TASK

Javascript I:

Variables and Operators

Visit our website

Introduction

Welcome to The First JavaScript Task!

Well done! The fact that you have reached this task means that you have been able to demonstrate your competence using HTML and CSS!

You are probably very eager to make your sites more dynamic now, though. For example, you have no doubt thought, “It’s great that I can make a form but how do I get it to work?”

You will learn to do this using JavaScript. This is a scripting language that allows you to create dynamic websites. In this task, you will learn what JavaScript is. You will also learn the basics of creating your first program using JavaScript! Once you have mastered some of the basics of creating a program with JavaScript in this task, you will learn to integrate JavaScript into your webpage in the next task.

Get in touch

Connect for support

Remember that with our courses, you’re not alone! You can contact your mentor to get support on any aspect of your course.

The best way to get help is to login to **www.hyperiondev.com/portal** to start a chat with your mentor. You can also schedule a call or get support via email.

Your mentor is happy to offer you support that is tailored to your individual career or education needs. Do not hesitate to ask a question or for additional support!

INTRODUCTION TO JAVASCRIPT

JavaScript is often used as a front-end scripting language. However, as you will learn later, it can also be used for server-side programming.

HTML, CSS and JavaScript are the three core technologies used for front-end web development. As you already know, HTML and CSS describe what content is shown on a webpage and what that content looks like. **JavaScript is used to make the webpage do things.** JavaScript is used to change web pages from static web pages (that always look the same) to dynamic web pages (pages that can dynamically change on the browser).

JAVASCRIPT BACKGROUND

Before you start learning to use JavaScript, it is important to quickly cover some theory and background related to JavaScript.

ECMAScript

ECMA is an international organization that creates standards for information and communication systems. ECMAScript is a standard, created and maintained by ECMA, that defines the JavaScript general-purpose programming language. In other words, *ECMA* is basically *the organization that has designed JavaScript* and *EMCAScript is basically JavaScript*. ECMAScript is based on several originating technologies including JavaScript and JScript. ECMAScript was first released in 1997. As you can probably imagine, standards for programming languages have had to be updated massively since the ECMAScript originated to accommodate the huge changes that have taken place in technology since then. This has resulted in several versions/editions of ECMAScript. The 6th edition of the ECMAScript (ES6), which was released in 2015, is the biggest revision of ECMAScript since 1997. ES6 is sometimes also referred to as ES2015 or Harmony.

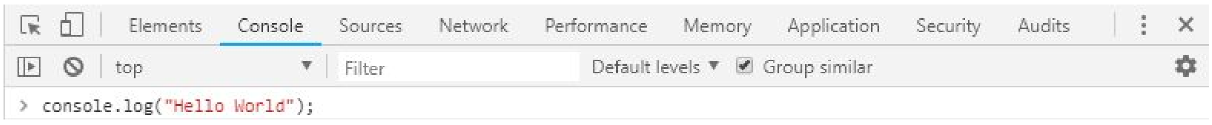
ES6, ES7 and ES8

ES6 "... is the most extensive update to ECMAScript since the publication of the first edition in 1997." ES6 has introduced many changes to JavaScript (which basically is the same thing as ECMAScript) which improve the programming language greatly. Besides ES6, though, there are also already versions ES7 and ES8. This is because, since the release of ES6, the ECMA has decided to release updates to ECMAScript every year. All versions of ECMAScript after ES6 should, therefore, have fewer revisions than ES6.

When you start coding in JavaScript you may notice that there are often different ways of writing the same code. The reason for this is often that some code will be written using the 'older' version of JavaScript while other code will be written using the changes to the language that have been introduced since ES6. The updates to JavaScript since ES6 will be highlighted in your tasks. *That's the theory out of the way! Let's get coding!*

USING THE JAVASCRIPT CONSOLE

All modern browsers offer built-in support for JavaScript (you're using it without even realising it). Browsers have a built-in console that can be used for debugging web pages. The functionality of the console may differ slightly based on the browser you use. In this task, we will be using **Chrome's** DevTools Console. To access this tool, open Chrome and then press either **Ctrl+Shift+J if you are using Windows / Linux** or **Cmd+Opt+J if you are using Mac**. You should see something similar to the screen shown in the image below. The console may already display some messages. You can clear these by right clicking on the console and then selecting "Clear console."



```
> console.log("Hello World");
```

