SCHOLAR Study Guide

Higher Computing Science
Unit 4: Web design and development

Authored by:

Ian King (Kelso High School)

Mark Tennant (Community School of Auchterarder)

Charlie Love (CompEdNet)

Andy McSwan (Knox Academy)

Reviewed by:

Jeremy Scott (George Heriot's School)

Previously authored by:

Jennifer Wilson (Denny High School)

Acknowledgements

Contents

Topic 1

Analysis

Contents

Prerequisites

From your studies at National 5 you should already know:

- the term end users relates to the people who will use the website;

- the different types of end users of websites;

- that some users have specific requirements when browsing a website;

- the functional requirements of a website are the tasks that it must perform;

- the functional requirements can be used to evaluate if the completed site is fit for purpose.

Learning objective

By the end of this topic you should be able to:

- identify the end-user requirements of a website as relates to the design and implementation at this level;

- identify the functional requirements of a website as relates to the design and implementation at this level.

1.1 End User Requirements

To create a successful product, the design team need to understand the intended users of the web site. In order to do this, the members of the team will spend a significant amount of time interviewing users, working with them in focus groups and engaging in workshops around the development of the web site.

During this process of determining the user, the design team will develop an understanding of what users need and the tasks that users will perform with the web site. In order to do this a number of documents are created to help inform the design team about the users and their motivations.

Personas

A first step for the design team is to develop Personas for each of the target user types. Personas are fictional characters created to represent the different user types that the design team have encountered or that the team believe are important to the project.

User stories

A user story is a brief statement that identifies the user and his / her need. It is a direct statement that relates to a specific persona. The personas created by the design team will have a general level of detail however the user story will provide specific details about a task that the user will be engaged in. One persona may have several user stories developed around it. User stories help to document the practical differences in need among those users.

User scenarios

A user scenario expands upon already developed user stories by including details about how a system might be interpreted, experienced, and used. User scenarios provide detail about the user's goal, detail any assumed knowledge the user has and the level of experience the user has.

Just as with user stories, the design team may imagine several scenarios for each persona group.

Use case

A use case is a list of steps a user would take to perform an action. A typical use case will start by detailing how the user got to the current position and then goes through every step until the user completes the operation or fails. Use cases can be informed and / or verified by the results of usability testing.

User devices and software

It is worth considering how the user is likely to access the website. What type of computer (desktop, laptop, tablet or smartphone), what resolution it will be displayed at and what web browsing software will they use.

To get an understanding of the current trends have a look at the stat counter website (http://gs.statcounter.com). You can find the most commonly used browsers, platforms and screen resolutions being used for browsing the web.

1.2   Functional Requirements

Functional requirements are defined as what the product should do and are contained in the scope within the product backlog.

Consider this example of a forum website which allows users to create a user account. As a simple rule of thumb, the functional requirements describe what the software will be able to do.

> **Example**
>
> The new user page should present the user with a form enabling them to create a new account by entering username and email address. On submitting the form, the user receives a welcome email containing a link to confirm the account. This link should take them to their account details where they can perform the following tasks:
>
> - enter /change personal information;
>
> - upload an avatar image;
>
> - change forum preferences;
>
> - change password;
>
> - delete account;
>
> - confirm their account details by pressing a submit button.
>
> The welcome email will warn the user that their account will expire if they do not go to the link and confirm their details within a certain time period.
>
> The site should reject usernames or email addresses which belong to existing users.
>
> The site should reject obscene or racist usernames.
>
> The site should be secure.

1.3   Learning points

Summary

You should know be able to:

- identify the end-user requirements of a website as relates to the design and implementation at this level;

- identify the functional requirements of a website as relates to the design and implementation at this level.

1.4 End of topic test

End of topic test: Analysis   Go online

Q1: The team creating an application write the following text:

"Sally has family around the world and shares photos and messages with them throughout her day. She needs quick access to her photos and her contacts and the ability to send messages."

This is an example of:

a) a use case
b) a user scenario
c) a user story
d) system requirements

....

Q2: Two activities that can be used to ensure user requirements are correctly understood are:

a) data dictionary and environment analysis.
b) performance and satisfaction criteria.
c) interviews and focus groups.
d) independent test groups and data flow.

....

Q3: What is a target audience?

a) People watching a show
b) People who the website is aimed at
c) The person who is designing the website
d) People who create the web design software

....

Q4: What is **not** contained in a description of purpose?

a) Who will be using the information system
b) Why the information is being stored
c) Who wrote the information
d) Date it was created

Topic 2

Design

Contents

Prerequisites

From your studies at National 5 you should already know how to:

- describe and create a website structure with a home page, linked multimedia pages, and external links;

- describe, exemplify and implement, taking into account end-user requirements, effective user-interface design (visual layout and readability) using wireframing:

  - navigational links;
  - consistency across multiple pages;
  - relative vertical positioning of the media displayed;
  - file formats of the media (text, graphics, video, and audio);

- understand how the Copyright, Designs and Patents Act 1988 applies to web content (text, graphics, video, and audio);

- compare a range of standard file formats:

  - audio standard file formats WAV and MP3 in terms of compression, quality, and file size;
  - bit-mapped graphic standard file formats JPEG, GIF, and PNG in terms of compression, animation, transparency, and colour depth;

- describe the factors affecting file size and quality, relating to resolution, colour depth, and sampling rate;

- describe the need for compression;

- describe, exemplify and implement prototyping (low fidelity) from wireframe design at this level.

Learning objective

By the end of this topic you should be able to:

- describe and develop the structure of a multi-level website with a home page and two additional levels;

- create a website taking into account end-user requirements and device type, an effective user-interface design (visual layout and readability) using wire-framing;

- create a prototype (low fidelity) web page from a wireframe design;

- design, describe and apply the key aspects of user-centred design including using analysis tools to capture user requirements;

- develop user profiles, user personas, user stories, user scenarios and use cases;

- describe, explain and apply the development of user interfaces using prototypes (from low to high fidelity including wireframes);

Learning objective continued

- create a wireframe design that can:

  - show a horizontal navigational bar;
  - plan the horizontal and vertical positioning of the media;
  - indicate form inputs;
  - indicate file formats of the media in a web page (text, graphics, video, and audio).

2.1   Multi level sites

> **Learning objective**
>
> By the end of this section you should be able to:
>
> - explain a multi-level structure;
>
> - identify the navigation structure for your website.

2.1.1   Linear and hierarchical structures

Navigation of a website describes how the user moves around the website, and what order they will visit pages in. Navigation can be achieved by the user clicking on hyperlinks or hotspots, clicking on buttons, or selecting options from a menu.

From your studies at National 5, you may have learned about linear and hierarchical navigational structures.

Figure 2.1 shows a linear structure.

Figure 2.1: Linear structure

Figure 2.2 shows a hierarchical structure.

Figure 2.2: Hierarchical structure
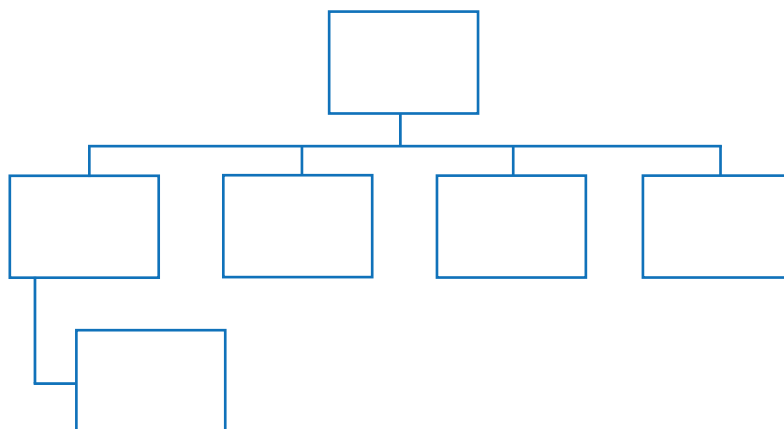
Linear and Hierarchical navigation structures are simplistic models of how pages can be organised in a website. In reality, links between pages (and even external sites) can be much more complex. A hierarchical structure may underlie the site, but in reality there will be many ways of navigating through the content.

Figure 2.3: A more realistic site structure
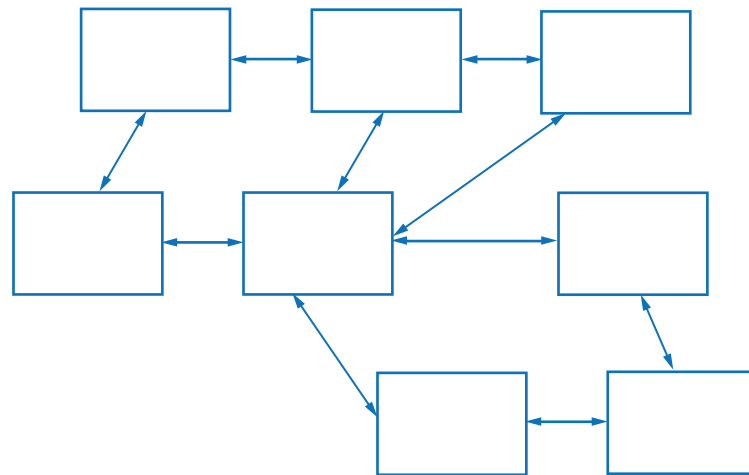
2.1.2 'Multi-level' navigation

'Multi-level' navigation is often used to organise pages in a logical fashion, meaning that hierarchical structures are extended to include sub-pages, sub-sub-pages, and so on. Often it is possible through the use of menus to return 'up' several layers in one click, though it may have taken several clicks to get to the page you are on. (As a general rule of thumb, most users get fed-up if it takes more than 3 clicks to find the page required - another challenge for designing a good navigation structure).

Figure 2.4: Multi-level navigation, showing the multiple return paths
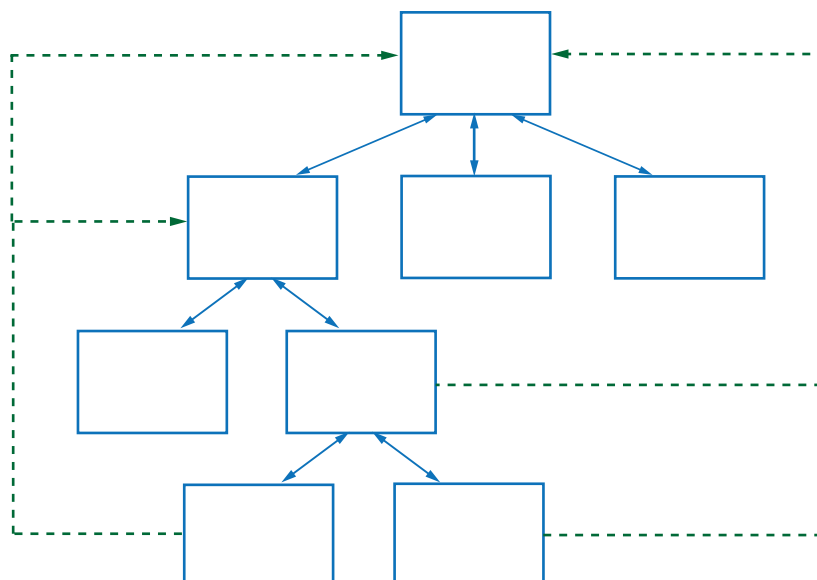
2.1.3   Navigation bar

In the example below you can see that the second level of the diagram has a shaded box around it, this is the format that will be used to identify what pages will be used in the site's navigation bar.
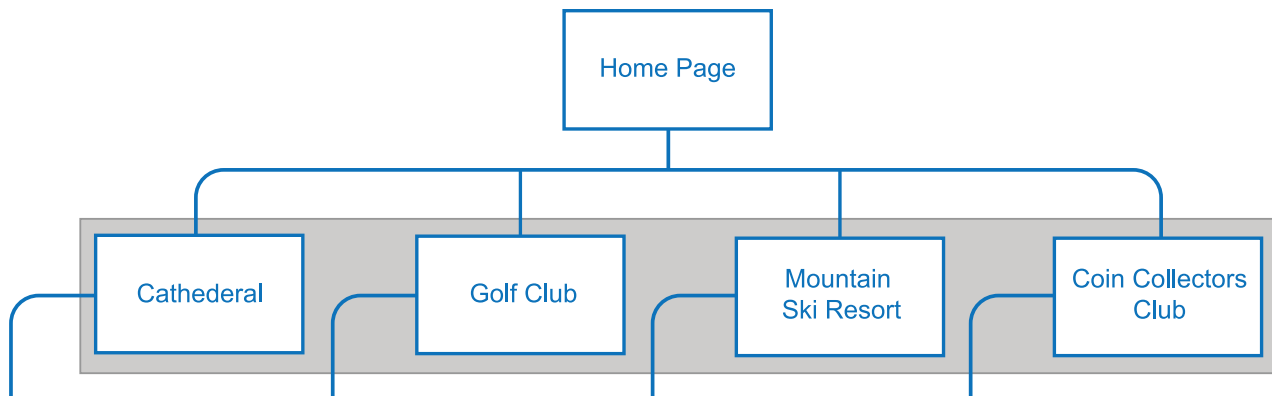
2.1.4   Navigational aids

A major aspect of designing navigation structures is considering how users will be informed of their location within the website. There are a number of techniques to achieve this.

You need to ensure that users cannot get confused when navigating your website. With this in mind you will have to consider a number of different navigational aids:

- **History** - A complete list of all nodes or links that have been visited. Each previous screen is represented once in the history file. The user can return by simply clicking on the relevant highlighted link. The history file will only contain recently visited links and will not contain older links.

- **Breadcrumbs** - A 'trail' which has been left and can lead the user back to wherever they have originated.

- **Highlighting (nodes/links)** - Clicking on hyperlinks to follow links to different screens. They will change colour once they have been selected.

- **Backtracking** - Use of a back button which will link to the last screen visited.

- **Bookmarks** - A list of pages that have been selected and saved by the user so that they can visit these pages again.

Quiz: Site structure (5 min)

Answer the following questions:

Q1: Explain the difference between a linear structure and a hierarchical structure. You may include a diagram.

....

Q2: Explain what is meant by a 'multi-level' structure.

....

Q3: Explain why 'breadcrumbs' may be useful in a multi-level navigation structure.

Activity: Website investigation

Investigate a website you use frequently and create a navigational structure for it.

Some websites that you could use are:

- Your school's website

- Heriot-Watt University's website (https://www.hw.ac.uk/)

- The BBC website (www.bbc.co.uk)

- The Register (https://www.theregister.co.uk)

- Wired (http://www.wired.co.uk)

2.2 End-user requirements

Learning objective

By the end of this section you should be able to:

- design, describe and apply the key aspects of user-centred design including using analysis tools to capture user requirements;

- develop user profiles, user personas, user stories, user scenarios and use cases.

User-centred design (UCD) is a method that ensures that a program, product, app or website will be easy to use. This approach to the design of a product focuses on the user: their needs, their wants, and their limitations and these are considered at each and every stage of development.

User-centred design is a multi-stage problem solving process. At regular points during the development of a product, designers must analyse and imagine how users are likely to use the

product. User-centred design also requires that designers carry out usability testing, with real world users, to ensure that the product meets the needs of the users and functions in the way that users would expect it to function.

This kind of testing is necessary as it is often very difficult for the designers of a product to understand intuitively what a first-time user of their design experiences, and what each user's learning curve may look like.

The key strength of user-centred design is that it focuses on how users can, want or need to use the product rather than forcing the users to change their behaviour to accommodate the product.

2.2.1   Principles of User centred design

The model for user-centred design is an agreed international standard (**ISO 9241-210 Human-centred design for interactive systems**) which describes the approach to design as "an approach to systems design and development that aims to make interactive systems more usable by focusing on the use of the system and applying human factors/ergonomics and usability knowledge and techniques."

This standard approach to user-centred design defines the principles of the approach as:

1. The design is based upon an explicit understanding of users, tasks and environments.

2. Users are involved throughout design and development.

3. The design is driven and refined by user-centred evaluation.

4. The process is iterative.

5. The design addresses the whole user experience.

6. The design team includes multi-disciplinary skills and perspectives.

This approach ensures that users are always considered during the development of the product.

The design is based upon an explicit understanding of users, tasks and environments

It is key that the design team understand how the intended users will use the product: the context of use. To achieve this the design team need to consider the three main features of user experience.

To understand:

- the user;

- what the user wants to do with the product;

- understand the environment in which the product is used.

As an example, consider the interface aimed at a teenager downloading music on a mobile phone with a businessperson accessing corporate data on a device. What makes a positive user experience in one situation may not be acceptable in another so understanding the context of use is vital to developing a successful product.

Users are involved throughout design and development

Users should be actively involved in all the design phases. These means that user involvement should go beyond initial focus groups and interviews (such as at the beginning of the project) or user surveys (at the end of the project).

This "active involvement" is normally achieved through significant engagement with the user, not just demonstrating a product but also actively involving the user in the design process.

The design is driven and refined by user-centred evaluation

Usability testing is used throughout the development to shape how the product works with the user. Usability testing consists of a number of formal procedures to test a product on real users doing real tasks. This regular testing of working prototypes is one example of how users can be involved throughout the design and development of the product.

Because issues relating to the usability of the product can occur at any point in the product development, it is important to regularly test the working elements of the product with users. This approach fits well with the Agile Model and Agile's focus on working code, short development times and with fewer functions delivered early in the project.

The process is iterative

To be effective this process must be iterative. Users are very poor at explaining what they require from a system. In most cases, users have to be shown an initial design and then work out what they do and don't like. In this way, the user then influences the next version of the product and so on. This iterative approach again matches closely with the approach of Agile development.

The design addresses the whole user experience

User experience is more complex that just making something easy to use. It also includes how the user sees and feels about the product. Making a product easy to use is a useful starting position but usability, and a positive user experience, is about more than simplifying the user interface.

If the design team are engaged in iterative user-centred evaluation and frequently involving the user in the development of the product then the team will learn what is important to the user and can then attempt to incorporate these features.

The design team includes multi-disciplinary skills and perspectives

If the product is to be successful, the design team needs a range of experience and to drawn on ideas and inspiration from across the team. Too often software development teams are too specialised, often predominately composed of programmers or system analysts.

An overly specialised team is often the wrong way to approach user centred design. To be successful the design team should draw views from a range of personnel: graphics designers, programmers, user experience experts, end users, project managers and many others.

2.2.2   Stages of User centred design

In common with many design methodologies, UCD is iterative and consists of four stages.

Figure 2.5: User Centred Design Process
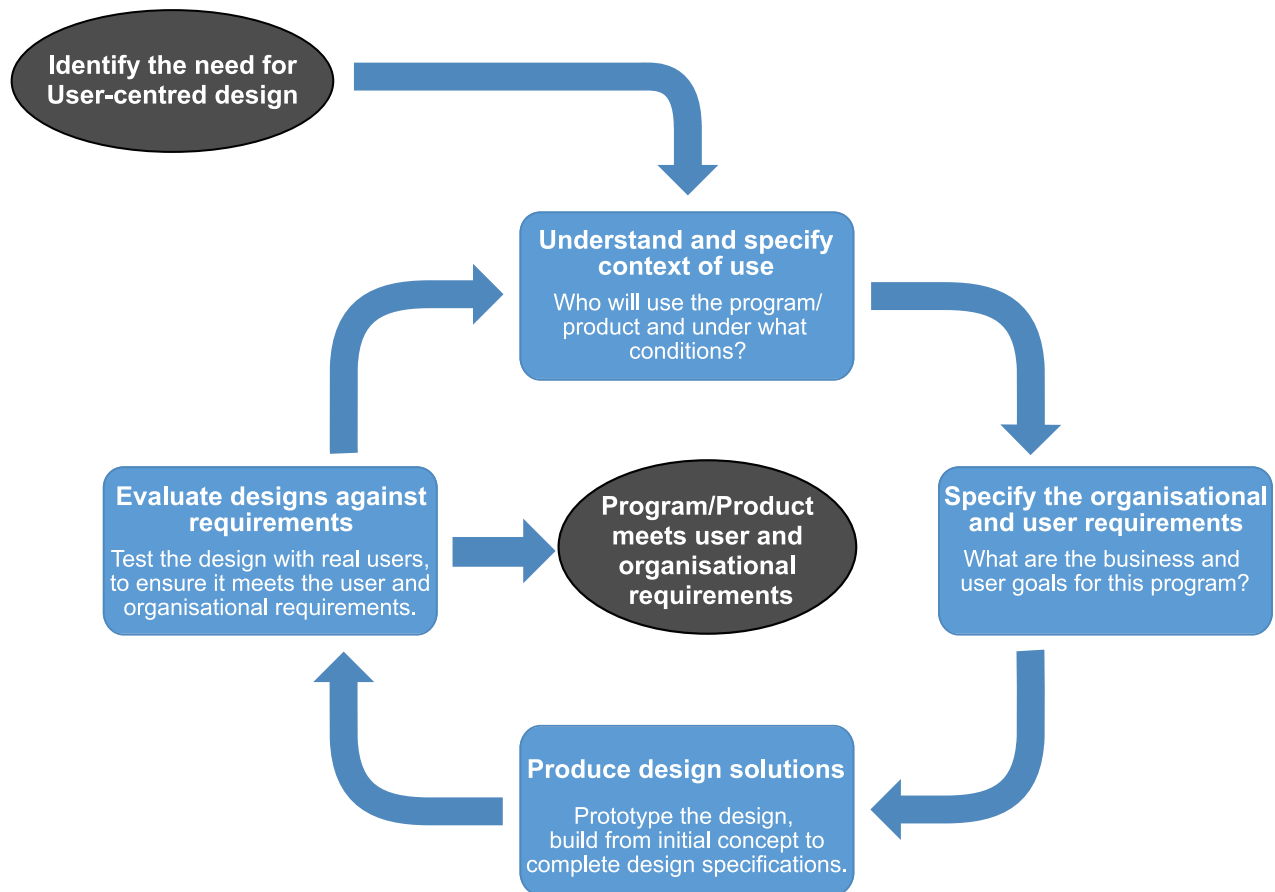
Understand and specify context of use

At this initial stage it important to understand the user, the tasks the user will carry out and the environment in which this will happen. A teenager, downloading and managing a music library on his laptop is a significantly different **context of use** from a nuclear scientist, controlling a maintenance robot in a nuclear reactor.

At this initial stage, the design team would look to describe three main areas: the **environment**, the **user** and the user **tasks** (and associated risks).

To assist with understanding users it is common to develop **User profiles** and from these profiles various **User personas**. These documents describe the user, his/her wants, needs and expectations from the product.

To develop a better understand on the tasks related to the product, the design team will develop **User stories**/**User scenarios** and **Use cases**. These documents describe the tasks and the motivation to complete them.

The process will also record the technical and environmental constraints on development (for example: what types of hardware will be used, in which organisation, technical and physical environments?)

