Competency Based Learning Materials (CBLMs)

Web Design and Development for Freelancing

Level-3

Module 4: Design styles with CSS and CSS framework

Code: CBLM-ICT-WDF-04-L3-EN-V1.1

National Skills Development Authority
Prime Minister's Office
Government of the People's Republic of Bangladesh

Copyright

National Skills Development Authority
Prime Minister's Office
Level: 10-11, Biniyog Bhaban,
E-6 / B, Agargaon, Sher-E-Bangla Nagar Dhaka-1207, Bangladesh.
Email: ec@nsda.gov.bd
Website: www.nsda.gov.bd.
National Skills Portal: http:\\skillsportal.gov.bd

The CBLM on "Design styles with CSS and CSS framework" is developed based on NSDA approved Competency Standards and Competency Based Curriculum under Web Design and Development for Freelancing Level-3 Occupation. It contains the information required to implement the Web Design and Development for Freelancing Level-3 standard.

This document has been prepared by NSDA with the help of relevant experts, trainers/professionals.

All Government-Private-NGO training institutes in the country accredited by NSDA can use this CBLM to implement skill-based training of Web Design and Development for Freelancing Level-3 course.

How to use this Competency Based Learning Materials (CBLMs)

The module, Maintaining and enhancing professional & technical competency contains training materials and activities for you to complete. These activities may be completed as part of structured classroom activities or you may be required you to work at your own pace. These activities will ask you to complete associated learning and practice activities in order to gain knowledge and skills you need to achieve the learning outcomes.

1. Review the **Learning Activity** page to understand the sequence of learning activities you will undergo. This page will serve as your road map towards the achievement of competence.

2. Read the **Information Sheets.** This will give you an understanding of the jobs or tasks you are going to learn how to do. Once you have finished reading the **Information Sheets** complete the questions in the **Self-Check.**

3. **Self-**Checks are found after each **Information Sheet**. **Self-Checks** are designed to help you know how you are progressing. If you are unable to answer the questions in the **Self-Check** you will need to re-read the relevant **Information Sheet**. Once you have completed all the questions check your answers by reading the relevant **Answer Keys** found at the end of this module.

4. Next move on to the **Job Sheets. Job Sheets** provide detailed information about *how to do the job* you are being trained in. Some **Job Sheets** will also have a series of **Activity Sheets**. These sheets have been designed to introduce you to the job step by step. This is where you will apply the new knowledge you gained by reading the Information Sheets. This is your opportunity to practise the job. You may need to practise the job or activity several times before you become competent.

5. Specification **sheets**, specifying the details of the job to be performed will be provided where appropriate.

6. A review of competency is provided on the last page to help remind if all the required assessment criteria have been met. This record is for your own information and guidance and is not an official record of competency

When working though this Module always be aware of your safety and the safety of others in the training room. Should you require assistance or clarification please consult your trainer or facilitator.

When you have satisfactorily completed all the Jobs and/or Activities outlined in this module, an assessment event will be scheduled to assess if you have achieved competency in the specified learning outcomes. You will then be ready to move onto the next Unit of Competency or Module

Approved by

---th Executive Committee (EC) Meeting of NSDA

Held on ----------

Table of Contents

Module Content

Module: Design Styles with CSS and CSS Framework

Module Descriptor: This module covers the knowledge, skills, and attitude to design styles with CSS and CSS framework. It specifically includes planning a website, designing it using CSS, enhancing it using front-end framework, and testing and confirming the website.

Nominal Hours: 90

Learning Outcomes: After completion of the module, trainees will be able to:

1. Plan a website.
2. Design the website using cascading style sheets (CSS)
3. Enhance website using CSS framework.
4. Test and confirm the website.

Assessment Criteria:

1.1 The purpose and intended audience of the website are identified.
1.2 The design requirements and constraints are identified.
1.3 A conceptual design is developed.
1.4 Necessary software installed as per requirement.
2.1 Web layout is selected as per design requirement.
2.2 Web layout is designed using CSS as per client's requirements.
2.3 HTML and CSS file is integrated as required.
2.4 Web site is saved and executed.
3.1 Framework is collected and configured with website.
3.2 HTML and CSS framework are integrated.
3.3 CSS framework is customized using CSS as per requirements.
3.4 Web site is saved and executed.
4.1 The website is tested to ensure functionality and errors are corrected per standard operating procedure.
4.2 The website is opened with common browsers and checked for accessibility, readability, legibility, and presentation in accordance with client requirements.
4.3 The website is evaluated for fitness in terms of the purpose, target audience and specifications of client requirements.

Learning Outcome 1: Plan a website

Assessment Criteria	1. The purpose and intended audience of the website are identified. 2. The design requirements and constraints are identified. 3. A conceptual design is developed. 4. Necessary software installed as per requirement.
Conditions and Resources	1. Applicable tools, utensils, and equipment as prescribed by competency standards. 2. Supply materials 3. Relevant ingredients 4. CBLM related to the learning outcome. 5. Instructions, job sheets, activity sheets, and standard operating procedures 6. Personal protective equipment 7. Module/reference
Contents	1. Identifying the Purpose and Intended Audience of the Website 2. Identifying Design Requirements and Constraints 3. Developing a Conceptual Design 4. Installing Necessary Software as per Requirements
Training Methods	1. CBLM 2. Handouts 3. Books, Manuals 4. Module/ Reference 5. Paper 6. Pen
Assessment Methods	1. Written Test 2. Demonstration 3. Oral Questioning

Learning Experience 1: Plan a website

You must perform the learning steps below to achieve the objectives stated in this learning guide. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Steps	Resources specific instructions
1. Students will ask the instructor about Design styles with CSS and CSS framework	1. The instructor will provide the learning materials for` plan a website.
2. Read the **Information sheet/s**	2. Information Sheet No 1: Plan a website 1. Identifying the Purpose and Intended Audience of the Website 2. Identifying Design Requirements and Constraints 3. Developing a Conceptual Design 4. Installing Necessary Software as per Requirements
3. Complete the **Self-Checks & Answer key sheets.**	3. Self-Check No 1: Plan a website Answer key No 1: Plan a website
4. Read the **Job/ Task sheet** and **Specification Sheet**	4. Job/ task sheet and specification sheet Individual Activity: Job Sheet No 1: Plan a website. Specification Sheet 1: Plan a website.

Information Sheet 1 Plan a website

Learning Objective: After completing this information sheet, the learners will be able to select the image editing tool.

1.1 Purpose and Intended Audience of the Website
1.2 Design Requirements and Constraints
1.3 Developing a Conceptual Design
1.4 Necessary Software as per Requirements

1.1 Purpose and Audience Identification:

Purpose of the Website

A website's purpose is its foundation, the reason it exists. Before diving into design and development, defining why you're creating a website is crucial. Is it for showcasing your portfolio, selling products, sharing information, or providing services? The purpose is to give your website direction. When embarking on the journey of creating a website, defining its purpose is your very first step. Think of this as the North Star guiding your entire web development process. Why does your website exist, and what role will it play in the digital realm? To aid self-learning, let's dive deeper into understanding this concept with examples and discussions.

1.1.1 Why Purpose Matters

- Clarity and Focus: A well-defined purpose ensures clarity. It's like setting a destination before starting a journey. It keeps you focused on what you want to achieve with your website.

Examples:

  - Personal Portfolio: Imagine you're a photographer whose website aims to showcase your best work. In this case, every design choice, from layout to image quality, should revolve around presenting your portfolio effectively.
  - E-commerce: If you're running an online store, your website's purpose is clear: selling products. Everything, from product listings to checkout, should be optimized for this purpose.

- Audience Alignment: Knowing your purpose helps you connect with the right audience. Different purposes attract different visitors.

Examples:

  - Educational Platform: If your website aims to provide educational content, it's more likely to attract students and educators interested in learning and teaching.

- Entertainment Blog: If your purpose is to entertain through blogs or videos, your audience will be people seeking entertainment or information on specific topics.

How to Define Your Website's Purpose

- Identify Your Goals: Ask yourself what you want to achieve with your website. Are you looking to inform, entertain, sell, or connect? Your goals are closely tied to your website's purpose.
- Understand Your Audience: Consider who your target audience is. Understanding their needs and preferences helps align your purpose with what your audience seeks.

Identifying the Intended Audience

Understanding your audience is fundamental. Are you targeting consumers, businesses, students, hobbyists, or a specific niche? Knowing your audience helps tailor content, design, and functionality to meet their needs and preferences.

1.2 Design Requirements and Constraints

Defining Design Requirements

Design requirements encompass the visual and functional aspects of your website. Consider factors like colour schemes, typography, layout, and branding. Think about the user interface (UI) and user experience (UX) elements that will enhance navigation and engagement. They encompass both the aesthetic and functional elements that will make your site visually appealing and user-friendly. Let's break down this concept further:

Visual Elements:

- **Colour Schemes**: Determine the colour palette that reflects your brand or theme. Colours can evoke emotions and convey messages, so choose wisely.
- **Typography**: Select fonts that align with your website's personality and readability. Typography impacts how users consume content.
- **Layout**: Plan the arrangement of text, images, and interactive elements on each page. An intuitive layout enhances user experience.

Functional Elements:

- **User Interface (UI):** Design the interactive elements that users will engage with, such as buttons, menus, and forms. A well-designed UI streamlines navigation
- **User Experience (UX):** Consider users' overall experience on your site. This encompasses factors like load times, responsiveness, and ease of use

Recognizing Constraints

Constraints are limitations that affect your website's design and development. These might include budget, timeline, technical limitations, and platform compatibility. Identifying constraints upfront ensures realistic expectations and a smoother process.

