JavaScript

# 基本知识

10——livescript——JavaScript,JScript——微软，ECMA——ECMAScript,BOM,DOM

JavaScript 是属于网络的脚本语言！

JavaScript 被数百万计的网页用来改进设计、验证表单、检测浏览器、创建cookies，以及更多的应用。

JavaScript 是因特网上最流行的脚本语言。

JavaScript的组成部分：ECMAScript,BOM,DOM

什么是 JavaScript？

JavaScript 被设计用来向 HTML 页面添加交互行为（动画，验证）。

JavaScript 是一种脚本语言（脚本语言是一种轻量级的编程语言）。

JavaScript 由数行可执行计算机代码组成。

JavaScript 通常被直接嵌入 HTML 页面。

JavaScript 是一种解释性语言（就是说，代码执行不进行预编译）。

所有的人无需购买许可证均可使用 JavaScript。

# 第一个程序

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>JavaScript第一个程序</**title**>  <**script type="text/javascript"**>  */\*js的代码写于script标签中 js也有3种写入方式  行内js----在元素中直接写js的代码  内部js------在html页面中 嵌入script标签，代码写入标签中。script标签  可以在页面的任何位置。但是要注意js是顺序执行的。  外部js------引入外部的js文件  \*/  //JavaScript的单行注释 JavaScript简称为js  /\*  \* 多行注释  \* \*/* **function** *test01*(){  *//alert是系统自带的函数 用于弹出警告框，用于输出  alert*(**"hello world"**)  }  *//test01();//函数被调用* </**script**> </**head**> <**body**>  <**input type="button" value="test01" onclick="***test01*();**"**/>  <**input type="button" value="test02" onclick="javascript**:*alert*(**'行内'**)**"**/> </**body**> </**html**> |

# 变量

变量是存储数据空间的表示。变量是用于存储信息的容器。

在js中声明变量需要使用var来表示这是一个变量，也可以不用。

如果在函数的内部使用var声明变量，那么该变量是局部变量，如果不使用var,那么该变量将变为全局变量。在函数的外面声明的变量都是全局变量。

JavaScript 变量名称的规则：

变量对大小写敏感（y 和 Y 是两个不同的变量）

变量必须以字母或下划线开始

在js中，可以通过var来声明变量然后不立即赋值。

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>变量</**title**>  <**script type="text/javascript"**>  **x** = 100  **function** *test01*() {  **x** = 5  }  **function** *test02*() {  *alert*(**x**)  }  </**script**> </**head**> <**body**> <**button onclick="***test01*()**"**>test01</**button**> <**button onclick="***test02*()**"**>test02</**button**> </**body**> </**html**> |

# 数据类型

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>数据类型</**title**>  <**script type="text/javascript"**>  */\*  \* js的数据类型：大的分类两类：原生类型，对象(自定义)  \* 原生类型：  字符串类型 string 在js中定义字符串有两种方式 一种使用单引号，一种使用双引号  数字类型 number 在js中 number即表示整数又表示小数  布尔类型 boolean 在js中 boolean使用小写true和false  undefined 未定义  object 对象  null  \* \*/* **function** *test01*() {  **var** t = **"hello world"** *//typeof函数查看数据类型  // alert(typeof(t))* **var** num1 = 20.2  *// alert(typeof(num1))* **var** flag = **true** *// alert(typeof(flag))* **sname** = **null** *alert*(**typeof**(**null**))  }  </**script**> </**head**> <**body**>  <**button onclick="***test01*()**"**>test01</**button**> </**body**> </**html**> |

# 运算符

算术运算符

赋值运算符

Boolean 运算符

逻辑 NOT 运算符的行为如下：

* 如果运算数是对象，返回 false
* 如果运算数是数字 0，返回 true
* 如果运算数是 0 以外的任何数字，返回 false
* 如果运算数是 null，返回 true
* 如果运算数是 NaN，返回 true
* 如果运算数是 undefined，发生错误

