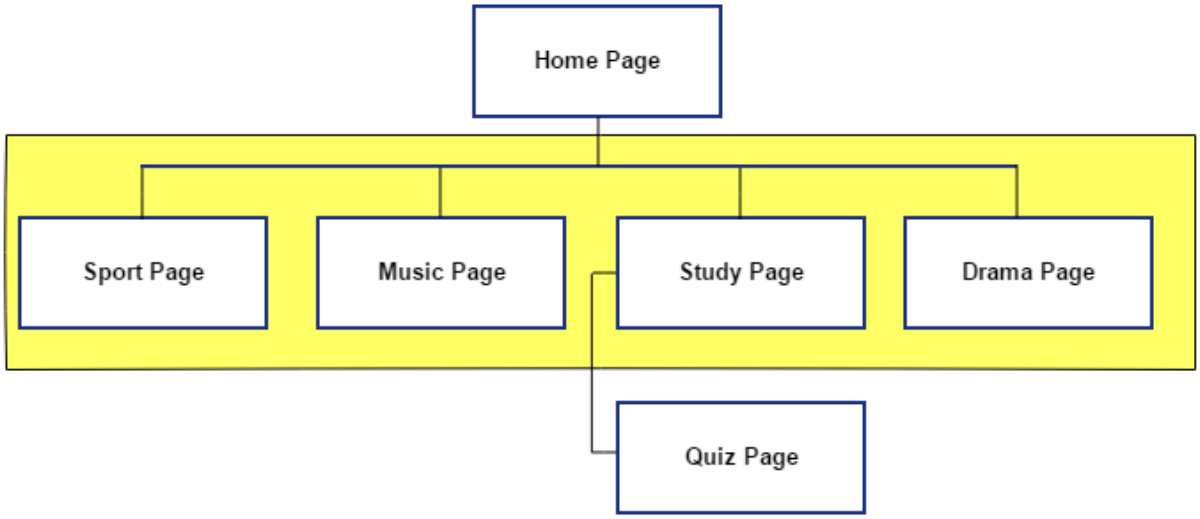
Appendix 12: design (WDD)

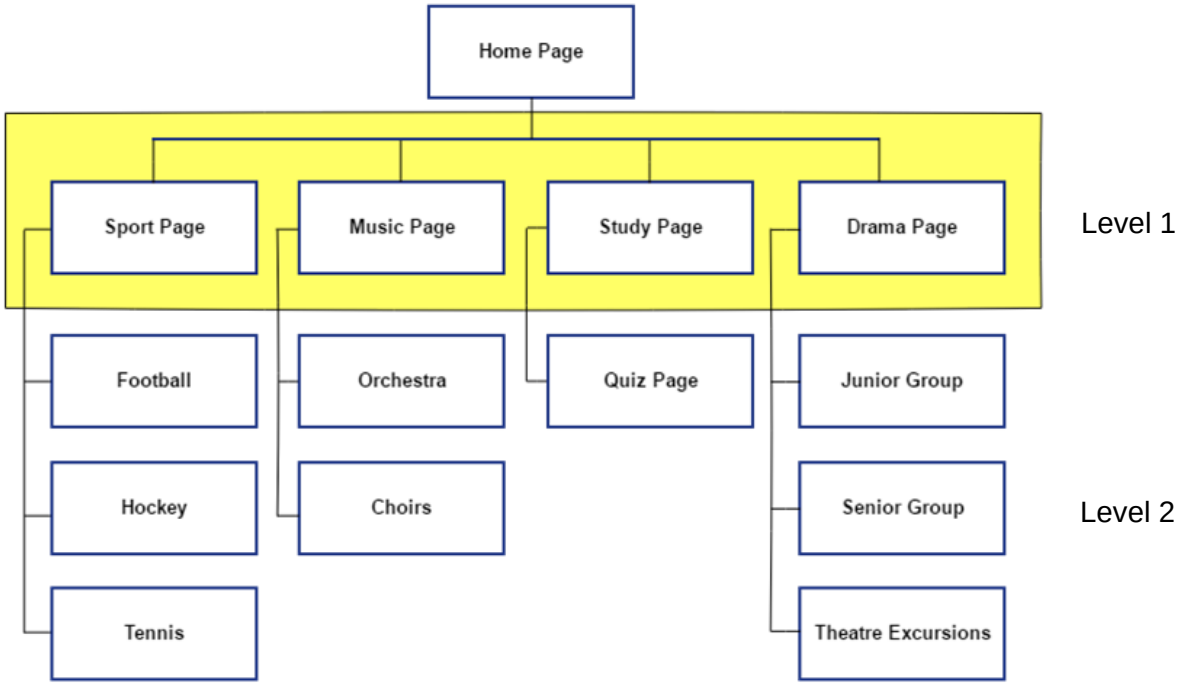
The design of the structure of a website is vital to ensure that users can easily find information on what could be thousands of individual pages. Hierarchical structures, using navigational bars, allow navigation through multi-level websites to be logical and straightforward.

There are many design notations used to demonstrate the navigational structure. However, SQA will always use a shaded background to show the pages that are part of the horizontal navigational bar, which are at the top of each page in the site. The home page naturally is included in the navigational bar. The horizontal lines show the multi-links that link them all together. The vertical lines show the links that are only one way, from top to bottom.



Example
Penny High School is developing a new website. The website will have a multi-level structure, consisting of a home page with a horizontal navigation bar that gives clear links to four main areas (pages). Each of the four main pages will have links to relevant sub-pages.

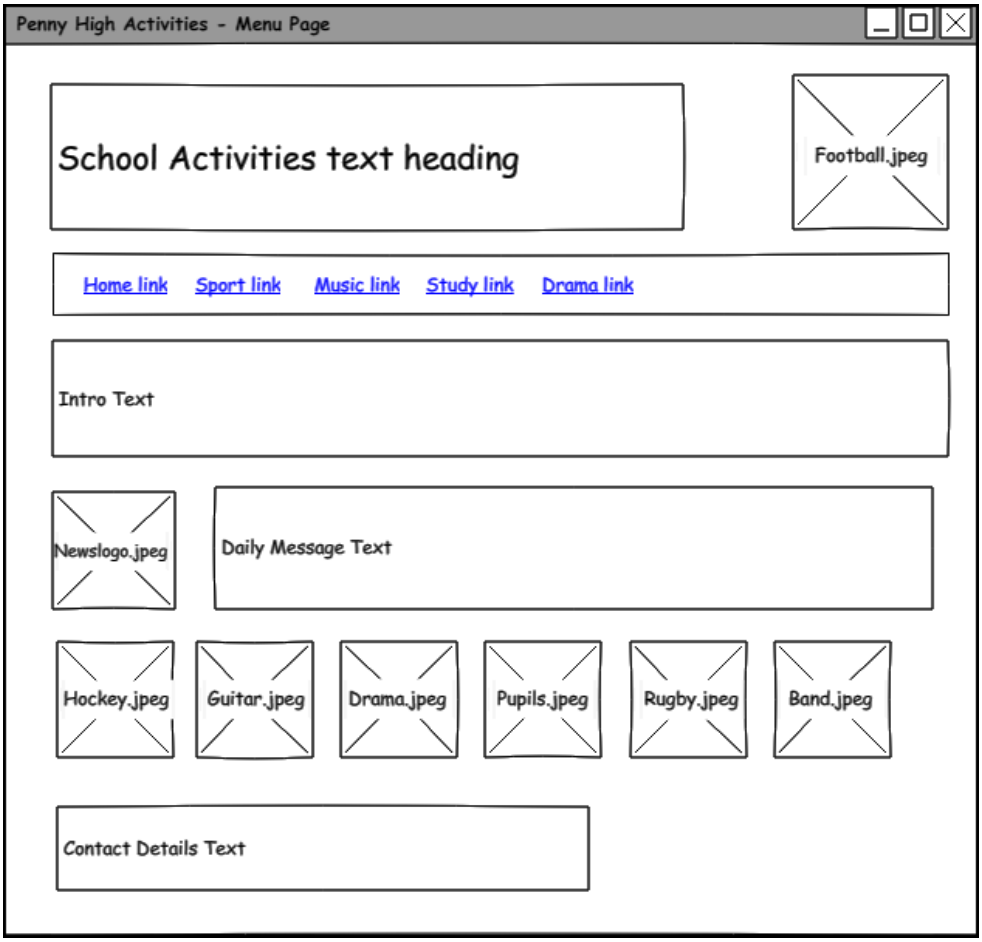
The following diagram shows the navigational structure of the Penny High School website:

