Javascript 学习记录

1.切换图片

```
<body>

<script>
function changeImage()
{
    element=document.getElementById('myimage')
    if (element.src.match("xiazai-fd"))
    {
        element.src="https://pic3.zhimg.com/v2-
58d652598269710fa67ec8d1c88d8f03_r.jpg?source=1940ef5c";
    }
    else
    {
        element.src="https://xiazai-fd.zol-
img.com.cn/t_s960x600/g1/M01/03/06/Cg-
4jVONmIiIa6NpAATdgesQtisAAN9YQLYqJcABN2Z899.jpg";
    }
}
</script>
<img id="myimage" onclick="changeImage()" src="https://xiazai-fd.zol-
img.com.cn/t_s960x600/g1/M01/03/06/Cg-
4jVONmIiIa6NpAATdgesQtisAAN9YQLYqJcABN2Z899.jpg" width="200" height="120">
<p>点击图片进行切换</p>

</body>
```

运行结果：

点击图片进行切换

运行结果：

点击图片进行切换

2.判断输入文本类型，网页提示后清空输入框

```
<input id="demo" type="text">
<script>
function myFunction()
{
    var x=document.getElementById("demo").value;
    if(x==""||isNaN(x))
    {
        alert("不是数字");
    }
    else
    {
        alert("输入合法");
    }
    document.getElementById("demo").value="";
}
</script>
<button type="button" onclick="myFunction()">点击这里</button>
```

```
<p></p>
<input id="demo1" type="text">
<script>
function myFunction1()
    {
        var y=document.getElementById("demo1").value;
        if(y==""||typeof(y)!='string'||!isNaN(y))
        {
            alert("不是字符");
        }
        else
        {
            alert("输入为字符");
```

```
        }
        document.getElementById("demo1").value="";
    }
    </script>
<button type="button" onclick="myFunction1()">GUN</button>
```

我的第一段 JavaScript

请输入数字。如果输入值不是数字，浏览器会弹出提示框。

点击这里

GUN

3.document.write(向html中输入)

```
<body>

<p>
JavaScript 能够直接写入 HTML 输出流中：
</p>
<script>
document.write("<h1>这是一个标题</h1>");
document.write("<p>这是一个段落。</p>");
</script>
<p>
您只能在 HTML 输出流中使用 <strong>document.write</strong>。
如果您在文档已加载后使用它（比如在函数中），会覆盖整个文档。
document.write输入流边写边加载，加载结束后，采用按钮的方式触发就会覆盖之前的输出流。
</p>
<button type="button" onclick="myFunction()">点击这里</button>
<script>
    function myFunction()
    {
        document.write("宇宙很神奇！");
    }
</script>

</body>
```

JavaScript 能够直接写入 HTML 输出流中：

这是一个标题

这是一个段落。

您只能在 HTML 输出流中使用 **document.write**。 如果您在文档已加载后使用它（比如在函数中），会覆盖整个文档。

点击这里

点击“点击这里”之后，界面上原有的输出被覆盖为了函数中的输入流。

宇宙很神奇！

4.在 head 或者 body 标签内的JavaScript

您可以在 HTML 文档中放入不限数量的脚本。

脚本可位于 HTML 的 或 部分中，或者同时存在于两个部分中。

通常的做法是把函数放入 部分中，或者放在页面底部。这样就可以把它们安置到同一处位置，不会干扰页面的内容。

两者中均含javascript的脚本

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>菜鸟教程(runoob.com)</title>
<script>
    function myFunction()
    {
        document.getElementById("demo1").innerHTML="时间到底是什么？";
    }
</script>
</head>
<body>
    <p id="demo1">lalala</p>
    <p></p>
    <button type="button" onclick="myFunction()">点击这里</button>
    <p></p>
    <input id="demo" type="text">
    <script>
    var a=["你点了一下按钮","你又点了一下按钮","你点了第三下了"];
    var flag=0;
```

```
     function duociclick(){
         if(flag>2){
            flag=0;
        }
            document.getElementById("demo").value=a[flag];
        flag++;
 }
    </script>
    <button type="button" onclick="duociclick()">点击这里</button>
</body>
</html>
```

5.调用外部.js-写在<script src=>中

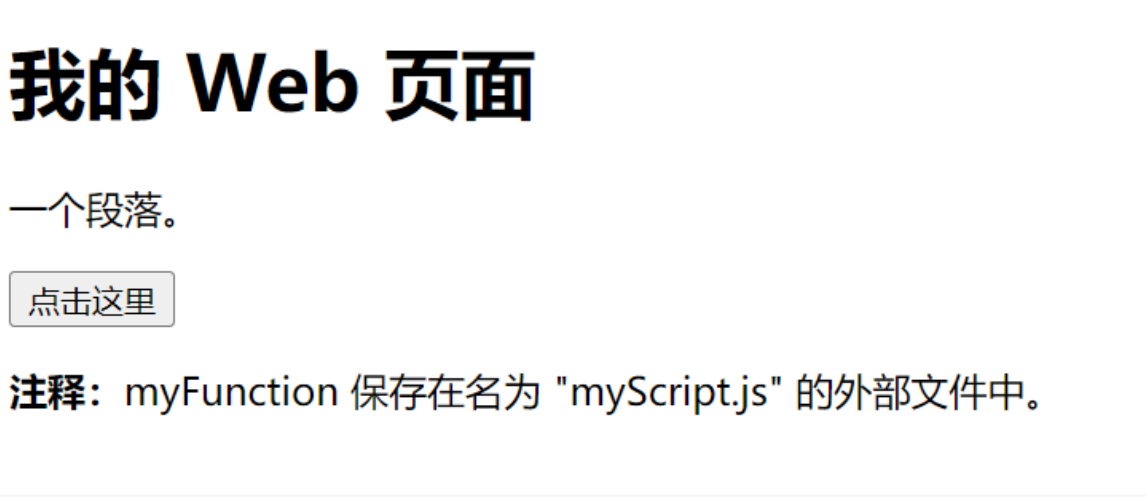
```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>菜鸟教程(runoob.com)</title>
</head>
<body>

