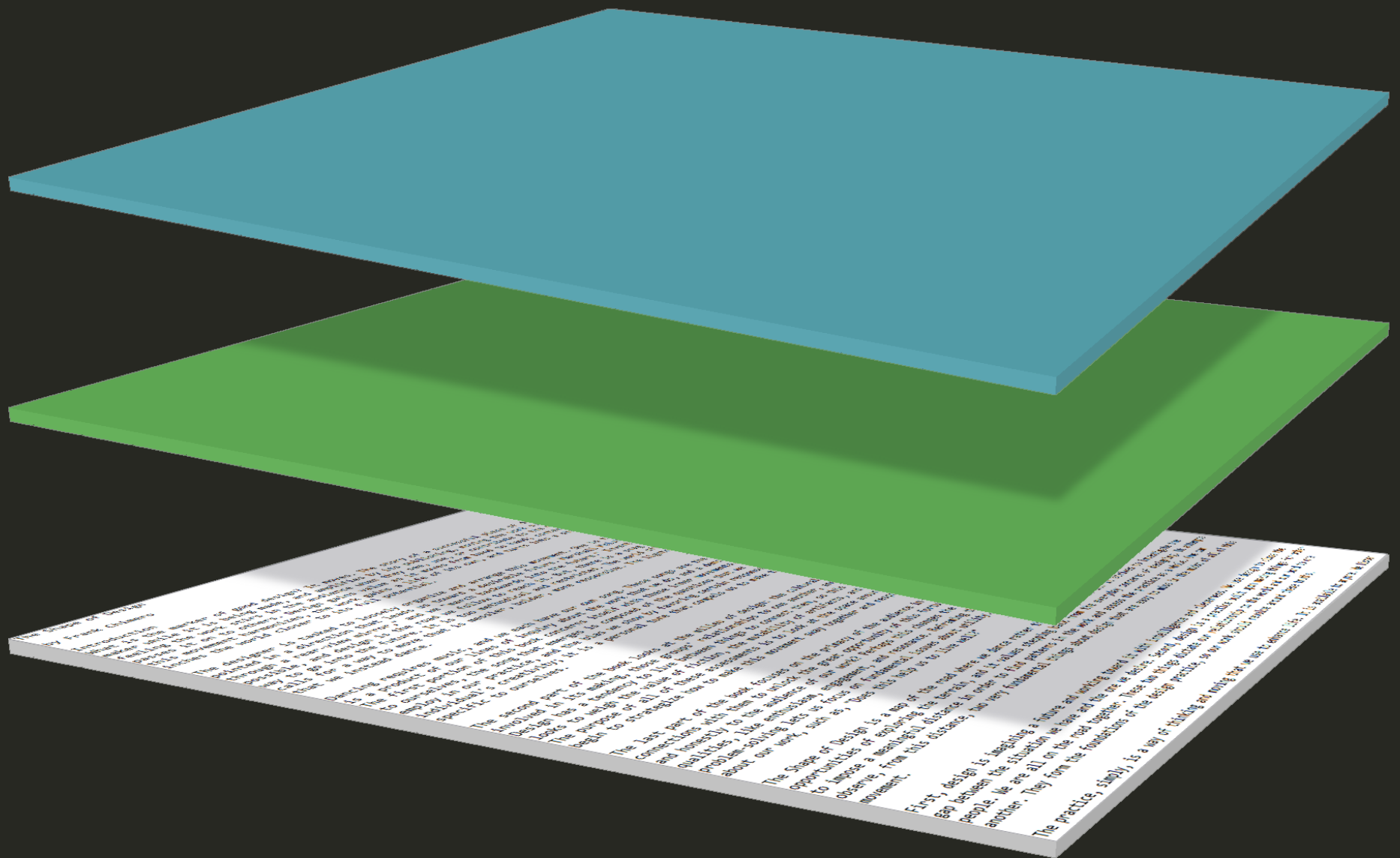
The behaviour layer
JavaScript: Part 1

Content Management



