# LEVEL 4

# Front-End Web Development

# Student Guide

# Modification History

| Version | Date | Revision Description |
|---|---|---|
| V1.0 | March 2024 | For release |
| | | |
| | | |
| | | |
| | | |
| | | |



**Published by: NCC Education Limited, Adamson House, Towers Business Park, Wilmslow Road, Didsbury, Manchester M20 2EZ, UK.**

Tel: +44 (0) 161 438 6200 Fax: +44 (0) 161 438 6240 Email: info@nccedu.com
http://www.nccedu.com

# CONTENTS

# 1. Unit Overview and Objectives

This unit aims to give the learner a thorough knowledge of web coding in HTML and CSS, and an understanding of website design and testing. The first topic introduces the World Wide Web (WWW) and the fundamental challenges facing the designers of websites. Topics 2-8 explore HTML and CSS in detail, equipping students with the skills and knowledge to build effective websites. The module finishes by considering the design of websites and particular user-centred approaches to the design and evaluation.

At the end of the unit, students should be able to:

1. Describe the concepts of website development and underlying technologies, key concepts and tools and technologies necessary for font-end development.
2. Use web development tools to build HTML- and CSS-based websites with a range of CSS frameworks to address well-defined specifications.
3. Build interactive webpages and dynamic webpages using JavaScirpt and associated libraries.
4. Build interactive webpages and dynamic webpages using JavaScirpt and associated libraries.
5. Understand and be proficient with front end development frameworks.
6. Understand and use a version control system (VCS) for project collaboration with other developers and explain the process of deploying websites on servers.
7. Develop appropriate test strategies and apply these to a website.

# 2. Learning Outcomes and Assessment Criteria

| **Learning Outcomes;**<br>The Learner will: | **Assessment Criteria;**<br>The Learner can: |
|---|---|
| 1. Describe the concepts of website development and underlying technologies, key concepts and tools and technologies necessary for font-end development. | 1.1 Describe the underlying technologies for building websites<br>1.2 Explain fundamental concepts of website design including the client-server model<br>1.3 Distinguish between front-end web development and back-end web development<br>1.4 Identify key concepts and technologies needed for front-end web development |
| 2. Use web development tools to build HTML- and CSS-based websites with a range of CSS frameworks to address well-defined specifications. | 2.1 Describe the use of HTML to develop websites<br>2.2 Describe how to use CSS to standardise the overall style of a website<br>2.3 Write the source code for a simple web page in clean HTML according to a specification.<br>2.4 Write the source code for a CSS according to a specification<br>2.5 Explain the contextual application of a variety of web development tools<br>2.6 Explain a range of CSS frameworks including Bootstrap, foundation, Semantic UI<br>2.7 Explain the advantages and disadvantages of various web development methodologies and technologies |

| | |
|---|---|
| 3. Create and design responsive website to scale well across laptop, tablet and mobile phone. | 3.1 Use Bootstrap concepts and develop responsive web pages.<br>3.2 Use CSS3 layout model: Flexbox for responsive and mobile-friendly design.<br>3.3 Explain the difference between CSS Grid Layout and CSS Flexbox Layout. |
| 4. Build interactive webpages and dynamic webpages using JavaScirpt and associated libraries. | 4.1 Write and apply JavaScript programming skills for biulding dynamic and interactive sites<br>4.2 Explain DOM concepts<br>4.3 Effectively use frameworks and libraries for Javascript<br>4.4 Create anmated, interactive web pages using jQuery library.<br>4.5 Build data visualization webpages using d3.js library. |
| 5. Understand and be proficient with front end development frameworks. | 5.1 Develop front end application by using Bootstrap architecture, services, and concepts.<br>5.2 Explain the advantages and disadvantages of using frameworks.<br>5.3 Employ JavaScript libraries to build dynamic, interactive, informative websites |
| 6. Understand and use a version control system (VCS) for project collaboration with other developers and explain the process of deploying websites on servers. | 6.1 Explain the concepts of Version Control and its significance<br>6.2 Track changes and collaboration with GitHub<br>6.3 Explain the flow of GitHub |
| 7. Develop appropriate test strategies and apply these to a website. | 7.1 Develop and apply a test strategy consistent with the design<br>7.2 Develop Cross Broswer COmpetibility Testing |

# 3. Syllabus

| Syllabus | | | |
|---|---|---|---|
| Topic No | Title | Proportion | Content |
| 1 | Overview of Web Application Architecture | 1/12<br><br>2 hours of lectures<br><br>4 hours of laboratory sessions | • What is Web application architecture?<br>• How does the Web application architecture work?<br>• 3-Tier architecture<br>• Layers of Modern We Application Architecture: Presentation Layer, Application Layer, and Data Layer<br>• Web application architecture components<br>• Key concepts and technologies for front-end web development<br>***Learning Outcomes: 1*** |
| 2 | Introduction to HTML5 | 1/12<br><br>2 hours of lectures<br><br>4 hours of laboratory sessions | • What is HTML?<br>• HTML document structure<br>• HTML Tags<br>• Ordered and Unordered Lists<br>• Images<br>• Forms<br>• HTML best practices<br>***Learning Outcomes: 2*** |
| 3 | Introduction to CSS | 1/12<br><br>2 hours of lectures<br><br>4 hours of laboratory sessions | • CSS Introduction<br>• CSS backgrounds, borders, margins, padding<br>• CSS Front Styling<br>• Gradients<br>• CSS best practices<br>***Learning Outcomes:2*** |
| 4 | HTML and CSS | 1/12<br><br>2 hours of lectures<br>4 hours of laboratory | • DIV element<br>• SPAN element<br>• CSS Selectors<br>• Coding with HTML and CSS<br>***Learning Outcomes: 2*** |

| | | | |
|---|---|---|---|
| 5 | CSS Flexbox and Grid Layout | 1/12<br><br>2 hours of lectures<br><br>4 hours of laboratory sessions | • What is CSS Grid Layout?<br>• Grid container<br>• What is CSS Flexbox?<br>• Flex container and flex items<br>• Differences with Grid and Flexbox<br>***Learning Outcomes: 3*** |
| 6 | JavaScript - I | 1/12<br><br>2 hours of lectures<br><br>4 hours of laboratory sessions | • JavaScript's key fundamental features<br>• JavaScript data types and objects<br>***Learning Outcomes: 4*** |
| 7 | JavaScript - II | 1/12<br><br>2 hours of lectures<br><br>4 hours of laboratory sessions | • Understanding and working with DOM<br>• DOM Manipulation and Events<br>***Learning Outcomes: 4*** |
| 8 | jQuery | 1/12<br><br>2 hours of lectures<br><br>4 hours of laboratory sessions | • About jQuery and how jQuery works<br>• jQuery UI<br>• jQuery Events and Effects<br>***Learning Outcomes: 4*** |
| 9 | Responsive CSS Framework: Bootstrap - I | 1/12<br><br>2 hours of lectures<br><br>4 hours of laboratory sessions | • Bootstrap Installation<br>• Bootstrap containers<br>• Spinners, Cards<br>• Responsive grid system<br>***Learning Outcomes: 3*** |

| | | | |
|---|---|---|---|
| 10 | Responsive CSS Framework: Bootstrap - I | 1/12<br><br>2 hours of lectures<br><br>4 hours of laboratory sessions | • Advanced features of Bootstrap for responsive web pages<br>• Bootstrap form, form controls, buttons, utilities and Cards<br>• Bootstrap elements<br>***Learning Outcomes: 5*** |
| 11 | Code Review, Testing and Collaboration | 1/12<br><br>2 hours of lectures<br><br>4 hours of laboratory sessions | • Webapp testing<br>• SDLC<br>• SDLC models<br>• Version Control<br>***Learning Outcomes:6 and 7*** |
| 12 | Unit Summary | 1/12<br><br>2 hours of lectures<br><br>4 hours of laboratory sessions | • Summary and recap of previous units<br>***Learning Outcomes: All*** |

# 4. Related National Occupational Standards

The UK National Occupational Standards describe the skills that professionals are expected to demonstrate in their jobs in order to carry them out effectively. They are developed by employers and this information can be helpful in explaining the practical skills that students have covered in this module.

| **Related National Occupational Standards (NOS)** |
|---|
| **Sector Subject Area:** IT Users<br>**Related NOS:** TECHDUDC2, TECHDUDC3, TECHDUDM1 |

# 5. Resources

Lecturer Guide: This guide contains notes for lecturers on the organisation of each topic, and suggested use of the resources. It also contains all of the suggested exercises and model answers.

PowerPoint Slides: These are presented for each topic for use in the lectures. They contain many examples which can be used to explain the key concepts. Handout versions of the slides are also available; it is recommended that these are distributed to students for revision purposes as it is important that students learn to take their own notes during lectures.

Student Guide: This contains the topic overviews and all of the suggested exercises. Each student will need access to this and should bring it to all of the taught hours for the module.

## 5.1 Additional Hardware and Software Requirements

Hardware: Desktop computer or laptop with 16 GB RAM, 512 GB Hard Drive or larger.

Software:

1. **Code Editor**: Visual Studio Code (VS Code)
    - Free and supports HTML syntax highlighting, live preview, and other useful web development features.
2. **Web Browser**: Google Chrome or Mozilla Firefox
3. **jQuery Library**:

    Ensure students know how to include the jQuery library in their projects. This can be done by downloading the jQuery file and linking it locally or by linking to a CDN (Content Delivery Network) version of jQuery in the HTML file. For example, adding **<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>** in the **<head>** section of the HTML file.

4. **jQuery UI Library** (for the jQuery UI exercise):

    Similar to jQuery, include the jQuery UI library by linking to it in the HTML file. It also requires linking to the jQuery UI CSS file for styles. For instance, you can add **<link rel="stylesheet" href="//code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css">** and **<script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>** in the **<head>** section.

5. **Bootstrap Library**:
    - Ensure that students include the Bootstrap CSS file in their HTML documents. This can be done by linking to the Bootstrap CDN in the **<head>** section of the HTML file, like so:

    ```
    <link rel="stylesheet"
    href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
    ```
    - For Bootstrap 4 or whichever version is being taught. Ensure the version in the link is updated to match the version you intend to teach.

6. **Optional: jQuery and Popper.js** (for Bootstrap components that require JavaScript):

    - If the exercises extend to Bootstrap components that rely on JavaScript (like modals, dropdowns, etc.), including jQuery and Popper.js might be necessary. For Bootstrap 4, these can be included before the closing **</body>** tag in your HTML:

```
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
 <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.2/dist/umd/popper.min.js"></script>
 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
```

    - Note: Bootstrap 5 and newer versions do not require jQuery.

7. **Git**:

    - **Git Software**: Download and install Git from git-scm.com to enable version control on the student's machines.

    - **GitHub**: Students should create accounts on GitHub for online repository hosting and collaboration.

8. **Optional: Online References and Documentation**:

    - jQuery Official Website and Documentation (https://jquery.com/) - Essential for students to learn more about jQuery syntax, methods, and best practices.

    - jQuery UI Official Website (https://jqueryui.com/) - Useful for exploring more widgets and features available in jQuery UI beyond the Datepicker widget.

    - Bootstrap Official Documentation (https://getbootstrap.com/) - Critical for students to understand the framework, explore components, and learn about the grid system, utilities, and JavaScript plugins Bootstrap offers.

# 6. Pedagogic Approach

| Suggested Learning Hours | | | | | | |
|---|---|---|---|---|---|---|
| Guided Learning Hours | | | | Assessment | Private Study | Total |
| Lecture | Tutorial | Seminar | Laboratory | | | |
| 24 | - | - | 48 | 40 | 38 | 150 |

The teacher-led time for this module is comprised of lectures, laboratory sessions and tutorials. The breakdown of the hours is also given at the start of each topic, with 5 hours of contact time per topic.

### 6.1 Lectures

Lectures are designed to introduce students to each topic; PowerPoint slides are presented for use during these sessions. Students should also be encouraged to be active during this time and to discuss and/or practice the concepts covered. Lecturers should encourage active participation and field questions wherever possible.

