# 第1章 JavaScript 教程

JavaScript 是属于 HTML 和 Web 的编程语言。

编程令计算机完成您需要它们做的工作。

JavaScript 很容易学习。

本教程涵盖 JavaScript 基础和高级教程。

为何学习 JavaScript？

JavaScript 是 web 开发者必学的三种语言之一：

HTML 定义网页的内容

CSS 规定网页的布局

JavaScript 对网页行为进行编程

本教程提供关于 JavaScript，以及 JavaScript 如何与 HTML 和 CSS 协同工作的知识。

您知道吗？

JavaScript 和 Java 是完全不同的语言，不论是概念还是设计。

JavaScript 在 1995 年由 Brendan Eich 发明，并于 1997 年成为一部 ECMA 标准。

ECMA-262 是其官方名称。ECMAScript 6 （发布于 2015 年）是最新的 JavaScript 版本。

学习速度

在本教程中，您可以自由控制学习速度。

一切都取决于您。

如果您遇到困难，请休息一会儿，然后重新读一读材料。

请始终确保您理解了“亲自试一试”中的实例和练习题。

JavaScript 参考手册

W3School 提供并维护了完整的 JavaScript 参考手册，包括所有 HTML DOM 以及浏览器对象。

本参考手册包含了所有对象、属性及方法的示例，并根据最新 web 标准持续更新。

# 第2章 JavaScript 简介

JavaScript 能够改变 HTML 内容

getElementById() 是多个 JavaScript HTML 方法之一。

本例使用该方法来“查找” id="demo" 的 HTML 元素，并把元素内容（innerHTML）更改为 "Hello JavaScript"：

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 能做什么</h2>

<p id="demo">JavaScript 能够改变 HTML 内容。</p>

<button type="button" onclick='document.getElementById("demo").innerHTML = "Hello JavaScript!"'>点击我！</button>

</body>

</html>

提示：JavaScript 同时接受双引号和单引号：

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript 能做什么？</h1>

<p id="demo">JavaScript 可以更改 HTML 内容。</p>

<button type="button" onclick="document.getElementById('demo').innerHTML = 'Hello JavaScript!'">单击这里</button>

</body>

</html>

JavaScript 能够改变 HTML 属性

本例通过改变 <img> 标签的 src 属性（source）来改变一张 HTML 图像：

灯泡

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 能做什么？</h2>

<p>JavaScript 能够改变 HTML 属性值。</p>

<p>在本例中，JavaScript 改变了图像的 src 属性值。</p>

<button onclick="document.getElementById('myImage').src='/i/eg\_bulbon.gif'">开灯</button>

<img id="myImage" border="0" src="/i/eg\_bulboff.gif" style="text-align:center;">

<button onclick="document.getElementById('myImage').src='/i/eg\_bulboff.gif'">关灯</button>

</body>

</html>

JavaScript 能够改变 HTML 样式 (CSS)

改变 HTML 元素的样式，是改变 HTML 属性的一种变种：

实例

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 能够做什么</h2>

<p id="demo">JavaScript 能够改变 HTML 元素的样式。</p>

<button type="button" onclick="document.getElementById('demo').style.fontSize='35px'">

点击我！

</button>

</body>

</html>

JavaScript 能够隐藏 HTML 元素

可通过改变 display 样式来隐藏 HTML 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 能够做什么</h2>

<p id="demo">JavaScript 能够隐藏 HTML 元素。</p>

<button type="button" onclick="document.getElementById('demo').style.display='none'">

点击我！

</button>

</body>

</html>

JavaScript 能够显示 HTML 元素

可通过改变 display 样式来显示隐藏的 HTML 元素：

实例

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 能够做什么</h2>

<p>JavaScript 能够显示隐藏的 HTML 元素。</p>

<p id="demo" style="display:none">Hello JavaScript!</p>

<button type="button" onclick="document.getElementById('demo').style.display='block'">

