/img/DC-Logo.png",
        width: "500",
        height: "1000",
        SaveContentInFile: true,
        KeepWidthHeightRate: false
    };
    arr.push(options2);
    ctl.InsertImageByJSONText(arr);
```

注意：

插入网络图片需要注意地址得在项目上，否则会出现跨域

相同效果：

SetChildElements 中也支持插入图片

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。阅读模式、只读模式、打印预览模式下无

兼容接口：

兼容四代。

接口优化：

五代编辑器下接口名称进行大小写优化。

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=InsertImageByJSONText>

8.9.48. *在当前位置处插入批注（选中内容设置批注）

接口:

```
bool NewComment(json parameter)
```

描述:

在当前位置插入批注。

参数: 数组形式的 json 对象, 它的属性包含

参数名	是否必填	数据类型	说明
text	必填	String	批注文本内容
author	必填	String	操作者名称
authorid	非必填	String	操作者编号
backcolor	非必填	String	批注显示的背景色
forecolor	非必填	String	批注显示的前景色
bordercolor	非必填	String	批注显示的边框色
username	非必填	String	同 author, 兼容性参数
userid	非必填	String	同 authorid, 兼容性参数

返回值:

数据类型	说明	备注
Bool	操作是否成功	

代码示例:

```
var opts = {
  BackgroundColor: "#64ff00", //控制批注背景颜色 (20220324)
  ForeColor: "#D9D919", //字体颜色
  elebgColor: "#084ffc", //被添加批注的元素的背景颜色 (20220324)
  Superscript: false, //顺序标记是否显示 (20220324)
  Text: "这是文档批注",
  UserName: "李四主任医师",
  UserID: "lisi"
}
```



```
var result=ctl.NewComment(opts);
```

注意:

当编辑器处于只读模式下，默认无法插入批注，需要设置文档选项 DocumentOptions.BehaviorOptions.CommentEditableWhenReadonly=true 才允许指定模式下插入批注

相同效果:

ctl.DCExecuteCommand('InsertComment',false,options);五代插入批注推荐使用该方式

结合使用:

无。

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。阅读模式、打印预览模式下无效

兼容接口:

兼容四代。

接口优化:

跟四代使用一致。

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=NewComment>

8.9.49. 获取文档中所有批注列表

接口:

Array getCommentList()

描述:

获取文档中所有的批注。

参数:

无

返回值:

以数组的形式返回文档中的所有批注信息。单个批注的信息如下:

名称	数据类型	说明

Text	String	批注的文本信息
Author	String	操作者名称
AuthorID	String	操作者编号
BackColor	String	批注显示的背景色
BorderColor	String	批注显示的边框色
ForeColor	String	批注显示的前景色
Attributes	Array	批注的自定义属性信息

代码示例:

```
let res=ctl.getCommentList();
```

相同效果:

无

结合使用:

无。

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

兼容四代。

接口优化:

跟四代使用一致。

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=getCommentList>

8.9.50. *根据批注索引修该指定批注文本内容

接口:

bool setCommentContent(int index, string newContent)

描述:

修改指定的批注的内容。

参数:

参数名	是否必填	数据类型	说明
index	必填	Int	指定的批注在所有批注中的序号。从上到下，从 0 开始。
newContent	必填	String	文本

返回值：

数据类型	说明	备注
Bool	返回操作是否成功	

代码示例：

```
var currentcomment=ctl.GetCurrentComment();  
let res=ctl.setCommentContent(currentcomment.IndexByList, "123");
```

相同效果：

无

结合使用：

先获取到当前批注索引：var currentcomment=ctl.GetCurrentComment();var index=currentcomment.IndexByList;然后根据 index 值修改指定批注内容

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。阅读模式下无法修改

兼容接口：

兼容四代。

接口优化：

五代修改需要跟其它接口组合使用。

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=setCommentContent>

8.9.51. *删除当前选中批注信息

接口：

```
bool DeleteComment(int index)
```

描述：

删除指定的批注。

参数：

参数名	是否必填	数据类型	说明
index	必填	Int	指定的批注在所有批注中的序号。从上到下，从 0 开始。

返回值：

数据类型	说明	备注
Bool	返回操作是否成功	

代码示例：

```
var currentcomment=ctl.GetCurrentComment();  
var index=current.IndexByList;  
let res=ctl.DeleteComment (index);
```

相同效果：

ctl.DCExecuteCommand('DeleteComment',false,true)//删除当前选中的批注信息

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。

兼容接口：

兼容四代。

接口优化：

五代修改需要跟其它接口组合使用。

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname>DeleteComment>

8.9.52. *获取当前光标处批注信息

接口：

Json GetCurrentComment()

描述：

获取当前的批注的信息。

参数：

无

返回值：

名称	数据类型	说明
Text	String	批注的文本信息
Author	String	操作者名称
AuthorID	String	操作者编号
BackColor	String	批注显示的背景色
BorderColor	String	批注显示的边框色
ForeColor	String	批注显示的前景色
Attributes	Array	批注的自定义属性信息
Index	int	插入批注先后顺序索引值
IndexByList	int	批注索引值用来修改批注和删除批注

代码示例：

```
let res=ctl.GetCurrentComment();
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetCurrentComment>

8. 9. 53. *修改点击批注的信息

接口:

bool SetCurrentComment (json parameter)

描述:

设置当前批注的内容。

参数:

参数名	是否必填	数据类型	说明
text	必填	String	批注文本内容
author	必填	String	操作者名称
authorid	非必填	String	操作者编号
backcolor	非必填	String	批注显示的背景色
forecolor	非必填	String	批注显示的前景色
bordercolor	非必填	String	批注显示的边框色
username	非必填	String	同 author，兼容性参数
userid	非必填	String	同 authorid，兼容性参数

返回值:

数据类型	说明	备注
Bool	操作是否成功	

代码示例:

```
var opts = {
    BackgroundColor: "#64ff00", //控制批注背景颜色 (20220324)
    ForeColor: "#D9D919", //字体颜色
    elebgColor: "#084ffc", //被添加批注的元素的背景颜色 (20220324)
    Superscript: false, //顺序标记是否显示 (20220324)
    Text: "这是文档批注",
    UserName: "李四主任医师",
    UserID: "lisi"
}
var result=ctl.SetCurrentComment(opts);
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。阅读模式下无法修改

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetCurrentComment>

8.9.54. *清空文档中所有输入域内容

接口:

Void ClearAllFields()

描述:

清空所有输入域的文本。

参数:

无

返回值:

无

代码示例:

```
ctl.ClearAllFields();
```

注意:

清空的内容无法撤销操作且当输入域里面嵌套了内容, 输入域里面的内容也会清空

相同效果:

ctl.DCExecuteCommand('ClearAllFieldValue', false, true); 清空文档正文中所有输入域内容, 但是当输入域设置了不允许用户编辑或者只读, 该命令无效

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下无效

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=ClearAllFields>

8. 9. 55. *根据不同条件获取文档中对应输入域集合

接口：

Array GetAllInputFields (Bool excludeReadonly, Bool excludeHiddenElement, String specifyRootElement)

描述：

当方法没有传递参数时，获取文档中所有的输入域集合列表。且可以根据参数不同获取对应的输入域集合列表

参数：

参数名	是否必填	数据类型	说明
excludeReadonly	非必填	Bool	是否排除只读元素
excludeHiddenElement	非必填	Bool	是否排除隐藏元素
specifyRootElement	非必填	String	获取指定容器对象下的输入域集合列表。 比如需要获取一个输入域里面所有子输入域元素，传父元素的编号即可 比如需要获取一个表格所有输入域，传表格编号即可 比如需要获取一个病程中所有输入域，传病程编号即可

			比如需要获取页眉中所有输入域传参为 header 比如需要获取正文中所有输入域传参为 body 比如需要获取页脚中所有输入域传参为 footer 不传参数默认获取全文输入域

返回值:

返回输入域集合对象列表

代码示例:

```
let result = ctl.GetAllInputFields();
```

注意:

当通过该方式返回出输入域对象集合，循环操作处理业务，比如获取输入域的属性、修改输入域属性、取输入域内容、设置输入域内容等操作，一律获取到输入域对象中的 NativeHandle 值来进行业务处理，不建议直接传元素对象或者元素编号处理业务，或者会出现获取输入域对象不准的问题（文档中存在相同编号的元素）

相同效果:

ctl.GetElementsByTypeName('XTextInputFieldElement');获取输入域类型的集合

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

兼容四代。

接口优化:

对比四代，五代接口可以单独获取页眉、正文、页脚中所有输入域

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetAllInputFields>

8.9.56. *获取转科信息中标签文本元素信息

接口：

Json GetLabelElementContactSettings(objcet parame)

描述：

获取标签的连接信息。

参数：

参数名	是否必填	数据类型	说明
parame	必填	object	标签元素属性对象

返回值：

名称	数据类型	说明
ID	String	编号
Width	Float	宽度
Height	Float	高度
Text	String	文本
StrictMatchPageIndex	Bool	调用 PageTexts 时是否严格匹配页码
ContactAction	String	连接模式
AttributeNameForContactAction	String	连接模式提取值的来源属性名
LinkTextForContactAction	String	连接字符串

代码示例：

```
var labelele=ctl.GetElementPrperties('label1')  
let res=ctl.GetLabelElementContactSettings(labelele);
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetLabelElementContactSettings>

8.9.57. *根据元素编号（ID）获取元素打印状态

接口：

String GetElementPrintVisibility(string id)

描述：

获取元素的 PrintVisibility 属性

参数：

参数名	是否必填	数据类型	说明
Id	必填	String	文档元素的编号

返回值：

数据类型	说明	备注
String	返回元素的打印可见性属性，包含三个值 Visible, Hidden, None。	

代码示例：

```
let res=ctl.GetElementPrintVisibility("field1");
```

相同效果：

```
var ele=ctl.GetElementProperties("field1");  
var result=ele.PrintVisibility;
```

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetElementPrintVisibility>

8.9.58. *根据当前光标获取元素.NET 对象

接口：

Json CurrentElement(string typename)

描述：

获取当前指定类型的元素的属性。元素存在嵌套时，获取当前元素时需要指定的类型。

参数：

参数名	是否必填	数据类型	说明
typename	非必填	String	元素的类型名称，元素类型名称可参考本文的附录中的结构化元素描述。当为空时，直接获取当前元素。

返回值：

返回 json 格式的结构化元素的属性信息

代码示例：

```
var ele = ctl.CurrentElement();  
var eleoptions = ctl.GetElementProperties(ele);
```

相同效果：

无

结合使用：

ctl.GetElementProperties(ele);根据当前光标处元素判断其元素类型

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。但是接口有实际效果需要编辑器纯在输入

焦点

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=CurrentElement>

8.9.59. *根据表格编号和行列索引获取指定单元格内容

接口:

`string GetTableCellText(string tableID, int rowIndex, int colIndex)`

描述:

获取指定单元格的文本内容。

参数:

参数名	是否必填	数据类型	说明
tableID	必填	字符串	表格编号
rowIndex	必填	Int	表格行序号, 从 0 开始计算的表格行在表格中的序号
colIndex	必填	Int	表格列序号, 从 0 开始计算的表格列在表格中的序号

返回值:

数据类型	说明	备注
String	单元格文本内容	

代码示例:

```
let res= ctl.GetTableCellText(“table1”, 0, 0); //获取编号为 table1 的表格中第一行第一列单元格内容
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetTableCellText>

8.9.60. *根据表格编号和行列索引设置单元格内容

接口:

```
bool SetTableCellText(string tableID, int rowIndex, int colIndex,
string newText)
```

描述:

设置指定单元格的文本内容。

参数:

参数名	是否必填	数据类型	说明
tableID	必填	字符串	表格编号
rowIndex	必填	Int	表格行序号, 从 0 开始计算的表格行在表格中的序号
colIndex	必填	Int	表格列序号, 从 0 开始计算的表格列在表格中的序号
newText	必填	字符串	单元格文本

返回值:

数据类型	说明	备注
Bool	操作是否成功	插入失败方法返回 false

代码示例:

```
let res=ctl.SetTableCellText(“table1”,0,0,“newtext”)
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetTableCellText>

8.9.61. *重置输入域固定宽度

接口:

Bool removeSpecifywidth(object element)

描述:

兼容四代清除输入域的宽度，恢复初始值 0。

参数:

参数名	是否必填	数据类型	说明
element	必填	Object	该接口接受参数包含：字符串类型的元素编号、元素属性对象（GetElementProperties）。

返回值:

数据类型	说明	备注
Bool	操作是否成功	

代码示例:

```
let res=ctl.removeSpecifywidth ("field1");
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

兼容四代-五代调用方式跟四代保持一致。

接口优化:

无

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=removeSpecifywidth>

8. 9. 62. *根据表格编号行列索引给指定单元格插入片段内容

接口:

```
bool InsertContentByTableCellID(string content, string format,
string tableID, int rowIndex, int colIndex, bool clearold)
```

描述:

在指定的单元格内追加带有格式的片段内容。格式文档支持 XML 和 JSON、html、text。

参数:

参数名	是否必填	数据类型	说明
content	必填	字符串	文档内容
format	必填	字符串	文档内容格式，支持 XML、json、text、html
tableID	必填	字符串	表格编号
rowIndex	必填	Int	表格行序号，从 0 开始计算的表格行在表格中的序号

colIndex	必填	Int	表格列序号, 从 0 开始计算的表格列在表格中的序号
clearold	必填	Bool	是否先清除单元格原有内容

返回值:

数据类型	说明	备注
Bool	操作是否成功	

代码示例:

```
let res= ctl.InsertContentByTableCellID('哈哈', "text", "table1",1, 1, true);//对表格第二行第二个单元格覆盖插入内容
```

注意:

插入内容成功后, 无法撤销

相同效果:

无

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

五代独有接口。

接口优化:

无。

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=InsertContentByTableCellID>

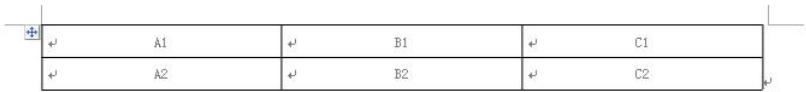
8.9.63. *获取表格中单元格背景编号

接口:

Void ShowBackgroundCellID()

描述:

显示单元格背景编号。执行后会在文档中的单元格内容显示单元格的编号信息。如下图所示：



A1	B1	C1
A2	B2	C2

该编号在打印时不显示，是为了开发者快速了解单元格的编号信息。该编号参考 excel 的单元格编号方式，横向根据英文字母的顺序，纵向根据行的序号从 1 开始。可以用来进行单元格的求和表达式中。

参数：

无

返回值：

无

代码示例：

```
ctl.ShowBackgroundCellID()
```

相同效果：

命令：ctl.DCExecuteCommand('ShowBackgroundCellID',false,null)
文档选项：ctl.DocumentOptions.ViewOptions.ShowBackgroundCellID= true;
ctl.ApplyDocumentOptions();
两种方式效果一致

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。当文档中存在表格时，调用方法即可看到单元格背景编辑，在阅读模式、打印预览模式下无效

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=ShowBackgroundCellID>

8. 9. 64. *根据参数设置当前光标所在单元格垂直水平对齐方式

接口:

Void TableCellAlign(string style)

描述:

设置当前单元格的对齐方式。

参数:

参数名	是否必填	数据类型	说明
Style	必填	字符串	对齐方式的枚举字符串，包含 AlignTopLeft: 顶端左对齐, AlignTopRight: 顶端右对齐, AlignTopCenter: 顶端中间对齐, AlignMiddleLeft: 垂直居中水平左对齐, AlignMiddleRight: 垂直居中水平右对齐, AlignMiddleCenter: 垂直居中水平中间对齐, AlignBottomLeft: 底端左对齐, AlignBottomRight: 底端右对齐, AlignBottomCenter: 底端中间对齐

返回值:

无

代码示例:

```
ctl. TableCellAlign ( “AlignMiddleLeft” )
```

相同效果:

命令: ctl.DCExecuteCommand('AlignMiddleLeft',false,null)ctl.ApplyDocumentOptions();
与对应命令效果一致

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。当编辑器中单元格存在输入焦点方法才有效果，只读模式、阅读模式、打印预览模式下调用无效

兼容接口：

五代独有接口。

接口优化：

无。

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=TableCellAlign>

8. 9. 65. *获取当前光标所在单元格对象

接口：

Object CurrentTableCell()

描述：

获取当前单元格的引用对象。

参数：

无

返回值：

数据类型	说明	备注
Object	单元格引用对象	

代码示例：

```
let res=ctl.CurrentTableCell()
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。打印预览模式下调用该接口无效

兼容接口：

兼容四代接口。

接口优化：

五代调用该方法跟四代一致

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=CurrentTableCell>

8.9.66. *根据表格编号和行/列索引获取对应单元格网格线信息

接口：

Json GetTableCellGridLineInfo(Json parameter)

描述：

获取单元格的网格线设置。

参数：

参数是 json 格式，包含属性

参数名	是否必填	数据类型	说明
tableID	必填	字符串	表格编号
rowIndex	必填	Int	表格行序号，从 0 开始计算的表格行在表格中的序号
colIndex	必填	Int	表格列序号，从 0 开始计算的表格列在表格中的序号

返回值：

返回 json 格式，属性包含

属性	数据类型	说明
AlignToGridLine	String	网格线是否对齐
ColorValue	String	网格线颜色
GridSpanInCM	Float	网格线的跨度，单位 cm
LineStyle	String	网格线的线型
LineWidth	Float	网格线的粗细，单位
Printable	Bool	是否打印网格线

Visible	Bool	是否可见网格线

代码示例:

```
var settings = {
    tableid: "table1",
    rowindex: 0,
    colindex: 0
};
var result = ctl.GetTableCellGridLineInfo(settings);
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。单元格设置了网格线即可获取网格线信息

兼容接口:

兼容四代接口。

接口优化:

五代通过 json 为参数获取指定单元格网格线信息、四代通过单元格对象获取网格线信息

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetTableCellGridLineInfo>

8.9.67. 根据表格编号和行/列索引设置对应单元格网格线信息

接口:

bool SetTableCellGridLineInfo(Json cell, Json settings)

描述:

设置单元格的网格线设置。

参数:

cell 参数是 json 格式，包含属性

参数名	是否必填	数据类型	说明

tableID	必填	字符串	表格编号
rowIndex	必填	Int	表格行序号, 从 0 开始计算的表格行在表格中的序号
colIndex	必填	Int	表格列序号, 从 0 开始计算的表格列在表格中的序号

settings 参数是 json 格式, 包含属性

参数名	是否必填	数据类型	说明
AlignToGridLine	必填	String	网格线是否对齐
ColorValue	必填	String	网格线颜色
GridSpanInCM	必填	Float	网格线的跨度, 单位 cm
LineStyle	必填	String	网格线的线型
LineWidth	必填	Float	网格线的粗细, 单位
Printable	必填	Bool	是否打印网格线
Visible	必填	Bool	是否可见网格线

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
var cell = {
    tableid: "table1",
    rowindex: 0,
    colindex: 1
};
var settings = {
    AlignToGridLine: "True", //文本行对齐到网格线
    ColorValue: "#000000", //网格线颜色
    GridSpanInCM: 1, //网格线之间的宽度
    LineStyle: "Solid", //网格线样式
    LineWidth: 1, //网格线宽度
    Printable: "True", //打印预览是否显示
    Visible: "True", //网格是否显示
};
ctl.SetTableCellGridLineInfo(cell, settings);
```

注意:

该方法第一个参数也可以传单元格 ID、.NET 引用对象、单元格的 NativeHandle 值。

相同效果：

通过 `ctl.cellGridlineDialog` 方法弹出单元格网格线对话框来设置单元格网格线

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。打印预览模式下调用方法预览界面看不到效果，退出预览模式单元格存在网格线

兼容接口：

兼容四代接口。

接口优化：

可以通过四代方式设置单元格网格线、也可以通过 json 指定那个表格下的第几行下的第几个单元格设置网格线

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetTableCellGridLineInfo>

8. 9. 68. *根据参数设置指定表格边框样式

接口：

`bool SetTableBorder (Json table, Json settings)`

描述：

根据表格编号或者.NET 对象或者表格 NativeHandle 值设置表格的边框样式。

参数：

table 参数是 json 格式，包含属性

参数名	是否必填	数据类型	说明
object	必填	字符串	表格编号

settings 参数是 json 格式，包含属性

参数名	是否必填	数据类型	说明

BorderBottom	必填	Bool	是否设置下边框
BorderBottomColorString	必填	String	下边框颜色
BorderLeft	必填	Bool	是否设置左边框
BorderLeftColorString	必填	String	左边框颜色
BorderRight	必填	Bool	是否设置右边框
BorderRightColorString	必填	Bool	右边框颜色
BorderStyle	必填	String	边框的线型, 包含 Solid, Dash, Dot, DashDot, DashDotDot, Custom
BorderTop	必填	Bool	是否设置上边框
BorderTopColorString	必填	String	上边框颜色
BorderWidth	必填	Float	边框的宽度, 单位

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
let Style = {
  BorderBottom: true,
  BorderBottomColorString: "#000000",
  BorderLeft: true,
  BorderLeftColorString: "#000000",
  BorderRight: true,
  BorderRightColorString: "#000000",
  BorderStyle: "Solid",
  BorderTop: true,
  BorderTopColorString: "#000000",
  BorderWidth: 0,
}
ctl.SetTableBorder("table1", Style)
```

相同效果:

通过 `ctl.bordersShadingDialog` 方法弹出表格边框对话框来设置表格边框线

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下无效

兼容接口：

兼容四代接口。

接口优化：

与四代编辑器调用一致

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetTableBorder>

8. 9. 69. *根据参数设置指定单元格边框样式

接口：

bool SetTableCellBorder(Json cell, Json settings)

描述：

根据单元格编号或者.NET 对象或者表格 NativeHandle 值设置对应单元格的边框样式

参数：

cell 参数是 json 格式，包含属性

参数名	是否必填	数据类型	说明
Id	必填	字符串	表格编号
rowIndex	必填	Int	表格行序号, 从 0 开始计算的表格行在表格中的序号
colIndex	必填	Int	表格列序号, 从 0 开始计算的表格列在表格中的序号

settings 参数是 json 格式，包含属性

参数名	是否必填	数据类型	说明
BorderBottom	必填	Bool	是否设置下边框
BorderBottomColorString	必填	String	下边框颜色
BorderLeft	必填	Bool	是否设置左边框

BorderLeftColorString	必填	String	左边框颜色
BorderRight	必填	Bool	是否设置右边框
BorderRightColorString	必填	Bool	右边框颜色
BorderStyle	必填	String	边框的线型，包含 Solid, Dash, Dot, DashDot, DashDotDot, Custom
BorderTop	必填	Bool	是否设置上边框
BorderTopColorString	必填	String	上边框颜色
BorderWidth	必填	Float	边框的宽度，单位

返回值：

数据类型	说明
Bool	操作是否成功

代码示例：

```
let Style = {
  BorderBottom: true,
  BorderBottomColorString: "#000000",
  BorderLeft: true,
  BorderLeftColorString: "#000000",
  BorderRight: true,
  BorderRightColorString: "#000000",
  BorderStyle: "Solid",
  BorderTop: true,
  BorderTopColorString: "#000000",
  BorderWidth: 0,
}
ctl.SetTableCellBorder ("DYG", Style)
```

相同效果：

通过 `ctl.borderShadingcellDialog` 方法弹出单元格边框对话框来设置单元格边框线

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下无效

兼容接口：

兼容四代接口。

接口优化：

与四代编辑器调用一致

示例地址：

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetTableCellBorder>

8.9.70. *获取当前光标所在表格对象

接口：

Object CurrentTable()

描述：

获取当前表格.NET 引用对象。

参数：

无

返回值：

数据类型	说明	备注
Object	表格引用对象	

代码示例：

```
let res=ctl.CurrentTable ()
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口无效

兼容接口：

兼容四代接口。

接口优化:

五代调用该方法跟四代一致

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=CurrentTable>

8. 9. 71. *获取当前光标所在表格行对象

接口:

Object CurrentTableRow()

描述:

获取当前表格行的.NET 引用对象。

参数:

无

返回值:

数据类型	说明	备注
Object	表格行引用对象	

代码示例:

```
let res=ctl.CurrentTableRow ()
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口无效

兼容接口:

兼容四代接口。

接口优化:

五代调用该方法跟四代一致

示例地址:

<https://dcwriter.cn:8765/apiDocument/edit/index.html?apiname=CurrentTableRow>

8.9.72. *根据表格编号获取表格对象

接口:

```
Json GetTableElementById(string id)
```

描述:

获取指定表格的信息。表格属性对话框中封装的属性集合对象

参数:

参数名	是否必填	数据类型	说明
Id	必填	字符串	表格编号

返回值:

返回 json 信息, 包含属性

属性	数据类型	说明
ID	String	表格 ID
Width	Float	表格宽度
Height	Float	表格高度
Visible	Bool	是否可见
RowCount	Int	行的总数
ColumnsCount	Int	列的总数
AllowUserToResizeRows	Bool	是否允许用户调整行高
AllowUserToResizeColumns	Bool	是否允许用户调整列宽
AllowUserDeleteRow	Bool	是否允许用户删除行
AllowUserInsertRow	Bool	是否允许用户插入行
Deleteable	Bool	是否允许用户删除表格
PrintBothBorderWhenJumpPrint	Bool	续打时是否打印边框
CompressOwnerLineSpacing	Bool	压缩表格的间距。默认情况下, 在两个表格之间始终存在一个空隙, 如果这个属性为 true, 表示不显示这个空隙。

ContentReadonly	String	是否只读
AllowUserToResizeEvenInFormViewMode	Bool	是否允许用户在表单模式下调整表格的大小
Attributes	Array	自定义属性信息，每一项都是一个键值对的数据
Style	String	样式信息
EnablePermission	String	是否启用痕迹权限信息

代码示例：

```
let res=ctl.GetTableElementById(“table1”)
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口无效

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://www.dcwritr.cn:8765/apiDocument/edit/index.html?apiname=GetTableElementById>

8.9.73. *根据表格编号修改表格属性值

接口：

```
bool SetTableElementPoperties(string elementID, JsonElement parameter)
```

描述：

设置指定表格的属性信息。当第一个参数为空时或根据第一个参数无法获取到表格，表示设置当前表格信息。

参数：

elementID 参数是字符串格式，包含表格编号。

Parameter 参数是 json 格式，包含属性

属性	数据类型	说明
ID	String	表格 ID
Visible	Bool	是否可见
AllowUserToResizeRows	Bool	是否允许用户调整行高
AllowUserToResizeColumns	Bool	是否允许用户调整列宽
AllowUserDeleteRow	Bool	是否允许用户删除行
AllowUserInsertRow	Bool	是否允许用户插入行
Deleteable	Bool	是否允许用户删除表格
PrintBothBorderWhenJumpPrint	Bool	续打时是否打印边框
CompressOwnerLineSpacing	Bool	压缩表格的间距。默认情况下，在两个表格之间始终存在一个空隙，如果这个属性为 true，表示不显示这个空隙。
ContentReadonly	String	是否只读
AllowUserToResizeEvenInFormViewMode	Bool	是否允许用户在表单模式下调整表格的大小
Attributes	Array	自定义属性信息，每一项都是一个键值对的数据
EnablePermission	String	是否启用痕迹权限信息

返回值：

数据类型	说明
Bool	操作是否成功

代码示例：

```
let para= {
  "ContentReadonly":"True"
}
```


ctl.SetTableElementPoperties ("table1", para)//设置表格只读

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口无效

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SetTableElementPoperties>

8. 9. 74. *根据表格编号和对应行索引修改指定表格行属性

接口:

bool SetTableRowElementPoperties(string elementID, int rowIndex, JsonElement parameter)

描述:

设置指定表格行的属性信息。当第一个参数为空时或根据第一个和第二个参数无法获取到表格行，表示设置当前表格行信息。

参数:

elementID 参数是字符串格式，表示表格编号。

rowIndex 参数是 int 格式，表示表格行的序号。从 0 开始计数。

Parameter 参数是 json 格式，包含属性

属性	数据类型	说明
AllowInsertRowDownUseHotKey	Bool	是否允许通过快捷键在当前行下方插入表

		格行
AllowUserToResizeHeight	Bool	是否允许用户调整行高
CloneType	String	表格行的复制模式
ExpendForDataBinding	Bool	
HeaderStyle	Bool	是否是标题行
NewPage	Bool	是否强制分页
PrintCellBackground	Bool	是否打印单元格背景
PrintCellBorder	Bool	是否打印单元格边框
SpecifyHeight	Float	表格行的高度。当为负数时，表示表格行的高度固定，在内容较多时不会自动撑高。
CanSplitByPageLine	Bool	是否允许表格行跨页显示

返回值：

数据类型	说明
Bool	操作是否成功

代码示例：

```
let para= {
  "ContentReadonly": "True"
}
ctl. SetTableRowElementPoperties ("table1", 2, para)
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口无效

兼容接口：

五代独有接口。

接口优化：

无

示例地址:

<https://www.dcwritr.cn:8765/apiDocument/edit/index.html?apiname=SetTableRowElementPoperties>

8. 9. 75. *根据表格编号和对应行索引列索引修改指定单元格属性

接口:

```
bool SetTableCellElementPoperties(string elementID, int rowIndex,
int columnIndex, JsonElement parameter)
```

描述:

设置指定单元格的属性信息。当第一个参数为空时或根据第一个、第二个、第三个参数无法获取到表格行，表示设置当前表格行信息。

参数:

elementID 参数是字符串格式，表示表格编号。

rowIndex 参数是 int 格式，表示表格行的序号。从 0 开始计数。

columnIndex 参数是 int 格式，表示表格列的序号。从 0 开始计数。

parameter 参数是 json 格式，包含属性

属性	数据类型	说明
AutoFixFontSizeMode	Bool	单元格自动缩小字体的模式
BorderPrintedWhenJumpPrint	Bool	续打时是否打印边框
CloneType	String	单元格的复制模式
ColSpan	Int	跨列数
MirrorViewForCrossPage	Bool	
MoveFocusHotKey	String	移动焦点快捷键
RowSpan	Int	跨行数
SlantSplitLineStyle	String	
GridLine	Json	
BorderRenderVisibility	String	
ContentRenderVisibility	String	

Style	Json	格式信息

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
let para= {
  "ContentReadonly":"True"
}
ctl.SetTableCellElementPoperties("表格1", 2, 3, para)
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口无效

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritr.cn:8765/apiDocument/edit/index.html?apiname=SetTableCellElementPoperties>

8.9.76. *根据参数获取指定表格下自定义属性列表

接口:

Array GetTableAttribute (object element)

描述:

获取表格的自定义属性信息。返回的自定义属性信息如下:



该返回值是一个数值，每个数组项是自定义的属性名称和属性对应的值。

参数：

该接口支持三种类型参数

- 1. 参数是字符串格式，表示表格编号。
- 2. 参数是 json 格式，属性是 id，表示表格编号。
- 3. 参数是后台引用对象
- 4. 参数是 NativeHandle 对象。

返回值：

数据类型	说明
Array	数组形式的自定义属性信息

代码示例：