### 6.2 Tutorials

Tutorials provide tasks to involve group work, investigation and independent learning for certain topics. The details of these tasks are provided in this guide and also in the Student Guide. They are also designed to deal with the questions arising from the lectures, laboratory sessions and private study sessions.

### 6.3 Laboratory Sessions

During these sessions, students are required to work through practical tutorials and various exercises. The details of these are provided in this guide and also in the Student Guide. Some sessions will require more support than others as well as IT resources. More detail is given in this guide.

### 6.4 Private Study

In addition to the taught portion of the module, students will also be expected to undertake private study. Exercises are provided in the Student Guide for students to complete during this time. Teachers will need to set deadlines for the completion of this work. These should ideally be before the tutorial session for each topic, when Private Study Exercises are usually reviewed.

# 7. Assessment

This module will be assessed by means of an assignment worth 100% of the total mark. These assessments will cover the learning outcomes and assessment criteria given above. Sample assessments are available through the NCC Education Virtual Learning Environment (http://vle.nccedu.com/login/index.php) for your reference.

# 8. Further Reading List

A selection of sources of further reading around the content of this module must be available in your Accredited Partner Centre's library. The following list provides suggestions of some suitable sources:

**Introduction to Front-End Web Development**

1. **"HTML and CSS: Design and Build Websites" by Jon Duckett**
    - This book provides a clear introduction to the basics of HTML and CSS, perfect for beginners. Its visual approach helps readers understand the concepts through examples and illustrations.
2. **"Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics" by Jennifer Robbins**
    - An excellent resource for beginners, this book covers the fundamentals of web design, including HTML, CSS, JavaScript, and web graphics, providing a solid foundation for front-end development.

**HTML, CSS**

1. **"HTML5: The Missing Manual" by Matthew MacDonald**
    - This manual covers the latest HTML5 standards and provides practical advice on building web pages that work across different devices.
2. **"CSS3: The Missing Manual" by David Sawyer McFarland**
    - A comprehensive guide to CSS3, offering explanations on how to use new features to create dynamic web pages.

**CSS Flexbox and Grid Layouts**

1. **"CSS Secrets: Better Solutions to Everyday Web Design Problems" by Lea Verou**
    - Focuses on practical tips and tricks, including how to best use Flexbox and Grid to solve common web design challenges.

2. **"A Complete Guide to Grid" by Rachel Andrew and "A Complete Guide to Flexbox" by Chris Coyier** (available online at CSS-Tricks)
    - These are not books but are considered essential readings for anyone wanting to master CSS Grid and Flexbox.

**JavaScript**

1. **"Eloquent JavaScript: A Modern Introduction to Programming" by Marijn Haverbeke**
    - Covers JavaScript from the basics to advanced topics, including programming fundamentals and real-world web development tasks.
2. **"JavaScript: The Definitive Guide" by David Flanagan**
    - A comprehensive resource for JavaScript developers, covering everything from basic syntax to advanced topics.

**jQuery**

1. **"jQuery: Novice to Ninja" by Earle Castledine and Craig Sharkie**
    - A practical guide to mastering jQuery, focusing on practical applications and techniques.

**CSS Bootstrap Framework**

1. **"Bootstrap 4 – Responsive Web Design" by Silvio Moreto, Matt Lambert, Benjamin Jakobus, and Jason Marah**
    - Covers Bootstrap 4 in detail, from installation to customization, along with tips on how to create responsive designs.

**Basics of Web Development Code Review and Testing**

1. **"Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability" by Steve Krug**
    - Although not solely about code review and testing, this book provides invaluable insights into web usability, which is crucial for effective web development and testing.
2. **"JavaScript Testing" by Jasmine Framework**
    - Focuses on using the Jasmine framework for testing JavaScript applications, covering basics to advanced testing techniques.

**Software Development Lifecycle**

1. **"Agile Web Development with Rails 6" by Sam Ruby, David Bryant Copeland, and Dave Thomas**
    - Provides insights into the Agile software development process using Rails as an example, though the principles can be applied more broadly.

**Version Control**

1. **"Pro Git" by Scott Chacon and Ben Straub**
    - An in-depth guide to using Git, from basic commands to advanced version control techniques.

# Topic 1: Overview of Web Application Architecture

## 1.1 Learning Objectives

This topic describes the concepts of website development and underlying technologies, key concepts and tools and technologies necessary for front-end development.

On completion of the topic, students will be able to:

- Describe the underlying technologies for building websites.
- Explain fundamental concepts of website design including the client-server model.
- Distinguish between front-end web development and back-end web development.
- Identify key concepts and technologies needed for front-end web development.

## 1.2 Pedagogic Approach

Information will be communicated to the students during the lectures. These are designed to be as interactive as possible. Students will undertake practical activities to develop skills and reinforce their understanding of the subjects introduced in the lecture. Finally private study and the subsequent tutorial will be used to consolidate learning and explore some aspects in greater detail.

## 1.3 Timings

Lectures: 2 hours

Private Study: 3 hours

Laboratory Sessions: 4 hours

## 1.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time and should be read in conjunction with the slides provided.

The structure of this topic is as follows:

- Overview of Internet and WWW
- Evolution of Website Development
- Web Application Architecture
- Types of Web Application Architecture
- Overview of Front-end Web Development

## 1.5 Laboratory Sessions

The laboratory time allocation for this topic is 4 hours.

These exercises are designed to be interactive and explorative, pushing students to apply critical thinking and observation skills to real-world web development practices. This approach helps bridge the gap between theoretical knowledge and practical understanding.

For students to effectively engage with and learn from the observational web development exercises described above, it's essential to provide them with a structured approach and clear guidance on how to execute these tasks. It might be more effective to encourage students to perform the exercises in groups.

**Hardware/Software requirements:**
For these exercises, students will need access to several different web browsers. At the time of writing, the following are recommended:

- Internet Explorer
- Mozilla Firefox
- Google Chrome
- Opera

Here are four revised lab exercises that focus on observation and analysis:

**1. Exploring Website Structures**

**Objective:** Analyse the structure and design of various websites to understand basic HTML elements and their purpose.

**Exercise Steps:**

- Visit a selection of websites, including a blog, an e-commerce site, a news portal, and an educational site.
- Observe and note the different sections of each website (e.g., header, navigation, content area, sidebar, footer).
- Use browser developer tools to inspect the HTML structure of these sections.
- Discuss how different HTML elements (like **<header>**, **<nav>**, **<article>**, **<aside>**, and **<footer>**) are used to structure web content.

**Learning Outcome:** Understand the common structural elements of web pages and their semantic significance.

**2. CSS and Theming Exploration**

**Objective:** Investigate how CSS is used to theme and style websites differently across various industries.

**Exercise Steps:**

- Select websites from different sectors (e.g., fashion, tech, food, sports).
- Compare and contrast the use of colours, fonts, and layout styles.
- Use the browser's developer tools to explore specific CSS rules applied to elements.
- Identify trends in styling for different types of websites (e.g., bold colours for entertainment sites, minimalistic designs for tech sites).

**Learning Outcome:** Recognise the role of CSS in creating visually distinct and thematic designs for websites.

**3. Responsiveness and Mobile Design**

**Objective:** Explore how websites adapt to different screen sizes and devices, focusing on responsive design techniques.

**Exercise Steps:**

- Visit a variety of websites and manually resize the browser window to observe changes in layout and functionality.
- Use the developer tools' device simulation feature to view websites on different simulated devices (e.g., tablets, smartphones).
- Note which elements adjust for small screens (like navigation menus transforming into hamburger menus) and how images or text blocks are resized.
- Reflect on the importance of responsive design for user experience across devices.

**Learning Outcome:** Gain insight into how responsive design principles is applied to ensure websites are functional and visually appealing on any device.

**4. Cross-Browser Compatibility Investigation**

**Objective:** Understand the challenges and importance of cross-browser compatibility in web design.

**Exercise Steps:**

- Choose a set of websites and open them in different web browsers (e.g., Chrome, Firefox, Safari, Edge).
- Identify any discrepancies in appearance or functionality across browsers.
- Explore how modern web standards and practices, such as using CSS prefixes and fallbacks, contribute to a consistent user experience.

- Discuss the findings and consider the impact of browser compatibility on web development strategies.

**Learning Outcome:** Learn about the variability in web page rendering across browsers and the techniques used to address these challenges.

**Additional Notes:**

- Encourage students to take screenshots or notes on their observations to share and discuss with peers.
- Facilitate discussions on the importance of web standards, accessibility, and user-centric design principles.

### 1.6 Private Study

The time allocation for private study in this topic is expected to be 3 hours.

Review the lecture slides and do any additional reading necessary to make sure you understand and feel confident about the content. Much of the practical work we will do in future topics will assume you are familiar with the basic challenges of designing websites.

**1. Article Reviews on Web Structure**
**Activity:** Students are to find and read articles on the importance of web structure and semantic HTML. They should look for content that discusses the role of different HTML elements in creating accessible and SEO-friendly websites.

**2. CSS Styling Trends Analysis**
**Activity:** Students should search for online tutorials or articles on current CSS styling trends, such as dark mode, glassmorphism, or minimalism. They can also explore websites like Awwwards or CSS Design Awards for real-world examples of these trends.

**3. Responsive Design Case Studies**
**Activity:** Identify websites known for their excellent responsive design. Students can use resources like media queries galleries or design showcases that highlight responsive web design examples.

**4. Browser Compatibility Research**
**Activity:** Research the common issues developers face with browser compatibility today. This can include reading documentation from web development communities, articles, or forums that discuss specific problems and solutions.
**Additional Private Study Activities:**

- **Technical Blogs and Podcasts:** Encourage students to follow blogs, podcasts, and YouTube channels focused on web development. This can help them stay updated on the latest trends, tools, and best practices in the industry.
- **Online Courses and Tutorials:** Recommend students to take advantage of free or paid online courses and tutorials that cover HTML, CSS, and responsive web design. Platforms like

Coursera, edX, FreeCodeCamp, and Codecademy offer structured learning paths from beginner to advanced levels.

- **Practice Exercises:** Utilize platforms like CodePen or JSFiddle to practice HTML and CSS by trying to recreate the layout of popular websites or by experimenting with different design concepts.
- **Join Web Development Communities:** Encourage participation in online forums and communities such as Stack Overflow, Reddit's web development subreddits, or specific web development Discord servers. Engaging in these communities can provide peer support, feedback, and insights into real-world web development challenges and solutions.

# Topic 2: Introduction to HTML5

## 2.1 Learning Objectives

This topic introduces the basic concepts of website development and underlying technologies, key concepts and tools and technologies necessary for front-end development using HTML.

On completion of the topic, students will be able to:

- Describe the use of HTML to develop websites
- Understand the basics of HTML5
- Explain the HTML structure
- Demonstrate understanding of HTML basics

## 2.2 Pedagogic Approach

Information will be communicated to the students during the lectures. These are designed to be as interactive as possible. Students will undertake practical activities to develop skills and reinforce their understanding of the subjects introduced in the lecture. Finally private study will be used to consolidate learning and explore some aspects in greater detail.

## 2.3 Timings

| | |
|---|---|
| Lectures: | 2 hours |
| Laboratory Sessions: | 4 hours |
| Private Study: | 3 hours |

## 2.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time and should be read in conjunction with the slides provided.

The structure of this topic is as follows:

- Introduction to HTML
- Role of HTML
- HTML Document Structure
- Text-level Elements
- HTML Lists
- HTML Forms
- HTML Best Practices

## 2.5 Laboratory Sessions

The laboratory time allocation for this topic is 4 hours.

**Hardware/Software requirements:**

1. **Code Editor**: Visual Studio Code (VS Code)
    - Free and supports HTML syntax highlighting, live preview, and other useful web development features.
2. **Web Browser**: Google Chrome or Mozilla Firefox

**General instructions for VS code installation and running HTML, CSS, and Javascript:**
**Installing Visual Studio Code**

1. **Download VS Code**:
    - Go to the Visual Studio Code website and download the installer for your operating system (Windows, macOS, or Linux).
2. **Install VS Code**:
    - Run the downloaded installer and follow the on-screen instructions to complete the installation.
3. **Launch VS Code**:
    - Open VS Code from your applications or programs menu.

