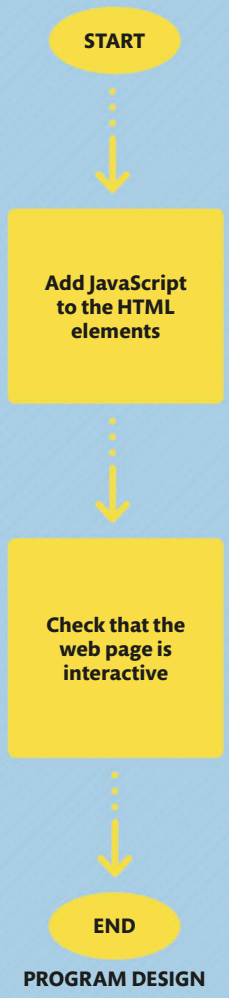
Animating the web page

JavaScript is used to extend the functionality of a website and make it more dynamic. Here, it will allow you to add intelligent and interactive behaviors to the existing HTML framework, completing the web page project.

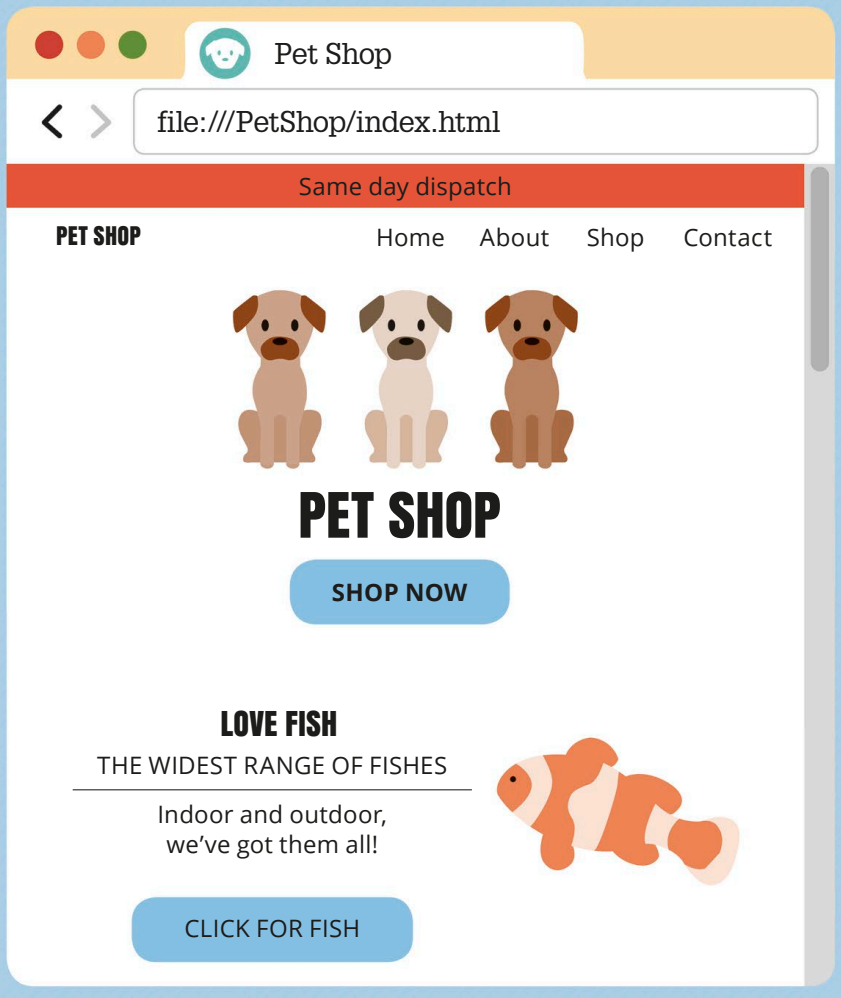
What the program does

In this part of the project, JavaScript is added to the structured and styled web page created in Styling a web page (see pp.242–263). The functionalities added in this part of the project will allow the web page to handle customized user interactions.



PROGRAM DESIGN

Interactive website
The web page will now have interactive elements. The promo bar on top will cycle through four messages, and the scroll button will be visible at the end of the web page. It will scroll up to the top when clicked.



YOU WILL LEARN

- How to create JavaScript files
- How to use JQuery
- How to animate HTML elements

Time: 1 hour

Lines of code: 89

Difficulty level

WHERE THIS IS USED

JavaScript is used in almost all websites. It helps web developers make web pages more attractive and interactive by implementing custom client-side scripts. It even allows the use of cross-platform run time engines, such as Node.js, to write server-side code.

Project requirements

For this project, you will need the previously created HTML and CSS files. You can continue using the same IDE.

