Higher Computing Science

Web Design & Development

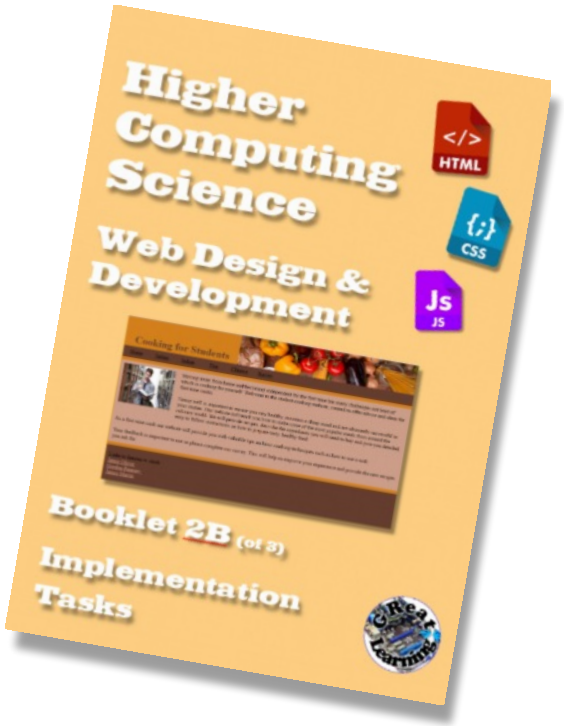
Booklet 2A (of 3)

Implementation Examples

Contents

Note that Tasks 1 to 12 are supplied separately in:

Booklet 2B - Implementation Tasks.



What's Included in this Booklet?

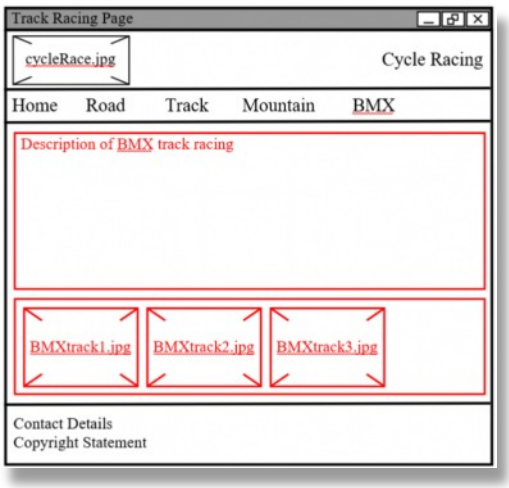
This booklet has been created to cover the web design and development area of the Scottish Higher Computing Science course. This is booklet 2 of 3.

While working through the three booklets you shall follow the development phases shown below:

Booklet 1	
Analysis	Identify the purpose of a website, the target audience, what these end-users require and what will be required to build the website.
Design	Consider the layout of content, navigation structure and user interface required to build an effective website.

Booklets 2A and 2B	
Implementation	Code the website using: • HTML - to define the content of the website. • CSS - to control the look of the content. • JavaScript - to introduce an element of interactivity with the website.

Booklet 3	
Testing	Consider and implement strategies to test the finished website.
Evaluation	Evaluate if the website is useable and fit for purpose.



Booklet 2 provides example implementations for the Bicycle Racing website pages designed in booklet 1.

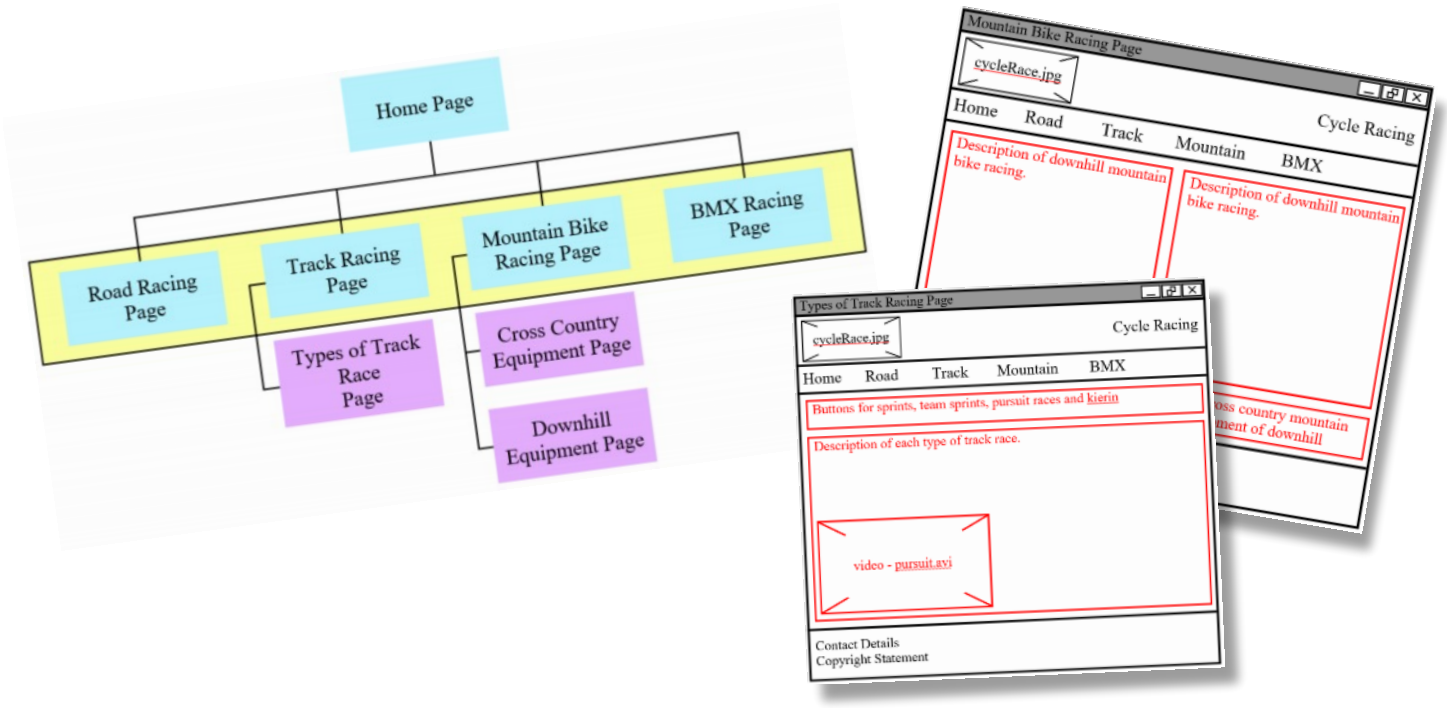
As you read through the booklet you will be asked to complete similar implementation tasks using the Student Cooking scenario seen in the booklet 1 analysis and design tasks.

The examples in this booklet should be used to complete the implementation of the Student Cooking website.

All the implementation tasks will be provided in separate task booklet. The tasks will be available from your teacher/lecturer and may be used in class or at home.

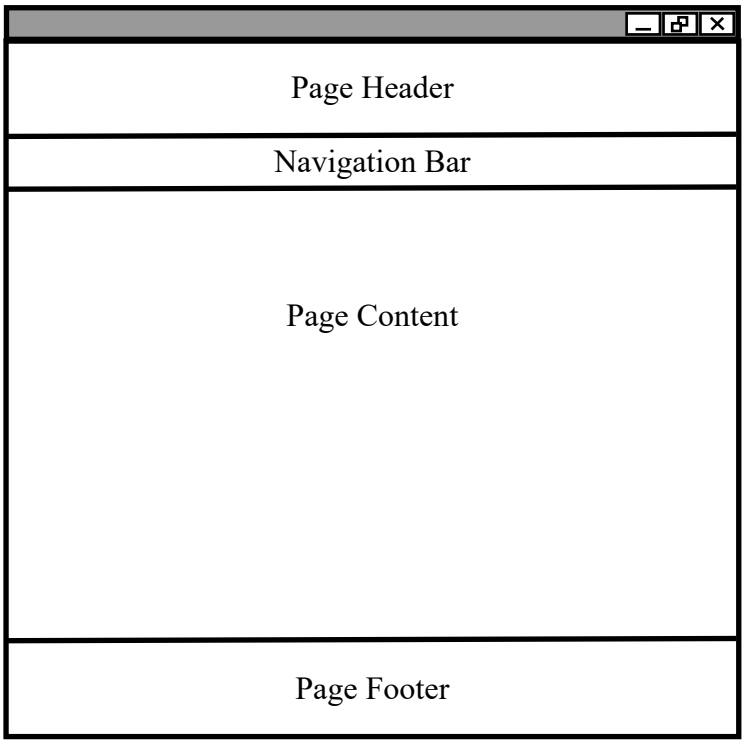
From Design to Reality

Once a website design is agreed with the client, programmers use the website structure and wireframe designs for reference when implementing (or coding) the website.



Stage 1 - Website Layout

A natural starting point is to code the general layout and content that will appear on every page of the website. The website layout was identified in the wireframe shown below.



Coding the Website Layout

The pages in the cycling website have four main areas:

Header	<header>	This contains a banner for each page which includes the name of the website and a graphic.
Navigation	<nav>	This contains hyperlinks to the main sub-topics of the website.
Content	<main>	The actual content for each page.
Footer	<footer>	Additional information such as contact details or copyright/legal declarations that should appear on every page.

Programmers often use template files as a starting point when creating different websites. A template file, suitable for the cycling website, may look like this.

```
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5  <title>   </title>
6  <link rel="stylesheet" type="text/css" href="../CSS/  .css">
7  </head>
8
9  <body>
10
11 <!-- Page Header -->
12 <header>
13 <h1></h1>
14 <img class="imageBanner" src="../Images/  ">
15 </header>
16
17 <!-- Navigation Bar -->
18 <nav>
19 <ul>
20 <li><a href=" .html">    </a></li>
21 <li><a href=" .html">    </a></li>
22 <li><a href=" .html">    </a></li>
23 <li><a href=" .html">    </a></li>
24 <li><a href=" .html">    </a></li>
25 </ul>
26 </nav>
27
28 <!-- The main content of the page -->
29 <main>
30
31 </main>
32
33 <!-- Page Footer -->
34 <footer>
35
36 </footer>
37
38 </body>
39 </html>
```

A title can be added for each page.

Space has been left to add the name of the CSS file the website will use.

The website name can be inserted into the <h1> element.

The file name of the website's banner image can be inserted here.

An unordered list <ul> has been set up to include the hyperlinks <a> to the website's sub-pages.

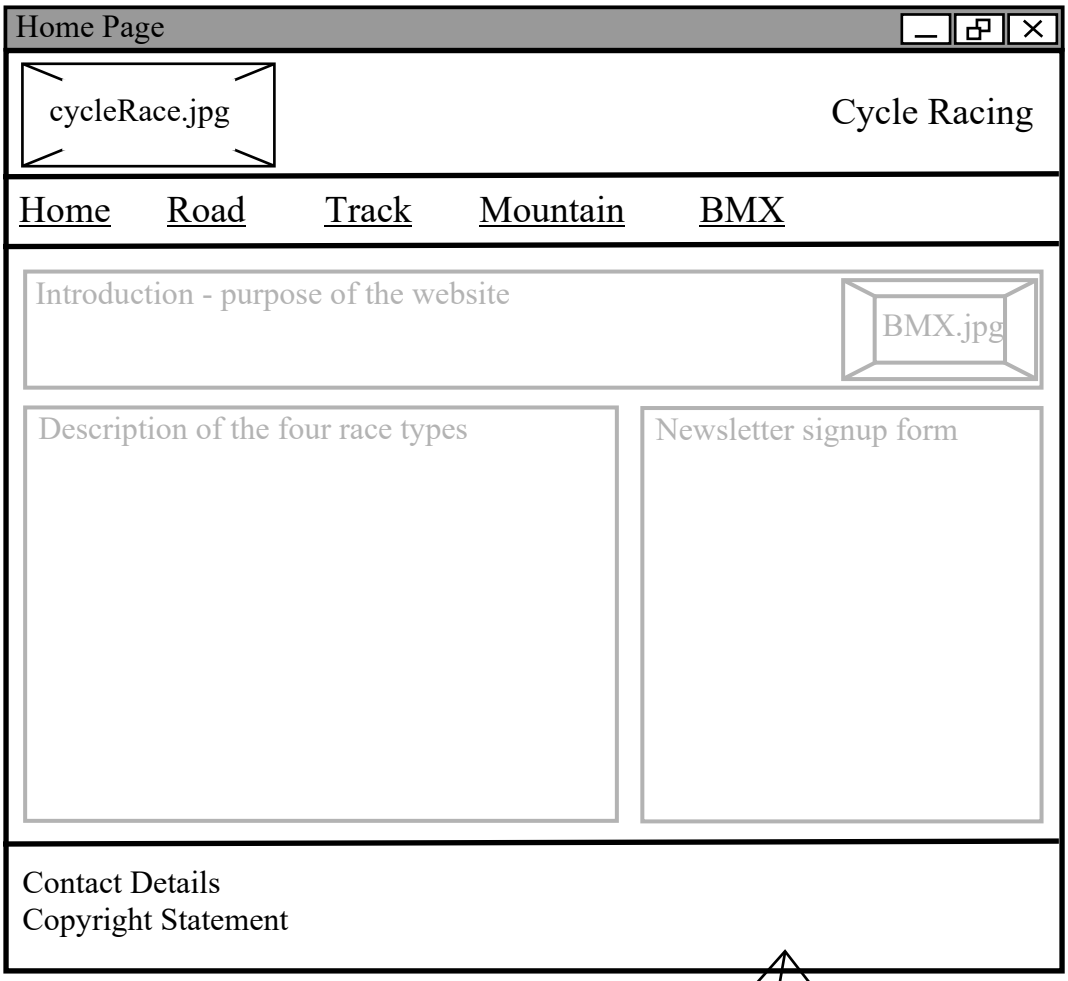
The names of the sub-pages can be inserted into the list items <li>.

The content for each page can be added to the <main> element.

The footer information can be added here.

Coding the Consistent Website Content

Three of the areas (header, nav, main) in the template file will have exactly the same content on every page. This consistent content is shown in the Home Page wireframe.



<header>

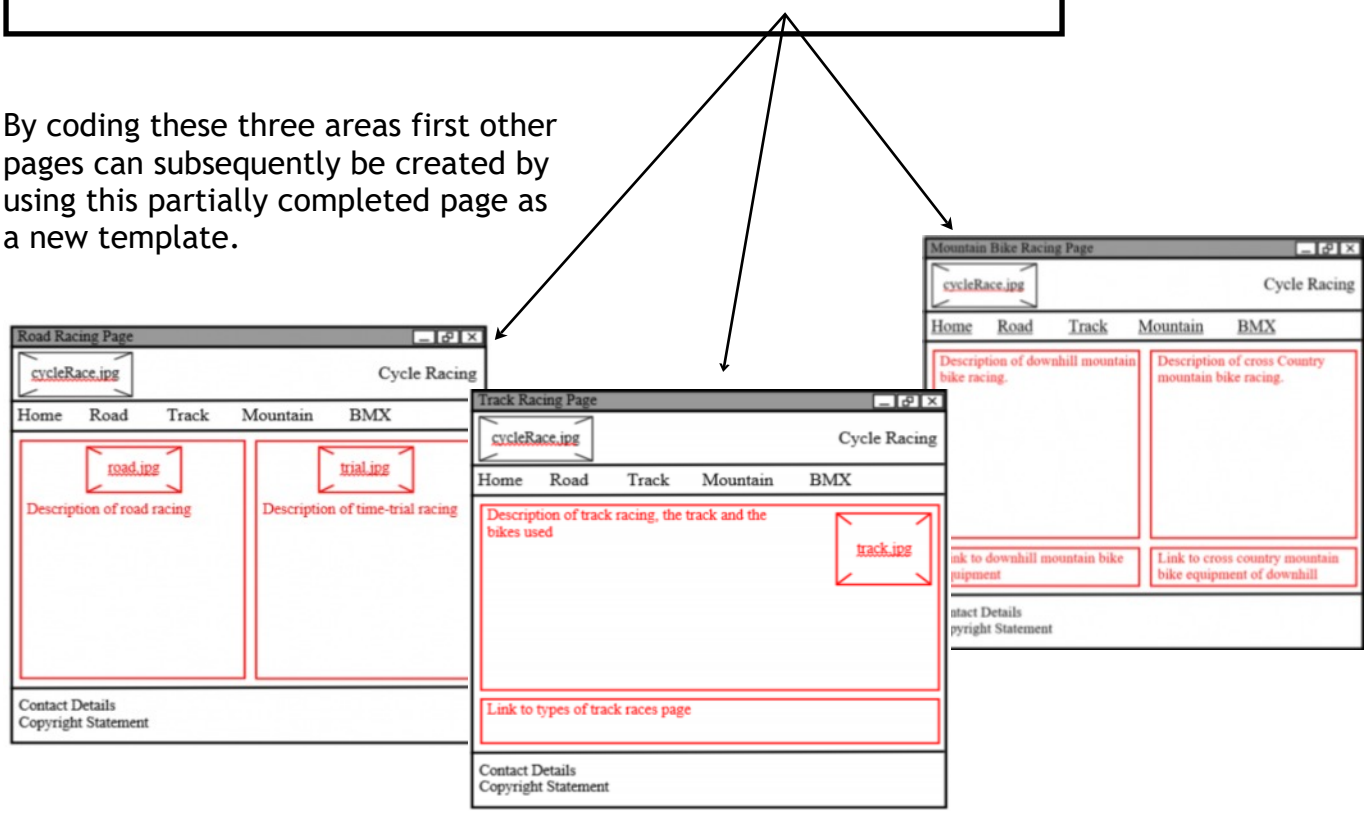
</header>

<nav> </nav>

<footer>

</footer>

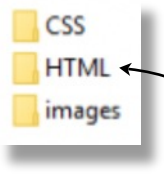
By coding these three areas first other pages can subsequently be created by using this partially completed page as a new template.



