RWD Week Two

Starting your portfolio page

You are going to develop a single page portfolio. This one page introduces you to the basics of RWD (responsive web design), mobile first web development and further elements of page layout, typography, web design and coding.

Task 1 - Getting started

- Create a folder called `term_two_project` to contain this project.
- Get the template portfolio from github - Select the green *Clone or download* button, select *Download ZIP*. Locate the ZIP file and extract the contents.
- You only need the `portfolio` folder - move this into your `term_two_project` folder.
- Using *Visual Studio Code* - open folder to define on the `portfolio` folder. Open `index.html` and `layout.css`.

VIDEO -

- Test `index.html` in a browser.
- Check `layout.css` is working by adding the line:

```
* { border:1px solid red; }
```

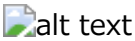
- Everything should get a red border. Yes? Then you can delete this line.

Working *mobile first*

Keep your browser window small (mobile size - no more than half) - don't develop your site with your browser set to fill the screen - we are developing *mobile first*.

Task 2 - The structure - review of HTML template

We will provide an overview of the semantic structure and how this relates to the visual design of the template in class.



Download full PDF

- **Technical Solutions Apprentice students** - Page Layout & code structure explained.
- **Digital Media & Communications students** - Page Layout & code structure explained.

Task 3 - Adding your content

Add your content (created over the last week) to the HTML file in the About and Work sections.

Tag up your content, remembering to use headings, paragraphs, links, and lists as appropriate.

- **About me** – engaging text to attract and inform the user. 150 words maximum plus relevant links.
- **Work experience** – details of your current and previous work experience or change this section to **Interests** in relation to your course. 200 words maximum.

Before you go further

- Validate your HTML additions. Fix any problems.

Task 4 - The CSS re-set

Making your CSS *sensible*.

Our template includes `layout.css`. We have included a basic re-set (to remove some of the browsers default settings) and comments to guide your initial styling.

The following is a breakdown of each of the re-set styles applied to `layout.css`. Please read through and ask questions if you don't understand something.

```
body {
    margin: 0;
    padding: 0;
    font-family: sans-serif;
```

Update the font-family with a Google Font, web safe font and generic fall back.

```
    font-size: 100%;
```

Remember that the default browser font size is 16px.

```
    line-height: 1 ;
```

update to suit your type choices

```
    color: #000;
```

```
/* base text colour - update to suit your design */
```

```
}
```

Default element re-sets (headings, paragraphs and lists).

```
h1,
h2,
h3,
p,
ol,
ul {
    margin: 0;
    padding: 0;
    font-weight: normal;
}
```

The final re-set provided improves default font styles for forms. By default form input and text areas do not inherit the font-family (like all other elements) applied to the body.

```
input,
textarea {
    font-family: sans-serif;
}
```

As with the body you will need to update the font-family with a Google Font, web safe font and generic fall back.

Save any changes to `layout.css`.

Task 5 - Two further CSS re-sets

We would like you to now add two more essential CSS re-sets to `layout.css`.

```
/* box sizing */
```

Find the above comment in `layout.css` and add the border box model CSS code from this article. Copy & paste the code, and save.

You will remember that by default browsers define a box size as `width or height + padding + border` = actual width or height. Applying `box-sizing: border-box;` means the width or height

includes any padding and/or border. This tends to make life easier - you want a 200px wide box, that's what you now get. For a more detailed explanation see Box Sizing explained on CSS Tricks.

At the moment this won't make any difference to you page. This comes later. Now add our final re-set.

```
/* flexible content images code */
```

Looking at `index.html` in your browser you will have noticed that the images we have provided do not fit (*when your browser window is mobile sized*) and cause an ugly horizontal scroll bar.

To fix this problem add this flexible image CSS code from Derren's CodePen to `layout.css`.

Save the changes to `layout.css` and reload `index.html` in your browser. You will now see that the dummy images we have provided size themselves to the browser window. Whilst this is not true *responsive images* - that would serve differnt size images to different size screens, it does provide a simple solution.

Remember as you further develop your CSS to add comments as you go to your code *about* what you've added. The syntax for CSS comments:

```
/* your comment */
```

Task 6 - Selecting & adding Google Fonts

Web Fonts offers a much wider array of options beyond the basic web safe font families. It is now possible create or buy Web Fonts to enhance the design of your site. Google Fonts provides a free and easy to use option.

There are some drawbacks. Adding Web Fonts to your page increases the overall download time and *weight* of the page. Fonts can be costly. Free doesn't always mean good.

We would you to select either a single Google Font for both your headings and text, or a pair of complimentary fonts (one for headings, one for the text).

Read the MDN Web Fonts article for much more detail if you would like a deeper understanding.

