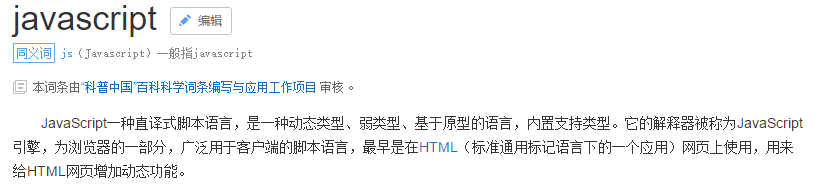
JAVASCRIPT

# Javascript概述



## 什么是javascript

JavaScript是因特网上最流行的脚本语言。

\*脚本语言不能单独使用，必须嵌入到其他语言中组合使用

JavaScript不能单独使用，必须和其他语言（HTML）结合使用

\***JavaScript由浏览器解释执行（解释一条，执行一条。故写代码时能用一条语句就不要用两条语句）**

**JavaScript代码：必须效率和可读性**

**JavaScript代码遇到错误，解析引擎停止**

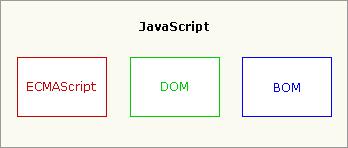
## 什么作用

* 可以控制前端页面的逻辑操作书写规范
* 可以控制CSS的样式；
* JS可以对表单项进行校验（重点）
* JS可以对HTML元素进行动态控制（非重点）

**\*特点：**

* + **安全性**（**没有访问系统文件权限**，无法用来做木马病毒）
  + 跨平台性（浏览器有JS的解析器，只要有浏览器就能运行JS代码，和平台无关）
  + 交互性（JavaScript控制HTML元素）

## 组成部分



**[ECMAScript](http://baike.baidu.com/item/ECMAScript" \t "http://baike.baidu.com/_blank)**，描述了该javascript组成言的语法和基本[对象](http://baike.baidu.com/item/%E5%AF%B9%E8%B1%A1/2331271" \t "http://baike.baidu.com/_blank)

**[文档对象模型](http://baike.baidu.com/item/%E6%96%87%E6%A1%A3%E5%AF%B9%E8%B1%A1%E6%A8%A1%E5%9E%8B" \t "http://baike.baidu.com/_blank)（DOM）**，描述处理网页内容的方法和接口

**浏览器对象模型（[BOM](http://baike.baidu.com/item/BOM/1801" \t "http://baike.baidu.com/_blank)）**，描述与浏览器进行交互的[方法](http://baike.baidu.com/item/%E6%96%B9%E6%B3%95/3009367" \t "http://baike.baidu.com/_blank)和[接口](http://baike.baidu.com/item/%E6%8E%A5%E5%8F%A3" \t "http://baike.baidu.com/_blank)。

## JavaScript和HTML的结合方式

* 两种使用方式

>内部使用

\*<script type=”text/javascript”>JavaScript的代码</script>

>外部引用

\*<script type=”text/javascript” src=”javascript文件路径”></script>

**\*外部引用时script标签内不能有script代码，即使写了也不会执行**

# Javascript基本语法

## 注释

\*单行注释

//

\*多行注释

/\* \*/

## 变量

对于学习完java的我们来说变量并不陌生。标志内存中的一块空间，用于存储数据，数据据是可以被更改的 。

### 变量的声明

==> JavaScript中所有类型的变量声明都用var关键字

===> **变量值**的数据类型（原始数据类型和引用数据类型）

**定义 :**

* 变量必须以字母开头
* 变量也能以 $ 和 \_ 符号开头（不过我们不推荐这么做）
* 变量名称对大小写敏感（y 和 Y 是不同的变量）

### 基本数据类型变量

|  |  |
| --- | --- |
| string | 字符串类型  “”和’’都表示字符串 |
| boolean | 布尔类型  true,false |
| number | 数字类型  整数、小数和NaN（Not a Number）  NaN!=NaN |
| null | 空，对象的占位符  表示引用类型的对象不存在 |
| undefined | 值未定义。即声明变量未赋值，系统会自动赋值undefined  变量声明未赋值时使用/对象的属性未赋值时使用 都会出现undefined  但如果变量 未声明 就使用，JavaScript会报 变量未定义并截停代码。  例如：alert(str);//JavaScript代码在此行停止解析运行，并抛出 变量str未定义 的错误  undefined容易和 变量未声明就使用 混淆    变量声明未定义：  var str;  alert(str);//undefined    变量未声明就使用：  alert(aaaccc);//报错，并且截断JAVASCRIPT代码  undefined==undefined |