The following code shows the template file once it has been completed to include the <title>, <link>, <header>, <nav> and <footer> content. Note that this has been set as the home page.

```
<!DOCTYPE html>
<html>

<head>
<title>Home Page</title>
<link rel="stylesheet" type="text/css" href="../CSS/styles.css">
</head>

<body>

<!-- Page Header -->
<header>
<h1>CYCLE RACING</h1>
<img class="imageBanner" src="../images/cycleRaceBanner.jpg">
</header>

<!-- Navigation Bar -->
<nav>
<ul>
<li><a href="home.html"> Home </a></li>
<li><a href="road.html"> Road </a></li>
<li><a href="track.html"> Track </a></li>
<li><a href="mountain.html"> Mountain </a></li>
<li><a href="bmx.html"> BMX </a></li>
</ul>
</nav>

<!-- The main content of the page -->
<main>

</main>

<!-- Page Footer -->
<footer>
<p> Email us at: getintocycleracing@lovecycling.org <br>
All information copyright lovecycling.org </p>
</footer>
```

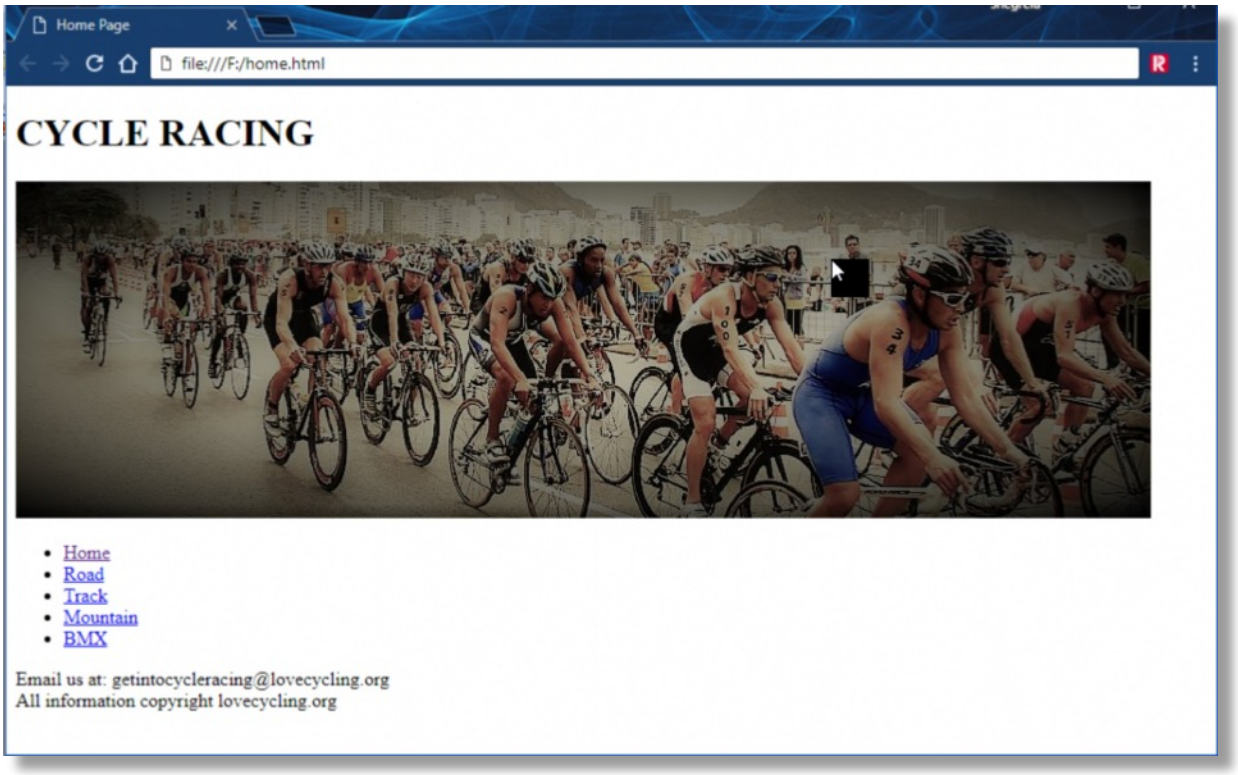
The website will be saved with separate folders for HTML, images and CSS.



This can now be saved in the HTML folder as “home.html”.

Each time you complete some web coding you should ensure that the content you've added displays correctly.

When the cycle racing home page is view in a browser it look like this.



Note that as no CSS styles have been added yet the content is simply stacked down the page with no colour, spacing or sizes. Don't worry about this at this stage. We will style the pages later using CSS declarations.

Creating Additional Pages

To complete the setting up of the website this file can now be saved four more times to the HTML folder. To ensure the web pages have no errors the following rules should be obeyed each time a new file is saved:

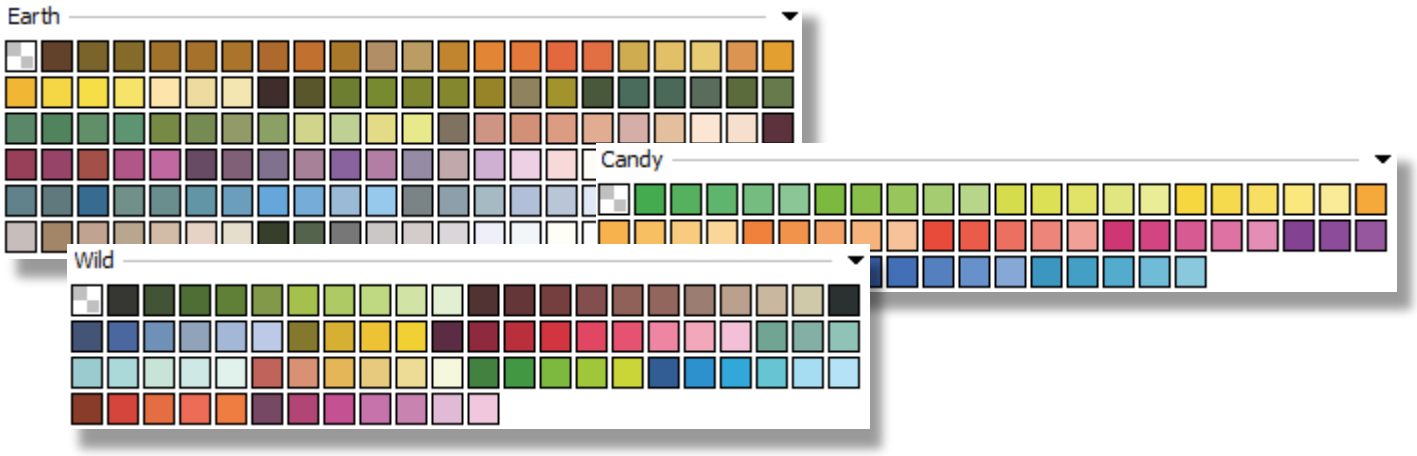
- Change the <title> element to match the new page.
- Use "save as" to change the name of the file.
- The saved names must match those already stated in the template page.

```
20  <li><a href="home.html">Home</a></li>
21  <li><a href="road.html">Road</a></li>
22  <li><a href="track.html">Track</a></li>
23  <li><a href="mountain.html">Mountain</a></li>
24  <li><a href="bmx.html">BMX</a></li>
```

You should now complete "Implementation Task 1 - Setting up the Pages for the Student Cooking Website". This can be found in the separate Implementation Tasks booklet.

Stage 2 - Colouring the Main Page Areas

Professional web developers often work using colour pallets when designing and implementing websites. These are sets of colours which naturally compliment each other.



There are a variety of website or graphics applications that you can use to research colour pallets to use. For example:

- 50 Gorgeous Colour Schemes for Web Design
- Trendy Web Colour Pallets

Colours are implemented as follows:

CSS Declaration			
Property	**Values**	**Example Declarations**	**Explanations**

`background-color`	`name`
	`rgb`
	`hex`
		`body {background-color: lightBlue}`
		
		`ul {background-color: rgb(255,99,71)}`
		
		`h1 {background-color: #ff6347}`
			Colours can simply be named. For a list of colours visit: https://www.w3schools.com/cssref/css_colors.asp
			
			A colour can be given as a red, green, blue (rgb) value. Each component colour is stated as a value ranging from 0 to 255.
			
			A hexadecimal number is simply an alternative way of writing an rgb value. Pair of digits identifies one component colour.

`color`	`name`
	`rgb`
	`hex`
		`p {color: #fb4}`	Color on its own is used to colour text within a page.
			Note that a hex value can be abrieviated if each pair of values is the same.
			#fb4 = #ffbb44

The colour pallet selected for the cycling website is shown on the right.

The important elements of the web pages can be allocated colours by adding the following declarations to the CSS file (styles.css) found in the CSS folder.

```
/* Background Colours */
header {background-color:#edd9c0}
footer {background-color:#edd9c0}
nav {background-color:#7d4627}
main {background-color:#a8b6bf}
div {background-color: #c9d8c5}
section {background-color: #c9d8c5}
```

Note that two of the colours in the page are used in more than element.

Grouping Selectors

Where declarations repeat grouping selectors can be used to reduce the amount of code.

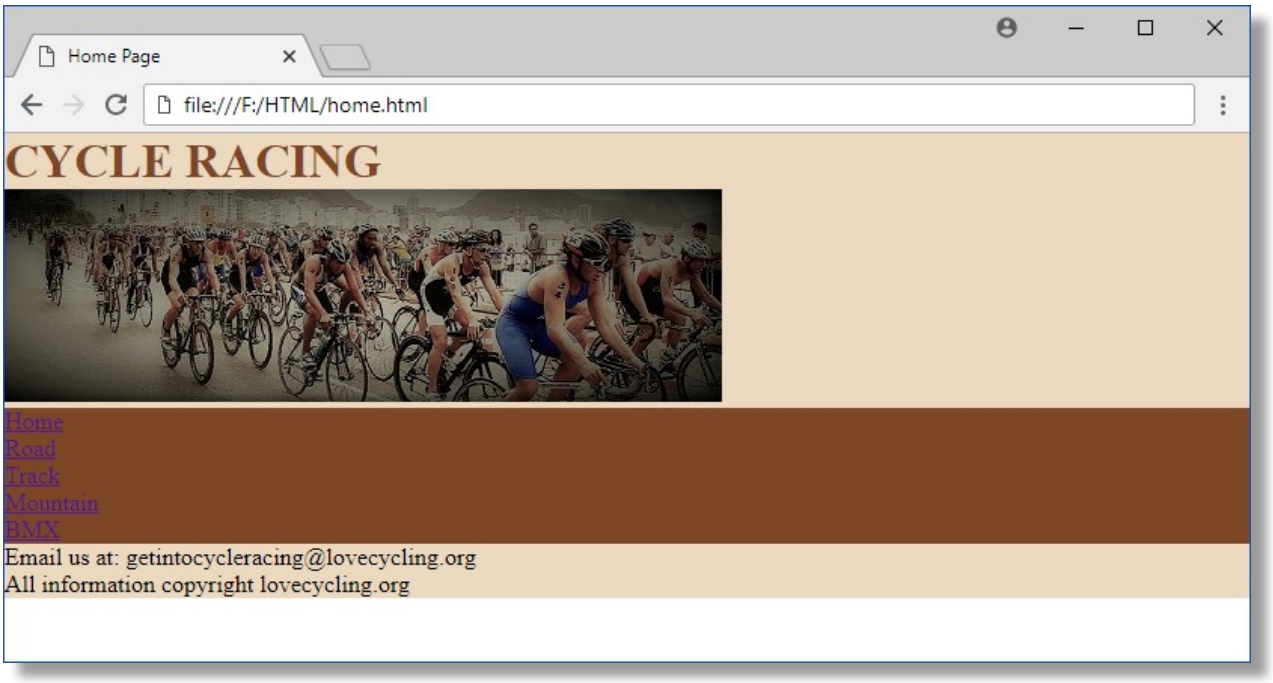
In the above code header, footer and div, section can be grouped as shown below.

```
/* Background Colours */
header, footer {background-color:#edd9c0}
nav {background-color:#7d4627}
main {background-color:#a8b6bf}
div, section {background-color: #c9d8c5}
```

The main heading used in the website is given a text colour that is also from the chosen colour pallet.

```
/* Text Properties */
h1 {color:#7d4627}
```

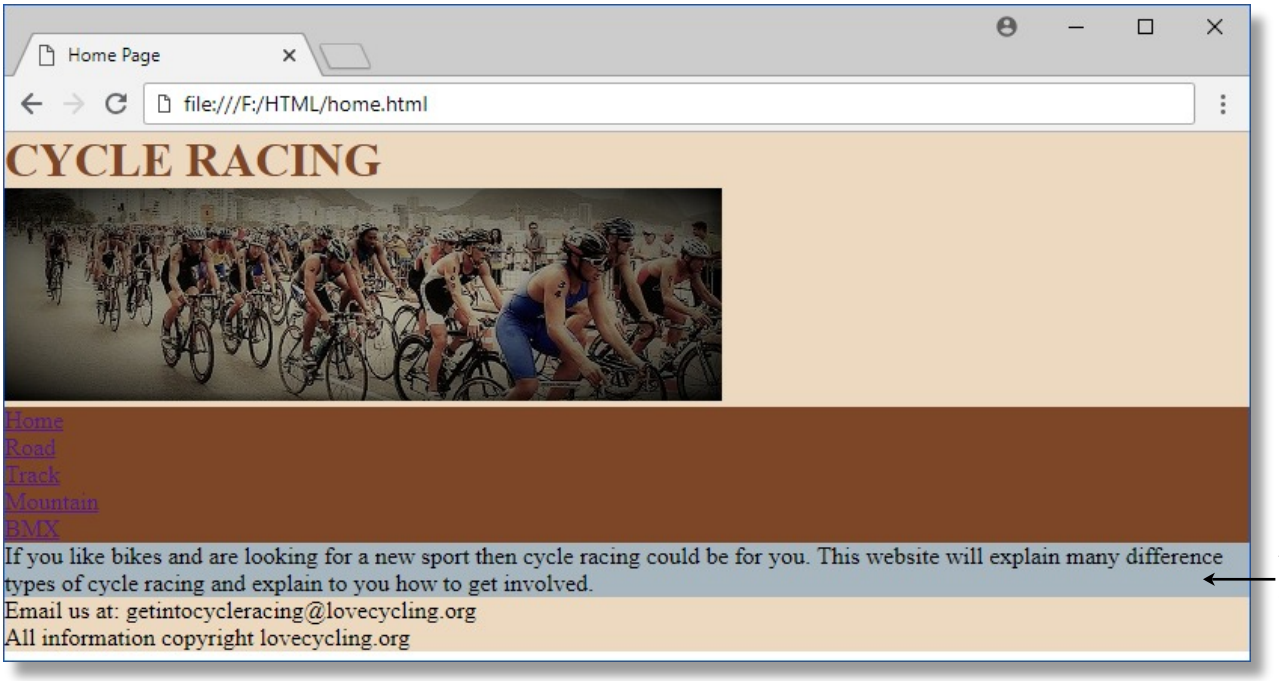
When tested the cycling page now has the following colours.



Note that the <main> element of the above page is not visible so it is impossible to test if the colour it has been coded correctly. This is because the element has no content.

The following code is added to the <main> element of the home page.

```
<p> If you like bikes and are looking for a new sport then cycle racing
could be for you.  This website will explain many difference types of
cycle racing and explain to you how to get involved. </p>
```



You should now complete "Implementation Task 2 - Setting up Colours on the Student Cooking Website". This can be found in the separate Implementation Tasks booklet.