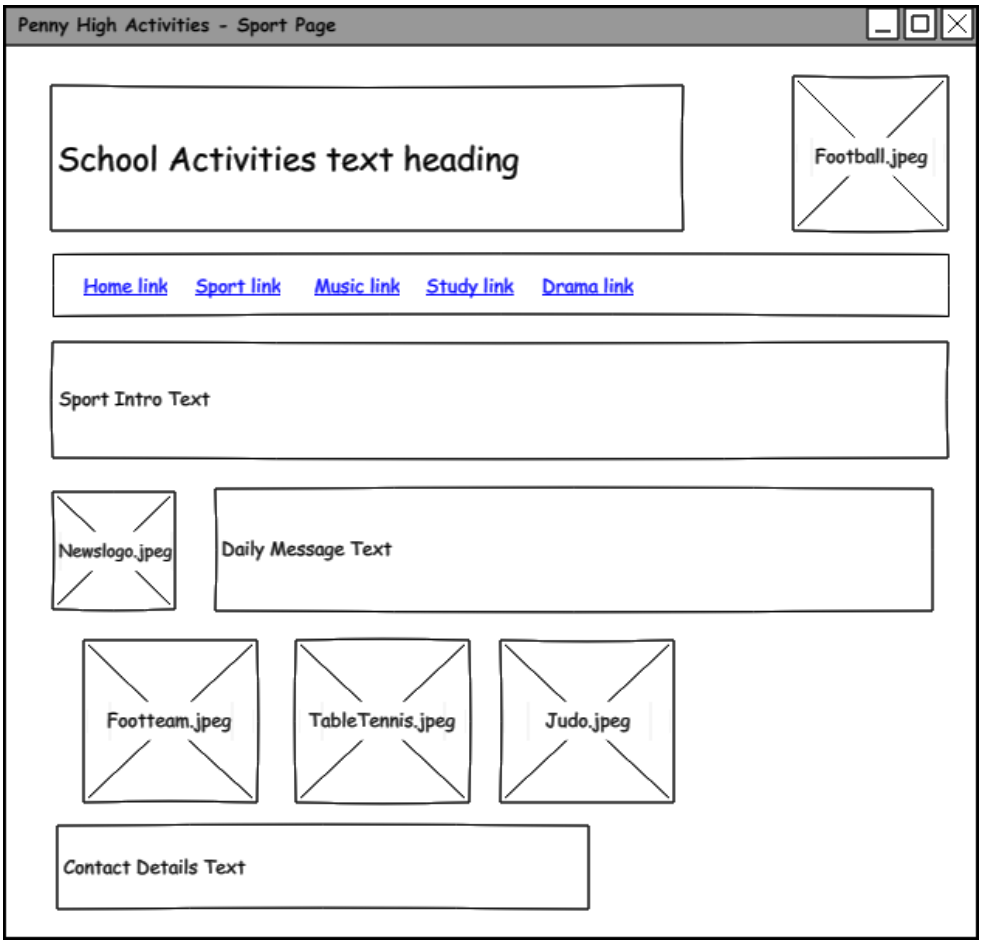


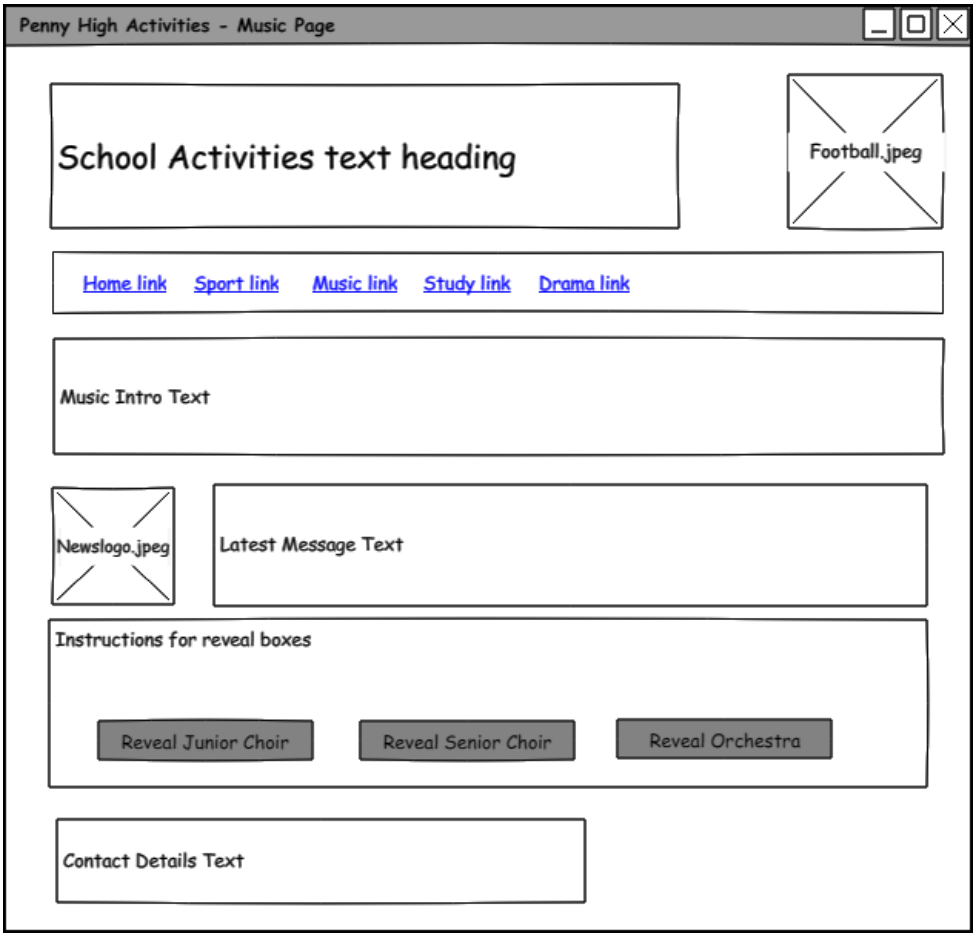
The user-interface planning should be illustrated using **wireframes**. A separate wireframe is needed for each page on a website. Each wireframe indicates the intended layout of the page and shows the horizontal and vertical position of:

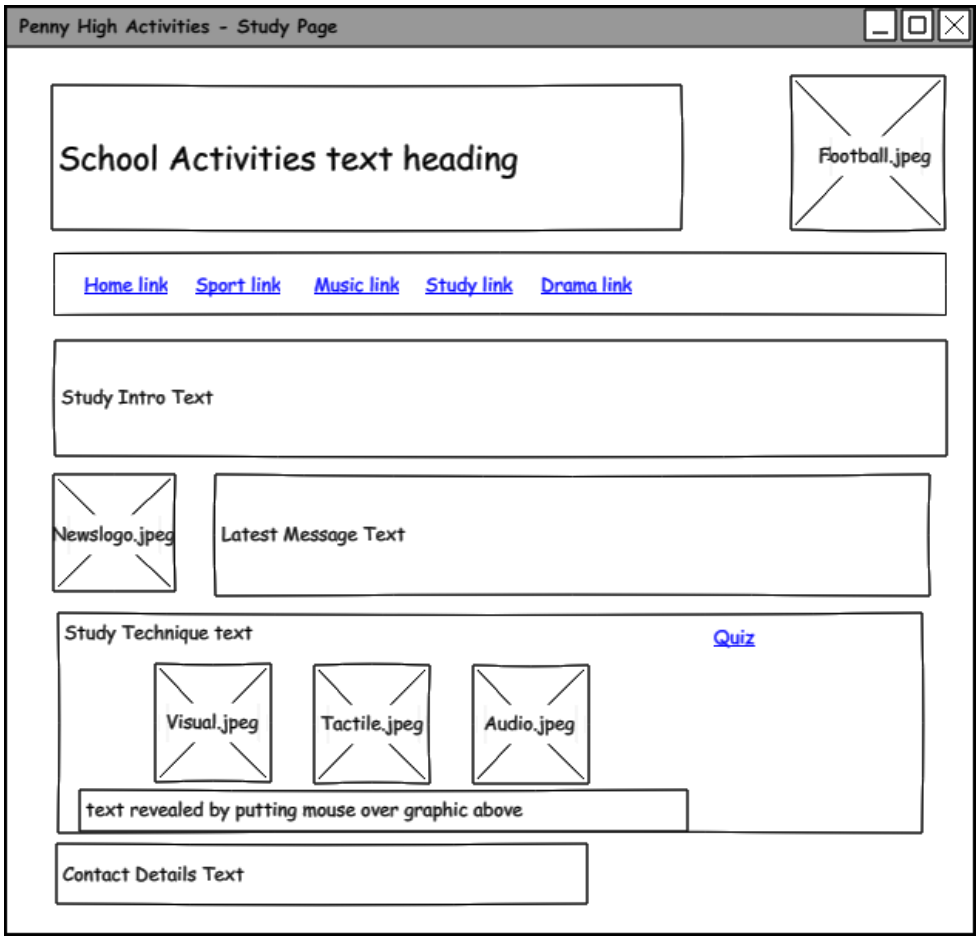
- navigational bars
- all text elements on the page
- any media elements (images, audio clips and video clips)
- elements that allow the user to interact with the page
- any form inputs

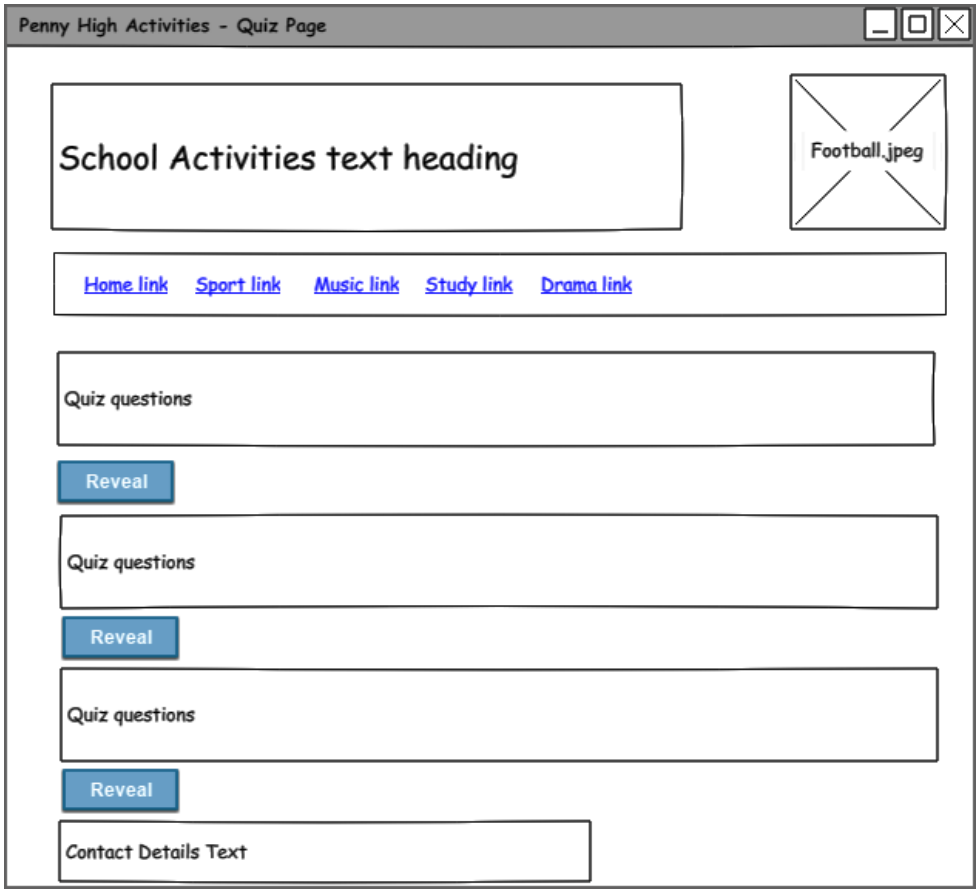
together with intended position and type of all hyperlinks on the page.

