第三部分 BOM和DOM（持续更新）

第12章 BOM

1. 浏览器对象模型（BOM，Browser Object Model）,是以window对象为基础，这个对象代表了浏览器窗口和页面可见的区域。window对象在浏览器中有两重身份，一是ES中的Global对象，一是浏览器窗口的JavaScript接口；
2. 窗口关系：window对象、top对象（最上层窗口，浏览器窗口本身）、parent对象（当前窗口的父窗口）、self对象（始终指向window）；

   CSS像素是web开发中使用的统一像素单位；`window.devicePixelRatio`表示物理像素与逻辑像素之间的缩放系数；（DPI，dots per inch，单位像素密度）；outerWidth返回浏览器窗口自身的大小，包括工具栏滚动条；innerWidth返回浏览器窗口中页面视口的大小；
   `document.compatMode`检查页面是否处于标准模式；
3. `window.open()`方法可以用于导航指定URL，也可以用于打开新浏览器窗口，接收四个参数：要加载的URL，目标窗口，特性字符串（用于指定新窗口的配置）、表示新窗口在浏览器历史记录中是否替代当前加载页面的布尔值；
4. `setTimeout()`用于指定在一段时间后执行某些代码；`setInterval()`用于指定每隔一段时间执行某些代码；一般来说，不要用`setInterval()`；

   系统提示框：`alert()`、`confirm()`、`prompt()`
5. window的对象：
   - `location对象`:以编程方式操纵浏览器的导航系统；它既是window的属性，又是document的属性；

        - URLSearchParams构造函数提供了一组标准的API方法，通过它们可以检查和修改查询字符串。
        - `location.assign("")` 修改location对象修改浏览器的地址；等同于 `window.location=""、location.href=""`；
    - `navigator对象`：提供浏览器的信息；通常用于确定浏览器的类型；

      检测插件：window.navigator.plugins
    - `screen对象`：保存着客户端显示器的信息；（使用不多）
    - `history对象`：提供了操纵浏览器历史纪录的能力；（出于安全考虑，这个对象不会暴露用户访问过的URL，但可以通过它在不知道实际URL的情况下前进和后退）
        - go（num）//num为负表示后退几步，为正表示前进几步；
        - go（“#”）；可以用`back()`和`forward()`代替;
        - `history.pushState()` 接受三个参数：一个state对象、一个新状态的标题、一个（可选的）相对URL；可以通过history.state获取当前的状态对象，也可以使用replaceState()并传入与pushState()同样的前两个参数来更新状态；

          区别：pushState()可以创建历史，可以配合popstate事件，而replaceState()则是替换掉当前的URL，不会产生历史，只会覆盖当前状态。

第13章 客户端检测

1. 能力检测：检测浏览器是否支持某种特性；尽量使用typeof操作符，

   注：!! 做类型判断，如`if(!!a)`

   ```
   if(object.propertyInQuestion){
   //使用object.propertyInQuestion
   }
   ```

2. 用户代理检测：通过用户代理字符串确定浏览器,用户代理字符串包含在每个HTTP请求的头部，在JavaScript中可以通navigator.userAgent访问；

第14章 DOM

文档对象模型（DOM，Document Object Model）是HTML和XML文档的编程接口。实践中要尽量减少DOM操作的数量，因为NodeList对象是“实时更新”的，这意味着每次访问它都会执行一次新的查询。

在HTML页面中，文档元素始终是`<html>`元素；在XML文档中，如果没有这样的预定义，任何元素都可能成为文档元素；

1、DOM中一共有12种节点类型：

节点类型由定义在Node类型上的十二个数值常量表示；

```
Element元素节点                  Node.ELEMENT_NODE(1)
Attr属性节点                   Node.ATTRIBUTE_NODE(2)
Text文本节点                   Node.TEXT_NODE(3)
CDATA节点                Node.CDATA_SECTION_NODE(4)
实体引用名称节点          Node.ENTRY_REFERENCE_NODE(5)
实体名称节点              Node.ENTITY_NODE(6)
处理指令节点              Node.PROCESSING_INSTRUCTION_NODE(7)
Comment注释节点                Node.COMMENT_NODE(8)
Document文档节点                Node.DOCUMENT_NODE(9)
DocumentType文档类型节点             Node.DOCUMENT_TYPE_NODE(10)
DocumentFragment文档片段节点              Node.DOCUMENT_FRAGMENT_NODE(11)
DTD声明节点               Node.NOTATION_NODE(12)
```

- Node类型

  appendChild()用于在childNodes列表末尾添加节点；**insertBefore()**在指定位置前面插入节点；如果参照节点是null，则二者效果相同；**replaceChild()、removeChild()**

- Document类型

  在浏览器中，文档对象document是HTMLDocument的实例，表示整个HTML页面。

  特征如下：

  1. nodeType的值为9；
  2. nodeName的值为“#document"；
  3. nodeValue、parentNode、ownerDocument的值为null；
  4. 其子节点可能是一个DocumentType（最多一个），Element（最多一个）或ProcessingInstrucion，Comment类型；

