WebAPI

1.认识DOM

概念: 文档对象模型（Document Object Model，简称DOM），是W3C组织推荐的处理可扩展标记语言的标准编程接口。

## DOM 中常用的操作

1.获取元素

2.对元素进行操作(设置其属性或调用其方法)

3.动态创建元素

4.给元素注册事件

3. document对象 ====== **概念:** document对象代表在浏览器中加载的页面

4.获取页面中的元素

什么是元素?

​ html中的标签在DOM中称为元素

为什么要获取页面上的元素呢?

​ 因为:我们想要操作页面上的元素,首先需要获取到对应的元素，然后才能进行后续操作

**4.1 根据id获取元素**

**语法:** document.getElementById('id名');

**作用:** 在整个文档中查找id名为传入的值的元素,如果没有返回null

### **4.2 根据标签名获取元素**

**语法:** document.getElementsByTagName('标签名');

**作用:** 在整个文档中查找所有标签名为传入的值的元素,将这些所有符合条件元素的存放到一个伪数组中返回出来,如果没有就返回**一个空的伪数组**

### **4.3 根据name属性获取元素 (有兼容问题,不同的浏览器实现不同)**

**语法:** document.getElementsByName('name属性的值')

**作用:** 在整个文档中查找所有name属性值为传入的值的元素,将这些所有符合条件元素的存放到一个伪数组中返回出来,如果没有就返回**一个空的伪数组**

### **4.4 根据类名获取元素 (有兼容问题,ie9+支持)**

**语法:** document.getElementsByClassName('类名')

**作用:** 在整个文档中查找所有class属性值为传入的值的元素,将这些所有符合条件元素的存放到一个伪数组中返回出来,如果没有就**返回一个空的伪数组**

### **4.5 根据选择器获取元素 (有兼容问题,ie8+支持)**

**语法:** document.querySelector('选择器');

**作用:** 在整个文档中查找所有符合选择器值的元素,但是只返回其中的第一个元素,如果**没有返回null**

**注意:** 如果想要所有符合选择器值的元素,请使用 querySelectorAll方法

**querySelectorAll返回的是一个伪数组,如果没有则返回空的伪数组**

### **4.6 DOM中元素可以使用的获取元素的方法**

element.getElementsByTagName('标签名')

element.getElementsByClassName('类名')

element.querySelector('选择器')

element.querySelectorAll('选择器')

以上这些方法也可以使用获取到的DOM对象调用

使用document调用这些方法是在整个页面中查找

使用获取到的DOM对象调用这些方法,是在当前DOM对象的里面查找

**小结：**

1.通过document这个对象调用获取元素的方法,是从整个页面的范围内查找 2.通过元素(elelement)调用方法,是在元素当前范围内查找 3.document.getElementById 返回的是对应的DOM元素, 如果没有返回null

4.document/element.querySelector 返回的是符合条件的第一个DOM元素,如果没有返回null

5.document/elementgetElementsByTagName 返回的是存储DOM元素的伪数组,如果没有返回空的伪数组

6.document/elementgetElementsByClassName 返回的是存储DOM元素的伪数组,如果没有返回空的伪数组

7.document.getElementsByName 返回的是存储DOM元素的伪数组,如果没有返回空的伪数组

8.document/element.querySelectorAll 返回的是存储所有符合条件的DOM元素的伪数组,如果没有返回空的伪数组

5.事件

#### **5.3.1 事件的三要素**

#### 1.事件源 : 监听事件的元素

2.事件名 : 监听的事件类型

3.事件处理函数 : 触发事件时,要调用的函数

#### **5.3.2 如何注册事件 ===== **语法:** 事件源.on + 事件名 = 事件处理函数**

### **5.4 给a标签注册事件时,需要注意的问题**

a标签是超链接标签,超链接标签默认是用来做页面跳转的,虽然我们没有在href属性中书写正确的url地址,但是当我们点击a标签时,页面也会重新刷新。所以当我们点击a标签时,触发了事件处理函数,并且执行了console.log(1)这行代码,但是当这行代码执行完毕之后,很快a标签的默认行为导致浏览器重新刷新了一遍,由于执行的速度非常快,所以我们肉眼根本看不到控制台上输出的1. 所以为了解决这个问题,我们需要阻止a标签默认行为执行

