

1

THE ABC OF
PROGRAMMING

Before you learn how to read and write the JavaScript language itself, you need to become familiar with some key concepts in computer programming. They will be covered in three sections:

A

What is a script and how do I create one?

B

How do computers fit in with the world around them?

C

How do I write a script for a web page?

Once you have learned the basics, the following chapters will show how the JavaScript language can be used to tell browsers what you want them to do.

1/a

WHAT IS A SCRIPT
AND HOW DO I
CREATE ONE?

A SCRIPT IS A SERIES OF INSTRUCTIONS

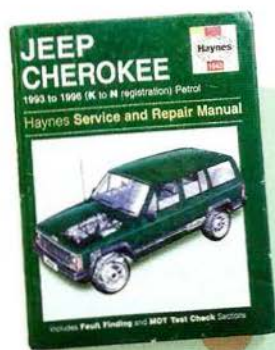
A script is a series of instructions that a computer can follow to achieve a goal. You could compare scripts to any of the following:

RECIPES

By following the instructions in a recipe, one-by-one in the order set out, cooks can create a dish they have never made before.

Some scripts are simple and only deal with one individual scenario, like a simple recipe for a basic dish. Other scripts can perform many tasks, like a recipe for a complicated three-course meal.

Another similarity is that, if you are new to cooking or programming, there is a lot of new terminology to learn.



EMPLOYEE HANDBOOK

HANDBOOKS

Large companies often provide handbooks for new employees that contain procedures to follow in certain situations.

For example, hotel handbooks may contain steps to follow in different scenarios such as when a guest checks in, when a room needs to be tidied, when a fire alarm goes off, and so forth.

In any of these scenarios, the employees need to follow only the steps for that one type of event. (You would not want someone going through every single step in the entire handbook while you were waiting to check in.) Similarly, in a complex script, the browser might use only a subset of the code available at any given time.

MANUALS

Mechanics often refer to car repair manuals when servicing models they are not familiar with. These manuals contain a series of tests to check the key functions of the car are working, along with details of how to fix any issues that arise.

For example, there might be details about how to test the brakes. If they pass this test, the mechanic can then go on to the next test without needing to fix the brakes. But, if they fail, the mechanic will need to follow the instructions to repair them.

The mechanic can then go back and test the brakes again to see if the problem is fixed. If the brakes now pass the test, the mechanic knows they are fixed and can move onto the next test.

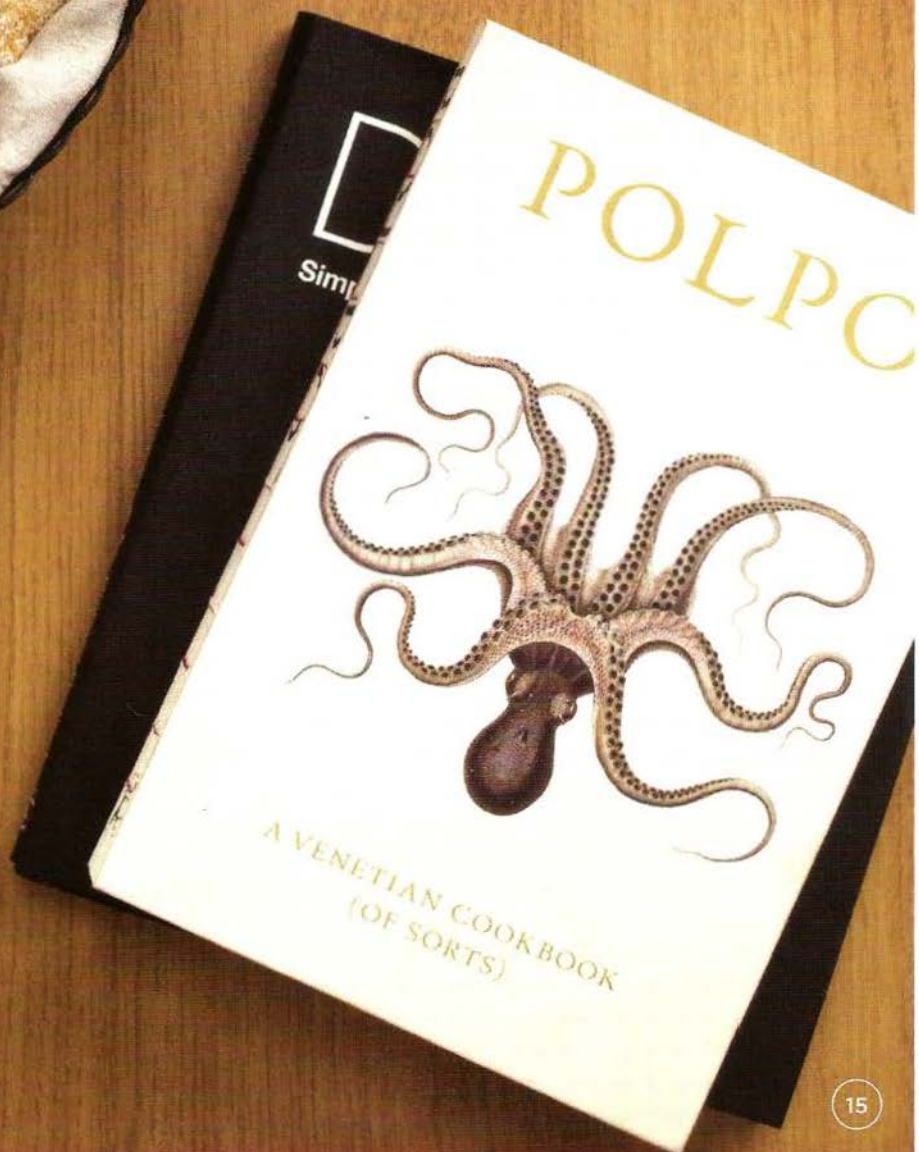
Similarly, scripts can allow the browser to check the current situation and only perform a set of steps if that action is appropriate.



Scripts are made up of instructions a computer can follow step-by-step.

A browser may use different parts of the script depending on how the user interacts with the web page.

Scripts can run different sections of the code in response to the situation around them.