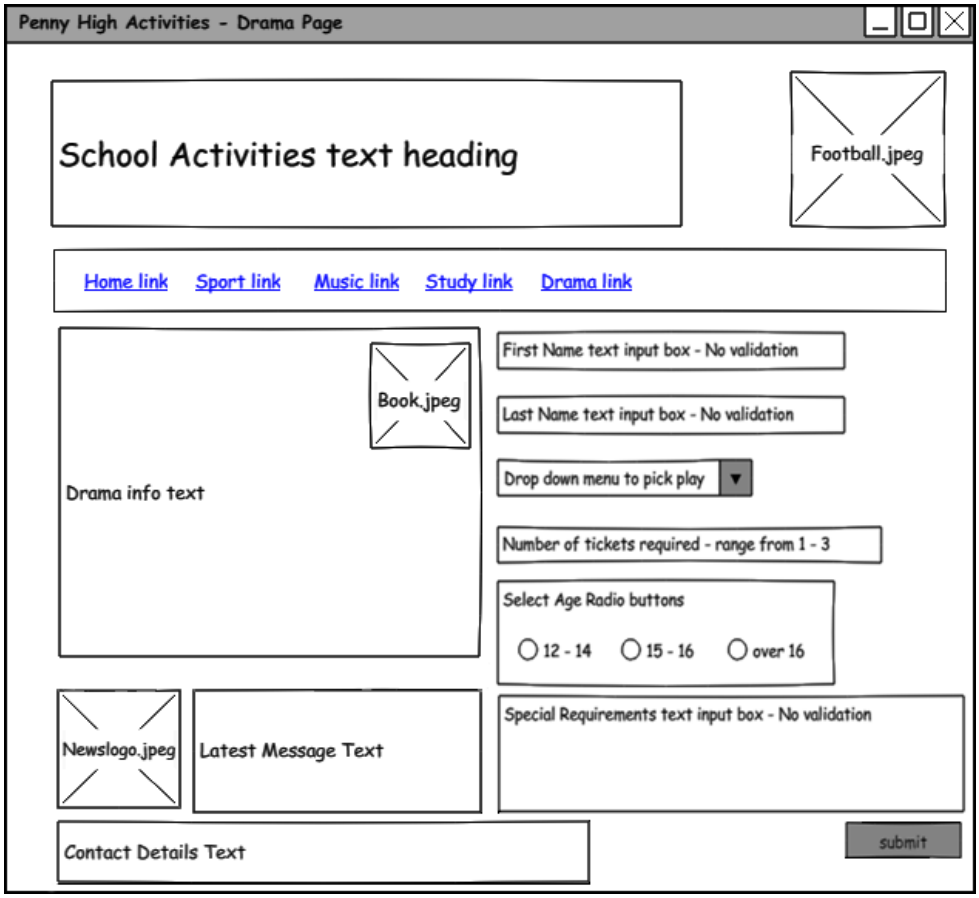

Penny High Activities - Menu Page
School Activities text heading
Football.jpeg
Home link
Sport link
Music link
Study link
Drama link
Intro Text
Newslogo.jpeg
Daily Message Text
Hockey.jpeg
Guitar.jpeg
Drama.jpeg
Pupils.jpeg
Rugby.jpeg
Band.jpeg
Contact Details Text


Penny High Activities - Sport Page
School Activities text heading
Football.jpeg
Home link
Sport link
Music link
Study link
Drama link
Sport Intro Text
Newslogo.jpeg
Daily Message Text
Footteam.jpeg
TableTennis.jpeg
Judo.jpeg
Contact Details Text


Penny High Activities - Music Page
School Activities text heading
Football.jpeg
Home link
Sport link
Music link
Study link
Drama link
Music Intro Text
Newslogo.jpeg
Latest Message Text
Instructions for reveal boxes
Reveal Junior Choir
Reveal Senior Choir
Reveal Orchestra
Contact Details Text


Penny High Activities - Study Page
School Activities text heading
Football.jpeg
Home link
Sport link
Music link
Study link
Drama link
Study Intro Text
Newslogo.jpeg
Latest Message Text
Study Technique text
Quiz
Visual.jpeg
Tactile.jpeg
Audio.jpeg
text revealed by putting mouse over graphic above
Contact Details Text


Penny High Activities - Quiz Page
School Activities text heading
Football.jpeg
Home link
Sport link
Music link
Study link
Drama link
Quiz questions
Reveal
Quiz questions
Reveal
Quiz questions
Reveal
Contact Details Text


Penny High Activities - Drama Page
School Activities text heading
Football.jpeg
Home link
Sport link
Music link
Study link
Drama link
Book.jpeg
Drama info text
First Name text input box - No validation
Last Name text input box - No validation
Drop down menu to pick play
Number of tickets required - range from 1 - 3
Select Age Radio buttons
12 - 14
15 - 16
over 16
Newslogo.jpeg
Latest Message Text
Special Requirements text input box - No validation
Contact Details Text
submit

Appendix 13: Cascading Style Sheets (CSS) — controlling appearance and positioning (WDD)

The following CSS declarations control the appearance and position of HTML page elements:

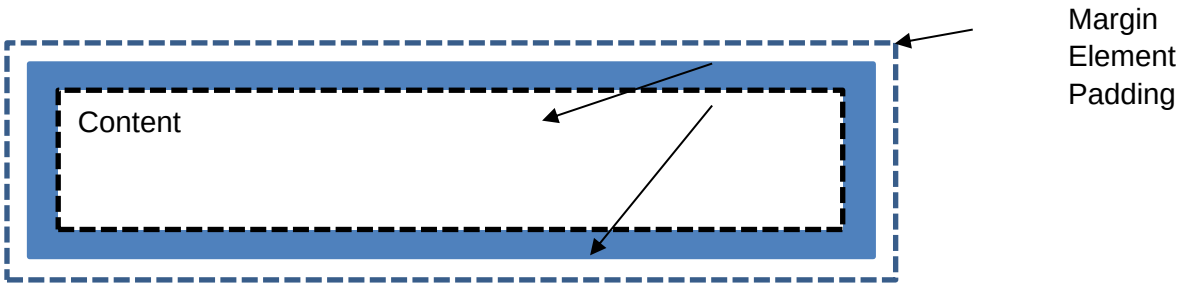
- display — block, inline, none
- float — left, right
- clear — both
- margins/padding
- sizes — height, width

The following examples have been taken from the *Higher Computing Science example website*. The example website can be downloaded as a zip file from the Higher Computing Science page on SQA's website. To view the full code in context, open and view the source code from the pages noted.

The box model

Margins and padding are used to push content away from the outer and inner edge of elements.

The box model allows us to define the space between elements.



Margin: declares a transparent area around the outside of an element. This pushes the element away from other adjacent elements.

Padding: declares a transparent area inside the edge of the element. This pushes content in from the edge of the element.

Universal selector

Browsers use default settings for margins and padding when displaying HTML elements. To override these defaults, a universal selector (*) can be used at the top of a stylesheet to set the margin and padding of every element to 0.

```
* {margin:0;padding:0}
```

Using the universal selector this way allows candidates to witness only the margins and paddings they actually code.

Note: universal selectors are not included in Higher content but can be useful when teaching.

Margins (all pages): main page areas
Margins can be declared as:

- margin
- margin-top
- margin-bottom
- margin-left
- margin-right

These properties may have the values:

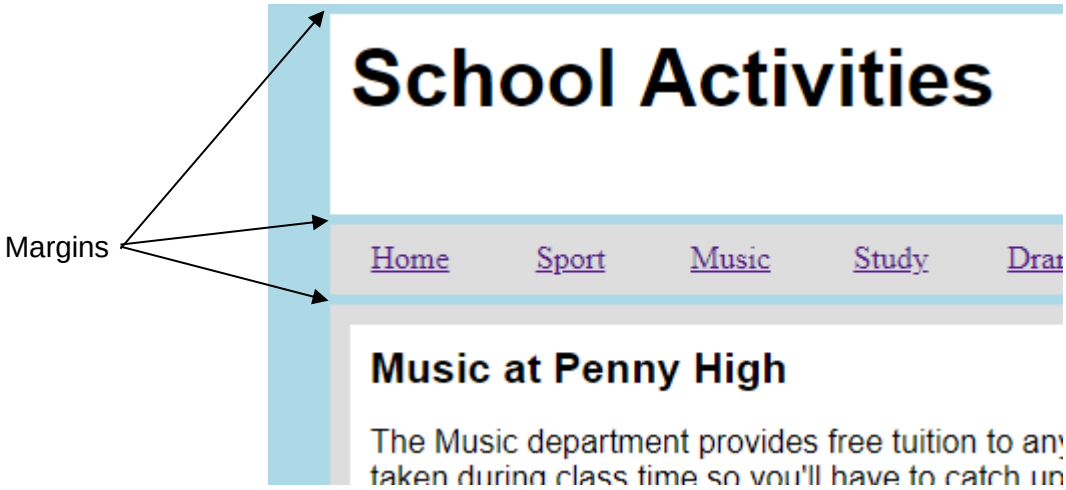
- auto (calculated by the browser)
- length (usually in pixels)

Margins can be declared on all four sides of an element simultaneously using the abbreviation: `margin:10px` .

In the example website, CSS declarations for the main page elements (<header>, <nav>, and <footer>) are used to separate out content when displayed.

The small gap at the top of each area is implemented by styling a margin of 5 pixels at the top of the four elements. For example:

```
header, nav, main, footer {margin-top:5px}
```



Grouping selectors using commas reduces the amount of code required and is more efficient than writing each CSS declaration separately.

Separate declarations

```
header {margin-top:5px}
nav {margin-top:5px}
main {margin-top:5px}
footer {margin-top:5px}
```

Grouped declaration

```
header, nav, main, footer {margin-top:5px}
```

Auto margins are used to position an element in the middle of the browser's window or within another element. In the example website, the 800 pixel wide page <body> element is always positioned in the middle of the window:

```
body{margin:auto}
```

Padding (drama page): sub-division of content
Padding can be declared as:

- padding
- padding-top
- padding-bottom
- padding-left
- padding-right

These properties may have the value:

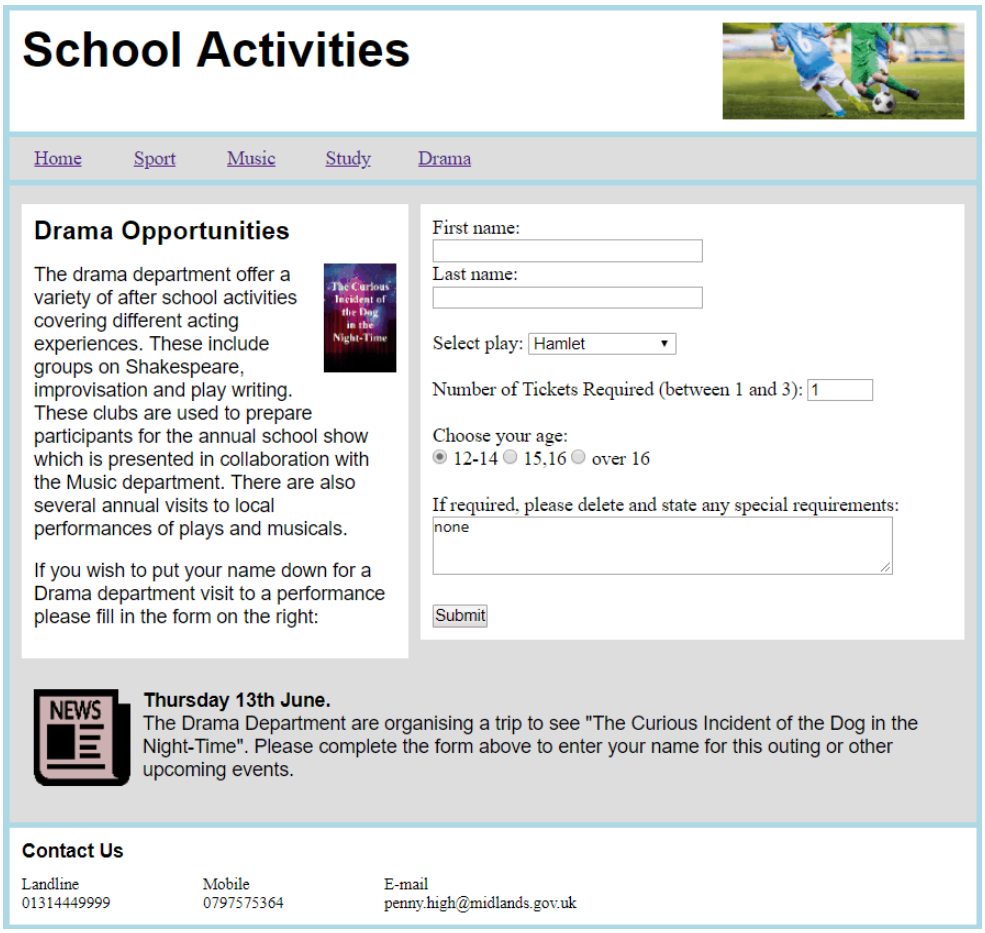
- length (usually in pixels)

