# Web APIs基础学习

Web APIs是w3c组织的标准，主要包括DOM和BOM

# DOM

##### 1、innerText和innerHTML

innerText是不标准的，不会识别html标签，且去除空格和换行。

innerHTML则是w3c的标准，可以识别html标签、空格和换行。

##### 2、setAttribute（属性，值）和element.属性 = 值

setAttribute主要用于自定义属性

element.属性 = 值主要用于DOM自带的属性

element.style只能获取到行内样式

##### 3、h5规定自定义属性以data开头，如“data-myAttr”

##### 4、javascript:void(0);

**void** 是 JavaScript 中非常重要的关键字，该操作符指定要计算一个表达式但是不返回值。

创建了一个死链接：

<a href="javascript:void(0)">单击此处什么也不会发生</a>

##### 5、注册事件

①传统的注册事件方式

button.onclick=function(){},同一个元素的同一个事件只能设置一个处理函数，后注册的处理函数会覆盖之前的处理函数。

②事件监听注册方式

w3c推荐，eventTarget.addEventListener（事件类型，处理函数[,useCapture]）。同一元素同一事件可注册多个处理函数，且按顺序执行。

useCapture默认为false，表示在事件冒泡阶段调用事件处理函数。

指的是在事件捕获阶段调用处理函数。

③DOM事件流，事件的传播过程

捕获->目标->冒泡

每个事件处理函数只能指定在捕获阶段或者在冒泡阶段其中一个阶段执行，对于onclick只能在事件冒泡阶段执行，而addEventListener则用useCapture来指定。

实际开发很少使用实际捕获，我们更多地关注事件冒泡。

有些事件没有冒泡，如onblur，onfocus，onmouseenter，onmouseleave

##### 6、事件对象

e.target跟this的区别

e.target指向触发事件的元素，this指向绑定事件的元素。this与e.currentTarget相似。

e.type：当前事件类型。

e.preventDefault()：阻止默认事件，如链接不跳转或提交按钮不提交。

e.stopPropagation(): 阻止事件冒泡。

##### 7、事件委托

给父节点添加事件监听器，用户点击子节点，通过事件冒泡传播到父节点，使用e.target得到触发事件的子节点。

因此可以不用遍历每个子节点绑定事件，提高效率。

##### 8、阻止右键菜单

document.addEventListener('contextmenu',function(e){  
 e.preventDefault();  
})

##### 9、禁止选中文字

document.addEventListener('selectstart',function(e){  
 e.preventDefault();  
})

