# 第六章 Javascrip中

本节所讲内容：

6.1 JavaScript对象

6.2 JavaScript流程控制语句

6.3 JavaScript函数

## 6.4 变量作用域

6.5 JavaScript BOM

6.6 JavaScript DOM

6.1 JavaScript对象

JavaScript 中的所有事物都是对象：字符串、数值、数组、函数...此外，JavaScript 允许自定义对象。

1.使用new运算符创建Object

var people= new Object(); //new方式

people.name='For'; //创建属性字段

people.age=18; //创建属性字段

2.new关键字可以省略

var people = Object(); //省略了new关键字

people.name='For';

people.age=18;

3.使用字面量方式创建Object

var people = { //字面量方式

name : 'for', //创建属性字段

age : 28 //最后一个属性不要加逗号

};

4.使用字面量及传统复制方式

var people ={};

people.name='for'; //字面量声明为空对象

people.age=18; //点符号给属性复制

5.两种属性输出方式

alert(people.age); //点表示法输出

alert(people ['age']); //中括号表示法输出，注意引号

6.给对象创建方法

var people ={

run:function(){

return '跑步中！！！'; //对象中的方法

}

}

alert(people.run()); //调用对象中的方法

7.使用delete删除对象属性

在实际开发过程中，一般我们更加喜欢字面量的声明方式。因为它清晰，语法代码少，而且还给人一种封装的感觉。

6.2 JavaScript流程控制语句

6.2.1.条件分支语句if () {} else {}

if (条件表达式) {语句;} else if (条件表达式) {语句;} ... else {语句;}

<script>  
 var *box* = 100;  
 if (*box* >= 100) { *//如果满足条件，不会执行下面任何分支  
 alert*('甲');  
 } else if (*box* >= 90) {  
 *alert*('乙');  
 } else if (*box* >= 80) {  
 *alert*('丙');  
 } else if (*box* >= 70) {  
 *alert*('丁');  
 } else if (*box* >= 60) {  
 *alert*('及格');  
 } else { *//如果以上都不满足，则输出不及格  
 alert*('不及格');  
 }  
</script>

6.2.2.多重分支语句switch () { case n : ...};

switch语句是多重条件判断，用于多个值相等的比较。

<script>  
 var *box* = 1;  
 switch (*box*) { *//用于判断box相等的多个值* case 1 :  
 *alert*('one');  
 break; *//break;用于防止语句的穿透* case 2 :  
 *alert*('two');  
 break;  
 case 3 :  
 *alert*('three');  
 break;  
  
 default : *//相当于if语句里的else，否则的意思  
 alert*('error');  
 }  
</script>

6.2.3.do...while语句

do...while语句是一种先运行，后判断的循环语句。也就是说，不管条件是否满足，至少先运行一次循环体。

<script>  
 var *box* = 1; *//如果是1，执行五次，如果是10，执行1次* do {  
 *alert*(*box*);  
 *box*++;  
 } while (*box* <= 5); *//先运行一次，再判断*</script>

6.2.4. while语句

while语句是一种先判断，后运行的循环语句。也就是说，必须满足条件了之后，方可运行循环体。

<script>  
 var *box* = 1; *//如果是1，执行五次，如果是10，不执行* while (*box* <= 5) { *//先判断，再执行  
 alert*(*box*);  
 *box*++;  
 }  
</script>

6.2.5. for语句

for语句也是一种先判断，后运行的循环语句。但它具有在执行循环之前初始变量和定义循环后要执行代码的能力。

<script>  
 for (var *box* = 1; *box* <= 5 ; *box*++) { *//第一步，声明变量var box = 1;  
 alert*(*box*); *//第二步，判断box <=5* } *//第三步，alert(box)  
 //第四步，box++  
 //第五步，从第二步再来，直到判断为false*</script>

