目录

[JavaScript 1](#_Toc10959994)

[内部JS 1](#_Toc10959995)

[外部JS 1](#_Toc10959996)

[变量（Variable） 2](#_Toc10959997)

[变量的不同数据类型 2](#_Toc10959998)

[Array 3](#_Toc10959999)

[object 3](#_Toc10960000)

[注释 3](#_Toc10960001)

[运算符 3](#_Toc10960002)

[字符串 4](#_Toc10960003)

[检索特定字符串字符 5](#_Toc10960004)

[indexOf() 5](#_Toc10960005)

[slice() 5](#_Toc10960006)

[替换字符串replace 6](#_Toc10960007)

[数组 6](#_Toc10960008)

[字符串和数组转换 6](#_Toc10960009)

[添加和删除数组项 6](#_Toc10960010)

[条件语句 7](#_Toc10960011)

[If…else 7](#_Toc10960012)

[switch语句 7](#_Toc10960013)

[函数（function） 8](#_Toc10960014)

[定义函数 8](#_Toc10960015)

[事件 8](#_Toc10960016)

[Loop循环 9](#_Toc10960017)

[对象object 9](#_Toc10960018)

[添加图像切换器 10](#_Toc10960019)

[添加个性化欢迎信息 10](#_Toc10960020)

[对象基础 12](#_Toc10960021)

[object()构造函数 12](#_Toc10960022)

[create() 12](#_Toc10960023)

[原型prototype 12](#_Toc10960024)

# JavaScript

JS是一门动态编程脚本语言，为网站添加动态交互功能。

内部JS

打开.html文件在<head>里添加以下代码

<script>

// 在此编写 JavaScript 代码

</script>

外部JS

<script src="scripts.js" async></script>

也可以将<script>放在 body底部。因为浏览器会按照代码在文件中的顺序解析 HTML。如果先加载的 JavaScript 期望修改其下方的 HTML，那么它可能由于 HTML 尚未被加载而失效。

**async与defer**

async脚本之间彼此独立，且不依赖于本页面的其它任何脚本

defer脚本将按照在页面中出现的顺序加载和运行

let myHeading = document.querySelector('h1');

myHeading.textContent = 'Hello world!';

结果是页面标题改成了“Hello World”

**querySelector(‘h1’)**函数获取（选择）标题，把标题的引用储存在**myHeading变量**中，之后把myHeading变量的**属性textContent**修改为Hello World。



# 变量（Variable）

变量是存储值的容器，**声明（定义）变量**先输入关键字**let**：

let myVariable;

var myName;

JS对大小写敏感，myVariable和myvariable是不同的。

变量定义后可以进行**赋值**：

myVariable = '李雷';

也可以将定义和赋值写在同一行：

let myVariable = '李雷';

赋值之后可以直接通过变量名取得变量值，即通过myVariable取得李雷

变量在赋值后是可以更改：

let myVariable = '李雷';

myVariable = '韩梅梅';

变量值由李雷变为韩梅梅

变量的不同数据类型

|  |  |  |
| --- | --- | --- |
| 类型 | 解释 | 示例 |
| String | 字符串。字符串的值必须用引号（单双都可）扩起来 | let myVariable =‘李雷’ ； |
| Number | 数字。无需引号 | let myVariable =10； |
| Boolean | 布尔值（true/false）。 true/false是JS里的特殊关键字，无需引号 | let myVariable =true; |
| Array | 数组。在单一引用中存储多个值 | let myVariable =[1,’李雷’,’韩梅梅’,10];  元素引用方法：myVariable[0]，myVariable[1]…… |
| Object | 对象。JavaScript里一切皆对象，一切都能储存在变量里 | let myVariable=document.querySelector(‘h1’);以及上面所有示例都是对象 |

Array

单个对象包含多个值

var myNameArray = ['Chris', 'Bob', 'Jim'];

访问各自值（从0开始）

myNameArray[0]; // should return 'Chris'

object

单个对象的多种信息

var dog = { name : 'Spot', breed : 'Dalmatian' };

