# JavaScript

干什么：

可以与用户进行交互

学什么：

学习js的各种语法

## 1.JavaScript概述

### 1.1.JavaScript是干什么的

1. HTML 定义了网页的内容

2. CSS 描述了网页的布局

3. JavaScript 指定了网页的行为

### 1.2.JavaScript历史

1995年5月，网景公司，叫Brendan Eich的哥们，10天，LiveScript

1995年12月，改名为JavaScript

1996年8月，微软，Jscript

1997-1999年，ECMA国际组织，ECMAScript，基于已有的JavaScript和Jscript，提出了标准的Script语法规则，JavaScript和Jscript都遵循这套标准。

1999-以后，ECMAScript不断的更新

### 1.3.JavaScript特点

JavaScript是脚本语言，不需要编译，解释运行

JavaScript可以插入html页面的

JavaScript是弱类型语言

JavaScript很容易学习

### 1.4.JavaScript优点

交互性：可以与用户进行动态交互

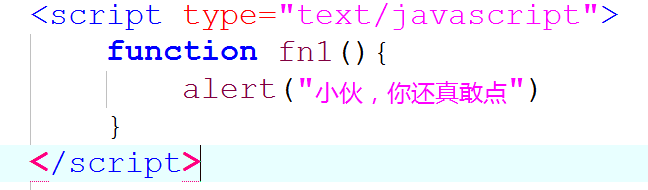
安全性：只能在浏览器内部运行，不能访问浏览器之外的资源

跨平台：有浏览器即可运行，与平台无关

## 2.在html中引入JavaScript（案例1）

### 2.1.通过script标签引入

在html中可以在script标签内写入js代码，次标签通常写到head中



### 2.2.引入外部的js文件

如果js代码特别多，可以将js代码抽取到js文件中，在html中通过script标签引入外部的js文件

注意：1.此script标签不要自闭

2.此script标签内不要添加任何其他代码



## 3.JavaScript语法

### 3.1.注释

单行注释： //单行注释

多行注释： /\* 多行注释 \*/

### 3.2.数据类型

#### 3.2.1.基本数据类型

1)数值类型（number）

2)字符串型（string）

3)布尔类型（boolean）

4)undifined类型

5)null类型

**（1）数值类型**

在js中, 所有的数值底层都是浮点型¸ 在需要时, 整型和浮点型会自动的进行类型的转换.

例如：2.4+3.6=6

特殊值：

Infinity 正无穷大

-Infinity 负无穷大

NaN (not a number) 非数字 ：NaN和任何数值都不相等, 包括它本身. 如果要判断一个值是否是非数字, 不能用 NaN == xx 来判断, 而是应该使用 isNaN(xx) 进行判断。

在js中, 为number类型提供了对应的包装对象 – Number。

**（2）字符串型**

在js中, 字符串类型是基本数据类型的一种, 字符串常量, 可以用单引号或者双引号引起来！

例如：var s1 = “aaa”; var s2 = ‘bbb’;

在js中, 为string类型提供了对应的包装对象 – String

**（3）布尔类型**

boolean, 值为true或false

在js中, 为boolean类型提供了对应的包装对象 – Boolean。

**（4）undefined**

undefined类型的值只有一个, 就是undefined. 表示变量未定义。如果声明了一个变量, 但是没有初始化值, 该变量的值就是undefined

例如：var a; 则a的值即为undefined。

**（5）null**

null类型的值也只有一个, 就是null, 表示此处的值现在为空。常用来作为函数的返回值, 表示该返回值是一个没有任何内容的空对象

#### 3.2.2.复杂数据类型

比如：对象（函数，数组，普通对象）后面详讲

#### 3.2.3.数据类型的转换（案例2）

js中的数据类型在需要时会自动的进行类型转换, 转换时遵循如下规则:

**数值类型:**

转字符串类型, 直接转成对应值的字符串, 3 --> "3"

转布尔类型, 0和NaN转成false, 其他数值转成true

在需要时, 会自动转成对应的包装对象 100 --> new Number(100)

**字符串:**

空字符串(""): 转数值为0, 转布尔值为false

非空纯数值字符串("123"): 转数值为对应的数值, 转布尔值为true

非空非数值字符串("123abc"): 转数值为NaN, 转布尔值为true

在需要时, 会自动转成对应的包装对象. "aaa" --> new String("aaa")

**布尔类型:**

true: 转数值为 1, 转字符串为"true"

false: 转数值为 0, 转字符串为"false"

在需要时, 会自动转成对应的包装对象.