Specify the User and organisational requirements

Figure 2.6: Requirements in the User-centred design process
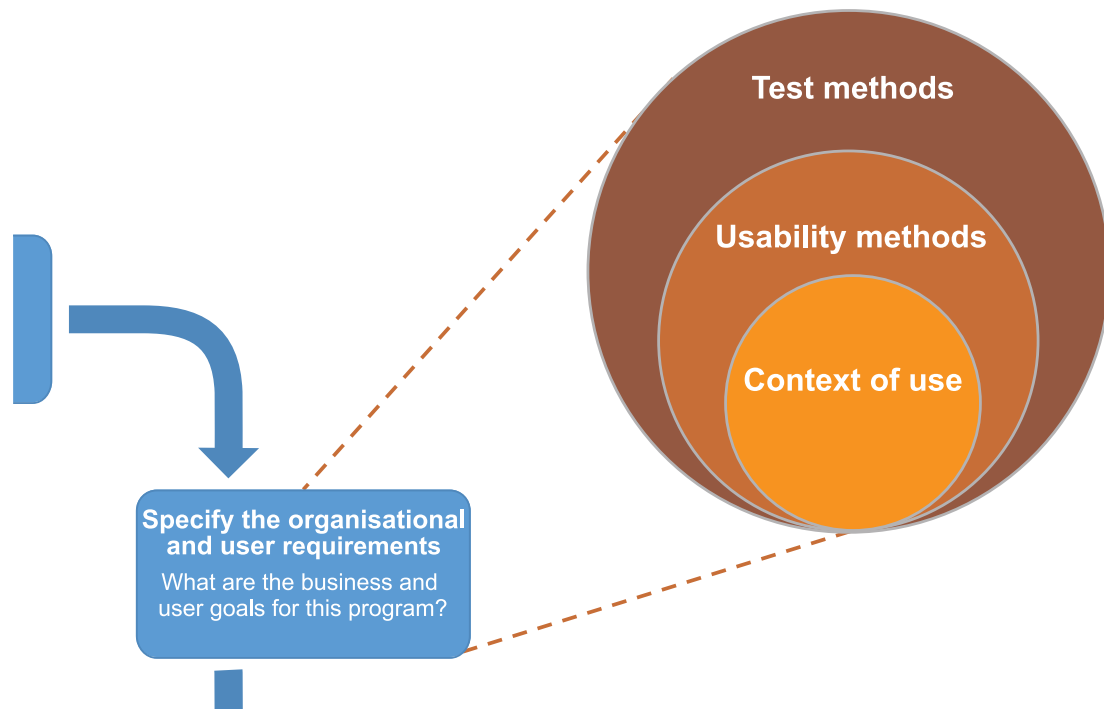
Now that the design team have a good understanding of the user, the tasks and the environment in which the product will operate they can produce a full specification for the product. This specification will include the documentation generated when developing an understanding of the "context of use" but will also include **performance and satisfaction criteria** (measures of usability for the product) and test methods (the means of determining whether the usability requirements have been met).

To ensure that the requirements are correct, other user centred design activities (such as **interviews**, **surveys**, **focus groups**, **field studies**, **task analysis**, **benchmark usability tests**, or **paper prototyping**) can be used early in the development process to obtain feedback from users to iteratively refine requirements.

Produce design solutions

The production of the solution will generally start with simple **wireframes** such as initial pencil sketches and move through formal wireframes to **low-fidelity prototypes** and then to **high-fidelity prototypes** and then the fully implemented and functional solution.

Figure 2.7: Different iterations of solution prototype

Initial Sketch   Low Fidelity Prototype   High Fidelity Prototype
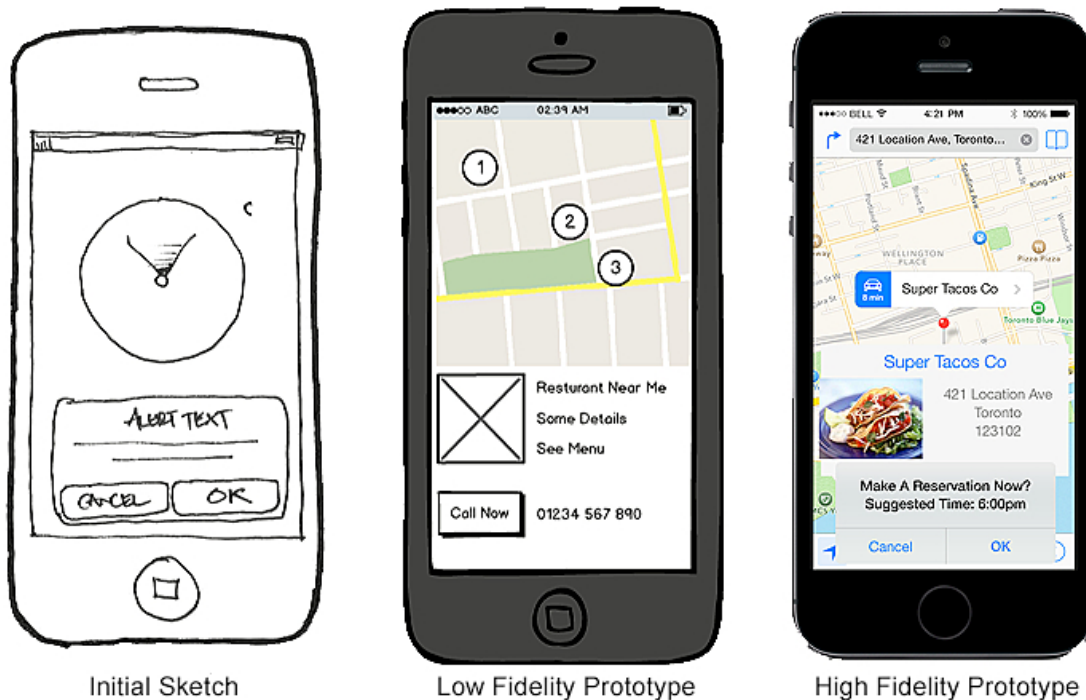
Typically, at each iteration of the cycle, some complexity/refinement is added to the solution, based on the updates to the requirements (as the result of the previous user testing).

Evaluate design against requirements

The evaluation of the solution is carried out against the requirements using a number of techniques. Usability testing to determine if the requirements have been met is the focus of the testing and this will be carried out using real users and real tasks (these are detailed in the User stories/User scenarios and Use cases).

While the solution is in development the feedback from testing can be documented in a **Prototyping session report**. This document records the results of the testing and includes recommendations to make changes to the "Context of Use" documentation, to the performance and satisfaction criteria or to the redefine the test methods.

2.2.3   Capturing user requirements

The requirements of the users of an application are captured using a number of tools. Because end users are not always available (there may be many different end users - a single **product owner** cannot directly represent all the views of these users). The team developing the application will start by understanding each user type and then develop what each user is looking for from the application.

Personas

A first step for the design team is to develop personas for each of the target user types. Personas are fictional characters created to represent the different user types that the design team have encountered or that the team believe are important to the project.

Personas can be developed iteratively and their development can involve users and the design team members. Developing the personas with a number of people increases the accuracy of the personas

and creates a level of awareness about the users that helps the design team. As people become familiar with the personas, they start talking about them as if they were actual people.

Personas do not need to be complex to be useful. The design team can begin by creating brief outlines of personas based on initial conversations with those involved in the project.

An example persona would be:

Figure 2.8: An example persona

Nikki Taylor

PROFILE:	Busy professional
GENDER:	Female
AGE:	26
LOCATION:	Central London
OCCUPATION:	Lawyer

Motivations

Likes to meet friends after work but often organises this at the last minute.

Nikki has started to use her mobile phone to organise her social life and wants to be able to book a restaurant table easily.

She also wants to know which places are the best to eat in locally without asking her colleagues or friends.

Goals

Needs to organise a meal out for herself and four friends.

Would like a choice of restaurants to match her budget.

Frustrations

Not being able to search for restaurants locally online.

Doesn't find the reviews she reads online match the places she eats in.

Hates making calls. She's too busy!

Activity: Personas

Consider a simple application that you wish to develop.

Create two personas that describe two typical users of the application. Use the same format as the sample persona above.

2.2.4 User stories

A user story is a brief statement that identifies the user and his/her need. It is a direct statement that relates to a specific persona. The personas created by the design team will have a general level of detail however the user story will provide specific details about a task that the user will be engaged in. One persona may have several user stories developed around it.

Here's an example user story for a restaurant finder app:

> "As a busy lawyer, Nikki wants to arrange a quick lunchtime meeting with colleagues. She needs quick access to restaurant information in the local area and a quick way to see which have tables available because she is making arrangements between meetings."

Notice that the user story identifies who the user is, what she needs, and why she needs it.

Here's another example of a user story that might apply to the same restaurant finder app designed to meet our first user's need:

> "Jack is a single parent and is meeting his children after school. He wants to find a local family friendly restaurant for a surprise birthday party for one of his children."

This second example maintains the same structure, but represents a very different user. A restaurant finder app is likely to have a wide array of users with different needs, big and small. User stories help to document the practical differences in need among those users.

Activity: User stories

For each user develop a small user story that defines what drives a user to make use of your application.

2.2.5   User scenarios

A user scenario expands upon already developed user stories by including details about how a system might be interpreted, experienced, and used. User scenarios provide detail about the user's goal, detail any assumed knowledge the user has and the level of experience the user has.

Just as with user stories, the design team may imagine several scenarios for each persona group.

Here's an example of a user scenario, again from our hypothetical restaurant finder app:

> "Jerry is a vegan and is arranging a night out with two friends: one is a vegetarian and the other eats meat. Jerry finds it difficult to find restaurants that meet all their needs and, in the past, nights out haven't been very good because there is often a problem with the food order. He has tried searching online but, when he does find somewhere, frequently he rings them to find out they have no tables available. He now has only one day left to make arrangements for the night out and is getting worried. He still has to book a restaurant and then call his friends to make arrangements to meet."

Notice how this scenario gives the user some backstory, gives some context about his needs, and tries to specify the gaps in his knowledge that might lead to interaction difficulties.

Activity: User scenarios

Extend your user stories from the previous section, development them to become user scenarios.

2.2.6 Use case

A use case is a list of steps a use would take to perform an action. A typical use case will start by detailing how the user got to the current position and then goes through every step until the user completes the operation or fails.

Here is an example based on Jerry from the scenario previously.

Jerry is booking a restaurant for himself and 2 friends. He:

1.	Opens the Restaurant finder app.
2.	Selects the search options from the app.
3.	Taps the Vegan and Vegetarian options.
4.	Taps the local (less than 5 miles) option.
5.	Taps the 3 stars or more option.
6.	Selects the Search button.
7.	Search results appear.
8.	Taps Stars and the results are sorted with the best rating at the top.
9.	Selects the restaurant at the top of the list.
10.	When the restaurant is displayed there are no tables available.
11.	Taps the back button to return to the search results.
12.	Taps the Stars again to sort the list.
13.	Selects the restaurant second on the list.
14.	The restaurant is displayed and it has a table for three available.
15.	Selects 8pm from the book field and taps the Book Now button.
16.	The booking is made.
17.	A confirmation text message is sent to his phone.

This detailed case highlights several points at which Jerry's experience could be improved. With this case as a guide, improvements could be made the listing of the restaurants (so that only those with availability are shown) and to the order of the list (so that it remembers how it was sorted when the user returns to it). Use cases can be informed and/or verified by the results of usability testing.

Activity: Use case

For each user story that you have created, develop a Use case which details what each user will do with the application.

2.3   User interface design

> **Learning objective**
>
> By the end of this section you will be able to:
>
> - describe, explain and apply the development of user interfaces using prototypes (from low to high fidelity including wireframes);
>
> - describe how an application style guide is used to ensure a consistent look and feel to an application;
>
> - create an initial application style guide for an application.

The solutions designed as part of user-centred design, become increasingly more sophisticated as the design team develop their solution and test it on users. Early versions of the solution may be simple pencil drawings/wireframes. Towards the end of the development process the solutions will be sophisticated computer based products that users can interact with fully.

These increasingly complex solutions are developed using a number of prototyping and design methods.

Figure 2.9: Development of prototypes
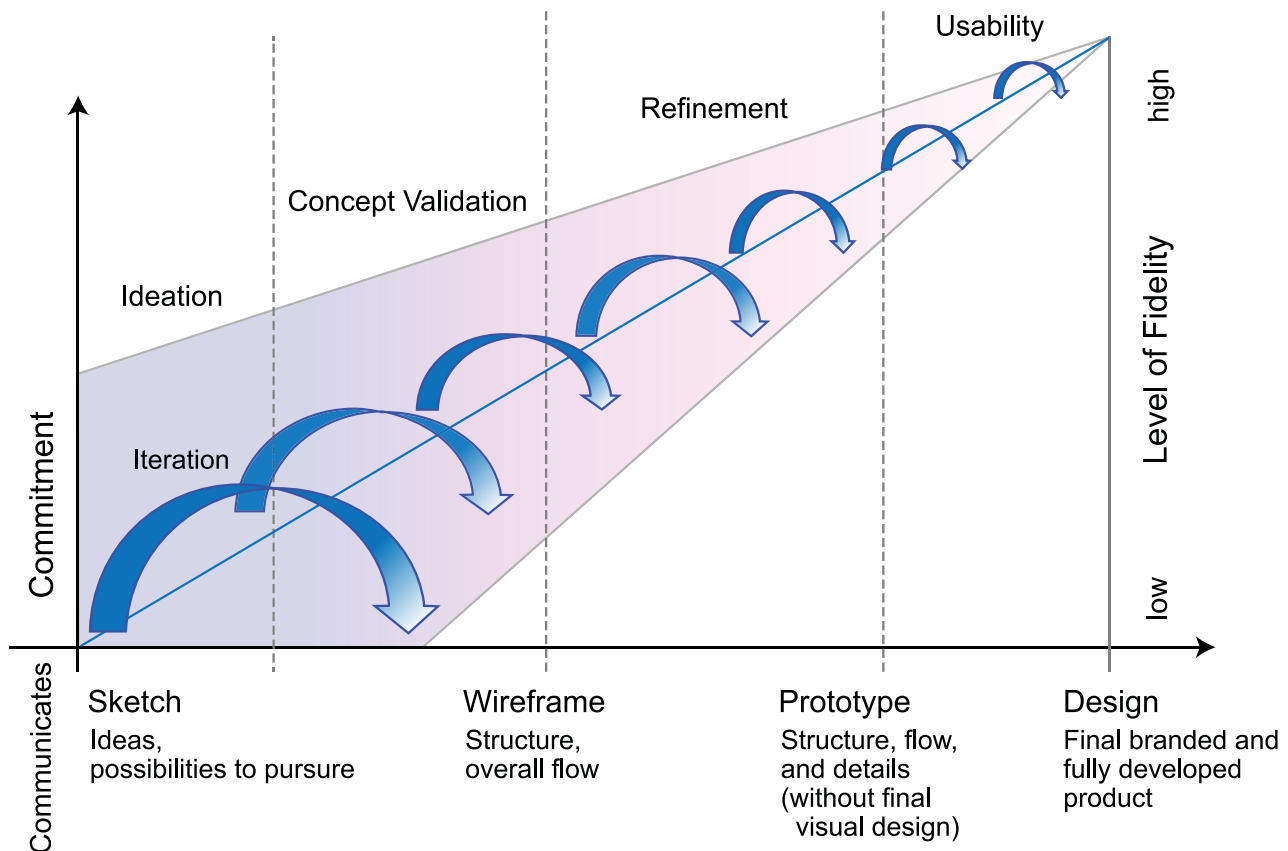
2.3.1 Wireframing

A wireframe can start out as a simple pencil sketch of a user interface. The wireframe can be developed from a low level of detail and functionality up to a high level of detail and functionality.

Figure 2.10: A simple sketch wireframe
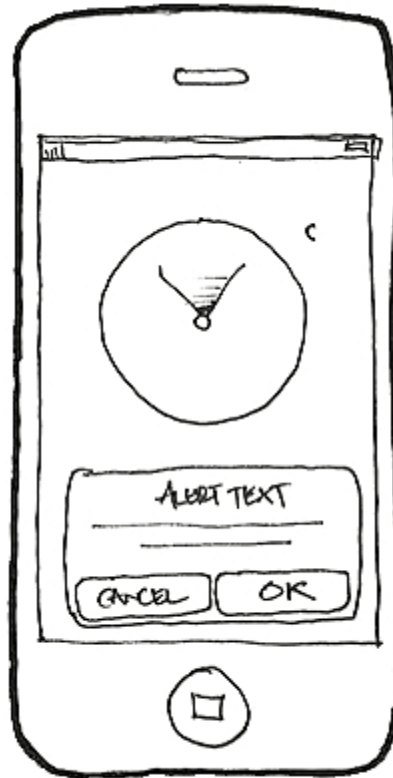
Activity: Visit Scholaria!

Visit Scholaria want to create 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.

An example of the homepage can be seen here:
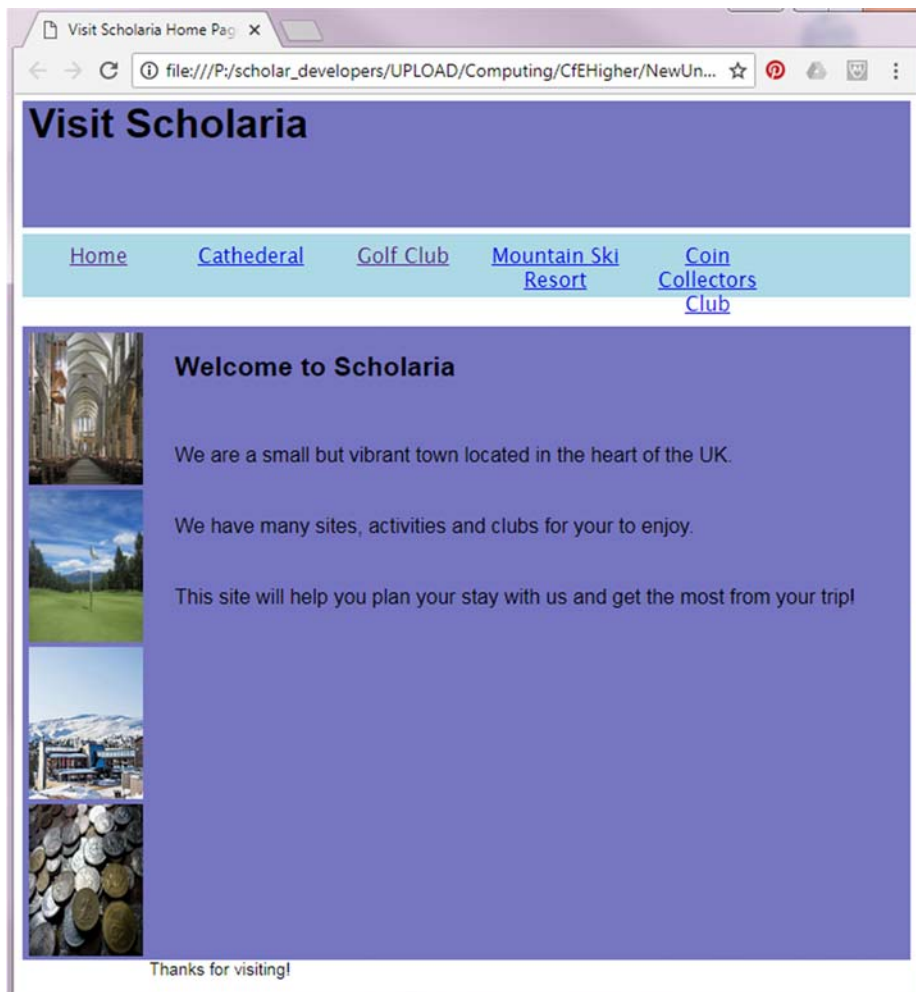
This diagram shows the navigational structure for the site. You should now create a set of wireframe designs for the Visit Upper Scholaria site we looked at previously.

The full structure for the site can be seen here:
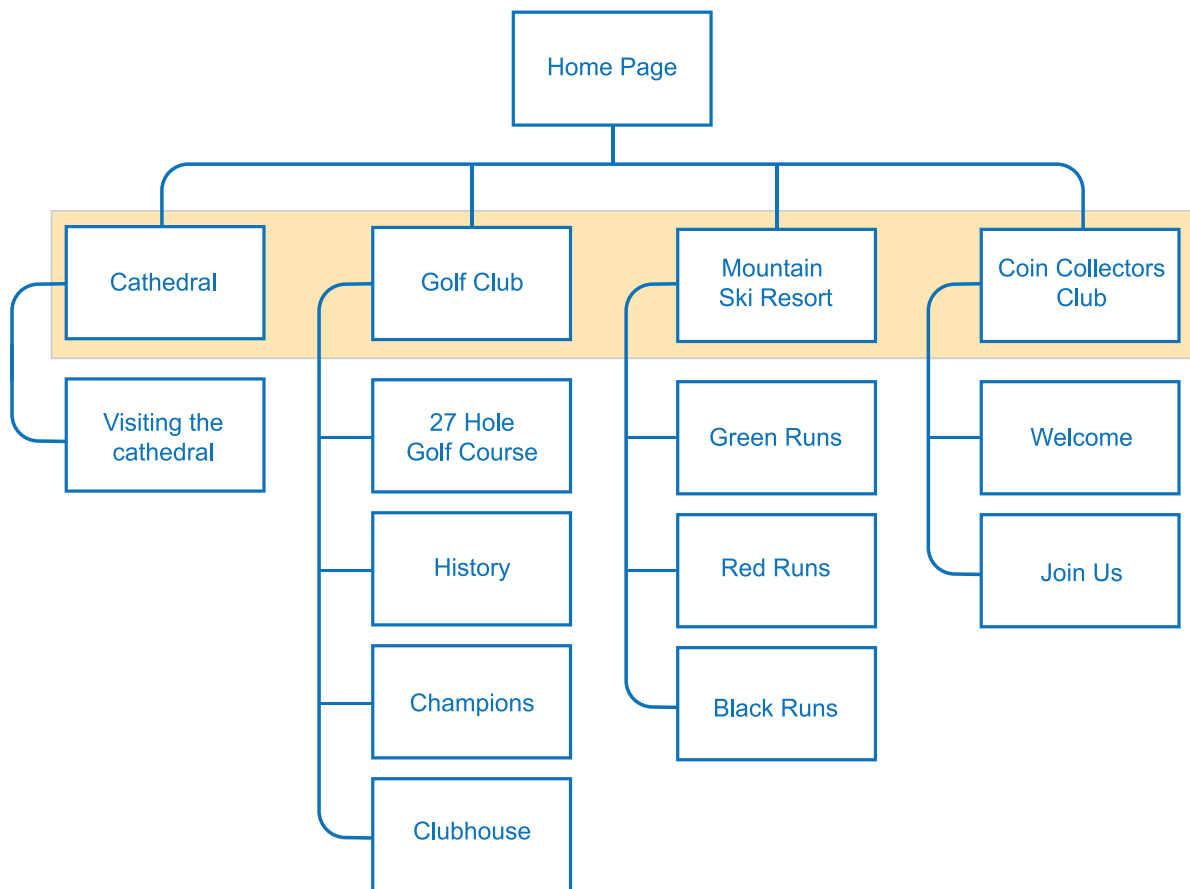
Create a wireframe for each of the 5 main pages (Homepage, Cathedral, Golf Club, Ski & Mountain Resort and Coin Collectors Club).

Each wireframe should indicate the intended layout of the page and show 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;
- any hyperlinks on the page.

2.3.2   Low fidelity prototype

Low fidelity prototypes may be paper-based (and without user interactions) or computer based with little or no interaction. They range from a series of hand drawn mock-ups to printouts to simple models of the user interface with limited interaction.