Presentation

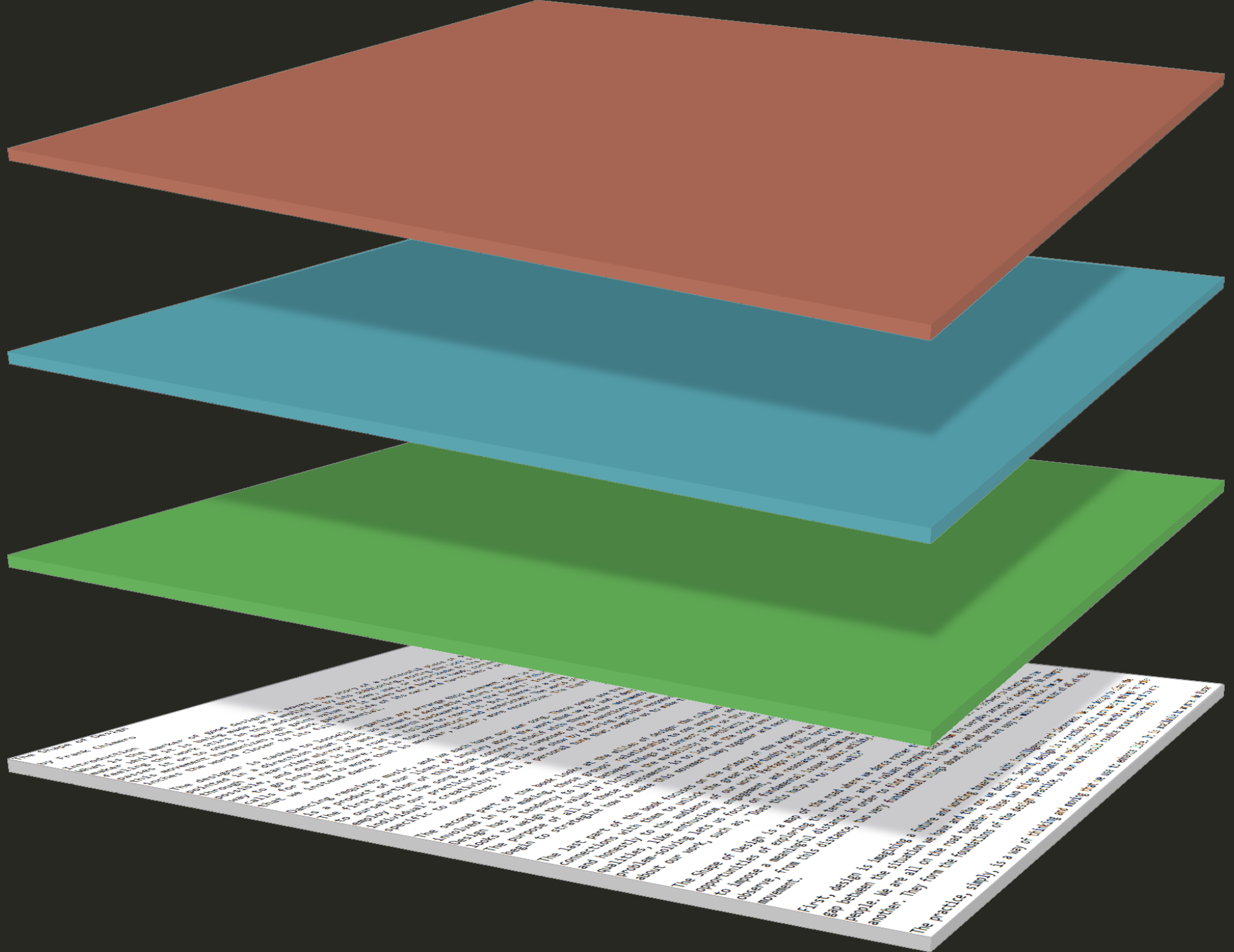
CSS

Structure

HTML

Content

Text, images etc.