**undefined**

转数值为NaN, 转字符串为"undefined", 转布尔值为false, 转对象会抛出异常!

**null**

转数值为 0, 转字符串为"null", 转布尔值为false, 转对象会抛出异常!

### 3.3.变量的定义

在js中有数据类型, 但是变量不区分类型, 所以称js是一门弱类型的语言。

在js中通过var关键字声明变量, 变量不区分类型, 可以指向任意类型的值!

例如：var s1 = “aaa”; var i = 100;

### 3.4.运算符(案例3)

JS中的运算符和Java中的运算符大致相同

JS中与java运算符不同之处：

比较运算符除了“==”，还有“===”

== 在进行比较时, 如果两端的值不是同一种数据类型, 先自动转成同一种数据类型, 再进行比较, 如果转换之后再比较, 相等则返回true, 不相等则返回false;

=== 为严格相等，在进行比较时, 如果两端的值不是同一种数据类型, 直接返回false; 如果是同一种类型再次比较值是否相等, 是则返回true, 不是则返回false

typeof: 用来返回变量/常量/表达式的数据类型

格式：var x = 123; typeof x

delete: 用来删除数组中的元素或者删除对象中的属性或方法。。。

格式：var arr = [123,”abc”,true]; delete arr[1];

### 3.5.语句

JS中的语句和Java中的语句也大致相同

if条件语句

如果判断条件不是布尔类型, 会自动转换成布尔值!

switch语句

语法和Java中大致相同, 并且也支持字符串类型

循环语句

while(){} do{}while() for(){} 语法和Java中的很相似, 但是js中不支持增强for循环

### 3.6.函数（案例4）

#### 3.6.1.通过function关键字定义一个函数

function 函数名(形参列表){

函数体

}

调用函数：函数名(实参)

#### 3.6.2.通过函数表达式定义一个函数

var 函数名 = function(形参列表){

函数体

}

调用函数：函数名(实参)

### 3.7.数组

**3.7.1.通过Array()构造函数来创建数组**

var arr1 = new Array(); //创建一个空数组

var arr2 = new Array(10); //创建一个长度为10的空数组

var arr3 = new Array(“aa”, 55, true); //创建一个具有指定初始值的数组

**3.7.2.通过数组直接量来创建数组**

var arr4 = []; //创建一个空数组

var arr5 = [“x”, 100, false]; //创建一个具有指定初始值的数组

**数组中的细节问题:**

>> 数组的长度可以被任意改变

var arr =[“aa”, “bb”, “cc”]; arr.length=1; //[“aa”]

>> 如果数组中某一个位置没有元素, 该位置的值为undefined

var arr =[“aa”, “bb”, “cc”]; arr[10]; //undefined

>> JS中的数组可以存放任意类型的数据

var arr = [“x”, 100, false];

### 3.8.内置对象（案例5）

#### 3.8.1.String对象

创建：

var str1 = new String(“xxx”); //创建字符串对象

var str2 =“xxx”; //创建字符串基本类型，但是也可以

常用属性和方法：

str.length 返回字符串长度。

str.charAt(index) 返回指定索引位置处的字符。

例如：var str = "abcdeabc"; str.charAt(3) // d

str.indexOf(subString, startIndex) 返回第一次出现子字符串的位置。

例如：var str = "abcdeabc"; str.indexOf("bc",2) // 6，从第3个开始数

str.lastIndexOf(substring, startindex) 返回最后出现子字符串的位置。

例如：var str = "abcdeabc"; str.lastIndexOf("bc",8); // 6，从后往前数

str.split(delimiter) 将字符串按照指定字符分割为字符串数组。

例如：var str = "a:b:c:d"; str.split(":") //返回为一个数组[a,b,c,d]

str.slice(start, end) 提取字符串的某个部分，含头不含尾。

例如：var str = "abcdeabc"; str.slice(2,4); // cd

str.substr(start, length) 返回从指定位置开始的指定长度的子字符串。

例如：var str = "abcdeabc"; str.substr(2,4); //cdea

str.toLowerCase( ) 字符串中的字母转换为小写。

例如：var str = "ABCDE"; str.toLowerCase(); //abcde

str.toUpperCase( ) 字符串中的字母转化为大写。

例如：var str = "abcde"; str.toUpperCase(); //ABCDE

**str.match(regexp) 在字符串中查找指定匹配正则表达式的值。**

**str.replace(regexp, replaceText) 字符串中匹配正则的值替换为其他值。**

**str.search(regexp) 查找与正则表达式匹配的第一个子字符串的位置。**

#### 3.8.2.RegExp正则

创建：

var reg1 = new RegExp("", "");

var reg2 = /xxx/;