**Running HTML, CSS, and JavaScript Code**

1. **Create a New File**:
    - In VS Code, go to **File** > **New File** or use the shortcut **Ctrl + N** (Windows/Linux) or **Cmd + N** (macOS) to create a new file.
2. **Write Your Code**:
    - For HTML: Start with a basic HTML structure, then save the file with a **.html** extension (e.g., **index.html**).
    - For CSS: Write your CSS styles, then save the file with a **.css** extension (e.g., **styles.css**). Link this CSS file in your HTML document within the **<head>** section using **<link rel="stylesheet" href="styles.css">**.
    - For JavaScript: Write your JavaScript code, then save the file with a **.js** extension (e.g., **script.js**). Link this JS file in your HTML document before the closing **</body>** tag using **<script src="script.js"></script>**.
3. **Preview Your HTML File**:
    - To view your HTML file in a web browser, right-click the file in VS Code and select **Open with Live Server**. If you don't see this option, you may need to install the Live Server extension:

- Go to the Extensions view by clicking on the square icon on the side menu or pressing **Ctrl+Shift+X** (Windows/Linux) or **Cmd+Shift+X** (macOS).
- Search for "Live Server" and click **Install** on the extension by Ritwick Dey.
- Once installed, you can right-click your HTML file and select **Open with Live Server** to view it in your default web browser.
- Live Server provides a live preview of your HTML file, and it will automatically reload the page when you make changes to your HTML, CSS, or JavaScript files.

4. **View Output in a Web Browser**:
    - Your default web browser should open, displaying the HTML file. You can inspect the elements, view the console, and debug JavaScript by right-clicking on the page and selecting **Inspect**.

Here are four lab exercises:

**1. HTML Document Structure**

**Objective:** To understand the basic structure of an HTML document, including the <!DOCTYPE html>, <html>, <head>, <title>, and <body> tags.

**Exercise:** Create a simple HTML page about a favourite animal.

- Task 1: Start by declaring the document type and setting up the basic structure of an HTML page.
- Task 2: In the <head> section, add a <title> that says "My Favourite Animal."
- Task 3: In the <body>, include an <h1> tag with the name of the animal, followed by a paragraph (<p>) describing why it is your favourite.
- Extension: Add an image of the animal using the <img> tag (you can use a placeholder image URL).

**2. Text-level Elements**

**Objective:** Learn to use text-level HTML elements for emphasizing text, creating links, and styling.

**Exercise:** Write a short biography of a famous scientist.

- Task 1: Use <p> tags to write two paragraphs about the scientist's life and achievements.
- Task 2: Emphasize key words or phrases using  and .
- Task 3: Include a link to a website where students can learn more about the scientist using the <a> tag.
- Extension: Use the <blockquote> tag to add a famous quote by the scientist.

### 3. HTML Lists

**Objective:** Understand how to create ordered (<ol>), unordered (<ul>), and description lists (<dl>) in HTML.

**Exercise:** Create a webpage that lists your top 5 favourite books, movies, or video games.

- Task 1: Use an <h2> tag to title each list (Books, Movies, Video Games).
- Task 2: For books and movies, use an unordered list (<ul>) to list the titles.
- Task 3: For video games, use an ordered list (<ol>) to rank them from most to least favourite.
- Extension: Use a description list (<dl>) to provide a brief description of each item in one of the lists.

### 4. HTML Forms

**Objective:** Learn to create a basic HTML form with different types of input fields.

**Exercise:** Create a feedback form for a class website.

- Task 1: Use <form> tags to create the form structure. Include input fields for name (type="text"), email (type="email"), and a dropdown to rate the class (<select> with options from 1 to 5).
- Task 2: Add a textarea for comments.
- Task 3: Include a submit button with the label "Submit Feedback."
- Extension: Use the <fieldset> and <legend> tags to group related elements in the form, such as personal information and feedback.

## 2.6 Private Study

The time allocation for private study in this topic is expected to be 3 hours.

To complement the classroom lab exercises and enhance the students' understanding of front-end web development, particularly HTML, here are recommended private study activities. These activities are designed to reinforce the concepts learned in the lab and encourage independent exploration and learning.

**1. HTML Document Structure**

**Activity:** Explore different **<!DOCTYPE>** declarations and their effects on HTML documents. Research the history of HTML and how it has evolved over time. Create a simple webpage that includes all major structural elements (**<header>**, **<footer>**, **<nav>**, **<article>**, and **<section>**) and experiment with their layouts.

**Resources:** Use online tutorials and documentation like http://www.w3schools.com/ to understand the semantics and best practices of using these elements.

**2. Text-level Elements**

**Activity:** Write a short blog post in HTML, making use of various text-level semantic tags (****, ****, **<mark>**, **<cite>**, **<blockquote>**, etc.). Focus on appropriately using each tag to enhance the readability and SEO of your content.

**Resources:** Look for articles on best practices for semantic HTML and SEO. Experiment with different tags and validate your HTML using online validators to ensure it's well-formed.

**3. HTML Lists**

**Activity:** Create an interactive to-do list HTML document. Structure the document with **<ul>** or **<ol>** for tasks. Include a form with input fields for adding new tasks to the list. While the interaction might require JavaScript, focus on structuring your HTML to support future functionality.

**Resources:** Review tutorials on form handling and list manipulation in HTML and JavaScript. This activity prepares you for understanding how HTML forms the basis for dynamic web content.

**4. HTML Forms**

**Activity:** Design a more complex form that could be used for a survey or a registration page for an event. Include various types of input fields (**<input type="text">**, **<input type="radio">**, **<input type="checkbox">**, **<select>**, etc.). Pay special attention to creating a user-friendly and accessible form.

# Topic 3: Introduction to CSS

## 3.1 Learning Objectives

This topic introduces the basic concepts of CSS programming language and provides an overview of some of the CSS concepts including CSS backgrounds, borders, margins, padding, front styling, gradients and CSS box models.

On completion of the topic, students will be able to:

- Describe the role of CSS in front end web development
- Understand the syntax used in CSS coding
- Explain concepts of CSS Class
- Differentiate between CSS stylesheets
- Identify the box structure of CSS

## 3.2 Pedagogic Approach

Information will be communicated to the students during the lectures. These are designed to be as interactive as possible. Students will undertake practical activities to develop skills and reinforce their understanding of the subjects introduced in the lecture. Finally private study and the subsequent tutorial will be used to consolidate learning and explore some aspects in greater detail.

## 3.3 Timings

| | |
|---|---|
| Lectures: | 2 hours |
| Laboratory Sessions: | 4 hours |
| Private Study: | 3 hours |

## 3.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time and should be read in conjunction with the slides provided.

The structure of this topic is as follows:

- CSS Definition
- CSS Syntax
- CSS Class
- CSS Style Sheets
- CSS Background
- CSS Box model

### 3.5 Laboratory Sessions

The laboratory time allocation for this topic is 4 hours.

**Hardware/Software requirements:**

1. **Code Editor**: Visual Studio Code (VS Code)
    - Free and supports HTML syntax highlighting, live preview, and other useful web development features.
2. **Web Browser**: Google Chrome or Mozilla Firefox

Here are four lab exercises:

**1. CSS Syntax: ID, Class, Selector, and Comment**

**Objective:** Familiarise students with basic CSS syntax, including how to use IDs, classes, selectors, and comments.

**Exercise:**

- Create an HTML document with various elements (divs, paragraphs, headers).
- Apply styles using ID selectors to individual elements.
- Use class selectors to style multiple elements with common styles.
- Utilize element selectors to define general styles for specific HTML tags.
- Include comments in the CSS file to describe the purpose of different style sections.
- **Task:** Students will create a simple webpage layout using divs and style it is using IDs and classes. They must use comments to explain their choice of styling.

**2. CSS Background and Border**

**Objective:** Teach students how to manipulate the background and border properties of web page elements.

**Exercise:**

- Provide a basic HTML template with multiple div elements.
- Guide students to apply different background colours, images, and gradients to these elements.
- Show how to adjust the border properties (width, style, colour) of various elements.
- Introduce the concept of border-radius for creating rounded corners.
- **Task:** Students will design a "card" for a character of their choice, setting a background image or colour, and customiing the border to make its visually appealing.

### 3. CSS Text Formatting

**Objective:** Explore various CSS properties to format text, including alignment, spacing, and decoration.

**Exercise:**

- Start with an HTML document filled with paragraphs, headers, and lists.
- Apply text-align to demonstrate different alignment options (left, right, center, justify).
- Use text-decoration to underline, overline, or strike-through text.
- Experiment with line-height and letter-spacing to adjust the readability of text.
- **Task:** Students will format a short biography or introduction paragraph for a fictional character, focusing on making the text visually appealing and easy to read.

### 4. CSS Fonts and Colors

**Objective:** Introduce students to font styling and color schemes in CSS.

**Exercise:**

- Provide an HTML document with text content, including headers, paragraphs, and links.
- Teach students how to embed and use web fonts from services like Google Fonts.
- Show how to apply font properties such as font-family, font-size, font-weight, and font-style.
- Experiment with colour and background-color properties to enhance text readability and aesthetic appeal.
- **Task:** Each student will create a themed "event invitation" (e.g., a birthday party, a science fair) using custom fonts and a harmonious color scheme. They must ensure the text is both attractive and readable.

### 3.6 Private Study

The time allocation for private study in this topic is expected to be 3 hours.

To complement the classroom lab exercises and enhance the students' understanding of front-end web development, particularly HTML, here are recommended private study activities. These activities are designed to reinforce the concepts learned in the lab and encourage independent exploration and for students looking to deepen their understanding of front-end web development, particularly in CSS, private study can significantly complement what they learn in the classroom.

Here are some resources and strategies for each of the topics covered:

**1. CSS Syntax: ID, Class, Selector, and Comment**

- **MDN Web Docs on CSS:** The Mozilla Developer Network (MDN) offers comprehensive guides and reference materials on CSS. Start with their CSS basics to understand how CSS works, then move on to more specific topics like selectors and syntax.
- **W3Schools CSS Tutorial:** W3Schools provides straightforward tutorials and examples that are perfect for beginners. Their sections on selectors, classes, and IDs are particularly relevant.
- **Practice on Codecademy:** Codecademy offers interactive CSS courses that include exercises on syntax, selectors, and best practices.
- **Book Recommendation:** "CSS: The Definitive Guide" by Eric Meyer provides an in-depth look at CSS, including selectors and structure.

**2. CSS Background and Border**

- **CSS Tricks:** This site is a treasure trove of tips and tricks for CSS, including detailed articles on background and border properties.
- **Frontend Mentor Challenges:** Engage with real-world HTML & CSS challenges that range in difficulty, focusing on applying background and border styles.
- **Kevin Powell's YouTube Channel:** Kevin Powell has many tutorials focused on CSS, including video guides on backgrounds, borders, and box models which are easy to follow and implement.

**3. CSS Text Formatting**

- **MDN Web Docs:** MDN's sections on text and font styling offer detailed explanations and examples. Pay special attention to the documentation on text formatting properties.
- **CSS-Tricks Almanac:** Look up properties related to text, like **text-align**, **line-height**, and **letter-spacing**, to see varied examples and use cases.
- **Experiment on CodePen:** Use CodePen to practice different text formatting techniques and see their immediate impact on the design.

**4. CSS Fonts and Colors**

- **Google Fonts:** Explore Google Fonts to find and experiment with different web fonts. The site also provides guidance on how to incorporate these fonts into your projects.
- **Color Hunt:** A great tool for finding and experimenting with color schemes. Understanding how to choose and combine colors is crucial for front-end design.
- **Adobe Color:** Use Adobe Color to learn about color theory and generate color schemes that can be used in your projects.

# Topic 4: HTML and CSS

## 4.1 Learning Objectives

This topic discusses how HTML and CSS are used in conjunction for front end web development and introduces the basic concepts of DIV and SPAN elements, CSS selectors, and coding with HTML and CSS.

