CAB230 - Minty Yard HTML/CSS workshop

NOTE: This is an *OLD* workshop for people to learn HTML and CSS. We now use much the same example, but we now work with React components instead. This Worksheet is *NOT* something you have to do, but we include it because of the explanations it gives. We just leave it here for those who want to look at it.

The material below is unchanged from earlier years. We do not supply the `starter-files` any more - just the solution files.

The Old Worksheet

This workshop will illustrate how Cascading Style Sheets (CSS) can be used to style HTML pages. Our goal is to design a website for a newly opened fictional restaurant, Minty Yard.

To begin, download and unzip the `starter-files` folder on Blackboard. This folder contains a HTML file, CSS stylesheet and a handful of images.

HTML

Open the HTML file `index.html` in your favourite code editor. The HTML code for this workshop has been written for you. Ensure you understand the provided HTML. You may need to refer to a resource such as W3Schools to refresh your knowledge.

Now open the HTML file in your browser. Currently, no styles are applied to this HTML page. Any styling that you can see is a browser default.

CSS

Open the CSS file `style.css` in your favourite code editor. This file is contained in the css folder. A few basic styles have been provided for you. We will build upon these styles in the remainder of this workshop.

Using the CSS provided in this workshop document will inevitably produce the desired styled webpage. However, blindly copying and pasting the code is not a particularly productive learning exercise. When writing CSS, ensure that you understand what each style is trying to achieve. If you are unsure: try viewing the page with the style removed, then add the style back in and observe what has changed. If in doubt, ask your tutor.

Linking an external stylesheet

To display CSS on a HTML page, a reference must be included using a `<link>` element. Add the following `<link>` element inside the `<head>` section in the HTML file:

```
<link rel="stylesheet" href="css/style.css" />
```

Save the HTML file and reload the page in the browser. The HTML page should now display the styles provided in the stylesheet.

Each time you modify the CSS file, you will need to refresh your browser to see the updated styles.

CSS reset

To deal with styling inconsistencies across browsers, a CSS reset is often applied. There are a number of open source CSS resets available, such as Normalize.css. For the purposes of this workshop we have provided you with some basic CSS reset styling. You can find these styles at the top of the `style.css` file.

```
/* This code has been included in styles.css */
/* simple browser reset */
* {
  box-sizing: border-box;
}

html, body {
  margin: 0;
  padding: 0;
  line-height: 1.5;
  font-size: 16px;
}
```

Styling the body & header

We will begin by setting a background colour and font for the body of the webpage:

```
body {
  background: #fff;
  font-family: Arial, Helvetica, sans-serif;
}
```

We will also style the header by adding a green background colour, setting a white font colour and adding a small amount of vertical and horizontal padding.

```
header {
  background: #2e8b57;
  color: #fff;
  padding: 10px 30px;
}
```

Styling the footer

We can also quickly style the footer at the bottom of the page. We will set a dark background color, add some padding and align the text to the center.

```
footer {
  background: #222;
  padding: 40px;
  text-align: center;
  color: #d8d8d8;
}
```

Styling the navigation bar

The header contains both the icon and navigation links. The styles for the icon have been provided for you.

Currently the three navigation links are aligned on the left hand side of the page: Menu, Book & About. Our goal for this section is to align the items horizontally and position them on the far right side of the navigation bar.

Current style: 

Desired style: 

In the HTML, the navigation items are contained within an unordered list, which is itself contained within a `nav` section. By default, `<li>` elements are block elements. To display the items all on one line, we need to change the `display` property of the list items to inline.

We will do this by targeting the li elements within the nav class and unordered list. Add the following style to your stylesheet:

```
nav ul li {
  display: inline-block;
}
```

Refresh the browser. The list items should now be horizontally positioned.

To move the list to the right hand side of the screen, we will style the `text-align` property of the nav element to the right, change the `display` property to an inline block and provide the element with some width.

```
nav {
  width: 50%;
  display: inline-block;
  text-align: right;
}
```

We will style the unordered list within the navigation element by removing the default list style and margin.

```
nav ul {
  list-style: none;
  margin: 0;
}
```

For each link within the unordered list, we will add some additional padding, remove the default text decoration for links, and also change the text colour to white. The `border-radius` property is used to style the radius of the element's corners.