点击我！

</button>

</body>

</html>

# 第3章 JavaScript 使用

<script> 标签

在 HTML 中，JavaScript 代码必须位于 <script> 与 </script> 标签之间。

实例

<!DOCTYPE html>

<html>

<body>

<h2>Body 中的 JavaScript</h2>

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML = "我的第一段 JavaScript";

</script>

</body>

</html>

注释：旧的 JavaScript 例子也许会使用 type 属性：<script type="text/javascript">。

注释：type 属性不是必需的。JavaScript 是 HTML 中的默认脚本语言。

JavaScript 函数和事件

JavaScript 函数是一种 JavaScript 代码块，它可以在调用时被执行。

例如，当发生事件时调用函数，比如当用户点击按钮时。

提示：您将在稍后的章节学到更多有关函数和事件的知识。

<head> 或 <body> 中的 JavaScript

您能够在 HTML 文档中放置任意数量的脚本。

脚本可被放置与 HTML 页面的 <body> 或 <head> 部分中，或兼而有之。

<head> 中的 JavaScript

在本例中，JavaScript 函数被放置于 HTML 页面的 <head> 部分。

该函数会在按钮被点击时调用：

实例

<!DOCTYPE html>

<html>

<head>

<script>

function myFunction() {

document.getElementById("demo").innerHTML = "段落已被更改。";

}

</script>

</head>

<body>

<h2>Head 中的 JavaScript</h2>

<p id="demo">一个段落。</p>

<button type="button" onclick="myFunction()">试一试</button>

</body>

</html>

<body> 中的 JavaScript

在本例中，JavaScript 函数被放置于 HTML 页面的 <body> 部分。

该函数会在按钮被点击时调用：

实例

<!DOCTYPE html>

<html>

<body>

<h2>Body 中的 JavaScript</h2>

<p id="demo">一个段落。</p>

<button type="button" onclick="myFunction()">试一试</button>

<script>

function myFunction() {

document.getElementById("demo").innerHTML = "段落已被更改。";

}

</script>

</body>

</html>

提示：把脚本置于 <body> 元素的底部，可改善显示速度，因为脚本编译会拖慢显示。

外部脚本

脚本可放置与外部文件中：

外部文件：myScript.js

function myFunction() {

document.getElementById("demo").innerHTML = "段落被更改。";

