# day28-前端【javascript基础】

# 学习目标

01\_能够说出五种原始的数据类型

02\_能够使用JS中常用的运算符

03\_能够使用JS中的流程控制语句

04\_能够在JS中定义命名函数和匿名函数

05\_能够使用JS中常用的事件

06\_能够使用数组中常用的方法

07\_能够使用日期对象常用的方法

# 项目清单

|  |
| --- |
|  |

# 回顾与提问

1\_表格和表单有什么作用？

|  |
| --- |
| 表格table：对页面进行布局的。  表单form：收集客户端信息，并提交到服务器。 |

2\_说出类选择器的声明和使用语法？

|  |
| --- |
| 声明：.a{...}  使用：<div class=”a”> |

# 问题引入

HTML是做网页的结构和内容的，

CSS是对网页的样式进行更加精确性的修饰的，

但我们希望网页有时能够和用户进行直接的交互，从而引导用户进行操作，如：表单验证。

那么是什么技术可以实现网页的交互功能呢？

答案是：JavaScript，它是客户端网页脚本编程语言。

# 第01章 JavaScript概述

## 什么是JavaScript

在1995年时，由Netscape公司在网景导航者浏览器上首次设计实现而成。因为Netscape与Sun合作，Netscape管理层希望它外观看起来像Java，因此取名为JavaScript。

JavaScript是一种属于网页的脚本语言，已经被广泛用于Web应用开发，常用来为网页添加各式各样的动态功能，为用户提供更流畅美观的浏览效果。通常JavaScript脚本是通过嵌入在HTML中来实现自身的功能的。

|  |
| --- |
|  |

## JavaScript的作用

HTML与用户没有交互的功能，网页只能看，不能操作。为了提升HTML的交互功能，使用一种脚本语言来提升用户体验。脚本语言运行在网页上。

|  |  |
| --- | --- |
| **技术** | **作用** |
| **HTML** | **制作网页，搭建网页的结构和内容。** |
| **CSS** | **对网页进行美化，装饰网页。** |
| **JavaScript** | **让网页有交互功能，提升用户的体验。** |

## JavaScript的特点

|  |  |  |
| --- | --- | --- |
| **特点** | **Java** | **JavaScript** |
| **面向对象** | **完全面向对象** | **基于对象，有些面向对象的特性没有** |
| **运行方式** | **编译型**  **生成中间代码**  **字节码文件** | **解释型**  **解释一行，运行一行**  **不会生成中间文件** |
| **跨平台** | **安装JVM就可以在**  **任何系统上运行** | **运行在浏览器中**  **只要系统有浏览器就可以执行** |
| **大小写** | **区分大小写** | **区分大小写**  **username和userName是不同的二个变量** |
| **数据类型** | **强类型**  **先声明，后使用** | **弱类型**  **可以不声明，直接使用**  **var username = “赵君”;**  **username = “另一个赵君”;** |

## JavaScript的语法组成

|  |  |
| --- | --- |
| **组成部分** | **作用** |
| **ECMA Script** | **脚本语言规范，制作了JS核心的语法基础，（JS第1天）** |
| **BOM** | **Browser Object Model （JS第2天）**  **浏览器对象模型，操作浏览器中各种对象，如：location地址栏** |
| **DOM** | **Document Object Model （JS第2天）**  **文档对象模型，操作网页中的各种元素/标签，如：div标签** |

## 案例：用JS向浏览器输出文字

### 1.5.1案例需求

使用JS在网页上输出5个“你好世界，Hello World”。

### 1.5.2案例效果

|  |
| --- |
|  |

### 1.5.3案例分析

1\_创建一个hello.html文件。

2\_在html文件的<body>标签中，书写<script></script>标签。

3\_在<script></script>标签中用window.document.write("你好世界，Hello World<br/>"); 这句话向浏览器输出内容。

### 1.5.4案例代码

01\_hello.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>用JS向浏览器输出文字</**title**> </**head**> <**body**>  <**script**>  ***window***.document.write(**"你好世界，Hello World<br/>"**);  ***window***.document.write(**"你好世界，Hello World<br/>"**);  ***window***.document.write(**"你好世界，Hello World<br/>"**);  ***window***.document.write(**"你好世界，Hello World<br/>"**);  ***window***.document.write(**"你好世界，Hello World<br/>"**);  </**script**> </**body**> </**html**> |

# 第02章 JavaScript的基础语法

## 2.1 JavaScript的存放位置

JS主要存在两个位置：

1\_放在HTML网页中，用<script>标签包裹。

|  |
| --- |
| **<body>**  <**script**>  ***window***.document.write(**"你好世界，Hello World<br/>"**);  ***window***.document.write(**"你好世界，Hello World<br/>"**);  ***window***.document.write(**"你好世界，Hello World<br/>"**);  ***window***.document.write(**"你好世界，Hello World<br/>"**);  ***window***.document.write(**"你好世界，Hello World<br/>"**);  </**script**>  **</body>** |

