# word

# JavaScript简介JS

## 1.1 什么是JavaScript

* javaScript 是因特网上最流行的**脚本语言**，它存在于全世界所有 Web 浏览器中，能够增强用户与 Web 站点和 Web 应用程序之间的交互。
* 脚本语言：脚本语言又被称为扩建的语言，或者动态语言，是一种解释型语言，由解释器来运行，用来控制软件应用程序，脚本通常以文本保存，只在**被调用时进行解释或编译**
* JavaScript 是可插入 HTML 页面的编程代码。
* JavaScript 插入 HTML 页面后，可由所有的现代浏览器执行。
* JavaScript 与 Java 是两种完全不同的语言

## 1.2 JavaScript作用

* 我们通过javascript可以改变html内容,改变html样式,进行验证输入等。

## 1.3 JavaScript历史

* Netscape（网景） 发明了 JavaScript
* 1997年由来自Netscape、Sun、微软、Borland 和其他一些对脚本编程感兴趣的公司的程序员组成的 TC39 锤炼出了 ECMA-262，该标准定义了名为 ECMAScript 的全新脚本语言。
* 从此，Web 浏览器就开始努力（虽然有着不同的程度的成功和失败）将 **ECMAScript 作为 JavaScript 实现的基础。**
* ECMAScript 仅仅是一个描述，定义了脚本语言的所有属性、方法和对象。其他语言可以实现 ECMAScript 来作为功能的基准，JavaScript 就是这样：

|  |
| --- |
| ECMAScript、JavaScript、ActionScript、ScriptEase |

## 1.4 JavaScript组成

* javascript的核心ECMAScript描述了语言的语法和基本对象。
* 一个完整的javaScript实现是由下面**三个不同部分**组成的。

|  |
| --- |
|  |

# JavaScript的使用

## 2.1 导入js方式

### 第一种：

* 在html页面中直接插入javascript
* 如需在 HTML 页面中插入 JavaScript，请使用 <script> 标签。

例如:

|  |
| --- |
|  |

* **那些老旧的实例可能会在 <script> 标签中使用 type="text/javascript"。现在已经不必这样做了**。
* JavaScript 是所有现代浏览器以及 HTML5 中的默认脚本语言。
* 注意:<script>标签可以出现在html页面的任意位置，但实际操作中，一般会写在<head>标签中。

### 第二种：

* **引入外部的javascript**
* 有时，你也许希望在若干个页面中运行 JavaScript，同时不在每个页面中写相同的脚本。
* 为了达到这个目的，你可以将 JavaScript 写入一个外部文件之中。然后以 .js 为后缀保存这个文件。

例如:

|  |
| --- |
|  |

* 注意:
* 外部的js文件中不能包含<script>标签。
* 在导入外部js文件的<script>标签中不能在写javascript代码。

|  |
| --- |
|  |

## 2.2 Javascript基础语法

### 变量声明

* 变量是用于存储信息的容器
* javascript中的变量与我们在java中的变量不一样。
* 在javascript中使用 **var 运算符（variable 的缩写）加变量名定义的**
* 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
* 注意：由于 **JavaScript 对大小写敏感**，变量名也对大小写敏感。

Int i=10;

var i = 10;

### 数据类型

在ECMAScript中，变量可以存在两种数据类型，即**原始类型**与**引用类型**。

原始类型 相当于 Java基本数据类型

#### 原始类型

* 原始类型，可以理解成是一个**值类型**。简单说，就是一个**字面值**，它是不可变的，例如: 10 “abc”
* ECMAScript有五种原始类型
* **String** 在javascript中字符串字符串字面值，可以使用单引号或双引号声明。
* **Number** 在javascript中任何数字都看成是Number类型，它即可以表示32位整数，也可以表示64位浮点数。八进制首数字必须是0,十六进制使用0x开始。
* **Boolean** 它有两个值true和false.
* **Undefined**:该类型只有一个值undefined.表示的是未初始化的变量
* **Null** 该类型只有一个值null,表示尚未存在的对象。**值undefined实际上是从值null派生出来的。因此null==undefined得到的结果是true.**
* **通过typeof方法可以获取一个变量的类型**

|  |
| --- |
|  |

|  |
| --- |
| **注意：您也许会问，为什么 typeof 运算符对于 null 值会返回 "Object"。这实际上是 JavaScript 最初实现中的一个错误，然后被 ECMAScript 沿用了。现在，null 被认为是对象的占位符，从而解释了这一矛盾，但从技术上来说，它仍然是原始值。** |