<h1>我的 Web 页面</h1>
<p id="demo">一个段落。</p>
<button type="button" onclick="myFunction()">点击这里</button>
<p><b>注释:</b>myFunction 保存在名为 "myScript.js" 的外部文件中。</p>
<script src="myScript.js"></script>

</body>
</html>
```

外部myScript.js文件代码如下:

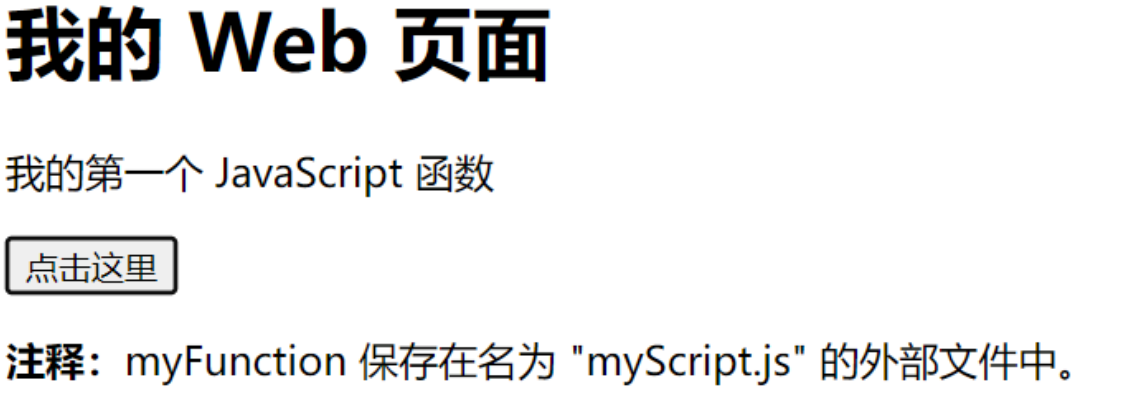
```
function myFunction()
{
    document.getElementById("demo").innerHTML="我的第一个 JavaScript 函数";
}
```

外部脚本不能包含 标签。

运行结果：



运行结果：



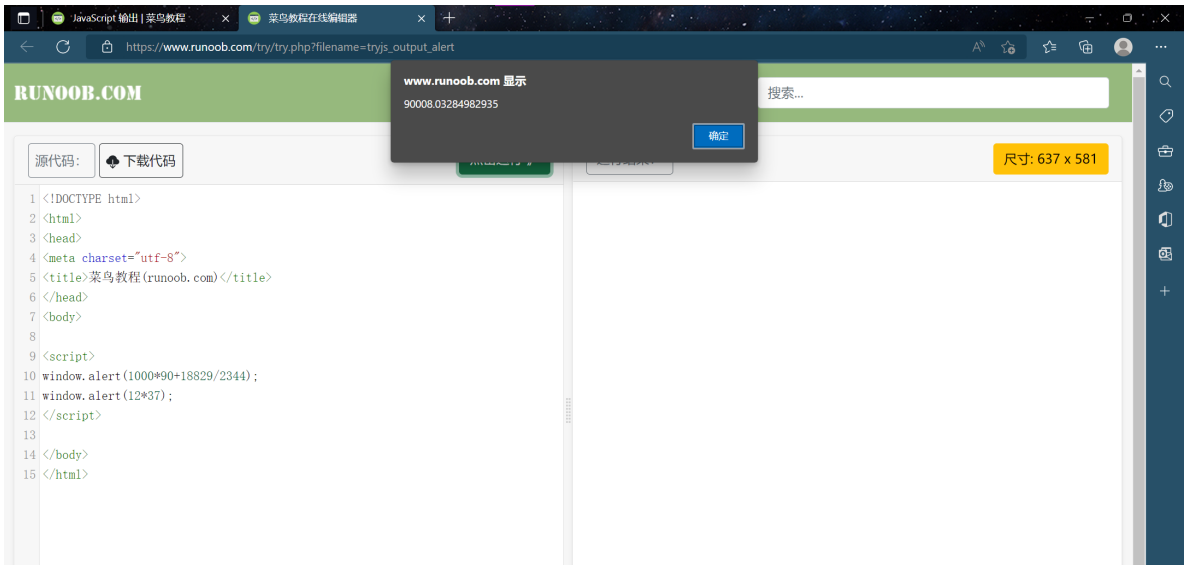
6.javascript的四大输出方式

```
JavaScript 可以通过不同的方式来输出数据：

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 写入到 HTML 元素。
使用 console.log() 写入到浏览器的控制台。
```

window.alert()

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
```

```
<title>菜鸟教程(runoob.com)</title>
</head>
<body>

<script>
window.alert(1000*90+18829/2344);
window.alert(12*37);
</script>

</body>
</html>
```



document.write()

```
<body>

<h1>时间到底是属性，还是熵增带给我们的感知！</h1>
<p>今天也要了解宇宙！</p>
<script>
document.write(Date());
</script>

</body>
```