Common Constraints:

- **Budget**: Determine the financial resources available for your project. A limited budget may impact design choices and the scope of development.
- **Timeline**: Set a realistic timeline for your project. Tight deadlines can influence design simplicity and the features you can include.
- **Technical Limitations**: Understand the technical capabilities and limitations of your platforms and tools. Compatibility issues can affect design choices.
- **Platform Compatibility**: Ensure your website functions seamlessly across various devices and browsers. Compatibility constraints may necessitate responsive design.

Why These Are Essential:

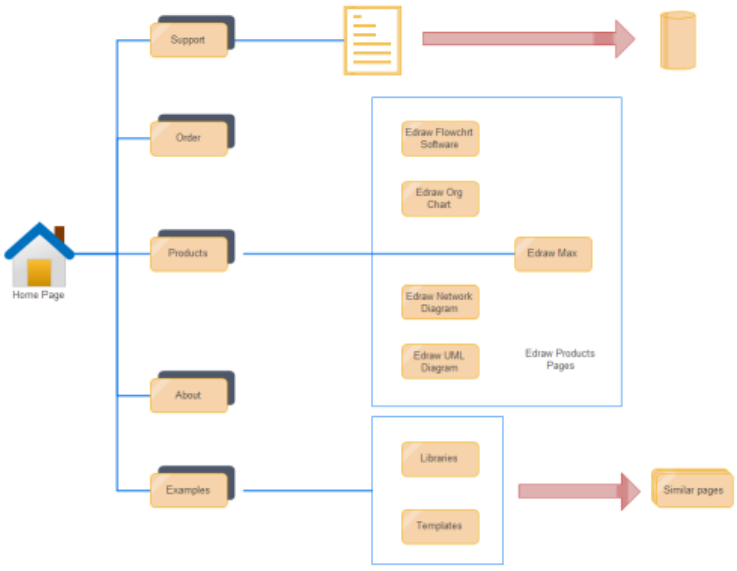
- **Balance Creativity and Realism**: Design requirements help strike a balance between creative aspirations and practicality. They provide a roadmap for achieving your vision.
- **Efficient Development**: Recognizing constraints early prevents costly design changes down the road. It allows you to work within limitations and meet project goals effectively.

Real-World Scenario Examples:

- *E-commerce Website*: Design requirements specify a user-friendly product catalogue, secure payment options, and an intuitive shopping cart. Constraints might include a tight budget for hiring developers and a fixed launch date for holiday sales.
- *Personal Blog*: Design requirements prioritise readability with a clean, legible font and an uncluttered layout. Constraints may include limited coding skills, affecting the complexity of interactive features.

1.3 Conceptual Design Development

Creating a Conceptual Design



With a clear purpose, audience, requirements, and constraints, you can start sketching a conceptual design. This is the blueprint for your website's structure and layout. Picture your website as a canvas awaiting your creative strokes. With a well-defined purpose, an identified audience, and clear design requirements and constraints, it's time to bring your vision to life. This is where the conceptual design takes centre stage. Here's a breakdown:

Blueprint for Structure and Layout:

- Think of conceptual design as the architectural blueprint for your website. It outlines elements' structure, layout, and spatial arrangement across each page.
- To visualise this, consider employing wireframes and mockups. These tools allow you to sketch out the skeletal framework of your site, including the placement of headers, menus, content sections, and interactive features.

Incorporating Content Strategy

A well-thought-out content strategy is integral. Decide what type of content (text, images, videos) will be on each page. Ensure it aligns with your purpose and resonates with your audience. While structural design is crucial, it's equally important to consider what your website will communicate to its visitors. Content strategy plays a pivotal role in this phase. Let's explore why:

Content Types and Alignment:

- Define the types of content that will grace each page of your website. This could include text, images, videos, or a combination.
- Ensure the chosen content aligns seamlessly with your website's purpose and resonates with your target audience. For example, an educational website may feature informative articles and tutorial videos, while a photography portfolio would showcase captivating images.

Purpose-Driven Content:

- Every piece of content should serve a purpose: to inform, entertain, inspire, or persuade. Your content strategy ensures that each word, image, or video contributes meaningfully to the user experience.

Audience-Centric Approach:

- Keep your audience at the forefront when crafting your content strategy. What are their preferences? What questions might they have? Tailor your content to address their needs and interests.

\

Real-World Application:

- E-commerce Website: In the conceptual design phase, you may selectively feature product images on the homepage with clear calls to action. Your content strategy includes product descriptions, prices, and customer reviews that help visitors make purchase decisions.
- Personal Blog: For a blog, the conceptual design could involve a simple yet elegant layout with ample space for text content. Your content strategy would focus on creating engaging and informative blog posts on topics of interest to your readers.

1.4 Installing Necessary Software

Identify the software tools required for website design and development. Common choices include web design software like Adobe XD, Sketch, or Figma and development tools like code editors (e.g., Visual Studio Code, Atom, Bracket, Sublime Text, Text pad, and Notepad++).

Install Visual Studio

Step 1. Make sure your computer is ready for Visual Studio.

Before you begin installing Visual Studio:

- Check the system requirements

  - https://learn.microsoft.com/en-us/visualstudio/releases/2022/system-requirements

  These requirements help you know whether your computer supports Visual Studio 2022.
- Ensure that the user performing the installation has administrator permissions on the machine.
- Apply the latest Windows updates. These updates ensure that your computer has the latest security updates and the required system components for Visual Studio.
- Reboot. The reboot ensures that pending installs or updates don't hinder your Visual Studio install.
- Free up space. Remove unneeded files and applications from your system drive by, for example, running the Disk Cleanup app.

Step 2. Initiate the installation.

The latest release of Visual Studio 2022 is hosted on Microsoft servers. To install this, click the following link and choose your desired edition. A small "bootstrapper" file will be downloaded into your Downloads folder.

- https://visualstudio.microsoft.com/downloads/?cid=learn-onpage-download-cta

- Double-click the bootstrapper from your Downloads folder, VisualStudioSetup.exe, or name something like vs_community.exe to start the installation.
- If you receive a User Account Control notice, choose Yes. A window will ask you to acknowledge the Microsoft License Terms and the Microsoft Privacy Statement. Choose Continue.

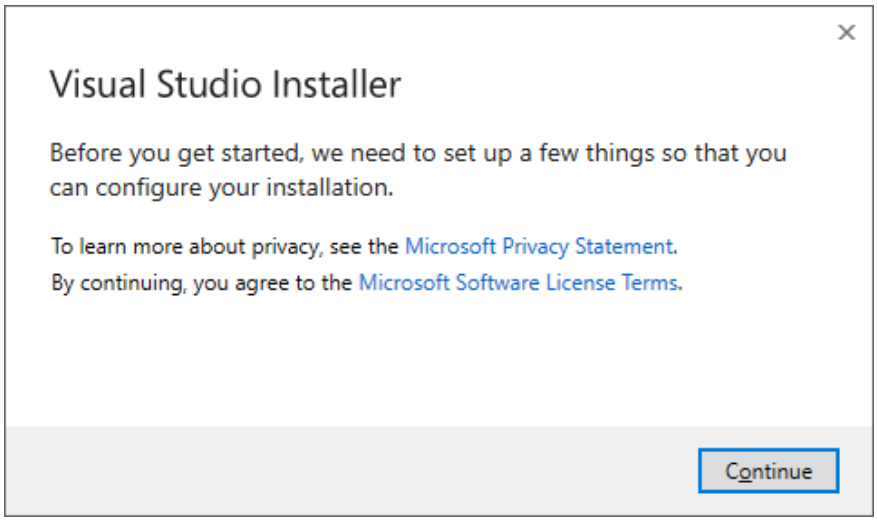


Figure 1: Video Studio Installer.

Step 3. Choose workloads.

After the Visual Studio Installer is installed, you can use it to customize your installation by selecting the feature sets—or workloads—that you want. After you choose the workload(s) you want, select **Install**.

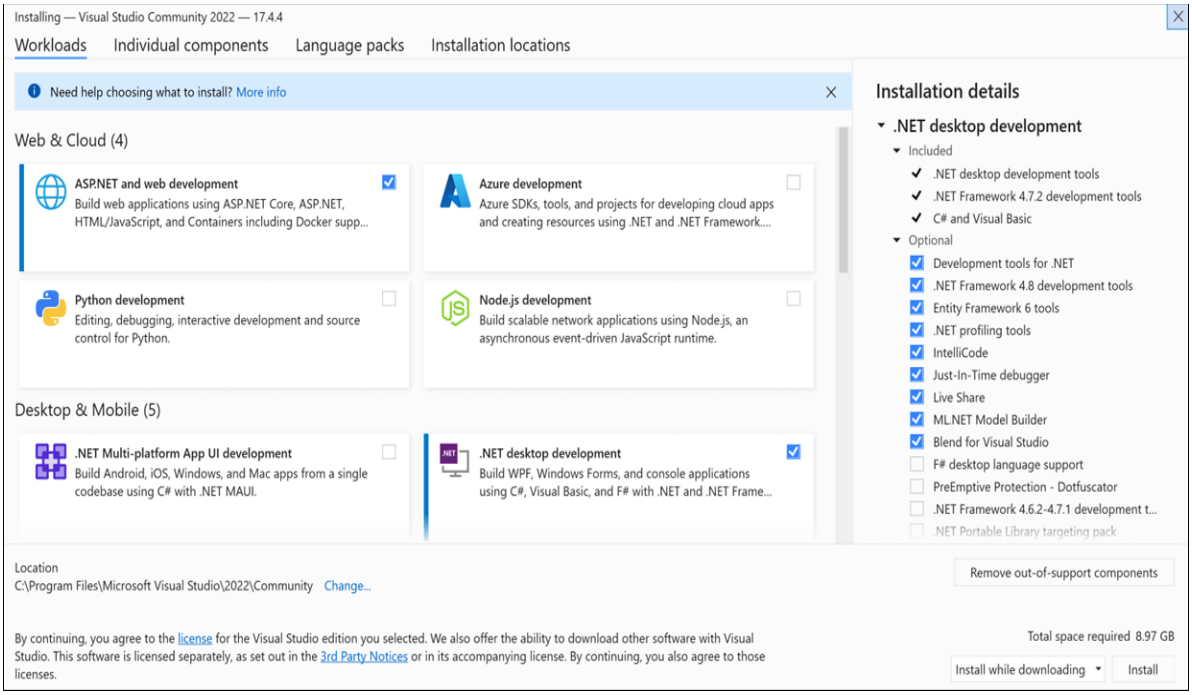


Figure 2: Choose Workloads in Visual Studio.