2\_放在HTML外部，单独以JS文件的方式存在。推荐使用这种，降低代码的耦合度。

|  |
| --- |
| <**head**>  <**meta charset="UTF-8"**>  <**title**>用JS向浏览器输出文字</**title**>  <**script type="text/javascript" src="js/my.js"**></**script**> </**head**> |

|  |
| --- |
| js/my.js文件：  ***window***.document.write(**"你好世界，Hello World<br/>"**); ***window***.document.write(**"你好世界，Hello World<br/>"**); ***window***.document.write(**"你好世界，Hello World<br/>"**); ***window***.document.write(**"你好世界，Hello World<br/>"**); ***window***.document.write(**"你好世界，Hello World<br/>"**); |

|  |
| --- |
|  |

File Encoding 中选择 UTF-8。

语法解释：

<script>中的src属性和type属性：

type： 指定脚本的类型，固定写法：text/javascript，默认值。

src： 指定外部JS文件的地址。

<script>标签个数：可以出现多个标签，每个标签会从上到下依次运行。

<script>标签出现的位置： 可以放在网页中的任何位置，甚至是放在HTML根元素的外面，

建议放在<head></head>或<body></body>中。

<script>标签中关于语句后面的分号：如果一行代码一条语句，后面的分号可以省略。

多句代码写在同一行，就会出错。建议每一行代码结束加上分号。

## 2.2 JavaScript的注释

|  |  |
| --- | --- |
| **语言** | **注释语法** |
| **HTML** | **<!-- 注释 -->** |
| **CSS** | **/\* 注释 \*/** |
| **JavaScript** | **// 单行注释**  **/\* 多行注释 \*/** |

## 2.3变量

### 2.3.1变量的定义

|  |  |  |
| --- | --- | --- |
| **数据类型** | **Java中定义变量** | **JS中定义变量** |
| **整数** | **int i = 5;** | **var i = 5; //整数** |
| **浮点数** | **float f = 3.14F; 或 double d=3.14D;** | **var f = 3.14; //浮点** |
| **布尔** | **boolean b = true;** | **var b = true; //布尔** |
| **字符** | **char c = 'a';** | **var c = 'a'; //字符串，单引双引都可以** |
| **字符串** | **String str = "abc";** | **var str = "abc"; //字符串，单引双引都可以** |

变量的命名规则和Java相同，不要用关键字，保留字，尽量做到见名知意。

### 2.3.2关于JS的弱类型

什么是弱类型：一个变量可以赋值为不同的数据类型，运行时，才能确认具体数据类型。

如：

a = 10;

a = “abc”;

a = true;

最终运行时，a为布尔类型。

### 2.3.3字符和字符串类型的说明

在JS中没有字符和字符串之分，都是字符串类型，而且既可以使用双引号，也可以使用单引号，原则是：外面字符串使用双引号，里面字符串就使用单引号，反之亦然。

### 2.3.4变量定义的特点

1\_var关键字可以省略，建议不要省略。

2\_在JS中一个变量可以重复定义，后面的会覆盖前面的。

3\_<script>标签对变量的作用域没有影响。

02\_var.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>变量定义的特点</**title**> </**head**> <**body**>   <**script**>  **var *tip*** = **"你好世界，'&lt;Hello World&gt;'<br/>"**;  </**script**>   <**script**>  *//程序从上向下执行  //这个script片段，是可以访问上面script片段的tip变量* ***window***.document.write(***tip***);  </**script**>  </**body**> </**html**> |

## 2.4案例：输出不同类型的数据

### 2.4.1案例需求

分别输出每一种类型JS的变量。

### 2.4.2案例效果

|  |
| --- |
|  |

### 2.4.3案例分析

1\_在一个<script>标签中分别定义多种不同类型的变量，

如整型，浮型，字符型，字符串型，布尔型。

2\_在另一个<script>标签中，将上一个<script>标签中的变量，输出到浏览器。

### 2.4.4案例代码

03\_dataType.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>输出不同类型的数据</**title**> </**head**> <**body**>  <**script**>  **var *i*** = 10;  **var *f*** = 3.14;  **var *c*** = **'a'**;  **var *str*** = **"abc"**;  **var *flag*** = **true**;  </**script**>   <**script**>  ***window***.document.write(***i***+**"<br/>"**);  ***window***.document.write(***f***+**"<br/>"**);  ***window***.document.write(***c***+**"<br/>"**);  ***window***.document.write(***str***+**"<br/>"**);  ***window***.document.write(***flag***+**"<br/>"**);  </**script**> </**body**> </**html**> |

# 第03章 数据类型

## 3.1五种原始数据类型

|  |  |
| --- | --- |
| **关键字** | **说明** |
| **number** | **数值型：整数（314）、浮点（3.14），包括正负数** |
| **boolean** | **布尔类型，包含：true/false** |
| **string** | **包含字符和字符串，既可以使用双引号又可以使用单引号** |
| **object** | **对象类型，如：Array/Date** |
| **undefined** | **未定义类型，一个变量没有赋值之前的状态** |