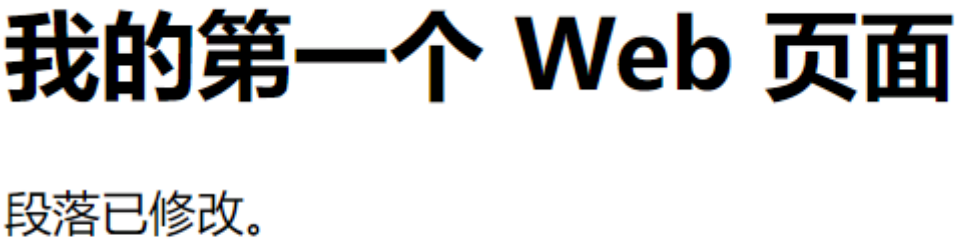
时间到底是属性，还是熵增带给我们的感知！

今天也要了解宇宙！

Mon Nov 07 2022 15:36:41 GMT+0800 (中国标准时间)

innerHTML

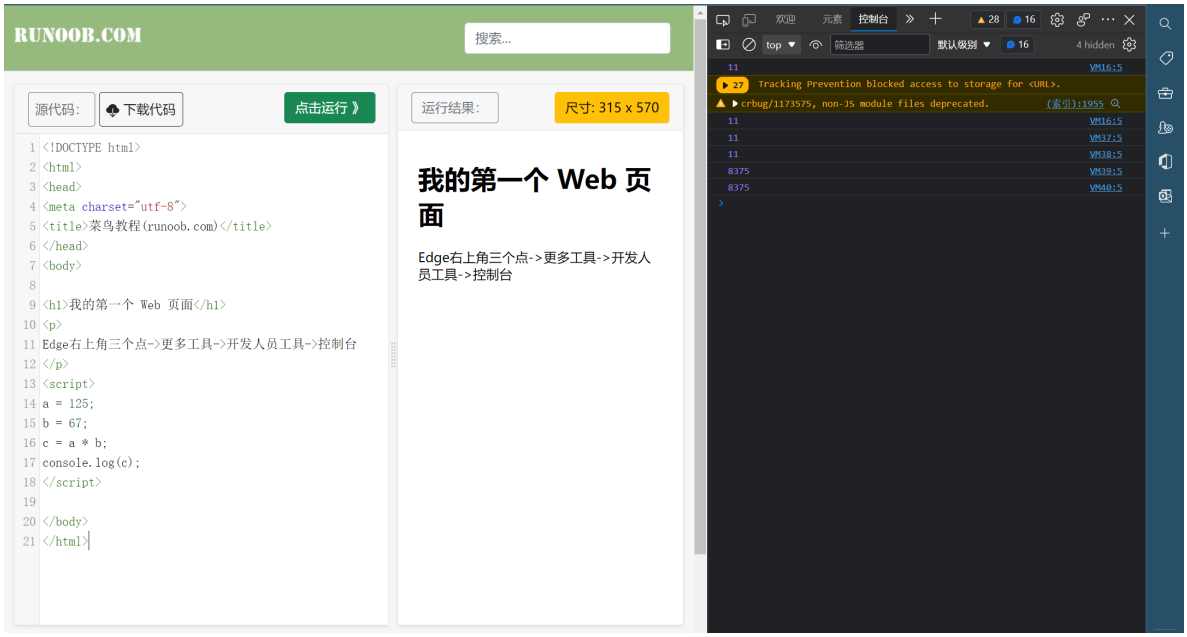
```
<body>

<h1>我的第一个 Web 页面</h1>
<p id="demo">我的第一个段落。</p>
<script>
document.getElementById("demo").innerHTML="段落已修改。";
</script>

</body>
```



段落已修改。

console.log()

```
<body>

<h1>我的第一个 Web 页面</h1>
<p>
Edge右上角三个点->更多工具->开发人员工具->控制台
</p>
<script>
a = 125;
b = 67;
c = a * b;
console.log(c);
</script>

</body>
```



7.javascript语法

1)javascript字面量

在编程语言中，一般固定值称为字面量，如 3.14。
数字（Number）字面量 可以是整数或者是小数，或者是科学计数(e)。

```
<body>

<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = 123e5;
</script>
<p>e是科学计数法，e5=10的5次方</p>

</body>
```

12300000

e是科学计数法，e5=10的5次方

字符串（String）字面量 可以使用单引号或双引号:

```
<body>

<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = 'John Doe';
</script>

</body>
```

John Doe

表达式字面量 用于计算：

```
<body>

<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML = 59 * 19;
</script>

</body>
```

1121

数组（Array）字面量 定义一个数组：[40, 100, 1, 5, 25, 10]

```
<body>

<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML =[40, 100, 1, 5, 25, 10];

</script>

</body>
```

对象（Object）字面量 定义一个对象：{firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"}

```
<body>

<p id="demo"></p>
<script>
var obj={firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
var arr = Object.keys(obj) //返回一个包含所有属性的数组
document.getElementById("demo").innerHTML=obj.firstName+","+obj.lastName+","+obj.age+","+obj.eyeColor;
</script>
</body>
```

函数（Function）字面量 定义一个函数：function myFunction(a, b) { return a * b;}

John,Doe,50,blue

2)javascript变量

在编程语言中，变量用于存储数据值。
JavaScript 使用关键字 var 来定义变量， 使用等号来为变量赋值：

```
<body>

<p id="demo"></p>
<p id="demo1"></p>
<script>
var uname;
uname = "aaaa";
var hight;
hight = 189;
document.getElementById("demo").innerHTML = uname;
document.getElementById("demo1").innerHTML = hight;
</script>

</body>
```

aaaa

189

3)javascript操作符

JavaScript语言有多种类型的运算符：

□类型	实例	描述
赋值，算术和位运算符	= + - * /	在 JS 运算符中描述
条件，比较及逻辑运算符	== != < >	在 JS 比较运算符中描述

```
<body>

<p id="demo"></p>
<script>
var x, y, z;
x = 5;
y = 6;
z = (x + y) * 10;
document.getElementById("demo").innerHTML = z;
</script>

</body>
```

