Getting Started With Website Design

6 – Finishing Touches

JavaScript

JavaScript is a programming language that is commonly used in the world wide web since it is good for adding interactivity to web pages. Originally called Live Script, the name was changed to JavaScript due to the popularity of the java programming language in the early days of the web, though the Java and JavaScript languages are not connected beyond the similarity in names. JavaScript is widely supported in modern web browsers.

Client-side scripting vs server-side scripting

JavaScript is a client-side language. This means that the programming code runs on the computer of the person visiting your website (the client). It works directly with the HTML and CSS in your website at the user end. Server-side scripting languages such as PHP and Python work on the server end. This means that they work on the server where your website is stored online and run the code before the pages are downloaded to the user's computer.

Exercise 1 – JavaScript and CSS

We will add some simple JavaScript to our website that will be used to hide and display the different team fixtures on the news page. It will do this by modifying the display property when the user clicks on links to change which table is showing and which ones are hidden.

Note If you would like to learn more about the JavaScript language, there are plenty of free tutorials online including W3schools and Sololearn. The JavaScript shown below is to give you an example of what the language can be used for.

Adding IDs for the Tables

We need to provide a name for each fixture on the page so that our combination of CSS and JavaScript can make them hide and display as needed. For this we will use **** tags, which can be used as a container for sections of the page. In this case each heading and table will be enclosed in a span tag, though span tags can be used for as little as a single word or character.

In our page, a user will click on a link. The link will then refer to a JavaScript function which will hide and display parts of the page as needed.

1. Add a span tag with the id *mens* at the start of the mens fixture and add a closing tag after the mens fixture table like in the following example.

```
<span id="mens"><h3>Mens Team</h3>
<table>
        * Table html in between *
</table></span>
```

2. Add similar span tags with ids for the other three fixtures. Remember to add the closing **** tag after each table.

```
<span id="womens"><h3>Womens Team</h3>

<span id="teen"><h3>Teen Team</h3>

<span id="kids"><h3>Kids Team</h3>
```

Creating the .js File

Like CSS, JavaScript can be placed inside tags, inside the head section of the page, or in a separate file depending on where it needs to be used. Although we will only be using our script in a single page, we will put the script in a separate file with a .js extension. This will allow us to use it in more than one page later if needed.

3. Using your editor, create a new blank file. Save it with the filename **fixtures.js** inside your **scripts** folder. You might need to set the file type in your editor to JavaScript when you save it.

Write the JavaScript

Inside the js file, write the following scripts. A link on the News page will activate one of these functions. The function will then hide or display the necessary fixture table. It does this by referring to the id of certain elements on the page and then setting the CSS display property of that element to either block or none (hidden).

Note We could do this more efficiently by writing a single script that checks to see which link has been clicked to decide which sections to hide and display. The separate functions as we have them here can demonstrate how the JavaScript works without getting too deep into the language.

```
function displayMen() {
        var menLink = document.getElementById('mens');
        var womenLink = document.getElementById('womens');
        var teenLink = document.getElementById('teen');
        var kidsLink = document.getElementById('kids');
                menLink.style.display = 'block';
                womenLink.style.display = 'none';
                teenLink.style.display = 'none';
                kidsLink.style.display = 'none';
}
```

```
function displayWomen() {
        var menLink = document.getElementById('mens');
        var womenLink = document.getElementById('womens');
        var teenLink = document.getElementById('teen');
        var kidsLink = document.getElementById('kids');
                menLink.style.display = 'none';
                womenLink.style.display = 'block';
                teenLink.style.display = 'none';
                kidsLink.style.display = 'none';
}

function displayTeen() {
        var menLink = document.getElementById('mens');
        var womenLink = document.getElementById('womens');
        var teenLink = document.getElementById('teen');
        var kidsLink = document.getElementById('kids');
                menLink.style.display = 'none';
                womenLink.style.display = 'none';
                teenLink.style.display = 'block';
                kidsLink.style.display = 'none';
}

function displayKids() {
        var menLink = document.getElementById('mens');
        var womenLink = document.getElementById('womens');
        var teenLink = document.getElementById('teen');
        var kidsLink = document.getElementById('kids');
                menLink.style.display = 'none';
                womenLink.style.display = 'none';
                teenLink.style.display = 'none';
                kidsLink.style.display = 'block';
}
```

4. Save the changes to the js file

Add our Fixture Links

We will add links above the fixtures that the user can click to show the desired fixture.

Modify the text above the fixtures to the following.

