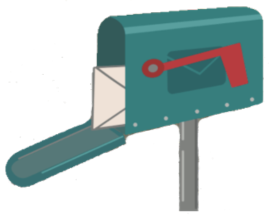
Variables and Constants

Variables and constants are named spaces in memory used to store data or values. Think of them like labeled mailboxes; just as a mailbox holds different pieces of mail according to its address, variables and constants hold data that our program can reference and use by their names.



Variables are used for storing data that may change. For instance, if you're coding a game, you might create a variable called level and initially set it to 1 for a new player. As the player progresses, you'll update the level variable to reflect their new status. This flexibility is key when working with data that will evolve over time.

The new value replaces the old value. Use the keyword ‘let’ to declare the variable, or in other words; initially set up the variable. Read the equal sign (=) as an assignment operator. For example, level is being assigned the value of 2.

```
let level = 1;

level = 2;
```

Constants, on the other hand, are meant to hold data that should remain unchanged. Use the keyword ‘const’ to declare the constant. For example, a user's name might be stored as a constant since the value won't need to change throughout the program. If we try to change the value that is assigned to the constant later in the code, we will get an error.

```
const name = 'Sue';
```

Let's consider a case with dates. While signupDate might remain the same, a value like lastLogin could be updated frequently to reflect each new login. In this case, signupDate would be a constant, while lastLogin would be a variable.

```
const signupDate = "2024-11-01";
let lastLogin = "2024-11-01";

// User logs in again
lastLogin = "2024-11-02";
```

Notice how these variables are named. It's easy to tell what will be stored in them. It's important to have meaningful names for your variables and constants. These names contain multiple words, so camelCase has been used where each word following the first word is capitalized. You can also use an underscore to separate words. Names cannot start with a number. Also, some reserved words like let or return can't be used as a variable name.

In JavaScript, variables and constants can store any type of data, including numbers, strings, and more.

```
42  is a number

“abc”  is a string

true is a Boolean

{a: 1} is an object

[1,2,3] is an array
```

Unlike some other programming languages, JavaScript doesn't require you to specify a data type when declaring a variable. This flexibility allows you to change the data type of a variable if needed, like switching from a string to a number, without any additional steps.

```
let myVariable = "advanced";

myVariable = 1;
```

You may come across the var keyword in older code for declaring variables. While var still works, using let (for variables) and const (for constants) is now the preferred approach. let and const improve code readability, scope management, and prevent accidental re-declarations, making your code more robust and easier to debug.

```
var age = 24;
```

```
let age = 24;
```

Let's look at scope, or in other words, the context in which variables and constants are accessible or visible. The way we have been declaring variables has been in global scope, meaning the variable is accessible from anywhere in your code. But if the variable or constant is declared within a block of code, like conditionals or loops, they can only be seen within that block of code. This is called block scope or local scope. The same thing happens if they are declared inside of a function; it restricts the variable to the function they are declared in. You can't reference the variable outside of the function or block of code where they were declared because they won't be recognized.

Understanding variable scope helps you write clearer and more predictable code. We will see scope in action later when we begin using functions and conditionals.

In this conditional, the variable firstname, which is assigned the value of 'Bob', can only be recognized within the if statement. If you try to reference it outside of the if statement, it won't be recognized.

```
if (true) {
  let firstname = 'Bob';
  console.log(firstname);
}

console.log(firstname); // Error: firstname is not defined
```

If we move the declaration of the variable to outside of the if conditional, then it is now global and can be recognized inside and outside of the if statement.

```
let firstname = 'Bob';

if (true) {
  console.log(firstname);
}

console.log(firstname); // Output: Bob
```

So, there we have an introduction to variable and constants, named spaces in memory used to store data or values that we use in our code.