#### 引用类型

* 引用类型通常叫做**类(class)**,也就是说，遇到引用值，所**处理的就是对象**。
* 从传统意义上来说，ECMAScript 并**不真正具有类**。
* ECMAScript 定义了“对象定义”，逻辑上等价于其他程序设计语言中的类。
* 对象是由 new 运算符加上要实例化的对象的名字创建的var obj=new Object();
* Object对象自身用处不大，但是 ECMAScript 中的 Object 对象与 Java 中的 java.lang.Object 相似，ECMAScript 中的所有对象都由这个对象继承而来，Object 对象中的所有属性和方法都会出现在其他对象中
* 常用的javascript对象有 **Boolean,Number,Array,String,Date,Math,RegExp**.
* 可以使用**instanceof**运算符来判断对象的类型。

### 类型转换

* ECMAScript为开发者提供了大量的类型转换方法。
* 大部分类型具有进行简单转换的方法，还有几个全局方法可以用于更复杂的转换。
* Boolean 值、数字和字符串的原始值它们是**伪对象**，这意味着它们实际上**具有属性和方法**。

#### 转换成字符串

toString()方法.把**number转成string,boolean转string**

|  |
| --- |
|  |

#### 转换成数字

parseInt() 和 parseFloat()。前者把值转换成整数，后者把值转换成浮点数。

**只有对 String 类型调用这些方法**，它们才能正确运行；对其他类型返回的都是 **NaN**。

|  |
| --- |
|  |

#### 强制类型转换

* ECMAScript 中提供了三种强制类型转换：

1.把给定的值转换成Boolean类型

2.把给定的值转换成数字

3.把给定的值转换成字符串

|  |  |
| --- | --- |
|  |  |
|  |  |

### 运算符和表达式

JavaScript的运算符和表达式与Java 大部份相似

#### (1)赋值运算符

var x = 5;

#### (2)算数运算符

+ - \* / %

+: 遇到字符串变成连接

-：先把字符串转成数字然后进行运算

\*: 先把字符串转成数字然后进行运算

/: 先把字符串转成数字然后进行运算

|  |
| --- |
|  |

#### (3)逻辑运算符

&& ||

#### (4)比较运算符

< > >= <= != ==

**===**:全等：类型与值都要相等

|  |
| --- |
|  |

#### (5)三元运算符

3<2?"大于":"小于"

#### (6)void运算符

<a href="javascript:void(0);">xxxxxx</a>

#### (7)类型运算符

typeof:判断数据类型 返回我的数据类型

instanceof：判断数据类型 是否是某种类型

var obj = new Object();

alert(typeof obj);//object

alert(obj instanceof Object);//true

### 逻辑语句

* JS的逻辑语句也跟Java 大部份相似

#### if-else

|  |
| --- |
|  |
|  |

#### Switch

|  |
| --- |
|  |

#### For

|  |
| --- |
|  |

#### for in

|  |
| --- |
|  |

# javascript常用对象

## 3.1 Boolean

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。

语法：

|  |
| --- |
| * new Boolean(value); //构造函数 * Boolean(value); //转换函数 * 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"）。

## 3.2 Number

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**。

语法：

|  |
| --- |
| var myNum=new Number(value);  var myNum=Number(value); |
|  |

## 3.3 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); |
|  |
|  |

## 3.4 String

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。

创建 String 对象的语法：

|  |
| --- |
| new String(s);  String(s); |
|  |
|  |

## 3.5 Date

Date 对象用于处理日期和时间。

创建 Date 对象的语法：

|  |
| --- |
| var myDate=new Date() |
|  |
| 案例： |

## 3.6 Math

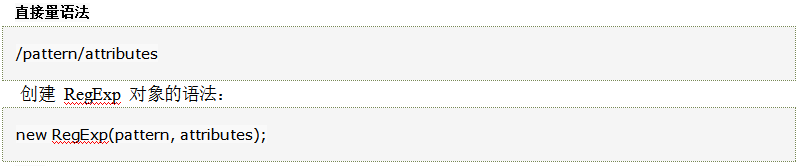
Math 对象用于执行数学任务。

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。 |
| **案例：** |
|  |

## 3.7 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。



|  |
| --- |
| 创建方式：  var reg = new RegExp(pattern);  var reg = **/^正则规则$/;**  规则的写法：  [0-9]  [A-Z]  [a-z]  [A-z]  \d 代表数据  \D 非数字  \w 查找单词字符  \W 查找非单词字符  \s 查找空白字符  \S 查找非空白字符  n+ 出现至少一次  n\* 出现0次或多次  n? 出现0次或1次  {5} 出现5  {2,8} 2到8次  方法：  test(str):检索字符串中指定的值。返回 true 或 false  需求：  校验邮箱：  //不能有中划线  **var** email = "78\_1234@qq.com"  **var** reg = /^[A-z0-9\_]+@[A-z0-9]+\.[A-z]+$/  console.log(reg.test(email)); |