```
var table = ctl.CurrentTable();  
var result = ctl.GetTableAttribute(table);
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口都能获取自定义属性列表

兼容接口：

兼容四代接口。

接口优化：

五代该接口重新定位，修改成获取表格自定义属性，四代该接口是获取表格属性（五代对应接口为：GetTableElementPoperties）

示例地址：

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=GetTableRowAttribute>

8. 9. 77. *根据参数获取表格中指定表格行自定义属性

接口:

Array GetTableRowAttribute (object element)

描述:

获取表格行的自定义属性信息。返回的自定义属性信息如下:



该返回值是一个数值，每个数组项是自定义的属性名称和属性对应的值。

参数:

element 参数是 json 格式，包含属性

属性	数据类型	说明
tableid	String	表格编号
rowindex	Int	行的序号

返回值:

数据类型	说明
Array	数组形式的自定义属性信息

代码示例:

```

var row = ctl.CurrentTableRow();
var result = ctl.GetTableRowAttribute(row);

```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口都能获取自定义属性列表

兼容接口:

兼容四代接口。

接口优化:

五代该接口重新定位，修改成获取表格自定义属性，四代该接口是获取表格属性（五代对应接口为：GetTableRowElementPoperties）

示例地址:

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetTableRowAttribute>

8.9.78. *根据参数获取表格中指定单元格自定义属性

接口:

Array GetTableCellAttribute (object element)

描述:

获取单元格的自定义属性信息。返回的自定义属性信息如下:



该返回值是一个数值，每个数组项是自定义的属性名称和属性对应的值。

参数:

element 参数是 json 格式，包含属性

属性	数据类型	说明
tableid	String	表格编号
rowindex	Int	行的序号
colindex	Int	列的序号

返回值:

数据类型	说明
Array	数组形式的自定义属性信息

代码示例:

```
var cell = ctl.CurrentTableCell();  
var result = ctl.GetTableCellAttribute(cell);
```

相同效果:

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口都能获取自定义属性列表

兼容接口：

兼容四代接口。

接口优化：

五代该接口重新定位，修改成获取表格自定义属性，四代该接口是获取表格属性（五代对应接口为：GetTableCellElementProperties）

示例地址：

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=GetTableCellAttribute>

8.9.79. *根据参数设置表格自定义属性值

接口：

```
Bool SetTableAttribute (object element, json settings)
```

描述：

设置表格的自定义属性信息。

参数：

第一个参数支持三种类型参数

1. 参数是字符串格式，表示表格编号。
2. 参数是 json 格式，属性是 id，表示表格编号。
3. 参数是后台引用对象
4. 参数是 NativeHandle 对象。

第二个参数就是自定义的属性信息，格式如下：

```
var opts = {  
    属性 1: "内容 1",  
};
```


可以同时设置多个。

返回值：

数据类型	说明
Bool	操作是否成功

代码示例：

```
var table = ctl.CurrentTable();
var opts = {
    t1: "TDDDDDD",
    t2: "valuevalue",
    tA: "myAA"
};
var result = ctl.SetTableAttribute(table, opts);
```

注意：

调用该接口设置表格自定义属性，会先把原先表格中设置的自定义属性值先清空，然后再重新设置方法调用的自定义属性值。建议通过接口（GetTableAttribute）先获取表格本身自定义属性，在这个基础上进行追加。

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口无效

兼容接口：

兼容四代接口。

接口优化：

五代该接口重新定位，修改成设置表格自定义属性，四代该接口是设置表格属性（五代对应接口为：SetTableElementPoperties）

示例地址：

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SetTableAttribute>

8.9.80. *根据参数设置表格行自定义属性值

接口:

Bool SetTableRowAttribute (object element, json settings)

描述:

设置表格行的自定义属性信息。

参数:

第一个参数支持三种类型参数

- 1.参数是字符串格式，表示表格行编号。
- 2.参数是 json 格式，属性是 id，表示表格行编号。
- 3.参数是后台引用对象
- 4.参数是 NativeHandle 对象

第二个参数是自定义属性信息，格式如下:

```
var opts = {  
    属性 1: "内容 1",  
};
```

可以同时设置多个。

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
var row = ctl.CurrentTableRow();  
var opts = {  
    R1: "rDDDDDD",  
    r2: "valuevalue",  
    rA: "myAA"  
};  
var result = ctl.SetTableRowAttribute(row, opts);
```

注意:

调用该接口设置表格行自定义属性，会先把原先表格行中设置的自定义属性值先清空，然后再重新设置方法调用的自定义属性值。建议通过接口 (GetTableRowAttribute) 先获取表格行本身自定义属性，在这个基础上进行追加。

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口无效

兼容接口:

兼容四代接口。

接口优化:

五代该接口重新定位，修改成设置表格行自定义属性，四代该接口是设置表格行属性（五代对应接口为：SetTableRowElementProperties）

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SetTableRowAttribute>

8.9.81. *根据参数设置单元格自定义属性

接口:

Bool SetTableCellAttribute (object element, json settings)

描述:

设置单元格的自定义属性信息。

参数:

第一个参数支持三种类型参数

1. 参数是字符串格式，表示单元格编号。
2. 参数是 json 格式，属性是 id，表示单元格编号。
3. 参数是后台引用对象
4. 参数是 NativeHandle 对象

第二个参数是自定义属性信息，格式如下：

```
var opts = {  
    属性 1: "内容 1",  
};
```

可以同时设置多个。

返回值:

数据类型	说明
Bool	操作是否成功

代码示例：

```
var cell = ctl.CurrentTableCell();
var opts = {
    c1: "cDDDDDD",
    c2: "valuevalue",
    cA: "myAA"
};
var result = ctl.SetTableCellAttribute(cell, opts);
```

注意：

调用该接口设置表格行自定义属性，会先把原先表格行中设置的自定义属性值先清空，然后再重新设置方法调用的自定义属性值。建议通过接口（GetTableRowAttribute）先获取表格行本身自定义属性，在这个基础上进行追加。

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口无效

兼容接口：

兼容四代接口。

接口优化：

五代该接口重新定位，修改成设置单元格自定义属性，四代该接口是设置单元格属性（五代对应接口为：SetTableCellElementProperties）

示例地址：

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SetTableCellAttribute>

8.9.82. *表格列超出正文边界修改所以列宽使其不超出正文边界

接口：

Bool AutoFixTableWidth (object element)

描述:

自动修复表格列宽使其自适应容器宽度。主要运用场景就是页面设置的修改导致表格宽度超出了正文宽度之后调用。

参数:

该接口支持三种类型参数

- 1. 参数是字符串格式，表示表格编号。
- 2. 参数是 json 格式，属性是 id，表示表格编号。
- 3. 参数是后台引用对象
- 4. 参数是 NativeHandle 对象。

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
var result = ctl.AutoFixTableWidth ("table1");
```

注意:

表格对应的表格列没有超出正文边界该接口没有实际效果，当超出正文边界会把超出的多余宽度在各列中进行平均修改列宽。

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口都能把超出部分进行修改

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetTableCellAttribute>

8. 9. 83. *根据表格参数和指定列索引设置指定列隐藏

接口:

Bool SetTableColumnVisible (object element, int columnIndex, bool visible)

描述:

兼容第四代设置表格列是否隐藏

参数:

第一个参数支持三种类型参数

- 1. 参数是字符串格式，表示表格编号。
- 2. 参数是 json 格式，属性是 id，表示表格编号。
- 3. 参数是后台引用对象
- 4. 参数是 NativeHandle 对象。

第二个参数表示列的序号，从左到右从 0 开始计算。

第三个参数表示是否显示或隐藏

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
var result = ctl.SetTableColumnVisible("table1", 1, true);
```

注意:

当表格列中有单元格设置了合并、拆分等操作，建议不要调用该接口。

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口都能进行隐藏

兼容接口:

兼容四代接口。

接口优化:

与四代编辑器调用效果一致

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SetTableColumnVisible>

8.9.84. *获取选中单元格属性值和单元格所在的表格属性值

接口:

Json GetSelectTableAndCell()

描述:

获取当前选择的表格和单元格。

返回信息如下图所示:

```
> document.WriterControl.GetSelectTableAndCell();
(25) [{"TableID": null, "ID": null, "Width": "306.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "318.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "372.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "308.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "315.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "253.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "306.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "318.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "372.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "308.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "315.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "253.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "306.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "318.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "372.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "877.37494", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "315.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "253.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "306.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "318.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "372.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "308.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "315.4583", "Height": "74", "ValueBinding": null, ...},
{"TableID": null, "ID": null, "Width": "253.4583", "Height": "74", "ValueBinding": null, ...},
{"ID": null, "Width": "1874.7498", "Height": "658", "Visible": true, "RowCount": 8, ...}
length: 25
```

参数:

无

返回值:

数据类型	说明
Json	Json 形式的选择的表格和单元格的信息，

代码示例：

```
var result = ctl.GetSelectTableAndCell();
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式下调用接口能获取选中单元格属性值、打印预览模式下调用该接口无效

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetSelectTableAndCell>

8.9.85. *根据表格编号修改所有表格行高度

接口：

Bool SetTableHeight (string tableid, object newHeight)

描述：

设置表格行高度。

参数：

参数名	是否必填	数据类型	说明
tableid	必填	字符串	表格编号
newHeight	必填	Object	支持两种： 1. 字符串类型，带 px 单位或不带单位都可

			以。 2. Float 类型，单位是像素 注意：设置高度值太少无效，内部会把值转成 1/300 英寸

返回值：

数据类型	说明
Bool	操作是否成功

代码示例：

```
var result = ctl.SetTableHeight ("table1", 25);
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式下调用接口能修改表格行高。阅读模式、打印预览模式下调用该接口无效

兼容接口：

兼容四代接口。

接口优化：

与四代编辑器调用一致

示例地址：

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SetTableHeight>

8. 9. 86. *根据表格编号和行索引、列索引获取指定单元格.NET 对象值

接口：

Object GetTableCell(string id, string rowIndex, string ColIndex)

描述：

获取给定表格内的指定行列的单元格后台引用.NET 对象。

参数：

参数名	是否必填	数据类型	说明
tableid	必填	字符串	表格编号
rowIndex	必填	Int	行的序号
ColumnIndex	必填	Int	列的序号

返回值：

数据类型	说明
Object	后台引用.NET 对象

代码示例：

```
var cell = ctl.GetTableCell("table1", 0, 0);
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口都能获取到指定单元格对象

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=GetTableCell>

8.9.87. *根据表格编号和行索引获取指定表格行.NET 对象值

接口：

```
Object GetTableRow(string id, string rowIndex)
```

描述：

获取给定表格行的后台引用.NET 对象。

参数：

参数名	是否必填	数据类型	说明
tableid	必填	字符串	表格编号
rowIndex	必填	Int	行的序号

返回值：

数据类型	说明
Object	后台引用.NET 对象

代码示例：

```
var row = ctl. GetTableRow("table1", 0);
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口都能获取到指定单元格对象

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=GetTableRow>

8.9.88. *根据表格编号和列索引获取指定列.NET 对象值（暂不开放）

接口：

Object CurrentTableColumn(string id, string columnIndex)

描述：

获取给定表格行的后台引用.NET 对象。

参数：

参数名	是否必填	数据类型	说明
tableid	必填	字符串	表格编号
columnIndex	必填	Int	列的序号

返回值：

数据类型	说明
Object	后台引用.NET 对象

代码示例：

```
var column = ctl.CurrentTableColumn("table1", 0);
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口都能获取到指定单元格对象

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=CurrentTableColumn>

8.9.89. *修改选中的单元格边框样式

接口：

```
bool SetSelectTableCellBorder (json settings)
```

描述：

为选择的单元格设置边框。支持选择多个设置。

参数:

settings 参数是 json 格式, 包含属性

参数名	是否必填	数据类型	说明
BorderBottom	必填	Bool	是否设置下边框
BorderBottomColorString	必填	String	下边框颜色
BorderLeft	必填	Bool	是否设置左边框
BorderLeftColorString	必填	String	左边框颜色
BorderRight	必填	Bool	是否设置右边框
BorderRightColorString	必填	Bool	右边框颜色
BorderStyle	必填	String	边框的线型, 包含 Solid, Dash, Dot, DashDot, DashDotDot, Custom
BorderTop	必填	Bool	是否设置上边框
BorderTopColorString	必填	String	上边框颜色
BorderWidth	必填	Float	边框的宽度, 单位

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
let Style = {
Style: {
  BorderBottom: true,
  BorderBottomColorString: "#000000",
  BorderLeft: true,
  BorderLeftColorString: "#000000",
  BorderRight: true,
  BorderRightColorString: "#000000",
  BorderStyle: "Solid",
  BorderTop: true,
  BorderTopColorString: "#000000",
  BorderWidth: 1,
}
}
ctl.SetSelectTableCellBorder (Style)
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式调用接口能修改选中单元格边框样式。打印预览模式下调用该接口无效

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SetSelectTableCellBorder>

8.9.90. *根据表格编号判断表格是否跨页显示

接口:

bool IsRowHasNewPage(string id)

描述:

判断指定的表格的行是否存在换页。表格中的任意一行存在跨页时，就返回 true。

参数:

参数名	是否必填	数据类型	说明
Id	必填	String	表格编号

返回值:

数据类型	说明	备注
Bool	是否存在换页	

代码示例:

```
let res=ctl. IsRowHasNewPage ("table1");
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口都能判断表格是否跨页显示

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=IsRowHasNewPage>

8.9.91. *根据表格编号修改所有表格行高度进行平均分配

接口:

bool AverageTableRows(string id)

描述:

平均分布各行的高度。

参数:

参数名	是否必填	数据类型	说明
Id	必填	String	表格编号

返回值:

数据类型	说明	备注
Bool	操作是否成功	

代码示例:

```
let res=ctl.AverageTableRows ("table1");
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口都能平均分配表格行高

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=AverageTableRows>

8. 9. 92. *根据表格编号修改所有表格列宽度进行平均分配

接口:

bool AverageTableColumns(string id)

描述:

平均分布各列。

参数:

参数名	是否必填	数据类型	说明
Id	必填	String	表格编号

返回值:

数据类型	说明	备注
Bool	操作是否成功	

代码示例:

```
let res=ctl.AverageTableColumns("table1");
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口都能平均分配表格列宽

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=AverageTableColumns>

8. 9. 93. *根据表格编号和行索引、列索引对指定单元格赋值纯文本内容

接口:

Bool SetTableCellTextExt(object element, int rowIndex, int colIndex, string newText)

描述:

设置指定单元格的内容。

参数:

参数名	是否必填	数据类型	说明
tableid	必填	Object	该接口支持三种类型参数 1. 参数是字符串格式，表示表格编号。 2. 参数是 json 格式，属性是 id，表示表格编号。 3. 参数是后台引用对象 4. 参数是 NativeHandle 对象。
rowIndex	必填	Int	行的序号

ColumnIndex	必填	Int	列的序号
newText	必填	String	文本内容

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
var result = ctl.SetTableCellTextExt(table1", 1, 2, "哈哈")
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式下调用接口无效。打印预览模式下调用该接口能对指定单元格赋值

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SetTableCellTextExt>

8. 9. 94. *根据当前光标所有行设置表格行高亮度颜色

接口:

Bool FocusTableRow (string color)

描述:

设置当前单元行的背景颜色。

参数:

参数名	是否必填	数据类型	说明
color	必填	String	颜色字符串

返回值：

数据类型	说明
Bool	操作是否成功

代码示例：

```
var result = ctl.FocusTableRow("red")
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口无效

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=FocusTableRow>

8.9.95. *根据表格编号和行索引、列索引设置高亮度颜色

接口：

```
Bool HighlightTableRowColumn (object element, int rowIndex, int colIndex, string colorString, bool cancelUnset)
```

描述：

使用指定的背景色高亮化表格指定的行或列

参数：

参数名	是否必填	数据类型	说明
tableid	必填	Object	该接口支持三种类型参数 1. 参数是字符串格式，表示表格编号。 2. 参数是 json 格式，属性是 id，表示表格编号。 3. 参数是后台引用对象 4. 参数是 NativeHandle 对象。
rowIndex	必填	Int	行的序号
colorString	必填	Int	列的序号
cancelUnset	必填	Bool	未亮化的行或列是否恢复背景色为透明色

返回值：

数据类型	说明
Bool	操作是否成功

代码示例：

```
var ri = document.getElementById("rowindex");
var rowindex = parseInt(ri.selectedOptions[0].value);
var ci = document.getElementById("colindex");
var colindex = parseInt(ci.selectedOptions[0].value)
var colortxt = document.getElementById("colortext").value;
var reset = document.getElementById("chkunset").checked;
console.log(rowindex, colindex, colortxt, reset);
ctl.HighlightTableRowColumn("table1", rowindex, colindex, colortxt, reset);
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口都能设置高亮度颜色

兼容接口：

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=HighlightTableRowColumn>

8. 9. 96. *根据单元格参数修改单元格内边距

接口:

bool SetTableCellPadding (object element, json settings)

描述:

设置单元格的内边距。

参数:

第一个参数是单元格的.NET 引用对象。为空的话尝试取当前单元格，一般是接口 CurrentTableCell 的返回值。

第二个参数是 settings，参数是 json 格式，包含属性

参数名	是否必填	数据类型	说明
paddingLeft	必填	Float	左内边距的值，单位
paddingTop	必填	Float	上内边距的值，单位
paddingRight	必填	Float	右内边距的值，单位
paddingBottom	必填	Float	下内边距的值，单位

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
var opts = {
    paddingleft: 55,
    paddingtop: 55,
    paddingright: 55,
    paddingbottom: 55,
};
```

```
var cell = ctl.CurrentTableCell();
ctl.SetTableCellPadding(cell, opts);
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式下调用接口能修改单元格内边距。打印预览模式下调用该接口无效

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SetTableCellPadding>

8.9.97. *根据选中的单元格设置单元格网格线

接口:

bool SetSelectTableCellGridLineInfo (json settings)

描述:

设置选择的一个或多个单元格的网格线。

参数:

settings 参数是 json 格式，包含属性

参数名	是否必填	数据类型	说明
AlignToGridLine	必填	String	网格线是否对齐
ColorValue	必填	String	网格线颜色
GridSpanInCM	必填	Float	网格线的跨度，单位 cm

LineStyle	必填	String	网格线的线型
LineWidth	必填	Float	网格线的粗细，单位
Printable	必填	Bool	是否打印网格线
Visible	必填	Bool	是否可见网格线

返回值：

数据类型	说明
Bool	操作是否成功

代码示例：

```
var settings = {
    AlignToGridLine: "True", //文本行对齐到网格线
    ColorValue: "#000000", //网格线颜色
    GridSpanInCM: 1, //网格线之间的宽度
    LineStyle: "Solid", //网格线样式
    LineWidth: 1, //网格线宽度
    Printable: "True", //打印预览是否显示
    Visible: "True", //网格是否显示
};
ctl1.SetSelectTableCellGridLineInfo(settings);
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式下调用接口能修改单元格网格线。打印预览模式下调用该接口无效

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetSelectTable>

[CellGridLineInfo](#)

8. 9. 98. *根据选中的单元格获取选中单元格属性值

接口:

Array GetSelectTableCells()

描述:

获取当前选择的单元格数组。

返回信息如下图所示:

```
> document.WriterControl.GetSelectTableCells();
(4) [{"ID": null, "Width": 940.375, "Height": 111, "Visible": true, "Attributes": null, ...}, {"ID": null, "Width": 940.375, "Height": 111, "Visible": true, "Attributes": null, ...}, {"ID": null, "Width": 940.375, "Height": 111, "Visible": true, "Attributes": null, ...}, {"ID": null, "Width": 940.375, "Height": 111, "Visible": true, "Attributes": null, ...}]
length: 4
```

参数:

无

返回值:

数据类型	说明
Json	Json 形式的选择的单元格的信息,

代码示例:

```
var result = ctl.GetSelectTableAndCell();
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式下调用接口能获取单元格属性值。打印预览模式下调用该接口无效

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=GetSelectTableCells>

8. 9. 99. *根据表格编号和行索引、列索引对指定单元格拆分出同行中设置单元格网格线产生相同数量的单元格

接口:

Object SplitTableCellForRowLine(string id, string rowIndex, string ColIndex, bool isAllCell)

描述:

根据单元格的网格线数量拆分单元格。

拆分前如下图所示:

		签名
	1111111111111111111111111111	放大
	1111111111111111111111111111	
	1111111111111111111111111111	
	1111111111111111111111111111	
	11111	
	2222222222222222222222222222	放大
	2222222222222222222222222222	
	222222222222222222222222	
	3333333333333333333333333333	放大
	3333333333333333333333333333	
	3333333333333333333333333333	
	3333333333333333333333333333	
	333333333333333333333333	

拆分后如下图所示:

用该接口无效

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SplitTableCellForRowLine>

8. 9. 100. *根据元素对象进行焦点切换

接口：

Bool FocusAdjacent(string where, object element)

描述：

移动输入焦点到指定元素。

参数：

参数名	是否必填	数据类型	说明
where	必填	String	参数包含：beforebegin、afterbegin、beforeend、Afterend。 当根据第二个参数查找到输入域时，beforebegin 表示输入域外部前面、afterbegin 表示输入域内部前面、beforeend 表示输入域内部最后、Afterend 表示输入域外部后面。 当根据第二个参数查找不到输入域时，beforebegin 表示文档开头、afterbegin 表示文档开头、beforeend 表示文档结尾、Afterend 表示文档结尾。
element	非必填	objetc	元素字符串编号 .net 对象 Handle 对象

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
var result = ctl.FocusAdjacent('beforebegin', null);
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口无效

兼容接口:

兼容四代接口。

接口优化:

与四代接口调用一致, 但是进行了优化, 当第二个参数不存在, 焦点进行文档开头、文档结尾进行切换

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=FocusAdjacent>

8.9.101. *根据元素类型获取文档中所有相同类型的元素列表

接口:

Array GetElementsByTypeName(string elementTypeName)

描述:

获取指定类型的所有元素。

参数:

参数名	是否必填	数据类型	说明
elementTypeName	必填	String	结构化元素类型的名称, 包含

			xtextmediaelement：多媒体元素 xtextnewmedicalexpressionelement：医学表达式元素 xtextinputfieldelement：输入域元素 xtextradioboxelement：单选框元素 xtextcheckboxelement：复选框元素 xtextlabelement：标签文本元素 xtextnewbarcodeelement：条形码元素 xtexttdbarcodeelement：二维码元素 xtexthorizontallineelement：水平线元素 xtextimageelement：图片元素 xtextpageinfoelement：页码元素 xtextbuttonelement：按钮元素 xtexttableelement：表格元素

返回值：
以数组的形式返回所有相同类型的元素列表

代码示例：

```
let result = ctl.GetElementsByTypeName("xtextinputfieldelement")
```

相同效果：
无

结合使用：
无

接口类型：
纯前端接口，不经过服务端

有无限制：
编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口无效

兼容接口：
五代独有接口。

接口优化：

无

示例地址:

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetElementsByTypeName>

8.9.102. *获取当前光标处元素类型

接口:

String GetCurrentElementTypeName()

描述:

获取当前光标所在的元素的类型名称。

参数:

无

返回值:

数据类型	说明
String	结构化元素的类型的名称，包含 xtextmediaelement 多媒体元素、xtextnewmedicalexpressionelement 医学表达式元素、xtextinputfieldelement 输入域元素、xtextradioboxelement 单选框元素、xtextcheckboxelement 复选框元素、xtextlabelelement 标签文本元素、xtextnewbarcodeelement 条形码元素、xtexttdbarcodeelement 二维码元素、xtexthorizontallineelement 水平线元素、xtextimageelement 图片元素、xtextpageinfoelement 页码元素、xtextbuttonelement 按钮元素、xtexttableelement 表格元素、xtextsignimageelement 电子签名元素。

代码示例:

```
let result = ctl.GetCurrentElementTypeName()
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口无效

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetCurrentElementTypeName>

8.9.103. *根据下拉项的 Value 值选择对应的下拉项显示

接口：

Bool SetFieldDropListItemByValue (object element, string valuestring)

描述：

指定下拉列表的默认选择项

参数：

参数名	是否必填	数据类型	说明
Element	必填	Object	该接口支持三种类型参数 1. 参数是字符串格式，表示元素编号。 2. 参数是 json 格式，属性是 id，表示元素编号。 3. 参数是后台引用对象 4. 参数是 NativeHandle 对象。
valuestring	必填	String	下拉项的值

返回值：

数据类型	说明
Bool	操作是否成功

代码示例：

```
let result = ctl.SetFieldDropListItemByValue('field1', '数值1')
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口无效

兼容接口:

兼容四代接口。

接口优化:

与四代编辑器调用一致

示例地址:

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetFieldDropListItemByValue>

8.9.104. *根据单、复选框编号控制是否勾选

接口:

bool SetElementCheckedByID(string id, bool newChecked)

描述:

设置单 / 复选框的勾选状态，如果没找到元素则返回 false。功能和 SetElementChecked 相同。

参数:

参数名	是否必填	数据类型	说明
Id	必填	String	单复选框文档元素的编号
newChecked	必填	Bool	是否勾选

返回值:

数据类型	说明	备注

Bool	返回操作是否成功	

代码示例:

```
let res=ctl.SetElementCheckedByID("chk1",true);
```

相同效果:

```
ctl.SetElementChecked("chk1",true)
```

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口无效

兼容接口:

兼容四代接口。

接口优化:

与四代编辑器调用一致

示例地址:

<https://www.dcwritr.cn:8765/apiDocument/edit/index.html?apiname=SetElementCheckedByID>

注意:

当用户登录之后，调用该接口会产生痕迹。

8.9.105. *获取相同类型元素状态改变元素列表

接口:

```
Array getModifiedElements(string typename)
```

描述:

获取所有修改的指定类型的元素。

参数:

参数名	是否必填	数据类型	说明
elementTypename	必填	String	结构化元素类型的名称，包含

			xtextmediaelement: 多媒体元素 xtextnewmedicalexpressionelement: 医学表达式元素 xtextinputfieldelement: 输入域元素 xtextradioboxelement: 单选框元素 xtextcheckboxelement: 复选框元素 xtextlabelement: 标签文本元素 xtextnewbarcodeelement: 条形码元素 xtexttdbarcodeelement: 二维码元素 xtexthorizontallineelement: 水平线元素 xtextimageelement: 图片元素 xtextpageinfoelement: 页码元素 xtextbuttonelement: 按钮元素 xtexttableelement: 表格元素 xtextsubelement: 病程元素

返回值:

以数组的形式返回所有Modified值为true的相同元素列表

代码示例:

```
let result = ctl.getModifiedElements("xtextinputfieldelement");//只读下无效
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口无效

兼容接口:

兼容四代接口。

接口优化:

功能与四代效果不一样

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=getModifiedElements>

8.9.106. *根据文本标签对象修改文本标签的连接对象

接口:

```
void SetLabelElementContactSettings(Json label, Json parameter)
```

描述:

设置标签的连接信息。它和 GetLabelElementContactSettings 是相反的接口

参数:

参数名	是否必填	数据类型	说明
label	必填	Object	该接口支持三种类型参数 1. 参数是字符串格式,表示元素编号。 2. 参数是 json 格式,属性是 id,表示元素编号。 3. 参数是后台引用对象 4. 参数是 NativeHandle 对象。
parameter	必填	Json	属性信息如下:
+StrictMatchPageIndex	必填	Bool	调用 PageTexts 时是否严格匹配页码
+ContactAction	必填	String	连接模式
+AttributeNameForContactAction	必填	String	连接模式提取值的来源属性名
+LinkTextForContactAction	必填	String	连接字符串

返回值:

bool型: 设置成功返回true、设置失败返回false

代码示例:

```
let element = ctl.GetElementProperties(ctl.CurrentElement())
var options = {
    //ContactAction: "Normal", //连接模式 AttributeNameForContactAction: "科室",
    //连接模式提取值的来源属性名 LinkTextForContactAction: "-" //连接字符串 };
    LinkTextForContactAction: "-"
}
ctl.SetLabelElementContactSettings(element, options);
```

相同效果:

无

结合使用:

ctl.RefreshInnerView(true);调用该接口，需要执行刷新方法才有效果

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口无效

兼容接口:

兼容四代接口。

接口优化:

与四代编辑器调用一致

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SetLabelElementContactSettings>

8.9.107. *根据元素对象获取所在文档坐标位置

接口:

Json GetAbsBoundsInDocument(object element)

描述:

获取元素位置兼容四代接口。

参数:

参数名	是否必填	数据类型	说明

label	必填	Object	该接口支持三种类型参数 1. 参数是字符串格式，表示元素编号。 2. 参数是 json 格式，属性是 id，表示元素编号。 3. 参数是后台引用对象 4. 参数是 NativeHandle 对象。

返回值：

返回 json 格式，属性包含：

属性	数据类型	说明
Left	Float	
Top	Float	
AbsLeft	Float	
AbsTop	Float	
Width	Float	
Height	Float	
Right	Float	
Bottom	Float	
TopInOwnerPage	Float	
LeftInOwnerPage	Float	

代码示例：

```
let res=ctl.GetAbsBoundsInDocument ("chk1");
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口无效

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=GetAbsBoundsInDocument>

8.9.108. CReSignature

接口:

bool CReSignature(Json parameter, function)

描述:

CA 签名与四代兼容接口。第二个参数表示回调函数。

参数:

参数名	是否必填	数据类型	说明
id	必填	String	如果为空，则以 signImg 为前缀，以 yyyyMMddHHmmss 格式的当前时间为后缀，自动添加 ID
base64img	必填	String	签名使用的图片的 base64 数据
scope	必填	String	签名范围，包含 None 无设置、Parent 当前元素的父元素、InputField 当前输入域、TableCell 当前单元格、TableRow 当前表格行、Table 当前表格、Section 当前病程、Body 当前正文、Document 当前文档
userid	必填	String	用户编号
username	必填	String	用户姓名
clientname	必填	String	客户端名称
role	必填	String	自定义属性 role 对应的值
Width	必填	Float	签名元素的宽度

height	必填	Float	签名元素的高度
imageinfrontoftext	必填	String	图片与文字的位置关系，包含 Normal 无设置、UnderText 图片位于文字下方、InFrontOfText 图片位于文字上方。
providername	必填	String	签名厂商，1 表示软签，2 表示江苏诺安科技有限公司。
bstrpfx	必填	String	自定义属性 PER 对应的值
Bstrpwd	必填	String	自定义属性 PWD 对应的值
resign	必填	Bool	是否是重新签名
specifycontainer	必填		

返回值：

数据类型	说明
Bool	操作是否成功

代码示例：

