JavaScript学习

JavaScript简介

JavaScript是一种轻量级的编程语言，是可插入HTML页面的编程代码，广泛应用于服务器、pc、笔记本电脑，是目前互联网上最流行的脚本语言。

JavaScript简单使用

HTML中的JavaScript脚本代码必须位于标签之间

JavaScript脚本代码可被放置在HTML页面的和部分中

JS输出

- 使用window.alert()弹出警告框
- 使用document.write()方法将内容写到HTML文档中
- 使用innerHTML写入HTML元素
- 使用console.log()写入浏览器的控制台

JS语法

JS字面量

decr：固定值称为字面量，如3.14

- 数字(Number)字面量可以是整数或者小数，或者是科学计数法
- 字符串(String)字面量可以使用单引号或双引号
- 表达式字面量用于计算
- 数组(Array)字面量定义一个数组
- 对象(Object)字面量定义一个对象
- 函数(Function)字面量定义一个函数

JS变量

decr：变量用于存储数据值

JavaScript中使用var关键字来定义变量。可以使用等号来为变量赋值

```
var x,length
x = 5
length = 10
```

注意：字面量是一个恒定的值；变量是可变的。

JS语句

decr：JS语句是发给浏览器的命令。JS语句使用分号间隔

JS注释

```
// 注释内容
```

JS数据类型

```
var length = 16;                                       // Number 通过数字字面量赋值
var points = x * 10;                                   // Number 通过表达式字面量赋值
var lastName = "Johnson";                              // String 通过字符串字面量赋值
var cars = ["Saab", "Volvo", "BMW"];                   // Array  通过数组字面量赋值
var person = {firstName:"John", lastName:"Doe"};       // Object 通过对象字面量赋值
```

JS函数

```
function 函数名称(参数列表){
    函数体;
}
```

JS字母大小写

decr：JavaScript对字母大小写敏感

JS字符集

decr：JavaScript默认使用unicode作为字符集

JS注释

单行注释

```
// 注释内容
```

多行注释

```
/*
注释内容1
注释内容2
*/
```

JS变量

decr：变量是用于存储信息的“容器”