You can input your JavaScript code directly into this console to test and debug it. To write information to the console we use the instruction:

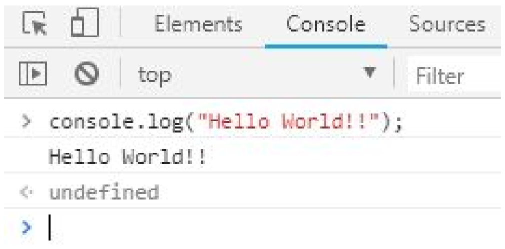
console.log("Whatever you would like to write to the console");

The above is a line of JavaScript code. It instructs your computer to log (or write) the text in the quotation marks to the console.

Try this:

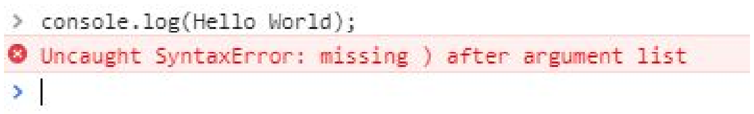
1. Open the Console.
2. Type the code console.log("Hello World!!"); into the console.
3. Press enter.

If you typed everything correctly, "Hello World!!" should be shown in the console as shown below:



```
> console.log("Hello World!!");
  Hello World!!
<· undefined
>
```

If you haven't typed the instruction *exactly* as shown (for example if you forgot to add the quotation marks, or you used capital letters where it should have been lowercase letters, or you spelled "console" incorrectly) you may get an error. See an example of an error below:



```
> console.log(Hello World);
✖ Uncaught SyntaxError: missing ) after argument list
> |
```

Like all programming languages, JavaScript has **syntax rules** that must be followed closely otherwise your instructions will not be correctly interpreted and executed.

A note from our coding mentor
Melanie

Syntax is the "spelling and grammar rules" of a programming language and determines how you write correct, well-formed statements. These syntax rules are very strict. If you don't follow the rules exactly, your code won't be recognised and therefore, won't execute. A common Syntax error you could make is forgetting to add a semicolon ';' at the end of each statement. The process of removing all the errors from your code is referred to as debugging.

Now try typing the following code into the console:

```
alert("Hello World!! I can write an instruction using JavaScript");
```

Press enter and see what happens.

Imagine that you wanted to write the code that prints the message "Welcome to my web page Tom" to the console but instead of using the name "Tom" every time, you'd like to be able to change the name based on who is currently viewing your web page. To do this we need a variable.

VARIABLES

A script is basically a set of instructions that are interpreted to tell the computer what to do in order to accomplish a certain task. Programs usually process data that is somehow input into the program and output the results of the processing. This data must be stored somewhere so that it can be used and processed by the instructions we code. When coding, we use **variables** to store the data we need to manipulate.

A variable is a way to store information. It can be thought of as a type of "container" that holds information. We use variables in calculations to hold values that can be changed.

DECLARING VARIABLES

Before we can use variables in our code, we need to **declare** them. To declare a variable is to assign it a storage space in memory and give it a name for us to reference it with. This tells JavaScript that we want to set aside a chunk of space in the computer's memory for our program to use.

In JavaScript we use the following format to create a variable and assign a value to it:

```
var variable_name = value_you_want_to_store;
```

1. Notice from the code above that the first thing you need to do to declare a variable is to use the keyword 'var'. Since ES6, this is not always the case, but more on this later.
2. After that, you declare the name of the variable. You can name a variable anything you like as long as the name:
    a. Contains only letters, numbers, underscores and dollar signs. No other characters can be used in variable names including spaces. "My name" would thus not be an acceptable variable name.
    b. Starts with a letter.
    c. Is not a reserved word. In JavaScript, certain words are reserved. For example, you would not be able to name a variable "var", "console" or "log", because these are reserved words.

   If you follow these rules you can call your variables whatever you want. It is, however, good practice to give your variables *meaningful names*.

Below is an example of bad naming conventions vs good naming conventions.

- myName = "Tom"   # Good variable name

- variableOne = "Tom"   # Bad variable name
- string_name = "Tom"   # Good variable name
- h4x0r = "Tom"   # Bad variable name

myName and *string_name* are examples of descriptive variables as they reveal what what content they store.
variableOne and *h4x0r* are terrible names because they are not descriptive.