```
var obj = {
  "id": "img1",
  "base64Img": imgaebase64,
  "scope": "InputField",
  "userID": "zhangsan",
  "userName": "zhangsan",
  "imageInFrontOfText": "false",
  "clientName": "(local)",
  "width": "80",
  "height": "80",
  "providerName": "1" //1:软签 2: 诺安签名
};

var result = ctl.CAResignature(obj, function () {
  console.log("签名成功")
  console.log(result, "result");
});
```

8. 9. 109. *根据元素对象刷新元素

接口：

Bool EditorRefreshContainerView(object element)

描述:

刷新指定的容器元素, 当调用了某个接口对元素进行修改, 没有其作用可以调用下该接口

参数:

参数名	是否必填	数据类型	说明
Element	必填	Object	该接口支持三种类型参数 1. 参数是字符串格式, 表示元素编号。 2. 参数是 json 格式, 属性是 id, 表示元素编号。 3. 参数是后台引用对象 4. 参数是 NativeHandle 对象。

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
let result = ctl.EditorRefreshContainerView('field1')
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口无效

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=EditorRefreshContainerView>

8. 9. 110. *根据元素对象设置元素样式

接口:

Bool EditorSetContentStyle(object element, json style)

描述:

设置指定容器元素的样式

参数:

参数名	是否必填	数据类型	说明
Element	必填	Object	该接口支持三种类型参数 1. 参数是字符串格式，表示元素编号。 2. 参数是当前光标处元素对象
Style	必填	Json	样式属性信息，包含
+colorstring	必填	String	前景色
+fontname	必填	String	字体名称

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
let result = ctl.EditorSetContentStyle(ctl.CurrentTableCell(), { "ColorString": 'red' });
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口可以修改指定元素样式

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=EditorSetContentStyle>

8.9.111. *根据单、复选框对象修改单、复选框 Text 和 Value 值

接口：

Bool SetCheckboxOrRadioTextAndValue(object element, json options)

描述：

设置单选框或复选框的文本和值

参数：

参数名	是否必填	数据类型	说明
Element	必填	Object	该接口支持三种类型参数 1. 参数是字符串格式，表示元素编号。 2. 参数是.NET 对象。
options	必填	Json	属性信息，包含
+ text	必填	String	文本
+ value	必填	String	值

返回值：

数据类型	说明
Bool	操作是否成功

代码示例：

```
var opt = { "text": "修改的text值", "value": "修改的value值", };
let res = ctl.SetCheckboxOrRadioTextAndValue("radio1", opt);
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式下调用接口都能修改指定单、复选框的文本和数字。打印预览模式下调用该接口无效

兼容接口:

兼容四代接口。

接口优化:

与四代编辑器调用一致

示例地址:

<https://www.dewriter.cn:8765/apiDocument/edit/index.html?apiname=SetCheckboxOrRadioTextAndValue>

8. 9. 112. *根据元素对象修改元素状态为未修改状态

接口:

Bool ResetModifiedSpecifyElement (object element)

描述:

重置指定的元素的修改状态为未修改状态

参数:

参数名	是否必填	数据类型	说明

Element	必填	Object	该接口支持三种类型参数 1. 参数是字符串格式，表示元素编号。 2. 参数是 json 格式，属性是 id，表示元素编号。 3. 参数是后台引用对象 4. 参数是 NativeHandle 对象。

返回值：

数据类型	说明
Bool	操作是否成功

代码示例：

```
let res = ctl.ResetModifiedSpecifyElement("radio1");
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式下调用该接口能重置元素状态。

阅读模式、打印预览模式下调用该接口无效

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=ResetModifiedSpecifyElement>

8.9.113. *根据元素编号修改元素状态为未修改状态

接口：

Bool ResetModifiedSpecifyElementByID(string id)

描述:

重置指定 ID 的元素的修改状态为未修改状态

参数:

参数名	是否必填	数据类型	说明
Id	必填	String	表示元素编号。

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
let res = ctl.ResetModifiedSpecifyElementByID ("radio1");
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式下调用该接口能重置元素状态。

阅读模式、打印预览模式下调用该接口无效

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=ResetModifiedSpecifyElementByID>

8.9.114. *选中当前光标处内容

接口:

Bool SelectCurrentElement()

描述:

选择当前元素。

光标在输入域里面选中整个输入域

光标在单元格中（单元格空白）选中整个单元格

光标在正文纯文本中，选中整个正文

参数:

无

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
var result = ctl.SelectCurrentElement()
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式下调用该接口能进行选中。阅读模式调用该接口光标在输入域中无效、打印预览模式下调用该接口无效

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SelectCurrentElement>

8.9.115. *根据当前光标处所在输入域向上一个输入域进行焦点跳转

接口:

Bool FocusPreviousInput(bool isEditor)

描述:

聚焦前一个输入域

参数:

参数名	是否必填	数据类型	说明
isEditor	必填	Bool	参数表示在聚焦时，是否启动编辑激活功能。 在编辑激活时，如果是下拉或时间输入域，会自动显示下拉列表或时间选择框

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
let res = ctl.FocusPreviousInput(false);
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式下调用该接口能进行跳转。阅读模式、打印预览模式下调用该接口无效

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=FocusPreviousI>

[nput](#)

8. 9. 116. *根据当前光标处所在输入域向下一个输入域进行焦点跳转

接口:

Bool FocusNextInput (bool isEditor)

描述:

聚焦下一个输入域。GetNextFocusFieldElement 表示获取下一个可以聚焦的输入域。

参数:

参数名	是否必填	数据类型	说明
isEditor	必填	Bool	参数表示在聚焦时，是否启动编辑激活功能。 在编辑激活时，如果是下拉或时间输入域，会自动显示下拉列表或时间选择框

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
let res = ctl.FocusNextInput(false);
```

注意:

当焦点在页眉中输入域时，无法从页眉把焦点切换到正文

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式下调用该接口能进行跳转。阅读模式、打印预览模式下调用该接口无效

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=FocusNextInput>

8.9.117. EditorRefreshContainerView

接口:

Bool EditorRefreshViewElement (object element)

描述:

刷新指定的元素。EditorRefreshContainerView 刷新的是容器元素。

参数:

参数名	是否必填	数据类型	说明
Element	必填	Object	该接口支持三种类型参数 1. 参数是字符串格式，表示元素编号。 2. 参数是 json 格式，属性是 id，表示元素编号。 3. 参数是后台引用对象 4. 参数是 NativeHandle 对象。

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
let result = ctl. EditorRefreshViewElement ('field1')
```

8.9.118. *根据元素对象对元素进行用户锁定

接口:

Bool SetElementContentLock(object element, json settings)

描述:

设置元素的内容锁定相关属性。

参数:

参数名	是否必填	数据类型	说明
Element	必填	Object	该接口支持三种类型参数 1. 参数是字符串格式，表示元素编号。 2. 参数是 json 格式，属性是 id，表示元素编号。 3. 参数是后台引用对象 4. 参数是 NativeHandle 对象。
settings	必填	Json	包含属性如下:
+OwnerUserID	必填	String	容器元素所有者用户 ID
+CreationTime	必填	String	创建时间
+AuthorisedUserIDList	必填	String	允许编辑元素的用户 ID 列表，用英文逗号分隔

返回值:

数据类型	说明
Bool	操作是否成功

代码示例:

```
let ctl = document.getElementById("myWriterControl");
    var options={
        OwnerUserID:"zhangsan",
        CreationTime:"2024-04-28",
        AuthorisedUserIDList:"lisi"
    }
var result = ctl.SetElementContentLock("姓名",options)
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式下调用该接口能进行用户锁定。

阅读模式、打印预览模式下调用该接口无效

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetElementContentLock>

8.9.119. *判断当前光标处是否允许插入元素

接口:

Array CanInsertElementAtCurrentPosition (string elementTypeName)

描述:

判断当前能否插入指定类型的元素。

参数:

参数名	是否必填	数据类型	说明
elementTypeName	必填	String	结构化元素类型的名称，包含 xtextmediaelement 多媒体元素、xtextnewmedical expressionelement 医学表达式元素、xt extinputfieldelement 输入域元素、xte xtradioboxelement 单选框元素、xtextc heckboxelement 复选框元素、xtextlabe lelement 标签文本元素、xtextnewbarco deelement 条形码元素、xtexttdbarcode element 二维码元素、xtexthorizontal lineelement 水平线元素、xtextimageele ment 图片元素、xtextpageinfoelement 页码元素、xtextbuttonelement 按钮元 素、xtexttableelement 表格元素、xtex

			tsignimageelement 电子签名元素。

返回值:

bool: 判断是否可以插入指定类型的元素

代码示例:

<pre>let result = ctl.CanInsertElementAtCurrentPosition("xtextinputfieldelement")</pre>

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式下、阅读模式、打印预览模式下调用该接口无效

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=CanInsertElementAtCurrentPosition>

8. 9. 120. *获取指定元素列表在文档中坐标位置

接口:

Json GetElementCoordinateByID(object ids, bool ispreviewmode, string specifyxml)

描述:

获取元素位置兼容四代接口。

参数:

参数名	是否必填	数据类型	说明

ids	必填	Object	该接口支持两种类型参数 1. 参数是字符串格式，表示元素编号。 2. 参数是 array 格式，数组项是元素的 ID
ispreviewmode	必填	Bool	是否为预览模式
specifyxml	必填	String	传递的 XML，为空则使用当前文档

返回值：

返回 json 格式，属性包含：

属性	数据类型	说明
PaperKind	Float	页面纸张名称
PaperWidth	Float	页面宽度
PaperHeight	Float	页面高度
Elements	数组	元素属性组成的数组，数组项的内容如下
+ Width	Float	元素宽度
+ Height	Float	元素高度
+OwnerPageIndex	Float	元素的页面序号
+ ID	Float	元素编号
+ Left	Float	左侧位置
+ Top	Float	顶部位置

代码示例：

```
var result = ctl.GetElementCoordinateByID(arr, true, null);
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式下、阅读模式、打印预览模式下调用该接口都能获取到指定元素坐标位置

兼容接口：

兼容四代接口。

接口优化:

与四代编辑器调用一致

示例地址:

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetElementCoordinateByID>

8. 9. 121. *根据元素对象获取下一个输入域. NET 对象值

接口:

Object GetNextFocusFieldElement(object element)

描述:

获取指定输入域的下一个定位焦点的输入域元素。

参数:

参数名	是否必填	数据类型	说明
Element	必填	Object	该接口支持三种类型参数 1. 参数是字符串格式，表示元素编号。 2. 参数是 json 格式，属性是 id，表示元素编号。 3. 参数是后台引用对象 4. 参数是 NativeHandle 对象。

返回值:

数据类型	说明
Object	输入域的后台引用对象

代码示例:

```
var result = ctl.GetNextFocusFieldElement(ctl.CurrentElement())
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式下、阅读模式、打印预览模式下调用该接口无效

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetNextFocusFieldElement>

8. 9. 122. *根据元素对象获取该元素下子元素.NET 对象列表

接口：

Array GetChildElements(object element)

描述：

获取指定容器元素的内部子元素列表。该接口调用的是 GetElementProperties 接口，返回值是 GetElementProperties 返回值的 Elements。

参数：

参数名	是否必填	数据类型	说明
element	必填	Object	它可以是三种类型参数：字符串类型的元素 ID、NativeHandle 对象、引用对象。

返回值：

数据类型	说明	备注
Array	元素组成的数组对象	

代码示例：

```
var result = ctl.GetChildElements('txtAge');
alert(JSON.stringify(result));
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式下、阅读模式、打印预览模式下调用该接口无效

兼容接口：

兼容四代接口。

接口优化：

五代该方法只有一个参数，功能都是获取指定容器下的子元素列表

示例地址：

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=GetChildElements>

8. 9. 123. *自定义单、复选框框的样式

接口：

Bool SetStandardImage(string strKey, string bsImage)

描述：

设置内部视图的一些图标自定义样式。

参数：

参数名	是否必填	数据类型	说明
strKey	必填	String	需要设置的图标名称，包含： CheckBoxChecked 勾选状态的复选框，必须为 16 * 16 的图片格式 CheckBoxUnchecked 不是勾选状态的复选框，必须为 16 * 16 的图片格式。 RadioBoxChecked 勾选状态的单选框，必须为 16 * 16 的图片格式。

			RadioBoxUnchecked 不是勾选状态的单选框，必须为 16 * 16 的图片格式。
bsImage	必填	String	图片的 base64 数据

返回值：

数据类型	说明	备注
Bool	操作是否成功	

代码示例：

```
ctl.SetStandardImage("CheckBoxChecked", checkedImage);
ctl.SetStandardImage("RadioBoxChecked", checkedImage);
ctl.SetStandardImage("CheckBoxUnchecked", uncheckedImage);
ctl.SetStandardImage("RadioBoxUnchecked", uncheckedImage);
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式下、阅读模式、打印预览模式下调用该接口都能自定义

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://www.dewriter.cn:8765/apiDocument/edit/index.html?apiname=SetStandardImage>

8.9.124. *根据区域获取相同类型元素列表

接口：

Array GetRootElementsByTypeName (string elementTypeName, string part)

描述:

获取指定区域下的指定类型的所有元素。

参数:

参数名	是否必填	数据类型	说明
elementTypeName	必填	String	结构化元素类型的名称，包含 xtextmediaelement 多媒体元素、xtextnewmedicalexpressionelement 医学表达式元素、xtextinputfieldelement 输入域元素、xtextradioboxelement 单选框元素、xtextcheckboxelement 复选框元素、xtextlabellement 标签文本元素、xtextnewbarcodeelement 条形码元素、xtexttdbarcodeelement 二维码元素、xtexthorizontalallineelement 水平线元素、xtextimageelement 图片元素、xtextpageinfoelement 页码元素、xtextbuttonelement 按钮元素、xtexttableelement 表格元素、xtextsignimageelement 电子签名元素。
Part	必填	String	指定的区域，包含 body 正文区域、header 页眉区域、footer 页脚区域、all 所有区域。

返回值:

以数组的形式返回所有的指定类型的元素列表

代码示例:

```
let result = ctl.GetRootElementsByTypeName("xtextinputfieldelement", "body")
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式下、阅读模式、打印预览模式下调用该接口都能获取到相同类型元素列表

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=GetRootElementByTypeName>

8. 9. 125. *根据折线图对象获取折线图元素的数据集对象

接口:

Json GetChartDataByID(object element)

描述:

获取折线图元素的数据集对象。该接口调用的是 GetElementProperties 接口，返回值是 GetElementProperties 返回值的 ChartDatas 属性。

参数:

参数名	是否必填	数据类型	说明
Element	必填	Object	该接口支持三种类型参数 1. 参数是字符串格式，表示元素编号。 2. 参数是 json 格式，属性是 id，表示元素编号。 3. 参数是后台引用对象 4. 参数是 NativeHandle 对象。

返回值:

数据类型	说明
Object	折线图元素的数据集对象

代码示例:

```
let element = ctl.GetElementProperties(ctl.CurrentElement());  
let result = ctl.GetChartElementDataByID(element)
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式下、阅读模式、打印预览模式下调用该接口都能获取到折线图数据点数据

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=GetChartElementDataByID>

8. 9. 126. *根据折线图对象设置折线图元素的数据集对象

接口:

Json SetChartElementDataByID(object element, json options)

描述:

设置折线图元素的数据集对象。该接口调用的是 SetElementProperties 接口。

参数:

参数名	是否必填	数据类型	说明
Element	必填	Object	该接口支持三种类型参数

			1. 参数是字符串格式，表示元素编号。 2. 参数是 json 格式，属性是 id，表示元素编号。 3. 参数是后台引用对象 4. 参数是 NativeHandle 对象。
options	必填	Json	该参数包含一个属性 ChartDatas，这个属性就是拆线图元素的数据集对象。

返回值：

数据类型	说明
Bool	操作是否成功

代码示例：

```
var options = {
  "GridYSplitNum": 8,
  "LegendVisible": false,
  "PaddingBottom": 50,
  "PaddingLeft": 100,
  "PaddingRight": 30,
  "PaddingTop": 30,
  "ViewStyle": "Bar",
  "YAxes": [
    {
      "Visible": true,
      "Color": "#FF0000",
      "GridStep": 10,
      "LeftSide": true,
      "MaxValue": 100,
      "MinValue": 0,
      "Name": "Y轴名称1"
    },
    {
      "Visible": true,
      "Color": "#00FFFF",
      "GridStep": 10,
      "LeftSide": true,
      "MaxValue": 50,
      "MinValue": 0,
      "Name": "Y轴名称2"
    }
  ]
}
```

```
    ],
    "Values": [
      {
        "SeriesName": "Y轴名称1",
        "GroupName": "分组0",
        "SymbolType": "Diamond",
        "Value": 69.433784
      },
      {
        "SeriesName": "Y轴名称1",
        "GroupName": "分组1",
        "SymbolType": "Diamond",
        "Value": 20.2027
      }
    ]
  }
}
var result = ctl.SetChartElementDataByID(ctl.CurrentElement('XTextCharElement'),
options);
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式下能进行修改。阅读模式、打印预览模式下调用该接口无效

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SetChartElementDataByID>

8.9.127. *根据表格编号和行索引、列索引所在单元格进行光标移动到指

定位

接口：

Bool MoveToTableCellTag(json options)

描述：

光标移动到单元格的指定位置

参数：

参数名	是否必填	数据类型	说明
options	必填	Json	属性信息，包含
+ tableid	必填	String	表格编号，当为空时表示使用当前光标的表格
+ rowindex	必填	Int	从 0 开始计算的表格行的行号，当为空时表示使用当前光标的表格单元格
+ colindex	必填	Int	从 0 开始计算的表格行的列号，当为空时表示使用当前光标的表格单元格
+ tag	必填	String	指定单元格的相对位置，可选值包含：CellHome 单元格内部前面、CellEnd 单元格内部后面、PreCell 前一个单元格、AfterCell 后一个单元格、BeforeTable 表格前、AfterTable 表格后。

返回值：

数据类型	说明
Bool	操作是否成功

代码示例：

```
var obj = {
    "tableid": "table1",
    "rowindex": 0,
    "colindex": 0,
    "tag": "BeforeTable"
};
var result = ctl.MoveToTableCellTag(obj);
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式下、阅读模式、打印预览模式下调用该接口无效

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=MoveToTableCellTag>

8. 9. 128. *根据元素对象设置元素自定义属性

接口：

Bool SetElementCustomAttributes(object element, json options)

描述：

设置指定元素的自定义属性。如果自定义属性名称相同，会覆盖原来的自定义属性。

参数：

参数名	是否必填	数据类型	说明
element	必填	Object	它可以是三种类型参数：字符串类型的元素 ID、NativeHandle 对象、引用对象。
options	必填	Json	自定义属性 json 对象

返回值：

数据类型	说明	备注
Bool	操作是否成功	

代码示例：


```
var opts = {
    d1: "DDDDDD",
    d2: "valuevalue",
    AA:"myAA"
};
var result = myWriterControl.SetElementCustomAttributes(element,opts);
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式下调用接口能设置自定义属性。

阅读模式、打印预览模式下调用该接口无效

兼容接口:

兼容四代接口。

接口优化:

与四代编辑器调用一致

示例地址:

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=SetElementCustomAttributes>

8.9.129. *根据元素对象获取元素自定义属性

接口:

Json GetElementCustomAttributes(object element)

描述:

获取指定元素的自定义属性。

参数:

参数名	是否必填	数据类型	说明
element	必填	Object	它可以是三种类型参数：字符串类型的元素 ID、NativeHandle 对象、引用对象。

返回值:

数据类型	说明	备注
Json	自定义属性 json 对象	

代码示例:

```
var result = myWriterControl.GetElementCustomAttributes(element);
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口能获取自定义属性

兼容接口:

兼容四代接口。

接口优化:

与四代编辑器调用一致

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=GetElementCustomAttributes>

8.9.130. *根据元素编号获取自定义属性中指定 Key 对应的 Value 值

接口:

```
Object GetElementProperty(string id, string name)
```

描述:

获取指定元素的指定属性的值。

参数:

参数名	是否必填	数据类型	说明
Id	必填	String	字符串类型的元素 ID

name	必填	String	字符串类型的元素属性名称

返回值:

数据类型	说明	备注
Object	返回属性的内容，以字符串的形式返回	

代码示例:

```
var result = myWriterControl.GetElementProperty("field1", "name");
```

相同效果:

```
ctl.GetElementAttribute("field1", "name")
```

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口能获取指定自定义属性中 key 对应的 Value 值

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=GetElementProperty>

8.9.131. *根据元素编号设置自定义属性中指定 Key 对应的 Value 值

接口:

```
bool SetElementProperty(string id, string name, string Value)
```

描述:

设置指定元素的自定义属性。

参数:

参数名	是否必填	数据类型	说明

Id	必填	String	字符串类型的元素 ID
name	必填	String	字符串类型的元素属性名称
Value	必填	String	字符串类型的元素值

返回值:

数据类型	说明	备注
Bool	操作是否成功	

代码示例:

```
var result = myWriterControl.SetElementProperty("field1", "id", "field2");
```

相同效果:

```
ctl.SetElementAttribute("field1", "name", "占山")
```

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式下调用该接口能设置指定自定义属性中 key 对应的 Value 值

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=SetElementProperty>

8.9.132. *用户操作权限不足显示自定义提示信息

接口:

```
bool SetStringResources(json options)
```

描述:

删除设置了不允许删除元素，不允许编辑元素删除、内容设置内保护进行删除出现的提示信息自定义。

参数：

参数名	是否必填	数据类型	说明
options	必填	json	
+PromptProtectedContent	非必填	String	删除内容保护内容自定义提示信息
+ReadOnlyContentProtect	非必填	String	内容保护
+ReadOnlyContentLocked	非必填	String	内容锁定
+ReadOnlyElementMarkUndeleteable_ID	非必填	String	删除不允许删除的内容提示
+ReadOnlyInputFieldUserEditable_ID	非必填	String	操作不允许编辑的内容提示
+ReadOnlyLowPermissionLevel_CurLevel_NeedLevel	非必填	String	权限等级不足提示
+ReadOnlyLowPermissionLevel_CurLevel_NeedLevel	非必填	String	系统粘贴字符控制粘贴字符数提示
+PageBreak	非必填	string	自定义插入分页符时，分页线上显示内容。
+Header	非必填	string	自定义双击进入页眉时，左下角出现的页眉提示文本内容
+Footer	非必填	string	自定义双击进入页脚时，左下角出现的页脚提示文本内容
+PromptRejectFormat_Format	非必填	string	自定义编辑器支持数据格式提示内容

返回值：

数据类型	说明	备注
无返回值		

代码示例：

```
var options = {
    PromptProtectedContent: "有内容受到保护，操作受到限制或无法执行。111111111",
    ReadonlyContentProtect: "由于强制设置了内容保护而导致内容只读。2222222",
    ReadonlyContentLocked: "内容被锁定。33333333",
    ReadonlyElementMarkUndeleteable_ID: "元素 "{0}" 被标记为不能删除。444444",
    ReadonlyInputFieldUserEditable_ID: "输入域[{0}]的内容设置为不能直接修改。5555",
    ReadonlyLowPermissionLevel_CurLevel_NeedLevel: "权限等级不够，当前等级为{0}，所需等级{1}。666666",
    PromptMaxTextLengthForPaste_Length: "系统设置为粘贴或插入内容时不能接收超过{0}个字符。777777"
};
ctl.SetStringResources(options);
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。但是在阅读模式，只读模式，打印预览模式下即使调用成功，也没有任何实际作用，所以推荐客户在编辑模式下进行调用接口

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

无

8.9.133. *获取鼠标选中内容包含的结构化元素列表

接口:

```
bool GetCurrentLineElements()
```

描述:

获取鼠标选中一行内容里面的结构化元素列表，其中不包含表格的结构化元素。且当选

中的内容超过了一行，获取后面一行的结构化元素列表。

参数：

参数名	是否必填	数据类型	说明
无			

返回值：

数据类型	说明	备注
json	返回结构化元素详细列表（包含完整元素属性）	

代码示例：

```
var result = myWriterControl.GetCurrentLineElements();
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。需要有选中状态该接口才有效

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

无

8.9.133. *根据元素编号激活输入域下拉弹窗

接口：

```
bool BeginEditElementValue(object id)
```

描述：

当用户不能通过鼠标点击激活输入域下拉弹窗时，可通过本方法进行激活下拉弹窗。

参数：

参数名	是否必填	数据类型	说明
id	必填	object	支持元素编号、NativeHandle、.net 后台对象

返回值：

数据类型	说明	备注
bool	接口调用是否有效	

代码示例：

```
var result = myWriterControl.BeginEditElementValue("field1");
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。鼠标双击能弹出下拉框，该接口调用就能有效。

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

无

8.9.134. *根据元素编号获取文本内容

接口：

```
bool GetElementTextByIDPro(string id, bool includeBorderText, bool  
includeLabelUnitText, bool ignoreDeleteElement, bool  
ignoreVisibleElement)
```

描述：

当输入域里面的内容包含上标、下标等操作，返回的文本数据会带上上、下标样式。

参数：

参数名	是否必填	数据类型	说明
id	必填	object	支持元素编号
includeBorderText	非必填	bool	是否包含前后边框
includeLabelUnitText	非必填	bool	是否包含标签和单位
ignoreDeleteElement	非必填	bool	是否忽略逻辑删除的元素
ignoreVisibleElement	非必填	bool	是否忽略隐藏的元素

返回值：

数据类型	说明	备注
string	返回带有上下标的字符串数据	

代码示例：

```
var result = myWriterControl.GetElementTextByIDPro("field1");
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<http://117.89.85.111:8086/apiDocument/edit/index.html?apiname=GetElementTextByIDPro>

8.9.135. *根据元素编号对指定元素赋文本内容

接口：

```
bool SetElementTextByIDPro(string id, string text)
```

描述:

对输入域里面的内容赋带有上标、下标样式的内容，或者正常的文本内容。

参数:

参数名	是否必填	数据类型	说明
id	必填	object	支持元素编号、NativeHandle、.net 后台对象
text	必填	string	赋值的字符串文本内容

返回值:

数据类型	说明	备注
bool	接口调用是否有效	

代码示例:

```
var result = myWriterControl.SetElementTextByIDPro("field1", "哈师大合适的哈师大  
啊收到货<sup>11</sup>1阿斯达撒大声地");
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。任意模式都能调用，打印预览模式下调用当前界面无效，退出打印预览就能看到赋值的内容。

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<http://117.89.85.111:8086/apiDocument/edit/index.html?apiname=SetElementTextByIDPro>

8.9.136. *获取打印预览指定页图片数据

接口:

```
bool PrintAsImage(string pageArr,function callBack)
```

描述:

对输入域里面的内容赋带有上标、下标样式的内容，或者正常的文本内容。

参数:

参数名	是否必填	数据类型	说明
pageArr	必填	string	文档页码值，当需要生成多页图片，使用英文逗号分割 ("1,3")，空字符(或者 null)表示导出所有页图片数据。
callBack	必填	function	从回调函数里面获取对应页图片数据

返回值:

数据类型	说明	备注
无		

代码示例:

```
var ctl=document.getElementById("myWriterControl");//获取当前编辑器对象
ctl.PrintAsImage("1,3",function (params) {
    console.log(params)
})
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。任意模式都能调用。

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<http://117.89.85.111:8086/apiDocument/edit/index.html?apiname=PrintAsImage>

注意：

编辑器版本需要大于 2024-11-11 11:17:34 才有效。

8. 9. 137. *全局修改文本样式

接口：

bool UpdateDocumentAllStyle(json cssstyle)

描述：

修改病历文本中所有内容的样式。

参数：

参数名	是否必填	数据类型	说明
json	必填		
colorstring	必填	string	修改文本颜色

返回值：

数据类型	说明	备注
bool		

代码示例：

```
var ctl=document.getElementById("myWriterControl");//获取当前编辑器对象
ctl.UpdateDocumentAllStyle({
    colorstring: "red"
});
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。任意模式都能调用。

兼容接口：

五代独有接口。

接口优化：

无

示例地址:

<http://117.89.85.111:8086/apiDocument/edit/index.html?apiname=UpdateDocumentAllStyle>

注意:

无。

8. 9. 138. *当前输入焦点处批量插入元素

接口:

bool InsertElements(json ele)

描述:

当前输入焦点处批量插入内容，与 SetChildElements 方法第二个参数一致。

参数:

参数名	是否必填	数据类型	说明
json	必填		

返回值:

数据类型	说明	备注
bool		

代码示例:

```
var ctl=document.getElementById("myWriterControl");//获取当前编辑器对象
var options = [
    {
        //输入域 参数可以参考 3.5.1
        'input': {
            "ContentReadonly": false,
            "ToolTip": "请输入年龄",
            "BackgroundText": "年龄",
            "EnableValueValidate": false,
            "Name": "年龄",
            "ID": "txtAge",
        },
    },
    {
        'text': "BLABLABLA 我是普通文本 BLABLABLA",
    },
    {
        //带有上标下标的文本
    },
]
```

```
        'html': "<sub>1</sub>"
    }
    ];
    ctl.InsertElements(options);
}
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。任意模式都能调用。

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

http://117.89.85.111:8086/pingcode/DUWRITER5_0-1960.html

注意:

无。

8.9.138. *获取指定容器元素内容 xml 的 base6 数据

接口:

bool GetElementInnerBase64XmlByID(string id, bool includeThis)

描述:

获取当前输入域内容的 xml-base64 数据并且病历数据同步。

参数:

参数	是否必填	类型	说明
Id	必填	object	它可以是三种类型参数：字符串类型的元素 ID、NativeHandle 对象、引用对象。
includeThis	非必填	Bool	生成的 XML 中是否包含元素自身，默认是 true，

			包含元素本身结构。

返回值:

数据类型	说明	备注
string	返回 base64 数据	

代码示例:

```
var ctl=document.getElementById("myWriterControl");//获取当前编辑器对象
ctl.GetElementInnerBase64XmlByID("field1",false);
```

