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

引用数据类型：对象(Object)、数组(Array)、函数(Function)。

//对象方法定义

methodName : function() {

// 代码

}

name =objectName.methodName() //调用对象方法返回给name变量

name =objectName.methodName //将对象方法赋给name变量

// 此处可调用 carName 变量，也可使用 window.carName

function myFunction() {

carName = "Volvo";

// 此处可调用 carName 变量

}

// 此处不可调用 carName 变量

function myFunction() {

var carName = "Volvo";

// 此处可调用 carName 变量

}

在 HTML 中, 全局变量是 window 对象: 所有数据变量都属于 window 对象。

var txt = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

var sln = txt.length; //字符串长度

var x = "John";

var y = new String("John");

typeof x // 返回 String

typeof y // 返回 Object

=== 为绝对相等，即数据类型与值都必须相等。



把两个或多个字符串变量连接起来，请使用 + 运算符。

"Hello"+5 结果为"Hello5"

//遍历对象属性

var person={fname:"Bill",lname:"Gates",age:56}; for (x in person) // x 为属性名 { txt=txt + person[x]; }

//list是一个代码块，break+代码块名可以跳出此代码块

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>");

}

//可以设置为 null 和undefined来清空对象:

var person = null;           // 值为 null(空), 但类型为对象

var person = undefined;     // 值为 undefined, 类型为 undefined

typeof undefined             // undefined  
typeof null                  // object  
null === undefined           // false  
null == undefined            // true，即它俩值相等，但类型不等

如果对象是 JavaScript Array 或 JavaScript Date ，我们就无法通过 **typeof** 来判断他们的类型，因为都是 返回 object。

object分为Object、Date、Array

debugger 和设置断点可以中止程序，从而可以调试程序。

10==“10“ 为true； 但是 10===”10”为false

**约束验证 HTML 输入属性：required** ，min，max，disabled，pattern，type 等属性

使用Obj.checkValidity()来检测输入符不符合这些约束规则(返回true或false)。Obj.validationMessage属性返回错误信息

validity 属性包含很多属性: rangeOverflow, patternMismatch, valid等(都是boolean类型的)

如document.getElementById("id1").validity.rangeOverflow

在方法中，this 表示该方法所属的对象。

如果单独使用，this 表示全局对象。

在函数中，this 表示全局对象。

在函数中，在严格模式下，this 是未定义的(undefined)。

在事件中，this 表示接收事件的元素。

类似 call() 和 apply() 方法可以将 this 引用到任何对象。

let的块级作用域

{

var x = 2;

}

// 这里可以使用 x 变量

{

let x = 2;

}

// 这里不能使用 x 变量

使用 let 关键字声明的全局作用域变量不属于 window 对象:

let carName = "Volvo";

// 不能使用 window.carName 访问变量

使用 const 定义的对象或者数组，其实是可变的

// 创建常量对象 const car = {type:"Fiat", model:"500", color:"white"};

// 修改属性: car.color = "red";

car = {type:"Volvo", model:"EX60", color:"red"}; // 重新对象赋值错误

var声明的变量可以先使用再声明，但let、const不行

//json格式解析为js对象

var text = '{ "sites" : [' +

'{ "name":"Runoob" , "url":"www.runoob.com" },' +

'{ "name":"Google" , "url":"www.google.com" },' +

'{ "name":"Taobao" , "url":"www.taobao.com" } ]}';

obj = JSON.parse(text);

href="#id" //定位到网页上id的位置

href="javascript:void(0)" //创建一个死链接

子线程有一个局限：一旦发射了以后就会与主线程失去同步，我们无法确定它的结束，如果结束之后需要处理一些事情，比如处理来自服务器的信息，我们是无法将它合并到主线程中去的。

为了解决这个问题，JavaScript 中的异步操作函数往往通过回调函数来实现异步任务的结果处理。

setTimeout(function () {

document.getElementById("demo").innerHTML="RUNOOB!";

}, 3000);

//Promise 对象：参数是一个函数，这个函数在构造之后会直接被**异步运行**，所以我们称之为起始函数。该函数的参数resolve 和 reject 都是函数，其中调用 resolve 代表一切正常，reject 是出现异常时所调用的：

new Promise(function (resolve, reject) {

// 要做的事情...

});

Promise 类有 .then() .catch() 和 .finally() 三个方法。resolve() 中可以放置一个参数用于向下一个 then 传递一个值，then 中的函数也可以返回一个值传递给 then

new Promise(function (resolve, reject) {

setTimeout(function () {

console.log("First");

resolve();

}, 1000);

}).then(function () {

return new Promise(function (resolve, reject) {

setTimeout(function () {

console.log("Second");

resolve();

}, 4000);

});

}).then(function () {

setTimeout(function () {

console.log("Third");

}, 3000);

});

//改进

function print(delay, message) {

return new Promise(function (resolve, reject) {

setTimeout(function () {

console.log(message);

resolve();

}, delay);

});

}

print(1000, "First").then(function () {

return print(4000, "Second");

}).then(function () {

print(3000, "Third");

});

//再改进

async function asyncFunc() {

await print(1000, "First");

await print(4000, "Second");

await print(3000, "Third");

}

asyncFunc();

//自调用函数：

(function () {

var x = "Hello!!"; // 我将调用自己

})();

arguments.length 属性返回函数调用过程接收到的参数个数

toString()方法可以把函数转换成字符串

箭头函数：

(参数1, 参数2, …, 参数N) => { 函数声明 }

const x = (x, y) => x \* y;

const x = (x, y) => { return x \* y };

//闭包

var add = (function () {

var counter = 0;

return function () {return counter += 1;}

})();

add();

add();

add();

事件冒泡或事件捕获：冒泡先触发自己元素的事件，再触发外层的；捕获相反。

创建新的 HTML 元素 (节点) - appendChild()，insertBefore()，removeChild()和replaceChild()

<div id="div1">

<p id="p1">这是一个段落。</p>

<p id="p2">这是另外一个段落。</p>

</div>

<script>

var para = document.createElement("p");

var node = document.createTextNode("这是一个新的段落。");

para.appendChild(node);

var element = document.getElementById("div1");

element.appendChild(para);

</script>

window.navigator 对象包含有关访问者浏览器的信息，来自 navigator 对象的信息具有误导性，不应该被用于检测浏览器版本

jQuery框架库：引用下面这个

<script src="https://cdn.staticfile.org/jquery/1.8.3/jquery.min.js"></script>

主要的 jQuery 函数是 $() 函数