HTML FILE

CSS FILE

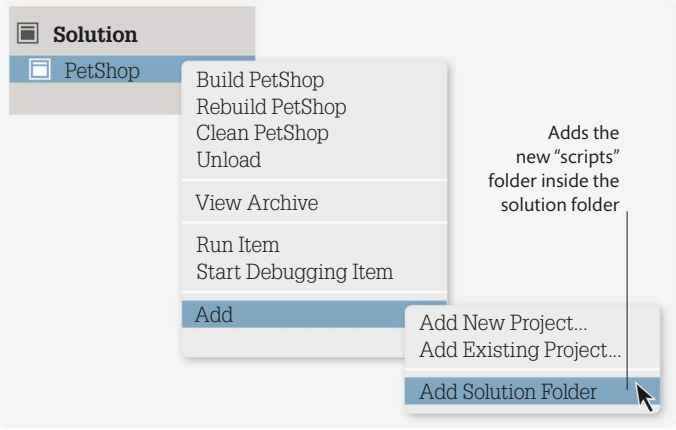
DEVELOPMENT ENVIRONMENT

1 Getting started

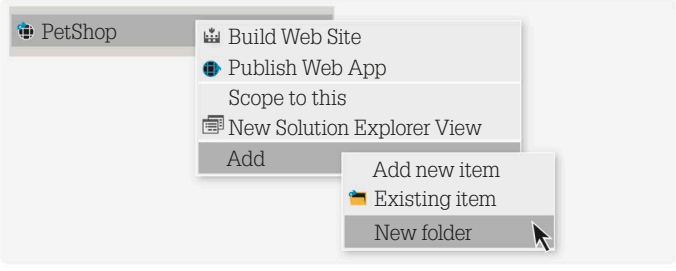
To add the interactive functionality of JavaScript to the website, you will require multiple JavaScript files. You will also link the HTML document to a JavaScript framework (see pp.284–285) to make programming easier.

1.1 ADD A FOLDER

Create a new folder, called "scripts", to contain the JavaScript files. On a Mac, open Finder and create the folder inside the website folder. Next, open Visual Studio, right-click on the project name, select Add, and choose Add Solution Folder. In Windows, right-click on the project name in Solution Explorer, select Add, and choose New Folder.



MAC



WINDOWS

1.2 ADD JQUERY

Before adding the custom JavaScript files, you need to add JQuery (see p.284) to the HTML file. You will use JQuery in the custom scripts to make it easier to target the HTML elements. In the "index.html" file, add a <script> tag inside the <head> tag, pointing to the online location that you can use to retrieve JQuery. This online location is called a CDN (content delivery network). You can download these files and host them in your own site, but it is often easier and quicker to use a CDN.

HTML

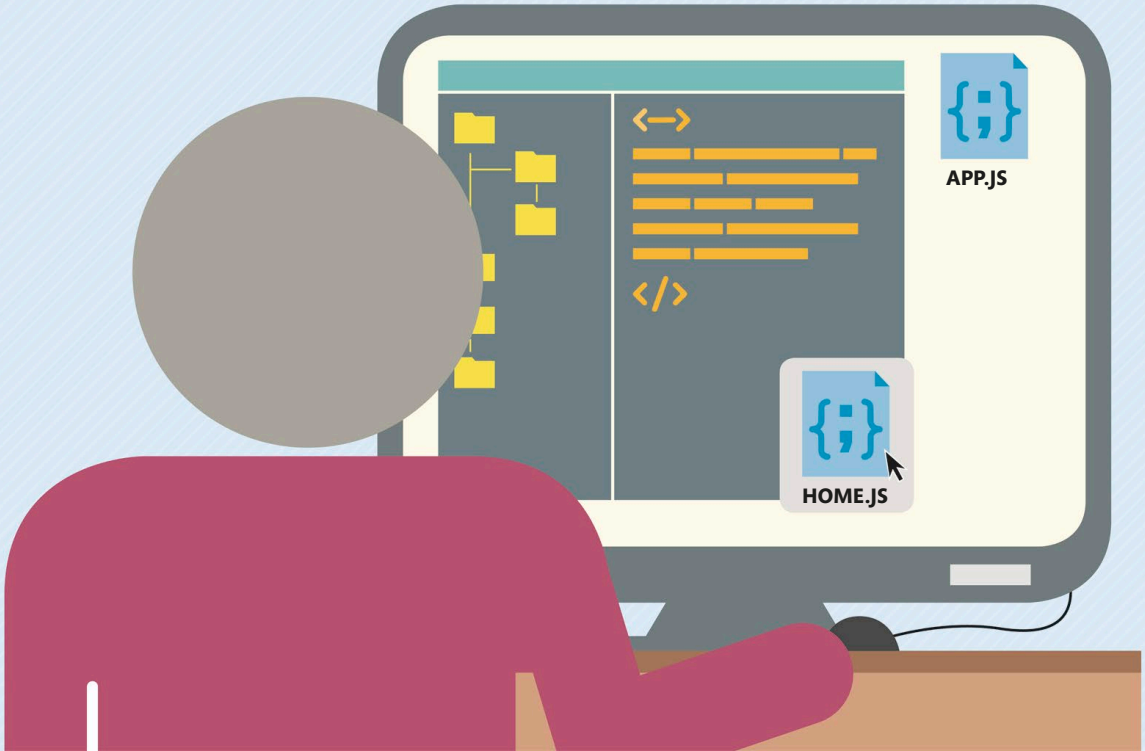
```
    <link href="https://fonts.googleapis.com/css?family=Anton|
    Open+Sans" rel="stylesheet">
    <script src="https://code.jquery.com/jquery-3.3.1.min.js">
    </script>
</head>
```

