

处理文本——JavaScript 中的字符串

接下来，我们将注意力转向字符串——这就是编程中对文本片段的称呼。在本文中，我们将介绍在学习 JavaScript 时您真正应该了解的关于字符串的所有常见知识，例如创建字符串、转义字符串中的引号以及将字符串连接在一起。

先决条件：	基本的计算机知识，对 HTML 和 CSS 的基本理解，对 JavaScript 是什么的理解。
客观的：	熟悉 JavaScript 中字符串的基础知识。

话语的力量

语言对人类非常重要——它们是我们交流方式的重要组成部分。由于 Web 是一种主要基于文本的媒体，旨在让人们交流和共享信息，因此控制出现在其上的文字对我们很有用。[HTML](#)为我们的文本提供结构和意义，[CSS](#)允许我们精确地设计它的样式，而 JavaScript 包含许多用于操作字符串、创建自定义欢迎消息和提示、在需要时显示正确的文本标签、将术语按所需顺序排序的功能，以及更多。

到目前为止，我们在课程中向您展示的几乎所有程序都涉及一些字符串操作。

字符串——基础

乍一看，字符串的处理方式与数字类似，但当您深入挖掘时，您会开始看到一些显着的差异。[让我们首先在浏览器开发人员控制台](#)中输入一些基本行来熟悉一下。

创建字符串

1. 首先，输入以下行：

```
const string = "The revolution will not be televised.";
console.log(string);
```

就像我们对数字所做的那样，我们声明一个变量，用一个字符串值初始化它，然后返回该值。这里唯一的区别是，在编写字符串时，需要用引号将值括起来。

2. 如果您不这样做，或者遗漏其中一个引号，您将收到错误消息。尝试输入以下行：

```
const badString1 = This is a test;
const badString2 = 'This is a test;
const badString3 = This is a test';
```

这些行不起作用，因为任何没有引号的文本都被假定为变量名称、属性名称、保留字或类似名称。如果浏览器找不到它，则会引发错误（例如“缺少；前语句”）。如果浏览器可以看到字符串的开始位置，但找不到字符串的结尾（如第二个引号所示），它会报错（带有“未终止的字符串文字”）。如果您的程序出现此类错误，请返回并检查所有字符串以确保没有遗漏引号。

3. 如果您之前定义了变量，则以下内容将起作用 string ——现在试试看：

```
const badString = string;
console.log(badString);
```

badString 现在设置为与 具有相同的值 string 。

单引号与双引号

1. 在 JavaScript 中，您可以选择单引号或双引号来包裹您的字符串。以下两种方式都可以正常工作：

```
const sgl = 'Single quotes.';
const dbl = "Double quotes";
```

```
console.log(sgl);  
console.log(dbl);
```

2. 两者之间的区别很小，您使用哪种取决于个人喜好。但是，您应该选择一个并坚持下去；不同引用的代码可能会造成混淆，尤其是当您在同一个字符串上使用两个不同的引号时！以下将返回错误：

```
const badQuotes = 'What on earth?';
```

3. 浏览器会认为该字符串尚未关闭，因为您未用于包含您的字符串的其他类型的引号可能会出现在该字符串中。例如，这两个都可以：

```
const sglDbl = 'Would you eat a "fish supper"?';  
const dblSgl = "I'm feeling blue.";  
console.log(sglDbl);  
console.log(dblSgl);
```

4. 但是，如果字符串用于包含它们，则不能在字符串中包含相同的引号。以下将出错，因为它会使浏览器混淆字符串的结束位置：

```
const bigmouth = 'I've got no right to take my  
place...';
```

这很好地引导我们进入下一个主题。

转义字符串中的字符

要修复我们之前的问题代码行，我们需要转义问题引号。转义字符意味着我们对它们做一些事情以确保它们被识别为文本，而不是代码的一部分。在 JavaScript 中，我们通过在字符前放置一个反斜杠来实现这一点。尝试这个：

```
const bigmouth = 'I\'ve got no right to take my place...';  
console.log(bigmouth);
```

这很好用。其他字符也可以用同样的方式进行转义，例如 `\"`，另外还有一些特殊的代码。有关详细信息，请参阅[转义序列](#)。

连接字符串

连接只是意味着“连接在一起”。要在 JavaScript 中连接字符串，您可以使用不同类型的字符串，称为 *模板文字*。

模板文字看起来就像一个普通的字符串，但不是使用单引号或双引号（' 或 "），而是使用反引号（`）：

```
const greeting = `Hello`;
```