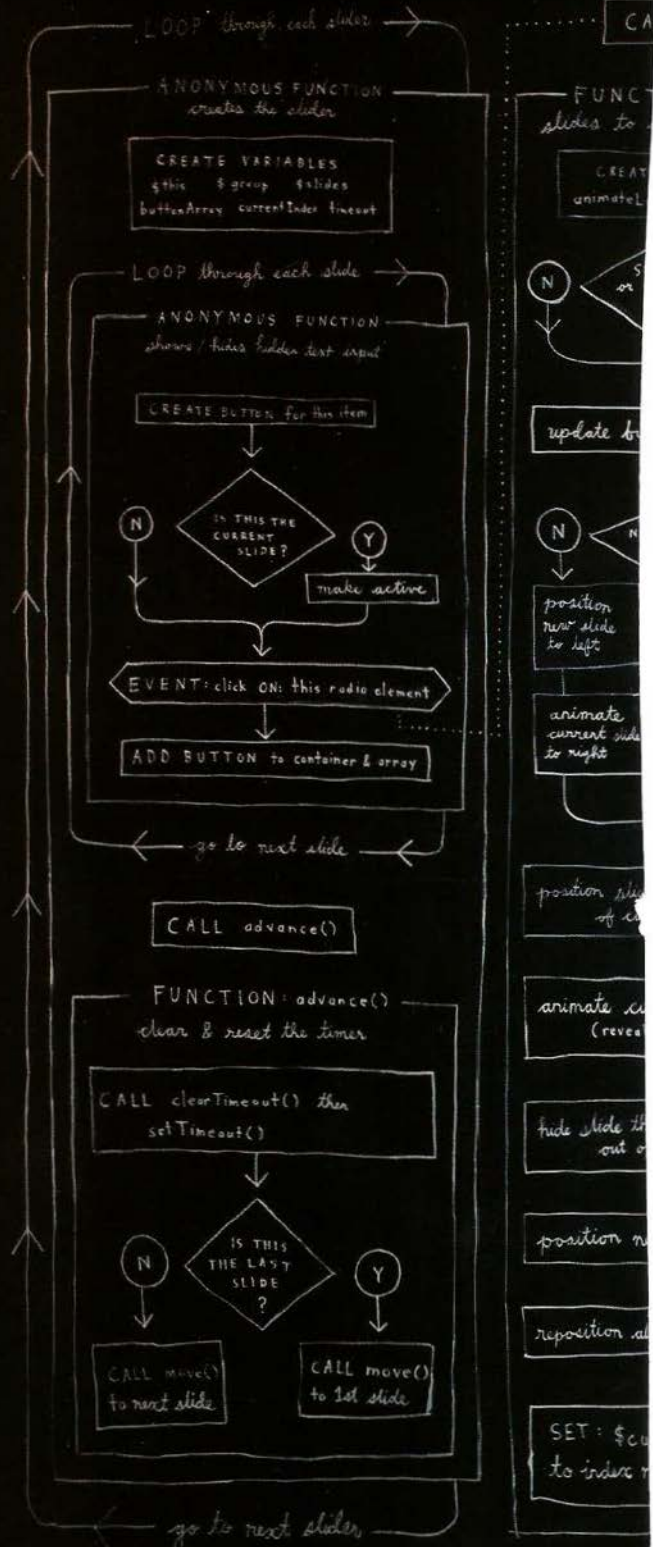
WRITING A SCRIPT

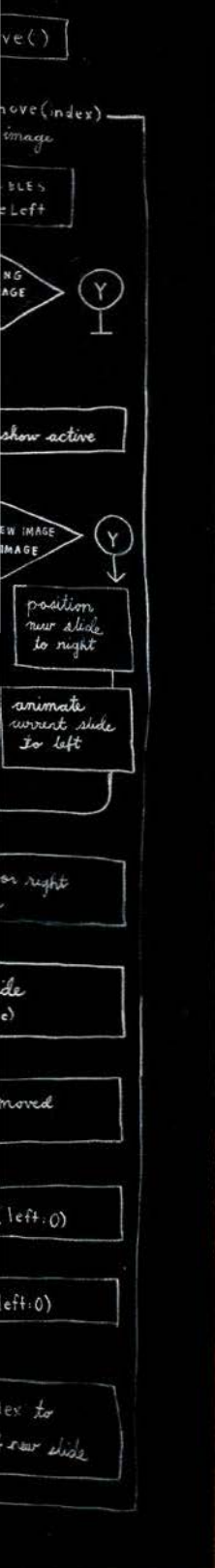
To write a script, you need to first state your goal and then list the tasks that need to be completed in order to achieve it.

Humans can achieve complex goals without thinking about them too much, for example you might be able to drive a car, cook breakfast, or send an email without a set of detailed instructions. But the first time we do these things they can seem daunting. Therefore, when learning a new skill, we often break it down into smaller tasks, and learn one of these at a time. With experience these individual tasks grow familiar and seem simpler.

Some of the scripts you will be reading or writing when you have finished this book will be quite complicated and might look intimidating at first. However, a script is just a series of short instructions, each of which is performed in order to solve the problem in hand. This is why creating a script is like writing a recipe or manual that allows a computer to solve a puzzle one step at a time.

It is worth noting, however, that a computer doesn't learn how to perform tasks like you or I might; it needs to follow instructions every time it performs the task. So a program must give the computer enough detail to perform the task as if every time were its first time.





Start with the big picture of what you want to achieve, and break that down into smaller steps.

1: DEFINE THE GOAL

First, you need to define the task you want to achieve. You can think of this as a puzzle for the computer to solve.

2: DESIGN THE SCRIPT

To design a script you split the goal out into a series of tasks that are going to be involved in solving this puzzle. This can be represented using a flowchart.

You can then write down individual steps that the computer needs to perform in order to complete each individual task (and any information it needs to perform the task), rather like writing a recipe that it can follow.

3: CODE EACH STEP

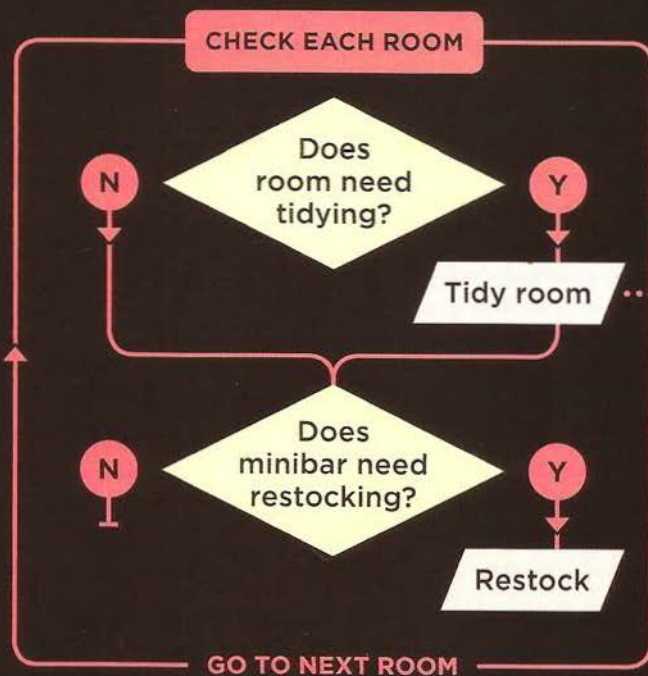
Each of the steps needs to be written in a programming language that the computer understands. In our case, this is JavaScript.

As tempting as it can be to start coding straight away, it pays to spend time designing your script before you start writing it.

DESIGNING A SCRIPT: TASKS

Once you know the **goal** of your script, you can work out the individual tasks needed to achieve it. This high-level view of the tasks can be represented using a flowchart.

