Variables and Operators

Foundations Course

Introduction

In the previous sections you learned how to structure webpages with HTML and style them with CSS. The next step is to make the webpage *interactive*, which is exactly what JavaScript is for.

In this section, we will focus on the fundamentals of JavaScript and how you can use it to manipulate all the various interactions between the web page and user.

Lesson overview

This section contains a general overview of topics that you will learn in this lesson.

- Running JavaScript code using an HTML file.
- Declaring variables with `let` and `const`.
- Performing number operations.
- Performing string operations.
- Using logical and mathematical operators.

How to run JavaScript code

All JavaScript we will be writing in the majority of the Foundations course will be run via the browser. Later lessons in Foundations and the NodeJS path will show you how to run JavaScript outside of the browser environment. This means that we will let the browser execute our code, even if it comes from a file we created.

Outside of these lessons, for now you should always default to running your JavaScript in the browser unless otherwise specified, otherwise you may run into unexpected errors.

The simplest way to get started is to create an HTML file with the JavaScript code inside of it. Use the VS Code snippet `!` (`tab`) to create the basic HTML skeleton in a file on your computer somewhere. Be sure to include the `.script` tag:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>

  <script>
    // Your JavaScript goes here!
    console.log("Hello, World!")
  </script>

</body>
</html>
```

Save and open this file up in a web browser and then open up the browser's console:

1. Right-click on the blank webpage.
2. Click on "Inspect" or "Inspect Element" to open the Developer Tools.
3. Find and select the "Console" tab, where you should see the output of our `console.log` statement.

> **Live preview**
>
> You can use Live Preview extension in Visual Studio Code to automatically update the browser when you save your file instead of having to manually refresh the page to see any changes when you edit your code. Try editing the text to say something different!

`console.log()` is the command to print something to the developer console in your browser. You can use this to print the results from at least of the following articles and exercises to the console. We encourage you to code along with all of the examples in this and future lessons.

Another way to include JavaScript in a webpage is through an external script. This is very similar to linking external CSS docs to your website.

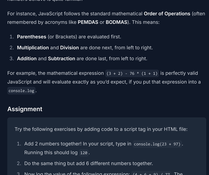
```
<script src="javascript.js"></script>
```

JavaScript files have the extension `.js` similar to `.css` for stylesheets. External JavaScript files are used for more complex scripts.

We named our file `javascript.js` but we could have chosen any name like `my-script.js` or even no name `.js`. What is really important is the `.js` extension.

Variables

These are the building blocks of any program. You can think of variables as "storage containers" for data in your code.



You can declare variables using the `let` keyword. Let's try it! (No pun intended).

```
let firstName = "John";
let lastName = "Doe";

console.log(firstName);
console.log(lastName);
```

What will the `console.log` output? Try it out!

You can also re-assign variables:

```
let age = 31;
console.log(age); // outputs 31 to the console

age = 54;

console.log(age); // what will be output now?
```

Notice the lack of `let` on line 4 - we don't need it since the variable has already been declared earlier and we are just re-assigning it here!

Re-assigning is cool and all, but what if we don't want it to happen? For example we might have a constant `pi` which will never need to be re-assigned. We can accomplish this using the `const` keyword.

```
const pi = 3.14;
pi = 10;