相同:

与方法 GetElementInnerXmlByID 效果一致, 返回值不同

结合使用:

无

接口型:

纯前端接口, 不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。任意模式都能调用。

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

[http://117.89.85.111:8086/apiDocument/edit/index.html?apiname=GetElementInnerBase64XmlBy
ID](http://117.89.85.111:8086/apiDocument/edit/index.html?apiname=GetElementInnerBase64XmlByID)

注意:

无。

8.9.139. *修改指定页文本标签元素内容

接口:

```
bool SetPageLabelText(object id, int pageindex, string textstr)
```

描述:

修改指定页中文本标签元素内容，常用于护理记录单场景。

参数：

参数	是否必填	类型	说明
Id	必填	object	它可以是三种类型参数：字符串类型的元素 ID、NativeHandle 对象、引用对象。
pageIndex	必填	int	页码，从 0 开始计算
textstr	必填	string	新文本标签元素内容

返回值：

数据类型	说明	备注
bool		true: 接口执行成功

代码示例：

<pre>var ctl=document.getElementById("myWriterControl");//获取当前编辑器对象 ctl.SetPageLabelText("label1",1,"啊啊啊啊")</pre>

相同：

```
var PageTexts11 = [{
    PageIndex: 1,
    PageIndexText: "2222"
}]
var options2 = {
    PageTexts: PageTexts11
};
ctl.SetElementProperties("label1", options2);
```

结合使用：

无

接口型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。任意模式都能调用。

兼容接口：

五代独有接口。

接口优化:

无

示例地址:

<http://117.89.85.111:8086/apiDocument/edit/index.html?apiname=SetPageLabelText>

注意:

文本标签元素连接模式(ContactAction)值需要设置成 **disable** 接口才有效果。

8.10. 开发辅助

8.10.1. *判断编辑器是否开启日志功能

接口:

Bool IsAPILogRecord()

描述:

判断编辑器是否正在记录 API 日志。如果启用了日志记录，则返回 true。

参数:

无。

返回值:

数据类型	说明	备注
Bool	如果启用了日志记录，则返回 true。	

代码示例:

```
let res=ctl.IsAPILogRecord();
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritr.cn:8765/apiDocument/edit/index.html?apiname=IsAPILogRecord>

8. 10. 2. *编辑器开启日志功能

接口:

Bool StartAPILogRecord()

描述:

启用 API 日志记录功能。启用 API 日志记录功能，本功能会消耗很多内存, 减慢运行速度，谨慎使用。

参数:

无。

返回值:

数据类型	说明	备注
Bool	启用 API 日志记录功能是否成功	

代码示例:

```
let res=ctl.StartAPILogRecord();
```

注意:

当出现偶发编辑器报错问题, 可以执行该方法开启编辑器日志功能, 然后重新操作一次, 出现对应的错误, 之后执行编辑器方法 DownloadAPIRecordData() 生成的 json 文件和编辑器界面加载文书提供给都昌技术人员。

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://www.dcwritr.cn:8765/apiDocument/edit/index.html?apiname=StartAPILogRecord>

8. 10. 3. *编辑器停用日志功能

接口：

Bool StopAPILogRecord()

描述：

停止记录 API 日志的功能，清空所有数据。当下载完日志文件，就可以执行该接口停用日志功能

参数：

无。

返回值：

数据类型	说明	备注
Bool	停止 API 日志记录功能是否成功	

代码示例：

```
let res=ctl.StopAPILogRecord();
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。

兼容接口：

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritr.cn:8765/apiDocument/edit/index.html?apiname=StopAPILogRecord>

8.10.4. ClearAPILogRecordData

接口:

Bool ClearAPILogRecordData ()

描述:

清除已经记录的 API 日志数据。。

参数:

无。

返回值:

数据类型	说明	备注
Bool	清除数据否成功	

代码示例:

```
let res=ctl. ClearAPILogRecordData ();
```

8.10.5. *编辑器下载日志功能

接口:

Bool DownloadAPIRecordData()

描述:

下载 API 日志数据。该接口需要动态的添加 a 标签并创建下载链接，以 utf-8 编码下载一个 API 调用记录的 json，文件名是 APIRecord_开头，再追加当前日期时间字符串的随机值。

参数:

无。

返回值:

数据类型	说明	备注
Bool	下载数据否成功	

代码示例：

```
let res=ctl.DownloadAPIRecordData();
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=DownloadAPIRecordData>

8.10.6. *判断编辑器命令是否可用

接口：

Json GetCommandStatus(string commandName)

描述：

查询编辑器命令状态。获取编辑器是否支持指定的命令，该命令是否可用。如编辑器支持 fontName 命令，该命令是修改选择内容的字体名称的功能，在文档只读时该命令应该是失效的。

参数：

参数名	是否必填	数据类型	说明
commandName	必填	String	字符串形式的命令名称。

返回值：

该接口返回 json 数据，包含属性如下

属性名称	说明	备注
Name	返回该命令的名称，一般和参数相同。字符串类型。	命令名称
Supported	该命令是否受到系统支持。Bool 数据类型。	命令在编辑器是否支持
Enabled	该命令在当前状态下是否可用。Bool 数据类型。	命令在编辑器是否可以用
Checked	该命令是否处于勾选状态。Bool 数据类型。表示当前内容的状态是否对应该命令的状态。	

代码示例：

```
let res=ctl.GetCommandStatus("Undo")//判断编辑器内是否允许执行撤销操作
```

相同效果：

无

结合使用：

一般结合事件 EventUpdateToolarState 来对编辑器外的工具栏增加各个按钮的状态效果，比如撤销、重做、剪切、复制等图片的状态是否可用进行动态转换

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。

兼容接口：

五代独有接口。

接口优化：

无

示例地址：

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=GetCommandStatus>

8.10.7. *关闭编辑器释放资源

接口:

Void Dispose()

描述:

清理编辑器所有正在使用的资源。在删除控件时，建议先从 HTML-DOM 中删除控件的容器对象 DIV 元素，然后调用它的 Dispose() 方法，不正确的调用次序容易导致错误。

参数:

无。

返回值:

无

代码示例:

```
ctl.Dispose();
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

五代独有接口。

接口优化:

无

示例地址:

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=Dispose>

8.10.11. *GetCommandNameList

接口:

String GetCommandNameList()

描述:

获得所有支持的命令名称组成的字符串，各个名称之间用逗号分开。

参数:

参数名	是否必填	数据类型	说明
commandName	必填	String	字符串形式的命令名称。

返回值:

属性名称	说明	备注
String	返回所有支持的命令名称组成的字符串，各个名称之间用逗号分开。	

代码示例:

```
let res=ctl. GetCommandNameList ();
```

8. 10. 12. *获取编辑器信息

接口:

Void AboutControl()

描述:

显示系统的关于信息。该接口无返回值，但是弹出一个当前编辑器的信息。包括版本、授权等。弹出信息如下：

www.dcwriter.cn:8022 显示
第五代WEB版DCWriter文本编辑器，南京都昌信息科技有限公司版权所有。
软件发布时间:2024-01-17 08:30:00
服务器地址:dcwriter.cn
软件授权:[用户名:都昌][授权软件产品:编辑器WEB版,时间轴][旗舰版]
[更新限制日期: 2024-11-9][截止日期: 2024-5-8][语言: 简体中文,English]



参数:

无

返回值:

无

代码示例:

```
ctl.AboutControl ();
```

相同效果:

```
ctl.DCExecuteCommand('AboutControl',true,null)
```

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

兼容四代编辑器

接口优化:

与四代编辑器调用一致

示例地址:

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=AboutControl>

8.10.13. GetLastCustomCommandText（待客户存在业务，在开启该接口）

接口:

描述:

参数:

参数名	是否必填	数据类型	说明

返回值:

数据类型	说明	备注

代码示例:

8. 10. 14. *获取文档中设置导航节点信息

接口:

String GetNavigateNodesString()

描述:

获得导航节点字符串，各个名称之间用&分开。在选择内容后通过 TitleLevel 命令设置标题层次，会在对应的样式中添加 TitleLevel 属性，标记它的标题层次。再通过 GetNavigateNodesString 可以获取当前文档的所有的标记标题层次的信息。

参数:

无

返回值:

属性名称	说明	备注
String	返回所有导航节点组成的字符串，各个名称之间用&分隔。例如返回：1=基本信息&2=主诉&3=现病史&4=既往史&5=个人史&6=婚育史&7=家族史。	

代码示例:

let res=ctl.GetNavigateNodesString ();

相同效果:

无

结合使用:

ctl.NavigateByNodeID(导航节点值)//也就是&左边内容中=左边的数字。

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口:

不兼容四代接口。

接口优化:

无

示例地址:

<http://117.89.85.111:8086/apiDocument/edit/index.html?apiname=GetNavigateNodesString>

8. 10. 14. *点击导航节点列表编辑视图自动定位节点

接口:

String NavigateByNodeID(string id)

描述:

根据导航节点值定位视图到对应导航节点。

参数:

string id

返回值:

类型	说明	备注
bool		true 接口调用成功

代码示例:

```
let res=ctl.NavigateByNodeID ("2");
```

相同效果:

无

结合使用:

ctl.GetNavigateNodesString();//根据返回节点解析数据,把导航值记录在字段创建导航列表节点中,通过点击事件获取导航值进行视图定位

接口类型:

纯前端接口,不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。只读模式下、阅读模式、打印预览模式下都能调用,但是只有在编辑模式下调用该接口导航节点才能正常定位。

兼容接口:

不兼容四代接口。

接口优化:

无

示例地址:

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=index&pageUrl=document>

[Navigation.html](#)

8.10.15. GetOptionValue

接口:

```
String GetOptionValue(string name)
```

描述:

获得指定名称的选项数值, 选项名称为“选项组名. 选项名称”的格式, 比如“ViewOptions.ShowParagraphFlag”。比如 `string v = obj.GetOptionValue("ViewOptions.ShowParagraphFlag");` 返回 “true”或“false”。 `string v2 = obj.GetOptionValue("ViewOptions.TagColorForNormalField");` 返回类似“#AAAAAA”等表示颜色值的字符串。

参数:

属性名称	说明	备注
Name	选项的名称, 比如 ViewOptions.ShowParagraphFlag, 为“选项组名. 选项名称”的格式	

返回值:

属性名称	说明	备注
String	返回选项对象的值	

代码示例:

```
let res=ctl. GetOptionValue ( "ViewOptions.ShowParagraphFlag" );
```

8.10.16. GetStyleIndexByString

接口:

```
Int GetStyleIndexByString (string name)
```

描述:

获得样式在列表中的编号

参数:

属性名称	说明	备注

Name	样式字符串，比如“FontName: 宋体; FontSize: 9”	

返回值:

属性名称	说明	备注
Int	返回样式列表中对应的序号，未找到返回-1。	

代码示例:

```
let res=ctl. GetStyleIndexByString ( “FontName: 宋体; FontSize: 9” );
```

8. 10. 17. *根据页码获取指定页中根据设置的行高获取剩余空白行数

接口:

```
Int GetSurplusLinesOfSpeifyPage(int pageIndex, float  
specifyLineHeight)
```

描述:

获得指定页的剩余空白行数

参数:

属性名称	说明	备注
pageIndex	指定的页码，从 0 开始	
specifyLineHeight	计算行数时，使用的行高	

返回值:

属性名称	说明	备注
Int	返回指定页的剩余空白行数。	

代码示例:

```
let res=ctl.GetSurplusLinesOfSpeifyPage(1, 65); //一页正常多少空白行
```

相同效果:

无

结合使用:

无

接口类型:

纯前端接口，不经过服务端

有无限制:

编辑器加载成功之后在任何情况下都能调用。

兼容接口：

五代编辑器独有

接口优化：

无

示例地址：

<https://www.dcwritter.cn:8765/apiDocument/edit/index.html?apiname=GetSurplusLinesOfSpeifyPage>

8. 10. 18. *编辑器是否处理设计模式

接口：

Bool InDesignMode()

描述：

判断编辑器是否处于设计模式。设计模式下，此时页面边界出现蓝色的虚线，此时级联设置、表达式设置、下拉列表设置、时间选择设置都是无效的。

参数：

无

返回值：

属性名称	说明	备注
Bool	返回是否处于设计模式。	

代码示例：

```
let res=ctl.InDesignMode();
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。

兼容接口：

五代编辑器独有

接口优化：

无

示例地址：

<https://www.dcwriter.cn:8765/apiDocument/edit/index.html?apiname=InDesignMode>

8. 10. 19. 检验编辑器保存 xml 格式是否正常

接口：

Bool IsValidateXML(string content)

描述：

校验是否为 xml 格式。该接口的校验并不完全准确，只是要求 XML 的开头必须使用 XTextDocument 作为根节点。

参数：

参数名	是否必填	数据类型	说明
content	必填	String	判断参数是否是编辑器可以加载的 XML

返回值：

属性名称	说明	备注
Bool	返回是否是 xml 字符串。	

代码示例：

let res=ctl.IsValidateXML(str);

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。

兼容接口：

五代编辑器独有

接口优化：

无

示例地址：

无

8.10.20. IsValidateJSONString

接口：

```
Bool IsValidateJSONString (string content)
```

描述：

校验是否为 json 格式。该接口的校验并不完全准确，只是判断是否是 json 格式。该接口和 validateJSON 具体相同的功能。

参数：

参数名	是否必填	数据类型	说明
content	必填	String	Json 字符串

返回值：

属性名称	说明	备注
Bool	返回是否是 json 字符串。	

代码示例：

```
let res=ctl.IsValidateJSONString (str);  
let res=ctl.validateJSON (str);
```

8.10.21. GetBehaviorOptions（待议）

接口：**描述：****参数：**

参数名	是否必填	数据类型	说明

返回值:

数据类型	说明	备注

代码示例:

8.10.22. GetEditOptions（待议）

接口:

描述:

参数:

参数名	是否必填	数据类型	说明

返回值:

数据类型	说明	备注

代码示例:

8.10.23. GetSecurityOptions（待议）

接口:

描述:

参数:

参数名	是否必填	数据类型	说明

返回值:

数据类型	说明	备注

代码示例:

8.10.24. GetViewOptions（待议）

接口:

描述:

参数:

参数名	是否必填	数据类型	说明

返回值:

数据类型	说明	备注

代码示例:

8.10.25. refreshDocumentOptions

接口:

```
Void refreshDocumentOptions ()
```

描述:

刷新文档选项。为防止文档选项错误，在部分接口执行后，需要执行该接口，防止选项丢失，或选项失效，表示重新执行选项。

参数:

无

返回值:

无

代码示例:

```
ctl.refreshDocumentOptions ();
```

8.10.26. ApplyDocumentOptions

接口:

```
Void ApplyDocumentOptions ()
```

描述:

启用文档选项。有的文档选项需要全部绘制才能有效，为防止文档选项正确执行，在设置完文档选项后，需要执行该接口，附加新的文档选项。一般该接口不能单独执行，需要在文档选项执行后再执行。

该接口执行后，在内部需要全部重新绘制全文，并再次调用 refreshDocumentOptions。

参数:

无

返回值:

无

代码示例:

```
ctl.DocumentOptions.ViewOptions.ShowHeaderBottomLine= false;  
ctl.DocumentOptions.ViewOptions.ShowParagraphFlag = false;  
ctl.ApplyDocumentOptions ()
```

8.10.27. ExecuteAllEffectExpressions

接口:

```
Bool ExecuteAllEffectExpressions ()
```

描述:

手动引发执行所有数值表达式。在部分情况下表达式未执行，需要调用该接口执行。

参数:

无

返回值:

数据类型	说明
Bool	设置是否成功

代码示例:

```
let res=ctl. ExecuteAllEffectExpressions ();
```

8.10.28. GetNotSupportFontNames

接口:

```
String GetNotSupportFontNames ()
```

描述:

获得最后一次加载的文档中不支持的字体名称列表，各个名称之间用逗号分开。

参数:

无

返回值:

数据类型	说明
String	获得最后一次加载的文档中不支持的字体名称列表，各个名称之间用逗号分开

代码示例:

```
let res=ctl. GetNotSupportFontNames ();
```

8.10.29. CanInsertAtCurrentPosition

接口:

```
Bool CanInsertAtCurrentPosition (string typeName)
```

描述:

判断当前能否插入指定类型的元素。

参数:

参数名	是否必填	数据类型	说明
typeName	必填	String	结构化元素类型名称, 类型名称可参考本文的结构化元素的描述。

返回值:

数据类型	说明
Bool	返回当前能否插入指定类型的元素

代码示例:

```
let res=ctl.CanInsertAtCurrentPosition ( "XTextInputFieldElement" );
```

8.10.30. UpdateAllRuleView

接口:

```
Void UpdateAllRuleView ()
```

描述:

在部分框架下会出现标尺重叠, 需要调用该接口, 重新绘制标尺。

参数:

无

返回值:

无

代码示例:

```
ctl. UpdateAllRuleView ();
```

8.10.31. GetSelectionNativeStartIndex

接口:

```
Int GetSelectionNativeStartIndex ()
```

描述:

获取选择内容的开始位置。

参数:

无

返回值:

数据类型	说明
Int	开始位置坐标信息

代码示例:

```
let res=ctl. GetSelectionNativeStartIndex ();
```

8.10.32. GetSelectionNativeLength

接口:

```
Int GetSelectionNativeLength ()
```

描述:

获取选择内容的长度。

参数:

无

返回值:

数据类型	说明
Bool	选择内容的长度信息

代码示例:

```
let res=ctl. GetSelectionNativeLength ();
```

8.10.33. GetSelectionStartIndex

接口:

```
Int GetSelectionStartIndex ()
```

描述:

获取选择内容的开始位置。该接口不如 GetSelectionNativeStartIndex 准确，建议使用 GetSelectionNativeStartIndex。

参数:

无

返回值:

数据类型	说明
Bool	选择内容的开始位置

代码示例：

```
let res=ctl. GetSelectionStartIndex ();
```

8.10.34. GetSelectionEndIndex

接口：

```
Int GetSelectionEndIndex ()
```

描述：

获取选择内容的结束位置。

参数：

无

返回值：

数据类型	说明
Bool	选择内容的结束位置

代码示例：

```
let res=ctl. GetSelectionEndIndex ();
```

8.10.35. SelectContentByStartEndIndex

接口：

```
Bool SelectContentByStartEndIndex (int startIndex, int endIndex)
```

描述：

根据开始位置和结束位置选择。位置信息，可通过 GetSelectionStartIndex 和 GetSelectionEndIndex 返回。

参数：

参数名	是否必填	数据类型	说明
startIndex	必填	Int	开始位置
endIndex	必填	Int	结束位置

返回值:

数据类型	说明	备注
Bool	是否执行成功	

代码示例:

```
ctl.SelectContentByStartEndIndex(1, 50);
```

8.10.36. CollectrResizeCanvas--不展示

接口:

```
Void CollectrResizeCanvas (element)
```

描述:

此方法用于处理 canvas 上的 resizeObserver 监听, 并处理 EventPageResize 事件。

参数:

参数名	是否必填	数据类型	说明
startIndex	必填	Int	开始位置
endIndex	必填	Int	结束位置

返回值:

数据类型	说明	备注
Bool	是否执行成功	

代码示例:

```
let canvas = document.querySelector('[dctype=page]')
try {
    var resizeObserver = new ResizeObserver(entries => {
        ctl.CollectrResizeCanvas(canvas);
    })
    alert('调用成功')
} catch (error) {
    alert('调用失败')
}
```

8.10.37. SetDocumentInfos

接口:

Bool SetDocumentInfos (json parameter)

描述:

设置文档的信息。在 XML 文档中有一个节点存储的 DocumentInfo 节点，存储文档的信息。该接口获取的就是这个节点的属性信息。开发时可根据业务系统功能和属性的数据类型，由客户自由定义各种参数。

参数:

参数名	是否必填	数据类型	说明
parameter	非必填	Json	以下参数是 json 的内容
+ readonly	非必填	Bool	是否只读
+ ShowHeaderBottomLine	非必填	Bool	是否显示页眉下划线
+ FieldBorderElementWidth	非必填	Float	输入域的边框的宽度
+ KEntryRangeMask	非必填	String	兼容参数
+ RuntimeTitle	非必填	String	标题
+ ID	非必填	String	编号
+ IsTemplate	非必填	Bool	是否模板
+ MRID	非必填	Bool	
+ TimeoutHours	非必填	Bool	
+ Version	非必填	Bool	
+ Title	非必填	Float	
+ Description	非必填	String	
+ LicenseText	非必填	String	
+ CreationTime	非必填	String	
+ LastModifiedTime	非必填	String	
+ EditMinute	非必填	String	
+ LastPrintTime	非必填	String	
+ Author	非必填	String	
+ AuthorName	非必填	String	

+ AuthorPermissionLevel	非必填	Int	
+ DepartmentID	非必填	String	
+ DepartmentName	非必填	String	
+ DocumentFormat	非必填	String	
+ DocumentType	非必填	String	
+ DocumentProcessState	非必填	String	
+ Comment	非必填	String	
+ Operator	非必填	String	
+ Printable	非必填	Bool	
+ StartPositionInPringJob	非必填	Float	
+ HeightInPrintJob	非必填	Float	

返回值:

属性名称	说明	备注
Bool	返回操作是否成功。	

代码示例:

```
var result = ctl.GetCurrentParagraphStyle ();
```

8.10.38. SetOptionValue

接口:

```
bool SetOptionValue(string name, string Value)
```

描述:

设置指定选项的值。由于很多选项是需要刷新才有效果，该接口需要添加 refreshDocumentOptions。

参数:

参数名	是否必填	数据类型	说明
name	必填	字符串	选项名称，选项名称为“选项组名. 选项名称”

			的 格 式 ， 比 如 “ViewOptions.ShowParagraphFlag”。。
Value	必填	字符串	指定的选项的值

返回值:

数据类型	说明	备注
Bool	设置是否成功	

代码示例:

```
obj.SetOptionValue("ViewOptions.ShowParagraphFlag", "true");
obj.SetOptionValue("ViewOptions.TagColorForNormalField", "#AAAAAA");
ctl.refreshDocumentOptions ();
```

8.10.39. ReportExceptionDialog

接口:

Void ReportExceptionDialog ()

描述:

显示运行中的错误消息对话框。在编辑器运行中如果出现错误消息，系统自动捕获这些消息，并在对话框中显示。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj. ReportExceptionDialog ();
```

8.10.40. GetDocumentInfos

接口:

```
Json GetDocumentInfos ()
```

描述:

获取文档的参数信息。该接口的返回值就是 SetDocumentInfos 设置的参数。

参数:

无

返回值:

数据类型	说明	备注
Json	返回文档的参数信息	

代码示例:

```
let info=ctl. GetDocumentInfos ();
```

8.10.41. DocumentSettingsDialog

接口:

```
Void DocumentSettingsDialog ()
```

描述:

显示文档的页面设置对话框。在该对话框中可设置。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj. DocumentSettingsDialog ();
```

8.10.42. DocumentGutterDialog

接口:

Void DocumentGutterDialog()

描述:

显示文档的装订线对话框。在该对话框中可设置。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj. DocumentGutterDialog ();
```

8.10.43. DocumentGridLineDialog

接口:

```
Void DocumentGridLineDialog ()
```

描述:

显示文档的网格线对话框。在该对话框中可设置。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj. DocumentGridLineDialog ();
```

8.10.44. WatermarkDialog

接口:

```
Void WatermarkDialog ()
```

描述:

显示文档的水印设置对话框。在该对话框中可设置。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj. WatermarkDialog ();
```

8.10.45. SynchroServerTimeByParameters

接口:

```
void SynchroServerTimeByParameters(int year, int month, int day, int hour24, int minute, int second)
```

描述:

本函数用于对不支持 DateTime 类型的编程语言的支持。本函数不会修改本地计算机时钟, 而是修改本软件内部维护的一个虚拟时钟。虚拟时钟会依赖本地计算机时钟。不过同步服务器时间后, 如果本机时钟修改了, 则需要重新调用本函数来同步服务器时间。如果不重新同步则会采用本机时刻。因此建议定期 (比如一分钟) 来调用本函数同步服务器时间。

参数:

参数名	是否必填	数据类型	说明

year	必填	Int	年
month	必填	Int	月
day	必填	Int	日
hour24	必填	Int	24 小时的时
minute	必填	Int	分
second	必填	Int	秒

返回值:

无

代码示例:

```
var result = ctl.GetCurrentParagraphStyle ();
```

8.10.46. CheckboxAndRadioDialog

接口:

```
Void CheckboxAndRadioDialog ()
```

描述:

显示当前单选框或复选框的属性对话框。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj. CheckboxAndRadioDialog ();
```

8.10.47. InputFieldDialog

接口:

Void InputFieldDialog ()

描述:

显示当前输入域的属性对话框。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj. InputFieldDialog ();
```

8.10.48. ButtonDialog

接口:

Void ButtonDialog ()

描述:

显示当前按钮元素的属性对话框。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj. ButtonDialog ();
```

8.10.49. HorizontalLineDialog

接口:

Void HorizontalLineDialog ()

描述:

显示当前水平线元素的属性对话框。弹出的对话框如下:

水平分割线属性

编号:

hr1

可见性表达式:

示例

颜色:

线条样式:

Solid

线条粗细:

10

线条长度:

100

确认

取消

参数:

无

返回值:

无

代码示例:

```
obj.HorizontalLineDialog ();
```

8.10.50. PageNumberDialog

接口:

Void PageNumberDialog ()

描述:

显示当前页码元素的属性对话框。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj. PageNumberDialog ();
```

8.10.51. LabelDialog

接口:

Void LabelDialog ()

描述:

显示当前标签元素的属性对话框。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj. LabelDialog ();
```

8.10.52. QRCodeDialog

接口:

Void QRCodeDialog ()

描述:

显示当前二维码元素的属性对话框。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj. QRCodeDialog ();
```

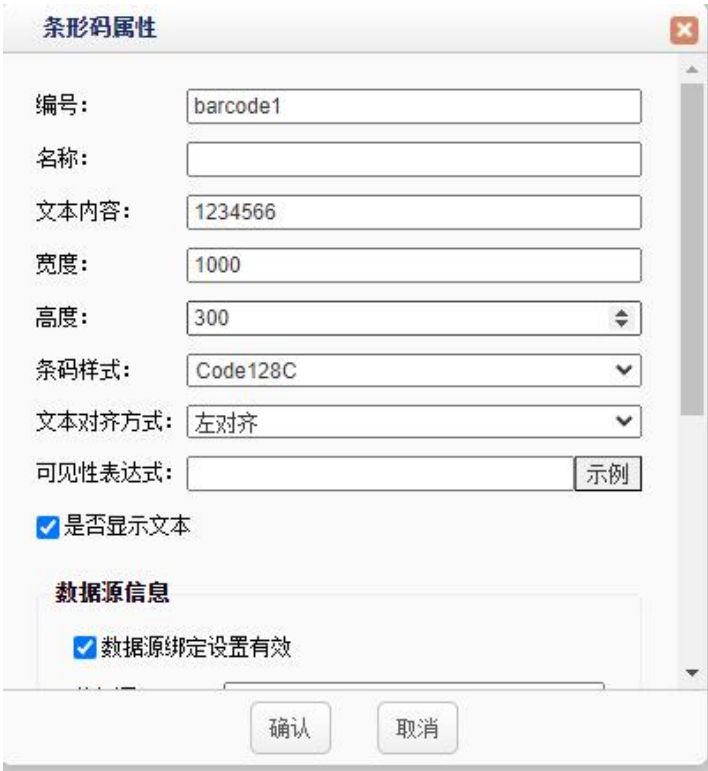
8.10.53. BarCodeDialog

接口:

Void BarCodeDialog ()

描述:

显示当前条形码元素的属性对话框。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj.BarCodeDialog ();
```

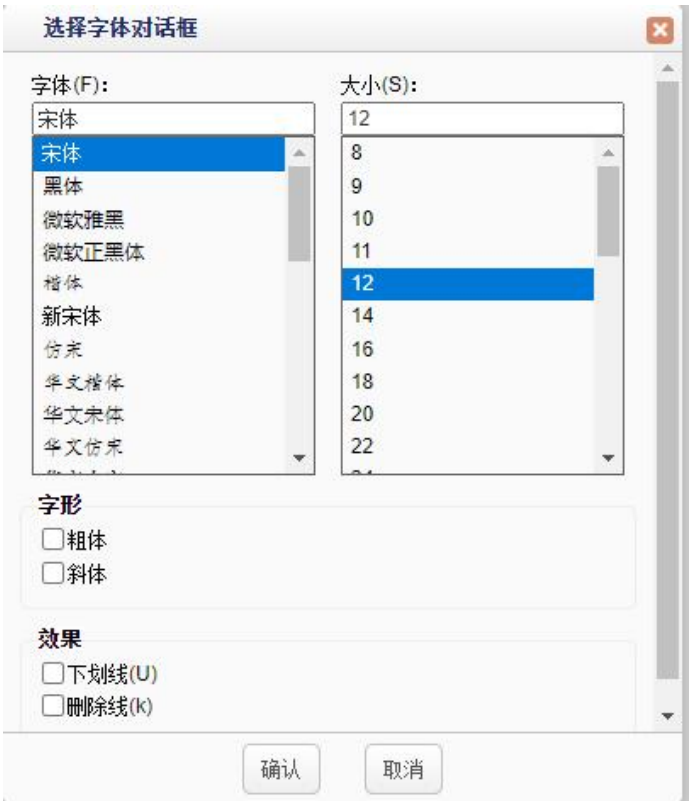
8.10.54. FontSelectionDialog

接口:

```
Void FontSelectionDialog ()
```

描述:

显示当前字体格式的属性对话框。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj. FontSelectionDialog ();
```

8.10.55. FetalHeartDialog

接口:

Void FetalHeartDialog (json element)

描述:

显示当前胎心图的属性对话框。弹出的对话框如下:



参数:

参数为当前的胎心图的属性

返回值:

无

代码示例:

```
obj.FetalHeartDialog ({});
```

8.10.56. PainIndexDialog

接口:

Void PainIndexDialog (json element)

描述:

显示当前标尺元素的属性对话框。弹出的对话框如下:



参数:

参数为当前的标尺元素的属性

返回值:

无

代码示例:

```
obj.CheckboxAndRadioDialog ({});
```

8.10.57. FractionDialog

接口:

Void FractionDialog (json element)

描述:

显示当前分数公式的属性对话框。弹出的对话框如下:



参数:

参数为当前的分数公式

返回值:

