1. 在html文档中插入JavaScript
2. 文档页面的加载顺序是从上往下，得先有标签才有程序操作。因此，将script标签写在</body>的前面

**奇技淫巧：i. console.log( )在控制台输出**

**ii. console.dir( )在控制台打印元素对象，输出其属性及方法**

1. 获取元素的方法 **记得document**
2. document . getElementById(“ 字符串 ”)，返回一个元素对象
3. document . getElementsByTagName(“ 标签名 ”)，返回一个对象数组。使用通配符“\*”，则返回所有的元素
4. document . getElementsByClassName(“ 字符串 ”)，返回一个对象数组
5. document . querySelector(“ 符号+字符串 ”)，返回指定选择器的**第一个元素对象**。例如：document . querySelector(“ h3 ”) document . querySelector(“ #nav ”)

document . querySelector(“ .box ”)

1. document . querySelectorAll(“ 符号+字符串 ”)，返回指定选择器的**所有元素对象的集合**
2. **获取body元素：document . body**

**获取html元素：document . documentElement**

1. 事件

事件三要素：i. 事件源：事件被触发的对象 var btn = document.getElementById(“btn”)

ii．事件类型：例如onclick

iii. 事件处理函数：btn . onclick = function(){ 语句; }

相对应的，有：获取事件源、绑定事件（btn.onclick）、添加处理函数。

常见的事件类型

onclick 鼠标左键点击

onmouseover 鼠标经过

onmouseout 鼠标离开

onfocus 获得焦点

onblur 失去焦点

onmousemove 鼠标移动

onmouseup 鼠标弹起

onmousedown 鼠标按下

手动调用事件函数：函数名.事件名（） 可直接调用事件函数，不需触发事件

1. 操作元素
2. 改变元素内容 元素.innerText=” ” 自动去除内容中空格和换行，不识别html标签

元素.innerHTML=” ” 保留空格和换行，识别html标签

例如：li.innerHTML=text.value+”<a href=’javascript:;’[[1]](#footnote-1)>删除</a>”可以在输入的内容后添加一个a标签

亦可通过这两个属性来获取相对应元素的内容，console.log(p.innerHtml)

1. 获取当前时间 year=date.get**Full**Year() month=date.getMonth()**+1**

的函数 date=date.getDate()获取日期 day=date.getDay()获取星期几

1. 修改元素属性 **元素.属性=“新的属性值”**
2. **表单元素的内容是通过value属性修改的**
3. **this指向事件函数的调用者** btn.onclick=function(){ }，则，this指btn
4. 修改css样式
5. 通过 **元素 . style . 样式=“新样式”**  修改

特别记住：设置背景图片

document.body.style.backgroundImage=’url(‘ +链接+ ’)’ 绿色代表空格

1. 事先使用一个新的类名，在css中写好新样式，然后通过 **元素.className=”新类名”** 替换到新的样式组。 若想保留原样式，可使用 **“旧类名 新类名”** 的方式保留
2. 自定义属性的操作
3. 获取属性值

* element.属性 （仅可以获取**元素本身自带的属性**）
* element . getAttribute( ' 属性 ‘ ) （可以获得程序员**自定义的属性**）

1. 设置元素属性值

* element.属性=‘值’ （获取元素本身自带的属性）
* element . setAttribute( ‘ 属性 ’ , ‘ 值 ’ ) （主要针对自定义属性）

通过这种方法设置类名，**需要使用class，**不能使用className

1. 移除属性

element . removeAttribute( ‘属性’ )

1. H5的新增自定义属性方式、

规定：**data- 开头的属性为自定义属性**

* 赋值方式 1.直接在html中定义 eg. <div **data-index=”1”**> </div>

2.element.setAttribute(‘属性’，‘值’)

* 获取属性值 1. dataset是一个存放所有data-开头的自定义属性的集合

获取方式：element.dataset.index

element.dataset [ ‘index’ ] **ie11以上才兼容**

这两种方式**不需写data-开头**；对于包含-连接的属性名，使用**驼峰命名法**

1. **节点操作**
2. **节点概述**

**节点一般具有三个基本属性：nodeType节点类型 nodeName节点名称 nodeValue节点值**

**常见的节点有三类：元素节点nodeType=1；**

**属性节点nodeType=2**

**文本节点nodeType=3（包含文字 空格 换行等等）**

1. **父级节点 node.parentNode**

**获取的是与该节点最近的父节点，即亲爸爸**

**若无父节点，则返回null，如 html.parentNode=null**

1. **子节点**
2. **node.childNodes返回的是所有子节点的集合，不仅仅包括属性节点**
3. **node.children（非标准，但是推荐）返回所有子元素节点的集合**
4. **node.firstChild和node.lastChild，返回的不仅仅是元素节点，并且一般是文本节点，不推荐使用**
5. **node.firstElementChild第一个子元素节点**