##### 10、鼠标事件对象MouseEvent

e.clientX,e.clientY：鼠标在窗口的坐标

e.pageX,e.pageY：鼠标在页面文档的坐标

##### 11、键盘事件

onkeyup：按键弹起触发

onkeydown：按键按下触发，长按一直触发

onkeypress：按键按下触发，长按一直触发，但不能识别功能键如（ctrl，shift，方向键等）

执行顺序keydown--keypress--keyup

键盘事件对象KeyboardEvent

e.keyCode:返回该键的ASCII值，keyup和keydown不区分大小写，按a或A得到都是65，而keypress区分大小写，按A得65，按a得97

# BOM(浏览器对象模型)

补充：浏览器私有前缀。

历史背景：浏览器厂商对css做新的尝试，而CSS标准发布相对来说一直是滞后的。

1、为什么需要写私有前缀？

老版本的浏览器实现css新特性时，w3c标准还未发布。这些特性只作为浏览器的私有特性，只支持后缀的写法，所以需要加上私有前缀。

2、什么时候不需要写？

当一个属性成为标准，并且被Firefox、Chrome等浏览器的最新版普遍兼容时，如果不想兼容老版本浏览器，可以不写。使用工具也可以免手写。

BOM缺乏标准，兼容性差，但各浏览器的BOM大部分是一样的。



window对象是浏览器的顶级对象，是BOM的核心对象。

window对象有一个特殊属性，window.name，所以定义全局变量的时候注意避开name。

##### 1、页面加载事件

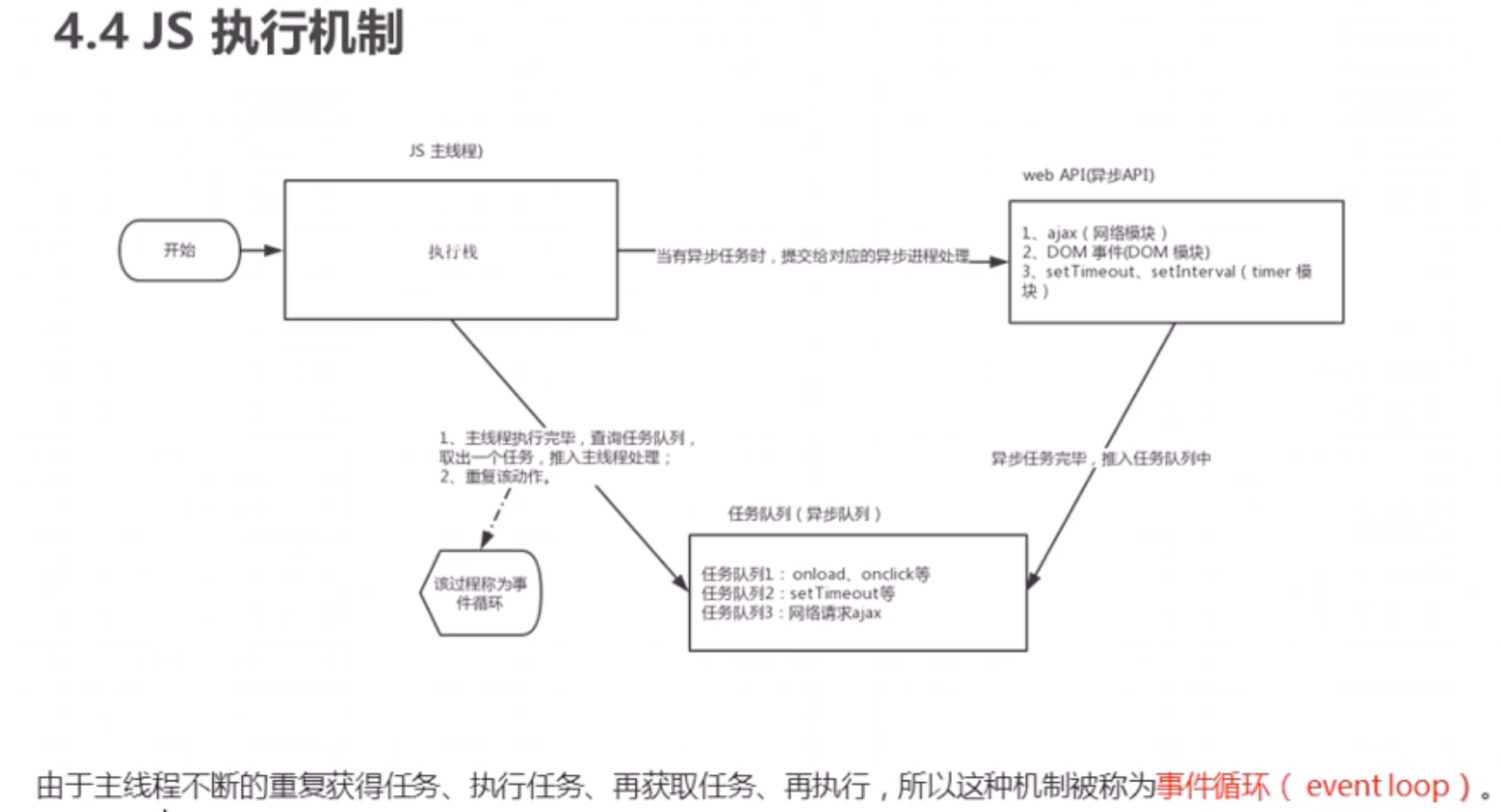
window.addEventListener('load',function(){}),页面完全加载，包括图片、css、flash等

document.addEventListener('DOMContentLoad',function(){}),DOM加载完毕时触发，不包括图片、css、flash等。

##### 2、window.onresize

窗口大小改变时触发，

##### 3、事件循环



##### 4、location对象

location.href:获取当前页面url，亦可跳转页面。

location.assign:可以跳转页面，跟href一样，记录浏览历史，可以后退

location.replace:替换当前页面，不记录浏览历史，所以不可以后退

location.reload:相当于刷新，加参数ture则强刷

##### 5、history对象

back、foward、go方法

# 其他WebAPIs

1、offsetTop和offsetLeft，元素与带有定位的父元素的偏移，不带单位。

2、element.offsetWidth和element.offsetHeight，元素的宽度和高度（border+padding+content）

3、element.clientWidth和element.clientHeight与element.offsetWidth和element.offsetHeight区别在于不包含边框的大小。

element.clientTop和element.clientLeft则返回其上边框、左边框的大小。

# 第十二章 BOM

## 1、Global作用域

浏览器以window对象实现了ECMAScript 中的 Global 对象。

var在全局作用域定义的变量，会添加到window对象中。而使用const和let定义的变量，则不会把他们添加给window。浏览器将其放在一个块级作用域中。



## 2、窗口关系

window.parent:当前窗口的父窗口

window.top:始终指向最上层(最外层)窗口，即浏览器窗口本身。

window.self === window

### 3、窗口位置与像素比

screenLeft 和 screenTop 属性，用于表示窗口相对于屏幕左侧和顶部的位置 ，返回值的单位是 CSS 像素。浏览器全屏，则它们都是0。

逻辑分辨率是显示屏分辨率。两者间由 Scale Factor（缩放因子） 计算得到，通常 物理分辨率 > 逻辑分辨率，而 物理分辨率 \* 缩放因子 = 逻辑分辨率；但在使用 台式电脑、和 笔记本 的情况下 物理分辨率 = 逻辑分辨率。

window.devicePixelRatio 表示物理像素与逻辑像素之间的缩放系数。对于分辨率从 1920×1080 转换为 640×320 的设备，window. devicePixelRatio 的值就是 3。这样一来，12 像素(CSS 像素)的文字实际上就会用 36 像素的物理 像素来显示。



### 4、窗口大小

在移动设备上，window.innerWidth 和 window.innerHeight 返回视口的大小。

### 5、滚动

// 正常滚动   
 window.scrollTo({  
 left: 100,  
 top: 100,  
 behavior: 'auto'  
 });

// 平滑滚动  
 window.scrollTo({   
 left: 100,  
 top: 100,  
 behavior: 'smooth'  
 });