检索方法

dog.name

注释

/\* \*/可用于跨行注释 //用于单行注释

运算符

|  |  |  |  |
| --- | --- | --- | --- |
| 运算符 | 解释 | 符号 | 示例 |
| 加 | 两个数字相加，或拼接两个字符串 | + | 6+9；  ‘Hello‘+‘World’； |
| 减、乘、除 | 数字间 | -，\*，/ | 9-6；  6\*9；  9/6； |
| 赋值运算符 | 赋值变量 | = | let myVariable=’李雷’ |
| 等于 | 测试两个值是否相等，并返回一个布尔（true/false）值 | === | let myVariable=3;  myVariable===4;  //false |
| 不等于 | 同上 | !== | let myVariable=3;  myVariable!==4;  //true |
| 取非 | 返回逻辑相反的值 | ! | 原式为真，则经取非后的值为false：  let myVariable=3;  !(myVariable===3);  //false |

var x = 3; // x 的值是 3

var y = 4; // y 的值是 4

x = y; // x 和 y 有相同的值, 4(赋值)

| **运算符** | **名称** | **作用** | **示例** | **等价于** |
| --- | --- | --- | --- | --- |
| += | 递增赋值 | 右边的数值加上左边的变量，然后再返回新的变量。 | x = 3; x += 4; | x = 3; x = x + 4; |
| -= | 递减赋值 | 左边的变量减去右边的数值，然后再返回新的变量。 | x = 6; x -= 3; | x = 6; x = x - 3; |
| \*= | 乘法赋值 | 左边的变量乘以右边的数值，然后再返回新的变量。 | x = 2; x \*= 3; | x = 2; x = x \* 3; |
| /= | 除法赋值 | 左边的变量除以右边的数值，然后再返回新的变量。 |  |  |

# 字符串

字符串可以由单引号或双引号包裹，但不能在字符串中包含相同的引号

可使用**转义字符**

var bigmouth = 'I\'ve got no right to take my place...';

[Number](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Number) 对象将把传递给它的任何东西转换成一个数字

var myString = '123';

var myNum = Number(myString);

typeof myNum;

[toString()](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number/toString) 的方法，它将把它转换成等价的字符串

检索特定字符串字符

var browserType = 'mozilla';

browserType[0];

方括号内，您可以包含要返回的字符的编号（从0开始）

browserType[browserType.length-1];

表示检索字符串的最后一个字符

indexOf()

browserType.indexOf('zilla');

结果是2

当主字符串中找不到子字符串时返回-1

if(browserType.indexOf('mozilla') !== -1) {

// do stuff with the string

}

slice()

browserType.slice(0,3);

返回moz

browserType.slice(2);

返回zilla因为2的字符位置是字母z，并且因为没有包含第二个参数，所以返回的子字符串是字符串中的所有剩余字符

[toLowerCase()](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/String/toLowerCase)和[toUpperCase()](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/String/toUpperCase" \o "toUpperCase() 将调用该方法的字符串值转换为大写形式，并返回。)字符串并将所有字符分别转换为小写或大写

var radData = 'My NaMe Is MuD';

radData.toLowerCase();

radData.toUpperCase();

替换字符串replace

browserType = browserType.replace('moz','van');

# 数组

多维数组

random[2][2];

**获取数组长度**

sequence.length;

字符串和数组转换

[split()](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/String/split)

var myArray = myData.split(',');

myArray;

[join()](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Array/join)

var myNewString = myArray.join(',');

myNewString;

join() 可以指定不同的分隔符

var dogNames = ["Rocket","Flash","Bella","Slugger"];

dogNames.toString(); //Rocket,Flash,Bella,Slugger

添加和删除数组项

**添加在末尾**

var myArray.push('Bradford', 'Brighton');

**删除末尾项**

var removedItem = myArray.pop();

[unshift()](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Array/unshift) 和 [shift()](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/Array/shift) 作用于数组开始

计算长度

var newLength = myArray.push('Bristol');

