CS4116 Week 3 Lab Guide

Software Development Project

Teaching Assistant Guide

Prepared by: Zia Ur Rehman
Course: CS4116 - Software Development Project
Professor: Conor Ryan

University of Limerick
2026

Contents

Software Development Project
Teaching Assistant Guide

1 Table of Contents

2 Introduction and Overview

Welcome to Week 3 Lab! This week builds directly on what you learned in Week 2. You'll now move beyond basic HTML structure and learn how to **style** your web pages using **CSS (Cascading Style Sheets)**.

2.1 What You'll Learn This Week

- **CSS fundamentals**: How to control the appearance of HTML elements
- **Three methods of applying CSS**: Inline styles, internal stylesheets, and external stylesheets
- **CSS selectors and properties**: How to target elements and change their appearance
- **Page layout with DIVs**: How to structure complex page layouts using boxes
- **Class attributes**: How to apply styles to specific groups of elements
- **Responsive design basics**: Using relative and absolute sizing

2.2 Lab Tasks Overview

This week's lab consists of several interconnected tasks:

1. **Tasks 2-5**: Create three versions of the same page with different styling methods
2. **Tasks 7-9**: Build a card-based layout page (”My Favorite App”)
3. **Task 10**: Create a portfolio page for ”Jane Doette” with complex layout
4. **Task 11**: Create an index page linking to all your HTML pages

3 Important Reminders

3.1 Mid-term Exam Preparation

Mid-term exam reminder

Remember: The mid-term in **Week 9** will include **coding questions without AI access**. Everything you learn in these labs will be tested, so:

- Take detailed notes
- Practice typing code yourself (don't just copy-paste)
- Understand WHY code works, not just WHAT it does
- Review lecture videos multiple times

3.2 Development Environment

Set up your tools

Make sure you have:

- **VS Code** installed with **Live Server** extension
- **Proper folder structure**: Create a `week03` folder for this week's work
- **Browser Developer Tools** open while working (F12 in Chrome/Firefox)

4 What is CSS?

4.1 CSS = Cascading Style Sheets

Key idea

CSS controls the APPEARANCE of HTML elements.

- **HTML** = Structure and content (“What is this?”)
- **CSS** = Presentation and styling (“How should it look?”)

4.2 Why Separate Content from Style?

Problem without CSS

Inline styles mix content and presentation, and make global changes very hard.

Listing 1: Inline styles (not recommended)

```
<h1 style="color:blue; background-color:black; font-size:24px; padding
    :10px;">Title</h1>

<h2 style="color:red; background-color:yellow; font-size:18px;">
    Subtitle</h2>

<h2 style="color:red; background-color:yellow; font-size:18px;">Another
     Subtitle</h2>
```

What if you want to change all `<h2>` colours? You would need to update every single element separately.

Solution with CSS

Move styling into CSS rules so one change updates all matching elements.

Listing 2: Separating style into CSS

```
h1 {
    color: blue;
    background-color: black;
    font-size: 24px;
    padding: 10px;
}

h2 {
    color: red;
    background-color: yellow;
    font-size: 18px;
}
```

Now change **one** rule in the CSS file, and **all** `<h2>` elements update automatically.

4.3 Benefits of CSS

Why CSS matters

CSS makes your sites easier to change, reuse, and optimise.

1. **Maintainability**: Change styling in one place
2. **Consistency**: All pages on your site can use the same stylesheet
3. **Separation of concerns**: Designers work on CSS, developers work on HTML
4. **Reusability**: One stylesheet can style multiple pages
5. **Performance**: Browser caches CSS files

5 Three Ways to Apply CSS

Overview

CSS can be applied in three different ways. Understanding when to use each method is crucial.

5.1 Method 1: Inline Styles (Least Preferred)

Definition

Style attributes written directly inside HTML elements.

Syntax:

Listing 3: Inline CSS syntax

```
<element style="property: value; property: value;">
```

Example:

Listing 4: Inline styles example

```
<h1 style="color: blue; background-color: black;">
    Conor's Jedward Appreciation Page
</h1>

<h2 style="color: red; background-color: yellow;">
    Welcome message
</h2>

<p style="font-size: 16px; color: gray;">
    This is a paragraph with inline styling.
</p>
```

When to use

- Quick testing
- One-off styling that will not be reused
- Overriding other styles in special cases

Disadvantages

- Hard to maintain (must update each element individually)
- Mixes content with presentation
- Cannot be cached by the browser
- Least reusable

5.2 Method 2: Internal Stylesheet (Good for Single Pages)

Definition

CSS rules defined in the `<head>` section of an HTML document inside a `<style>` element.

Syntax:

Listing 5: Internal stylesheet example

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Page Title</title>
    <style>
        /* CSS rules go here */
        h1 {
            color: blue;
            background-color: black;
        }
        h2 {
            color: red;
        }
    </style>
</head>
<body>
    <h1>Title</h1>
    <h2>Subtitle</h2>
</body>
</html>
```

When to use

- Single-page websites
- Page-specific styles that will not be used elsewhere
- Quick prototyping

Advantages over inline

- Styles centralised in one location
- Can target multiple elements with one rule
- Cleaner HTML body

Disadvantages

- Still mixes CSS with the HTML document
- Cannot be reused across multiple pages
- Makes the HTML file larger

5.3 Method 3: External Stylesheet (Best Practice)

Definition

CSS rules stored in a separate `.css` file, linked to HTML documents.

Step 1: Create CSS file (e.g., `styles.css`):

Listing 6: Example external stylesheet (styles.css)

```
/* styles.css */
body {
    background-color: pink;
}

h1 {
    color: blue;
    background-color: black;
}

h2 {
    color: red;
}
```

Step 2: Link CSS file in HTML `<head>`:

Listing 7: Linking an external stylesheet

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Page Title</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Title</h1>
    <h2>Subtitle</h2>
</body>
</html>
```

Understanding the `<link>` element:

Listing 8: The link element for CSS

```
<link rel="stylesheet" href="week3-2.css">
```

- `rel="stylesheet"` → Relationship: this is a stylesheet
- `href="week3-2.css"` → Location: filename of the CSS file

When to use

- Multi-page websites (ALWAYS)
- Professional projects
- Any reusable styling

Advantages

- **Complete separation** of content (HTML) and presentation (CSS)
- **One file styles multiple pages** (change CSS once, all pages update)
- **Browser caches CSS** (faster page loads)
- **Team collaboration** (designers work on CSS, developers on HTML)
- **Easier maintenance**

Best practice

This is the **best practice** method for professional web development.

6 Task Analysis: Week 3 Lab Requirements

Big picture

The lab asks you to create **demonstration pages** showing how the **same HTML content** can look **completely different** with different stylesheets.

6.1 Understanding the Lab Structure

Key concept: This proves that **CSS controls appearance, not HTML**.

What you are demonstrating

- Same HTML structure
- Different CSS files
- Very different visual designs

6.2 Tasks Breakdown

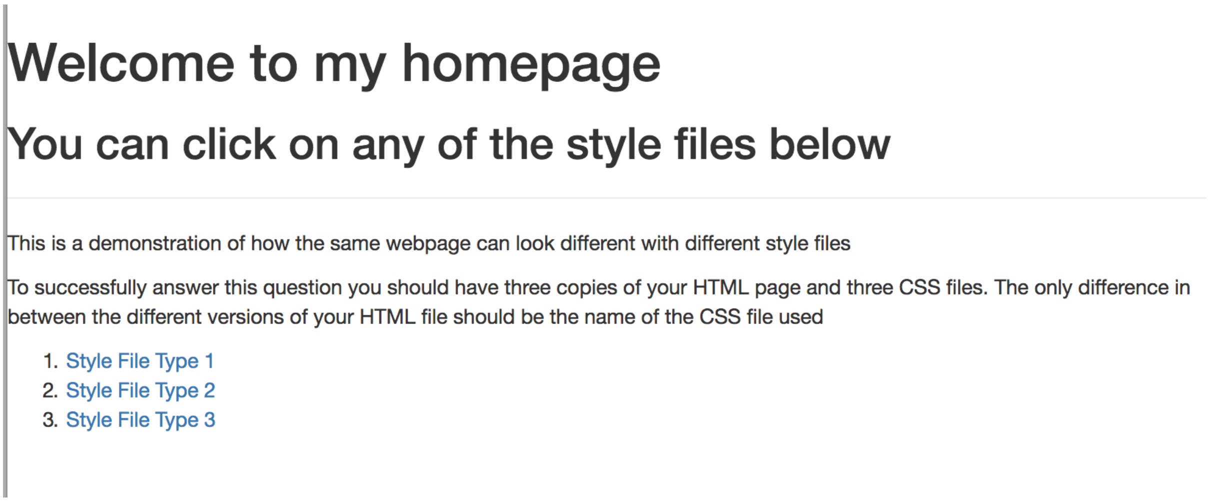
Task Number	Requirement	File(s) to Create
Task 2	Create webpage with basic content	`week3-1.html`
Task 3	Copy Task 2, add external stylesheet	`week3-2.html`, `week3-2.css`
Task 4	Copy Task 3 HTML, use different CSS	`week3-3.html`, `week3-3.css`
Task 5	Ensure links work correctly	Update links in all files
Task 6	Boxify design (practice identifying layout)	Paper/annotation exercise
Tasks 7–9	Create “My Favorite App” page with DIVs	`week3-4.html`
Task 10	Create “Jane Doette” portfolio page	`week3-5.html`
Task 11	Create index page linking to all pages	`index.html`

