

# 1

# The world of JavaScript

We are going to start our journey by looking at the world of JavaScript. We'll begin by considering just what it is that a programming language does. Then we'll investigate the JavaScript programming language and discover how JavaScript programs get to run on your computer. We'll learn how web pages provide an environment for JavaScript and how to use HyperText Markup Language (HTML) and Cascading StyleSheets (CSS) to create containers for our JavaScript programs. We'll discover just how powerful modern web browsers are as software development tools and how to have a conversation with JavaScript from within a browser. We'll also learn how to manage our software source code and share it with others.

# 1

## Running JavaScript

### What you will learn

Programmers have a set of tools and techniques they use when they create programs. In this chapter, you're going to discover how JavaScript programs run on a computer. You'll also have your first of many conversations with the JavaScript command prompt and investigate your first JavaScript program. Finally, you'll download the Git and Visual Studio Code tools and the example programs for this book and do some simple editing.

### What is JavaScript?

Before we go off and look at some JavaScript it's worth considering just what we are running. JavaScript is a *programming language*. In other words, it's a language that you use to write programs. A program is a set of instructions that tells a computer how to do something. We can't use a "proper" language like English to do this because "proper English" is just too confusing for a computer to understand. As an example, I give you the doctor's instructions:

"Drink your medicine after a hot bath."

We would probably have a hot bath and then drink our medicine. A computer would probably drink the hot bath and then drink its medicine. You can interpret the above instructions either way because the English language allows you to write ambiguous statements. Programming languages must be designed so that instructions written using them are not open to interpretation, they must tell the computer precisely what to do. This usually means breaking actions down into a sequence of simple steps:

Step1: Take a hot bath Step2: Drink your medicine
--

We can get this effect in English (as you can see above) but a programming language forces us to write instructions in this way. JavaScript is one of many programming languages which have been invented to provide humans with a way of telling the computer what to do.

In my programming career, I've learnt many different languages over the years and I confidently expect to have to learn even more in the future. None of them are perfect, and I see each of them as a tool that I would use in a particular situation, just like I would choose a different tool depending on whether I was making a hole in a brick wall, a pane of glass or a piece of wood.

Some people get very excited when talking about the "best" programming language. I'm quite happy to discuss what makes the best programming languages, just like I'm happy to tell you all about my favorite type of car, but I don't see this as something to get worked up about. I love JavaScript for its power and the ease with which I can distribute my code. I love Python for its expressiveness and how I can create complex solutions with tiny bits of code. I love the C# programming language for the way it pushes me to produce well-structured solutions. I love the C++ programming language for the way that it gives me absolute control of hardware underneath my program. And so on. JavaScript does have things about it which make me want to tear my hair out in frustration. But that's true of the other languages too. And all programming languages have things about them I love. But most of all I love JavaScript for the way that I can use it to pay my bills.

## Programmer's Point

### The best programming language for you is the one that pays you the most

I think it is very fitting that the first programmer's point is one that has a strong commercial focus. Whenever I get asked which is the "best" programming language I always say that my favorite language is the one that I get paid the most to use. It turns out that I'll write in any programming language if the price is right.

I strongly believe that you can enjoy programming well in any language, and that includes JavaScript. Conversely, you can have a horrible time writing bad programs in any language. The language is just the medium which you use to express your ideas.

So, if you tell someone that you're writing JavaScript programs and they tell you that it's not a very good programming language for reasons that you don't understand, just show them how many jobs there are out there for people

who can write JavaScript code.

## JavaScript origins

You might think that programming languages are a bit like space rockets, in that they are designed by white-coated scientists with mega-brains who get everything right first time and always produce perfect solutions. However, this is not the case. Programming languages have been developed over the years for all kinds of reasons, including ones like “it seemed a good idea at the time”.

JavaScript was invented by Brendan Eich of Netscape Communications Corporation and first appeared in a Netscape web browser that was released at the end of 1995. The language had a variety of names before the company decided on JavaScript. It turned out to be a poor choice of name because it makes it easy to confuse JavaScript with the Java programming language, which is actually quite different from JavaScript.

JavaScript was intended as a simple way of making web pages interactive. Its name reflects the way that it was supposed to be used alongside Java applications (called *applets*) running in a web browser. However, JavaScript was extended beyond all the expectations of its creator and is now one of the most popular programming languages in the world. Whenever you visit a web site it is almost certain that you will be talking to a JavaScript program.

This book will teach you JavaScript, but actually I’m trying to turn you into a programmer. The fundamentals of program creation are the same for JavaScript and pretty much all programming languages. Once you’ve learned how to write JavaScript you’ll be able to transfer this skill into many other languages, including C++, C#, Visual Basic and Python. It’s a bit like the way that once you have learned to drive you can drive any vehicle. When you are using a strange car you just need to find out where the various switches and controls are, and then you can set off on your journey.

## JavaScript and the web browser

The inventor of JavaScript intended for it to be used in a web browser and that is where we are going to start using it. It is possible to create JavaScript programs that run outside the browser, we will consider how to do this in the third part of this text. You can use any modern browser, but the exercises in this text use a browser based on the Chromium framework. I’m using Microsoft Edge Chromium which is available for Windows PC and Apple Macintosh. You can use Google Chrome or the Chromium browser for Linux if you prefer.

