



TASK

Your First Computer Program

Visit our website

Introduction

WELCOME TO YOUR FIRST COMPUTER PROGRAM TASK!

In this task, you are introduced to the JavaScript programming language. 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! In doing so, you will become familiar with the structure of a JavaScript program.



Get in touch
Connect for support

Remember that with our courses, you're not alone! You can contact an expert code reviewer to get support on any aspect of your course.

The best way to get help is to login to Discord at <https://discord.com/invite/hyperdev> where our specialist team is ready to support you.

Our expert code reviewers are 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. 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 cover some theory and background related to JavaScript first.

ECMAScript

ECMA is an international organisation 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 organisation that has designed JavaScript and ECMAScript 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

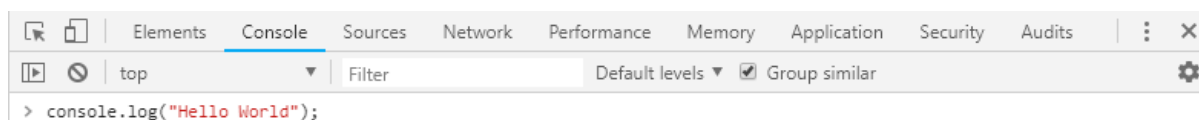
ES6 has introduced many changes to JavaScript (which basically is the same thing as ECMAScript) that improve the programming language greatly. Besides ES6, though, there are also already versions ES7 and ES8. This is because, since the release of ES6, the ECMA has decided to release updates to ECMAScript 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 right-click → Inspect, or 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."



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:

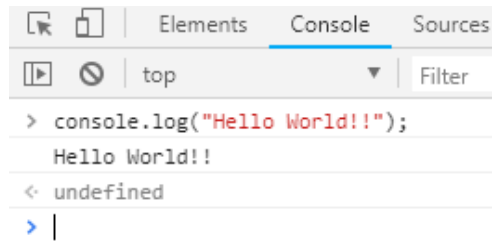
```
console.log("Whatever you would like to write to the console");
```

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:

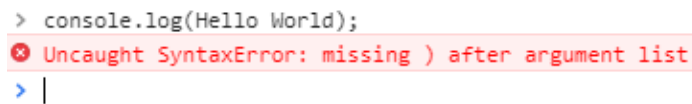
1. Open the Console.
2. Type the code `console.log("Hello World!!");` into the console.
3. Press enter.

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

A screenshot of a web browser's developer console. The 'Console' tab is selected. The command `console.log("Hello World!!");` has been entered and executed. The output shows `Hello World!!` on the first line and `undefined` on the second line. The prompt `>` is visible at the bottom left of the console area.

```
> 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 spelt "console" incorrectly) you may get an error. See an example of an error below:

A screenshot of a web browser's developer console showing a syntax error. The command `console.log(Hello World);` has been entered. Below it, a red error message is displayed: `Uncaught SyntaxError: missing) after argument list`. The prompt `>` is visible at the bottom left.

```
> 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.

SYNTAX RULES

All programming languages have *syntax* rules. Syntax is the "spelling and grammar rules" of a programming language and determines how you write correct, well-formed statements.

A common syntax error you could make above is forgetting to add a closing quotation mark ("). Remember that all opening quotation marks (") require a closing one! Another common syntax error that you could make above is by forgetting to add a closing bracket). Remember that all opening brackets (require a matching closing one!

Any program you write must be exactly correct. All code is case sensitive. This means that 'Console' is not the same as 'console'. If you enter an invalid JavaScript command, misspell a command or misplace a punctuation mark, you will get a syntax error when trying to run your program.