In theory, low fidelity sketches are quick to create. Low fidelity prototypes are helpful in enabling early development of alternative designs, which helps provoke innovation and improvement. An additional advantage to this approach is that when using rough sketches or very simple models, users may feel more comfortable suggesting changes.

Figure 2.11: A low-fidelity prototype

2.3.3 High fidelity prototype

High fidelity prototypes are computer-based, and usually allow realistic (mouse-keyboard) user interactions. High-fidelity prototypes take you as close as possible to a true representation of the user interface.

High fidelity prototypes are assumed to be much more effective in collecting true human performance data (e.g., time to complete a task), and in demonstrating actual products to users, clients, management, and others.

Figure 2.12: A high fidelity prototype

2.4   Learning points

Summary

You should now be able to:

- describe and develop the structure of a multi-level website with a home page and two additional levels;

- create a website taking into account end-user requirements and device type, an effective user-interface design (visual layout and readability) using wireframing;

- create a prototype (low fidelity) web page from a wireframe design;

- design, describe and apply the key aspects of user-centred design including using analysis tools to capture user requirements;

- develop user profiles, user personas, user stories, user scenarios and use cases;

- describe, explain and apply the development of user interfaces using prototypes (from low to high fidelity including wireframes);

- create a wireframe design that can:

  - show a horizontal navigational bar;
  - plan the horizontal and vertical positioning of the media;
  - indicate form inputs;
  - indicate file formats of the media in a web page (text, graphics, video, and audio).

2.5   End of topic test

End of topic test: Design   Go online

Q4:   Designers use this technique to generate an outline design of the user interface of an application/website:

a) persona
b) wireframe
c) pseudocode
d) use case

. . . .

Q5:   A paper prototype is a:

a) version of the application drawn on paper with which users can interact.
b) sketch of the user interface which users can comment on.
c) pseudocode design.
d) template of buttons, menus, and text boxes that users can discuss.

Topic 3

Implementation: HTML

Contents

Prerequisites

From your studies at National 5 you should already know how to:

- create a basic HTML page structure using the Head, Title and Body tags;

- use the following HTML tags: heading (h1-h6); paragraph <p>; DIV, link; anchor <a>; IMG; audio, video;

- create lists that are ordered and unordered and add list items to them;

- create Internal Hyperlinks between pages in a website and External Hyperlinks to pages that are not a part of your website;

- use relative and absolute addressing in a website.

Learning objective

By the end of this topic you should be able to:

- describe and exemplify the structure of a web page;

- use semantic elements (HTML5) to structure a basic web page;

- create an internal stylesheet to control the layout of a page;

- create a basic form structure to collect information;

- apply validation to form elements that check:

    ◦ the user has entered data (presence check);

    ◦ data is within a set range (range check);

    ◦ the data entered is within a set maximum/minimum length.

3.1 Revision

Quiz: Revision Go online

Q1: The following CSS code sets style rules for:

.paragraph { color: red; margin-top: 20px; border: 1px #000; }

a) a HTML tag.
b) a unique ID.
c) a class.
d) a paragraph.

....

Q2: The head of an HTML document cannot contain?

a) Links to external stylesheets.
b) The title of the HTML document.
c) A DIV grouping element.
d) JavaScript code.

....

Q3: Where would you find the title tag and what is its purpose?

....

Q4: What is included in the body tag?

Activity: Revising your HTML

Before getting started we want to keep things organised. Create a new folder called Web Design and than create a new folder for this task called "Revising HTML and CSS".

```
1   <!DOCTYPE html>
2   <!-- Your name and today's date -->
3   <!-- Basic HTML and stylesheets -->
4   <html>
5   <head>
6       <title>Basic revision of HTML and Stylesheets!</title>
7       <link rel="stylesheet" type="text/css" href="stylesheet.css">
8   </head>
9   <body>
10      <div class=header>
11          <h1>Revising HTML and Stylesheet rules</h1>
12      </div>
13      <div id="HTML">
14          <img class="logo" src="images/HTML_logo.png">
15          <h2>HTML</h2>
16          <p>HTML stands for Hyper Text Markup Language and was
                created by Sir Tim Berners-Lee in 1989 while working at
                CERN in Geneva.</p>
17          <p>HTML's development is now overseen by a group called
                The World Wide Web Consortium (W3C)</p>
18      </div>
19      <div id="Stylesheets">
20          <img class="logo" src="images/CSS.png">
21          <h2>Stylesheets</h2>
22          <p>HTML is designed to define the content of a web page
                but not the format. So the World Wide Web Consortium
                (WC3) created CSS. All modern browsers support CSS
                files so they should always be able to play the web
                pages correctly. CSS is a standard way of implementing
                and displaying web pages that use style sheets.</p>
23          <p>CSS stands for Cascading Style Sheets and they define
                how the different elements of a web page are to be
                displayed.</p>
24      </div>
25      <div class=footer>
26          <p>For more information have a look at the W3C's website
                <a href="https://www.w3.org/">Here</a></p>
27      </div>
28  </body>
29  </html>
```

Using a text editor, create a file using the text from the file above and save this file to your "Revising HTML and CSS" folder as **revising.html**.

Test it in your web browser and you should see the following:
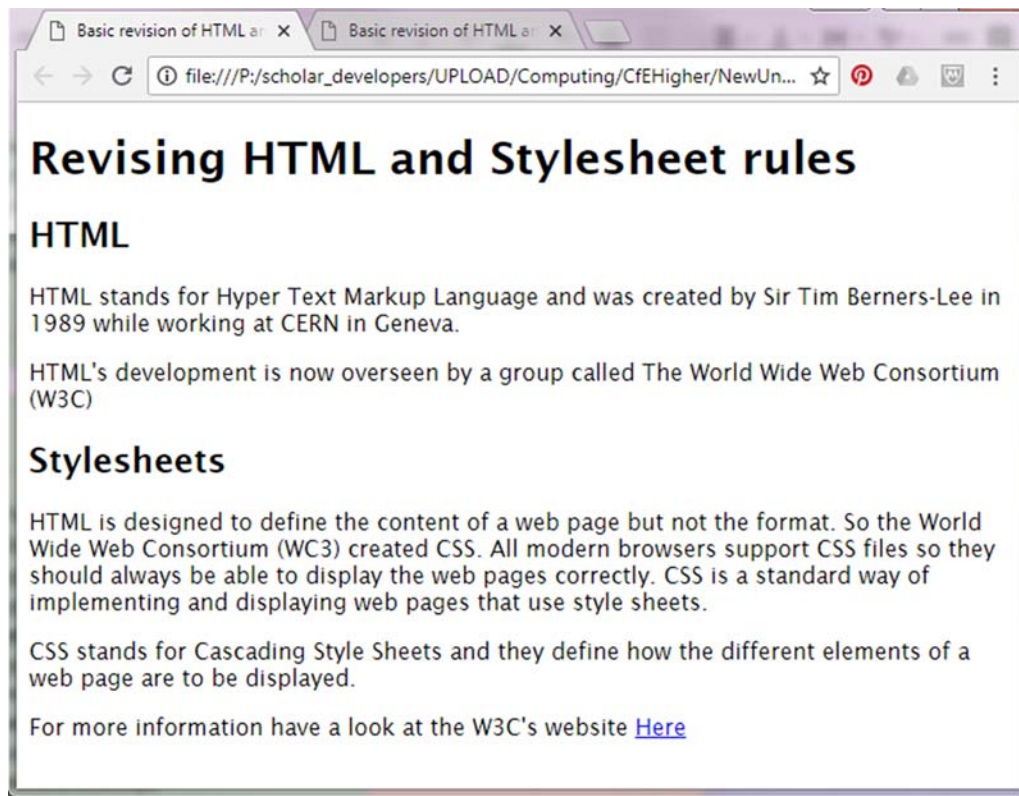
Notice that the images have not been displayed yet, you can download and save them into your website's folder:

Your page should now look like this:
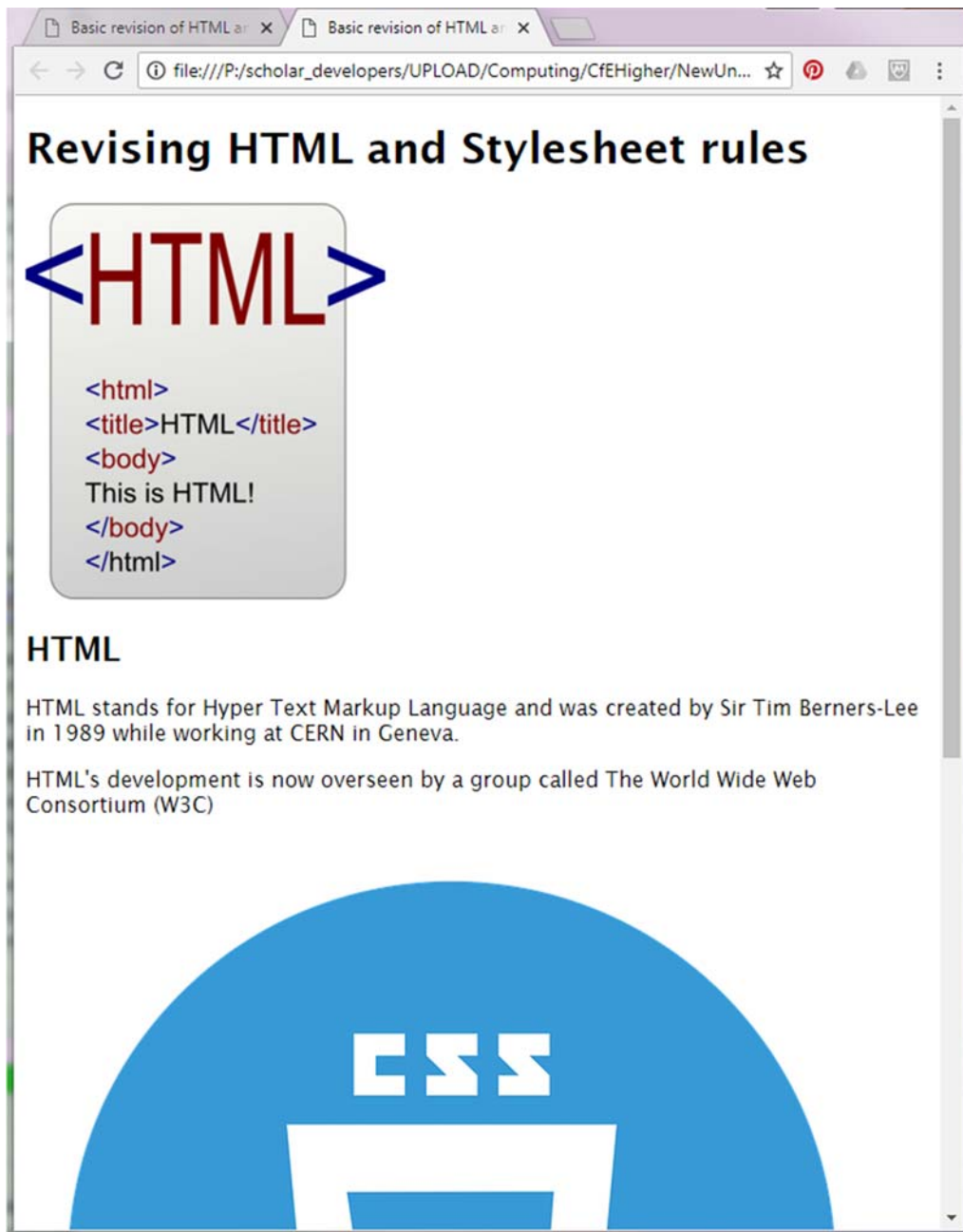
Activity: Internal Stylesheet

In the head section of your index.html page add the following stylesheet rules:

```
1  <style>
2  /*This area will be used to control elements (tags) in the
       webpage*/
3  body{
4      background-color: lightseagreen;
5      color: purple;
6  }
7  img {
8      height:150px;
9  }
10 /*This area will be used to control any objects that have an id
       property in the webpage*/
11 #HTML{
12     background-color: lightblue;
13     height:auto;
14 }
15 #Stylesheets{
16     background-color: lightblue;
17     height:auto;
18 }
19 /*This area will be used to control any objects that have an class
       property in the webpage*/
20 .header{
21     background-color: aquamarine;
22     text-align: center;
23 }
24 .footer{
25     background-color: aquamarine;
26     text-align: center;
27 }
28 </style>
```

Your page should now look like this:
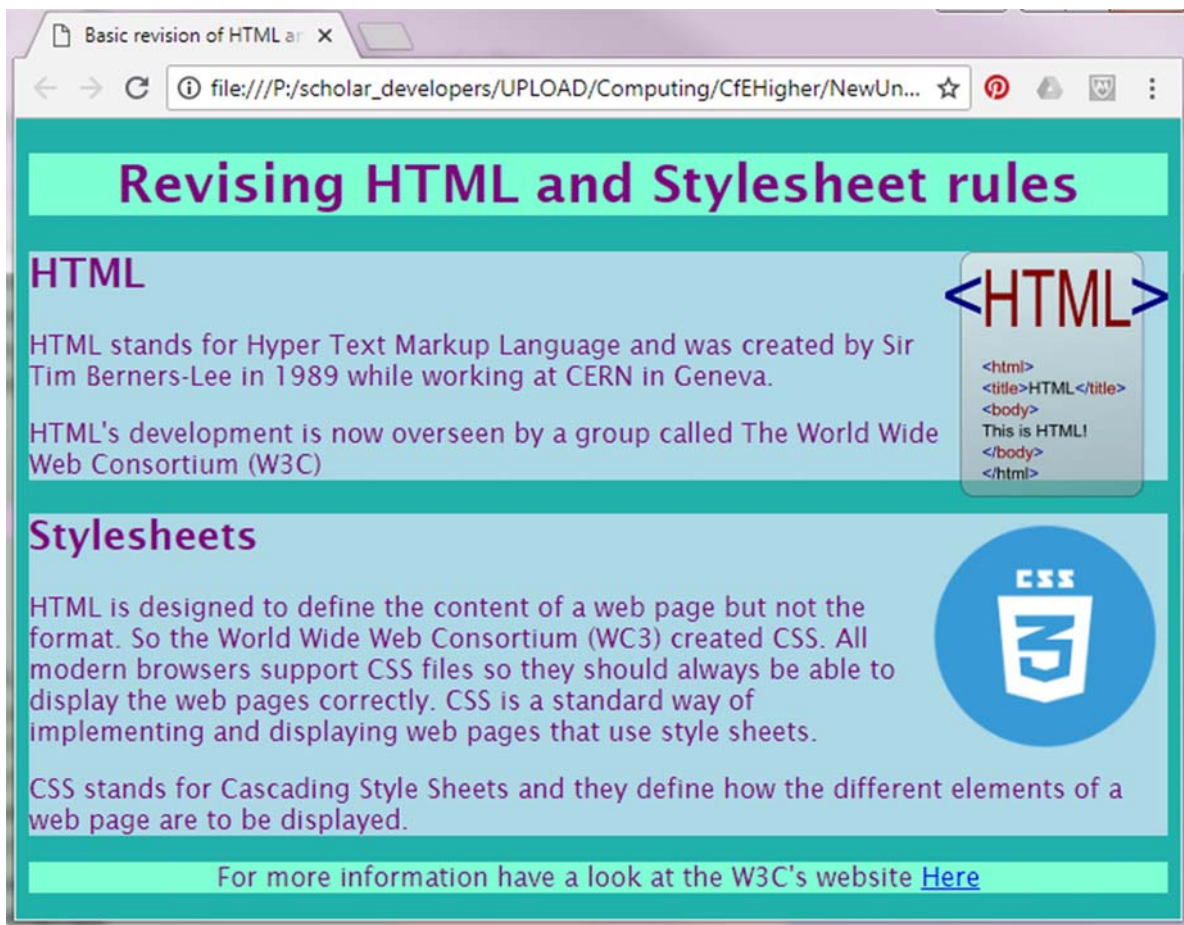
Take some time to review the rules that have been created. You may want to change some of the rules to see what happens.

3.2 Semantic elements

Learning objective

By the end of this section you should be able to:

- describe and exemplify the structure of a web page;

- use semantic elements (HTML5) to structure a basic web page;

- create an internal stylesheet to control the layout of a page

At Higher level you will be expected to be able to produce webpages and sites that meet the following requirements:

- web sites will use a horizontal navigation bar to allow users to access the different pages;

- users will experience a consistent theme throughout with additional information to help navigate a multi-level site;

- all videos and images display correctly and appropriately as designed (horizontally and vertically);

- the page has been created with appropriate tags so the web browser can display it correctly (i.e. header, title and a footer sections have been created using appropriate semantic elements).

With previous versions of HTML developers created IDs and classes in order to style and control different elements of their web page (i.e. id="nav", class=article). With the creation of HTML 5 the W3C sought to streamline this and so **semantic elements** were created.

```
<header>
```

```
<nav>
```

```
<section>
```

```
<article>
```

```
<footer>
```
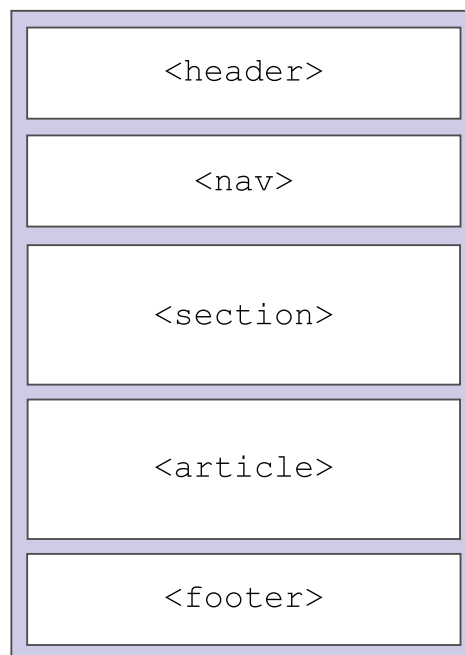
The elements that you need to know for Higher are:

Nav - <nav>

- This will be used to access the main navigational links for your website, very often this will be the same links on multiple pages.

- Not all links on a page have to be placed here.

Header - <header>

- Similar to a header in a word processing document the header tag in HTML is placed at the top of webpages. It will typically contain the name and logo for the website.

Footer - <footer>

- Similar to a footer in a word processing document the footer tag in HTML is placed at the bottom of webpages and is usually the last tag in the body of the webpage.

- It will typically contain a link back to the top of the page (anchor link), copyright information and contact information.

Section - <section>

- The section element represents a generic section of a document or application. A section, in this context, is a thematic grouping of content. Each section should be identified, typically by including a heading (h1-h6 element) as a child of the section element. (*w3.org*)

Main - <main>

- This tag is used to contain all the content that is unique to that page. Items such as navigation links, headers and footers should be in their own section.

form - <form>

- This is used to create an area to get input from the website's user, this will be looked at in more detail in the section on Form elements.

id attribute

- This should be used to uniquely identify elements in an HTML page. No other element in the page should have the same ID. This allows the CSS and Javascript to correctly access and manage the element.

You may find some others useful and you can see the full list here:
https://www.w3schools.com/htmL/html5_semantic_elements.asp

Activity: "Visit Scholaria - Coin Collectors Club"

Create a new website folder called "**Visit Scholaria**", download https://courses.scholar.hw.ac.uk/vle/asset/Downloads/H-CCMP/Course%20Downloads/211E6F85-1C03-7F52-D3C9-EF14685A6D2E/Revising_HTML_and_stylesh, extract the files and preview the homepage (index.html) in your web browser.

Here is an example of a page that contains a heading, 2 navigation sections, the main content and a footer. Use it to create a page called "**coinClub.html**" in your "**Visit Scholaria**" folder.

```
 1  <html>
 2  <head>
 3      <title>Scholaria Coin Collectors</title>
 4  </head>
 5  <body>
 6      <header>
 7          <h1>Scholaria Coin Collectors Club</h1>
 8      </header>
 9      <nav>
10          <ul>
11              <li><a href="index.html">Home</a></li>
12              <li><a href="cathederal.html">Cathederal</a></li>
13              <li><a href="golfClub.html">Golf Club</a></li>
14              <li><a href="skiResort.html">Ski Resort</a></li>
15              <li><a href="coinClub.html">Coin Collectors</a></li>
16          </ul>
17      </nav>
18      <!-- Second sub menu for the coin club -->
19      <nav>
20          <ul>
21              <li><a href="meetings.html">Meetings</a></li>
22              <li><a href="joinUs.html">Join Us</a></li>
23          </ul>
24      </nav>
25      <main>
26          <h2>Who are we</h2>
27          <p>Scholaria Coin Collectors Club are a group of
                enthusiastic coin collectors. We meet in Scholaria
                village hall on Tuesday evenings</p>
28      </main>
29      <footer>
30          <h3>Contact us</h3>
31          <p><a href="mailto:mail@sccc.co.sc">Email Us</a></p>
32          &copy; SCCC 2014
33      </footer>
34  </body>
35  </html>
```

Here is the rendered page in a browser:
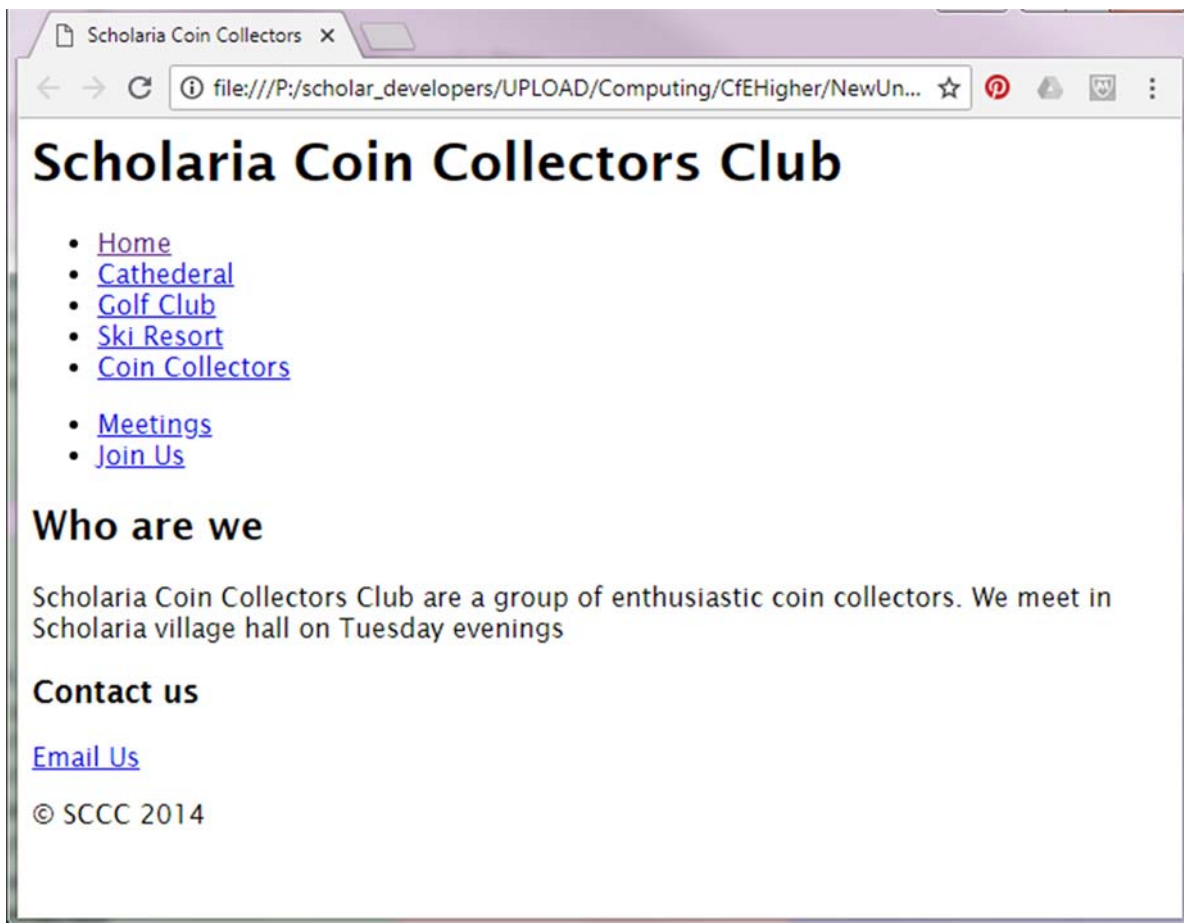
You will notice that the page still looks very plain and 'linear' when viewed in a browser. That's because HTML is only used to describe the structure and contents of the page. To change the layout and looks of the page another supporting language called CSS - Cascading Style Sheets - is needed. There is more about this language coming up in this unit.