4)javascript语句

```
在 HTML 中，JavaScript 语句用于向浏览器发出命令。
语句是用分号分隔：
x = 5 + 6;
y = x * 10;
```

5)javascript关键字

```
JavaScript 关键字用于标识要执行的操作。

和其他任何编程语言一样，JavaScript 保留了一些关键字为自己所用。

var 关键字告诉浏览器创建一个新的变量：
var x = 5 + 6;
var y = x * 10;
JavaScript 同样保留了一些关键字，这些关键字在当前的语言版本中并没有使用，但在以后 JavaScript 扩展中会用到。

以下是 JavaScript 中最重要的保留关键字（按字母顺序）：
```

abstract	else	instanceof	super
boolean	enum	int	switch
break	export	interface	synchronized
byte	extends	let	this
case	false	long	throw
catch	final	native	throws
char	finally	new	transient
class	float	null	true
const	for	package	try
continue	function	private	typeof
debugger	goto	protected	var
default	if	public	void
delete	implements	return	volatile
do	import	short	while
double	in	static	with

6)javascript注释

```
不是所有的 JavaScript 语句都是"命令"。双斜杠 // 后的内容将会被浏览器忽略：
// 我不会执行
```

7)javascript数据类型

```
JavaScript 有多种数据类型：数字，字符串，数组，对象等等：

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
```

8)javascript字母大小写

```
JavaScript 对大小写是敏感的。
当编写 JavaScript 语句时，请留意是否关闭大小写切换键。
函数 getElementById 与 getElementbyID 是不同的。
同样，变量 myVariable 与 MyVariable 也是不同的。
```

8.JavaScript语句

javascript语句标识符

语句	描述
break	用于跳出循环。
catch	语句块，在 try 语句块执行出错时执行 catch 语句块。
continue	跳过循环中的一个迭代。
do ... while	执行一个语句块，在条件语句为 true 时继续执行该语句块。
for	在条件语句为 true 时，可以将代码块执行指定的次数。
for ... in	用于遍历数组或者对象的属性（对数组或者对象的属性进行循环操作）。
function	定义一个函数
if ... else	用于基于不同的条件来执行不同的动作。
return	退出函数
switch	用于基于不同的条件来执行不同的动作。
throw	抛出（生成）错误。
try	实现错误处理，与 catch 一同使用。
var	声明一个变量。
while	当条件语句为 true 时，执行语句块。

9.JavaScript注释

```
单行注释用//
多行注释用/* */
可用于调试
```

10.JavaScript变量

```
<body>

<script>
var x=5;
var y=6;
var z=x+y;
var m=x*y;
document.write(x + "<br>");
document.write(y + "<br>");
document.write(z + "<br>");
document.write(m + "<br>");
</script>

</body>
```

```
与代数一样，JavaScript 变量可用于存放值（比如 x=5）和表达式（比如 z=x+y）。

变量可以使用短名称（比如 x 和 y），也可以使用描述性更好的名称（比如 age, sum, totalvolume）。

变量必须以字母开头
变量也能以 $ 和 _ 符号开头（不过我们不推荐这么做）
变量名称对大小写敏感（y 和 Y 是不同的变量）
```

11.JavaScript数据类型

JavaScript 变量还能保存其他数据类型，比如文本值 (name="Bill Gates")。

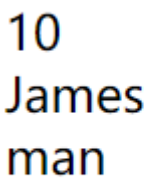
在 JavaScript 中，类似 "Bill Gates" 这样一条文本被称为字符串。

JavaScript 变量有很多种类型，但是现在，我们只关注数字和字符串。

当您向变量分配文本值时，应该用双引号或单引号包围这个值。

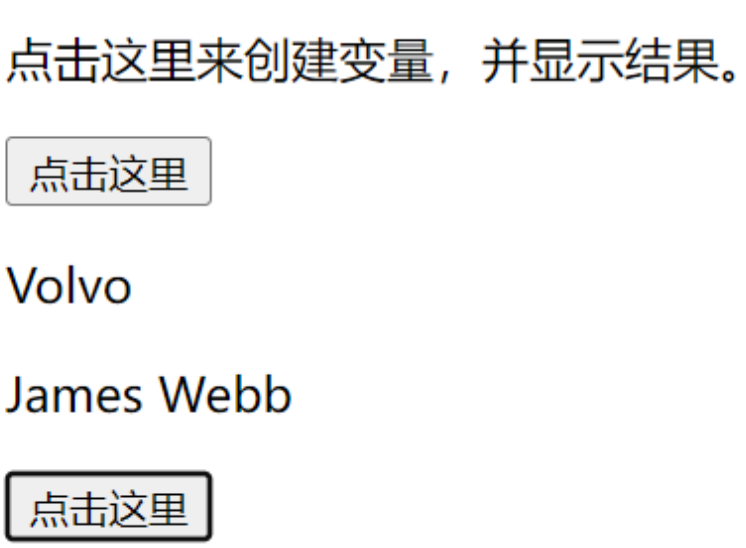
当您向变量赋的值是数值时，不要使用引号。如果您用引号包围数值，该值会被作为文本来处理。

```
<body>

<script>
var num=10;
var name="James";
var sex='man';
document.write(num + "<br>");
document.write(name + "<br>");
document.write(sex + "<br>");
</script>

</body>
```




```
<body>

<p>点击这里来创建变量，并显示结果。</p>
<button onclick="myFunction()">点击这里</button>
<p id="demo"></p>
<script>
function myFunction(){
    var carname="Volvo";
    document.getElementById("demo").innerHTML=carname;
}
</script>
<p id="demo1"></p>
<button type="button" onclick="myFunction1()">点击这里</button>
<script>
    function myFunction1(){
        var username = "James Webb";
        document.getElementById("demo1").innerHTML=username;
    }
</script>
```

```
</body>
```



