Authoring for the Web (HTML)

Introduction to HTML and CSS

These exercises were produced by Dr Paul Gooding (University of Glasgow) for the Arts and Humanities in the Digital Age programme (2014-) through the Consortium for Humanities and the Arts South-East England (CHASE). They are adapted here with permission of the author.

Exercise 1: Viewing Existing Websites

Go to https://chasedigitalage.wordpress.com/, right click and select "view page source" – see if you can identify the web page elements that we discussed in the presentation.

- If you're using Firefox, you can also select "Tools", "Web Developer", and select "Inspector".
- If you're using Google Chrome, go to "Tools" then "More Tools" then select "Developer Tools".
- If you're using Microsoft Edge, go to "Tools" then select "Developer Tools".

Exercise 2: HTML

As you can see with WordPress, it is possible to create websites using a Content Management System (CMS) simple WYSIWYG (What You See is What You Get) interface. However, as we're starting from the basic principles of web design we are going to build a webpage using the bare minimum.

- An editor to create HTML files (Brackets)
- A browser to view and test your website (We have recommended Chrome because it plays nicely with Brackets)

You can create an HTML page in any text editor such as NotePad, but an Open Source code editor has many benefits – not least because it recognizes the type of code that you are working with and highlights it for ease of use. If you click on the file **index.html** in the left-hand column, you will see this in practice – although currently you may not recognize many of the HTML elements that you see.

HTML Structure

Our first step is to create a basic structure for the webpage.

```
<!DOCTYPE html>
<html>
      <head>
          <meta charset="utf-8">
          <title>Insert title here</title>
      </head>
      <body>
          <h1>Insert header here</h1>
          <p>Write anything you want to here!</p>
      </body>
</html>
```

<!DOCTYPE HTML> tells your browser that it is reading an HTML file

<html> indicates where the HTML file starts and ends.

<head> contains additional information about the page, including the character set that you are using in **<meta charset="…">.** It also contains the title of the webpage, **<title>**, which displays in the title bar at the top of your browser window.

Everthing inside the **<body**> element is displayed in the main browser window. **<h1>** defines the main heading. Text between **<p>** and **/p>** is a paragraph.

Note that both the **<head>** and **<body>** are "nested" within the HTML file. Other than the Doctype Declaration, every other element should be nested within the HTML element in the manner that you see above.

EXERCISE 3: CREATING YOUR FIRST WEB PAGE IN BRACKETS

In a little while we're going to use Brackets to create a web page. First, though, we need to set up our working space. On your computer, create a new folder entitled **webpage.** Every file that you create in this practical should be saved in this folder! Then open your install of brackets and go through the following steps:

1.) Click on **file** in the top menu bar, followed by **new**. This will open a blank window in Brackets. If you were to type anything into this window then it would not be recognized as an HTML file and would therefore not be marked up accordingly.
2.) To allow your computer to recognize this as HTML, return to the **file** menu and click on **save as.** A pop-up window will open, so navigate to your folder and name your file **index.html**. This is essential, as it flags this up as your website's home page.

Great! Now Brackets should recognize that this is an HTML file, which means we can start creating the web page.

We need to insert the following tags to create a functional web page:

```
<! DOCTYPE html>
<html>
      <head>
          <meta charset="utf-8">
      </head>
      <body>
      </body>
</html>
```

Type these in as you see above, and then right click on the file name and click **save.** Well done – if you have done this stage correctly, you now have the most basic webpage structure possible. Now do the next two things:

1.) On the line after the characterset element, insert the following: **<title>My First Webpage</title>.**
2.) Inside the body element, insert the following, without a tag around it: **Hello World**

One thing you will notice is that Brackets automatically closes tags for you. As you close an HTML element, it immediately places the closing tag afterwards – this is very useful, but it does mean that you have to pay attention to ensure that elements are nested in the right order.

Right now, your webpage is basic, but we can at least see how it looks in a web browser. If you have Chrome installed on your computer, you can use Brackets' **Live Preview** to view your webpage as you make it. Right click on the file name and click **save**, then click on the lightning bolt at the top right of the **Brackets** window. Chrome should load and display your webpage. It hopefully works, but it's somewhat… basic currently. From now on we're going to add a bit more to it.

1.) Start by creating a heading **<h1></h1>** around the word **Hello!** Click **save** and look at the results in Chrome. What is the difference between words contained within a Header tag, and those which are not?