}

外部脚本很实用，如果相同的脚本被用于许多不同的网页。

JavaScript 文件的文件扩展名是 .js。

如需使用外部脚本，请在 <script> 标签的 src (source) 属性中设置脚本的名称：

实例

<!DOCTYPE html>

<html>

<body>

<h2>外部 JavaScript</h2>

<p id="demo">一个段落。</p>

<button type="button" onclick="myFunction()">试一试</button>

<p>（myFunction 存储在名为 "myScript.js" 的外部文件中。）</p>

<script src="/demo/myScript.js"></script>

</body>

</html>

您可以在 <head> 或 <body> 中放置外部脚本引用。

该脚本的表现与它被置于 <script> 标签中是一样的。

注释：外部脚本不能包含 <script> 标签。

外部 JavaScript 的优势

在外部文件中放置脚本有如下优势：

分离了 HTML 和代码

使 HTML 和 JavaScript 更易于阅读和维护

已缓存的 JavaScript 文件可加速页面加载

如需向一张页面添加多个脚本文件 - 请使用多个 script 标签：

实例

<script src="myScript1.js"></script>

<script src="myScript2.js"></script>

外部引用

可通过完整的 URL 或相对于当前网页的路径引用外部脚本：

本例使用完整的 URL 来链接至脚本：

实例

<!DOCTYPE html>

<html>

<body>

<h1>外部 JavaScript</h1>

<p id="demo">一个段落。</p>

<button type="button" onclick="myFunction()">试一试</button>

<p>（myFunction 存储在名为 "myScript.js" 的外部文件中。）</p>

<script src="https://www.w3school.com.cn/demo/myScript.js"></script>

</body>

</html>

本例使用了位于当前网站上指定文件夹中的脚本：

实例

<!DOCTYPE html>

<html>

<body>

<h2>外部 JavaScript</h2>

<p id="demo">一个段落。</p>

<button type="button" onclick="myFunction()">试一试</button>

<p>（myFunction 存储在名为 "myScript.js" 的外部文件中。）</p>

<script src="/demo/myScript.js"></script>

</body>

</html>

本例链接了与当前页面相同文件夹的脚本：

实例

<!DOCTYPE html>

<html>

<body>

<h2>外部 JavaScript</h2>

<p id="demo">一个段落。</p>

<button type="button" onclick="myFunction()">试一试</button>

<p>（myFunction 存储在名为 "myScript.js" 的外部文件中。）</p>

<script src="/demo/myScript.js"></script>

</body>

</html>

# 第4章 JavaScript 输出

JavaScript 不提供任何内建的打印或显示函数。

JavaScript 显示方案

JavaScript 能够以不同方式“显示”数据：

使用 window.alert() 写入警告框

使用 document.write() 写入 HTML 输出

使用 innerHTML 写入 HTML 元素

使用 console.log() 写入浏览器控制台

使用 innerHTML

如需访问 HTML 元素，JavaScript 可使用 document.getElementById(id) 方法。

id 属性定义 HTML 元素。innerHTML 属性定义 HTML 内容：

实例

<!DOCTYPE html>

<html>

<body>

<h2>我的第一张网页</h2>

<p>我的第一个段落。</p>

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML = 5 + 6;

</script>

</body>

</html>

提示：更改 HTML 元素的 innerHTML 属性是在 HTML 中显示数据的常用方法。

使用 document.write()

出于测试目的，使用 document.write() 比较方便：

实例

<!DOCTYPE html>

<html>

<body>

<h2>我的第一张网页</h2>

<p>我的第一个段落。</p>

<script>

document.write(5 + 6);

</script>

</body>

</html>

注意：在 HTML 文档完全加载后使用 document.write() 将删除所有已有的 HTML ：

实例

<!DOCTYPE html>

<html>

<body>

<h2>我的第一张网页</h2>

<p>我的第一个段落。</p>

<button type="button" onclick="document.write(5 + 6)">试一试</button>

</body>

</html>

提示：document.write() 方法仅用于测试。

使用 window.alert()

您能够使用警告框来显示数据：

实例

<!DOCTYPE html>

<html>

<body>

<h2>我的第一张网页</h2>

<p>我的第一个段落。</p>

<script>

window.alert(5 + 6);

</script>

</body>

</html>

使用 console.log()

在浏览器中，您可使用 console.log() 方法来显示数据。

请通过 F12 来激活浏览器控制台，并在菜单中选择“控制台”。

实例

<!DOCTYPE html>

<html>

<body>

<h2>按 F12 启动</h2>

<p>在 debugger 中选择 "Console"。然后再次点击运行按钮。</p>

<script>

console.log(5 + 6);

</script>

</body>

</html>

# 第5章 JavaScript 语句

在 HTML 中，JavaScript 语句是由 web 浏览器“执行”的“指令”。

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 语句</h2>

<p><b>JavaScript 程序</b> 是一系列由计算机执行的 <b>语句</b>。</p>

<p id="demo"></p>

<script>

var x, y, z; // 语句 1

x = 22; // 语句 2

y = 11; // 语句 3

z = x + y; // 语句 4

document.getElementById("demo").innerHTML =

"z 的值是" + z + "。";

</script>

</body>

</html>

JavaScript 程序

计算机程序是由计算机“执行”的一系列“指令”。

在编程语言中，这些编程指令被称为语句。

JavaScript 程序就是一系列的编程语句。

注释：在 HTML 中，JavaScript 程序由 web 浏览器执行。

JavaScript 语句

JavaScript 语句由以下构成：

值、运算符、表达式、关键词和注释。

这条语句告诉浏览器在 id="demo" 的 HTML 元素中输出 "Hello Kitty."：

实例

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 语句</h2>

<p>在 HTML 中，JavaScript 语句由浏览器执行。</p>

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML = "Hello Kitty.";