On completion of the topic, students will be able to:

- Explain the concepts of SPAN and DIV element in HTML and CSS.
- Differentiate between SPAN and DIV element.
- Identify different type of CSS selectors.
- Demonstrate the ability to write code for front end web development using HTML and CSS.

## 4.2 Pedagogic Approach

Information will be communicated to the students during the lectures. These are designed to be as interactive as possible. Students will undertake practical activities to develop skills and reinforce their understanding of the subjects introduced in the lecture. Finally private study will be used to consolidate learning and explore some aspects in greater detail.

## 4.3 Timings

Lectures: 2 hours

Private Study: 3 hours

Laboratory: 4 hours

## 4.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time and should be read in conjunction with the slides provided.

The structure of this topic is as follows:

- DIV element
- SPAN element
- CSS Selectors
- Coding with HTML and CSS

## 4.5 Laboratory Sessions

The laboratory time allocation for this topic is 4 hours.

These exercises are designed to be observational and analytical, requiring students to engage critically with real-world web pages to understand the practical applications of web development concepts.

Each of these exercises should be hands-on and encourage creativity, while also reinforcing the technical skills needed for front-end web development. Students should be encouraged to experiment with styles and structures beyond the basics outlined in the exercises.

**Hardware/Software requirements:**

1. **Code Editor**: Visual Studio Code (VS Code)
    - Free and supports HTML syntax highlighting, live preview, and other useful web development features.
2. **Web Browser**: Google Chrome or Mozilla Firefox

### 1. DIV Element

**Objective:** Understand the block-level behaviour of the ** ** element and practice grouping content.

**Exercise:**

- Create a simple webpage with a header that says "My Webpage".
- Below the header, use ** ** elements to create three separate sections: "About Me", "My Hobbies", and "Contact Information".
- Inside each ** **, add a paragraph describing each section. For example, "About Me" could have a short bio, "My Hobbies" could list favourite pastimes, and "Contact Information" could contain fictional contact details.
- Style each ** ** with a different background colour using inline CSS.

**Expected Outcome:** Students will see how ** ** elements stack vertically and can be used to organize content on a webpage.

### 2. SPAN Element

**Objective:** Learn the inline behaviour of the **** element and how it can be used to style text.

**Exercise:**

- Create a webpage with a descriptive paragraph about an interesting animal.
- Within the paragraph, use **** elements to highlight scientific names, important facts, or keywords by changing their color or font weight.
- Example: "The Panthera leo (lion) is known for its majestic mane and its status as a 'king of the jungle'."

**Expected Outcome:** Students will understand how **** elements can be used within block elements to apply styles to specific text portions without breaking the flow.

**3. CSS Selectors**

**Objective:** Practice using different types of CSS selectors to apply styles to HTML elements.

**Exercise:**

- Create an HTML document with a variety of elements: headers, paragraphs, an unordered list, and a table.
- Write a CSS stylesheet that:
    - Changes the text colour of all paragraphs using a type selector.
    - Applies a unique background to the unordered list using an ID selector.
    - Highlights one of the table rows using a class selector.
    - Bonus: Use a pseudo-class to change the colour of a link when it is hovered over.

**Expected Outcome:** Students will gain hands-on experience targeting HTML elements with CSS selectors and understand the specificity hierarchy.

**4. Coding with HTML and CSS**

**Objective:** Combine HTML and CSS to create a small project that incorporates both structure and design.

**Exercise:**

- Task the students to create a personal profile page that includes a photo, a brief introduction, a list of skills, and a mini-portfolio of projects or artwork.
- Use HTML to structure the content into appropriate sections.
- Use CSS to style the page, including layout techniques like Flexbox or Grid, and add visual appeal with colours, fonts, and spacing.
- Encourage creativity in the design but ensure the layout is responsive and maintains usability.

**Expected Outcome:** Students will demonstrate their understanding of HTML and CSS by building a complete webpage from scratch, reinforcing their skills in both coding and design.

### 4.6 Private Study

The time allocation for private study in this topic is expected to be 3 hours.

Recommendations for Private Study on HTML and CSS.

1. DIV Element
   - **Study Resources**: Look for tutorials that explain the box model, which is fundamental to understanding how **div** elements are styled and laid out on the page.
   - **Practical Tips**: Use developer tools in browsers like Chrome or Firefox to inspect **div** elements on various websites to see how they are used in real-world layouts.
   - **Project Ideas**: Create a simple web page layout using only **div** elements for structuring different areas like the header, footer, sidebar, and main content area.

2. SPAN Element
   - **Study Resources**: Find articles or videos that explain inline versus block elements. Pay particular attention to examples that use **span** for styling and semantic purposes.
   - **Practical Tips**: Practice by trying to style specific words or phrases in a paragraph without affecting the rest of the content. Use **span** elements to apply classes and inline styles.
   - **Project Ideas**: Make a mock-up of a blog post and use **span** elements to highlight certain words for emphasis or to display icons next to specific types of content.

3. CSS Selectors
   - **Study Resources**: CSS Tricks, Mozilla Developer Network (MDN), and W3Schools have excellent references on selectors. Focus on understanding the difference between class, ID, and attribute selectors.
   - **Practical Tips**: Experiment with different selectors by styling a simple HTML document. Practice using combinators like child (**>**), descendant (space), and adjacent sibling (**+**).
   - **Project Ideas**: Create a styled form and use attribute selectors to apply styles to different form elements based on their type (e.g., **input[type="text"]**).

4. Coding with HTML and CSS
   - **Study Resources**: Use online courses from platforms like Codecademy, freeCodeCamp, or Coursera to get a structured learning path on HTML and CSS.
   - **Practical Tips**: Practice building small web components, like a navigation bar or a footer, and gradually work your way up to full web pages.
   - **Project Ideas**: Try replicating a simple page from your favourite website using HTML and CSS only. Focus on understanding layout techniques like Flexbox and Grid.

# Topic 5: CSS Flexbox and Grid Layout

## 5.1 Learning Objectives

This topic introduces the concepts of CSS flexbox and grid layout. It provides an overview of some of the CSS grid related concepts including CSS Grid layout, flexbox, flex container and items, and its comparison.

On completion of the topic, students will be able to:

- Create and design responsive website using Flexbox and Grid Layout.
- Use CSS3 layout model: Flexbox for responsive and mobile-friendly design.
- Explain the difference between CSS Grid Layout and CSS Flexbox Layout.

## 5.2 Pedagogic Approach

Information will be communicated to the students during the lectures. These are designed to be as interactive as possible. Students will undertake practical activities to develop skills and reinforce their understanding of the subjects introduced in the lecture. Finally private study and the subsequent tutorial will be used to consolidate learning and explore some aspects in greater detail.

## 5.3 Timings

| | |
|---|---|
| Lectures: | 2 hours |
| Laboratory Sessions: | 4 hours |
| Private Study: | 3 hours |

## 5.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time and should be read in conjunction with the slides provided.

The structure of this topic is as follows:

- CSS Grid
- CSS Grid Layout
- CSS Flexbox Layout

## 5.5 Laboratory Sessions

The laboratory time allocation for this topic is 4 hours.

These exercises are designed to be observational and analytical, requiring students to engage critically with real-world web pages to understand the practical applications of web development concepts. These exercises provide a practical introduction to CSS Grid and Flexbox layouts, enabling students to experiment with and understand the foundational concepts of CSS layout techniques.

**Hardware/Software requirements:**

1. **Code Editor**: Visual Studio Code (VS Code)
    - Free and supports HTML/CSS syntax highlighting, live preview, and other useful web development features.
2. **Web Browser**: Google Chrome or Mozilla Firefox

**Exercise 1: CSS Grid**

**Objective:** Create a simple 3x3 grid layout using CSS Grid.

**Instructions:**

1. Create an HTML file with a ** ** element that has a class of "grid-container".
2. Inside the "grid-container", add nine ** ** elements, each with a class of "grid-item" and numbered from 1 to 9.
3. In your CSS file, select the "grid-container" class and apply the following styles:
    - Display property set to grid.
    - Define grid-template-columns and grid-template-rows to have three columns and three rows of equal size.
4. Style the "grid-item" class to have a border, some padding, and centre the text inside each grid item.

**Expected Outcome:** Students will create a 3x3 grid where each cell contains a number from 1 to 9, demonstrating the basic use of CSS Grid layout.

**Exercise 2: CSS Grid Layout with Column and Row Span**

**Objective:** Modify the previous 3x3 grid layout so that one of the items spans two columns and another item spans two rows.

**Instructions:**

1. Using the grid created in Exercise 1, select two "grid-item" elements to modify.
2. For the first selected item, apply a grid-column span to make it span across two columns.
3. For the second selected item, apply a grid-row span to make it span across two rows.
4. Adjust the grid-template-columns and grid-template-rows if needed to accommodate the spanning items.

**Expected Outcome:** Students will understand how to make grid items span multiple columns or rows, showcasing the flexibility of CSS Grid layout.

**Exercise 3: Basic CSS Flexbox Layout**

**Objective:** Create a simple navigation bar using CSS Flexbox.

**Instructions:**

1. Create an HTML file with a **<nav>** element containing an unordered list **<ul>**. Inside the **<ul>**, add four **<li>** elements representing navigation links (e.g., Home, About, Services, Contact).
2. In your CSS file, select the **<ul>** element and apply the following styles:
    - Display property set to flex.
    - Justify-content to space-around to evenly space the navigation links.
3. Style the **<li>** elements to display inline and add some padding for visual separation.

**Expected Outcome:** Students will create a simple, horizontally aligned navigation bar using the flexbox layout, demonstrating the basic use of CSS Flexbox for aligning items in a container.

**Exercise 4: CSS Flexbox Layout with Alignment**

**Objective:** Modify the navigation bar created in Exercise 3 to align the items vertically and center them within the container.

**Instructions:**

1. Using the navigation bar from Exercise 3, change the flex-direction to column in your CSS file for the **<ul>** element.
2. Apply align-items to center to align the navigation links in the center of the **<nav>** container.
3. Adjust padding and margin as needed to ensure the navigation links are visually appealing and centered.

**Expected Outcome:** Students will learn how to change the orientation of flex items and align them vertically, utilising the flex-direction and align-items properties in CSS Flexbox.

## 5.6 Private Study

The time allocation for private study in this topic is expected to be 3 hours.

To enhance understanding and skills in CSS Grid and Flexbox layouts, students can engage in private study through a variety of resources and practices. Here are some recommendations to facilitate learning and mastery of these powerful CSS layout techniques.

**Online Tutorials and Courses**

1. **MDN Web Docs (Mozilla Developer Network):** Start with the CSS Grid and Flexbox guides provided by MDN. These are comprehensive resources that explain the concepts from the ground up, complete with examples and interactive demos.
    - **CSS Grid:** MDN CSS Grid Layout Guide
    - **CSS Flexbox:** MDN CSS Flexible Box Layout Guide
2. **CSS Tricks:** CSS Tricks offers excellent guides that visually explain both Flexbox and Grid. The Flexbox guide, in particular, is famous for its simplicity and effectiveness.
    - **A Complete Guide to Flexbox:** CSS-Tricks Flexbox Guide
    - **A Complete Guide to Grid:** CSS-Tricks Grid Guide
3. **FreeCodeCamp:** FreeCodeCamp offers an interactive learning experience where you can write code and see the results immediately. It has tutorials and projects specifically for CSS Grid and Flexbox.
    - Website: FreeCodeCamp
4. **Codecademy:** Codecademy provides hands-on CSS courses that include lessons on Flexbox and Grid, allowing you to apply what you learn in practice projects.
    - Website: Codecademy Web Development Path

**Practice Platforms**

1. **Flexbox Froggy & Grid Garden:** These are interactive web games designed to teach you Flexbox and Grid in a fun and engaging way.
    - **Flexbox Froggy:** Flexbox Froggy
    - **Grid Garden:** Grid Garden
2. **CodePen & JSFiddle:** Use these online code editors to practice your CSS Grid and Flexbox skills. You can start from scratch or fork existing projects to modify and understand how different properties affect layout.
    - CodePen
    - JSFiddle

# Topic 6: JavaScript (1)

## 6.1 Learning Objectives

This topic introduces the concepts of JavaScript. It provides an overview of some of the basic concepts of JavaScript including topics such as fundamental features, data types, and related features.