#### 定义变量

|  |
| --- |
| **var** a = 2;  **var** b = "2";  **var** c = **true**;  alert(**typeof** (a));//number  alert(**typeof** (b));//string  alert(**typeof** (c));//boolean  alert(a - b);//0  alert(a + b);//22  alert(a == b);//true  alert(a === b);//false |

## 运算符

### 2.3.1算数运算符

算术运算加

字符串连接符

number转换符(其他类型转number)：

-符号

算术运算减

数字转换符

\*会把其他值转换成数字，进行算术运算减

string转number

如果string可以转换成number，那么转换后算术运算

如果不能转换成number，那么整体NaN

boolean转number

True--->1

False---> 0

null转number

转换成0

undefined转number

转成NaN

NaN和其他数字运算都是NaN

number转换符(其他类型转number)：

string转number ：将值转成数字类型，如果值不是合法数字，就转为NaN 例如：var str = -"11";

boolean转number ： true转为1 ，false转为0

null转number: 转为0

undefined转number：转为 NaN

注：使用算术运算符时，都会向数字方向进行转换，如果不是数字类型，则转换。

,

### 2.3.2、赋值运算符（了解）

= 赋值

+= 原值先加后赋值（若原值不为可运算数字，自动转换为对应数字,string类型值除外）

-= 原值先减后赋值（若原值不为可运算数字，自动转换为对应数字）

不可运算数字：

string,boolean,null,undefined

### 2.3.3、比较运算符（重点）

== 比较值

=== 比较值和类型。 如果变量值和变量类型都相同，那么true,只要有一个不同 false

!= 比较值

!== 全不等。比较值或比较类型，有一个不同即为不同，true. 值和类型都相同，那么false

|  |
| --- |
| 练习：  var x=8;  alert(x==8);  alert(x===8);  alert(x=="8");  alert(x==="8");    alert(x!=8);  alert(x!=1);  alert(x!==8);  alert(x!==1);  alert(x!="8");  alert(x!="1");  alert(x!=="8");  alert(x!=="1"); |

### 2.3.4**、逻辑运算符（重点）**

&& 逻辑与

|| 逻辑或

！ 非

注：使用以上逻辑运算符，最终都会用boolean类型运算，若不是boolean类型，则进行转换。

number转boolean：-0或0或NaN 转换成false ，其他为true

string转boolean：””为false，其他为true

null转boolean：转为false

undefined转boolean：转为false

对象转boolean：转为true

综上：0、-0、null、""、false、undefined 或 NaN，为false

否则为true

### 2.3.5、条件运算符(三元运算符)

var val=(表达式)?value1:value2

1、条件运算符 最终都是为了赋值

2、表达式，最终都是boolean类型的运算

## 流程控制语句（重点）

### 判断语句

IF语句

if(条件){

}else{

}

|  |
| --- |
| var a=15;  if(a=15){  alert(15);  }else{  alert(“else”);  } |
| 15 |

注：若写 = 则为赋值

Switch语句

switch(n)

{

case 1:

执行代码块 1

Break

case 2 :

执行代码块 2

break

default:

如果n即不是1也不是2，则执行此代码

}

注：JS中的switch接收任何类型的值

Switch会对值和值的类型进行比较

### 循环语句

for循环：

for(var i=0;i<=8;i++){

// 循环体

}

For循环和JAVA中的FOR循环只有在条件定义时不一样的

JavaScript中。 var

for..in循环：

类似于Java的增强for循环（但有本质区别）

For...In 声明用于对数组或者对象的属性进行循环操作。

for ... in 循环中的代码每执行一次，就会对数组的元素进行一次操作。

for(变量 in 对象或数组){

// 循环体

}

区别于Java：

遍历数组时，其中变量代表的是数组的下标。

例如：

数组遍历：

var p=[1,2,3];

for(i in p){

alert("数组下标："+i);

alert("数组元素："+p[i]);

}

while(表达式){

// 循环体

}

# Javascript的方法

什么是函数： 可以理解位java中的方法 。

## 方法的创建方式

**#第一种**

function 函数名称(参数列表){

函数体;

}

**#第二种**

var 函数名称=function(参数列表){

函数体;

}

**#第三种(了解)**

Var 函数名称=new Function(参数列表,函数体);

## 调用方法

函数可以通过其名字加上括号中的参数进行调用。

示例1

<script type="text/javascript">

//使用function 函数名称(参数列表 ){}方式来创建函数

function add(a, b) {

alert(a + b);

}

//调用函数

add(10, 20);

</script>

示例2

<script type="text/javascript">

//使用 var 函数名称=function(参数列表)来创建函数

var add = function(a, b) {

alert(a + b);

}

//函数调用

add(10, 20);

</script>

示例3

<script type="text/javascript">

var add = new Function("a", "b", "alert(a+b)");

add(10, 20);

</script>

问题:如果我们在调用函数时，使用add(1,2,3,4)是否会出现问题哪?

