存储你需要的信息——变量

阅读最后几篇文章后，您现在应该知道 JavaScript 是什么、它能为您做什么、如何将它与其他 Web 技术一起使用，以及它的主要功能从高层次看是什么样的。在本文中，我们将深入了解真正的基础知识，看看如何使用 JavaScript 最基本的构建块——变量。

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 变量的基础知识。

你需要的工具

在整篇文章中，您将被要求输入代码行来测试您对内容的理解。如果您使用的是桌面浏览器，则键入示例代码的最佳位置是浏览器的 JavaScript 控制台（有关如何访问此工具的更多信息，请参阅[什么是浏览器开发人员工具）。]

什么是变量?

变量是值的容器，例如我们可能在求和中使用的数字，或者我们可能用作句子一部分的字符串。

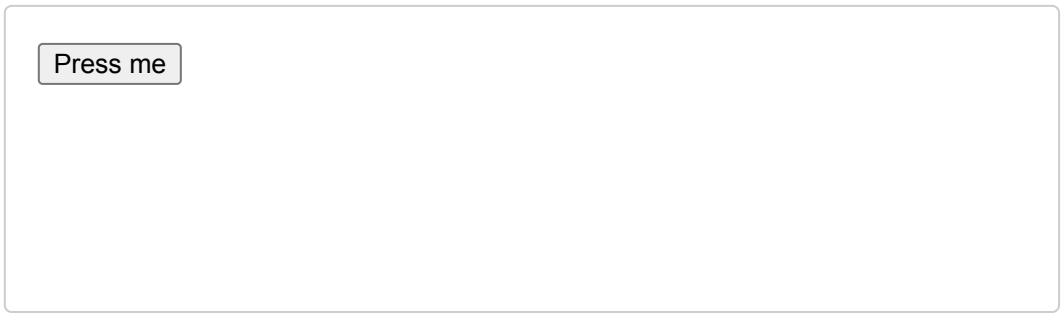
变量示例

让我们看一个简单的例子：

```
<button id="button_A">Press me</button>
<h3 id="heading_A"></h3>
```

```
const buttonA = document.querySelector('#button_A');
const headingA = document.querySelector('#heading_A');

buttonA.onclick = () => {
  const name = prompt('What is your name?');
  alert(`Hello ${name}, nice to see you!`);
  headingA.textContent = `Welcome ${name}`;
}
```



在这个例子中按下按钮运行一些代码。第一行在屏幕上弹出一个框，要求读者输入他们的名字，然后将值存储在一个变量中。第二行显示一条欢迎消息，其中包括他们的名字，取自变量值，第三行在页面上显示该名字。

没有变量

要理解为什么它如此有用，让我们考虑一下如何在不使用变量的情况下编写此示例。它最终看起来像这样：

```
<button id="button_B">Press me</button>
<h3 id="heading_B"></h3>



const buttonB = document.querySelector('#button_B');
const headingB = document.querySelector('#heading_B');

buttonB.onclick = () => {
    alert(`Hello ${prompt('What is your name?')}, nice to see
you!`);
    headingB.textContent = `Welcome ${prompt('What is your
name?')}`;
}
```