On completion of the topic, students will be able to:

- Gain an understanding of JavaScript programming language.
- Build interactive webpages and dynamic webpages using JavaScript.
- Write and apply JavaScript programming skills for building dynamic and interactive sites.
- Explain the different concepts of JavaScript including data types, variables, operators, and functions.

## 6.2 Pedagogic Approach

Information will be communicated to the students during the lectures. These are designed to be as interactive as possible. Students will undertake practical activities to develop skills and reinforce their understanding of the subjects introduced in the lecture. Finally private study l will be used to consolidate learning and explore some aspects in greater detail.

## 6.3 Timings

| | |
|---|---|
| Lectures: | 2 hours |
| Laboratory Sessions: | 4 hours |
| Private Study: | 3 hours |

## 6.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- JavaScript introduction
- Data types
- Variable
- Operators
- Functions
- Popup boxes
- Objects

## 6.5 Laboratory Sessions

The laboratory time allowance for tutorials in this topic is 4 hours.

These exercises are designed to be interactive and explorative, pushing students to apply critical thinking and observation skills to real-world web development practices. This approach helps bridge the gap between theoretical knowledge and practical understanding.

For students to effectively engage with and learn from the observational web development exercises described below, it's essential to provide them with a structured approach and clear guidance on how to execute these tasks. It might be more effective to encourage students to perform the exercises in groups.

**Hardware/Software requirements:**

1. **Code Editor**: Visual Studio Code (VS Code)
    - Highly recommended for JavaScript development due to its integrated support for JavaScript editing, debugging, IntelliSense (code completion), and extensions for further enhancing JavaScript coding. The live preview feature can also be beneficial for immediate feedback while coding.
2. **Web Browser**: Google Chrome or Mozilla Firefox
    - Essential for testing and viewing the output of JavaScript code. Both Chrome and Firefox offer powerful Developer Tools, which include a JavaScript console to view the output of **console.log()** statements, inspect variables, and debug scripts. The element inspector is also useful for understanding how JavaScript interacts with HTML elements.

### 1. JavaScript Data Types

**Objective:** Understand and work with different data types in JavaScript including strings, numbers, and Booleans.

**Exercise:**

1. Create a new HTML file and link a JavaScript file to it.
2. In the JavaScript file, declare three variables: one with a string value, one with a number value, and one with a Boolean value.
3. Use **console.log()** to print each variable's value and its type. To get the type, use **typeof** operator.
4. View the output in the browser's console.

**Expected Outcome:** Students will learn how to declare variables of different data types and how to check the type of a variable.

### 2. JavaScript Variables and Operators

**Objective:** Learn to declare variables and use basic arithmetic operators.

**Exercise:**

1. Declare two variables and assign them numerical values.
2. Perform addition, subtraction, multiplication, and division operations on those variables.
3. Store the result of each operation in a new variable.
4. Use **console.log()** to print the results of these operations.
5. Bonus: Try using the increment and decrement operators and print the results.

**Expected Outcome:** Students will understand how to use variables to store data and perform and display basic arithmetic operations.

### 3. JavaScript Function

**Objective:** Create and call a simple JavaScript function that performs a calculation.

**Exercise:**

1. Write a function named **calculateArea** that takes two parameters: **length** and **width**.
2. Inside the function, calculate the area of a rectangle using the formula **area = length * width**.
3. Return the area from the function.
4. Call the function with different sets of values and use **console.log()** to print the results of these calls.

**Expected Outcome:** Students will learn how to declare functions, pass parameters, return a value, and call functions in JavaScript.

### 4. Pop-up Box Using JavaScript

**Objective:** Use JavaScript to create and handle a simple pop-up box.

**Exercise:**

1. Create a button in your HTML file with the text "Click Me!".
2. In your JavaScript file, write a function called **showAlert** that uses the **alert()** function to show a pop-up box with the message "Hello, World!".
3. Add an event listener to the button so that when it is clicked, the **showAlert** function is called.
4. Test the button in your browser to see if the pop-up appears.

**Expected Outcome:** Students will learn how to interact with the DOM using JavaScript, create a function that triggers a pop-up alert, and understand event handling through button clicks.

## 6.6 Private Study

The time allocation for private study in this topic is expected to be 3 hours.

For students looking to deepen their understanding of front-end web development, particularly in the areas covered by the lab exercises (JavaScript data types, variables and operators, functions, and pop-up boxes), self-study can be a highly effective way to reinforce learning.

Here are some recommendations for resources and study methods that can help:

**1. Online Tutorials and Courses**

- **MDN Web Docs (Mozilla Developer Network):** MDN provides comprehensive and beginner-friendly documentation on JavaScript. It's a great place to start for understanding the fundamentals, including data types, variables, functions, and DOM manipulation. MDN JavaScript
- **freeCodeCamp:** Offers an interactive JavaScript curriculum that covers everything from the basics to more advanced topics. It's hands-on and includes exercises similar to the lab exercises provided. freeCodeCamp
- **Codecademy:** Offers interactive programming courses for various languages, including JavaScript. The courses range from beginner to more advanced topics, making it a good resource for deepening your understanding. Codecademy JavaScript Course

**2. Books**

- **"Eloquent JavaScript" by Marijn Haverbeke:** This is a book that not only teaches JavaScript but does so in a manner that encourages good programming practices. It starts with the basics and moves on to more complex topics, including programming style, project building, and Node.js. The book is available online for free. Eloquent JavaScript
- **"JavaScript: The Good Parts" by Douglas Crockford:** For students looking to understand the core of JavaScript, this book focuses on the best practices and patterns. It's more suited for students who have grasped the basics and are looking to refine their knowledge.

**3. Practice Platforms**

- **LeetCode and HackerRank:** While these platforms are often used for interview preparation, they are also great for practicing JavaScript coding challenges. They offer problems ranging from easy to difficult, which can help solidify understanding of JavaScript fundamentals through practical application.
- **CodePen and JSFiddle:** These are online code editors that allow you to write HTML, CSS, and JavaScript code directly in your browser and see the results instantly. They're great for experimenting with code and practicing the creation of pop-up boxes and other DOM manipulations.

# Topic 7: JavaScript II

## 7.1 Learning Objectives

This topic introduces the advanced concepts of JavaScript. It provides an overview of some of the advanced concepts of JavaScript including topics such as DOM, DOM manipulation and Events.

On completion of the topic, students will be able to:

- Gain an understanding of DOM concepts.
- Explain DOM elements and attributes.
- Describe event listening concepts.
- Write and apply programming skills for DOM and JS events.

## 7.2 Pedagogic Approach

Information will be communicated to the students during the lectures. These are designed to be as interactive as possible. Students will undertake practical activities to develop skills and reinforce their understanding of the subjects introduced in the lecture. Finally private study and the subsequent tutorial will be used to consolidate learning and explore some aspects in greater detail.

## 7.3 Timings

| | |
|---|---|
| Lectures: | 2 hours |
| Laboratory Sessions: | 4 hours |
| Private Study: | 3 hours |

## 7.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- DOM introduction
- DOM elements
- DOM attributes
- JS events

## 7.5 Laboratory Sessions

The laboratory time allocation for this topic is 4 hours.

These exercises are designed to be observational and analytical, requiring students to engage critically with real-world web pages to understand the practical applications of web development concepts. These exercises cover basic manipulations and interactions with the DOM, providing a practical foundation for further exploration of front-end web development concepts.

Hardware/Software requirements:

1. **Code Editor**: Visual Studio Code (VS Code)
    - Highly recommended for JavaScript development due to its integrated support for JavaScript editing, debugging, IntelliSense (code completion), and extensions for further enhancing JavaScript coding. The live preview feature can also be beneficial for immediate feedback while coding.
2. **Web Browser**: Google Chrome or Mozilla Firefox
    - Essential for testing and viewing the output of JavaScript code. Both Chrome and Firefox offer powerful Developer Tools, which include a JavaScript console to view the output of **console.log()** statements, inspect variables, and debug scripts. The element inspector is also useful for understanding how JavaScript interacts with HTML elements.

**1. DOM Syntax and Method**

**Objective:** Introduce students to basic DOM manipulation by changing the text of an HTML element.

**Exercise:** Create a simple HTML page with a **<p>** element that has an id of "example". Write a JavaScript function that changes the text content of the **<p>** element when a button is clicked.

**2. DOM Elements**

**Objective:** Teach students how to create and append a new element to the DOM.

**Exercise:** Create a webpage that contains a button. When the button is clicked, a new **<li>** element with some predefined text should be added to an existing **<ul>** list.

**3. DOM Attributes**

**Objective:** Show students how to get and set attributes of DOM elements.

**Exercise:** Create an HTML page with an image and a button. When the button is clicked, change the **src** attribute of the image to another image URL.

**4. JS Events**

**Objective:** Introduce students to JavaScript event listeners by implementing a simple form validation.

**Exercise:** Create a form with a text input for an email address and a submit button. Use JavaScript to add an event listener to the form's **submit** event that checks if the email input is empty. If it is, prevent the form from being submitted and alert the user.

## 7.6 Private Study

The time allocation for private study in this topic is expected to be 3 hours.

For students looking to deepen their understanding of the topics covered in the classroom exercises (DOM syntax and methods, DOM elements, DOM attributes, and JavaScript events), engaging in private study is an excellent way to reinforce these concepts. Here are some recommendations for resources and activities that can enhance their learning experience:

**1. Online Tutorials and Courses**

- **MDN Web Docs (Mozilla Developer Network):** This is an invaluable resource for web developers. The MDN offers comprehensive guides and reference materials on all aspects of web development, including detailed documentation on the DOM and JavaScript. Students can start with the JavaScript guide and delve into the sections on working with the DOM and handling events.
- **Codecademy:** Offers interactive JavaScript courses that include lessons on DOM manipulation. Their hands-on approach helps reinforce the concepts learned in the classroom.
- **freeCodeCamp:** Provides a vast curriculum that covers front-end libraries and frameworks, with practical exercises on JavaScript and the DOM. Their projects and challenges are excellent for applying what you've learned.

**2. Practice Platforms**

- **JavaScript.info:** This site offers an in-depth JavaScript tutorial that includes sections on the DOM, events, and forms. It's great for both beginners and experienced developers.
- **Frontend Mentor:** Challenges you to build projects based on real designs. This is excellent for applying your knowledge of DOM manipulation and event handling in real-world scenarios.
- **CodePen or JSFiddle:** These online code editors are perfect for experimenting with HTML, CSS, and JavaScript. Students can practice by recreating the classroom exercises and experimenting with new ideas.

# Topic 8: jQuery

## 8.1 Learning Objectives

This topic introduces the advanced concepts of jQuery. It provides an overview of some of the basic concepts of jQuery including topics such as how jQuery works, jQuery UI, and jQuery events and effects.

On completion of the topic, students will be able to:

- Explain jQuery concepts
- Create animated, interactive web pages using jQuery library
- Make use of simple jQuery examples
- Explain jQuery Events and Effects
- Describe jQuery UI

## 8.2 Pedagogic Approach

Information will be communicated to the students during the lectures. These are designed to be as interactive as possible. Students will undertake practical activities to develop skills and reinforce their understanding of the subjects introduced in the lecture. Finally private study and the subsequent tutorial will be used to consolidate learning and explore some aspects in greater detail.

## 8.3 Timings

Lecture: 2 hours

Laboratory Sessions: 4 hours

Private Study: 3 hours

## 8.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- jQuery introduction
- jQuery Syntax
- jQuery Events
- jQuery Effects
- DOM
- jQuery UI

## 8.5 Laboratory Sessions

The laboratory time allocation for this topic is 4 hours.

These exercises are designed to be observational and analytical, requiring students to engage critically with real-world web pages to understand the practical applications of web development concepts. The exercises aim to build a foundational understanding of jQuery in a hands-on manner, preparing students for more complex tasks as they progress.