6.2.6. for...in语句

for...in语句是一种精准的迭代语句，可以用来枚举对象的属性。

<script>  
 var *box* = { *//创建一个对象* 'name' : 'For Teacher’, //键值对，左边是属性名，右边是值  
 'age' : 28,  
 'height' : 178  
 };  
 for (var *p* in *box*) { *//列举出对象的所有属性  
 alert*(*p*);  
 }  
</script>

break和continue语句用于在循环中精确地控制代码的执行。其中，break语句会立即退出循环，强制继续执行循环体后面的语句。而continue语句退出当前循环，继续后面的循环。

<script>  
 for (var *box* = 1; *box* <= 10; *box*++) {  
 if (*box* == 5) break; *//如果box是5，就退出循环  
 document*.write(*box*);  
 *document*.write('<br />');  
 }  
  
</script>

<script>  
 for (var *box* = 1; *box* <= 10; *box*++) {  
 if (*box* == 5) continue; *//如果box是5，就退出当前循环  
 document*.write(*box*);  
 *document*.write('<br />');  
 }  
  
</script>

6.3 JavaScript函数

函数是由事件驱动的或者当它被调用时执行的可重复使用的代码块。

1.函数是完成某一功能的代码段

2.函数是可重复执行的代码段

3.函数方便管理和维护

语法格式：

Function 函数名称（[参数][ 参数][ 参数]）{

代码段

Return 返回值

}

代码示例：

function box(name, age) { //带参数的函数

alert('你的姓名：'+name+'，年龄：'+age);

}

box('For老师',28); //调用函数，并传参

函数名称的规范：

1.函数名称不要包含特殊字符

2.函数名称最好遵循驼峰式的命名方法

3.函数名称严格区分大小写

4.函数名称如果重复会覆盖

5.函数可以有参数也可以没有参数，可以有一个参数也可以有多个参数

6.函数通过return返回值，如果没有默认返回undefined

7.函数不调用不执行

<!DOCTYPE html>  
<html>  
<for>  
 <meta charset="utf-8">  
 <title>js 函数</title>  
 <script>  
 function *PeopleNameAge*(name,age){  
 *alert*('你的姓名:'+name+'你的年龄:'+age);  
 }  
 *//box('For',18);  
 //box('teacher',20);  
 //驼峰式PeopleNameAge 每个单词的首字母大写  
 //小驼峰Peoplenameage 单词的首字母大写* function *People*(){  
 *alert*(10);  
 }  
  
 function *People*(){  
 *alert*(20);  
 }  
 *//People();* function *aza*(a,b){  
  
 return(a+b);  
 }  
 *alert*(*aza*(2,3));  
 </script>  
</for>  
<body>  
  
</body>  
</html>

## 6.4 变量作用域

变量作用域指的是变量的作用范围，javascript中的变量分为全局变量和局部变量。

1、全局变量：在函数之外定义的变量，为整个页面公用，函数内部外部都可以访问。

2、局部变量：在函数内部定义的变量，只能在定义该变量的函数内部访问，外部无法访问。

<script type="text/javascript">

//全局变量

var a = 12;

function myalert()

{

//局部变量

var b = 23;

alert(a);

alert(b);

}

myalert(); //弹出12和23

alert(a); //弹出12

alert(b); //出错

</script>

6.5 JavaScript BOM

浏览器对象模型 (BOM)浏览器对象模型（Browser Object Model (BOM)）尚无正式标准。

Browser [ˈbraʊzə(r)]：浏览器

由于现代浏览器已经（几乎）实现了 JavaScript 交互性方面的相同方法和属性，因此常被认为是 BOM 的方法和属性。

1.警告框

警告框经常用于确保用户可以得到某些信息。

当警告框出现后，用户需要点击确定按钮才能继续进行操作。

Alert(‘aaa’);