## 3.2 typeof操作符

作用： 用来判断变量是什么类型。

语法：typeof(变量) 或 typeof 变量

注意：对于返回值null与undefined的区别：

null: 对象类型，已经知道了数据类型，但对象为空。

undefined：未定义的类型，并不知道是什么数据类型。

## 3.3案例：获取数据类型

### 3.3.1案例需求

分别输出整数、浮点数、字符串（单引号和双引号）、布尔、

未定义、对象、null的数据类型。

### 3.3.2案例效果

|  |
| --- |
|  |

### 3.3.3案例分析

1\_在一个<script>标签中分别定义多种不同类型的变量，

如整型，字符串型，布尔型，对象型，未定义型。

2\_在另一个<script>标签中，将上一个<script>标签中的变量，输出到浏览器。

### 3.3.4案例代码

04\_typeof.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>用typeof函数来获取数据类型</**title**> </**head**> <**body**>  <**script**>  **var *i*** = 5; *//整型* **var *f*** = 3.14; *//整型* **var *b*** = **true**; *//布尔型* **var *c*** = **'a'**; *//字符串型* **var *str*** = **"abc"**; *//字符串型* **var *d*** = **new *Date***(); *//对象型* **var *u***; *//未定义型* **var *n*** = **null**; *//对象型* </**script**>   <**script**>  ***window***.document.write(***i***+**"为："**+**typeof**(***i***)+**"<br/>"**);  ***window***.document.write(***f***+**"为："**+**typeof**(***f***)+**"<br/>"**);  ***window***.document.write(***b***+**"为："**+**typeof**(***b***)+**"<br/>"**);  ***window***.document.write(***c***+**"为："**+**typeof**(***c***)+**"<br/>"**);  ***window***.document.write(***str***+**"为："**+**typeof**(***str***)+**"<br/>"**);  ***window***.document.write(***d***+**"为："**+**typeof**(***d***)+**"<br/>"**);  ***window***.document.write(***u***+**"为："**+**typeof**(***u***)+**"<br/>"**);  ***window***.document.write(***n***+**"为："**+**typeof**(***n***)+**"<br/>"**);  </**script**> </**body**> </**html**> |

## 3.4字符串转换成数字类型

全局函数（方法），就是可以在JS中任何的地方直接使用的函数，不用导入对象。不属于任何一个对象。

|  |  |
| --- | --- |
| **转换函数** | **作用** |
| **parseInt()** | **将一个字符串转成整数，如果一个字符串包含非数字字符，那么parseInt函数会从首字母开始取数字字符，一旦发现非数字字符，马上停止获取内容。如果转换失败，则返回NaN=Not a Number，不是一个数。**  **如：**  **parseInt(“123”)结果为：123**  **parseInt(“12ABC3”)结果为：12**  **parseInt(“ABC123”)结果为：NaN（Not a Number）** |
| **parseFloat()** | **将一个字符串转成小数，转换原理同上。** |
| **isNaN()** | **转换前判断被转换的字符串是否是一个数字，**  **非数字返回true**  **isNaN = is not a number** |

## 3.5案例：字符串转换成数字类型

### 3.5.1案例需求

用parseInt()和parseFloat()这二个函数，将字符串转成数字。

### 3.5.2案例效果

|  |
| --- |
|  |

### 3.5.3案例分析

1\_在一个<script>标签中分别定义多种不同类型的字符串类型，字母和数字的组合。

2\_在另一个<script>标签中，将上一个<script>标签中的字符串转成数字后，输出到浏览器。

### 3.5.4案例代码

05\_stringToNumber.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>将字符串转成数字</**title**> </**head**> <**body**>  <**script**>  **var *str1*** = **"123"**;  **var *str2*** = **"1A23"**;  **var *str3*** = **"A123"**;   **var *str4*** = **"1.23"**;  **var *str5*** = **"1.ABC23"**;  **var *str6*** = **"ABC1.23"**;  </**script**>   <**script**>  ***window***.document.write(***str1***+**"：字符串转数字为："**+*parseInt*(***str1***)+**"<br/>"**);  ***window***.document.write(***str2***+**"：字符串转数字为："**+*parseInt*(***str2***)+**"<br/>"**);  ***window***.document.write(***str3***+**"：字符串转数字为："**+*parseInt*(***str3***)+**"<br/>"**);  ***window***.document.write(***str4***+**"：字符串转数字为："**+*parseInt*(***str4***)+**"<br/>"**);  ***window***.document.write(***str5***+**"：字符串转数字为："**+*parseInt*(***str5***)+**"<br/>"**);  ***window***.document.write(***str6***+**"：字符串转数字为："**+*parseInt*(***str6***)+**"<br/>"**);   </**script**> </**body**> </**html**> |