FLOWCHART: TASKS OF A HOTEL CLEANER



DESIGNING A SCRIPT: STEPS

Each individual task may be broken down into a sequence of steps. When you are ready to code the script, these steps can then be translated into individual lines of code.

LIST: STEPS REQUIRED TO TIDY A ROOM

- STEP 1** Remove used bedding
- STEP 2** Wipe all surfaces
- STEP 3** Vacuum floors
- STEP 4** Fit new bedding
- STEP 5** Remove used towels and soaps
- STEP 6** Clean toilet, bath, sink, surfaces
- STEP 7** Place new towels and soaps
- STEP 8** Wipe bathroom floor

As you will see on the next page, the steps that a computer needs to follow in order to perform a task are often very different from those that you or I might take.

FROM STEPS TO CODE

Every step for every task shown in a flowchart needs to be written in a language the computer can understand and follow.

In this book, we are focussing on the JavaScript language and how it is used in web browsers.

Just like learning any new language, you need to get to grips with the:

- **Vocabulary:** The words that computers understand
- **Syntax:** How you put those words together to create instructions computers can follow

Along with learning the language itself, if you are new to programming, you will also need to learn how a computer achieves different types of goals using a **programmatic** approach to problem-solving.

Computers are very logical and obedient. They need to be told every detail of what they are expected to do, and they will do it without question. Because they need different types of instructions compared to you or I, everyone who learns to program makes lots of mistakes at the start. Don't be disheartened; in Chapter 10 you will see several ways to discover what might have gone wrong - programmers call this **debugging**.





You need to learn to “think” like a computer because they solve tasks in different ways than you or I might approach them.

Computers solve problems **programmatically**; they follow series of instructions, one after another. The type of instructions they need are often different to the type of instructions you might give to another human. Therefore, throughout the book you will not only learn the vocabulary and syntax that JavaScript uses, but you will also learn how to write instructions that computers can follow.

For example, when you look at the picture on the left how do you tell which person is the tallest? A computer would need explicit, step-by-step instructions, such as:

1. Find the height of the first person
2. Assume he or she is the “tallest person”
3. Look at the height of the remaining people one-by-one and compare their height to the “tallest person” you have found so far
4. At each step, if you find someone whose height is greater than the current “tallest person”, he or she becomes the new “tallest person”
5. Once you have checked all the people, tell me which one is the tallest

So the computer needs to look at each person in turn, and for each one it performs a test (“Are they taller than the current tallest person?”). Once it has done this for each person it can give its answer.

DEFINING A GOAL & DESIGNING THE SCRIPT

Consider how you might approach a different type of script. This example calculates the cost of a name plaque. Customers are charged by the letter.

The first thing you should do is detail your goals for the script (what you want it to achieve):

Customers can have a name added to a plaque; each letter costs \$5. When a user enters a name, show them how much it will cost.

Next, break it into a series of tasks that have to be performed in order to achieve the goals:

1. The script is triggered when the button is clicked.
2. It collects the name entered into the form field.
3. It checks that the user has entered a value.
4. If the user has not entered anything, a message will appear telling them to enter a name.
5. If a name has been entered, calculate the cost of the sign by multiplying the number of letters by the cost per letter.
6. Show how much the plaque costs.

(These numbers correspond with the flowchart on the right-hand page.)



CUSTOM SIGNAGE

Enter name:

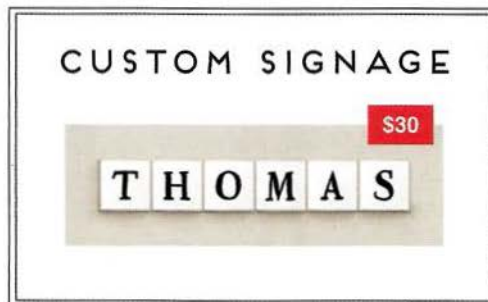
SHOW COST



CUSTOM SIGNAGE

Enter name: *Please enter a name below...*

SHOW COST



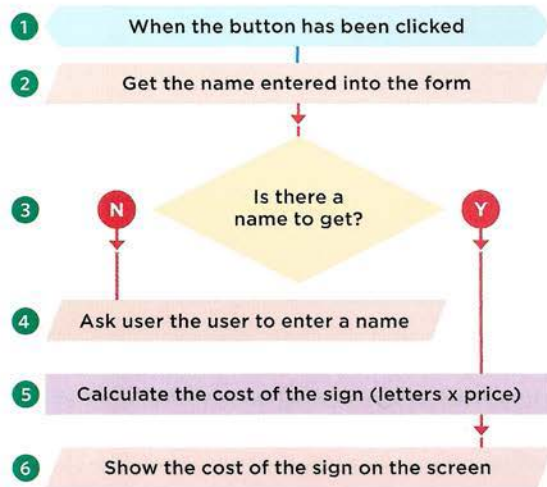
CUSTOM SIGNAGE

\$30

THOMAS

SKETCHING OUT THE TASKS IN A FLOWCHART

Often scripts will need to perform different tasks in different situations. You can use flowcharts to work out how the tasks fit together. The flowcharts show the paths between each step.

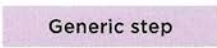

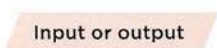



Arrows show how the script moves from one task to the next. The different shapes represent different types of tasks. In some places there are decisions which cause the code to follow different paths.

You will learn how to turn this example into code in Chapter 2. You will also see many more examples of different flowcharts throughout the book, and you will meet code that helps you deal with each of these types of situations.

Some experienced programmers use more complex diagram styles that are specifically designed to represent code - however, they have a steeper learning curve. These informal flowcharts will help you understand how scripts work while you are in the process of learning the language.

FLOWCHART KEY

 Generic step	 Event
 Input or output	 Decision

SUMMARY

THE ABC OF PROGRAMMING

A: What is a script and how do I create one?

- ▶ A script is a series of instructions that the computer can follow in order to achieve a goal.
- ▶ Each time the script runs, it might only use a subset of all the instructions.
- ▶ Computers approach tasks in a different way than humans, so your instructions must let the computer solve the task programmatically.
- ▶ To approach writing a script, break down your goal into a series of tasks and then work out each step needed to complete that task (a flowchart can help).



1/b

HOW DO COMPUTERS
FIT IN WITH THE
WORLD AROUND
THEM?

COMPUTERS CREATE MODELS OF THE WORLD USING DATA

Here is a model of a hotel, along with some model trees, model people, and model cars. To a human, it is clear what kind of real-world object each one represents.



A computer has no predefined concept of what a hotel or car is. It does not know what they are used for. Your laptop or phone will not have a favorite brand of car, nor will it know what star rating your hotel is.

So how do we use computers to create hotel booking apps, or video games where players can race a car? The answer is that programmers create a very different kind of model, especially for computers.

Programmers make these models using data. That is not as strange or as scary as it sounds because the data is all the computer needs in order to follow the instructions you give it to carry out its tasks.