**Hardware / Software requirements:**

1. **Code Editor**: Visual Studio Code (VS Code)

   Highly recommended for its support for JavaScript and jQuery with syntax highlighting, IntelliSense, and extensions that can enhance jQuery development.
2. **Web Browser**: Google Chrome or Mozilla Firefox

   Both have excellent developer tools for debugging JavaScript and jQuery code. These browsers allow you to test the functionality of your jQuery scripts effectively.
3. **jQuery Library**:

   Ensure students know how to include the jQuery library in their projects. This can be done by downloading the jQuery file and linking it locally or by linking to a CDN (Content Delivery Network) version of jQuery in the HTML file. For example, adding **<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>** in the **<head>** section of the HTML file.
4. **jQuery UI Library** (for the jQuery UI exercise):

   Similar to jQuery, include the jQuery UI library by linking to it in the HTML file. It also requires linking to the jQuery UI CSS file for styles. For instance, you can add **<link rel="stylesheet" href="//code.jquery.com/ui/1.12.1/themes/base/jquery-ui.css">** and **<script src="https://code.jquery.com/ui/1.12.1/jquery-ui.js"></script>** in the **<head>** section.
5. **Optional: Online References and Documentation**:

   jQuery Official Website and Documentation (https://jquery.com/) - Essential for students to learn more about jQuery syntax, methods, and best practices.
   jQuery UI Official Website (https://jqueryui.com/) - Useful for exploring more widgets and features available in jQuery UI beyond the Datepicker widget.

### 1. jQuery Syntax

**Objective:** Familiarize students with the basic syntax of jQuery, including selecting elements, applying methods, and chaining.

**Exercise:** Create a simple HTML page with several paragraphs (**<p>** elements) with different classes. Use jQuery to select all paragraphs, change their text colour, and then hide them with a button click.

- Start by including the jQuery library in your HTML.
- Use the **$()** selector to select all **<p>** elements.
- Apply **.css()** method to change the text color.
- Use **.click()** event on a button to hide all paragraphs using the **.hide()** method.

**Expected Outcome:** Students will understand how to include jQuery, select elements, modify CSS properties, and respond to user events.

**2. jQuery Events**

**Objective:** Teach students how to handle events in jQuery, focusing on mouse events.

**Exercise:** Create an HTML page with a button. When the button is clicked, display an alert with a message. Additionally, have a ** ** element change colour when the mouse hovers over it.

- Include jQuery in your HTML.
- Use the **.click()** method to show an alert when the button is clicked.
- Use the **.hover()** method on a ** ** to change its background colour when the mouse hovers over it. Reset the colour when the mouse leaves.

**Expected Outcome:** Students learn to handle click and hover events, demonstrating the event handling capabilities of jQuery.

**3. jQuery Effects**

**Objective:** Introduce students to jQuery effects, such as show, hide, fade, and slide.

**Exercise:** Create an HTML page with several images. Use buttons to apply different effects to these images, such as fading out, sliding up, or showing after being hidden.

- Include jQuery in the HTML.
- Place buttons for each effect: Fade Out, Slide Up, and Show.
- Use **.fadeOut()**, **.slideUp()**, and **.show()** methods on images when the respective button is clicked.

**Expected Outcome:** Students will learn how to manipulate the visibility and appearance of elements with animations, providing a dynamic user experience.

**4. jQuery UI**

**Objective:** Get students started with jQuery UI by using a simple widget.

**Exercise:** Implement a jQuery UI Datepicker in a form.

- Include both jQuery and jQuery UI libraries in your HTML.
- Add an input field where the datepicker will be attached.
- Use **$("#yourInputField").datepicker();** to attach a datepicker to the input field.

**Expected Outcome:** Students will be introduced to jQuery UI widgets, enhancing the user interface beyond what's available with basic HTML and jQuery.

## 8.6 Private Study

The time allocation for private study in this topic is expected to be 3 hours.

For students looking to deepen their understanding of jQuery, jQuery events, jQuery effects, and jQuery UI through private study, here are several recommendations that encompass online resources, books, and practice projects. These resources are designed to cater to different learning styles, from textual content and video tutorials to hands-on coding exercises.

**Online Courses and Tutorials**

1. **Codecademy's jQuery Course:** An interactive platform that offers a hands-on approach to learning jQuery. It covers the basics up to more advanced topics. The exercises allow students to write code directly in the browser, receiving immediate feedback.
2. **Udemy jQuery Courses:** Udemy offers a wide range of jQuery courses tailored to different levels of experience. Look for courses with high ratings and reviews that cover jQuery syntax, events, effects, and UI to ensure comprehensive learning.
3. **W3Schools jQuery Tutorial:** A great free resource for beginners. It provides simple, easy-to-understand examples and allows students to practice directly in the browser. W3Schools also covers jQuery UI and offers a "Try it Yourself" feature.
4. **jQuery Learning Center:** An official resource that offers guides and tutorials on various aspects of jQuery. It's a good place to understand the core concepts and advanced features directly from the source.

# Topic 9: Responsive CSS Framework: Bootstrap (1)

## 9.1 Learning Objectives

This topic introduces the concepts of CSS Framework - Bootstrap. This is the first part of two weeks sessions. It provides an overview of some of the basic concepts of Bootstrap framework including topics such as what is bootstrap, why it is needed, its features, containers, the grid system and bootstrap content elements.

On completion of the topic, students will be able to:

- Use Bootstrap concepts and develop responsive web pages
- Demonstrate understanding of basic bootstrap concepts including Bootstrap installation
- Identify different types of Bootstrap containers and explain its concepts
- Explain the grid system under bootstrap framework and some of its content elements

## 9.2 Pedagogic Approach

Information will be communicated to the students during the lectures. These are designed to be as interactive as possible. Students will undertake practical activities to develop skills and reinforce their understanding of the subjects introduced in the lecture. Finally private study and the subsequent tutorial will be used to consolidate learning and explore some aspects in greater detail.

## 9.3 Timings

| | |
|---|---|
| Lectures: | 2 hours |
| Laboratory Sessions: | 4 hours |
| Private Study: | 3 hours |

## 9.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- Bootstrap introduction
- Bootstrap containers
- Grid system in bootstrap framework
- Bootstrap content elements

## 9.5 Laboratory Sessions

The time allocation for this topic is 4 hours.

These exercises are designed to be observational and analytical, requiring students to engage critically with real-world web pages to understand the practical applications of web development concepts.

These exercises are focused on the Bootstrap framework, specifically targeting Bootstrap containers, the grid system, and Bootstrap content elements. Each exercise aims to provide hands-on experience with Bootstrap, encouraging students to apply their learning in a practical context.

**Hardware / software requirements:**

1. **Code Editor**: Visual Studio Code (VS Code)
    - A popular choice for its support for HTML, CSS, and Bootstrap classes. Its features like live preview, IntelliSense, and extensions specifically for Bootstrap can enhance the learning and development experience.
2. **Web Browser**: Google Chrome or Mozilla Firefox
    - Necessary for testing and viewing the Bootstrap-styled pages. Both browsers come with developer tools that are helpful for inspecting HTML elements and debugging.
3. **Bootstrap Library**:
    - Ensure that students include the Bootstrap CSS file in their HTML documents. This can be done by linking to the Bootstrap CDN in the **<head>** section of the HTML file, like so:

```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
```

    - For Bootstrap 4 or whichever version is being taught. Ensure the version in the link is updated to match the version you intend to teach.
4. **Optional: jQuery and Popper.js** (for Bootstrap components that require JavaScript):
    - If the exercises extend to Bootstrap components that rely on JavaScript (like modals, dropdowns, etc.), including jQuery and Popper.js might be necessary. For Bootstrap 4, these can be included before the closing **</body>** tag in your HTML:

```
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
 <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.5.2/dist/umd/popper.min.js"></script>
 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></script>
```

    - Note: Bootstrap 5 and newer versions do not require jQuery.
5. **Optional: Online Resources and Documentation**:
    - Bootstrap Official Documentation (https://getbootstrap.com/) - Critical for students to understand the framework, explore components, and learn about the grid system, utilities, and JavaScript plugins Bootstrap offers.

**1. Exercise on Bootstrap Containers**

**Objective:** Understand and apply Bootstrap containers for layout management.

**Task:**

- Create an HTML file and link the Bootstrap CSS file.
- Inside the **<body>**, define two different sections using ** ** elements.
- For the first section, apply a **container** class to center your content and give it some padding.
- For the second section, use a **container-fluid** class for a full-width container, spanning the entire width of the viewport.
- Inside each container, add a simple paragraph (**<p>**) with some text to see the effect of the container styles.

**Expected Learning Outcome:**

- Students will learn the difference between **container** and **container-fluid** and how to use them to manage the layout of web pages.

**2. Exercise on Grid System in Bootstrap Framework**

**Objective:** Learn to create responsive layouts using Bootstrap's grid system.

**Task:**

- Use the same HTML file from the first exercise or create a new one and link the Bootstrap CSS file.
- Create a ** ** with a **container** class.
- Inside this container, create a row using a ** ** with a **row** class.
- Within this row, create three ** ** elements with classes **col-sm-4**. Inside each, place a simple piece of text or an image.
- Resize your browser window to see how the layout adjusts on different screen sizes.

**Expected Learning Outcome:**

- Students will understand the basics of the Bootstrap grid system and how it enables responsive design.

**3. Exercise on Using Bootstrap Content Elements: Typography**

**Objective:** Explore Bootstrap's typography and content styling options.

**Task:**

- Create or modify an existing HTML file and ensure Bootstrap CSS is linked.
- In a **container** class, experiment with different heading levels (**<h1>** through **<h6>**), including paragraph text using the **<p>** tag.
- Apply Bootstrap's text utility classes such as **text-primary**, **text-center**, **font-weight-bold**, etc., to modify the appearance of the text.

- Include a blockquote using **<blockquote>** with **blockquote** and **blockquote-footer** classes.

**Expected Learning Outcome:**
- Students will learn how to utilize Bootstrap's classes to style typography and content elements effectively.

**4. Exercise on Bootstrap Buttons and Images**

**Objective:** Get familiar with Bootstrap's components for buttons and images.

**Task:**
- Continue with the previously used or a new HTML file with Bootstrap CSS linked.
- Add a section in your document where you create buttons with different styles (primary, secondary, success, etc.) by using Bootstrap's button classes (e.g., **btn btn-primary**).
- Experiment with button sizes (e.g., **btn-lg**, **btn-sm**) and the **disabled** attribute.
- Include a few images and apply Bootstrap's image classes to make them responsive (e.g., **img-fluid**) and to round their corners (e.g., **rounded**).

**Expected Learning Outcome:**
- Students will learn how to incorporate and style buttons and images using Bootstrap's classes, enhancing the visual appeal and responsiveness of web pages.

## 9.6 Private Study

The time allocation for private study in this topic is expected to be 3 hours.

To further enhance their understanding and skills in front-end web development, particularly focusing on Bootstrap and its components, students can engage in private study activities. Here are some recommendations tailored to the topics covered in the classroom exercises:

**1. Online Tutorials and Courses**
- **Bootstrap Official Documentation:** Start with the Bootstrap official documentation. It's comprehensive and includes examples, which are excellent for learning how to use Bootstrap effectively for various components and layouts.
- **Codecademy or Udemy Courses:** Platforms like Codecademy, Udemy, or Coursera offer structured courses on Bootstrap and front-end development. These can provide a more guided learning path with projects and quizzes to test knowledge.

**2. Practice with Design and Layout**
- **CSS Tricks and Layout Techniques:** Bootstrap is built on CSS, so understanding CSS deeply will help you customize Bootstrap components beyond the defaults. Websites like CSS-Tricks offer valuable tips and tricks for CSS layout techniques, animations, and more.
- **Responsive Design Practices:** Focus on creating responsive designs using Bootstrap's grid system and utilities. Experiment with making websites that look great on all devices, from phones to desktops.