Now insert the following text into your webpage, between the **<body>** tags:

```
<h1>Hello World!</h1>

<p>This is a nice basic web page for us to use as an example. You
can't break it, so follow the exercises on the handouts, experiment
with a few of the tags introduced in Paul's presentation and, if you
have time, go to http://www.w3schools.com/tags/ - The W3 Schools HTML
Reference page - and see what some of the more advanced tags do.</p>

<p>By adding HTML to a simple string of text, we can create a list
with a few entries in. Follow the instructions to see how we do
this</p>

<h2>Shopping list</h2>



Tabasco Sauce

Pop Tarts

Maltesers

Orange Juice

Painkillers...
```

Click **SAVE** and observe the results.

2.) We can turn the W3 Schools URL into a working link by inserting an **<a>** tag so it looks like this:
<a href="http://www.w3schools.com/tags/">W3 Schools</a>. Click **SAVE** and then click the link in Chrome to see whether it works.

3.) Create an Unordered list from the shopping list contents as follows:

```
<ul>
<li>Tobasco Sauce</li>
<li>Pop Tarts</li>
<li>Maltesers</li>
 <li>Orange Juice</li>
 <li>Painkillers...</li>
```

```
</ul>
```

Click **SAVE** and observe the results in Chrome. Now change the **<ul><ul>** tags to **<ol></ol>** for Ordered List and click **RUN** again. What is the difference between the two types of list?

4.) Next, try to add some emphasis to the text. Add **<em></em>** around the words **You can't break it,** then click save and observe the results in Chrome. Then change the tag to <strong></strong> and observe the changes.

5.) Finally, we're going to try adding a picture to your webpage. Choose an image you're your computer and save it as a JPEG entitled **image.jpg** (this may vary depending on your computer, but ask if you're not sure how!). Make sure that the image is saved in the same folder as your HTML file. Directly after the second paragraph, and before the **<h2>** element, add the following:

```
<img src="image.jpg" alt="…"
style="width:304px;height:228px;float:right;">
```

Replace the ellipsis with a description of your own image. Then save the file and check out what your webpage looks like in Chrome.

Well done! You've now created a basic website with a working link, embedded image, text emphasis, and an ordered list. If you follow the basic rules that we've learned here, you can now be confident in playing with other elements to see what they might add to your website. Once you start creating a website in WordPress, you will also be able to use basic HTML to edit it, rather than relying entirely on the CMS to do all the work for you – this will be a useful skill!

Exercise 4: Viewing Advanced CSS

Take some time to explore www.csszengarden.com. All the different versions of this website are created from an identical HTML file: the only difference is the stylesheet that has been applied. Look at a few of the alternative layouts and think about them from a technical viewpoint to start with – view the page source and CSS files to see which elements you recognize, and how each version differs. Then take some time to think about the design decisions that have been made: what do you think works well, or badly, for each version, and what would you do differently if you were designing a new layout for this website?

Exercise 5: Creating CSS

If you get the time, it's worth trying to create a basic CSS file for your webpage. To do so, create a new text file in Brackets, and save it as **style.css**. Make sure that it is saved in the same folder on your computer as your index file. We'll start working on the CSS file in just a second, but first click on your HTML file **index.html** and add the following element which must be nested within the **<head>** element:

<link rel="stylesheet" type="text/css" href="style.css">. Make sure that your file name matches the link in your HTML, otherwise it won't work!

We're first going to make a couple of changes to your HTML file. Edit only the section within the body so that it reads as follows:

```
<section id="intro">

     <h1>Hello!</h1>
```

```
    <p>This is a nice basic web page for us to use as an example. <strong>You can't break it</strong>, so follow the exercises on the handouts, experiment with a few of the tags introduced in Paul's presentation and, if you have time, go to <a href="http://www.w3schools.com/tags/">W3Schools</a> - The W3 Schools HTML Reference page - and see what some of the more advanced tags do.</p>

      <p>By adding HTML to a simple string of text, we can create a list with a few entries in. Follow the instructions to see how we do this</p>
</section>

<section id="ShoppingList">

     <h2>Shopping list</h2>
         <ul class="indigestion">
             <li>Tabasco Sauce</li>
             <li>Pop Tarts</li>
             <li>Maltese’s</li>
             <li>Orange Juice</li>
             <li>Painkillers...</li>
          </ul>
</section>
```

1.) Now open your CSS file. Start by changing the font of your paragraphs:

```
p {
      font-family: Helvetica, Verdana, Sans-Serif;
}
```

Now observe the result in Chrome.