OBJECTS & PROPERTIES

If you could not see the picture of the hotel and cars, the data in the information boxes alone would still tell you a lot about this scene.

OBJECTS (THINGS)

In computer programming, each physical thing in the world can be represented as an **object**. There are two different **types** of objects here: a hotel and a car.

Programmers might say that there is one **instance** of the hotel object, and two **instances** of the car object.

Each object can have its own:

- Properties
- Events
- Methods

Together they create a working model of that object.

PROPERTIES (CHARACTERISTICS)

Both of the cars share common characteristics. In fact, all cars have a make, a color, and engine size. You could even determine their current speed. Programmers call these characteristics the **properties** of an object.

Each property has a **name** and a **value**, and each of these name/value pairs tells you something about each individual instance of the object.

The most obvious property of this hotel is its name. The value for that property is Quay. You can tell the number of rooms the hotel has by looking at the value next to the rooms property.

The idea of name/value pairs is used in both HTML and CSS. In HTML, an attribute is like a property; different attributes have different names, and each attribute can have a value. Similarly, in CSS you can change the color of a heading by creating a rule that gives the color property a specific value, or you can change the typeface it is written in by giving the font-family property a specific value. Name/value pairs are used a lot in programming.

HOTEL OBJECT

The hotel object uses property names and values to tell you about this particular hotel, such as the hotel's name, its rating, the number of rooms it has, and how many of these are booked. You can also tell whether or not this hotel has certain facilities.

OBJECT TYPE: HOTEL	
PROPERTIES	
name	Quay
rating	4
rooms	42
bookings	21
gym	false
pool	true

CAR OBJECTS

The car objects both share the same properties, but each one has different values for those properties. They tell you the make of car, what speed each car is currently traveling at, what color it is, and what type of fuel it requires.

OBJECT TYPE: CAR	
PROPERTIES	
make	BMW
currentSpeed	30mph
color	silver
fuel	diesel

OBJECT TYPE: CAR	
PROPERTIES	
make	Porsche
currentSpeed	20mph
color	silver
fuel	gasoline



EVENTS

In the real world, people interact with objects. These interactions can change the values of the properties in these objects.

WHAT IS AN EVENT?

There are common ways in which people interact with each type of object. For example, in a car a driver will typically use at least two pedals. The car has been designed to respond differently when the driver interacts with each of the different pedals:

- The accelerator makes the car go faster
- The brake slows it down

Similarly, programs are designed to do different things when users interact with the computer in different ways. For example, clicking on a contact link on a web page could bring up a contact form, and entering text into a search box may automatically trigger the search functionality.

An **event** is the computer's way of sticking up its hand to say, "Hey, this just happened!"

WHAT DOES AN EVENT DO?

Programmers choose which events they respond to. When a specific event happens, that event can be used to trigger a specific section of the code.

Scripts often use different events to trigger different types of functionality.

So a script will state which events the programmer wants to respond to, and what part of the script should be run when each of those events occur.

HOTEL OBJECT

A hotel will regularly have bookings for rooms. Each time a room is reserved, an event called `book` can be used to trigger code that will increase the value of the `bookings` property. Likewise, a `cancel` event can trigger code that decreases the value of the `bookings` property.

CAR OBJECTS

A driver will accelerate and brake throughout any car journey. An `accelerate` event can trigger code to increase the value of the current `Speed` property and a `brake` event can trigger code to decrease it. You will learn about the code that responds to the events and changes these properties on the next page.

OBJECT TYPE: HOTEL

EVENT happens when:

<code>book</code>	reservation is made
<code>cancel</code>	reservation is cancelled

OBJECT TYPE: CAR

EVENT happens when:

<code>brake</code>	driver slows down
<code>accelerate</code>	driver speeds up

OBJECT TYPE: CAR

EVENT happens when:

<code>brake</code>	driver slows down
<code>accelerate</code>	driver speeds up



METHODS

Methods represent things people need to do with objects. They can retrieve or update the values of an object's properties.

WHAT IS A METHOD?

Methods typically represent how people (or other things) interact with an object in the real world.

They are like questions and instructions that:

- Tell you something about that object (using information stored in its properties)
- Change the value of one or more of that object's properties

WHAT DOES A METHOD DO?

The code for a method can contain lots of instructions that together represent one task.

When you use a method, you do not always need to know *how* it achieves its task; you just need to know how to ask the question and how to interpret any answers it gives you.

HOTEL OBJECT

Hotels will commonly be asked if any rooms are free. To answer this question, a method can be written that subtracts the number of bookings from the total number of rooms. Methods can also be used to increase and decrease the value of the bookings property when rooms are booked or cancelled.