console.log(pi); // What will be output?
```

Your intuition may tell you that `3.14` will be output. Try it!

An error is thrown. It doesn't even reach the `console.log`! You may wonder why we would want an error in our code. Truth be told, errors are incredibly helpful at telling us what is wrong with our code and exactly where the issue is. Without them, our code would still not do what we may want it to, but it would be a major pain to try and find what's wrong!

So in summary, there are two ways to declare a variable:

- `let`, which we can re-assign,
- `const` which we **can't** re-assign and will throw an error if we try.

There is also a third way, `var`, which was the original way variables were declared in JavaScript. `var` is similar to `let` in that variables assigned this way can be reassigned, but it has other quirks that were cleaned up when the language introduced `let` and `const`. By and large, it is not used anymore. However, you will likely come across code which uses `var` at some point, so it is useful to know that it exists.

Numbers

Numbers are the building blocks of programming logic! In fact, it's hard to think of any useful programming task that doesn't involve at least a little basic math, so knowing how numbers work is obviously quite important. Luckily, in JavaScript, the way numbers behave is quite familiar.

For instance, JavaScript follows the standard mathematical **Order of Operations** (often remembered by acronyms like **PEMDAS** or **BODMAS**). This means:

1. **Parentheses** (or Brackets) are evaluated first.
2. **Multiplication** and **Division** are done next, from left to right.
3. **Addition** and **Subtraction** are done last, from left to right.

For example, the mathematical expression `(3 + 2) - 70 * (1 + 1)` is perfectly valid JavaScript and will evaluate exactly as you'd expect, so try put that expression into a `console.log`.

Assignment

Try the following exercises by adding code to a script tag in your HTML file:

1. Add 2 numbers together! In your script, type in `console.log(23 + 97)`. Running this should log `120`.
2. Do the same thing but add 6 different numbers together.
3. Now log the value of the following expression: `(4 + 6 * 9) / 72`. The console should log approximately `0.8611`.
4. Let's use some variables!
   1. Add this statement to the script tag: `let a = 10`.
   2. Below it, add `console.log(a)`. When you run this, the browser console should log `10`.
   3. Afterwards, re-assign `a` with a different number value. Log `a` again afterwards and it should show the updated value (the previous log should still show the old value of `10` since that was before `a` was re-assigned).
   4. Now add to the bottom of the script `let b = 7 * a`.
   5. Log what `b` is. It should log the result of 7 multiplied by whatever you've assigned `a` with.
5. Try this sequence:
   1. Declare a const variable `max` with the value `57`.
   2. Declare another const variable `actual` and assign it `max - 13`.
   3. Declare another const variable `percentage` and assign it `actual / max`.
   4. Now if you log `percentage`, you should see a value in the console like `0.7719`.
6. Take a few minutes to keep playing around with various things in this script tag. Eventually, we will learn how to actually make these things show up on the webpage, but all of this logic will remain the same. Make sure you're comfortable with it before moving on.

Go through the following articles to deepen your knowledge.

1. Read up on variables in JavaScript from JavaScript.info.
2. This W3Schools lesson on JavaScript arithmetic followed by this on JavaScript numbers, are good introductions to what you can accomplish with numbers in JavaScript.
3. This MDN article on JavaScript math covers the same info from a slightly different point of view, while also teaching you how to apply some basic math in JavaScript. There's much more that you can do with numbers, but this is all you need at the moment.
4. Read through (and code along with) this article on JavaScript operators. Don't forget to do the "Tasks" at the bottom of the page! It will give you a pretty good idea of what you can accomplish with numbers (among other things!) in JavaScript.

Knowledge check

The following questions are an opportunity to reflect on key topics in this lesson. If you can't answer a question, click on it to review the material, but keep in mind you are not expected to memorize or master this knowledge.

- What three keywords can you use to declare new variables?
- Which of the three variable declarations should you avoid and why?
- What rules should you follow when naming variables?
- What happens when you add numbers and strings together?
- How does the Modulo (%), or Remainder, operator work?
- What's the difference between `==` and `===`?
- When would you receive a `NaN` result?
- How do you increment and decrement a number?
- What's the difference between prefixing and postfixing increment/decrement operators?
- What is operator precedence and how is it handled in JS?
- How do you access developer tools and the console?
- How do you log information to the console?
- What does unary plus operator do to string representations of integers? eg, `+"10"`

Additional resources

This section contains helpful links to related content. It isn't required, so consider it supplemental.

- MDN's "What is JavaScript?" explains a bit more about it on a high-level.