如果是在java中我们这样调用一定会有问题的，但是对于javascript来说，是不会有问题的。

简单的说，也就是函数调用时传递的参数不一定要与函数声明时的参数个数一致。

问题:如果调用时传递的参数与函数定义时参数个数不一致，那么我们在函数中如果获取多余的参数哪?

在javascript中有一个特殊的对象arguments,我们可以通过它来获取所有函数中的参数。

<script type="text/javascript">

//创建函数

function add(a, b) {

//在函数中如果要获取所有参数列表中的信息可以使用argument来操作

for(var i=0;i<arguments.length;i++){

alert(arguments[i]);

}

}

add(1,2,3,4);

</script>

**函数参数与返回值**

在java中我们定义一个方法，如果这个方法有返回值，我们必须做到以下两点:

在方法声明时，必须声明其返回值类型，如果无返回值需要使用void。

方法有返回值，我们在方法中需要使用return来返回结果。

那么我们在javascript中应该怎样处理一个函数的返回值哪?

对于javascript来说，我们不必须在创建函数时，来声明其返回值类型，无返回值也不需要使用void来声明,但是，如果一个函数要想返回结果，我们在函数中必须使用return来返回结果。

<script type="text/javascript">

//创建函数

function add(a, b) {

//返回a与b的和

return a+b;

}

//在调用函数时，我们就可以直接使用一个变量来接收函数的返回结果

var sum=add(1,2,3,4);

alert(sum)

</script>

### 引用数数据类型

* + - Javascript常用对象-Boolean
    - Javascript常用对象-Number
    - Javascript常用对象-Array
    - Javascript常用对象-String
    - Javascript常用对象-Date
    - Javascript常用对象-Math
    - Javascript常用对象-RegExp

#### Javascript常用对象-Boolean

Boolean对象描述

在 JavaScript 中，布尔值是一种基本的数据类型。Boolean 对象是一个将布尔值打包的布尔对象。Boolean 对象主要用于提供将布尔值转换成字符串的 toString() 方法。

Boolean对象创建

Boolean 对象表示两个值："true" 或 "false"。

创建 Boolean 对象的语法：

new Boolean(value); //构造函数

Boolean(value); //转换函数

**参数**

参数 value 由布尔对象存放的值或者要转换成布尔值的值。

**返回值**

当作为一个构造函数（带有运算符 new）调用时，Boolean() 将把它的参数转换成一个布尔值，并且返回一个包含该值的 Boolean 对象。

如果作为一个函数（不带有运算符 new）调用时，Boolean() 只将把它的参数转换成一个原始的布尔值，并且返回这个值。

注释：如果省略 value 参数，或者设置为 0、-0、null、""、false、undefined 或 NaN，则该对象设置为 false。否则设置为 true（即使 value 参数是字符串 "false"）。

|  |
| --- |
| var b = true;  var c = new Boolean("true");  var d = new Boolean(true);    // alert(typeof(b));//boolean  // alert(typeof(c));//object  // alert(typeof(d));//object |

#### Javascript常用对象-Number

**Number对象描述**

在 JavaScript 中，数字是一种基本的数据类型。JavaScript 还支持 Number 对象，该对象是原始数值的包装对象。在必要时，JavaScript 会自动地在原始数据和对象之间转换。

**Number对象创建**

Number 对象是原始数值的包装对象。

创建 Number 对象的语法：

var myNum=new Number(value);

var myNum=Number(value);

**参数**

参数 value 是要创建的 Number 对象的数值，或是要转换成数字的值。

**返回值**

当 Number() 和运算符 new 一起作为构造函数使用时，它返回一个新创建的 Number 对象。如果不用 new 运算符，把 Number() 作为一个函数来调用，它将把自己的参数转换成一个原始的数值，并且返回这个值（如果转换失败，则返回 NaN）。

|  |
| --- |
| //Number对象  var f = 12;  var g = new Number(2);  // alert(typeof(f));//number  // alert(typeof(g));//object |

#### Javascript常用对象-Array

Array 对象用于在单个的变量中存储多个值。

创建 Array 对象的语法：

new Array();

new Array(size);

new Array(element0, element1, ..., elementn);