The "src" attribute points to the CDN for JQuery

Link to retrieve JQuery

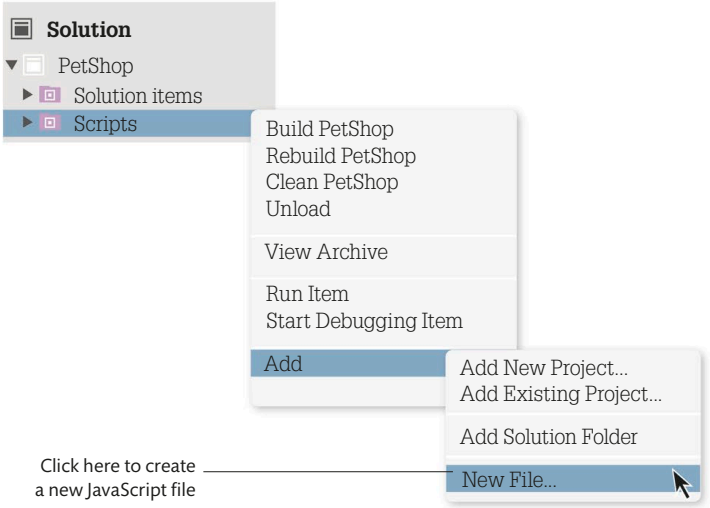
2 Adding JavaScript files

The web page in this project needs three custom JavaScript files. In this section, you will create the first two files—one to contain the global scope variables and another to contain functionality for the features of the home page.



2.1 ADD NEW FILE

The first custom script you need to add is called "app.js". This will add an "app" class/function that you can instantiate to hold all of the global scope variables. Right-click on the "scripts" folder, select Add, and choose New File to create a JavaScript file. Name this file "app.js".



2.2 REFERENCE APP.JS

You now need to link the JavaScript file to the HTML file. In the "index.html" file , add a <script> tag that will point to the newly created JavaScript file. Place this link inside the <head> tag, just below the <script> tag for JQuery. The order in which you declare the JavaScript files is important, because the scripts must be loaded into the JavaScript Engine (see p.264) before they can be called. For example, JQuery must be loaded before you can use any of its methods.

HTML

```
    <script src="https://code.jquery.com/jquery-3.3.1.min.js">
    </script>
    <script src="scripts/app.js"></script>
```

Adds a new <script> tag that points to the new JavaScript file

2.3 CREATE FUNCTION

Inside "app.js", declare a variable called **app** that is a self-executing function (see p.279). Next, add a property called "websiteName" and a method called "getWebsiteName" inside the function. This is an example of how to add functionality to the app class.

```
var app = (function () {
    /* Properties */
    var websiteName = "PetShop";
    /* Methods */
    return {
        getWebsiteName: function () {
            return websiteName;
        }
    }
})();
```

Round brackets instruct the JavaScript Engine to run that function immediately

Code after double slashes (//) or between /* and */ is treated as a comment in JavaScript

JS

SAVE

2.4 ADD ANOTHER FILE

Next, add another new custom script with all of the logic you will need for the home page. Follow the same steps as before to create a new JavaScript file inside the scripts folder and name it "home.js".

HOME.JS

2.5 REFERENCE HOME.JS

As before, you will need to reference this new JavaScript file in the HTML document. In the "index.html" file, add a <script> tag that points to the "home.js" file. Ensure that this is placed below the reference for "app.js" added in step 2.2.