6.3 Expected Learning Outcome

By the end of this lab you should

1. Understand how CSS separates presentation from content
2. See how the cascade lets styles override each other
3. Be able to write CSS selectors and properties
4. Use `divs` and classes for page layout
5. Create multi-page websites with consistent navigation

7 Task 2: Creating `week3-1.html` (Basic HTML)



Welcome to my homepage

You can click on any of the style files below

This is a demonstration of how the same webpage can look different with different style files

To successfully answer this question you should have three copies of your HTML page and three CSS files. The only difference in between the different versions of your HTML file should be the name of the CSS file used

1. Style File Type 1
2. Style File Type 2
3. Style File Type 3

Figure 1: Reference output for Task 2

7.1 Task Requirements

Goal for Task 2

Create a webpage called `week3-1.html` that matches the unstyled design shown in the lab.

Expected appearance:

- Heading: “Welcome to my homepage”
- Text: “You can click on any of the style files below”
- Description paragraph about demonstrating different style files
- Three links: “Style File Type 1”, “Style File Type 2”, “Style File Type 3”

Important

The Task 2 page uses only **plain HTML**. No custom CSS yet.

7.2 HTML Concepts Needed

You already learned these in Week 2:

- Document structure (`<!DOCTYPE html>`, `<html>`, `<head>`, `<body>`)
- Headings (`<h1>`, `<h2>`)
- Paragraphs (`<p>`)
- Links (`<a href="...">`)

7.3 Solution: `week3-1.html`

Listing 9: Task 2 basic HTML page

```html
<!DOCTYPE html>
<!-- HTML5 document declaration -->

<html lang="en">
<!-- Root element with language specification -->

<head>
    <meta charset="UTF-8">
    <!-- Character encoding for proper text display -->

    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!-- Responsive design viewport settings -->

    <title>Welcome to my homepage</title>
    <!-- Page title shown in browser tab -->
</head>

<body>
    <!-- Body contains all visible content -->

    <h1>Welcome to my homepage</h1>
    <!-- Main heading -->

    <h2>You can click on any of the style files below</h2>
    <!-- Subheading -->

    <p>
        This is a demonstration of how the same webpage can look different
        with different style files.
    </p>
    <!-- Introductory paragraph -->

    <p>
        To successfully answer this question you should have three copies
        of your HTML page and three CSS files. The only difference between
        the different versions of your HTML file should be the name of the
        CSS file used.
    </p>
    <!-- Explanation paragraph -->

    <ol>
        <li><a href="week3-1.html">Style File Type 1</a></li>
        <li><a href="week3-2.html">Style File Type 2</a></li>
        <li><a href="week3-3.html">Style File Type 3</a></li>
    </ol>
    <!-- Ordered list with navigation links -->

</body>

</html>
```

7.4 Testing Task 2

How to test Task 2

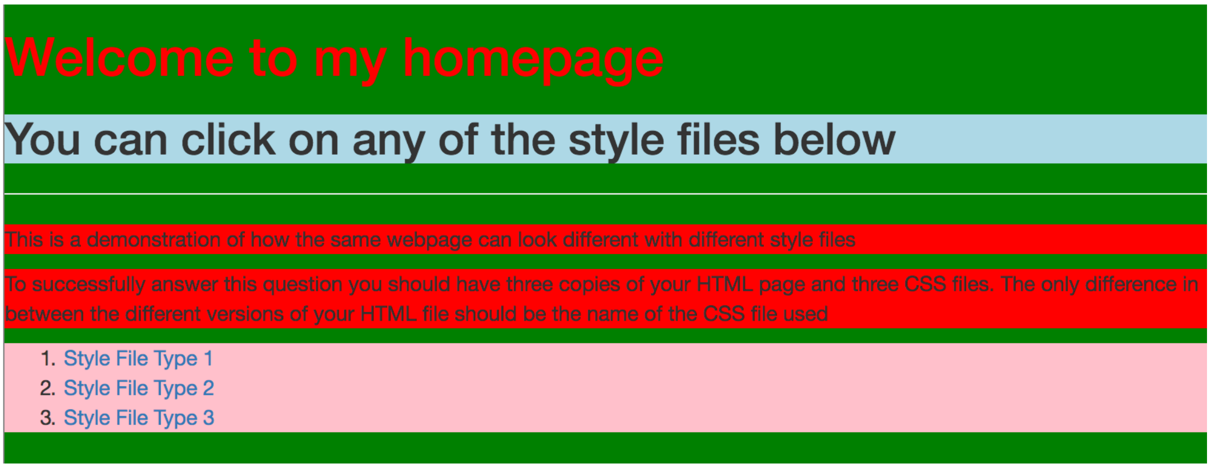
1. Save the file as `week3-1.html` in your `week03` folder.
2. Open with Live Server (right-click → *Open with Live Server*).
3. Verify the checklist below.

- Heading displays correctly
- Text is readable
- Three links are visible (will not work yet – later tasks create those pages)
- Page structure matches the requirement

Reminder

At this point, the page has **no custom styling** — it uses browser defaults.

8 Task 3: Creating `week3-2.html` and `week3-2.css` (External Stylesheet)



Welcome to my homepage

You can click on any of the style files below

This is a demonstration of how the same webpage can look different with different style files

To successfully answer this question you should have three copies of your HTML page and three CSS files. The only difference in between the different versions of your HTML file should be the name of the CSS file used

1. Style File Type 1
2. Style File Type 2
3. Style File Type 3

Figure 2: Reference output for Task 3

8.1 Task Requirements

What you must do

1. **Copy** `week3-1.html` to create `week3-2.html`.
2. Create an **external stylesheet** called `week3-2.css`.
3. Link the stylesheet to `week3-2.html`.
4. Style the page to look like the provided image (green and red colour scheme).

8.2 Understanding the Target Appearance

Based on the lab image, `week3-2.html` should have:

- **Green background** for headings
- **Red background** for description paragraphs
- **Pink background** for list items

Concept check

The HTML is almost identical to Task 2 — **only CSS changes the look**.