**参数**

参数 size 是期望的数组元素个数。返回的数组，length 字段将被设为 size 的值。

参数 element ..., elementn 是参数列表。当使用这些参数来调用构造函数 Array() 时，新创建的数组的元素就会被初始化为这些值。它的 length 字段也会被设置为参数的个数。

**返回值**

返回新创建并被初始化了的数组。

如果调用构造函数 Array() 时没有使用参数，那么返回的数组为空，length 字段为 0。

当调用构造函数时只传递给它一个数字参数，该构造函数将返回具有指定个数、元素为 undefined 的数组。

当其他参数调用 Array() 时，该构造函数将用参数指定的值初始化数组。

当把构造函数作为函数调用，不使用 new 运算符时，它的行为与使用 new 运算符调用它时的行为完全一样。



|  |
| --- |
| <script>  //定义数组  **var** arr1 = **new** Array();//正确 长度为0  **var** arr2 = **new** Array(6);//6指的是长度  **var** arr3 = **new** Array(5, 6);//5,6指的是元素  **var** arr4 = [ "1", 1, **true** ];//弱数据类型  arr1[0] = 12;//向数组中添加元素  arr1.push("abc");//向数组中添加元素  arr1.pop();//删除最后一个元素  alert(arr1.length);//1  //join方法测试  //alert(arr4);  //alert( arr4.join("~~~~~~~~~~~~~~~~~~~~~"));//按指定的分隔符输出数组的元素  arr4[10] = "hello";//代表给下标为10的索引位置赋值为hello  // arr4.sort();//排序按字典序  arr4[4] = 4;  arr4[5] = 12;  arr4[6] = 200;  arr4[7] = 30;  arr4[8] = 300;  arr4.sort();  alert(arr4);//1,1,12,200,30,300,4,itheima,true 排序按字典序  // shift() 在头部删除 unshift()在头部添加  // pop() 在末尾删除 push() 在末尾添加  </script> |

#### Javascript常用对象-String

**String 对象描述**

字符串是 JavaScript 的一种基本的数据类型。

String 对象的 length 属性声明了该字符串中的字符数。

String 类定义了大量操作字符串的方法，例如从字符串中提取字符或子串，或者检索字符或子串。

需要注意的是，JavaScript 的字符串是不可变的（immutable），String 类定义的方法都不能改变字符串的内容。像 String.toUpperCase() 这样的方法，返回的是全新的字符串，而不是修改原始字符串。

**String对象创建**

String 对象用于处理文本（字符串）。

创建 String 对象的语法：

new String(s);

String(s);

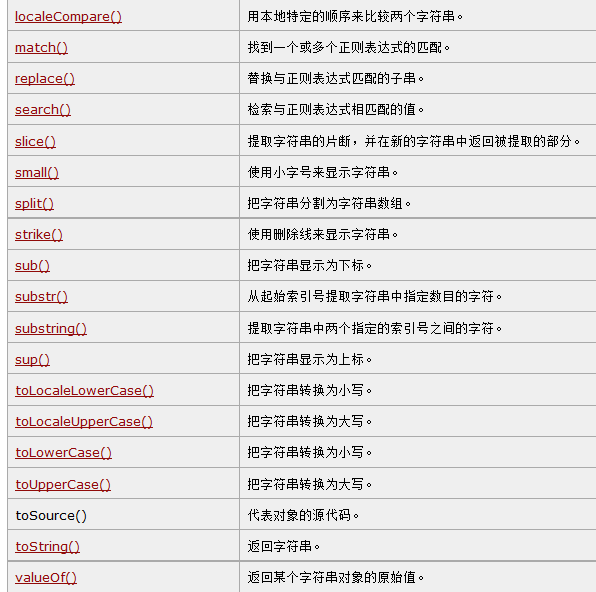
**参数**

参数 s 是要存储在 String 对象中或转换成原始字符串的值。

**返回值**

当 String() 和运算符 new 一起作为构造函数使用时，它返回一个新创建的 String 对象，存放的是字符串 s 或 s 的字符串表示。

当不用 new 运算符调用 String() 时，它只把 s 转换成原始的字符串，并返回转换后的值。



|  |
| --- |
| **var** s = "Hello";  **var** t = **new** String("world");  // alert(typeof(s));//string  // alert(typeof(t));//object  // document.write(s);  //document.write(s.big());//变成大号字  // 比较substr与substring两个函数  **var** r = "abcdefgh";  //alert(r.substring(1, 2));//b 第一个参数：1代表起始位置（从0计算)第二个参数：结束位置,含头不含尾）  // alert(r.substr(1,2));//bc 第一个参数：代表起始位置 第二个参数：代表长度  **var** personinfo = "wangwu,zhangsan,lisi,zhaoliu";  **var** perarray = personinfo.split(",");//实现字符串切割  **for** (**var** i = 0; i < perarray.length; i++) {  // alert(perarray[i]);  }  **var** email = "598196583@qq.com";  email.indexOf("@");//方法返回-1或找到的索引号  **if** (email.indexOf("@") == -1) {  alert("邮箱中必须有@");  } **else** {  alert("验证通过"); } |

