**javascript学习笔记**

# 1 JS和DOM的关系

(1)JS语言本身的语法。

(2)DOM对象，例如把div、p等元素作为对象。

(3)BOM对象，例如把浏览器地址栏、历史记录、DOM对象等装在一个对象里。

# 2 引入JS的方式

(1)在HTML页面内引入，用标签script标注，例如：<script>JS代码</script>。注：JS代码可以放到页面head和body。

(2)外部引入JS文件，用标签script标注，导入JS路径。例如：<script src="./test.js">注意：这里的JS代码无效。</script>

# 3 变量类型

声明变量用var关键字。

(1)数字类型

**var *a*** = 10;  
**var *b*** = 3.14;

(2)字符类型

**var *c*** = **'hello'**;  
**var *d*** = **"world"**;

(3)布尔类型

**var *e*** = **false**;

(4)undefined型

**var *f*** = **undefined**; *// 原生类型没有定义叫undefined*

(5)null型

**var *g*** = **null**; *// 对象值为空叫null*

(6)数组

**var *h*** = [**'hello'**, 10, **true**]; *// 键始终是数字0、1、2……递增。*

获取数组单元h[1];

(7)对象

**var *obj*** = {**name**: **'zhangsan'**, **gender**: 男, **age**: 28};

获取对象单元obj.name或者obj[name](常用在遍历上)。

注意：var a=10和a=10的区别，var是声明，没有var是赋值语句，如果变量在局部没有声明就赋值，就会向上一级查找变量，如果没有找到，会一直找到Windows全局。

# 4 运算符

+ - \* / > < == % || &&等运算符。

(1)拼接运算符：+

规则：碰到非法数字后，后面一律为拼接下去。

|  |
| --- |
| **var *a*** = 1; **var *b*** = 2; **var *c*** = 3; **var *d*** = 4; **console**.log(**'hello'** + ***a*** + ***b***); *// hello12* **console**.log(***a*** + ***b*** + **'hello'**); *// 3hello* **console**.log(***a*** + ***b*** + **'hello'** + ***c*** + ***d***); *// 3hello34* |

(2)逻辑运算：或|| 与&&

规则：返回最早能判断整体结果的值。

|  |
| --- |
| **var *a*** = **false**; **var *b*** = 10; **var *c*** = **true**; **console**.log(***a*** || ***b***); *// 返回b = 10* **console**.log(***a*** && ***b***); *// 返回a = false* **console**.log(***a*** || ***b*** || ***c***); *// 返回b = 10;* |

# 5 控制结构

条件控制if 、else if、else

选择控制switch case

循环控制for、for in、while

|  |
| --- |
| **var *arr*** = [**'刘备'**, **'关羽'**, **'张飞'**, **'赵云'**, **'黄忠'**]; **for** (**var *i*** = 0; ***i*** < ***arr***.**length**; ***i***++) { *// 遍历数组* **console**.log(***arr***[***i***]); }  **var *obj*** = {**name**: **'zhangsan'**, **gender**: **'男'**, **age**: 28}; **for** (**var *k* in *obj***) { *// 遍历对象* **console**.log(***k***, ***obj***[***k***]); } |

# 6 对象操作

JS很多原生态的变量都理解为对象，对象就有其属性和方法。

## 6.1 String(字符串对象)

**属性**:  
 length ：字符串中的字符数。  
  
**方法**:

|  |
| --- |
| **charAt( )**  *// 抽取字符串中指定位置处的字符。* **charCodeAt( )***// 返回字符串中指定位置处的字符编码。* **concat( )** *// 把一个或多个值连接到字符串上。* **indexOf( )** *// 在字符串中检索一个字符或一个子串。* **lastIndexOf( )** *// 在字符串中向后检索一个字符或一个子串。* **match( )** *// 用正则表达式执行模式匹配。* **replace( )** *// 用正则表达式执行查找、替换操作。* **search( )** *// 检索字符串中与正则表达式匹配的子串。* **slice( )** *// 返回字符串的一个片段或一个子串。* **split( )** *// 把字符串分割成一个字符串数组，在指定的分界字符处或正则表 达式处执行分割。* **substring( )** *// 从字符串中抽取一个子串。* **substr( )** *// 从字符串中抽取一个子串。该方法是substring()的一个变体。* **toLowerCase( )** *// 将字符串中的所有字符都转换成小写的，然后返回一个副本。* **toString( )** *// 返回原始的字符串值。* **toUpperCase( )** *// 将字符串中的所有字符都转换成大写的，然后返回一个副本。* **valueOf( )** *// 返回原始字符串值。* |