Selecting and adding your Google Font(s)

- Identify a single font (or font pairing) on Google Fonts: it could be for your headings, for your *body copy*, or for both.

Help in selecting your fonts

- Layouts designed with google fonts
- The best google fonts according to Typewolf

Apply your selected Google Font(s)

Once you have chosen your font(s), they need to be added to your HTML and CSS.

- Add the Google Font code for your *body copy* to the `<head>` of your `index.html` file, **before the link to** `layout.css`. Repeat this if you have selected a second font.
- Add the font-family CSS rule from Google Fonts to your body rule in `layout.css`.
- By default Google provides a family of the font you have selected and a generic fallback. Insert between these a suitable web safe font.
- Base font (body) should read - Google font, web safe font, fallback.

Save `layout.css` and view the changes to `index.html` in a browser.

VIDEO - Demonstration of selecting and implementing Google Fonts.

```
/* element styling (e.g. standard p, h1, h2, a, etc) */
```

Find the above comment in `layout.css` and add styles to headings and paragraphs.

- If you have selected a second Google Font for headings you will need to apply this (usually to all headings).

```
h1, h2, h3 {
font-family: Google font, web safe font, generic fallback;
}
```

- Add a font-size in rems. 1 rem will be 16px.

```
h1 {
  font-size: 2rem;
}

h2 {
  font-size: 1.5rem;
}

p {
  font-size: 1rem;
}
```

Experiment to see what works best for your chosen fonts. Are the paragraphs legible? Are the headings suitably big enough?

- Next adjust the line-height in your body rule. Adjust to improve the legibility of your text.

- Save `layout.css` and check it's working by viewing `index.html` in your browser. Does it look right? **Adjust it until it does**.
- Style your header `<h1>` and section `<h2>` headings by changing one or two things about them: font-weight, font-size, color, text-transform: uppercase, letter-spacing etc.

As the project develops you will adjust and enhance these settings. You may spot a *better* font. You will improve your writing and you will undoubtedly add more sophisticated styles.

Task 7 - Selecting a colour scheme

On our illustration of the layout we have alternate grey and white background colours for each section, plus a black background for both the navigation and footer. We would like you to select an appropriate colour scheme. This would include colours for:

- Body text
- Headings
- Navigation (background)
- Sections (background) - two complimentary/contrasting colours
- Footer (background)
- Link colours (all states)

Research a colour scheme - try coolors.co.

Choose one primary colour and use tints and shades of that colour for the backgrounds. You may also want to select an accent colour.

Colour contrast - check that that each colour/tint has a strong contrast with your text colour. Test each section colour by using Lea Verou's Contrast ratio checker. You are looking for a AAA (green) pass for each section.

We will apply the colours next week.

If you want to learn more read Building Your Color Palette. This will help in selecting your pallete, including selecting primary and accent colours.

Colour resources

- Coolors.co - a really good colour palette chooser.
- Adobe colour wheel - you may also want to try out this colour tool which helps you come up with colour palettes for your site.
- CSS for Backgrounds - a reference site for all the ways we can control backgrounds in CSS.
- WebAIM Colour Contrast Checker and WebAIM Link Contrast Checker.

Task 8 - The Style Guide

In your pages folder you will find `style-guide.html`. This is a standalone style guide for your site.

Edd Sowden writing for GOV.UK says...

> When trying to develop a site with a consistent look and feel it's common to develop a style guide with patterns which can be reused across the site. This lets designers easily reuse standard patterns and lets developers know how to make things look.

Edit `style-guide.html` to:

- Add the hex codes for your chosen colours to the swatches.
- Add your Google Font(s) to the page.

See prompts and comments on the page to guide you.

Save `style-guide.html` and view in the browser. Test and review your font and colour choices.

Testing

At the end of this session validate your HTML and validate your CSS. **Fix any problems**.

Week 1 Checklist

Are you up to date?

Before you start Week 2 - Have you completed everything?

- Download the template files from GitHub
- Define the site in Visual Studio Code and view the files (`index.html`, `layout.css` and `style-guide.html`)
- Working mobile first - resize your browser
- Review the structure - understand how the design is reflected in the semantic markup.
- View the full (relevant) layout PDF.
- Add your content (tag it up).
- Validate to check for errors.
- Review the CSS re-sets we have provided - ask if you don't understand.
- Add the box sizing and flexible image re-sets.
- Select and add your Google fonts.
- Select a colour scheme ready for implementing next week.
- Edit `style-guide.html` to add your colours and font(s).

And finally:

- Fix any problems before you continue.
- Keep your browser window small - don't use it at full screen yet. We are still *mobile first*.

End