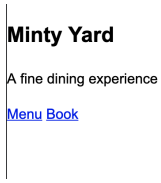
```
nav ul li a {
  color: #fff;
  text-decoration: none;
  display: block;
  padding: 10px 15px;
  border-radius: 3px;
  transition: 0.3s ease all;
}

nav ul li a:hover {
  background: #444;
}
```

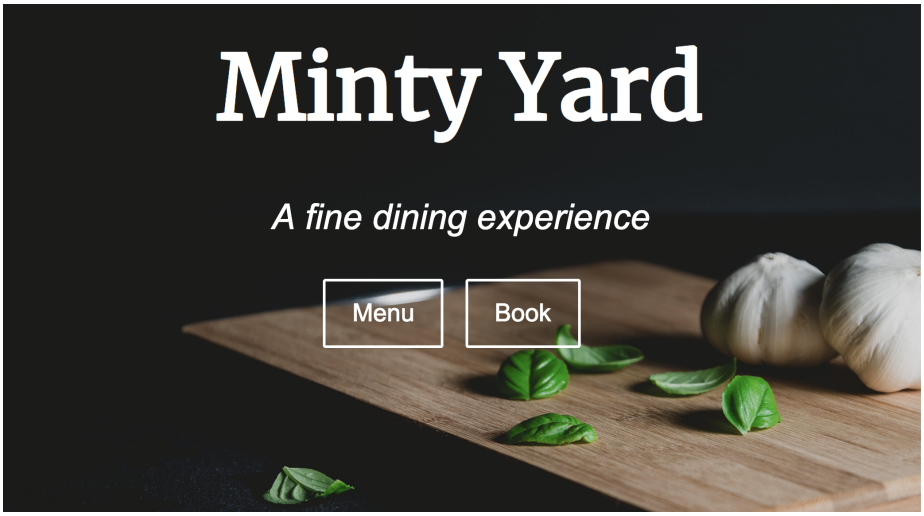
When you now hover over the links, the background should change to a darker colour. This is done by using the `:hover` selector. We have also applied a `transition` to the link of 0.3 seconds. The `transition` property allows us to change an element's values smoothly over a given duration. Try removing the transition to see the difference it makes.

Styling the main content

Current style:



Desired style:



In web design, a hero image is a large web banner image, generally positioned on the front and center of the webpage. As in our case, the hero often consists of a large background image with accompanying text.

The title and subtitle on our page are currently lacklustre. Let's start by adding some text styles to the hero classes:

```
.hero__title {
  font-size: 100px;
  margin: 0;
  font-family: "Merriweather", serif;
}

.hero__subtitle {
  font-size: 40px;
  font-style: italic;
}
```

We also want to set a background image for the hero class and align all of the text to the center of the page. The background image has been provided for you in the img folder. The `background-size: cover` property resizes the background image to cover the entire hero container, even if that requires stretching the

image.

```
.hero {
  min-height: 800px;
  text-align: center;
  background-image: url("../img/dining.jpg");
  background-size: cover;
}
```

We will now add some styles to the two links - menu and booking. Although these are links we will attempt to style them to look like buttons using the `border` property. We will first target the `hero__content` class and add a text colour and padding:

```
.hero__content {
  color: #ffffff;
  padding-top: 25px;
}
```

Then we will target the links within the `hero__content` class. We will style the links, focusing on the `border` and `border-radius` properties to give them a button-like design. We will also remove the default link text decoration.