#### Javascript常用对象-Date

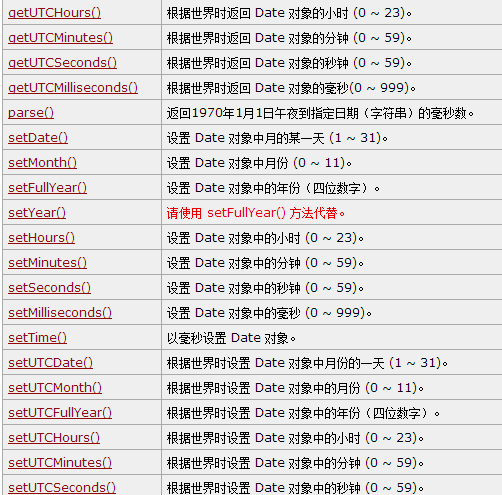
**Date 对象创建**

Date 对象用于处理日期和时间。

创建 Date 对象的语法：

var myDate=new Date()

注释：Date 对象会自动把当前日期和时间保存为其初始值。



动态时钟案例

|  |
| --- |
| <**script type="text/javascript"**>  **function** *clock*() {  **var** a = **new** Date();  **var** H = a.getHours();  **var** M = a.getMinutes();  **var** S = a.getSeconds();  H = (H < 10) ? **"0"** + H : H;  M = (M < 10) ? **"0"** + M : M;  S = (S < 10) ? **"0"** + S : S;  **document**.getElementById(**"clock"**).**value** = H + **":"** + M + **":"** + S;  }  setInterval(**"***clock***()"**, 1000);*//定时器。切记不要写在被调函数的方法内 方法加引号* </**script**> </**head**> <**body**> <**input type="text" value=“” id="clock" style="border**:0 ;**text-align**: **center**;**color**: **red**; **font-size**:60**px"**> </**body**> |

|  |
| --- |
| **var *a*** = Date.parse(**"2015/01/01"**); **var *b*** = Date.parse(**"2015/01/031"**); alert((***b***-***a***)/1000/60/60/24);*//30*  计算两个日期之间的天数 |

#### Javascript常用对象-Math

**Math对象创建**

Math 对象用于执行数学任务。

使用 Math 的属性和方法的语法：

var pi\_value=Math.PI;

var sqrt\_value=Math.sqrt(15);

注意：Math 对象并不像 Date 和 String 那样是对象的类，因此没有构造函数 Math()，像 Math.sin() 这样的函数只是函数，不是某个对象的方法。您无需创建它，通过把 Math 作为对象使用就可以调用其所有属性和方法。



#### Javascript常用对象-RegExp

RegExp 对象表示正则表达式，它是对字符串执行模式匹配的强大工具。

**直接量语法**

/pattern/attributes

创建 RegExp 对象的语法：

new RegExp(pattern, attributes);

**参数**

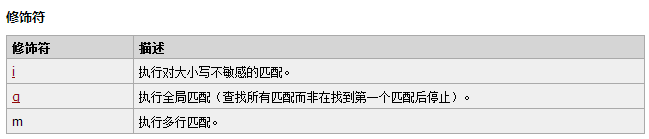
参数 pattern 是一个字符串，指定了正则表达式的模式或其他正则表达式。

参数 attributes 是一个可选的字符串，包含属性 "g"、"i" 和 "m"，分别用于指定全局匹配、区分大小写的匹配和多行匹配。ECMAScript 标准化之前，不支持 m 属性。如果 pattern 是正则表达式，而不是字符串，则必须省略该参数。

**返回值**

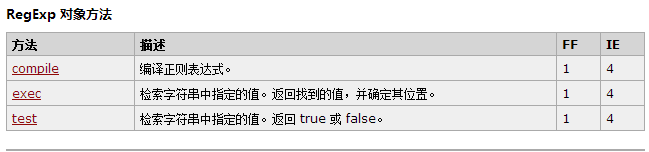
一个新的 RegExp 对象，具有指定的模式和标志。如果参数 pattern 是正则表达式而不是字符串，那么 RegExp() 构造函数将用与指定的 RegExp 相同的模式和标志创建一个新的 RegExp 对象。