Step 4. Choose individual components (optional)

If you don't want to use the Workloads feature to customize your Visual Studio installation, or you want to add more components than a workload installs, you can do so by installing or adding individual components from the **Individual components** tab. Choose what you want, and then follow the prompts.

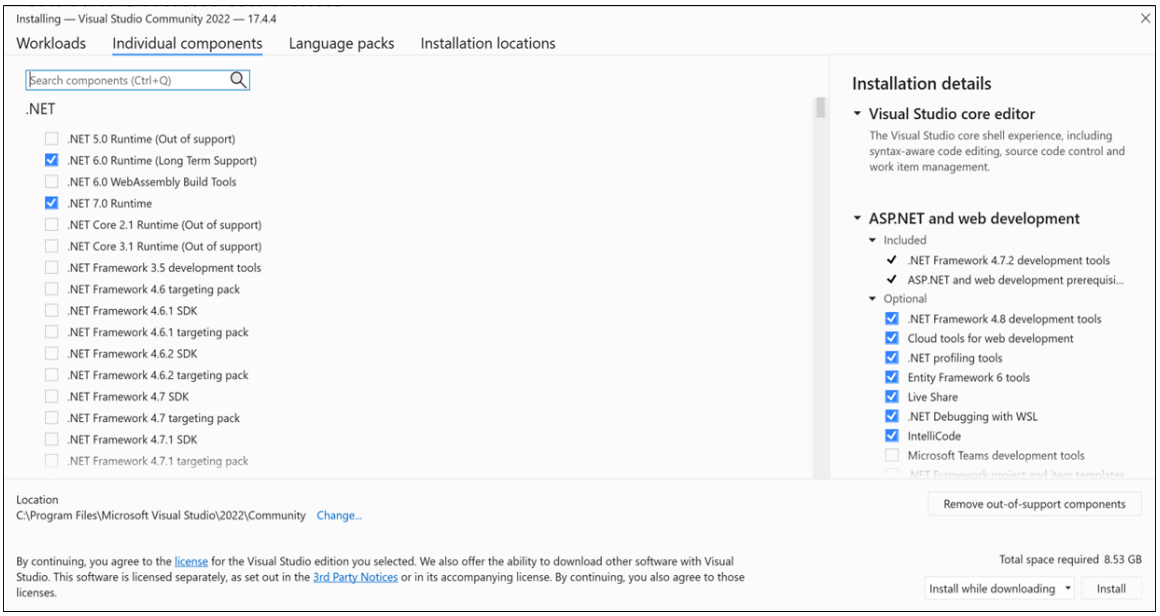


Figure 3: Choose Individual Component in VS Studio.

Step 5. Install language packs (optional)

By default, the installer program tries to match the operating system's language when it runs for the first time. To install Visual Studio in your chosen language, choose the Language Packs tab from the Visual Studio Installer and then follow the prompts.

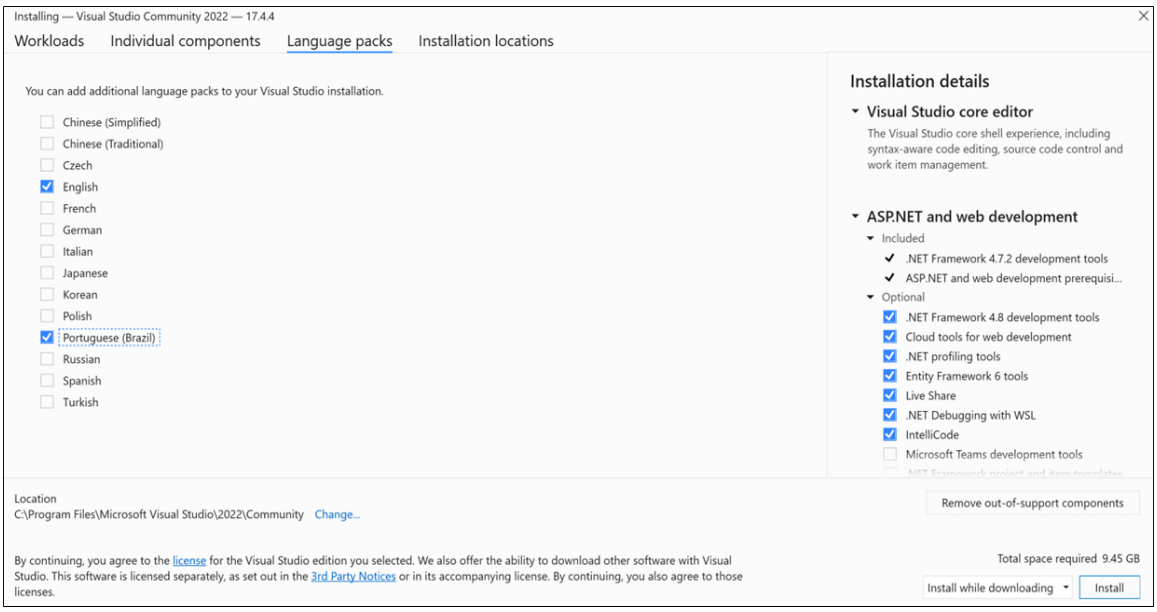


Figure 4: Install language packs in Visual Studio.

Step 6. Select the installation location (optional)

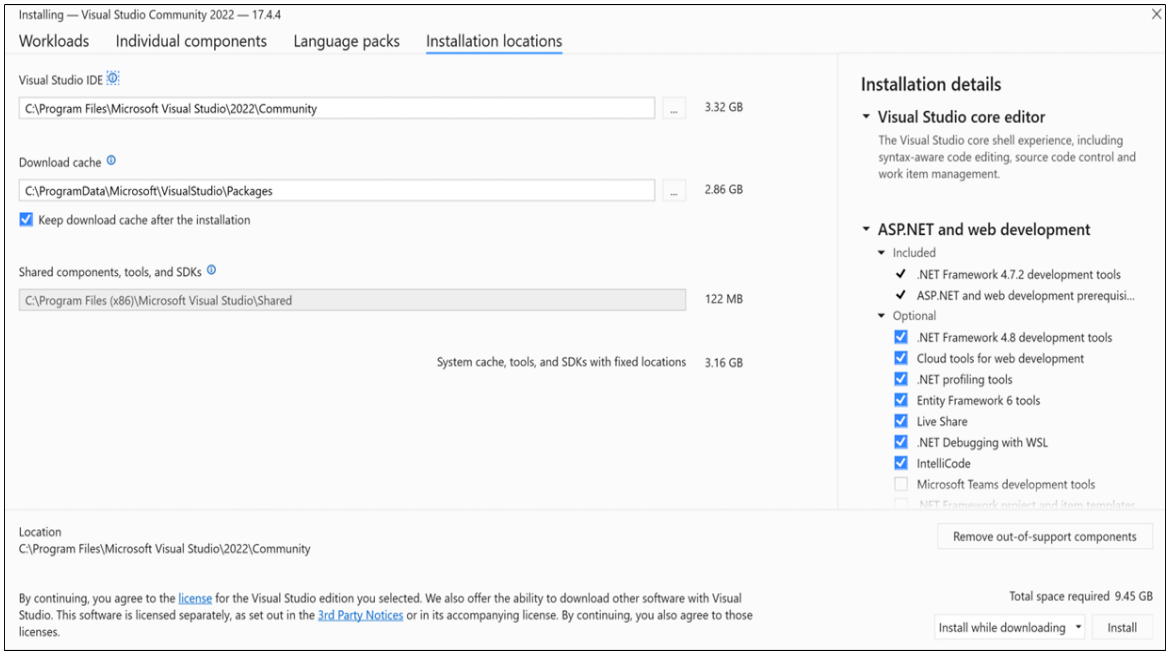


Figure 5: Select the installation location of Visual Studio.

You can select a different drive for Visual Studio IDE or Download cache only when you first install Visual Studio. If you've installed Visual Studio on your computer before, you won't be able to change the Shared components, tools, and SDKs path, and it will appear greyed out. All installations of Visual Studio share this location.

Step 7. Start developing.

- After completing your Visual Studio installation, select the Launch button to start developing with Visual Studio.
- On the start window, choose Create a new project.
- In the template search box, enter the type of app you want to create to see a list of available templates. The list of templates depends on the workloads that you choose during installation. To see different templates, choose different workloads.

  You can also filter your search for a specific programming language by using the Language drop-down list. You can filter by using the Platform list and the Project type list, too.
- Visual Studio opens your new project, and you're ready to code!