Original HTML

```
<p>Check the information below to see when your team's next games will be played.</p>
```

New HTML

```
<p>Click on a team name below to see when that team's next games will be played.</p>

<ul>

    <li><a class="fixture-link" onclick="displayMen()">Mens Team</a></li>

    <li><a class="fixture-link" onclick="displayWomen()">Womens Team</a></li>

    <li><a class="fixture-link" onclick="displayTeen()">Teen Team</a></li>

    <li><a class="fixture-link" onclick="displayKids()">Kids Team</a></li>

</ul>
```

Since these **a** tags are not referring to a particular address (no **href** attribute) and are used to trigger a script, they might not appear as normal links in a browser. Each of these links refers to a CSS class (which we will create in a moment) that formats these links to look more like a typical href link.

Each link includes an attribute that refers to one of the functions we have written when the link is clicked.

Hide the Fixtures

Add the following rules in your CSS file (in the main section).

5. These will set each of the fixture sections to be hidden when the page loads so that they will only display when clicking a link runs the appropriate script.

```
#mens {
  display: none;
}
#womens {
  display: none;
}
#teen {
  display: none;
}
#kids {
  display: none;
}
```

6. Add the following CSS rule to format the links.

```
.fixture-link {
  color: blue;
  cursor: pointer;
}
```

Link to the .js File

7. Add a line in the head section of your news.html page like the following. This will load the .js file when the html loads so that our functions are ready for use.

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

8. Save all files and preview your page so that you can test the show / hide fixture links.

When the page loads, each of the fixtures should be hidden. Then you can click the links to display each fixture.

Add an HTML Form

Forms are used to allow a user to either send or retrieve information through your web page. For example, a form could be used so that a visitor can send information to an email address or send information for an order. A form could also retrieve information such as when a user performs a search. Forms are often used in combination with a database on your website (such as a membership database when a student uses a form to log on or a products database when a form is used to search and order products). Forms are also commonly used with a scripting language which determines what will happen with a form is used.

In our website, we will create a simple form that will send information to an email address.

At the bottom of our Memberships page, we left a comment for a place where we intended to add a form later. Now is the time to add that form.

Exercise 2 – Expression of Interest Form

1. Open the memberships page and locate the following HTML near the bottom.

```
<h2 id="expression-of-interest">Expression of Interest</h2>

<p>If you are interested in joining <abbr title="Morley Hooperoos Basketball Club">MHBC</abbr>, complete the following form and one of our committee members will contact you.</p>

<p><!-- a form that can be filled in will be added here later --></p>
```

2. Delete the line that is shaded in this example as this is where we will place the form.

We will first create a table to line up the form controls and the labels next to each control.

3. Add the following HTML. This will create a table with a column for labels, and a column where the form controls such as text boxes will be placed. Some rows have a single merged cell instead of two cells. A final row with a merged cell will be used for the submit and clear buttons.

```
<table>
    <tr>
        <td>Name: </td><td></td>
    </tr>
    <tr>
        <td>Email: </td><td></td>
    </tr>
    <tr>
        <td>Phone: </td><td></td>
    </tr>
    <tr>
        <td colspan="2">Have you played for a basketball team before? </td>
    </tr>
    <tr>
```

```
            <td>Skill level: </td><td></td>
        </tr>
        <tr>
            <td>Team you wish to play on: </td><td></td>
        </tr>
        <tr>
            <td>Other comments: </td><td></td>
    </tr>
    <tr>
        <td colspan="2"></td>
    </tr>
</table>
```

4. Modify the first row of the table to look like the following:

```
<tr>
    <td><label for="interest-name">Name: </label></td>
    <td><input type="text" id=" interest-name" size="40"></td>
</tr>
```

The **<input>** tag is one of the more common form controls and can display differently depending on the **type** attribute. In this case, the type attribute is used to make the input display as a text box. The **size** attribute determines how wide the box will appear while the **id** tag gives the control a name.

The **<label>** tag has been added to associate the text in the first column with the form control of the same name. This won't make any difference to how the form looks but it can help screen readers to know which text is related to which form control.

Add form controls to the next two table rows.

```
        <tr>
            <td><label for="interest-email">Email: </label></td>
            <td><input type="email" id="interest-email" placeholder="email@address.com"
        size="40"></td>
        </tr>
        <tr>
            <td><label for="interest-phone">Phone: </label></td>
            <td><input type="tel" id="interest-phone" size="40"></td>
        </tr>
```

5. The next row will use a checkbox type input.

```
<tr>
    <td colspan="2"><label for="interest-played">Have you played for a basketball team before?</label> <input type="checkbox" id="interest-played"></td>
</tr>
```