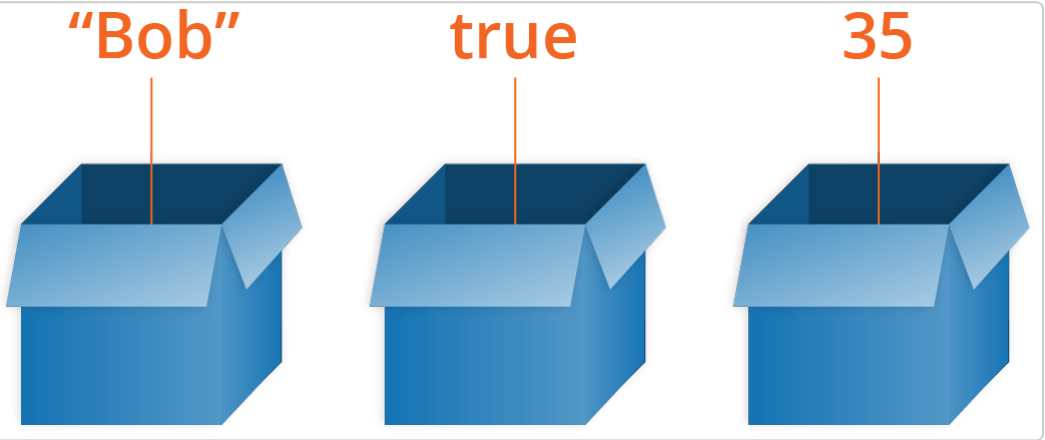
Press me

您可能不完全理解我们正在使用的语法（还！），但您应该能够理解这个想法。如果我们没有可用的变量，我们每次需要使用它时都必须询问读者的名字！

变量只是有意义的，随着你对 JavaScript 的了解越来越多，它们将开始成为你的第二天性。

变量的一个特别之处在于它们几乎可以包含任何东西——而不仅仅是字符串和数字。变量还可以包含复杂的数据甚至整个函数来完成令人惊奇的事情。随着您的进行，您将了解更多相关信息。

> **注意：**我们说变量包含值。这是一个重要的区别。变量本身不是值；它们是价值的容器。您可以将它们想象成可以存放东西的小纸板箱。



声明一个变量

要使用一个变量，您首先必须创建它——更准确地说，我们称之为声明变量。为此，我们键入关键字 `let` ，然后键入要调用变量的名称：

```
let myName;
let myAge;
```

在这里，我们创建了两个名为 `myName` 和 的变量 `myAge` 。尝试在网络浏览器的控制台中输入这些行。之后，尝试使用您自己选择的名称创建一个（或两个）变量。

> **注意：**在 JavaScript 中，所有代码指令都应以分号（ `;` ）结尾——您的代码可能对单行代码可以正常工作，但当您同时编写多行代码时可能不会。尝试养成包含它的习惯。

您可以通过仅键入变量的名称来测试这些值现在是否存在于执行环境中，例如

```
myName;
myAge;
```

它们目前没有价值；它们是空容器。当您输入变量名称时，您应该得到一个返回值 `undefined` 。如果它们不存在，您将收到一条错误消息——尝试输入

```
scoobyDoo;
```

> **注意：**不要将存在但没有定义值的变量与根本不存在的变量混淆——它们是完全不同的东西。在您上面看到的盒子类比中，不存在意味着没有盒子（变量）可以让值进入。没有定义值意味着有一个盒子，但它里面没有值。

初始化一个变量

一旦你声明了一个变量，你就可以用一个值来初始化它。为此，您可以键入变量名称，后跟等号（ `=` ），然后是您要赋予它的值。例如：

```
myName = 'Chris';
myAge = 37;
```

现在尝试返回控制台并输入这些行。在每种情况下，您都应该看到您已分配给控制台返回的变量的值以确认它。同样，您可以通过在控制台中键入变量名称来返回变量值——再试一次：

```
myName;
myAge;
```

您可以同时声明和初始化一个变量，如下所示：

```
let myDog = 'Rover';
```

这可能是您大多数时候会做的事情，因为它比在两条单独的行上执行这两个操作更快。

关于 var 的注释

您可能还会看到使用关键字声明变量的不同方式 `var`：

```
var myName;
var myAge;
```

回到最初创建 JavaScript 时，这是声明变量的唯一方法。的设计 `var` 令人困惑且容易出错。So `let` 是在现代版本的 JavaScript 中创建的，这是一个用于创建变量的新关键字，其工作方式与 有所不同 `var` ，修复了该过程中的问题。

下面解释了几个简单的区别。我们现在不会深入探讨所有差异，但是随着您对 JavaScript 的了解越来越多，您会开始发现它们（如果您现在真的想了解它们，请随时查看我们的 let 参考页面）。

首先，如果您编写声明和初始化变量的多行 JavaScript 程序，您实际上可以 `var` 在初始化变量后声明一个变量，它仍然有效。例如：

```
myName = 'Chris';

function logName() {
  console.log(myName);
}

logName();

var myName;
```

> **注意：** 当在 JavaScript 控制台中输入单独的行时，这将不起作用，仅当在 Web 文档中运行多行 JavaScript 时。

这是因为**提升**- 阅读var 提升以获取有关该主题的更多详细信息。

提升不再适用于 `let` . 如果我们在上面的示例中更改 `var` 为 `let` ， 它将因错误而失败。这是一件好事——在初始化变量后声明它会导致代码混乱、难以理解。