## Our first brush with JavaScript

You’ve reached a significant point in the process of learning how to program. You’re about to begin exploring how programs work. This is a bit like opening the front door of a new apartment or house or getting into a shiny new car you’ve bought. It’s an exciting time, so take a deep breath, find a nice cup (or glass) of something you like to drink and settle down comfortably.

You are going to start by doing something that you’ve done thousands of times in the past. You are going to visit a

site on the World Wide Web. But then, with a single press of a key, you're going to explore a world behind the web page and get a glimpse of the role that JavaScript plays in making it work.

## Make Something Happen

## A web page with secrets

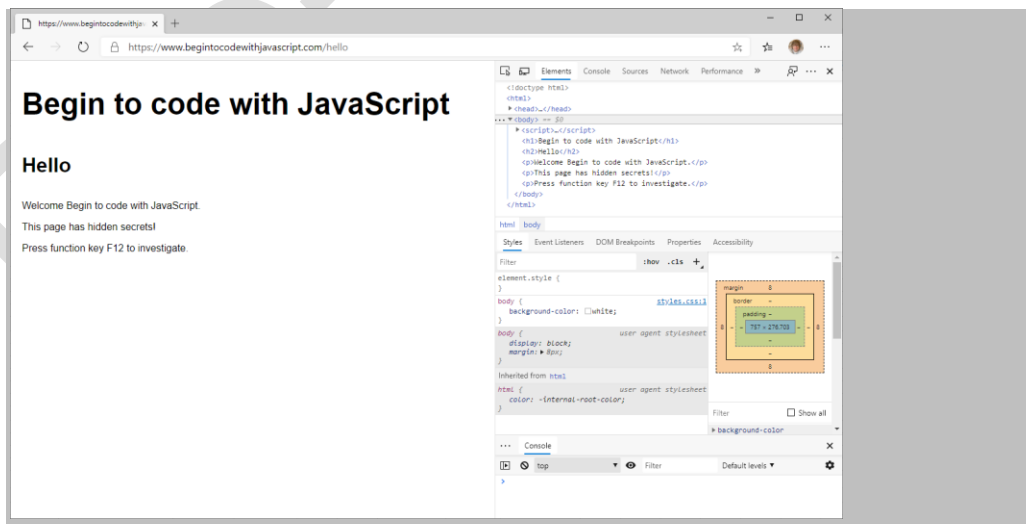
First you need to open your browser. Then visit the web page:

<http://www.begintocodewithjavascript.com/hello>



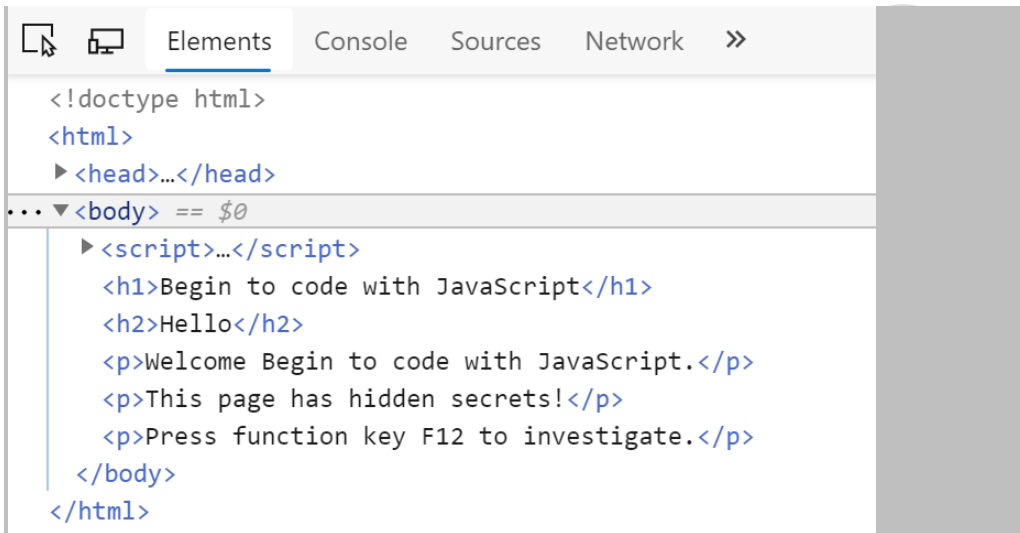
### 1.1 Ch01\_inset01\_01 A web page with secrets

This looks like a very ordinary web page. But it holds a secret behavior that you can find by pressing the F12 key on your keyboard.



## 1.2 Ch01\_inset01\_02 Developer View

This is called *Developer View*. It shows all the elements that make up the web page. A complete description of everything you can do in this view would not fit in this book. Don't be worried about how complicated it all looks, we are only going to use a couple of the features. We are going to start by looking at the elements that make up the text on the page. Make sure that the Elements tab is selected as you can see in the figure above. Then look at the text underneath.