无

代码示例:

```
obj. FractionDialog ({});
```

8.10.58. FourValuesDialog

接口:

Void FourValuesDialog (json element)

描述:

显示当前月经史公式的属性对话框。弹出的对话框如下:



参数:

参数是当前的月经史公式信息

返回值:

无

代码示例:

```
obj. FourValuesDialog ({});
```

8.10.59. FourValues1Dialog

接口:

Void FourValues1Dialog (json element)

描述:

显示当前单选框或复选框的属性对话框。弹出的对话框如下:

参数:

参数是当前的月经史公式信息

返回值:

无

代码示例:

```
obj. FourValues1Dialog ({});
```

8.10.60. FourValues2Dialog

接口:

Void FourValues2Dialog (json element)

描述:

显示当前单选框或复选框的属性对话框。弹出的对话框如下:

参数:

参数是当前的月经史公式信息

返回值:

无

代码示例:

```
obj. FourValues2Dialog ({});
```

8.10.61. FourValues3Dialog

接口:

Void FourValues3Dialog (json element)

描述:

显示当前单选框或复选框的属性对话框。弹出的对话框如下:

参数:

参数是当前的月经史公式信息

返回值:

无

代码示例:

```
obj. FourValues3Dialog ({});
```

8.10.62. LightPositioningDialog

接口:

```
Void LightPositioningDialog (json element)
```

描述:

显示当前光定位图公式的属性对话框。弹出的对话框如下:

参数:

参数是当前的光定位信息

返回值:

无

代码示例:

```
obj. LightPositioningDialog ({});
```

8.10.63. PupilDialog

接口:

```
Void PupilDialog (json element)
```

描述:

显示当前瞳孔图的属性对话框。弹出的对话框如下:

参数:

参数为当前的瞳孔图信息

返回值:

无

代码示例:

```
obj. PupilDialog ({});
```

8.10.64. borderShadingcellDialog

接口:

Void borderShadingcellDialog ()

描述:

显示单元格边框设置对话框。弹出的对话框如下:



参数:

参数是当前单元格信息

返回值:

无

代码示例:

```
obj. borderShadingcellDialog ();
```


8.10.65. splitCellDialog

接口:

```
Void splitCellDialog ()
```

描述:

显示拆分当前单元格的对话框。弹出的对话框如下:



参数:

参数是当前的单元格信息

返回值:

无

代码示例:

```
obj. splitCellDialog ();
```

8.10.66. EditDocumentCommentsDialog

接口:

```
Void EditDocumentCommentsDialog ()
```

描述:

显示当前批注信息对话框。弹出的对话框如下:

编辑文档批注

批注内容

颜色设置

背景颜色:

前景颜色:

作者

姓名:

编号:

自定义属性

名称

值

操作

删除

添加

清空

确认

取消

参数:

参数是当前批注信息

返回值:

无

代码示例:

```
obj. EditDocumentCommentsDialog ();
```

8.10.67. formModeDialog

接口:

Void formModeDialog ()

描述:

显示当前表单模式设置的对话框。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj. formModeDialog ();
```

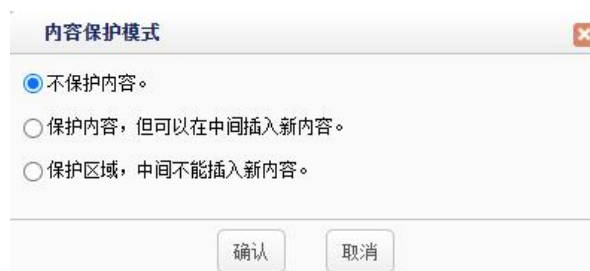
8.10.68. contentProtectedModeDialog

接口:

Void contentProtectedModeDialog ()

描述:

显示当前选择内容的内容保护设置对话框。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj. contentProtectedModeDialog ();
```

8.10.69. paragraphDialog

接口:

```
Void paragraphDialog ()
```

描述:

显示当前段落的属性信息对话框。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj.paragraphDialog ();
```

8.10.70. tableCellDialog

接口:

```
Void tableCellDialog ()
```

描述:

显示当前单元格属性对话框。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj. tableCellDialog ();
```

8.10.71. tableRowDialog

接口:

```
Void tableRowDialog ()
```

描述:

显示当前表格行的属性对话框。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj. tableRowDialog ();
```

8.10.72. cellGridlineDialog

接口:

```
Void cellGridlineDialog ()
```

描述:

显示当前单元格网格线设置对话框。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj.cellGridlineDialog ();
```

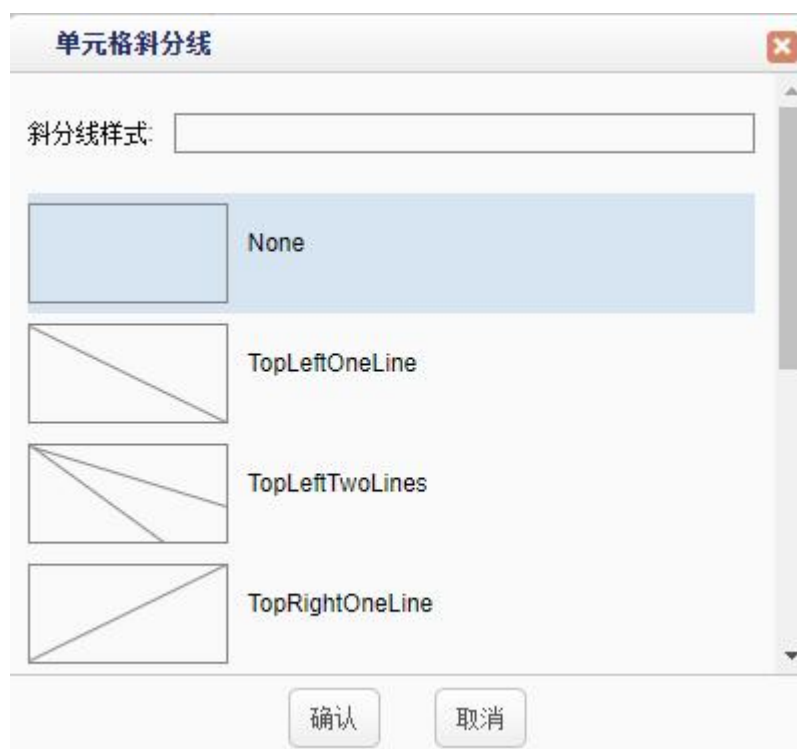
8.10.73. cellDiagonalLineDialog

接口:

Void cellDiagonalLineDialog ()

描述:

显示当前单元格斜分线的设置对话框。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj. cellDiagonalLineDialog ();
```

8.10.74. imgEditDialog

接口:

```
Void imgEditDialog (json element)
```

描述:

显示编辑图片的对话框。弹出的对话框如下:



参数:

参数是当前的图片信息

返回值:

无

代码示例:

```
obj. imgEditDialog ({});
```

8.10.75. loginDialog

接口:

Void loginDialog ()

描述:

显示用户登录信息设置对话框。弹出的对话框如下:



参数:

无

返回值:

无

代码示例:

```
obj. loginDialog ();
```

8.10.76. ImageDialog

接口:

```
Void ImageDialog ()
```

描述:

显示当前图片元素的属性对话框。弹出的对话框如下:



参数：
无

返回值：
无

代码示例：

```
obj. ImageDialog ();
```

8.10.77. ShowTooltip

接口：
Ctl.ShowTooltip

描述：
获取或设置结构化元素是否允许显示提示文本。在结构化元素的属性中包含 Tooltip，本属性可以控制全文的结构化元素是否显示提示文本。

参数：

Bool 值

返回值:

无

代码示例:

```
Ctl.ShowTooltip=true
```

8. 10. 78. *获取选中内容对应的数据

接口:

ctl.Selection

描述:

获取选择内容的信息。

参数:

无

返回值:

返回选择的内容信息，返回值是 json 格式，属性包含:

参数名	数据类型	说明
AbsEndIndex	Int	结束位置的序号
AbsStartIndex	Int	开始位置的序号
Cells	Array	选择的单元格数组对象
Length	Int	选择内容的长度
Mode	String	选择内容的类型，一般包含 cell 和 Content
SelectionParagraphFlags	Int	选择内容的段落数组对象
StartIndex	Int	开始位置的序号
Text	String	选择内容的文本信息
XMLText	String	选择内容的 XML 文本信息

代码示例:

```
let res=ctl.Selection;
```

相同效果:

`ctl.DocumentSelection()` 方法返回的值；跟 `var xx=ctl.Selection;xx.XmlText` 返回的值一致。

结合使用：

无

接口类型：

纯前端属性，不经过服务端

有无限制：

编辑器初始化成功之后都能调用。但是前提是用户需要先选择内容。注意打印预览模式下该属性无效。

兼容接口：

五代编辑器独有接口。

接口优化：

无任何区别

示例地址：

暂无

8.10.79. SelectedText

接口：

`Ctl. SelectedText`

描述：

获取选择内容的文本信息。

参数：

无

返回值：

返回选择的文本信息

代码示例：

```
Let res=Ctl. SelectedText
```

8.10.80. SelectionLength

接口:

Ctl. SelectionLength

描述:

获取选择内容的文本长度信息。

参数:

无

返回值:

返回选择的文本长度信息

代码示例:

```
Let res=Ctl. SelectionLength
```

8.10.81. SelectionStartPosition

接口:

Ctl. SelectionStartPosition

描述:

获取选择内容的开始位置信息。

参数:

无

返回值:

返回选择的开始位置信息

代码示例:

```
Let res=Ctl. SelectionStartPosition
```

8.10.82. DocumentTitle

接口:

Ctl. DocumentTitle

描述:

获取或设置文档的 Title 属性。该属性存储在 XML 节点的 Info 属性下的 Title 属性。

可关联 SetDocumentInfos、GetDocumentInfos 接口一起查看。

参数:

Bool 值

返回值:

无

代码示例:

```
Ctl. DocumentTitle = "AAA"  
  
Let res= Ctl. DocumentTitle
```

8.10.83. PageIndex

接口:

Ctl. PageIndex

描述:

获取当前页面的从 0 开始的页码数。

参数:

无

返回值:

Int

代码示例:

```
Let res=Ctl. PageIndex
```

8.10.84. PageCount

接口:

Ctl. PageCount

描述:

获取当前文档的总页数。

参数:

无

返回值:

Int

代码示例:

```
Let res=Ctl. PageCount
```

8.10.85. PageBackColorString

接口:

Ctl. PageBackColorString

描述:

获取控件显示的背景（非输入区域）颜色。

参数:

无

返回值:

String

代码示例:

```
Let res=Ctl. PageBackColorString
```

8.10.86. PageBorderColorString

接口:

Ctl. PageBorderColorString

描述:

获取控件显示的页面边框的颜色。

参数:

无

返回值:

String

代码示例:

```
Let res=Ctl. PageBorderColorString
```

8.10.87. CurrentPageBorderColor

接口:

Ctl. CurrentPageBorderColor

描述:

获取当前页面的页面边框的颜色。

参数:

无

返回值:

String

代码示例:

```
Let res=Ctl. CurrentPageBorderColor
```

8.10.88. CurrentPageBorderColorString

接口:

Ctl. CurrentPageBorderColorString

描述:

获取当前页面的页面边框的颜色。

参数:

无

返回值:

String

代码示例:

```
Let res=Ctl. CurrentPageBorderColorString
```

8.10.89. RuleEnabled

接口:

```
Ctl. RuleEnabled
```

描述:

获取或设置标尺是否可用

参数:

Bool 值

返回值:

无

代码示例:

```
Ctl. RuleEnabled=true  
  
Let res= Ctl. RuleEnabled
```

8.10.90. HeaderFooterFlagVisible

接口:

```
Ctl. HeaderFooterFlagVisible
```

描述:

获取或设置页眉页脚的标记

参数:

String

返回值:

String

代码示例:

```
Ctl. HeaderFooterFlagVisible=true  
  
Let res= Ctl. HeaderFooterFlagVisible
```

8.10.91. GrayingDisabledHeaderFooter

接口:

Ctl. GrayingDisabledHeaderFooter

描述:

获取或设置页眉页脚的区域是否以浅灰色覆盖。

参数:

String

返回值:

String

代码示例:

```
Ctl. GrayingDisabledHeaderFooter=true  
Let res= Ctl. GrayingDisabledHeaderFooter
```

8.10.92. MoveFocusHotKey

接口:

Ctl. MoveFocusHotKey

描述:

获取或设置移动焦点使用的快捷键。

参数:

String

返回值:

String

代码示例:

```
Ctl. MoveFocusHotKey= “tab”  
Let res= Ctl. MoveFocusHotKey
```

8.10.93. FocusedPageIndexBase0

接口:

Ctl. FocusedPageIndexBase0

描述:

获取当前光标所在的页码数, 从 0 开始计算。GetElementTextById

参数:

无

返回值:

Int

代码示例:

```
Var res=Ctl. FocusedPageIndexBase0
```

8.10.94. CurrentLineIndexInPage

接口:

Ctl. CurrentLineIndexInPage

描述:

获取当前光标所在行在当前页的序号, 从 1 开始计算。

参数:

无

返回值:

Int

代码示例:

```
Var res=Ctl. CurrentLineIndexInPage
```

8.10.95. CurrentColumnIndex

接口:

Ctl. CurrentColumnIndex

描述:

获取当前光标所在列在当前行的序号，从 1 开始计算。

参数:

无

返回值:

Int

代码示例:

```
Var res=Ctl. CurrentLineIndexInPage
```

8.10.96. CurrentFontName

接口:

Ctl. CurrentFontName

描述:

获取当前光标所在的字体名称。

参数:

无

返回值:

String

代码示例:

```
Var res=Ctl. CurrentLineIndexInPage
```

8.10.97. CurrentFontSize

接口:

Ctl. CurrentFontSize

描述:

获取当前光标所在的字体大小。

参数:

无

返回值:

Float

代码示例:

```
Var res=Ctl. CurrentFontSize
```

8.10.98. CurrentItalic

接口:

Ctl. CurrentItalic

描述:

获取当前光标所在的字体是否是斜体。

参数:

无

返回值:

Bool

代码示例:

```
Var res=Ctl. CurrentItalic
```

8.10.99. CurrentSubscript

接口:

Ctl. CurrentSubscript

描述:

获取当前光标所在的字体是否是下标样式。

参数:

无

返回值:

Bool

代码示例:

```
Var res=Ctl. CurrentSubscript
```

8.10.100. CurrentSuperscript

接口:

```
Ctl. CurrentSuperscript
```

描述:

获取当前光标所在的字体是否是上标样式。

参数:

无

返回值:

Bool

代码示例:

```
Var res=Ctl. CurrentSuperscript
```

8.10.101. CurrentUnderline

接口:

```
Ctl. CurrentUnderline
```

描述:

获取当前光标所在的字体是否是下划线样式。

参数:

无

返回值:

Bool

代码示例:

```
Var res=Ctl. CurrentUnderline
```

8.10.102. ForceShowCaret

接口:

Ctrl. ForceShowCaret

描述:

是否强制显示光标而不管控件是否获得输入焦点

参数:

Bool

返回值:

Bool

代码示例:

```
Var res=Ctrl. ForceShowCaret
```

8.10.103. Focus

接口:

bool Focus ()

描述:

设置编辑器获取输入焦点

参数:

无

返回值:

数据类型	说明	备注
Bool	设置是否成功	

代码示例:

```
Var res=Ctrl. Focus ()
```

8.10.104. MoveDownOneLine

接口:

Void MoveDownOneLine ()

描述:

移动光标到下一行

参数:

无

返回值:

无

代码示例:

```
Ctl. MoveDownOneLine ( )
```

8.10.105. MoveEnd

接口:

Void MoveEnd ()

描述:

移动光标到行尾

参数:

无

返回值:

无

代码示例:

```
Ctl. MoveEnd ( )
```

8.10.106. MoveHome

接口:

Void MoveHome ()

描述:

移动光标到行首

参数:

无

返回值:

无

代码示例:

```
Ctl. MoveHome ( )
```

8.10.107. MoveLeft

接口:

```
Void MoveLeft ( )
```

描述:

向左移动光标

参数:

无

返回值:

无

代码示例:

```
Ctl. MoveLeft ( )
```

8.10.108. MoveRight

接口:

```
Void MoveRight ( )
```

描述:

向右移动光标

参数:

无

返回值:

无

代码示例:

```
Ctl. MoveRight ()
```

8.10.109. MoveTo

接口:

Void MoveTo (string tag)

描述:

移动光标到指定位置。

参数:

参数名	是否必填	数据类型	说明
Tag	必填	String	字符串是 MoveTarget 枚举的字符串值，包含： None 无操作， DocumentHome 文档正文开头 DocumentEnd 文档正文结尾 CellHome 单元格开头 CellEnd 单元格结尾 PreCell 上一个单元格 NextCell 下一个单元格 BeforeTable 表格前 AfterTable 表格后 BeforeField 输入域外部前面 AfterField 输入域外部后面 ParagraphHome 段落开头 ParagraphEnd 段落结尾 PreLine 上一行 NextLine 下一行 FieldHome 输入域内部开头

			FieldEnd 输入域内部结尾
			Home 行的开头
			End 行的结尾
			PageHome 页面开头
			PageEnd 页面结尾
			ViewHome 当前视图最上面（暂时无效）
			ViewEnd 当前视图最下面（暂时无效）

返回值：

无

代码示例：

```
Ctrl. MoveTo ( “DocumentHome” )
```

8.10.110. MoveToClientPosition（待议）

接口：

描述：

参数：

参数名	是否必填	数据类型	说明

返回值：

数据类型	说明	备注

代码示例：

8.10.111. MoveToPosition（待议）

接口：

描述:

参数:

参数名	是否必填	数据类型	说明

返回值:

数据类型	说明	备注

代码示例:

8.10.112. MoveUpOneLine

接口:

```
Void MoveDownOneLine ( )
```

描述:

移动光标到上一行

参数:

无

返回值:

无

代码示例:

Ctl. MoveUpOneLine ()

8.10.113. MoveToSelectionSrart

接口:

```
Void MoveToSelectionSrart ( )
```

描述:

光标定位在选择内容的最前面

参数:

无

返回值:

无

代码示例:

```
Ctl. MoveToSelectionSrart ()
```

8.10.114. MoveToTableCellTag

接口:

Bool MoveToTableCellTag (json parameter)

描述:

定位到单元格的相对位置

参数:

参数名	是否必填	数据类型	说明
tableid	必填	String	表格编号
rowindex	必填	Int	行号
colindex	必填	Int	列号。根据行号和列号可以确定指定的单元格。
tag	必填	String	相对位置，参数包含： Beforetable 表格前,此时 colindex 和 rowindex 两个参数可以不输入。 Aftertable 表格后,,此时 colindex 和 rowindex 两个参数可以不输入。 Cellhome 单元格内部最前面， Cellend 单元格内部最后面， Prece11 前一个单元格， Aftercell 后一个单元格，

返回值:

属性名称	说明	备注
Bool	操作是否成功。	

代码示例:

```
let res=ctl. MoveToTableCellTag (str);
```

8.10.115. tableDialog

接口:

Void tableDialog ()

描述:

显示当前的表格属性对话框:



参数:

无

返回值:

无

代码示例:

```
obj. tableDialog ();
```

8.10.116. insertTableDialog

接口:

```
Void insertTableDialog ()
```

描述:

在当前位置插入表格对话框:



参数:

无

返回值:

无

代码示例:

```
obj.insertTableDialog ();
```

8.10.117. bordersShadingDialog

接口:

```
Void bordersShadingDialog ()
```

描述:

设置表格边框对话框:



参数:

无

返回值:

无

代码示例:

```
obj.bordersShadingDialog ();
```

8. 10. 118. *根据页码集合获取对应页图片数据

接口:

Array GetDocumentSpecifyPageImages(json options, function)

描述:

获取指定页码的图片数据。一般需要通过回调函数来实现获取数据。

参数:

参数名	是否必填	数据类型	说明
options	必填	Json	获取图片的参数，属性如下：
+showmarginline	必填	Bool	是否显示页面的页面边距标记。默认时，页面显示内容的时候，会在页面四角显示页面边距信息，如下图：

			都昌测试[提示：5天后本软件授权到期。] 。属性设置为 false，则生成的图片不带该标记。
+specifypageindexes	必填	String	需要获取图片的页码，从 0 开始计算。如果设置为 all 表示获取所有的；如何设置 3-6 表示从 3 到 6；如果时逗号分隔，则每一个部分代表一个页码。
function	必填	回调函数	需要通过回调函数实现获取数据

返回值：

数据类型	说明	备注
Array	图片的字节数组	

代码示例：

```
var opts = {
    showmarginline: false,
    specifypageindexes:"1"
};
ctl.GetDocumentSpecifyPageImages(opts, function (result) {
    for (var i = 0; i < result.length; i++) {
        console.log(result[i]); //获取第一张图片的base64数据
    }
});
```

相同效果：

无

结合使用：

无

接口类型：

纯前端接口，不经过服务端

有无限制：

编辑器加载成功之后在任何情况下都能调用。只读模式、阅读模式、打印预览模式获取出来的图片数据跟正常编辑模式获取出来的图片数据是一样的。且图片数据跟编辑模式下的效果一模一样，表格的虚边框一样会显示，只有批注不会在图片上显示

兼容接口：

兼容四代接口。

接口优化：

五代该接口方法参数中对比四代多了一个回调函数，返回图片数据五代编辑器是通过回调函数里面的参数进行返回的。

示例地址：

<http://117.89.85.111:8086/apiDocument/edit/index.html?apiname=GetDocumentSpecifyPageImages>

注意：

当编辑未注册时，生成的图片数据会带上未注册标识文字

8.10.119. SetDocumentGlobalJavaScript

接口：

Bool SetDocumentGlobalJavaScript (string strString)

描述：

设置文档的脚本信息。该接口会覆盖文档中原有的脚本。

参数：

参数名	是否必填	数据类型	说明
strString	必填	String	脚本信息

返回值：

数据类型	说明	备注
Bool	是否执行成功	

代码示例：

```
let res=ctl. SetDocumentGlobalJavaScript (str);
```

8.10.120. GetDocumentGlobalJavaScript

接口:

String GetDocumentGlobalJavaScript ()

描述:

获取文档中的脚本。

参数:

无

返回值:

数据类型	说明	备注
String	全文的脚本字符串	

代码示例:

```
let res=ctl. GetDocumentGlobalJavaScript ();
```

8.10.121. GetNowDateTime

接口:

String GetNowDateTime()

描述:

获取当前日期时间。该接口是通过后端代码获取的，不是通过 JS 获取。

参数:

无。

返回值:

数据类型	说明	备注
String	返回日期时间。默认格式是“2024-01-11 11:11:11”，但根据操作系统的设置不同，可能会有差异，如显示为“2024/01/11 11:11:11”，或其他格式。	

代码示例:

```
let res=ctl.GetNowDateTime();
```

8.10.122. 下拉输入域两栏展示

接口:

`DropDownList_TwoColumnDisplay="true"`

描述:

编辑器 div 中添加上面属性, 当下拉输入域同时设置 Text 和 Value 值, 下拉框自动为双栏展示。

参数:

无。

代码示例:

```
<div id="myWriterControl" dctype="WriterControlForWASM"
  DropDownList_TwoColumnDisplay="true"></div>
```

相同效果:

无。

结合使用:

无

接口类型:

纯前端属性, 不经过服务端

有无限制:

编辑器初始化成功之后都能调用。

兼容接口:

五代编辑器独有属性。

接口优化:

无任何区别

示例地址:

暂无

8.10.123. 高亮度显示文档中已经隐藏的元素

接口:

`bool DisplayAllVisibleElement()`

描述:

通过可见性表达式隐藏的元素，设置 Visible 属性隐藏的元素，调用方法可以全部高亮度显示。

参数：

参数名	是否必填	数据类型	说明
visileBackgroundColor	必填	String	通过 Visible 隐藏的元素，以哪种颜色高亮度显示
runtimeVisibleBackgrou ndColor	必填	String	通过可见性表达式隐藏的元素，以哪种颜色高亮度显示

代码示例：

```
var ctl=document.getElementById("myWriterControl")
ctl.DisplayAllVisibleElement("red","yellow");
```

相同效果：

无。

结合使用：

无

接口类型：

纯前端方法，不经过服务端

有无限制：

编辑器初始化成功之后都能调用。

兼容接口：

五代编辑器独有方法。

接口优化：

无任何区别

示例地址：

暂无

8. 10. 124. 隐藏通过 DisplayAllVisibleElement 方法高亮度显示的元素

接口：

```
bool HiddenAllVisibleElement()
```

描述：

DisplayAllVisibleElement 方法显示的元素，在次进行隐藏。

参数：

无

代码示例：

```
var ctl=document.getElementById("myWriterControl")
ctl.HiddenAllVisibleElement();
```

相同效果：

无。

结合使用：

无

接口类型：

纯前端方法，不经过服务端

有无限制：

编辑器初始化成功之后都能调用。

兼容接口：

五代编辑器独有方法。

接口优化：

无任何区别

示例地址：

暂无

8.10.125. 编辑器内置工具条

接口：

```
bool SetToolBarVisibility()
```

描述：

调用方法编辑器内置显示工具条，无须开发人员二次开发。

参数：

true：显示工具栏，false：隐藏工具栏

代码示例：

```
var ctl=document.getElementById("myWriterControl")
ctl.SetToolBarVisibility(true);
```

相同效果：

无。

结合使用：

无

接口类型：

纯前端方法，不经过服务端

有无限制：

编辑器初始化成功之后都能调用。

兼容接口：

五代编辑器独有方法。

接口优化：

无任何区别

示例地址：

暂无

9. 前端命令

通过调用方法 `DCExecuteCommand` 方法可以实现编辑器下的格式功能、比如粗体、前景色、字体、字号、上标等功能、也能通过这个方法插入结构化元素：输入域、标签文本、单、复选框、图片、表格、条形码、二维码等。

9.1. 开启管理员模式

描述：编辑器开启管理员模式：用户除了受到编辑器控件的内容只读限制（`Readonly` 属性为 `true`）及表单视图模式的限制外，其他没有任何限制，比如授权只读控制、输入域只读控制、内容保护、文档签名保护等等统统无效。

代码示例：


```
ctl.DCExecuteCommand('AdministratorViewMode', false, null)
```

备注：无。

9.2. 单元格底端居中对齐

描述：单元格底端居中对齐。

代码示例：

```
ctl.DCExecuteCommand('AlignBottomCenter', false, null)
```

备注：无。

9.3. 单元格底端左对齐

描述：单元格底端左对齐。

代码示例：

```
ctl.DCExecuteCommand('AlignBottomLeft', false, null)
```

备注：无。

9.4. 单元格底端右对齐

描述：单元格底端右对齐。

代码示例：

```
ctl.DCExecuteCommand('AlignBottomRight', false, null)
```

备注：无。

9.5. 段落居中对齐

描述：段落居中对齐。

代码示例：

```
ctl.DCExecuteCommand('AlignCenter', false, null)
```

备注：无。

9.6. 段落分散对齐

描述：段落分散对齐。

代码示例：

```
ctl.DCExecuteCommand('AlignDistribute', false, null)
```

备注：无。

9.7. 段落左对齐

描述：段落左对齐。

代码示例：

```
ctl.DCExecuteCommand('AlignLeft', false, null)
```

备注：无。

9.8. 单元格垂直水平居中

描述：单元格垂直水平居中。

代码示例：

```
ctl.DCExecuteCommand('AlignMiddleCenter', false, null)
```

备注：无。

9.9. 单元格垂直水平左对齐

描述：单元格垂直水平左对齐。

代码示例：

```
ctl.DCExecuteCommand('AlignMiddleLeft', false, null)
```

备注：无。

9.10. 单元格垂直水平右对齐

描述：单元格垂直水平右对齐。

代码示例：

```
ctl.DCExecuteCommand('AlignMiddleRight', false, null)
```

备注：无。

9.11. 段落右对齐

描述：段落右对齐。

代码示例：

```
ctl.DCExecuteCommand('AlignRight', false, null)
```

备注：无。

9.12. 单元格顶端居中对齐

描述：单元格顶端居中对齐。

代码示例：

```
ctl.DCExecuteCommand('AlignTopCenter', false, null)
```

备注：无。

9.13. 单元格顶端居左对齐

描述：单元格顶端居左对齐。

代码示例：

```
ctl.DCExecuteCommand('AlignTopLeft', false, null)
```

备注：无。

9.14. 单元格顶端居右对齐

描述：单元格顶端居右对齐。

代码示例：

```
ctl.DCExecuteCommand('AlignTopRight', false, null)
```

备注：无。

9.15. 全局修改字体名称

描述：全局修改字体名称。

代码示例：

```
ctl.DCExecuteCommand('allfontname', false, null)
```

备注：无。

9.16. 全局修改字体大小

描述：全局修改字体大小。

代码示例：

```
ctl.DCExecuteCommand('allfontsize', false, null)
```

备注：无。

9.17. 对文档中没有痕迹的信息附加当前用户的痕迹。

描述：对文档中没有痕迹的信息附加当前用户的痕迹。

代码示例：

```
ctl.DCExecuteCommand('AttachCurrentUserTrackToBodyContent', false, null)
```

备注：无。

9.18. 设置背景色

描述：设置背景色。

代码示例：

```
ctl.DCExecuteCommand('BackColor', false, null)
```

备注：无。

9.19. 设置后台运行模式。

描述：设置后台运行模式：内容保护机制无效、三级权限控制无效。不过痕迹保留机制还在运行。

代码示例：

```
ctl.DCExecuteCommand('BackgroundMode', false, null)
```

备注：无。

9.20. 退格删除

描述：退格删除。

代码示例：

```
ctl.DCExecuteCommand('Backspace', false, null)
```

备注：无。

9.21. 对选中的内容设置粗体

描述：对选中的内容设置粗体。

代码示例：

```
ctl.DCExecuteCommand('Bold', false, null)
```

备注：无。

9.22. 设置圆点列表

描述：设置圆点列表。

代码示例：

```
ctl.DCExecuteCommand('BulletedList', false, null)
```

备注：无。

9.23. 字符套圈

描述：字符套圈。

代码示例：

```
ctl.DCExecuteCommand('CharacterCircle', true, null)
```

备注：无。

9.24. 整洁视图模式-清洁视图模式

描述：整洁视图模式：把文档中的痕迹给隐藏不进行展示。

代码示例：