即 ：在事件处理函数中的最后一行写return false,就会阻止a标签默认行为的执行

## **DOM中常用的非表单元素属性**

#### **为什么要学习DOM中常用的非表单元素属性?**

我们学习DOM就是为了通过访问元素的属性或者方法,来动态的改变元素的状态,比如,用js修改a标签href属性的值

#### **常用的非表单元素属性有哪些?**

#### **href 超链接的url地址**

#### **title 标签的标题属性**

#### **id 标签的id属性**

#### **src 引入外部文件的路径**

#### **innerText 标签内的文本**

#### **innerHTML 标签内的内容**

#### **6.3 获取属性的值/设置属性的值**

#### **6.4 innerText 和 innerHTML**

**作用:** 给双标签的元素设置内容/获取双标签里面的内容

**不同点:**

innerText 只是用于获取文本或设置文本

innerHTML 不仅可以用于设置/获取文本,还可以识别html

**相同点: ========** 如果是赋值的话,都会覆盖元素内本身的内容

**注意: ========** 这两个是用于双标签的属性

### **6.4.2 innerText 和 innerHTML的兼容问题**

### **1.**innerHTML是非标准属性(非w3c标准),但是所有的浏览器都支持

### 2.innerText属性存在兼容性问题,早期的火狐浏览器不支持该属性,使用textContent替代

Day02

## DOM 中常用的表单元素的属性

* 1. value 用于大部分表单元素的内容获取(option除外)
  2. type 可以获取input标签的类型(输入框或复选框等)
  3. disabled 禁用属性
  4. checked 复选框选中属性
  5. selected 下拉菜单选中属性

注意：

disabled, checked, selected 这些布尔值属性,在DOM中通过true/false修改状态

true 是让属性的作用生效,false 为不生效

## DOM中操作样式的两种方式

### **通过style属性操作样式**

**注意:** 通过style属性设置样式时,css中要写单位的属性,在js代码中也要加单位

通过style属性设置样式执行完毕之后的html标签的变化

相当于给标签添加了行内样式

### **通过className操作样式**

通过className属性设置样式执行完毕之后的html标签的变化

相当于给标签添加了类名

小结 ：

1. 通过style属性操作样式,相当于给标签添加了行内样式.获取的时候,只能获取到行内写好的样式
2. 通过className属性操作样式,相当于给标签添加了类名, 赋值的时候,会覆盖原来的

Day03

1. 标签上自定义属性的操作
   1. **获取自定义属性的值**

element.getAttribute('属性名') 返回对应属性的值 ,如果没有返回null

### **1.2设置自定义属性的值**

element.setAttribute('属性名', '属性的值') 返回undefined

**1.3移除自定义属性的值**

element.removeAttribute('属性名') 返回undefined

**小结：**

1.操作自定义属性的这三个方法,不仅可以操作自定义属性,同时也可以操作html规范中的属性

2.设置的属性的值,最终都转换成了字符串的形式

## **节点**

## 1.什么是节点：

根据 W3C 的 HTML DOM 标准，HTML 文档中的所有内容都是节点(文档,标签,文本,属性,注释)

（1）整个文档是一个文档节点

（2）每个HTML标签是一个标签节点/元素节点

（3）HTML 标签内的文本是文本节点

（4）每个HTML标签的属性是属性节点

（5）注释是注释节点

### 2 什么是DOM树

当浏览器加载文档的时候,会将文档中每一个节点,以树状的结构组织起来,我们将这种结构称为之节点树或DOM树

### 3 节点中常用的属性

1.nodeType 节点的类型 ======== 元素节点的nodeType 是1

2.nodeName 节点的名称 ========= 元素节点的nodeName 是大写的标签名

3.nodeValue 节点值 ========= 元素节点的nodeValue始终是null

4.找儿子的属性:

（1）children 返回包含所有子元素的伪数组

（2）childNodes 返回包含所有的子节点的伪数组

（3）firstChild 返回第一个子节点

（4）lastChild 返回最后一个子节点

（5）firstElementChild 返回第一个子元素

（6）lastElementChild 返回最后一个子元素

5.找父亲的属性: -====== parentNode 返回父节点

6.找兄弟

（1）nextElementSibling 返回下一个兄弟元素

（2）previousElementSibling 返回上一个兄弟元素

（3）nextSibling 返回下一个兄弟节点

（4）previousSibling 返回上一个兄弟节点

Day04

## 1.操作节点的方法

（1）appendChild() 往节点的内容后面添加一个子节点

****语法:** 父节点.appendChild(子节点)**

（2）insertBefore() 往节点里面添加一个子节点,这个子节点的位置在参考子节点之前

****语法:** node.insertBefore(要插入的节点, 参考子节点)**

（3）removeChild() 移除某个子节点

****语法:** node.removeChild(要被删除的节点); 返回被移除的节点**

（4）replaceChild() 替换节点使用其他节点替换掉自己的某个子节点

****语法:** element.replaceChild(替换的节点, 被替换的子节点);**

（5）cloneNode() 克隆节点

****语法:** element.cloneNode([true]) ;**

参数是boolean值,不传参数默认是false,为浅拷贝.传true,则是深拷贝

### **小结:**

1.有剪切效果的方法:

（1）appendChild() （2）insertBefore() （3）replaceChild()

2.移除的方法

removeChild()

3.需要传递两个参数的方法:

insertBefore(要插入的节点, 参考节点)

replaceChild(替换的节点, 被替换的节点)

## 2. 创建元素的三种方式

### 1. innerHTML

### ****语法:** element.innerHTML = '标签字符串'**

### 2. docuement.write()

****语法:** document.write('标签字符串')**

### 3. document.createElement()

****语法:** document.createElement('标签名')**

### **小结:**

innerHTML 会覆盖原来元素里面的内容,并且会将新的元素直接渲染到页面上,但是不推荐用来创建表格(常用)

document.write() 也会直接渲染到页面上,并且在事件中执行的话,会覆盖原来页面的内容(慎用)

docuement.createElement() 只是在内存中创建出来一个元素,但是并没有添加到DOM树中(常用

## **3. 创建元素的三种方式的效率问题**

（1）document.write() 和 innerHTML都是解析字符串执行效率相同

（2）document.createElement 直接创建元素,执行效率比前两者高很多

（3）document.write() 和 innerHTML优化后的代码效率跟document.createElement类似

Day05

1. 注册事件的其他方式

以前是 on+事件名

后来,W3C DOM 规范中提供了注册事件监听的另外一种方式 : addEventListener

**优点: 1.**它允许给一个事件注册多个 listener。

2.它提供了一种更精细的手段控制 listener 的触发阶段。（即可以选择捕获或者冒泡）。

**语法：element.addEventListener('事件名', 事件处理函数, useCapture);**

**注意： 1.**事件名不需要写on

2.useCapture 是一个boolean值,用于控制事件触发的阶段,不写默认是false

3.false 是在冒泡阶段执行

4.true 是在捕获阶段执行

2. 移除事件的方式

### **2.1 on+事件名 的移除方式 =======** on + 事件名 = null;

### **2.2 addEventListener的移除方式 =======** removeEventListener()

总结：

**1.注册事件的两种方式**

on + 事件名 = function(){}

addEventListener('事件名', 事件处理函数, useCapture )

**2.移除事件对应的两种方式**

on + 事件名 = null;

removeEventListener('事件名', 事件处理函数, useCapture )

注意 : 注册时,事件处理函数不能使用匿名函数的形式

### 3.1 事件对象的常用属性

### 1.event.type 返回事件类型(也就是事件名);

### 2.event.target 返回事件目标

### 3.clientX/clientY 返回鼠标在可视窗口的坐标

### 4.pageX/pageY 返回鼠标在当前页面的坐标

### 5.keyCode 返回键盘按键对应的数字

### 3.2 事件对象的常用方法

**1.event.preventDefault() 取消默认行为** （如果使用addEventListener注册事件的话,return false 是无效的.所以我们需要使用 event.preventDefault()来阻止a标签的默认行为）

**2.event.stopPropagation() 阻止事件传递**

## 4. 事件流 ======== 共有三个事件阶段：捕获阶段，目标阶段和冒泡阶段。

### **捕获阶段：**

### 事件对象通过目标的祖先从窗口传播到目标的父项。这个阶段也被称为捕获阶段

### **目标阶段：** 事件对象到达事件目标。这个阶段也被称为目标阶段。

### **冒泡阶段：**

### 事件对象以相反的顺序通过目标的祖先传播，从目标的父项开始，以窗口结束。这个阶段也被称为冒泡阶段

## 事件委托

### **什么是事件委托 :**

本来是要注册给自己的事件,注册给了父元素.事件触发后的事情,委托给父元素执行

### **5.2 事件委托的好处 :** 代码简洁 节省内存

### **5.3 事件委托的原理 : 事件冒泡**

Day06

## 1.BOM

**概念:** BOM(Browser Object Model) 是指浏览器对象模型,浏览器对象模型提供了独立于内容的、可以与浏览器窗口进行互动的对象结构。BOM由多个对象组成，其中代表浏览器窗口的Window对象是BOM的顶层对象，其他对象都是该对象的子对象。