这可以像普通字符串一样工作，除了你可以在其中包含变量，包裹在 \${ } 字符中，并且变量的值将被插入到结果中：

```
const name = "Chris";
const greeting = `Hello, ${name}`;
console.log(greeting); // "Hello, Chris"
```

您可以使用相同的技术将两个变量连接在一起：

```
const one = "Hello, ";
const two = "how are you?";
const joined = `${one}${two}`;
console.log(joined); // "Hello, how are you?"
```

上下文中的连接


让我们看一下实际使用的串联：

```
<button>Press me</button>

const button = document.querySelector("button");

function greet() {
  const name = prompt("What is your name?");
  alert(`Hello ${name}, nice to see you!`);
}
```

```
button.addEventListener("click", greet);
```



在这里，我们使用了该 [window.prompt\(\)](#) 函数，它要求用户通过弹出对话框回答问题，然后将他们输入的文本存储在给定变量中——在本例中为 `name`。然后，我们使用该 [window.alert\(\)](#) 函数显示另一个弹出窗口，其中包含一个将名称插入通用问候消息的字符串。

使用“+”连接

您还可以使用 `+` 运算符连接字符串：

```
const greeting = "Hello";
const name = "Chris";
console.log(greeting + ", " + name); // "Hello, Chris"
```

然而，模板字面量通常会给你更易读的代码：

```
const greeting = "Hello";
const name = "Chris";
console.log(`${greeting}, ${name}`); // "Hello, Chris"
```

数字与字符串

那么当我们尝试组合字符串和数字时会发生什么？让我们在我们的控制台中尝试一下：

```
const name = "Front ";
const number = 242;
console.log(`${name}${number}`); // "Front 242"
```

您可能希望这会返回一个错误，但它工作得很好。尝试将字符串表示为数字并没有多大意义，但将数字表示为字符串确实有意义，因此浏览器会将数字转换为字符串并将这两个字符串连接起来。

如果您有一个数字变量要转换为字符串但不作其他更改，或者有一个字符串变量要转换为数字但不作其他更改，则可以使用以下两种构造：

- 如果可以，该 [Number\(\)](#) 函数会将传递给它的任何内容转换为数字。尝试以下操作：

```
const myString = "123";
const myNum = Number(myString);
console.log(typeof myNum);
```

- 相反，每个数字都有一个称为 [toString\(\)](#) 将其转换为等效字符串的方法。尝试这个：

```
const myNum2 = 123;
const myString2 = myNum2.toString();
console.log(typeof myString2);
```

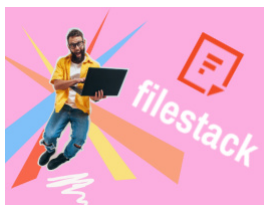
这些结构在某些情况下非常有用。例如，如果用户在表单的文本字段中输入一个数字，它就是一个字符串。但是，如果你想将这个数字添加到某物中，你需要它是一个数字，这样你就可以传递它 `Number()` 来处理这个问题。[我们在第 59 行的猜数字游戏](#) 中正是这样做的。

在字符串中包含表达式

您可以在模板文字中包含 JavaScript 表达式，以及简单的变量，结果将包含在结果中：

```
const song = "Fight the Youth";
const score = 9;
const highestScore = 10;
const output = `I like the song ${song}. I gave it a score of ${
  (score / highestScore) * 100
}%.`;
console.log(output); // "I like the song Fight the Youth. I gave
it a score of 90%."
```

多行字符串



需要免费的专业文件
上传器吗？立即试用
Filestack。

 mdn web docs

不想看广告？

模板文字尊重源代码中的换行符，因此您可以像这样编写跨越多行的字符串：

```
const output = `I like the song.  
I gave it a score of 90%.`;  
console.log(output);
```

```
I like the song.  
I gave it a score of 90%.  
*/
```

\n 要使用普通字符串获得等效输出，您必须在字符串中包含换行符 ()：

```
const output = "I like the song.\nI gave it a score of 90%.";  
console.log(output);  
  
/*  
I like the song.  
I gave it a score of 90%.  
*/
```

有关高级功能的更多示例和详细信息，请参阅我们的[模板文字](#)参考页面。

结论

这就是 JavaScript 中涵盖的字符串的基础知识。在下一篇文章中，我们将以此为基础，研究 JavaScript 中字符串可用的一些内置方法，以及我们如何使用它们将字符串操纵成我们想要的形式。

此页面最后修改于 2023 年 2 月 26 日由[MDN 贡献者](#)提供。