如果某个运算数不是原始的 Boolean 型值，逻辑 AND 运算并不一定返回 Boolean 值：

* 如果一个运算数是对象，另一个是 Boolean 值，返回该对象。
* 如果两个运算数都是对象，返回第二个对象。
* 如果某个运算数是 null，返回 null。
* 如果某个运算数是 NaN，返回 NaN（not a number）。
* 如果某个运算数是 undefined，发生错误。

与逻辑 AND 运算符相似，如果某个运算数不是 Boolean 值，逻辑 OR 运算并不一定返回 Boolean 值：

* 如果一个运算数是对象，另一个是 Boolean 值，返回该对象。
* 如果两个运算数都是对象，返回第一个对象。
* 如果某个运算数是 null，返回 null。
* 如果某个运算数是 NaN，返回 NaN。
* 如果某个运算数是 undefined，发生错误。

在js中除以0是有值得。

乘法还有一些特殊行为：

* 如果结果太大或太小，那么生成的结果是 Infinity 或 -Infinity。
* 如果某个运算数是 NaN，结果为 NaN。
* Infinity 乘以 0，结果为 NaN。
* Infinity 乘以 0 以外的任何数字，结果为 Infinity 或 -Infinity。
* Infinity 乘以 Infinity，结果为 Infinity。

除法运算符也有一些特殊行为：

* 如果结果太大或太小，那么生成的结果是 Infinity 或 -Infinity。
* 如果某个运算数是 NaN，结果为 NaN。
* Infinity 被 Infinity 除，结果为 NaN。
* Infinity 被任何数字除，结果为 Infinity。
* 0 除一个任何非无穷大的数字，结果为 NaN。
* Infinity 被 0 以外的任何数字除，结果为 Infinity 或 -Infinity。

取模运算符也有特殊的行为：

* 如果被除数是 Infinity，或除数是 0，结果为 NaN。
* Infinity 被 Infinity 除，结果为 NaN。
* 如果除数是无穷大的数，结果为被除数。
* 如果被除数为 0，结果为 0。

# 类型转换

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>类型转换</**title**>  <**script type="text/javascript"**>  **function** *test01*() {  **num** = 0  **r1** = **num**&&**true***//0* **r2** = **num**+**true***//1* **r3** = **true**&&**num***//0* **r4** = **num**+**'23'***//做拼接023  //parseInt()将字符串转换为整数  //parseFloat()将字符串转换为浮点数* **r5** = **num**+*parseInt*(**'23px'**)*//23* **r5** = **num**+*parseInt*(**'px23'**)*//NaN  //任何数与NaN运算的结果都是NaN* **r6** = 1+**null***//1* **r6** = 1+**undefined** *//NaN* **r6** = **true**&&**undefined** *//undefined* **r6** = **true**&&**null** *//null* **r6** = !**null** *//true* **r6** = **false**&&7 *//false* **r6** = **false** == 0 *//true  // null<--->number<---->boolean* **r6** = **false** == **null** *//false  //null<--->string<---->number* **r6** = **null**+**'ab'** *//nullab* **t1** = **true  t2** = 1  *// == 等值符：值相等即等 ===等同符 值相等并且同类型* **r6** = **t1** === **t2** *alert*(**r6**)  }  </**script**>  </**head**> <**body**>  <**button onclick="***test01*()**"**>test01</**button**> </**body**> </**html**> |

# 分支语句

## If语句

语法：

|  |
| --- |
| if (条件)  {  条件成立时执行代码  } |

## If-else语句

如果希望条件成立时执行一段代码，而条件不成立时执行另一段代码，那么可以使用 if....else 语句。

|  |
| --- |
| if (条件)  {  条件成立时执行此代码  }else  {  条件不成立时执行此代码  } |

## If...else if...else 语句

当需要选择多套代码中的一套来运行时，请使用 if....else if...else 语句。

|  |
| --- |
| if (条件1)  {  条件1成立时执行代码  }else if (条件2)  {  条件2成立时执行代码  }else  {  条件1和条件2均不成立时执行代码  } |