2.确认框

确认框通常用于验证是否接受用户操作。

当确认卡弹出时，用户可以点击 "确认" 或者 "取消" 来确定用户操作。

当你点击 "确认", 确认框返回 true， 如果点击 "取消", 确认框返回 false。

<script>  
 if (*confirm*('你喜欢我吗？')){  
 *alert*('喜欢！')  
 } else {  
 *alert*('不喜欢！')  
 }  
</script>

3.提示框

提示框经常用于提示用户在进入页面前输入某个值。

当提示框出现后，用户需要输入某个值，然后点击确认或取消按钮才能继续操纵。

如果用户点击确认，那么返回值为输入的值。如果用户点击取消，那么返回值为 null。

<script>  
*// 相当于python中的input*var *name* = *prompt*('请输入你的姓名:');  
 if (*name*){  
 *alert*('欢迎你'+ *name*);  
 }  
</script>

通过使用 JavaScript，我们有能力作到在一个设定的时间间隔之后来执行代码，而不是在函数被调用后立即执行。我们称之为计时事件。

在 JavaScritp 中使用计时事件是很容易的，两个关键方法是:

setInterval() - 间隔指定的毫秒数不停地执行指定的代码。

setTimeout() - 暂停指定的毫秒数后执行指定的代码

clearTimeout() - 关闭只执行一次的定时器;

clearInterval() - 关闭反复执行的定时器;

Note: setInterval() 和 setTimeout() 是 HTML DOM Window对象的两个方法。

setInterval() 方法

setInterval() 间隔指定的毫秒数不停地执行指定的代码

语法

window.setInterval("javascript function",milliseconds);

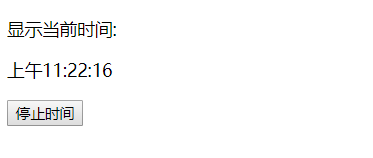
window.setInterval() 方法可以不使用window前缀，直接使用函数setInterval()。

setInterval() 第一个参数是函数（function）。第二个参数间隔的毫秒数

注意: 1000 毫秒是一秒。

<script>  
 var *myVar* = *setInterval*(*myTimer*, 1000);  
 function *myTimer*() {  
 var d = new *Date*();  
 var t = d.toLocaleTimeString(); *//获取当前字符串时间  
 document*.getElementById("demo").innerHTML = t;*//定位标签插入数据* }  
 function *myStopFunction*() {  
 *clearInterval*(*myVar*);  
 };  
</script>

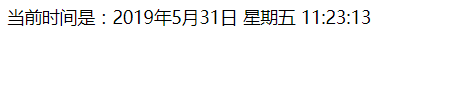
运行结果展示：



时间展示：

<div id="div1"></div>  
<script type="text/javascript">  
  
 var *oDiv* = *document*.getElementById('div1');  
 function *timego*(){  
 var now = new *Date*();*//获取当前时间* var year = now.getFullYear();*//获取当前年份* var month = now.getMonth()+1;*//获取当前月份* var date = now.getDate();*//获取当前的日期* var week = now.getDay();*//获取星期* var hour = now.getHours();*//获取时间* var minute = now.getMinutes();*//获取分钟* var second = now.getSeconds();*//获取秒* var str = '当前时间是：'+ year + '年'+month+'月'+date+'日 '+*toweek*(week)+' '+*todou*(hour)+':'+*todou*(minute)+':'+*todou*(second);  
 *oDiv*.innerHTML = str;*//在当前的div标签中插入html文本* }  
 *// timego();  
 setInterval*(*timego*,1000);  
  
 function *toweek*(n){  
 if(n==0)  
 {  
 return '星期日';  
 }  
 else if(n==1)  
 {  
 return '星期一';  
 }  
 else if(n==2)  
 {  
 return '星期二';  
 }  
 else if(n==3)  
 {  
 return '星期三';  
 }  
 else if(n==4)  
 {  
 return '星期四';  
 }  
 else if(n==5)  
 {  
 return '星期五';  
 }  
 else  
 {  
 return '星期六';  
 }  
 }  
  
 function *todou*(n){  
 if(n<10)  
 {  
 return '0'+n;  
 }  
 else  
 {  
 return n;  
 }  
 }  
</script>