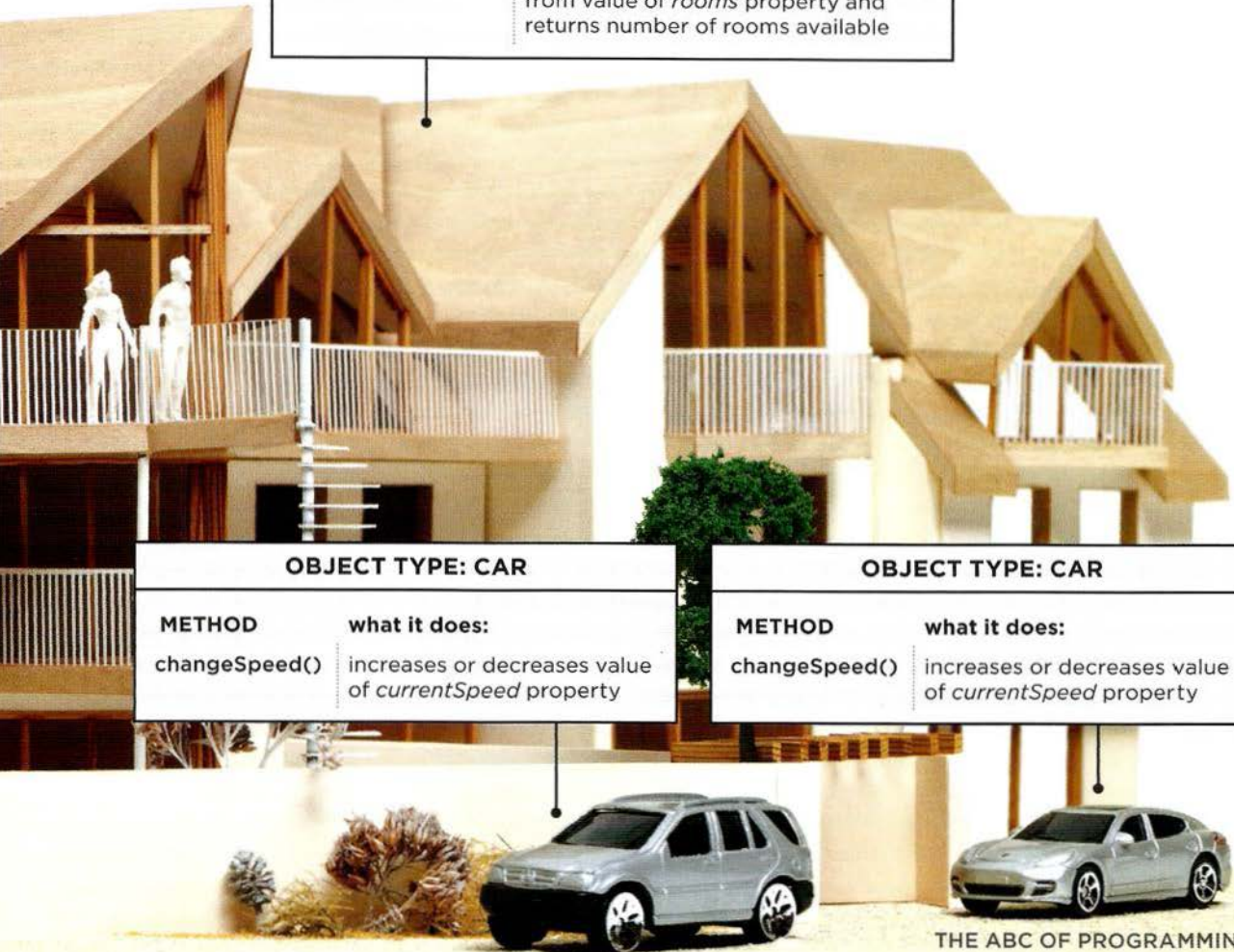
CAR OBJECTS

The value of the `currentSpeed` property needs to go up and down as the driver accelerates and brakes. The code to increase or decrease the value of the `currentSpeed` property could be written in a method, and that method could be called `changeSpeed()`.

OBJECT TYPE: HOTEL	
METHOD	what it does:
<code>makeBooking()</code>	increases value of <i>bookings</i> property
<code>cancelBooking()</code>	decreases value of <i>bookings</i> property
<code>checkAvailability()</code>	subtracts value of <i>bookings</i> property from value of <i>rooms</i> property and returns number of rooms available

OBJECT TYPE: CAR	
METHOD	what it does:
<code>changeSpeed()</code>	increases or decreases value of <i>currentSpeed</i> property

OBJECT TYPE: CAR	
METHOD	what it does:
<code>changeSpeed()</code>	increases or decreases value of <i>currentSpeed</i> property



PUTTING IT ALL TOGETHER

Computers use data to create models of things in the real world. The events, methods, and properties of an object all relate to each other: Events can trigger methods, and methods can retrieve or update an object's properties.

OBJECT TYPE: HOTEL			
1	EVENT	happens when:	method called:
	book	reservation is made	makeBooking()
	cancel	reservation is cancelled	cancelBooking()
2	METHOD	what it does:	
	makeBooking()	increases value of <i>bookings</i> property	
	cancelBooking()	decreases value of <i>bookings</i> property	
	checkAvailability()	subtracts value of <i>bookings</i> property from value of <i>rooms</i> property and returns number of rooms available	
	PROPERTIES		
	name	Quay	
	rating	4	
	rooms	42	
	bookings	22	
	gym	false	
	pool	true	

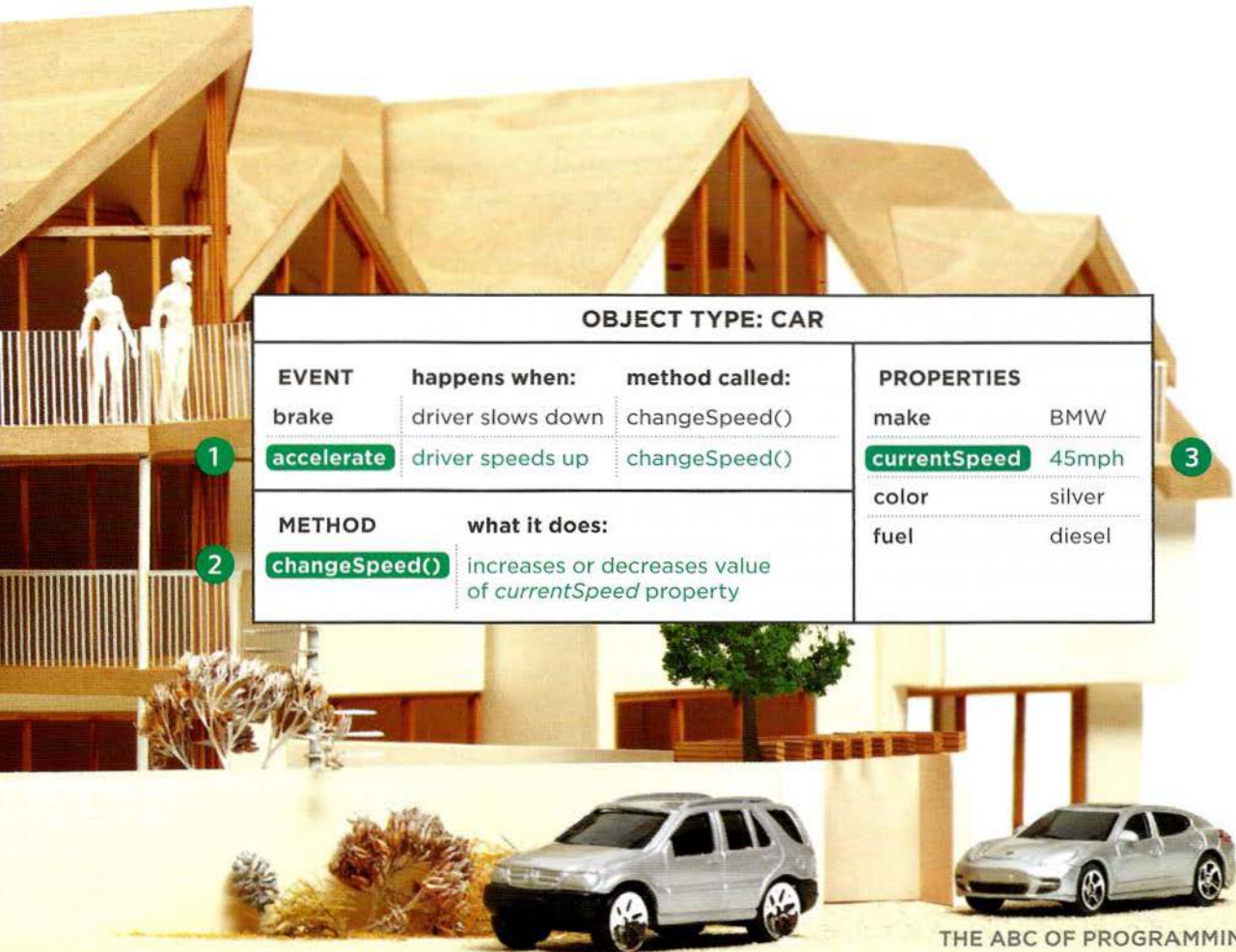
QUAY
HOTEL

HOTEL OBJECT

1. When a reservation is made, the `book` event fires.
2. The `book` event triggers the `makeBooking()` method, which increases the value of the `bookings` property.
3. The value of the `bookings` property is changed to reflect how many rooms the hotel has available.