8.3 Step 1: Create `week3-2.html`

Listing 10: week3-2.html with linked stylesheet

```html
<!DOCTYPE html>
<!-- HTML5 document declaration -->

<html lang="en">
<!-- Root element -->

<head>
    <meta charset="UTF-8">
    <!-- Character encoding -->

    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!-- Responsive viewport -->

    <title>Welcome to my homepage</title>
    <!-- Page title -->

    <link rel="stylesheet" href="week3-2.css">
    <!-- CRITICAL: Link to external stylesheet -->
</head>

<body>

    <h1>Welcome to my homepage</h1>
    <!-- Main heading - will be styled green -->

    <h2>You can click on any of the style files below</h2>
    <!-- Subheading - will be styled green -->

    <p>
        This is a demonstration of how the same webpage can look different
        with different style files.
    </p>
    <!-- Paragraph - will be styled red -->

    <p>
        To successfully answer this question you should have three copies of your
        HTML page and three CSS files. The only difference between the different
        versions of your HTML file should be the name of the CSS file used.
    </p>
    <!-- Paragraph - will be styled red -->

    <ol>
        <li><a href="week3-1.html">Style File Type 1</a></li>
        <li><a href="week3-2.html">Style File Type 2</a></li>
        <li><a href="week3-3.html">Style File Type 3</a></li>
    </ol>
    <!-- List items - will be styled pink -->

</body>

</html>
```

8.4 Step 2: Create week3-2.css

Listing 11: week3-2.css – external stylesheet

```
/* week3-2.css */
/* External stylesheet for week3-2.html */

/* Style the body element */
body {
    background-color: white;
    font-family: Arial, sans-serif;
    margin: 20px;
}

/* Style all h1 elements (main heading) */
h1 {
    background-color: green;
    color: white;
    padding: 15px;
}

/* Style all h2 elements (subheading) */
h2 {
    background-color: green;
    color: black;
    padding: 10px;
}
```

```

/* Style all paragraph elements */
p {
    background-color: red;
    color: white;
    padding: 10px;
    margin: 10px 0;
}

/* Style list items */
li {
    background-color: pink;
    padding: 5px;
    margin: 5px 0;
}

/* Style links */
a {
    color: blue;
    text-decoration: none;
}

a:hover {
    text-decoration: underline;
}
```

8.5 Understanding the CSS

Breaking down key rules

Each rule targets a different HTML element and changes its appearance.

1. Body styling:

```
body {
    background-color: white;
    font-family: Arial, sans-serif;
    margin: 20px;
}
```

- Sets page background to white
- Uses Arial for all text
- Adds 20px margin around the page

2. `h1` styling:

```
h1 {
    background-color: green;
    color: white;
    padding: 15px;
}
```

- Green background
- White text

- 15px padding inside the heading

3. `h2` styling:

```
h2 {
    background-color: green;
    color: black;
    padding: 10px;
}
```

- Same green background as `h1`
- Black text
- 10px padding

4. Paragraph styling:

```
p {
    background-color: red;
    color: white;
    padding: 10px;
    margin: 10px 0;
}
```

- Red background, white text
- 10px padding inside
- 10px margin above and below each paragraph

5. List item styling:

```
li {
    background-color: pink;
    padding: 5px;
    margin: 5px 0;
}
```

- Pink background
- 5px padding and vertical spacing

6. Link styling:

```
a {
    color: blue;
    text-decoration: none;
}

a:hover {
    text-decoration: underline;
}
```

- Blue links without underline by default
- Underline appears on hover (`:hover` is a **pseudo-class**)

8.6 CSS Syntax Rules

General CSS rule pattern

Every CSS rule follows this structure:

```
selector {
    property: value;
    property: value;
}
```

Components:

1. **Selector**: What element(s) to style (e.g., `h1`, `p`, `body`)
2. **Curly braces** {}: Contain all properties for this selector
3. **Property**: What aspect to change (e.g., `color`, `background-color`, `padding`)
4. **Value**: What to change it to (e.g., `red`, `15px`, `Arial`)
5. **Semicolon** `;`: Ends each property–value pair

8.7 Testing Task 3

How to test your styled page

1. Save both `week3-2.html` and `week3-2.css` in your `week03` folder.
2. Open `week3-2.html` with Live Server.
3. Verify the checklist below.

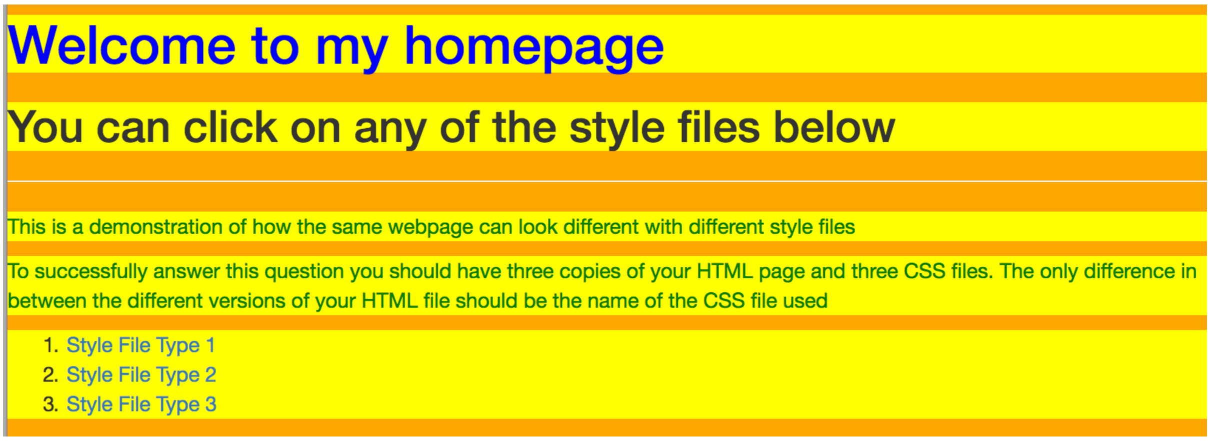
- Headings have green background
- Paragraphs have red background
- List items have pink background
- Colours match the lab image
- Clicking “Style File Type 2” shows the styled page

If styling does not appear

Check the following:

- CSS filename matches exactly (`week3-2.css`)
- The `<link>` tag is inside the `<head>` section
- No typos in property names or values
- CSS file is in the same folder as the HTML file

9 Task 4: Creating `week3-3.html` with Different Stylesheet



Welcome to my homepage

You can click on any of the style files below

This is a demonstration of how the same webpage can look different with different style files

To successfully answer this question you should have three copies of your HTML page and three CSS files. The only difference in between the different versions of your HTML file should be the name of the CSS file used

1. Style File Type 1
2. Style File Type 2
3. Style File Type 3

Figure 3: Reference output for Task 4

9.1 Task Requirements

What you must do

1. Create a **copy** of `week3-2.html` named `week3-3.html`.
2. Create a **different** CSS file called `week3-3.css`.
3. Link `week3-3.html` to `week3-3.css`.
4. Style the page to look like the provided image (yellow colour scheme).

9.2 Understanding the Goal

Key idea

This task demonstrates that **the same HTML can have completely different appearances** just by changing the CSS file reference.

Changes needed:

- `week3-2.html` uses `week3-2.css` (green/red theme).
- `week3-3.html` uses `week3-3.css` (yellow theme).
- HTML content is **identical**; only the CSS file reference changes.

9.3 Step 1: Create `week3-3.html`

Listing 12: week3-3.html with alternative stylesheet

```
<!DOCTYPE html>
<!-- HTML5 document declaration -->

<html lang="en">
<!-- Root element -->
```

```

<head>
    <meta charset="UTF-8">
    <!-- Character encoding -->

    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!-- Responsive viewport -->

    <title>Welcome to my homepage</title>
    <!-- Page title -->

    <link rel="stylesheet" href="week3-3.css">
    <!-- NOTICE: Different CSS file than week3-2.html -->
</head>

<body>

    <h1>Welcome to my homepage</h1>

    <h2>You can click on any of the style files below</h2>

    <p>
        This is a demonstration of how the same webpage can look different
        with different style files.
    </p>

    <p>
        To successfully answer this question you should have three copies of
        your HTML page and three CSS files. The only difference between the
        different versions of your HTML file should be the name of the CSS
        file used.
    </p>

    <ol>
        <li><a href="week3-1.html">Style File Type 1</a></li>
        <li><a href="week3-2.html">Style File Type 2</a></li>
        <li><a href="week3-3.html">Style File Type 3</a></li>
    </ol>

</body>

</html>
```

9.4 Step 2: Create `week3-3.css`

Listing 13: week3-3.css – yellow/orange theme

```
/* week3-3.css */
/* Alternative stylesheet with yellow colour scheme */

/* Style the body element */
body {
    background-color: lightyellow;
    font-family: ’Georgia’, serif;
    margin: 20px;
}

/* Style all h1 elements */
h1 {
    background-color: yellow;
    color: black;
    padding: 15px;
    border: 2px solid orange;
}

/* Style all h2 elements */
h2 {
    background-color: orange;
    color: black;
    padding: 10px;
}

/* Style all paragraph elements */
p {
    background-color: yellow;
    color: black;
    padding: 10px;
    margin: 10px 0;
    border-left: 5px solid orange;
}

/* Style list items */
li {
    background-color: lightyellow;
    padding: 5px;
    margin: 5px 0;
    border-bottom: 1px solid orange;
}

/* Style links */
a {
    color: darkblue;
    text-decoration: underline;
}

a:hover {
    color: orange;
}
```

9.5 Comparing the Two Stylesheets

Element	week3-2.css (Green/Red Theme)	week3-3.css (Yellow Theme)
`body`	White background, Arial font	Light yellow background, Georgia font
`h1`	Green background, white text	Yellow background, black text, orange border
`h2`	Green background, black text	Orange background, black text
`p`	Red background, white text	Yellow background, black text, orange left border
`li`	Pink background	Light yellow background, orange bottom border
`a`	Blue, no underline	Dark blue, underlined

Table 1: Visual differences between `week3-2.css` and `week3-3.css`

9.6 Testing Task 4

How to test Task 4

1. Save both files in your `week03` folder.
2. Open `week3-3.html` with Live Server.
3. Verify the checklist below.

- Page has yellow/orange colour scheme
- Colours are noticeably different from `week3-2.html`
- All three links work and show different styling
- HTML content is identical across pages

Click through all three style links:

- Style File Type 1 → Unstyled (browser defaults)
- Style File Type 2 → Green/red theme
- Style File Type 3 → Yellow/orange theme

What this shows

Same content, completely different looks — this is the **power of CSS**.