Activity: Expanding "Visit Scholaria"

Using a text editor and the coin collectors page from the previous practical task as a template, create the 3 other web pages for the site:

- The Cathedral

- The Golf Club

- Mountain Ski Resort

Using the design we've looked at previously adapt the sub-navigation sections with links to the pages given.
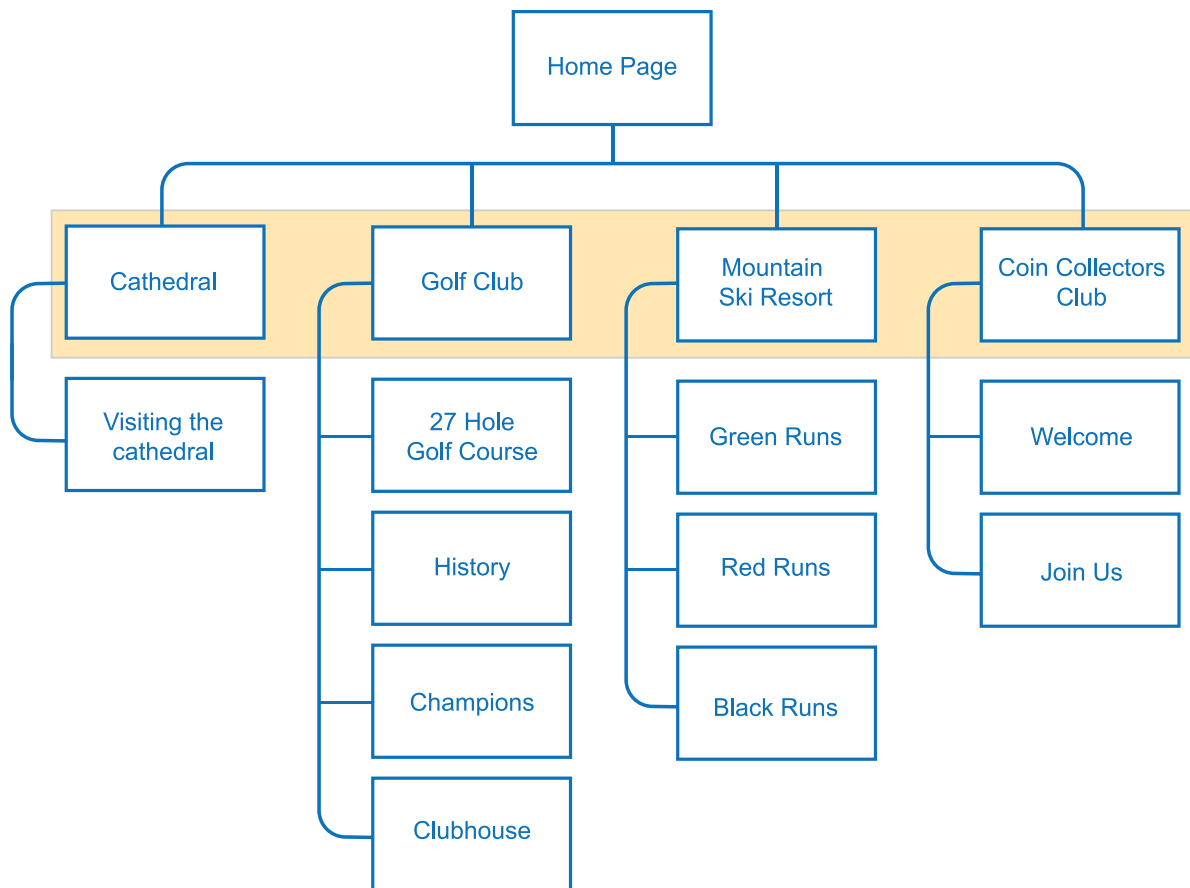
The website should encourage visitors to explore the (fictitious!) region of Scholaria's best landmarks and attractions.

You should include the following:

- A suitable heading area;
- A navigation menu for each of the 3 pages;
- A main area with a picture;
- A footer with the tourist board address and email.

Let your imagination go wild! (The place is fictitious, after all!).

Later on, we will use CSS to make the pages look more appealing.

3.3   Form elements

> **Learning objective**
>
> By the end of this section you should be able to:
>
> - create a basic form structure to collect information using:
>   - Text - a one line box for data entry
>   - Number - a number value;
>   - Textarea - a large box for data entry;
>   - Radio buttons - to make a selection from a list;
>   - Submit button - to give the user a way to send data entered to the server.

Forms in web development are typically used to submit data for processing. A web form has two parts: the **HTML** "front end" and a "back end" form processor. The front end operation occurs in the client (the browser) and the back end processing occurs on the server side.

The interactions with a web form are typically:

1. A user visits a web page that contains a form.

2. The web browser displays the HTML form.

3. The user completes the form and submits it.

4. The browser sends the submitted form data to the web server.

5. A form processor script running on the web server processes the form data.

6. A response page is sent back to the browser.

In the following activity you will create a simple HTML form and perform client side validation of the form.

Forms are created using the <form> HTML element. This element is used to open a collection of form associated elements (the things that form is made up of) and the close tag </form> is used to end the collection.

Activity: Sign up form: HTML

Create the following HTML code and save in a file called signup_form.html in the root folder of your testing web server e.g. htdocs or httpdocs.

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <meta charset="utf-8">
5       <title>Sign-up form</title>
6       <link rel="stylesheet" media="screen" href="styles.css">
7     </head>
8     <body>
9       <!--start form for signup -->
10      <form class="signup_form" name="signup_form">
11        <!-- collection of form associated elements goes here -->
12      </form>
13    </body>
14  </html>
```

This simple HTML document contains a <meta> element to force it to use Unicode encoding, a title for the document and a link to cascading style sheet that does not yet exist (we will create the HTML elements first then add the **CSS** and **JavaScript** later). The <form> element has a number of attributes:

- class (the CSS class applied);

- and a unique name to identify the form - "signup_form".

Form Input elements

Forms may use a number of input elements. These are can be text boxes, check boxes, radio buttons, password fields, buttons and many more.

Single line text box	A single line text box can be used to collect name, email and other details that normally occupy a single line.

The <input> element is used to create the text box using the text type.

```
<input type="text" name="firstname" />
```

As is the convention, because the **input** element doesn't have a closing tag, we use /> to close it.

Common attributes used with the **input** element:

- **type="text"** specifies that the browser should display a single line text input box;

- **name="firstname"** means that when the form is submitted the contents of this input will be referred to as firstname;

- **value="Please enter"** Value specifies a value to place in the text box when the form is created;

- **maxlength="60"** gives the text box a maximum number of characters that it can hold;

- **size="40"** the size of the text box as it appears in the web page;

- **placeholder="Jenny Smith"** The placeholder attribute specifies a short hint that describes the expected value of the input and displays this hint in the input area before the user enters a value.

Activity: Sign up form: Notification

Add the following code to the signup_form.html under the comment line `<!- collection of form associated elements goes here ->`

```
 1  <ul>
 2      <li>
 3          <h2>Sign Up</h2>
 4          <span class="required_notification">* Denotes Required
                Field</span>
 5      </li>
 6      <li>
 7          <label for="username">Username:</label>
 8          <input id="username" type="text" name="username" value=""
                size="40"
 9          maxlength="40" placeholder="jensmith72" />
10      </li>
11      <li>
12          <label for="realname">Real Name:</label>
13          <input id="realname" type="text" name="realname" value=""
                size="40"
14          maxlength="40" placeholder="Jenny Smith" />
15      </li>
16      <li>
17          <label for="email">Email:</label>
18          <input id="email" type="email" name="email" value=""
                size="40"
19          maxlength="60" placeholder="jenny.smith@example.com"/>
20      </li>
21      <!-- end of user ID sign up fields -->
22  </ul>
```

This code produces a form which looks like this:

Hints can be added to the form to assist users when completing it. e.g.

```
1   <li>
2    <label for="email">Email:</label>
3    <input type="text" name="email" value="" size="40" maxlength="60"/>
4    <span class="form_hint">Use the format "name@domain.com"</span>
5   </li>
```

Activity: Sign up form: Hints

Add suitable hints for the three single line text inputs in the form using the code `<span class="form_hint">Use the format "name@domain.com"</span>` above.

Change the contents of the hint to be appropriate to each input.

- A username should contain only alphabetic characters and numbers.

- A real name should be at least a firstname and lastname.

- An email address must be in a valid format.

To complete the form, we will add a small text area for multi-line text input, a check box to agree to some terms and conditions and a submit button.

Text area	A multi line text area can be used to collect an extended about of text that would occupy multiple lines.

The <textarea> element is used to create the text area.

`<textarea name="message" cols="40" rows="6">`

Common attributes used with the **textarea** element:

- **name="message"** means that when the form is submitted the contents of this textarea will be referred to as message;

- **cols="40"** Sets how wide the textarea will be in terms of number of characters;

- **rows="6"** Sets the number of visible lines of text in the text area.

Button	A clickable button used to perform an action of some sort.

The <button> element is used to create a button. Unlike the button which can be created with the <input> element, the <button> element can hold content, like text or images.

`<button class="submit" type="submit">Sign Up</button>`

Common attributes used with the **button** element:

- **type="<value>"** details the type of button. A value of button creates a "button" without a form action which can be used with JavaScript and an OnClick function. A value of "reset" creates a reset button that clears the values entered in the form. A value of "submit" creates a button that sends the forms contents for processing.

Activity: Sign up form: Additional fields

Amend the signup_form.html file to include the following lines of code after the HTML comment `<!- end of user ID sign up fields ->` and before the closing tag.

```
1  <li>
2   <label for="message">Message:</label>
3   <textarea id="message" name="message" cols="40" rows="6"
       ></textarea>
4   <span class="form_hint">A brief message, why you want to sign
       up.</span>
5  </li>
6  <li>
7   <label for="terms_and_conditions">Agree to terms and
       conditions</label>
8   <input id=" terms_and_conditions" type="checkbox"
       name="terms_and_conditions">
9  </li>
10 <li>
11  <button class="submit" type="submit">Submit Form</button>
12 </li>
13 <!-- end of additional fields -->
```

Preview the form in your web browser. The form will now appear as:

Activity: Sign up form: Apply CSS rules to the form

Create a stylesheet file called *styles.css* and save this in the same location as your *signup_form.html* file. We will be looking at CSS in more detail in the next topic, so for now enter the following rules into the file.

```css
 1  @charset "UTF-8";
 2  /* CSS Document */
 3  /* set default fonts and styles for type */
 4  body {
 5    font: 14px/21px "Lucida Sans", "Lucida Grande", "Lucida Sans
          Unicode",
 6   sans-serif;
 7  }
 8
 9  .signup_form h2, .signup_form label {
10    font-family:Georgia, Times, "Times New Roman", serif;
11  }
12  .required_notification {
13    font-size: 11px;
14  }
15
16  .form_hint {
17    font-size: 11px;   vertical-align:top;
18  }
19
20  /* remove the focus style which looks odd */
21  *:focus {
22    outline: none;
23  }
24
25  /*make the form fields more attractive by redefining the list
        style*/
26  .signup_form ul {
27      width:950px;
28      list-style-type:none;
29      list-style-position:outside;
30      margin:0px;
31      padding:0px;
32  }
33  .signup_form li{
34      padding:12px;
35      border-bottom:1px solid #eee;
36      position:relative;
37  }
38
39  /*add some visual style to the top and bottom of the form*/
40  .signup_form li:first-child, .signup_form li:last-child {
41      border-bottom:1px solid #777;
42  }
43
44  /*add a better header style and put the "required" field on the
        right*/
45  .signup_form h2 {
46      margin:0;
```

```
47        display: inline;
48  }
49  .required_notification {
50        color:#d45252;
51        margin:5px 0 0 0;
52        display:inline;
53        float:right;
54  }
55
56  /*space out the form input elements to make them more attractive*/
57  .signup_form label {
58        width:200px;
59        margin-top: 3px;
60        display:inline-block;
61        float:left;
62        padding:3px;
63  }
64  . signup _form input {
65        height:20px;
66        width:220px;
67        padding:5px 8px;
68  }
69
70  .signup _form textarea {
71      padding:8px; width:300px;
72  }
73
74  .signup _form button {
75      margin-left:156px;
76  }
77
78  /* add some enhanced visual styles */
79  .signup_form input, . signup_form textarea {
80        border:1px solid #aaa;
81        box-shadow: 0px 0px 3px #ccc, 0 10px 15px #eee inset;
82        border-radius:2px;
83  }
84  . signup_form input:focus, . signup_form textarea:focus {
85        background: #fff;
86        border:1px solid #555;
87        box-shadow: 0 0 3px #aaa;
88  }
89  /* Button Style */
90  button.submit {
91        background-color: #68b12f;
92        background: -webkit-gradient(linear, left top, left bottom,
93        from(#68b12f), to(#50911e));
94        background: -webkit-linear-gradient(top, #68b12f, #50911e);
95        background: -moz-linear-gradient(top, #68b12f, #50911e);
96        background: -ms-linear-gradient(top, #68b12f, #50911e);
97        background: -o-linear-gradient(top, #68b12f, #50911e);
98        background: linear-gradient(top, #68b12f, #50911e);
99        border: 1px solid #509111;
100       border-bottom: 1px solid #5b992b;
101       border-radius: 3px;
```

```
102        -webkit-border-radius: 3px;
103        -moz-border-radius: 3px;
104        -ms-border-radius: 3px;
105        -o-border-radius: 3px;
106        box-shadow: inset 0 1px 0 0 #9fd574;
107        -webkit-box-shadow: 0 1px 0 0 #9fd574 inset ;
108        -moz-box-shadow: 0 1px 0 0 #9fd574 inset;
109        -ms-box-shadow: 0 1px 0 0 #9fd574 inset;
110        -o-box-shadow: 0 1px 0 0 #9fd574 inset;
111        color: white;
112        font-weight: bold;
113        padding: 6px 20px;
114        text-align: center;
115        text-shadow: 0 -1px 0 #396715;
116    }
117    button.submit:hover {
118        opacity:.85;
119        cursor: pointer;
120    }
121    button.submit:active {
122        border: 1px solid #20911e;
123        box-shadow: 0 0 10px 5px #356b0b inset;
124        -webkit-box-shadow:0 0 10px 5px #356b0b inset ;
125        -moz-box-shadow: 0 0 10px 5px #356b0b inset;
126        -ms-box-shadow: 0 0 10px 5px #356b0b inset;
127        -o-box-shadow: 0 0 10px 5px #356b0b inset;
128    }
129
130    /* interactive CSS3 */
131    .signup_form input:focus, .signup_form textarea:focus {
132    /* add this to the already existing style */
133        padding-right:30px;
134    }
135
136    .signup_form input, .signup_form textarea {
137    /* add this to the already existing style */
138        -moz-transition: padding .25s;
139        -webkit-transition: padding .25s;
140        -o-transition: padding .25s;
141        transition: padding .25s;
142    }
```

  This CSS code significantly enhances the visual appearance and behaviour of the form.

Finally, we can use the HTML5 required attribute to tell the browser that an input / textarea element must have a value before the form can be submitted e.g.

```
1    <input type="text" name="username" value="" size="40" maxlength="40"
2    placeholder="jensmith72" required />
```

Activity: Sign up form: Additional required attributes

Add the **required** attribute to the inputs for username, realname and email. Leave the message textarea as an optional item.

Some additional styling can be added to these fields. These rules will add a red asterisk to the background of each required field.

Activity: Sign up form: Additional CSS

Add the following rules to your *styles.css* file.

```
1   /*add red asterisk for the required elements*/
2   .signup_form input, .signup_form textarea {
3       padding-right:30px;
4   }
5
6   input:required, textarea:required {
7       background: #fff url(images/red-asterisk.png) no-repeat 98%
8           top;
    }
```

Download this image of a red asterisk and place it in a subfolder called "images" inside your web folder. Name the downloaded file "red-asterisk.png" to match the CSS rule.

    red-asterisk.png

Please go online to download this file.

For further reading you can find a complete list of HTML tags here:
https://www.w3schools.com/tags/

3.4 Learning points

Summary

You should now be able to:

- describe and exemplify the structure of a web page;

- use semantic elements (HTML5) to structure a basic web page;

- create an internal stylesheet to control the layout of a page;

- create a basic form structure to collect information;

- apply validation to form elements that check:

  ◦ the user has entered data (presence check);
  ◦ data is within a set range (range check);
  ◦ the data entered is within a set maximum/minimum length.

3.5 End of topic test

End of topic test: HTML   Go online

Q5: HTML comments start with //

a) True
b) False

....

Q6: Which HTML element is used to create a footer in an HTML document?

a) <section>
b) <footer>
c) <bottom>
d) <foot>

....

Q7: Which attribute is used to specify that an input field is mandatory?

a) placeholder
b) validate
c) formvalidate
d) required

....

Q8:   Which HTML element is used for creating the site navigation structure?

a)   <nav>
b)   <navigate>
c)   <menu>
d)   <div id-navigation>

. . . .

Q9:   Which HTML element is used to create a header in an HTML Document?

a)   <top>
b)   <head>
c)   <header>
d)   <section>

Topic 4

Implementation: CSS

Contents

Prerequisites

From your studies at National 5 you should already know how to:

- create a stylesheet:
  - internal and;
  - external;
- use selectors, classes and IDs in your stylesheet to control sections of your webpage;
- set the following text properties:
  - Font
    - Font-family
    - Font-size
  - Color
  - Alignment
- set the background colour of different sections in a webpage using:
  - Selectors (body, H1);
  - Classes (.headingClass);
  - IDs (#currentPage);
- create and link to a CSS document from a webpage;
- read and explain all of the above in terms of a Stylesheet / CSS document.

Learning objective

By the end of this topic you should be able to know how to efficiently use:

- inline styles, internal stylesheets and external Cascading Style Sheets (CSS);
- understand the order of priority given to each method of styling a web page;
- grouping selectors to increase efficiency of CSS and web pages;
- descendant selectors to increase efficiency of CSS and web pages;
- using the following properties and rules to control appearance and positioning of elements in a web page:
  - display (block, inline, none);
  - float (left, right);
  - clear (both);
  - margins / padding;
  - sizes (height, width);
- grouping and descendant selectors to create horizontal navigation bars;

Learning objective continued

- use the following properties and rules when creating a horizontal navigation bar:

    ◦ list-style-type:none;

    ◦ hover;

- read and explain code that makes use of the above CSS.

4.1   Revision

We're now going to go back to our "Revising HTML and CSS" website that we were introduced to in the Implementation: HTML topic and we will move our styles into a separate CSS document.

Activity: Revising your HTML and CSS

Using a text editor create a new file in your website folder called "**stylesheet.css**".

Enter the CSS below and save it as "**stylesheet.css**".