```
ctl.DCExecuteCommand('CleanViewMode', false, null)
```

备注：在编辑器只读的时候可以进行清洁-复杂模式切换。

9.25. 清空文档中所有输入域内容

描述：清空文档中所有输入域内容：只清空正文里面的输入域内容，且当输入域元素设置了内容只读、不允许用户编辑清空不了、编辑器设置只读、阅读模式、打印预览模式清空不了。

代码示例：

```
ctl.DCExecuteCommand('ClearAllFieldValue', false, null)
```

备注：无。

9.26. 清空文档中所有用户书写的痕迹

描述：清空文档中所有用户书写的痕迹，被逻辑删除的内容，只会把逻辑删除的样式给清除掉，但是逻辑删除的内容还保留，该操作不可恢复。

前提：需要开启管理员模式

代码示例：

```
ctl.DCExecuteCommand('ClearAllUserTrace', false, null)
```

备注：无。

9.27. 清空文档中正文的内容

描述：清空文档中正文的内容，当编辑器通过 Readonly 设置全文只读，该命令无效。

代码示例：

```
ctl.DCExecuteCommand('ClearBodyContent', false, null)
```

备注：无。

9.28. 清空当前光标处输入域的内容

描述：清空当前光标处所在输入域的内容，当输入域元素设置了内容只读、不允许用户编辑清空不了、编辑器设置只读、阅读模式、打印预览模式清空不了。

代码示例：

```
ctl.DCExecuteCommand('ClearCurrentFieldValue', false, null)
```

备注：无。

9.29. 退出续打模式

描述：退出续打模式。

代码示例：

```
ctl.DCExecuteCommand('ClearJumpPrintMode', false, null)
```

备注：无。

9.30. 清除撤销信息操作列表

描述：清除撤销信息操作列表。

代码示例：

```
ctl.DCExecuteCommand('ClearUndo', false, null)
```

备注：无。

9.31. 清除用户留下的痕迹

描述：清除用户留下的痕迹，被逻辑删除的内容，只会把逻辑删除的样式给清除掉，但是逻辑删除的内容还保留，该操作可以恢复。

前提：需要开启管理员模式

代码示例：

```
ctl.DCExecuteCommand('ClearUserTrace', false, null)
```

备注：无。

9.32. 清空校验结果

描述：清空校验结果

代码示例：

```
ctl.DCExecuteCommand('ClearValueValidateResult', false, null)
```

备注：无。

9.33. 设置前景色

描述：设置前景色

代码示例：

```
ctl.DCExecuteCommand('Color', false, 'red')
```

备注：无。

9.34. 提交所有的用户痕迹信息效果

描述：提交所有的用户痕迹信息，用户痕迹全部提交（相当于清除掉），被逻辑删除的内容变成物理删除，xml 文档里面不存在该内容

前提：需要开启管理员模式

代码示例：

```
ctl.DCExecuteCommand('CommitUserTrace', false, null)
```

备注：无。

9.35. 复杂（留痕）视图模式

描述：复杂（留痕）视图模式：把文档中的痕迹给显示出来。

代码示例：

```
ctl.DCExecuteCommand('ComplexViewMode', false, null)
```

备注：开启复杂视图模式，需要执行了用户登录接口才行，否则编辑器打开一个带有痕迹的文档，然后不执行登录接口，对病历操作会产生痕迹不准的问题，在编辑器只读的时候可以进行清洁-复杂模式切换。

9.36. 设置内容保护

描述：设置内容保护。

代码示例：

```
ctl.DCExecuteCommand('ContentProtect', true, null)
```

备注：无。

9.37. 将选中的文档内容转换成输入域对象

描述：将选中的文档内容转换成输入域对象。

代码示例：

```
ctl.DCExecuteCommand('ConvertContentToField', false, null)
```

备注：无。

9.38. 将当前光标所在的输入域转成普通文本

描述：将当前光标所在的输入域转成普通文本。

代码示例：

```
ctl.DCExecuteCommand('ConvertFieldToContent', false, null)
```

备注：无。

9.39. 文档内容转换为表单元素的标签文本

描述：文档内容转换为表单元素的标签文本。

代码示例：

```
ctl.DCExecuteCommand('ConvertTextContentToElementLabel', false, null)
```

```
ctl.DCExecuteCommand('ConvertTextContentToElementLabel', false, {Bold: true})//转换之
```

后文本标签元素自动加粗

备注：无。

9.40. 复制

描述：复制。

代码示例：

```
ctl.DCExecuteCommand('Copy', false, null)
```

备注：无。

9.41. 纯文本复制

描述：纯文本复制。

代码示例：

```
ctl.DCExecuteCommand('CopyAsText', false, null)
```

备注：无。

9.42. 剪切

描述：剪切。

代码示例：

```
ctl.DCExecuteCommand('Cut', false, null)
```

备注：无。

9.43. 插入图片

描述：插入图片。

代码示例：

```
ctl.DCExecuteCommand('DCInsertImage', false, null)
```

备注：无。

9.44. 删除

描述：当开启痕迹功能，该删除为逻辑删除。

代码示例：

```
ctl.DCExecuteCommand('Delete', false, null)
```

备注：无。

9.45. 物理删除

描述：物理删除。

代码示例：

```
ctl.DCExecuteCommand('DeleteAbsolute', false, null)
```

备注：无。

9.46. 删除所有批注

描述：删除所有批注。

代码示例：

```
ctl.DCExecuteCommand('DeleteAllComment', false, null)
```

备注：无。

9.47. 除当前光标处的批注

描述：删除当前光标处的批注。

代码示例：

```
ctl.DCExecuteCommand('DeleteComment', false, null)
```

备注：无。

9.48. 删除文本输入域

描述：删除文本输入域：但当输入域元素设置不允许删除，该命令无效。

代码示例：

```
ctl.DCExecuteCommand('DeleteField', false, null)
```

备注：无。

9. 49. 删除当前光标处的一行文本

描述：删除当前光标处的一行文本。

代码示例：

```
ctl.DCExecuteCommand('DeleteLine', false, null)
```

备注：无。

9. 50. 删除文档后面多余的空白行

描述：删除文档后面多余的空白行。

代码示例：

```
ctl.DCExecuteCommand('DeleteRedundant', false, null)
```

备注：无。

9. 51. 删除当前光标处文档节

描述：删除当前光标处文档节--可以删除病程记录。

代码示例：

```
ctl.DCExecuteCommand('DeleteSection', false, null)
```

备注：无。

9. 52. 设计模式

描述：设计模式：下拉输入域处于设计模式下失效，表达式功能也失效。

代码示例：

```
ctl.DCExecuteCommand('DesignMode', false, null)
```

备注：无。

9. 53. 显示所有隐藏的输入域

描述：显示所有隐藏的输入域。

代码示例：

```
ctl.DCExecuteCommand('DisplayWhole', false, null)
```

备注：无。

9.54. 修改文档默认的字体样式

描述：修改文档默认的字体样式。

代码示例：

```
var obj = {  
  "fontFamily": '幼圆',  
  "fontSize": 24  
}
```

```
ctl.DCExecuteCommand('DocumentDefaultFont', false,obj)
```

备注：无。

9.55. 文档中的数据进行校验

描述：文档中的数据进行校验。

代码示例：

```
ctl.DCExecuteCommand('DocumentValueValidate', false, null)
```

备注：无。

9.56. 批注式校验

描述：对文档中的数据进行校验，并通过批注的方式进行显示。

代码示例：

```
ctl.DCExecuteCommand('DocumentValueValidateWithCreateDocumentComments', false, null)  
ll)
```

备注：无。

9.57. 元素属性对话框

描述：元素属性对话框。

代码示例：

```
ctl.DCExecuteCommand('ElementProperties', true, null)
```

备注：无。

9.58. 将选中的内容设置着重号

描述：将选中的内容设置着重号。

代码示例：

```
ctl.DCExecuteCommand('EmphasisMark', false, null)
```

备注：无。

9.59. 命令对话框

描述：弹出命令对话框，当编辑器处于阅读模式、只读模式的时候，`shift+f12` 无效时，可以调用命令来弹出命令对话框。

代码示例：

```
ctl.DCExecuteCommand('ExecuteCommand', true, null)
```

备注：无。

9.60. 取消/设置输入域固定宽度

描述：取消/设置输入域固定宽度。

代码示例：

```
ctl.DCExecuteCommand('FieldSpecifyWidth', false, null)
```

备注：无。

9.61. 新建空白文档

描述：新建空白文档。

代码示例：

```
ctl.DCExecuteCommand('FileNew', false, null)
```

备注：无。

9.62. 加载本地文档

描述：加载本地文档。

代码示例：

```
ctl.DCExecuteCommand('FileOpen', false, null)
```

备注：支持回调函数，加载文档后触发回调函数实现特殊业务需求。

```
var CallBackFunction = function () {  
    console.log("%cFileOpen=>文档加载成功!");  
} //支持第四个参数传回调函数
```

9.63. 首行缩进

描述：首行缩进。

代码示例：

```
ctl.DCExecuteCommand('FirstLineIndent', false, null)
```

备注：无。

9.64. 选中一段内容设置字体

描述：选中一段内容设置字体。

代码示例：

```
ctl.DCExecuteCommand('Font', true, null)
```

备注：无。

9.65. 设置字符边框

描述：设置字符边框。

代码示例：

```
ctl.DCExecuteCommand('Fontborder', false, null)
```

备注：无。

9.66. 选中一段内容设置字体名称

描述：选中一段内容设置字体名称。

代码示例：

```
ctl.DCExecuteCommand('FontName', false, '楷体')
```

备注：无。

9.67. 选中一段内容设置字体大小

描述：选中一段内容设置字体大小。

代码示例：

```
ctl.DCExecuteCommand('FontSize', false, 24)
```

备注：无。

9.68. 格式刷

描述：格式刷。

代码示例：

```
ctl.DCExecuteCommand('FormatBrush', false, null)
```

备注：无。

9.69. 段落悬挂缩进

描述：段落悬挂缩进。

代码示例：

```
var options = {  
    firstlineindent: "236",//首行缩进--单位：1/300 英寸  
    leftindent: "472"//左侧缩进--单位：1/300 英寸  
};
```

```
ctl.DCExecuteCommand('hangingIndent', false, options)
```

备注：无。

9.70. 表单模式

描述：表单模式。

代码示例：

```
ctl.DCExecuteCommand('FormViewMode', true, null)
```

备注：无。

9.71. 设置标题的数字列表 1 样式

描述：设置标题的数字列表 1 样式。

代码示例：

```
ctl.DCExecuteCommand('Header1', false, null)
```

备注：无。

9.72. 设置标题的数字列表 1 样式

描述：设置标题的数字列表 1 样式。

代码示例：

```
ctl.DCExecuteCommand('Header1WithMultiNumberlist', false, null)
```

备注：无。

9.73. 隐藏被标记为自动隐藏的元素

描述：隐藏被标记为自动隐藏的元素。

代码示例：

```
ctl.DCExecuteCommand('HideElementMarkAutoHide', false, null)
```

备注：无。

9.74. 段落缩进

描述：段落缩进。

代码示例：

```
ctl.DCExecuteCommand('Indent', false, null)
```

备注：无。

9. 75. 设置或取消输入域的下划线

描述：设置或取消输入域的下划线。

代码示例：

```
ctl.DCExecuteCommand('InputFieldUnderLine', false, null)
```

备注：无。

9. 76. 插入会计数字----接口文档不显示。

描述：插入会计数字。

代码示例：

```
ctl.DCExecuteCommand('InsertAccountingNumber', false, null)
```

备注：无。

9. 77. 插入条形码

描述：插入条形码。

代码示例：

```
ctl.DCExecuteCommand('InsertBarcodeElement', true, null)
```

备注：无。

9. 78. 插入按钮

描述：插入按钮。

代码示例：

```
ctl.DCExecuteCommand('InsertButton', true, null)
```

备注：无。

9.79. 插入折现图

描述：插入折现图。

代码示例：

```
ctl.DCExecuteCommand('InsertChartElement', true, null)
```

备注：无。

9.80. 插入单、复选框组

描述：插入单、复选框组。

代码示例：

```
ctl.DCExecuteCommand('insertcheckboxorradio', true, null)
```

备注：

"ValueBinding":{"DataSource": "default","BindingPath": "BAA010"}//设置数据源绑定，注意大小写，第三个参数中定义数据源绑定。

9.81. 插入批注

描述：插入批注。

代码示例：

```
ctl.DCExecuteCommand('InsertComment', true, null)
```

备注：无。

9.82. 插入时间输入域

描述：插入时间输入域。

代码示例：

```
ctl.DCExecuteCommand('InsertDateTimeField', true, null)
```

备注：无。

9.83. 插入时间字符串

描述：插入时间字符串。

代码示例：

```
ctl.DCExecuteCommand('InsertDateTimeString', false, null)
```

备注：无。

9.84. 插入水平线

描述：插入水平线。

代码示例：

```
ctl.DCExecuteCommand('InsertHorizontalLine', true, null)
```

备注：无。

9.85. 插入 html 内容

描述：插入 html 内容。

代码示例：

```
ctl.DCExecuteCommand('InsertHtml', false,'html 片段')
```

备注：无。

9.86. 插入图片

描述：插入图片。

代码示例：

```
ctl.DCExecuteCommand('InsertImage', true,null)
```

备注：无。

9.87. 插入输入域元素

描述：插入输入域元素。

代码示例：

```
ctl.DCExecuteCommand('InsertInputField', true,null)
```

备注：无。

9.88. 插入标签文本元素

描述：插入标签文本元素。

代码示例：

```
ctl.DCExecuteCommand('InsertLabelElement', true,null)
```

备注：无。

9.89. 插入软回车

描述：插入软回车

代码示例：

```
ctl.DCExecuteCommand('InsertLineBreak', false,null)
```

备注：无。

9.90. 插入视频音频元素

描述：插入视频音频元素。

代码示例：

```
var options = {  
    Width: '1871',  
    Height: '1000',  
    PrintVisibility: 'none',  
    FileName: 'http://www.dcwritter.cn/static/images/websiteexplain.mp4',  
}
```

```
ctl.DCExecuteCommand('InsertMediaElement', false,options)
```

备注：无。

9.91. 插入医学表达式

描述：插入医学表达式。

代码示例：

```
var options = {  
    "ID": "expression1",  
    "ExpressionStyle": "FourValues1",  
    "Type": "XTextNewMedicalExpressionElement",  
    "FontSize": "12",  
    "Width": "112px",  
    "Height": "46px",  
    "Values": "Value1:14;Value2:14;Value3:14;Value4:14"  
};
```

```
ctl.DCExecuteCommand('InsertMedicalExpression', false,options)
```

备注：通过 ExpressionStyle 属性来插入不同的医学表达式的类型。

9.92. 改成读/写模式

描述：改成读/写模式

代码示例：

```
ctl.DCExecuteCommand('InsertMode', false,null)
```

备注：点击键盘 Insert 键相同效果。

9.93. 设置有序列表

描述：设置有序列表

代码示例：

```
ctl.DCExecuteCommand('Insertorderedlist', false,null)
```

备注：无。

9.94. 插入分页符

描述：插入分页符

代码示例：

```
ctl.DCExecuteCommand('InsertPageBreak', false,null)
```

备注：无。

9.95. 插入页码元素

描述：插入页码元素

代码示例：

```
ctl.DCExecuteCommand('InsertPageInfoElement', true,null)
```

备注：无。

9.96. 插入段落符号

描述：插入段落符号

代码示例：

```
ctl.DCExecuteCommand('InsertParagrahFlag', false,null)
```

备注：无。

9.97. 插入标尺元素

描述：插入标尺

代码示例：

```
ctl.DCExecuteCommand('InsertRuler', false,null)
```

备注：无。

9.98. 插入文档节

描述：插入文档节

代码示例：

```
ctl.DCExecuteCommand('InsertSection', false,null)
```

备注：无。

9.99. 插入字符串数据

描述：插入字符串数据

代码示例：

```
ctl.DCExecuteCommand('InsertString', false, "哈哈哈哈哈\r\n 啊实打实大")
```

备注：不支持 html 数据。

9.100. 插入表格

描述：插入表格

代码示例：

```
ctl.DCExecuteCommand('InsertTable',true,null)
```

备注：无。

9.101. 插入二维码

描述：插入二维码

代码示例：

```
ctl.DCExecuteCommand('InsertTDBarcodeElement',true,null)
```

备注：无。

9.102. 插入无序列表

描述：插入无序列表

代码示例：

```
ctl.DCExecuteCommand('Insertunorderedlist',false,null)
```

备注：无。

9.103. 插入空格使后面的内容右对齐

描述：插入空格使后面的内容右对齐

代码示例：

```
ctl.DCExecuteCommand('InsertWhiteSpaceForAlignRight',false,null)
```

备注：无。

9.104. 插入 xml 小片段内容

描述：插入 xml 小片段内容

代码示例：

```
ctl.DCExecuteCommand('InsertXML',false,'xml 小片段')
```

备注：无。

9.105. 设置斜体

描述：设置斜体

代码示例：

```
ctl.DCExecuteCommand('Italic',false,null)
```

备注：无。

9.106. 开启续打模式

描述：开启续打模式

代码示例：

```
ctl.DCExecuteCommand('JumpPrintMode',false,null)
```

备注：无。

9.107. 全局修改行间距

描述：设置行间距

代码示例：

```
ctl.DCExecuteCommand('alllineheight', false, "2");
```

备注：无。

9.108. 设置行间距

描述：设置行间距

代码示例：

```
ctl.DCExecuteCommand('lineheight',false,'2')
```

备注：无。

9.109. 向下移动一行

描述：向下移动一行

代码示例：

```
ctl.DCExecuteCommand('MoveDown',false,null)
```

备注：无。

9.110. 移动到行尾

描述：移动到行尾

代码示例：

```
ctl.DCExecuteCommand('MoveEnd',false,null)
```

备注：无。

9.111. 移动到行首

描述：移动到行首

代码示例：

```
ctl.DCExecuteCommand('MoveHome',false,null)
```

备注：无。

9.112. 向左移动一列

描述：向左移动一列

代码示例：

```
ctl.DCExecuteCommand('MoveLeft',false,null)
```

备注：无。

9.113. 向下翻页

描述： 向下翻页

代码示例：

```
ctl.DCExecuteCommand('MovePageDown',false,null)
```

备注：无。

9.114. 向上翻页

描述： 向上翻页

代码示例：

```
ctl.DCExecuteCommand('MovePageUp',false,null)
```

备注：无。

9.115. 向右移动一列

描述： 向右移动一列

代码示例：

```
ctl.DCExecuteCommand('MoveRight',false,null)
```

备注：无。

9.116. 移动插入点位置

描述： 移动插入点位置

代码示例：

```
ctl.DCExecuteCommand('MoveTo',false,'Home')
```

备注：无。

9.117. 移动插入点位置

描述： 移动插入点位置

代码示例：

```
ctl.DCExecuteCommand('MoveToPage',false,1)
```

备注：无。

9.118. 移动插入点到指定的位置

描述：移动插入点到指定的位置

代码示例：

```
ctl.DCExecuteCommand('MoveToPosition',false,100)
```

备注：无。

9.119. 向上移动一行

描述：向上移动一行

代码示例：

```
ctl.DCExecuteCommand('MoveUp',false,null)
```

备注：无。

9.120. 普通视图模式

描述：普通视图模式

代码示例：

```
ctl.DCExecuteCommand('NormalViewMode',false,null)
```

备注：无。

9.121. 数字列表

描述：数字列表

代码示例：

```
ctl.DCExecuteCommand('NumberedList',false,null)
```

备注：无。

9.122. 偏移续打模式

描述：偏移续打模式

代码示例：

```
ctl.DCExecuteCommand('OffsetJumpPrintMode',false,null)
```

备注：无。

9.123. 页面视图方式

描述：页面视图方式

代码示例：

```
ctl.DCExecuteCommand('PageViewMode',false,null)
```

备注：无。

9.124. 段落格式

描述：段落格式

代码示例：

```
ctl.DCExecuteCommand('ParagraphFormat',true,null)
```

备注：无。

9.125. 粘贴

描述：粘贴。只能在编辑器内部复制的内容，执行该命令才有效果，外部复制的内容，调用该命令得通过 https 访问的项目才行

代码示例：

```
ctl.DCExecuteCommand('Paste',false,null)
```

备注：无。

9.126. 纯文本粘贴

描述：纯文本粘贴。只能在编辑器内部复制的内容，执行该命令才有效果，外部复制的

内容，调用该命令得通过 https 访问的项目才行

代码示例：

```
ctl.DCExecuteCommand('PasteAsText',false,null)
```

备注：无。

9.127. 阅读版式视图方式

描述：阅读版式视图方式

代码示例：

```
ctl.DCExecuteCommand('ReadViewMode',false,null)
```

备注：无。

9.128. 撤销

描述：撤销

代码示例：

```
ctl.DCExecuteCommand('Redo',false,null)
```

备注：无。

9.129. 刷新文档

描述：刷新文档

代码示例：

```
ctl.DCExecuteCommand('RefreshDocument',false,null)
```

备注：无。

9.130. 撤销所有用户修订

描述：撤销所有用户修订

前提：管理员模式下

代码示例：

```
ctl.DCExecuteCommand('RejectUserTrace',false,null)
```

备注：无。

9.131. 清除字体，把选择的内容字体还原成宋体

描述：清除字体，把选择的内容字体还原成宋体

代码示例：

```
ctl.DCExecuteCommand('RemoveFontFamily',false,null)
```

备注：无。

9.132. 清除字号，把选择的内容字体还原成 12 号字体

描述：清除字号，把选择的内容字体还原成 12 号字体

代码示例：

```
ctl.DCExecuteCommand('Removefontsize',false,null)
```

备注：无。

9.133. 重置表单数据

描述：重置表单数据：

代码示例：

```
ctl.DCExecuteCommand('ResetFormValue',false,null)
```

备注：无。

9.134. 设置段前距、段后距

描述：设置段前距、段后距：

代码示例：

```
ctl.DCExecuteCommand('rowspacing',false,"10,top")//段前距
```

```
ctl.DCExecuteCommand('rowspacing',false,"10,bottom")//段后距
```

备注：无。

9.135. 全选

描述：全选：

代码示例：

```
ctl.DCExecuteCommand('SelectAll',false,null)
```

备注：无。

9.136. 选中光标所在的当前行

描述：选中光标所在的当前行

代码示例：

```
ctl.DCExecuteCommand('SelectLine',false,null)
```

备注：无。

9.137. 带选择的向下移动一行

描述：带选择的向下移动一行

代码示例：

```
ctl.DCExecuteCommand('ShiftMoveDown',false,null)
```

备注：无。

9.138. 带选择的移动到行尾

描述：带选择的移动到行尾

代码示例：

```
ctl.DCExecuteCommand('ShiftMoveEnd',false,null)
```

备注：无。

9.139. 带选择的移动到行首

描述：带选择的移动到行首

代码示例：

```
ctl.DCExecuteCommand('ShiftMoveHome',false,null)
```

备注：无。

9.140. 带选择的向左移动一列

描述：带选择的向左移动一列

代码示例：

```
ctl.DCExecuteCommand('ShiftMoveLeft',false,null)
```

备注：无。

9.141. 带选择的向下翻页

描述：带选择的向下翻页

代码示例：

```
ctl.DCExecuteCommand('ShiftMovePageDown',false,null)
```

备注：无。

9.142. 带选择的向上翻页

描述：带选择的向上翻页

代码示例：

```
ctl.DCExecuteCommand('ShiftMovePageUp',false,null)
```

备注：无。

9.143. 带选择的向右移动一列

描述：带选择的向右移动一列

代码示例：

```
ctl.DCExecuteCommand('ShiftMoveRight',false,null)
```

备注：无。

9.144. 带选择的向上移动一行

描述：带选择的向上移动一行

代码示例：

```
ctl.DCExecuteCommand('ShiftMoveUp',false,null)
```

备注：无。

9.145. 显示背景单元格编号

描述：显示背景单元格编号

代码示例：

```
ctl.DCExecuteCommand('ShowBackgroundCellID',false,null)
```

备注：无。

9.146. 显示被标记为自动隐藏的元素

描述：显示被标记为自动隐藏的元素

代码示例：

```
ctl.DCExecuteCommand('ShowElementMarkAutoHide',false,null)
```

备注：无。

9.147. 显示表单按钮

描述：显示表单按钮

代码示例：

```
ctl.DCExecuteCommand('ShowFormButton',false,null)
```

备注：无。

9.148. 插入特殊字符

描述：插入特殊字符

代码示例：

```
ctl.DCExecuteCommand('Spechars',false,"啊哈哈就<br/>萨达撒大声地")
```

备注：五代该命令不能插入 html 字符。

9.149. 插入删除线

描述：插入删除线

代码示例：

```
ctl.DCExecuteCommand('Strikeout',false,null)
```

备注：无。

9.150. 插入删除线

描述：插入删除线

代码示例：

```
ctl.DCExecuteCommand('Strikethrough',false,null)---兼容四代
```

备注：无。

9.151. 插入上标

描述：插入上标

代码示例：

```
ctl.DCExecuteCommand('Subscript',false,null)
```

备注：无。

9.152. 插入下标

描述：插入下标

代码示例：

```
ctl.DCExecuteCommand('Superscript',false,null)
```

备注：无。

9.153. 删除表格列

描述：删除表格列

代码示例：

```
ctl.DCExecuteCommand('Table_DeleteColumn',false,null)
```

备注：无。

9.154. 删除表格行

描述：删除表格行

代码示例：

```
ctl.DCExecuteCommand('Table_DeleteRow',false,null)
```

备注：无。

9.155. 删除表格

描述：删除表格

代码示例：

```
ctl.DCExecuteCommand('Table_DeleteTable',false,null)
```

备注：无。

9.156. 设置表格标题行

描述：设置表格标题行

代码示例：

```
ctl.DCExecuteCommand('Table_HeaderRow',false,null)
```

备注：无。

9.157. 修改表格行的高度，使其扩展到页面底端

描述：修改表格行的高度，使其扩展到页面底端

代码示例：

```
ctl.DCExecuteCommand('Table_IncreaseRowHeightToPageBottom',false,null)
```

备注：无。

9. 158. 在当前表格列的左边插入新的表格列

描述：在当前表格列的左边插入新的表格列

代码示例：

```
ctl.DCExecuteCommand('Table_InsertColumnLeft',false,null)
```

备注：无。

9. 159. 在当前表格列的右边插入新的表格列

描述：在当前表格列的右边插入新的表格列

代码示例：

```
ctl.DCExecuteCommand('Table_InsertColumnRight',false,null)
```

备注：无。

9. 160. 在当前表格行的下边插入新的表格行

描述：在当前表格行的下边插入新的表格行

代码示例：

```
ctl.DCExecuteCommand('Table_InsertRowDown',false,null)
```

备注：无。

9. 161. 在当前表格行的上边插入新的表格行

描述：在当前表格行的上边插入新的表格行

代码示例：

```
ctl.DCExecuteCommand('Table_InsertRowUp',false,null)
```

备注：无。

9.162. 合并单元格

描述：合并单元格

代码示例：

```
ctl.DCExecuteCommand('Table_MergeCell',false,null)
```

备注：无。

9.163. 获取光标所在的表格内容

描述：获取光标所在的表格内容

代码示例：

```
ctl.DCExecuteCommand('Table_ReadDataFromTable',false,'表格对象')
```

备注：无。

9.164. 删除最后一页中的空白行

描述：删除最后一页中的空白行

代码示例：

```
ctl.DCExecuteCommand('Table_RemoveEmptyRowsInLastPage',false,null)
```

备注：调用该命令，会把在另外一页的表格行删除掉，跨页的表格行会保留。

9.165. 拆分单元格

描述：拆分单元格

代码示例：

```
ctl.DCExecuteCommand('Table_SplitCellExt',false,null)
```

备注：无。

9.166. 根据内容行数来拆分表格行

描述：根据内容行数来拆分表格行

代码示例：

```
ctl.DCExecuteCommand('Table_SplitRowsByContentLines',false,null)
```

备注：无。

9.167. 表格数据绑定

描述：表格数据绑定

代码示例：

```
var options = {  
    RowIndex: 0,//从第几行开始赋值  
    TableElement: "table1",  
    DataArray: data,  
    TableRowExtendable: false  
};  
ctl.DCExecuteCommand("Table_WriteDataToTable", false, options);  
ctl.RefreshDocument()
```

备注：无。

9.168. 单元格属性对话框

描述：单元格属性对话框

代码示例：

```
ctl.DCExecuteCommand('TableCellProperties',true,null)
```

备注：无。

9.169. 表格属性对话框

描述：表格属性对话框

代码示例：

```
ctl.DCExecuteCommand('TableProperties',true,null)
```

备注：无。

9.170. 表格行属性对话框

描述：表格行属性对话框

代码示例：

```
ctl.DCExecuteCommand('TableRowProperties',true,null)
```

备注：无。

9.171. 表格列属性对话框

描述：表格行属性对话框

代码示例：

```
ctl.DCExecuteCommand('tableColumnproperties',true,null)
```

备注：无。

9.172. 字母转小写

描述：字母转小写

代码示例：

```
ctl.DCExecuteCommand('TolowerCase',false,null)
```

备注：无。

9.173. 字母转大写

描述：字母转大写

代码示例：

```
ctl.DCExecuteCommand('ToupperCase',false,null)
```

备注：无。

9.174. 设置下划线

描述：设置下划线

代码示例：

```
ctl.DCExecuteCommand('UnderLine',false,null)
```

备注：无。

9.175. 重做

描述：重做

代码示例：

```
ctl.DCExecuteCommand('Undo',false,null)
```

备注：无。

9.176. 设置图片在文字中四周环绕

描述：设置图片在文字中四周环绕

代码示例：

```
ctl.DCExecuteCommand('TextSurroundings',false,true)//设置图片在文字中四周环绕
```

备注：无。

9.177. 取消图片在文字中四周环绕

描述：取消图片在文字中四周环绕

代码示例：

```
ctl.DCExecuteCommand('EmbedInText',false,true)//取消图片在文字中四周环绕
```

备注：无。

9.178. 页眉下边框线显示/隐藏

描述：第三个参数为 true 时显示页眉下边框线 false 反之隐藏。只能是 bool 类型

代码示例：

```
ctl.DCExecuteCommand('HeaderBottomLineVisible',false,true)
```

备注：无。

9.179. 最后一页空白区域新增表格行填充至页底

描述：第三个参数为 `null` 时对文档最后一页空白区域上面光标所在的表格进行新增行填充至页底，或者第三个参数传表格编号新增行填充至页底。

代码示例：

