JavaScript Variables

(https://github.com/learn-co-curriculum/phase-0-pac-1-js-variables) (https://github.com/learn-co-curriculum/phase-0-pac-1-js-variables/issues/new)

Learning Goals

- Define what a variable is.
- Variable names in JavaScript.
- Initializing variables in JavaScript.
- Retrieving and changing the value of variables.
- Identify when to use `const`, `let`, and `var` for declaring variables.

Introduction

In an earlier lesson, we talked about using the *assignment expression* to save information into a variable. "Saving" to a variable allows us to *store* a result so we can use it again later. Storing calculations to *temporary storage places* is the heart of making efficient programs. It's a simple idea that has powerful consequences.

What is a Variable?

A variable is a container in which we can store values for later retrieval.

Imagine a box that can hold any type of data: a number, a string, etc. We take some data that we want to store, place it inside the box, and hand the box off to the JavaScript engine, which stores it in memory. All done! Our data is safely cached until we need to access it again.

But wait! When we ask for the data back, how will the JavaScript engine know *which* box to retrieve? We need to assign a name to our variable — a label for our box — so that we can use the *variable lookup expression* to tell the engine exactly which piece of stored data we want to access.

Variable Names in JavaScript

Variable names in JavaScript can sometimes be complicated, but if you follow these three rules you'll be fine:

- Start every variable name with a lowercase letter. Variable names starting with a number are not valid.
- Don't use spaces. If a variable name consists of multiple words, `camelCaseYourVariableNames` (see the camel humps?) instead of `snake_casing_them` (think of the underscore as a snake that swallowed the words).
- Don't use JavaScript **reserved words (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Lexical_grammar#Reserved_keywords_as_of_ECMAScript_2015)** or **future reserved words (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Lexical_grammar#Future_reserved_keywords)** .

It's important to note that case matters, so `javaScript` , `javascript` , `JavaScript` , and `JAVASCRIPT` are four different variables.

Initializing Variables in JavaScript

Initializing variables in JavaScript is really a two-step process: first, we *declare* the variable, then we *assign a value* to it.

To declare the variable, we use either the `let` or `const` reserved word. (And, in fact, there's a third option as well: `var`. We'll talk about when to use each a bit later in this lesson. For now, just know that when you see `const`, `let`, or `var`, those are reserved words that are used to declare a variable.)

```
let pi;
//=> undefined
```

The JavaScript engine sets aside a chunk of memory to store the declared variable. Then, we assign a value to that variable using the *assignment expression*:

```
pi = 3.14159;
//=> 3.14159
```

We can make our code a bit more efficient by packaging both initialization steps — declaration and assignment — in a single line of code:

```
let pi = 3.14159;
//=> undefined
```

You will encounter cases later on where it makes sense to declare a variable without immediately assigning a value to it, but combining the two steps will work most of the time.

Retrieving and Changing the Value of Variables

Say we've declared a variable `pi` and set its value:

```
let pi = 3.14159;
```

To retrieve the value of `pi`, we use the *variable lookup expression*, i.e., we simply type in its name:

```
pi;
//=> 3.14159
```

If we want to change (reassign) that value, we use the *assignment expression*:

```
pi;
//=> 3.14159
pi = 3.14;
pi;
//=> 3.14;
```

Identify When to Use `const`, `let`, and `var` for Declaring Variables

In the past, `var` was the only option available in JavaScript for declaring variables. Then, in 2015, the language underwent a major revision that, among many other changes, added two new options: `const` and `let`.

> **NOTE:** Delving into the history of JavaScript and the many changes and improvements that were made to it in 2015 is outside the scope of this lesson. However, as you're learning to code in JavaScript you may see references to `ECMAScript 2015`, `ES2015`, or `ES6`. These terms are interchangeable and refer to that major revision. (ECMAScript is the "official" name of JavaScript.) There are some resources listed at the end of this lesson if you're interested in learning more about `ES2015` and the history of JavaScript.