Padding can be declared on all four sides of an element simultaneously using the abbreviation: `padding:10px`

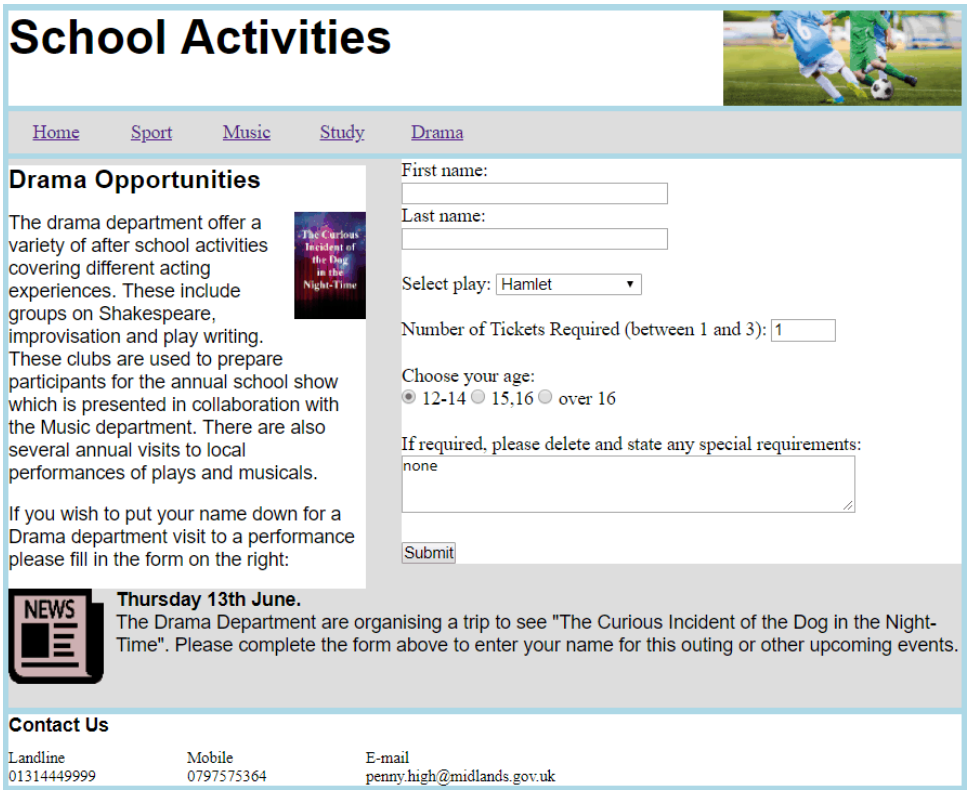
White space around page content, aids the usability of a web page. Appropriate use of white space can also aid readability of text.

Using padding to move content in from the edge of an element is one method to generate white space.

With padding: `header, div, main, footer, section {padding:10px}`



Without padding: `header, div, main, footer, section {padding:0px}`



Sizes — height/width (all pages)
Web page design and implementation involves creating areas that either have a fixed size or change size with content or display (changing window size or resolution).

In the example website, the <body> of each web page is set to a fixed width of 800px:

```
body{width:800px}
```

The height of the <header>, <nav> and <footer> elements are all set to a fixed size, remaining constant throughout the website:

```
header {height:80px}
footer {height:60px}
nav {height:35px}
```

The <main> element, which holds content of the pages, is not declared as a fixed size, as it changes according to the differing amount of content on each page.

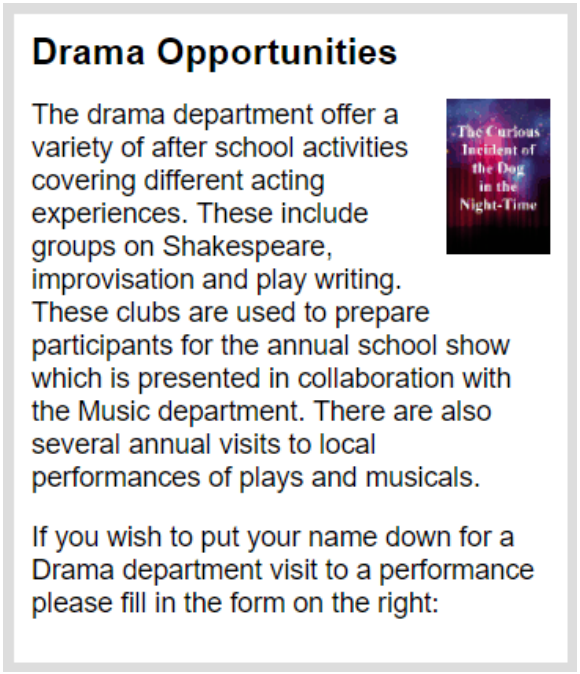
Float — left/right (all pages)
An element can be positioned on the left or right of its container, using the float property.

In the example website, the <img> element has been positioned on the right of its container, the <header> element:

HTML
<header>
<h1>School Activities</h1>
<img class="imageBanner" src="../images/activitiesBanner.jpg">
</header>

CSS
.imageBanner {width:200px;height:80px;float:right}

Float can also be used to word-wrap text around an image, as demonstrated in the drama page below. The addition of margins creates white space between the graphic and the text.



HTML

```
<p> <img class="imageIconRight"
src="../images/curiousIncident.jpg">
The drama department offer a variety of after school
activities covering different acting experiences;
Shakespeare, improvisation, play writing.  These clubs are
used to prepare participants for the annual school show
which is presented in collaboration with the Music
department.  There are also several annual visits to local
performances.</p><p>If you wish to put your name down for a
Drama department visit to a performance please fill in the
form:</p>
```

CSS

```
.imageIconRight {width:60px;height:90px;float:right;margin-
left:10px;margin-bottom:10px}
```

Clear (all pages):

The effect of floating elements continues until cancelled, using the clear property on a subsequent element.

To ensure that the four main page elements <header>, <nav>, <main> and <footer> start a new line and remain unaffected by any float properties applied elsewhere in the page, `clear:both` was implemented for these elements.

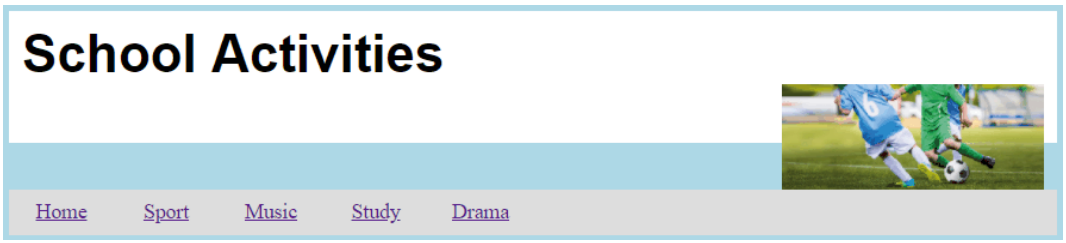
```
header, nav, main, footer {display:block;clear:both}
```

Display — block/inline/none (all pages):
HTML elements have default display values depending on the type of element. This value specifies how or if an element is to be displayed. The default display value for most elements is block or inline:

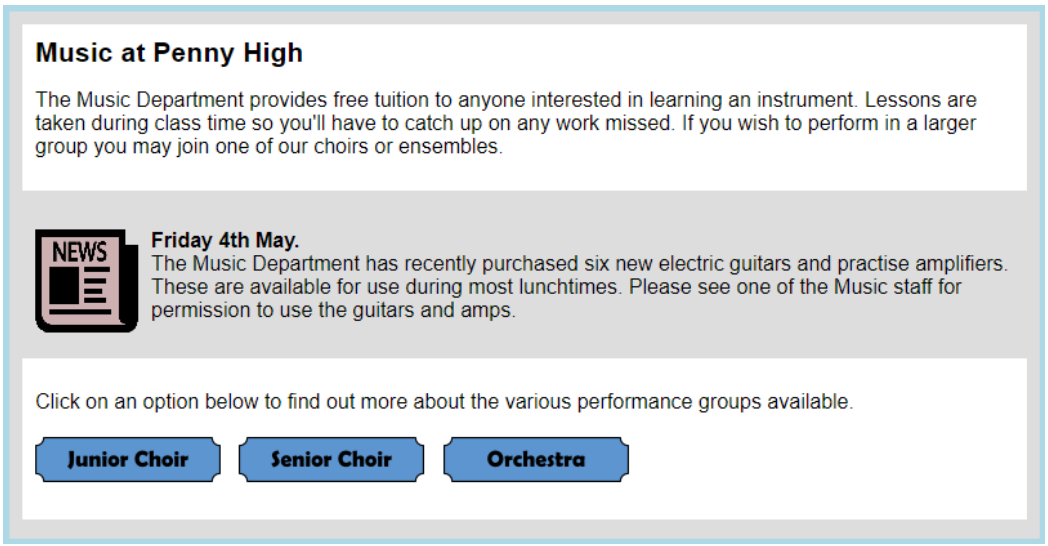
`display:block` — an element takes up the entire width of its container
`display:inline` — an element takes up only as much room as necessary
`display:none` — the element is not visible. The space where the element should be collapses as if there was no content in that place.

In the <header> element of the example website, the <h1> heading uses `display:inline`. This allows the image to be floated level with the heading.

If the <h1> element used `display:block` it would force the image to float on a new line.