代码

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>if语句</**title**>  <**script type="text/javascript"**>  **function** *test01*() {  *//Date 是js中内置的日期对象 new一个日期就是当前时间* **var** d = **new *Date***()  */\*  \* if语句后的大括号 表示if条件成立时要执行的代码块  \* if语句后的小括号中是条件  \* 当if语句的代码块只有一条语句时 可以省略大括号  \* \*/* **if**(d.getHours()>12){  *alert*(**"good afternoon"**)  }  }  **function** *test02*() {  **var** d = **new *Date***()  */\*  \* if语句和else语句必须配套  \* \*/* **if**(d.getHours()<10){  *alert*(**"good morning"**)  }  **else**{  *alert*(**"good afternoon"**)  }  }  **function** *test03*() {  **var** score = 90  **if** (score>=90){  *alert*(**"A"**)  }**else if**(score>=80){  *alert*(**"B"**)  }**else if**(score>=60){  *alert*(**"C"**)  }**else**{  *alert*(**"D"**)  }  }  </**script**> </**head**> <**body**>  <**button onclick="***test01*()**"**>test01</**button**>  <**button onclick="***test02*()**"**>test02</**button**>  <**button onclick="***test03*()**"**>test03</**button**> </**body**> </**html**> |

## Switch语句

如果希望选择执行若干代码块中的一个，你可以使用 switch 语句：

语法：

|  |
| --- |
| 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  } |

**工作原理**：switch 后面的 (n) 可以是表达式，也可以（并通常）是变量。然后表达式中的值会与 case 中的数字作比较，如果与某个 case 相匹配，那么其后的代码就会被执行。**break** 的作用是防止代码自动执行到下一行。

Switch后面的表达式可以是整数，也可以是字符串。Case后一定是常量

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>Switch结构</**title**>  <**script type="text/javascript"**>  *//给定一个月份 判断月份是哪个季节* **function** *test01*() {  **var** m = *parseInt*(*prompt*(**"输入月份"**,1))  **switch** (m){  **case** 12:  **case** 1:  **case** 2:  *alert*(**"冬季"**)  **break  case** 3:  **case** 4:  **case** 5:  *alert*(**"春季"**)  **break  case** 6:  **case** 7:  **case** 8:  *alert*(**"秋季"**)  **break  case** 9:  **case** 10:  **case** 11:  *alert*(**"秋季"**)  **break** }  }  **function** *test02*() {  **t** = **'A'  switch** (**t**){  **case 'A'**:  *alert*(**"优"**)  **break  case 'B'**:  *alert*(**"良"**)  **break  default**:  *alert*(**"中"**)  }  }  </**script**> </**head**> <**body**>  <**button onclick="***test01*()**"**>test01</**button**>  <**button onclick="***test02*()**"**>test02</**button**> </**body**> </**html**> |

# 循环结构

## for循环

作用：将一段代码循环执行指定的次数

在脚本的运行次数已确定的情况下使用 for 循环。

语法：

|  |
| --- |
| for (变量=开始值[表达式1];变量<=结束值[表达式2];变量=变量+步进值[表达式3])  {  需执行的代码  } |

工作原理：

循环开始时，先执行表达式1，在判断表达式2，如果条件为真，那么执行循环代码块，执行循环代码块以后，执行表达式3，执行完表达式3后，判断表达式2，如果条件为真，继续执行循环代码块，直到表达式2条件为假时，循环停止执行。

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>for循环</**title**>  <**script type="text/javascript"**>  **function** *test01*() {  *//计算1到10的和* **sum** = 0  **for**(**i** = 1;**i**<=10;**i**++){  **sum**+=**i** }  *alert*(**sum**)  }  *//计算1到100的奇数和* **function** *test02*() {  *//计算1到10的和* **sum** = 0  **for**(**i** = 1;**i**<=100;**i**+=2){  **sum**+=**i** }  *alert*(**sum**)  }  *//打印99乘法表* **function** *test03*() {  **var** str=**""  for**(**i**=1;**i**<10;**i**++){  **for**(**j**=1;**j**<=**i**;**j**++){  str+=**i**+**"\*"**+**j**+**"="**+(**i**\***j**)+**" "** }  str+=**"\n"** }  *alert*(str)  }  </**script**> </**head**> <**body**>  <**button onclick="***test01*()**"**>test01</**button**>  <**button onclick="***test02*()**"**>test02</**button**>  <**button onclick="***test03*()**"**>test03</**button**> </**body**> </**html**> |