10 Task 5: Ensuring Proper Navigation Links

10.1 Task Requirements

Navigation consistency

Make sure that the links on all pages point to the **correct pages**.

Current state:

- `week3-1.html` links to `week3-1.html`, `week3-2.html`, `week3-3.html`
- `week3-2.html` links to `week3-1.html`, `week3-2.html`, `week3-3.html`
- `week3-3.html` links to `week3-1.html`, `week3-2.html`, `week3-3.html`

10.2 Understanding Link Requirements

What the lab means

"Style File 1 links to `week3-1.html`, and Style File 2 links to `week3-2.html`, etc."

This means:

- "Style File Type 1" → `week3-1.html` (unstyled version)
- "Style File Type 2" → `week3-2.html` (green/red theme)
- "Style File Type 3" → `week3-3.html` (yellow theme)

10.3 Verification Checklist

For **all three HTML files** (`week3-1.html`, `week3-2.html`, `week3-3.html`), ensure the links section looks like this:

Listing 14: Consistent navigation block

```html
<ol>
    <li><a href="week3-1.html">Style File Type 1</a></li>
    <li><a href="week3-2.html">Style File Type 2</a></li>
    <li><a href="week3-3.html">Style File Type 3</a></li>
</ol>
```

10.4 Testing Navigation

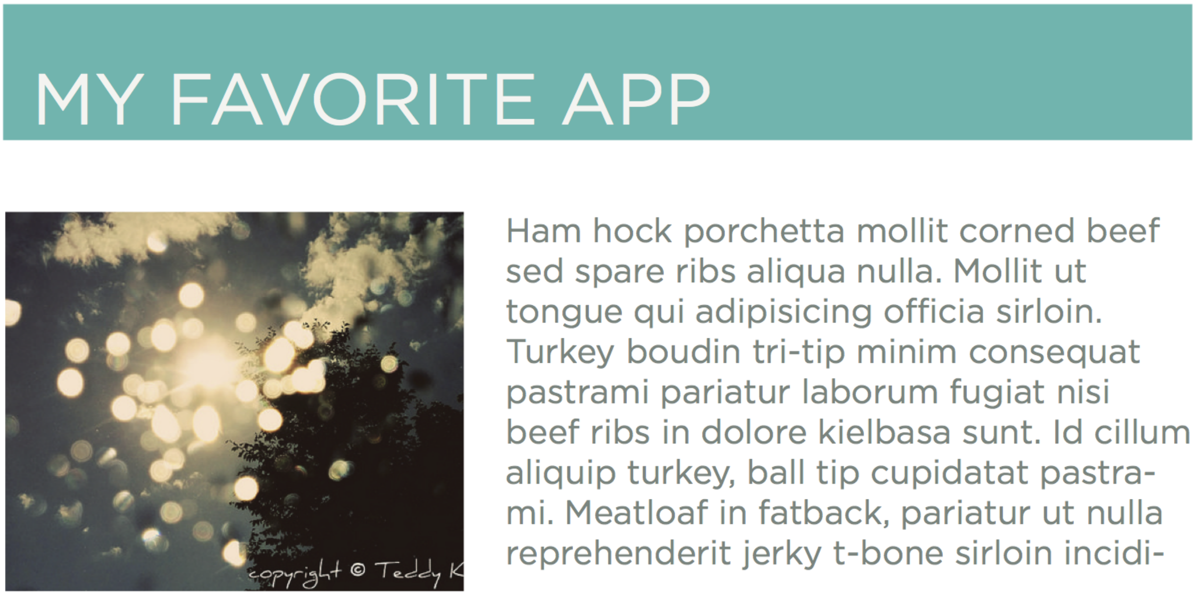
Navigation test plan

1. Open `week3-1.html`.
2. Click "Style File Type 1" → Should stay on unstyled page.
3. Click "Style File Type 2" → Should navigate to green/red themed page.
4. Click "Style File Type 3" → Should navigate to yellow themed page.
5. Repeat from each page to verify all links work correctly.

Expected behaviour:

- All three pages have identical navigation.
- You can switch between style versions seamlessly.
- Content remains the same; only styling changes.

11 Task 6: Boxifying the Design



MY FAVORITE APP

Ham hock porchetta mollit corned beef sed spare ribs aliqua nulla. Mollit ut tongue qui adipisicing officia sirloin. Turkey boudin tri-tip minim consequat pastrami pariatur laborum fugiat nisi beef ribs in dolore kielbasa sunt. Id cillum aliquip turkey, ball tip cupidatat pastrami. Meatloaf in fatback, pariatur ut nulla reprehenderit jerky t-bone sirloin incidi-

Figure 4: Reference design for boxifying (Task 6)

11.1 Task Requirements

What you must submit

"Boxify (on paper) the design of this page — it is acceptable to use Acrobat Reader to annotate it."

11.2 What Does "Boxify" Mean?

Key idea

Boxifying is the process of identifying the rectangular "boxes" (DIVs) that make up a page layout.

Key concept from lectures:

> "Everything's a box. DIVs are used to create 'boxes', into which items are placed."

11.3 Why Boxify?

Before writing HTML/CSS for complex layouts:

- **Visual planning**: Identify structure before coding
- **Hierarchy understanding**: See which boxes contain other boxes
- **Layout strategy**: Decide how to arrange elements

11.4 How to Boxify

Step 1: Print or open the page image

Step 2: Draw rectangles around visual sections

For the “My Favorite App” page, you would identify:

- Header box (title)
- Image box
- Description text box
- Container box holding multiple elements

Step 3: Label boxes with semantic names

Instead of “box1”, “box2”, use meaningful names:

- `header`
- `card`
- `image`
- `description`

Step 4: Note nesting relationships

```
Container Box
 Title Box
 Content Box
     Image Box
     Text Box
```

11.5 Example Boxification

For the “My Favorite App” page:

```
 MY FAVORITE APP                      ← Title box (cyan background)




  [Image]     Ham hock porchetta...   ← Card box
              (Lorem ipsum text)        Image box (left)
                                        Description box (right)
```

Annotations you might add:

- “Title: cyan background, white text”
- “Card: flexbox layout, two columns”
- “Image: 460px width”
- “Text: 705px width”

11.6 Task 6 Deliverable

Ways to complete the boxifying task

You can:

1. **Print the image** and draw boxes with a pen
2. **Use Acrobat Reader** annotation tools to draw rectangles
3. **Use image editing software** to add coloured rectangles
4. **Use PowerPoint/Google Slides** to overlay shapes

The goal is visual understanding, not perfect technical drawings.

12 Introduction to DIVs and Page Layout

Before tackling Tasks 7–10, you need to understand **DIVs** and **class attributes**.

12.1 What is a DIV?

`<div>` = **Division**

A `<div>` is a **container element** that groups related content together.

Key characteristics:

- Generic block-level container
- No semantic meaning by itself (unlike `<header>`, `<article>`, etc.)
- Used to create "boxes" for layout
- Can contain any other elements (text, images, other `div`s)

Basic syntax:

Listing 15: Basic div syntax

```
<div>
    Content goes here
</div>
```

12.2 Why Use DIVs?

Problem without DIVs

It is hard to treat related elements as a single unit or control layout.