JS变量命名规范

- 变量名必须以字母开头
- 变量名也能以$和_符号开头(不推荐)
- 变量名称对大小写敏感

声明JS变量

```
var 变量名 = 值;
或者
var 变量名;
变量名 = 值;
```

一条语句声明多个变量

```
var 变量名1 = 值1,变量名2 = 值2,变量名3 = 值3;
```

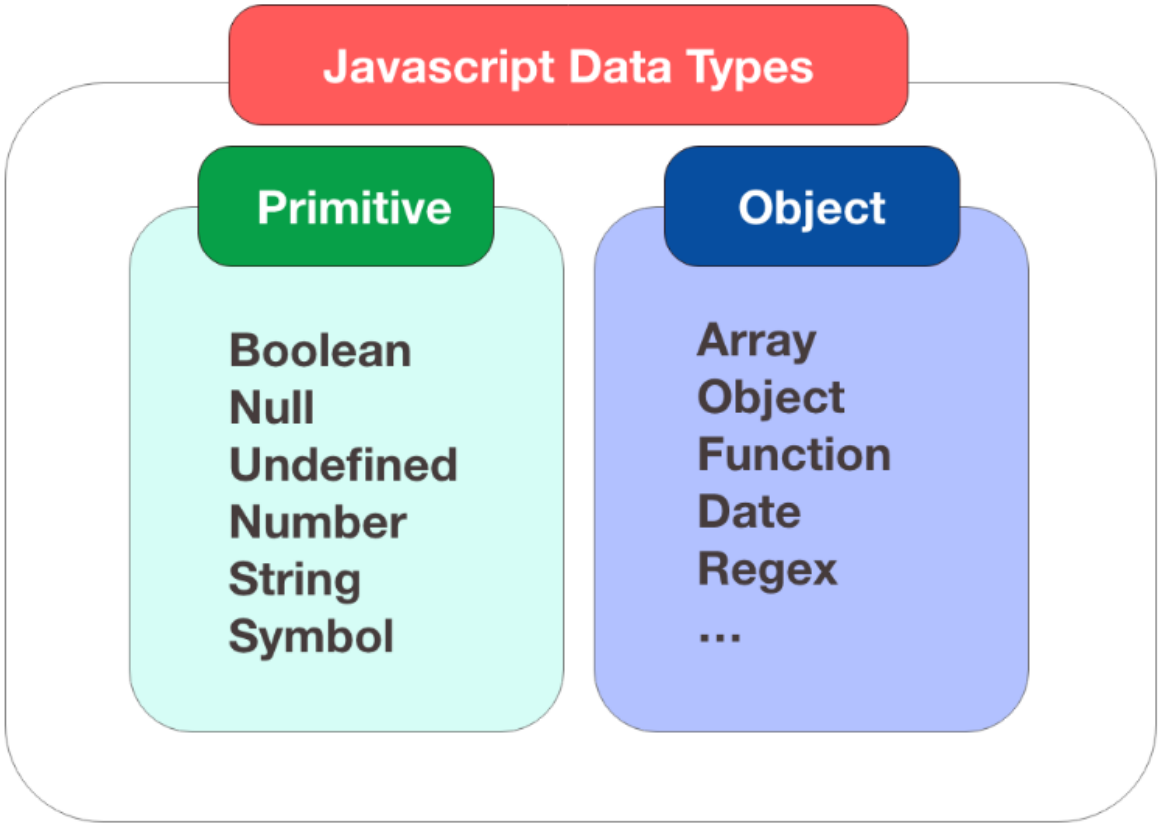
注意：

- 若声明变量但未赋值，则该变量为undefined
- 若声明变量且赋值，重新声明变量无法改变其值

JS数据类型

值类型(基本类型) 字符串(String)、数字(Number)、布尔(Boolean)、空(Null)、未定义(Undefined)、Symbol

引用数据类型(对象类型) 对象(Object)、数组(Array)、函数(Function)、正则(RegExp)和日期(Date)



decr：JS拥有动态类型，相同变量可用作不同的类型

```
var x;               // x 为 undefined
var x = 5;           // 现在 x 为数字
var x = "John";      // 现在 x 为字符串
```

查看变量的数据类型，可以使用typeof操作符来查看：

```
typeof "John"                // 返回 string
typeof 3.14                  // 返回 number
typeof false                 // 返回 boolean
typeof [1,2,3,4]             // 返回 object
typeof {name:'John', age:34} // 返回 object
```

JS字符串

字符串是存储字符的变量。

字符串可以是引号中的任意文本，可以使用单引号或双引号。

可以在字符串中使用引号，只要不匹配包围字符串的引号即可

JS数字

JS只有一种数字类型。数字可以带小数点，也可以不带，极大或极小的数字可以通过科学计数法来书写

JS布尔

JS中的布尔逻辑只有两个值true或false

JS数组

```
var cars = new Array();
cars[0] = "Saab";
cars[1] = "Volvo";
cars[2] = "BMW";
```

或者

```
var cars = new Array("Saab","Volvo","BMW");
```

或者

```
var cars = ["Saab","Volvo","BMW"];
```

注意：JS数组的下标同Python数组一样，从0开始！！

JS对象

对象由花括号分隔。在括号内部，对象的属性以键值对的形式来定义，属性之间使用逗号隔开。

对象的属性有两种寻址方式：

```
变量名 = 对象名称.属性名称
或者
变量名 = 对象名称["属性名称"]
```

JS对象方法

对象的方法定义了一个函数，并作为对象的属性存储；对象方法通过添加()调用。

```
对象名称.对象方法()
```

创建JS对象方法

```
var 变量名 = {
    name1:value1,
    name2:value2,
    对象方法名 : function() {
        // 方法体
    }
}
```

注意：如果不添加()去调用对象方法，则会返回对象方法定义语句

Undefined和Null

Undefined这个值表示变量不含有值

可以通过将变量的值设置为null来清空变量

JS函数

函数是由时间驱动的或者当它被调用时，执行的可重复使用的代码块。

JS定义函数语法

```
function 函数名称()
{
    // 函数体
}
```

JS定义带有参数的函数

```
function 函数名称(形参1,形参2)
{
    // 函数体
}
```

调用

```
函数名称(实参1,实参2)
```

JS定义带有返回值的函数

```
function myFunction()
{
    var x=5;
    return x;
}
```

注意：通过return返回数据之后，仅仅是函数停止，JS不会停止

如果仅仅希望退出函数时，也可以使用return语句。

```
function myFunction(a,b)
{
    if (a>b)
    {
        return;
    }
    x=a+b
}
```