Behaviour

JavaScript

Presentation

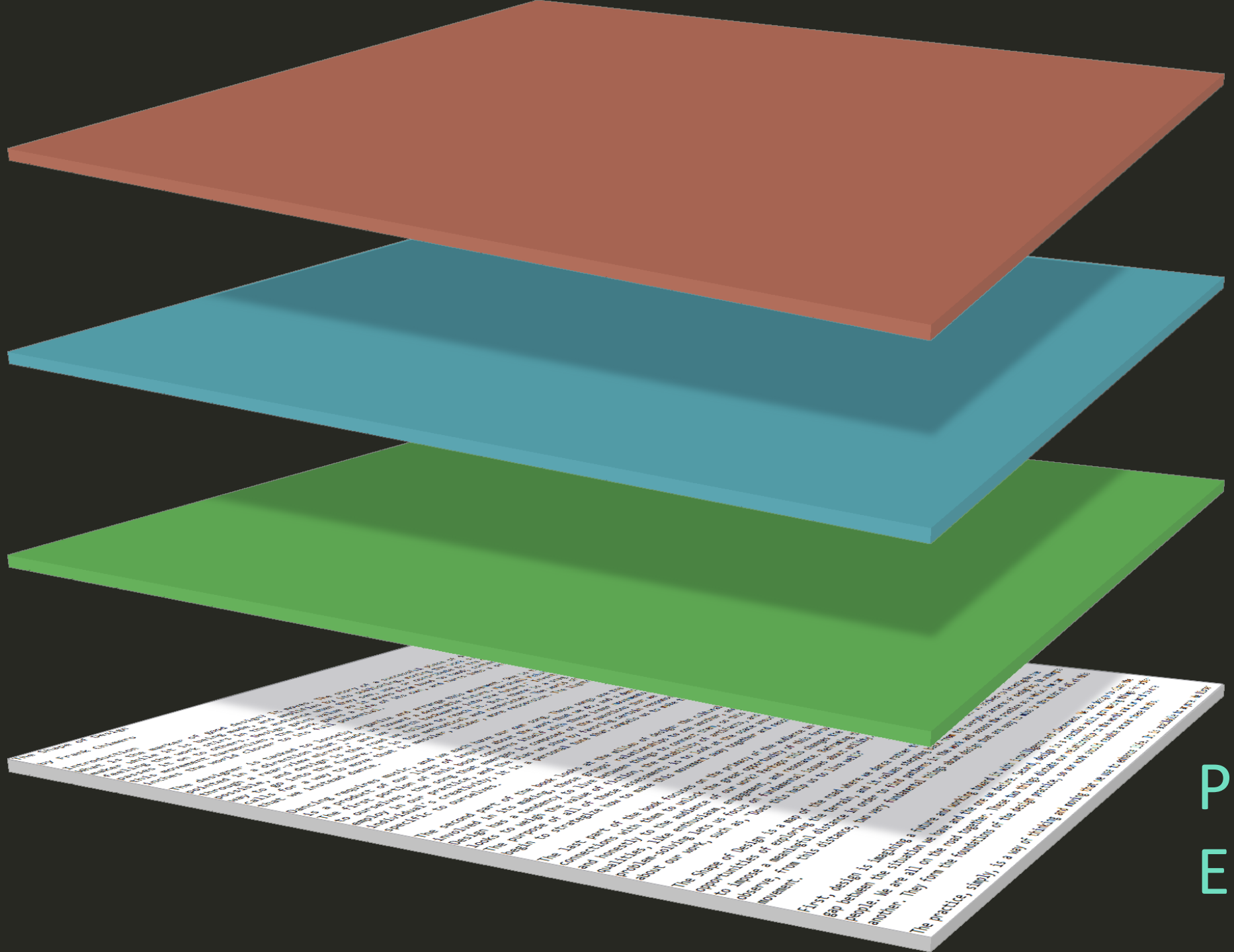
CSS

Structure

HTML

Content

Text, images etc.



Richness of experience

Progressive Enhancement

All three layers of the web standards model combined…

structure | presentation | behaviour | webpage in browser

The browser loads the HTML file and then any linked resources such as the CSS file, the JavaScript file and any media files (e.g. images). A single webpage is built from many separate files.

The purpose of JavaScript is to add…
Behaviours to the user interface

```
var newLink = document.createElement( "a" );
var allParagraphs = document.getElementsByTagName( "p" );
var moreParagraph = allParagraphs[ 1 ];
newLink.setAttribute( "href", "#" );
newLink.setAttribute( "class", "more-link" );
newLink.innerHTML = "Read more";
moreParagraph.appendChild( newLink );
```

It does this with scripts that change how the content looks or behaves in the browser depending on events. For example, JavaScript can be used to change CSS values or add new CSS properties to HTML elements.

What is a script?

```
<script>
1. Do something;
2. Do something else;
3. Do something with the results of 1 and 2;
4. Print the output of 3;
</script>
```

A script is a logical sequence of *statements*. Each statement tells the browser to do something, so that, step-by-step, the required outcome is achieved.

Hands on Javascript!

Head over to Moodle and download the folder entitled “JavaScript Greeting – coding exercise”.

What does this script do?

```
<script>
var dateTime = new Date( );
var hours = dateTime.getHours( );
var minutes = dateTime.getMinutes( );
document.write ("<p>The time is: " + hours + ":"
+ minutes + "</p>");
</script>
```

index.html

JavaScript can be placed in the HTML file but must be enclosed by script tags. However, this is not ideal.

Tip: `var` is short for “variable”

Think of a variable as being a small container that is given a name and is used to store information. You may also see the `let` and `const` keywords used to declare variables.

Separating JS from HTML

```
<script src="script.js"></script>
```

index.html

The script tags must be placed in the HTML, where you want the output to appear.

```
var dateTime = new Date( );
var hours = dateTime.getHours( );
var minutes = dateTime.getMinutes( );
document.write ("<p>The time is: " + hours
+ ":" + minutes + "</p>");
```

script.js

The script is placed in an external file and linked to the HTML file in a script tag.

An even better solution

```
<p id="time">Time goes here</p>
<script src="script.js"></script>
```

index.html

The script tags should be placed immediately before the closing body tag and the paragraph goes wherever you want the output to appear.

```
var dateTime = new Date();
var hours = dateTime.getHours();
var minutes = dateTime.getMinutes();
var msg = "The time is: " + hours + ":" + minutes;
var el = document.querySelector('#time');
el.textContent = msg;
```

script.js

The script is updated so that it changes an existing element rather than creating a new one.

Assigning elements to variables

```
<p>Some content…</p>
var paragraph = document.getElementsByTagName('p')[0];
```

Result = assigns the first paragraph in the current document to a variable called “paragraph”.

```
<p id="time">JavaScript not available</p>
var time = document.getElementById('time');
```

Result = assigns the paragraph with ID “time” to a variable called “time”.

```
<body>… </body>
var body = document.querySelector('body');
```

Result = assigns the body element to a variable called “body”.

The DOM provides JavaScript with several methods for accessing elements within the HTML document such as *getElementsByTagName, getElementById* and *querySelector*. Once elements have been selected, JavaScript can be used to manipulate them in various ways.