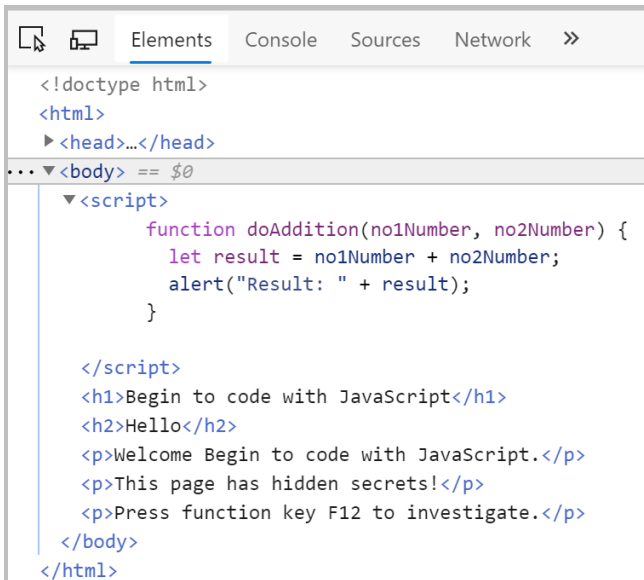


The screenshot shows the Chrome Developer Tools interface with the 'Elements' tab selected. The HTML structure is displayed as follows:

```
<!doctype html>
<html>
  <head>...</head>
  <body> == $0
    <script>...</script>
    <h1>Begin to code with JavaScript</h1>
    <h2>Hello</h2>
    <p>Welcome Begin to code with JavaScript.</p>
    <p>This page has hidden secrets!</p>
    <p>Press function key F12 to investigate.</p>
  </body>
</html>
```

## 1.3 Ch01\_inset01\_03 Page elements

The Figure above shows the elements on this page. You can see that the text that appears on the page is here. Parts of the text are enclosed in what look like formatting instructions, for example some is marked as `<h1>` and some as `<p>`. If you look back at the web page as displayed you will notice that the `<h1>` text is in a large heading font, whereas the `<p>` text is in smaller text. This is how pages are formatted. You can see how the page works, but where is the secret? To answer this, click the right-pointing arrowhead at the left of the word `script` to open this part of the view.



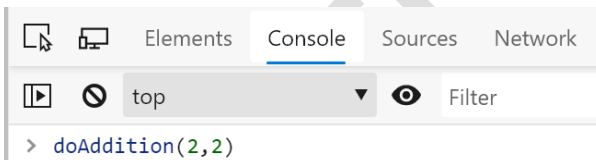
```
<!doctype html>
<html>
  <head>...</head>
  <body> == $0
    <script>
      function doAddition(no1Number, no2Number) {
        let result = no1Number + no2Number;
        alert("Result: " + result);
      }

    </script>
    <h1>Begin to code with JavaScript</h1>
    <h2>Hello</h2>
    <p>Welcome Begin to code with JavaScript.</p>
    <p>This page has hidden secrets!</p>
    <p>Press function key F12 to investigate.</p>
  </body>
</html>
```

#### 1.4 Ch01\_inset01\_04 JavaScript function

Clicking the arrow opens that part of the listing. The hidden feature is a function called `doAddition`. This takes two numbers, adds them together and displays the result using an alert. Later in the text we will go into detail of how this JavaScript works, but even at this stage it is quite clear what is going on.

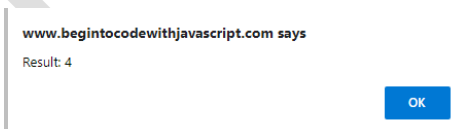
However, this function is never actually used in the webpage. We can use it ourselves by entering it into the JavaScript console which is built into the browser. This performs JavaScript statements that you type in. You can open the console by selecting the tab at the top of the window.



```
> doAddition(2,2)
```

#### 1.5 Ch01\_inset01\_05 JavaScript console

This is the Console window. I've typed in the name of the function and given it two numbers to work on. When the function runs it display an alert with the result it has calculated.



#### 1.6 Ch01\_inset01\_06 Result alert

We can use the JavaScript console to type in other JavaScript commands. You can use JavaScript to perform calculations by just entering them. When you press the Enter key the answer will be displayed. In fact, the console is often keen to give you an answer even before you press Enter. The console will also try to help as you type in by

suggesting what you might be typing. You can accept any suggestion by using the cursor key to select the suggestion and then pressing the Tab key.

## Code Analysis

We can learn a little about the way JavaScript works by giving the JavaScript console some commands and considering the responses.

```
> 2+3
```

This looks like a sum, and as you might expect, you get a number for an answer

```
<-5
```

We can repeat this with something other than a number:

```
> "Rob"+" Miles"
```

Some text enclosed in double quotes is interpreted by JavaScript as a string of text and it is perfectly happy to use + to add two strings together. Note that if you want to have a space between the two words in the sum you have to actually put it into the strings that you add together (in my example above there is a space in front of the 'M' in the second word.

```
<- "Rob Miles"
```

We can do other kinds of sums, for example we can subtract using minus (-).

```
> 6-5
```

This produces the result that you would expect.

```
<- 1
```

JavaScript seems quite happy when we ask it to do sensible things. Now, let's try asking it to do something stupid. What do you think would happen if we tried to subtract one string from another using the following statement?

```
> "Rob"- " Miles"
```

While it seems sensible to regard + as meaning "add these strings together" there doesn't seem to be a sensible interpretation of minus when you are working with strings. In other words, it is meaningless to try and subtract one string from another. If you enter this into the console you get a strange response from JavaScript

```
<- NaN
```

The JavaScript console is not calling for grandma to come and sort the problem out. The value "NaN" means "not a number". It is a way that JavaScript indicates the result of a calculation has no meaning. Some programming



languages would display an error message and stop a program if you tried to use them to subtract one string from another. JavaScript does not work like this. It just generates a result value that means “this result is not a number” and keeps going. We will consider how to a program can manage errors like this later in the text.

