**Introduction**

JavaScript is a widely used web-based programming language that powers the dynamic behavior on most websites, including this one.

In this lesson, you will learn about *data types* and *built-in methods*, essential knowledge for all aspiring JavaScript developers. When you finish, you'll know how to write programs that solve mathematical expressions, compute the length of a word, and generate random numbers.

This lesson covers the building blocks of JavaScript. Make sure to take notes and pace yourself. This foundation will set you up for understanding more complex concepts later.

**Console**

Before you learn about data types and methods, you need to know how to print values to the console. The console is a tool that developers use to record the output of their JavaScript programs.

The console.log() command is used to print, or log, text to the console. Consider the following example:

console.log("Hello!");

In this example, Hello! is logged to the console. Notice, we denote the end of the line with a semicolon. Although your code will usually run as intended without a semicolon, it is best practice to always include one to ensure your code works as expected in situations when it does need one.

You'll see in the next exercise that you can put any data type inside the parentheses to print it to the console.

**Instructions**

**1.**

Use the console.log code in the editor to log your favorite pizza topping to the console.

Run your code when you are ready to see the result.

Hint

Create a word inside of quotations marks and put it between opening and closing parentheses, like:

console.log('put topping here');

**2.**

On the next line log your favorite book or movie to the console.

Hint

Create a word inside of quotations marks and put it between opening and closing parentheses, like:

console.log('put movie or book here');

|  |
| --- |
| console.log("Pizza");  console.log('Movie'); |
| Pizza  Movie |

# Data Types

Data types are the building blocks of all languages and essential pieces of any program.

Below are examples of four primitive data types that lay the foundation for all JavaScript programs. Primitive data types, as their name implies, are the simplest built-in forms of data.

console.log('New York City');

console.log(40.7);

console.log(true);

console.log(null);

In the example above, the computer logs each of the four primitive data types to the console. The types include:

* Strings — Any grouping of keyboard characters (letters, spaces, numbers, or symbols) surrounded by single quotes ('Hello') or double quotes ("World!"). In the example above, 'New York City' is a string.
* Numbers — Any number, including numbers with decimals: 4, 1516, .002, 23.42. In the example above, 40.7 is a number.
* Booleans — Either true or false, with no quotations. In the example above, true is a boolean.
* Null — Can only be null. It represents the absence of value.

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

**Math Operators**

Don't worry! Math doesn't need to be your strong-suit to learn JavaScript. However, there are operators you'll need to know to make useful programs.

JavaScript supports the following math *operators*:

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

These all work how you might guess:

console.log(3 + 4); // Equals 7 console.log(5 - 1); // Equals 4 console.log(4 \* 2); // Equals 8 console.log(9 / 3); // Equals 3

In the example above, each line uses a different mathematical operator to log a value to the console.

# Properties

When you introduce a new piece of data into a JavaScript program, the browser saves it as an instance of the data type. An instance is an individual case (or object) of a data type.

JavaScript will save a new piece of data, like 'Hello', as a string instance in the computer's memory. Another example, the number 40.7, is stored as an instance of the number data type.

An instance, like the string 'Hello', has additional information attached to it.

For example, every string instance has a property called length that stores the number of characters in it. You can retrieve property information by appending the string with a period and the property name:

console.log('Hello'.length);

In the example above, the value saved to the length property is retrieved from the string, 'Hello'. The program prints 5 to the console, because Hello has five characters in it.

**Built-in Methods**

While the length of a string is calculated when an instance is created, a string instance also has *methods* that calculate new information as needed. When these built-in methods are called on an instance, they perform actions that generate an output.

Built-in methods are called, or used, by appending an instance with a period, the name of the method, and opening (() and closing ()) parentheses. Consider the examples below:

console.log('Hello'.toUpperCase()); // 'HELLO' console.log('Hey'.startsWith('H')); // true

Let's look at each line separately:

* On the first line, the .toUpperCase()method is called on the string instance 'Hello'. The result is logged to the console. This method returns a string in all capital letters: 'HELLO'.
* On the second line, the .startsWith()method is called on the string instance "Hey". This method also accepts the character 'H' as an input between the opening and closing parentheses. Since the string 'Hey' does start with the letter 'H', the method returns the boolean true.