Manipulating HTML elements

```
<p>Some content</p>
```

```
var paragraph = document.getElementsByTagName('p')[0];
paragraph.innerHTML = "Some replacement content";
```

Result = replaces the existing content with new content for the selected element.

```
<p>Some replacement content</p>
```

JavaScript being used to change the content of a HTML document, using the *innerHTML* method.

```
<body>… </body>
```

```
var body = document.querySelector('body');
body.setAttribute ('class', 'page1');
```

Result = adds the class “page1” to the body element.

```
<body class="page1">… </body>
```

JavaScript being used to change the styling of a HTML document, using the *setAttribute* method.

JavaScript: context

- Originally developed by Netscape with Sun Microsystems is 1995
- Microsoft followed with Jscript
- JavaScript now standardised by ECMA* (not W3C)
- Got a bad name because of misuse by some designers
- Has now been rehabilitated and used with the DOM and other web standards. This is sometimes referred to as “DOM scripting”.
- ES6 (introduced in 2020) in the most recent version.

* European Computer Manufacturers Association

The nature of JavaScript

- JavaScript is a “client-side” scripting language.
- That means it runs in the browser on the client's own device (phone, tablet, laptop).
- Scripts will therefore only work if the device supports JavaScript and the user has it turned on.
- JavaScript should not be used for essential functions (such as navigation) unless there is a fall-back that works if JavaScript does not.
- JavaScript is used to *progressively enhance* web pages.

JavaScript and me

- In the past, web designers could get away with just using HTML and CSS, leaving JavaScript to the “geeks”.
- However, most front-end developers are now expected to have at least a rudimentary understanding of JavaScript. Fortunately, JavaScript has become much easier to use.
- Many developers also use JavaScript frameworks/libraries like jQuery to make life easier but developing a good understanding of “vanilla” JavaScript should be your objective.

Where does it go?

- Just like CSS, scripts can be embedded in your HTML document, or they can be placed in an external file and linked to from the HTML document.
- Where scripts will be used by many pages, it makes sense to store them in external files but if they are unique to a page, it may be better to include them in the HTML document, thereby saving an *http request* (more efficient).
- If scripts are added to the HTML file, they are usually best located just before the closing </body> tag because the browser will have completed the DOM (Document Object Model) at that point.

Adding Scripts

```
<script>
  JavaScript goes here
</script>
```

Embedded scripts are enclosed in a <script> tag.

```
<script src="script.js"></script>
```

Linked scripts also use the <script> tag but a src (source) attribute is added to point to the file. Usually, the file has a .js extension. Notice that this forms an empty element – there's nothing between the opening and closing tags.

Note: in HTML5, the type="text/javascript" attribute is not required but it is required for XHTML.

Hello world!

A very simple script:

```
<script>
// A very simple script that prints "Hello World!"
document.write("<h1>Hello World!</h1>");
</script>
```

Example

The above example uses a single statement to write some markup to the document. The script is placed between opening and closing script tags.

The text in quotes (including the html tags) is written to the document by JavaScript using the *document.write* method.

Notice that the script includes a comment, which describes what the statement does. Single-line comments begin with a double slash "//".

Getting the time and printing it

A more useful script:

```
<script>
// This script will print the time
var dateTime = new Date( );
var hours = dateTime.getHours( );
var minutes = dateTime.getMinutes( );
document.write ("<p>The time is: " + hours +
":" + minutes + "</p>");
</script>
```

Example

This script uses four statements to write the current time to the document. The script uses the Date() object and then the *getHours* and *getMinutes* methods to assign those values to the variables “hours” and “minutes”.

The text in quotes is added to the variable values for hours and minutes before being written to the document by JavaScript using the *document.write* method.

Day of the week

Not so simple:

```
<script>
// Sometimes, doing simple things in JavaScript is more complicated
// This script gets the name of the day of the week and prints it
var dateTime = new Date();
var dayNumber = dateTime.getDay();
var weekday = new Array(7);
weekday[0]="Sunday";
weekday[1]="Monday";
weekday[2]="Tuesday";
weekday[3]="Wednesday";
weekday[4]="Thursday";
weekday[5]="Friday";
weekday[6]="Saturday";
document.write("<p>Today is " + weekday[dayNumber] + "</p>");
</script>
```

Example

We need to build an array of day names because JavaScript (unlike PHP) only knows the day number. Arrays always begin with 0, so the week runs from day 0 to day 6.

Testing for conditions

```
<script>
// The following code prints something different on a Wednesday
var dateTime = new Date();
var n = dateTime.getDay();
var weekday = new Array(7);
weekday[0]="Sunday";
weekday[1]="Monday";
weekday[2]="Tuesday";
weekday[3]="Wednesday";
weekday[4]="Thursday";
weekday[5]="Friday";
weekday[6]="Saturday";
if (weekday[n]==="Wednesday"){
        document.write("<p>Hurrah! Today is " + weekday[n] + ", it's a Greenwich
day.</p>");
}else{
        document.write("<p>Today is " + weekday[n] + ", just an ordinary day.</p>");
}
</script>
```

Example

This script uses an *if else* clause to decide what message should be printed. If today is Wednesday, a special message is printed otherwise (else) a standard message is printed. Using this technique, web pages can be dynamic, with content that changes depending on time.