Errors appear in the JavaScript console when you try to run a program and it fails. Be sure to read all errors carefully to discover what the problem is. Error reports will even

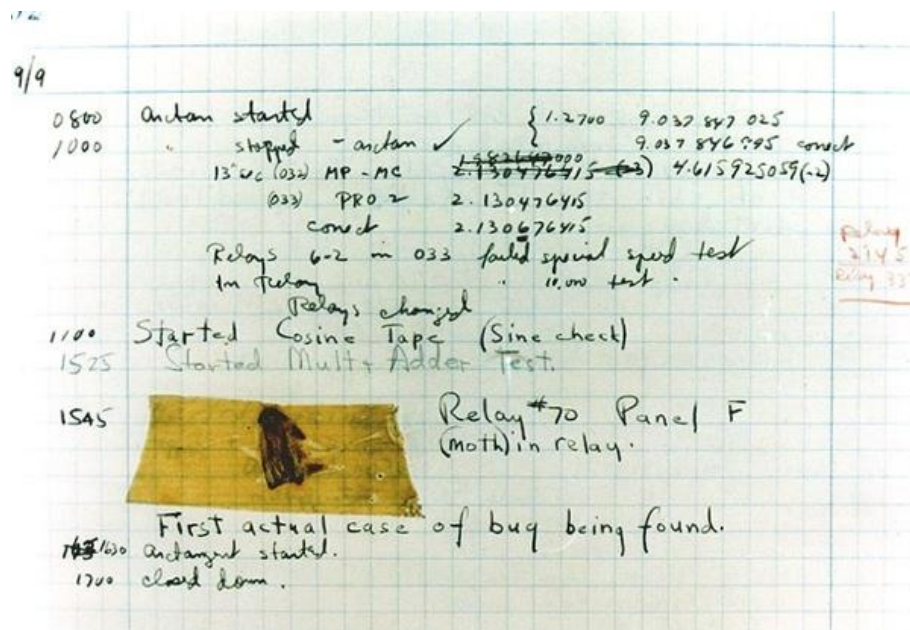
show you what line of your program had an error. The process of resolving errors in code is known as *debugging*.



A note from Riaz...

Sorry to interrupt, but did you know that the first computer “bug” was named after a real bug? Yes, you read that right! Although the term “bug”, in the sense of a technical error, was first coined by Thomas Edison in 1878, it was only 60 years later that someone else popularised the term.

In 1947, Grace Hopper, a US Navy admiral, recorded the first computer ‘bug’ in her logbook as she was working on a Mark II computer. A moth was discovered stuck in a relay and thus hindering the operation. She proceeded to remove the moth, thereby ‘debugging’ the system, and taped it in her logbook. In her notes, she wrote, “First actual case of bug being found.”



- **Riaz Moola**, Founder and CEO

WRITING A JAVASCRIPT PROGRAM

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.

The code above will always produce the same output, but sometimes we want some aspects of the output to change each time the code is run. For example, 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:

```
let 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 **let**. You can also use the keywords **var** and **const** to declare variables. **var** is the old way of creating a variable. **const** is used when you want to declare a variable such that its value can't change, i.e. the variable stores a constant value. **const** is a great option when you want to make sure a variable doesn't accidentally get reassigned a different value.
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 and underscores. All other characters are prohibited from use in variable names, including spaces. "My name" would thus not be an acceptable variable name, whereas "My_Name" would be acceptable.
 - 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 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. Finally, we finish the line with a semicolon (;). For example, consider the line of JavaScript code below:

```
let 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:

```
let 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**.

We can also ask the user to give us the value for a variable. We do this using `prompt()`. Have a look below:

```
let myName = prompt("What is your name?");    // Asks the user their name  
console.log("Hi there, " + myName);           // Will show "Hi there, [myName]"
```

The code above prompts the user to input their name, and then logs “Hi there, “ and their name to the console.



Take note:

ES6 Update: No more ‘var’

With previous versions of ECMAScript, you could only declare variables with the keyword **var**. With ES2015 (or ES6), variables are declared with either the keywords **let** or **const**. The concept of variable **scope** () becomes important here. For more on this, and an introduction to local and global variables, see [here](#). Variables declared with the keywords **let** or **const** are scoped within the code block they are declared in. Variables declared using the keyword **var**, on the other hand, are declared either locally or globally in relation to a function (which sometimes leads to errors). **const** variables must be assigned a value when they are declared

and the value they are assigned cannot change; **let** variables can be re-assigned a value within the block where they have been declared but they cannot be redeclared.

In summary, with ES2015:

- Don't use **var** any more.
- Use **const** whenever possible to initialise variables
- Use **let** whenever the value of that variable needs to be changed in a block of code.

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, you'll need to create JavaScript files. In this course, we will be using Visual Studio Code to write and save files from now on. To set up Visual Studio Code, follow the step-by-step guide in this task file.

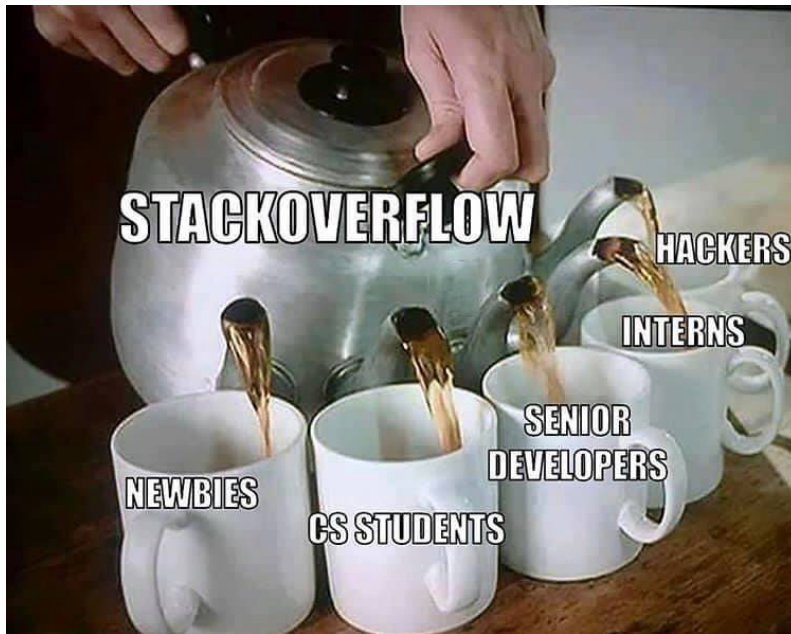
Once everything is installed and set up, simply create a new file and save it with the .js extension.

Remember: Save all your compulsory tasks as JavaScript files!



Take note:

Coding can be an intimidating process, especially for a beginner. Be sure to look up anything you feel unsure about — the internet is full of very clever and helpful people! Remember that looking up your problems is something that ALL programmers do, from brand new coders to the most advanced senior developer. Don't believe me? Check out [this](#) post. Keep in mind the difference between looking up problems and plagiarism (check back to Compulsory Task 3 in IFS L1t01 if you need a reminder).



Instructions

Open **example.js** and **index.html** in Visual Studio Code 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 Visual Studio Code (or another editor of your choice) to execute and debug JavaScript in the next few tasks. To see how to configure Visual Studio Code for debugging and executing JavaScript, please see the additional reading that accompanies this task.
- Remember that if you really get stuck, you can contact an expert code reviewer for help.

Compulsory Task 1

Follow these steps:

- Create a new JavaScript file in this folder called **greeting.js**
- Inside this file, write JavaScript code to take in a user's name and then print out the name.
- Also, take in a user's age and print out the age.
- Finally, output a newline character (essentially the same as pressing enter, to move the output to a new line) and the string "It's nice to meet you!"

Compulsory Task 2

Follow these steps:

- Create a new JavaScript file in this folder called **shopping.js**
- Write a program that takes items to make a shopping list. Store them as variables *item1*, *item2* and *item3*. The user needs to input each item.
- Output each item separately.
- E.g. your program could print:

My Shopping List:

Milk

Lettuce

Frozen pizza



Rate us

Share your thoughts

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

Think that 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.