2.) Now we’ll look at sections. Input the following:

```
#intro {
      background-color: blue;
      }
#ShoppingList {
      background-color: red;
      }
```

Now observe the results in Chrome.

3.) Now to make the strong text really stand out. Insert the following:

```
strong {
      color: pink;
      }
```

Look at Chrome observe what is now looking like an almighty mess…

4.) Let’s finish the job by defining the appearance of the indigestion class with the following code (note the full stop at the start!):

```
.indigestion {
      border-style: double;
      border-color: black;
      border-width: 4px;
      float: left;
      width:40%;
      }
```

5.) By any standard, what you've created looks awful. If you have time, then use what you've learnt, and some of the extra tags from W3 Schools and this handout, and try to make the website look presentable.

Exercise 6: Accessibility

First, we're going to see how people with disabilities experience aspects of the Web. Individuals with blindness or other sight problems might use a screen reader to translate web pages into audible form. Windows 10 and Mac OSX both have in-built screen readers, and we're going to have a go at using the web in this way.

1.) PC – press the **Windows Logo key** + **Enter**.
2.) Max – press **CMD + F5.**
3.) Now go to www.bbc.co.uk. We're going to try to access today's weather report using nothing but the keyboard and voiceover function. You can navigate from link to link by pressing **TAB** and select a link by pressing **RETURN.** Keep going until you can find the weather report for today in Norwich, relying only on the screenreader for prompts.
    a. Make a note of the elements which the screenreader focuses on first – why would it concentrate on presenting websites in this manner?
    b. In pairs, discuss your experience of using a screenreader – what are the challenges for users of this software for web browsing?
4.) Now go to http://wave.webaim.org/ and type www.bbc.co.uk into the search box. Wave is a free tool for assessing the accessibility of websites. If you're unfamiliar with web design, then the results may seem quite intimidating, but you can see the importance of a number of HTML elements for assisting with accessibility. Click on the Flag icon in the left hand menu to see the list of elements which relate to accessibility:
    a. Which attribute groups can you recognize?
    b. It's worth going back to this after learning more about coding in HTML and CSS, as the elements will start making more sense!
5.) Now for readability: navigate to http://juicystudio.com/services/readability.php and input the BBC URL in the search box. You will see the website's score broken down into various readability scales. How does the BBC website do? Feel free to try other websites.
6.) There is also a website which allows you to view websites as seen by individuals with various types of colour blindness. This can be found at http://www.vischeck.com/, but unfortunately the website is down at the moment. It's worth looking at in future, though, as it's an extremely useful tool to help ensure your colour schemes and images are appropriate for the widest possible audience.
7.) For more insights into how people with disabilities use the Web, then check out the following link: https://www.w3.org/WAI/intro/people-use-web/Overview.html.

Some Follow-Up Work:

Understanding Domain Names

Go to **https://whois.net/** and search for some sites – this will show you what sort of information is available about domain owners.

www.bl.uk

www.academia.edu

www.uea.ac.uk

www.chase.ac.uk

History of the Web

If you're interested in learning more about the history of the Internet, and the growth of the Web, take a look at some of the following links:

http://www.nethistory.info/

http://webhistory.org/

http://www.w3.org/history.html

HTML Cheat Sheet

This page lists some basic HTML elements. These are tags which mark certain areas of your page such as links, paragraphs, headings or lists.

Elements are normally opened using a tag at their start **<p>**, and a closing tag at their end **</p>**: the forward slash allows the browser to differentiate between opening and closing tags. Some elements are self-closing, so they have the forward slash inside **<img />.**

Elements can have attributes, which are written as **attribute="value". This** is used for link tags **href=""**, and for labeling elements to allow styles to be applied (eg. **id=""** and **class=""**).

The following table contains a few basic tags, but for a more complete list see the following links:

http://www.w3schools.com/tags/

http://www.webmonkey.com/2010/02/html_cheatsheet/

Element	Description
`<html></html>`	Creates an HTML document.
`<head></head>`	Indicates information that won't be displayed on the web page, such as language, title and metadata.
`<body></body>`	Defines the **body**, or visible section of the document.
`<h1></h1>` to `<h6></h6>`	Defines **headings** from H1 to H6.
`<p></p>`	Defines a **paragraph**.
`<br />`	Adds a **line break** within a paragraph. This allows sections of text to be split into smaller portions while maintaining a consistent style.
`<ul></ul>`	Defines an **unordered list**, with items marked with bullet points.
`<ol></ol>`	Defines an **ordered list**, with items number sequentially unless other markers are specifically defined.
`<a href="URL"></a>`	Defines an **anchor** or hyperlink, commonly to an internal or external webpage.
`<img src="SRC" />`	Defines an **image** at the location specificed in the SRC attribute
`<em></em>`	Defines text which should be given **emphasis** (italics by default in most browsers)
`<strong></strong>`	Defines **important** text (bold by default)