如果不用 new 运算符，而将 RegExp() 作为函数调用，那么它的行为与用 new 运算符调用时一样，只是当 pattern是正则表达式时，它只返回 pattern，而不再创建一个新的 RegExp 对象。







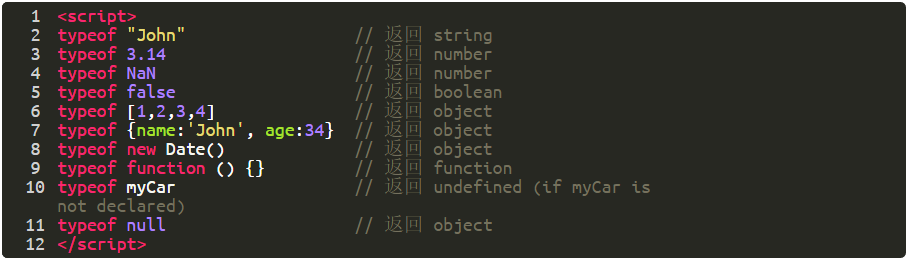




#### **两个不包含任何值的数据类型：**

* null
* Undefined 未定义

你可以使用 **typeof** 操作符来查看 JavaScript 变量的数据类型。

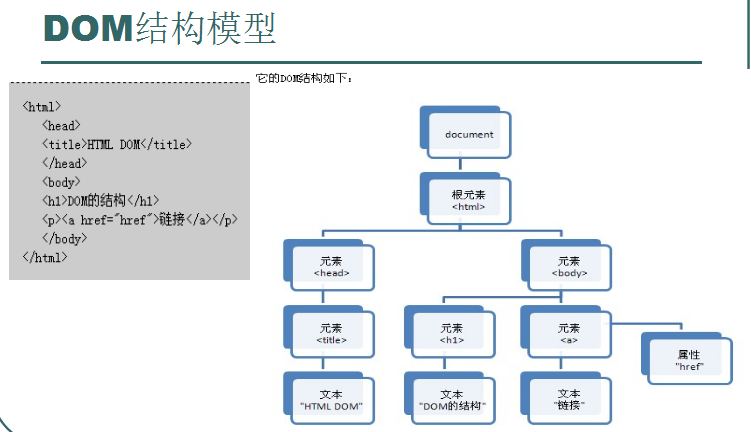


# JS的dom对象

## 什么是dom对象

是W3C织制订的一套用于访问XML和HTML文档的标准.允许脚本动态地访问和更新文档的内容、结构和样式. 。

当网页被加载时，浏览器会创建页面的文档对象模型（Document Object Model），页面加载HTML文档，HTML文档内部本身就遵循文档对象模型规范：内部维护一个如下的都没树模型 。Dom对象就是用来操作 。



## Dom 对象

相当于获得操作整个文档的对象 document 此对象可以操作dom文档中的任何元素 元素的属性 和文本

例子

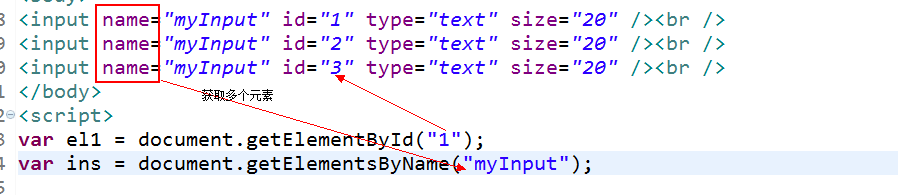
### Document对象的常用方法

**常用方法**

getElementById()返回对拥有指定 id 的第一个对象的引用

document.getElementsByName 数组

document.write("写到页面");



### DOM节点常见操作

**# DOM 获取节点**

**# DOM 改变节点**

**# DOM 删除节点**

**# DOM 替换节点**

**# DOM 创建节点**

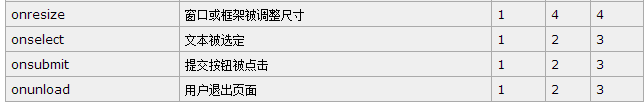
**# DOM 添加节点**

# Javascript事件介绍

## 常见事件:

事件通常与函数配合使用，这样我们可以通过发生的事件来驱动函数执行.