## 3.6在浏览器中的调试

几乎所有的浏览器都支持JS代码的调式，IE、Chrome、FireFox中调试的快捷键：F12。

### 3.6.1设置断点

在源代码左边的行号外，单击一下，会出现一个圆形的图标，这就是断点，程序运行时，

会在圆形的图标外停止下来。

|  |
| --- |
|  |

### 3.6.2查看错误

重新刷新页面才会在断点停下来，并显示出错信息。

|  |
| --- |
|  |

# 第04章 流程控制语句

高级语言中的三种基本结构： 顺序、判断、循环。

## 4.1 if判断

### 4.1.1 if语句

在一个指定的条件成立时执行代码。

|  |
| --- |
| if(条件表达式) {  //代码块;  } |

### 4.1.2 if..else语句

在指定的条件成立时执行代码，当条件不成立时执行另外的代码。

|  |
| --- |
| if(条件表达式) {  //代码块;  }  else {  //代码块;  } |

### 4.1.3 if..elseif..else语句

使用这个语句可以选择执行若干块代码中的一个。

|  |
| --- |
| if (条件表达式) {  //代码块;  }  else if(条件表达式) {  //代码块;  }  else {  //代码块;  } |

JS中，判断条件除了true/false外，还可以使用非布尔类型的结果。

|  |  |  |
| --- | --- | --- |
| **数据类型** | **为真** | **为假** |
| **number** | 非0 | **0** |
| **string** | 非空串 | **空串""** |
| **undefined** |  | **为假** |
| **NaN(Not a Number)** |  | **为假** |
| **object** | **非空** | **null** |

## 4.2 swtich多分支

### 4.2.1 case后使用变量

|  |
| --- |
| switch(变量名) {  case 常量值:  break;  case 常量值：  break;  default:  break;  } |

### 4.2.2 case后使用表达式

|  |
| --- |
| switch(true) { //这里的变量名写成true  case 表达式: //如：n > 5 && n < 10  break;  case 表达式:  break;  default:  break;  } |

## 4.3 案例：判断一个学生的等级

### 4.3.1案例需求

通过prompt()输入的分数，如果90~100之间，输出优秀。80~90之间输出良好。60~80输出及格。60以下输出不及格。其它分数输出:分数有误。

|  |  |
| --- | --- |
| **window对象的方法名** | **作用** |
| **string prompt("提示信息","默认值”)** | 出现一个给用户输入信息的对话框，可以指定默认值。返回字符串类型。不同浏览器样子有区别，不能修改。 |

### 4.3.2案例效果

|  |
| --- |
|  |

### 4.3.3案例分析

1\_使用prompt()得到输入的分数。

2\_如果分数是数字，则进行switch操作；如果分数不是数字，则提示用户。

3\_使用switch对分数进行判断。

4\_如果在90到100之间，则输出优秀，其它依次类推。

### 4.3.4案例代码

06\_ifSwitch.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>判断一个学生的等级</**title**> </**head**> <**body**>  <**script**>  *//提示用户输入分数：* **var *strScore*** = ***window***.prompt(**"你的分数："**,**"80"**);  *//判断分数是否为数字* **var *flag*** = *isNaN*(***strScore***);  *//如果是数字* **if**(!***flag***){  *//将字符串转为整数* **var *score*** = *parseFloat*(***strScore***);  **var *tip*** = **null**;  **switch**(**true**){  **case *score***>=0 && ***score***<60 : ***tip*** = **"不合格"**;**break**;  **case *score***>=60 && ***score***<70 : ***tip*** = **"勉强"**; **break**;  **case *score***>=70 && ***score***<80 : ***tip*** = **"良好"**;**break**;  **case *score***>=80 && ***score***<90 : ***tip*** = **"很好"**;**break**;*//break执行完后，退出switch代码块* **case *score***>=90 && ***score***<=100 : ***tip*** = **"优秀"**;**break**;  **default** : ***tip*** = **"数据有误"**; **break**;  }  ***window***.alert(**"你的分数是"**+***score***+**",评级为"**+***tip***);  }**else**{  ***window***.alert(**"分数得是number类型"**);  }  </**script**> </**body**> </**html**> |