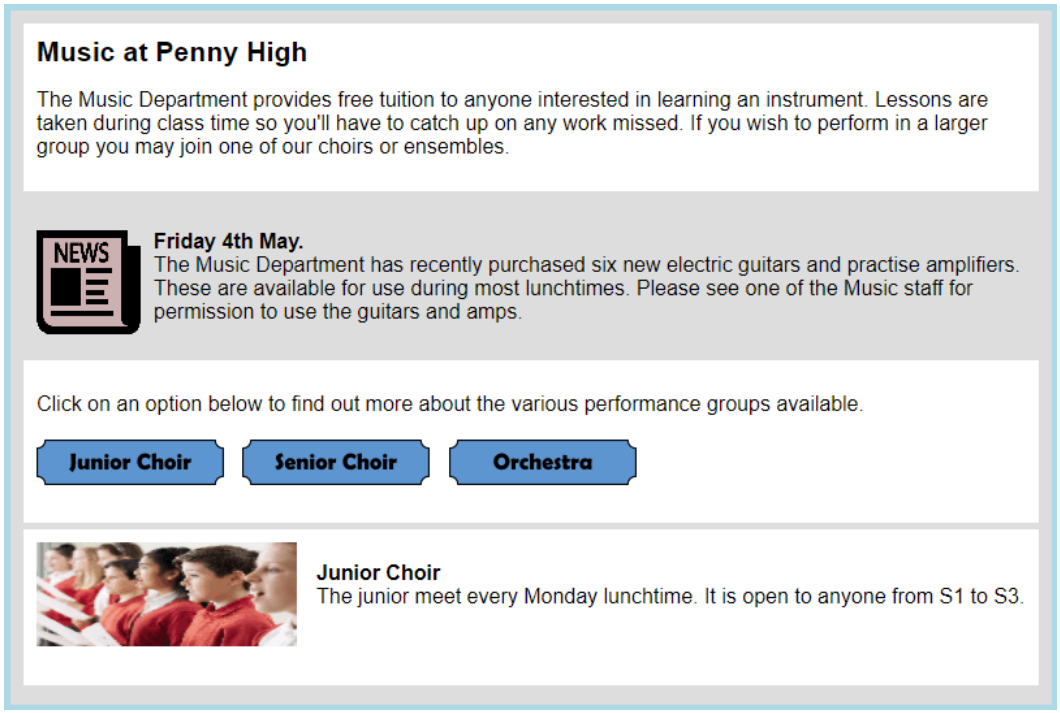
In the music page, `display:none` is used to hide three <section> elements when the page loads.



HTML

```
<section id="junior" style="display:none;height:100px">
<img class="largeImages" src="../images/juniorChoir.jpg">
<p><b>Junior Choir</b><br>The junior meet every Monday
lunchtime.  It is open to anyone from S1 to S3.</p>
</section>
```

The above section is revealed when the 'Junior Choir' image on the page is clicked. This is achieved using a JavaScript onclick event to change the display property of the <section> to block.



For more details, see appendix 17: JavaScript.

Appendix 14: Cascading Style Sheets (CSS) — horizontal navigation bar (WDD)

The HTML 5 <nav> element is used to contain website navigation links, usually as a navigation bar. A navigation bar is declared as an unordered list of hyperlinks:

```
<nav>
    <ul>
        …
        <li>hyperlink</li>
        <li>hyperlink</li>
        …
    </ul>
</nav>
```

The look and feel of a navigation bar is then created using CSS declarations.

List of hyperlinks before and after CSS declarations

Before CSS declarations are added, the list of HTML hyperlinks are displayed as a simple bullet point list:

HTML

```
<nav>
  <ul>
   <li><a href="home.html">Home</a></li>
   <li><a href="sport.html">Sport</a></li>
   <li><a href="music.html">Music</a></li>
   <li><a href="study.html">Study</a></li>
   <li><a href="drama.html">Drama</a></li>
  </ul>
</nav>
```

The addition of CSS declarations alter the position, look and behaviour of the list to create a navigation bar.

CSS

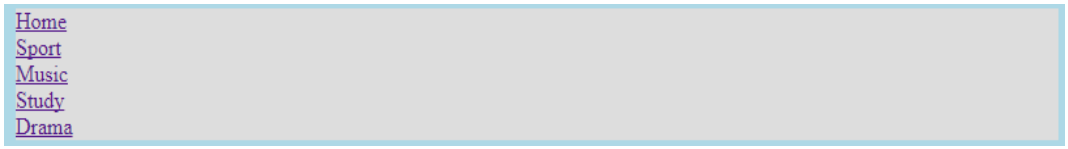
```
nav ul {list-style-type:none;}
nav ul li {float:left;width:80px;text-align:center}
nav ul li a {display:block;padding:8px}
nav ul li a:hover {background-color:#000;color:White}
```

The following examples breakdown the above code and provide explanations of each declaration. They have been taken from the navigation bar used within the *Higher Computing Science example website*. The example website can be downloaded as a zip file from the Higher Computing Science page on SQA's website. To view the full code in context, open and view the source code from any of the example web pages along with the external CSS file.

Example: removing bullets points

Declaring the `list-style-type` of the unordered list element as `none` removes the bullet points from the list:

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



Using descendant selectors (nav ul) ensures that the style is only applied to the ul element within the nav element.

Example: positioning list items horizontally

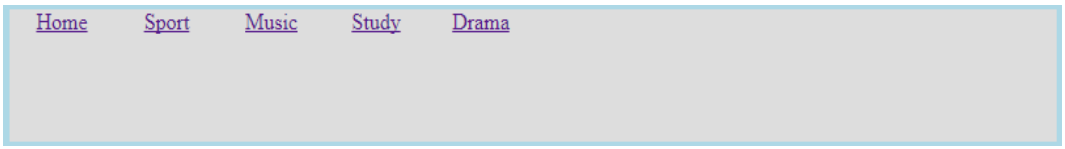
To ensure the list of hyperlinks is distributed horizontally, each <li> element is floated to the left:

```
nav ul li {float:left}
```

Example: spacing out each list item

To create space between each list item, a width is declared and the link text is centred within each width:

```
nav ul li {float:left;width:80px;text-align:center}
```



Example: creating clickable boxes
In most navigation bars, clicking the area around the hyperlink also selects the link. A clickable box area around the link text is achieved by displaying the <a> element as a block:

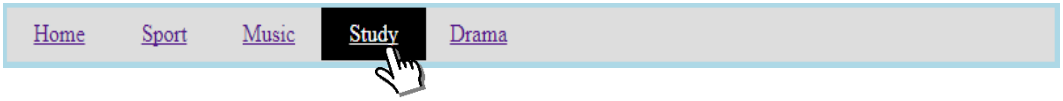
```
nav ul li a {display:block}
```

Example: controlling vertical alignment
The vertical positioning of the link text in the navigation bar is controlled by declaring the height of the <nav> element and including padding within the <a> element:

```
nav {height:35px}
nav ul li a {display:block;padding:8px}
```

Example: adding interactive colours
The state of the <a> element can be styled to change the background colour and text colour of each link, when the mouse hovers over an <a> element:

```
nav ul li a:hover {background-color:#000;color:White}
```

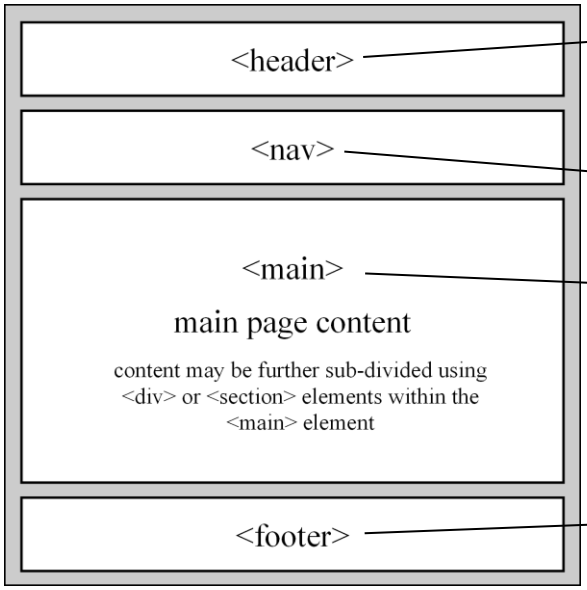


Appendix 15: HTML — page layout (WDD)

HTML 5 introduces new elements used to define different areas of a web page. Elements implemented at Higher level are:

- <header>
- <nav>
- <section>
- <footer>

These are implemented as shown below:



Contains the page banner, often including a title, graphic, company icon or link to the home page.

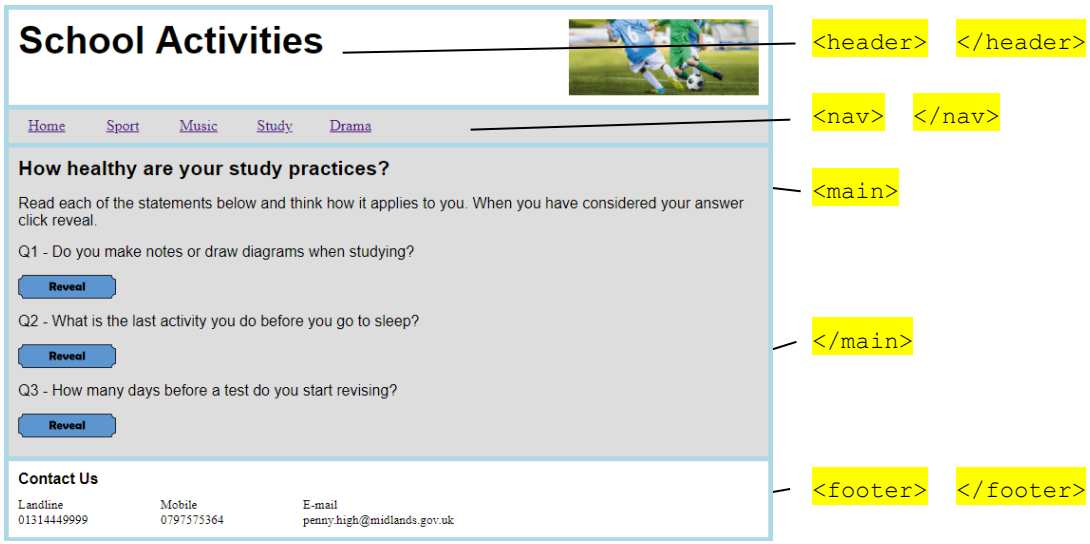
Contains the navigation bar/links.

Contains the page content: graphics, text, sound and video.

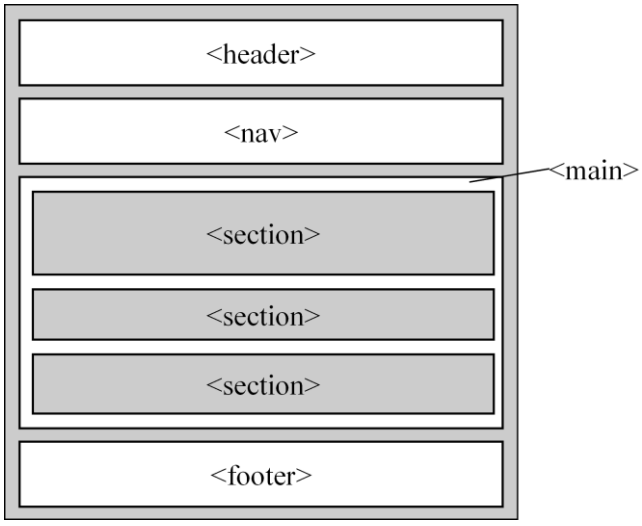
Contains general information, for example links to contact details, FAQs, copyright information or legal agreements.