Stage 3 - Positioning, Sizes and Spacing

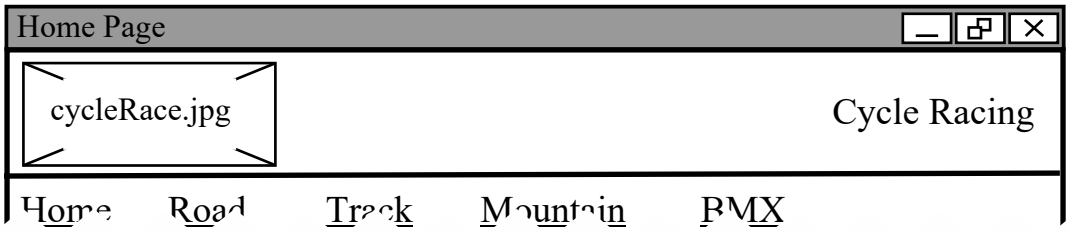
After the initial layout of the pages has been coded developers will often set the position, size and spacing of the main page elements. This allows the developer to ensure that the subsequent content of the pages will fit correctly in the allocated space.

Positioning

When browsers interpret HTML code they display elements in order they appear in the code.

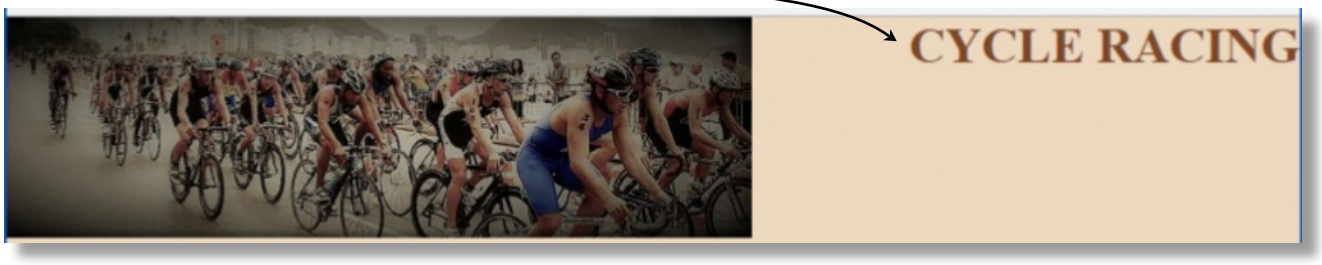
```
12  <header>
13    <h1>CYCLE RACING</h1>
14    <img class="imageBanner" src="../images/cycleRaceBanner.jpg">
15  </header>
```

The wireframe for the each page of the cycle racing website shows that the website name and the banner image are to be positioned side-by-side.



This can be achieved by "floating" the <h1> element to the right of the page using a CSS declaration.

```
25  /* Positioning (Float, Display and Clear) */
26  h1 {float:right}
```

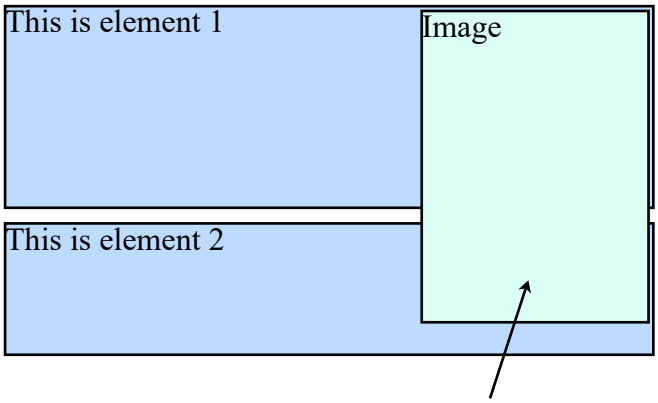


Problems Associated with Positioning Content

Problem 1

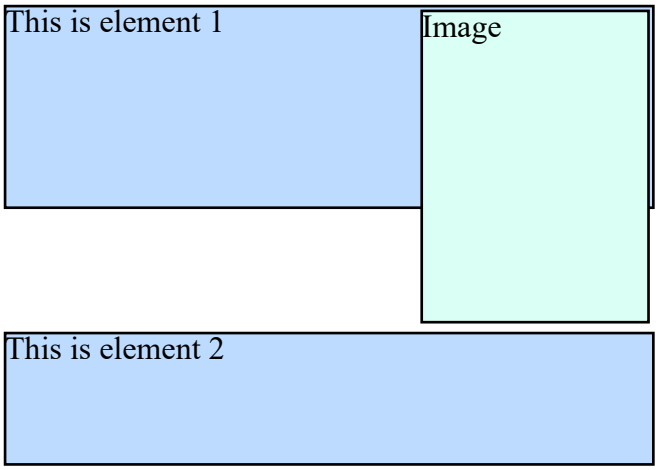
When an object is floated it may affect elements other than the ones it was intended to interact with.

In the example shown, an image is floated to the right of element 1.



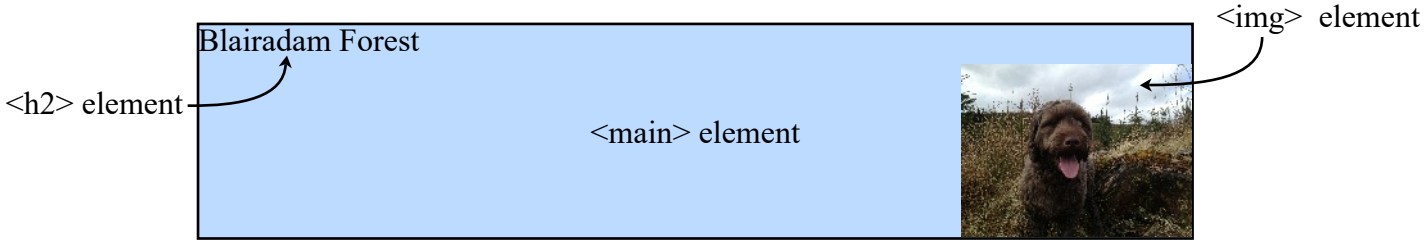
However, as the image is taller than the content of element 1 the image also overlaps part of element 2.

This can be corrected by adding a "clear" declaration to element 2. Clear cancels the effect of the previous float declaration ensuring that element 2 appears after the image as shown below.

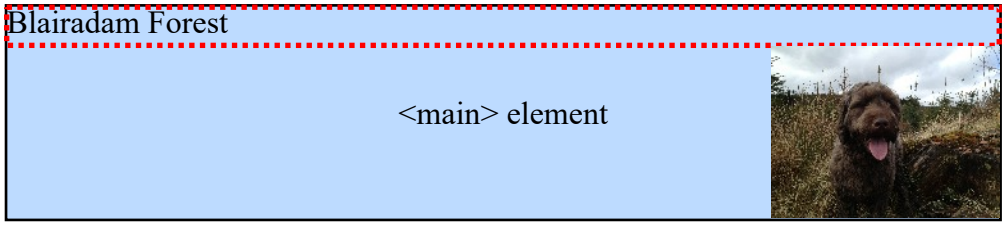


Problem 2

Sometimes when an element is floated it fails to align side-by-side with another element. In the example below the img has been floated to the right and should be inline with the h2 element. Instead the image is sitting below the text.



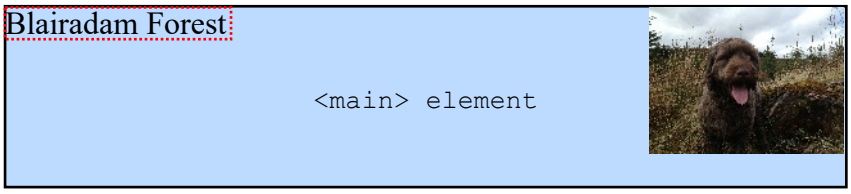
This occurs because the h2 element is actually taking up the full width of the <main> element forcing the image to float below it.



The CSS "display" property can be used to control the width of an element.

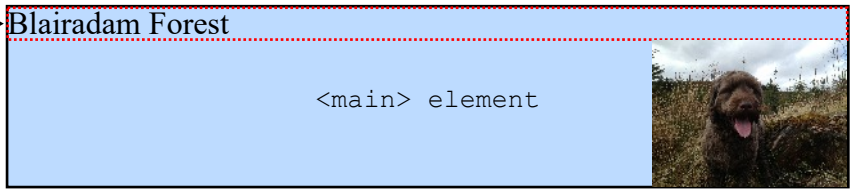
`h2 {display:inline}`

The `inline` value ensures the h2 element takes up only as much space as it requires (in other words the width of the text)



`h2 {display:block}`

The `block` value ensures the element takes up the full width of its container element. In this case the <main> element.



Some elements automatically default to `display:block` unless coded to do otherwise.

`<div>` `<h1>-<h6>` `<p>` `<form>` `<header>` `<footer>` `<section>`

Some other elements default to `display:inline`.

`<a>` `<img>`

Float, clear and display declarations are implemented as follows:

CSS Declaration			
Property	**Values**	**Example Declarations**	**Explanations**

`float`	`left, right`	`#siteIcon {float:left}`
		
		`.newsP {float:right}`
			As `float` is often applied to individual page objects it's common to use it along with an ID(#).
			
			Floats a paragraph of text to the right of its container element.

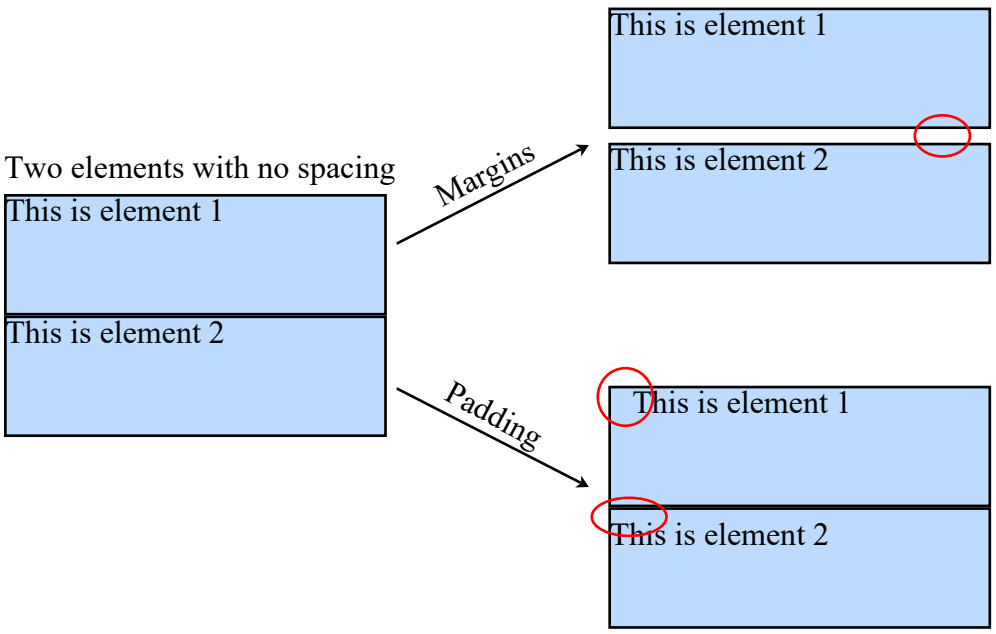
`clear`	`both`	`nav, main, footer {clear:both}`	It would be unusual to have content overlapping between the main areas of a web page. A simple way to prevent this is to add a clear statement to all of the main page elements.

`display`	`block, inline`	`p {display:inline}`
		
		`a {display:block}`
			The paragraph width will be limited to the width of the text.
			
			All hyperlinks would display on a line of their own.

You should now complete "Implementation Task 3 - Floating Images on the Student Cooking Website". This can be found in the separate Implementation Tasks booklet.

Spacing

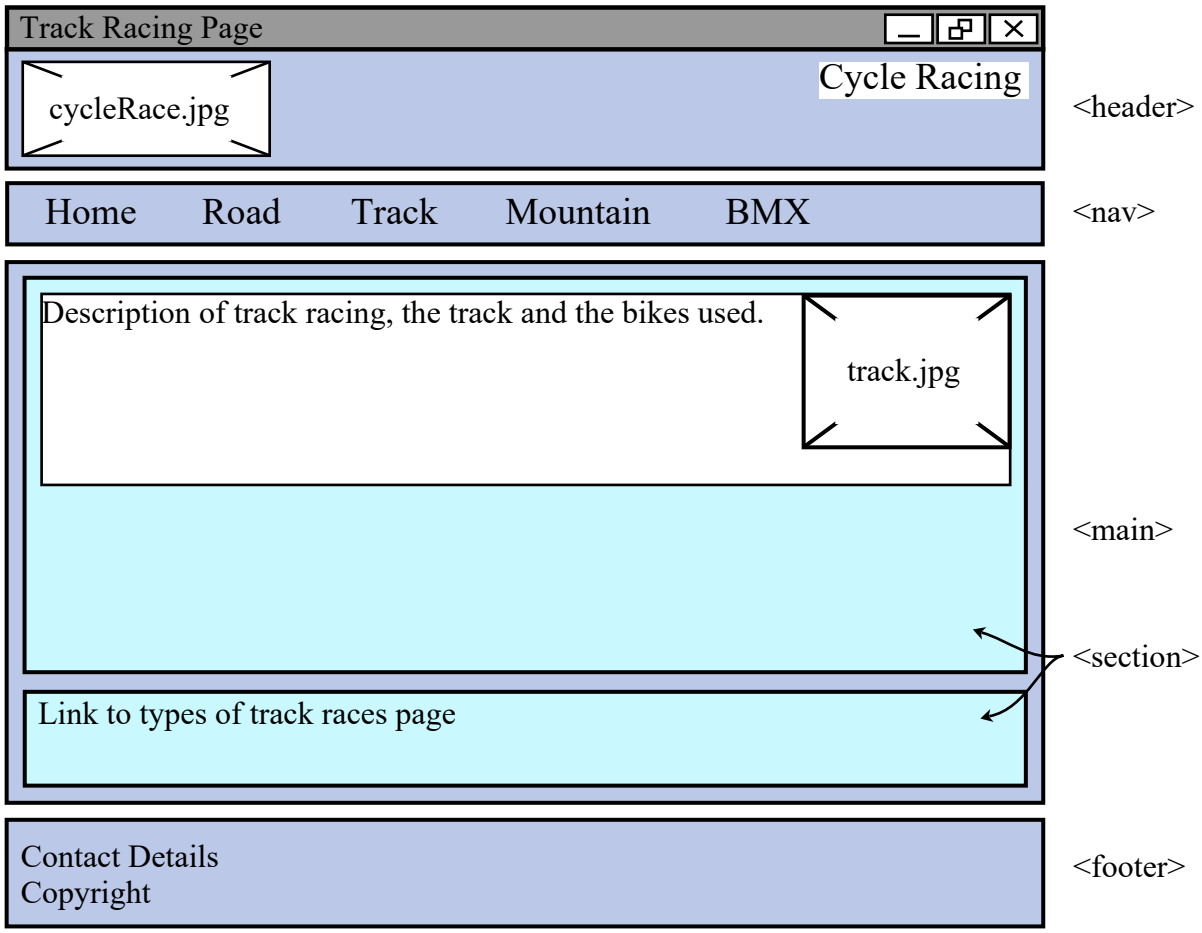
Spacing is controlled using CSS: margins for adding space outside elements and padding for creating space inside elements.



Margins and Padding are implemented as follows:

CSS Declaration			
Property	**Values**	**Example Declarations**	**Explanations**
`margin (left, right, top, bottom)`	`auto, length`	`body {margin: auto}`	The browser calculates the margin. This centres one element inside another.
		`div {margin: 10px}`	Pushes all elements away from any adjacent elements by a distance of 10 pixels.
		`li {margin-left: 10px}`	Pushes the left hand edge of any elements 10 pixels away from the edge of its container element.
		`p {margin: 5px 10x 25px 10px}`	Assigns the paragraph element margins of: top 5px, right 10px, bottom 25px and left 10px.
`padding (left, right, top, bottom)`	`length`	`section {padding: 15px}`	Pushes all content contained within the section elements 15 pixels in from all four sides of the element.
		`p {padding-top: 10px}`	Pushes any text within a paragraph element 10 pixels down from the top of the element.

The correct use of margins and padding often requires some thought and problem solving. Consider the wireframe below for the track racing page.



1. The header, nav, main and footer elements are separated by a small gap at the top or bottom of each element. This could be achieved by either:
   - applying bottom margins - `header, nav, main (margin-bottom:5px)`
   - or applying top margins - `nav, main, footer (margin-top:5px)`

   In this case either method could be applied.