To assign a value to a variable, you need to use the assignment operator. This is the equal-to sign (=) we usually use in maths. It takes the value on the right-hand side of the = and stores it in the variable on the left-hand side. For example, consider the line of JavaScript code below:

```
var myName = "Tom";
```

That statement would cause your computer to create an area in memory (i.e. a variable) called "myName" and put the value "Tom" into that area in memory.

It is important that you always put the value you want to assign (or put into) your variable on the right-hand side of the assignment operator (=).

Try this:
Add the following code to your console:

```
var myName = "Tom";
console.log("Hi there " + myName);
```

You have now successfully used a variable! In the example above, "myName" is referred to as a variable because the value stored in the position in memory named "myName" will *vary*/change throughout our program. You will learn how to get values to store in variables from users and other sources in later tasks.

Variables store data and the type of data that is stored by a variable is intuitively called the **data type**.

ESSENTIAL DATA TYPES

Within JavaScript, you'll find yourself working with many data types based on the kind of application you're dealing with. There are four main data types with which you should familiarise yourself as they form the basis of any JavaScript program:

Data Type	Declaration
Numeric	var someNumber = 25;
String	var someName = "Joe";
Boolean	var someBool = true;
Array	var someArray = [15, 17, 19];
Object	var someObject = {firstName: "James", lastName: "Bond"}

Numeric data types describe any numbers that you store. **Strings** refer to a combination of characters. "Joe", "23 Main Street", "Hyperion" are all examples of strings. We use strings to store and manipulate text. String values must always be put within quotation marks (" "). **Booleans** are data types that can store only two values: either true or false. An **array** is a data type that we use to store multiple values. As shown in the table above, we put all the values we want to store in an array variable in a comma-separated list that is enclosed by square brackets ([ ]). You will learn a lot more about **objects** later, but for now, it is important to know that an object is a data type that stores a collection of related data. If you wanted to create a person object, for example, you would store a collection of related information that describes a person such as their name, surname, date of birth, address etc.

TYPE IDENTIFICATION

In some programming languages, you have to tell the computer what data type you want a variable to store when you declare the variable. This is not true with JavaScript. JavaScript is smart in the sense that it's able to detect variable types automatically based on the value that you assign to your variable. If a value is in quotation marks, JavaScript knows that the value is a string. It would likewise automatically know that is should store the value 12 as a number.

```
var myString = "12";
var myNumber = 12;
```

JAVASCRIPT INTELLIGENCE

So what happens if you were to code this line?

```
var unknownType = 53 + "Bond";
```

In many programming languages this would cause a *type conflict error* because mathematically you can't add a number and a string, but JavaScript simply solves this issue by converting the entire variable contents into a string. The logic is that JavaScript first sees a number and assumes an number variable type but, once it detects a string, the variable is reclassified as a string.

FINDING THE TYPE

Sometimes, you may want to check some data to inspect its data type property. This is done by making use of the 'typeof' built-in function.

```
typeof "Bond";  // this returns a string
typeof [4,6,2];   // this returns an object
```

Try this:

1. Enter the following code into your console and then press enter.

```
var myName = "Tom";
var num = 33.33;
var pass = true;

console.log(myName);
console.log(num);
console.log(pass);

var myDataType = typeof num;
console.log(myDataType);

console.log(typeof pass);
```

2. Take careful note of the output. Be sure to understand how the code you entered resulted in the output displayed in the console.

MATHEMATICAL CALCULATIONS WITH JAVASCRIPT

Doing calculations with numbers in JavaScript is similar to the way they would be done in normal math. The only difference between calculations in real mathematics and programming is the symbols you use as shown below:

Arithmetic Operations	Symbol used in Python
Addition	+
Subtraction	-
Multiplication	*
Division	/
Modulus (Divides left-hand operand by right-hand operand and returns remainder e.g. 5%2 = 1)	%
Add one to a variable (e.g. 2++ = 3)	++
Subtract one from a variable (e.g. 2-- = 1)	--

THE MODULUS OPERATOR

It is important that we discuss one special operator. It is crucial for a programmer to know how to use this operator because it is used to solve many computational problems. It is the modulus operator. Here is an example of it in use:

```
var remainder = 152 % 10;
```

Given the division problem 152 / 10, the answer is given as two parts. The quotient is 15 and the remainder is 2. We informally say the answer is 15 remainder 2. The modulus operator is a means of getting the remainder of a division problem directly. So the result of the expression above would be 2.