Without DIVs:

Listing 16: Unstructured song profile

```
<h1>Song Profile</h1>
<img src="album.png">
<p>Description text here...</p>
```

How do you style these as a group? How do you position them side-by-side?

Solution with DIVs:

Listing 17: Song profile grouped with divs

```html
<div class="song-profile">
    <div class="image-container">
        <img src="album.png">
    </div>
    <div class="description">
        <h1>Song Profile</h1>
        <p>Description text here...</p>
    </div>
</div>
```

Now you can:

- Style the entire song profile as a unit
- Position image and description side-by-side
- Apply consistent spacing and borders

12.3 The Class Attribute

What classes are for

Classes allow you to target specific groups of elements with CSS.

Syntax:

Listing 18: Single class on a div

```html
<div class="class-name">Content</div>
```

Multiple classes:

Listing 19: Multiple classes on a div

```html
<div class="card featured">Content</div>
```

CSS targets classes with a dot .:

Listing 20: CSS rule for a class

```css
.class-name {
    /* Styles for elements with class="class-name" */
}
```

12.4 Example: Using Classes

HTML:

Listing 21: Card layout HTML

```html
<div class="title">MY FAVORITE APP</div>

<div class="card">
    <div class="card-image">
        <img src="image.png" alt="App screenshot">
    </div>
    <div class="card-description">
        <p>Description text...</p>
    </div>
</div>
```

CSS:

Listing 22: Card layout CSS

```css
.title {
    background-color: cyan;
    color: white;
    padding: 20px;
    text-align: center;
}

.card {
    display: flex;
    background-color: lightgray;
}

.card-image {
    width: 460px;
}

.card-description {
    width: 705px;
    padding: 20px;
}
```

12.5 Naming Classes: Best Practices

Choose good class names

Good class names:

- Descriptive: `header`, `navigation`, `footer`, `card`, `title`
- BEM style: `card__image`, `card__description`
- Purpose-based: `primary-button`, `warning-message`

Avoid these names

Bad class names:

- Non-descriptive: `box1`, `div2`, `thing`
- Style-based: `red-text`, `big-font` (design will change)
- Too generic: `content`, `wrapper` (not meaningful)

13 Tasks 7–9: Creating "My Favorite App" Page (`week3-4.html`)

13.1 Task Requirements

What Tasks 7–9 are about

Task 7: Using `divs` and images downloaded from the web (about 460×300px), create a page approximating the "My Favorite App" design.
Task 8: Boxify (on paper) the design.
Task 9: Using `divs` and images downloaded from the web (460×300px), create a page approximating the design.

Note: Tasks 7 and 9 are effectively the same implementation task: build the app showcase page.

13.2 Understanding the Design

Based on the lab image, the "My Favorite App" page has:

Structure:

1. **Title section**: "MY FAVORITE APP" with cyan/teal background
2. **Card section**: Contains image and text side-by-side

- Left: Image (approximately 460×300px)
- Right: Lorem ipsum placeholder text

Visual appearance:

- Title: Cyan background, white uppercase text
- Card: Light grey background, flexbox layout
- Image: On the left, fills its container
- Text: On the right, wrapped paragraph

Design goal

You are practising how layout (with `divs` and flexbox) separates **structure** from **content**.

13.3 Step 1: Find an Image

Requirements:

- Approximately 460×300px (exact size does not matter)
- Any suitable image (app screenshot, stock photo, etc.)
- Download and save in your `week03` folder (or an `Images` subfolder)

Image sources:

- Unsplash – Free stock photos
- Pixabay – Free images
- Pexels – Free stock photos
- Or use any image you already have

Save as: `app-image.png` or `app-image.jpg` in your `week03` folder.

13.4 Step 2: Get Placeholder Text

Getting lorem ipsum

Use a lorem ipsum generator such as:

Lorem ipsum generator: https://www.lipsum.com/
Generate 1–2 paragraphs of placeholder text to fill the description area.

13.5 Step 3: Create week3-4.html

Listing 23: My Favorite App page (week3-4.html)

```
<!DOCTYPE html>
<!-- HTML5 document declaration -->

<html lang="en">
<!-- Root element -->

<head>
    <meta charset="UTF-8">
    <!-- Character encoding -->

    <meta name="viewport" content="width=device-width, initial-scale
        =1.0">
    <!-- Responsive viewport -->

    <title>My Favorite App</title>
    <!-- Page title -->

    <style>
        /* Internal CSS for this page */

        /* Reset default margins */
        body {
            margin: 0;
            font-family: Arial, sans-serif;
        }

        /* Title section styling */
        .title {
            background-color: #00CED1; /* Cyan/Dark Turquoise */
            color: white;
            padding: 30px;
            text-align: center;
            font-size: 36px;
            font-weight: bold;
            letter-spacing: 2px;
        }

        /* Card container - uses flexbox for side-by-side layout */
        .card {
            display: flex;
            background-color: #f5f5f5; /* Light gray */
            margin: 20px;
            box-shadow: 0 4px 6px rgba(0,0,0,0.1);
        }

        /* Image section (left side) */
        .card-image {
            width: 460px;
            flex-shrink: 0; /* Prevents image from shrinking */
        }

        .card-image img {
            width: 100%;
            height: auto;
            display: block;
        }

        /* Description section (right side) */
        .card-description {
            padding: 30px;
            line-height: 1.6;
            color: #333;
        }
    </style>
</head>

<body>

    <!-- Title section -->
    <div class="title">
        MY FAVORITE APP
    </div>

    <!-- Card section -->
    <div class="card">

        <!-- Left: Image -->
        <div class="card-image">
            <img src="app-image.png" alt="App Screenshot">
        </div>

        <!-- Right: Description -->
        <div class="card-description">
            <p>
                Ham hock porchetta mollit corned beef sed spare ribs aliqua nulla.
                Mollit ut tongue qui adipisicing officia sirloin. Turkey boudin tri-tip
                minim consequat pastrami pariatur laborum fugiat nisi beef ribs in dolore
                kielbasa sunt. Id cillum aliquip turkey, ball tip cupidatat pastrami.
                Meatloaf in fatback, pariatur ut nulla reprehenderit jerky t-bone sirloin
                incididunt.
            </p>
        </div>

    </div>

</body>

</html>
```

13.6 Understanding the CSS

Why flexbox here?

Flexbox makes it easy to place the image and description **side-by-side** inside a single card.

1. Flexbox layout:

```
.card {
    display: flex;
}
```

This makes the `.card` a **flex container**, allowing child elements (`.card-image` and `.card-description`) to sit side-by-side.

Without flexbox: Elements would stack vertically (default block behaviour).

With flexbox: Elements are arranged horizontally in a row.

2. Fixed width for image:

```
.card-image {
    width: 460px;
    flex-shrink: 0;
}
```

- `width: 460px` → Image container is exactly 460px wide.
- `flex-shrink: 0` → Prevents the container from shrinking when the window is resized.

3. Image sizing:

```
.card-image img {
    width: 100%;
    height: auto;
    display: block;
}
```

- `width: 100%` → Image fills the width of its container.
- `height: auto` → Maintains the correct aspect ratio.
- `display: block` → Removes the extra space below the image.

13.7 Testing Task 7/9

Checklist for your app page

1. Save `week3-4.html` with your chosen image.
2. Open it with Live Server.
3. Verify the following:

- Title has cyan background
- Image and text are side-by-side
- Layout approximates the lab image

- Image loads correctly

Responsive test:

- Resize the browser window.
- Image should maintain 460px width.
- Text should reflow but remain in the right column.