6. The next row will use radio buttons. This makes a group of related controls (each one needs the same name) so that one can be selected at a time. Notice that the first radio button has a **checked** attribute meaning it will already be selected when the page loads.

```
<tr>
    <td>Skill level: </td>
    <td><label><input name="interest-skill" type="radio" id="beginner" value="beginner" checked="checked">beginner</label>
    <label><input type="radio" name="interest-skill" value="medium" id="medium">medium</label>
    <label><input type="radio" name="interest-skill" value="advanced" id="advanced">advanced</label></td>
</tr>
```

7. The next table row will contain an option list that creates a drop-down list of choices for the user to select from. The text inside the option tag is what appears in the list, while the value tag determines what response is sent with the form when an option is selected.

```
<tr>
    <td><label for="interest-team">Team you wish to play on: </label></td>
    <td><select name="Team" id="interest-team">
    <option value="None">No team selected</option>
    <option value="Mens Team">Mens Team</option>
    <option value="Womens Team">Womens Team</option>
    <option value="Teen Team">Teen Team</option>
    <option value="Kids Team">Kids Team</option>
    </select></td>
</tr>
```

8. The comments row will contain a larger text box that is intended for longer amounts of text.

```
<tr>
    <td><label for="interest-comments">Other comments: </label></td>
    <td><textarea name="Comments" cols="35" rows="3" id="interest-
    comments"></textarea></td>
</tr>
```

9. The last row will contain buttons to submit and clear the form

```
<tr>
    <td colspan="2" style="text-align: center">
    <input type="submit" name="submit" id="submit" value="Submit Form">
    <input type="reset" name="reset" id="reset" value="Clear Form"></td>
</tr>
```

If you preview the page and test the form now, you will be able to test the form controls such as text boxes and option list. The buttons won't do anything though since we haven't yet specified what is supposed to happen to the completed form. This is done by enclosing the whole form inside a **<form>** tag.

A form tag will have a few properties. One of the most important one is the **method** property which specifies whether the form will be sending information (post) or retrieving information (get). A form also needs to have an **action** property. This will determine what happens when the form is submitted. Often it is a reference to a server-side script with programming code that determines what to do with the form (client-side scripts such as JavaScript are not intended for this purpose largely for security reasons).

In our example, rather than use a server-side script, we will simply set the action to send the form results directly to an email address. This is much simpler than using a server-side script, though a server-side script can be used to make the results from the form look a lot more neat when they arrive at the destination email address, as well as allowing for the generation of a "thank-you" page that can appear after the form is submitted.

10. Add a form tag before the start of your table. Put your own email address inside the action property so you can test the form.

```
<form action="mailto:your@email.com" method="post" name="Expression of Interest"
id="interest">
<table>
```

11. Add a closing form tag after the end of the table.

```
</table>
</form>
```

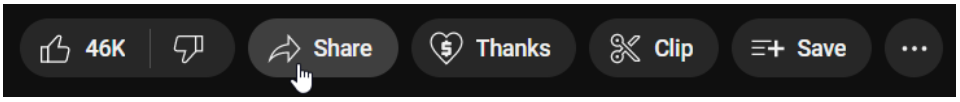
If you save and test the form now, clicking the clear button will reset the form, while clicking the submit button will open your computer's email application with a new email including details from the form.

Exercise 3 – Add a Video using a Google API

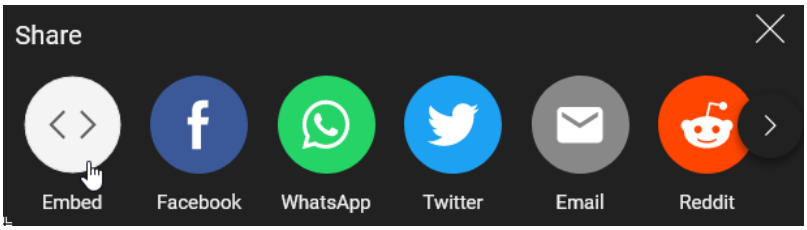
In the last part of the Links page, we will make use of an API (Application Programming Interface). This is a way for your website to make use of computer programs that someone else has created. In this case, we will make use of google's Youtube API to place a video on our page.

You can find your own video to insert using the steps below or you can use the same video shown in the example further down.

1. Visit **Youtube** and find a suitable video.
2. Find the **Share** button beneath the video and click on it.



3. Click **Embed** in the share options.



Youtube will now provide some code that can be placed inside your own HTML wherever you would like this video to appear in your site.

4. Click the **Copy** button in the bottom corner.
5. In your HTML, Paste the copied code right below the Training videos heading in the links page.

```
<h2>Training Videos</h2>
<iframe width="560" height="315" src="https://www.youtube.com/embed/XIHqHouUHoY?si=Kcl-vWXTFqr8thB0" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" allowfullscreen></iframe>
</article>
```

You could use a similar method to add a Google map to one of your pages.