The following examples have been taken from the *Higher Computing Science example website*. The example website can be downloaded as a zip file from the Higher Computing Science page on SQA's website. To view the full code in context, open and view the source code from the pages noted in each of the examples.

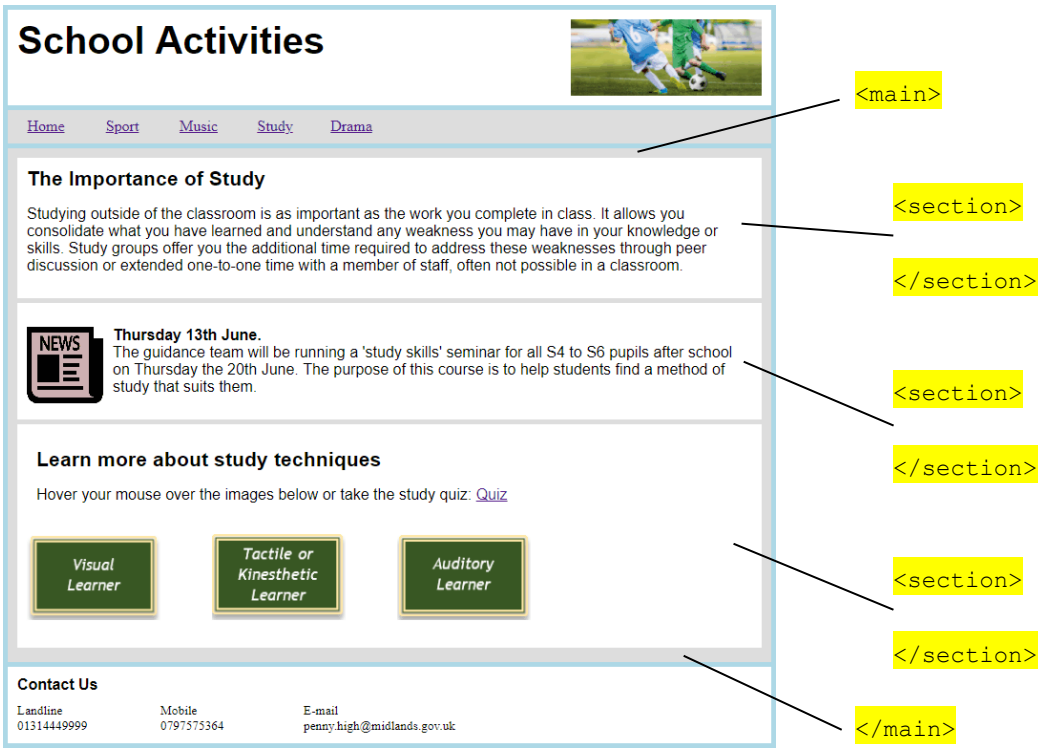
Simple page layout (study quiz page): example 1
The example below implements a simple page, with a main element used to contain the page content. The content is then sub-divided using section elements:



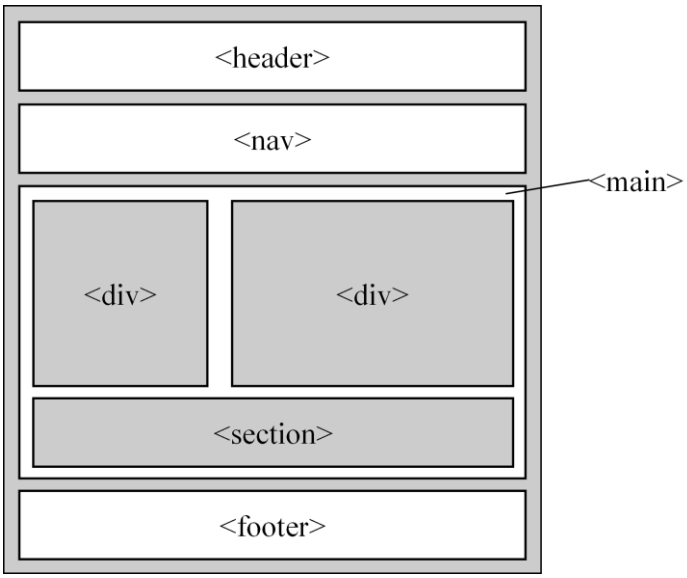
Page layout with sub-divided content (study page): example 2
Page content can be sub-divided into different areas using both `<section>` and `<div>` elements. Using both elements allows the sub-divided content to be independently styled, without the need to implement classes or ids:



The following example website uses the study page to demonstrate the above layout:

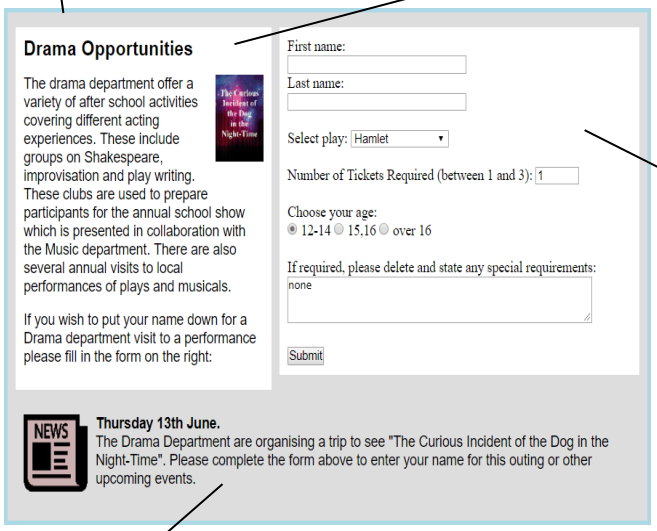


Page layout with side-by-side content (drama page): example 3
Page content can also be styled with CSS, to position page components side-by-side. In this example, the content has been sub-divided using three section elements, which are then controlled using float and clear CSS declarations:



The drama page shows the area containing the form, positioned to the right of the drama text area using CSS.

<main> — this is used to wrap around all the main content of the page. This is styled a light-grey colour:



 — the first div is styled white and uses `float:left` and `width:300px` to implement position and size.

 — the float left applied to the first div ensures that this area positions itself in line with the first. The margin between the two sections was created by pushing the left-hand edge of this section away from the left-hand edge of the page using `margin:330px`.

<section> — the style `clear:both` ensures that the final section sits on a new line and does not overlap the two sections above. The outer edge of the section is not apparent as it uses styles contained within the id ‘newsArticle’ to appear the same light-grey colour as the main element.

Appendix 16: HTML — forms (WDD)

The HTML <form> element defines a form that is used to collect user input:

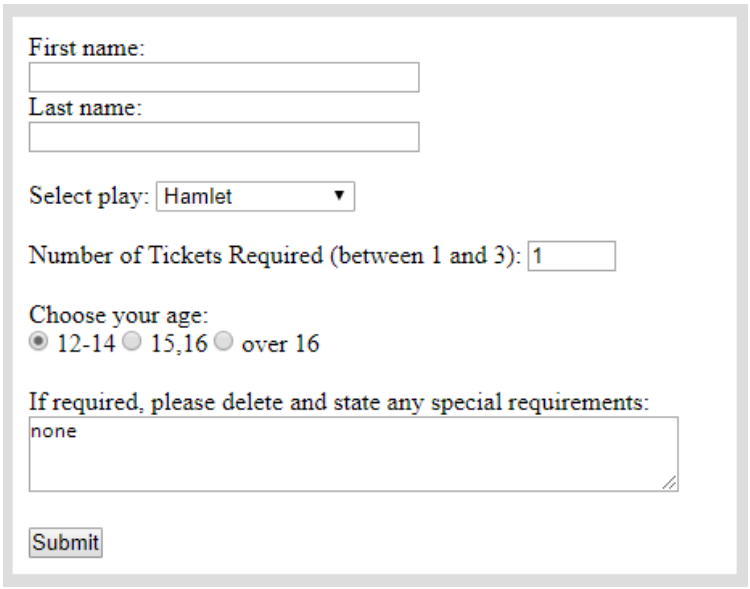
```
<form>
.
form elements
.
</form>
```

HTML 5 can be used to create and perform client-side validation on forms, without the need for JavaScript. Form elements implemented at Higher level are:

- input
  - text
  - number
  - textarea
  - radio
  - submit
- select

Where appropriate, form inputs will be validated using length, presence and range checks.

The following examples have been taken from the form used within the drama page of the *Higher Computing Science example website*. The example website can be downloaded as a zip file from the Higher Computing Science page on SQA's website. To view the full code in context, open and view the source code from the drama page.



Input: example 1 — text
The example below implements two text input boxes, including length and presence checks:

```
<form>
First name:<br>
<input type="text" name="firstname" size="30" maxlength="15"
required><br>
Last name:<br>
<input type="text" name="lastname" size="30" maxlength="15"
required>
</form>
```

The above code includes the following:

`type="text"`	Identifies input type of the form element as text.
`name="firstname"`	The name attribute is required when a form's data is submitted to a server for processing. Although submitting a form to a server is not required until Advanced Higher, it is good practice to include the name attribute in all form elements.
`size="30"`	Width of the input box, in characters, when displayed in a browser.
`maxlength="15"`	Length check limiting input to 15 characters.
`required`	A presence check is applied to the input.

Input: example 2 — number
Number input may be limited to a minimum value, maximum value or both:

```
Number of Tickets Required (between 1 and 3):
<input type="number" name="tickets" min="1" max="3">
```

The above code includes the following:

`type="number"`	Identifies input type of the form element as numeric.
`min="1" max="3"`	A range check to ensure values entered are >=1 and <=3.

Input: example 3 — textarea
A larger text box, for use with extended text input, can be implemented using the textarea form element:

```
If required, please delete and state any special requirements:
<textarea name="message" rows="3" cols="55"> </textarea>
```

The width and height of the textarea element is set using rows and columns.
If required, a length and presence check can be applied to the above input element.