一条语句，多个变量

您可以在一条语句中声明很多变量。该语句以 var 开头，并使用逗号分隔变量即可：

```
var lastname="Doe", age=30, job="carpenter";
```

声明也可横跨多行：

```
var lastname="Doe",
age=30,
job="carpenter";
```

一条语句中声明的多个变量不可以同时赋同一个值:

```
var x,y,z=1;
```

x,y 为 undefined， z 为 1。

Value = undefined

在计算机程序中，经常会声明无值的变量。未使用值来声明的变量，其值实际上是 undefined。

在执行过以下语句后，变量 carname 的值将是 undefined：

```
var carname;
```

重新声明 JavaScript 变量

如果重新声明 JavaScript 变量，该变量的值不会丢失：

在以下两条语句执行后，变量 carname 的值依然是 "Volvo"：

```
var carname="Volvo";
var carname;
```

javascript算数

```
<body>

<p>假设 y=5，计算 x=y+2，并显示结果。</p>
<button onclick="myFunction()">点击这里</button>
<p id="demo"></p>
<script>
function myFunction(){
    var y=5;
    var x=y+2;
```

```
    var demoP=document.getElementById("demo");
    demoP.innerHTML="x=" + x;
}
</script>

</body>
```

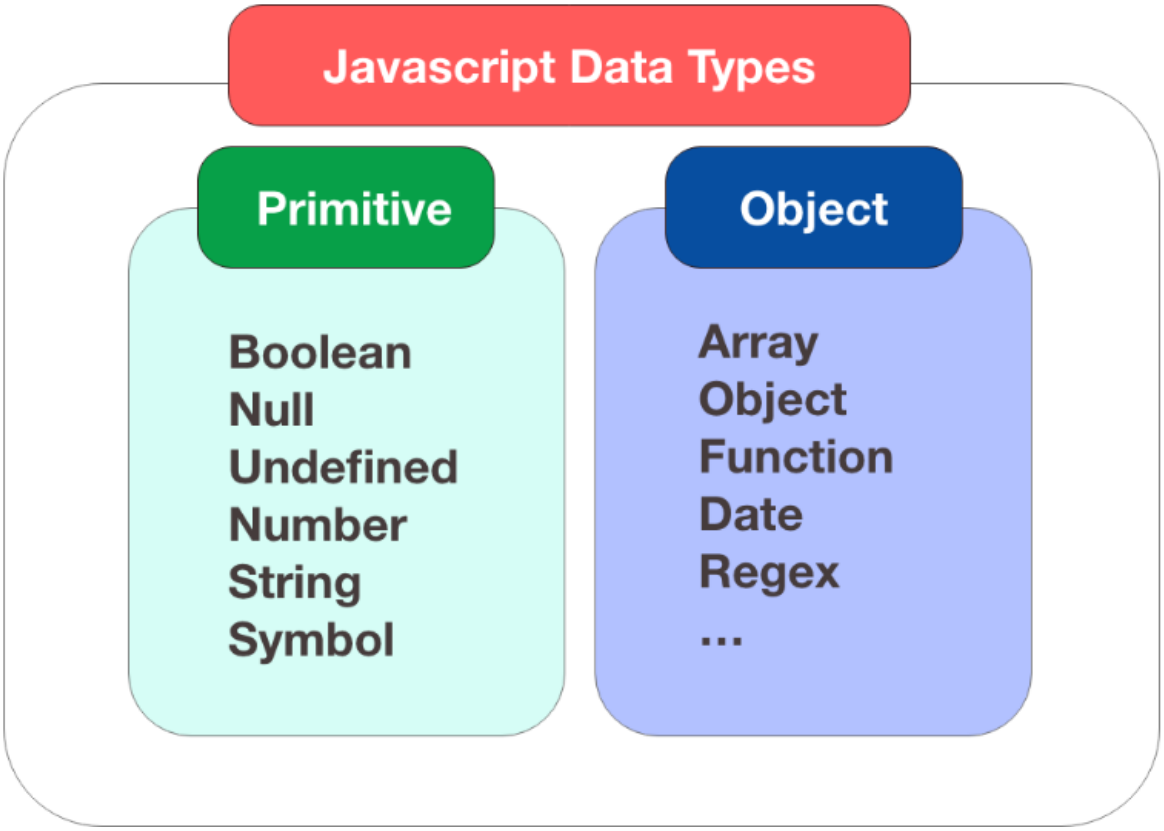
假设 y=5，计算 x=y+2，并显示结果。

点击这里

x=7

基本数据类型

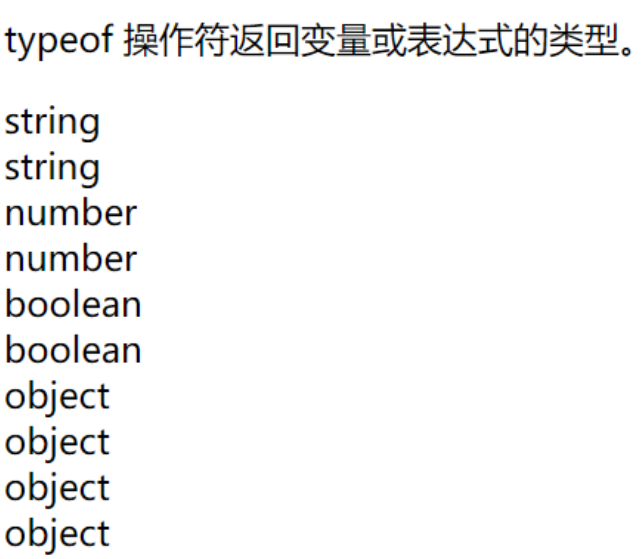
值类型(基本类型)：字符串（String）、数字(Number)、布尔(Boolean)、空（Null）、未定义（Undefined）、Symbol。
引用数据类型（对象类型）：对象(Object)、数组(Array)、函数(Function)，还有两个特殊的对象：正则（RegExp）和日期（Date）。



```
JavaScript 拥有动态类型
JavaScript 拥有动态类型。这意味着相同的变量可用作不同的类型：

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

可以用typeof查看数据类型

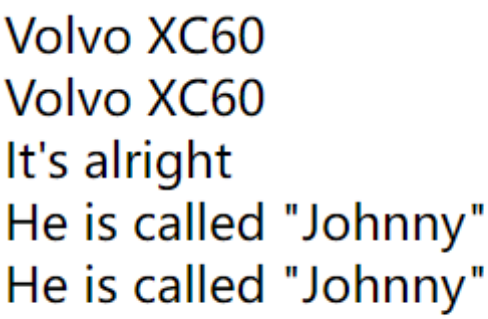
```
<body>