And since we are talking about errors, how about asking JavaScript to do some silly math:

```
> 1/0
```

When I got my first pocket calculator the first I tried to do with it was calculate one divided by zero. I was richly rewarded by a result that just kept counting upwards. What do you think JavaScript will do?

```
<- Infinity
```

JavaScript says that the result of the calculation is the value “Infinity”. This is another special value that is generated by JavaScript when it does calculations. Talking of calculations, how about asking JavaScript to do another one for us.

```
> 2/10+1/10
```

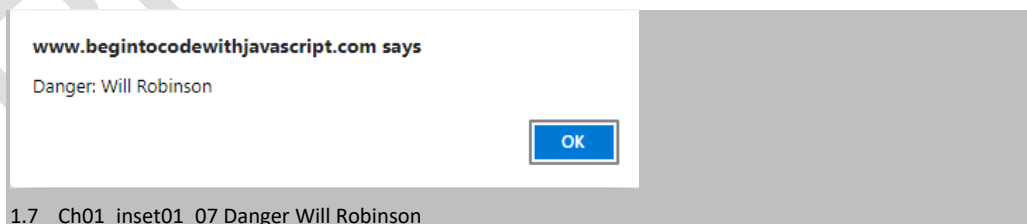
This calculation involves real numbers (i.e. ones with a fractional part). The calculation is adding 0.2 to 0.1 (a fifth to a tenth). This should produce the result 0.3 but what we get is interesting:

```
<- 0.30000000000000004
```

This number is very, very close to 0.3 (the correct answer) but is ever so slightly larger than it should be. This illustrates an important aspect of the way that computers work. Some values that we can express very easily on paper are not held exactly by the machine. This is only usually a problem if we start performing tests with the values that we calculate, for example a check to see if the calculated result above was equal to the value 0.3 might fail because of the tiny difference. Let’s see if we can use JavaScript to do some things for us. How about this:

```
> alert("Danger: Will Robinson")
```

This statement doesn’t calculate a result. Instead it calls a function called alert. The function is provided with a string of text. It asks the browser to display the string as a message in an alert box.



This is how the `doAddition` function displays the result it has calculated. Finally, lets try another function called `print`:

```
> print()
```

What you will see next depends on the computer and the browser that you are using. But you should see a print window appear which offers you the chance to print the web page. If you've ever wondered what happens when you press the print button on a web page; you know now.

Congratulations. You now know how web pages work. Your browser fetches a file from the server and then follows the instructions in that file to build a page for you to look at. The file contains text that is to be displayed on the page along with formatting instructions. A page file can also contain JavaScript program code.

The instructions that the browser follows are expressed in a language called Hyper-Text Markup Language (HTML). In the next chapter we'll take a detailed look at HTML. But before we do that, we need to get some tools that will let us fetch the sample code for this book onto our computer and work with HTML and JavaScript.

## Tools

You will need some software tools to get the best out of the exercises in this book. We are going to start with two, a program called "git" that will manage the program files that we work on and a program called "Visual Studio Code" which we will use to work on the files. Neither of these will cost you any money, and they are available for Windows, Macintosh and Linux based computers. You can follow through the printed instructions below, or you can use one of my Video Walkthroughs that you can find here:

<https://www.begintocodewithjavascript.com/media>

### Programmer's Points

### Git and Visual Studio Code are professional tools

When you learned to ride a bike you probably had one with training wheels. And people learning to drive a car usually start in something small and easy to handle. You are learning programming with the tools that professionals work with. This is a bit like learning to drive using a Formula 1 racing car. However, there is nothing to worry about here. A Formula 1 car might look a bit scary, but it still has a steering wheel and the usual set of pedals. You don't have to drive it fast if you don't want to and the consequences of a crash are much less.

GitHub and Visual Studio code have a huge range of features, but you don't have to use them. Just like there are buttons on my car dashboard that I don't press because I'm not sure what they do, you don't have to know about every feature of these tools to make good use of them.

It is very sensible to start developing with "proper" tools as recruiters are often as interested in the tools that you are familiar with as they are with the programming languages that you can work with.

## Getting Git

The source code of programs that you write is stored on your computer as files of text. You work on your programs

by changing the contents of these files. When I was starting out programming, I learned very quickly that you can go backwards as well as forwards when writing software. Sometimes I would spend a lot of time making changes to my programs that would turn out to be a bad idea and I would have to go back and undo them all. I solved this problem by making copies of my program code before I did any major edits. That way if anything went bad, I could go back to my original files.

Lots of other programmers noticed this problem too. They also noticed that if you release a program to users it is very useful to have a “snapshot” of that code so that you can keep track of any changes that you make. The best programmers are great at being “intelligently lazy” and so they created software to manage this. One of the most popular programs is called Git.