```
 1  body{
 2      background-color: lightseagreen;
 3      color: purple;
 4      }
 5
 6  img {
 7      height:150px;
 8  }
 9
10  #HTML, #Stylesheets{
11      background-color: lightblue;
12      height:auto;
13  }
14
15  .header, .footer{
16      background-color: aquamarine;
17      text-align: center;
18  }
19
20  .logo {
21      float: right;
22  }
```

In your index file enter the following line in between the <**head**> tags:

<link rel="stylesheet" type="text/css" href="stylesheet.css">

Save both of these files and test that your website is displaying properly again with all style rules working.

4.2   CSS Priorities

HTML is designed to define the content of a web page but not the format. So the World Wide Web Consortium (WC3) created CSS. All modern browsers support CSS files so they should always be able to display the web pages correctly. CSS is a standard way of implementing and displaying web pages that use style sheets.

When developing for the web it is important to remember the priorities that are applied for styles in

---

pages.

This graphic shows the priorities that are applied when styling a web page.
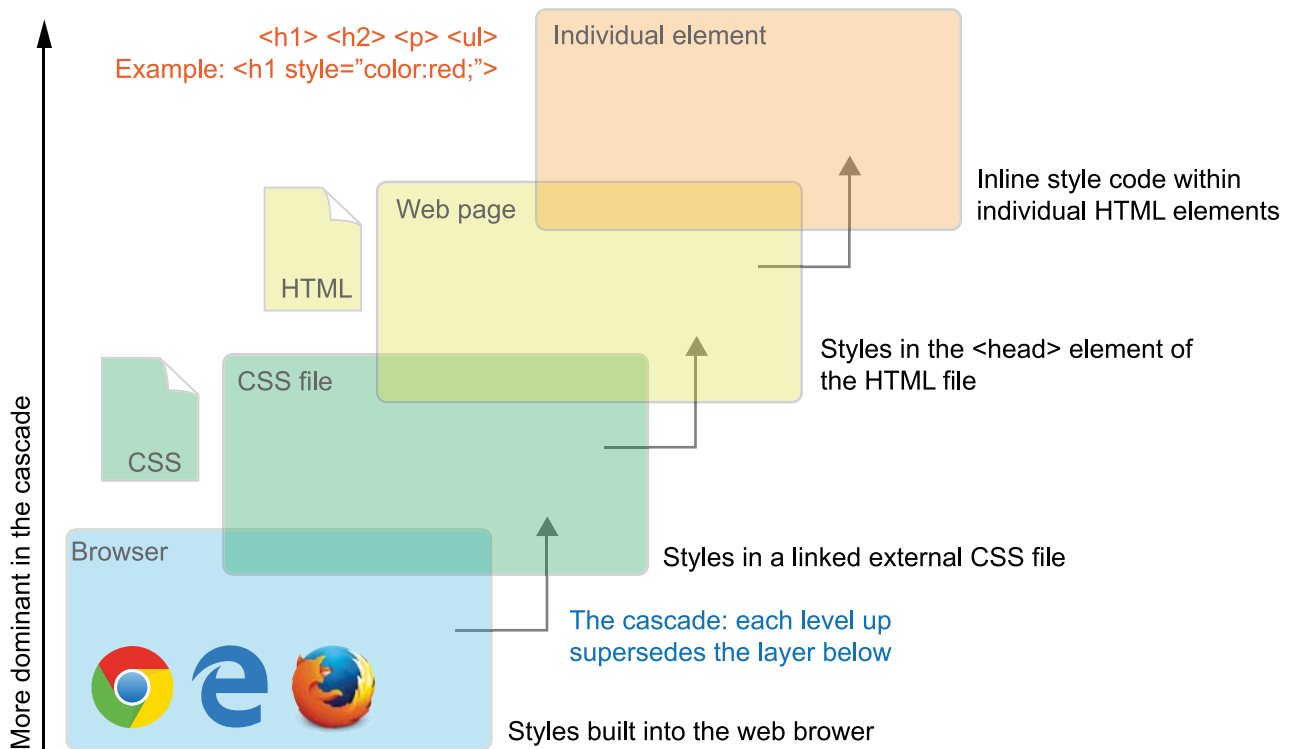
4.2.1   Web browsers

Web Browsers have a default style they will use to display web pages. If no styles are provided for the page or if styles specified by the web developer cannot be displayed on the user's computer this basic style will be used.

This can be problematic as these default styles differ from browser to browser, not good if we want to give the user a consistent theme regardless of what device of web browser they are using.

4.2.2   Cascading Style Sheet (CSS)

Cascading Style Sheets define how the different elements of a web page are to be displayed. In effect every web page created in this way has two files. One which contains what is to be displayed and another that explains how it is to be displayed. Styles are saved in a .css file.

The advantages of CSS are:

- The layout of many documents or even an entire website can be controlled from one style sheet, reducing the amount of code required for formatting.

- The developer gets a more precise control over layout than by using HTML tags.

- The developer can apply different layouts to different media types, screen or print etc.

- Several different stylesheets can be used to change the look of a site according to users' preferences.

Websites have to be able to be viewed on a variety of different devices and screen sizes and CSS can help with this. CSS can create rules for different devices that have different screen widths so that the website will display slightly differently on each device. This ensures that the user is always viewing the website correctly.

If a style sheet is created for a web page then the programmer does not need to insert hundreds of different tags for formatting. If the design is to be changed then it is only the style sheet that needs to be changed - not the entire website.

4.2.3   Internal stylesheet

In the Implementation: HTML topic we created an internal stylesheet as part of the "Revision of HTML and CSS" task.

We insert Style tags <style> in the head of our HTML document and use the same structure for the rules as in external CSS documents. The rules in an internal stylesheet will override any styles given in the external stylesheet.

Look at these sets of stylesheet rules and the resultant output.

External stylesheet rules	*Internal stylesheet rules*
| ```
1  body {
2      background-color: white;
3      color: purple;
4      }
5
6  img {
7      height:150px;
8  }
``` | ```
1  <link rel="stylesheet"
      href="style.css">
2
3  <style>
4    body {
5      background-color: lightblue;
6      font-family:sans-serif;
7    }
8
9    h1 {
10      background-color: lightcyan;
11      font-size: 32px;
12    }
13  </style>
``` |

Output:

> **Key point**
>
> In the example above the link to the stylesheet called "style.css" still exists but as the internal stylesheet is given below it is executed after the CSS styles have been loaded and overrides the CSS file.

4.2.4 In-line styles

We've previously looked at how Internal style sheets take priority over external stylesheets, In-Line styles can be used to style an individual element in the page and is placed inside the tag that is being edited.

HTML Page

```
1   <head>
2     <link rel="stylesheet" href="style.css">
3
4     <style>
5       body {
6         background-color: lightblue;
7         font-family:sans-serif;
8       }
9
10      h1 {
11        background-color: lightcyan;
12        font-size: 32px;
13      }
14    </style>
15  </head>
16  <body>
17    <h1 style="background-color:lightgreen;">Welcome to CSS</h1>
18    <p>
19      In this page we will demonstrate how priorities between External
            and internal pages effect the display of the page.
20    </p>
21  </body>
```

Final output:

> **Key point**
>
> As you can see from the above, even when the external stylesheet is linked and the internal stylesheet is present the in-line style has again overruled both of these and used the individual elements style.

4.2.5   Review

When used effectively each of the methods of styling a web page has its merit. However, for ease of maintaining a website having a separate document is preferable with items that are to be individually altered given an ID to be directly targeted from a stylesheet. If necessary we can even use multiple CSS files to style one page.

It may be easy to create an internal stylesheet to test the look and function of your webpage without affecting other pages in the site before moving it into an external stylesheet.

4.3   Increasing efficiency in CSS

Learning objective

By the end of this section you should be able to:

- efficiently use grouping and descendant selectors in a web page to control appearance and positioning using the following rules:
  - display (block, inline, none);
  - float (left, right);
  - clear (both);
  - margins / padding;
  - sizes (height, width).

4.3.1   Grouping selectors

You may have noticed in the CSS that a number of the selectors have the same rules. To save time and increase efficiency we can group these together.

In the example from the CSS below you can see that three different selectors are using the same colour rule for the text. To group these together we can use a comma (,) to separate the selectors and reuse the rules multiple times.

Original	*Grouped*
| ```
1  p {
2      color: DarkBlue;
3  }
4  ul {
5      color: DarkBlue
6  }
7  ol {
8      color: DarkBlue;
9  }
``` | ```
1  p, ul, ol {
2      color: DarkBlue;
3  }
``` |

4.3.2 Descendant selectors

There may be times where we don't want all elements such as paragraphs in a page to use the same styles. For this we can use **Descendent Selectors**.

We might want all paragraphs in a <div> to have the same format but use a different format for the next div in the page.

In the table you can see that the heading 2 from the header is different to the <section> area of the page by adding the following to the CSS document:

CSS	Page Output
| ```
1  section h2{
2      color: yellow;
3  }
``` |  |

Activity: The Guid Scottish Dog Walk: HTML

To look at descendent selectors we are going to use a new website called "**The Guid Scottish Dog Walk**".

Create a new folder for it and download https://courses.scholar.hw.ac.uk/vle/asset/Downloads/H-CCMP/Course%20Downloads/4821994B-A624-D6AF-5024-308AB84CD322/TheGuidScottishDogWalk.zip , extract the files and preview the homepage (index.html) in your web browser.

You should see this page displayed:

Activity: The Guid Scottish Dog Walk: CSS

Now create a CSS file called "style.css", save it in the same folder as the index.html file you downloaded, and enter the following into it:

```css
body {
    background-color: lightskyblue;
    font-family: sans-serif;
}
header{
    background-color: lightcoral;
    overflow: auto;
}
section {
    background-color: lightcoral
}
h1 {
    font-family: Comic Sans MS;
    font-size: 24;
    text-align: center;
}
h2 {
    font-family: Comic Sans MS;
    font-size: 14;
}
p {
    color: DarkBlue;
}
ul {
    color: DarkBlue
}
ol {
    color: DarkBlue;
}
.dates{
    padding-left: 60px;
}
```
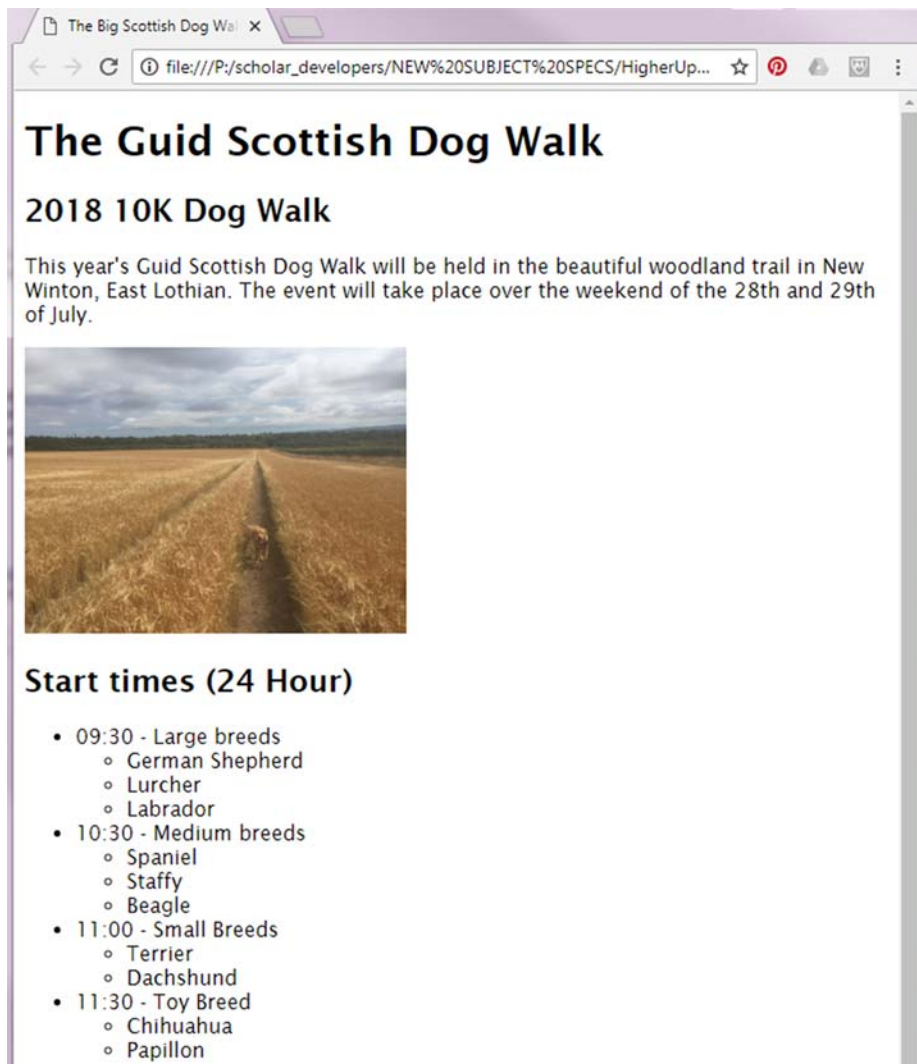
Insert the link from the HTML page to the CSS in the head section by using the line below:

<link rel="stylesheet" type="text/css" href="styles.css">

Your page should now look like this:

Make sure you read both the HTML and CSS to make sure you understand what each part is doing to display the webpage.

4.4   Appearance and positioning

Over the next four sections we will look more closely at how to control the appearance and positioning of elements on our website using:

- display properties;

- image attributes;

- margins and padding;

- defined size properties.

4.4.1   Display

There are many properties that can be applied to the display rule however the three we will focus on are:

- **Block**
  Many elements in HTML have the display block rule applied by default (doesn't need to be specified in the CSS). It displays an element on its own as a block element. It starts on a new line, and takes up the whole width of the page.

- **Inline**
  Displays an element as an inline element. Any height and width properties will have no effect. This will be used later on to help create a horizontal navigation bar.

- **None**
  The element is hidden on the page, most commonly used in pages with Javascript.

You can find out more about the display property from the W3Schools website here:
https://www.w3schools.com/cssref/pr_class_display.asp

4.4.2   Float and clear

The float property will be most useful when getting text to wrap around another element or an image on your webpage. We can tell an element to float to the left, right, none or inherit from the element it's contained in.

> **Key point**
>
> For this to work properly we need to place the image tag above the text we want to wrap around it.

The clear property helps us control what elements can float around the current element.

Example : Float

Now apply the following property to the image selector (img) from "The Guid Scottish Dog Walk" by adding the following to our CSS file:

```
1  img{
2      float:none;
3  }
```

Once we've entered this rule we can try out the other options for float (left, right and inherit) and we would see the following outcomes:

`float:none` (default)	`float:left`

`float:right`	`float:inherit`

4.4.3 Margins and padding: The box model

We're now going to have a look at how we can use margins and padding in our webpages. The diagram below indicates where each are located for an element.
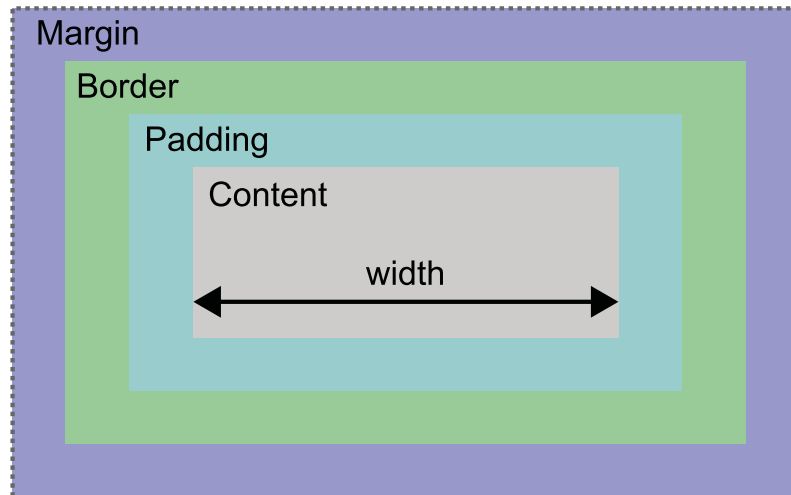
Margins and padding can be specified in 2 ways, using individual CSS rules for top, left, right and bottom or a shorthand rule that can specify all 4 in one line.

For example, if we want to have all the <div> elements in a page have a margin of 5 pixels all around the element and a padding of 10 pixels between the margin and the content we would can create the following CSS rules.

Examples

1. Individual rules

```
 1  div{
 2     /* Margins */
 3     margin-top: 5px;
 4     margin-right: 5px;
 5     margin-bottom: 5px;
 6     margin-left: 5px;
 7
 8     /* Padding */
 9     Padding-top: 10px;
10     Padding-right: 10px;
11     Padding-bottom: 10px;
12     Padding-left: 10px;
13  }
```

....

2. Shorthand rules 1

```
1  div{
2      margin: 5px 5px 5px 5px;
3      padding: 5px 5px 5px 5px;
4  }
```

....

3. Shorthand rules 2

```
1  div{
2    margin: 5px;
3    padding: 10px;
4  }
```

All three of the above CSS rules accomplish the same thing but the most efficient is example 3 as it accomplishes the objective in the least number of lines.

If, however, we have different dimensions then learning to use example 2 will be the most efficient and the example below we'll look at how the version of example 2 is structured:

Selector   Rule   Top   Right Bottom Left   Selector   Rule   Top   Right Bottom Left

```
p{                                              p{
    margin: 5px 5px 5px 5px;                        padding: 5px 5px 5px 5px;
}                                               }
```

Here is another example of how this can be implemented using different dimensions:

Selector Rule   Top   Right   Bottom   Left   Selector   Rule   Top   Right   Bottom   Left

```
p{                                                 p{
    margin: 25px 50px 75px 100px;                      padding: 25px 50px 75px 100px;
}                                                  }
```

Activity: Margins and padding

Use the following html to make a page with a <header> and 2 <div> elements.

HTML

```
 1  <!DOCTYPE html>
 2  <html>
 3    <head>
 4      <title>Margins and Padding</title>
 5      <link rel="stylesheet" href="style.css">
 6    </head>
 7
 8    <body>
 9      <header>
10        <h1>Margins and Padding</h1>
11      </header>
12
13      <main>
14        <div class=definition>
15          <h2>CSS Margins</h2>
16          <p>The CSS margin properties are used to create space
                around elements, outside of any defined borders.</p>
17
18          <p>With CSS, you have full control over the margins. There
                are properties for setting the margin for each
19                          side of an element (top, right, bottom, and
                                left).</p>
20        </div>
21        <div class=definition>
22          <h2>CSS Padding</h2>
23          <p>The CSS padding properties are used to generate space
                around an element's content, inside of any defined
                borders.</p>
24
25          <p>With CSS, you have full control over the padding. There
                are properties for setting the padding for each side of
                an
26                          element (top, right, bottom, and left).</p>
27        </div>
28      </main>
29    </body>
30  </html>
```

CSS: You will need to add the missing properties and rules.

```
 1  body {
 2     background - color:  lightblue;
 3  }
 4  header {
 5     background - color:  cadetblue;
 6     /* Margin for the header section is 2px all around */
 7     margin: 2px;
 8  }
 9  .definition{
10     background - color:  aliceblue;
11     /*Add a margin of 10 for this class*/
12
13     /*Add some padding of 5px from the top and bottom
14     and 10px from the left and right for this class*/
15
16  }
17  h2{
18     /* Add a margin of 5px for all h2 tags*/
19  }
```

Follow the instructions in the CSS page to add margins and padding to the webpage.

4.4.4   Sizes (height and width)

You may have already used height and width when controlling an image in a webpage but we can control the height and size of any individual element using CSS rules also.

There are different measurements that can be used some of them are:

- Pixels (px)

- Percentage (%)

- Centimetres (cm)

The syntax for each of these are very simple:

```
div{                        div{
    height: 500px;              width: 500px;
}                           }


div{                        div{
    height: 50%;                width: 50%;
}                           }
```

Using the Margins and Padding example webpage you've created previously set the width of the definition class to be 40% and the height to auto (this should allow you to resize the window and still

display the <div> tags correctly).

Finally use the float rule to set the two <div> tags in the same line (float:left). You should see something like the image below now:
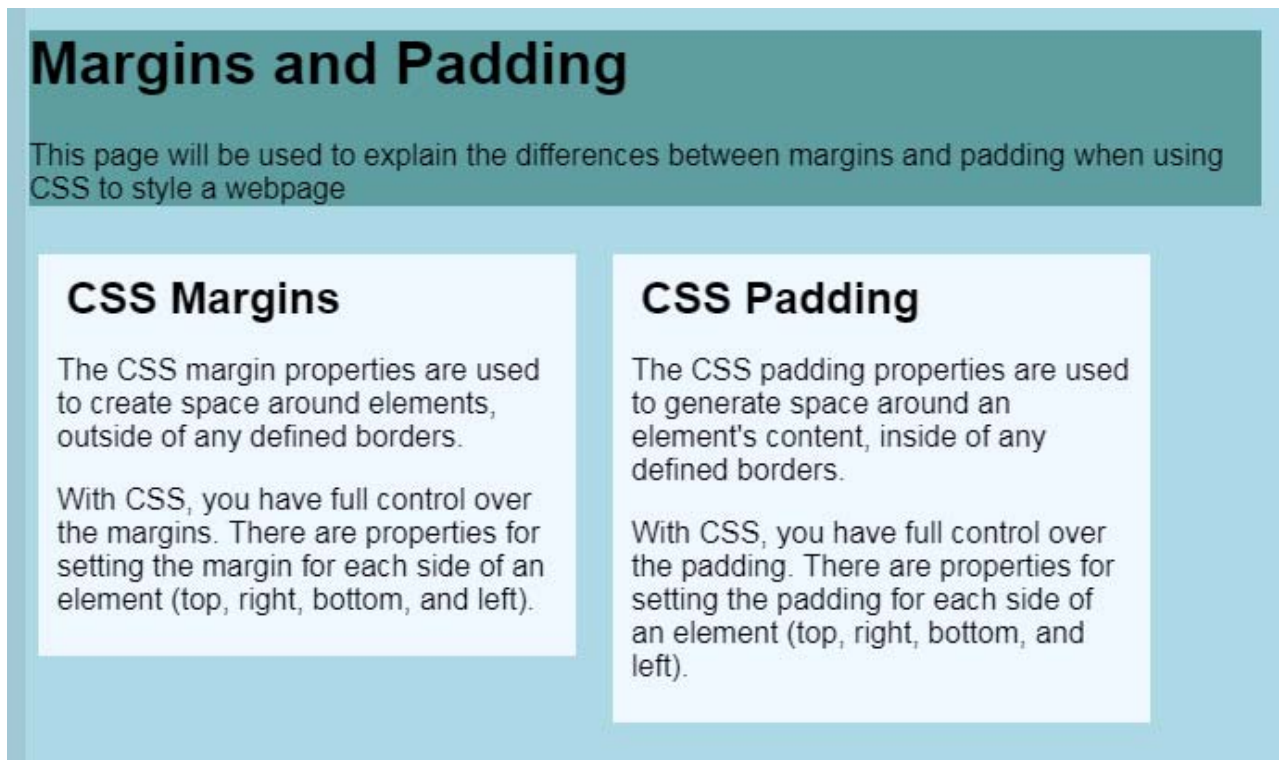
4.5 Navigation bars

Learning objective

By the end of this topic you should be able to know how to efficiently use:

- create a navigation structure using hyperlinks for your website using the <nav></nav> tags;

- remove the bullet points by altering the style of an unordered list::

  ○ list-style-type:none;

- efficiently use grouping and descendant selectors to create horizontal navigation bars using the following rules:

  ○ list-style-type:none;

  ○ hover.

In the Implementation: HTML topic we looked at the new semantic elements created for HTML 5, we will be using the nav element <nav> to help us create a horizontal navigation bar.

Activity: Scholaria Circus

Creating the pages

The first step is to create our new web page and link the CSS to it. Create a new folder, name it appropriately i.e. ScholariaCircus. Now create an **index.html** file and a **style.css**, saving both of the files into your new folder. Now you can add the content of each page using the code here:

HTML - index.html

```
 1  <!DOCTYPE html>
 2  <html>
 3    <head>
 4     <title>Scholaria Circus</title>
 5      <link rel="stylesheet" href="CSS/style.css">
 6    </head>
 7
 8    <body>
 9      <header>
10        <img id="logo" src="images/circus2.png">
11        <h1>Scholaria Circus</h1>
12        <p>Welcome to the greatest show on earth!</p>
13      </header>
14
15      <nav>
16        <ul>
17          <li><a href="index.html">Home</a></li>
18          <li><a href="cast.html">The Cast</a></li>
19          <li><a href="show.html">The Show</a></li>
20          <li><a href="gallery.html">Gallery</a></li>
21        </ul>
22      </nav>
23
24      <main>
25        <h2>Leave  all that is real behind!</h2>
26        <p>What happens when fantasy becomes reality, and strolling
              down the street, you find yourself face to face with not
              one but <strong>three</strong> giant elephants, the most
              amazing trapeze artists jump over your head?</p>
27
28        <p>Not sure? then maybe it's time to find out at the
              <strong>Scholaria Circus</strong></p>
29        <img src="images/circus1.jpg">
30      </main>
31    </body>
32  </html>
```

CSS - style.css

```
 1  body {
 2    background-color: red;
 3    font-family: sans-serif;
 4  }
 5  header{
 6    background-color: yellow;
 7    margin: 10px;
 8  }
 9  h1 {
10    font-size: 32px;
11    margin: 0px 0px 0px 0px;
12    padding: 5px
13  }
14  /* A group selector has been used here to set both h1 and h2 with
       the same margins and padding. */
15  h1, h2{
16    margin: 0px 0px 0px 0px;
17    padding: 5px
18  }
19  p{
20    margin: 0px 0px 0px 0px;
21    padding: 5px;
22  }
23  #logo{
24    float:left;
25    padding-right:2px
26  }
27  main{
28    padding: 10px;
29    background-color: white;
30    margin: 10px;
31  }
```

Take some time to look through both the HTML and CSS to make sure you understand what is going on. If you are unsure or need some help speak to your tutor.