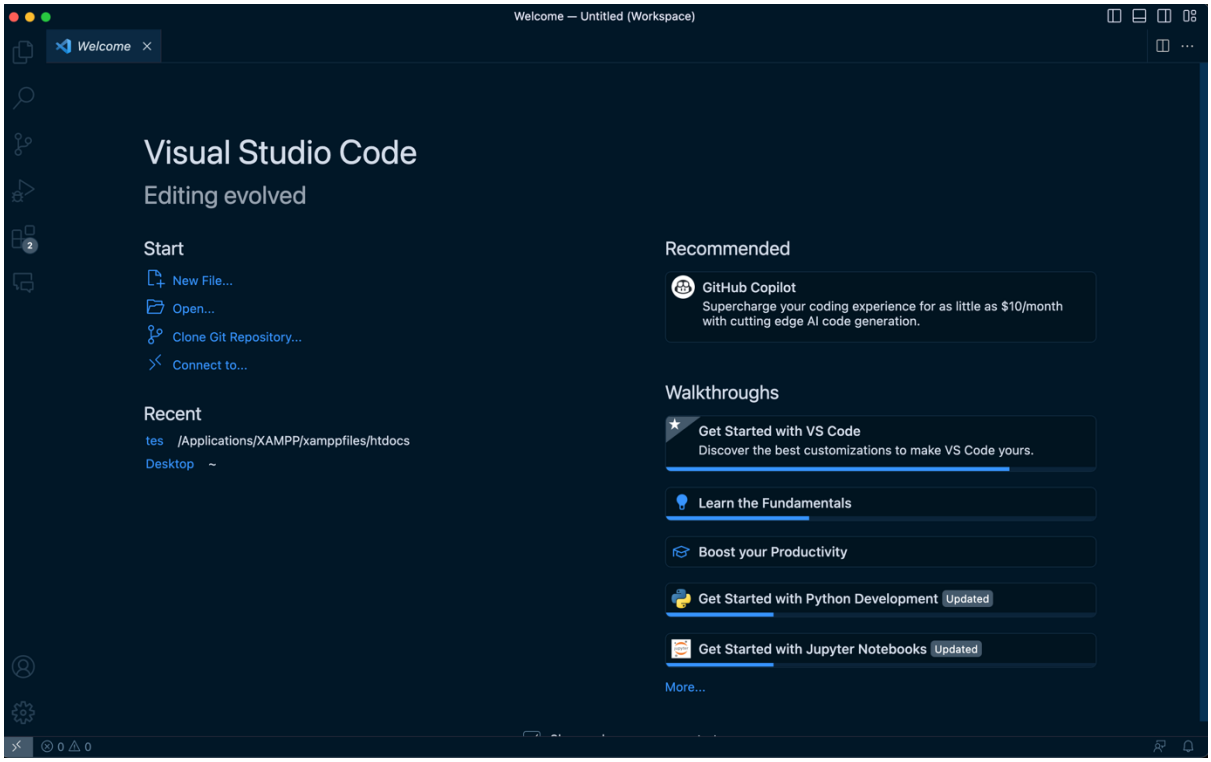


Figure 6: Visual Studio.

***Like this, you can install any other IDE as per your choice. You can also see the documentation of respective IDE sites.*

Self-Check Sheet 1 Plan a website

1. Why is it essential to define the purpose of a website before starting the design and development process?
   Answer:
2. Can you provide an example of how the purpose of a website influences its design?
   Answer:
3. What are the key steps in defining the purpose of a website?
   Answer:
4. Why is selecting an appropriate colour scheme important in design requirements?
   Answer:
5. How does typography impact user experience, and why is it a design requirement?
   Answer:
6. Why is a well-designed user interface (UI) important, and how does it enhance a website's usability?
   Answer:
7. Can you give an example of how technical limitations can impact design choices?
   Answer:
8. How does recognizing constraints upfront benefit the design process?
   Answer:
9. Can you provide an example of how budget constraints might influence design choices?
   Answer:
10. Why is creating a conceptual design important, and how does it relate to the website's purpose?
    Answer:
11. How can wireframes and mockups be helpful in the conceptual design phase?
    Answer:
12. Why is content strategy integral to the conceptual design phase?
    Answer:
13. How does a content strategy reflect the website's purpose and audience?
    Answer:
14. Can you provide an example of purpose-driven content for a website?
    Answer:

Answer Sheet 1 Plan a website

1. Defining a website's purpose is crucial because it provides clarity and focus. It helps set clear goals for the website and ensures that all design and content decisions align with those goals. Without a defined purpose, a website may lack direction and effectiveness.
2. Certainly, consider an e-commerce website. Its primary purpose is to sell products. Therefore, the design should focus on showcasing products effectively, with features like product listings, search functionality, and a smooth checkout process. This purpose guides the design choices.
3. The key steps include identifying your goals for the website and understanding your target audience. Your goals should answer questions like what you want to achieve with the website (e.g., inform, entertain, sell) while understanding your audience helps tailor the content and design to their needs and preferences.
4. The right colour scheme is crucial because colours can evoke emotions and convey messages. It sets the visual tone for the website and can influence how users perceive and interact with it.
5. Typography affects how users consume content. The choice of fonts impacts readability and the overall aesthetics of the website. It's a design requirement because it significantly affects user experience.
6. A well-designed UI enhances usability by providing intuitive elements like buttons and menus. It simplifies navigation, making it easier for users to interact with the website. UI is crucial for a positive user experience.
7. Certainly. If a website has a technical limitation where it must be compatible with older web browsers, this might restrict the use of modern design elements and features that aren't supported in older browsers. Design choices must accommodate this constraint.
8. Recognizing constraints upfront ensures realistic expectations and prevents costly design changes later. It allows designers to work within limitations, whether they are budget, timeline, technical, or compatibility-related, leading to a smoother design process.
9. Certainly. If a project has a limited budget, designers might opt for a simpler design with fewer custom features to save on development costs. This constraint impacts the complexity of the design.
10. A conceptual design is essential because it serves as the blueprint for a website's structure and layout. It ensures the design aligns with the website's purpose by visually organising elements and pages to meet the intended goals.
11. Wireframes and mockups help visualise the layout and arrangement of elements on each page. They provide a clear visual representation of the conceptual design, making planning and communicating design ideas easier.
12. Content strategy is vital because it defines the type of content (text, images, videos) on each page. It ensures that the content aligns with the website's purpose and resonates with the target audience, creating a cohesive user experience.
13. A content strategy tailors the content to address the needs and interests of the target audience while aligning with the website's purpose. For example, an educational website's content strategy focuses on informative content to serve students and educators.
14. Certainly. For an e-commerce website, purpose-driven content includes product descriptions, prices, and customer reviews. These elements aim to inform and persuade visitors to make purchase decisions, aligning to sell products.

Job Sheet 1 IDE Installation for CSS Coding

Job Sheet Title: IDE Installation for CSS Coding

Objective: Install and set up an Integrated Development Environment (IDE).

Tasks:

Step 1. Choose an IDE

- Research and select an IDE suitable for CSS coding. Popular choices include Visual Studio Code, Sublime Text, and Atom.
- Consider factors like user-friendliness, extensions/plugins available, and your operating system's compatibility.

Step 2. Download and Install the IDE

- Visit the official website of the selected IDE using a web browser.
- Locate the download section and download the installer for your operating system (e.g., Windows, macOS, Linux).
- Run the downloaded installer and follow the installation instructions.

Step 3. Basic Setup

- Launch the IDE after installation.
- Customise the IDE settings according to your preferences (e.g., theme, font size, indentation).
- Familiarise yourself with the IDE's user interface.

Step 4. Install CSS Extensions/Plugins

- Most modern IDEs support extensions or plugins to enhance functionality.
- Search for and install CSS-related extensions or plugins. These may include linters, autocompletion, and colour pickers.
- Configure the extensions as needed.

Step 5. Create a Sample CSS File

- Open your new IDE.
- Create a sample CSS file with a .css extension.
- Add some CSS rules to the file to test your environment.

Step 6. Test the Environment

- Write CSS code in your IDE.
- Save the CSS file.
- Open a web browser and create an HTML file.
- Link your HTML file to the CSS file you created.
- View the webpage in the browser to ensure your CSS is applied correctly.

Specification Sheet 1 IDE Installation for CSS Coding

Job Title: IDE Installation for CSS Coding

Necessary tools and equipment

Sl. No	Name of Tools & Equipment	Specification	Unit	Quantity
1	Computer/Laptop	Minimum Corei3 with 4GB RAM	No.	1
3	Software (Browser)	Latest Version	No.	01
4	Internet connections	High Speed	No.	01

Other Specifications:

1. Chosen IDE for CSS - Visual Studio Code
2. Latest version available at the time of installation
3. Operating System: Windows 10
4. Customization:
   - Theme: Dark+
   - Font Size: 16px
   - Tab Width: 4 spaces
   - Plugins/Extensions Installed:
   - "Close Tag" for HTML tag completion
5. Test Procedure:
   1. Launch Visual Studio Code.
   2. Open a sample CSS file.
   3. Write and save CSS code.
   4. Create an HTML file.
   5. Link the HTML file to the CSS file.
   6. Verify that CSS styles are correctly applied to the HTML elements.