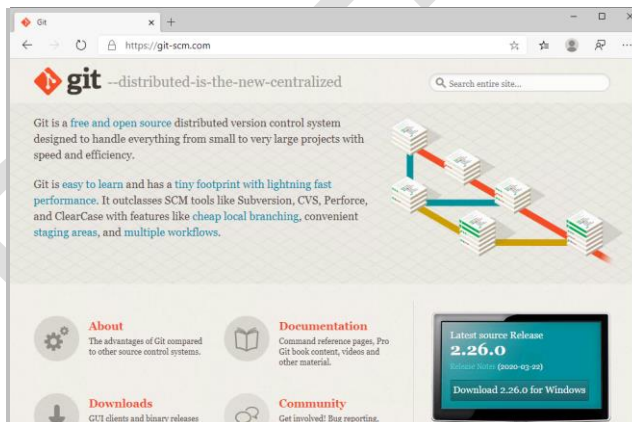
Git was created in 2005 by Linus Torvalds who was writing the Linux operating system at the time. He needed a tool that could track what he was doing and make it easy for him to work with other people. So he created his own. Git is a professional tool and very powerful. It lets large numbers of developers work together on a single project. Different teams can work on their own versions of the code which can then be merged. There is no need for you to use all these powerful features though. You’re just going to use Git to keep track of our work and as a way of obtaining the example programs.

## Make Something Happen

### Install Git

I’m going to give you instructions for Windows 10. The instructions for macOS are very similar. First you need to open your browser and visit the web page:

<https://git-scm.com>



2.1 Ch01\_inset02\_01 Git Install page

Follow the installation process selecting all the defaults.

# Getting Visual Studio Code

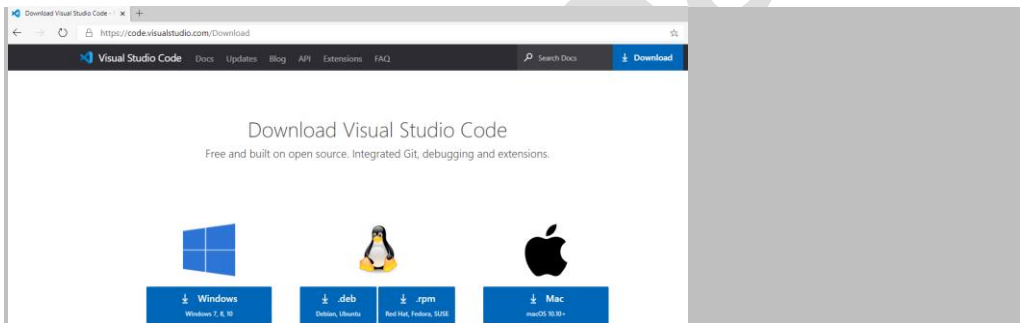
If you want to write a letter you would use a Word processor. To perform calculations, you might use a spreadsheet. Visual Studio Code is a tool that you can use to edit your program files. It can do a lot more than this, as we shall see later. But for now, we are going to use it as a super powerful program editor. Visual Studio Code is free.

Make Something Happen

## Install Visual Studio Code

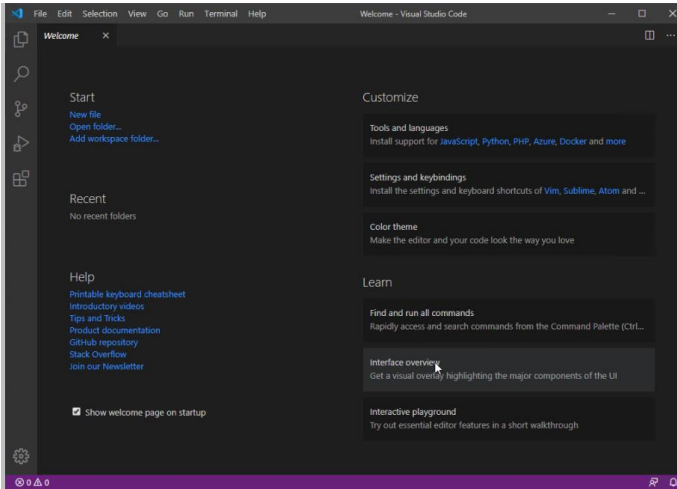
I'm going to give you instructions for Windows 10. The instructions for macOS are very similar. First you need to open your browser and visit the web page:

<https://code.visualstudio.com/Download>



3.1 Ch01\_inset03\_01 Visual Studio Code download page

Click the version of Visual Studio Code that you want and follow the instructions to install it. Once it is installed you will see the start page.



### 3.2 Ch01\_inset03\_02 Visual Studio Code start page

Now that you have Visual Studio installed the next thing you need to do is fetch the sample files to work on.

## Getting the Sample Files

The sample programs, along with a lot of other stuff, are stored on *GitHub*. GitHub is a service that is underpinned by the Git system. You can store your own files on GitHub (and not just programs). You can also use GitHub to host web pages containing JavaScript programs that you create. This makes programs that you write accessible by anyone in the world. To do all this you will need to create a GitHub username and download some software onto your computer. We will create your username later. For now, we are going to just download the sample repository and edit `hello.html`, the file that we worked with at the start of this chapter.