14 Task 10: Creating “Jane Doette” Portfolio Page (`week3-5.html`)

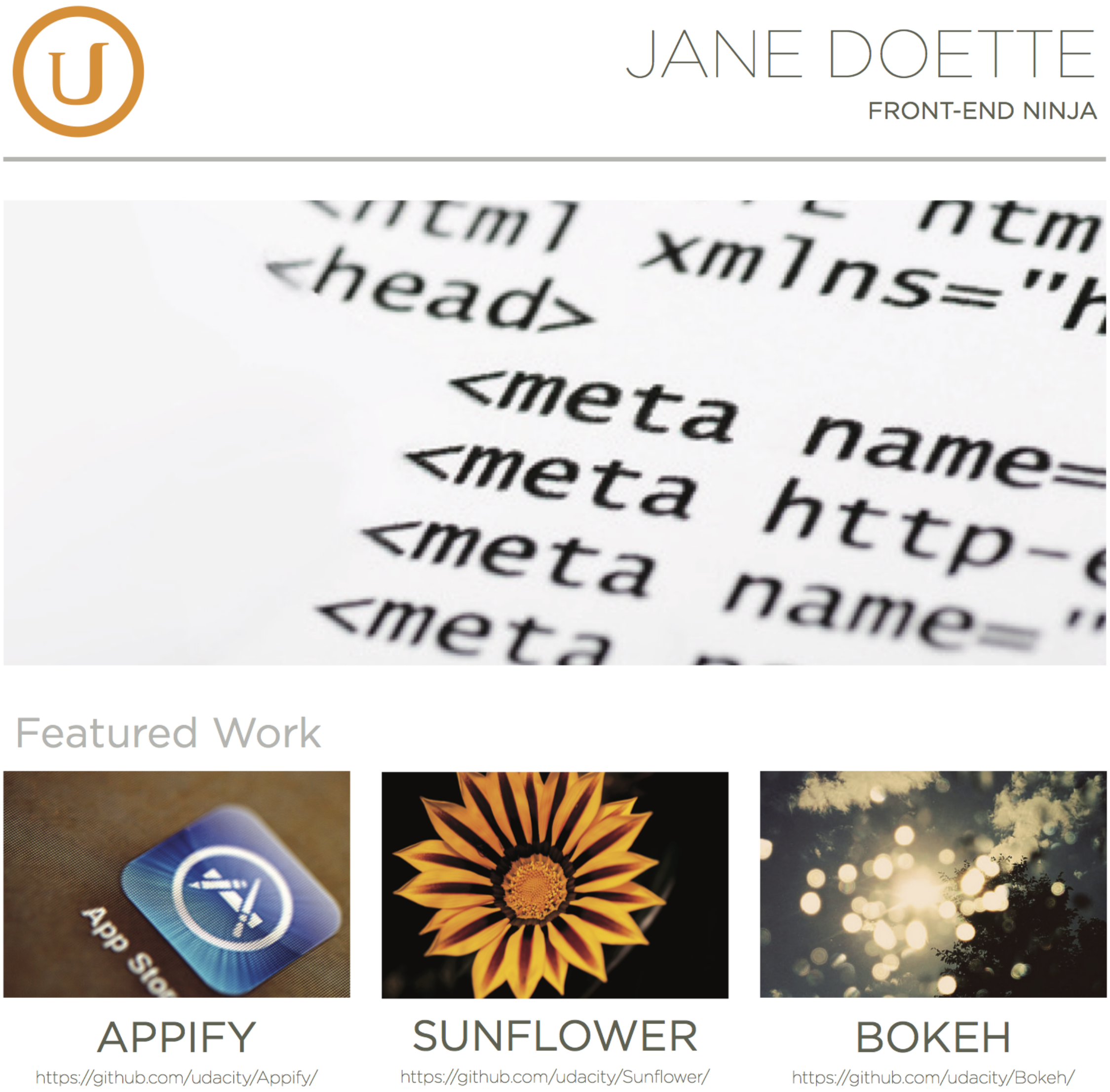

JANE DOETTE
FRONT-END NINJA
Featured Work
APPIFY
https://github.com/udacity/Appify/
SUNFLOWER
https://github.com/udacity/Sunflower/
BOKEH
https://github.com/udacity/Bokeh/

Figure 5: Reference design fo (Task 10)

14.1 Task Requirements

Goal of Task 10

Using `divs` and images downloaded from the web (about 460×300px), create a page approximating the "Jane Doette – Front-End Ninja" design shown in the lab.

14.2 Understanding the Design

The Jane Doette page has a more complex layout:

Structure:

1. **Header section**: Circle logo (left) and name/title (right)
2. **Code preview section**: Shows HTML code snippet
3. **Featured Work section**: Title + three project cards
4. **Project cards**: Three side-by-side cards with images and titles

Visual elements:

- Orange circle logo with "U"
- Name: "JANE DOETTE" in large text
- Subtitle: "FRONT-END NINJA"
- Code preview with light grey background
- Three project cards: "APPIFY", "SUNFLOWER", "BOKEH"

Design insight

This page combines **flexbox**, **cards**, and **typography** to create a professional portfolio layout.

14.3 Step 1: Find Images

You need **three images** (approximately 460×300px each) for the project cards.

Option 1: Download from free stock photo sites. **Option 2:** Use placeholder images from https://placeholder.com/.

Example placeholder URL:
`https://via.placeholder.com/460x300`
Save images as:

- `project1.png`
- `project2.png`
- `project3.png`

14.4 Step 2: Create `week3-5.html`

Listing 24: Jane Doette portfolio page (week3-5.html)

```
<!DOCTYPE html>
<!-- HTML5 document declaration -->

<html lang="en">
<!-- Root element -->

<head>
    <meta charset="UTF-8">
    <!-- Character encoding -->

    <meta name="viewport" content="width=device-width, initial-scale
       =1.0">
    <!-- Responsive viewport -->

    <title>Jane Doette - Front-End Ninja</title>
    <!-- Page title -->

    <style>
        /* Internal CSS for portfolio page */

        /* Reset and base styles */
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: Arial, sans-serif;
            background-color: #f5f5f5;
        }

        /* Header section */
        .header {
            display: flex;
            align-items: center;
            padding: 40px;
            background-color: white;
        }

        /* Circle logo */
        .logo {
            width: 100px;
            height: 100px;
            background-color: #FF8C00; /* Dark orange */
            border-radius: 50%;         /* Makes it circular */
            display: flex;
            align-items: center;
            justify-content: center;
            font-size: 60px;
            color: white;
            font-weight: bold;
            margin-right: 30px;
        }

        /* Name and title */
        .header-text {
            flex-grow: 1;
        }

        .name {
            font-size: 42px;
            font-weight: bold;
            letter-spacing: 3px;
            color: #333;
        }

        .title {
            font-size: 18px;
            color: #888;
            margin-top: 5px;
            letter-spacing: 2px;
        }

        /* Code preview section */
        .code-preview {
            background-color: #e8e8e8;
            padding: 30px 40px;
            margin: 20px 40px;
            border-left: 5px solid #FF8C00;
        }

        .code-preview pre {
            font-family: 'Courier New', monospace;
            font-size: 14px;
            color: #555;
            line-height: 1.8;
        }

        /* Featured Work section */
        .featured-work {
            padding: 40px;
        }

        .section-title {
            font-size: 24px;
            color: #888;
            margin-bottom: 30px;
            letter-spacing: 1px;
        }

        /* Projects container - flexbox for three columns */
        .projects {
            display: flex;
            gap: 20px;
            justify-content: space-between;
        }

        /* Individual project card */
        .project-card {
            background-color: white;
            flex: 1;
            box-shadow: 0 2px 8px rgba(0,0,0,0.1);
        }

        .project-card img {
            width: 100%;
            height: auto;
            display: block;
        }

        .project-title {
            padding: 20px;
            text-align: center;
            font-size: 18px;
            font-weight: bold;
            color: #333;
        }

        .project-link {
            display: block;
            padding: 10px;
            text-align: center;
            color: #888;
            text-decoration: none;
            font-size: 14px;
        }

        .project-link:hover {
            text-decoration: underline;
        }
    </style>
</head>

<body>

    <!-- Header section with logo and name -->
    <div class="header">
        <div class="logo">U</div>
        <div class="header-text">
            <div class="name">JANE DOETTE</div>
            <div class="title">FRONT-END NINJA</div>
        </div>
    </div>

    <!-- Code preview section -->
    <div class="code-preview">
        <pre>
&lt;!item&gt;
&lt;heads&gt;
    &lt;meta name=
    &lt;meta http-e
    &lt;meta name=
    &lt;met
        &lt;meta name=
    &lt;/met&gt;
        </pre>
    </div>

    <!-- Featured Work section -->
    <div class="featured-work">
```

```
        <div class="section-title">Featured Work</div>

        <div class="projects">

            <!-- Project 1: APPIFY -->
            <div class="project-card">
                <img src="project1.png" alt="Appify Project">
                <div class="project-title">APPIFY</div>
                <a href="#" class="project-link">http://github.com/
                   appify</a>
            </div>

            <!-- Project 2: SUNFLOWER -->
            <div class="project-card">
                <img src="project2.png" alt="Sunflower Project">
                <div class="project-title">SUNFLOWER</div>
                <a href="#" class="project-link">http://github.com/
                   sunflower</a>
            </div>

            <!-- Project 3: BOKEH -->
            <div class="project-card">
                <img src="project3.png" alt="Bokeh Project">
                <div class="project-title">BOKEH</div>
                <a href="#" class="project-link">http://github.com/
                   bokeh</a>
            </div>

        </div>
    </div>

</body>

</html>
```

14.5 Understanding the Complex Layout

Layout techniques used

This page uses flexbox in two places: the header and the project card row.