<p> typeof 操作符返回变量或表达式的类型。</p>
<p id="demo"></p>
<script>
document.getElementById("demo").innerHTML =
    typeof "john" + "<br>" +
    typeof 'yes' + "<br>" +
    typeof 3.14 + "<br>" +
    typeof 345 + "<br>" +
    typeof false + "<br>" +
    typeof true + "<br>" +
    typeof [1,2,3,4] + "<br>" +
    typeof ['of','哈哈哈','goood'] + "<br>" +
    typeof {name:'john', age:34} + "<br>" +
    typeof {uname:'随行佯醉',interest:'看宇宙'};
</script>

</body>
```



javascript字符串（字符串内也可以加引号，建议内部引号前加\，避免与外部引号匹配而报错）

```
<body>

<script>
```

```
var carname1="Volvo XC60";
var carname2='Volvo XC60';
var answer1='It\'s alright';
var answer2="He is called \"Johnny\"";
var answer3='He is called "Johnny"';
document.write(carname1 + "<br>")
document.write(carname2 + "<br>")
document.write(answer1 + "<br>")
document.write(answer2 + "<br>")
document.write(answer3 + "<br>")
</script>

</body>
```



javascript数字

JavaScript 只有一种数字类型。数字可以带小数点，也可以不带：

实例

```
var x1=34.00;      //使用小数点来写
var x2=34;         //不使用小数点来写
```

极大或极小的数字可以通过科学（指数）计数法来书写：

实例

```
var y=123e5;      // 12300000
var z=123e-5;     // 0.00123
```

尝试一下 »

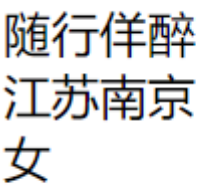
您将在本教程的高级部分学到更多关于数字的知识。

javascript布尔

```
布尔（逻辑）只能有两个值：true 或 false。

var x=true;
var y=false;
```

javascript数组

```
<body>

<script>
var person = new Array();
person[0] = '随行佯醉';
person[1] = '江苏南京';
person[2] = '女';
for(var j=0;j<person.length;j++){
    document.write(person[j] + "<br>");
}
</script>

</body>
```



javascript对象

对象由花括号分隔。在括号内部，对象的属性以名称和值对的形式 (name : value) 来定义。属性由逗号分隔：

```
var person={firstname:"John", lastname:"Doe", id:5566};
```

上面例子中的对象 (person) 有三个属性：firstname、lastname 以及 id。

空格和折行无关紧要。声明可横跨多行：

```
var person={
firstname : "John",
lastname  : "Doe",
id        :  5566
};
```

对象属性有两种寻址方式：

```
<body>

<script>
var person=
{
    name : "随行佯醉",
    interest : "play",
    id :  777
};
document.write(person.name + "<br>");
document.write(person["name"] + "<br>");
document.write(person.interest);
</script>

</body>
```

随行佯醉
随行佯醉
play

Undifined 和 NULL

```
<body>

<script>
var person;
var car="Volvo";
document.write(person + "<br>");
document.write(car + "<br>");
var car=null
document.write(car + "<br>");
</script>

</body>
```

undefined
Volvo
null

使用关键词new来声明创建的变量类型

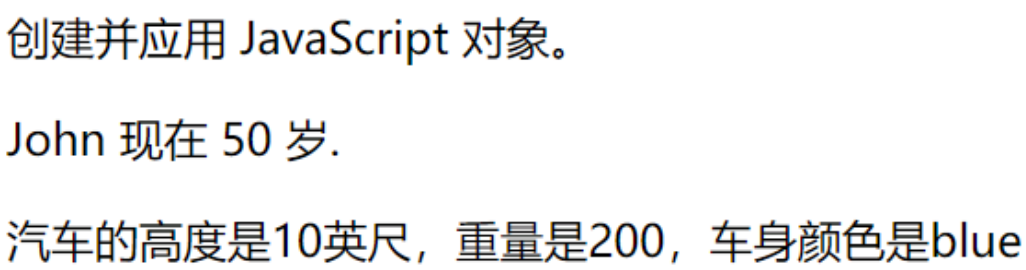
```
当您声明新变量时，可以使用关键词 "new" 来声明其类型：

var carname=new String;
var x=      new Number;
var y=      new Boolean;
var cars=   new Array;
var person= new Object;
```

12.JavaScript对象

对象定义

```
<body>

<p>创建并应用 JavaScript 对象。</p>
<p id="demo"></p>
<script>
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"};
document.getElementById("demo").innerHTML =
    person.firstName + " 现在 " + person.age + " 岁.";
</script>
<p id = "demo1"></p>
```

```
<script>
    var car = {height:10,weight:200,color:"blue"};
    document.getElementById("demo1").innerHTML="汽车的高度是" + car.height + "英尺，重量是" + car.weight + "，车身颜色是" + car.color;
</script>