## 6.2 Array（数组对象）

**属性**：  
 length: 数组中元素的个数。**方法**：

|  |
| --- |
| **concat( )** *// 给数组添加元素。* **join( )** *// 将数组中所有元素都转换为字符串，然后连接在一起。* **pop( )** *// 从数组尾部删除一个项目。* **push( )** *// 把一个项目添加到数组的尾部。* **reverse( )**  *// 在原数组上颠倒数组中元素的顺序。* **shift( )** *// 将数组的头部元素移出数组头部。* **slice( )** *// 返回一个数组的子数组。* **sort( )** *// 在原数组上对数组元素进行排序。* **splice( )**  *// 插入、删除或替换一个数组元素。* **toLocaleString( )**  *// 把数组转换为一个局部字符串。* **toString( )**  *// 把数组转换为字符串。* **unshift( )**  *// 在数组的头部插入一个元素。* |

# 7 浏览器对象

window对象是浏览器宿主对象，和JS无关，但可以在JS调用。

## 7.1 window对象的方法

### 7.1.1Window 对象属性

|  |  |
| --- | --- |
| **属性** | **描述** |
| [**closed**](http://www.runoob.com/jsref/prop-win-closed.html) | 返回窗口是否已被关闭。 |
| [**defaultStatus**](http://www.runoob.com/jsref/prop-win-defaultstatus.html) | 设置或返回窗口状态栏中的默认文本。 |
| [**document**](http://www.runoob.com/jsref/dom-obj-document.html) | 对 Document 对象的只读引用。(请参阅[对象](http://www.runoob.com/jsref/dom-obj-document.html)) |
| [**frames**](http://www.runoob.com/jsref/prop-win-frames.html) | 返回窗口中所有命名的框架。该集合是 Window 对象的数组，每个 Window 对象在窗口中含有一个框架。 |
| [**history**](http://www.runoob.com/jsref/obj-history.html) | 对 History 对象的只读引用。请参数 [History 对象](http://www.runoob.com/jsref/obj-history.html)。 |
| [**innerHeight**](http://www.runoob.com/jsref/prop-win-innerheight.html) | 返回窗口的文档显示区的高度。 |
| [**innerWidth**](http://www.runoob.com/jsref/prop-win-innerheight.html) | 返回窗口的文档显示区的宽度。 |
| [**length**](http://www.runoob.com/jsref/prop-win-length.html) | 设置或返回窗口中的框架数量。 |
| [**location**](http://www.runoob.com/jsref/obj-location.html) | 用于窗口或框架的 Location 对象。请参阅 [Location 对象](http://www.runoob.com/jsref/obj-location.html)。 |
| [**name**](http://www.runoob.com/jsref/prop-win-name.html) | 设置或返回窗口的名称。 |
| [**navigator**](http://www.runoob.com/jsref/obj-navigator.html) | 对 Navigator 对象的只读引用。请参数 [Navigator 对象](http://www.runoob.com/jsref/obj-navigator.html)。 |
| [**opener**](http://www.runoob.com/jsref/prop-win-opener.html) | 返回对创建此窗口的窗口的引用。 |
| [**outerHeight**](http://www.runoob.com/jsref/prop-win-outerheight.html) | 返回窗口的外部高度，包含工具条与滚动条。 |
| [**outerWidth**](http://www.runoob.com/jsref/prop-win-outerheight.html) | 返回窗口的外部宽度，包含工具条与滚动条。 |
| [**pageXOffset**](http://www.runoob.com/jsref/prop-win-pagexoffset.html) | 设置或返回当前页面相对于窗口显示区左上角的 X 位置。 |
| [**pageYOffset**](http://www.runoob.com/jsref/prop-win-pagexoffset.html) | 设置或返回当前页面相对于窗口显示区左上角的 Y 位置。 |
| [**parent**](http://www.runoob.com/jsref/prop-win-parent.html) | 返回父窗口。 |
| [**screen**](http://www.runoob.com/jsref/obj-screen.html) | 对 Screen 对象的只读引用。请参数 [Screen 对象](http://www.runoob.com/jsref/obj-screen.html)。 |
| [**screenLeft**](http://www.runoob.com/jsref/prop-win-screenleft.html) | 返回相对于屏幕窗口的x坐标 |
| [**screenTop**](http://www.runoob.com/jsref/prop-win-screenleft.html) | 返回相对于屏幕窗口的y坐标 |
| [**screenX**](http://www.runoob.com/jsref/prop-win-screenx.html) | 返回相对于屏幕窗口的x坐标 |
| [**screenY**](http://www.runoob.com/jsref/prop-win-screenx.html) | 返回相对于屏幕窗口的y坐标 |
| [**self**](http://www.runoob.com/jsref/prop-win-self.html) | 返回对当前窗口的引用。等价于 Window 属性。 |
| [**status**](http://www.runoob.com/jsref/prop-win-status.html) | 设置窗口状态栏的文本。 |
| [**top**](http://www.runoob.com/jsref/prop-win-top.html) | 返回最顶层的父窗口。 |