## 4.4 循环

### 4.4.1 while语句

当指定的条件为 true 时循环执行代码。

最少执行0次循环。

|  |
| --- |
| while (条件表达式) {  需要执行的代码;  } |

### 4.4.2 do-while语句

当指定的条件为 true 时循环执行代码。

最少执行1次循环。

|  |
| --- |
| do {  需要执行的代码;  }while (条件表达式); |

### 4.4.3 for语句

循环指定次数。

最少执行0次循环。

|  |
| --- |
| for (var i=0; i<10; i++) {  需要执行的代码;  } |

### 4.4.4 break和continue语句

break: 结束循环，跳出while/do-while/for语法。

continue：跳过本次循环，执行下一次循环，没有跳出while/do-while/for语法。

## 4.5 案例：乘法表

### 4.5.1案例需求

以表格的方式输出乘法表行列，其中行数通过用户输入。

### 4.5.2案例效果

|  |
| --- |
|  |

### 4.5.3案例分析

1\_由用户输入乘法表的行数。

2\_使用循环嵌套的方式，每个外循环是一行tr，每个内循环是一个td。

3\_输出每个单元格中的计算公式。

### 4.5.4案例代码

07\_whileFor.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**>  <**head**>  <**meta charset="UTF-8"**>  <**title**>乘法表</**title**>  <**style**>  **td**{  **border-style**: **solid**;  **text-align**: **center**;  */\*background-color: aqua;\*/* }  </**style**> </**head**> <**body**>  <**script**>  *//提示用户输入行数* **var *strLine*** = ***window***.prompt(**"输入行数"**,**"9"**);  *//如果是数字* **if**(!*isNaN*(***strLine***)){  *//显示将string转成number* **var *line*** = *parseInt*(***strLine***);  ***document***.write(**"<table align='center' width='50%' cellspacing='4'>"**);  *//外循环控制行* **for**(**var *i***=1;***i***<=***line***;***i***++){  ***document***.write(**"<tr>"**);  *//内循环控制列* **for**(**var *j***=1;***j***<=***i***;***j***++){  ***document***.write(**"<td>"**);  ***document***.write(***i***+**"\*"**+***j***+**"="**+(***i***\****j***));  ***document***.write(**"</td>"**);  }  ***document***.write(**"</tr>"**);  }  ***document***.write(**"</table>"**);  }**else**{  ***window***.alert(**"请输入正确的数字"**);  }  </**script**> </**body**> </**html**> |

# 第05章 函数的使用

## 5.1函数的基本概述

函数function（小写字母）是当它被调用时可重复使用的代码块，JS中的函数类似于Java中的方法。

JS中函数分为： 有名/命名函数和无名/匿名函数。

有名/命名函数的格式：

|  |
| --- |
| **function 函数名( [参数列表] ) {  函数体;  [return 返回值]; }** |

[]表示可选。

无名/匿名函数的格式：

|  |
| --- |
| **var 变量名 = function([参数列表]) {  函数体; }** |

参数列表：add(num1,num2)是对的。

add(var num1,var num2)是错的，无需声明。

## 5.2案例：有名/命名函数

### 5.2.1案例需求

定义一个函数实现加法功能。

### 5.2.2案例效果

|  |
| --- |
|  |

### 5.2.3案例分析

1\_定义一个有名函数叫add，接收num1和num2这二个形参，函数体中将

num1和num2这二个形参相加后的和，通过return语句返回。

2\_调用有名函数add，并传入二个实参，将结果显示在浏览器中。

### 5.2.4案例代码

08\_addFunction.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>有名/命名函数</**title**> </**head**> <**body**>  <**script**>  *//函数定义，不调用函数，是不会执行的* **function** *add*(num1,num2){  **return** num1 + num2;  }  </**script**>  <**script**>  **var *num1*** = 10;  **var *num2*** = 100;  *//函数调用* ***window***.document.write(***num1*** + **"+"** + ***num2*** + **"="** + *add*(***num1***,***num2***));  </**script**> </**body**> </**html**> |

注意：

1\_形参的类型不用指定，因为是弱类型，前面var省略。

2\_函数的返回值，如果有返回值就使用return返回，如果没有返回值不写return。

## 5.3案例：无名/匿名函数

### 5.3.1案例需求

定义一个函数实现加法功能。

### 5.3.2\_案例效果

|  |
| --- |
|  |

### 5.3.3案例分析

1\_定义一个无名函数叫add，接收num1和num2这二个形参，函数体中将

num1和num2这二个形参相加后的和，通过return语句返回。

2\_调用有名函数add，并传入二个实参，将结果显示在浏览器中。

### 5.3.4案例代码

09\_addFunction.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>无名/匿名函数</**title**> </**head**> <**body**>  <**script**>  //add是函数名  **var** *add* = **function**(num1,num2){  **return** num1 + num2;  }  **var *num1*** = 100;  **var *num2*** = 1000;  *//函数调用* ***window***.document.write(***num1*** + **"+"** + ***num2*** + **"="** + *add*(***num1***,***num2***));  </**script**> </**body**> </**html**> |

## 5.4变量的作用域

### 5.4.1局部 JavaScript 变量

函数内部声明的变量（使用 var）是局部变量，所以只能在函数内部访问它。（该变量的作用域是局部的）。您可以在不同的函数中使用名称相同的局部变量，因为只有声明过该变量的函数才能识别出该变量。只要函数运行完毕，局部变量就会被删除。

|  |
| --- |
| <script>  function add1() {  var num = 5;//num在add1中有效，add1执行完毕，num变量消失，局部变量  }  function add2() {  var num = 5;//num在add2中有效，add2执行完毕，num变量消失  //add1和add2是不同的函数，可以定义相同名字的局部变量  }  document.write("num=" + num); //提示变量未定义，因为在add1和add2函数外，  //是不能访问num变量的。  </script> |