1. Header flexbox:

```
.header {
    display: flex;
    align-items: center;
}
```

- `display: flex` → Lays out logo and text horizontally.
- `align-items: center` → Vertically centres logo and text.

2. Circular logo:

```
.logo {
    width: 100px;
    height: 100px;
    border-radius: 50%;
}
```

- Equal width and height make a square.
- `border-radius: 50%` turns the square into a circle.

3. Three-column layout:

```
.projects {
    display: flex;
    gap: 20px;
}

.project-card {
    flex: 1;
}
```

- `display: flex` → Arranges the three project cards in a row.
- `gap: 20px` → Adds space between cards.
- `flex: 1` → Each card takes equal width (roughly one-third).

4. Code preview styling:

```
.code-preview pre {
    font-family: 'Courier New', monospace;
}
```

- `<pre>` preserves whitespace and line breaks, so it looks like code.
- Monospace font reinforces the code-editor feel.

14.6 Testing Task 10

Checklist for the portfolio page

1. Save `week3-5.html` with your three project images.
2. Open it with Live Server.
3. Verify:

- Orange circle logo displays
- Name and title are positioned correctly
- Code preview section appears under the header
- Three project cards are side-by-side
- Images load in all cards

Responsive test:

- Resize the browser window.
- Cards should remain side-by-side and keep equal widths (until the viewport is very small).
- Layout should scale proportionally without breaking.

15 Task 11: Creating `index.html` Navigation Page

15.1 Task Requirements

Goal of Task 11

Create an `index.html` page that links to all the HTML pages you created for Week 3.

15.2 Purpose of `index.html`

An **index page** serves as the **homepage** or **navigation hub** for your website.

Why the name `index.html`?

- Web servers automatically serve `index.html` when you visit a directory.
- Example: `http://yoursite.com/week03/` → automatically loads `index.html`.

15.3 Solution: `index.html`

Listing 25: index.html navigation page

```html
<!DOCTYPE html>
<!-- HTML5 document declaration -->

<html lang="en">
<!-- Root element -->

<head>
    <meta charset="UTF-8">
    <!-- Character encoding -->

    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!-- Responsive viewport -->

    <title>CS4116 Week 3 - Lab Navigation</title>
    <!-- Page title -->

    <style>
        /* Internal CSS for index page */

        body {
            font-family: Arial, sans-serif;
            max-width: 800px;
            margin: 50px auto;
            padding: 20px;
            background-color: #f9f9f9;
        }

        h1 {
            color: #333;
            border-bottom: 3px solid #4CAF50;
            padding-bottom: 10px;
        }

        h2 {
            color: #555;
            margin-top: 30px;
        }

        ul {
            list-style-type: none;
            padding: 0;
        }

        li {
            margin: 10px 0;
        }

        a {
            display: block;
            padding: 15px;
            background-color: white;
            color: #4CAF50;
            text-decoration: none;
            border-left: 5px solid #4CAF50;
            transition: all 0.3s ease;
        }

        a:hover {
            background-color: #4CAF50;
            color: white;
            padding-left: 25px;
        }

        .description {
            color: #777;
            font-size: 14px;
            margin-top: 5px;
        }
    </style>
</head>

<body>

    <h1>CS4116 Week 3 Lab - HTML &amp; CSS Exercises</h1>

    <p>
        This page provides navigation to all HTML pages created for Week 3 lab exercises.
    </p>

    <h2>CSS Styling Demonstrations (Tasks 2  5  )</h2>
    <ul>
        <li>
            <a href="week3-1.html">
                Task 2: Basic HTML Page (Unstyled)
                <div class="description">
                    Shows default browser styling with no custom CSS.
                </div>
            </a>
        </li>
        <li>
            <a href="week3-2.html">
```

```
                Task 3: External Stylesheet Version 1 (Green/Red Theme)
                <div class="description">
                    Demonstrates external CSS with green and red colour scheme.
                </div>
            </a>
        </li>
        <li>
            <a href="week3-3.html">
                Task 4: External Stylesheet Version 2 (Yellow Theme)
                <div class="description">
                    Same HTML content with different CSS (yellow colour scheme).
                </div>
            </a>
        </li>
    </ul>

    <h2>DIV Layouts and Page Design</h2>
    <ul>
        <li>
            <a href="week3-4.html">
                Tasks 7 9 : My Favorite App Page
                <div class="description">
                    Card layout with image and text using flexbox.
                </div>
            </a>
        </li>
        <li>
            <a href="week3-5.html">
                Task 10: Jane Doette Portfolio Page
                <div class="description">
                    Complex layout with header, code preview, and project cards.
                </div>
            </a>
        </li>
    </ul>

    <h2>Resources</h2>
    <ul>
        <li>
            <a href="https://www.w3schools.com/css/" target="_blank">
                W3Schools CSS Tutorial
                <div class="description">
                    Comprehensive CSS reference and examples.
                </div>
            </a>
        </li>
        <li>
            <a href="https://developer.mozilla.org/en-US/docs/Web/CSS" target="_blank">
                MDN Web Docs - CSS
                <div class="description">
                    Detailed CSS documentation from Mozilla.
                </div>
            </a>
        </li>
    </ul>

</body>

</html>
```

15.4 Understanding the Index Page

Navigation structure

1. **Navigation structure:**
   - Grouped by task type.
   - Descriptive links with short explanations.
   - External resources listed at the bottom.

2. **Visual enhancements:**

- Hover effects on links (colour change and left indent).
- Left border accents highlighting each link.
- Clean, centred layout for readability.

3. **Accessibility:**

- Descriptive link text (not just "click here").
- Clear heading hierarchy (`h1`, then `h2` groups).
- Legible font sizes and good colour contrast.

15.5 Testing Task 11

How to test the index page

1. Save `index.html` in your `week03` folder.
2. Open it with Live Server.
3. Verify the checklist below.

- All internal links work (`week3-1.html` through `week3-5.html`).
- External links open in new tabs.
- Hover effects work on all navigation links.
- Page content is well-organised and easy to scan.

File structure check

```
week03/
 index.html
 week3-1.html
 week3-2.html
 week3-2.css
 week3-3.html
 week3-3.css
 week3-4.html
 week3-5.html
 app-image.png
 project1.png
 project2.png
 project3.png
```

16 CSS Properties Reference

How to use this section

This section provides a quick reference for CSS properties used in this week's lab, grouped by topic.

16.1 Color Properties

Text colour examples:

Listing 26: Text colour

```
color: red;
color: #FF0000;
color: rgb(255, 0, 0);
```

Background colour examples:

Listing 27: Background colour

```
background-color: blue;
background-color: #0000FF;
background-color: rgba(0, 0, 255, 0.5); /* With transparency */
```

Colour formats:

- **Named colours**: `red`, `blue`, `green`, `cyan`, etc.
- **Hex codes**: `#RRGGBB` (e.g., `#FF5733`)
- **RGB**: `rgb(red, green, blue)` values 0–255
- **RGBA**: `rgba(red, green, blue, alpha)` where alpha is transparency 0–1

Colour picker tool

A handy tool: HTML Color Codes.

16.2 Spacing Properties

Padding (space inside element, around content):

Listing 28: Padding examples

```
padding: 10px;              /* All sides */
padding: 10px 20px;         /* Top/bottom 10px, left/right 20px */
padding: 10px 20px 15px 25px;/* Top, right, bottom, left */

padding-top: 10px;
padding-right: 20px;
padding-bottom: 15px;
padding-left: 25px;
```

Margin (space outside element, between elements):

Listing 29: Margin examples

```
margin: 10px;      /* All sides */
margin: 10px 20px; /* Top/bottom 10px, left/right 20px */
margin: 10px 0;    /* Top/bottom 10px, left/right 0 */

margin-top: 10px;
```

Padding vs margin

- **Padding**: Inside the element's border (increases element size).
- **Margin**: Outside the element's border (creates space between elements).