JS作用域

作用域是可访问变量的范围

JS变量

JS局部变量

局部变量是指在函数体中定义的变量，它的作用域只能在函数开始到函数。

JS全局变量

全局变量是指在函数外声明的变量，网页上的所有脚本和函数都能访问它。

JS变量的生命周期

- JS所有变量的生命周期都是从它们被声明的时间开始。
- 局部变量会在函数运行以后被删除。
- 全局变量会在页面关闭后被删除。

注意：在HTML中，全局变量是window对象，所以window对象可以调用函数内的局部变量且**所有数据变量都属于window对象**

JS事件

HTML事件是发生在HTML元素上的事情。

HTML事件

HTML事件可以是浏览器行为，也可以是用户行为。

HTML元素中可以添加事件属性，使用JS代码来添加HTML元素。

HTML事件语法

单引号

```
<html标签 事件名称='JS代码'>
```

双引号

```
<html标签 事件名称="JS代码">
```

常见的HTML事件

事件	描述
onchange	HTML 元素改变
onclick	用户点击 HTML 元素
onmouseover	鼠标指针移动到指定的元素上时发生
onmouseout	用户从一个 HTML 元素上移开鼠标时发生
onkeydown	用户按下键盘按键
onload	浏览器已完成页面的加载

JS字符串

JS字符串用于存储和处理文本。

JS字符串元素索引

JS字符串支持使用索引位置来访问字符串中的每个元素。例如：

```
var carname = "Volvo XC60";
var character = carname[7];
```

注意：JS中字符串位置索引形同Python中字符串位置索引！！！

JS字符串中的引号

JS字符串中的可以使用引号，但分为两种情况：

- 字符串内部引号不冲突于外围引号
- 字符串内部引号冲突于外围引号

针对不冲突情况：

```
var answer = "It's alright";
var answer = "He is called 'Johnny'";
var answer = 'He is called "Johnny"';
```

针对冲突情况：

```
var x = 'It\'s alright';
var y = "He is called \"Johnny\"";
```

JS字符串长度

要获取字符串的长度，可以使用JS内置属性length来获取。

```
var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
var sln = txt.length;
```

JS字符串中的特殊字符

代码	输出
\'	单引号
\"	双引号
\\	反斜杠
\n	换行
\r	回车
\t	tab(制表符)
\b	退格符
\f	换页符

JS字符串也可以是对象

```
var x = "json";
var y = new String("json");
typeof x // 返回String
typeof y // 返回Object
```

注意：创建String对象会拖慢执行速度，并可能产生其他副作用！！！

JS字符串属性和方法

JS原始字符串没有属性和方法(因为它们不是对象)，原始字符串可以使用JS字符串的属性和方法，是因为JS把原始字符串当成对象。

JS字符串属性

属性	描述
constructor	返回创建字符串属性的函数
length	返回字符串的长度
prototype	允许您向对象添加属性和方法

JS字符串方法

方法	描述
charAt()	返回指定索引位置的字符
charCodeAt()	返回指定索引位置字符的 Unicode 值
concat()	连接两个或多个字符串，返回连接后的字符串
fromCharCode()	将 Unicode 转换为字符串
indexOf()	返回字符串中检索指定字符第一次出现的位置
lastIndexOf()	返回字符串中检索指定字符最后一次出现的位置
localeCompare()	用本地特定的顺序来比较两个字符串
match()	找到一个或多个正则表达式的匹配
replace()	替换与正则表达式匹配的子串
search()	检索与正则表达式相匹配的值
slice()	提取字符串的片断，并在新的字符串中返回被提取的部分
split()	把字符串分割为子字符串数组
substr()	从起始索引号提取字符串中指定数目的字符
substring()	提取字符串中两个指定的索引号之间的字符
toLocaleLowerCase()	根据主机的语言环境把字符串转换为小写，只有几种语言（如土耳其语）具有地方特有的大小写映射
toLocaleUpperCase()	根据主机的语言环境把字符串转换为大写，只有几种语言（如土耳其语）具有地方特有的大小写映射
toLowerCase()	把字符串转换为小写
toString()	返回字符串对象值
toUpperCase()	把字符串转换为大写
trim()	移除字符串首尾空白
valueOf()	返回某个字符串对象的原始值

JS运算符

- 运算符 = 用于赋值
- 运算符 + 用于加值

JS算术运算符

用于算术运算

运算符	描述	例子	x 运算结果	y 运算结果	在线实例
+	加法	x=y+2	7	5	实例 »
-	减法	x=y-2	3	5	实例 »
*	乘法	x=y*2	10	5	实例 »
/	除法	x=y/2	2.5	5	实例 »
%	取模（余数）	x=y%2	1	5	实例 »
++	自增	x=++y	6	6	实例 »
		x=y++	5	6	实例 »
--	自减	x=--y	4	4	实例 »
		x=y--	5	4	实例 »