Learning Outcome 2: Design the website using cascading style sheets (CSS)

Assessment Criteria	1. Web layout is selected as per design requirement. 2. Web layout is designed using CSS as per client's requirements. 3. HTML and CSS file is integrated as required. 4. Web site is saved and executed.
Conditions and Resources	1. Applicable tools, utensils, and equipment as prescribed by competency standards. 2. Supply materials 3. Relevant ingredients 4. CBLM related to the learning outcome. 5. Instructions, job sheets, activity sheets, and standard operating procedures 6. Personal protective equipment 7. Module/reference
Contents	1. Selecting a Web Layout Based on Design Requirements 2. Designing the Web Layout Using CSS to Meet Client Requirements 3. Integrating HTML and CSS Files as Needed 4. Saving and Executing the Website
Training Methods	1. CBLM 2. Handouts 3. Books, Manuals 4. Module/ Reference 5. Paper 6. Pen
Assessment Methods	1. Written Test 2. Demonstration 3. Oral Questioning

Learning Experience 2: Design the website using cascading style sheets (CSS)

You must perform the learning steps below to achieve the objectives stated in this learning guide. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Steps	Resources specific instructions
1. Students will ask the instructor about design styles with CSS and CSS framework.	1. The instructor will provide the learning materials for design the website using cascading style sheets (SCC)
2. Read the **Information sheet/s**	2. Information Sheet No 1: Design the website using cascading style sheets (CSS) ▪ Web Layout Based on Design Requirements ▪ Web Layout Using CSS to Meet Client Requirements ▪ Integrating HTML and CSS Files as Needed ▪ Saving and Executing the Website
3. Complete the **Self-Checks & Answer key sheets.**	3. Self-Check No 2: Design the website using cascading style sheets (CSS) Answer key No 2: Design the website using cascading style sheets (CSS)
4. Read the **Job/ Task sheet** and **Specification Sheet**	4. Job/ task sheet and specification sheet Job Sheet No 2: Design the website using cascading style sheets (CSS) Specification Sheet 2: Design the website using cascading style sheets (CSS)

Information Sheet 2 Design the website using cascading style sheets (CSS)

Learning Objective: After completing this information sheet, the learners will be able to identify the client-server architecture and its components.

2.1 Web Layout Based on Design Requirements
2.2 Web Layout Using CSS to Meet Client Requirements
2.3 Integrating HTML and CSS Files as Needed
2.4 Saving and Executing the Website

2.1 Web layout Based on design Requirements

Web layout is a critical aspect of web design. It determines how the content is structured and presented to the audience. Choosing the right layout is essential to meet design goals and user expectations when creating web content, whether for a personal blog, e-commerce site, or corporate webpage.

Why Layout Matters

- Aesthetic Appeal: The layout of a website significantly influences its visual appeal. A well-structured layout can captivate users and make a positive first impression.
- User Experience: Layout affects how users navigate and interact with your site. An intuitive layout enhances the user experience by making it easy to find information and perform actions.
- Content Organization: Different types of content require specific layouts. For instance, a news website may use a grid layout for articles, while an e-commerce site might employ a card-based layout for products.
- Responsive Design: Layout choices impact how a website adapts to various devices and screen sizes. Responsiveness is crucial for ensuring a consistent user experience.

Selecting the Right Layout

To choose the appropriate web layout for your project, consider the following factors:

- Content Type: Determine the type of content you'll showcase. Is it primarily text, images, videos, or a combination? Each content type may require a different layout.
- Audience: Understand your target audience's preferences. Different layouts may resonate better with specific demographics.
- Branding: Ensure that the layout aligns with your brand's identity and message. Consistency in design reinforces brand recognition.
- Functionality: Consider the website's functionality. If it's an e-commerce site, the layout should support product listings and a shopping cart. For a blog, it should emphasize readability.

Examples of Layout Types

- Grid Layout: Organizes content into rows and columns, ideal for showcasing products or articles.
- Card-Based Layout: Uses cards to display individual pieces of content, suitable for varied content types.
- Full-Screen Background Layout: Features a large, captivating background image or video with overlaid content, often used for landing pages.
- Single-Column Layout: Presents content in a linear, easy-to-read format, common in blogs and articles.

2.2 Web Layout Using CSS to Meet Client Requirements

Cascading Style Sheets (CSS) is a powerful tool for creating and customising web layouts. CSS allows you to control elements' positioning, size, spacing, and appearance. Applying CSS rules to HTML elements can achieve the desired layout effects.

Understanding CSS: Cascading Style Sheets (CSS) is a stylesheet language that describes a document's presentation and formatting in HTML or XML. It is commonly used in web development to control the appearance of web pages, including elements such as layout, colours, fonts, and animations.

CSS defines rules that determine how specific elements in an HTML document should be displayed. These rules consist of selectors and declarations. Selectors target specific HTML elements, while declarations specify the desired styles for those elements.

Here's a breakdown of the main components of CSS:

- Selectors: Selectors are used to target HTML elements for styling. They can target elements based on their type (e.g., **p** for paragraphs), class (e.g., .**my-class**), ID (e.g., **#my-id**), attributes, or hierarchical relationships between elements (e.g., **div > p** selects paragraphs that are direct children of a div).
- **Declarations:** Declarations define the styles to be applied to the selected elements. A declaration consists of a property and a value separated by a colon. For example, color: red; sets the text color of the selected elements to red.
- **Properties:** Properties define the aspects of an element's appearance that can be modified, such as color, font-size, background-image, margin, and many others.
- **Values:** Values are assigned to properties to specify the desired style. For example, color: red; sets the color property to the value red.
- **Style Rules:** Style rules combine selectors and declarations to define how specific elements should be styled. Multiple style rules can be grouped together in a CSS file or embedded within the <style> tag in an HTML document.

CSS provides a wide range of features and capabilities, including:

- **Box Model:** CSS allows you to control the size, padding, margin, and border of elements.
- **Layout:** CSS provides various techniques to position and arrange elements on a web page, including floating, positioning, and flexible box layout (flexbox) or grid layout (CSS Grid).
- **Typography:** CSS enables you to control the font family, size, weight, style, and other text-related properties.
- **Colors and Backgrounds:** CSS provides options to set colors for text, backgrounds, and borders, including gradients and transparency.
- **Transitions and Animations:** CSS allows you to create smooth transitions and animations, defining how elements should change over time.
- **Media Queries:** CSS supports media queries that let you define different styles for different screen sizes or devices, enabling responsive web design.

CSS can be included in an HTML document using inline, internal, or external stylesheets. External stylesheets are commonly used, where a separate CSS file is linked to the HTML document, allowing consistent styling across multiple web pages.

Why Use CSS?

CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: lightblue;
}

h1 {
  color: white;
  text-align: center;
}

p {
  font-family: verdana;
  font-size: 20px;
}
</style>
</head>
<body>

<h1>My First CSS Example</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Output:

CSS Solved a Big Problem:

HTML was NEVER intended to contain tags for formatting a web page!

HTML was created to describe the content of a web page, like:

```
<h1>This is a heading</h1>
<p>This is a paragraph. </p>
```

When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.

To solve this problem, the World Wide Web Consortium (W3C) created CSS.

CSS removed the style formatting from the HTML page!

Identifying Types of CSS

Cascading Style Sheet (CSS) sets the style in web pages containing HTML elements. It sets the background colour, font size, font family, colour, … etc., properties of elements on a web page.

There are three types of CSS which are given below:

- Inline CSS
- Internal or Embedded CSS
- External CSS

Inline CSS: Inline CSS contains the CSS property in the body section attached to the element, known as inline CSS. This style is specified within an HTML tag using the style attribute.

Example: This example shows the application of inline-CSS

```
<!DOCTYPE html>
<html>
<head>
        <title>Inline CSS</title>
```

```
</head>
<body>
        <p style="color:#009900; font-size:50px;
                        font-style:italic; text-align:center;">
                GeeksForGeeks
        </p>
</body>
</html>
```

Output:

External CSS: External CSS contains separate CSS files that contain only style properties with the help of tag attributes (For example class, id, heading, … etc). CSS property is written in a separate file with a .css extension and should be linked to the HTML document using a link tag. It means that, for each element, style can be set only once and will be applied across web pages.

Example: The file given below contains CSS property. This file saves with .css extension. For Ex: geeks.css

```
body {
    background-color:powderblue;
}
.main {
    text-align:center;
}
.GFG {
    color:#009900;
    font-size:50px;
    font-weight:bold;
}
#geeks {
    font-style:bold;
    font-size:20px;
}
```

Below is the HTML file that is making use of the created external style sheet.

- link tag is used to link the external style sheet with the html webpage.
- href attribute is used to specify the location of the external style sheet file.

```
<!DOCTYPE html>
<html>
<head>
        <link rel="stylesheet" href="geeks.css" />
</head>

<body>
        <div class="main">
                <div class="GFG">GeeksForGeeks</div>
                <div id="geeks">
                        A computer science portal for geeks
                </div>
        </div>
</body>
</html>
```

Priorities of CSS

Inline CSS has the highest priority, then comes Internal/Embedded followed by External CSS which has the least priority. Multiple style sheets can be defined on one page. For an HTML tag, styles can be defined in multiple style types and follow the below order.

- As Inline has the highest priority, any styles that are defined in the internal and external style sheets are overridden by Inline styles.
- Internal or Embedded stands second in the priority list and overrides the styles in the external style sheet.
- External style sheets have the least priority. If there are no styles defined either in inline or internal style sheet then external style sheet rules are applied for the HTML tags.

Inline CSS: Inline CSS refers to including CSS styles directly within individual HTML elements using the style attribute. Inline CSS allows you to apply specific styles to an element without needing an external or internal stylesheet. You'll only need to add the style attribute to each HTML tag for this CSS style, without using selectors.

This CSS type is not really recommended, as each HTML tag needs to be styled individually. Managing your website may become too hard if you only use inline CSS.