</body>
```



访问对象属性

```
<body>

<p>
有两种方式可以访问对象属性:
</p>
<p>
你可以使用 .property 或 ["property"]。
</p>
<p id="demo"></p>
<script>
var person = {
    firstName: "John",
    lastName : "Doe",
    id : 5566
};
document.getElementById("demo").innerHTML =
    person["firstName"] + " " + person["lastName"];
document.write(person.firstName + " " + person.lastName);
</script>

</body>
```

有两种方式可以访问对象属性:

你可以使用 .property 或 ["property"]。

John Doe

John Doe

对象方法

```
<body>

<p>创建和使用对象方法。</p>
<p>对象方法是一个函数定义,并作为一个属性值存储。</p>
<p id="demo1"></p>
<p id="demo2"></p>
<p id="demo3"></p>
<script>
var person = {
    firstName: "John",
    lastName : "Doe",
    id : 5566,
    fullName : function()
    {
       return this.firstName + " " + this.lastName;
    },
    whole : function()
    {
        return this.firstName + " " + this.lastName + " " + this.id;
    }
};
document.getElementById("demo1").innerHTML = "不加括号输出函数表达式："  +
person.fullName;
document.getElementById("demo2").innerHTML = "加括号输出函数执行结果："  +
 person.fullName();
document.getElementById("demo3").innerHTML = "白洞什么时候被证实：" +
person.whole();
</script>

</body>
```

对象属性和方法之间都需要逗号

创建和使用对象方法。

对象方法是一个函数定义,并作为一个属性值存储。

不加括号输出函数表达式：function() { return this.firstName + " " + this.lastName; }

加括号输出函数执行结果：John Doe

白洞什么时候被证实：John Doe 5566

通常 fullName() 是作为 person 对象的一个方法， fullName 是作为一个属性。
如果使用 fullName 属性，不添加 ()，它会返回函数的定义：

13.JavaScript函数

javascript函数语法

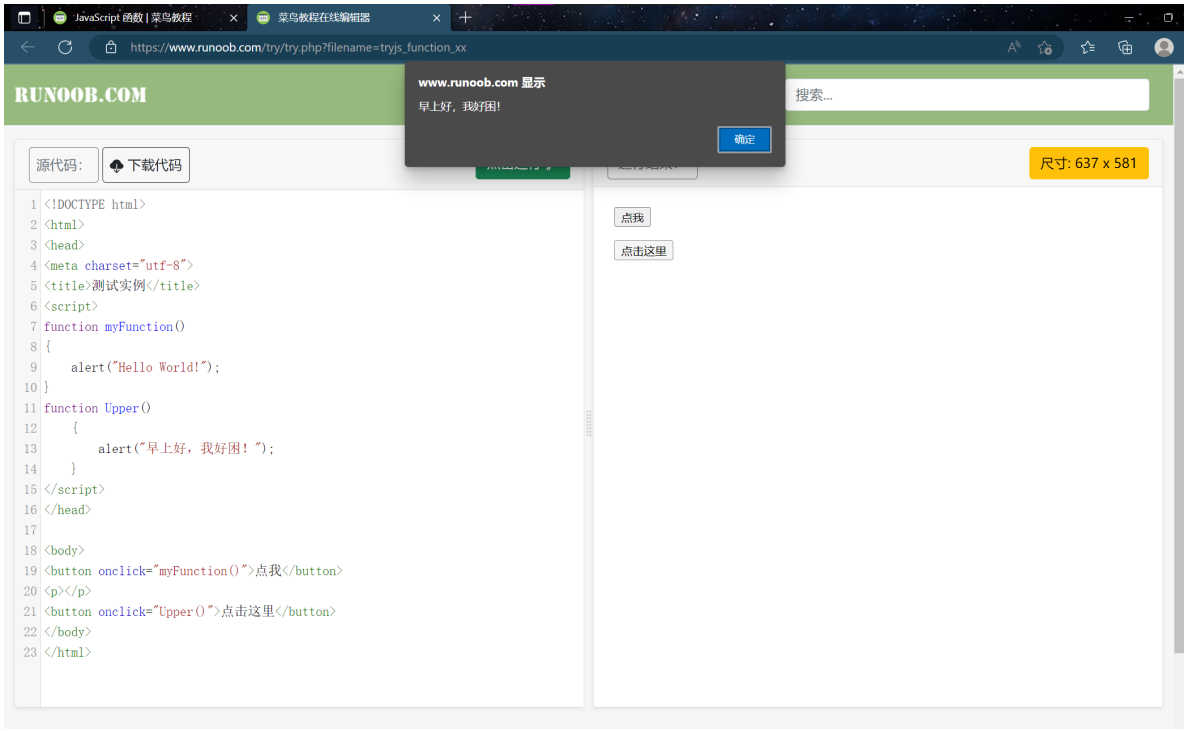
函数就是包裹在花括号中的代码块，前面使用了关键词 function：