- Element类型

  - 可以通过nodeName或tagName属性来获取元素的标签名；
  - getAttribute()、setAttribute()、removeAttribute()、
  - Element类型是唯一使用attributes属性的DOM节点类型；attributes属性最有用的场景是需要迭代元素上所有属性的时候；

  特征如下：

  1. nodeType的值为1；
  2. nodeName值为元素的标签名；
  3. nodeValue的值为null
  4. parentNode的值为Document或Element对象；
  5. 其子节点可以是Element、Text、Comment、ProcessingInstrucion、CDATASection、EntityReference类型；

2、DOM编程

动态脚本：有两种方法通过`<script>`动态为网页添加脚本：引入外部文件和直接插入源代码；

动态样式：`<link>`(添加在`<head>`中)和`<style>`

3、MutationObserver接口

可以在DOM被修改时异步执行回调；取代了之前的MutationEvent；

```
let observer=new MutationObserver(()=>console.log('DOM was mutated!'));
observer.observe(document.body,{attributes;true});
```

通过`disconnect()`可提前终止回调，但可以通过`observer.observe()`恢复关联；

第15章 DOM扩展

三个规范：

1. Selectors API：`querySelector()`、`querySelectorAll()`(该方法返回一个NodeList的静态实例)、`matches()`
2. Element Traversal
3. HTML5

跨站脚本攻击XSS

第16章 DOM2和DOM3

1、DOM2 Core在一些类型上新增了与XML命名空间有关的新方法。这些变化只有在使用XML或XHTML文档时才会用到，在HTML文档中则没有用处。

2、XML

命名空间 - xmlns 属性（命名空间是在元素的开始标签的 **xmlns 属性**中定义的）；XML 文档中的所有文本均会被解析器解析，只有 CDATA 区段中的文本会被解析器忽略；

Document类型的更新中唯一与命名空间无关的方法时importNode()；

- XML 被设计用来传输和存储数据，其焦点是数据的内容。
- HTML 被设计用来显示数据，其焦点是数据的外观。

XML语法规则：

- XML 文档必须有一个根元素
- XML元素都必须有一个关闭标签
- XML 标签对大小写敏感
- XML 元素必须被正确的嵌套
- XML 属性值必须加引号

3、DOM样式

- 样式的定义方式：外部样式表（使用`link`元素）、文档样式表（使用`<style>`元素）、元素特定样式（使用style属性）
- DOM2 Style规范：
  - cssText：设置cssText是一次性修改元素多个样式最快捷的方式；
  - getPropertyValue()返回CSS属性值的字符串表示，这个方法会返回CSSValue对象，这个对象有两个属性：cssText和cssValueType；
  - removeProperty()方法用于从元素样式中删除指定的CSS样式；
- 通过document.styleSheets集合可以访问文档上所有的样式表；

3、元素尺寸

- 偏移尺寸：**offsetHeight**（高度、水平滚动条高度、上下边框的高度）；**offsetTop**（元素上边框外侧距离包含元素上边框内侧的像素数） **(offsetLeft、offsetWidth)**
- 客户端尺寸（client dimensions）：元素内容+内边距； **(clientHeight、clientWidth)**
- 滚动尺寸：**scrollHeight**：没有滚动条出现时，元素内容的总高度；**scrollTop**：内容区顶部隐藏的像素数，设置这个属性可以改变元素的滚动位置；（scrollWidth、scrollLeft）

4、遍历

- NodeIterator和TreeWalker可以对DOM树执行深度优先的遍历；
  - NodeIterator可以通过`document.creatNodeIterator()`方法创建其实例；其接收四个参数：`root`、`whatToShow`（数值代码）、`filter，NodeFilter对象或者函数`（表示是否接收或者跳过特定节点）、`entityReferenceExpansion`（布尔值，表示是否扩展实体引用）

```
//创建一个简单的遍历所有节点的NodeIterator
let  iterator = document.creatNodeIterator( root,NodeFilter.SHOW_ELEMENT,
filter, false)
```

  - TreeWalker是NodeIterator的高级版；可以通过`document.creatTreeWalker()`方法创建其实例；其接收同上四个参数：
- 通过范围的选区可以在保持文档结构完好的同时从文档中移除内容，也可复制文档中相应的部分；范围在常规DOM操作的粒度不够时可以发挥作用；

```
let range=document.creatRange();//创建DOM范围对象
```