其次，当您使用 时 `var` ， 您可以根据需要多次声明同一变量，但 `let` 不能。以下将起作用：

```
var myName = 'Chris';
var myName = 'Bob';
```

但以下内容会在第二行引发错误：

```
  let myName = 'Chris';
  let myName = 'Bob';
```

你必须这样做：

```
let myName = 'Chris';
myName = 'Bob';
```

同样，这是一个明智的语言决定。没有理由重新声明变量——它只会让事情变得更混乱。

出于这些原因以及更多原因，我们建议您 `let` 在代码中使用，而不是 `var` 。除非您明确编写对古代浏览器的支持，否则不再有任何理由使用， `var` 因为自 2015 年以来所有现代浏览器都支持 `let` 。

> **注意：** 如果您在浏览器的控制台中尝试此代码，请将每个代码块作为一个整体复制并粘贴到此处。Chrome 的控制台中 有一项功能，允许使用 `let` 和重新声明变量： `const`
>
> ```
> > let myName = 'Chris';
>   let myName = 'Bob';
> // As one input: SyntaxError: Identifier 'myName' has
> already been declared
>
> > let myName = 'Chris';
> > let myName = 'Bob';
> // As two inputs: both succeed
> ```

更新变量

使用值初始化变量后，您可以通过给它一个不同的值来更改（或更新）该值。尝试在您的控制台中输入以下行：

```
myName = 'Bob';
myAge = 40;
```

关于变量命名规则的旁白

您几乎可以随心所欲地调用变量，但是有一些限制。通常，您应该坚持只使用拉丁字符（0-9、az、AZ）和下划线字符。

- 您不应使用其他字符，因为它们可能会导致错误或难以被国际观众理解。

- 不要在变量名的开头使用下划线——这在某些 JavaScript 结构中用于表示特定的事物，因此可能会造成混淆。
- 不要在变量的开头使用数字。这是不允许的，会导致错误。
- 一个安全的约定是所谓的“小驼峰式” ，你把多个单词放在一起，第一个单词全部小写，后面的单词大写。到目前为止，我们一直在文章中使用它作为我们的变量名。
- 使变量名称直观，以便它们描述它们包含的数据。不要只使用单个字母/数字或大长短语。
- 变量区分大小写—— `myage` 与 `myAge` .
- 最后一点：您还需要避免使用 JavaScript 保留字作为变量名——这里指的是构成 JavaScript 实际语法的字！因此，您不能使用 `var` 、 `function` 、 `let` 和 之类的词 `for` 作为变量名。浏览器将它们识别为不同的代码项，因此您会收到错误。

> **注意：您可以在**Lexical grammar — keywords中找到一个相当完整的要避免的保留关键字列表。

好名字例子：

```
age
myAge
init
initialColor
finalOutputValue
audio1
audio2
```

坏名字示例：

```
1
a
_12
myage
MYAGE
var
```

```
Document
skjfndskjfnbdskjfb
thisisareallylongvariablenameman
```

牢记上述指导，现在尝试创建更多变量。

变量类型

我们可以将几种不同类型的数据存储在变量中。在本节中，我们将简要介绍这些，然后在以后的文章中，您将更详细地了解它们。

到目前为止，我们已经看过前两个，但还有其他的。

数字

您可以将数字存储在变量中，可以是整数，如 30（也称为整数）或小数，如 2.456（也称为浮点数或浮点数）。与其他一些编程语言不同，您不需要在 JavaScript 中声明变量类型。当你给一个变量一个数字值时，你不包括引号：

```
let myAge = 17;
```

字符串

字符串是文本片段。当你给一个变量一个字符串值时，你需要用单引号或双引号把它包起来；否则，JavaScript 会尝试将其解释为另一个变量名。

```
let dolphinGoodbye = 'So long and thanks for all the fish';
```

布尔值

布尔值是真/假值——它们可以有两个值， `true` 或者 `false` . 这些通常用于测试条件，然后根据需要运行代码。因此，例如，一个简单的案例是：

```
let iAmAlive = true;
```

而实际上它会更像这样使用：

```
let test = 6 < 3;
```

这是使用“小于”运算符（ < ）来测试 6 是否小于 3。如您所料，它返回 `false`，因为 6 不小于 3！在本课程的后面，您将学到更多关于此类运算符的知识。

