1.富文本编辑器入门

什么是富文本编辑器

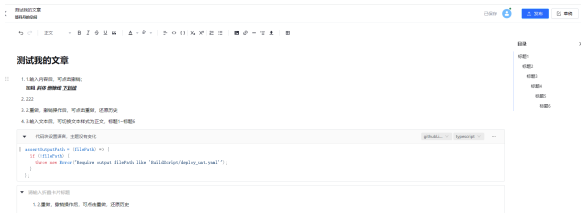
普通的文本框（如 input, textarea）只能输入纯文本以及 Emoji（只不过是特殊编码的文本），如果只是简简单单写些纯文本的内容（比如表单，简单的评论等），这是一个非常不错的选择；

但是如果需要输入图片、视频、加粗文本等内容，这种方式就行不通了；

Markdown 格式语法可以满足多种输入诉求，但是这样会提高用户的学习成本，需要学习markdown的输入规则；这时候富文本编辑器就是比较好的解决方案了

富文本编辑器是一种内嵌于浏览器，所见即所得的文本编辑器，提供类似于office word的编辑功能，方便那些不太了解html的用户使用。不需要像 GitHub 输入 Markdown 内容那样需要在输入视图和预览视图切换才能看到最终效果。主流的浏览器都已经支持这一功能。

招乎文档富文本编辑器样例：https://oa-doc.cmbchina.com/f/s/v?id=15000443



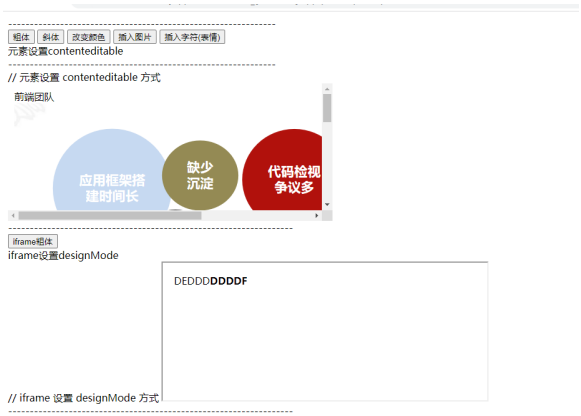
现有的编辑器实现，大多借助开源的工具包进行二次开发，对于底层的实现细节进行了封装和抽象，对上层用户提供了一些基本的配置入口；例如wangEditor提供了，工具栏、菜单配置，能够满足一些基础场景的使用，比如有话说的发帖和评论；

有话说发帖场景支持加粗、列表、图片上传、撤销、重做、对齐、链接等；基于wangEditor修改一下ui，配置一些功能菜单就可以完成；

招乎文档的业务需求是，开发一个支持多用户协同的富文本编辑器；并且技术实现上要兼容confluence的数据格式的编辑器，所以需要选用一些底层结构尽可能多可以自主可控的一些框架进行开发，比如prosemirror；要从底层自定义文档结构、开发插件，就需要对编辑器的基础知识有一些了解；

一个简单的富文本编辑器的例子

http://localhost:63342/myapp/richText.html?_ijt=ihfsi43jcrppq7akedap5m3q7e



如何在网页上实现富文本编辑

使用designMode 属性

实现这一技术的本质就是在页面中嵌人一个包含空 HTML 页面的 iframe。通过设置 designMode 属性，这个的 HTML 页面可以被编辑，而编辑对象则是该页面<body>元素的 HTML代码。designMode个可能的值:"off"(默认值)和"on"。在设置为"on"时，整个文档都会变得可以编辑，像使用字处理软件一样，通过键盘将文本内容加粗、变成斜体，等等。

可以给 iframe 指定一个非常简单的 HTML 页面作为其内容来源。例如:

```
<html>
<head>
        <title>Blank Page for Rich Text Editing</title>
    </head>
    <body>
    </body>
</html>
```

这个页面在iframe 中可以像其他页面一样被加载。要让它可以编辑，必须要将 designMode设置为"on"，但只有在页面完全加载之后才能设置这个属性。因此，在包含页面中，需要使用onload事处理程序来在恰当的时刻设置 designMode，如下面的例子所示:

```
EventUtil.addHandler(window, "load", function(){    frames["richedit"].
document.designMode = "on";
});
```

等到以上代码执行之后，你就会在页面中看到一个类似文本框的可编辑区字段。这个区字段具有与其他网页相同的默认样式;不过，通过为空白页面应用 CSS 样式，可以修改可编辑区字段的外观

使用 contenteditable 属性

另一种编辑富文本内容的方式是使用名为 contenteditable 的特殊属性，这个属性也是由IE最早实现的。可以把contenteditable 属性应用给页面中的任何元素,然后用户立即就可以编辑该元素了。这种方法之所以受到欢迎，是因为它不需要 iframe、空白页和 JavaScript，只要为元素contenteditable属性即可。

```
<div id="richedit"  contenteditable></div>
```

这样，元素中包含的任何文本内容就都可以编辑了,就好像这个元素变成了<textarea>元素一样 。

```
HTMLElement.contentEditable enumerated attribute

"true"
"false"
"inherit"
```

各个浏览器的支持 contenteditable属性的情况

	Chrome	Edge	Firefox	Opera	Safari	Chrome Android	Firefox for Android	Opera Android	Safari on iOS	Samsung Internet	WebView Android
contentEditable	✓ 1	✓ 12	✓ 3	✓ 9	✓ 3	✓ 18	✓ 4	✓ 10.1	✓ 1	✓ 1.0	✓ 4.4