### 7.1.2 Window 对象方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [**alert()**](http://www.runoob.com/jsref/met-win-alert.html) | 显示带有一段消息和一个确认按钮的警告框。 |
| [**blur()**](http://www.runoob.com/jsref/met-win-blur.html) | 把键盘焦点从顶层窗口移开。 |
| [**clearInterval()**](http://www.runoob.com/jsref/met-win-clearinterval.html) | 取消由 setInterval() 设置的 timeout。 |
| [**clearTimeout()**](http://www.runoob.com/jsref/met-win-cleartimeout.html) | 取消由 setTimeout() 方法设置的 timeout。 |
| [**close()**](http://www.runoob.com/jsref/met-win-close.html) | 关闭浏览器窗口。 |
| [**confirm()**](http://www.runoob.com/jsref/met-win-confirm.html) | 显示带有一段消息以及确认按钮和取消按钮的对话框。 |
| [**createPopup()**](http://www.runoob.com/jsref/met-win-createpopup.html) | 创建一个 pop-up 窗口。 |
| [**focus()**](http://www.runoob.com/jsref/met-win-focus.html) | 把键盘焦点给予一个窗口。 |
| [**moveBy()**](http://www.runoob.com/jsref/met-win-moveby.html) | 可相对窗口的当前坐标把它移动指定的像素。 |
| [**moveTo()**](http://www.runoob.com/jsref/met-win-moveto.html) | 把窗口的左上角移动到一个指定的坐标。 |
| [**open()**](http://www.runoob.com/jsref/met-win-open.html) | 打开一个新的浏览器窗口或查找一个已命名的窗口。 |
| [**print()**](http://www.runoob.com/jsref/met-win-print.html) | 打印当前窗口的内容。 |
| [**prompt()**](http://www.runoob.com/jsref/met-win-prompt.html) | 显示可提示用户输入的对话框。 |
| [**resizeBy()**](http://www.runoob.com/jsref/met-win-resizeby.html) | 按照指定的像素调整窗口的大小。 |
| [**resizeTo()**](http://www.runoob.com/jsref/met-win-resizeto.html) | 把窗口的大小调整到指定的宽度和高度。 |
| **scroll()** |  |
| [**scrollBy()**](http://www.runoob.com/jsref/met-win-scrollby.html) | 按照指定的像素值来滚动内容。 |
| [**scrollTo()**](http://www.runoob.com/jsref/met-win-scrollto.html) | 把内容滚动到指定的坐标。 |
| [**setInterval()**](http://www.runoob.com/jsref/met-win-setinterval.html) | 按照指定的周期（以毫秒计）来调用函数或计算表达式。 |
| [**setTimeout()**](http://www.runoob.com/jsref/met-win-settimeout.html) | 在指定的毫秒数后调用函数或计算表达式。 |