HTML

```
    <script src="scripts/app.js"></script>
    <script src="scripts/home.js"></script>
```

Indicates that the file is inside the scripts folder

SAVE

2.6 ADD FUNCTIONALITY TO HOME.JS

In "home.js", create a function called **HomeIndex()**, which will contain all of the functionality required by the home page. Below this function, add an **on document ready()** function. This is a JQuery command that instructs the JavaScript Engine to wait until all of the elements on the page have finished loading before running the code within it. Inside the **on document ready()** function, you will instantiate the **HomeIndex()** function as a property of the "app" object, which has already been instantiated in the "app.js" file.

JS

```
function HomeIndex() {
}
$(document).ready(function () {
    /* Instantiate new Home class */
    app.homeIndex = new HomeIndex();
});
```

The dollar sign denotes the JQuery function

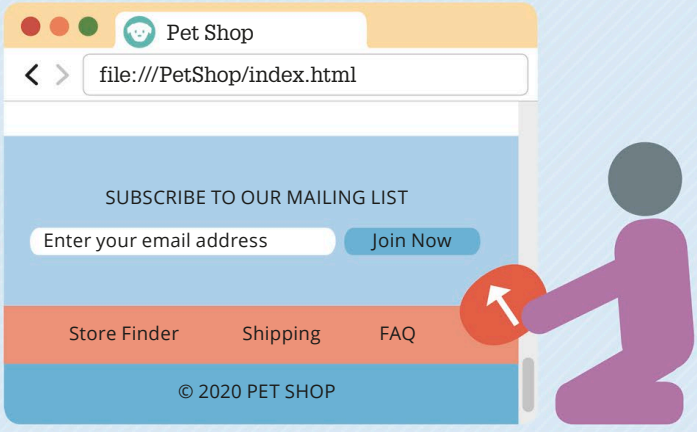
Makes a function available after the document is loaded

Comments are ignored and will not be executed

Links "home.js" to "app.js"

3 Managing the Scroll to top button

In the next few steps, you will add functionality to the Scroll to top button. You need to add code to control when the button becomes visible and to make it scroll back to the top of the page when clicked.



3.1 DEFINE PROPERTIES

Add a property inside the **HomeIndex()** function to set the height from the top of the page at which the Scroll to top button should become visible.

```
function HomeIndex() {
    /* Properties */
    const heightFromTop = 300;
}
```

The height will not change, so it is defined as constant

3.2 DEFINE METHODS

Now add a method to initialize the Scroll to top button. This method will control two aspects of the button. It will add an "on scroll" event handler so that every time the user scrolls up or down in the browser, the JavaScript Engine checks if the scroll distance is more than the amount defined in the **heightFromTop** value. The button is then made visible or hidden accordingly. The method also adds an "on click" event handler, so that every time the user clicks the Scroll to top button, the page will scroll back to the top. Add this code within the **HomeIndex()** function.

```
    const heightFromTop = 300;
    /* Methods */
    this.initialiseScrollToTopButton = function () {
    }
}
```

The keyword "this" refers to the owner object—in this case, the **HomeIndex()** function

3.3 ADD CALL TO INITIALIZE

In the **document ready()** function, add this code below the "app.homeIndex" declaration. This will add the call to run the **initialiseScrollToTopButton()** method.

```
$(document).ready(function () {
    /* Instantiate new Home class */
    app.homeIndex = new HomeIndex();
    /* Initialize the Scroll To Top button */
    app.homeIndex.initialiseScrollToTopButton();
});
```

Initializes the Scroll to top button

3.4 SHOW THE BUTTON

Add the "on scroll" event handler in the `initialiseScrollToTopButton()` function. This determines the current scroll distance by using the JavaScript command "scrollTop()". It then compares the current scroll distance with the "heightToTop" value to see if the scroll button needs to fade in or fade out.

```
    /* Methods */
    this.initialiseScrollToTopButton = function () {
        /* Window Scroll Event Handler */
        $(window).scroll(function () {
            /* Show or Hide Scroll to Top Button based on
            scroll distance*/
            var verticalHeight = $(this).scrollTop();
            if (verticalHeight > heightFromTop) {
                $("#scrollToTop").fadeIn();
            } else {
                $("#scrollToTop").fadeOut();
            }
        });
    }
```

This instruction tells JQuery to run the code block every time the user scrolls the page

JQuery selector that targets the "window" object

This selector tells JQuery to use the element that triggered the event—the window object, in this instance

This JQuery selector targets the HTML element with id="scrollToTop"

JQuery methods that automatically animate the button

3.5 CLICKING THE BUTTON

Next, you need to add an event handler to manage what happens when the Scroll to top button is clicked. To do this, add the JQuery `click()` function that detects when the button is clicked. Use the `animate()` command to instruct JQuery to animate the "html" and "body" elements when the button is clicked.

```
                $("#scrollToTop").fadeOut();
            }
        });
        /* Scroll to Top Click Event Handler */
        $("#scrollToTop").click(function () {
            $("html, body").animate({ scrollTop: 0 },"slow");
        });
    }
```