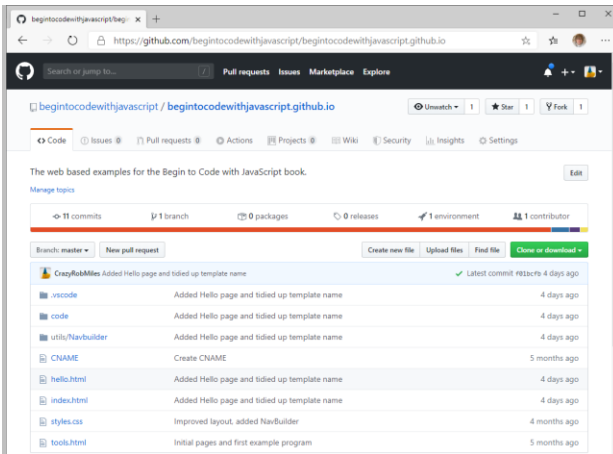
### Make Something Happen

## Clone the sample repository

A *repository* in Git is a collection of files. Whenever I start working on something new I create a repository to hold all the files I'm going to create. I've got a private repository that contains all the text of this book. And I've made a public repository to hold the sample files. Repositories on GitHub can be accessed directly from the browser. The sample files for this book are at the repository with this url (uniform resource locator):

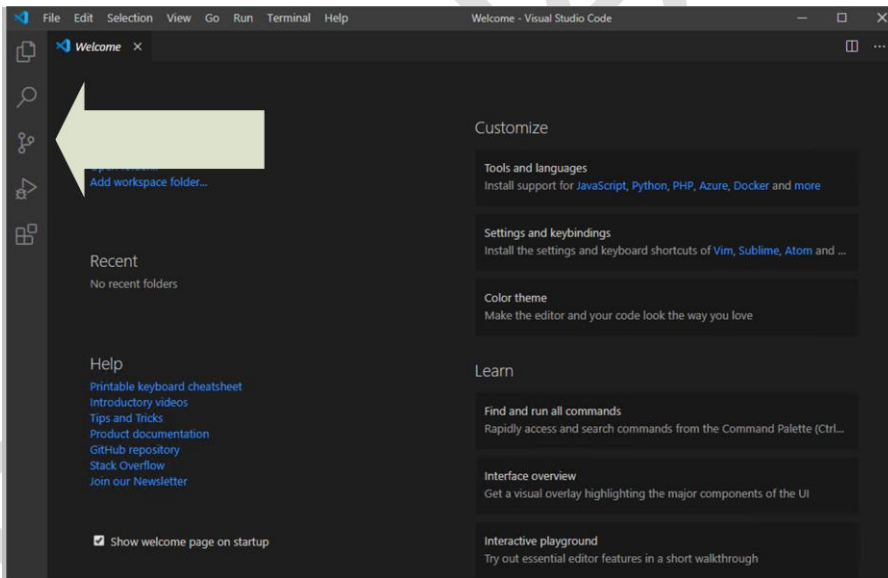
<https://github.com/begintocodewithjavascript/begintocodewithjavascript.github.io>

If you visit this url with your browser you will find that you can navigate all the files, including the file "hello.html" that we investigated earlier and take a look at what is inside them.



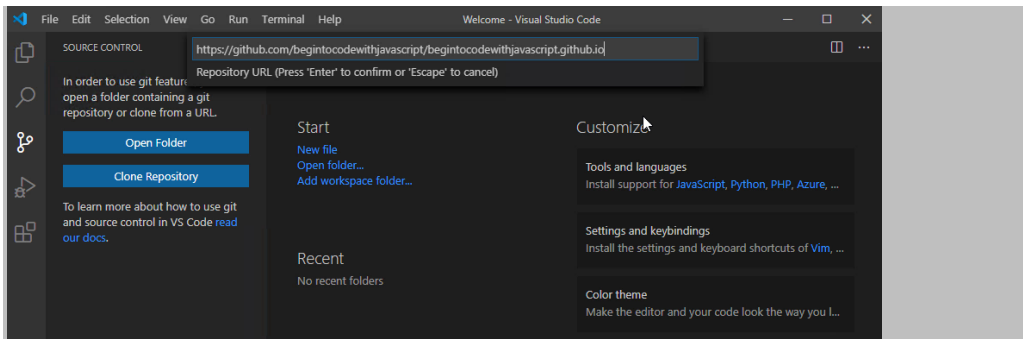
#### 4.1 Ch01\_inset04\_01 Repository home page

You can see that GitHub is keeping track of the changes that I have made to the example programs. We are going to use Visual Studio Code to clone this repository.



#### 4.2 Ch01\_inset04\_02 Visual Studio Code Source Control Button

Start Visual Studio Code and click the Source Control button as shown on the figure above. This opens the Source Control dialog as shown below. Next click the Clone Repository button to begin the process of fetching a repository from GitHub.

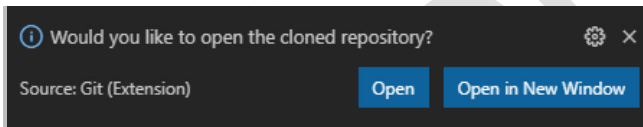


#### 4.3 Ch01\_inset04\_03 Visual Studio Code Source Clone Repository

Visual Studio code is going to download the contents of the repository and store them on your machine. Enter the url of the repository in the dialog that appears. The url you want to use is:

<https://github.com/begintocodewithjavascript/begintocodewithjavascript.github.io>

When you press enter at the end of the url you will be asked where on your computer you want to put the files that are about to be copied. I suggest that you create a folder called GitHub in your “documents” folder and use that, but you can put the repository anywhere you like. Once you’ve selected the folder Visual Studio will copy all the files in the repository from the GitHub site onto your computer.



#### 4.4 Ch01\_inset04\_04 Visual Studio Code Repository Clone Complete

When all the files have been copied Visual Studio Code will ask if you want to open the repository. Click Open to open it.

