## Javascript介绍

JavaScript是一种是运行在客户端的、面向对象的、单线程脚本语言。它是一种弱类型的、解释型语言。

Netscape 发明了 JavaScript

## Javascript作用

1. 通过javascript对html代码内容进行修改，对html中的css样式进行修改。
2. 为web页面添加交互行为，提高用户体验：Tab栏切换、地图
3. 制作页面特效：各种炫酷的动画、轮播图
4. 表单验证：提高响应速度（Ajax），减轻服务器的压力
5. 跨平台运行
6. **JavaScript**
7. **ECMAScript**
8. BOM
9. DOM

ECMA 国际，前身是欧洲计算机制造协会，ECMAScript 是其发布的一套语法标准，因为除了JavaScript，还有其他公司开发的很多某某Script，而ECMAScript 统一规定了一系列的标准语法，比如变量如何声明、关键字、保留字、运算符、对象、函数。

## Html与javascript结合方式

需求：

1. 行间

<input type=”button” onclick=”alert(1);” value=”Click”/>

1. 内嵌：在html页面上任意位置写上<script>标签。
2. 外链js文件

<script src=”demo.js”></script>

## Javascript基本语法

1. 严格区分大小写。
2. 是一种弱类型语言。
3. 一行的结束用分号，如果不使用，默认是换行做为结束。
4. 注释：//单行注释 /\* \*/多行注释

## 变量和数据类型

变量声明用**var**，变量名可以包含数字**字母**下划线，但是不能以数字开头。

|  |  |  |
| --- | --- | --- |
| **数据类型** | | |
| 简单数据类型 | | 复杂数据类型 |
| string | 字符串 | Array |
| number | 数字（NaN\*） | Function |
| boolean | 布尔 | Date |
| undefined |  | Math |
| null |  | RegExp |

|  |  |  |
| --- | --- | --- |
| 运算符 | | |
| 数学运算符 | + | 拼接字符串 |
| - \* / % ++ -- |  |
| 比较运算符 | > < >= <= |  |
| == === |  |
| 赋值运算符 | = |  |
| 逻辑运算符 | && | 一flase皆flase |
| || | 一true皆true |
| ! |  |
| 三元运算符 | 值?表达式1:表达式2 |  |

|  |  |  |
| --- | --- | --- |
| 流程控制 | | |
| 条件判断 | if ( ) { }  else if ( ) { }  else if ( ) { }  else { } | if(2) { } |
| switch ( ) {  case 常量1: ... break;  case 常量2: ... break;  default: ... } |  |
| 循环 | do { ... } while ( ); | 至少执行一次 |
| while ( ) { } |  |
| for( ) { } |  |

在java中if语句后面的条件必须是true/false，而在javascript 中我们描述true/false有多种方式，其中包含了隐式转换。

|  |  |
| --- | --- |
| 转成true | 转成false |
|  | Undefined |
| object | null |
| 非0数字 | 0 |
| 非空字符串 | 空字符串 |

## Javascript常用对象

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| String | Math | Date | Array | RegExp |

### String对象

常用属性：length

eg: Your girlfriend two-timed you.

|  |  |  |
| --- | --- | --- |
|  | **返回值** | **示例** |
| **concat** | 连接两个或多个字符串，类似 + | str1.concat(str2) |
| **indexOf** | 某个指定的字符串值在字符串中首次出现的位置 | a.indexOf(“b”) |
| **[lastIndexOf](mk:@MSITStore:E:\\工具书\\w3c手册.chm::/www.w3school.com.cn/js/jsref_lastIndexOf.asp.htm)**[()](mk:@MSITStore:E:\\工具书\\w3c手册.chm::/www.w3school.com.cn/js/jsref_lastIndexOf.asp.htm) | 一个指定的字符串值最后出现的位置 | a.lastIndexOf(“a”) |
| charAt(index) | 返回指定索引处的字符 |  |
| split( , ) | 把一个字符串分割成字符串数组，并截取前面若干项 | a.split(“ ”, 3) |
| **slice**(start,end) | 一个新的字符串。从start开始（包括）到end结束（不包括）为止的所有字符 | a.slice(1,3) |
| replace | 每次只替换一个，全部替换完要循环 |  |

### Math对象

|  |  |  |  |
| --- | --- | --- | --- |
| abs() | ceil() | floor() | random() |
| max( , ) | min( , ) | round() | sqrt() |

### Date对象

eg: travel back

|  |  |
| --- | --- |
| getFullYear() | setFullYear() |
| getMonth() | setMonth() |
| getDate() | setDate() |
| getHours() | setHours() |
| getMinutes() | setMinutes() |
| getSeconds() | setSeconds() |
| getTime() | toLocal**e**String() |

### Array对象

创建数组

var arr1 = new Array();

var arr2 = new Array(4);

var arr3 = new Array("a", "b");

var arr4 = ["d","f","a","t"];