</script>

</body>

</html>

大多数 JavaScript 程序都包含许多 JavaScript 语句。

这些语句会按照它们被编写的顺序逐一执行。

注释：JavaScript 程序（以及 JavaScript 语句）常被称为 JavaScript 代码。

分号 ;

分号分隔 JavaScript 语句。

请在每条可执行的语句之后添加分号：

a = 5;

b = 6;

c = a + b;

如果有分号分隔，允许在同一行写多条语句：

a = 5; b = 6; c = a + b;

您可能在网上看到不带分号的例子。

提示：以分号结束语句不是必需的，但我们仍然强烈建议您这么做。

JavaScript 空白字符

JavaScript 会忽略多个空格。您可以向脚本添加空格，以增强可读性。

下面这两行是相等的：

var person = "Bill";

var person="Bill";

在运算符旁边（ = + - \* / ）添加空格是个好习惯：

var x = y + z;

JavaScript 行长度和折行

为了达到最佳的可读性，程序员们常常喜欢把代码行控制在 80 个字符以内。

如果 JavaScript 语句太长，对其进行折行的最佳位置是某个运算符：

实例

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 语句</h2>

<p>

折行的最佳位置是运算符或逗号之后。

</p>

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML =

"Hello Kitty.";

</script>

</body>

</html>

JavaScript 代码块

JavaScript 语句可以用花括号（{...}）组合在代码块中。

代码块的作用是定义一同执行的语句。

您会在 JavaScript 中看到成块组合在一起的语句：

实例

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 语句</h2>

<p>JavaScript 代码块在 { 与 } 之间编写。</p>

<button type="button" onclick="myFunction()">点击我！</button>

<p id="demo1"></p>

<p id="demo2"></p>

<script>

function myFunction() {

document.getElementById("demo1").innerHTML = "Hello Kitty.";

document.getElementById("demo2").innerHTML = "How are you?";

}

</script>

</body>

</html>

注释：在本教程中我们为代码块用了 4 个空格的缩进。

提示：您将稍后在本教程中学到更多有关函数的内容。

JavaScript 关键词

JavaScript 语句常常通过某个关键词来标识需要执行的 JavaScript 动作。

下面的表格列出了一部分将在教程中学到的关键词：

|  |  |
| --- | --- |
| 关键词 | 描述 |
| break | 终止 switch 或循环。 |
| continue | 跳出循环并在顶端开始。 |
| debugger | 停止执行 JavaScript，并调用调试函数（如果可用）。 |
| do ... while | 执行语句块，并在条件为真时重复代码块。 |
| for | 标记需被执行的语句块，只要条件为真。 |
| function | 声明函数。 |
| if ... else | 标记需被执行的语句块，根据某个条件。 |
| return | 退出函数。 |
| switch | 标记需被执行的语句块，根据不同的情况。 |
| try ... catch | 对语句块实现错误处理。 |
| var | 声明变量。 |

注释：JavaScript 关键词指的是保留的单词。保留词无法用作变量名。

# 第6章 JavaScript 语法

JavaScript 语法是一套规则，它定义了 JavaScript 的语言结构。

var x, y; // 如何声明变量

x = 7; y = 8; // 如何赋值

z = x + y; // 如何计算值

JavaScript 值

JavaScript 语句定义两种类型的值：混合值和变量值。

混合值被称为字面量（literal）。变量值被称为变量。

JavaScript 字面量

书写混合值最重要的规则是：

写数值有无小数点均可：

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 数值</h2>

<p>写数值有无小数点均可。</p>

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML = 15.90;