## 事件绑定

在javascript中事件经常与函数一起使用，那么我们介绍一下怎样将一个事件与函数绑定到一起，简单说，就是某个事件被触发后，函数如果被调用。

html事件属性

<button onclick="displayDate()">点击这里</button>

html dom分配事件

<script>

document.getElementById("myBtn").onclick=function(){displayDate()};

</script>

阻止默认事件

if(e&&e.stopPropagation){

e.preventDefault();

}else{

window.event.returnValue = false;

}

阻止事件传播

if(e&&e.stopPropagation){

e.stopPropagation();

}else{

window.event.cancelBubble = true;

}

# Js对cookie的操作

设置cookie并遍历取出所有的cookie

|  |
| --- |
| <**script type="text/javascript"**>  *//cookie是以键值对的形式存在，设置cookie* **document**.**cookie**=**"userId=828"**;  **document**.**cookie**=**"userName=hulk"**;  *//取出所有的cookie* **var *cookies***=**document**.**cookie**;  ***cookies*** = ***cookies***.split(**";"**);  **for**(**i** in ***cookies***){  alert(***cookies***[**i**]);  } </**script**> |

取出指定名称的cookie

|  |
| --- |
| <**script type="text/javascript"**>  *//cookie是以键值对的形式下旬在，设置cookie* **document**.**cookie**=**"userId=828"**;  **document**.**cookie**=**"userName=hulk"**;  *//取出指定名称的cookie值* **var *cName*** = **document**.**cookie**.indexOf(**"userId="**);//828  alert(***cName***); </**script**> |

J

# JavaScript综合案例

## 表单校验案例

**Js**

|  |
| --- |
| **function** checkForm(){    /\*\*校验用户名\*/  //1.获取用户输入的数据  **var** uValue = document.getElementById("user").value;  //alert(uValue);  **if**(uValue==""){  //2.给出错误提示信息  alert("用户名不能为空!");  **return** **false**;  }    /\*校验密码\*/  **var** pValue = document.getElementById("password").value;  **if**(pValue==""){  alert("密码不能为空!");  **return** **false**;  }    /\*\*校验确认密码\*/  **var** rpValue = document.getElementById("repassword").value;  **if**(rpValue!=pValue){  alert("两次密码输入不一致!");  **return** **false**;  }    /\*校验邮箱\*/  **var** eValue = document.getElementById("eamil").value;  **if**(!/^([a-zA-Z0-9\_-])+@([a-zA-Z0-9\_-])+(.[a-zA-Z0-9\_-])+/.test(eValue)){  alert("邮箱格式不正确!");  **return** **false**;  }    } |

**提高**

|  |
| --- |
| **function** showMsg(id,info){  //用户获取焦点的时候  document.getElementById(id+"span").innerHTML="<font color='gray'>"+info+"</font>";  }  //用户失去焦点时调用次方法  **function** check(id,info){  //1.获取用户输入的用户名数据  **var** uValue = document.getElementById(id).value;  //2.进行校验  **if**(uValue==""||" "==uValue){  document.getElementById(id+"span").innerHTML="<font color='red'>"+info+"</font>";  //页面会显示除<html标签> 会吧内容当作文本输出  //document.getElementById(id+"span").innerText="<font color='red'>"+info+"</font>";  }**else**{  document.getElementById(id+"span").innerHTML="";  }  } |

**Form表单**

|  |
| --- |
| <form action=*"http://localhost:8080/com.xiaoniu/CreatServlet"*  method=*"post"*>  <table border=*"1px"* width=*"750px"* height=*"400px"* align=*"center"*  cellpadding=*"0px"* cellspacing=*"0px"* bgcolor=*"white"*>  <tr height=*"40px"* align=*"center"*>  <td colspan=*"2"*><font size=*"4"*>会员注册</font>  &nbsp;&nbsp;&nbsp;USER REGISTER</td>  </tr>  <tr>  <td>用户名</td>  <td><input type=*"text"* name=*"user"* size=*"34px"* id=*"user"*  onfocus="showTips('user','用户名必填！')"  onblur="check('user','用户名不能为空！')" /><span id=*"userspan"*></span></td>  </tr>  <tr>  <td>密码</td>  <td><input type=*"password"* name=*"password"* size=*"34px"*  id=*"password"* onfocus="showTips('password','密码必填')"  onblur="check('password','密码不能为空!')" /><span id=*"passwordspan"*></span>  </td>  </tr>  <tr>  <td>确认密码</td>  <td><input type=*"password"* name=*"repassword"* size=*"34px"*  id=*"repassword"*></input></td>  </tr>  <tr>  <td>Emaile</td>  <td><input type=*"text"* name=*"email"* size=*"34px"* id=*"eamil"* /></td>  </tr>  <tr>  <td>姓名</td>  <td><input type=*"text"* name=*"username"* size=*"34px"* /></td>  </tr>  <tr>  <td>性别</td>  <td><input type=*"radio"* name=*"sex"* value=*"男"* />男 <input  type=*"radio"* name=*"sex"* value=*"女"* />女</td>  </tr>  <tr>  <td>出生日期</td>  <td><input type=*"text"* name=*"birthday"* size=*"34px"* /></td>  </tr>  <tr>  <td colspan=*"2"*><input type=*"submit"* value=*"注册"* /></td>  </tr>  </table>  </form> |