## 7.2 window对象的子对象

### navigator浏览器信息对象

|  |
| --- |
| **appCodeName**; *// 内部代码* **appName**; *// 浏览器名称* **appVersion**; *// 浏览器版本* **platform**; *// 操作系统类型* **userAgent**; *// 用户代理信息* **cookieEnabled**; *// 是否支持cookie* |

### location地址栏对象

|  |
| --- |
| **host**; *// 主机* **port**; *// 端口* **href**; *// 地址，页面跳转* **pathname**; *// 路径* **protocol**; *// 协议* search**()**; *// 查询字符串* assign(url); *// 页面跳转* |

### history历史记录对象

|  |
| --- |
| **window**.**history**.**length**; *// 历史记录数据* **window**.**history**.forward(); *// 向前* **window**.**history**.back(); *// 后退* **window**.**history**.go(); *// 指定调到历史的哪一个页面* |

# 8 查找DOM对象方式

### 用id查找，返回值是对象

**document**.getElementById(**"test"**);

### 按类名查找，返回值是对象集合

var ts = **document**.getElementsByClassName(**"ts"**);

ts[0]表示第一个name为ts的类

### 用标签查找，返回值是对象集合

var dv = **document**.getElementsByTagName(**"div"**);

dv[0]表示第一个标签div

### 对于表单元素，按属性name查询，返回值是对象集合

var usr = **document**.getElementsByName(**"username"**);

usr[0]表示第一个名称username

### 按子节点查询，返回值是对象集合

**document**.getElementById(**"test1"**).**childNodes**;

**document**.getElementById(**"usr"**).**firstChild**; *// 第一个子节点***document**.getElementById(**"usr"**).**lastChild**; *// 最后一个子节点*

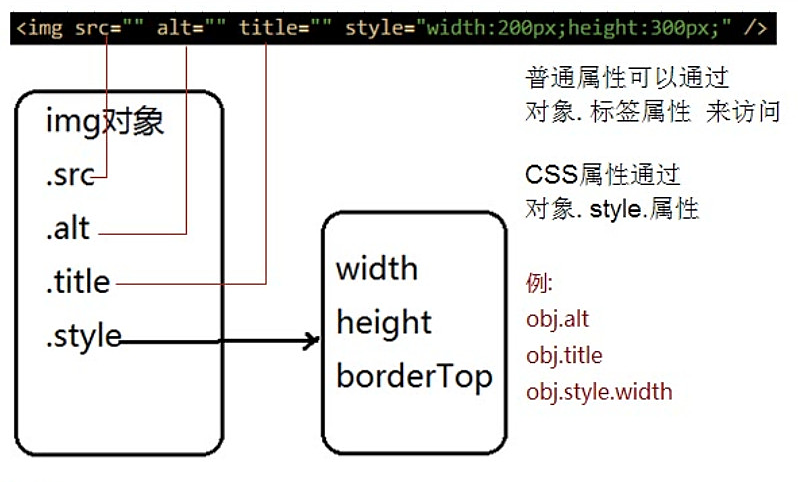
### 按父节点查询，返回对象

**document**.getElementById(**"test1"**).**parentNode**;*// 节点*

### 按兄弟节点查询，返回对象

**document**.getElementById(**"usr"**).**nextSibling**;*// 下一个兄弟节点***document**.getElementById(**"usr"**).**previousSibling**;*// 上一个兄弟节点*

# 9 操作DOM对象的属性



标签的属性和DOM对象的属性绝大多数情况是相同的。

例如：<**img** id=**"bg"** src=**"./test1.jpg"** alt=**""** title=**""**>

操作src属性：**document**.getElementById(**"bg"**).**src**=**"./test2.jpg"**;

但有一个例外：标签的属性是class，而DOM对象的属性是className。

<**div** class=**"testClass"**></**div**>

操作class属性: **document**.getElementsByTagName(**"div"**).**className**=**"test2"**;