JS赋值运算符

用于赋值运算

运算符	例子	等同于	运算结果	在线实例
=	x=y		x=5	实例 »
+=	x+=y	x=x+y	x=15	实例 »
-=	x-=y	x=x-y	x=5	实例 »
*=	x*=y	x=x*y	x=50	实例 »
/=	x/=y	x=x/y	x=2	实例 »
%=	x%=y	x=x%y	x=0	实例 »

字符串加法运算

- 字符串 + 字符串 = 字符串
- 字符串 + 数字 = 字符串

JS比较运算符

比较运算符在逻辑语句中使用，以测定变量或值是否相等。

运算符	描述	比较	返回值
==	等于	x==8	*false*
		x==5	*true*
===	绝对等于（值和类型均相等）	x==="5"	*false*
		x===5	*true*
!=	不等于	x!=8	*true*
!==	不绝对等于（值和类型有一个不相等，或两个都不相等）	x!=="5"	*true*
		x!==5	*false*
>	大于	x>8	*false*
<	小于	x<8	*true*
>=	大于或等于	x>=8	*false*
<=	小于或等于	x<=8	*true*

JS逻辑运算符

逻辑运算符用于测定变量或值之间的逻辑。

运算符	描述	例子
&&	and	(x < 10 && y > 1) 为 true
||	or	(x==5 || y==5) 为 false
!	not	!(x==y) 为 true

条件运算符

```
variablename=(condition)?value1:value2
```

注意：如果条件成立，返回value1，否则返回value2！！

JS流程控制语句

JS条件语句

- if 语句
- if ... else 语句
- if ... else if ... else 语句
- switch 语句

if 语句

只有当指定条件成立时，该语句才会执行代码。

```
if (条件)
{
    当条件为true的代码块
}
```

if ... else 语句

当条件成立时，执行响应的代码，条件不成立，执行其他代码。

```
if (条件)
{
    当条件为true时 执行该部分代码
}
else
{
    当条件为false时，执行该部分代码
}
```

if ... else if ... else ...语句

使用if ... else if ... else ...语句来选择多个代码块之一来执行。

```
if (条件1)
{
    当条件1成立时，执行该部分代码
}
else if (条件2)
{
    当条件2成立时，执行该部分代码
}
else
{
    当条件1和条件2都不成立时，执行该部分代码
}
```

switch 语句

switch语句用于在多个要执行的代码块中选择一个执行。

```
switch(n)
    {
        case 1:
            执行代码块1
            break;
        case 2:
            执行代码块2
            break;
        default:
            与case1和case2不同时执行的代码
    }
```

工作原理：使用switch括号中的数据，与每种case进行对比，如果匹配成功，则执行相应case下的代码块，如果执行的代码块中没有break；语句，则会顺延执行下一个代码块，直到找到break语句或执行完所有代码块；switch语句中default语句是为了防止找不到匹配条件的情况。

JS循环语句

JS支持不同类型的循环

- for 循环代码块一定的次数
- for/in 循环遍历对象的属性
- while 当指定的条件为true时循环指定的代码块
- do/while 同样当指定的条件为true时循环指定的代码块

for循环

```
for (语句1;语句2;语句3)
{
    被执行的代码块
}
```

- 语句1通常用于初始化循环变量
- 语句2通常用于给定循环继续的条件
- 语句3通常用于更新循环变量的值

for循环语句省略的条件

语句1：在循环开始之前已经初始化好循环变量并赋予初值

语句2：在循环代码体中添加break语句

语句3：在循环代码体中增加更新循环变量的代码

for/in 循环

JS中的for/in语句循环变量对象的属性

```
var person={fname:"Bill",lname:"Gates",age:56};

for (x in person)  // x 为属性名
{
    txt=txt + person[x];
}
```

while循环

while循环会在指定条件为真时循环执行代码块。

```
while (条件)
{
    代码块
}
```

注意：如果没有加循环停止的条件，则会一直运行下去导致浏览器崩溃哦！！！

do/while 循环

do/while循环是while循环的变体。该循环先执行代码块，然后判断条件是否满足，若条件成立，则继续执行代码块。

```
do
{
    需要执行的代码
}while (条件);
```

break和continue

- break语句用于跳出循环，执行循环体后的语句。
- continue用于跳过循环中某个迭代，继续执行循环的下一个迭代。

JS标签

对JS语句进行标记

```
label:
JS语句
```

使用continue无论带不带标签，都只能应用于循环中。

使用break带标签，可以用在任何JS代码中。

使用标签+break语句

```
cars=["BMW","Volvo","Saab","Ford"];
list:
{
    document.write(cars[0] + "<br>");
    document.write(cars[1] + "<br>");
    document.write(cars[2] + "<br>");
    break list;
    document.write(cars[3] + "<br>");
    document.write(cars[4] + "<br>");
    document.write(cars[5] + "<br>");
}
```