### 5.4.2全局JavaScript 变量

不是声明在函数体内部的变量，网页上的所有脚本和函数都能访问它。

|  |
| --- |
| <script>  var num = 5;//全局变量，生命周期为整个网页关闭  function add() {  document.write("num=" + num);//既可以在函数内部访问全局变量  }  document.write("num=" + num);//又可以在函数外部访问全局变量  </script> |

注意：当全局变量和局部变量同名时，在获取值时，采用就近原则。

### 5.4.3向未声明的 JavaScript 变量来分配值

如果您把值赋给尚未声明的变量，该变量将被自动作为全局变量声明。

如：name="传智播客"; //注：前面没有var

将声明一个全局变量，哪怕这个变量是声明在函数内部它也是一个全局变量。

10\_varScope.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>变量的作用域</**title**> </**head**> <**body**>  <**script**>  **function** *add*(){  **num** = 100;//相当于全局变量  ***document***.write(**"内部num="**+**num**+**"<br/>"**);//内部num=100  }   *add*();  ***document***.write(**"外部num="**+**num**+**"<br/>"**);外部num=100  </**script**> </**body**> </**html**> |

建议：局部变量声明在函数内，全局变量声明在函数外。

局部变量和全局变量声明时，都带var关键字。

## 5.5案例：实现轮播图

### 5.5.1 案例需求

实现每过3秒中切换一张图片的效果，一共3张图片，当显示到最后1张的时候，再次显示第1张。

|  |  |
| --- | --- |
| **使用到的方法** | **描述** |
| **document.getElementById("id")** | 通过标签的id属性得到唯一的元素 |
| **window.setInterval("函数名()"/无名函数，时间)** | 每隔一段时间调用一次函数，单位是毫秒 |

### 5.5.2 案例效果

|  |
| --- |
|  |

### 5.5.3 案例分析

1\_创建HTML页面，页面中有一个div标签，div标签内包含一个img标签。

2\_为div标签添加样式，align=”center”，为img标签添加样式，设置width和height属性。

3\_三张图片的名字依次是0~2.jpg，放在项目的img文件夹下，

图片一开始的src为第0张图片。

4\_编写函数：changePicture()，使用setInterval()函数，每过3秒调用一次。

5\_定义全局变量：num=1。

6\_在changePicture()方法中，设置图片的src属性为img/0~2.jpg。

7\_判断num是否等于4，如果等于4，则num=0；否则num++。

### 5.5.4 案例代码

11\_changePic.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>实现轮播图</**title**> </**head**> <**body**>  <**div align="center"**>  <**img id="pic" src="img/0.jpg" width="300px" height="300px"**>  </**div**>   <**script**>  **var *num*** = 1;   **function** *changePicture*(){  **var** img = ***document***.getElementById(**"pic"**);  img.**src** = **"img/"** + ***num*** + **".jpg"**;  **if**(***num*** == 2){  ***num*** = 0;  }**else**{  ***num***++;  }  }  //调用函数时：函数名()，是字符串类型，记住()。  ***window***.setInterval(**"***changePicture***()"**,3000);  </**script**> </**body**> </**html**> |

# 第06章 事件的处理

## 6.1事件的基本概述

在网页中用户操作的时候会激活各种事件，如按钮点击，鼠标移动，键盘按下等，这些都叫事件（Event）。我们可以通过编写代码对这些事件进行处理，如：当用户点击一个删除按钮时，我们给用户一些在删除之前的提示等等。

设置事件的两种方式：

方式一： 有名/命名函数

方式二： 无名/匿名函数

示例：

|  |  |
| --- | --- |
| **window对象的方法** | **作用** |
| **alert("提示信息")** | 用于弹出一个信息框 |

12\_setEvent.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>设置事件的两种方式</**title**> </**head**> <**body**>  <**input type="button" value="方式一：有名函数" onclick="***fun1*()**"**/>  <**input type="button" value="方式二：无名函数" id="btn2"**/>    <**script**>  **function** *fun1*(){  *alert*(**"方式一：有名函数"**);  }   **var *button2*** = ***document***.getElementById(**"btn2"**);  ***button2***.onclick = **function**(){  *alert*(**"方式二：无名函数"**);  }  </**script**> </**body**> </**html**> |

## 6.2常用的事件

### 6.2.1加载完成事件

onload：元素加载完毕以后执行，如：window.onload

示例：

浏览器加载完页面时，弹出警告框。

|  |
| --- |
|  |

13\_onload.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>加载完成事件</**title**> </**head**> <**body**>  <**script**>  ***window***.onload = **function**(){  *alert*(**"abc"**);  }  ***window***.onload = **function**(){  *alert*(**"ABC"**);  }  </**script**> </**body**> </**html**> |