用DOM对象操作标签的CSS属性

html: <**div** id=**"test"** ></**div**>

js:

|  |
| --- |
| **var *div*** =**document**.getElementById(**"test"**); ***div***.**style**.**width**=**"400px"**; ***div***.**style**.**height**=**"200px"**; ***div***.**style**.**border**=**"5px solid red"**; ***div***.**style**.**backgroundColor** = **"blue"**; ***div***.**style**.**margin**=**"0 auto"**; ***div***.**style**.**paddingTop** = **"10px"**; |

**注意：**如果标签的CSS属性名称带有横线的，例如border-top-style，对应的DOM对象操作名称是把横线去掉并且横线后的首字母大写，例如borderTopStyle。

## 获取对象在内存中的样式

|  |
| --- |
| *// html：*  <**div** id=**"test"**></**div**>  *// CSS：*  <**style**>  **#test**{  width: 400**px**;  height: 200**px**;  } </**style**>  *// JS：*  <**script**>  **var *div*** = **document**.getElementById(**"test"**);  **console**.log(**"方式一："**, ***div***.**style**.**width**, ***div***.**style**.**height**);*// 这种方式只能获取内联CSS的style值，当html和CSS分开后无法获取其内存的值* **function** *getStyle*(el, attr) { *// 兼容性封装，注意这种方式获取的对象是只读，要想更改还是需要用obj.style.属性=值方式。* **return** el.**currentStyle** ? el.**currentStyle**[attr] : getComputedStyle(el, **null**)[attr];  }  **console**.log(**"方式二："**, *getStyle*(***div***, **"width"**), *getStyle*(***div***, **"height"**));*// 结果：400px 200px* </**script**> |

# 10 创建和删除DOM对象

创建对象步骤：

1. 创建对象。
2. 找到父对象parentObj。
3. 让父对象添加，parentObj. appendChild ()。

注这种方式添加对象比较麻烦，不建议使用，一般使用下面的添加对象方式：

直接把父对象下面的内容读出来，读出来内容可以修改，然后把添加的对象插入或追加， parentObj.innerHTML +=添加的对象

删除对象步骤：

1. 找到对象。
2. 找到父对象parentObj。
3. 让父对象删除，parentObj.removedChild()。

|  |
| --- |
| <!DOCTYPE html> <**html** lang=**"en"**> <**head**>  <**meta** charset=**"UTF-8"**>  <**title**>测试</**title**>  <**script**>  **function** *addLi*() { *// 添加DOM对象* **var** ul = **document**.getElementsByTagName(**"ul"**)[0];  ul.**innerHTML** += **"<li>东</li> <li>南</li> <li>西</li> <li>北</li>"**;  }   **function** *delLi*() { *// 删除DOM对象* **var** li = **document**.getElementsByTagName(**"li"**);  **if** (li.**length** < 1) {  alert(**"li已经删除完毕。"**)  **return**;  }  **var** lastLi = li[li.**length** - 1];  lastLi.parentNode.removeChild(lastLi);  }  </**script**> </**head**> <**body**>  <**button** onclick=**"***addLi*()**"**>添加li</**button**> <**button** onclick=**"***delLi*()**"**>删除最后一个li</**button**> <**ul**>  <**li**>东</**li**>  <**li**>南</**li**>  <**li**>西</**li**>  <**li**>北</**li**> </**ul**>  </**body**> </**html**>  <**body**>  <**button** onclick=**"***addLi*()**"**>添加li</**button**> <**button** onclick=**"***delLi*()**"**>删除最后一个li</**button**> <**ul**>  <**li**>东</**li**>  <**li**>南</**li**>  <**li**>西</**li**>  <**li**>北</**li**> </**ul**>  </**body**> </**html**> |

