JavaScript – theory and tasks

Introduction

JavaScript is the most widely used programming language on the web and is used on most websites.

By the end of this course, you'll have a strong foundation in JavaScript. You'll be able to write full-fledged programs, make HTML and CSS dynamic, and write impressive web applications.

This unit has two parts:

1. Learn the building blocks of JavaScript
2. Do some fun stuff with those parts

Because this course starts with the building blocks of JavaScript, make sure to take notes and pace yourself. This foundation will take us through the rest of the JavaScript course.

The *building* blocks of JavaScript make up every program and line of JavaScript.

Just like a language has nouns, verbs, and prepositions, JavaScript has its own building blocks. Instead of calling them building blocks, however, developers call them *data types*.

There are three essential data types to know for now, and here is your first test of memorization (don't worry, we will practice):

1. *String*: Any grouping of words or numbers surrounded by single quotes: *' ... '* or double quotes *" ... "*.
2. *Number*: Any number, including numbers with decimals, without quotes: *4, 8, 1516, 23.42*.
3. *Boolean*: This is always one of two words. Either *true* or *false*, with no quotations.

Let's review: a string is any grouping of words, a number's a number, and a boolean is a ghostly halloween costume. Or no, that's false! It's either *true* or *false*.

***Instructions***

*Write your code in the types.js file.*

1. In the code editor, there are three variables (we will learn about variables in the next lesson). For now, replace each variable's value with each corresponding type.

On line 1, there's a variable named myString equal to undefined. Replace undefined, with a string of your name.

1. On line 2, there's a variable named myNumber, also equal to undefined. Replace undefined with your lucky number.
2. On line 3, answer this question with a boolean:

Did you like the Halloween joke?

If so, replace the word undefined with the boolean true. Otherwise, replace undefined with false.

1. Notice that your string, number, and boolean printed to the black box on the right. That is the console, and is a program that can run JavaScript programs and show their results.

We will learn how to use the JavaScript with the console in the next exercise.

We can't do much programming with our knowledge of types right now, so let's build something cool. Let's learn how to ask JavaScript to talk to us.

To do this, we need two things:

1. A way to ask JavaScript to talk.
2. Something for JavaScript to say.

We can ask JavaScript to print words to the console with this line of code:

console.log('Your message here.');

In human-speak, this is saying: "Hey console, please print/log this thing inside the parentheses. Bye, thanks!"

By writing this line, we've also solved the second thing we need: Something for JavaScript to say. We can put a String, Number, or Boolean (or any data type) inside the parentheses of a *console.log* statement.

***Instructions***

*Write your code in the console.js file.*

1. Use console.log to log a string of your favorite pizza topping.

Stuck? Get a hint

1. Fun fact: We can log multiple things at once by separating them with commas, like this:

console.log('bacon', 'pesto');

Use console.log to print out two of your favorite pizza toppings (plural).

Don't worry, math does not need to be your strong-suit to learn JavaScript. There are just a few operations we'll need to know to make some awesome programs later in the course!).

JavaScript includes the general math operators that you can find on a calculator:

1. Add: +
2. Subtract: -
3. Multiply: \*
4. Divide: /

These all work how you might guess: *3 + 4* will equal *7, 50 / 10* will equal *5*.

Let's use each one of these math operators.

***Instructions***

*Write your code in the mathOprerators.js file.*

1. Inside of a console.log, add 3.5 to your age.

This is the age you'll be when we start sending people to live on Mars.

1. On a new line write another console.log. Inside the new console.log's parentheses, take the current year and subtract 1969.

The answer is how many years it's been since the 1969 moon landing.

1. Create another console.log, then divide 65 by 240.
2. Create one last console.log, then multiply the full answer from step 3 by 100.

That's the percent of the sun that is made up of helium. Assuming we could stand on the sun, we'd all sound like chipmunks!

Now let's generate a space fact while we learn a brand new operator called – drum roll please – the *modulus*.

The idea behind the modulus is to show you the remainder after you divide a number.