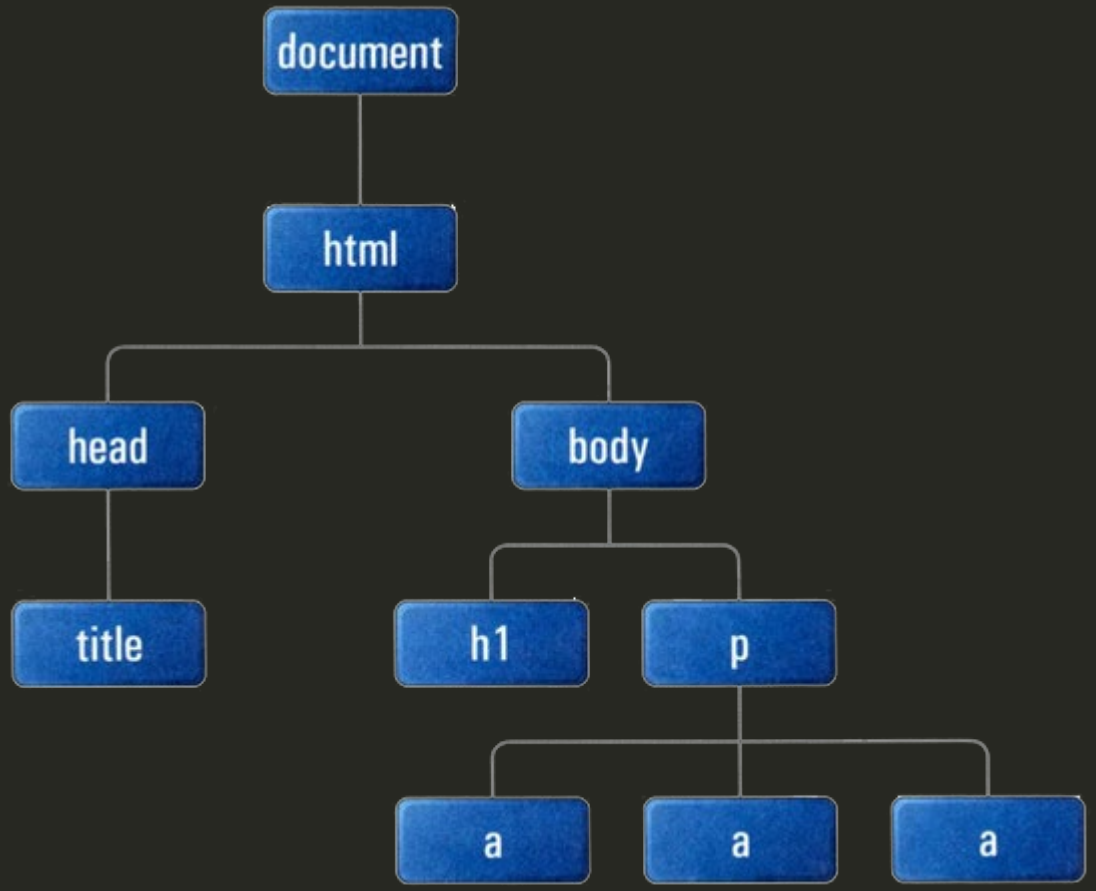
Objects, Properties & Methods

- Modern JavaScript supports Object Oriented Programming (OOP).
- OOP envisions software as a group of co-operating objects.
- Each object (e.g. the Date object) has properties and methods (e.g. the getHours method).
- A property is a value (say the number 5).
- A method is an action that the object can perform (e.g. tell you how many properties an object has).
- Almost all the scripting you do will use this model.
- A web page is just a bunch of objects…

DOM

WHAT'S THIS DOCUMENT OBJECT MODEL THING?

The Document Object Model



The DOM is jointly controlled by the W3C and WHATWG and is a standard method for browsers and scripting languages to access page elements. Effectively, the DOM is a map of the markup in an HTML file.

Each object in the DOM is known as a *node*. This diagram shows only the element nodes. In reality, the DOM also maps any attributes and the content within elements.

You probably didn't realise it, but you use the DOM every time you use a CSS selector. The browser uses the DOM to locate elements you select in your CSS.

Taken from: “Simply JavaScript” by Kevin Yank and Cameron Adams

What is the purpose of the DOM?

The Document Object Model has two purposes:

1. Providing JavaScript with a map of all the elements on our page.
2. Providing us with a set of methods for accessing those elements, their attributes, and their contents.

Taken from: “JavaScript for Web Designers” by Mat Marquis

HTML markup

When a browser receives an HTML file, it must create a “map” of the markup in the form of a hierarchical model before the web page can be rendered.