</script>

</body>

</html>

字符串是文本，由双引号或单引号包围：

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 字符串</h2>

<p>字符串可由双引号或单引号包围。</p>

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML = 'Bill Gates';

</script>

</body>

</html>

JavaScript 变量

在编程语言中，变量用于存储数据值。

JavaScript 使用 var 关键词来声明变量。

= 号用于为变量赋值。

在本例中，x 被定义为变量。然后，x 被赋的值是 7：

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 变量</h2>

<p>在本例中，x 被定义为变量。然后 x 被赋值 7：</p>

<p id="demo"></p>

<script>

var x;

x = 7;

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

JavaScript 运算符

JavaScript 使用算数运算符（+ - \* /）来计算值：

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 运算符</h2>

<p>JavaScript 使用算数运算符来计算值（类似代数）。</p>

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML = (7 + 8) \* 10;

</script>

</body>

</html>

JavaScript 使用赋值运算符（=）向变量赋值：

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 赋值</h2>

<p>在 JavaScript 中，= 运算符用于向变量赋值。</p>

<p id="demo"></p>

<script>

var x, y;

x = 7;

y = 8;

document.getElementById("demo").innerHTML = x + y;

</script>

</body>

</html>

JavaScript 表达式

表达式是值、变量和运算符的组合，计算结果是值。

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 表达式</h2>

<p>表达式计算为值。</p>

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML = 6 \* 10;

</script>

</body>

</html>

表达式也可包含变量值：

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 表达式</h2>

<p>表达式计算为值。</p>

<p id="demo"></p>

<script>

var x;

x = 6;

document.getElementById("demo").innerHTML = x \* 10;

</script>

</body>

</html>

值可以是多种类型，比如数值和字符串。

例如，"Bill" + " " + "Gates"，计算为 "Bill Gates"：

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 表达式</h2>

<p>表达式计算为值。</p>

<p id="demo"></p>

<script>

document.getElementById("demo").innerHTML = "Bill" + " " + "Gates";

</script>

</body>

</html>

JavaScript 关键词

JavaScript 关键词用于标识被执行的动作。

var 关键词告知浏览器创建新的变量：

<!DOCTYPE html>

<html>

<body>

<h2>var 关键词创建变量</h2>

<p id="demo"></p>

<script>

var x, y;

x = 7 + 8;

y = x \* 10;

document.getElementById("demo").innerHTML = y;

</script>

</body>

</html>

JavaScript 注释

并非所有 JavaScript 语句都被“执行”。

双斜杠 // 或 /\* 与 \*/ 之间的代码被视为注释。

注释会被忽略，不会被执行：

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 注释<b>不会</b>被执行。</h2>

<p id="demo"></p>

<script>

var x;

x = 7;

// x = 8; 我不会被执行

document.getElementById("demo").innerHTML = x;

</script>

</body>

</html>

JavaScript 标识符

标识符是名称。

在 JavaScript 中，标识符用于命名变量（以及关键词、函数和标签）。

在大多数编程语言中，合法名称的规则大多相同。

在 JavaScript 中，首字符必须是字母、下划线（-）或美元符号（$）。

连串的字符可以是字母、数字、下划线或美元符号。

提示：数值不可以作为首字符。这样，JavaScript 就能轻松区分标识符和数值。

JavaScript 对大小写敏感

所有 JavaScript 标识符对大小写敏感。

变量 lastName 和 lastname，是两个不同的变量。

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 对大小写敏感</h2>

<p>请试着把 lastName 改为 lastname。</p>

<p id="demo"></p>

<script>

var lastname, lastName;

lastName = "Gates";

lastname = "Jobs";

document.getElementById("demo").innerHTML = lastName;

</script>

</body>

</html>

JavaScript 不会把 VAR 或 Var 译作关键词 var。

JavaScript 与驼峰式大小写

历史上，程序员曾使用三种把多个单词连接为一个变量名的方法：

连字符：

first-name, last-name, master-card, inter-city.