Try this:

1. Type the following JavaScript into the console.

```
var num1 = 12;
var num2 = 34;
```

```
console.log("num1 = " + num2);
console.log("num2 = " + num2);
console.log("num1 + num2 = " + num1+num2);
console.log("num1 / num2 = " + num1/num2);
console.log("num2 % num1 = " + num2%num1);
console.log("num1++ = " + num1++);
console.log("num2-- = " + num2--);
```

2. Press enter and take careful note of the output. Be sure to understand how the code you entered resulted in the output displayed in the console.

A note from our coding mentor
Joseph

Note:
In JavaScript the + sign can be used to add two numbers OR to concatenate a value to a string. To concatenate things means to link things together. Consider this line of code in the example above: console.log("num1 + num2 = " + num1+num2);. Here the variables num1 and num2 are added together and then the result is concatenated with the string "num1 + num2 = ".

CREATING .js FILES

All the code we have written so far has been written directly into Google's DevTools Console. As soon as you close your browser though, all the code you wrote in the console will be lost. To write JavaScript code that you can save locally and reuse for your websites, create JavaScript files. To do this, simply create a new file (using a text editor like Notepad++ or Sublime Text), enter the JavaScript instructions in this file one instructions per line and then save this file with the .js extension. You will learn how to get your JavaScript files to work with your HTML files very soon.

Save all your compulsory tasks as JavaScript files!

Use either the JavaScript console or Sublime Text (or another editor of your choice) to execute and debug JavaScript in the next few tasks. To see how to configure

Sublime Text for debugging and executing JavaScript, please see the FAQs document that accompanies this task.

Instructions

Open *example.js* in Sublime Text and read through the comments before attempting these tasks.

Getting to grips with JavaScript takes practice. You will make mistakes in this task, this is completely to be expected as you learn the keywords and syntax rules of this programming language. It is vital that you learn to debug your code. To help with this remember that you can:

- Use either the JavaScript console or Sublime Text (or another editor of your choice) to execute and debug JavaScript in the next few tasks. To see how to configure Sublime Text for debugging and executing JavaScript, please see the FAQs document that accompanies this task.
- Remember that if you really get stuck you can contact your mentor for help.

Compulsory Task 1

Follow these steps:

- Create a JavaScript file called “myVariables.js”.
- Create a variable called “language” and assign it the value “JavaScript”. Then create a variable called “score” and assign it the value “10”. Write both variables to the console.
- Create two variables called “length” and “width”. Assign each variable a value. Use the variables to calculate the area of a rectangle with the length and width specified (remember area = length x width). Write the following to the console: “The area of the rectangle is ...” where ... is the area you calculated.
- Create two variables called “num1” and “num2.” Assign each variable a value. Calculate and display what the remainder is if num1 is divided by num2.
- OPTIONAL: Using the variables you created in the previous step, output the results of the division in the following format: “num1 / num2 = x remainder y.” (Research Math.trunc to get this to work.)
- Use HTML-CSS-JS Prettify to improve the readability of your code before submitting it to your mentor.

Compulsory Task 2

Follow these steps:

- Create a .js file called 'dataTypes.js'.
- Define the following data types, with your own contents:
    - Integer
    - Double
    - String
    - Array
    - Object
- Once you've created the variables, display them to the console.
- Use the 'typeof' function to check each variable.
- Create an array with numbers and strings. Comment what type you expect this to be.
- Use the 'typeof' function on this array.
- Create a null variable and an undefined variable (you'll need to do some research to do this).
- Use HTML-CSS-JS Prettify to improve the readability of your code before submitting it to your mentor.

Once you have completed the task in line with the instructions above, click the button below to request your mentor to review your work and provide feedback. If you have any questions while attempting the task, leave your mentor a note on the comments.txt file in your Student Dropbox folder.

Completed the task(s)?

Ask your mentor review your work!

Review work

Things to look out for:

1. Make sure that you have installed and setup all programs correctly. Follow the instructions in the FAQs document that accompanies this task to configure Sublime Text for debugging and executing JavaScript.

2. If you are not using Windows, please ask your mentor for alternative instructions if needed.

Rate us

Share your thoughts

Hyperion strives to provide internationally-excellent course content that helps you achieve your learning outcomes.

Think the content of this task, or this course as a whole, can be improved or think we've done a good job?

Click here to share your thoughts anonymously.