2. The cycle racing image and text are slightly in from the edge of the header element. This could be achieved in two ways.
   - apply ids with margin styles to the text and image - `#bannerImage {margin:5px}` `#bannerText {margin-top:5px;margin-right:5px}`
   - applying padding to the header element - `header {padding:5px}`

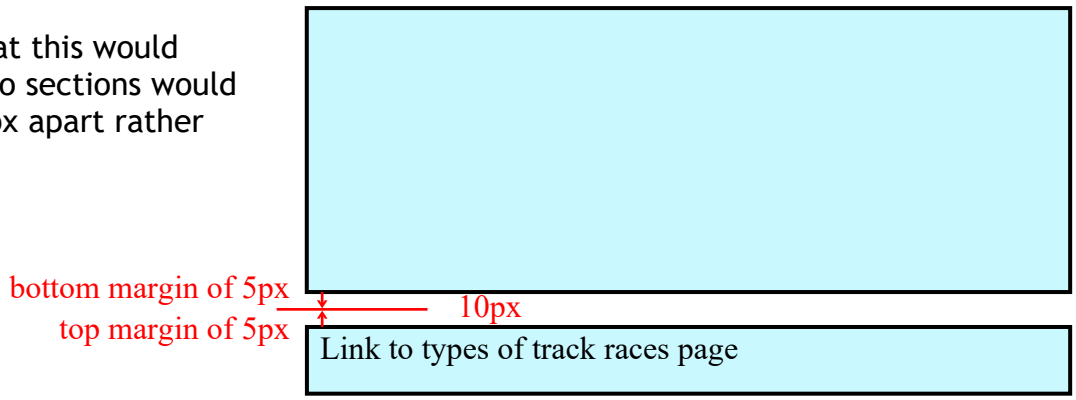
   Padding provides the most efficient solution as this moves both the text and image in from the edge of the header element using a single style.

3. The two section elements are slightly in from the edge of the main element and also slightly apart from each other. This could be achieved by:
   - either applying a padding to the main element and a bottom margin to the section elements -
     `main {padding:5px}  section {margin-bottom:5px}`
   - or by applying a margin to the section elements - `section {margin:5px}`

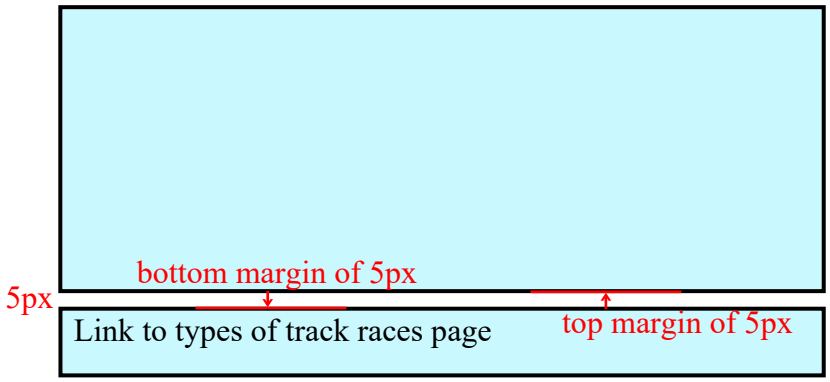
   The second method is more efficient as the desired positioning can be achieved using a single style.

   Note that this applies a 5 pixel margin to the bottom of the first section element and 5 pixel margin to the top of the second section element.

   You may think that this would mean that the two sections would be positioned 10px apart rather than 5px.

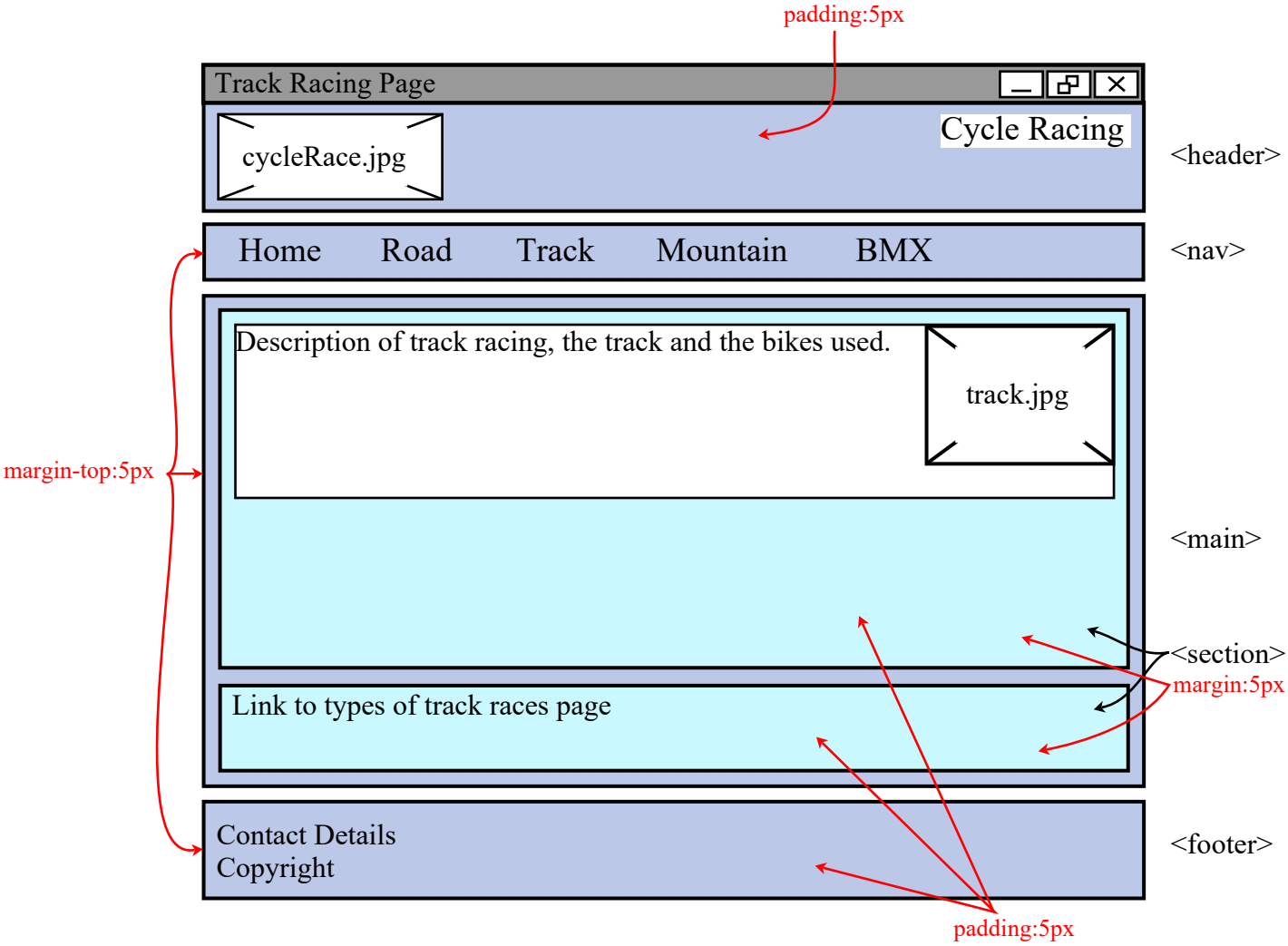


   In reality each margin pushes each section away from the edge of the other element and not from the limit of the other margin. The effect of this is that the margins overlap each other ensuring the desired 5px margin is implemented.



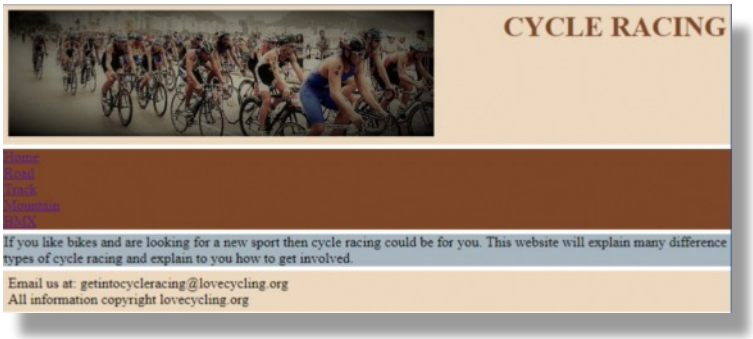
4. The text and links in the section and footer elements have been moved in slightly from the edge of the elements. This can be achieved using padding with a grouping selector.
   - `section, footer {padding:5px}`

The decisions made can be used to annotate the wireframe design before coding the CSS for the page.



The coded CSS rules are shown on the right.

```
/* Margins */
nav, main, footer {margin-top:5px}
section {margin:5px}


/* Padding */
header, section, footer {padding:5px}
```

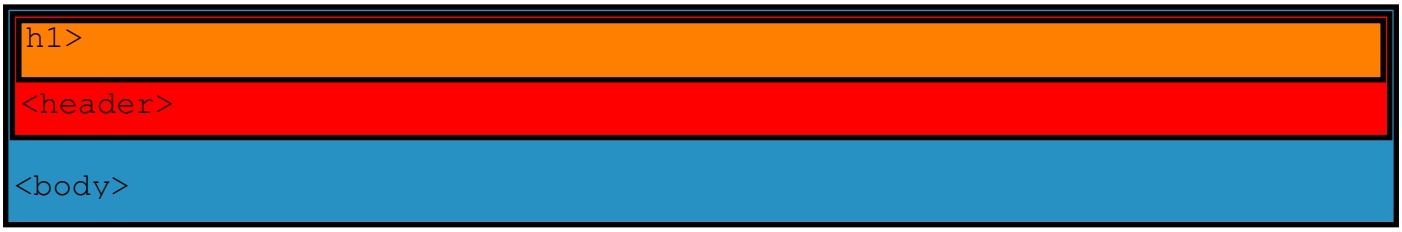


Testing the home page shows the effects of some of the margins and padding. More of the website will have to be implemented before all of the spacing rules are visible.

You should now complete "Implementation Task 4 - Setting up Spacing on the Student Cooking Website". This can be found in the separate Implementation Tasks booklet.

Size

HTML elements automatically expand to fit the element they are contained within (known as their 'container element'). For example a `<body>` element will automatically expand to fit the width of a browser window. A `<header>` element may expand to fill the container element `<body>`. An `<h1>` element will expand to fit the width of the `<header>` element it is contained within.



Sometimes this automatic expansion is exactly the behaviour we are looking for but at other times we may wish to exert some control over the height and width of page elements.

Sizes are controlled using the CSS height and width properties.

Heights and widths usually have a value stated in pixels (px) but may also be declared using cm or as a percentage % of their container element.

This is element 1

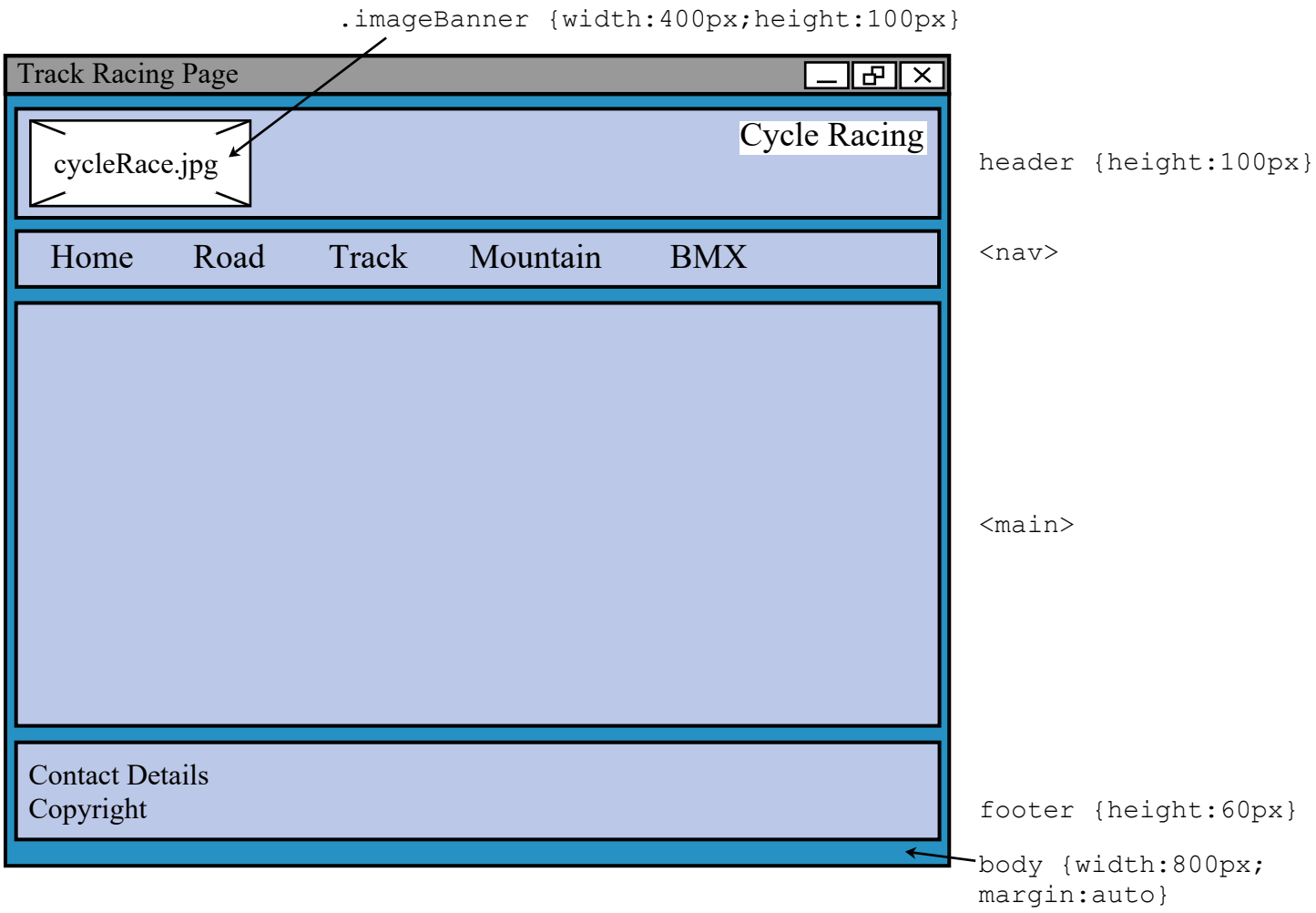
A width of 300px with a height of 35px.

This is element 2

A width of 150px with a height of 80px.

When setting the width of all the pages in a website the `<body>` element is assigned a width. To ensure the page is always positioned in the centre of the browser window (i.e. the margins to the left and right of the body element are always the same width) an automatic margin is assigned.

The sizes of the content that will appear on each page are shown below.



Note that there are two elements that do not have sizes:

main - a height has not been set on the main element as this would mean the content on each page of the cycling website would have to fit into the same fixed area on every page.

nav - a height will be set for this in the next chapter of this booklet.

The CSS code for the above design looks like this when implemented.

```
/* Element Sizes */
body {width:800px}
.imageBanner {width:400px;height:100px}
header {height:100px}
footer {height:60px}
```

Remember that an automatic margin should be added to the body element to ensure the website is centred in the browser window.

```
/* Margins */
nav, main, footer {margin-top:5px}
section {margin:5px}
body {margin:auto}
```

Sizes (heights and widths) are implemented as follows:

CSS Declaration			
Property	**Values**	**Example Declarations**	**Explanations**
height, width	px, cm %	body {width:800px} img {height:100px; width:200px} p {width:50%}	Sets the width of the web page to 800 pixels. Assigns an image a height and width of 100 x 200 pixels. Sets every paragraph to be half the width of its container element.

You should now complete "Implementation Task 5 - Setting sizes on the Student Cooking Website". This can be found in the separate Implementation Tasks booklet.

Stage 4 - Using CSS to Create a Navigation Bar

When creating the template for the cycling website the site navigation was coded in HTML as a set of links within a bullet point list.

 bullet point list

 list items within the bullet point list

<a> hyperlinks to other main pages in the website

```
17   <!-- Navigation Bar -->
18   <nav>
19   <ul>
20     <li><a href="home.html"> Home </a></li>
21     <li><a href="road.html"> Road </a></li>
22     <li><a href="track.html"> Track </a></li>
23     <li><a href="mountain.html"> Mountain </a></li>
24     <li><a href="bmx.html"> BMX </a></li>
25   </ul>
26   </nav>
```

To create a navigation bar for the website styles should be added to `<ul>`, `<li>` and `<a>` elements within the `<nav>` element. The styles will alter the look of the list items and links.

Before

- Home
- Road
- Track
- Mountain
- BMX

After

Home Road Track Mountain BMX

Coding the Navigation Bar

Step 1 - Removing the bullet points

The look of bullet points in a list can be styled using the `list-style-type` property.

```
ul {list-style-type:circle}
```
○ example

or

```
ul {list-style-type:square}
```
■ example

To remove the bullet points completely the value of the `list-style-type` property is set to `none`.

```
ul {list-style-type:none}
```
example

A text colour for the links can also be set at this time.

```
a {color:black}
```

Child Selectors

The above style would remove the bullets from every bulleted list in the website. To ensure only the navigation list is styled the code can be written as shown below:

```
nav ul {list-style-type:none}
```

The above code can be read as “apply this style to ul elements that are contained within nav elements”.