16.3 Text Properties

Font:

Listing 30: Font properties

```
font-family: Arial, sans-serif;
font-size: 16px;
font-weight: bold; /* or normal, 100  900   */
```

Text alignment:

Listing 31: Text alignment

```
text-align: left;   /* or center, right, justify */
```

Text decoration:

Listing 32: Text decoration

```
text-decoration: none;     /* Removes underline from links */
text-decoration: underline;
```

Letter spacing and line height:

Listing 33: Spacing within text

```
letter-spacing: 2px;
line-height: 1.6;   /* 1.6 times font size */
```

16.4 Box Properties

Width and height:

Listing 34: Size properties

```
width: 460px;
max-width: 1200px;
height: 300px;
```

Border:

Listing 35: Border examples

```
border: 1px solid black;   /* Width, style, colour */
border-left: 5px solid orange;
border-radius: 50%;        /* Circular corners */
```

Box shadow:

Listing 36: Box shadow

```
box-shadow: 0 4px 6px rgba(0,0,0,0.1);
/* horizontal offset, vertical offset, blur, colour */
```

16.5 Display Properties

Display type:

Listing 37: Display types

```
display: block;  /* Takes full width */
display: inline; /* Takes only content width */
display: flex;   /* Flexbox container */
display: none;   /* Hides element */
```

Flexbox container properties:

Listing 38: Flex container properties

```
display: flex;
flex-direction: row;      /* or column */
justify-content: center;  /* or flex-start, flex-end, space-between */
align-items: center;      /* or flex-start, flex-end, stretch */
gap: 20px;                /* Space between flex items */
```

Flex item properties:

Listing 39: Flex item properties

```
flex: 1;         /* Grow to fill space */
flex-shrink: 0;  /* Do not shrink */
```

16.6 Pseudo-classes

Link states:

Listing 40: Link pseudo-classes

```
a:hover {
    /* When mouse hovers over link */
}

```

```
a:active {
    /* When link is being clicked */
}

a:visited {
    /* After link has been visited */
}
```

17 Troubleshooting Guide

17.1 Problem 1: “CSS styles aren’t showing”

Possible causes and fixes

1. **CSS file not linked correctly**
   - Check the `<link>` tag in the `<head>` section.
   - Verify `href` matches the CSS filename exactly.
   - Ensure the CSS file is in the same folder as the HTML.
2. **Typo in CSS filename**

Listing 41: CSS filename typo

```
<!-- Wrong -->
<link rel="stylesheet" href="week3-2.cs">

<!-- Correct -->
<link rel="stylesheet" href="week3-2.css">
```

3. **CSS syntax error**
   - Missing semicolon `;` at end of property.
   - Missing closing brace }.
   - Typo in property name.
4. **Selector does not match**

Listing 42: Selector mismatch

```
/* HTML has class="card" */

.cards {   /* Wrong - extra ’s’ */
    color: red;
}

.card {    /* Correct */
    color: red;
}
```

Debugging steps:

1. Open Browser Developer Tools (F12).
2. Go to “Elements” or “Inspector” tab.

3. Select the element you are trying to style.
4. Check the "Styles" panel to see which CSS is applied.
5. Look for strikethrough styles (overridden by other rules).

17.2 Problem 2: "Flexbox layout not working"

Common flexbox mistakes

1. **Forgot `display: flex` on container**

Listing 43: Missing display:flex

```css
/* Wrong */
.card {
    background: gray;
}

/* Correct */
.card {
    display: flex;
    background: gray;
}
```

2. **Applied flex to wrong element**
   - Flex must be on the **parent container**, not the child elements.
   - Child elements automatically become flex items.
3. **Overriding layout with other display values**
   - Avoid setting conflicting `display` on flex items unless needed.

17.3 Problem 3: "Images not loading"

Image loading issues

Possible causes:

1. **Wrong file path**

Listing 44: Image paths

```html
<!-- If image is in same folder -->
<img src="image.png" alt="Description">

<!-- If image is in Images subfolder -->
<img src="Images/image.png" alt="Description">
```

2. **Case sensitivity**
   - On some servers, `image.PNG image.png`.
   - Use lowercase consistently.
3. **Missing alt attribute**
   - Always include `alt` for accessibility and better debugging.

17.4 Problem 4: “Page looks different in different browsers”

Why this happens

- Different browsers have different **user agent stylesheets** (default styles).
- A simple solution is to use a **CSS reset** to normalise styles.

Simple reset:

Listing 45: Basic reset

```css
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}
```

17.5 Problem 5: “Colours look different than expected”

Check:

- **Hex code typos**: `#00CED1` (cyan) vs `#00CED` (invalid).
- **RGB values out of range**: must be 0–255 (e.g., `rgb(300, 0, 0)` is invalid).
- **Monitor differences**: colours vary; test on multiple devices if critical.

17.6 Problem 6: “Text overflows container”

Possible solutions:

Listing 46: Handling overflow

```css
/* Option 1: Hide overflow */
.container {
    overflow: hidden;
}

/* Option 2: Allow scrolling */
.container {
    overflow: auto;
}

/* Option 3: Break long words */
.container {
    word-wrap: break-word;
}
```

18 Summary and Next Steps

18.1 What We Covered This Week

Week 3 achievements

CSS fundamentals

- Three methods of applying CSS (inline, internal, external).
- CSS syntax (selectors, properties, values).
- Cascade and specificity.

CSS properties

- Colours, spacing, typography.
- Borders, shadows, and display types.

Page layout

- `div`s as container elements.
- Class attributes for targeting.
- Flexbox for side-by-side layouts.

Complete lab tasks

- Three versions of the same page with different styling (`week3-1`, `week3-2`, `week3-3`).
- "My Favorite App" card layout page (`week3-4`).
- "Jane Doette" portfolio page (`week3-5`).
- Navigation index page.

18.2 Key Takeaways

1. **CSS separates presentation from content** → easier maintenance.
2. **External stylesheets are best practice** → reusability and caching.
3. `div`s **+ classes** give powerful layout control.
4. **Flexbox simplifies side-by-side layouts** (no tables needed).
5. **Browser DevTools are essential** for debugging and learning.

18.3 What's Next

Coming next (likely topics):

- More advanced CSS layouts (Grid, positioning).
- Responsive design and media queries.
- CSS transitions and animations.
- Introduction to JavaScript basics.

To prepare:

- Review all Week 3 files.
- Experiment with different CSS properties.
- Practise creating layouts from scratch.
- Use DevTools to study other websites.

18.4 Practice Exercises (Optional)

Extra challenges

Challenge 1: Customise colour schemes

- Create an alternative `week3-4.css`.
- Switch everything to a purple theme.

Challenge 2: Add more project cards

- Modify `week3-5.html` to show 4–5 projects.
- Adjust flexbox layout accordingly.

Challenge 3: Make the index page more visual

- Add thumbnails/screenshots for each page.
- Try a grid layout instead of a list.

18.5 Resources

Official documentation:

- MDN CSS Reference
- W3C CSS Specifications

Learning resources:

- W3Schools CSS Tutorial
- CSS-Tricks
- Flexbox Froggy – interactive flexbox game

Tools:

- HTML Color Codes
- CSS Gradient Generator
- Can I Use – browser compatibility checker

18.6 Getting Help

During lab sessions:

- Ask TAs (Zia Rehman, Mritunjay Musale).
- Work with your group members.
- Use DevTools to debug.

Outside lab:

- Review lecture videos on Brightspace.
- Consult the course guide.
- Email: `rehman.zia@ul.ie` (include `[CS4116]` in the subject).

18.7 Final Reminders

Action items

Complete all lab tasks (`week3-1` through `week3-5`, `index.html`).

Test all pages in a browser.

Verify all links work.

(Optional) Validate HTML and CSS.

Practise flexbox layouts.

Exam reminder

"The mid-term will include coding questions. Practise writing code yourself—do not rely on copy-paste. Understanding **why** code works is more important than just seeing that it does."

Prepared by Zia Ur Rehman – CS4116 Teaching Assistant
University of Limerick – 2026