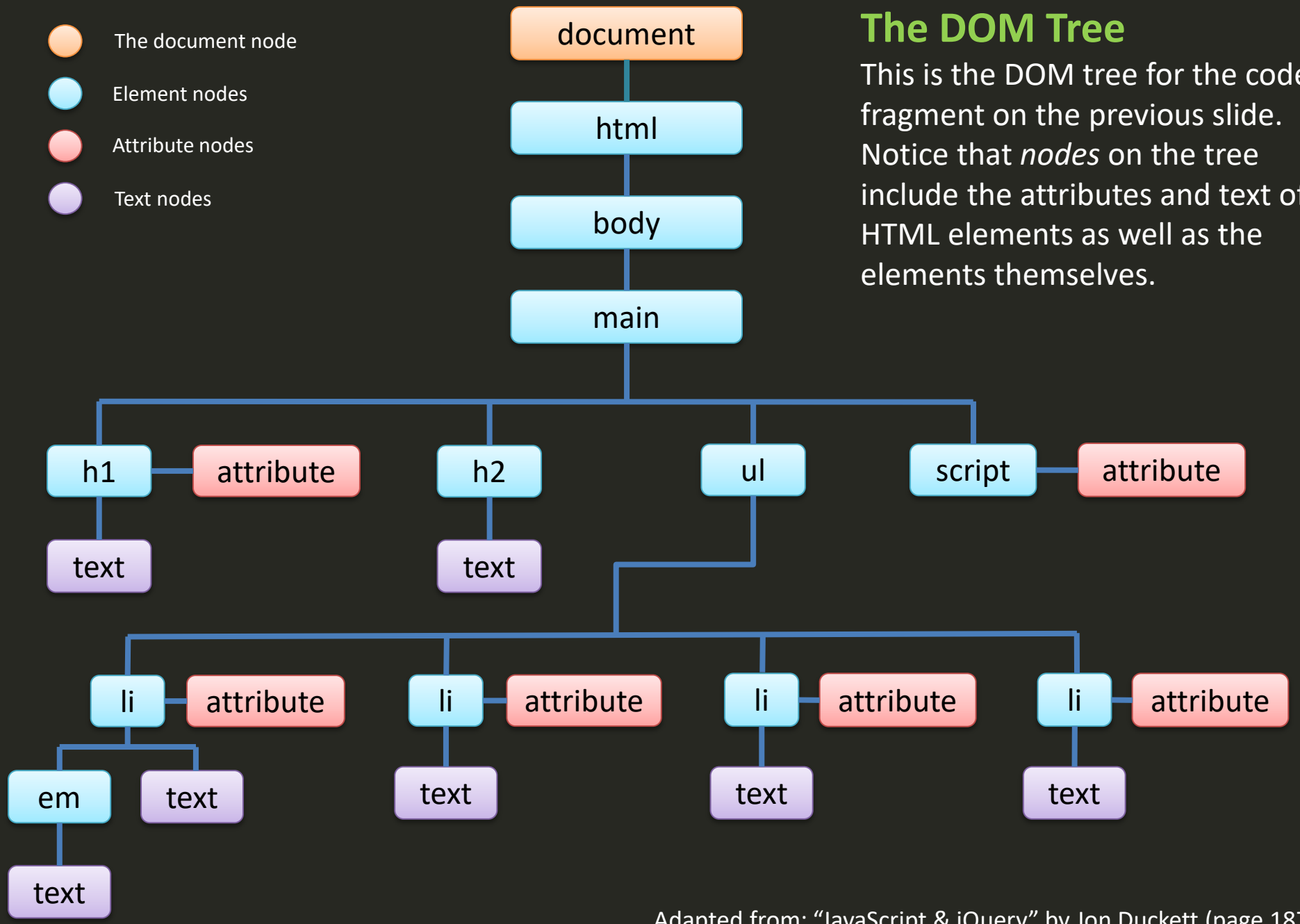
```
<html>
  <body>
    <main>
      <h1 class="page1">List</h1>
      <h2>Buy Groceries</h2>
      <ul>
        <li id="one"><em>fresh</em> figs</li>
        <li id="two">pine nuts</li>
        <li id="three">honey</li>
        <li id="four">balsamic vinegar</li>
      </ul>
    </main>
    <script src="js/list.js"></script>
  </body>
</html>
```

The code fragment above is interpreted by the browser and stored in memory as a *DOM tree*. The next slide shows the DOM tree for this code. To keep things simple, <head> is intentionally omitted.

Adapted from: “JavaScript & jQuery” by Jon Duckett (page 186)

The DOM Tree

This is the DOM tree for the code fragment on the previous slide. Notice that *nodes* on the tree include the attributes and text of HTML elements as well as the elements themselves.



Adapted from: “JavaScript & jQuery” by Jon Duckett (page 187)

Which version of the DOM?

- Just like HTML and CSS, the DOM has evolved through a number of versions.
- The DOM Level 1 was standardised in 1997 by the W3C.
- DOM4, which is part of the HTML5 collection of technologies, is developed jointly by W3C and WHATWG.
- Modern browsers support all the features of DOM3 and much of DOM4 e.g. *document.querySelector()*
- DOM4 reached *recommended* status on 19th November 2015.
- However, DOM is now a “living standard” and, like HTML will continue to evolve without numbered iterations.
- The standard was most recently updated on 19th December 2021.

https://dom.spec.whatwg.org/

DOM Scripting

Most of the work you will do with JavaScript can be described as “DOM Scripting” because it mainly involved manipulating the DOM in various ways. For example, we can use JavaScript to add, remove or change HTML elements by editing the DOM. The great thing is, it’s not as difficult as it sounds.