```
function functionname()
{
    // 执行代码
}
```

当调用该函数时，会执行函数内的代码。

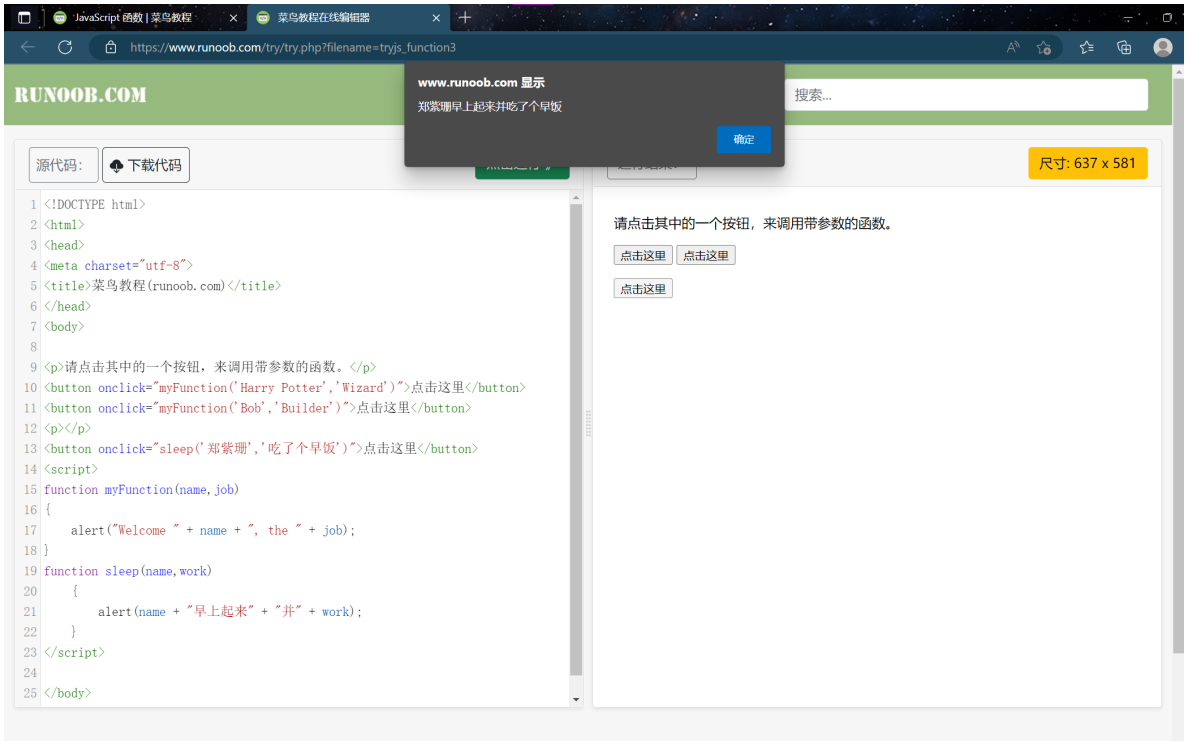
可以在某事件发生时直接调用函数（比如当用户点击按钮时），并且可由 JavaScript 在任何位置进行调用。

```
<head>
<meta charset="utf-8">
<title>测试实例</title>
<script>
function myFunction()
{
    alert("Hello World!");
}
function Upper()
    {
        alert("早上好，我好困！");
    }
</script>
</head>

<body>
<button onclick="myFunction()">点我</button>
<p></p>
<button onclick="Upper()">点击这里</button>
</body>
```



含参数的函数调用

```
<body>

<p>请点击其中的一个按钮，来调用带参数的函数。</p>
<button onclick="myFunction('Harry Potter','Wizard')">点击这里</button>
<button onclick="myFunction('Bob','Builder')">点击这里</button>
<p></p>
<button onclick="sleep('郑紫珊','吃了个早饭')">点击这里</button>
<script>
function myFunction(name,job)
{
    alert("Welcome " + name + ", the " + job);
}
function sleep(name,work)
    {
        alert(name + "早上起来" + "并" + work);
    }
</script>

</body>
```



带返回值的函数

有时，我们会希望函数将值返回调用它的地方。

通过使用 return 语句就可以实现。

在使用 return 语句时，函数会停止执行，并返回指定的值。

语法

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

上面的函数会返回值 5。

注意： 整个 JavaScript 并不会停止执行，仅仅是函数。JavaScript 将继续执行代码，从调用函数的地方。

函数调用将被返回值取代：

```
var myVar=myFunction();
```

myVar 变量的值是 5，也就是函数 "myFunction()" 所返回的值。

即使不把它保存为变量，您也可以使用返回值：

```
document.getElementById("demo").innerHTML=myFunction();
```

"demo" 元素的 innerHTML 将成为 5，也就是函数 "myFunction()" 所返回的值。

您可以使返回值基于传递到函数中的参数：

```
<body>

<p>本例调用的函数会执行一个计算，然后返回结果：</p>
<p id="demo"></p>
<script>
function myFunction(a,b){
    return a*b;
}
document.getElementById("demo").innerHTML=myFunction(4,3);
document.write(myFunction(3,4));
</script>

</body>
```

本例调用的函数会执行一个计算，然后返回结果：

12

12

return可用于退出函数，结束循环

```
function myFunction(a,b)
{
    if (a>b)
    {
        return;
    }
    x=a+b
}
//如果 a 大于 b，则上面的代码将退出函数，并不会计算 a 和 b 的总和。
```

局部/全局javascript变量

局部 JavaScript 变量
在 JavaScript 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，本地变量就会被删除。

全局 JavaScript 变量
在函数外声明的变量是全局变量，网页上的所有脚本和函数都能访问它。

JavaScript 变量的生存期
JavaScript 变量的生命期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。

向未声明的 JavaScript 变量分配值
如果您把值赋给尚未声明的变量，该变量将被自动作为 window 的一个属性。

这条语句:

carname="Volvo";
将声明 window 的一个属性 carname。

非严格模式下给未声明变量赋值创建的全局变量，是全局对象的可配置属性，可以删除。

var var1 = 1; // 不可配置全局属性
var2 = 2; // 没有使用 var 声明，可配置全局属性

console.log(this.var1); // 1

```
console.log(window.var1); // 1
console.log(window.var2); // 2

delete var1; // false 无法删除
console.log(var1); //1

delete var2;
console.log(delete var2); // true
console.log(var2); // 已经删除 报错变量未定义
```