```
ctl.DCExecuteCommand('Table_InsertRowsToPageBottom',false,null)
```

```
ctl.DCExecuteCommand('Table_InsertRowsToPageBottom',false,'table1')
```

备注：无。

9.180. 修改当前光标所在输入域里面的文本内容颜色

描述：修改当前光标所在输入域的内容文本颜色或者修改当前选中内容的文本颜色。

代码示例：

```
ctl.DCExecuteCommand('colorforfieldtextcolor',false,null)
```

备注：无。

9.181. 设置选中内容打印文本颜色

描述：内容编辑模式下显示黑色，打印时为红色。

代码示例：

```
ctl.DCExecuteCommand('PrintColor',false,null)
```

备注：无。

9.182. 设置选中内容字号大小

描述：修改选中内容的字号大小，其中包含当选中内容中存在输入域设置了不允许用户编辑也能修改字号大小（`FontSize` 不能修改输入域设置不允许修改的内容字号大

小)。

代码示例：

```
ctl.DCExecuteCommand('FontSizePro',false,"16");
```

备注：单位为 pt, 16px 底层也会解析成功 16pt。

9.183. 插入承载控件

描述：支持用户自定义 canvas 样式，通过插入承载控件插入到病历文档中，比如生物遗传图谱。

代码示例：