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

The style attribute is added to the <p> (paragraph) element in the above example. Within the **style** attribute, CSS properties and values are specified, separated by semicolons. In this case, the text color is set to blue **(color: blue**;) and the font size is set to 16 pixels (**font-size: 16px**;).

Inline CSS can be applied to any HTML element and allows you to override or add specific styles directly to that element. This approach can be useful when you want to apply unique or temporary styles to individual elements without affecting the rest of the page. However, it is generally not recommended for large-scale styling because it can make the code harder to maintain and reuse.

It's worth noting that inline CSS takes precedence over other CSS rules, including styles defined in an external stylesheet or internal <**style**> tags. Therefore, if conflicting styles are applied to the same element through different methods, the inline CSS will take priority.

However, inline CSS in HTML can be useful in some situations. For example, in cases where you don't have access to CSS files or need to apply styles for a single element only.

Let's take a look at an example. Here, we add an inline CSS to the **<p>** and **<h1>** tag:

```
<!DOCTYPE html>
<html>
<body style="background-color:black;">
```

```
<h1 style="color:white;padding:30px;">Hostinger Tutorials</h1>
<p style="color:white;">Something usefull here.</p>

</body>
</html>
```

Advantages of Inline CSS:

- You can easily and quickly insert CSS rules to an HTML page. That's why this method is useful for testing or previewing the changes, and performing quick-fixes to your website.
- You don't need to create and upload a separate document as in the external style.

Disadvantages of Inline CSS:

- Adding CSS rules to every HTML element is time-consuming and makes your HTML structure messy.
- Styling multiple elements can affect your page's size and download time.

Embedded/ Internal: Internal or embedded CSS requires you to add <style> tag in the <head> section of your HTML document. This CSS style is an effective method of styling a single page. However, using this style for multiple pages is time-consuming as you need to put CSS rules on every website page.

Here's how you can use internal CSS:

- Open your HTML page and locate <head> opening tag.
- Put the following code right after the <head> tag
- Add CSS rules on a new line. Here's an example:

```
<style type="text/css">
body {
  background-color: blue;
}
h1 {
  color: red;
  padding: 60px;
}
</style>
```

- Your HTML file will look like this:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: blue;
}
h1 {
  color: red;
  padding: 60px;
}
</style>
</head>
<body>
<h1>Hostinger Tutorials</h1>
<p>This is our paragraph.</p>
</body>
</html>
```

Here's another example of embedded CSS:

```
<!DOCTYPE html>
<html>
<head>
  <title>Embedded CSS Example</title>
  <style>
      p {
          color: blue;
          font-size: 16px;
      }
      h1 {
          color: red;
          font-size: 24px;
      }
```

```
    </style>
</head>
<body>
    <h1>This is a heading with embedded CSS</h1>
    <p>This is a paragraph with embedded CSS</p>
</body>
</html>
```

In the above example, the CSS styles are defined within the <**style**> tags in the <**head>** section. The **p** selector specifies that all **<p>** (**paragraph**) elements should have a text color of blue and a font size of 16 pixels. Similarly, the h1 selector specifies that all **<h1**> (**heading level 1**) elements should have a text color of red and a font size of **24** pixels.

Embedded CSS allows you to keep the styles within the HTML file itself, making it convenient for small-scale projects or when you want to quickly apply styles to a single web page. However, it can become less manageable as the project grows larger or when you need to maintain consistent styles across multiple pages.

When using embedded CSS, the styles defined within the <**style**> tags will take precedence over external stylesheets but will be overridden by inline styles. It follows the cascading nature of CSS, where the order of style application is inline styles < **embedded styles** > external stylesheets.

For more complex or extensive styling, it is generally recommended to use external stylesheets, as they offer better organization, reusability, and ease of maintenance across multiple HTML files.

Advantages of Internal CSS:

- You can use class and ID selectors in this style sheet. Here's an example:

```
.class {
    property1 : value1;
    property2 : value2;
    property3 : value3;
}
```

```
#id {
  property1 : value1;
  property2 : value2;
  property3 : value3;
}
```

- Since you'll only add the code within the same HTML file, you don't need to upload multiple files.

Disadvantages of Internal CSS:

- Adding the code to the HTML document can increase the page's size and loading time.

External CSS: External CSS refers to the practice of linking a separate CSS file to an HTML document. This method allows you to define styles in a separate file and apply them consistently across multiple HTML files. It promotes better organization, reusability, and maintainability of styles in web development.

Here's how external CSS is used:

- Create a CSS file: First, create a separate file with a **.css** extension, such as **styles.css**. This file will contain all the CSS rules and styles.

- Link the CSS file to HTML: In your HTML document's <head> section, add a <**link**> tag that references the **CSS** file. The **href** attribute should point to the location of the CSS file.

```
<!DOCTYPE html>
<html>
<head>
  <title>External CSS Example</title>
  <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
  <h1>This is a heading</h1>
  <p>This is a paragraph</p>
</body>
</html>
```

- Define styles in the CSS file: Open the **CSS** file (**styles.css** in this example) and define the styles using CSS selectors, properties, and values.

```
/* styles.css */

h1 {
  color: red;
  font-size: 24px;
}

p {
  color: blue;
  font-size: 16px;
}
```

In the above example, the CSS file (styles.css) defines styles for <**h1**> and <**p**> elements. The <**h1>** elements will have a red text colour and a font size of **24** pixels, while the **<p>** elements will have a blue text color and a font size of 16 pixels.

External CSS provides several advantages:

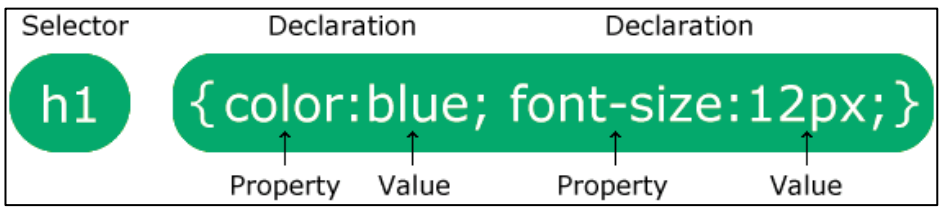
- **Reusability**: You can link the same CSS file to multiple HTML documents, ensuring consistent styles across the website.
- **Maintainability**: Keeping styles separate from the HTML makes managing and updating styles easier without modifying each HTML file.
- **Performance**: By using an external CSS file, the browser can cache the file, resulting in faster subsequent page loads.

Organising the CSS file into sections or groups based on the elements or components they style is common practice, making it easier to find and modify styles when needed.

Overall, external CSS is widely used in web development to achieve scalable, maintainable, and consistent styling across web pages.

Interpreting CSS Syntax

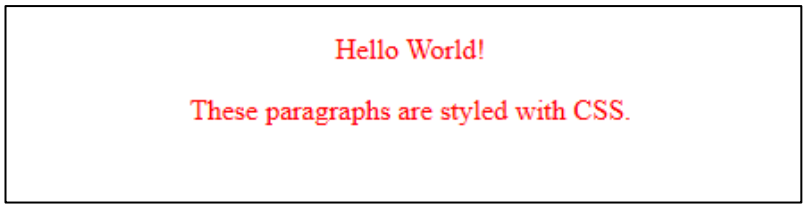
To interpret CSS syntax, it's important to understand the various components and rules that make up valid CSS code. Here's a breakdown of the key aspects of CSS syntax:



- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
- Multiple CSS declarations are separated with semicolons, and curly braces surround declaration blocks.

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  color: red;
  text-align: center;
}
</style>
</head>
<body>

<p>Hello World!</p>
<p>These paragraphs are styled with CSS.</p>

</body>
</html>
```

Output:



Example Explained:

- p is a selector in CSS (it points to the HTML element you want to style: <p>).
- color is a property, and red is the property value
- text-align is a property, and center is the property value

CSS Selectors:

CSS selectors are patterns used to select and target specific HTML elements to apply styles or perform actions. They allow you to define which elements should be affected by the CSS rules you write. CSS provides a wide range of selectors to accommodate different targeting needs. Here are some commonly used CSS selectors:

CSS selectors are used to "find" (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

This page will explain the most basic CSS selectors.

The CSS element Selector

The element selector selects HTML elements based on the element name.

Example: Here, all <p> elements on the page will be center-aligned, with a red text color:

```
p {
  text-align: center;
  color: red;
}
```

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<p>Every paragraph will be affected by the style.</p>
<p id="para1">Me too!</p>
<p>And me!</p>

</body>
</html>
```

Output:

Every paragraph will be affected by the style.

Me too!

And me!

The CSS id Selector

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element is unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

Example
The CSS rule below will be applied to the HTML element with id="para1":

```
#para1 {
  text-align: center;
  color: red;
}
```

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>

</body>
</html>
```

Hello World!

This paragraph is not affected by the style.

The CSS class Selector

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.

```
.center {
  text-align: center;
  color: red;
}
```

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>

</body>
</html>
```

The CSS Universal Selector

The universal selector (*) selects all HTML elements on the page.