This code runs every time the button is clicked

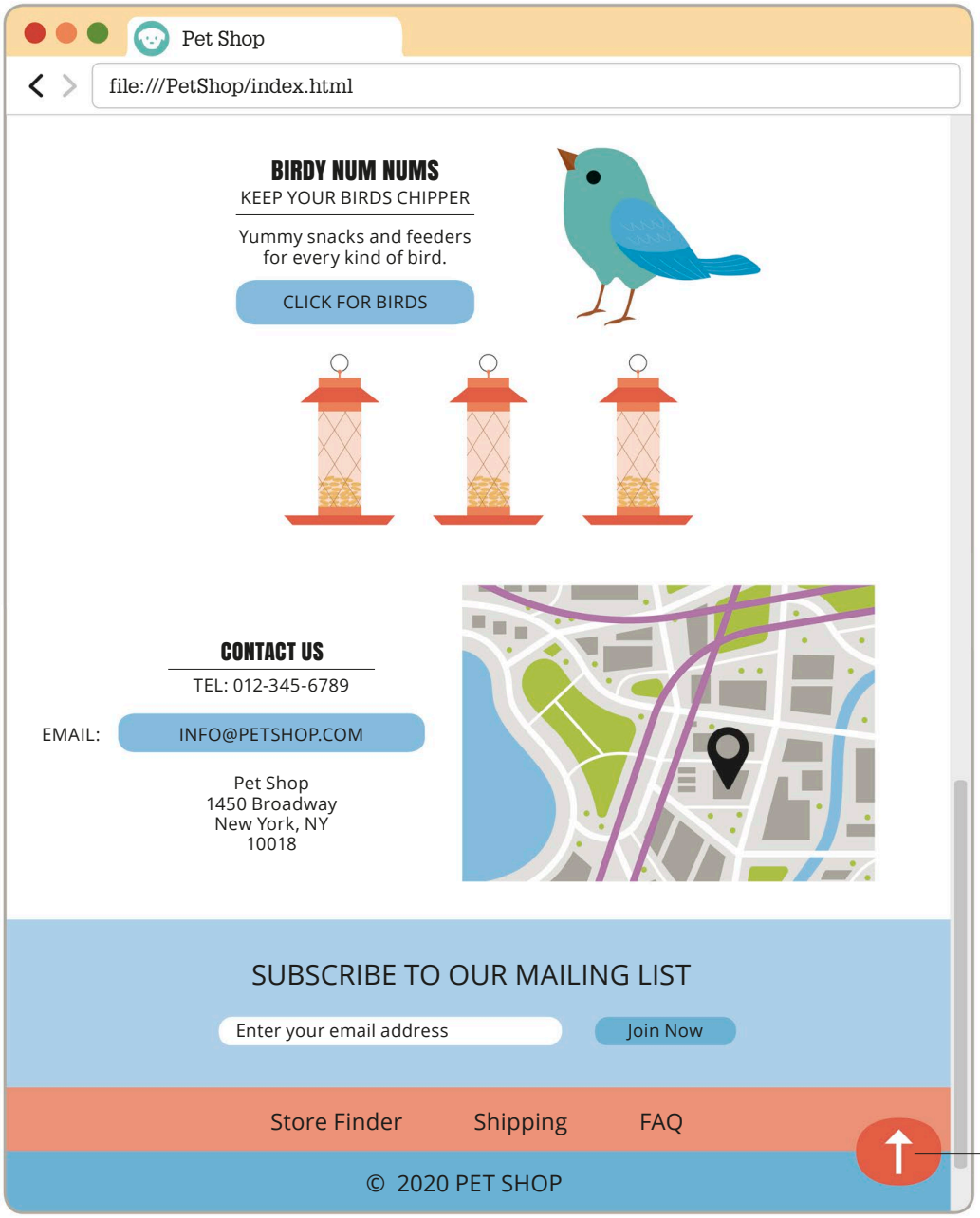
Animates the "html" and "body" elements

Stops when the scroll reaches the top of the page

SAVE

3.6 **VIEW PAGE**

Test the Scroll to top button. Open the browser and enter the URL for the web page into the address bar. In Windows, the URL to the file on your local computer will be "file:///C:/petshop/index.html". On a Mac, the URL will be "file:///Users/[user account name]/PetShop/index.html". The Scroll to top button should be visible now. Click on it and make sure that the page scrolls back up to the top.



The Scroll to top button will now scroll to the top of the page when clicked

4 Managing promotional messages

The next element that needs to be managed is the promotional bar that appears on top of the web page. The promo section in HTML contains four different messages (see p.221). Using JavaScript, you will program the promo bar to cycle through these messages, making them appear one at a time.