**node.lastElementChild最后一个子元素节点**

**两者均有兼容性问题，仅支持ie9以上**

1. **因此，实际开发中，我们使用**

**node.children [ 0 ]以及node.children [ node.children.length-1 ]来替代**

1. **removeChild删除某个元素**
2. **兄弟节点**

**node.nextSibring下一个兄弟节点，一般是文本节点，不推荐**

**node.nextElementSibring下一个兄弟元素节点**

**此外还有node.previousSibring和node.previousElementSibring**

**带element的都是ie9以上才支持**

**解决兼容性问题：封装一个兼容性函数**

**function getNextElementSibring（element）{**

**var el=element；**

**while（el=el.nextSibring）{**

**if（el.nodeType===1）{**

**return el；**

**}**

**}**

**return null；**

**}**

* **节点操作**

1. **创建节点**

**var x = document.createElement（'元素名或者标签名’）**

1. **添加节点**

**node.appendChild ( 子节点 ) 子节点不需要使用双引号括起来**

**在node的末端添加一个子节点**

**node.insertBefore( 想要加入的元素, 指定的元素 )**

**将指定的元素之前加入一个兄弟元素节点，注意，node指这两个元素的父节点**

**注意，并没有insertAfter**

1. **控制节点**

**node.cloneNode()**

**若括号为空，或者为false，称为浅拷贝。仅拷贝本标签，而不复制其中的内容及子节点。**

**若括号内容为true，则为深度拷贝。将复制节点本身以及其内容、所有的子节点。**

* **三种动态创建元素的方法**
* **document.write() 会导致整个页面重绘，只显示write的内容**
* **element.innerHTML 将内容写入某个DOM节点，创建多个元素效率更高（应该采用数组形式拼接，而非字符串拼接），但是结构复杂**
* **createElement() 创建多个元素效率较低，但是结构清晰**
* **element . insertAdjacentHTML ( position , ‘ text ‘ ); 把字符串格式的元素添加到element中**