```
* {
  text-align: center;
  color: blue;
}
```

```
<!DOCTYPE html>
<html>
<head>
<style>
* {
  text-align: center;
  color: blue;
}
</style>
</head>
<body>

<h1>Hello world!</h1>

<p>Every element on the page will be affected by the style.</p>
<p id="para1">Me too!</p>
<p>And me!</p>

</body>
</html>
```

The CSS Grouping Selector

The grouping selector selects all the HTML elements with the same style definitions.

Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

```
h1 {
  text-align: center;
  color: red;
}

h2 {
  text-align: center;
  color: red;
}

p {
  text-align: center;
  color: red;
}
```

It will be better to group the selectors, to minimize the code.

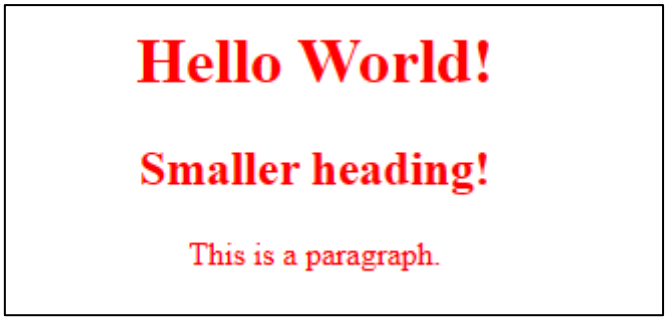
To group selectors, separate each selector with a comma.

Example

In this example we have grouped the selectors from the code above:

```
h1, h2, p {
  text-align: center;
  color: red;
}
```

```
<!DOCTYPE html>
<html>
<head>
<style>
h1, h2, p {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<h1>Hello World!</h1>
<h2>Smaller heading!</h2>
<p>This is a paragraph.</p>

</body>
</html>
```



All CSS Simple Selectors

Selector	Example	Example description
#id	#firstname	Selects the element with id="firstname"
.class	.intro	Selects all elements with class="intro"
element.class	p.intro	Selects only <p> elements with class="intro"
*	*	Selects all elements
element	p	Selects all <p> elements
element,element,...	div, p	Selects all elements and all <p> elements

Creating a CSS File

Creating a CSS (Cascading Style Sheets) file is a straightforward process. CSS defines web documents' visual styles and layout, including HTML pages. Here's a step-by-step guide to creating a CSS file:

In this session, we will write and save our first CSS file. Let's begin by opening a text editing program. If you are on a Microsoft Windows PC, open the program Notepad (hold down the Windows Key on your keyboard and press R, then type Notepad and press enter). If you are using a Macintosh computer, launch the " TextEdit " application (which can be found in your Apps folder).

Let's Write Our First Bit of CSS

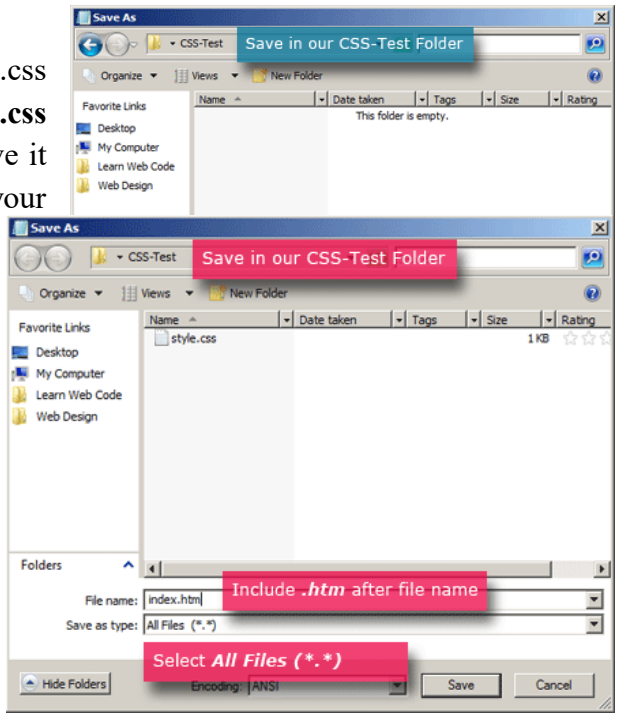
Let's imagine we have a simple web page with a heading and want the heading to be orange and centre-aligned. Add the following code to your new blank text document:

```
h1 {
color: orange;
text-align: center;
}
```

Hopefully, you remember this code from our previous lesson. The task for today is to save our CSS file and link it to an HTML page.

Step 1: Saving the CSS File

Save the file: Save the new file with a .css extension. For example, you can name it **styles.css** or any other relevant name. Make sure to save it in a location where it will be accessible by your HTML files.



Step 2: Linking CSS File to an HTML Page

Link the CSS file to your HTML document: To apply the CSS styles to your HTML document, you need to link the CSS file to it. In the <head> section of your HTML file, add a <link> tag with the rel, type, and href attributes.

Here's an example

```
<link rel="stylesheet" type="text/css" href="styles.css">
```

The HTML File:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>CSS-Test</title>
</head>
<body>
<h1>CSS-Test</h1>
<div id="box-one">
<p>This is box one.</p>
</div>
<div id="box-two">
<p>This is box two.</p>
</div>
</body>
</html>
```

Step 3: Now save this document in our "CSS-Test" folder and name it "index.htm".

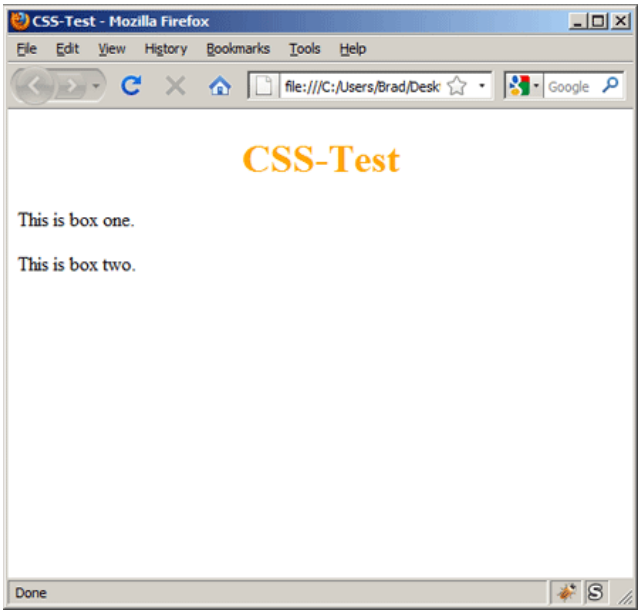
Linking the Two Files Together

We still need to tell the web browser to load our "style.css" file when the "index.htm" page is viewed. Add the following code to "index.htm" directly above our </head> closing tag:

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

This line of code tells our browser that we want to link a Style Sheet, that it's located in the same folder as our HTML page, and that it's named "style.css".

Now, when you view "index.htm" page in a web browser you should see a centered, orange heading:

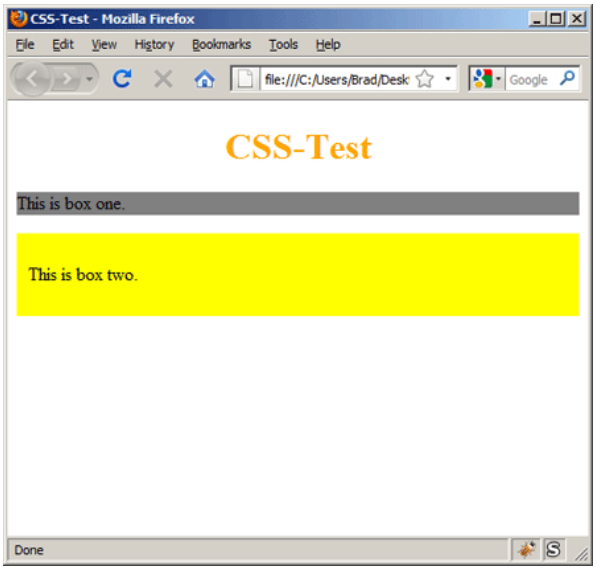


Let's Style Those Two Boxes

If you look at the code of our HTML page, you'll see two elements. We added an HTML attribute of "id" for these two elements and assigned them values of "box-one" and "box-two." We can use an element's "id" to select and style it with CSS. For example, let's make the first box gray, and the second box yellow. Add the following code to your CSS file, directly below our original <h1> rule:

```
#box-one
{
background-color: gray;
}
#box-two {
background-color: yellow;
padding: 10px;
}
```

When an element has an “id” we can access it with a CSS selector by placing a pound sign (#) in front of its id value. So to select the first element we simply type “#box-one” and then begin our CSS Rule.



Our New Fancy Boxes
When you save your CSS file and refresh our HTML page in a web browser you should see something very similar to this:

That's it! Your CSS file is now ready to be used to style your HTML document. Remember to refer to the appropriate selectors in your CSS file to target the desired HTML elements and apply the desired styles.

2.3 Integrating HTML and CSS Files as Needed

Integrating CSS files into your HTML document involves linking the CSS file(s) to your HTML file. Here's a step-by-step guide on how to integrate CSS files:

- **Create or locate your CSS file:** If you haven't already created a CSS file, follow the instructions in the previous response to create one. Make sure you know the file path or location of your CSS file.
- **Open your HTML file:** Open the HTML file you want to integrate the CSS file into using a text editor.
- **Identify the <head> section**: In the HTML file, locate the <**head**> section. This is where you typically include meta tags, title, and other document-related information.
- **Link the CSS file**: Inside the <**head**> section, add a <**link**> tag with the following attributes:
- **rel:** Specifies the relationship between the HTML file and the linked file. In this case, set it to "**stylesheet**".
- **type:** Specifies the MIME type of the linked file. For CSS files, set it to "**text/css**".
- **href:** Specifies the path or URL to the CSS file. Provide the appropriate path to your CSS file.

Here's an example of the <link> tag:

```
<link rel="stylesheet" type="text/css" href="path/to/styles.css">
```

Replace "path/to/styles.css" with the actual path or URL to your CSS file.

- Save your HTML file: Save the changes made to your HTML file.

Now, when you open your HTML file in a web browser, it will load the CSS file specified in the <**link**> tag, and the styles defined in the CSS file will be applied to the HTML elements according to the selectors used in the CSS rules.