4.1 ADD A NEW CUSTOM SCRIPT

You will want the promotional messages to be visible on all pages of the website. Create a new JavaScript file called "common.js". The code in this file will provide functionality for the promotional messages section at the top of the web page. Follow the same steps as earlier to create a new JavaScript file within the scripts folder.

4.2 REFERENCE FILE

Now use a <script> tag to reference the new file "common.js" in the "index.html" file. Add this line of code below the reference for the "home.js" file.

HTML

```
    <script src="scripts/home.js"></script>
    <script src="scripts/common.js"></script>
```

Links the JavaScript file to the HTML file

SAVE

4.3 CREATE A FUNCTION

Inside "common.js", add a new function called `Common()`. This function will act as a class (see pp.282–283) that can be instantiated as a property of the "app" object defined previously. Add an `on document ready()` function below this to instantiate the "Common" class as a property of the "app" object.

JS

```
function Common() {
}
$(document).ready(function () {
    /* Instantiate new Common class */
    app.common = new Common();
});
```

Instantiates the "Common" class as a property

4.4 ADD PROPERTIES

Next, inside the **Common()** function, add a property called "promoBar". This is a JavaScript object (see p.269) that contains all of the variables used by the "Promotional Messages" section to manage itself.

```
function Common() {
    let self = this;
    /* Properties */
    this.promoBar =
        {
            promoItems: null,
            currentItem: 0,
            numberOfItems: 0,
        };
}
```

Creates a reference to the object that can be used later in its methods

"null" indicates that the variable has no value

This is the list of tags with messages

This is the index of the that is currently visible

This is the number of tags with messages

4.5 INITIALIZE THE PROMOTIONAL MESSAGES

Add a method to initialize the "Promotional Messages" section. This method will set the values of the properties contained in the "promoBar" object and will start the loop to show the next message item.

```
            numberOfItems: 0,
        };
    /* Methods */
    this.initialisePromo = function () {
        /* Get all items in promo bar */
        let promoItems = $("#promo > div");
        /* Set values */
        this.promoBar.promoItems = promoItems;
        this.promoBar.numberOfItems = promoItems.length;
        /* Initiate promo loop to show next item */
        this.startDelay();
```

This JQuery selector returns an array of all of the divs inside an element with id="promo"

Returns the number of elements in this array

```
    }
        this.startDelay = function () {
            /* Wait 4 seconds then show the next message */
            setTimeout(function () {
                self.showNextPromoItem()
            }, 4000);
    }
```

This function instructs JavaScript to repeat the call every 4,000 milliseconds

This function will fade out the current promo message and fade in the next message

4.6 CYCLE THROUGH THE PROMOTIONAL MESSAGES

Add a new method below the `initialisePromo()` function. This method will hide the current message and then determine the index of the next message before displaying it onscreen. If the current message is the last item in the message list, then the next message must be the first message in the list. The array index property (see p.268) will be used here. The value of the variable `currentItem` indicates the index number of the displayed message. As this value changes, the message being displayed will also change.

```
    this.showNextPromoItem = function () {
        /* Fade out the current item */
        $(self.promoBar.promoItems).fadeOut("slow").promise().
        done(function () {
            /* Increment current promo item counter */
            if (self.promoBar.currentItem >= (self.promoBar.
            numberOfItems - 1)) {
                /* Reset counter to zero */
                self.promoBar.currentItem = 0;
            } else {
                /* Increase counter by 1 */
                self.promoBar.currentItem++;
            }
            /* Fade in the next item */
```

This command instructs JQuery to extract the array item with the given index number

Ensures that the `currentItem` never exceeds the index number

Cycles through the promotional messages

```
            $(self.promoBar.promoItems).eq(self.promoBar.currentItem).
            fadeIn("slow", function () {
                /* Delay before showing next item */
                self.startDelay();
            });
        });
    }
}
```

Displays the next promotional message in the list

4.7 ADD CALL TO INITIALIZE

Finally, add a call to start cycling through the promotional messages. In "common.js", in the **`on document ready()`** function, add a call to run the **`app.common.initialisePromo()`** function.