CAR OBJECTS

1. As a driver speeds up, the `accelerate` event fires.
2. The `accelerate` event calls the `changeSpeed()` method, which in turn increases the value of the `currentSpeed` property.
3. The value of the `currentSpeed` property reflects how fast the car is traveling.



OBJECT TYPE: CAR			
EVENT	happens when:	method called:	PROPERTIES
brake	driver slows down	changeSpeed()	make BMW
1 accelerate	driver speeds up	changeSpeed()	3 currentSpeed 45mph
2 METHOD	what it does:		color silver
changeSpeed()	increases or decreases value of currentSpeed property		fuel diesel

WEB BROWSERS ARE PROGRAMS BUILT USING OBJECTS

You have seen how data can be used to create a model of a hotel or a car. Web browsers create similar models of the web page they are showing and of the browser window that the page is being shown in.

WINDOW OBJECT

On the right-hand page you can see a model of a computer with a browser open on the screen.

The browser represents each window or tab using a `window` object. The `location` property of the `window` object will tell you the URL of the current page.

DOCUMENT OBJECT

The current web page loaded into each window is modelled using a `document` object.

The `title` property of the `document` object tells you what is between the opening `<title>` and closing `</title>` tag for that web page, and the `lastModified` property of the `document` object tells you the date this page was last updated.



OBJECT TYPE: WINDOW

PROPERTIES

location <http://www.javascriptbook.com/>

OBJECT TYPE: DOCUMENT

PROPERTIES

URL <http://www.javascriptbook.com/>

lastModified 09/04/2014 15:33:37

title Learn JavaScript & jQuery -
A book that teaches you
in a nicer way

THE DOCUMENT OBJECT REPRESENTS AN HTML PAGE

Using the document object, you can access and change what content users see on the page and respond to how they interact with it.

Like other objects that represent real-world things, the document object has:

PROPERTIES

Properties describe characteristics of the current web page (such as the title of the page).

METHODS

Methods perform tasks associated with the document currently loaded in the browser (such as getting information from a specified element or adding new content).

EVENTS

You can respond to events, such as a user clicking or tapping on an element.

Because all major web browsers implement the document object in the same way, the people who create the browsers have already:

- Implemented properties that you can access to find out about the current page in the browser
- Written methods that achieve some common tasks that you are likely to want to do with an HTML page

So you will be learning how to work with this object. In fact, the document object is just one of a set of objects that all major browsers support. When the browser creates a model of a web page, it not only creates a document object, but it also creates a new object for each element on the page. Together these objects are described in the **Document Object Model**, which you will meet in Chapter 5.

OBJECT TYPE: DOCUMENT

PROPERTIES

URL	http://www.javascriptbook.com/
lastModified	09/04/2014 15:33:37
title	Learn JavaScript & jQuery - A book that teaches you in a nicer way

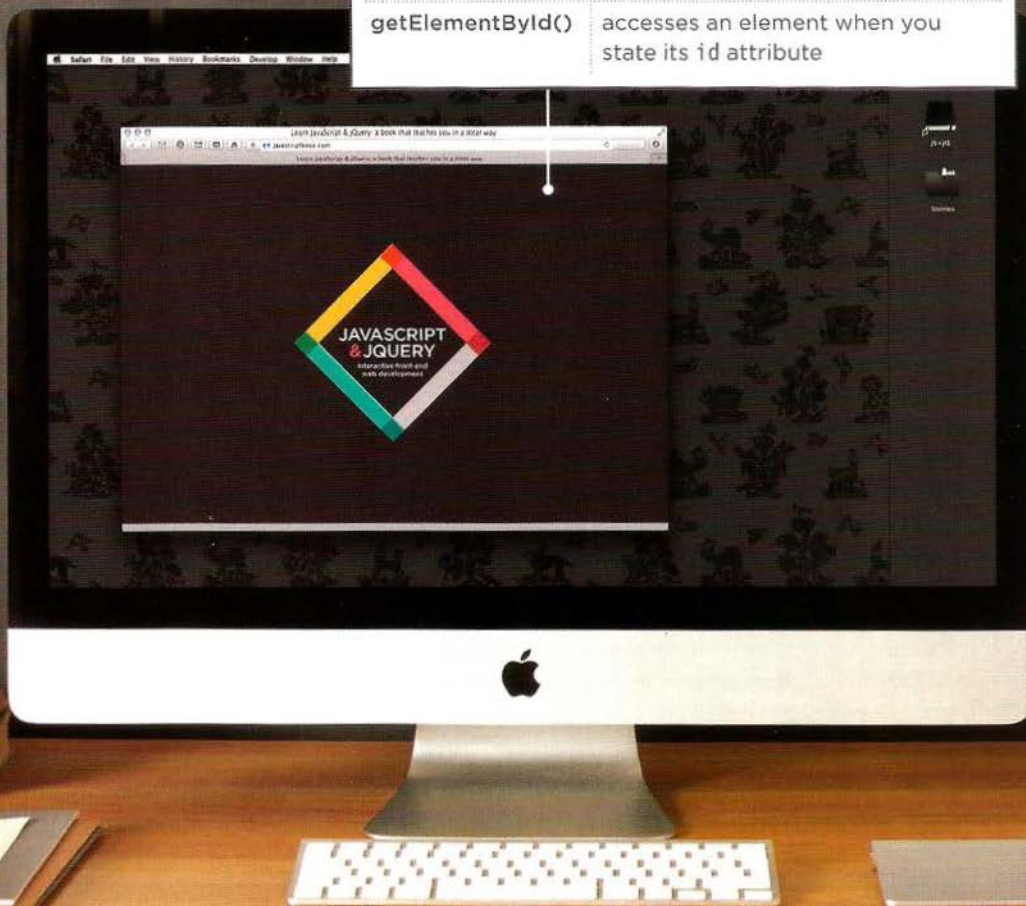
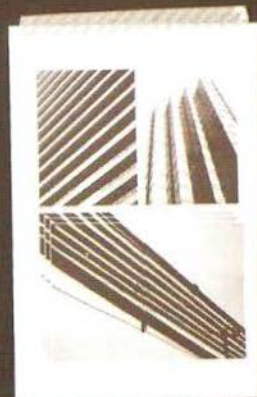
EVENT happens when:

load	page and assets have finished loading
click	user clicks the mouse over the page
keypress	user presses down on a key

METHOD

what it does:

write()	adds new content to the document
getElementById()	accesses an element when you state its id attribute



HOW A BROWSER SEES A WEB PAGE

In order to understand how you can change the content of an HTML page using JavaScript, you need to know how a browser interprets the HTML code and applies styling to it.