注释：JavaScript 中不能使用连字符。它是为减法预留的。

下划线：

first\_name, last\_name, master\_card, inter\_city.

驼峰式大小写（Camel Case）：

FirstName, LastName, MasterCard, InterCity.

JavaScript 程序员倾向于使用以小写字母开头的驼峰大小写：

firstName, lastName, masterCard, interCity

JavaScript 字符集

JavaScript 使用 Unicode 字符集。

Unicode 覆盖世界上几乎所有的字符、标点和符号。

如需了解详情，请访问我们完整的 Unicode 参考手册。

# 第7章 JavaScript 注释

JavaScript 注释用于解释 JavaScript 代码，增强其可读性。

JavaScript 注释也可以用于在测试替代代码时阻止执行。

单行注释

单行注释以 // 开头。

任何位于 // 与行末之间的文本都会被 JavaScript 忽略（不会执行）。

本例在每条代码行之前使用单行注释：

<!DOCTYPE html>

<html>

<body>

<h1 id="myH"></h1>

<p id="myP"></p>

<script>

// 改变标题：

document.getElementById("myH").innerHTML = "JavaScript 注释";

// 改变段落：

document.getElementById("myP").innerHTML = "我的第一个段落。";

</script>

</body>

</html>

多行注释

多行注释以 /\* 开头，以 \*/ 结尾。

任何位于 /\* 和 \*/ 之间的文本都会被 JavaScript 忽略。

本例使用多行注释（注释块）来解释代码：

实例

<!DOCTYPE html>

<html>

<body>

<h1 id="myH"></h1>

<p id="myP"></p>

<script>

/\*

下面的代码会改变

id = "myH" 的标题

以及 id = "myp" 的段落

\*/

document.getElementById("myH").innerHTML = "JavaScript 注释";

document.getElementById("myP").innerHTML = "我的第一个段落。";

</script>

</body>

</html>

注释：使用单行注释最为常见。

提示：注释块常用于官方声明。

使用注释来阻止执行

使用注释来防止代码执行很适合代码测试。

在代码行之前添加 // 会把可执行的代码行更改为注释。

本例使用 // 来阻止代码行的执行：

实例

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 注释</h2>

<h1 id="myH"></h1>

<p id="myP"></p>

<script>

//document.getElementById("myH").innerHTML = "我的首页";

document.getElementById("myP").innerHTML = "我的第一个段落。";

</script>

<p>// 开头的代码行不会被执行。</p>

</body>

</html>

本例使用注释块来阻止多行代码的执行：

<!DOCTYPE html>

<html>

<body>

<h2>JavaScript 注释</h2>

<h1 id="myH"></h1>

<p id="myP"></p>

<script>

/\*

document.getElementById("myH").innerHTML = "欢迎访问我的首页";

document.getElementById("myP").innerHTML = "这是我的第一个段落。";

\*/

document.getElementById("myP").innerHTML = "注释块不会被执行。";

</script>

</body>

</html>

# 第8章 JavaScript 变量

JavaScript 变量

JavaScript 变量是存储数据值的容器。

在本例中，x、y 和 z 是变量：

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript 变量</h1>

<p>在本例中，x、y 以及 z 都是变量。</p>

<p id="demo"></p>

<script>

var x = 7;

var y = 8;

var z = x + y;

document.getElementById("demo").innerHTML =

"z 的值是：" + z;

</script>

</body>

</html>

从上例中，您可获得：

x 存储值 7

y 存储值 8

z 存储值 15

类似代数

在本例中，price1、price2 以及 total 是变量：

<!DOCTYPE html>

<html>

<body>

<h1>JavaScript 变量</h1>

<p id="demo"></p>

<script>

var price1 = 7;

var price2 = 8;

var price3 = 12;

var total = price1 + price2 + price3;

document.getElementById("demo").innerHTML = "总计：" + total;

</script>

</body>

</html>