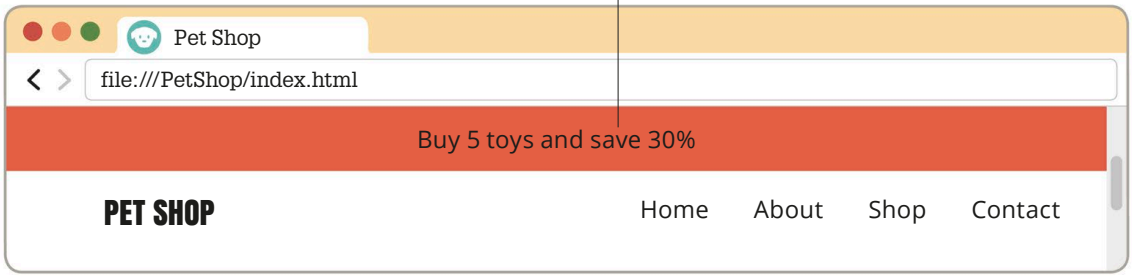
```
$(document).ready(function () {
    /* Instantiate new Common class */
    app.common = new Common();
    /* Initialize the Promo bar */
    app.common.initialisePromo();
});
```

SAVE

4.8 VIEW PAGE

Now go to the browser and refresh the page. Check that the orange promotional messages bar on top of the web page is cycling through the four messages specified in the HTML file.

The promotional messages appear one at a time

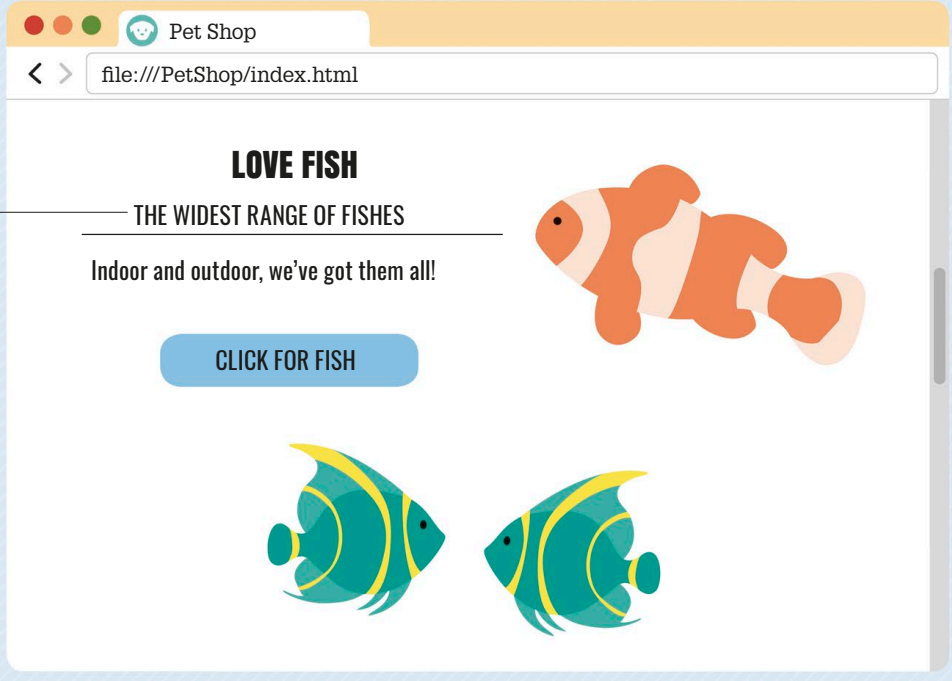


Hacks and tweaks

Explore fonts and icons

To make a website unique and personalized, you can change the way it looks by using different images and fonts. Websites also use icons to distinguish between different sections and pages of the site easily, such as Home, Contact, and Shopping Cart.

This project uses Google Fonts for all of the text elements on the web page. You can explore and try different font options to enhance the look and feel of your web page. You can also search for other favicon images.



You can change the font of any text element on your web page, from the body text to the headings on the website

1 CHOOSE NEW ICONS OR FONTS

You can use the Google Material Icons website to look for more options for the favicon. To change the fonts used on the web page, both the HTML and the CSS files need to be updated. First, choose your new font from the Google Fonts website. Next, go to "index.html" and under the <head> tag, edit the <link> tag that contains the reference to the fonts. Replace the font name with the new font you want to work with.

https://fonts.google.com/ — Link to the Google Fonts website

https://material.io/ — Link to the Google Material Icons website

HTML

```
<link href="https://fonts.googleapis.com/css?
family=Anton|Oswald" rel="stylesheet">
```

The font "Open Sans" has been replaced with "Oswald"

2 **APPLY NEW FONTS**

Now, go to the "global.css" file. Try changing the font of the body elements, as shown here. You will have to edit the "font-family" style definitions within the body selector to do this. Save both of the files and test the program. Your text for the body elements will appear in the new font (see opposite).

```
body {
    margin: 0;
    padding: 0;
    font-family: "Oswald", sans-serif;
    font-size: 15px;
    background-color: white;
    color: #333;
}
```

Font changed from "Open Sans" to "Oswald"

The other elements remain the same

CSS

SAVE

3 **LOAD UPDATED FILE**

At times, even when the code files have been updated, the browser still displays the old web page. This may happen because the browser could be using a saved version of the file rather than downloading it again. Because users may not refresh the page every time they visit the site, you can use a querystring—part of a URL, which assigns values to specified parameters—to force the browser to download the latest version of the file. You can use any querystring, as long as it is different from the previous versions that are already stored. The querystring is added to the <link> tag in the HTML file.