```
ctl.DCExecuteCommand('InsertControlHost', true, options);
```

备注：http://117.89.85.111:8086/pingcode/DUWRITER5_0-2866.html。

10. 附录

10.1. 附录：数值表达式

DCWriter 支持文档元素数值表达式，支持的文档元素类型有 XTextTabelCellElement、XTextInputFieldElement、XTextLabelElement、XTextSectionElement。元素的 ValueExpression 属性值就指定了数值表达式文本。

要运行文档元素数值表达式，首先要设置文档选项“DocumentOptions.BehaviorOptions.EnableExpression”的值为 true。

加载文档的时候会对文档中所有的文档元素数值表达式执行一遍，当用户编辑单元格内容时，系统会实时的检索文档中的表达式中引用文档元素的关系，若发现有其他元素表达式引用了当前编辑的文档元素，则会重新计算其他文档元素的数值表达式结果。

在表达式文本中，使用 “[元素 ID 号]” 来引用文档中的其他文档元素的值。比如有两个输入域，ID 分别为 “field1”、“field2”，则表达式可以写成 “[field1]+[field2]” 或者 “[field1]*[field2]+999” 等等。

特别的对于单元格的数值表达式，可以设置为类似 Excel 中的单元格表达式，而且可以使用类似“A3”、“B7：D9”这样的形式来引用其他单元格的文本，也可以使用“Table1!A3:B5”的形式来引用其他表格中的单元格。表达式中引用单元格编号时必须在编号两边加上方括号。比如 “[A1]”， “[A3:B4]”。

DCWriter 编辑器对单元格表达式提供了很好的设计支持，当用户插入和删除了表格行或列时，设计环境会自动更新表达式中的单元格的引用，降低设计工作量。比如某个单元格使用了表达式“PRODUCT([A16:B16])”，则当用户在 A16 单元格上面插入新的表格行时，设计环境会自动将表达式修正为“PRODUCT([A17:B17])”；若用户删除了 A16 单元格上面的某个表格行时；设计环境会自动的将表达式修正为“PRODUCT([A15:B15])”；若用户在 A16 单元格的左边插入新的表格列时，设计环境会自动将表达式修正为“PRODUCT([B16:C16])”；若用户删除了 A16 单元格所在的表格列，则设计环境会自动将表达式修正为“PRODUCT([A16:A16])”。

DCWriter 还提供 ShowBackgroundCellID 来显示文档中单元格背景编号，这样也能帮助用户快速定位单元格，辅助写出正确的表达式。

数值表达式支持加减乘除数值运算，还支持“%”进行求模计算，支持圆括号改变运算顺序，不支持逻辑运行和字符串拼凑。单元格中文本在参与表达式运算时都会试图转换为数值，若不能转换则标记为“NaN”。标记为“NaN”的数据在加减乘除时视作 0，但在某些函数中其处理方式可能不同。表达式支持以下的函数，在表达式中的函数名全部采用大写形式。

函数名	参数	说明
ABS	一个数值	返回绝对值，若参数为 NAN 则返回 NAN。 例如 ABS([feidl1])。
ACOS	一个数值	返回指定弧度的反余弦值，若参数为 NAN 则返回 NAN。 例如 ACOS([field1])。
ASIN	一个数值	返回指定弧度的反正弦值，若参数为 NAN 则返回 NAN。
ATAN	一个数值	返回指定弧度的反正切值，若参数为 NAN 则返回 NAN。
ATAN2	X 值和 Y 值	返回指定弧度的正切值，第一个参数为 X 值，第二个参数为 Y 值。若某个参数为 NAN 则返回 NAN。
AVERAGE	数值个数不定	返回算术平均值，若某个参数为 NAN 则不参与计算。
CEILING	一个数值	获得最近的整数，若参数为 NAN 则返回 NAN。
COS	一个数值	返回指定弧度的 COS 值，若参数为 NAN 则返回 NAN。
COUNT	参数个数不定	返回参数的个数，若某个参数为 NAN 则返不参与计算。
EXP	一个数值	返回 e 的 n 次方，若参数为 NAN 则返回 NAN。
FLOOR	一个数值	返回向下舍入取整数，若参数为 NAN 则返回 NAN。
INT	一个数值	将数值向小取整为最接近的整数，若参数为 NAN 则返回 NAN。

LOG	Number , base	根据给定底数返回数字的对数，第一个参数为数值，第二个参数为底数。若某个参数为 NAN 则返回 NAN。
LOG10	一个数值	返回以 10 为底的对数，若参数为 NAN 则返回 NAN。
MAX	参数个数不定	返回最大值，若某个参数为 NAN 则不参与计算。
MIN	参数个数不定	返回最小值，若某个参数为 NAN 则不参与计算。
MOD	Number,divisor	返回两个数相除的余数，第一个参数为被除数，第二个参数为除数。若某个参数为 NAN 则返回 NAN。
ODD	一个数值	将正（负）数向上（下）舍入到最接近的奇数。，若参数为 NAN 则返回 NAN。
PI	无参数	返回常数 3.14159265358979
POW	Number, power	返回某数的乘幂，第一个参数为基数，第二个参数为幂数。若某个参数为 NAN 则返回 NAN。
PRODUCT	参数个数不定	计算所有参数的乘积，若某个参数为 NAN 则不参与计算。 例如 PRODUCT([A1:C3])。
RADIANS	一个数值	将角度转换为弧度，若参数为 NAN 则返回 NAN。
RAND	无参数	返回一个介于 0 到 1 之间的随机数。
ROUND	一个数值	进行四舍五入计算，若参数为 NAN 则返回 NAN。
ROUNDDOWN	一个数值	向下舍入数字，若参数为 NAN 则返回 NAN。
ROUNDUP	一个数值	向上舍入数字，若参数为 NAN 则返回 NAN。
SIGN	一个数值	为正数返回 1，为零返回 0，为负数返回-1，若参数为 NAN 则返回 NAN。
SIN	一个数值	返回指定角度的正弦值，若参数为 NAN 则返回 NAN。
SQRT	一个数值	返回数值的平方根，若参数为 NAN 则返回 NAN。
SUM	参数不定	计算所有数值的和，若某个参数为 NAN 则返不参与计算。 例如 SUM([A1:C3])。
TAN	一个数值	返回指定角度的正切值，若参数为 NAN 则返回 NAN。
LOOKUP	多个参数	进行数组比较，返回比较结果。 例如 LOOKUP([field1],0,'不及格',60,'及格',70,'中',80,'良',90,'优')
IF	三个参数	对一个参数值转换为布尔值，如果为 true 则返回第二个参数值，如果为 false 则返回第三个参数值。 例如 IF([field1],'高兴','不高兴')
AGE	一个参数	计算周岁年龄。参数为表示时间日期数值。 例如 AGE([field1])
AGEMONTH	一个参数	计算月龄。参数为表示时间日期的数值。 例如 AGEMONTH([field1])。
AGEWEEK	一个参数	计算周龄。参数为表示时间日期的数值。 例如 AGEWEEK([field1])。
AGEDAY	一个参数	计算日龄。参数为表示时间日期的数值。 例如 AGEDAY ([field1])。
AGEHOUR	一个参数	计算小时龄。参数为表示时间日期的数值。

		例如 AGEHOUR([field1])。
CDOUBLE	一个数值	将数据转换为一个双精度浮点数
CINT	一个数值	将数据转换为一个整数
LEN	一个数值	字符串长度
FIND	两个参数	用于在第二个文本串中定位第一个文本串，并返回第一个文本串在第二个字符串位置的值，该值从第二个文本串的第一个字符算起。从 0 开始。
IF		Boolean 转换。

一些函数其参数个数不定，则为集合运算函数，比如 AVERAGE、MAX 等等，这些函数用于处理一个数据集合，在表达式中可以引用单元格区间来作为函数参数值，比如“MAX([A1:B3])”表示计算单元格区域“A1:B3”中的最大值。非集合运算函数就不能引用单元格区域来作为参数，只能引用指定的一个单元格。比如“SIN([A1])”。

10.2. 附录：页面设置

页面设置是指页面纸张的大小及页边距设置。它包含的属性及说明如下：

属性	数据类型	名称	说明
PaperKind	字符串	纸张名称	页面纸张名称，常用的纸张名称在下面的表中有描述。
PaperHeight	数值型（单位 1 /100 英寸）	纸张高度	在 PaperKind 设置成 Custom，用户可以修改，否则不允许修改。
PaperWidth	数值型（单位 1 /100 英寸）	纸张宽度	在 PaperKind 设置成 Custom，用户可以修改，否则不允许修改。
Landscape	boolean	打印方向	false:纵向打印、true:横向打印
LeftMargin	数值型（单位 1 /100 英寸）	左边距	文档正文与页面最左边的边距
RightMargin	数值型（单位 1 /100 英寸）	右边距	文档正文与页面最右边的边距
TopMargin	数值型（单位 1 /100 英寸）	上边距	文档正文与页面最上边的边距
BottomMargin	数值型（单位 1 /100 英寸）	下边距	文档正文与页面最下边的边距
HeaderDistance	数值型（单位 1 /100 英寸）	页眉位置	页眉处与页面最上边的边距
FooterDistance	数值型（单位 1 /100 英寸）	页脚位置	页脚处与页面最下边的边距
ShowGutterLine	boolean	显示装订线	false:不显示、true:显示
GutterPosition	数值型（单位 厘米）	装订线位置	装订线显示的位置与页面的距离

SwapGutter	boolean	双面打印切 换装订线	false:不切换、true:切换
GutterStyle	字符串	装订线位置	Top:上方、Left:左方、Right:右方
LeftMarginInCM	数值型（单位 厘米）	左边距	文档正文与页面最左边的边距
RightMarginInCM	数值型（单位 厘米）	右边距	文档正文与页面最右边的边距
TopMarginInCM	数值型（单位 厘米）	上边距	文档正文与页面最上边的边距
BottomMarginInCM	数值型（单位 厘米）	下边距	文档正文与页面最下边的边距
PaperHeightInCM	数值型（单位 厘米）	纸张高度	在 PaperKind 设置成 Custom，用户 可以修改，否则不允许修改。
PaperWidthInCM	数值型（单位 厘米）	纸张宽度	在 PaperKind 设置成 Custom，用户 可以修改，否则不允许修改。
HeaderFooterDiffe rentFirstPage	boolean	首页页眉页 脚不同	false:不启用、true:启用
PageIndexsForHide HeaderFooter	字符串	隐藏页码	"1, 3"值表示单独隐藏文档中的第 一页和第三页页码值
PageIndexsForPrin tBackgroundImage	字符串	打印背景图 片的页码	"1, 3"值表示单独隐藏文档中的第 一页和第三页页码值
PageIndexsForShow BackgroundImage	字符串	显示背景图 片的页码	"1, 3"值表示单独隐藏文档中的第 一页和第三页页码值

注：PageIndexsForPrintBackgroundImage、PageIndexsForShowBackgroundImage 这两个属性设置为：各个页码之间用英文逗号分开，从 1 开始计算。例如“1, 3, 5-12, -9”。中间有横杆表示页码区间，例如“5-12”；纯负数表示不包含指定页，例如“-9”表示只读不包含第 9 页。如果不设置则表示匹配任意页

常用的纸张尺寸设置如下：

纸张	尺寸	纸张	尺寸
A2 纸	420 毫米 × 594 毫米	Note 纸	216 毫米 × 279 毫米
A3 纸	297 毫米 × 420 毫米	PRC 16K 纸	146 × 215 毫米
A3 extra 纸	322 毫米 × 445 毫米	PRC 32K 纸	97 × 151 毫米
A3 extra trans verse 纸	322 毫米 × 445 毫米	PRC 32K(Big) 纸	97 × 151 毫米
A3 rotated 纸	420 毫米 × 297 毫米	PRC #1 envelope	102 × 165 毫米

纸张	尺寸	纸张	尺寸
A3 transverse 纸	297 毫米 × 420 毫米	PRC #2 envelope	102 × 176 毫米
A4 纸	210 毫米 × 297 毫米	PRC #3 envelope	125 × 176 毫米
A4 extra 纸	236 毫米 × 322 毫米	PRC #4 envelope	110 × 208 毫米
A4 plus 纸	210 毫米 × 330 毫米	PRC #5 envelope	110 × 220 毫米
A4 rotated 纸	297 毫米 × 210 毫米	PRC #6 envelope	120 × 230 毫米
A4 small 纸	210 毫米 × 297 毫米	PRC #7 envelope	160 × 230 毫米
A4 transverse 纸	210 毫米 × 297 毫米	PRC #8 envelope	120 × 309 毫米
A5 纸	148 毫米 × 210 毫米	PRC #9 envelope	229 × 324 毫米
A5 extra 纸	174 毫米 × 235 毫米	PRC #10 envelope	324 × 458 毫米
A5 rotated 纸	210 毫米 × 148 毫米	PRC 16K rotated 纸	146 × 215 毫米
A5 transverse 纸	148 毫米 × 210 毫米	PRC 32K rotated 纸	97 × 151 毫米
A6 纸	105 毫米 × 148 毫米	PRC #1 rotated envelope	165 × 102 毫米
A6 rotated 纸	148 毫米 × 105 毫米	PRC #2 rotated envelope	176 × 102 毫米
B4 纸	250 毫米 × 353 毫米	PRC #3 rotated envelope	176 × 125 毫米
B4 信封	250 毫米 × 353 毫米	PRC #4 rotated envelope	208 × 110 毫米
B5 纸	176 毫米 × 250 毫米	PRC #5 rotated envelope	220 × 110 毫米
B5 信封	176 毫米 × 250 毫米	PRC #6 rotated envelope	230 × 120 毫米
B6 信封	176 毫米 × 125 毫米	PRC #7 rotated envelope	230 × 160 毫米
C 纸	432 毫米 × 559 毫米	PRC #8 rotated envelope	309 × 120 毫米

纸张	尺寸	纸张	尺寸
Custom		自定义纸张	
C3 信封	324 毫米 × 458 毫米	PRC #9 rotated envelope	229 × 324 毫米
C4 信封	229 毫米 × 324 毫米	PRC #10 rotated envelope	458 × 324 毫米
C5 信封	162 毫米 × 229 毫米	Quarto 纸	215 毫米 × 275 毫米
C6 信封	114 毫米 × 162 毫米	Standard 纸	305 毫米 × 279 毫米
C65 信封	114 毫米 × 229 毫米	Statement 纸	140 毫米 × 216 毫米
D 纸	559 毫米 × 864 毫米	Standard 纸	254 毫米 × 356 毫米
DL 信封	110 毫米 × 220 毫米	Standard 纸	279 毫米 × 432 毫米
E 纸	864 毫米 × 1118 毫米	Standard 纸	229 毫米 × 279 毫米
Executive 纸	184 毫米 × 267 毫米	Standard 纸	254 毫米 × 279 毫米
Folio 纸	216 毫米 × 330 毫米	Standard 纸	381 毫米 × 279 毫米
German standard fanfold	216 毫米 × 305 毫米	SuperA/SuperA/A4 纸	227 毫米 × 356 毫米
German legal fanfold	216 毫米 × 330 毫米	SuperB/SuperB/A3 纸	305 毫米 × 487 毫米
ISO B4	250 毫米 × 353 毫米	Tabloid 纸	279 毫米 × 432 毫米
ISO B5 extra 纸	201 毫米 × 276 毫米	Tabloid extra 纸	297 毫米 × 457 毫米
Italy envelope	110 毫米 × 230 毫米	US standard fanfold	378 毫米 × 279 毫米
Japanese double	200 毫米 × 148 毫米	#9 envelope	3.875 英寸 × 8.8

纸张	尺寸	纸张	尺寸
e postcard			75 英寸
Japanese postcard	100 毫米 × 148 毫米	#10 envelope	105 × 241 毫米
Japanese rotated double postcard	148 毫米 × 200 毫米	#11 envelope	4.5 英寸 × 10.375 英寸
Japanese rotated postcard	148 毫米 × 100 毫米	#12 envelope	4.75 英寸 × 11 英寸
JIS B4 rotated 纸	364 毫米 × 257 毫米	#14 envelope	5 英寸 × 11.5 英寸
JIS B5 rotated 纸	257 毫米 × 182 毫米	6 3/4 envelope	92 毫米 × 165 毫米
JIS B5 transverse 纸	182 毫米 × 257 毫米	邀请函信封	220 毫米 × 220 毫米
JIS B6 纸	128 毫米 × 182 毫米	Letter extra transverse 纸	236 毫米 × 305 毫米
JIS B6 rotated 纸	182 毫米 × 128 毫米	Letter plus 纸	216 毫米 × 322 毫米
Ledger 纸	432 毫米 × 279 毫米	Letter rotated 纸	279 毫米 × 216 毫米
Legal 纸	216 毫米 × 356 毫米	Letter small 纸	216 毫米 × 279 毫米
Legal extra 纸	236 毫米 × 381 毫米	Letter transverse 纸	210 毫米 × 279 毫米
Letter 纸	216 毫米 × 279 毫米	Monarch envelope	98 毫米 × 191 毫米
Letter extra 纸	236 毫米 × 304 毫米		

10.3. 附录：输入域属性

输入域的数据类型是 XTextInputFieldElement，红色的标记表示常用。它的属性如下：

属性	数据类型	说明

AbsLeft		对象在文档中的绝对坐标位置（左侧）
AbsTop		对象在文档中的绝对坐标位置（上侧）
AcceptChildElementTypes		允许接收的内容样式
AcceptTab		能否接受制表符，默认 false
Alignment		内容对齐方式，包含 Near 左对齐、Center 居中、Far 右对齐。
Attributes		用户自定义属性列表：{键 1: '值 1', 键 2: '值 2'}
AutoFixTextMode		文本自动修复模式
AutoHideMode		自动隐藏模式 枚举值： None：不做设置 HideWhenEmpty：内容为空时自动隐藏 HideWhenChildFieldEmpty：子输入域内容 为空时自动隐藏
BackgroundText		背景文字
BackgroundTextColor		背景文字颜色
BorderElementColor		输入域边框颜色
BorderTextPosition		边框文本对齐方式 枚举值： Top：靠上对齐 Middle：垂直居中 Bottom：靠下对齐 LeftTopRightBottom：从左上到右下 LeftBottomRightTop：从左下到右上
BorderVisible		输入域边框可见性 枚举值： Default：默认 Hidden：隐藏 Visible：可见 AlwaysVisible：一直可见
ContentLock		内容锁
ContentReadonly	字符串	内容是否可读(字符串值) true：只读 false：不只读 Inherit：继承父节点
ContentReadonlyExpression		元素内容只读性表达式
CopySource		内容复制来源
DefaultEventExpression		默认使用的事件表达式字符串
Deleteable		对象是否能被删除
DisplayFormat		显示的格式化对象,支持自定义显示的格式

DynamicListItems		动态加载列表项目
EditorActiveMode		编辑器激活模式，默认双击激活 枚举值： None：无效状态（下拉类型无法激活） Default：采用默认样式(DocumentOptions.BehaviorOptions.DefaultEditorActiveMode 设置的数据来激活) MouseDbClick：双击激活 MouseDClick：单击激活 Program：编辑器激活 F2：按下 F2 激活 GotFocus：对象获得输入焦点就激活 MouseRightClick：鼠标左击激活 Enter：Enter 键激活
Elements		该元素下的所有子元素列表
EnableHighlight		是否允许高亮度显示状态 枚举值： Default：默认状态（用 DocumentOptions.ViewOptions.DefaultInputFieldHighlight）控制 Enable：可用 Disabled：禁用高亮度状态
EnablePermission		是否启用权限控制
EnableValueValidate		是否启用校验。设置了元素校验，自动设置成 true
EndBorderText		结束边框文本
EndingLineBreak		输入域结尾处强制换行。默认值 false
EventTemplateName		文档元素事件模板名称
GetValueOrderByTime		多选时，是否按照勾选的时间排序下拉选项
Height		高度
HiddenPrintWhenEmpty		内容为空时不打印
ID		元素编号
InnerEditStyle		输入域类型，包含多种： Text：纯文本输入域 DropDownList：下列列表 Date：日期类型 DateTime：日期时间类型 DateTimeWithoutSecond：不含秒的日期时间类型 Time：时间类型 Numeric：数值类型
InnerID		元素内部 ID
InnerListSourceName		动态列表

InnerMultiSelect		是否多选
InnerText		元素内容的文本内容
InnerValue		内置的数值
IsLogicDeleted		是否已经被逻辑删除
JavaScriptForClick		单击时执行的 javascript 脚本
JavaScriptForDoubleClick		双击时执行的 javascript 脚本
LabelText		标签文本
Left		左端位置
LeftInOwnerPage		文档元素在文档页中的距离页面纸张左上角的水平距离
LimitedInputChars		输入域可输入的数据类型 如: '1, 2, 3, 4, 5, 6, 7, 8, 9, 0', 只允许输入数字 1, 2, 3, 4, 5, 6, 7, 8, 9, 0
ListItems		下拉框列表(插入静态下拉输入域时设置的属性) "ListItems": [{Text: "男", Value: "0"}, {Text: "女", Value: "1"}]
ListValueFormatString		列表数值格式化字符串
ListValueSeparatorChar		多选下拉列表值之间的分隔字符
MaxInputLength		用户最大可输入的字符数
Modified		元素的状态(内容是否被修改过)
MoveFocusHotKey		移动焦点使用的快捷键 枚举值 None: 无快捷键 Default: 默认方式(由 DocumentOptions.BehaviorOptions.MoveFocusHotKey) 值决定 Tab: tab 键切换 Enter: enter 键切换
Name		元素名称
NativeHandle		该元素在文档中的唯一编号(即使文档纯在相同元素 ID 该值也是唯一)
NextElement		获得元素的同一层次的下一个元素
OwnerCell		该元素所在的单元格对象
OwnerLastPageIndex		获得一个从 0 开始计算的元素下边缘所能到达的页码
OwnerPageIndex		获得一个从 0 开始计算的当前元素所在的页码
OwnerSection		获得该元素所在的文档节对象
OwnerTable		对象所属表格对象

Parent		对象所属父节点对象
PreviousElement		获得元素的同一层次的上一个元素
PrintVisibility		打印时是否可见 Visible: 显示 Hidden: 隐藏, 但占据排版位置 None: 隐藏, 而且不占据排版位置
PrintVisibilityExpression		打印可见性表达式
RepulsionForGroup		多选下拉列表项目分组互斥
SpecifyWidth		输入域元素的宽度
StartBorderText		起始边框
Style		元素样式列表 (参考附录元素样式列表)
TabIndex		焦点切换序号
TabStop		获取或设置一个值, 该值指示用户能否使用 Tab 键将焦点放到该元素中上
Text		文本内容
TextColor		文本颜色
ToolTip		光标悬浮时的提示文本
Top		顶端位置
TopInOwnerPage		文档元素在文档页中的距离页面纸张左上角的垂直距离
TypeName		元素类型
UnitText		单位 (后缀) 文本
UserEditable	bool	用户是否允许编辑 true: 允许 false: 不允许
UserFlags		用户标记
ValidateStyle	Json 值	内容校验格式, (详情看附录 校验列表属性)
ValueBinding	Json 值	数据源绑定赋值 数据源类下的属性: AutoUpdate: 自动更新内容 - 布尔型值 BindingPath: 数据源路径-绑定输入域 Value 值-字符串值 BindingPathForText: 绑定路径 (Text) - 字符串值 DataSource: 数据源名称-字符串 Enabled: 该数据源是否可用-布尔型 ProcessState: 操作状态 (枚举值) +Always: 总是执行 +Once: 只执行一次 +Never: 从不执行 ReadOnly: 只读标记

ValueExpression	字符串值	数值表达式
ViewEncryptType	字符串值	内容脱敏 枚举值 None: 不加密 Partial: 部分加密, 前面和后面一个字符不加密 Both : 全部加密, 所有内容加密
Visible		元素是否可见
VisibleExpression		可见性表达式
Width		宽度

输入域数据校验属性:

属性名称	说明
BinaryLength	进行的是二进制长度判断
CheckDecimalDigits	是否需要判断小数位数
CheckMaxValue	检查数值或者日期值的最大值
CheckMinValue	检查数值或者日期值的最小值
CustomMessage	自定义提示信息
DateTimeMaxValue	最大时间日期值
DateTimeMinValue	最小时间日期值
ExcludeKeywords	排除关键字
IncludeKeywords	允许包含的关键字, 各个关键字之间用英文逗号分开。
Level	校验等级
MaxDecimalDigits	小数最大位数
MaxLength	最大文本长度
MaxValue	数值最大值
Message	验证结果
MinLength	最小文本长度
MinValue	数值最小值
Range	数据取值范围
RegExpression	正则表达式文字
Required	必填数据
RequiredInvalidateFlag	由于必填项而导致的数据校验无效的标记
ValueName	数据的名称
ValueType	数值类型

10.4. 附录：单选框、复选框属性

单选框、复选框属性

属性	数据类型	说明
AbsLeft		对象在文档中的绝对坐标位置（左侧）
AbsTop		对象在文档中的绝对坐标位置（上侧）
Attributes		用户自定义属性列表：{键 1: '值 1', 键 2: '值 2'}
Caption		复选框后面跟着的文本
CaptionFlowLayout		流式排版
CheckAlignLeft		勾选框靠左侧对齐
Checked		是否勾选
ContentReadonly		元素只读状态
Deleteable		是否允许删除
EnableHighlight		是否允许高亮度显示 枚举值： Default: 默认状态（用 DocumentOptions.ViewOptions.DefaultInputFieldHighlight）控制 Enable: 可用 Disabled: 禁用高亮度状态
Enabled		是否可用
EventTemplateName		文档元素事件模板名称
Height		高度
ID		元素 ID
InnerID		元素内部 ID
InnerText		元素内容的文本内容
IsLogicDeleted		是否已经被逻辑删除
IsRadio		判断是单选框还是复选框
JavaScriptForClick		单击时执行的 javascript 脚本
JavaScriptForDoubleClick		双击时执行的 javascript 脚本
Left		左端位置
LeftInOwnerPage		文档元素在文档页中的距离页面纸张左上角的水平距离
Modified		元素是否修改
Multiline		文本多行
Name		元素名称
NativeHandle		该元素在文档中的唯一编号（即使文档纯在相同元素 ID 该值也是唯一）
NextElement		获得元素的同一层次的下一个元素
OffsetX		内部属性，暂不开放

OffsetY		内部属性，暂不开放
OwnerCell		该元素所在的单元格对象
OwnerLastPageIndex		获得一个从 0 开始计算的元素下边缘所能到达的页码
OwnerPageIndex		获得一个从 0 开始计算的当前元素所在的页码
OwnerSection		获得该元素所在的文档节对象
OwnerTable		对象所属表格对象
Parent		对象所属父节点对象
PreviousElement		获得元素的同一层次的上一个元素
PrintVisibility		打印时是否可见 Visible: 显示 Hidden: 隐藏，但占据排版位置 None: 隐藏，而且不占据排版位置
Requried		是否必选
StringTag		附加数据
Style		元素样式列表
Text		元素内容
ToolTip		元素提示信息
Top		顶端位置
TopInOwnerPage		文档元素在文档页中的距离页面纸张左上角的垂直距离
TypeName		元素类型
UserFlags		用户标记
Value		内容数值
ValueBinding		数据源绑定赋值 数据源类下的属性： AutoUpdate: 自动更新内容 -布尔型值 BindingPath: 数据源路径-绑定输入域 Value 值-字符串值 BindingPathForText: 绑定路径 (Text) -字符串值 DataSource: 数据源名称-字符串 Enabled: 该数据源是否可用-布尔型 ProcessState: 操作状态 (枚举值) +Always: 总是执行 +Once: 只执行一次 +Never: 从不执行 Readonly: 只读标记
ValueExpression		数值表达式
Visible		元素是否可见
VisibleExpression		可见性表达式

VisualStyle		显示样式。 枚举字符串值： Default：默认值 RadioBox：单选框样式 CheckBox：复选框样式
Width		宽度
ZOrderStyle		内部属性，暂不开放
isnewparagraph		单选框与单选框直接进行换行显示

10.5. 附录：表格属性

元素属性	数据类型	说明
AbsLeft		对象在文档中的绝对坐标位置（左侧）
AbsTop		对象在文档中的绝对坐标位置（上侧）
AcceptChildElementType		能接收的子元素类型
AcceptTab		能否接受制表符，默认 false。
AllowUserDeleteRow		用户可删除表格行 布尔值
AllowUserInsertRow		用户可新增表格行 布尔值
AllowUserToResizeColumns		用户可调整列宽 布尔值
AllowUserToResizeRows		用户可调整行高 布尔值
Attributes		用户自定义属性列表
AutoFixTextMode		文本自动修复模式
Cells		当前表格单元格对象集合
ColumnsWidth		列宽
CompressOwnerLineSpacing		压缩所在文档行的行间距
ContentLock		内容锁
ContentReadonly		元素只读状态
ContentReadonlyExpression		元素内容只读性表达式。
CopySource		内容复制来源
Deleteable		用户可删除表格
Elements		子对象列表
EnablePermission		启用授权控制 布尔值
EventTemplateName		文档元素事件模板名称
Height		高度
HiddenPrintWhenEmpty		内容为空时不打印

ID		元素编号
InnerID		元素内部 ID
InnerText		元素内容的文本内容
JavaScriptForClick		单击时执行的 javascript 脚本
JavaScriptForDoubleClick		双击时执行的 javascript 脚本
Left		左端位置
LeftInOwnerPage		文档元素在文档页中的距离页面纸张左上角的水平距离
LimitedInputChars		本容器元素能接受的字符清单, 比如本属性设置为“0123456789”, 则在用户界面中本容器元素只能输入规定的数字字符
MaxInputLength		最大可输入的字符的长度, 属性值小于等于 0 则无限制。
Modified		元素是否修改
OwnerLastPageIndex		获得一个从 0 开始计算的元素下边缘所能到达的页码
OwnerPageIndex		获得一个从 0 开始计算的当前元素所在的页码
PrintBothBorderWhenJumpPrint		续打时打印所有单元格边框
PrintVisibility		打印是否可见
PrintVisibilityExpression		打印时是否可见性表达式
Rows		当前表格表格行对象集合
RowCount		当前表格表格行行数
RowsHeight		行高
Style		元素样式列表
Text		表格内容
ToolTip		元素提示信息
Top		顶端位置
TopInOwnerPage		文档元素在文档页中的距离页面纸张左上角的垂直距离
TypeName		元素类型
UserFlags		用户标记
ValueExpression		数值表达式
Visible		元素是否可见
Width		宽度

10.6. 附录：表格行属性

元素属性	数据类型	说明
AbsLeft		对象在文档中的绝对坐标位置（左侧）
AbsTop		对象在文档中的绝对坐标位置（上侧）
AcceptChildElementTypes		允许接收的内容样式
AcceptTab		输入域接收 Tab 符号
AllowInsertRowDownUseHotKey		使用快捷键向下插入表格行时的行为 Disable: 禁止使用快捷键在下面插入表格行 EnableOnlyForLastRow: 只有是表格的最后一行才启用这种行为 EnableInAllCases: 在所有状态下都启用这种行为。无论表格行是否是表格的最后一行都启用这种行为
AllowUserToResizeHeight		是否允许用户鼠标拖拽操作改变表格行高度 False: 不只读 Inherit: 只读 True: 继承父节点
Attributes		用户自定义属性列表
AutoFixTextMode		文本自动修复模式
AutoHideMode		自动隐藏模式 None: 不做设置 HideWhenEmpty: 内容为空时自动隐藏 HideWhenChildFieldEmpty: 子输入域内容为空时自动隐藏
CanSplitByPageLine		能否被分页线分割，也就是是否允许被分配到两页上
Cells		表格中单元格集合列表
CellsCount		单元格数量
CloneType		复制方式 Default: 默认方式，只复制表格行及单元格，不复制内容。对于单元格是继承上级设置。 ContentWithClearField: 复制内容，但删除输入域中的内容。 Complete: 完整的复制，包括输入域中的内容。
ContentLock		内容锁
ContentReadonly		内容是否只读

		True: 只读 False: 不只读 Inherit: 继承父节点
ContentReadonlyExpression		元素内容只读性表达式。
CopySource		内容复制来源
Deleteable		是否允许删除
Elements		子元素列表
EnablePermission		启用文档内容安全和权限控制 True: 启用 False: 不启用 Inherit: 继承父节点
EventTemplateName		文档元素事件模板名称
ExpendForDataBinding		由于数据源绑定而扩展的表格行
HeaderStyle		标题行样式
Height		高度
HiddenPrintWhenEmpty		当为空时打印是否隐藏
ID		元素编号
InnerID		元素内部 ID
InnerText		元素内容的文本内容
JavaScriptForClick		单击时执行的 javascript 脚本
JavaScriptForDoubleClick		双击时执行的 javascript 脚本
Left		左端位置
LeftInOwnerPage		文档元素在文档页中的距离页面纸张左上角的水平距离
LimitedInputChars		本容器元素能接受的字符清单, 比如本属性设置为“0123456789”, 则在用户界面中本容器元素只能输入规定的数字字符
MaxInputLength		最大可输入的字符的长度, 属性值小于等于 0 则无限制。
Modified		元素是否修改
NewPage		是否强制分页
OwnerLastPageIndex		获得一个从 0 开始计算的元素下边缘所能到达的页码
OwnerPageIndex		获得一个从 0 开始计算的当前元素所在的页码
PrintCellBackground		打印单元格背景
PrintCellBorder		打印单元格边框线
PrintVisibility		打印时是否可见 Visible: 显示 Hidden: 隐藏, 但占据排版位置 None: 隐藏, 而且不占据排版位置

PrintVisibilityExpression		打印时可见表达式
RowIndex		要操作的表格行序号
SpecifyHeight		指定的高度
Style		元素样式列表
Text		元素内容
ToolTip		元素提示信息
Top		顶端位置
TopInOwnerPage		文档元素在文档页中的距离页面纸张左上角的垂直距离
TypeName		元素类型
UserFlags		用户标记
ValueBinding		数据源绑定信息。同输入域属性下的数据源，但是表格数据源绑定，表格行只需要设置数据源名称，行对应的单元格设置绑定路径
ValueExpression		数值表达式
Visible		元素是否可见
VisibleExpression		元素可见性表达式
Width		宽度

10.7. 附录：单元格属性

元素属性	数据类型	说明
AbsLeft		对象在文档中的绝对坐标位置（左侧）
AbsTop		对象在文档中的绝对坐标位置（上侧）
AcceptChildElementTypes		允许接收的内容样式
AcceptTab		输入域接收 Tab 符号
Attributes		用户自定义属性列表
AutoFixFontSizeMode		自动修改字体大小以适应内容
AutoFixTextMode		文本自动修复模式
AutoHideMode		自动隐藏模式 None：不做设置 HideWhenEmpty：内容为空时自动隐藏 HideWhenChildFieldEmpty：子输入域内容为空时自动隐藏
BorderPrintedWhenJumpPrint		在续打的时候已经打印了边框，和 XTextTableElement.PrintBothBorderWhenJumpPrint 属性搭配使用。

CellID		单元格背景 ID
CloneType		复制方式 Default: 默认方式, 只复制表格行及单元格, 不复制内容。对于单元格是继承上级设置。 ContentWithClearField: 复制内容, 但删除输入域中的内容。 Complete: 完整的复制, 包括输入域中的内容。
ColIndex		当前单元格所在列索引
ColSpan		跨列数合并
ContentLock		内容锁
ContentReadonly		内容是否只读 True: 只读 False: 不只读 Inherit: 继承父节点
ContentReadonlyExpression		元素内容只读性表达式。
CopySource		内容复制来源
Deleteable		是否允许删除
Elements		子元素列表
EnablePermission		启用文档内容安全和权限控制 True: 启用 False: 不启用 Inherit: 继承父节点
EventTemplateName		文档元素事件模板名称
GridLine		网格线设置
Height		高度
HiddenPrintWhenEmpty		当为空时打印是否隐藏
ID		元素编号
InnerID		元素内部 ID
InnerText		元素内容的文本内容
IsOverrided		判断本单元格是否被其他单元格合并了
JavaScriptForClick		单击时执行的 javascript 脚本
JavaScriptForDoubleClick		双击时执行的 javascript 脚本
Left		左端位置
LeftInOwnerPage		文档元素在文档页中的距离页面纸张左上角的水平距离
LimitedInputChars		本容器元素能接受的字符清单, 比如本属性设置为“0123456789”, 则在用户界面中本容器元素只能输入规定的数字字符

MaxInputLength		最大可输入的字符的长度，属性值小于等于 0 则无限制。
MirrorViewForCrossPage		跨页视图镜像
Modified		元素是否修改
MoveFocusHotKey		移动焦点使用的快捷键
OwnerLastPageIndex		获得一个从 0 开始计算的元素下边缘所能到达的页码
OwnerPageIndex		获得一个从 0 开始计算的当前元素所在的页码
PrintVisibility		打印时是否可见 Visible: 显示 Hidden: 隐藏，但占据排版位置 None: 隐藏，而且不占据排版位置
PrintVisibilityExpression		打印时可见表达式
RowIndex		要操作的表格行序号
RowSpan		跨行数合并
SlantSplitLineStyle		斜分割线样式
Style		元素样式列表
Text		元素内容
ToolTip		元素提示信息
Top		顶端位置
TopInOwnerPage		文档元素在文档页中的距离页面纸张左上角的垂直距离
TypeName		元素类型
UserFlags		用户标记
ValueBinding		数据源绑定信息。同输入域属性下的数据源，但是表格数据源绑定，表格行只需要设置数据源名称，行对应的单元格设置绑定路径
ValueExpression		数值表达式
Visible		元素是否可见
VisibleExpression		元素可见性表达式
Width		宽度

10.8. 附录：标签文本属性

元素属性	数据类型	说明
AbsLeft		对象在文档中的绝对坐标位置（左侧）
AbsTop		对象在文档中的绝对坐标位置（上侧）
Alignment		内容对齐方式
AllowUserEditCurrentPageLabelText		允许用户编辑当前页标签文本
AttributeNameForContactAction		执行连接动作使用的属性名（来自于元素的自定义属性值）
Attributes		用户自定义属性列表
Bold		是否加粗
ContactAction		连接模式 枚举值： Disable：禁止文本连接 Normal：正常模式 FirstSectionInPage：只显示当前页面中第一个文档节的文本 LastSectionInPage：只显示当前页面中最后一个文档节的文本 TableRow：表格行 FirstTableRowInPage：页面中的第一个表格行 LastTableRowInPage：页面中的最后一个表格行
ContentReadonly		元素只读状态
Delete		是否允许删除
Enabled		是否可用
EventTemplateName		文档元素事件模板名称
Height		高度
ID		元素编号
InnerID		元素内部 ID
InnerText		元素内容的文本内容
JavaScriptForClick		单击时执行的 javascript 脚本
JavaScriptForDoubleClick		双击时执行的 javascript 脚本
Left		左端位置
LeftInOwnerPage		文档元素在文档页中的距离页面纸张左上角的水平距离
LinkTextForContactAction		连接字符串
Modified		元素是否修改

Name		元素名称
OffsetX		内部属性，暂不开放
OffsetY		内部属性，暂不开放
OwnerLastPageIndex		获得一个从 0 开始计算的元素下边缘所能到达的页码
OwnerPageIndex		获得一个从 0 开始计算的当前元素所在的页码
PageTexts		各个页面的文本
PrintVisibility		打印时是否可见 Visible: 显示 Hidden: 隐藏，但占据排版位置 None: 隐藏，而且不占据排版位置
StringTag		附加数据
Style		元素样式列表
Text		元素内容
TitleLevel		标题层次, 默认值为-1
ToolTip		元素提示信息
Top		顶端位置
TopInOwnerPage		文档元素在文档页中的距离页面纸张左上角的垂直距离
TypeName		元素类型
UserFlags		用户标记
ValueExpression		数值表达式
Visible		元素是否可见
Width		宽度
ZOrderStyle		内部属性，暂不开放

10.9. 附录：图片属性

元素属性	数据类型	说明
AbsLeft		对象在文档中的绝对坐标位置（左侧）
AbsTop		对象在文档中的绝对坐标位置（上侧）
Attributes		用户自定义属性列表
ContentReadonly		元素只读状态
Delete		是否允许删除
EnableEditImageAdditionShape	bool	图片是否允许编辑，默认双击图片会弹出图片编辑框，设置 false, 不触发
Enabled		是否可用
EventTemplateName		文档元素事件模板名称
Height		高度

ID		元素编号
InnerID		元素内部 ID
InnerText		元素内容的文本内容
JavaScriptForClick		单击时执行的 javascript 脚本
JavaScriptForDoubleClick		双击时执行的 javascript 脚本
Left		左端位置
LeftInOwnerPage		文档元素在文档页中的距离页面纸张左上角的水平距离
Modified		元素是否修改
Name		元素名称
OffsetX		内部属性，暂不开放
OffsetY		内部属性，暂不开放
OwnerLastPageIndex		获得一个从0开始计算的元素下边缘所能到达的页码
OwnerPageIndex		获得一个从0开始计算的当前元素所在的页码
PrintVisibility		打印时是否可见 Visible: 显示 Hidden: 隐藏，但占据排版位置 None: 隐藏，而且不占据排版位置
SaveContentInFile		保存图片数据到文件中 true: 保存图片数据 false: 不保存图片数据
src		图片来源，或者为 base64 图片数据。 注意： src 小写时，当图片来源为网络地址时，保存文档时，图片元素节点为图片网络地址存储。 Src, S 大写时，即使图片来源为网络地址时，保存文档时，图片元素节点会变成 base64 图片数据存储，会受 SaveContentInFile 属性影响。
StringTag		附加数据
Style		元素样式列表
Text		元素内容
ToolTip		元素提示信息
Top		顶端位置
TopInOwnerPage		文档元素在文档页中的距离页面纸张左上角的垂直距离
TypeName		元素类型
UserFlags		用户标记
ValueBinding		内容绑定对象

ValueExpression		数值表达式
Visible		元素是否可见
Width		宽度
ZOrderStyle	枚举值	Normal: 普通 UnderText: 文本下方 InFrontOfText: 文本上方
VAlign	枚举值	垂直对齐方式 Top: 向上 Middle: 居中 Bottom: 向下
TransparentColorValue	string	透明颜色

10.10. 附录：按钮属性

元素属性	数据类型	说明
AbsLeft		对象在文档中的绝对坐标位置（左侧）
AbsTop		对象在文档中的绝对坐标位置（上侧）
Attributes		用户自定义属性列表
AutoSize		是否设置自动大小
CommandName		命令名称
ContentReadonly		元素只读状态
Delete		是否允许删除
Enabled		是否可用
EventTemplateName		文档元素事件模板名称
Height		高度
ID		元素编号
ImgBase64		按钮的 base64 数据图片
ImgBase64ForDown		鼠标点击按钮的 base64 数据图片
ImgBase64ForOver		鼠标悬浮按钮时的 base64 数据图片
InnerID		元素内部 ID
InnerText		元素内容的文本内容
JavaScriptForClick		单击时执行的 javascript 脚本
JavaScriptForDoubleClick		双击时执行的 javascript 脚本
Left		左端位置
LeftInOwnerPage		文档元素在文档页中的距离页面纸张左上角的水平距离
Modified		元素是否修改
Name		元素名称
OffsetX		内部属性，暂不开放
OffsetY		内部属性，暂不开放
OwnerLastPageIndex		获得一个从 0 开始计算的元素下边缘所

		能到达的页码
OwnerPageIndex		获得一个从 0 开始计算的当前元素所在的页码
PrintVisibility		打印时是否可见 Visible: 显示 Hidden: 隐藏, 但占据排版位置 None: 隐藏, 而且不占据排版位置
PrintAsText		打印时文字打印 (按钮样式不参与打印)
ScriptTextForClick		按钮点击时使用的脚本
StringTag		附加数据
Style		元素样式列表
Text		元素内容
ToolTip		元素提示信息
Top		顶端位置
TopInOwnerPage		文档元素在文档页中的距离页面纸张左上角的垂直距离
TypeName		元素类型
UserFlags		用户标记
ValueExpression		数值表达式
Visible		元素是否可见
Width		宽度
ZOrderStyle		内部属性, 暂不开放

10.11. 附录：条形码属性

元素属性	数据类型	说明
Attributes	json	元素自定义属性
BarcodeStyle	string	元素类型 (详细可见元素属性对话框)
ContentReadonly	string	元素内容只读 (详细可见元素属性对话框)
Deleteable	bool	元素是否允许删除
Enabled	bool	元素是否可用
Height	float	元素高度
ID	string	元素编号
InnerText	string	元素文本

IsLogicDeleted	bool	元素是否逻辑删除
Modified	bool	元素状态标记
Name	string	元素名称
NativeHandle	int	元素唯一标识
Parent	int	元素父元素对象
PrintVisibility	string	元素打印是否可见（详细可见元素属性对话框）
ShowText	string	元素文本是否显示在条形码下方
TextAlignment	string	元素文本对齐方式（详细可见元素属性对话框）
Text	string	元素文本
TypeName	string	元素类型 (XTextNewBarcodeElement)
ValueBinding	json	元素数据源绑定 数据源类下的属性： AutoUpdate : 自动更新内容-布尔型值 BindingPath : 数据源路径-绑定输入域 Value 值-字符串值 BindingPathForText : 绑定路径 (Text) -字符串值 DataSource : 数据源名称-字符串 Enabled : 该数据源是否可用-布尔型 ProcessState : 操作状态（枚举值） +Always : 总是执行 +Once : 只执行一次 +Never : 从不执行 Readonly : 只读标记
Visible	bool	元素是否可见

Width	float	元素宽度

10. 12. 附录：二维码属性

元素属性	数据类型	说明
Attributes	json	元素自定义属性
BarcodeType	string	元素类型（详细可见元素属性对话框）
ContentReadonly	string	元素内容只读（详细可见元素属性对话框）
Deleteable	bool	元素是否允许删除
Enabled	bool	元素是否可用
Height	float	元素高度
ID	string	元素编号
InnerText	string	元素文本
IsLogicDeleted	bool	元素是否逻辑删除
Modified	bool	元素状态标记
Name	string	元素名称
NativeHandle	int	元素唯一标识
Parent	int	元素父元素对象
PrintVisibility	string	元素打印是否可见（详细可见元素属性对话框）
ShowText	string	元素文本是否显示在条形码下方
TextAlignment	string	元素文本对齐方式（详细可见元素属性对话框）
Text	string	元素文本
TypeName	string	元素类型 (XTextTDBarcodeElement)

VAlign	string	垂直对齐方式
ValueBinding	json	元素数据源绑定 数据源类下的属性： AutoUpdate : 自动更新内容 - 布尔型值 BindingPath : 数据源路径 - 绑定输入域 Value 值 - 字符串值 BindingPathForText : 绑定路径 (Text) - 字符串值 DataSource : 数据源名称 - 字符串 Enabled : 该数据源是否可用 - 布尔型 ProcessState : 操作状态 (枚举值) +Always : 总是执行 +Once : 只执行一次 +Never : 从不执行 Readonly : 只读标记
Visible	bool	元素是否可见
Width	float	元素宽度

10. 13. 附录：水平线属性

元素属性	数据类型	说明
Color	string	水平线颜色
ID	string	元素编号
LineLengthInCM	float	水平线宽度 (单位厘米)
LineSize	float	水平线粗细
LineStyle	string	水平线样式 (详细可看元素属性对话框) 默认样式为 Solid

10.14. 附录：病程（主要介绍 Options 中参数）

元素属性	是否必填	数据类型	说明
id	必填	字符串	病程的编号
tooltip	非必填	字符串	病程的提示文本
title	非必填	字符串	病程的标题
newpage	非必填	Bool	是否强制换页
readonly	非必填	Bool	是否只读
enablepermission	非必填	Bool	是否启用权限
attributes	非必填	Json	自定义属性信息
parameters	非必填	Json	数据源绑定对象
databindingxml	非必填	字符串	绑定的 XML 信息
+titlelevel	非必填	Int	标题层级
loadoriginheaderfooter	非必填	Bool	当前端设置保存病程页眉页脚的时候, 将页眉页脚字符串存在病程的自定义属性内
autofittablewidth	非必填	Float	设置子文档内的表格列宽进行缩放适应主文档宽度
cssstyle	非必填	Json	自定义的样式。 比如: "border: 1px solid yellow;background-color: gray;" --设置病程边框颜色和病程背景颜色

10.15. 附录：水印

水印是指对文档设置文字水印、图片水印来进行文档的防伪。它包含的属性及说明如下：

属性	数据类型	名称	说明
type	字符串	水印类型	Text: 文字水印、Image: 图片水印
densityforrepeat	数值型	重复平铺密度	水印的平铺密度（0-1）
repeat	boolean	是否启用平铺	false: 不启用、true: 启用
alpha	数值型	水印透明度	数值越大，水印越清晰
colorvalue	字符串	水印颜色	设置的文字水印显示颜色
angle	数值型	水印倾斜度	0-360，设置水印的倾斜度

imagedatabase64string	字符串	水印图片数据	水印图片的数据来源
text	字符串	文字水印内容	文字水印内容
font	字符串	文字水印样式	文字水印显示字体字号:“宋体, 18”

10.16. 附录：装订线

装订线是指对文档设置装订线来进行文档的整理。它包含的属性及说明如下：

属性	数据类型	名称	说明
GutterPosition	数值型(单位 1/100 英寸)	装订线距离	装订线显示在页面上的距离
GutterStyle	字符串	装订线位置	Top:上方、Left:左方、Right:右方
ShowGutterLine	boolean	显示装订线	false:不显示、true:显示
SwapGutter	boolean	双面打印切换装订线	false:不切换、true:切换

10.17. 附录：文档网格线

文档网格线是指对文档设置网格线,类似于信纸的那种样式。它包含的属性及说明如下：

属性	数据类型	名称	说明
Visible	boolean	显示网格线	false: 不显示、true: 显示
ColorValue	字符串	网格线颜色	网格线显示的颜色
GridNumInOnePage	数值型	每页行数	每一行显示的网格线行数
LineStyle	字符串	网格线样式	Solid: 实线 Dash: 由划线段组成的直线 Dot: 由点构成的直线 DashDot: 由重复的划线点图案构成的直线 DashDotDot: 由重复的划线点点图案构成的直线
Printable	boolean	打印网格线	false: 不打印、true: 打印
AlignToGridLine	boolean	文本对齐网格线	false:不对齐、true: 对齐

10.18. 附录：段落列表样式

段落列表样式就是设置段落的有、无序列表的枚举值

枚举值	实际效果
None	无样式
ListNumberStyle	默认的数字样式：1. 2. 3. 4.
ListNumberStyleArabic1	阿拉伯 1 型数字样式：1, 2, 3,
ListNumberStyleArabic2	阿拉伯 2 型数字样式：1) 2) 3)
ListNumberStyleLowercaseLetter	小写字母样式：a) b) c)
ListNumberStyleLowercaseRoman	小写罗马样式：i) ii) iii)
ListNumberStyleNumberInCircle	带圈数字样式：①, ②, ③,
ListNumberStyleSimpChinNum1	简体中文数字 1 样式：一. 二. 三
ListNumberStyleSimpChinNum2	简体中文数字 2 样式：一) 二) 三)
ListNumberStyleTradChinNum1	繁体中文数字 1 样式：壹. 貳. 叁.
ListNumberStyleTradChinNum2	繁体中文数字 2 样式：壹) 貳) 叁)
ListNumberStyleUppercaseLetter	大写字母样式：A) B) C)
ListNumberStyleUppercaseRoman	大写罗马样式：I) II) III)
ListNumberStyleZodiac1	Zodiac1 样式：甲, 乙, 丙,

ListNumberStyleZodiac2	Zodiac2 样式: 子, 丑, 寅,
BulletedListHollowStar	空心星列表: ☆
BulletedList	圆头列表: ●
BulletedListBlock	方块列表: ■
BulletedListDiamond	菱形方块列表: ◆
BulletedListCheck	钩子列表: ✓
BulletedListRightArrow	向右的箭头列表: ➤

10. 19. 附录：医学表达式

编辑器内置大量医学表达式，以下就是各种表达式类型说明。shift+f12 快捷键找到命令 InsertMedicalExpression 会弹出医学表达式集合列表有公式对应的效果图

类型 ()	说明
FourValues1	月经史公式 1
FourValues2	月经史公式 2 (根据条件能生成视野公式，插入时设置自定义属性 ISPERIMETRIC 值为 true)
ThreeValues	月经史公式 3
FourValues	月经史公式 4
FourValues4	出血指数公式
Pupil	瞳孔图
FetalHeart	胎心图

PainIndex	标尺公式
PermanentTeethBitmap	恒牙牙位图
DeciduousTeech	乳牙牙位图
AdvancedTeech	混合牙位图
DiseasedTeethTop	病变上牙图类型
DiseasedTeethBotton	病变下牙图类型
Fraction	分数公式
ThreeValues2	眼球突出度公式
StrabismusSymbol	斜视符号公式
ElectricPulpTestingTeeth	电话力测试牙位图
StationaryBridgeTeeth	固定桥牙位图
PDTeech	探诊深度
PDTeech	根分叉
LightPositioning	光定位图

10.20. 附录：函数表达式

编辑器支持大量函数表达式，以下就是各个函数表达式的相关介绍

函数名称	说明	用法
POW(NUMER,POWER)	返回某数的乘幂。	示例：输入域数值表达式：POW([field1],[field2])--代表当前输入域值为输入域 ID 为 field1 的乘幂、ID 为 field2 的输入域值为底数（ $6^2=64$ ）
PRODUCT(V1,V2,V3...)	返回所有参数的乘积。	示例：输入域数值表达式：PRODUCT([field1],[field2])--代表当前输入域值为输入域 ID 为 field1、输入域 ID 为 field2 的乘积。（ $3*4=12$ ）
ROUND(V,i)	进行四舍五入计算。并且第二个参数控制保留多少位小数	示例：输入域数值表达式：ROUND([field1],1)--代表当前输入域值为输入域 ID 为 field1 进行四舍五入。（ $11.3764=11.4$ ）（ $11.511=11.5$ ）

SUM(V1,V2...)	返回所有参数的和。	示例：输入域数值表达式：POW([field1],[field2])--代表当前输入域值为输入域 ID 为 field1、ID 为 field2 的输入域值之和（1+2=3） 单元格数值表达式：SUM([C1:C3])--代表单元格背景编号 C1-C3 值之和
SUMINNERVALUE(v1)	计算一组复选框之和	示例：输入域数值表达式：SUMINNERVALUE([name001])---代表复选框勾选后值之和 示例：输入域数值表达式：[name001]--代表单选框勾选后的值赋值给输入域，当需要计算多个单选框组之间和，可以使用+把各个单选框组 name 相加。比如：[name001]+[name002]+[name003]
FIND	判断元素是否包含某些字符控制元素显隐	示例：输入域可见性表达式：FIND('男',[field1])>=0--代表当前输入域值为多选下拉输入域 ID 为 field1 值包 含男，该输入域就显示，否则隐藏。
LOOKUP	进行数组比较，返回比较结果。	示例：输入域数值表达式：LOOKUP([field1],0,'不及格',60,'及格',70,'中',80,'良',90,'优')--代表当前输入域值为输入域 ID 为 field1 值所属某个区间的值（45=不及格）（81=良）
IF	对一个参数数值转换为布尔值，如果为 true 则返回第二个参数值，如果为 false 则返回第三个参数值。	示例：输入域可见性表达式：if([field1]=null,true,false)--代表输入域 ID 为 field1 值为空时，当前输入域显示，不为空的时候，当前输入域隐藏。 示例：输入域数值表达式 if([field1]="",0,2)--代表当前输入域值为输入域 ID 为 field1 值为空时显示 0，否则显示 2
性别-月经史级联		示例：输入域可见性表达式[field1]='男'--当输入域 id 为男的时候，该输入域显示，否则隐藏

ADDDAY(V1,d)	计算一个日期 增加几天得到 另一个日期	示例：输入域数值表达式 ADDDAY([field1],7)。表示该输入域的日期为编号为 field1 日期输入域+7 天进行显示。
AddHour(V1,h)	计算一个时间 增加几小时得到 另一个时间	示例：输入域数值表达式 AddHour([field1],7)。表示该输入域的时间为编号为 field1 时间输入域+7 小时进行显示。
ADDMINUTES(V1,m)	计算一个时间 增加几分钟得到 另一个时间	示例：输入域数值表达式 ADDMINUTES([field1],7)。表示该输入域的时间为编号为 field1 时间输入域+7 分钟行显示。
DAYDIFF(V1,V2)	得出两个日期 输入域天数差	示例：输入域数值表达式 ADDDAY([field1],[field2])。表示用编号为 field2 日期减去编号为 field1 日期得到天数差
HOURLDIFF(V1,V2)	得出两个时间 输入域小时差	示例：输入域数值表达式 HOURLDIFF([field1],[field2])。表示用编号为 field2 时间减去编号为 field1 时间得到小时差
MINUTEDIFF2(V1,V2)	得出两个时间 输入域分钟差	示例：输入域数值表达式 MINUTEDIFF2([field1],[field2])。表示用编号为 field2 时间减去编号为 field1 时间得到分钟差
MINUTEDIFF(V1,V2)	得出两个日期 时间输入域时间差	示例：输入域数值表达式 MINUTEDIFF([field1],[field2])。表示用编号为 field2 时间减去编号为 field1 时间得到带有天数、小时、分钟、秒的时间差
Count(V1,V2,V3....)	判断文书中勾选 单选框或者 复选框个数和	示例：输入域数值表达式 Count([rad1],[rad2],[rad3],[rad4])。表示这四个单选框勾选之和
打印可见性表达式：设置元素属性 -PrintVisibilityExpression	根据可见性表达式条件，满足条件不参与打印，不满足者打印	示例：容器元素可见性表达式： if([field1]=null,false,true)。表示当编号为 field1 的输入域元素为空时，该元素不参与打印，否则参与打印

10. 21. 附录：音视频属性(常用属性)

元素属性	数据类型	说明
Attributes	json	自定义属性
ContentReadonly	string	内容是否只读
Deleteable	bool	元素是否允许删除
Enabled	bool	元素是否可用
FileName	string	视频文件来源地址（新版本写法（2024-9月后））
FileSystemName	string	音频文件来源地址
Height	folat	音视频高度
ID	string	音视频编号
ImgBase64	string	音视频插入时背景图片样式（base64 图片数据）
LoopPlay	bool	循环播放内容
NativeHandle	int	元素唯一标识
PrintVisibility	string	元素打印是否可见
TypeName	string	元素类型（XTextMediaElement）
Width	float	音视频宽度

10. 22. 附录：页码属性(常用属性)

元素属性	数据类型	说明
Attributes	json	自定义属性
ContentReadonly	string	内容是否只读
Deleteable	bool	元素是否允许删除
Enabled	bool	元素是否可用
FormatString	string	格式化字符串： 第 [%PageIndex%] 页 共 [%NumOfPages%] 页 界面表示为 第 1 页 共 5 页 效果
Height	folat	元素高度
ID	string	元素编号
NativeHandle	int	元素唯一标识
PrintVisibility	string	元素打印是否可见
TypeName	string	元素类型（XTextPageInfoElement）
ValueType	string	页码类型：默认值 PageIndex 当文档中页码类型是这个枚举值时，参与 多文档合并打印时，合并后文档页码展示 效果为 1, 2, 3, 4, 5, 6, 7, 8, 9 连续展示。 当不为该类型时，多文档合并打印使用文

		本自身页码值显示
Width	float	元素宽度

10. 23. 附录：编辑器标尺样式

样式名称	数据类型	说明
RuleVisible	bool	是否显示标尺 true：显示标尺 false：隐藏标尺