  - 一个范围具有两个边界点，即一个开始点和一个结束点。每个边界点由一个节点和那个节点的偏移量指定。该节点通常是 Element 节点、Document 节点或 Text 节点。对于 Element 节点和 Document 节点，偏移量指该节点的子节点。偏移量为 0，说明边界点位于该节点的第一个子节点之前。偏移量为 1，说明边界点位于该节点的第一个子节点之后，第二个子节点之前。但如果边界节点是 Text 节点，偏移量则指的是文本中两个字符之间的位置。
  - 方法：selectNode()、selectNodeContents()、setStart()、setEnd()、deleteContents()、extractContents()等十八种
  - range.setStart和range.setEnd(参照节点，偏移量)；
  - 调用 detach() 方法后，对这些属性的任何读操作都会抛出代码为 INVALID_STATE_ERR 的 DOMException 异常

第17章 事件

理解事件的原理和对性能的影响

1、事件流

事件是JavaScript与网页结合的主要方式；事件流描述了页面接收事件的顺序；

IE事件流被称为事件冒泡；

事件流主要分为三个阶段：事件捕获（1）、到达目标（2）、事件冒泡（3）；

2、事件处理程序

为响应事件而调用的函数被称为事件处理程序，事件处理程序的名字以“on”开头；特定元素支持的每个事件都可以使用事件处理程序的名字，以HTML属性的形式来指定，此时属性的值必须是能够执行的JavaScript代码；

在HTML事件中处理事件程序有一些问题：

- 时机问题，大多数HTML事件处理程序会封装在try/catch块中；
- 对事件处理程序作用域链的扩展，在不同浏览器中可能导致不同的结果；

1. DOM0事件处理程序（把属性赋值给一个函数）

```
btn.onclick=function(){}
btn.onclick=null;//移除事件处理程序
```

2. DOM2事件处理程序

   `addEventListener()`和`removeEventListener()`参数：事件名、事件处理函数、布尔值（true表示在捕获阶段调用事件处理程序，false（默认值）表示在冒泡阶段调用事件处理函数）；

3. IE事件处理程序

   `attachEvent`和`detachEvent`参数：事件处理程序的名字和事件处理函数；

   - 事件处理程序会以添加他们的顺序反向触发；
   - 无法移除作为事件处理程序添加的匿名函数；

3、事件类型

- 用户页面事件
  - load事件会在整个页面（包括所有外部资源：如图片、js文件、css文件）加载完成后触发，两种方式：JavaScript方式（常用）、指定load事件处理程序的方式是向`<body>`元素添加onload属性；
- 焦点事件
- 鼠标事件
- 滚轮事件
- 输入事件
- 键盘事件
- 合成事件

4、内存与性能的问题：

- 最好限制一个页面中事件处理程序的数量，因为它们会占用过多内存，导致页面响应缓慢；
- 利用事件冒泡，事件委托可以解决限制事件处理程序数量的问题；
- 最好在页面卸载之前删除所有事件处理程序；

事件委托

第18章 动画与Canvas图形

1、时间间隔问题

- `requestAnimationFrame()`：该方法接收一个参数，此参数是一个要在重绘屏幕前调用的函数；
- `cancelAnimationFrame()`：取消重回任务

减少调用频率：

- 函数防抖：将几次操作合并为一此操作进行；函数防抖只是在最后一次事件后才触发一次函数；
- 节流：使得一定时间内只触发一次函数； 函数节流不管事件触发有多频繁，都会保证在规定时间内一定会执行一次真正的事件处理函数；

2、2D上下文

`<canvas> </canvas>`是html5最受欢迎的特性；

`getContext()`方法可以获取对绘图上下文的引用，对于平面图形，传入参数“2d"，表示要获取2D上下文图像；

- 填充描边：`fillStyle`、`strokeStyle`

- 绘制矩形：`fillRect()`、`strokeRect`、`clearRect`
- 绘制弧形：
- 绘制路径：`beginPath()`、`stroke()`、`lineTo`、`moveTo`、`isPointPath()`
- 绘制文本：(方法)`fillText()`、`strokeText()`；（属性）`font、textAlign、textBaseLine`
- 变换：rotate()、scale()、translate()、transform()、
- 绘制图像、阴影、渐变（`createLinearGradient()、createRectLinearGradient()`）、图案、图像数据、合成、

html创建canvas画布失败"cannot read property 'getContext' of null"原因及解决方法：

- 在head标签中时，一般只是进行加载，并不立即自动执行，需要被触发后才会执行。故引入`window.onload=function(){}`等触发事件；
- 在body标签结束前，javascript会被加载并立即执行；故可以直接将script标签置于body标签结束之后，或者`<canvas>`之后；

3、3D上下文

WebGL、 OpenGL

第19章 表单脚本

1、Web表单在HTML中以

元素表示，在JavaScript中则以`HTMLFormElement`类型表示。`HTMLFormElement`类型继承自`HTMLElement`类型

2、*iframe*元素会创建包含另外一个文档的内联框架（即行内框架）

3、**表单序列化：**

4、**富文本编辑：**基本的技术就是在空白HTML文件中嵌入一个iframe，通过designMode属性，可以将这个空白文档变成可以编辑的，实际编辑的则是`<body>`元素的HTML。与富文本编辑器交互的主要方法是使用`document.exexCommand()`；