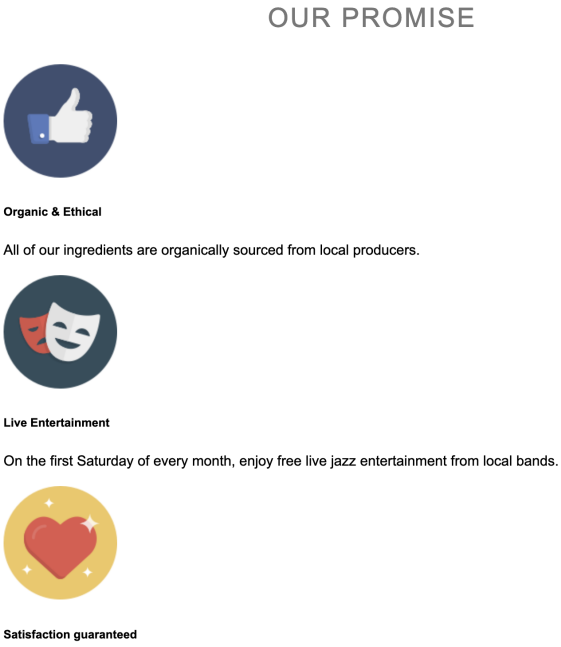
```
.hero__content a {
  display: inline-block;
  color: #fff;
  border: 3px solid #fff;
  border-radius: 3px;
  padding: 15px 30px;
  margin-right: 20px;
  text-decoration: none;
  font-size: 28px;
  transition: transform 1s;
}

.hero__content a:hover {
  -webkit-transform: scale(1.2);
  -ms-transform: scale(1.2);
  transform: scale(1.2);
}
```

A transform/transition effect has also been applied. This is similar to our styling of the navigation bar, except this time we are changing the scale (i.e. size) of the link rather than its background colour.

Styling the feature boxes

Current style:



Desired style:

The feature boxes are the last part of our website that need styling. Currently the boxes are stacked vertically. For our design we want them to be positioned horizontally, and centered in the middle of the screen.

To achieve the desired layout for this section we will use Flexbox, a relatively new CSS module for layouts. Before completing this section, first have a look at the following links. The second link is an online game for learning how Flexbox works:

- A Complete Guide to Flexbox
- Flexbox Froggy

First we will define a flexbox container. We will use the existing `.features__box-wrapper` class and change its display property to `flex`.

We will also `justify-content` and `align-items` to the center.

```
.features__box-wrapper {
  display: flex;
  justify-content: center;
  align-items: center;
}
```

Our feature boxes are now horizontally positioned, but still aligned to the left. We can change this by selecting each feature box and changing the `text-align` property. We will also add a small left margin to each box and set the flex value to 1.

```
.features__box {
  text-align: center;
  flex: 1;
  margin-left: 5px;
}
```

Finally, we will add some styles to the small images and titles:

```
.features__box img {
  margin-bottom: 12px;
  width: 80px;
}

.features__box h5 {
  font-size: 20px;
  color: #999;
  text-transform: uppercase;
  font-weight: 300;
  letter-spacing: 2px;
  margin: 0;
}
```

Media queries

The `@media` CSS rule can be used to check for things such as the width and height of the screen that is displaying the website. This makes it a popular tool for specifying styles that are only displayed depending on the size of the device. You may have noticed that when viewing some websites on your phone, the design of the site is different from when it is viewed on a laptop or desktop. Media queries help facilitate responsive design.

For our website, we will add a simple media query so that the columns of our features section collapse nicely to a singular vertical column when viewed on a small width device (<=700px).

```css
/* Responsive columns */
@media (max-width: 700px) {
  .features__box-wrapper {
    flex-wrap: wrap;
  }

  .features__box {
    flex: 1 100%;
  }
}
```

You can test how this CSS works by reducing the width of your browser window.

Exploration

Browser Developer Tools

All modern browsers ship with developer tools that allow web designers and developers to find out more information about the currently loaded HTML, CSS and JavaScript on a webpage.

Both Google and Mozilla have well written guides on how to use their developer tools. Work through the guide for your favourite browser:

- Chrome
- Firefox

Implementing your own design

One of the best ways to learn CSS is to experiment with your own styles. For this task, copy the existing restaurant webpage and turn it into something new. If you're stuck for ideas, try making a personal portfolio or a website for a sporting/hobby club.

We recommend that you first work on modifying the content of the website using HTML. Then try modifying the style with CSS. Feel free to share your new design in the CAB230 slack group.

Additional CSS resources

- HTML5 Boilerplate
- Bootstrap
- List of CSS Frameworks
- Google Fonts