**3. Experiment and Build**
- CodePen and JSFiddle: Use online code editors like CodePen or JSFiddle to experiment with Bootstrap components and layouts. These platforms allow you to quickly test ideas and share them with others for feedback.

# Topic 10: Responsive CSS Framework: Bootstrap - II

## 10.1 Learning Objectives

This topic continues the concepts of CSS Framework – Bootstrap from the previous lesson. This is the second part of two weeks sessions. It provides an overview of some of the concepts of Bootstrap framework including topics such as bootstrap forms, form controls, buttons, some of the utilities and bootstrap cards.

On completion of the topic, students will be able to:

- Use advanced features of Bootstrap and develop responsive web pages.
- Demonstrate understanding of various bootstrap concepts including Bootstrap form, form controls, buttons, utilities and Cards.
- Identify different types of Bootstrap elements and explain its working.

## 10.2 Pedagogic Approach

Information will be communicated to the students during the lectures. These are designed to be as interactive as possible. Students will undertake practical activities to develop skills and reinforce their understanding of the subjects introduced in the lecture. Finally private study and the subsequent tutorial will be used to consolidate learning and explore some aspects in greater detail.

## 10.3 Timings

| | |
|---|---|
| Lectures: | 2 hours |
| Laboratory Sessions: | 4 hours |
| Private Study: | 3 hours |

## 10.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- Bootstrap Forms
- Bootstrap Form Controls
- Bootstrap Buttons
- Bootstrap Utilities
- Bootstrap Cards

## 10.5 Laboratory Sessions

The laboratory time allocation for this topic is 4 hours.

These exercises are designed to be observational and analytical, requiring students to engage critically with real-world web pages to understand the practical applications of web development concepts. These exercises will help students get hands-on experience with Bootstrap, focusing on forms and form controls, buttons, utilities, and cards.

**Hardware / software requirements:**

1. **Code Editor**: Visual Studio Code (VS Code)
    - An excellent editor for HTML and CSS, providing support for Bootstrap syntax highlighting and autocomplete features. It's user-friendly for beginners and has powerful features for more advanced users.
2. **Web Browser**: Google Chrome or Mozilla Firefox
    - Both are suitable for testing and viewing Bootstrap-styled pages. Their developer tools are invaluable for debugging and understanding how Bootstrap styles are applied to HTML elements.
3. **Bootstrap CSS Link**:
    - It's essential for students to include the Bootstrap CSS link in the **<head>** section of their HTML documents to use Bootstrap's components and classes. Use the Bootstrap CDN for easy access:

      ```
      <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
      ```
    - This link points to Bootstrap version 4.5.2; however, you may want to use the link corresponding to the latest version or the version being taught.

### 1. Bootstrap Forms and Form Controls

**Objective:** Create a simple contact form using Bootstrap form controls.

**Instructions:**

1. Start by creating an HTML file and include the Bootstrap CSS link in the **<head>** section.
2. Inside the **<body>**, create a ** ** with a class of **container** to ensure your form is nicely centered and padded.
3. Within the **container**, use a **<form>** tag to start your form.
4. Add a form group with an **<input>** element for the user's name. Make sure to use Bootstrap classes such as **form-group** and **form-control**.
5. Repeat the above step to add more form fields for the user's email and a message (use a **<textarea>** for the message field).
6. Include placeholder attributes to guide users on what to enter in each field.
7. Finally, add a submit button using the **<button>** tag with Bootstrap button classes.

**Example:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Contact Form</title>
    <link                href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
<div class="container">
    <form>
        <div class="form-group">
            <label for="name">Name</label>
            <input type="text" class="form-control" id="name" placeholder="Enter name">
        </div>
        <div class="form-group">
            <label for="email">Email</label>
            <input type="email" class="form-control" id="email" placeholder="Enter email">
        </div>
        <div class="form-group">
            <label for="message">Message</label>
            <textarea        class="form-control"        id="message"        rows="3"        placeholder="Enter
message"></textarea>
        </div>
        <button type="submit" class="btn btn-primary">Submit</button>
    </form>
</div>
</body>
</html>
```

**2. Bootstrap Buttons**

**Objective:** Explore Bootstrap button styles and sizes.
**Instructions:**

1. Create a new HTML file and include Bootstrap in the **<head>**.
2. In the **<body>**, create a ** ** with a class of **container** for layout purposes.
3. Inside the container, add **<button>** elements for each Bootstrap button style (e.g., primary, secondary, success, info, warning, danger, dark, and light).
4. For each button, use the **btn** class, along with the **btn-{style}** class to apply the different Bootstrap themes.
5. Also, create buttons with different sizes using the **btn-lg**, **btn-sm**, and default size. Mention the size in the button text for clarity.
6. Include an action, such as **alert('Button clicked!');**, using the **onclick** attribute to demonstrate interaction.

**Example:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css" rel="stylesheet">
    <title>Bootstrap Buttons</title>
</head>
<body>
<div class="container mt-3">
    <button type="button" class="btn btn-primary" onclick="alert('Primary button clicked!');">Primary</button>
    <button type="button" class="btn btn-secondary">Secondary</button>
    <!-- Add more buttons for each style here -->
    <button type="button" class="btn btn-success btn-lg">Large Success Button</button>
    <button type="button" class="btn btn-danger btn-sm">Small Danger Button</button>
</div>
</body>
</html>
```

### 3. Bootstrap Utilities

**Objective:** Utilize Bootstrap utilities to style a simple profile card.

**Instructions:**

1. Create an HTML file and link Bootstrap in the **<head>**.
2. In the **<body>**, use a ** ** with classes **container** and **mt-5** for spacing.
3. Inside the container, create a simple profile card using a ** ** with classes **card** and **p-3**.
4. Add a **<h4>** tag for the name, a **<p>** tag for a short bio, and use utility classes like **text-muted**, **mb-2**, **text-center**, or **font-weight-bold** to style the text.
5. Experiment with spacing utilities (e.g., **mb-3**, **mt-3**) to adjust the margins and paddings of your elements.

**Example:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link                href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
    <title>Profile Card</title>
</head>
<body>
<div class="container mt-5">
    <div class="card p-3">
        <h4 class="text-center font-weight-bold">John Doe</h4>
        <p class="text-muted text-center">Front-end Developer</p>
    </div>
</div>
</body>
</html>
```

**4. Bootstrap Cards**

**Objective:** Create a card layout to display a product or service.

**Instructions:**

1. Start with a new HTML file and include Bootstrap.
2. Use a ** ** with a class of **container** in the **<body>** to hold your cards.
3. Inside the container, create a ** ** with classes **card** and set a **width** style. Add a **card-img-top** for an image, **card-body** for content.
4. Within the **card-body**, include a **card-title**, **card-text** for a brief description, and a button with classes **btn btn-primary**.
5. Encourage students to add custom text and images to personalize their cards.

**Example:**

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <link                href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
    <title>Product Card</title>
</head>
<body>
<div class="container mt-3">
    <div class="card" style="width: 18rem;">
        <img src="https://via.placeholder.com/150" class="card-img-top" alt="...">
        <div class="card-body">
            <h5 class="card-title">Product Name</h5>
            <p class="card-text">Some quick example text to build on the card title and make up the bulk
of the card's content.</p>
            <a href="#" class="btn btn-primary">Go somewhere</a>
        </div>
    </div>
</div>
</body>
</html>
```

## 10.6 Private Study

The time allocation for private study in this topic is expected to be 3 hours.

For students wishing to deepen their understanding of front-end web development, particularly with Bootstrap and its components, private study can be an incredibly effective way to complement classroom learning. Here are some recommendations for resources and study practices that can help:

**Online Courses and Tutorials**

1. **Codecademy's HTML & CSS Courses:** These offer a good foundation for beginners, with interactive lessons that include Bootstrap basics.
2. **Udemy and Coursera:** Look for courses specifically focused on Bootstrap or front-end development. Courses often range from beginner to advanced levels and are taught by industry professionals.
3. **freeCodeCamp:** Offers comprehensive tutorials and projects on front-end libraries, including Bootstrap. It's entirely free and includes a supportive community.

**Documentation and Reading Material**

1. **Bootstrap Official Documentation:** The best place to understand the framework's components, utilities, and classes. Reading documentation is a crucial skill for developers, and Bootstrap's documentation is well organized for learners.
2. **MDN Web Docs (Mozilla Developer Network):** For a deeper understanding of HTML and CSS, MDN provides extensive resources and guides that are highly respected in the industry.

**Practice Websites**

1. **CodePen and JSFiddle:** These online code editors are great for experimenting with HTML, CSS, and JavaScript. You can try out Bootstrap components and see the results in real-time.
2. **GitHub Pages:** Learn to use Git and GitHub to host your Bootstrap projects online. This practice is invaluable for understanding version control and showcasing your work to potential employers or collaborators.

# Topic 11: Code Review, Testing and Collaboration

## 11.1 Learning Objectives

This topic discusses the concepts of code review, testing and collaboration. The lecture introduces the concept of software development lifecycle, its models and how code collaboration can be achieved with version control tools.

On completion of the topic, students will be able to:

- Identify different test methods for front end web development.
- Develop and apply a test strategy consistent with the design.
- Describe the software development life cycle process and explain some of its models.
- Explain the concepts of Version Control and its significance.

## 11.2 Pedagogic Approach

Information will be communicated to the students during the lectures. These are designed to be as interactive as possible. Students will undertake practical activities to develop skills and reinforce their understanding of the subjects introduced in the lecture. Finally private study a will be used to consolidate learning and explore some aspects in greater detail.

## 11.3 Timings

Lectures: 2 hours

Laboratory Sessions: 4 hours

Private Study: 3 hours

## 11.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time. Please also refer to the slides.

The structure of this topic is as follows:

- Webapp testing
- SDLC
- SDLC models
- Version Control

### 11.5 Laboratory Sessions

The laboratory time allocation for this topic is 4 hours.

These exercises are designed to be observational and analytical, requiring students to engage critically with real-world web pages to understand the practical applications of web development concepts. These exercises are designed to give students practical experience with key aspects of front-end web development, including testing and version control, in a beginner-friendly manner.

**Hardware / software requirements:**
**Web App Testing**

1. **Functionality Testing**:
    - **Development Tools**: Any text editor or IDE like Visual Studio Code, equipped for web development (HTML, CSS, JavaScript).
    - **Web Browser**: Google Chrome or Mozilla Firefox for testing the web application's functionality.
    - **Optional**: Use browser developer tools for debugging and inspecting elements to help identify issues.
2. **Security Testing**:
    - **Code Analysis Tools**: Tools like SonarQube can help analyze code for potential security issues.
    - **Web Browser**: Use developer tools for testing and observing network requests and responses, particularly for testing input validation and other security flaws.
3. **Performance Testing**:
    - **Online Tools**: Google PageSpeed Insights for analyzing the performance of web pages and receiving specific suggestions for improvement.
    - **Web Browser**: Chrome Lighthouse (built into Chrome DevTools) for performance, accessibility, progressive web apps, and more.

**Version Control Using Git**

1. **Git**:
    - **Git Software**: Download and install Git from git-scm.com to enable version control on the student's machines.
    - **GitHub**: Students should create accounts on GitHub for online repository hosting and collaboration.
2. **Learning Resources**:
    - **Official Git Documentation** at git-scm.com/doc for comprehensive guides and reference materials.
    - **GitHub Learning Lab** (lab.github.com) for interactive tutorials on using Git and GitHub.

3. **Practice Tools**:
    - **Visual Studio Code** with integrated Git control for practicing basic Git operations directly within the IDE.
    - **Git GUI Clients** like GitHub Desktop, SourceTree, or GitKraken for those who prefer a graphical interface over command line.

**Web App Testing**

**1. Functionality Testing**
**Objective:** Test the functionality of a simple web application (e.g., a to-do list app).

- **Task:** Create a simple web application with basic CRUD (Create, Read, Update, Delete) functionality. Students should test the application by adding new items, reading/displaying them, updating existing items, and deleting them.
- **Procedure:**
    1. Students will manually test each function to ensure it works as expected.
    2. They will report any bugs or issues encountered during the process.
    3. Discuss the importance of testing each part of the operations in web apps.