Create a new folder called images (inside your website folder).
Download https://courses.scholar.hw.ac.uk/vle/asset/Downloads/H-CCMP/Course%20Downloads/17D2E11E-E6B0-8BB4-1CAA-BBC6742FA762/ScholariaCircus.zip , extract the images and save them to the new folder you created.

Circus 1	Circus 2

Preview your page in your browser and you should now see the following:
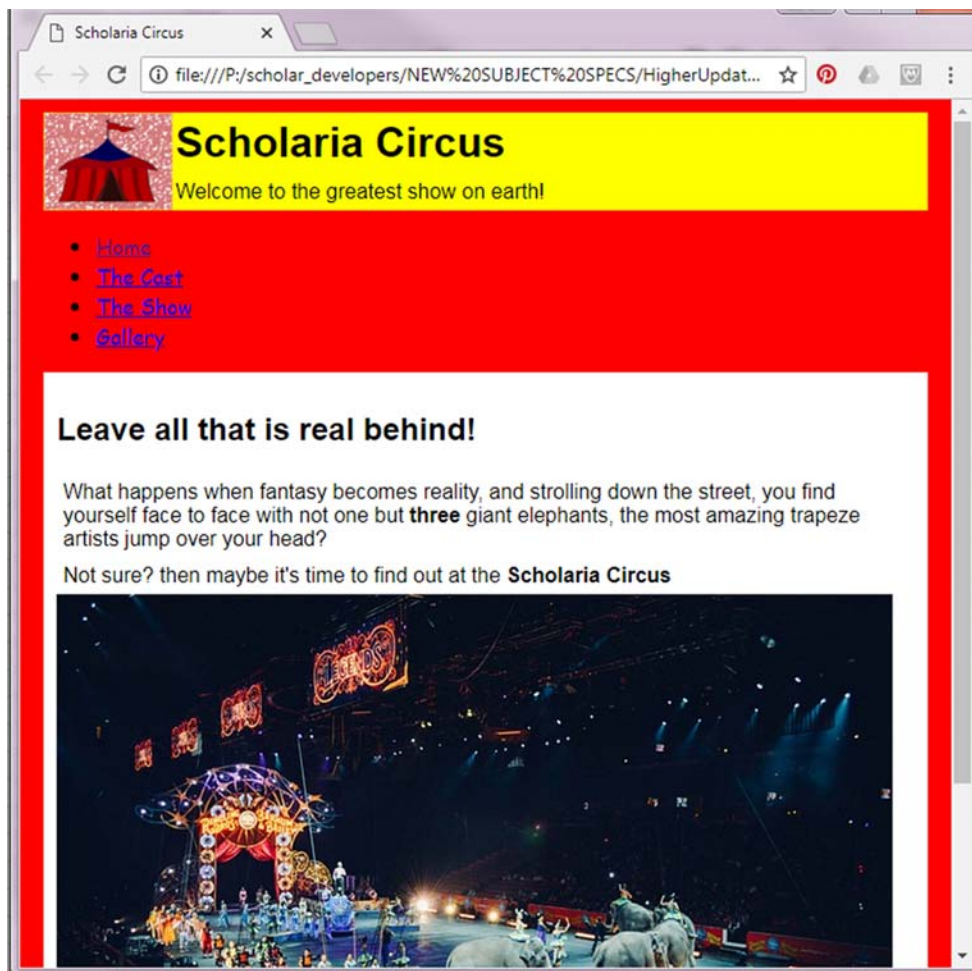
Removing the bullets

Now add the following to your CSS file:

```
1  /* This Section will control the navigation bar */
2  nav{
3    background-color: white;
4    font-family: cursive;
5    border: 1px solid black;
6    margin: 10px;
7  }
8  ul{
9    /* this property defines the bullets used, by setting it to none
         we are removing the bullets from our list*/
10   list-style-type:none;
11
12   /* By setting the margins and padding to 0 it will make our
         navigation bar fit our <nav> section */
13   margin: 0px;
14   padding: 0px;
15 }
```

Your page will now look like:
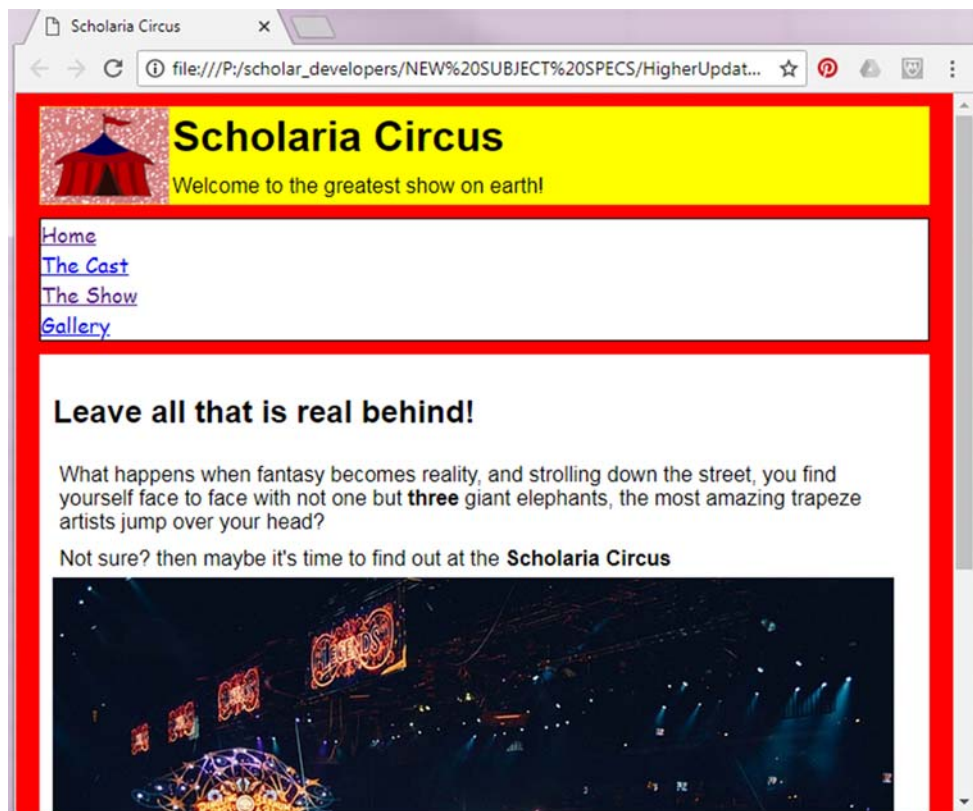
Creating the horizontal bar

At the moment we have a vertical navigation bar, to make this horizontal we need to change how each list item <li> displays.

In your CSS file enter the following rule:

```
1  li{
2      display: inline;
3  }
```

You should now see that your navigation bar has become a horizontal list as seen in the example below:
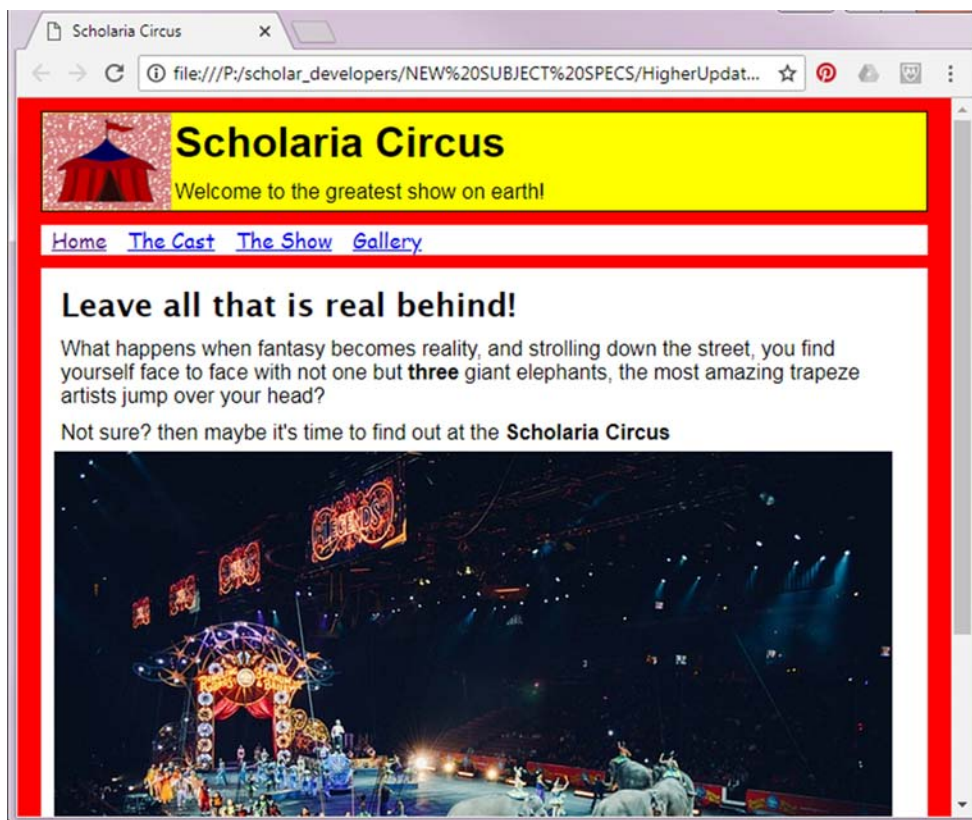
Using hover

Finally we're going to use the hover selector to highlight one of the links in our navigation bar if the user moves their mouse over it.

You may have seen this kind of functionality in when looking at JavaScript in the National 5 course however it can be implemented in CSS by using the descendant selector.

In your CSS file add the following selector and rule:

```
1  li a:hover{
2      background-color: yellow;
3  }
```

Save your CSS file, reload your web page and now hover over the hyperlinks in your menu and you'll see they change colour as with the example here:
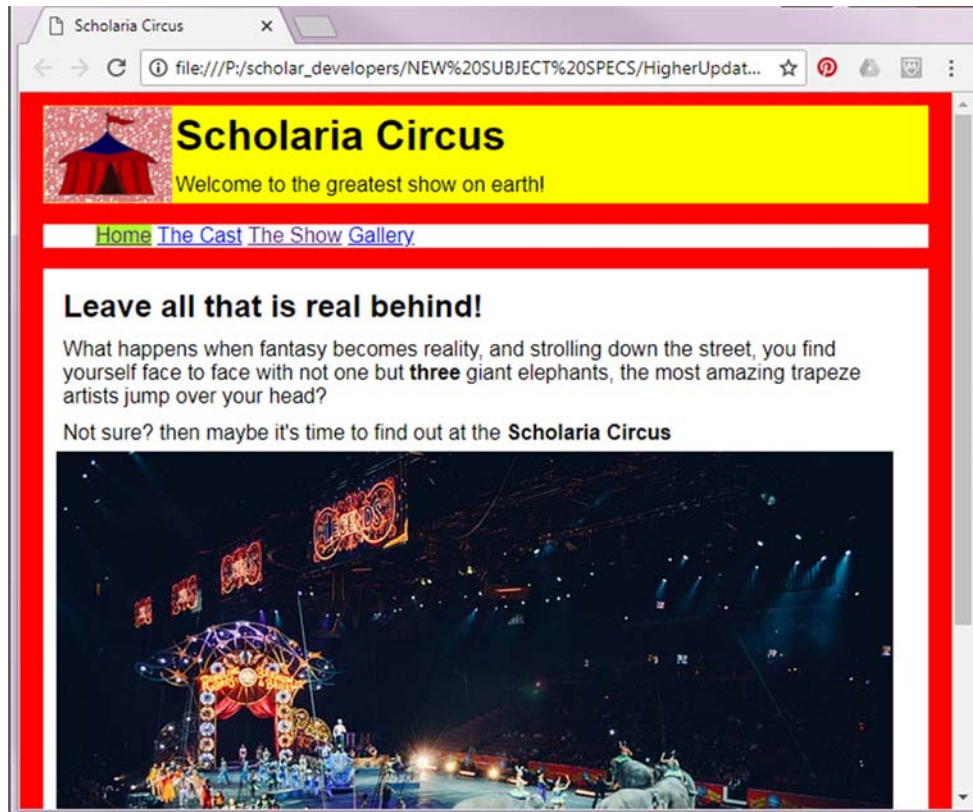
You may wish to learn more about the selectors in CSS and how they can add extra functionality to your website there is a full list available on W3Schools which can be accessed here:
https://www.w3schools.com/cssref/css_selectors.asp

4.6   Learning points

Summary

You should now be able to:

- inline styles, internal stylesheets and external Cascading Style Sheets (CSS);

- understand the order of priority given to each method of styling a web page;

- grouping selectors to increase efficiency of CSS and web pages;

- descendant selectors to increase efficiency of CSS and web pages;

- using the following properties and rules to control appearance and positioning of elements in a web page:

  ○ display (block, inline, none);
  ○ float (left, right);
  ○ clear (both);
  ○ margins / padding;
  ○ sizes (height, width);

- grouping and descendant selectors to create horizontal navigation bars;

- use the following properties and rules when creating a horizontal navigation bar:

  ○ list-style-type:none;
  ○ hover;

- read and explain code that makes use of the above CSS.

4.7   End of topic test

End of topic test: CSS   Go online

Q1:   CSS comments start with /* and end with */

a)   True
b)   False

....

Q2:   How do you add the following padding to an HTML element:

The top padding = 15 pixels
The bottom padding = 15 pixels
The left padding = 10 pixels
The right padding = 10 pixels

a)   padding:15px 15px 10px 10px;
b)   padding:15px 10px 15px 10px;
c)   padding:10px 10px 15px 15px;
d)   padding:15px, 10px, 15px, 10px;

....

Q3:   How do you select class name "active" in a CSS document?

a)   .active
b)   #active
c)   active
d)   *active

....

Q4:   How do you group selectors in a CSS document?

a)   Separate each selector with a plus sign
b)   Separate each selector with a comma
c)   Separate each selector with a space

Topic 5

Implementation: JavaScript

Contents

Prerequisites

From your studies at National 5 you should already know how to:

- understand that JavaScript is used to help users interact with a website;

- describe and identify the following JavaScript events:

    ○ onmouseover;
    ○ onmouseover.

Learning objective

By the end of this topic you should be able to know how to efficiently use:

- use a scripting language (JavaScript) to add interactivity to a webpage by using the following events:

    ○ onmouseover;
    ○ onmouseout;
    ○ onclick.

5.1 JavaScript

JavaScript is the default scripting language used in web development. This allows us to use the <script> tags in our HTML document to store our JavaScript code.

JavaScript can be used to add extra functionality or interactivity to a webpage. JavaScript can offer all the functionality of a programming language but must be run inside a web browser.

Similar to CSS rules we use JavaScript code directly inside the HTML tag, as a script within our HTML file or store it in an external file (.js) that is linked to from our HTML document.

Download https://courses.scholar.hw.ac.uk/vle/asset/Downloads/H-CCMP/Course%20Downloads/67FF6DFA-0A4A-3F50-4A60-FBFA30429E30/CodePositioning.zip , extract the files and preview in your Web Browser.

This page uses the three different ways of storing JavaScript code.

HTML

```
 1  <!DOCTYPE html>
 2  <html>
 3    <head>
 4      <meta charset="utf-8">
 5      <meta name="viewport" content="width=device-width, initial-scale=1">
 6      <title>Where to put your JavaScript</title>
 7      <link rel="stylesheet" href="CSS/styles.css">
 8    </head>
 9    <body>
10      <header>
11        <h1>Inline, Internal Script and External Scripts</h1>
12          <p>Move your mouse over the sections below.</p>
13      </header>
14
15      <main>
16        <body>
17          <div>
18          <p>The paragraph below has the JavaScript code stored in the
             &lt;p&gt; tag </p>
19            <p onmouseover="this.style.color='blue';
                 this.style.background='lightgreen';"
20               onmouseout="this.style.color='black';
                   this.style.background='lightsalmon';">
21               Used in Web pages, JavaScript is a "client-side"
                   programming language.</p>
22          </div>
23          <div>
24              <p>The paragraph below calls the JavaScript functions
                 stored in the &lt;script&gt; tag </p>
25            <p onmouseover="blueText(this)" onmouseout="blackText(this)">
26               This means JavaScript scripts are read, interpreted and
                   executed in the client, which is your Web browser.</p>
27          </div>
28          <div>
29              <p>The paragraph below calls the JavaScript functions
                 stored in the external JavaScript file
                 <strong>"scripts.js"</strong> </p>
30            <p onmouseover="extBlueText(this)"
                 onmouseout="extBlackText(this)">
```

```
31              By comparison , "server-side" programming languages run on
                   a remote computer , such as a server hosting a
                   website.</p>
32          </div >
33        </body >
34      </main >
35
36      <script >
37        function blueText(elem){
38          elem.style.color = "blue";
39          elem.style.background='lightgreen ';
40        }
41        function blackText(elem){
42          elem.style.color = "black";
43          elem.style.background='lightsalmon ';
44        }
45      </script >
46      <script src="Scripts/script.js"></script >
47    </body >
48 </html >
```

JavaScript

```
1  function extBlueText(elem) {
2    elem.style.color = "blue";
3    elem.style.background='lightgreen ';
4  }
5  function extBlackText(elem) {
6    elem.style.color = "black";
7    elem.style.background='lightsalmon ';
8  }
```

Key point

There is no rule as to where to put any JavaScript (or link to external JavaScript file) used in your webpage. Some developers will place it in the head section, some will have it stored in the page where it will be used and others will put it at the end of the file (within the body tags).

In most of the examples seen here the JavaScript or link to the script will be placed at the end of the document, this will ensure that all of the page to be displayed is loaded before any JavaScript code is executed.

A mark-up language can be called the language of the web and is used to create web pages. HTML stands for Hypertext Mark-up Language. HTML uses pairs of tags, which a browser interprets to know how to display a web page. Importantly, markup languages only contain information about the content of the page. They do not offer the sort of constructs found in programming languages, for example: data structures like variables; repetition with loops; selection using IF statements; arithmetical and logical operations. This is where a scripting language like JavaScript can be useful.

Javascript can be used for a number of tasks in a web page:

- To create a more active page than regular HTML (things happen based on user interaction)

- To validate input on forms (check someone has entered all the required fields)

- To customise page content based on

    ◦ The time of day
    ◦ Browser being used
    ◦ Location
    ◦ Cookies previously set

- JavaScript can control the web browser (so you've got JavaScript to thank for all those annoying popups!)

- JavaScript can be used for pop-up menus, image galleries, collapsible 'accordion' areas and much more!

Javascript also has some limitations:

- It cannot connect to a web-based database on its own

- It cannot access files on your computer, apart from cookies

- It cannot keep track of settings long-term (except with the help of cookies); if you leave the page or hit reload, any stored data the user has typed in will be gone.

JavaScript is called a **client-side scripting language** because the program is run by the 'client' - your browser. This saves the **web server** having to do all the processing, which reduces the overall demand on it. It can also make web pages quicker to respond: for example if you validate a form in the browser with JavaScript, you don't need to transmit the information back from the server to find out if it's correct or not.

There are also **server-side scripting languages**. As you might have worked out, they need to be run by the server, and send the finished results to you. This is useful as they can interact with databases of information based on the web server, generating database-driven HTML code on the server and sending finished HTML page to your browser.

Figure 5.1: Client-side scripting process

2. Page including JavaScript sent

3. JavaScript runs on client

1. Request sent
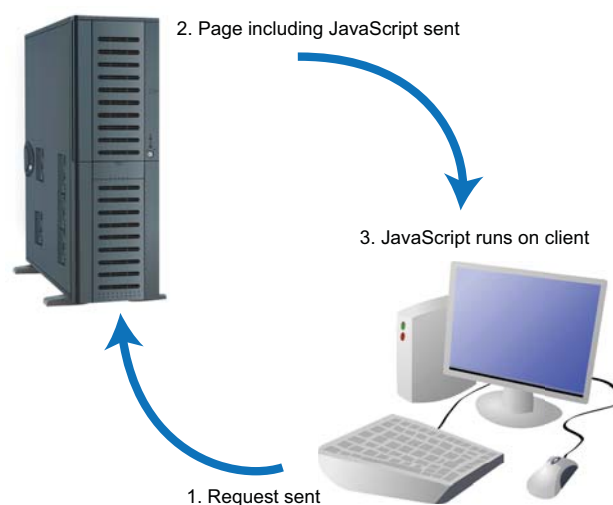
Client-side scripting and server-side scripting are two different ways to customise web pages and allow scripts to run. A script is a set of instructions for a web page and they provide a change to the way that a web page is displayed. Any page that changes every time you visit it will probably be using scripts. Logging into a website or selecting from a menu uses scripts. You will learn more about client-side scripting and server-side scripting in the rest of this topic.

Quiz: Scripting (10 min)   Go online

Q1: What is a mark-up language?

....

Q2: What does HTML stand for?

....

Q3: When considering a web page, what is a script?

....

Q4: Give three examples of what a scripting language can be used for in a web page.

5.2   HTML Document Object Model (DOM)

In order to understand how JavaScript can access and make changes to elements in an HTML document we need to look at the HTML Document Object Model (DOM).

The DOM is created when your web browser opens your webpage.
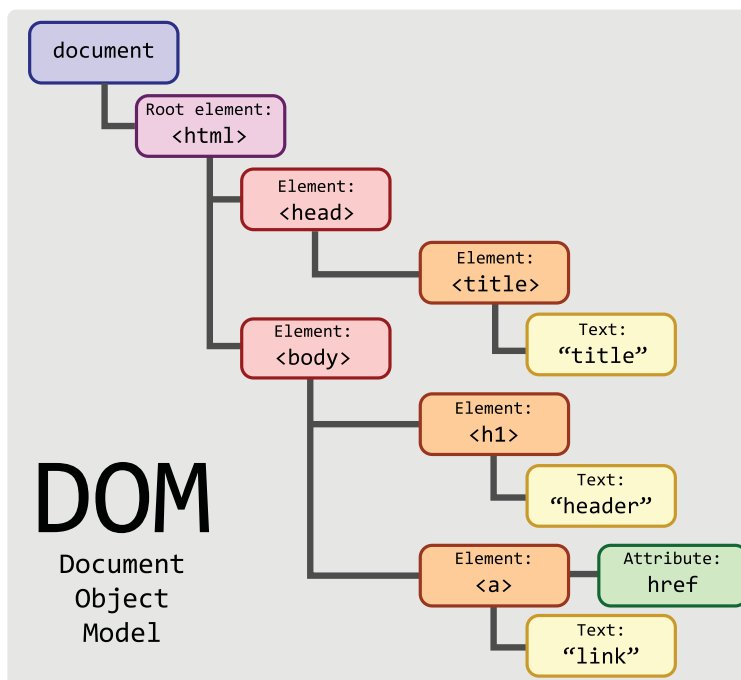
In this image you can see a tree like structure that details all the elements stored in a web page.

5.2.1 What is the DOM?

The Document Object Model is a platform and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents.

The document can be further processed and the results of that processing can be incorporated back into the presented page.

(Source: https://www.w3.org/DOM/)

5.2.2 What is the HTML DOM?

The HTML DOM is a standard that allows us to perform object oriented programming to effect changes to HTML elements.

This allows us when using JavaScript to do things such as:

- change all HTML elements and attributes in a page;

- change the styles used in a CSS document.