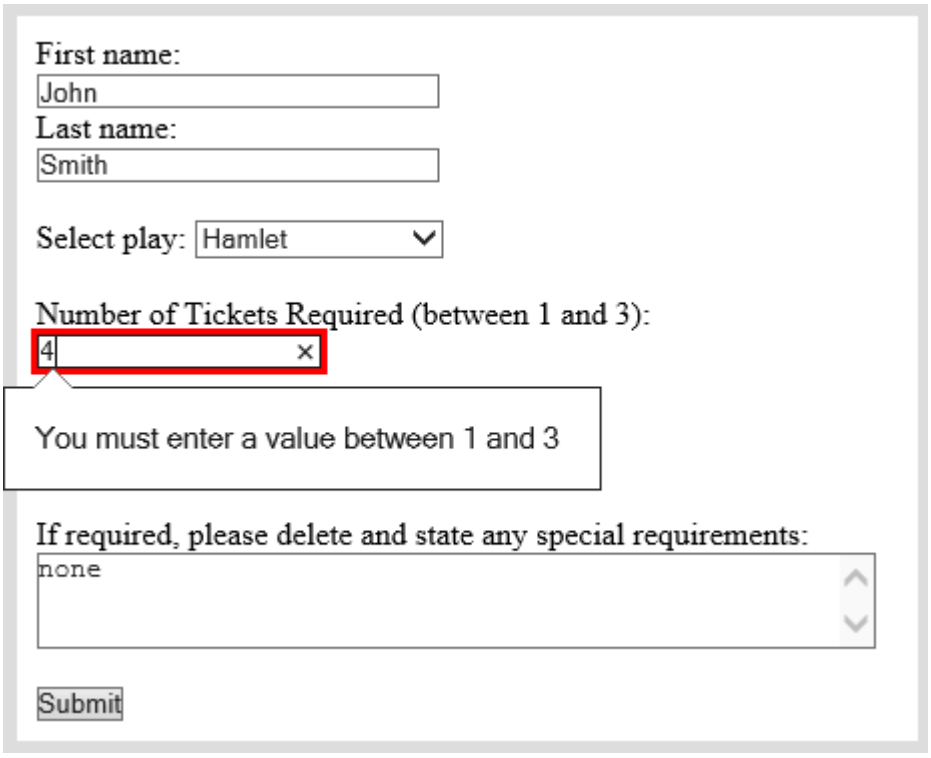
Input: example 4 — radio buttons
Radio input can be implemented using multiple input elements of type radio:

```
Choose your age:<br>
<input type="radio" name="age" value="12 to 14"> 12-14
<br>
<input type="radio" name="age" value="15 or 16"> 15,16
<br>
<input type="radio" name="age" value="17 or over"> over 16
```

When submitted, the above form would return the name attribute “age” along with one of the listed values: 12 to 14, 15 or 16, 17 or over. Although Higher forms are not submitted to a server, it is good practice to include both the name and value attributes.

If the
 elements are omitted from the form, the radio buttons align horizontally.

Input: example 5 — submit
When a form is submitted, the browser shows any validation errors (check browser versions, as this is browser dependent).



To allow candidates visual acknowledgement that an action is performed when the submit button is clicked, a JavaScript onclick event can be applied to the button as shown below:

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

Visual acknowledgement that the form is submitted can be helpful to candidates who are not yet experiencing a response generated by server-side processing.

Select: example 1 — drop-down menu
The select element is used to create a list of possible inputs in the form of a drop-down menu. Input choices are placed inside option elements:

```
Select play:
  <select name="play">
    <option value="hamlet">Hamlet</option>
    <option value="godo">Waiting for Godo</option>
    <option value="brothers">Blood Brothers</option>
    <option value="curious">Curious Incident</option>
  </select>
```

As previously stated, the name and value attributes are not required at Higher, but it is good practice to include both.

Select: example 2 — drop-down menu with size attribute
The size attribute can be used within the select element, to display a set number of options. If the number of options is larger than the size attribute, a scroll bar will automatically appear:

```
<select name="play" size="3">
```

Select: example 3 — drop-down menu with multiple attribute
To allow users to select more than one option, the multiple attribute is used with the select element:

```
<select name="play" size="4" multiple>
```

Pre-populating form input
To aid user input, form elements can be given values that are displayed when the web page loads. These can be left unchanged, deleted or edited by the user, when they are completing the form.

Example 1: text
The value attribute can be used to pre-populate text input elements:

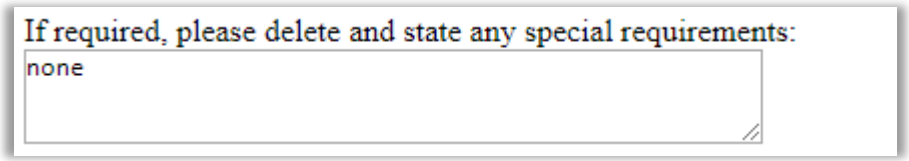
```
<input type="text" name="firstname" value="forename"
size="30" maxlength="15" required>
```

Example 2: number
The value attribute is also used to pre-populate numeric input elements:

```
<input type="number" name="tickets" value="1" min="1"
max="3">
```

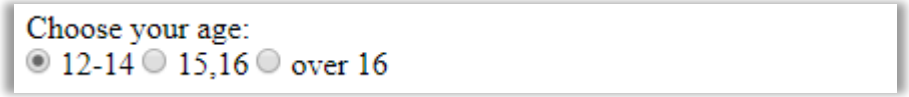
Example 3: textarea
To pre-populate a textarea element, the text is included between the start and end elements:

```
<textarea name="message" rows="3" cols="55">none</textarea>
```



Example 4: radio
Checked is used to initially check one of the radio buttons in a form:

```
<input type="radio" name="age" value="12 to 14" checked>
12-14 <br>
```



Appendix 17: JavaScript (WDD)

JavaScript events (onmouseover, onmouseout and onclick) are used to implement interactive web content.

Events are placed within HTML elements as shown below:

```
<img  onmouseover="       ">
```

The inverted commas contain the action to be executed when the event takes place. Possible actions include:

- hiding page elements
- revealing page elements
- changing the position of an element
- changing the size of an element
- changing the colour of an element
- changing the look of text

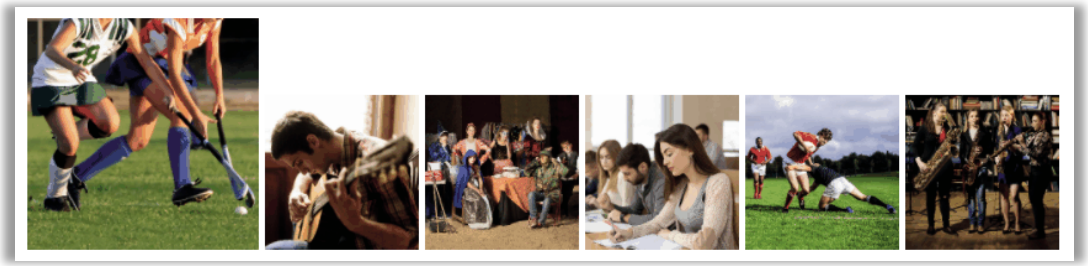
Actions can be executed by:

- referring to the element containing the JavaScript event
- referring to a different element, identified by an id
- calling a JavaScript function containing the actions

The following examples have been taken from the *Higher Computing Science example website*. The example website can be downloaded as a zip file from the Higher Computing Science page on SQA's website. To view the full code in context, open and view the source code from the pages noted in each example.

Interactively changing the size of a graphic (home page): example 1
Inline JavaScript can be used to increase and decrease the size of a graphic, as a mouse pointer passes over and out of a graphic.

```
<img src="../images/hockey.jpg"
onmouseover="this.style.width='150px';this.style.height='150px'"
onmouseout="this.style.width='100px';this.style.height='100px'">
```



This example contains two events, `onmouseover, onmouseout` used to execute four actions:

```
width='150px'  height='150px'  width='100px'  height='100px'
```

Each action in a JavaScript statement is separated by a semi-colon.

Each action in the above events are prefixed by `this.style`.

`this`	meaning this element. In example 1 this refers to the graphic (img) element containing the event.
`style`	the style of this element will be altered using a CSS declaration

Interactively changing the size of a graphic (home page): example 2
JavaScript functions may be called to increase and decrease the size of a graphic, as a mouse pointer passes over and out of a graphic.

```
<img src="../images/guitar.jpg"
onmouseover="displayLarger(this)"
onmouseout="displaySmaller(this)">
```

The JavaScript functions above are passed the parameter `this`, referring to the element containing the event `displayLarger(this)`

The JavaScript functions are placed within a <script> element in the <head> section of the HTML document:

```
<script>
function displayLarger(my_image)
      {my_image.style.width='150px';
      my_image.style.height='150px';}


function displaySmaller(my_image)
      {my_image.style.width='100px';
      my_image.style.height='100px';}
</script>
```

The parameter is used by each function to implement the same actions as example 1.

Implementing example 2 as a function, allows the same code to be called from multiple events:

```
<img src="../images/drama.jpg"
onmouseover="displayLarger(this)"
onmouseout="displaySmaller(this)">
<img src="../images/study.jpg"
onmouseover="displayLarger(this)"
onmouseout="displaySmaller(this)">
<img src="../images/rugby.jpg"
onmouseover="displayLarger(this)"
onmouseout="displaySmaller(this)">
```

Creating a rollover image (sports page): example 1
Rollover images can be created using two JavaScript events. The first event (`onmouseover`) displays an alternate graphic when the mouse passes over the image element. The second event (`onmouseout`) displays the original graphic when the mouse pointer leaves the image.

The following is the view in browser when the mouse pointer is not over any image:

The following is the view in browser when mouse passes over the right-hand image:

This can be implemented using the inline JavaScript shown below:

```
<img class="sportImage" src="../images/karate.jpg"
onmouseover="this.src='../images/basketball.jpg'"
onmouseout="this.src='../images/karate.jpg'">
```

The action `src=' '` points the image element towards a given address**.**

Creating a rollover image (sports page): example 2
A rollover image can also be achieved using two functions. The first function is called using onmouseover. The second is called using onmouseout. Each function is passed the img element as a parameter using this. As before each function changes the src of the image:

HTML
```
<img class="sportImage" src="../images/football.jpg"
onmouseover="displaySport1A(this)"
onmouseout="displaySport1B(this)">
```

JavaScript
```
<script>
function displaySport1A(my_image)
      {my_image.src ='../images/netball.jpg';}
function displaySport1B(my_image)
      {my_image.src ='../images/football.jpg';}
</script>
```

Highlighting text by dynamically changing its colour (home page):
Text may be highlighted by changing its style when the mouse pointer passes over the element containing the text.

The example below highlights a paragraph element dark red, by using an onmouseover event:

```
<p onmouseover="this.style.color='darkRed'">
<img class="imageIconLeft" src="../images/newsIcon.png" >
Wednesday 12th June.<br>  The Computing …  … Room B9.
</p>
```

Highlighting text by dynamically changing its colour (study page):
To implement an action on a different page element, an `id` is required to identify the element the action will be performed on.

An element is identified using `document.getElementById('      ')`

The example below highlights the paragraph text when the mouse passes over the image contained in the paragraph:

```
<p id="highlight"><img class="imageIconLeft"
src="../images/newsIcon.png"
onmouseover="document.getElementById('highlight').style.colo
r='darkRed'"><b>Thursday 13th June.</b><br>  The guidance …
… suits them. </p>
```

Revealing an element using onclick (study quiz page):
When a web page loads, elements may be initially hidden by applying the CSS declaration `display:none`. This can be implemented using an inline or internal style, or by implementing an external style (using a class, as shown below):

CSS

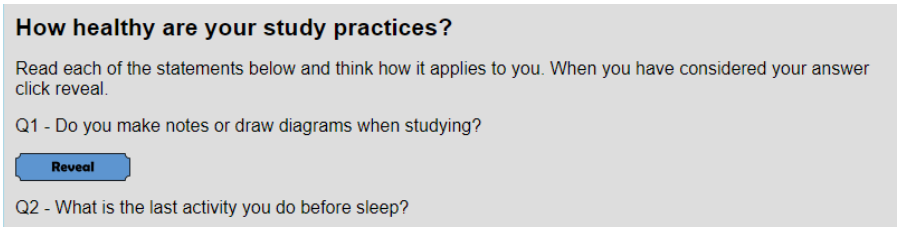
```
.hidden{display:none}
```

The graphic element in the example (reveal.png) contains an onclick event that uses the paragraph element's id to execute the action 'display=block' on the paragraph. The result of this action is that the paragraph becomes visible on the page.