This is referred to as a child selector. The ul element is contained within or is “a child of” the nav element.

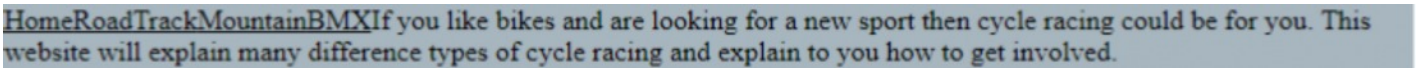
The use of child selectors in this way allows the navigation list to be styled without affecting any other lists found within the rest of the website code.

Step 2 - Changing the list from horizontal to vertical

The five list items can be stacked from left to right by floating the list items.

```
45  /* Navigation List Properties */
46  nav ul {list-style-type:none}
47  a {color:#000}
48  nav ul li {float:left}
```

Note the use of child selectors again to style list items within a bullet point list within the nav element.

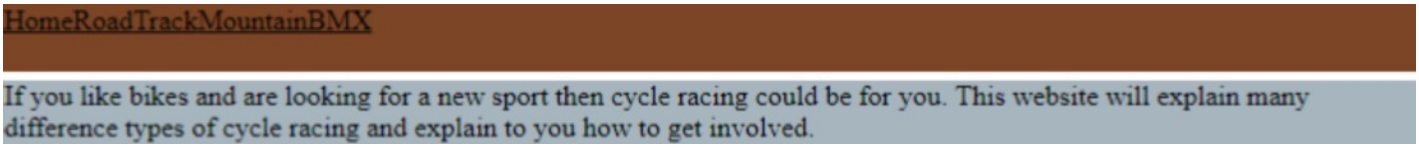


As the `<nav>` element has no set height, the floated items merge with the content in the `<main>` element below.

To ensure the floated list items behave correctly, the height of the <nav> element should be set at this same time.

This separates the two elements.

```
33  /* Element Sizes */
34  body {width:800px}
35  .imageBanner {width:400px;height:100px}
36  header {height:100px}
37  footer {height:60px}
38  nav {height:38px}
```



If you like bikes and are looking for a new sport then cycle racing could be for you. This website will explain many difference types of cycle racing and explain to you how to get involved.

Step 3 - Separating the list items

The list items, although now organised horizontally are crushed together on the left hand side of the page. To separate the items they are given a width.

By positioning the text in the centre of each list item the effect of a box, with the text aligned in the middle, is implemented.

```
45  /* Navigation List Properties */
46  nav ul {list-style-type:none}
47  a {color:#000}
48  nav ul li {float:left;width:120px;text-align:center}
```

Step 4 - Styling the hyperlinks.

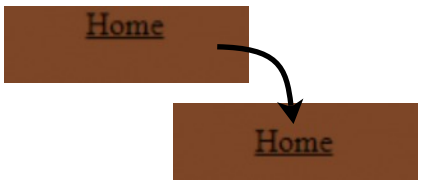
The following code can added to the link elements.

```
/* Navigation List Properties */
nav ul {list-style-type:none}
a {color:#000}
nav ul li {float:left;width:120px;text-align:center}
nav ul li a {display:block;padding:10px}
```

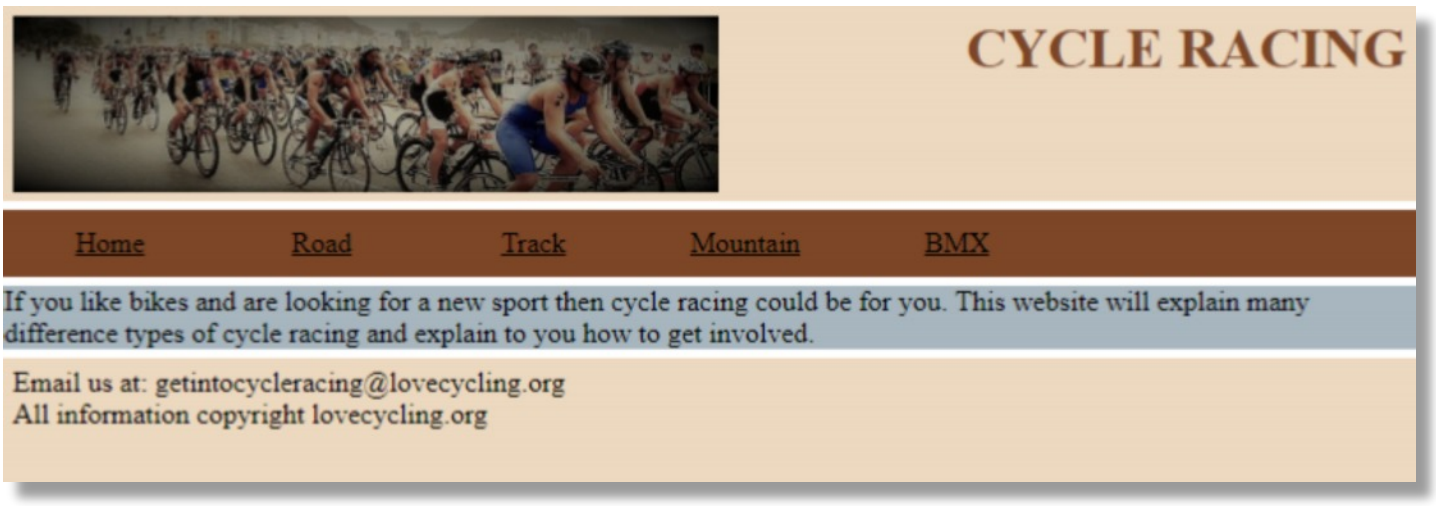
The two styles added to the <a> elements within the navigation bar (note the use of child selectors again) can be broken down as follows:

1. display:block — This has the effect of turning the whole area around list item text into a link.

2. padding:10px — This pushes the text down into the centre of the list item.



Once step 4 has been implemented the web page looks like it has a working navigation bar.



One more step is required to add some interactivity to the navigation bar.

Step 5 - Adding interactivity to the navigation bar.

Styles can be used make the navigation bar respond to a mouse pointer passing over, or 'hovering' over the hyperlinks.

```
/* Navigation List Properties */
nav ul {list-style-type:none}
a {color:#000}
nav ul li {float:left;width:120px;text-align:center}
nav ul li a {display:block;padding:10px}
nav ul li a:hover {background-color:#edd9c0;color:#7d4627}
```

The above code changes the background colour and text colour of an <a> element when the mouse hovers over it.

Editing the Navigation Bar Code

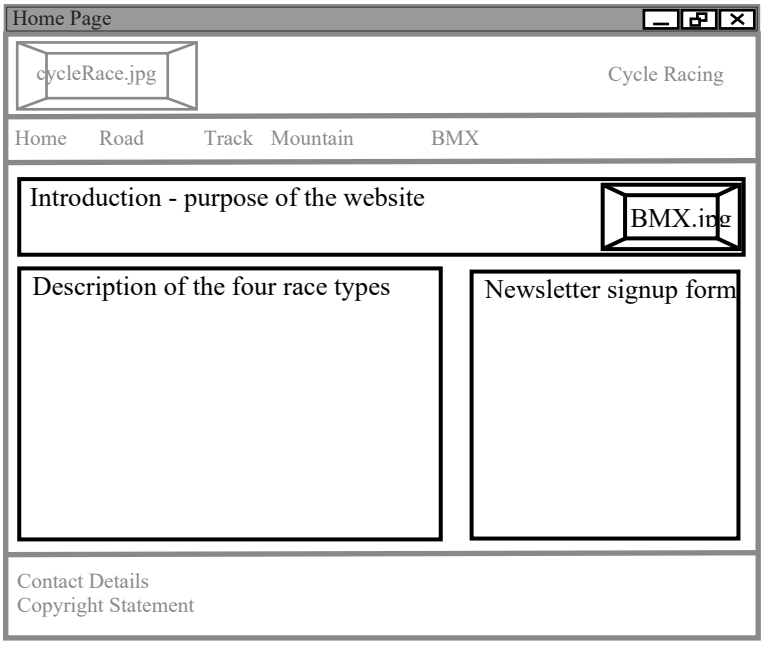
Creating your own navigation is simply a matter of editing the complete code given below. If you wish to alter colours, sizes, positions or inactivity simply edit, add or remove the styles below.

```
nav {height:38px}
```

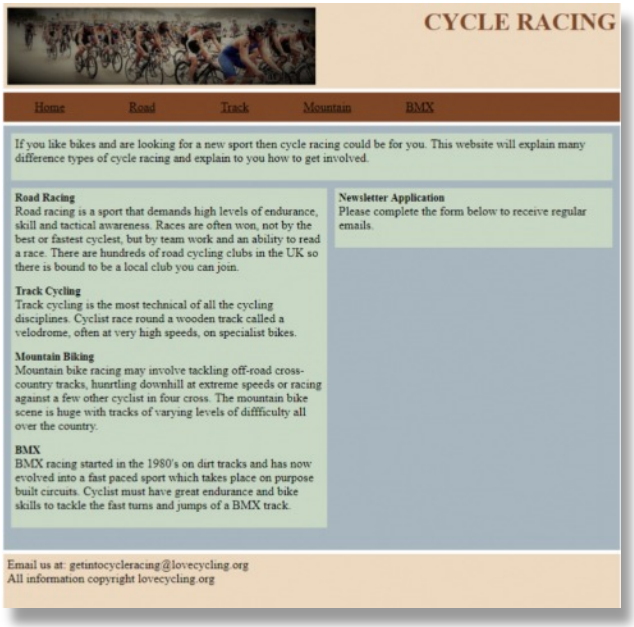
```
nav ul {list-style-type:none}
a {color:#000}
nav ul li {float:left;width:120px;text-align:center}
nav ul li a {display:block;padding:10px}
nav ul li a:hover {background-color:#edd9c0;color:#7d4627}
```

You should now complete "Implementation Task 6 - Styling the Navigation Bar of the Student Cooking Website". This can be found in the separate Implementation Tasks booklet.

Stage 5 - Adding a Form to the Home Page



During the analysis and design stages it was identified that a form would be added to the home page of cycling website. This would be added as one of three sections.



Note that to create the three sections, positioned and sized as shown above, the following code is required.

home.html

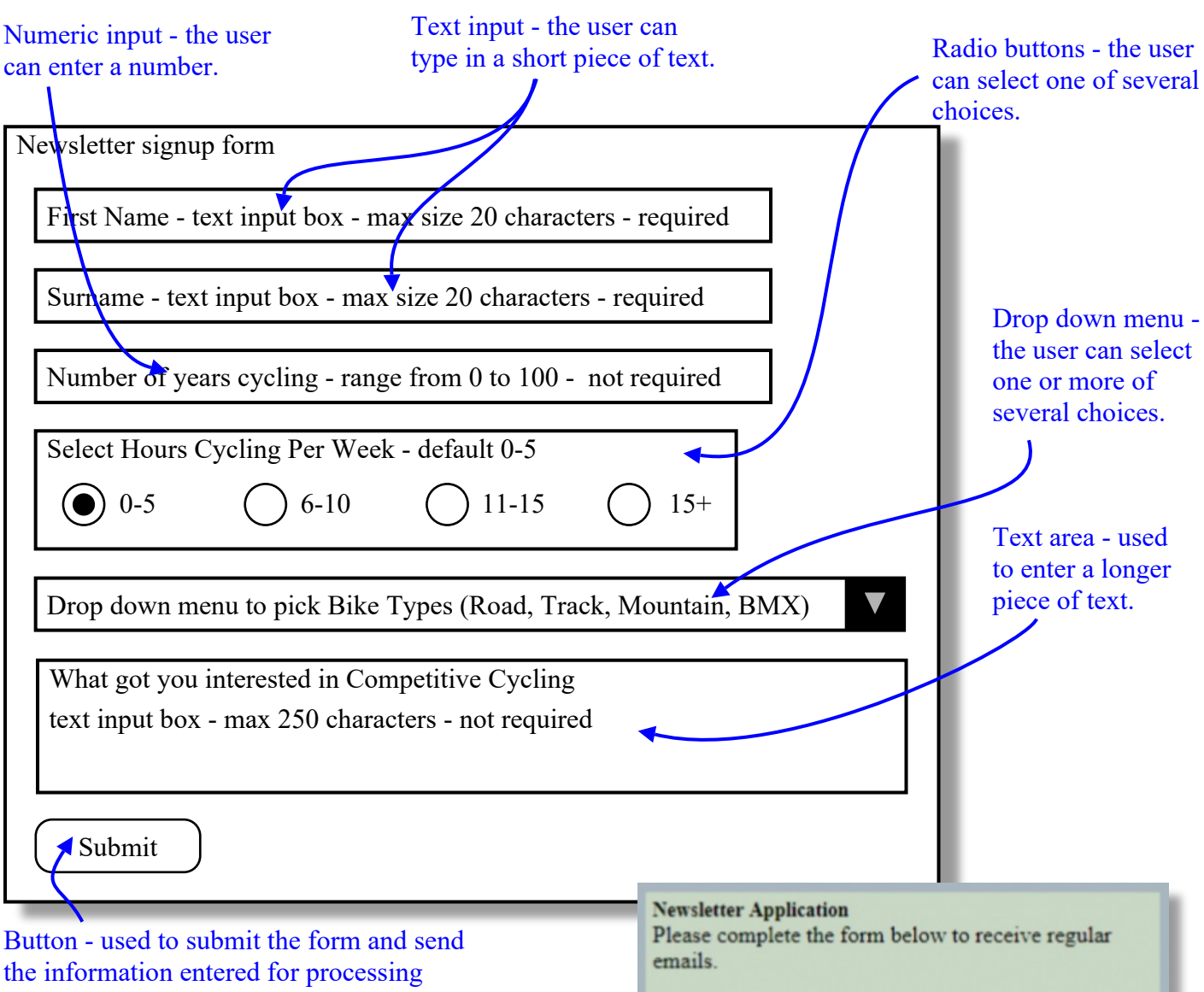
```
<!-- The main content of the page -->
<main style="height:540px">

<section>
<p> If you like bikes and are looking for a new sport then cycle racing could be for you. This website will explain many difference types of cycle racing and explain to you how to get involved. </p>
</section>

<section id="raceDescriptions">
<h2>Road Racing</h2>
<p>Road racing is a sport that demands high levels of endurance, skill and tactical awareness. Races are often won, not by the best or fastest cyclest, but by team work and an ability to read a race. There are hundreds of road cycling clubs in the UK so there is bound to be a local club you can join.</p>
<h2>Track Cycling</h2>
<p>Track cycling is the most technical of all the cycling disciplines. Cyclist race round a wooden track called a velodrome, often at very high speeds, on specialist bikes.</p>
<h2>Mountain Biking</h2>
<p>Mountain bike racing may involve tackling off-road cross-country tracks, hunrtling downhill at extreme speeds or racing against a few other cyclist in four cross. The mountain bike scene is huge with tracks of varying levels of diffficulty all over the country.</p>
<h2>BMX</h2>
<p>BMX racing started in the 1980's on dirt tracks and has now evolved into a fast paced sport which takes place on purpose built circuits. Cyclist must have great endurance and bike skills to tackle the fast turns and jumps of a BMX track.</p>
</section>

<section id="newsletterForm">
<h2>Newsletter Application</h2>
<p>Please complete the form below to receive regular emails.</p>

</section>

</main>
```

styles.css

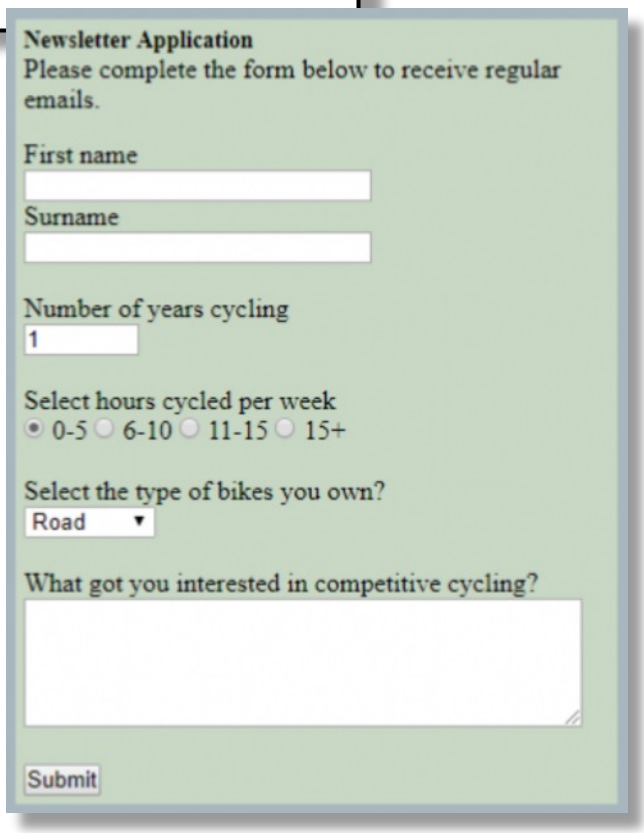
```
#raceDescriptions {float:left;width:400px}
#newsletterForm  {float:right;width:350px}
header, nav, main, footer {display:block;clear:both}
```

The two sections are given ids and positioned side-by-side using styles to `float` them `left` and `right`. Note that the `<main>` element requires a height or the grey background would only surround the top `<section>` element.