5.3 JavaScript Functions

JavaScript functions can be used to complete the following tasks:

- hiding and revealing page elements;

- changing the:

    ◦ position, size and colour of an element;
    ◦ look of text;

In this section we will be looking at three "MouseEvent" JavaScript functions:

1. **onmouseout**

   This function detects when the user's mouse pointer moves over a selected HTML element.

2. **onmouseover**

   This function detects when the user's mouse pointer moves away from a selected HTML element.

3. **onclick**

   This function is triggered when a user clicks their mouse button when the mouse pointer is over the selected HTML element.

These functions are triggered by the website user interacting with HTML elements that have JavaScript functions listed inside its tag.

Activity: onmouseover and onmouseout

Create a new folder called "javascript practical 1".

Create a new webpage in a text editor using this code:

```
1   <!DOCTYPE html>
2   <html>
3     <head>
4       <title>JavaScript </title>
5     </head>
6
7     <body>
8       <header>
9         <h1 onmouseover="redText(this)"
              onmouseout="blackText(this)">Welcome to Scholar!</h1>
10      </header>
11      <main>
12        <section>
13          <p>The onmouseover event calls the function redText() when
                the mouse pointer is moved over the text.</p>
14          <p>The onmouseout event calls the function blackText()
                when the mouse pointer is moved away from the text.</p>
15        </section>
16      <main>
17    </body>
18    <script>
19      function redText(x) {
20        x.style.color = 'red';
21      }
22      function blackText(x) {
23        x.style.color = 'black';
24      }
25    </script>
26  </html>
```

Test your webpage in a web browser and check that both functions work.

Now apply the existing onmouseover and onmouseout events to the two paragraphs in the page by adding this code to each <p> tag.

```
onmouseover="redText(this)" onmouseout ="blackText(this)"
```

Save your page and test that these now work and change the colour of each paragraph.

Extension 1

Create two new functions to use different colours and styles for the two paragraphs when the onmouseover event is triggered and have the text return to normal when the onmouseout event is triggered.

Extension 2

Using your text editor take the code from within your script tags and save it into a new file, save it as "**scripts.js**".

Replace your scripts tags above with the following line:

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

Now test your page is loading the scripts and that your functions all still work.

Activity: onclick

Create a new html file in your text editor and enter this code:

```
 1  <html>
 2    <head>
 3      <title>JavaScript onclick Function Example</title>
 4    </head>
 5
 6    <body>
 7      <header>
 8        <h1>Press the button below to call the function and display
              the message!</h1>
 9      </header>
10      <main>
11        <button type="button" onClick="message()">Click here</button>
12      </main>
13    </body>
14    <script>
15      function message()
16      {
17        alert("This is a function being called");
18      }
19    </script>
20  </html>
```

Save your page and test it in your web browser.

Further reading on JavaScript events can be found on:
https://www.w3schools.com/jsreF/default.asp.

5.4   Three Functions

In the HTML document below you can see that it contains three <div> elements each using a different JavaScript function call within the tag. Notice that each JavaScript function contains two internal elements:

- <p>

- <span>

```
1   <!-- onmouseover function -->
2          <div onmouseover="myOverFunction()">
3              <p>onmouseover: <br> <span id="mouseover">Mouse over
                   me!</span></p>
4          </div>
5
6          <!-- onmouseout function -->
7
8          <div onmouseout="myOutFunction()">
9              <p>onmouseout: <br> <span id="mouseout">Mouse over
                   me!</span></p>
10         </div>
11
12         <!-- onclick function -->
13         <div onclick="myClickFunction()">
14             <p>onclick: <br> <span id="onclick">Click me!</span></p>
15         </div>
```

These function calls trigger this JavaScript:

```
1   /* Variables needed for this script */
2   var x = 0;
3   var y = 0;
4   var z = 0;
5
6   /* The functions below will add 1 to the variables listed above when
       the user performs a specific action on the appropriate HTML element
       */
7
8   function myOverFunction() {
9      document.getElementById("mouseover").innerHTML = x+=1;
10  }
11  function myOutFunction() {
12     document.getElementById("mouseout").innerHTML = y+=1;
13  }
14  function myClickFunction() {
15     document.getElementById("onclick").innerHTML = z+=1;
16  }
```

Activity: Three Functions

Download https://courses.scholar.hw.ac.uk/vle/asset/Downloads/H-CCMP/Course%20Downloads/4DA6D916-EC05-4A55-3347-EE2BB0203A61/ThreeFunctions.zip , extract the files and preview in your web browser, you should see this page:
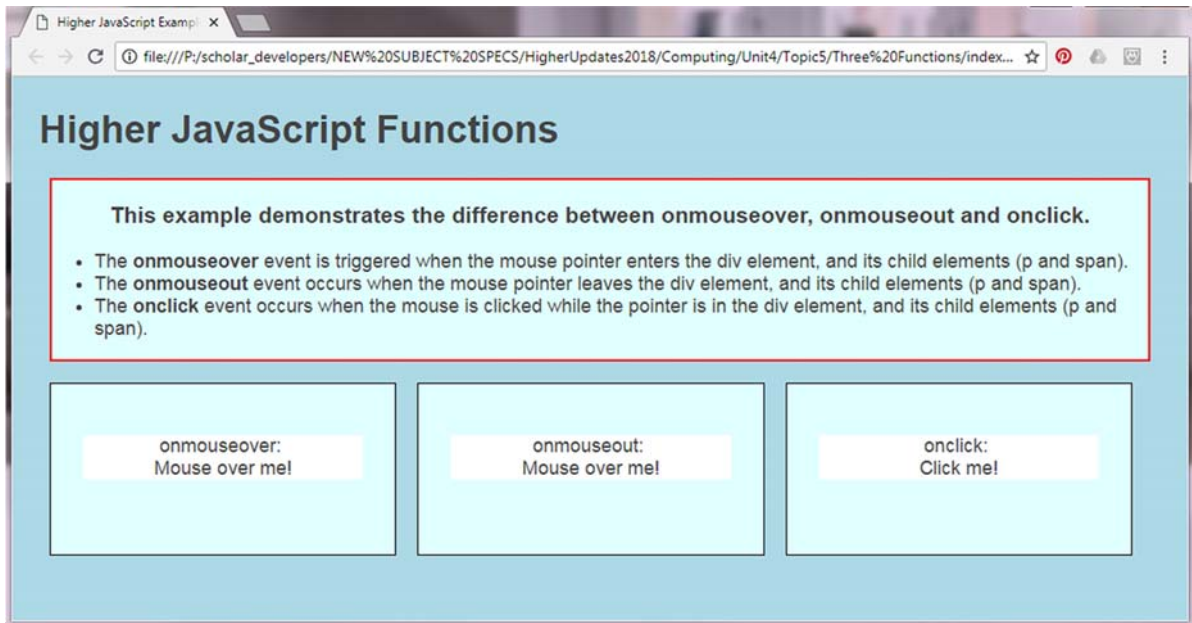
Try out the onmouseover and onmouseout functions by moving your mouse over each of the first two <div> blocks and note when the text underneath the function name changes to a number that will:

- count how many times you either moved your mouse over the element (<div>, <p> or <span>) or;

- count how many times the mouse pointer moves away from the element.

The third element will count how many times the user clicks their mouse button when the mouse pointer is over the element.

Activity: The Scottish Mental Health Society

You're going to create a website for the "The Scottish Mental Health Society".

They want a new website created to help young children understand their basic emotions (happy, excited, scared...). They want the user to be able to click a button and get a definition and see a picture of the matching emotion.

Download https://courses.scholar.hw.ac.uk/vle/asset/Downloads/H-CCMP/Course%20Downloads/F9DF729C-3CE0-6C4A-88E3-C18887A5FC38/TheScottishMentalHealthSo , extract the files and preview in your Web Browser.

You should now have the following inside of an HTML and CSS file along with associated images.

HTML

```
1  <!DOCTYPE html>
2  <!-- JavaScript onClick function -->
3  <!-- In this example we will look at how we can use the onclick
       function to interact with existing HTML elements in our page by
       showing or hiding them. -->
4  <html>
5  <head>
6      <meta charset="utf-8">
7      <meta name="viewport" content="width=device-width,
          initial-scale=1.0">
8      <title>Emotions</title>
9      <link rel="stylesheet" href="CSS/style.css">
10 </head>
11 <body>
12     <header>
13         <img class=feelings src="images/Emotions.png" />
14         <h1 >Emotions</h1>
15     </header>
16     <main>
17         <section id="start">
18             <div id="image">
19                 <img id="myImage" alt="" src="images/emoji.png">
20             </div>
21             <div id="Explanation">
22                 <h2 id="heading">Get to know your emotions</h2>
23                 <p id="exp">An explanation of each emotion will
                      appear here</p>
24             </div>
25         </section>
26         <section id="excited">
27             <div id="image">
28                 <img id="myImage" alt="" src="images/excited.png">
29             </div>
30             <div id="Explanation">
31                 <h2 id="heading">Excited</h2>
32                 <p id="exp">Feeling very happy and enthusiastic</p>
33             </div>
34         </section>
35         <section id="happy">
36             <div id="image">
37                 <img id="myImage" alt="" src="images/happy.png">
38             </div>
39             <div id="Explanation">
40                 <h2 id="heading">Happy</h2>
41                 <p id="exp">delighted, pleased, or glad, as over a
                      particular thing</p>
42             </div>
43         </section>
44         <section id="scared">
45             <div id="image">
```

```
46                    <img id="myImage" alt="" src="images/scared.png">
47                </div>
48                <div id="Explanation">
49                    <h2 id="heading">Scared</h2>
50                    <p id="exp">to feel frightened or worried about
                          something</p>
51                </div>
52            </section>
53        </main>
54        <div id="buttons">
55            <!-- Each of the buttons in this section when clicked
                   triggers a JavaScript to change the image and text in
                   the section above -->
56            <hr>
57            <button class=button type="button"
                   onclick=displayExcited()>Excited</button>
58            <button class=button type="button"
                   onclick="displayHappy()">Happy</button>
59            <button class=button type="button"
                   onclick="displayScared()">Scared</button>
60            <button class=button type="button"
                   onclick=displayStart()>Reset</button>
61        </div>
62        <footer>
63            <h6>The Scottish Mental Health Society</h6>
64        </footer>
65        <script>
66            /* insert the given JavaScript code here or link to an
                   external file containing the code you wish to use*/
67        </script>
68    </body>
69 </html>
```

CSS

```
 1  body {
 2      background-color: darkorchid;
 3      color: White;
 4      font-family: sans-serif;
 5  }
 6  img {
 7      max-width: 300px;
 8      border: 1px solid black;
 9      margin-right: 10px;
10      max-height: 200px;
11      padding: 10px;
12  }
13  section {
14      height: 250px;
15  }
16
17  #image {
18      float: left;
19  }
```

```
20  #buttons {
21      clear: both;
22      text-align: center;
23  }
24  #start {
25      display: block;
26  }
27  #excited,
28  #happy,
29  #scared {
30      display: none;
31  }
32
33  footer{
34      text-align: center;
35  }
36  .feelings{
37      height:1000px;
38      float:right;
39      border: none;
40  }
41
42  /*The CSS below is to add styles to the buttons*/
43  button {
44      background: #ffff00;
45      border-radius: 5px;
46      padding: 8px 20px;
47      font: normal 700 24px/1 sans-serif;
48      text-align: center;
49  }
```

Preview the page in your web browser and you should see this webpage:
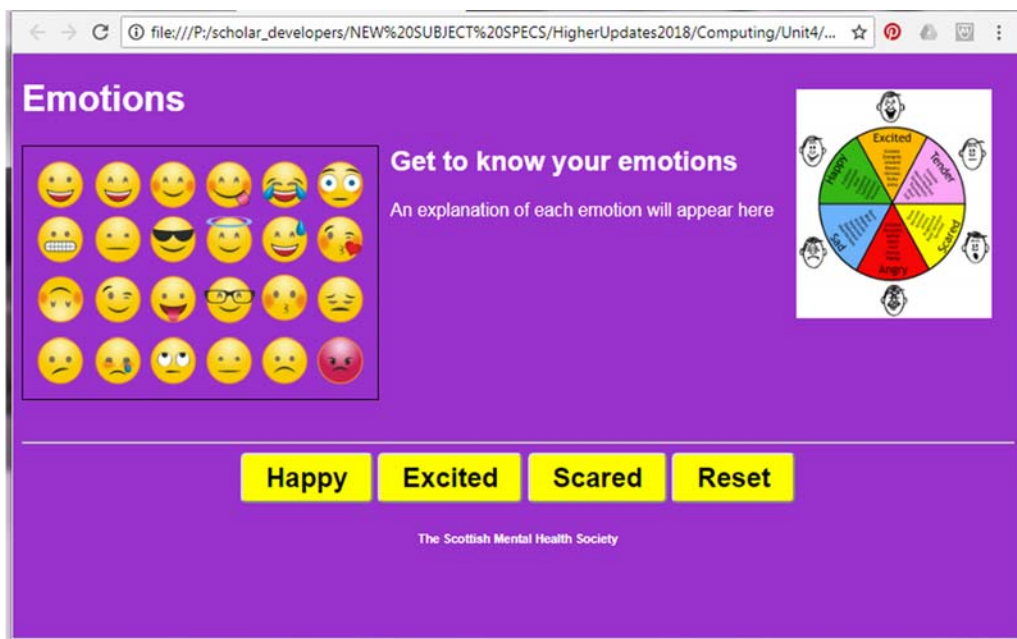
While the functions have already been referenced by each of the buttons the code does not yet exist so none of the buttons currently work.

Insert the following code into your web page JavaScript section (or create an external .js file):

```
 1  /*Note these functions edit the CSS display properties for the
        given elements.
 2  If it's to be displayed it is set to block and if it is to be
        hidden it is set to none*/
 3
 4  function displayStart() {
 5      document.getElementById("start").style.display = "block";
 6      document.getElementById("happy").style.display = "none";
 7      document.getElementById("excited").style.display = "none";
 8      document.getElementById("scared").style.display = "none";
 9  }
10
11  function displayHappy() {
12      document.getElementById("start").style.display = "none";
13      document.getElementById("happy").style.display = "block";
14      /* Complete the rest of this function */
15  }
16
17  function displayExcited() {
18      /* Complete this function yourself */
19  }
20
21  function displayScared() {
22          /* Complete this function yourself */
23  }
```

Now test the site again in your web browser and two button (Happy and Reset) will work correctly however, you will need to complete the JavaScript code for the three unfinished functions:

- displayHappy()

- displayExcited()

- displayScared()

5.5   Learning points

Summary

You should now be able to:

- use a scripting language (JavaScript) to add interactivity to a webpage by using the following events:

    ◦ onmouseover;

    ◦ onmouseout;

    ◦ onclick.

5.6   End of topic test

End of topic test: JavaScript   Go online

Q5:   How do you create a function in JavaScript?

a) function myFunction()
b) function = myFunction()
c) function:myFunction()

....

Q6:   How do you call a function named "myFunction"?

a) call myFunction()
b) myFunction()
c) call function myFunction()

....

Q7:   Which event occurs when the user clicks on an HTML element?

a) onmouseclick
b) onmouseover
c) onclick
d) onchange

....

Q8:   Which event occurs when the user's mousepointer leaves an HTML element?

a) onmouseclick
b) onmouseover
c) onclick
d) onchange

....

Q9: Where can JavaScript be placed in an HTML document?

a) The head section
b) Inside the element tag
c) After the </html> tag
d) Anywhere in the document between the <html> and </html> tags

Topic 6

Testing and evaluation

Contents

Prerequisites

From your studies at National 5 you should already know how to:

- that all:

  - webpages should match the user-interface design;
  - links (internal and External) and navigation work correctly;
  - media in the page displays correctly (text, graphics, and video);
  - pages in a site have a consistent theme;

- how webpages can be evaluated as being fit for purpose.

Learning objective

By the end of this topic you should be able to:

- understand and perform usability testing making use of:

  - personas;
  - test cases;
  - scenarios based on low fidelity prototypes;

- describe and perform testing on:

  - input validation for data being entered into a form (Client Side Scripting);
  - the site's navigational bar both displays correctly and all links work;
  - all of the media content displays correctly;

- understand and perform compatibility testing to ensure that websites display properly on a variety of different devices:

  - tablet;
  - smart phone;
  - desktop;

- understand the importance and perform compatibility testing to ensure that websites display properly on a variety of different software:

  - operating systems;
  - web browsers.

- evaluate:

  - if your website is fit for purpose;
  - your website's usability.

6.1 Usability testing

Introduction

All computing projects follow an iterative pattern. One such common example is the Software Development Process:

1. Analysis

2. Design

3. Implementation

4. Testing

5. Documentation

6. Evaluation

7. Maintenance

You may already know from studying the Software Design and Development unit at National 5 that after the implementation stage of the software development process comes testing. All stages of the software development process are important but testing is essential to ensure that a project has been completed to the highest standard possible. Errors can be detected at the testing stage and fixed before the project is published, or handed over to the client.

You may also have heard of different types of methodologies such as rapid application development or agile programming. Here, testing is also important but may be organized differently. Much of the testing will be done in small parts during implementation, with larger-scale beta testing coming near the completion.

At National 5 you will have learnt that a website should be tested to ensure that all the hyperlinks work as expected, that there are no spelling or grammar errors, that pictures display correctly and that any other features, such as a JavaScript application work correctly.

This type of testing is usually performed by the same team that created the website. This is sometimes called 'in-house testing' or **alpha testing**. Some developers may involve the client at this stage, though this is not strictly necessary.

6.2 Usability testing

Learning objective

By the end of this section you should be able to:

- understand and perform usability testing making use of:

  - personas;
  - test cases;
  - scenarios based on low fidelity prototypes;

Usability testing is a technique used in user-centred design to evaluate a product by testing it on real users performing real tasks. Usability testing should be focused on the most important user goals (such as being able to change a password) and/or the most important organisation goals (for an ecommerce web site this would be making a purchase). These tasks will be detailed in the User Stories, User Scenarios and Use Cases.

6.2.1   Usability testing techniques

Usability testing involves watching people trying to use something for its intended purpose. Data can be gathered using a number of techniques such as:

- Personas;

- test cases;

- low-fidelity protoypes.

Personas

At the testing stage the development team will revisit the personas that were previously created in the design stage and have testers take on this persona when testing the website. They will then use this to ensure that the goals created and the frustrations for each of the personas have all been addressed.

Test cases

A list of tests would be created from the initial user requirements, testers will work through this list to ensure all features and functions the clients have requested have been created and work as expected.

Low Fidelity prototypes

We have already looked at scenarios based on what Low Fidelity Prototypes are in the 'Design' topic, in the section 'User interface design: Low fidelity prototype'. Testers would now be asked to interact with the prototype to perform regular functions the site would need to be able to handle. Their actions would be watched and observations noted to help improve the site.

6.3   Functional testing

Learning objective

By the end of this section you should be able to:

- describe and perform testing on:

  ○ input validation for data being entered into a form (Client Side Scripting);

  ○ the site's navigational bar both displays correctly and all links work;

  ○ all of the media content displays correctly.

6.3.1 Input validation

Learning objective

By the end of this section you should be able to:

- apply validation to form elements that checks that:
    - the user has entered data (presence check);
    - data is within a set range (range check);
    - the data entered is within a set maximum / minimum length.

Validation is the key to protecting against code injection and cross-site scripting attacks. By validating the form data, we can protect against these exploits.

Client side validation techniques

Any validation carried out on the client site, using either JavaScript or another client technology, can be subverted. There are tools, such as interception proxy servers, that intercept the data between the client and the server and allow the hacker to change the requests before they can be sent effectively rendering all client side validation for the purpose of security useless.

However, for the typical user, carrying out some kind of client side validation is useful as it can catch errors before data is sent to the server.

Activity: Sign up form: Validation

We're now going to go back to the sign up form in the section on Form Elements that we created in the topic Implementation: HTML.

HTML5 carries out validation based on the type attributes of form collection elements. There is no specific markup required in order to activate form validation - it is on by default.

The first field, username, is text and is required. The web browser will validate the field so that it must not be empty and contains characters. As long as the user entered al least one character the field it will validate.

We can use the :valid and :invalid CSS selectors to format valid and invalid fields providing visual information to the user.

Download each of these files and place them in your images folder.

valid.png

invalid.png

Please go online to download these files.

Add the following CSS rules to your styles.css file.

```
1  .signup_form input:focus:invalid, .signup_form
       textarea:focus:invalid {
2    /* when a field is considered invalid by the browser */
3    background: #fff url(images/invalid.png) no-repeat 98% center;
```

```
 4        box-shadow: 0 0 5px #d45252;
 5        border-color: #b03535
 6   }
 7
 8   .signup_form input:required:valid, .signup_form
          textarea:required:valid {
 9   /* when a field is considered valid by the browser */
10        background: #fff url(images/valid.png) no-repeat 98% center;
11        box-shadow: 0 0 5px #5cd053;
12        border-color: #28921f;
13   }
```

Load the form and fill out the values to see how the use of the :valid and :invalid selectors affects the form.

Further validation can be performed using the HTML5 **pattern** attribute. The username must be a combination of only letters and numbers; no special characters or spaces are allowed.

Activity: Sign up form: CSS formatting

To complete the formatting of the form add the following CSS rules to your styles.css file.

```
 1   .form_hint {
 2     background: #d45252;
 3     border-radius: 3px 3px 3px 3px;
 4     color: white;
 5     margin-left:8px;
 6     padding: 1px 6px;
 7     z-index: 999; /* hints stay above all other elements */
 8     position: absolute; /*allows proper formatting if hint is two
          lines*/
 9     display: none;
10   }
11
12   .form_hint::before {
13     content: "\25C0"; /* left point triangle in escaped unicode */
14     color:#d45252;
15     position: absolute;
16     top:1px;
17     left:-6px;
18   }
19
20   .signup_form input:focus + .form_hint {display: inline;}
21
22   /* change form hint color when valid */
23   .signup_form input:required:valid + .form_hint {background:
          #28921f;}
24
25
26   /* change form hint arrow color when valid */
27   .signup_form input:required:valid + .form_hint::before
          {color:#28921f;}
```

> This CSS formats the hints so they are only shown when editing a field.

HTML5 can be used to carry out validation on the client side however every input must be validated on the server side and made suitable for use.

6.3.2   Site validation

We now need to perform some tests to ensure our users can navigate around the website and reach all pages.

We would test:

- navigational bar links all work and link to the correct page;

- all external links work;

- all pages link back to the home page;

- internal links between pages work correctly;

- for orphan pages.

It may be useful to create a test table similar to one you would use in software development. An example format is given here:

Area of webpage being tested	URL being tested	Internal / External link	Expected Outcome	Actual Outcome	Result (Pass / Fail)	Date tested	Actions required
Navigation Bar <nav>							
Body of webpage <body>							
Footer of webpage <footer>							

Example: Scholaria Circus website testing

This table was created and used to test the Scholaria Circus website we created in the 'Implementation: CSS' topic.