You can find a list of built-in string methods in the [JavaScript documentation](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/prototype). Developers use documentation as a reference tool. It describes JavaScript's keywords, methods, and syntax.

**1.**

Use the .toUpperCase() method to log the string Codecademy to the console in all capital letters.

Hint

Use console.log() to log the string to the console. Call .toUpperCase()on Codecademy inside of the parentheses.

**2.**

Use the [JavaScript documentation](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/prototype) to find a string method that trims the whitespace at the beginning and end of a string.

Use the method to remove the whitespace at the beginning and end of the following statement (also in **app.js**):

' Remove whitespace '

Hint

Append the string ' Remove whitespace ' with the method .trim().

|  |
| --- |
| https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/String/prototype |

|  |
| --- |
| // Log Codecademy in all uppercase letters  console.log('Codecademy'.toUpperCase());  // Use a string method to log the following statment without whitespace at the beginning and end of it.  console.log(' Remove whitespace '.trim()); |

**Libraries**

Instance methods, by definition, require that you create an instance before you can use them. What if you want to call a method without an instance? That's where JavaScript libraries come in. Libraries contain methods that you can call without creating an instance.

One such collection contains mathematical methods, aptly named the Math library.

Let's see how you call the .random()method from the Math library:

console.log(Math.random()); // random number between 0 and 1

In the example above, we called the .random() method by appending the library name with a period, the name of the method, and opening (() and closing ()) parentheses. This method returns a random number between 0 and 1. This code prints a random number between 0 and 1.

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

Math.random() \* 50;

The answer in the example above will most likely be a decimal. To ensure the answer is a whole number, JavaScript provides a built-in method called Math.floor(). Math.floor() takes a decimal number, and rounds down to the nearest whole number. You can use Math.floor() to round a random number like this:

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

In this case:

1. Math.random generates a random number between 0 and 1.
2. We then multiply that number by 50, so now we have a number between 0 and 50.
3. Then, Math.floor rounds the number down to the nearest whole number.

**Instructions**

**1.**

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

Hint

You can generate a random number between 0 and 100 with:

Math.random() \* 100

Since Math.random() will generate a number between 0 and 1, multiplying it by 100 will make the result between 0 and 100.

**2.**

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.

Hint

Math.floor(Math.random() \* 100);

By placing Math.random() \* 100inside of the parentheses of Math.floor, we can make sure that the number generated will be a whole number. For instance, if Math.random() \* 100 generated the number 64.65, Math.floor would make the result 64.

Note: Be careful of parentheses here. Each opening parentheses should have a closing parentheses.

**3.**

Find a method in the [JavaScript math library](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math) that returns the smallest integer greater than or equal to a decimal number.

Use this method with the number 43.8. Log the answer to the console.

Hint

Use the .ceil() method to calculate the smallest integer greater than or equal to 43.8.

**4.**

Use the [JavaScript documentation](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Number) to find a method in the Number library that checks if a number is an integer.

Put the number 2017 in the parentheses and use console.log()to print the result.

Hint

Use the isInteger() method from the Number library to check if 2017is an integer.

See an example of how to use isInteger() and log the result:

console.log(Number.isInteger(34.2));

|  |
| --- |
| console.log(Math.floor(Math.random()\*100));  console.log(Math.ceil(43.8));  console.log(Number.isInteger(2017)); |
| 39  44  true |

|  |
| --- |
| https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/Math/round |
| https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\_Objects/Number/isInteger |

# Comments

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

Programs do not evaluate comments when you run them. In other words, they exist just for human readers. Good comments are useful for you and other developers, because they describe what the code does.

There are two types of code comments in JavaScript:

1. 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);

1. 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); \*/

**Review Types and Operators**

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

* Four essential data types in JavaScript include strings, numbers, booleans, and null.
* Data is printed, or logged, to the console with console.log().
* Four built-in mathematical operators include +, -, \*, and /.
* JavaScript associates certain properties with different data types.
* JavaScript has built-in methods for different data types.
* Libraries are collections of methods that can be called without an instance.
* You can write single-line comments with // and multi-line comments between /\* and \*/.