### 6.2.2鼠标点击事件

onclick：单击事件

ondblclick：双击事件

示例：

在“姓名1”中输入完内容后，

单击按钮，会将“姓名1”的内容复制到“姓名2”中；

双击按钮，将将“姓名1”和“姓名2”中的内容清空。

|  |
| --- |
|  |

14\_onclickOndblclick.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>鼠标点击事件</**title**> </**head**> <**body**>  姓名1：<**input type="text" name="username" id = "u1"**/><**br**/>  姓名2：<**input type="text" name="username" id = "u2"**/><**br**/>  <**input type="button" value="单击复制/双击删除" id="btn"**/>   <**script**>  **var *button*** = ***document***.getElementById(**"btn"**);   ***button***.onclick = **function**(){  ***document***.getElementById(**"u2"**).**value** = ***document***.getElementById(**"u1"**).**value**;  };   ***button***.ondblclick = **function**(){  ***document***.getElementById(**"u1"**).**value** = **""**;  ***document***.getElementById(**"u2"**).**value** = **""**;  };  </**script**> </**body**> </**html**> |

### 6.2.3鼠标移动事件

onmouseover：鼠标在元素上面

onmouseout：鼠标离开元素

示例：

将鼠标移动到img上显示图片，移出则显示另一张图片。

|  |
| --- |
|  |

15\_onmouseoverOnmouseout.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>鼠标移动事件</**title**>  <**style type="text/css"**>  .**pic**{  **width**:300**px**;  **height**: 300**px**;  **border-style**: **solid**;  }  </**style**> </**head**> <**body**>  <**img class="pic" id="pic"**/>   <**script**>  **var *pic*** = ***document***.getElementById(**"pic"**);  ***pic***.onmouseover = **function**(){  ***pic***.**src** = **"img/1.jpg"**;  };  ***pic***.onmouseout = **function**(){  ***pic***.**src** = **"img/2.jpg"**;  };  </**script**> </**body**> </**html**> |

### 6.2.4焦点事件

什么是焦点： 当一个元素得到光标并且处于可以操作状态，得到焦点。

onblur：失去焦点

onfocus：得到焦点

示例：

当文本框获得到焦点时，文本框后面显示红色文字；

当文本框获失去焦点时，文本框后面的文字消失。

|  |
| --- |
|  |

16\_onblurOnfocus.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>焦点事件</**title**> </**head**> <**body**>  姓名：<**input type="text" id="username"**/><**span style="color**: **red" id="tip"**></**span**>   <**script**>  **var *input*** = ***document***.getElementById(**"username"**);  ***input***.onfocus = **function**(){  ***document***.getElementById(**"tip"**).innerHTML = **"&nbsp;&nbsp;&nbsp;&nbsp;请输入姓名"**;  };  ***input***.onblur = **function**(){  ***document***.getElementById(**"tip"**).innerHTML = **""**;  };  </**script**> </**body**> </**html**> |

### 6.2.5改变内容事件

onchange：改变事件

示例：

选中不同的城市出现显示信息框；

用户输入英文字母以后，文本框的字母全部变成大写，

注：文本框是要失去焦点以后才会激活改变事件。

|  |
| --- |
|  |

17\_onchange.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>改变内容事件</**title**> </**head**> <**body**>  <**select id="city"**>  <**option value="选择城市"**>选择城市</**option**>  <**option value="广州"**>广州</**option**>  <**option value="长沙"**>长沙</**option**>  <**option value="南宁"**>南宁</**option**>  </**select**>  <**hr**/>  英文字母：<**input type="text" id="english"**/>   <**script**>  **var *select*** = ***document***.getElementById(**"city"**);  ***select***.onchange = **function**(){  *alert*(**this**.**value**);  };    **var *input*** = ***document***.getElementById(**"english"**);  ***input***.onchange = **function**(){  **this**.**value** = **this**.**value**.toUpperCase();  };  </**script**> </**body**> </**html**> |

### 6.2.6键盘事件

onkeyup：键盘弹起

onkeydown：键盘按下

示例：

用户输入英文名后马上变成大写；

并且记录每个英文字母的Unicode码；

如果用户按了回车键，则用警告框显示已输入完的大写英文名。

|  |
| --- |
|  |

18\_onkeyupOnkeydown.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>键盘事件</**title**> </**head**> <**body**>  输入你的英文姓：<**input type="text" id="username"**/><**br**/>  对应的Code码为：<**input type="text" id="usernameCode"**/><**br**/>   <**script**>  **var *input*** = ***document***.getElementById(**"username"**);   ***input***.onkeyup = **function**(){  **var** code = ***event***.keyCode;  **if**(code != 13){  ***document***.getElementById(**"usernameCode"**).**value** += code;  }  **this**.**value** = **this**.**value**.toUpperCase();  }   ***input***.onkeydown = **function**(){  **var** code = ***event***.keyCode;  **if**(code == 13){  **var** username = ***document***.getElementById(**"username"**).**value**;  *alert*(username);  }  }  </**script**> </**body**> </**html**> |

|  |
| --- |
|  |

### 6.2.7小结

|  |  |
| --- | --- |
| **事件名** | **作用** |
| **onload** | **加载完毕** |
| **onclick** | **单击事件** |
| **ondblclick** | **双击事件** |
| **onfocus** | **得到焦点** |
| **onblur** | **失去焦点** |
| **onchange** | **改变事件** |
| **onmouseover** | **鼠标在上面** |
| **onmouseout** | **鼠标离开** |
| **onkeyup** | **键盘弹起** |
| **onkeydown** | **键盘按下** |
| **onsubmit** | **表单提交事件** |