两者的区别是，基于iframe的实现可以做到样式隔离，不存在与外部元素的样式污染；基于元素的contentEditable属性的实现，不需要增加额外的htmL页面，任意元素都可以设置为可编辑器区域，样式污染可以通过其他的方式规避；

后续的介绍都是基于contentEditable实现的富文本

操作富文本

与富文本编辑器交互的主要方式，就是使用 document.execCommand()。

```
jsbool = document.execCommand(aCommandName, aShowDefaultUI, aValueArgument)
aCommandName
 DOMString
aShowDefaultUI
 Boolean falseMozilla
aValueArgument
 insertImageinsertImage  image  url null
```

浏览器的一些预定义命令

（更多的命令参考：https://developer.mozilla.org/zh-CN/docs/Web/API/Document/execCommand#%E8%A7%84%E8%8C%83）

```
backColor

bold
IE  <strong><b>
insertHTML
 HTML  HTML (IE )
insertImage
  URL  URL IE  null
```

可以在任何时候使用这些命令来修改富文本区域的外观，如下面的例子所示。

```
//
document.execCommand("bold", false, null);//
document.execConmand("italic", false,null);
//www.wrox.com
document.execCommand("createlink", false,
"http://www.wrox.com");
//1
document.execCommand("formatblock", false, "<h1>");
```

需要注意的是，虽然所有浏览器都支持这些命令，但这些命令所产生的 HTML仍然有很大不同例如，执行 bold 命令时，正和 Opera 会使用标签包围文本，Safari 和Chrome使用<b>标签。转换的方式也不一样，因此不能指望富文本编辑器会产生一致的HTML。为了保证一致性的html，很多的第三方框架做了底层适配，上层用户就不用解决这些问题了。

除了命令之外，还有一些命令相关的方法。

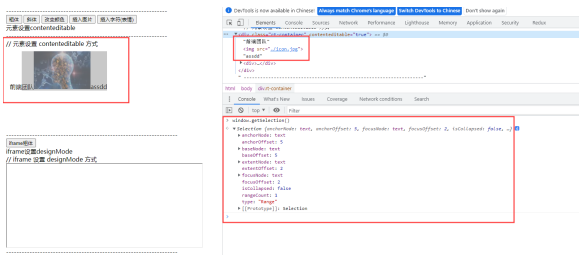
```
Document.queryCommandEnabled()

var isEnabled = document.queryCommandEnabled(command);
command



 Boolean true false

Document.queryCommandState()
Object.queryCommandState(String command)


1 0 -1

Document.queryCommandValue()
var fontSize = document.queryCommandValue('fontSize');

command
```

富文本选区

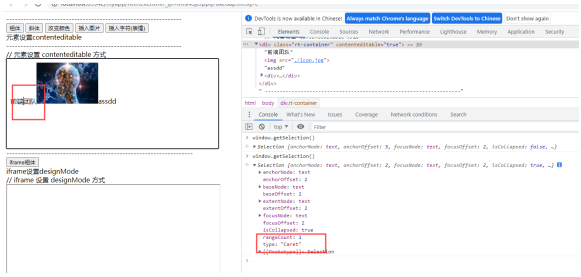
在富文本编辑器中，使用window. getSelection()方法，可以确定实际选择的文本。这个方法是window 对象和 document 对象的属性,调用它会返回一个表示当前选择文本的 Selection对象。每个 selection 对象都有下列属性。

```
anchorNode:
anchorOffset: anchorNode
focusNode:
focusOffset: focusNode
isCollapsed:
rangeCount:DOM
type( Range )( Caret )
```

示例：选中区域



光标闪烁在某一位置时



该对象提供的下列方法以支持选区的操作

```
addRange(range): DOM
collapse(node,offset):
 collapseToEnd():
 collapseTostart():
containsNode(node):
deleteFromDocument0 :,document .execCommandl"delet.falsenull)
extend(nodeoffset):focusNodefocusOffset
 getRangeAt(index): DOM
removeAllRanges(): DOM
reomveRange(range): DOM
selectAllChildren(node):
tostring():
```

Selection 对象的这些方法都极为实用，它们利用了DOM 范围来管理选区由于可以直接操作选择文本的 DOM 表现，因此访问 DOM 范围与使用 execCommand()相比，能够对富文本编辑器进行更加细化的控制。下面来看一个例子。

```
var selection = window.getSelection();//get the range representing the
selection
var range = selection.getRangeAt(0);
//highlight the selected text
var span = document.createElement("span");
span.style.backgroundColor = "yellow";
range.surroundContents(span);
```

以上代码会为富文本编辑器中被选择的文本添加黄色的背景。这里使用了默认选区中的DOM范围通过surroundContents()方法将选区添加到了带有黄色背景的元素中。

> *Range.surroundContents() 方法将 Range 对象的内容移动到一个新的节点，并将新节点放到这个范围的起始处*

富文本内容提取与回显

富文本内容提取与回显需要通过与innerHTML属性交互。

```
Element.innerHTML  HTML
const content = element.innerHTML;
element.innerHTML = htmlString;

DOMString  HTML  innerHTML  htmlString
```

参考文档

https://developer.mozilla.org/zh-CN/docs/Web/API/Document/execCommand

https://www.wangeditor.com/v5/menu-config.html#%E5%9B%BE%E7%89%87