The addition of `const` and `let` was in response to significant problems that the use of `var` can cause. In particular, it can create scope issues (which is a whole other topic that you'll learn about later) that lead to unpredictable and difficult to diagnose bugs in your code. You will likely see `var` used a lot in legacy code and older StackOverflow posts, so it's important to be familiar with it. ***However, there is no good reason to use `var` to declare variables in your own code, and good reasons not to***.

`let`

The main advantage of using `let` for declaring a variable is that, unlike `var`, it will throw an error if you try to declare the same variable a second time:

```
let pi = 3.14159;
//=> undefined

let pi = "the ratio between a circle's circumference and diameter";
//=> Uncaught SyntaxError: Identifier 'pi' has already been declared
```

Why is this a good thing? Well, you can imagine how easy it could be, especially in a lengthy program, to accidentally reuse a variable name. Doing so can cause unpredictable behavior because the value associated with the variable in one part of the program may be changed by code in a different part of the program. These types of problems can be very difficult to track down and debug. Using `let` to declare your variables will reduce the likelihood of introducing such errors into your code.

While we can't *redeclare* a variable that is declared using `let`, we can still *reassign* its value:

```
let pi = 3.14159;
//=> undefined

pi = "the ratio between a circle's circumference and diameter";
//=> "the ratio between a circle's circumference and diameter"

pi;
//=> "the ratio between a circle's circumference and diameter"
```

const

The `const` reserved word should be your go-to option for declaring variables in JavaScript. When you declare a variable with `const`, not only can it not be redeclared but it also ***cannot be reassigned***.

```
const pi = 3.14159;
//=> undefined
```

```
pi = 2.71828;
//=> Uncaught TypeError: Assignment to constant variable.
```

When you (or another developer) sees that `pi` has been declared with `const`, you immediately know that the variable points to the same value every other time it's referenced in the program. For variables declared with `let` (or `var`), you cannot be so sure and will have to keep track of how those variables change throughout the program. The extra information provided by `const` is valuable, and it comes at no extra cost to you! Just use `const` whenever possible and reap the benefits.

> **Note**: With `let`, it's possible to declare a variable without assigning a value:
>
> ```
> let pi;
> //=> undefined
>
> pi = 3.14159;
> //=> 3.14159
> ```
>
> However, because `const` doesn't allow reassignment after the variable is initialized, we **must** assign a value right away:
>
> ```
> const pi;
> //=> Uncaught SyntaxError: Missing initializer in const declaration
>
> const pi = 3.14159;
> //=> undefined
> ```

As your JavaScript powers increase with experience, you'll develop a more nuanced understanding of what to use where. However, for now, this is a good rule of thumb:

- ***Use `var` ...*** never.
- ***Use `let` ...*** when you know the value of a variable will change. For example, a `counter` variable that starts at `0` and is subsequently incremented to `1`, `2`, `3`, and so on. In the lessons on looping and iteration in JavaScript, `let` will have its moment in the spotlight.
- ***Use `const` ...*** for *every* other variable.

Best practice is to always declare variables with `const` and then, if you later realize that the value has to change over the course of your program, circle back to change it to `let`.

Conclusion

We covered what a variable is, how to initialize and retrieve it, and how to assign or reassign its value. We also looked at best practices for naming variables and for when to use `let`, `const`, and `var`.

Resources

- MDN — Language basics crash course: Variables (https://developer.mozilla.org/en-US/docs/Learn/Getting_started_with_the_web/JavaScript_basics#Variables)
- MDN — `let` (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/let)
- MDN — `const` (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/const)
- MDN — `var` (https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/var)
- JavaScript ES6+: `var`, `let`, or `const`? (https://medium.com/javascript-scene/javascript-es6-var-let-or-const-ba58b8dcde75)
- W3C - A Short History of JavaScript (https://en.wikipedia.org/wiki/JavaScript#History)
- Wikipedia - ECMAScript: Versions (https://en.wikipedia.org/wiki/ECMAScript#Versions)