# 条件语句

If…else

if (iceCream === 'chocolate') {

alert('我最喜欢巧克力冰激淋了。');

} else if(iceCream === 'strawberry') {

alert('草莓我也喜欢');

} else {

alert('但是巧克力才是我的最爱啊');

}

If（…）中表达式为条件，若返回true则运行第一个代码块，否则运行else之后的代码块。

* && — 逻辑与; 使得并列两个或者更多的表达式成为可能，只有当这些表达式每一个都返回true时，整个表达式才会返回true.
* || — 逻辑或; 当两个或者更多表达式当中的任何一个返回 true 则整个表达式将会返回 true.
* !  — 逻辑非; 对一个布尔值取反, 非true返回false,非false返回true.

switch语句

switch (choice) {

case 'sunny':

para.textContent = 'It is nice and sunny outside today.';

break;

case 'rainy':

para.textContent = 'Rain is falling outside.';

break;

default:

para.textContent = '';

}

# 函数（function）

如querySelector()和alert()是浏览器内置函数，()内的内容为**参数**。

**alert()**函数在浏览器窗口内弹出一个**警告框**，其中的字符串参数为警告框内显示的内容。

**typeof()**的特殊函数返回所传递给它的变量的数据类型

定义函数

function multiply(num1, num2) {

let result = num1 \* num2;

return result;

}

 return语句告诉浏览器当前函数返回result变量。这是一点很有必要，因为函数内定义的变量只能在函数内使用。

# 事件

如点击事件，鼠标点击操作会触发该事件。

document.querySelector('html').onclick = function() {

alert('别戳我，我怕疼。');

}

这里选用<html>元素，把一个匿名函数（未命名函数，这里的匿名函数包含单击鼠标时要运行的代码）赋值给了html的**onclick属性**。

document.querySelector('html').onclick = function() {};

等价于

let myHTML = document.querySelector('html');

myHTML.onclick = function() {};

# Loop循环

for (let i = 1; i < 21; i++) { console.log(i); }

1. 起始值②退出条件③增加器（console.log()在控制台打印出每次迭代时变量 i 的值）

let resetParas = document.querySelectorAll('.resultParas p');

for (let i = 0 ; i < resetParas.length ; i++) {

resetParas[i].textContent = '';

}

[querySelectorAll()](https://developer.mozilla.org/zh-CN/docs/Web/API/Document/querySelectorAll) 方法创建了一个包含 <div class="resultParas"> 内所有段落的变量，然后通过循环迭代，删除每个段落的文本内容。

# 对象object

guessField.focus();

focus() 是 [<input>](https://developer.mozilla.org/zh-CN/docs/Web/HTML/Element/input) 元素可用方法之一，因此我们可以使用这行代码将光标聚焦于此文本框上

guessField.value = 'Hello';

段落中不存在value，返回undefined

改变段落中的文本内容， 你需要用 [textContent](https://developer.mozilla.org/zh-CN/docs/Web/API/Node/textContent" \o "Node.textContent 属性表示一个节点及其后代的文本内容。) 属性来代替 value

guesses.textContent = '我的段落在哪里？';

每个元素都有一个 style 属性，它本身包含一个对象，其属性包含应用于该元素的所有内联 CSS 样式

guesses.style.backgroundColor = 'yellow';

guesses.style.fontSize = '200%';

guesses.style.padding = '10px';

guesses.style.boxShadow = '3px 3px 6px black';

# 添加图像切换器

用JS使图片在点击时进行切换。

let myImage = document.querySelector('img');

myImage.onclick = function() {

let mySrc = myImage.getAttribute('src');

if(mySrc === 'images/firefox-icon.png') {

myImage.setAttribute('src', 'images/firefox2.png');

} else {

myImage.setAttribute('src', 'images/firefox-icon.png');

}

}

1. 将<img>元素的引用存放在变量myImage里，再将这个变量的onclick事件与一个匿名函数绑定。
2. 点击图片时，会**获取这张图片的src属性值（getAttribute(‘src’)）**
3. 用一个条件句来判断src的值是否等于原始图像的路径，