**重点掌握**

数组的创建与常用方法

String中常用方法

Javascript中正则的使用

# javascript 函数和事件

函数就是方法

## 4.1创建函数

在ECMAScript中函数有以下几种创建方式

|  |
| --- |
| **第一种** |
| function 函数名称(参数列表){  函数体;  return 返回值  } |
| **第二种** |
| var 函数名称=function(参数列表){  函数体;  } |
| 第三种：【了解，不用】  Var 函数名称=new Function(参数列表,函数体); |
| 案例： |

## 4.2 函数注意事项：

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。

|  |
| --- |
|  |

## 4.3 全局函数介绍

|  |
| --- |
|  |
| 案例： |

## 4.4事件介绍

**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.

**常见事件:**

|  |
| --- |
|  |

## 4.5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。

事件的三要素：事件、事件源、响应行为

#### onclick事件

|  |
| --- |
|  |

#### Onchange事件

|  |
| --- |
|  |
|  |

#### onfoucus:获得焦点的事件

#### onblur:失去焦点的事件

|  |
| --- |
|  |

#### onmouseover:鼠标悬浮的事件

#### onmouseout:鼠标离开的事件

|  |
| --- |
|  |

#### onload:加载完毕的事件【注意】

等到页面加载完毕在执行onload事件所指向的函数

|  |
| --- |
|  |

#### this关键字

|  |
| --- |
|  |

### 阻止事件默认行为

不让a标签跳转,IE可以选择IE的版本来测试,如果不想跳转，可直接在onclick 写return false

|  |
| --- |
| <!-- event:固定事件 -->  <a href=*"http://www.baidu.com"* onclick="fun(event)">点击我</a>  <script type=*"text/javascript"*>  ////阻止事件的默认行为  //IE web技术：window.event.returnValue = false;  //W3c web技术: 传递过来的事件对象.preventDefault();  **function** fun(e) {  **if**(e && e.preventDefault){  alert("w3c");  e.preventDefault();  }**else**{  alert("IE");  window.event.returnValue = **false**;  }  }  </script> |
|  |

### 阻止事件的传播

IE：window.event.cancelBubble = true;

W3c: 传递过来的事件对象.stopPropagation();

|  |
| --- |
| <script type=*"text/javascript"*>  //阻止事件传递,黄色的不传递给红色的  **function** fun1(e) {  alert("yellow");  **if**(e && e.stopPropagation){  alert("w3c");  e.stopPropagation();  }**else**{  alert("IE");  window.event.cancelBubble = **true**;  }  }  **function** fun2() {  alert("red");  }  </script>  <div style="width: *200px*;height: *200px*;background-color: *red*" onclick="fun2()">  <div style="width: *100px*;height: *100px*;background-color: *yellow*" onclick="fun1(event)"></div>  </div> |

# javascript BOM

## 5.1 Javascript中bom介绍

* IE 3.0 和 Netscape Navigator 3.0 提供了一种特性 - **BOM（浏览器对象模型**），**可以对浏览器窗口进行访问和操作**。
* 使用 BOM，开发者可以移动窗口、改变状态栏中的文本以及执行其他与页面内容不直接相关的动作。
* 使 BOM 独树一帜且又常常令人怀疑的地方在于，它只是 JavaScript 的一个部分，没有任何相关的标准。
* BOM 主要处理浏览器窗口和框架，不过通常**浏览器特定的 JavaScript 扩展都被看做 BOM 的一部分**。这些扩展包括：
* 弹出新的浏览器窗口
* 移动、关闭浏览器窗口以及调整窗口大小
* 提供 Web 浏览器详细信息的定位对象
* 提供用户屏幕分辨率详细信息的屏幕对象
* 对 cookie 的支持
* IE 扩展了 BOM，加入了 ActiveXObject 类，可以通过 JavaScript 实例化 ActiveX 对象
* 由于没有相关的 BOM 标准，每种浏览器都有自己的 BOM 实现。**有一些事实上的标准**，如具有一个窗口对象和一个导航对象，不过每种浏览器可以为这些对象或其他对象定义自己的属性和方法。



## 5.2 Window对象

* Window 对象表示浏览器中打开的窗口
* 如果文档包含框架（frame 或 iframe 标签），浏览器会为 HTML 文档创建一个 window 对象，并为每个框架创建一个额外的 window 对象