Java中的数组特点：长度固定，数组中元素类型是单一、有顺序。

JavaScript的数组元素可以是任意类型，长度可以改变。

属性：length

|  |  |  |  |
| --- | --- | --- | --- |
| **Method** | | | |
|  |  | 返回值 | 示例 |
| join(分隔符) | 把数组的所有元素（忽略空格）放入一个字符串。元素通过指定的分隔符进行分隔 |  | arr.join("-") |
| **indexOf**(searchvalue,fromindex) | 返回指定值在数组中首次出现的位置 |  |  |
| **lastIndexOf**() |  |  |  |
| arr instanceof Array | 判断arr是不是数组类型 | true / false |  |
| Array.isArray(arr) |  |
| reverse() | 颠倒数组中元素的顺序 | 改变原来的数组，而不会创建新的数组 |  |
| **concat**() | 连接两个或更多的数组。  不改变原有数组 | 返回被连接数组的副本 | arr1.concat(arr2)  arr.concat(1); |
| sort(fn) | 对数组的元素进行排序 |  |  |
| push() | 向数组末尾添加元素 | 新的长度 |  |
| pop() | 删除最后一个元素 | 最后一个元素 |  |
| unshift() | 从前面加入元素 | 返回数组的长度 |  |
| shift() | 从前面删除元素 | 返回被删除的元素 |  |
| **slice**(start, end) | 从已有的数组中返回选定的元素，原数组保持不变 | 返回切出来的新数组 |  |

### RegExp对象：找什么，找多少，从哪里找

创建正则对象的方法：

|  |  |
| --- | --- |
| 1. **直接量**语法 | 1. new |
| var reg=/正则表达式/ | var reg=new RegExp(“正则表达式”); |

|  |  |  |
| --- | --- | --- |
|  |  | **含义** |
| 基本元字符 | . | 表示任一个非换行的字符 |
| () | 表示分组和提高优先级 |
| [] | 表示一个字符，如 [abc] 匹配a/b/c |
| | | 或（优先级最低）用法：正则表达式 | 正则表达式 |
| 次数限定元字符 | \* | 紧挨前面的一个字符或一组字符出现 0 次到多次 |
| + | 紧挨它前面的字符出现 1 次到多次 |
| ？ | 紧挨它前面的字符出现 0 次或1次 |
| {n} | 紧挨它前面的字符出现至少n次 |
| {n, m} | 紧挨它前面的字符出现n~m次 |
| 首尾元字符 | ˆ | 以某字符开头 |
| $ | 以某字符结尾 |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **单词** | **代表** | **含义** |
| \d | digit | [0-9] | 一个数字 |
| \D |  | [^0-9] | 非数字字符 |
| \w | word | [a-zA-Z0-9\_] | 一个单词字符(所有的数字、字母、下划线) |
| \W |  | [^a-zA-Z0-9\_] | 非单词字符 |
| \s | space | [\f\r\n\t\v] | 不可见字符：空格 / tab / enter |
| \S |  | [^\f\r\n\t\v] | 可见字符 |

匹配电话号码：1[3|5|7|8|]\d{9}

实现手机号前带86或+86的情况：^((\+86)|(86))?(13)\d{9}$

匹配邮箱：\w+@([0-9a-zA-Z]+[-0-9a-zA-Z]\*)(\.[0-9a-zA-Z]+[-0-9a-zA-Z]\*)+

### 函数

函数就是方法，是完成某个特定功能的代码块。对于函数必须调用它才可以执行。

|  |  |  |
| --- | --- | --- |
| 函数声明式 | 函数表达式 | 构造函数 |
| function fn () {  代码段；} | var fn = function() {  代码段 }**;** | var fn = new Function(参数，函数体); |

#### 关于参数：

1. 在js中，函数的参数是无数据类型的；
2. 函数在调用时，传入的参数的个数和类型不是必须指定的；
3. 函数内有一个arguments对象用来接收所有参数，它是一个数组；
4. JS没有重载，只要函数名相同，就认为是同一个函数，如果写了多个，后面的就会把前面的覆盖掉。

#### return的作用：

1. return后面有值：返回函数的结果（即返回值）；
2. return; 返回undefined，函数运行结束；
3. return不出现，默认返回undefined.

JS中的全局函数

|  |  |
| --- | --- |
| encodeURI() |  |
| encodeURIComponent() |  |
| decodeURI() |  |
| decodeURIComponent() |  |
| isNaN() | 用于检查其参数是否是非数字值 |
| Number() |  |
| String() |  |
| parseInt() |  |
| parseFloat() |  |

URI (uniform resource identifier)，统一资源标识符，用来唯一的标识一个资源。Web上可用的每种资源都由一个URI来进行定位。

URL (uniform resource locator)，统一资源定位器，是一种具体的URI，它不仅唯一标识资源，而且还提供了定位该资源的信息。



encodeURI()它不编码的有82个.

encodeURIComponent()它不编码的有71个。