Congratulations, you have cloned your first repository! Later in the text you will discover how to create your own repositories to store your programs. Remember that you can use GitHub to store anything that you might want to work on, not just program files. If you have an assignment to write you could create a repository to hold the documents and images. This would be an even better idea if you were working on the assignment with other people as GitHub is a great collaboration tool.

## Working on files with Visual Studio Code

We can round off this chapter by working the JavaScript program that we saw at the very start. The process we are going to follow will look like this:

1. Edit the program in the HTML file.
2. Save the file back to disk.

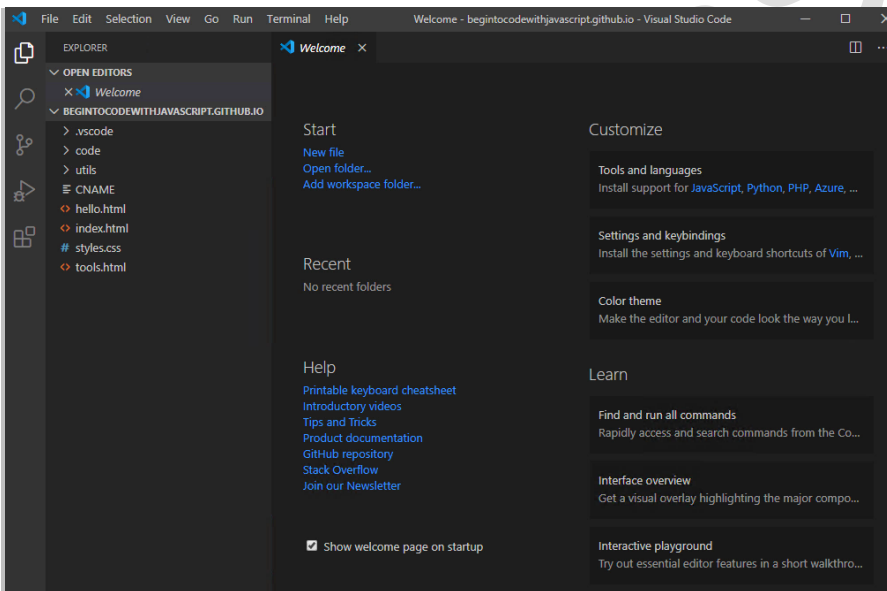
3. Use a web browser to view the HTML file and see what it does.

This is the process you will be using for a lot of the rest of the book. In the next chapter you will discover how to make your programs public so that anyone in the world can view them.

## Make Something Happen

### Edit the secret program

At the end of the last session you opened the example repository that you'd downloaded from GitHub. Now you get to edit the `hello.html` file that we saw contains the secret program.



#### 5.1 Ch01\_inset05\_01 Visual Studio Code Example Repository

The Explorer window at the left hand side of the Visual Studio Code window provides a view of all the files and folders in the repository. You can click the “>” in front of folders in the Explorer view to open them and view their contents. For now, you are just going to look in the `hello.html` file, so click the filename `hello.html` in the Explorer to open it.

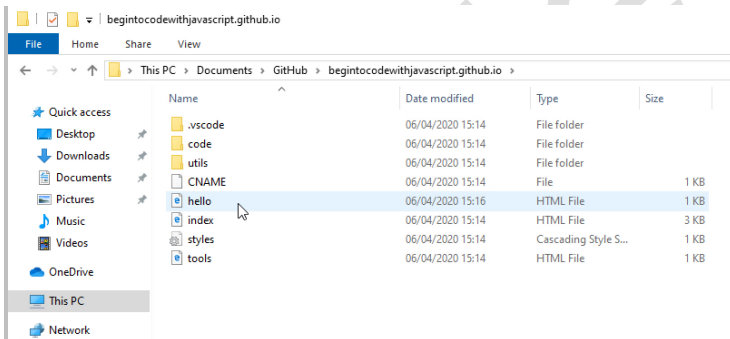


```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <link rel="stylesheet" href="styles.css">
5 </head>
6 <body>
7 <h1>
8   <script>
9     function addition(no1Number, no2Number) {
10      let result = no1Number + no2Number;
11      alert("result: " + result);
12    }
13 </script>
14 <!--begin to code with JavaScript:!!!-->
15 <!--Hello from Rob!!!-->
16 <!--welcome to begin to code with JavaScript.c!!-->
17 <!--this page has hidden secrets!!!-->
18 <!--Press function key F12 to investigate.c!!-->
19 </body>
20 </html>
```

## 5.2 Ch01\_inset05\_02 Visual Studio Code Editing hello.html

Once the page has opened you can make some changes to the text in the file. I've changed one heading so that it says "Hello from Rob". You can save the file by holding down the control key and pressing S (CTRL+S). Or you could use the Save command. Either way, you now want to view the changed file in a browser to see if the changes have worked.

When you cloned the repository, you told Visual Studio Code where to put the files, so now is the time to open File Explorer and navigate to that folder. If you've forgotten where you put the files you can find out just by resting your mouse pointer over the filename in Explorer. Visual Studio Code will then show you the path to that file.



## 5.3 Ch01\_inset05\_03 Finding the hello.html file

If we double-click this file it will be opened by the browser.



Ch01\_inset05\_04 Browsing the hello.html file

Now you will see the file in all its edited glory. Note that the address being browsed is now a file on your local storage, rather than on the web. Note also that you can press F12 if you like and view the contents of the file just like we did at the start of this chapter.