## 图片切换案例

|  |
| --- |
| <script type=*"text/javascript"*>  **var** i = 0;  **function** changImg() {  i++ ;  document.getElementById("img1").src="./"+i+".jpg" ;  **if**(i==3){  i=0 ;  }  }  </script>  </head>  <body>  <div id=*""*>  <img src=*"./3.jpg"* width=*"160"* height=*"180"* id=*"img1"* />  </div>  <input type=*"button"* value=*"切换图片"* onclick="changImg();" />  </body> |

使用js定时函数自动切换

|  |
| --- |
| <script type=*"text/javascript"*>  **function** init(){  setInterval("changImg()",2000);  }  **var** i = 0;  **function** changImg() {  i++ ;  document.getElementById("img1").src="./"+i+".jpg" ;  **if**(i==3){  i=0 ;  }  }  </script>  </head>  <body onload="init();">  <div id=*""*>  <img src=*"./3.jpg"* width=*"160"* height=*"180"* id=*"img1"* />  </div>  <input type=*"button"* value=*"切换图片"* onclick="changImg();" />  </body> |

## 动态添加元素案例

凡是涉及到操作元素节点的时候一定要在元素节点加载完成以后再操作，这个例子把js放在body上面就不会成功 。

|  |
| --- |
| <body>  <ul id=*"ul1"*>  <li>杨过</li>  <li>杨幂</li>  <li>杨颖</li>  </ul>  <input type=*"button"* value=*"添加新的人"* id=*"btn"*/>  </body>  <script type=*"text/javascript"*>  document.getElementById("btn").onclick = **function**(){  //1.获取ul元素节点  **var** ulEle = document.getElementById("ul1");  //2.创建城市文本节点  **var** textNode = document.createTextNode("杨康");//深圳  //3.创建li元素节点  **var** liEle = document.createElement("li");//<li></li>  //4.将城市文本节点添加到li元素节点中去  liEle.appendChild(textNode);//<li>深圳</li>  //5.将li添加到ul中去  ulEle.appendChild(liEle);  }  </script> |

## 城市二级级联案例

|  |
| --- |
| <body>  <table>  <tr>  <td>籍贯</td>  <td><select onchange=*"changeCity(this.value)"*>  <option>--请选择--</option>  <option value=*"0"*>北京</option>  <option value=*"1"*>湖南</option>  <option value=*"2"*>江苏</option>  <option value=*"3"*>河南</option>  </select>  <select id=*"city"*></select>  </td>  </tr>  </table>  </body>  <script>  //1.创建一个二维数组用于存储省份和城市  **var** cities = **new** Array(3);  cities[0] = **new** Array("朝阳", "海淀", "昌平");  cities[1] = **new** Array("长沙市", "株洲市", "岳阳市");  cities[2] = **new** Array("苏州", "常州", "徐州", "无锡");  cities[3] = **new** Array("郑州市", "洛阳市", "开封市", "安阳市");  **function** changeCity(val) {  //7.获取第二个下拉列表  **var** cityEle = document.getElementById("city");  //9.清空第二个下拉列表的option内容  cityEle.options.length = 0;  //2.遍历二维数组中的省份  **for** (**var** i = 0; i < cities.length; i++) {  //注意，比较的是角标  **if** (val == i) {  //3.遍历用户选择的省份下的城市  **for** (**var** j = 0; j < cities[i].length; j++) {  //alert(cities[i][j]);  //4.创建城市的文本节点  **var** textNode = document.createTextNode(cities[i][j]);  //5.创建option元素节点  **var** opEle = document.createElement("option");  //6.将城市的文本节点添加到option元素节点  opEle.appendChild(textNode);  //8.将option元素节点添加到第二个下拉列表中去  cityEle.appendChild(opEle);  }  }  }  }  </script> |

案例3 ，4 中涉及到元素的操作。使用window.onload=function(){};