**通俗理解:** 把浏览器当做对象,通过访问对象的属性,实现操作浏览器的一组方法

## 2. 页面加载事件

**window.onload事件 是页面所有资源加载完成时触发**

**window.onunload事件 是用户退出页面时触发**

## 3.location对象

location 相当于浏览器地址栏的抽象, 通过window.location可以访问到,window可以忽略不写

我们学习location对象,目的是为了操作地址栏里面的url地址

#### 什么是url

**概念:** 统一资源定位符 (Uniform Resource Locator, URL)

**通俗理解:** 互联网中的地址

#### URL的组成

<http://www.itheima.com:80/a/b/index.html?name=zs&age=18#bottom>

1.scheme:通信协议 ===== 常用的http,ftp,maito等

2.host:主机 (找计算机) ==== www.itheima.com

服务器(计算机)域名系统 (DNS) 主机名或 IP 地址。

3.port:端口号 (找软件) ======== :80

整数，可选，省略时使用方案的默认端口，如http的默认端口为80。

4.path:路径 ======= /a/b/index.html

由零或多个'/'符号隔开的字符串，一般用来表示主机上的一个目录或文件地址。

5.query:查询 ====== name=zs&age=18

可选，用于给动态网页传递参数，可有多个参数，用'&'符号隔开，每个参数的名和值用'='符号隔开。

6.fragment: 信息片断 ==== 字符串，锚点. ===== bottom

### 5.location有哪些成员？===== **成员: 属性和方法又叫做成员**

### 1.href 返回地址栏的整个url,设置值的时候,也可以起到跳转页面的作用 2.hash 返回地址栏url的fragment

### 3.host 返回地址栏url的主机名

### 4.search 返回地址栏url的键值对参数(query)

5.reload() 设置页面刷新 ===== ****语法:** location.reload([boolean])**

参数是一个布尔值,不传参数则默认是false. 普通刷新,可能会从缓存拿数据 传true的话,是强制刷新.强制浏览器去服务器获取数据

6.assign() 设置页面跳转,记录历史==== ****语法:** location.assign('url地址')**

7.replace() 设置页面跳转,不记录历史===****语法:**location.replace('url地址')**

## 6.history对象

1.back() 返回上一个页面 =====  ****语法:** history.back();**

2.forward() 前进到下一个页面 ===== ****语法:** history.forward();**

3.go(1/-1) 前进/ 返回 ===== **语法:** history.go(number);

正数是前进,负数是后退.

* history.go(1); 前进一步
* history.go(2); 前进两步
* history.go(-3); 后退三步

## navigator对象

userAgent 返回识别客户端设备和浏览器的字符串

****语法:** navigator.userAgent**

## 7. 定时器

### 1.设置定时器 ====

### （1）**setTimeout()**

**作用:** 到达间隔时间之后,只调用一次回调函数

**语法:** window.setTimeout(回调函数, 间隔时间)

1. window可以省略
2. 间隔时间以毫秒为单位
3. 返回这个定时器的标识符,是数字类型

#### **（2）setInterval()**

**作用:** 每隔一个间隔时间,就调用一次回调函数

**语法:** window.setInterval(回调函数, 间隔时间)

### 2.清除定时器

#### （1）**clearTimeout()**

**作用:** 清除以setTimeout方法设置的定时器

**语法:** widnow.clearTimeout(定时器的标识符) window可以忽略

#### （2）**clearInterval()**

**作用:** 清除以setInterval方法设置的定时器

**语法:** widnow.clearInterval(定时器的标识符) window可以忽略

Day07

## 特效

### （1）offset系列

* + 1. offsetParent 用于获取定位的父级元素
    2. offsetLeft 距离定位父元素的左偏移量
    3. offsetTop 距离定位父元素的上偏移量
    4. offsetWidth 当前元素的宽度
    5. offsetHeight 当前元素的高度

**offsetParent和parentNode的区别?**

offsetParent 返回的是离自己最近的定位父元素

parentNode 返回的是直接父元素

### client系列

* + 1. clientWidth 元素可视区的宽度
    2. clientHeight 元素可视区的高度

### scroll系列

* + 1. scrollLeft 元素中内容左侧滚动出去的距离
    2. scrollTop 元素中内容顶部滚动出去的距离
    3. scrollWidth 元素中内容的宽度
    4. scrollHeight 元素中内容的高度

### 获取浏览器可视区的大小

* + 1. window.innerWidth 浏览器可视区的宽度
    2. window.innerHeight 浏览器可视区的高度

### 获取页面滚动出去的距离

* + 1. window.pageYOffset 顶部滚动出去的距离
    2. window.pageXOffset 左侧滚动出去的距离