The wireframe design below shows a variety of input types are required on the form.



When implemented the cycling website form will look like this.



HTML 5 introduced many new features to HTML. One of these was the ability to create a form without the need for any other programming languages.

The form is broken down into elements to create:

- the text instructions
- input types
- sizes
- validation

The Form Element

A form is declared using the <form> element. Within the element the content of the form is created using a combination of text and input types.

```
50 <form action="">
51
52 </form>
```

When the user submits a form it is usually passed to a server. This "action" is beyond what will be covered in this booklet so the action property has been left empty.

Text Input

A text input box is declared as shown below. The width of the box is entered in characters (`size="25"`).


```
51 First name<br>
52     <input type="text" name="firstname" size="25">
53     <br>
```

It is good practice in forms to provide the user with instructions so text is added around the different types of input.

Points to Note

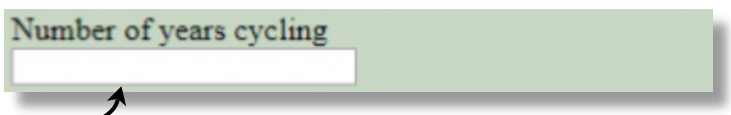
When the form is passed to a server the data entered by the user is identified by its name. As forms will not be processed in this course the code "`name="firstname"`" is not required. It is good practice however to name the different inputs within a form.

The use of line breaks (
) to control where the text input box appears in relation to the instruction ("First name").

Without the line break the text box would appear beside the instruction.

Numeric Input

Creating a box to allow numerical input by users is very similar to creating a text box.




```
57 Number of years cycling<br>
58     <input type="number" name="cyclingYears">
59     <br><br>
```

Radio Button Input

Radio button input offers a list of options from which the user can select only one.

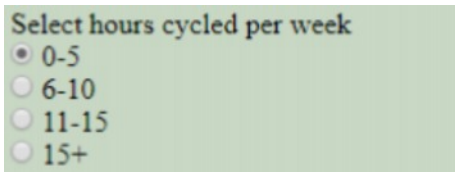
The radio buttons are identified as being part of the same group using the "name" property. Each option has a value that would be passed to a server when the form is submitted. The text instructions labelling each option is added to the end of each line.

```
60  Select hours cycled per week<br>
61      <input type="radio" name="hours" value="0 to 5" checked> 0-5
62      <input type="radio" name="hours" value="6 to 10"> 6-10
63      <input type="radio" name="hours" value="11 to 15"> 11-15
64      <input type="radio" name="hours" value="over 15"> 15+
65      <br><br>
```

One of the radio inputs can be coded as "checked". When the web page loads this option will already be selected on the form.

Additional Point

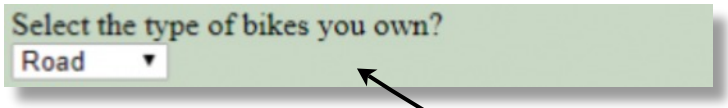
If a
 is added at the end of each line the radio buttons will stack vertically rather than be displayed horizontally.



Selection (Drop-down Menu)

The `<select>` element is used to enclose a lost of `<option>` elements. Each `<option>` creates an item in the list. A different "value" would be passed to the server depending on the option selected.

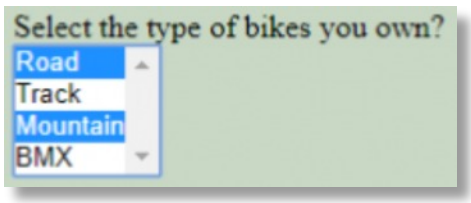
The code on the right shows how a drop-down menu would be implemented on the cycling website.




```
66  Select the type of bikes you own?<br>
67      <select name="bikeType">
68          <option value="lowest">Road</option>
69          <option value="second">Track</option>
70          <option value="third">Mountain</option>
71          <option value="most">BMX</option>
72      </select>
```

The design stated the user should be able to select more than one bike. This is achieved by adding the multiple property to the `<select>` element.

```
67      <select name="bikeType" multiple>
```

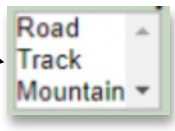


The user can select more than one option by holding down the Ctrl key when choosing options.

Additional Point

It is possible to change the number of options the user can see at any one time using the "size" property.

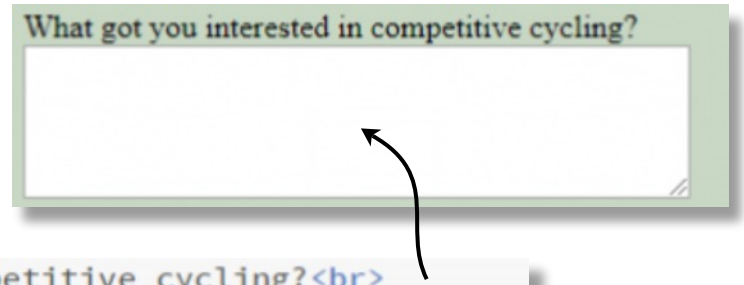
```
<select name="bikeType" multiple size="3">
```



Text Area

The <textarea> element is used to create a larger text box where the user can give an extended response.

The height and width of the box is set as rows and columns (cols) of characters.

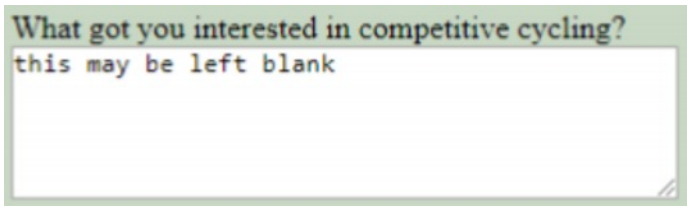



```
74  What got you interested in competitive cycling?<br>
75    <textarea name="message" rows="5" cols="45"> </textarea>
76    <br><br>
```

Additional Point

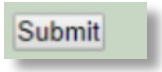
The text area can be pre-populated by adding text between `<textarea>` and `</textarea>`.

```
<textarea name="message" rows="5" cols="45">this may be left blank</textarea>
```



Submit Input (Button)

The `<input>` element with a "type" property set to "submit" is used to create the button. The text within the button is added using the "value" property.




```
78    <input type="submit" onclick="alert('Form Entered')" value="Submit">
```

The JavaScript onclick event is not required and has been added simply to give a visual acknowledgement (an alert message) that something has happened when the button is clicked.

Additional Point (beyond this course)

Forms would usually be passed (using the "action" property) to a server for processing by a language like PHP.

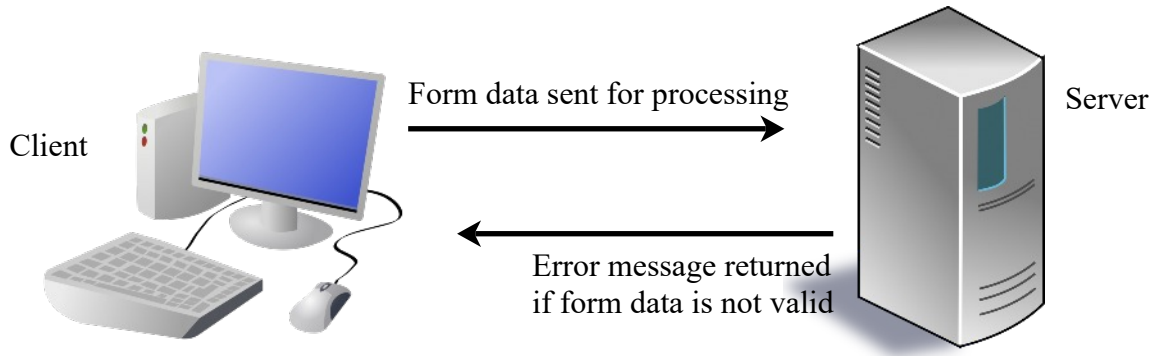
```
<form action="/action_page.php">
```

Form Validation

Data is validated to increase the robustness of a system. If the user can enter only valid data a program is less likely to crash when processing the data. It is important to ensure form data in a web page is valid because it may contain vital information such as a user's account details, an online purchase or a banking transaction.

Web form validation can take place in two locations:

- **Client side** validation takes place at the users computer system.
- **Server side** processing is carried out by the server. Any errors are sent back to the client which then asks the user to re-enter the data.



It would be perfectly possible to leave all the form validation to the server but this would be inefficient as, every time the user enters invalid data, communication must take place across a network between the client computer and the server computer.

Implementing client side validation using HTML5 within the form code itself (or using JavaScript with older versions of HTML) ensures that only valid form data is sent to the server reducing any inefficient communication.

Maximum and Minimum Values

`Max`, `min` and `maxlength` properties can be added to text input, numerical input and a text area in order to:

- limit the number of characters the user can enter
- set a range of valid numerical values.

Text Input

```
38    First name:<br>
39    <input type="text" name="firstname" size="30" maxlength="25" required>
40    <br>
```

Numerical Input

```
57  Number of years cycling<br>
58      <input type="number" name="cyclingYears" min="0" max="100">
59      <br><br>
```

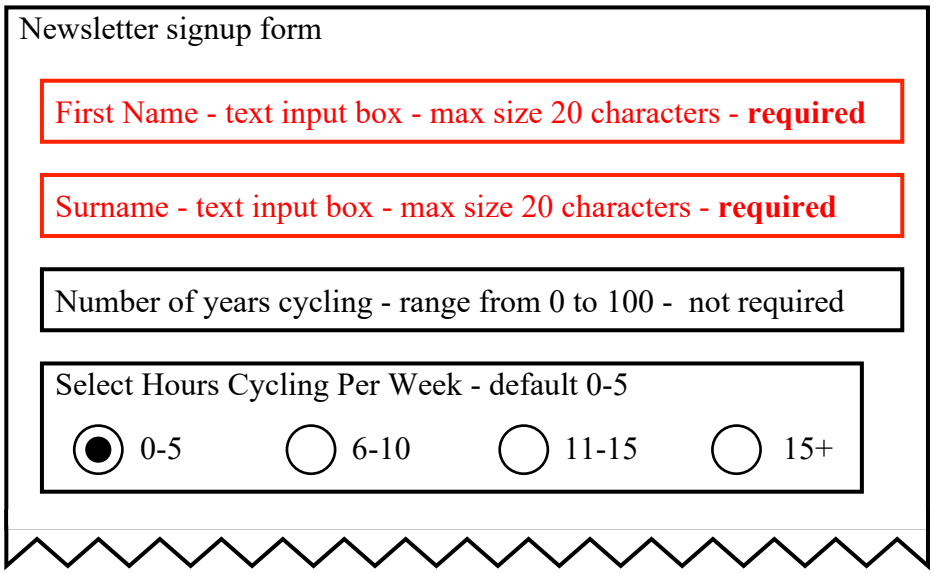
Text Area

```
74  What got you interested in competitive cycling?<br>
75    <textarea name="message" rows="5" cols="45" maxlength="250">  </textarea>
76    <br><br>
```

Presence Check

When asking a user to complete a form it is sometimes necessary to ensure that one or more of the inputs are not left blank by the user. For example, if you ask someone to register for a website you may wish to insist that they enter their name.

The wireframe design for the form notes that the following two inputs should always be completed by the user.



Required Attribute

The "required" attribute can be added to text, number and radio inputs to ensure that the user does not leave them blank.

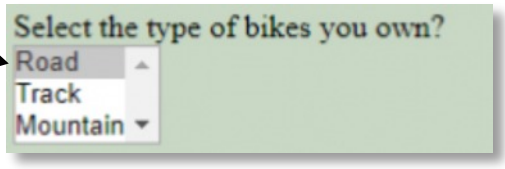
```
51  First name<br>
52      <input type="text" name="firstname" size="25" required>
53      <br>
54  Surname<br>
55      <input type="text" name="surname" size="25" required>
56      <br><br>
```

As noted in the design only the two text inputs for forename and surname are required.

Selected Attribute

When using the <select> element one of the options can be pre-selected forcing the user to enter at least one of the options.

```
66  Select the type of bikes you own?<br>
67    <select name="bikeType" multiple size="3">
68      <option value="lowest" selected>Road</option>
69      <option value="second">Track</option>
70      <option value="third">Mountain</option>
71      <option value="most">BMX</option>
72    </select>
```



You should now complete "Implementation Task 7 - Creating a Questionnaire for the Student Cooking Website". This can be found in the separate Implementation Tasks booklet.

Stage 6 - Using JavaScript to add Interactivity

JavaScript is a high-level, interpreted programming language which alongside HTML and CSS, is one of the three core technologies of the World Wide Web. JavaScript enables the implementation of interactive web pages. All major web browsers have a dedicated JavaScript engine used to execute JavaScript code.

Initially only implemented client-side in web browsers, JavaScript engines are now embedded in many other types of host software, including server-side in web servers and databases.

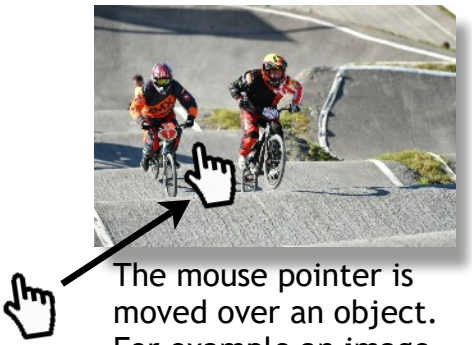
This course will cover the use of JavaScript to create interactivity by implementing client-side events initiated by mouse movements and clicks. Interactivity covered will include:

- changing the size of an image as the mouse pointer passes over the image
- changing the style of page content when the mouse passes over the image
- swapping one image for another as the mouse pointer passes over the image
- showing and hiding page content when the mouse is clicked on a page object.

Mouse Events

JavaScript mouse events can be used to trigger page interactivity.

Three JavaScript mouse events are:



JavaScript Example 1 - Changing the Size of an Image

JavaScript code can be used to change the size of an image when the mouse passes over it.

This is achieved using two events:

`onmouseover` - used along with JavaScript code to increase the size (width and height) of the image

The code for this is shown below:

```
onmouseover="this.style.width='300px';this.style.height='225px'"
```

This code can be broken up and explained as follows:

The element event (in this example an onmouseover attached to an <img> element) is followed by = " some *JavaScript code* ".

Each JavaScript instruction is separated with a semi-colon.

```
onmouseover="                ;                "
```

The width and height are changed as they would be in a css style. Note the different syntax JavaScript uses: *width* = ' '

```
width='300px'
```

```
this.style.height='225px'
```

```
this.style.
```

The code shown contains two instructions, each followed by a full stop:

`this` - meaning the following code refers to 'this' element. In the example, the <img> element containing the event.

`style` - meaning the following code refers to a 'style' property of the 'this' element. In the example, the 'width'.

`onmouseout` - used along with JavaScript code to return the graphic to it's original size

The code for this simply sets the size of the image back to its original 200 pixels by 150 pixels.

```
onmouseout="this.style.width='200px';this.style.height='150px'"
```

Both of the JavaScript events (onmouseover and onmouseout) are added to the <img> element to create the required interactivity.

```
35   <img class="BMXImages" src="../images/bmxRacing1.jpg"
     onmouseover="this.style.width='300px';this.style.height='225px'"
     onmouseout='this.style.width="200px";this.style.height="150px"'>
```

JavaScript Syntax Rule

In JavaScript, double and single quotes can be used interchangeably. The code below works as well as the above example.

```
onmouseout='this.style.width="200px";this.style.height="150px"'
```

As a rule if double quotes are used to surround the JavaScript instructions, the inner instructions must have single quotes. If single quotes are used to surround the JavaScript instructions, the inner instructions must have double quotes.

You should now complete "Implementation Task 8 - Interactive Image Sizes on the Student Cooking Website". This can be found in the separate Implementation Tasks booklet.

JavaScript Example 2 - Changing the Size of an Image Using Functions

An alternative method of implementing the interactive graphic size changes is to remove the code from the <img> element and place it within a separate JavaScript script.

JavaScript scripts are often placed inside the <head> section of an HTML document, as shown below, but they may also be located in the <body> or even in an external file.

```
4  <head>
5    <title>BMX Racing</title>
6    <link rel="stylesheet" type="text/css" href="../CSS/styles.css">
7
8  <script>
9
10
11   </script>
12
13   </head>
```

This implementation method requires two separate pieces of code.