如果是，则将src的值改为等二张图片的路径，并在<img>内加载该图片

如果不是，则把src的值重新设置为原始图片的路径。即点击两张图片循环切换。

# 添加个性化欢迎信息

在网页标题中添加用户名字，名字由Web Storage API保存下来，即使用户关闭页面之后再重新打开仍可得到之前的信息，可根据用户名字改变而更新信息。

function setHeading(name) {

let myHeading = document.querySelector('h1');

myHeading.textContent = 'Mozilla 酷毙了，' + name + '！';

}

function setUserName() {

let myName = prompt('请输入你的名字');

localStorage.setItem('name', myName);

setHeading(myName);

}

let storedName = localStorage.getItem('name');

if(!storedName) {

setUserName();

} else {

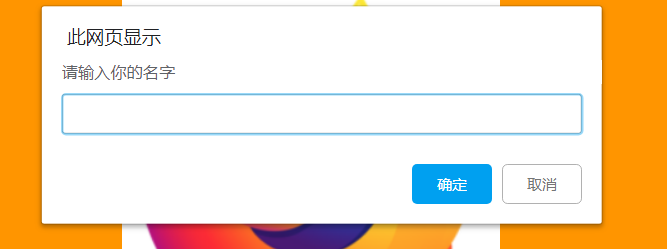
setHeading(storedName);

}

let myButton = document.querySelector('button');

myButton.onclick = setUserName;

1. 讲h1中元素的引用保存在myHeading变量中，然后设置变量myHeading的**textContent属性**。
2. setUserName函数将为页面设置用户名信息，先调用prompt()函数，与**alert()**类似，会弹出一个对话框，但是这里需要用户**输入数据**，并保存在myName变量里。



调用**localStorage API**，他可以将数据存储在浏览器中供后续获取。这里用localStorage的**setItem()函数**来创建一个‘name’数据项，并把myName变量复制给他。最后调用上述的setHeading函数来个性化欢迎信息。

1. let storedName=localStorage.getItem(‘name’);尝试取得浏览器中保存的‘name’数据项。
2. **if…else**是**初始化代码**（在页面初次读取时进行构造工作），条件**!storedName**在storedName不存在时值为真，此时调用setUserName函数，若条件为假时以storedName为参数调用setHeading函数。

此条件语句块可描述为：若 'name' 数据项不存在，则由用户输入用户名，若存在则直接将 'name' 中保存的用户名显示到个性化欢迎信息中。

1. button 元素的引用保存在 myButton 中，将这个按钮的 onclick 事件处理器设为 setUserName 函数。

注：用户名是保存在 localStorage 里的，网站关闭后也不会丢失，所以重新打开浏览器时所设置的名字信息将依然存在！

# 对象基础

var objectName = {

member1Name : member1Value,

member2Name : member2Value,

member3Name : member3Value

}

一个对象由许多成员组成，每个成员都拥有名字和值，成员之间用逗号分隔

**点表示法**访问对象属性

person.bio()

**括号表示法**

person.name.first

person['name']['first']

**this**指向了代码所在的对象(其实代码运行时所在的对象)

object()构造函数

var person1 = new Object({

name : 'Chris',

age : 38,

greeting : function() {

alert('Hi! I\'m ' + this.name + '.');

}}

);

create()

var person2 = Object.create(person1);

person2是基于person1创建的， 它们具有相同的属性和方法

原型prototype

每个函数都有一个特殊的属性叫作原型（prototype）

function doSomething(){}

doSomething.prototype.foo = "bar";

var doSomeInstancing = new doSomething();

doSomeInstancing.prop = "some value"; // add a property onto the object

console.log( doSomeInstancing );

定义一个构造器函数

function Person(first, last, age, gender, interests) {

// 属性与方法定义

};

创建对象实例

var person1 = new Person('Bob', 'Smith', 32, 'male', ['music', 'skiing']);