# What you have learned

You might feel that you've spent a lot of this chapter just following instructions, but actually you've learned rather a lot. You've discovered that JavaScript is a programming language, providing a means by which you can tell a computer how to do something. You've had a conversation with JavaScript itself. You've learned that looking after the source files of your programs is important, although great programmers sometimes think of very silly names (for example "git") for their programs sometimes. You've installed the git system and your program editor, Visual Studio Code. Finally, you've copied all the example code onto your machine by "cloning" the repository held on GitHub and even managed to edit one file and view the effects in your browser.

To reinforce your understanding of this chapter, you might want to consider the following "profound questions" about JavaScript, computers, programs, and programming.

What does the word "script" mean in the name JavaScript?

The word "script" in the name refers to the way that JavaScript programs were intended to run. The browser would read each JavaScript statement and then perform it; just like an actor would act out the script of a play. This is not how all programming languages work. Some programming languages are designed to be *compiled*. This means that the source code of the program is converted into the low-level instructions that are run by the computer hardware. These low-level instructions are then directly obeyed by the hardware to make the program run.

Compiled languages run faster than scripts because when the compiled program runs the computer doesn't have to put any effort into working out what the program source is doing, it can just obey the low-level instructions. However, you need to make a different version of the compiled code for each different type of computer. For example, a compiled file for a Windows PC would not run on a Raspberry Pi.

JavaScript was intended to perform simple tasks inside a browser, so it was created as a scripting language. However, it has now become so popular that modern browsers compile JavaScript before running it so that it runs as quickly as possible.

Does my JavaScript programs run on the Web Server?

No. The job of a web server is just to serve up files. The browser (the program running on the user's computer) is responsible for actually creating the display of a web page and running any JavaScript programs in that page.

Do JavaScript programs run at the same speed on all computers?

No. The faster the host computer, the faster the browser (and the JavaScript programs it is hosting) will run.

Do JavaScript programs run faster if I have a faster network connection?

No. A faster network connection will improve the speed at which the JavaScript programs will be loaded into the browser, but the actual speed the JavaScript program runs is determined by the speed of the host computer. Having said that, if the JavaScript program uses the host computer network connection these actions will of course happen more quickly.

Can we view the JavaScript programs in every page we visit?

Yes, you can. The F12 trick (pressing F12 when viewing a web page in a browser) will opens the development view of the page. You can use this to view the JavaScript source code in the page. If you are concerned about someone copying the JavaScript code you can use a tool called an *obfuscator* which is a piece of software that changes the appearance (but not the behavior) of a program so that it is very hard to understand. Take a look at <https://www.javascriptobfuscator.com/> for more details.

How big can a JavaScript program be?

A JavaScript program can be very large indeed. Modern web browsers are very good at handling large programs and the speed of modern networks means that the code can be downloaded very quickly. Some people have even created complete computer emulations in JavaScript that you can run in a browser.

Can you run JavaScript outside a web browser?

Yes you can. Some web pages can be converted into applications which then run on the local computer. There are also ways in which a computer can be made to host JavaScript applications in the same way that a browser does. We will look at these later in the text.

Why is “Git” called “Git”?

This is probably the hardest question in this book. In the UK the word “git” is a form of mild abuse. You would call someone a git if they spilled your drink on purpose. It seems that Linus Torvalds called his first version of the program “His stupid content tracker” and then hit upon the word git as a shorter version of this.

Can I do private work on GitHub?

Yes. GitHub is very popular with programmers who are working on Open Source projects, but you can also make a GitHub repository private so that only you can see it. If you use the free subscription the number of private repositories you can create is limited, as is the number of people you can work with on a shared project.

What do I do if I “break” my program?

Some people worry that things they do with a program on the computer might “break” it in some way. I used to worry about this too, but I've conquered this fear by making sure that whenever I do something, I always have a way back. Git and GitHub are very useful in this respect. Later in the text we will discover how to use GitHub to take “snapshots” of projects which we can return to if we break our program. We can also use the Git desktop program to search for changes that we have made to the files in a project.

Why is the Visual Studio Code display of the hello.html file in different colors?

This is called *source code highlighting*. Visual Studio Code has a list of words that are “special” as far as JavaScript and HTML are concerned. These special words are called *keywords*. For each keyword Visual Studio has a characteristic color, in the case of Visual Studio Code keywords are displayed in blue, functions are displayed in yellow, strings of text are orange and everything else is white. The intention is to make it easier for programmers to understand the structure of the program. Note that there is nothing in the program file that specifies the color of each element, this is something that Visual Studio Code does.

Will “artificial intelligence” mean that one day we won’t have to write programs?

This is a very deep question. To me, artificial intelligence is a field where lots of people are working very hard to make a computer really good at guessing. It turns out that by giving computers lots of information, and telling them how the information is related, a program can then use all this stuff to make a pretty good guess as to the context of a statement.

I suppose that all humans do is “guess” at the meaning of things. Maybe one day a doctor really will want me to drink a hot bath before I take my medicine (see the instructions above), in which case I’ll do the wrong thing. However, humans have a much greater capacity to store experiences and link them together, which puts the computer at a distinct disadvantage when it comes to showing intelligence. Maybe in time this will change. We are already seeing that in specific fields of expertise, for example finance and medical diagnosis, artificial intelligence can do very well.

However, in my opinion, when it comes to telling the computer exactly what we want them to do, we’ll be needing programmers for quite a long time. Certainly, long enough for you to pay off your mortgage.