## while 循环

while 循环用于在指定条件为 true 时循环执行代码。

语法：

|  |
| --- |
| while (变量<=结束值)  {  需执行的代码  } |

案例

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>while语句</**title**>  <**script type="text/javascript"**>  **function** *test01*() {  **sum** = 0;  **i** = 1  **while**(**i**<=10)  **sum**+=**i**++  *alert*(**sum**)  }  *//死循环* **function** *test02*() {  *//document.write写入文档中  // while (true)  // //console是打印到控制台  // console.info("hello world")* **for**(;;)  ***console***.info(**"hello world"**)  **for**(;**true**;)  ***console***.info(**"hello world"**)  }  </**script**> </**head**> <**body**>  <**button onclick="***test01*()**"**>test01</**button**> <**button onclick="***test02*()**"**>test02</**button**> </**body**> </**html**> |

## Do-while循环

do...while 循环是 while 循环的变种。该循环程序在初次运行时会首先执行一遍其中的代码，然后当指定的条件为 true 时，它会继续这个循环。所以可以这么说，do...while 循环为执行至少一遍其中的代码，即使条件为 false，因为其中的代码执行后才会进行条件验证。

语法

|  |
| --- |
| do  {  需执行的代码  }while (变量<=结束值) |

案例

|  |
| --- |
| **function** *test03*() {  **sum** = 0  **i**=1  **do**{  **sum**+=**i**++  }**while**(**i**<=10)  *alert*(**sum**) } |

## for...in循环

for ... in 循环中的代码每执行一次，就会对数组的元素或者对象的属性进行一次操作。

语法：

|  |
| --- |
| for (变量 in 对象)  {  在此执行代码  } |

案例

|  |
| --- |
| **function** *test04*() {  **var** arr = [1,2,3,4,5]*//js的数组* **for**(**i in** arr){  ***document***.write(arr[**i**]+**"<br/>"**)  } } |

# 消息框

## 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("文本") |

注意：警告框是需要中断程序的。

## 确认框

确认框用于使用户可以验证或者接受某些信息。

当确认框出现后，用户需要点击确定或者取消按钮才能继续进行操作。

如果用户点击确认，那么返回值为 true。如果用户点击取消，那么返回值为 false。

## 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。

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>message</**title**>  <**script type="text/javascript"**>  **function** *test01*() {  *//点击确定返回的true 点击取消 返回的false* **if**(*confirm*(**"是否删除？"**)){  *alert*(**"执行删除动作"**)  }  }  **function** *test02*() {  *//第二个参数默认值 如果不设置那么为null 建议写默认值* **t** = *prompt*(**"请输入一个数字"**,1)  *alert*(**t**)   }  </**script**> </**head**> <**body**>  <**button onclick="***test01*()**"**>test01</**button**>  <**button onclick="***test02*()**"**>test02</**button**> </**body**> </**html**> |

# Try...Catch 语句

可以测试代码中的错误。try 部分包含需要运行的代码，而 catch 部分包含错误发生时运行的代码。

语法：

|  |
| --- |
| try  {  //在此运行代码  }catch(err)  {  //在此处理错误  } |

案例

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>try</**title**>  <**script type="text/javascript"**>  **function** *test01*() {  **try**{  **i**/0  }  **catch** (err){  ***document***.write(err.toString()+**"<br>"**)  }  ***document***.write(**"tt"**)  }  </**script**> </**head**> <**body**>  <**button onclick="***test01*()**"**>test01</**button**> </**body**> </**html**> |

# 事件

事件是可以被 JavaScript 侦测到的行为。

事件包括了3个组成部分：事件源，事件本身，事件的处理

事件源指事件的发生地，事件本身指发生了什么样的事件，事件的处理一般发生事件都需要做出一定动作，通常通过实现函数来表示事件的处理。