Website: Scholaria Circus							Webpage: index.html

Area of webpage being tested	URL being tested	Internal / External link	Expected Outcome	Actual Outcome	Result (Pass / Fail)	Date tested	Actions required
Navigation Bar <nav>	Index.html	Internal	Homepage of the site loads	Page loads correctly	Pass	31/07	None
	Cast.html	Internal	Web page with cast list loads	Page loads correctly	Pass	31/07	None
	Show.html	Internal	Web page with show information loads	Page does not load	Fail	31/07	Check <a> tag and that page exists
	Gallery.html	Internal	Web page with photo gallery loads	Page loads correctly	Pass	31/07	None

Area of webpage being tested	URL being tested	Internal / External link	Expected Outcome	Actual Outcome	Result (Pass / Fail)	Date tested	Actions required
Body of webpage <body>	circusbookings.co.uk	External	Web page load in new browser tab to allow purchase of tickets	No page loads	Fail	31/07	Check url and <a> tag are correct. Check external site is working.
Footer of WebPage <footer>	Contact.html	Internal	Webpage with contact form for users	No page loads	Fail	31/07	Check <a> tag and that page exists.

Activity: Visit Scholaria testing table Go online

Create a navigation testing table for the homepage (index.html) of the "Visit Scholaria" website.

Website:					Webpage:		

Area of webpage being tested	URL being tested	Internal / External link	Expected Outcome	Actual Outcome	Result (Pass / Fail)	Date tested	Actions required
Navigation Bar <nav>							
Body of webpage <body>							
Footer of webpage <footer>							

6.3.3 Page content (media)

The final test is to check that all the content (text, graphics, videos) that was planned in the design stage appear in the correct position, using the correct styles, colours and sizes as indicated in the wireframe design.

Wireframe design	*Actual page*

Again when testing our web pages it is useful to create a table like the one used below to test the Circus website we looked at previously.

| Website: Scholaria Circus | | | | Webpage: index.html |

Media / Styles / Javascipt function	Expected Outcome	Result (Pass / Fail)	Date tested	Actions required
Text	All planned text appearing in page and in correct location.	Fail	01/08	Check all elements have been used correctly. Check to display property has been set to none.
Graphics	Image of a circus tent to appear in header of page.	Pass	01/08	
	Big image of the circus to appear in main section of the page.	Fail	01/08	Check image is in correct folder Check <img> tag in HTML document is correct.
Videos	None	N/A	N/A	None
CSS Styles	Header - yellow background	Pass	01/08	
	Navigation bar displayed inline	Pass	01/08	
	Hover over link colour	Pass	01/08	
JavaScript	None	N/A	N/A	None

Activity: Emotions testing table   Go online

Create a testing table for the "Emotions" website created in the 'Implementation: JavaScript' topic.

A blank table has been provided as a starting point, you can adapt it to suit the page you are testing.

| Website: | | | | Webpage: |

Media / Styles / Javascipt function	Expected Outcome	Result (Pass / Fail)	Date tested	Actions required

---

6.4 Compatibility

> **Learning objective**
>
> By the end of this section you should be able to:
>
> - understand and perform compatibility testing to ensure that websites display properly on a variety of different devices:
>   - tablet;
>   - smartphone;
>   - desktop;
>
> - understand the importance and perform compatibility testing to ensure that websites display properly on a variety of different software:
>   - operating systems;
>   - web browsers.

Users now view information systems on a variety of different devices, often via websites - desktops, laptops, notebooks, tablets and smartphones. Whilst all of these devices are able to display web pages they are still very different and have different features. For example, a smartphone has different input devices to a laptop, a different operating system, a smaller memory, smaller amount of RAM, smaller cache and a slower processor. This means that not all devices can display the same version of a website. Even on the same type of computer, different browsers can render page elements differently, and use different technologies.

> **Activity: Compatibility (20 min)**   Go online
>
> Use the internet to complete the table below:
>
> | Device | Screen Size | Amount of RAM | Backing storage | Processor Speed | Input Devices | Operating System |
> |---|---|---|---|---|---|---|
> | Desktop | | | | | | |
> | Laptop | | | | | | |
> | Tablet | | | | | | |
> | Smartphone | | | | | | |

You will find that the different types of devices all have different specifications. When you compare the more traditional devices, desktops and laptops, with the more current devices, tablets and smartphones, you will find that tablets and smartphones have less memory and processing power, and use different operating systems and input devices.

If you have used a smartphone before you may have found that when you try to view a web page you are directed to an app, or the smartphone version of the website. This is because the 'normal' version of the website cannot display on the smartphone for several possible reasons. Perhaps the screen size is too small? Is there enough space in main memory? Is there bandwidth constraints or

is it costly to use mobile data? Is it compatible with the mobile browser? Is the processor powerful enough?

When you are creating a website it is important to think about who is going to use the website and what type of device they are going to use. You might find that it is necessary to create different versions of the same website that are suitable for different devices. If your website cannot be viewed from a smartphone or a tablet then you are limiting the number of potential users and customers that you will receive.

Obviously, maintaining different versions of the same website would be inefficient, so there has been a gradual move towards 'responsive web design'. This means that one website can be coded in HTML where each element's size and position can then be controlled by CSS. The platform can be detected by a script on the page and the appropriate style sheet selected. Modern browsers usually support media-type definition in CSS by default.

Quiz: Compatibility (15 min)   Go online

Q1:   Why is the amount of space in memory that a website uses important to viewers?

....

Q2:   How can you ensure that your website can be viewed on all types of computer?

....

Q3:   Why do style sheets aid usability?

6.5   Fitness for purpose

When evaluating if a website is fit for purpose we need to consider several things.

- Has the intended purpose of the website been met?

- What were the requirements of the website and have they been met?

  - User requirements:
    - Can the users find what they are looking for?
    - Can the user access all pages of the site?
    - Do all links (internal & external) work?
  - Functional requirements:
    - Do all planned sections of the page appear in the correct place?
    - Does the navigational bar appear?
    - Are all style rules applied?
    - Do all scripts used in the page work?

These are areas that you will already have considered when creating your website but should be revisited for evaluation purposes.

6.6   Usability

A web site can be described as having good usability if it has clear and concise information and it is easy to navigate for a range of users.

We would start by looking at the feedback results from our compatibility testing carried out in the previous section, Compatibility.

- How does the website display on different devices?

  - Smartphone
  - Tablet
  - Laptop / Desktop computer

- How does the website display / function in different web browsers?

The websites can then be evaluated by using the following criteria:

- Are the screens laid out appropriately, with a consistent User Interface?

- Are navigation and features obvious and working correctly?

- Have you catered for your target audience appropriately?

- Can the user find the information or features easily?

Once a website has been completed then it is good practice to try to evaluate the questions above and evaluate the success and appropriateness of the website.

Screen layout

There should be a consistent layout throughout the product. Navigation menus, headings and controls should always appear in the same place. Screens should not be too cluttered with information. Where a large amount if information needs to be presented, it should be possible to hide areas of the information (e.g. by using collapsible headings).

Navigation

Are there enough visual clues to aid navigation? Are navigation links obvious? Are menus and sub-menus organized into sensible categories? Does the interface provide feedback about the navigation? such as "breadcrumbs" or identifying previously visited pages.

Target audience (Personas)

You may remember from National 5 that during the analysis stage a target audience will have been identified. This will give demographics such as the age group of the potential users and their level of technical expertise and would have been used at the design stage to create a number of different personas.

Usability testing will try to evaluate whether the target audience have been catered for, looking at the use of colours, navigational complexity and amount of information on each screen.

Often an early version of the product - or even a mock-up - will be trialled with a small selection of the target audience to ensure the interface is satisfactory. Feedback will be collected and given to the designers to act upon.

Finding information

There is a whole set of theories around identifying how to construct information systems so that users can navigate the product efficiently and find information on the resulting screens. Research involving eye-tracking software has been carried out so that important information (or adverts!) are placed in the optimum position.

Trends in User Interface (UI) design also influence how web sites are built or updated. In the early days of the World Wide Web, every link would consist of underlined text, this is now rarely the case, with colours, shapes and even the position of elements reflecting their intended purpose.

Evaluating usability

A number of techniques will be used to try to evaluate the usability of a product. These may provide quantitative data about the user experience (such as how long it takes to locate a piece of information on a mobile interface vs a desktop one) or qualitative data, such as a narrative of how the user accomplished a task in an unfamiliar product ("I saw the destinations menu on the airport website and figured there may be another link from there to a flight schedule for Alicante. This was

easy to find on the new page.")

6.6.1 Usability tools

There are other tools that can be used to evaluate a websites usability such as:

- Eye tracking software to see what users are looking at on your page. This can create heat maps to show you where users look most on your page;

- Heuristic Evaluation where the site is compared against the principles of usability;

- System Usability Scale (SUS) where the user is asked 10 questions and asked to rate their experience on a scale from 1 to 5 (strongly disagree - strongly agree).
  These scores are added together to give a usability score out of 100 and uses questions such as:

  1. I think that I would like to use this system frequently.
  2. I found the system unnecessarily complex.
  3. I thought the system was easy to use.
  4. I think that I would need the support of a technical person to be able to use this system.
  5. I found the various functions in this system were well integrated.
  6. I thought there was too much inconsistency in this system.
  7. I would imagine that most people would learn to use this system very quickly.
  8. I found the system very cumbersome to use.
  9. I felt very confident using the system.
  10. I needed to learn a lot of things before I could get going with this system.

6.7   Learning points

Summary

You should now be able to:

- understand and perform usability testing making use of:
    - personas;
    - test cases;
    - scenarios based on low fidelity prototypes;

- describe and perform testing on:
    - input validation for data being entered into a form (Client Side Scripting);
    - the site's navigational bar both displays correctly and all links work;
    - all of the media content displays correctly;

- understand and perform compatibility testing to ensure that websites display properly on a variety of different devices:
    - tablet;
    - smart phone;
    - desktop;

- understand the importance and perform compatibility testing to ensure that websites display properly on a variety of different software:
    - operating systems;
    - web browsers.

- evaluate:
    - if your website is fit for purpose;
    - your website's usability.

6.8   End of topic test

End of topic test: Testing   Go online

Q4:   Which of the following is the correct definition of what is included under usability?

a) Screen Layout, Use of HTML5, Style sheets, Help Guide.
b) Ease of Navigation, Screen Layout, Target Audience, Ease of locating information.
c) Robustness, navigational aids, accessibility, use of standard file formats.
d) HTML5, CSS3, JavaScript, PHP

....

Q5: Which of the following can help usability on a website:

a) Designing separate mobile and desktop versions.

b) Using Responsive Web Design methods.

c) Training users properly in school or college.

d) Varying the design of screens frequently.

. . . .

Q6: Which factors may affect the compatibility of an information system?

a) User preferences.
b) Bandwidth, Screen size, Web Server type.
c) Screen Size, RAM, Processor Speed, browser.
d) How useful the user finds the information.

. . . .

Q7: Style sheets help to aid maintainability by making it simpler to edit the style or format of an entire website.

a) True
b) False

. . . .

Q8: What is an advantage of beta testing?

a) Testers are biased
b) Less errors may be found as real world data is being used.
c) Errors that can only be found when running a project may be discovered.

Q9: A website could be considered fit for purpose if it matches:

a) most of the wireframe design and displays most of the content.
b) all the wireframe design and displays all planned content.

. . . .

Q10: When assessing the usability of a website we would evaluate how it works on:

a) multiple web browsers on one type of device.
b) the same web browser on multiple types of devices.
c) multiple web browsers and on multiple devices.

Topic 7

End of unit 4 test

End of unit 4 test   Go online

Q1:   Which image shows a linear structure for an information system?

a)

b)

....
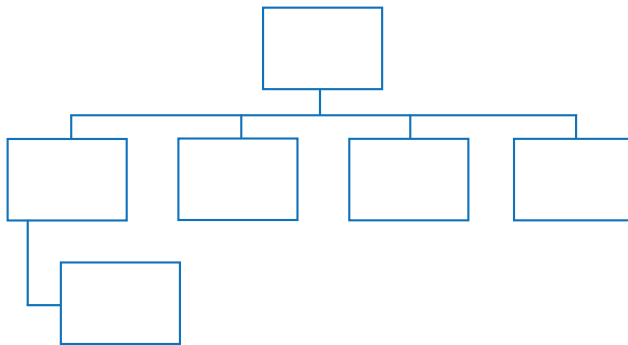
Q2:   Name the necessary tags for the structure of a web page. You may choose more than one option.

   a) $<$title$>$
   b) $<$body$>$
   c) $<$ head$>$
   d) $<$!doctype$>$

....

Q3:   Cascading Style Sheets (CSS) rules consist of a selector and a declaration.   The declaration is made up of:

a)   a list of attributes and the associated values.
b)   a list of different fonts in order of priority.
c)   metadata about the stylesheet.

....

Q4:   Which of the following makes a web page interactive? You may choose more than one option.

   a) Text
   b) Buttons
   c) Videos
   d) Tables

....

Q5: In client-side scripting where are scripts executed?

a) User's browser
b) On the web server

....

Q6: In server-side scripting where are scripts executed?

a) User's browser
b) On the web server

....

Q7: Define target audience.

a) People watching a show.
b) Who the website is aimed at.
c) Who is designing the website.
d) People who create the web design software.

....

Q8: An HTML form is being created that will require a user to enter a value in a text field. Select the attribute of the INPUT element that will force a user to enter a value before the form can be submitted.

a) validated
b) regex
c) required
d) necessary

....

Q9: A pattern defines a rule for the client side validation of data in a form field. Patterns as specified using:

a) regular expressions.
b) Boolean algebra.
c) JavaScript.
d) fuzzy logic.

....

Q10: Which one of the following techniques is used to represent the motivations, goals and frustrations of users so that the development team can understand their needs?

a) Wireframe
b) Use case
c) Persona
d) User scenario

....

Q11: A paper prototype is a:

a) sketch of the user interface which users can comment on.
b) version of the application drawn on paper with which users can interact.
c) pseudocode design.
d) template of buttons, menus, and text boxes that users can discuss.

....

Q12: What HTML is used to create a text input field?

a) <input type="textfield">
b) <textinput type="text">
c) <textfield>
d) <input type="text">

....

Q13: What HTML would we use to create a dropdown list?

a) <input type="list">
b) <list>
c) <select>
d) <input type="dropdown">

....

Q14: Elements with the property `display:block` would be displayed without starting a new line.

a) True
b) False

....

Q15: Which property is used to change the left margin of an element?

a) padding-left
b) margin-left
c) indent

....

Q16: How do you select an element with id "image"?

a) .image
b) image
c) *image
d) #image

....

Q17: How do you select all `h1` elements inside a section element?

a) section.h1
b) section + h1
c) section h1

....

Q18: How do you create a margin for an element with the following values?

The top margin = 20 pixels
The bottom margin = 8 pixels
The left margin = 10 pixels
The right margin = 5 pixel

a) margin:20px 5px 8px 10px;
b) margin:20px 8px 5px 10px;
c) margin:20px 10px 5px 8px;
d) margin:5px 10px 20px 8px;

Glossary

Alpha testing

testing of software within the development organisation which does not necessarily wait until the product is complete.

Benchmark usability tests

are different from conventional usability tests. Participants carry out testing from their home or place of work. Rather than providing feedback to a team member, the user works with an automated system that asks his/her to perform tasks, measures his/her performance and asks for subjective comments.

Beta testing

getting members of the public to check a website or program for errors. Website is checked using real world data to try to find as many errors as possible.

Client-side scripting

the scripts for a website are executed on the client's computer. This allows the code to be viewed and copied.

Compatibility

whether or not an information system will work on a particular device.

Context of use

is the actual conditions under which a given software product/application is used, or will be used in a normal day to day working situation.

CSS

Cascading Style Sheets, a style sheet language used to describe the look and formatting of a document written in a markup language. It is currently on version 3 of the CSS standard (CSS3).

Environment

is the situation in which an application or software product is used/will be used.

Field studies

research into the success of an application or to verify the needs of a target groups which is carried out "in the field" - in real life settings with real users.

Focus group

a group of people assembled to participate in a discussion about a product before it is developed/launched.

High-fidelity prototypes

from a user testing point of view, a high-fidelity prototype is close enough to a final product to be able to examine usability questions in detail and make strong conclusions about how behaviour will relate to use of the final product.

HTML

Hypertext Markup Language, a standardised system for tagging text most commonly used to define the structure and content of a document.

Human computer interaction (HCI)

relates to the design and use of computer technology, focusing particularly on the interfaces between people (users) and computers.

Interviews

a face to face meeting of between the developers and one or more target users.

ISO 9241-210 Human-centred design for interactive systems

ergonomics of human-system interaction, provides guidance on human-system interaction throughout the life cycle of interactive systems. An international standard detailing the development of iterative systems.

JavaScript

an object-oriented computer programming language commonly used to create interactive effects within web browsers.

Low-fidelity prototypes

an early prototype that is sketchy and incomplete, that has some characteristics of the target product but is otherwise simple, usually in order to quickly produce the prototype and test broad concepts.

onclick

This JavaScript function is triggered when a user clicks their mouse button when the mouse pointer is over the selected HTML element.

onmouseout

This JavaScript function detects when the user's mouse pointer moves over a selected HTML element.

onmouseover

This JavaScript function detects when the user's mouse pointer moves away from a selected HTML element.

Paper prototyping

in human-computer interaction, paper prototyping is a widely used method in the user-centered design process, a process that helps developers to create software that meets the user's expectations and needs-in this case, especially for designing and testing user interfaces.

Performance and satisfaction criteria

the criteria used by the client/end-user which detail what the applications/software product must do (and how well it should do it) in order to be accepted.

Product owner

is the member of the team responsible for defining and prioritizing the Product Backlog so as to streamline the execution of program priorities, while maintaining conceptual and technical integrity of the features or components the team is responsible for.

Prototyping session report

a formal report from a session with users working with prototypes of any time.

Screen reader

> software used by users who are blind or have impaired vision which reads the text on a web page or the user interface of a program.

Semantic elements

> Semantics is the study of the meanings of words and phrases in a language. Semantic elements are 'elements with a meaning'. A semantic element clearly describes its meaning to both the browser and the developer.

Server-side scripting

> the code for a website is executed on the server. This does not allow the code to be viewed or copied.

Surveys

> a series of questions used to evaluation or reflect on an application.

Task analysis

> is the process of learning about ordinary users by observing them in action to understand in detail how they perform their tasks and achieve their intended goals.

Tasks

> what the user has to achieve with the applications/software product.

Test methods

> are approved procedures for ensure that an application meets it's requirements.

URL

> Uniform Resource Locator, a reference (an address) to a resource on the Internet. A URL has two main components: Protocol identifier: For the URL http://scholar.hw.ac.uk/ , the protocol identifier is http . Resource name: For the URL http://scholar.hw.ac.uk/ , the resource name is scholar.hw.ac.uk/ - the resource can include the server, port, path and filename of the resource plus any internal anchor and/or data.

Usability

> a subjective assessment of how the user finds their experience of using the information system.

Use cases

> is an approach used in system analysis to identify, clarify, and organize system requirements. Use cases are made up of sets of possible sequences of interactions between systems and users in a particular environment and related to a particular goal.

User

> the end user of an application.

User experience (UX)

> the overall experience of a person using an application, especially in terms of how easy or pleasing it is to use.

User persona

is a representation of the goals and behavior of a hypothesized group of users. In most cases, personas are created from data collected from interviews with users.

User profile

is a simple representation a user. It is an initial step in the creation of a user persona.

User scenarios

describe the stories and context behind why a specific user or user group comes to your site or use your application. They note the goals and questions to be achieved and sometimes define the possibilities of how the user(s) can achieve them on the site.

User stories

are short, simple description of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system.

wireframes

an image or set of images which displays the functional elements of a website or application, typically used for planning interactions and interface elements.

Answers to questions and activities

Topic 1: Analysis

End of topic test: Analysis (page 5)

Q1: c) a user story

Q2: c) interviews and focus groups.

Q3: b) People who the website is aimed at

Q4: d) Date it was created

Topic 2: Design

Quiz: Site structure (page 13)

Q1: In a linear structure all the pages must be accessed in the order that they are positioned in, one after another. In a hierarchical structure the pages can be accessed in a random order.

Q2: A multi-level structure means that there are several levels in a hierarchical structure which can be navigated between. Often, tools such as searching and backtracking will be required to help manage navigation between pages.

Q3: Breadcrumbs are useful as they allow the user to see where the page they are on sits within a multi-level structure, and navigate to a parent page or category easily (even if they haven't visited it previously).

End of topic test: Design (page 28)

Q4: b) wireframe

Q5: a) version of the application drawn on paper with which users can interact.

Topic 3: Implementation: HTML

Quiz: Revision (page 31)

Q1: c) a class.

Q2: c) A DIV grouping element.

Q3: It is a compulsory tag within the head section and it contains a title that will appear on the top right (tab) of web pages.

Q4: Everything that is to appear on the web page.

End of topic test: HTML (page 51)

Q5: b) False

Q6: b) <footer>

Q7: d) required

Q8: a) <nav>

Q9: c) <header>

Topic 4: Implementation: CSS
End of topic test: CSS (page 79)

Q1: a) True

Q2: b) padding:15px 10px 15px 10px;

Q3: a) .active

Q4: b) Separate each selector with a comma

Topic 5: Implementation: JavaScript

Quiz: Scripting (page 86)

Q1: A language that is used to create web pages.

Q2: HyperText Markup Language

Q3: A script is extra code that is added to a web page to improve the interactivity.

Q4: Scripting languages can add interactive elements such as menus, change page contents, validate form data, and (if server-based) access and query databases.

End of topic test: JavaScript (page 96)

Q5: a) function myFunction()

Q6: b) myFunction()

Q7: c) onclick

Q8: a) onmouseclick

Q9: d) Anywhere in the document between the <html> and </html> tags

Topic 6: Testing and evaluation

Quiz: Compatibility (page 110)

Q1: If it uses too much space in memory then it may not be able to run on the user's device, and may take too long to load if they have limited bandwidth.

Q2: Test your website on all different devices and if necessary create different versions for all types of computer.

Q3: They make it simple to maintain consistency. They can be used to quickly adjust elements across the product. Alternative style sheets can be used for accessibility reasons or different media types.

End of topic test: Testing (page 114)

Q4: b) Ease of Navigation, Screen Layout, Target Audience, Ease of locating information.

Q5: a) Designing separate mobile and desktop versions.
b) Using Responsive Web Design methods.

Q6: b) Bandwidth, Screen size, Web Server type.

Q7: a) True

Q8: c) Errors that can only be found when running a project may be discovered.

Q9: b) all the wireframe design and displays all planned content.

Q10: c) multiple web browsers and on multiple devices.

Topic 7: End of unit 4 test

End of unit 4 test (page 118)

Q1: a)

```
┌──────┐   ┌──────┐   ┌──────┐   ┌──────┐   ┌──────┐
│      │───│      │───│      │───│      │───│      │
└──────┘   └──────┘   └──────┘   └──────┘   └──────┘
```

Q2: b) <body>, c) <head> and d) <!doctype>

Q3: a) a list of attributes and the associated values.

Q4: b) Buttons and c) Videos

Q5: a) User's browser

Q6: b) On the web server

Q7: b) Who the website is aimed at.

Q8: c) required

Q9: a) regular expressions.

Q10: c) Persona

Q11: b) version of the application drawn on paper with which users can interact.

Q12: d) <input type="text">

Q13: c) <select>

Q14: b) False

Q15: b) margin-left

Q16: d) #image

Q17: c) section h1

Q18: a) margin:20px 5px 8px 10px;