数组

数组是单个对象，它包含用方括号括起来并用逗号分隔的多个值。尝试在您的控制台中输入以下行：

```
let myNameArray = ['Chris', 'Bob', 'Jim'];
let myNumberArray = [10, 15, 40];
```

一旦定义了这些数组，您就可以通过它们在数组中的位置访问每个值。试试这些行：

```
myNameArray[0]; // should return 'Chris'
myNumberArray[2]; // should return 40
```

方括号指定一个索引值，该索引值对应于您要返回的值的位置。你可能已经注意到 JavaScript 中的数组是从零开始索引的：第一个元素的索引为 0。

在以后的文章中，您将学到更多关于数组的知识。

对象

在编程中，对象是模拟真实对象的代码结构。你可以有一个简单的对象来代表一个盒子并包含关于它的宽度、长度和高度的信息，或者你可以有一个对象来代表一个人，并包含关于他们的名字、身高、体重、他们说什么语言、如何向他们问好等等。

尝试在您的控制台中输入以下行：

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

要检索存储在对象中的信息，可以使用以下语法：

```
dog.name
```

我们现在不会再关注对象——您可以在以后的模块中了解更多关于对象的信息。

动态类型

JavaScript 是一种“动态类型语言”，这意味着与其他一些语言不同，您不需要指定变量将包含什么数据类型（数字、字符串、数组等）。

例如，如果您声明一个变量并给它一个用引号引起来的值，浏览器会将变量视为一个字符串：

```
let myString = 'Hello';
```

即使引号中的值只是数字，它仍然是一个字符串——而不是数字——所以要小心：

```
let myNumber = '500'; // oops, this is still a string
typeof myNumber;
myNumber = 500; // much better — now this is a number
typeof myNumber;
```

试着将上面的四行一行一行地输入到你的控制台，看看结果是什么。你会注意到我们使用了一个特殊的运算符 `typeof` ——它返回你在它后面键入的变量的数据类型。第一次调用时，它应该返回 `string` ，因为此时 `myNumber` 变量包含一个字符串 `'500'` 。看看你第二次调用它时返回了什么。

JavaScript 中的常量

除了变量，您还可以声明常量。这些类似于变量，除了：

- 你必须在声明它们时初始化它们
- 初始化后不能再给它们赋新值。

例如，使用 `let` 您可以在不初始化变量的情况下声明它：

```
let count;
```

如果您尝试这样做， `const` 您将看到一个错误：

```
  const count;
```

同样， `let` 你可以用 with 初始化一个变量，然后给它赋一个新值（这也称为*重新赋值*变量）：

```
let count = 1;
count = 2;
```

如果您尝试这样做， `const` 您将看到一个错误：

请注意，尽管 JavaScript 中的常量必须始终命名相同的值，但您可以更改它命名的值的内容。对于像数字或布尔值这样的简单类型，这不是一个有用的区别，但考虑一个对象：

```
const bird = { species : 'Kestrel'};
console.log(bird.species);  // "Kestrel"
```

您可以更新、添加或删除使用 声明的对象的属性 `const` ，因为即使对象的内容已更改，常量仍指向同一个对象：

```
bird.species = 'Striated Caracara';
console.log(bird.species);  // "Striated Caracara"
```

什么时候用const什么时候用let

如果你不能用 with 做尽可能多的事情 `const`， `let` 为什么你更愿意使用它而不是用 `let`？事实上 `const` 是非常有用的。如果你使用 `const` 命名一个值，它会告诉任何查看你的代码的人，这个名称永远不会被分配给一个不同的值。任何时候他们看到这个名字，就会知道它指的是什么。

在本课程中，我们采用以下关于何时使用 `let` 和何时使用的原则 `const`：

`const` 能用就用，`let` 该用就用。

这意味着如果您可以在声明变量时对其进行初始化，并且以后不需要重新分配它，请将其设为常量。

测试你的技能！

您已读完本文，但您还记得最重要的信息吗？在继续之前，您可以找到一些进一步的测试来验证您是否保留了这些信息——请参阅测试您的技能：变量。

概括

到目前为止，您应该对 JavaScript 变量以及如何创建它们有一定的了解。在下一篇文章中，我们将更详细地关注数字，看看如何在 JavaScript 中进行基本数学运算。

此页面最后修改于 2023 年 2 月 26 日由MDN 贡献者提供。