# 第07章 JavaScript的内置对象

参考<<JavaScript 语言参考.chm>>

## 7.1数组对象

### 7.1.1数组的四种方式

|  |  |
| --- | --- |
| **创建数组的方式** | **说明** |
| **new Array()** | **创建0个长度的数组** |
| **new Array(5)** | **创建5个长度的数组** |
| **new Array(2,4,10,6,41)** | **指定每个元素创建数组** |
| **[4,3,20,6]** | **使用中括号指定每个元素** |

### 7.1.2数组的特点

1\_元素类型是可以变化的，每个元素类型可以不同。

2\_数组的长度是可变的。

3\_数组是有方法的。

### 7.1.3常用方法

|  |  |
| --- | --- |
| **方法名** | **功能** |
| **concat()** | **连接两个或更多的数组，并返回结果** |
| **reverse()** | **将数组进行反转** |
| **join(separator)** | **与split()功能相反，将数组通过分隔符，拼成一个字符串** |
| **sort()** | **对字符串数组进行排序，按字符串的unicode码**  **如果要对数字进行排序，要指定比较器函数**  **sort(function(m,n)) 数字两两比较**  **1) 如果m大于n，则返回正整数，return m-n 是升序**  **2) 如果m小于n，则返回负整数，return n-m 是降序**  **3) 如果m等于n，则返回0** |

### 7.1.4常用方法的示例代码

演示数组的常用方法

|  |
| --- |
|  |

19\_array.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>数组常用方法1</**title**> </**head**> <**body**>  <**script**>  **var *names*** = [**"A"**,**"B"**,**"C"**,**"D"**];  **for**(**var *i***=0;***i***<***names***.**length**;***i***++){  ***document***.write(***names***[***i***]+**"&nbsp;"**);  }  ***document***.write(**"<hr/>"**);  ***names***.length = 7;  **for**(**var *i***=0;***i***<***names***.**length**;***i***++){  ***document***.write(***names***[***i***]+**"&nbsp;"**);  }  ***document***.write(**"<hr/>"**);  ***names***.length = 4;  **for**(**var *i***=0;***i***<***names***.**length**;***i***++){  ***document***.write(***names***[***i***]+**"&nbsp;"**);  }  ***document***.write(**"<hr/>"**);  **var *scores*** = [10,20,30,40];  **var *news*** = ***names***.concat(***scores***);  **for**(**var *i***=0;***i***<***news***.length;***i***++){  ***document***.write(***news***[***i***]+**"&nbsp;"**);  }  ***document***.write(**"<hr/>"**);  ***news*** = ***news***.reverse();  **for**(**var *i***=0;***i***<***news***.length;***i***++){  ***document***.write(***news***[***i***]+**"&nbsp;"**);  }  ***document***.write(**"<hr/>"**);  </**script**> </**body**> </**html**> |

|  |
| --- |
|  |

20\_array.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>数组常用方法2</**title**> </**head**> <**body**>  <**script**>  **var *scores*** = [10,30,20,40];  ***scores*** = ***scores***.join(**"#"**);  ***document***.write(***scores***);  ***document***.write(**"<hr/>"**);   ***scores*** = ***scores***.split(**"#"**);  **for** (**var *i* in *scores***) {  ***document***.write(***scores***[***i***]+**"&nbsp;"**);  }  ***document***.write(**"<hr/>"**);   *//升序* ***scores*** = ***scores***.sort(**function**(m,n){  **return** m-n;  });  **for** (**var *i* in *scores***) {  ***document***.write(***scores***[***i***]+**"&nbsp;"**);  }  ***document***.write(**"<hr/>"**);   *//降序* ***scores*** = ***scores***.sort(**function**(m,n){  **return** n-m;  });  **for** (**var *i* in *scores***) {  ***document***.write(***scores***[***i***]+**"&nbsp;"**);  }  ***document***.write(**"<hr/>"**);  </**script**> </**body**> </**html**> |

21\_date.html

|  |
| --- |
| <!DOCTYPE **html**> <**html lang="en"**> <**head**>  <**meta charset="UTF-8"**>  <**title**>演示JS内置对象Date</**title**> </**head**> <**body**>  <**script**>  *//创建当前时间对象* **var *now*** = **new *Date***();   *//分别获取当前时间：年-月-日* **var *year*** = ***now***.getFullYear();  **var *month*** = ***now***.getMonth()+1;  **var *date*** = ***now***.getDate();  *//alert(year+"-"+month+"-"+date);   //分别获取当前时间：时-分-秒* **var *h*** = ***now***.getHours();  **var *m*** = ***now***.getMinutes();  **var *s*** = ***now***.getSeconds();  *//alert(h+":"+m+":"+s);* **var *day*** = ***now***.getDay();  *alert*(***day***);     </**script**> </**body**> </**html**> |