标识符: g 🡪 Global 全局查找 i 🡪 IgnoreCase 忽略大小写

如果正则需要从头到尾都匹配，需要以”^”开头，以”$”结尾。

reg.test(str); 检查字符串是否符合该正则, 如果符合返回true, 不符合返回false！！！

#### 3.8.3.Array对象

创建：

var arr1 = new Array();

var arr2 = [];

常用属性和方法：

arr.length 返回数组中元素的个数

例如：var arr = ["a","b","c"]; arr.length; // 3

arr.concat(arr1,arr2,arr3...) 将多个数组合并成一个

例如：

var arr1 = ["a","b","c"]; var arr2 = ["d","e","f"]; var arr3 = ["x","y","z"];

arr1.concat(arr2,arr3); // ["a","b","c","d","e","f","x","y","z"]

arr.join(string) 将数组中所有元素按照指定字符串连接成一个字符串

例如：var arr = ["aaa",123,true]; arr.join("-"); // aaa-123-true

arr.reverse( ) 返回一个元素顺序被反转的 Array 对象。

例如：var arr = ["a","b","c"]; arr.reverse(); //["c","b","a"]

arr.slice(start, end) 返回一个数组的一段，含头不含尾。

例如：var arr = ["a","b","c",1,2,3]; arr.slice(2,4); // ["c",1]

arr.splice(start, deleteCount,value...)

从一个数组中移除一个或多个元素，如果必要，在所移除元素的位置上插入新元素，返回所移除的元素。

例如：var arr = ["a","b","c","d","e","f"]; arr.splice(2,2,"zzz");

// 返回值为["c","d"]，arr则变为["a","b","zzz","e","f"]，中间被替换

arr.pop( ) 移除数组中的最后一个元素并返回该元素

例如：var arr = ["aaa",123,"x",true,8]; arr.pop();

//返回值为 8，arr则变为["aaa",123,"x",true]

arr.push(value) 在数组的末尾加上一个或多个元素，并返回新数组的长度值

例如：var arr = ["aaa",123,"x",true,8]; arr.push("last")

//返回值为6，arr则变为 ["aaa",123,"x",true,8,"last"]

arr.shift( ) 移除数组中的第一个元素并返回该元素

例如：var arr = ["aaa",123,"x",true,8]; arr.shift()

//返回值为 "aaa" ，arr则变为[123,"x",true,8]

arr.unshift(value) 为数组的开始部分加上一个或多个元素,并且返回该数组的新长度

例如：var arr = ["aaa",123,"x",true,8]; arr.unshift("first")

//返回值为6，arr则变为["first","aaa",123,"x",true,8]

**arr.sort(sortfunction) 返回排序后的数组。**

#### 3.8.4.date对象

创建：

var date1 = new Date(); // 当前时间

var date2 = new Date(年,月,日[,时,分,秒]); //指定时间，可以只有年月日

注意：当创建指定时间的时候，月是从0开始，例如：

var date2 = new Date(2008,7,8,20,0,0);

这个时间是2008年8月8日晚上8点整

常用方法：

data.toLocaleString() 把日期(一个数值)转变成本地日期格式字符串

例如：var date2 = new Date(2008,7,8); date2.toLocaleString();

// "2008/8/8 上午12:00:00"

data.[getFullYear()](http://www.w3cschool.cn/tryrun/showhtml/tryjsref_getfullyear" \t "http://www.w3cschool.cn/javascript/_blank) 获取日期对象中所表示时间的年份

例如： var date = new Date(2008,7,8); date.getFullYear(); // 2008

data.[getMonth()](http://www.w3cschool.cn/tryrun/showhtml/tryjsref_getfullyear" \t "http://www.w3cschool.cn/javascript/_blank) 获取日期对象中所表示时间的月份，返回一个 0 到 11 之间的整数，0表示一月。

例如：var date = new Date(2008,7,8); date.getMonth(); // 7

data.[getDate()](http://www.w3cschool.cn/tryrun/showhtml/tryjsref_getfullyear" \t "http://www.w3cschool.cn/javascript/_blank) 获取日期对象中所表示一个月的日期值，返回值是一个处于 1 到 31 之间的整数，代表了相应的日期值。