So, if you divide *13 / 5, 5* goes into *13* two times, and there will be *3* remaining. A modulus, denoted by a *%,* would take *13 % 5* and return the remainder *3*.

How on Earth is this useful?

Let's ask a question a modulus can solve: What will the moon phase be one year from today?

***Instructions***

*Write your code in the mathOprerators.js file.*

1. Let's say it's a full moon tonight, and we want to know what the moon will look like one year from today. We know from the moon phase image to the right that the moon circles the Earth every 27 days, so let's start by dividing 365 by 27.

Remember to put your calculations inside console.log to print them to the screen.

1. That gives us a number (13), followed by a decimal (.518...).

To figure out what phase the moon will be in a year, we need to know how many extra days the moon will rotate around the earth before the end of the year. We need the decimal we just found displayed as remaining days.

Use the modulus operator to find the remaining days, then run your code.

1. Now we know how many days into the moon's phase it will be in exactly one year.

We can now figure out, based on the moon phase image, what the moon phase will be 365 days from today. If it's a full moon today, a year from now it will be a new moon.

As it turns out, JavaScript has some tricks up its sleeve to make our lives easier.

JavaScript has built in functions, which help us do everyday things. We'll learn more about functions later in the course, so don't worry about understanding what they are right now.

Sometimes it's necessary to generate a random number within a program. We can do that with this code:

Math.random();

This code will return a random number between 0 and 1. JavaScript will generate a random number for us with this code.

To generate a random number between 0 and 50, we could multiply this result by 50, like so:

Math.random() \* 50;

The problem with this is that the answer will most likely be a decimal. Luckily, JavaScript has our back with another built in function called *Math.floor*. *Math.floor* will take a decimal number, and round down to the nearest whole number. It is used like this:

Math.floor(Math.random() \* 50);

In this case:

1. *Math.random* will generate a random number between 0 and 1.
2. We then multiplied that number by *50*, so now we have a number between 0 and 50.
3. Then, *Math.floor* will round the number down to the nearest whole number.

Let's utilize these two methods to generate a random number between 0 and 100.

***Instructions***

*Write your code in the random.js file.*

1. Inside of a console.log, create a random number with Math.random, then multiply it by 100.

If you run the program few times, you'll see random numbers in the console.

1. Now, utilize Math.floor to make the output a whole number.

Inside the console.log you wrote in the last step, put Math.random() \* 100 inside the parentheses of Math.floor.

1. Run the program a few times to see random numbers between 0 and 100.

Math.random() is a function that returns us a random value each time. Later in the course, we will use this to function to simulate a decision from the computer.

As we write JavaScript, we can create comments in our code.

Comments are lines that are not evaluated when the code runs. They exist just for human readers, in other words. Comments can be extremely useful when we're looking back at code we've written later on and for other people who will be looking at your code.

There are two types of code comments in JavaScript:

A *single line comment* will comment out a single line, and is denoted with two forward slashes // preceding a line of JavaScript code.

// The first 5 decimals of pi

console.log('Pi is equal to ' + 3.14159);

A *multi-line comment* will comment out multiple lines, and is denoted with /\* to begin the comment, and \*/ to end the comment.

/\*

console.log('All of this code');

console.log('Is commented out');

console.log('And will not be executed);

\*/

***Instructions***

*Write your code in the comment.js file.*

1. Let's practice adding some code comments.

To the right, we've provided you with the beginning of the book Catch-22 by Joseph Heller.

On line 1, write a single line comment that says 'Opening line'.

1. Single line comments are great for adding context to your code.

Multi-line comments are often best suited to prevent a block of code from running. Comment out lines 4 - 9 with a multi-line comment.

Let's take one more glance at the concepts we just learned:

* There are three essential data types in JavaScript: strings, numbers, and booleans.
* We can write out anything to the console with console.log.
* We can do math with operators like +, -, \*, and /.
* We can find the remainder after dividing two numbers with a modulus: %.
* We can generate a random number with Math.random, then make it a whole number with Math.floor.
* We write a single line comment with // and a multi-line comment with /\* and \*/.

You've just finished one of the toughest parts of this course, nice work!