1: RECEIVE A PAGE AS HTML CODE

Each page on a website can be seen as a separate **document**. So, the web consists of many sites, each made up of one or more documents.

2: CREATE A MODEL OF THE PAGE AND STORE IT IN MEMORY

The model shown on the right hand page is a representation of one very basic page. Its structure is reminiscent of a family tree. At the top of the model is a **document object**, which represents the whole document.

Beneath the **document** object each box is called a **node**. Each of these nodes is another object. This example features three types of nodes representing elements, text within the elements, and attribute.

3: USE A RENDERING ENGINE TO SHOW THE PAGE ON SCREEN

If there is no CSS, the rendering engine will apply default styles to HTML elements. However, the HTML code for this example links to a CSS style sheet, so the browser requests that file and displays the page accordingly.

When the browser receives CSS rules, the rendering engine processes them and applies each rule to its corresponding elements. This is how the browser positions the elements in the correct place, with the right colors, fonts, and so on.

All major browsers use a JavaScript interpreter to translate your instructions (in JavaScript) into instructions the computer can follow.

When you use JavaScript in the browser, there is a part of the browser that is called an **interpreter** (or scripting engine).

The interpreter takes your instructions (in JavaScript) and translates them into instructions the browser can use to achieve the tasks you want it to perform.

In an **interpreted programming language**, like JavaScript, each line of code is translated one-by-one as the script is run.

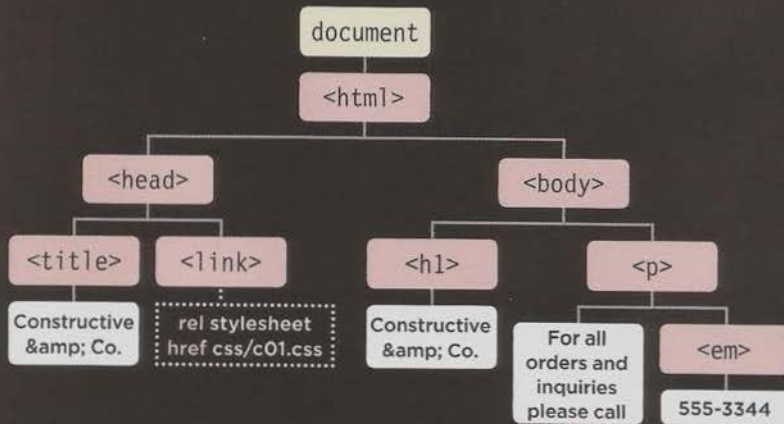
```

<!DOCTYPE html>
<html>
  <head>
    <title>Constructive & Co.</title>
    <link rel="stylesheet" href="css/c01.css" />
  </head>
  <body>
    <h1>Constructive & Co.</h1>
    <p>For all orders and inquiries please call
      <em>555-3344</em></p>
  </body>
</html>

```

1

The browser receives an HTML page.



2

It creates a model of the page and stores it in memory.



3

It shows the page on screen using a rendering engine.

SUMMARY

THE ABC OF PROGRAMMING

B: How do computers fit in with the world around them?

- ▶ Computers create models of the world using data.
- ▶ The models use objects to represent physical things. Objects can have: properties that tell us about the object; methods that perform tasks using the properties of that object; events which are triggered when a user interacts with the computer.
- ▶ Programmers can write code to say "When this event occurs, run that code."
- ▶ Web browsers use HTML markup to create a model of the web page. Each element creates its own node (which is a kind of object).
- ▶ To make web pages interactive, you write code that uses the browser's model of the web page.



1/c

HOW DO I WRITE A
SCRIPT FOR A
WEB PAGE?

HOW HTML, CSS, & JAVASCRIPT FIT TOGETHER

Before diving into the JavaScript language, you need to know how it will fit together with the HTML and CSS in your web pages.

Web developers usually talk about three languages that are used to create web pages: HTML, CSS, and JavaScript.

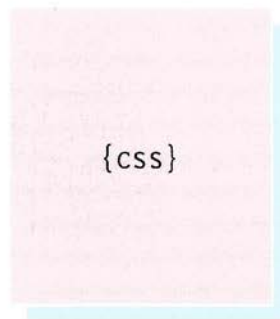
Where possible, aim to keep the three languages in separate files, with the HTML page linking to CSS and JavaScript files.

Each language forms a separate **layer** with a different purpose. Each layer, from left to right, builds on the previous one.



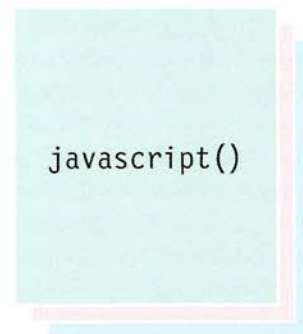
CONTENT LAYER
.html files

This is where the content of the page lives. The HTML gives the page structure and adds semantics.



PRESENTATION LAYER
.css files

The CSS enhances the HTML page with rules that state how the HTML content is presented (backgrounds, borders, box dimensions, colors, fonts, etc.).



BEHAVIOR LAYER
.js files

This is where we can change how the page behaves, adding interactivity. We will aim to keep as much of our JavaScript as possible in separate files.

Programmers often refer to this as a **separation of concerns**.

PROGRESSIVE ENHANCEMENT

These three layers form the basis of a popular approach to building web pages called progressive enhancement.

As more and more web-enabled devices come onto the market, this concept is becoming more widely adopted.

It's not just screen sizes that are varied - connection speeds and capabilities of each device can also differ.

Also, some people browse with JavaScript turned off, so you need to make sure that the page still works for them.

Constructive & Co.

For all orders and inquiries please call 555-3344



HTML ONLY

Starting with the HTML layer allows you to focus on the most important thing about your site: its content.

Being plain HTML, this layer should work on all kinds of devices, be accessible to all users, and load quite quickly on slow connections.

HTML+CSS

Adding the CSS rules in a separate file keeps rules regarding how the page looks away from the content itself.

You can use the same style sheet with all of your site, making your sites faster to load and easier to maintain. Or you can use different style sheets with the same content to create different views of the same data.

HTML+CSS+JAVASCRIPT

The JavaScript is added last and enhances the usability of the page or the experience of interacting with the site.

Keeping it separate means that the page still works if the user cannot load or run the JavaScript. You can also reuse the code on several pages (making the site faster to load and easier to maintain).

CREATING A BASIC JAVASCRIPT

JavaScript is written in plain text, just like HTML and CSS, so you do not need any new tools to write a script. This example adds a greeting into an HTML page. The greeting changes depending on the time of day.

1 Create a folder to put the example in called c01, then start up your favorite code editor, and enter the text to the right.

A JavaScript file is just a text file (like HTML and CSS files are) but it has a .js file extension, so save this file with the name `add-content.js`

Don't worry about what the code means yet, for now we will focus on how the script is created and how it fits with an HTML page.