**position：’beforebegin’ element之前 ‘afterbegin’ element第一个子节点之前**

**‘beforeend’ 元素内部最后一个子节点之后**

**‘afterend’ 元素自身的后面**

1. **事件高级**
2. **注册事件的两种方式**

**传统方式：利用on开头的是事件，例如onclick。**

**eventTarget . onclick=function(){ }**

**特点：唯一性，如果同一个元素同一个事件被不同处理函数注册，最后的处理函数会完全覆盖之前的其他函数**

**方法监听注册方式：addEventListener( )，在ie9之前使用attachEvent()替代**

**特点：同一个元素同一个事件可以注册多个监听器，会按照注册事件的顺序依次执行**

**格式：eventTarget . addEventLisenter( type , listener[ , useCapture ] )**

**eventTarget为被绑定的目标对象，当目标对象被触发指定的事件后，就会执行事件处理函数**

**type 事件类型，如click、mouseover，注意吧on字去掉，一定要加引号**

**listener 事件处理函数，事件发生时就会调用该监听函数**

**useCapture 可选参数，默认false**

**btns.addEventListener( ‘ click ’ , function(){处理语句} )**

**了解eventTarget . attachEvent ( eventNameWithOn, callback )**

**eventNameWithOn: 事件类型字符串，记得带on 和引号**

**callback：事件处理函数，当目标触发时回调函数被调用**

1. **事件解绑**
2. **传统解绑方式 element . onclick = null;**
3. **方法监听注册方式**

* **eventTarget . removeEventListener ( type , listener[ , useCapture] );**

**使用这种方法解绑，则在注册事件时不能使用匿名函数，应该事先写好一个带名字的函数再绑定。**

**function fn ( ) { 处理语句 }**

**btn . addEventListener ( ‘click’ , fn )**

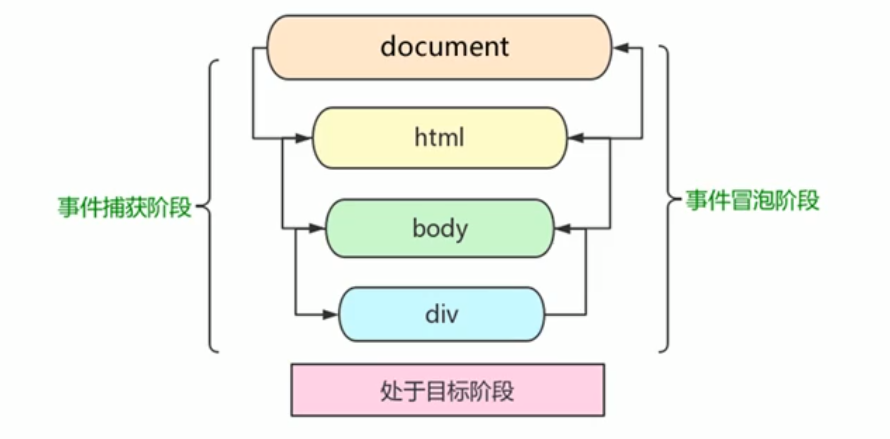
**btn . removeEventListener ( ‘click’ , fn ) 注意，调用函数不再需要添加小括号**

* **在ie9以前使用 detachEvent 代替**

1. **DOM事件流**

**事件发生时，它会在元素节点之间按照特定的顺序传播，这个传播过程就是DOM事件流**

**分为三个阶段：1.捕获阶段 2.当前目标阶段 3.冒泡阶段**



1. **js代码只能执行捕获或者冒泡中的一个阶段**
2. **onclick和attachEvent只能得到冒泡阶段**
3. **如果addEventListener中的useCapture参数为true，则处于捕获阶段**

**例如： son.addEventListeer( ‘click’, function(){ }, true )**

**默认为false，处于冒泡阶段**

**部分事件是没有冒泡的，比如onblur、onfocus、onmouseenter、onmouseleave**

1. **事件对象**
2. **div.onclick = function( event ){ } event就是一个事件对象，类似于形参**
3. **事件对象只有有了事件才会存在，是系统自动创建的，不需要我们传递参数**
4. **事件对象是一系列与事件相关的数据的集合，比如鼠标点击事件就包含了鼠标坐标等相关信息。**

**例如：div.addEventListener( ‘click’ , function( e ){ console.log(e); }，就可以输出鼠标点击对象**

1. **兼容性问题ie678只认识window.event。console.log(window.event)**
2. **事件对象常见的属性和方法**



**注意：a. e.target返回触发事件的对象，不一定指向绑定事件的对象，需要根据事件流分析。this返回绑定事件的对象，与e.currentTarget相似。**

**b. 阻止默认行为，比如不让链接跳转。e.preventDefault( )**

**e.returnValue ie678**

**直接使用return false只能适用于传统注册方法，并且会直接跳出处理函数，不再执行余下语句。都是这个方法没有兼容性问题。**

**c. 阻止冒泡 e.stopPropagation() e.cancelBubble ie678**

1. **事件委托（代理、委派）——事件冒泡的运用**

**不给每一个子节点单独添加事件监听器，而是给他们的父节点添加监听器，利用冒泡的原理影响子节点。**

**例如，点击无序列表中的某个列表项，改变列表项的颜色，其他列表项无色**

**var ul=document.querySelector(‘ul’);**

**ul.addEventListener( ‘click’ , function( e ){**

**for( var i=0, i < this.children.length, i++ )**

**{**

**this.childern[i].style.backgroundColor=’’;**

**}**

**e.target.style.backgroundColor = ‘red’;**

**} )**

1. **常用鼠标事件**
2. **常用鼠标事件**
3. **禁止右键菜单 contextmenu事件为打开右键菜单**

**使用document.addEvenListener( ‘contextvalue’ , function(e){**

**e.preventDefault();**

**} ) 可以禁用菜单**

1. **禁止鼠标选中 selectstart 鼠标开始选中**
2. **鼠标移动事件 mousemove 每移动一个像素会触发一次**
3. **鼠标事件对象 MouseEvent**



**这些值以px为单位，在使用时记得加上单位px**

1. **常用键盘事件**
2. **常用键盘事件**
3. **onkeyup 某个键盘按键被松开**
4. **onkeydown 某个键盘按键被按下，按下后会持续触发直至停止**
5. **onkeypress 某个键盘按键被按下 但是不识别功能键**
6. **同时存在时执行顺序：keydown keypress keyup**
7. **keydown keypress事件触发式，字符还未出现在文本框里；keyup事件触发时字符已经出现在文本框里**
8. **键盘事件对象**

**e. keycode返回按下键的ASCII码的值**

**keyup keydown 事件返回值不区分大小写，均视为大写**

**keypress 事件区分大小写**

1. **href=”javascript:;”**可以阻止链接跳转 [↑](#footnote-ref-1)