- Within the <script> element a function is declared. This will contain the JavaScript code we wish to run.

```
8  <script>
9
10   function showLarge() { }
11
12   </script>
```

- Within the <img> element a "function call" is added to tell the browser to run the code in the function.

```
53   <img class="BMXImages" src="../images/bmxRacing3.jpg"
     onMouseOver="showLarge()">
```

Code can now be added to the function to change the width and height.

```
10   function showLarge()
11   {style.width='300px';
12   style.height='225px';}
```

Note that each JavaScript instruction is separated by a semi-colon.

Finally the code must identify which element within the HTML document the width and height refer to.

This can be achieved by passing "`this`" (discussed in example 1) to the function.

```
53   <img class="BMXImages" src="../images/bmxRacing3.jpg"
     onMouseOver="showLarge(this)">
```

becomes

Within the function a variable is used to store the element being passed.

The variable is then used in the code as shown below to identify the correct page element.

```
10   function showLarge(thisGraphic)
11   {thisGraphic.style.width='300px';
12   thisGraphic.style.height='225px';}
```

A second function is created to change the graphic back to its original size. Once implemented the completed code looks like this.

```
 8 ▼ <script>
 9
10   function showLarge(thisGraphic)
11   {thisGraphic.style.width='300px';
12   thisGraphic.style.height='225px';}
13
14   function showNormal(thisGraphic)
15   {thisGraphic.style.width='200px';
16   thisGraphic.style.height='150px';}
17
18   </script>
```

```
51   <img class="BMXImages" src="../images/bmxRacing1.jpg"
     onMouseOver="showLarge(this)" onMouseOut="showNormal(this)">
52
53   <img class="BMXImages" src="../images/bmxRacing3.jpg"
     onMouseOver="showLarge(this)" onMouseOut="showNormal(this)">
54
55   <img class="BMXImages" src="../images/bmxRacing2.jpg"
     onMouseOver="showLarge(this)" onMouseOut="showNormal(this)">
```

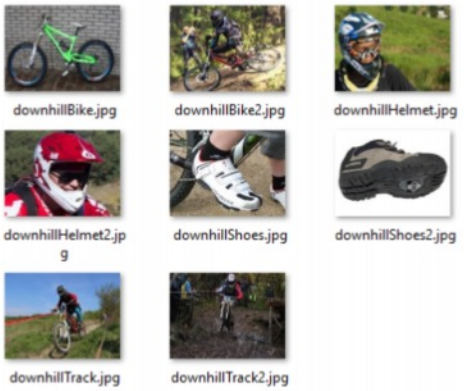
You should now complete "Implementation Task 9 - Creating JavaScript Functions to Interactively Change Image Properties". This can be found in the separate Implementation Tasks booklet.

JavaScript Example 3 - Creating Roll-Over Images

JavaScript code can be used to swap one image for another when the mouse pointer passes over the image. This is called a "roll-over" image. The image can be coded to change back to the original image or remain permanently swapped.

To create four roll-over images on the downhill.html page of the cycling website eight images were required.

Four original images and four replacements.



Each <img> element is coded with three URLs pointing towards the location of two graphic files.

Note the use of "this" again.

This URL is used to load the original image.

```
37   <img class="mountainImages" src="../images/downhillHelmet.jpg"
     onmouseover="this.src='../images/downhillHelmet2.jpg'"
     onmouseout="this.src='../images/downhillHelmet.jpg'">
```

This URL is used to reload the original image when the mouse mouse away from the image.

This URL is used to load the alternative, roll-over image.

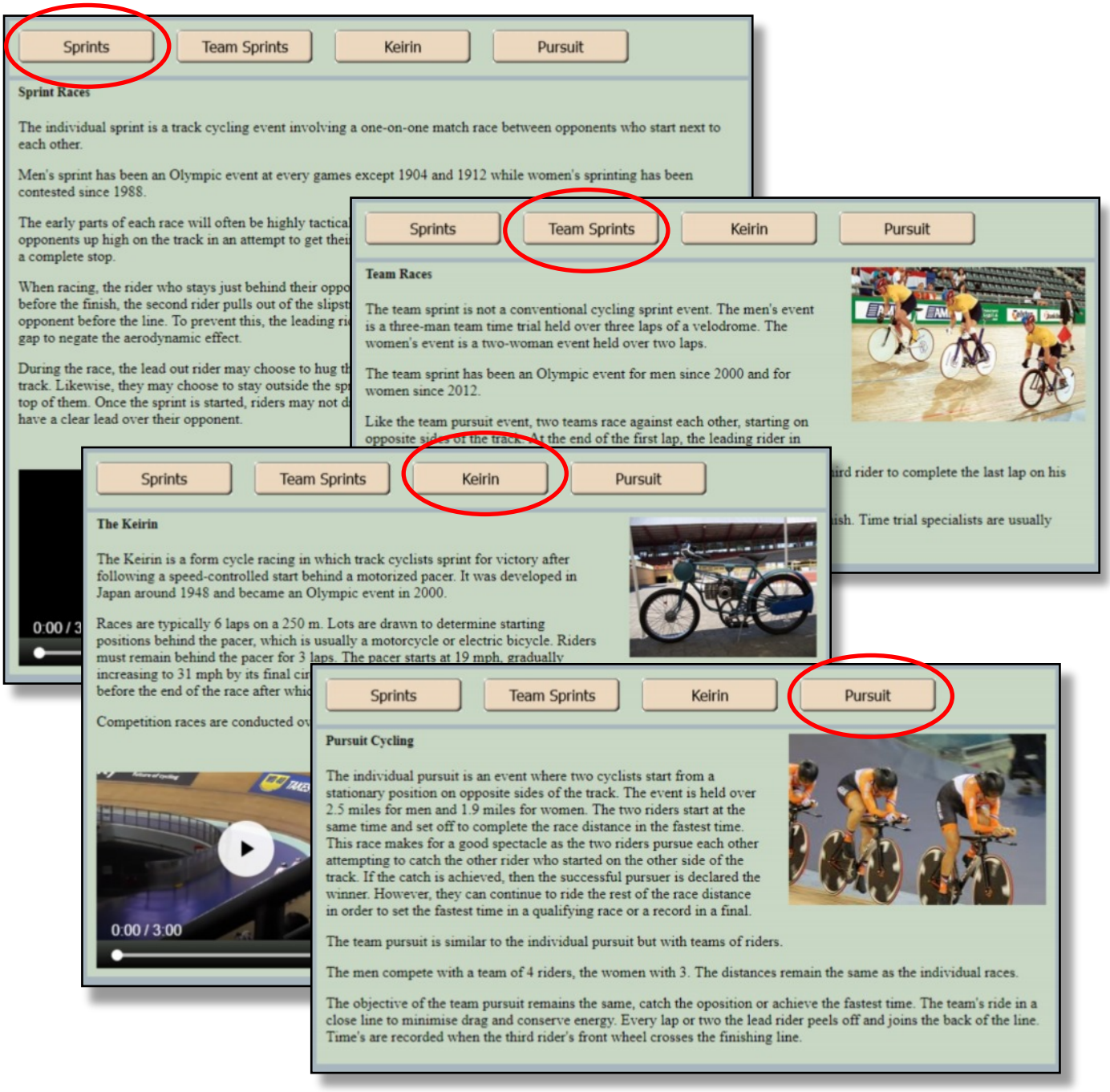
You should now complete "Implementation Task 10 - Creating Roll-Over Images on the Student Cooking Website". This can be found in the separate Implementation Tasks booklet.

JavaScript Example 4 - Showing and Hiding Page Elements

The CSS property "`display`" is used to control how an element appears within a web page. In addition to `inline` and `block`, display may also have the value '`none`'.

If an element is styled to display:none it becomes invisible and the space it would have taken up collapses as if the element code had been removed from the HTML file. This can be used along with JavaScript to create interactive content by showing and hiding various content in a web page.

The Types of track racing page, on the cycling website, has been set up so that the four buttons display different content when they are clicked.



The code for the Types of Track Racing web page is shown below.

```
<main>

<div>
    <img class="buttonTrack" src="../images/buttonSprint.png" onclick="displaySprint()">
    <img class="buttonTrack" src="../images/buttonTeam.png" onclick="displayTeam()">
    <img class="buttonTrack" src="../images/buttonKeirin.png" onclick="displayKeirin()">
    <img class="buttonTrack" src="../images/buttonPursuit.png" onclick="displayPursuit()">
</div>

<div>

<section id="sprint" style="display:block">
    <h2>Sprint Races</h2>
    <p>The individual sprint is a track cycling event involving a one-on-one match race
    between opponents who start next to each other.</p>
    <p>Men's sprint has been an Olympic event at every games except 1904 and 1912 while
    women's sprinting has been contested since 1988.</p>
    <p>The early parts of each race will often be highly tactical with riders pedalling
    slowly often trying to force their opponents up high on the track in an attempt to get
    their rivals to make the first move. Some even bring their bicycles to a complete
    stop.</p>
    <p>When racing, the rider who stays just behind their opponent expends less effort by
    reducing the aerodynamic drag. Just before the finish, the second rider pulls out of
    the slipstream, and with fresher legs, may be able to overtake their opponent before
    the line. To prevent this, the leading rider may choose to accelerate quickly to
    establish a large enough gap to negate the aerodynamic effect.</p>
    <p>During the race, the lead out rider may choose to hug the inside of the track
    giving them the shortest path around the track. Likewise, they may choose to stay
    outside the sprinter's red line to force their opponent to come higher over the top of
    them. Once the sprint is started, riders may not drop into the sprinter's lane or
    cross out of the lane unless they have a clear lead over their opponent. </p>
    <video width="320" height="240" controls>
      <source src="../images/sprint.mp4" type="video/mp4">
    </video>
</section>

<section id="team" style="display:none">
    <img src="../images/teamSprint.jpg" style="float:right;margin-left:30px;margin-
    bottom:30px"><h2>Team Races</h2>
    <p>The team sprint is not a conventional cycling sprint event.  The men's event is a
    three-man team time trial held over three laps of a velodrome.  The women's event is a
    two-woman event held over two laps.</p>
    <p>The team sprint has been an Olympic event for men since 2000 and for women since
    2012.</p>
    <p>Like the team pursuit event, two teams race against each other, starting on
    opposite sides of the track. At the end of the first lap, the leading rider in each
    team pulls up the banking leaving the second rider to lead for the next lap; at the
    end of the second lap, the second rider does the same, leaving the third rider to
    complete the last lap on his own. The team with the faster time is the winner.</p>
    <p>The last rider needs good endurance qualities to maintain high speed to the finish.
    Time trial specialists are usually chosen for this role.</p>
</section>

<section id="keirin" style="display:none">
    <img src="../images/derny.jpg" style="float:right;margin-left:30px;margin-
    bottom:30px">
    <h2>The Keirin</h2>
    <p>The Keirin is a form cycle racing in which track cyclists sprint for victory after
    following a speed-controlled start behind a motorized pacer. It was developed in Japan
    around 1948 and became an Olympic event in 2000.</p>
    <p>Races are typically 6 laps on a 250 m. Lots are drawn to determine starting
    positions behind the pacer, which is usually a motorcycle or electric bicycle. Riders
    must remain behind the pacer for 3 laps. The pacer starts at 19 mph, gradually
    increasing to 31 mph by its final circuit. The pacer leaves the track 750 m (820 yd)
    before the end of the race after which the sprint begins.</p>
    <p>Competition races are conducted over several rounds with one final. Eliminated
    cyclists can try again in repechages.</p>
    <video width="320" height="240" controls>
      <source src="../images/Keirin.mp4" type="video/mp4">
    </video>
</section>

<section id="pursuit" style="display:none">
    <img src="../images/teamPursuit.jpg" style="float:right;margin-left:30px;margin-
    bottom:30px"><h2>Pursuit Cycling</h2>
    <p>The individual pursuit is an event where two cyclists start from a stationary
    position on opposite sides of the track. The event is held over 2.5 miles for men and
    1.9 miles for women. The two riders start at the same time and set off to complete the
    race distance in the fastest time. This race makes for a good spectacle as the two
    riders pursue each other attempting to catch the other rider who started on the other
    side of the track. If the catch is achieved, then the successful pursuer is declared
    the winner. However, they can continue to ride the rest of the race distance in order
    to set the fastest time in a qualifying race or a record in a final. </p>
    <p>The team pursuit is similar to the individual pursuit but with teams of riders.</p>
    <p>The men compete with a team of 4 riders, the women with 3. The distances remain the
    same as the individual races.</p>
    <p>The objective of the team pursuit remains the same, catch the oposition or achieve
    the fastest time.  The team's ride in a close line to minimise drag and conserve
    energy.  Every lap or two the lead rider peels off and joins the back of the line.
    Time's are recorded when the third rider's front wheel crosses the finishing line.</p>
</section>

</div>

</main>
```

Four `<img>` elements (one for each button graphic) are placed at the top of the <main> element.

Each image contains an onclick event which calls a JavaScript function.

The functions are covered on the next page.

`<section>`

Four `<section>` elements containing the content for each type of race are coded to appear below the buttons.

`<section>`

One `<section>` is initially shown using `"display:block"`

The other three <section> elements are hidden using `"display:none"`

`<section>`

Each `<section>` element is given a unique `id` which will be used by the JavaScript functions to identify which part of the page to show or hide.

`<section>`

When one of the button images is clicked…

… the element onclick event calls one of four almost identical functions…

```
<img class="buttonTrack" src="../images/buttonSprint.png" onclick="displaySprint()">
<img class="buttonTrack" src="../images/buttonTeam.png" onclick="displayTeam()">
<img class="buttonTrack" src="../images/buttonKeirin.png" onclick="displayKeirin()">
<img class="buttonTrack" src="../images/buttonPursuit.png" onclick="displayPursuit()">
```

```
<script>

function displaySprint() {
    document.getElementById("sprint").style.display="block";
    document.getElementById("team").style.display="none";
    document.getElementById("keirin").style.display="none";
    document.getElementById("pursuit").style.display="none";
}

function displayTeam() {
    document.getElementById("sprint").style.display="none";
    document.getElementById("team").style.display="block";
    document.getElementById("keirin").style.display="none";
    document.getElementById("pursuit").style.display="none";
}

function displayKeirin() {
    document.getElementById("sprint").style.display="none";
    document.getElementById("team").style.display="none";
    document.getElementById("keirin").style.display="block";
    document.getElementById("pursuit").style.display="none";
}

function displayPursuit() {
    document.getElementById("sprint").style.display="none";
    document.getElementById("team").style.display="none";
    document.getElementById("keirin").style.display="none";
    document.getElementById("pursuit").style.display="block";
}
```

… which shows one of the `<section>` elements and hides the other three by changing the style of each `id`.

```
<section id="sprint" style="display:block">
    <h2>Sprint Races</h2>
    <p>The individual sprint is a track cycling ev
    between opponents who start next to each other
    <p>Men's sprint has been an Olympic event at e
    women's sprinting has been contested since 198
```

You should now complete “Implementation Task 11 - Interactive Hidden Content on the Student Cooking Website”. This can be found in the separate Implementation Tasks booklet.