例如：var date = new Date(2008,7,8); date.getDate(); // 8

data.getDay() 获取日期对象中所表示一个周的日期值。 返回的值是一个处于 0 到 6 之间的整数，0表示周日，6表示周六。

例如：var date = new Date(2008,7,8); date.getDay(); // 5，表示周五

data.getTime() 返回从 1970 年 1 月 1 日至指定时间的毫秒数。

例如：var date = new Date(2008,7,8); date.getTime(); //1218124800000

#### 3.8.5.Math对象

Math可以直接拿来用！

常用属性和方法：

Math.PI 返回圆周率的值，约等于 3.141592653589793。

例如：var num = Math.PI; // num 的值为3.141592653589793。

Math.exp(number) 返回 e（自然对数的底）的幂。

例如：Math.exp(1) // e1 约等于2.718281828459045

Math.abs(number) 返回数字的绝对值。

例如：Math.abs(-5) // 5

Math.ceil(number) 向上取整

例如：Math.ceil(3.14) // 4

Math.floor(number) 向下取整

例如：Math.floor(3.14) // 3

Math.round(number) 四舍五入到整数

例如：Math.round(5.67) // 6

**Math.random() 返回介于 0 和 1 之间的伪随机数。**

#### 3.8.6.全局对象globle

全局对象没有语法。直接调用其方法。

parseFloat(numString) 将字符串转换成浮点数。

例如：parseFloat("13.14") // 返回number类型的13.14

parseInt(numString) 将字符串转成整数，非四舍五入。

例如：parseInt("9.99") // 返回number类型的9

isNaN(numValue) 判断一个值是否为非数字。

例如： isNaN("abc") // true

isNaN("123") // false

eval(codeString) 判断一个字符串并将其以脚本代码的形式执行

例如：eval("alert(1+2)"); //会直接将"alert(1+2)"当作代码执行，弹出3

### 3.9.自定义对象

#### 3.9.1.通过构造函数定义对象

function Person(){} //定义一个空对象

function Person2(形参列表){

对象内容

}

//定义一个带参数，有内容的对象

#### 3.9.2.通过对象直接量定义

var p3 = {

"name" : "宋小宝",

"age" : 19,

"run" : function(){

alert(this.name+"?"+this.age)

}

}

#### 3.9.3.对象的取值

p3.name 或者 p3[“name”]/p3[key]

## 4.DHTML

### 4.1.概述

DHTML: Dynamic HTML 动态的 html

DHTML并不是一门新的技术, 而是将现有的HTML, CSS, JavaScript整合在一起, 形成了DHTML技术.

DHTML分为BOM和DOM。

BOM：Browse Object Modle 即浏览器对象模型，其中封装了浏览器操作相关对象。

DOM：Document Object Modle 即文档对象模型，将整个HTML文档按照文档结构组织成了树形结构。

### 4.2.BOM

#### 4.2.1.window对象

此对象为全局对象，因此，调用方法或属性时，可以省略window。

常用事件：

onclick事件 – 当窗口被点击时触发

onfocus事件 – 当窗口获得焦点时触发

onblur事件 – 当窗口失去焦点时触发

onload事件 – 当整个html文档加载完之后立即触发

常用方法：

alert() – 消息对话框

confirm() – 确认对话框 点击确定，返回true，取消则返回false

#### 4.2.2.location对象

属性：

href – 获取或设置浏览器地址栏url。。。

### 4.3.DOM

#### 4.3.1.获取元素

document.getElementById("id值"); 通过id值获取指定id的元素

div.getElementById("id值");

document.getElementsByName("name属性值"); 通过name属性获取指定name值的所有元素组成的集合数组

document.getElementsByTagName("元素名"); 通过元素的名称获取指定元素名的所有元素组成的集合数组

value属性： 获取或设置输入框的value值

innerText： (部分浏览器不支持) 获取或设置元素内的文本内容

innerHTML： 获取或设置元素的html内容

#### 4.3.2.元素的增删改

var oDiv = docuemnt.createElement("div"); 创建一个div元素

body.appendChild(oDiv); 往body中追加一个子元素

body.removeChild(oDiv); 删除oDiv子元素

body.replaceChild(oNewDiv, oDiv); 用oNewDiv替换已有的子元素oDiv

body.insertBefore(oDiv1, oDiv); 往body中插入一个子元素oDiv1, 子元素是插入在oDiv的前面

oDiv.cloneNode([boolean]); 克隆oDiv元素, 参数默认值为false, 表示只克隆元素本身, 不克隆oDiv内的所有子孙元素, 如果希望克隆元素还克隆元素内容所有子孙元素, 可以传入参数true！！！