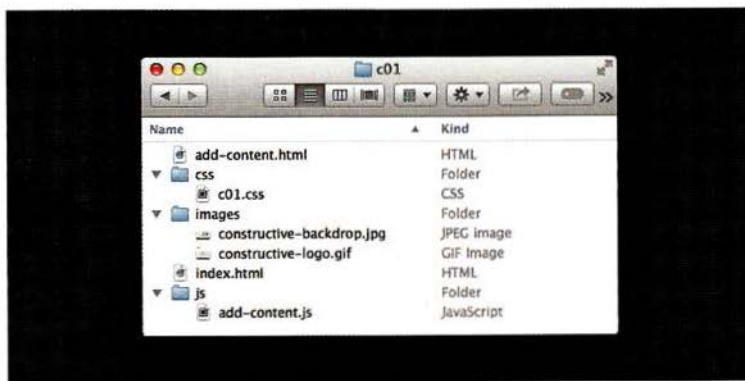
2 Get the CSS and images for this example from the website that accompanies the book: www.javascriptbook.com

To keep the files organized, in the same way that CSS files often live in a folder called `styles` or `css`, your JavaScript files can live in a folder called `scripts`, `javascript`, or `js`. In this case, save your file in a folder called `js`

```
var today = new Date();
var hourNow = today.getHours();
var greeting;

if (hourNow > 18) {
    greeting = 'Good evening!';
} else if (hourNow > 12) {
    greeting = 'Good afternoon!';
} else if (hourNow > 0) {
    greeting = 'Good morning!';
} else {
    greeting = 'Welcome!';
}

document.write('<h3>' + greeting + '</h3>');
```



Here you can see the file structure that you will end up with when you finish the example. Always treat file names as being case-sensitive.

LINKING TO A JAVASCRIPT FILE FROM AN HTML PAGE

When you want to use JavaScript with a web page, you use the HTML `<script>` element to tell the browser it is coming across a script. Its `src` attribute tells people where the JavaScript file is stored.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Constructive & Co.</title>
    <link rel="stylesheet" href="css/c01.css" />
  </head>
  <body>
    <h1>Constructive & Co.</h1>
    <script src="js/add-content.js"></script>
    <p>For all orders and inquiries please call
      <em>555-3344</em></p>
  </body>
</html>
```

3 In your code editor, enter the HTML shown on the left. Save this file with the name `add-content.html`

The HTML `<script>` element is used to load the JavaScript file into the page. It has an attribute called `src`, whose value is the path to the script you created.

This tells the browser to find and load the script file (just like the `src` attribute on an `` tag).



4 Open the HTML file in your browser. You should see that the JavaScript has added a greeting (in this case, *Good Afternoon!*) to the page. (These greetings are coming from the JavaScript file; they are not in the HTML file.)

Please note: Internet Explorer sometimes prevents JavaScript running when you open a page stored on your hard drive. If this affects you, please try Chrome, Firefox, Opera, or Safari instead.

THE SOURCE CODE IS NOT AMENDED

If you look at the source code for the example you just created, you will see that the HTML is still exactly the same.

5 Once you have tried the example in your browser, view the source code for the page. (This option is usually under the *View*, *Tools* or *Develop* menu of the browser.)



6 The source of the web page does not actually show the new element that has been added into the page; it just shows the link to the JavaScript file.

As you move through the book, you will see most of the scripts are added just before the closing `</body>` tag (this is often considered a better place to put your scripts).



PLACING THE SCRIPT IN THE PAGE

You may see JavaScript in the HTML between opening `<script>` and closing `</script>` tags (but it is better to put scripts in their own files).

```
<!DOCTYPE html>
<html>
  <head>
    <title>Constructive & Co.</title>
    <link rel="stylesheet" href="css/c01.css" />
  </head>
  <body>
    <h1>Constructive & Co.</h1>
    <script>document.write('<h3>Welcome!</h3>');
    </script>
    <p>For all orders and inquiries please call
      <em>555-3344</em></p>
  </body>
</html>
```



7 Finally, try opening the HTML file, removing the `src` attribute from the opening `<script>` tag, and adding the new code shown on the left between the opening `<script>` tag and the closing `</script>` tag. The `src` attribute is no longer needed because the JavaScript is in the HTML page.

As noted on p44, it is better not to mix JavaScript in your HTML pages like this, but it is mentioned here as you may come across this technique.

8 Open the HTML file in your web browser and the welcome greeting is written into the page.

As you may have guessed, `document.write()` writes content into the *document* (the web page). It is a simple way to add content to a page, but not always the best. Chapter 5 discusses various ways to update the content of a page.

HOW TO USE OBJECTS & METHODS

This one line of JavaScript shows how to use objects and methods. Programmers refer to this as **calling** a method of an object.

The **document** object represents the entire web page. All web browsers implement this object, and you can use it just by giving its name.

The **write()** method of the **document** object allows new content to be written into the page where the `<script>` element sits.

The diagram shows the code `document.write('Good afternoon!');` with brackets and lines pointing to labels. 'document' is labeled as 'OBJECT'. 'write' is labeled as 'METHOD'. The dot '.' is labeled as 'MEMBER OPERATOR'. The string 'Good afternoon!' is labeled as 'PARAMETERS'.

The **document** object has several methods and properties. They are known as **members** of that object.

Whenever a method requires some information in order to work, the data is given inside the parentheses.

You can access the members of an object using a dot between the object name and the member you want to access. It is called a **member operator**.

Each piece of information is called a **parameter** of the method. In this case, the **write()** method needs to know what to write into the page.

Behind the scenes, the browser uses a lot more code to make the words appear on the screen, but you don't need to know how the browser does this.

You only need to know how to call the object and method, and how to tell it the information it needs to do the job you want it to. It will do the rest.

There are lots of objects like the **document** object, and lots of methods like the **write()** method that will help you write your own scripts.

JAVASCRIPT RUNS WHERE IT IS FOUND IN THE HTML

When the browser comes across a `<script>` element, it stops to load the script and then checks to see if it needs to do anything.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Constructive & Co.</title>
    <link rel="stylesheet" href="css/c01.css" />
  </head>
  <body>
    <h1>Constructive & Co.</h1>
    <p>For all orders and inquiries please call <em>555-3344</em></p>
    <script src="js/add-content.js"></script>
  </body>
</html>
```

Note how the `<script>` element can be moved below the first paragraph, and this affects where the new greeting is written into the page.

This has implications for where `<script>` elements should be placed, and can affect the loading time of pages (see p356).



SUMMARY

THE ABC OF PROGRAMMING

C: How do I write a script for a web page?

- ▶ It is best to keep JavaScript code in its own JavaScript file. JavaScript files are text files (like HTML pages and CSS style sheets), but they have the `.js` extension.
- ▶ The HTML `<script>` element is used in HTML pages to tell the browser to load the JavaScript file (rather like the `<link>` element can be used to load a CSS file).
- ▶ If you view the source code of the page in the browser, the JavaScript will not have changed the HTML, because the script works with the model of the web page that the browser has created.