JavaScript Example 5 - Changing the Style of Page Elements with Mouse Movements

The previous tasks showed that a page element is identified and then manipulated by using either

```
<section id="sprint"
```

or

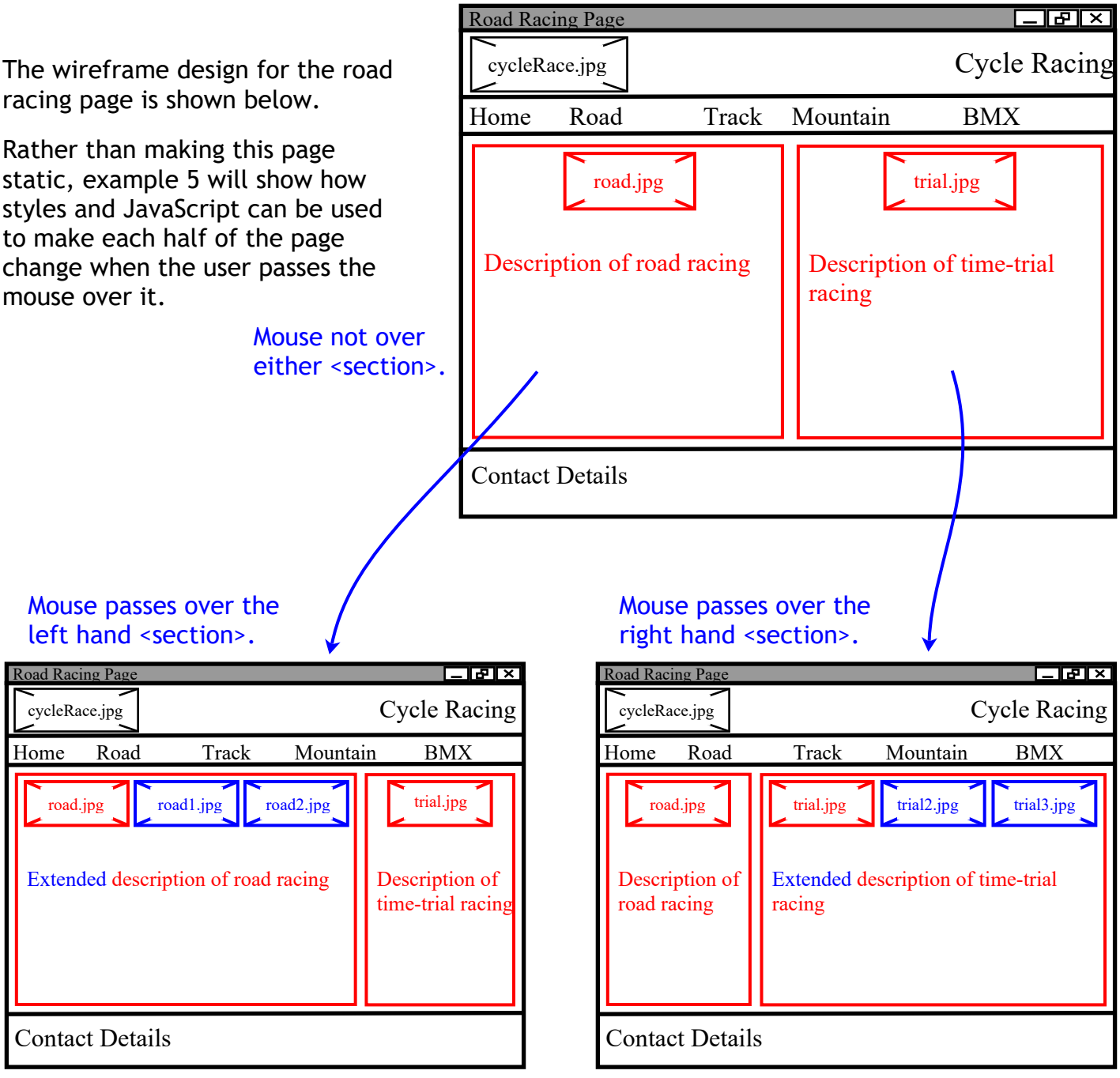
```
this.style.width='300px'
```

to change any CSS styles of the element.

The following example shows that, with a bit of imagination, JavaScript interaction can make pages visually more exciting to use.

The wireframe design for the road racing page is shown below.

Rather than making this page static, example 5 will show how styles and JavaScript can be used to make each half of the page change when the user passes the mouse over it.



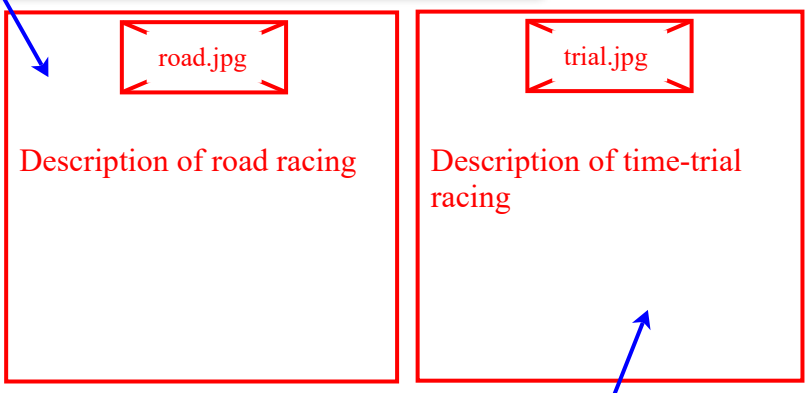
To create this effect **a lot of thought is required and its best to implement something complex like this a stage at a time.**

Stage 1

To manipulate the <section>elements each must initially be:

- given an id
- set up with initial widths and heights
- floated left and right

```
<section id="roadSide"
style="width:370px;height:400px;display:inline;float:left;
margin-left:10px">
```




```
<section id="trialSide"
style="width:370px;height:400px;display:inline;float:right
;margin-right:10px">
```

Stage 2

Four events are added to the <section> elements to implement interactivity:

- onmouseover in first <section>
- onmouseover in second <section>
- onmouseout in first <section>
- onmouseout in second <section>

The events are set up to call one of three functions:

```
<section id="roadSide"
style="width:370px;height:400px;display:inline;float:left;
margin-left:10px" onmouseover="changeLeft()"
onmouseout="changeBack()">
```

```
<section id="trialSide"
style="width:370px;height:400px;display:inline;float:right
;margin-right:10px" onmouseover="changeRight()"
onmouseout="changeBack()">
```

```
<script>

function changeLeft() {

}

function changeRight() {

}

function changeBack() {

}

</script>
```

Stage 3

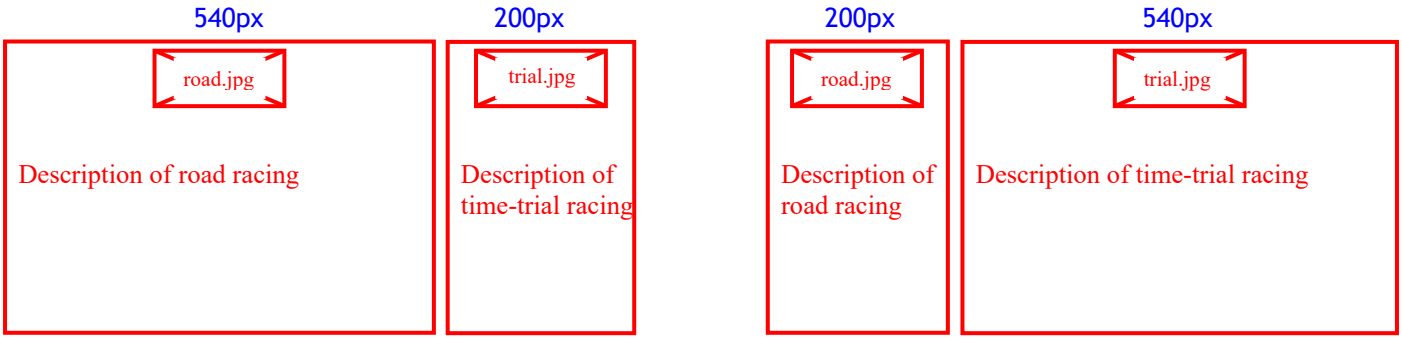
To change the size of the <section> elements code is added to the JavaScript functions:

```
function changeLeft() {
    document.getElementById("roadSide").style.width='540px';
    document.getElementById("trialSide").style.width='200px';
}

function changeRight() {
    document.getElementById("roadSide").style.width='200px';
    document.getElementById("trialSide").style.width='540px';
}

function changeBack() {
    document.getElementById("roadSide").style.width='370px';
    document.getElementById("trialSide").style.width='370px';
}
```

Left side gets wider while right side gets narrower.

Right side gets wider while left side gets narrower.

Both sides return to original widths.



Stage 4

To make the elements appear and disappear in the left hand `<section>` more JavaScript is added to the `changeLeft()` function. This code again uses the ids of the various elements to identify the elements to change.

At the same time the size of the original shown `<img>` is changed so ensure all three images are the same size and fit across the section.

```
function changeLeft() {
    document.getElementById("roadSide").style.width='540px';
    document.getElementById("trialSide").style.width='200px';

    document.getElementById("road").style.display='inline';
    document.getElementById("road1").style.display='inline';
    document.getElementById("road2").style.display='inline';
    document.getElementById("road").style.width='170px';
    document.getElementById("road").style.height='113px';
    document.getElementById("road").style.marginRight='10px';

    document.getElementById("roadParagraph").style.display='block';
}
```

The three 'road' graphics in left `<section>` are all set to `display:inline`. This changes the style of the original image and makes the other two images appear.

The size of the original image is changed to match the other two images (170x113) and the image is given a right margin of 10 pixels.

The additional paragraph about road racing is initially set to `display:none`. When the function is run it now appears.

Stage 5

To change the right side `<section>` exactly the same code is used but with different ids.

```
function changeRight() {
    document.getElementById("roadSide").style.width='200px';
    document.getElementById("trialSide").style.width='540px';

    document.getElementById("trial").style.display='inline';
    document.getElementById("trial").style.width='170px';
    document.getElementById("trial").style.height='113px';
    document.getElementById("trial").style.marginRight='10px';
    document.getElementById("trial1").style.display='inline';
    document.getElementById("trial2").style.display='inline';

    document.getElementById("trialParagraph").style.display='block';
}
```

Stage 6

Finally the `changeBack()` function is coded to change every style back to their original settings.

```
function changeBack() {
    document.getElementById("roadSide").style.width='370px';
    document.getElementById("trialSide").style.width='370px';
    document.getElementById("road").style.display='block';
    document.getElementById("road").style.width='200px';
    document.getElementById("road").style.height='133px';
    document.getElementById("road").style.margin='auto';
    document.getElementById("road1").style.display='none';
    document.getElementById("road2").style.display='none';

    document.getElementById("roadParagraph").style.display='none';

    document.getElementById("trial").style.display='block';
    document.getElementById("trial").style.width='200px';
    document.getElementById("trial").style.height='133px';
    document.getElementById("trial").style.margin='auto';
    document.getElementById("trial1").style.display='none';
    document.getElementById("trial2").style.display='none';

    document.getElementById("trialParagraph").style.display='none';
}
```

With so much going on this code was tested constantly during development to ensure everything works as expected after each stage.

You should now complete “Implementation Task 12 - Creating a Completely Interactive Page on the Student Cooking Website”. This can be found in the separate Implementation Tasks booklet.

Events Summary

Mouse events are implemented as follows:

Mouse Events		
Event	**Example Declarations**	**Explanations**
onmouseover	`<p onmouseover="bigImage()">` `Move the mouse over here to make the photo larger.</p>`	Calls a function called 'bigImage' when the mouse passes over the <p> element containing the event.
onmouseout	`<img src="bike.jpg" onmouseout="this.style.width='150px';this.style.height='100px'">`	Changes the width and height styles of "this" image (bike.jpg) when the mouse pointer moves away from the bike.jpg graphic.
onclick	`<img src="bike.jpg" onclick="document.getElementById('largerBike').style.border='2px solid black'">` `<img id="largerBike" src="largeBike.jpg"`	When the bike.jpg image is clicked the other image (largerBike.jpg identified by its id), will immediately appear to have a 2 pixel thick, black border round the image.

If you wish to experiment, dozens of other events exist that are not part of the Higher Course.

https://www.w3schools.com/tags/ref_eventattributes.asp

Here are a few examples:

onfocus	`<input type="text" id="fname" onfocus="warning(this.id)">`	This would execute a JavaScript function called warning( )when an input element in "in focus" (the cursor is inside the input box).
onkeydown	`<input type="text" onkeydown="showMessage( )">`	Executes a JavaScript function when a keyboard key is pressed.
ondblclick	`<button ondblclick="this.style.color= 'White'"> Double-click me </button>`	This is similar to onclick but responds to a double click instead.

JavaScript Summary

JavaScript code may be implemented as follows:

JavaScript		
Property	**Example Declarations**	**Explanations**
<script> </script>	<script> document.getElementById ("mainHeading").style. color='darkRed'; </script> <h1 id="mainHeading"> College Rooms </h1>	This HTML element is used to place JavaScript within an HTML file. The example code would change the text colour of "College Rooms" in the <h1> element to dark red. This instruction is broken down below.
document	document.getElementById ("mainHeading").style. color='darkRed';	Used to specify that the JavaScript instruction applies to this document.
.getElementById() this	document.getElementById ("mainHeading").style. color='darkRed'; this.style. color='darkRed';	Used to identify an element in the document (in this example "mainHeading"). The element must be given a matching id. "this" may be used instead of document.getElementById if the code is being applied to the element containing the JavaScript.
.style	document.getElementById ("mainHeading").style. color='darkRed';	Notes that the JavaScript will change the style of a page element.
.color=''	document.getElementById ("mainHeading").style. color='darkRed';	Changes the CSS "color" property of the element.

Once you have learned the above code. any CSS property can be added to the end of the instruction.

.display= 'none'	this.style. display='none';	Use of JavaScript to hide an element.
.padding='20px' .paddingTop='100px'	this.style. paddingTop='100px';	Note the CSS declaration padding-top:20px in JavaScript becomes paddingTop='20px'.
cssFloat = 'left'	this.style. cssFloat='left'	Note that the word float relates to real numbers in JavaScript. To use the CSS float property "cssFloat" is used instead.

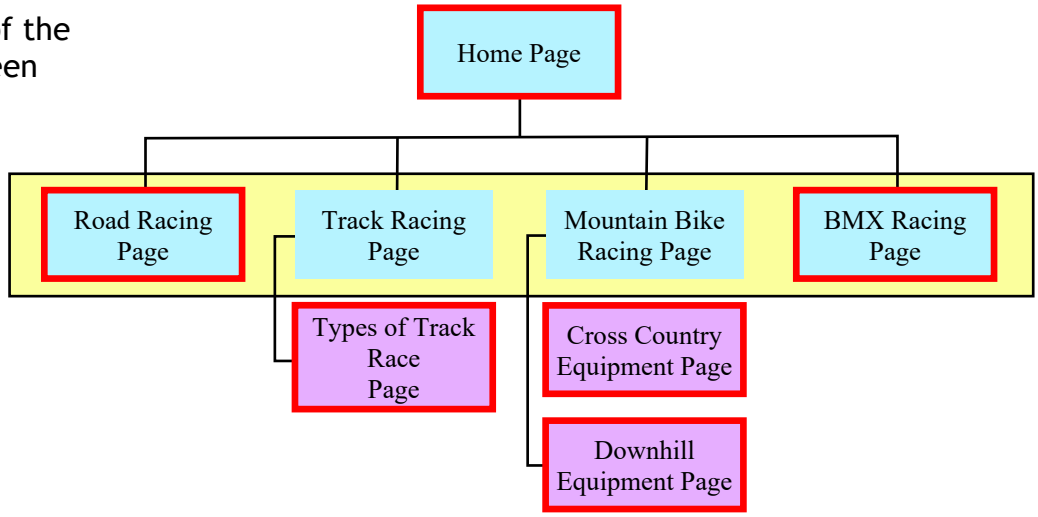
Stage 7 - Completing the Website

After the initial template pages are created, the order in which each page is coded is entirely up to the developer.

The developer may choose to create the pages in order of importance. Home, main topics then sub pages.

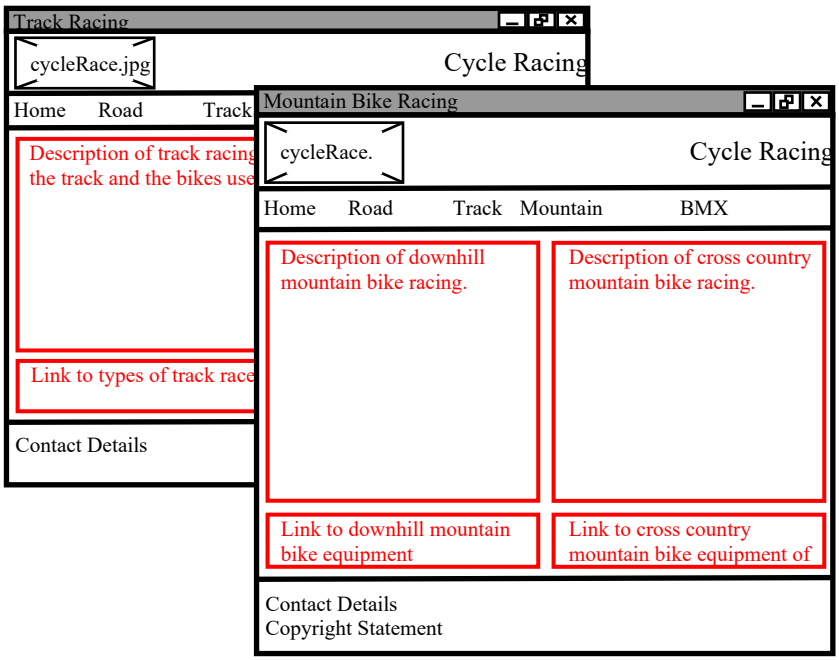
Alternatively the developer may work on random pages, pulling the website together as the project is completed.

The highlighted pages of the cycling website have been completed already.



All that remains is to:

- code the Track Racing Page
- code the Mountain Bike Racing Page
- ensure the Track Racing Page hyperlinks to the Types of Track Page
- ensure the Mountain Bike Racing Page hyperlinks to both the Cross Country and Downhill Equipment Pages



You should now complete "Implementation Task 13 - Completing the Student Cooking Website". This can be found in the separate Implementation Tasks booklet.

When this is complete it is important to thoroughly test and then evaluate the website. This will be covered in booklet 3.