# 11 事件

事件对象包含事件相关信息，例如鼠标、事件、触发DOM对象等。

事件对象被系统传递给事件函数的第一个参数。

事件对象的属性在低版本IE、w3c略有不同。

一个重要的事件属性：target，srcElement(低版本IE)，代表事件发生的所在对象。

## 11.1常用事件方法

**onclick** 当用户点击某个对象时调用的事件句柄。

**onmouseover** 鼠标移到某元素之上。

**onfocus** 元素获得焦点。

**onblur** 元素失去焦点。

**onload** 一张页面或一幅图像完成加载。

**onchange** 在属性值改变时还必须使得当前元素失去焦点(onblur)才可以激活该事件。

**onsubmit** 确认按钮被点击。

**onmousedown** 鼠标按钮被按下。

|  |
| --- |
| <!DOCTYPE html> <**html** lang=**"en"**> <**head**>  <**meta** charset=**"UTF-8"**>  <**title**>测试</**title**>  <**script**>  **window**.onload = **function** () {  **var** usr = **document**.getElementsByName(**'usr'**)[0];  **var** email = **document**.getElementsByName(**'email'**)[0];  usr.onfocus = **function** () { *// 获取焦点事件* **this**.**style**.**backgroundColor** = **"#11ee22"**;  **this**.**style**.**border** = **''**; *// 去掉边框* }  usr.onblur = **function** () { *// 失去焦点事件* **this**.**style**.**backgroundColor** = **""**;  }  usr.onchange = **function** () { *// 改变内容事件，在属性值改变时还必须使得当前元素失去焦点(onblur)才可以激活该事件* **this**.**style**.**border** = **'3px solid green'**;  }   **var** image = **document**.getElementsByTagName(**'img'**)[0];  image.onmouseover = **function** () { *// 鼠标悬浮标签事件* **this**.**style**.**border** = **'5px solid red'**;  }  image.onclick = **function** () { *// 点击事件* **this**.**style**.**border** = **'5px solid blue'**;  }   *// 表单验证* **document**.getElementsByTagName(**'form'**)[0].onsubmit = **function** () {  **var** verify = /^[A-Za-z0-9]{6,12}$/;*// 正则表达式*  **if** (!verify.test(usr.**value**)) {  alert(**'请输入6到12位字母和数字用户名。'**)  **return false**;  }  verify = /^\w+@\w+(\.\w+)+$/;*// 正则表达式*  **if** (!verify.test(email.**value**)) {  alert(**'请输入正确的email。'**)  **return false**;  }  }  }   **function** *mousedown*(ev) { *// 打印当前点击的标签和位置* **var** ev = **window**.**event** || ev;  **switch** (ev.**button**) {  **case** 0:  **console**.log(**'左键点击了对象'** + ev.**target**.**tagName**, **'('** + ev.**clientX**, ev.**clientY** + **')'**, **'('** + ev.**screenX**, ev.**screenY** + **')'**);  **break**;  **case** 1:  **console**.log(**'中键点击了对象'** + ev.**target**.**tagName**);  **break**;  **case** 2:  **console**.log(**'右键点击了对象'** + ev.**target**.**tagName**);  **break**;  }  }  </**script**> </**head**>  <**body** onmousedown=**"***mousedown*()**"**> <**form** action=**""**>  <**p**>用户<**input** type=**"text"** name=**"usr"** value=**""**></**p**>  <**p**>邮件<**input** type=**"text"** name=**"email"** value=**""**></**p**>  <**button**>提交</**button**> </**form**> <**br**> <**img** src=**"./bg.jpg"** alt=**""**> </**body**> </**html**> |

注：var regular=/\w+@\w+(\.\w+)+/ 如果把正则用来匹配，记得把起始符^和结束符$去掉，然后判断regular.exec(value)!==null

## 11.2 事件委托

下面是10\*10单元格的CSS和JS，子元素事件委托父元素事件来实现，提高效率。

|  |
| --- |
| <**style**>  **table** {  width: 500**px**;  height: 500**px**;  border-collapse: **collapse**;  border-spacing: 0**px**;  margin: 50**px auto**;  }  **td** {  border: 1**px solid darkgrey**;  } </**style**>  <**script**>  **window**.onload = **function** () {  **var** cnt = 0;  **document**.getElementsByTagName(**'table'**)[0].onclick = **function** (ev) {  cnt++ % 2 ? ev.**target**.**style**.**backgroundColor** = **'green'** : ev.**target**.**style**.**backgroundColor** = **'blue'**;  }  } </**script**> |