Most DOM Scripting involves just 3 steps:

1. Select — querySelector
2. Make interactive — addEventListener
3. Change styling — setAttribute

The *querySelector* method is particularly useful for us because it allows HTML elements to be selected using CSS selector syntax. For example:

```
document.querySelector('.content')
```

Will select the first element in the DOM tree with a class of “content”. We could select all elements with a class of “content” using the *querySelectorAll* method.

Introduction to the DOM

Case, White space, Semicolons and Comments

JAVASCRIPT SYNTAX

JavaScript is case sensitive

myNumber and mynumber are not the same

This is important because scripting decisions are often made on the basis of comparing one thing with another.

getElementById

Notice that variable names and methods etc. use **camel case**. This is used to ensure readability while avoiding the use of word separators such as hyphens and underscores.

White space is unimportant*

In JavaScript, this…

```
var newLink = document.createElement( "a" );
var allParagraphs = document.getElementsByTagName( "p" );
var moreParagraph = allParagraphs[ 1 ];
newLink.setAttribute( "href", "#" );
newLink.setAttribute( "class", "more-link" );
newLink.innerHTML = "Read more";
moreParagraph.appendChild( newLink );
```

…is the same as this…

```
var newLink=document.createElement("a");var
allParagraphs=document.getElementsByTagName("p");var
moreParagraph=allParagraphs[1];newLink.setAttribute("href","#");newLi
nk.setAttribute("class","more-link");newLink.innerHTML="Read
more";moreParagraph.appendChild(newLink);
```

*But one is much easier to read than the other. We aim to write one statement per line. This is the same principle as HTML and CSS.

Semicolons end statements

In JavaScript, the semicolon is used to terminate a statement just as semicolons are used in CSS to terminate declarations.

```
var newLink = document.createElement( "a" );
var allParagraphs = document.getElementsByTagName( "p" );
var moreParagraph = allParagraphs[ 1 ];
newLink.setAttribute( "href", "#" );
newLink.setAttribute( "class", "more-link" );
newLink.innerHTML = "Read more";
moreParagraph.appendChild( newLink );
```

Strictly speaking, the semicolons are not required because JavaScript interprets a line break as an end to a statement, but they are normally used – this is a convention (like always using lower-case for HTML elements).

Comments

Comments are really important in JavaScript where they are used to explain what a piece of code is doing. The syntax for a single line comment looks like this:

```
// This part of the script creates a link and adds it to the HTML.
// We want this to happen on page load. Each statement runs, one
// after the other and forms a logical sequence.
var newLink = document.createElement( "a" );
var allParagraphs = document.getElementsByTagName( "p" );
var moreParagraph = allParagraphs[ 1 ];
moreParagraph.appendChild( newLink );
```

Multi-line comments are just like CSS comments and look like this:

```
/* This part of the script creates a link and adds it to the HTML.
We want this to happen on page load. Each statement runs, one
after the other and forms a logical sequence.*/
```

Values, variables and operators

THE BUILDING BLOCKS OF JAVASCRIPT

Values (data types)

Numbers	56
Strings	"letters and words"
Boolean	true *or* false

Operators

+	plus	6+3 = 9
-	minus	6-3 = 3
*	multiply	6*3 = 18
/	divide	6/3 = 2
%	modulo	6%3 = 0

Modulo gives the remainder of one value divided by another (12%5=2).

Variables

```
var myNumber = 36;
```

You can think of variables as being named boxes into which data can be placed. This could be simple data like a number or a string of text, or it could be an array containing many values.

```
var myNumber = 36;
myNumber / 3 === 12;
true
```

Variable names can be used in place of the value they contain, and this is what makes programs dynamic. Two new ways to declare variables were introduced with ES6 in 2020. You may see the *let* and *const* keywords being used.

Working with variables

```
var myNumber = 36;
myNumber = myNumber / 3;
```

In the example above, the value contained in the myNumber variable is changed from 36 to 12 (36 divided by 3 equals 12).

Comparison operators

==	is equal	6+3 == "9"
!=	is not equal to	6-3 != 4
===	is identical to	6*3 === 18
>	Is greater than	9 > 2
>=	Is greater than or equal to	6 >= 3
<	Is less than	2 < 3
<=	Is less than or equal to	4 <= 4

All of the above will return the boolean value *true*.

The distinction between *is equal to* and *is identical to* is important. Two things can be equal, even if the data types are different (e.g. a number 9 and a string "9" are equal). To be identical, their value and data types must both be the same.