`<section></section>`	Defines a section in a document

CSS Cheat Sheet

CSS (Cascading Style Sheets) is a language used for describing the look and formatting of an HTML document. CSS files are made up of **selectors** and **declarations**. **Selectors** define an HTML element within the page, and **Declarations** define the rules to apply to those elements. So for instance, if you want to colour text in headings blue, you would write the following:

H1 {color: blue;}

Webmonkey provides an excellent introduction to CSS, and below is a table with some of the most common selectors included:

http://www.webmonkey.com/2010/02/css-guide/

Common selectors:

`h1`	Selects an HTML element. This can be any standard HTML tag., and is written as the tag is displayed without triangular brackets.
`#id`	Selects an HTML element with the specified ID, set in the HTML document using id="[id]". ID selectors are unique, and can therefore only be used once in each document.
`.class`	Selects any HTML element with the **class="[class]"** attribute. Note the full stop at the start. The class selector can be used multiple times in each document.

Declarations are formed of pairs in the format **property:value.** Property defines the element you want to style, while value specifies the style. Declarations are separated by a semicolon after the value. Some common declarations:

Property	**Description**
`font-size:`	Defines the size of text. Can be specified in a range of units.
`font-family:`	Changes the font used. Browsers do not have every font installed, so you can provide a comma-separated list of fonts which the browser will try in order. It's best to select your preferred font, and then move towards a general font family as the last in the last (**serif** or **sans-serif**).
`color:`	Sets the foreground colour of the specified element. For text, this will define the text colour. Colours can be specified either by their name or using a # followed by a combination of six letters and numbers: these are known as hexadecimal, and a colour picker is available at http://www.w3schools.com/tags/ref_colorpicker.asp.
`background-color:`	sets the background colour of the specific element.
`border-width:`	Sets the width of the specified element's border.
`border-style:`	Selects the border line for the specified element.
`padding:`	Sets the spacing between an element's borders and its contents.

`margin:`	Sets the properties for a margin in a given selector.
`text-align:`	Specifies the horizontal alignment of text.

ACCESSIBILITY & VALIDATION

Web Accessibility is about ensuring that people with a range of disabilities can view, navigate and interact with the web, and encompasses all disabilities that may affect access to the Web including visual, auditory, physical, speech, cognitive and neurological disabilities. It ensures that websites work properly when used alongside accessibility tools such as screen readers, and can encompass everything from making it easy to resize text to be more readable, through ensuring that colour schemes are visible to those with colour blindness, to utilizing well-formed HTML which works properly with screen readers.

There are plenty of resources out there which will help you to understand the concepts which underpin designing for web accessibility.

W3C provides an introduction to web accessibility at
http://www.w3.org/WAI/intro/accessibility.php.

To check whether your website is suitable for users with common forms of colour blindness, try www.vischeck.com.

The WAVE Web Accessibility Tool can be used to evaluate the accessibility of any website. It uses a visual interface to flag up problems clearly, and ranks them according to severity. Check it out at http://wave.webaim.org.

It's also worth trying a screen reader to see what it's like to use the web in this way: you can download a free trial of JAWS, http://www.freedomscientific.com/Products/Blindness/JAWS or your university may have similar software available to try.

Markup Validation is a way of testing your web code to ensure that it is 'valid': i.e., that it meets the W3 standards. Web documents are written using markup languages, and markup validation is an automatic process of checking your own code against the technical specifications of the chosen language. It will tell you whether your code is well formed, identify broken links, and generally make sure that your website meets the appropriate web standards. It won't necessarily solve all problems, but it will improve the quality of your site and make it more likely to meet requirements for accessibility.

There are a wide variety of validator tools, but here are a few to get you going (note: you may not understand all the results at this point, but they will make more sense as you increase your knowledge of web design!):

W3C HTML Validator: http://validator.w3.org/

W3C CSS Validator: http://jigsaw.w3.org/css-validator/

W3C Link checker: http://validator.w3.org/checklink