运行结果展示：



制作计时器

<form>  
 <input type="button" value="开始计时！" onClick="*timedCount*()">  
 <input type="text" id="txt">  
 <input type="button" value="停止计时！" onClick="*stopCount*()">  
</form>  
  
<script type="text/javascript">  
 var *c*=0;  
 var *t*;  
 function *timedCount*()  
 {  
 *document*.getElementById('txt').value=*c*;  
 *c*=*c*+1;  
 *t*=*setTimeout*("*timedCount*()",1000)  
 }  
 function *stopCount*()  
 {  
 *clearTimeout*(*t*)  
 }  
</script>

运行结果如下：



当然上面都是没加样式的计时器的操作：详情请见test.html



详情请看：timedemo.zip中的代码



6.6 JavaScript DOM

DOM（Document Object Model）即文档对象模型，针对HTML和XML文档的API（应用程序接口）。DOM描绘了一个层次化的节点树，运行开发人员添加、移除和修改页面的某一部分。

DOM中的三个字母，D（文档）可以理解为整个Web加载的网页文档；O（对象）可以理解为类似window对象之类的东西，可以调用属性和方法，这里我们说的是document对象；M（模型）可以理解为网页文档的树型结构。

W3C提供了比较方便简单的定位节点的方法和属性，以便我们快速的对节点进行操作。分别为：getElementById()、getElementsByTagName()、getElementsByName()、getAttribute()、setAttribute()和removeAttribute()。

元素节点方法：

|  |  |
| --- | --- |
| 方法 | 说明 |
| getElementById() | 获取特定ID元素的节点 |
| getElementsByTagName() | 获取相同元素的节点列表 |
| getElementsByName() | 获取相同名称的节点列表 |
| getAttribute() | 获取特定元素节点属性的值 |
| setAttribute() | 设置特定元素节点属性的值 |
| removeAttribute() | 移除特定元素节点属性 |

1. getElementById()方法

getElementById()方法，接受一个参数：获取元素的ID。如果找到相应的元素则返回该元素的HTMLDivElement对象，如果不存在，则返回null。

document.getElementById('box'); //获取id为box的元素节点

PS：上面的例子，默认情况返回null，这无关是否存在id="box"的标签，而是执行顺序问题。解决方法，1.把script调用标签移到html末尾即可；2.使用onload事件来处理JS，等待html加载完毕再加载onload事件里的JS。

window.onload = function () { //预加载html后执行

document.getElementById('box');

};

2.getElementsByTagName()方法

getElementsByTagName()方法将返回一个对象数组HTMLCollection(NodeList)，这个数组保存着所有相同元素名的节点列表。

document.getElementsByTagName('\*'); //获取所有元素

document.getElementsByTagName('li'); //获取所有li元素，返回数组

3.getElementsByName()方法

getElementsByName()方法可以获取相同名称(name)的元素，返回一个对象数组HTMLCollection(NodeList)。

document.getElementsByName('add') //获取input元素

4.getAttribute()方法

getAttribute()方法将获取元素中某个属性的值。它和直接使用.属性获取属性值的方法有一定区别。

document.getElementById('box').getAttribute('id');//获取元素的id值

5. setAttribute()方法

setAttribute()方法将设置元素中某个属性和值。它需要接受两个参数：属性名和值。如果属性本身已存在，那么就会被覆盖。

document.getElementById('box').setAttribute('align','center');//设置属性和值

6.removeAttribute()方法

removeAttribute()可以移除HTML属性。

document.getElementById('box').removeAttribute('style');//移除属性

总结：

6.1 JavaScript对象

6.2 JavaScript流程控制语句

6.3 JavaScript函数

## 6.4 变量作用域

6.5 JavaScript BOM

6.6 JavaScript DOM