HTML

```
<link href="styles/global.css?v=2" rel="stylesheet" />
```

Add social media

Most websites today find it necessary to advertise their social media accounts so that users can follow them on various platforms. It is possible for you to add social media links on the web page. Start by adding a button to encourage users to follow the website on Twitter.

1 **LOAD THE WIDGET**

To display the Twitter widget on the web page, you will need to load the "widgets.js" script from Twitter. Open the "index.html" file, then add a <script> tag inside the <head> tag to link the widget to the web page.

The new <script> tag can be placed anywhere inside the <head> tag

HTML

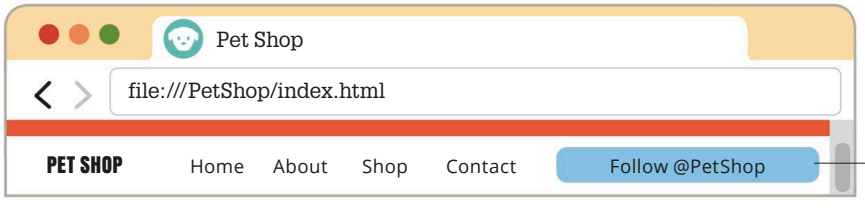
```
<script src="scripts/app.js"></script>
<script src="scripts/common.js"></script>
<script async src="https://platform.twitter.com/widgets.js"
charset="utf-8"></script>
```

2 ADD BUTTON

Now add the Twitter button to the Top Menu. Inside the "topLinks" div unordered list, add a new list item that contains a hyperlink to the Twitter page. If you refresh the browser and view the page onscreen, you will see a new social media button in the Top Menu.

```
...<div id="topLinks">
    <ul>
        <li>
            <a href="/">Home</a>
        </li>
        <li>
            <a href="/">About</a>
        </li>
        <li>
            <a href="/">Shop</a>
        </li>
        <li>
            <a href="/">Contact</a>
        </li>
        <li>
            <a href="https://twitter.com/PetShop"
            class="twitter-follow-button" data-show
            -count="false"> Follow @PetShop</a>
        </li>
    </ul>
</div>
```

Link to the Twitter page



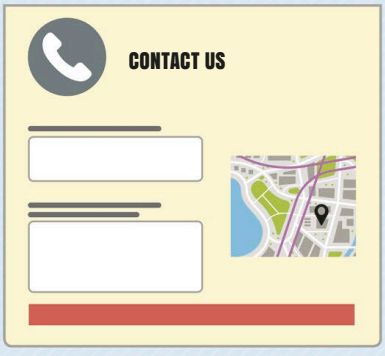
The button appears next to the menu items

Page template

This project only includes code for the website home page. However, other pages, such as About, Shop, and Contact in the Top Menu, are required to make a fully functioning website. In order to create these pages, you will need a template that contains all of the elements that are common to every page of the website. This template will include the meta links to the CSS and JavaScript files, along with all of the common HTML code.

ABOUT

SHOP



CONTACT

1 TEMPLATE HTML FILE

Create a new file called "template.html". Copy the HTML from the home page into this new file. Insert the code shown here into the template file.

Now try creating the other pages of the website. Start by copying the content of "template.html" and pasting it into a new file. Rename the file accordingly—for example, "contact.html". Next, using the instructions given for the home page (see pp.220–233), insert the HTML for the "Contact page" into the placeholder "`## Insert page content here ###`" given in the template code.

```
                <a href="/Shop">SHOP NOW</a>
            </div>
        </div>
    </div>
    <div class="clear spacer v80"></div>
    <div class="wrap">
        ## Insert page content here ###
    </div>
    <div id="footer">
```

Replace the code between the "banner" section and the "footer" section with these lines

Insert HTML for the new page between the div tag with class ="wrap"

2 SERVER-SIDE TEMPLATE OPTIONS

In order to automatically inject the template into each page, you will need to use a server-side language, such as C# MVC or Python Django. In this project, you repeatedly had to include the links to the CSS and JavaScript files into every HTML page on the website. This is obviously difficult to update and maintain, especially if there are lots of pages in the website. You may want to explore the "layout file" concept in C# MVC (*https://www.asp.net/mvc*) and the "template inheritance" feature in Python Django (*https://www.djangoproject.com*) to solve this problem.