JS typeof，null，undefined

typeof

typeof可以检测变量的数据类型

```
typeof "John"                 // 返回 string
typeof 3.14                   // 返回 number
typeof false                  // 返回 boolean
typeof [1,2,3,4]              // 返回 object
typeof {name:'John', age:34} // 返回 object
```

注意：在JS中数组是一种特殊的对象类型！！！

null

在JS中，null是一个只有一个值的特殊类型，表示一个空对象引用。

若使用typeof检测null返回是object，同时可以设置变量为null来清空对象。

undefined

在JS中，undefined是一个没有设置值的变量，若typeof一个没有值的变量会返回undefined。

也可以使用undefined来清空变量。

undefined和null的区别

两者值相同，但类型不同。

JS类型转换

JS变量可以转换为新变量或其他数据类型：

- 通过使用JavaScript函数
- 通过JavaScript自身自动转换

数字转换为字符串

使用String()方法可以将数字转换为字符串

```
String(数字 or 数字类型变量 or 表达式)
```

例如：

```
String(x)         // 将变量 x 转换为字符串并返回
String(123)       // 将数字 123 转换为字符串并返回
String(100 + 23)  // 将数字表达式转换为字符串并返回
```

Number方法toString()也有同样的效果

```
x.toString()
(123).toString()
(100 + 23).toString()
```

更多数字转换为字符串的方法：

方法	描述
toExponential()	把对象的值转换为指数计数法。
toFixed()	把数字转换为字符串，结果的小数点后有指定位数的数字。
toPrecision()	把数字格式化为指定的长度。

布尔值转换为字符串

```
String(false)        // 返回 "false"
String(true)         // 返回 "true"
false.toString()     // 返回 "false"
true.toString()      // 返回 "true"
```

日期转换为字符串

```
Date()      // 返回 Thu Jul 17 2014 15:38:19 GMT+0200 (W. Europe Daylight Time)
String(new Date())      // 返回 Thu Jul 17 2014 15:38:19 GMT+0200 (W. Europe
Daylight Time)
obj = new Date()
obj.toString()   // 返回 Thu Jul 17 2014 15:38:19 GMT+0200 (W. Europe Daylight
Time)
```

更多关于日期的函数：

方法	描述
getDate()	从 Date 对象返回一个月中的某一天 (1 ~ 31)。
getDay()	从 Date 对象返回一周中的某一天 (0 ~ 6)。
getFullYear()	从 Date 对象以四位数字返回年份。
getHours()	返回 Date 对象的小时 (0 ~ 23)。
getMilliseconds()	返回 Date 对象的毫秒(0 ~ 999)。
getMinutes()	返回 Date 对象的分钟 (0 ~ 59)。
getMonth()	从 Date 对象返回月份 (0 ~ 11)。
getSeconds()	返回 Date 对象的秒数 (0 ~ 59)。
getTime()	返回 1970 年 1 月 1 日至今的毫秒数。

字符串转换为数字

全局方法Number()可以将字符串转换为数字。

若字符串中包含数字，则会转换为数字。

空字符串会转换为0。

其他的字符串会转换为NaN(不是数字)。

更多字符串转为数字的方法：

方法	描述
parseFloat()	解析一个字符串，并返回一个浮点数。
parseInt()	解析一个字符串，并返回一个整数。

一元运算符+

operator + 可用于将变量转换为数字：

```
var y = "5";      // y 是一个字符串
var x = + y;      // x 是一个数字
var y = "John";   // y 是一个字符串
var x = + y;      // x 是一个数字 (NaN)
```

布尔值转换为数字

```
Number(false)     // 返回 0
Number(true)      // 返回 1
```

将日期转换为数字

```
d = new Date();
Number(d)          // 返回 1404568027739
d = new Date();
d.getTime()        // 返回 1404568027739
```

自动转换类型

当JavaScript尝试操作一个错误的数据类型时，会自动转换为正确的数据类型

```
5 + null    // 返回 5          null 转换为 0
"5" + null  // 返回"5null"   null 转换为 "null"
"5" + 1     // 返回 "51"       1 转换为 "1"
"5" - 1     // 返回 4          "5" 转换为 5
```

注意：当我们尝试输出一个对象或变量时，JS会自动调用该变量的toString()方法。