### window对象集合

|  |
| --- |
|  |

### window对象属性

|  |
| --- |
|  |

### window对象方法

|  |
| --- |
|  |

### 案例：

|  |
| --- |
| **window对弹框的三个方法,alert,confirm,prompt** |
| <script type=*"text/javascript"*>  **function** click1() {  //alert是window对象的方法,平时开如可以省略window  window.alert("党的十九大招开了");//提示框  }    **function** click2() {  //显示带有一段消息以及确认按钮和取消按钮的对话框。  //用户选择后，会返回true或者false  **var** rsl = window.confirm("你确定要分手吗？");  alert(rsl);  }    **function** click3(){  //显示可提示用户输入的对话框,点确定后，返回用户输入的密码，点取消，返回null  **var** rsl = prompt("请输入你的银行卡密码");  alert("你输入的密码是:" + rsl)  }  </script>  <button onclick="click1();">点击1</button>  <button onclick="click2();">点击2</button>  <button onclick="click3();">点击3</button> |
| **open方法** |
|  |
| 定时器(**指定时间执行任务**) |
|  |
| **定时器(每隔一定时间执行任务)** |
|  |
| **需求：注册后5秒钟跳转首页** |
|  |
| **history的go,back,forword函数** |
|  |

# javascript DOM介绍

## 6.1 DOM介绍

**DOM（文档对象模型）是 HTML 和 XML 的应用程序接口**（API）。DOM 将把整个页面规划成由节点层级构成的文档。HTML 或 XML 页面的每个部分都是一个节点的衍生物。

请看下面的 HTML 页面



这段代码可以用 DOM 绘制成一个**节点层次图**：

|  |
| --- |
|  |

**DOM 通过创建树来表示文档**，从而使开发者对文档的内容和结构具有空前的控制力。

W3C DOM 标准被分为 3 个不同的部分：

* 核心 DOM - 针对任何结构化文档的标准模型
* XML DOM - 针对 XML 文档的标准模型
* HTML DOM - 针对 HTML 文档的标准模型
* XML DOM 定义了所有 XML 元素的对象和属性，以及访问它们的方法。
* HTML DOM 是：
* HTML 的标准对象模型
* HTML 的标准编程接口
* W3C 标准
* HTML DOM 定义了所有 HTML 元素的对象和属性，以及访问它们的方法。

|  |
| --- |
|  |
| <a>百度</a> |

## 6.2 XML DOM与HTML DOM的关系

|  |
| --- |
| XML DOM 定义了**访问和处理 XML 文档的标准方法**  HTML**文档格式 符合XML语法标准**，所以可以使用XML DOM API  在XML DOM每个元素 都会被解析为一个节点Node，而常用的节点类型又分为  元素节点 Element  属性节点 Attr  文本节点 Text  文档节点 Document  HTML DOM 定义了针对 HTML文档的对象，可以说是**一套 更加适用于 JavaScript 技术开发 的API**  **HTML DOM是对XML DOM的扩展**  进行 JavaScript DOM开发 可以同时使用 XML DOM和 HTML DOM |

## 6.3 getElementById() 方法

|  |
| --- |
| getElementById() 方法 |
|  |

## 6.4 getElementsByTagName方法

|  |
| --- |
|  |

## 6.5 getElementsByName() 方法

|  |
| --- |
|  |

## 6.6 hasChildNodes()

判断是否有子节点

|  |
| --- |
|  |

## 6.7 firstChild/lastChild

注意：子标签之间不要换行或空格，

|  |
| --- |
|  |

## 6.8 replaceChild

|  |
| --- |
|  |

## 6.9 getAttribute/setAttribute

获取、设置标签中属性的值

|  |
| --- |
|  |

## 6.10 节点地增删改查

document.createElement

document.createTextNode

appendChild

insertBefore

insertAfter 方法，js没有这个方法，insertAfter 的jquery有

removeChild

|  |
| --- |
| 喜欢的城市:<br>  <ul id=*"city"*>  <li id=*"bj"*>北京</li>  <li id=*"sh"*>上海</li>  <li id=*"gz"*>广州</li>  </ul>  <button id=*"btn"* onclick="myClick()">点击</button>  <script type=*"text/javascript"*>  **function** myClick(){  //创建节点  **var** li = document.createElement("li");  li.setAttribute("id","sz");    //创建一个文本  **var** txt = document.createTextNode("深圳");  li.appendChild(txt);    //获取ul  **var** ul = document.getElementById("city");  //ul.appendChild(li);//尾部添加    **var** sh = document.getElementById("sh");  //ul.insertBefore(li,sh);//插入到上海前面  //ul.insertAfter(li,sh);//插入到上海后面  //ul.removeChild(sh);//删除上海  }  </script> |

## 6.11 innerHTML

div.innerHTML = "<h1>今天</h1>"