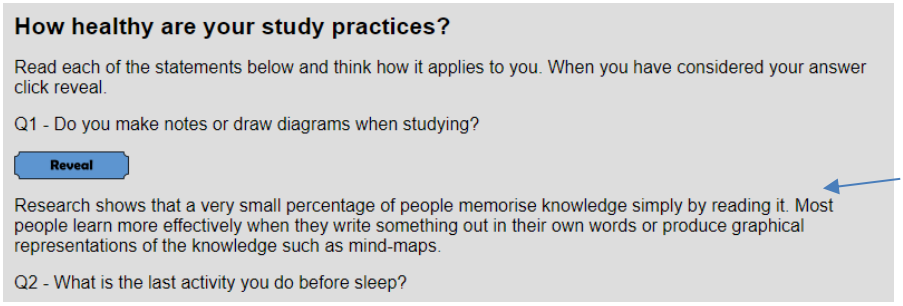
HTML

```
<p> Q1 - Do you make notes or draw diagrams when studying? </p>
<img class="reveal" src="../images/reveal.png"
onClick="document.getElementById('reveal1').style.display='block'">
<p id="reveal1" class="hidden"> Research shows that a very small
percentage of people memorise knowledge simply by reading it.  Most
people learn more effectively when they write something out in their
own words or produce graphical representations of the knowledge such
as mind-maps.</p>
```

The following is the view in browser when page loads:

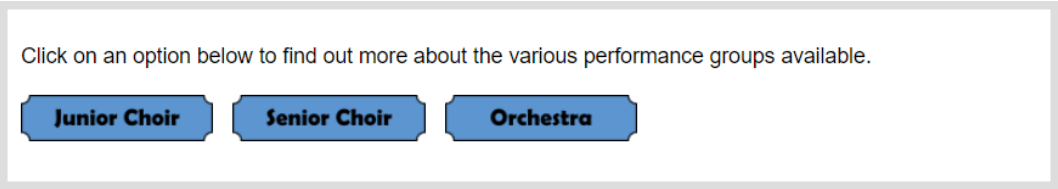


The following is the view in browser after 'Reveal' has been clicked:

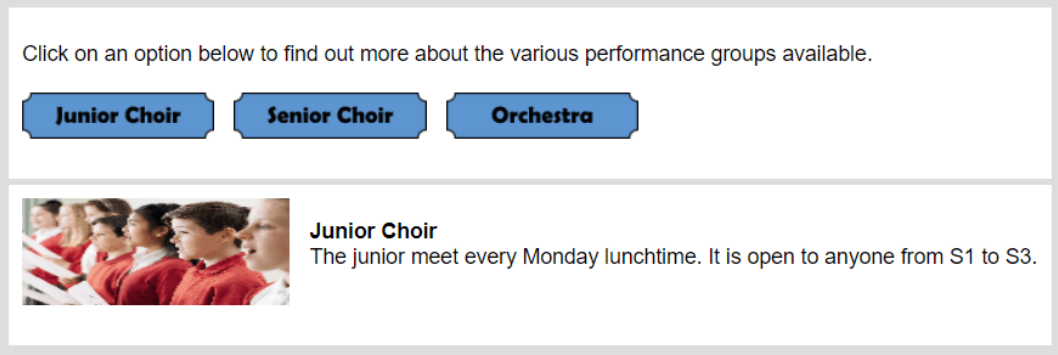


Revealing one of three hidden elements using onclick (music page):
Image elements, with associated onclick events, can be used to offer users a choice of what they view on a web page. Each onclick event is used to reveal a hidden <section> element containing a graphic and text relating to each option.

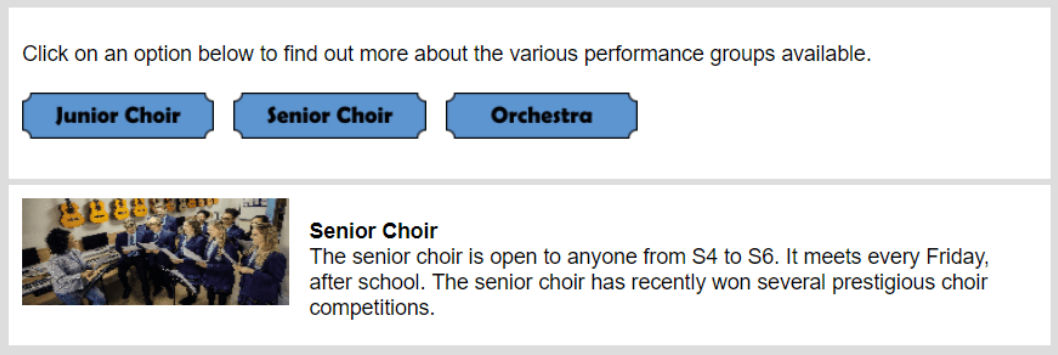
The following is the view in browser when page loads:



The following is the view in browser when the 'Junior Choir' graphic is then clicked.



The following is the view in browser when the 'Senior Choir' button in then clicked.



Each onclick event calls a function:

```
<p> Click on an option below to find out more about the
various performance groups available. <br><br>
<img class="musicOptions"
src="../images/juniorChoirOption.png" onclick="displayJC()">
<img class="musicOptions"
src="../images/seniorChoirOption.png" onclick="displaySC()">
<img class="musicOptions"
src="../images/orchestraOption.png" onclick="displayO()">
</p>
```

The three section elements, containing the graphic and text to be revealed, are each identified by an `id` and initially hidden using `display:none`.

```
<section id="junior" style="display:none">
<img class="largeImages" src="../images/juniorChoir.jpg">
<p><b>Junior Choir</b><br>The junior meet every Monday
lunchtime.  It is open to anyone from S1 to S3.</p>
</section>
<section id="senior" style="display:none">
<img class="largeImages" src="../images/seniorChoir.jpg">
<p><b>Senior Choir</b><br>The senior choir is open to anyone
from S4 to S6.  It meets every Friday, after school.  The
senior choir has recently won several prestigious choir
competitions.</p>
</section>
<section id="orchestra" style="display:none">
<img class="largeImages" src="../images/orchestra.png">
<p><b>Orchestra</b><br>The orchestra is not age limited,
instead being open to anyone who plays their instrument at
grade 3 level or higher.</p>
</section>
```

Each of the three JavaScript functions called by the onclick event reveals one of the three `<section>` elements and hides the other two using three `style.display` actions:

```
<script>
function displayJC() {
   document.getElementById("junior").style.display="block";
  document.getElementById("senior").style.display="none";
  document.getElementById("orchestra").style.display="none";
}

function displaySC() {
   document.getElementById("junior").style.display="none";
  document.getElementById("senior").style.display="block";
  document.getElementById("orchestra").style.display="none";
}
function displayO() {
   document.getElementById("junior").style.display="none";
  document.getElementById("senior").style.display="none";
  document.getElementById("orchestra").style.display="block";
}
</script>
```

Appendix 18: testing (WDD)

Usability testing
Usability testing involves systematic observation to determine how well people can use a product. In this case, the product will be the low-fidelity prototypes of a website. The goal with usability testing is to recreate real-world scenarios where the tester will actually be able to use your product. Then, by observing their behaviour, you will be able to understand what could be done better. In this case, the product will be the low-fidelity prototypes of a website. This helps to eliminate design problems at an early stage, before money has been spent implementing the design.

The testers may be given:

- a persona — this may relate to the age or experience that the tester should exhibit
- test cases — a set of actions executed to verify a particular feature or function of the website
- scenarios — they may be asked to use the website to place an order or book flights

They use the low-fidelity prototypes under a variety of conditions, while they are observed.
The observers make notes about any difficulties that the testers experienced and what alterations are required to the website design to make it easier to use the website.

Testing websites
There are a number of tests you should carry out on your website to ensure that it meets the functional requirements.

- Input validation:
  - check that every field in a form has the correct validation by trying to get every field on the form to accept incorrect data
- Links and navigation:
  - test the navigational bar links take you to the correct pages
  - test all external links work correctly
  - test that all pages can get back to the home page
  - test all internal links work correctly
  - test to check if there are any orphan pages (pages that are not linked to any others)
- Media content:
  - ensure that the text, graphics and video display correctly and in the position in which it was designed to appear

Compatibility testing
This is when you test your website to ensure that it works in the same way across a range of platforms.

Types of compatibility testing include:

Browser testing	It is important that your website will work on all the main browsers, for example Chrome, Firefox, Internet Explorer, Safari, and Opera. Your customers will not use your website if it does not function properly on their chosen browser.
Device type	You should check that your website is accessible on tablets, smartphones and desktop computers, as there are so many different types of hardware with different size screens available.

Common compatibility testing exposes the following types of problem:

- changes in font size
- changes in the user interface
- alignment issues
- changes in CSS style and colour
- scroll bar related issues
- content or label overlapping
- broken tables or frames

Copyright acknowledgements

All images Shutterstock:

Page 111,112,113,115,121 and 122: School activities banner — 553188940

Page 112,114 and 123: Drama opportunities — 329907647

Page 112,115,116,122 and 123: News icon — 735471220

Page 116 and 135: Junior choir — 432363805

Page 131: Hockey — 132311264

Page 131: Guitar — 145132324

Page 131: Drama — 752638612

Page 131: Studying — 549484090

Page 131: Rugby — 325167287

Page 131: Wind band — 723313330

Page 132: Footballer lifting trophy — 370092275

Page 132: Table tennis — 87611998

Page 132: Karate — 740133061

Page 132: Basketball — 198943043

Page 135: Senior choir — 588701573

Page 137: Man designing a website — 698323777

Administrative information

Published: August 2021 (version 2.3)

History of changes

Version	Description of change	Date
2.0	Table header on page 10 changed from ‘Software’ to ‘Database’. Course support notes added as appendix.	June 2018
2.1	Appendix 5 ‘range and precision’ section — references to 'accuracy' changed to 'precision'. Appendix 10 and appendix 11 — wildcard notations changed from Access specific (*) and (?) to generic (%) and (_), reflecting how they will be presented in the assessment. Appendix 12 — distorted graphics replaced.	August 2018
2.2	Page 101 (appendix 11 — SQL): we have amended the guidance on the UPDATE query to clarify the difference between National 5 and Higher.	October 2020
2.3	Page 37 — National 5 content changed from EU GDPR to UK GDPR, in line with change made to National 5 course specification.	August 2021

Note: you are advised to check SQA’s website to ensure you are using the most up-to-date version of this document.