What does this script do?

```
var dateTime = new Date();
var day = dateTime.getDay();
var el =  document.getElementById('message');
var msg;
if (day < 4) {
        msg = "Wednesday is " + (3 - day) + " day(s) away.";
}else if (day == 4) {
        msg = "It's Wednesday!";
}else{
        msg = "You'll have to wait until next week for Wednesday.";
}
el.innerHTML = msg;
```

script.js

This script demonstrates the basic form that simple scripts take. The first part gathers the data, the second part processes the data, and the third part does something with the processed data. In this case, we're getting the day number and then deciding what message should be displayed and finally, displaying the message.

JavaScript is power

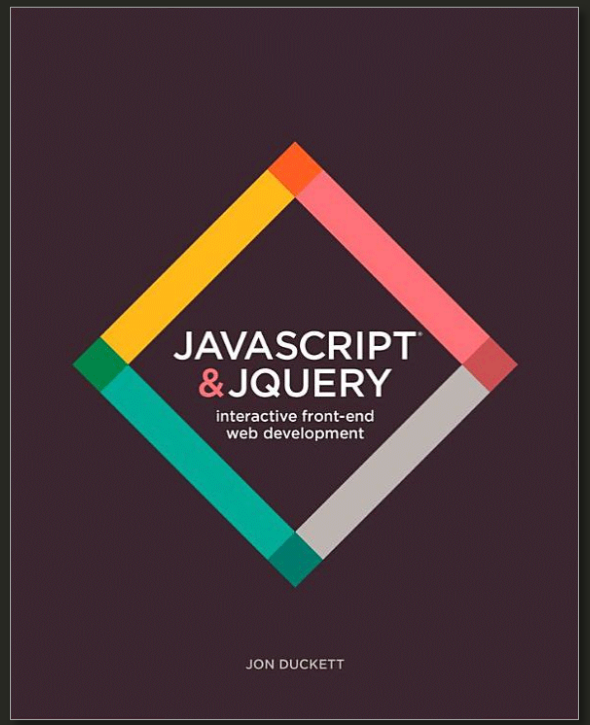
OK, I LIKE JAVASCRIPT, HOW CAN I LEARN MORE?

How can I learn JavaScript?

There are lots of online resources to help you learn JavaScript. However, just like HTML and CSS, JavaScript has changed a lot over the years. Do ensure that you use an up-to-date resource. The Code Academy courses are very good, as are the video tutorials at Pluralsight.

codecademy.com/learn/introduction-to-javascript

Any good beginner books?



If you're a complete beginner and you've never done any programming before, this book is written specially for you. It assumes nothing and is extremely clear. 3 cheers for Jon Duckett!

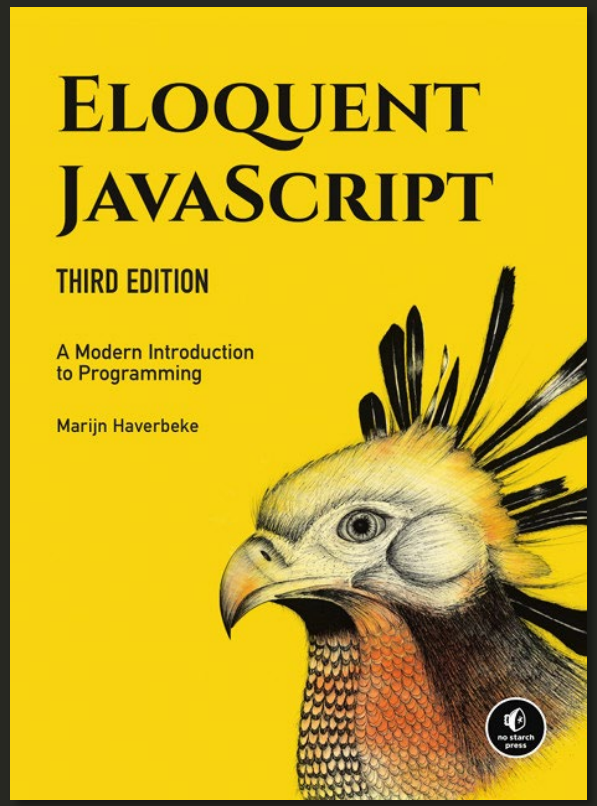
javascriptbook.com

Something I can read on the bus?

Also designed for complete beginners, Mat Marquis' book is an excellent, light read that gently introduces the core concepts of JavaScript and brings them all together in the final chapter with an exercise you can follow.

abookapart.com/products/javascript-for-web-designers

What other resources are there?



The third edition of Eloquent Javascript by Marijn Haverbeke is totally recommended if you want to get to grips with JavaScript and dig a little deeper. An online version of the book is also available for free. Check it out.

eloquentjavascript.net

Start simple

- Don't be put off if you've never done any scripting before or if JavaScript looks complicated.
- Start with the basics and work up from there.
- It takes time and a little frustration, but scripting can be very rewarding and will give you power over your web pages.
- Always follow a reliable reference and work from first principles.

https://developer.mozilla.org/en-US/docs/Web/JavaScript

```
function endSlideshow() {
    el = document.querySelector("slideshow");
    el.style.display = "none";
    return true;
}
```