**2. Security Testing**
**Objective:** Identify common security issues in a simple web application.

- **Task:** Provide students with a basic web application that has intentional security flaws (e.g., exposed API keys, lack of input validation).
- **Procedure:**
    1. Students will review the code to identify security flaws.
    2. They will suggest fixes or mitigations for the identified issues.
    3. Discuss why security testing is crucial in web development.

**3. Performance Testing**
**Objective:** Test the performance of a web page under load.

- **Task:** Use a simple online tool (e.g., Google PageSpeed Insights) to test the performance of a web page.
- **Procedure:**
    1. Students will run the performance test on a specified web page.
    2. They will analyse the results and identify areas for improvement.
    3. Discuss how performance impacts user experience and search engine rankings.

**Version Control Using Git**

4. Basic Git Operations

**Objective:** Familiarize students with basic Git operations.

- **Task:** Teach students how to use Git for version control of their projects.
- **Procedure:**
    1. Set up Git on each student's machine and create a GitHub account if they don't already have one.
    2. Practice cloning a repository from GitHub to their local machine.
    3. Make changes to a file, then add, commit, and push the changes back to the repository.
    4. Show how to pull changes made by others.
    5. Discuss the importance of version control in collaborative projects and how it can be used to track changes and revert to previous versions if needed.

## 11.6 Private Study

The time allocation for private study in this topic is expected to be 3 hours.

To enhance understanding and skills in web app testing and version control using Git, students can engage in private study through various resources and activities. Here are some recommendations for each topic:

**Web App Testing**

1. Online Courses

- **Coursera** and **Udemy** offer courses on web development and testing that cover functionality, security, and performance testing. Look for courses that provide hands-on projects.
- **Codecademy** has interactive courses on web development basics that can solidify understanding of how web apps work, which is crucial for effective testing.

2. Practice Platforms

- **HackerRank** and **LeetCode**: Engage in coding challenges that can improve your problem-solving skills and understanding of algorithms, which indirectly benefits testing by fostering a deeper understanding of how applications work.
- **OWASP Juice Shop**: An intentionally insecure web application for practicing security testing techniques.

3. Blogs and Online Resources

- Follow blogs and websites like **Smashing Magazine**, **CSS-Tricks**, and **MDN Web Docs** for the latest trends and best practices in web development and testing.
- **Google Developers** website for performance testing insights and tools like Lighthouse.

**Version Control Using Git**

1. Online Courses
    - **GitHub Learning Lab** offers guided courses for learning Git and GitHub directly through hands-on practice.
    - **Coursera** and **Udemy** have comprehensive courses on Git and GitHub for version control, catering to beginners and advanced users.

2. Practice Platforms
    - **GitKraken's Git GUI**: While learning command-line Git is crucial, using a GUI can help understand the process visually.
    - **Git Exercises**: A website offering exercises that range from basic to advanced levels, helping to solidify your understanding of Git operations.

3. Blogs and Online Resources
    - **Atlassian Git Tutorials**: Comprehensive guides and tutorials on Git, covering basic to advanced topics.
    - **GitHub Blog** and **GitLab Blog**: Stay updated with new features, best practices, and how to effectively use these platforms for version control and collaboration.

# Topic 12: Unit Summary

## 12.1 Learning Objectives

This topic provides an overview of the unit content and briefly discusses a number of topics related to the material we have covered in the unit.

On completion of the topic, students will be able to:

- Recap the main parts of the unit week by week.

## 12.2 Pedagogic Approach

Information will be presented to the students during the lectures. The lecture is a recap of previous topics. Students should participate as much as possible as it is a good way for them to test their understanding of the material we have covered. Private study and laboratory sessions will focus on revision.

## 12.3 Timings

Lectures: 2 hours

Laboratory Sessions: 4 hours

Private Study: 5 hours

## 12.4 Lecture Notes

The following is an outline of the material to be covered during the lecture time and should be read in conjunction with the slides provided.

The structure of this topic is as follows:

- Recap the main topics of each week lecture for this unit.

## 12.5 Laboratory Sessions

The laboratory time allowance for in this topic is 4 hours.

### Lab Exercise 1: Build a Personal Webpage

**Objective**: This exercise aims to introduce students to the basics of HTML, CSS, and JavaScript by building a simple personal webpage. The webpage will include personal information, a photo, and a section that changes dynamically based on user interaction.

**Skills Covered**:

- HTML: Structure of a webpage
- CSS: Styling webpages
- JavaScript: Adding interactive elements

**Requirements**:

1. **HTML**:
    - Use HTML to create the basic structure of your webpage. Include the following elements:
        - A header with your name.
        - A paragraph introducing yourself.
        - An unordered list of your hobbies or interests.
        - An image that represents you or your interests.
        - A button labelled "Change Theme".
2. **CSS**:
    - Style your webpage using an external CSS file. Include the following styles:
        - Change the background colour of the body.
        - Set a font family for your text.
        - Style your header to be larger and a different colour than the rest of your text.
        - Add margins and padding to make your content more visually appealing.
        - Style your button to change its appearance when hovered over.

3. **JavaScript**:
    - Add an external JavaScript file to your project.
    - Write JavaScript code to change the background colour of the webpage when the "Change Theme" button is clicked. You can have a set of predefined colours that the background cycles through.

**Instructions**:

1. Create an HTML file named **index.html**, a CSS file named **styles.css**, and a JavaScript file named **script.js**.
2. Link your CSS and JavaScript files to your HTML document.
3. Implement the HTML structure as described in the requirements.
4. Style your webpage using the CSS file following the styling requirements.
5. Implement the JavaScript functionality to allow theme changes on button click.

**Expected Outcome**: Students will have a personal webpage that showcases their ability to use HTML for structure, CSS for styling, and JavaScript for dynamic behaviour. This exercise will also encourage creativity and personal expression through web development.

**Evaluation Criteria**:

- Correct use of HTML tags and structure.
- Effective application of CSS for styling.
- JavaScript functionality works as expected without errors.
- Code readability and organisation.

**Lab Exercise 2: Create a To-Do List Application**

**Objective**: This lab exercise aims to provide hands-on experience with HTML for creating form elements, CSS for styling, and JavaScript for adding dynamic interactivity. Students will build a simple To-Do List application where users can add tasks, mark them as completed, and delete them.

**Skills Covered**:

- HTML: Creating forms and other elements
- CSS: Designing and layout
- JavaScript: DOM manipulation, event handling

**Requirements**:

1. **HTML**:
    - Create an HTML file with the following elements:
        - A title **<h1>** element for your application name, e.g., "My To-Do List".
        - An input field for entering new tasks.
        - A submit button to add tasks to the list.
        - An empty **<ul>** or **<ol>** element where your tasks will be listed.
2. **CSS**:
    - Use CSS to style your application with an external stylesheet. Include the following:
        - A distinct background colour for your application.
        - Styling for the input field and submit button to make them visually appealing.
        - Style the task list to distinguish between completed and pending tasks (e.g., strike-through text for completed tasks).
3. **JavaScript**:
    - Use JavaScript to add functionality to your To-Do List:
        - Adding a new task to the list when the submit button is clicked.
        - Marking a task as completed when it is clicked.
        - Optional: Adding a delete button to each task to remove it from the list.

**Instructions**:

1. Create an **index.html** file for the structure of your To-Do List application.
2. Link a **styles.css** file to your HTML for styling the application.
3. Link a **script.js** file to your HTML where you will write your JavaScript code.
4. In your HTML, include an input field, a submit button, and an empty list (**<ul>** or **<ol>**) for displaying the tasks.

5. Style your application using CSS. Ensure it is visually organized and appealing.
6. Write JavaScript code to:
    - Capture the value from the input field and add it as a new list item when the submit button is clicked.
    - Toggle the completion status of a task when it is clicked. Consider changing the style of the task to indicate its completion status.
    - Optional: Implement a way to delete tasks from the list, either through a delete button next to each task or another method you prefer.

**Expected Outcome**: Students will create a functional To-Do List application where users can add new tasks, mark tasks as completed, and optionally delete them. This exercise reinforces the practical application of HTML, CSS, and JavaScript in creating interactive web applications.

**Evaluation Criteria**:

- Correct implementation of HTML elements for user input and task display.
- Effective use of CSS for styling the application and distinguishing between task states.
- JavaScript correctly adds, toggles, and deletes tasks without errors.
- Overall usability and aesthetic of the application.

### 12.6 Private Study

The time allocation for private study in this topic is expected to be 5 hours.

Topic 12 should have alerted you to specific topics that you might need to review again in order to re-enforce your understanding. Review the lecture slides from all the topics and do any additional reading necessary so that you understand and feel confident about the content. Prepare and list specific points you would like the tutor to explain again and bring this to the final tutorial session for the module.

## Book References for the Unit

Creating a comprehensive reference book list for students interested in back-end as well as some front-end web development topics requires covering a variety of subjects. Below is a curated list of books that should serve well across the specified topics, providing a blend of foundational knowledge, practical skills, and advanced techniques. This list includes books that are highly regarded in the web development community for their clarity, depth, and practicality.

**Introduction to Front-End Web Development**

1. **"HTML and CSS: Design and Build Websites" by Jon Duckett**
    - This book provides a clear introduction to the basics of HTML and CSS, perfect for beginners. Its visual approach helps readers understand the concepts through examples and illustrations.
2. **"Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics" by Jennifer Robbins**
    - An excellent resource for beginners, this book covers the fundamentals of web design, including HTML, CSS, JavaScript, and web graphics, providing a solid foundation for front-end development.

**HTML, CSS**

1. **"HTML5: The Missing Manual" by Matthew MacDonald**
    - This manual covers the latest HTML5 standards and provides practical advice on building web pages that work across different devices.
2. **"CSS3: The Missing Manual" by David Sawyer McFarland**
    - A comprehensive guide to CSS3, offering explanations on how to use new features to create dynamic web pages.

**CSS Flexbox and Grid Layouts**

1. **"CSS Secrets: Better Solutions to Everyday Web Design Problems" by Lea Verou**
    - Focuses on practical tips and tricks, including how to best use Flexbox and Grid to solve common web design challenges.
2. **"A Complete Guide to Grid" by Rachel Andrew and "A Complete Guide to Flexbox" by Chris Coyier** (available online at CSS-Tricks)
    - These are not books but are considered essential readings for anyone wanting to master CSS Grid and Flexbox.

**JavaScript**

1. **"Eloquent JavaScript: A Modern Introduction to Programming" by Marijn Haverbeke**
    - Covers JavaScript from the basics to advanced topics, including programming fundamentals and real-world web development tasks.
2. **"JavaScript: The Definitive Guide" by David Flanagan**
    - A comprehensive resource for JavaScript developers, covering everything from basic syntax to advanced topics.

**jQuery**

1. **"jQuery: Novice to Ninja" by Earle Castledine and Craig Sharkie**
    - A practical guide to mastering jQuery, focusing on practical applications and techniques.

**CSS Bootstrap Framework**

1. **"Bootstrap 4 – Responsive Web Design" by Silvio Moreto, Matt Lambert, Benjamin Jakobus, and Jason Marah**
    - Covers Bootstrap 4 in detail, from installation to customization, along with tips on how to create responsive designs.

**Basics of Web Development Code Review and Testing**

1. **"Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability" by Steve Krug**
    - Although not solely about code review and testing, this book provides invaluable insights into web usability, which is crucial for effective web development and testing.
2. **"JavaScript Testing" by Jasmine Framework**
    - Focuses on using the Jasmine framework for testing JavaScript applications, covering basics to advanced testing techniques.

**Software Development Lifecycle**

1. **"Agile Web Development with Rails 6" by Sam Ruby, David Bryant Copeland, and Dave Thomas**
    - Provides insights into the Agile software development process using Rails as an example, though the principles can be applied more broadly.

**Version Control**

1. **"Pro Git" by Scott Chacon and Ben Straub**
    - An in-depth guide to using Git, from basic commands to advanced version control techniques.