You can also link multiple CSS files to a single HTML file by including multiple <**link**> tags, each pointing to a different CSS file. This allows you to modularize your stylesheets and keep them organized.

Implementing CSS for Layout

Implementing CSS for layout involves using CSS properties and techniques to structure and position elements on a webpage. Here are some commonly used CSS techniques for layout:

1 **Box Model:** The box model is fundamental to CSS layout. Each HTML element is represented as a rectangular box, consisting of content, padding, border, and margin. You can control the size and spacing of elements using properties like width, height, padding, border, and margin.

2 **Display Property:** The display property specifies how an element should be displayed. The most common values are:

- block: Makes an element a block-level element, taking up the full width of its parent container.
- inline: Makes an element an inline-level element, allowing other elements to be next to it.
- inline-block: Combines aspects of block and inline, allowing an element to have block properties (e.g., width, height) while still being inline.
- none: Hides an element by removing it from the document flow.

3 Positioning: The position property controls how elements are positioned within their parent container. Common values include:

- static: The default positioning. Elements follow the normal flow of the document.
- relative: Elements are positioned relative to their normal position using properties like top, bottom, left, and right.
- absolute: Elements are positioned relative to the nearest positioned ancestor. If there is no positioned ancestor, it's relative to the document body.

- fixed: Elements are positioned relative to the browser window, so they stay in the same position even when scrolling.

4 Floats: The float property allows elements to be positioned side by side, either to the left or right of their container. Floated elements are removed from the normal document flow, affecting the positioning of other elements around them.

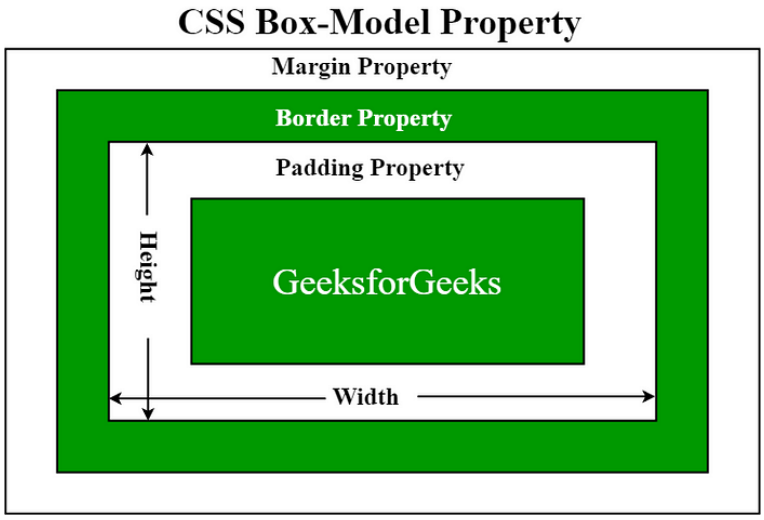
3. Flexbox: Flexbox is a powerful CSS layout module that provides flexible and responsive layouts. By setting the display property of a container to flex, you can use properties like flex-direction, justify-content, align-items, and flex to control the positioning and alignment of child elements.

4. Grid Layout: CSS Grid Layout allows you to create complex two-dimensional layouts. By setting the display property of a container to grid, you can define rows and columns and control the placement of elements within the grid using properties like grid-template-rows, grid-template-columns, and grid-area.

Applying CSS Box Model and Positioning

The CSS Box Model: All HTML elements can be considered as boxes. In CSS, the term "box model" is used when discussing design and layout.

The CSS box model is a box that wraps around every HTML element. It consists of margins, borders, padding, and content. The image below illustrates the box model:



- **Content —** The content of the box, where text and images appear
- **Padding —** Clears an area around the content. The padding is transparent
- **Border —** A border that goes around the padding and content
- **Margin —** Clears an area outside the border. The margin is transparent

The box model allows us to add a border around elements, and to define space between elements.

Width and Height of an Element
In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

Important: When you set the width and height properties of an element with CSS, you just set the width and height of the content area. To calculate the full size of an element, you must also add padding, borders and margins.

Example
This element below will have a total width of 350px:

```
div {
width: 320px;
padding: 10px;
border: 5px solid gray;
margin: 0;
}
```

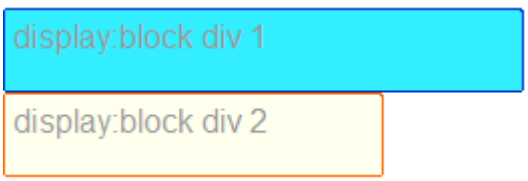
Understanding CSS Positioning

Positioning elements with CSS in web development isn't as easy as it seems. Things can get quickly complicated as your project gets bigger and without having a good understanding of how CSS deals with aligning HTML elements, you won't be able to fix your alignment issues.

There are different ways/methods for positioning elements with pure CSS. Using CSS float, display and position properties are the most common methods. In this article, I will be explaining one of the most confusing ways for aligning elements with pure CSS: the position property.

There are basically two types of ways to display an element in CSS: block and inline.

Display:block;
Block can be, quite literally, seen as a block. It has a specified width and height, optionally controlled by its content, but dimensions set by CSS have prevalence. Another property of elements with display:block is that they do not allow elements on the same "line" as their own. So while an element might have a small width, The next element will always be placed under it. So two simple divs with display block look like this:



Display:inline;
Display:inline is somewhat the opposite. Elements with display:inline always take their width and height from the contents, and will ignore any dimensions specified in the CSS. Another opposite is that, as the name suggest, these elements are displayed in-line, so they will always be next to the preceding element, providing that the preceding element is inline as well and that there is enough width left. If the width of the "line" is filled, an element with display:inline will wrap to the next line. That looks like this:

There is a multitude of other display: properties such as table and inline-block.

Flow
With positioning, the elusive "Flow" of CSS positioning comes into play. The flow in CSS is the logical way in which elements get placed on your screen. For example, when you turn off the CSS on this page, you see the HTML in its original order. That order is the flow, and dictates which element gets rendered and placed where. It's very simple: the flow of a page is from the top of the HTML to the bottom of it.

The position Property:
The position property specifies the type of positioning method used for an element.

There are five different position values:
- static
- relative
- fixed
- absolute
- sticky

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.
Now let's move on with the position property values.

position: static;
- HTML elements are positioned static by default.
- Static positioned elements are not affected by the top, bottom, left, and right properties.
- An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page.

```
div {
  position: static;
}
```

position: relative;
An element with position: relative; is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

```
div {
  position: relative;
  top: 20px;
  left: 30px;
}
```

position: fixed;
An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

```
div {
 position: fixed;
 top: 10px;
 right: 20px;
}
```

A fixed element does not leave a gap in the page where it would normally have been located.

position: absolute;
An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

```
div {
 position: absolute;
 top: 50px;
 left: 100px;
}
```

position: sticky;
An element with position: sticky; is positioned based on the user's scroll position.

A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

Overlapping Elements
When elements are positioned, they can overlap other elements.

The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).

An element can have a positive or negative stack order.

Applying CSS Transitions and Gradients

CSS Transitions and Gradients are powerful features that allow you to add smooth animations and gradient backgrounds to your elements. Here's an overview of how to apply CSS Transitions and Gradients:

CSS Transitions:

CSS Transitions enable smooth animations when a CSS property changes its value over a specified duration. You can specify which properties to transition, the duration of the transition, and the timing function that controls the transition speed.

Here's an example of applying a transition effect to a <**div**> element when its background color changes:

```
div {
  transition: background-color 0.3s ease;
}
div:hover {
  background-color: blue;
}
```

In this example, when you hover over the element, the background color will smoothly transition to blue over a duration of 0.3 seconds with an ease timing function. You can adjust the transition duration, timing function, and other properties as needed.

CSS Gradients:

CSS Gradients allow you to create smooth color transitions between two or more specified colors. You can apply gradients to backgrounds, borders, or even text.

Here's an example of applying a linear gradient background to a element:

```
div {
  background-image: linear-gradient(to right, red, blue);
}
```

In this example, the element will have a linear gradient background that transitions from red to blue from left to right. You can customize the gradient direction and add more color stops to create complex gradients.

Additionally, you can apply radial gradients using the radial-gradient() function and add color stops to control the smooth transition between colors.

```
div {
  background-image: radial-gradient(circle, yellow, orange, red);
}
```

In this example, the element will have a radial gradient background that transitions from yellow to orange to red in a circular pattern.

Adjust the CSS selectors and properties to target the appropriate HTML elements and achieve the desired effects. CSS Transitions and Gradients offer a wide range of possibilities to enhance the visual appeal of your web pages.

Applying 2D/3D Transformations and Animations

CSS 2D/3D Transformations and Animations allow you to manipulate and animate elements visually appealingly. Here's an overview of how to apply these CSS features:

CSS 2D/3D Transformations:
CSS Transformations allow you to modify elements' position, size, and rotation in 2D or 3D space. You can use properties like transform, translate, rotate, scale, and more to apply transformations.

CSS also supports 3D transformations.
Mouse over the elements below to see the difference between a 2D and a 3D transformation:

In this chapter you will learn about the following CSS property:

- **Transform**

Browser Support

The numbers in the table specify the first browser version that fully supports the property.

Property					
transform	36.0	10.0	16.0	9.0	23.0

CSS 3D Transforms Methods

With the CSS transform property you can use the following 3D transformation methods:

- rotateX()
- rotateY()
- rotateZ()

The rotateX() Method
The rotateX() method rotates an element around its X-axis at a given degree:

Example

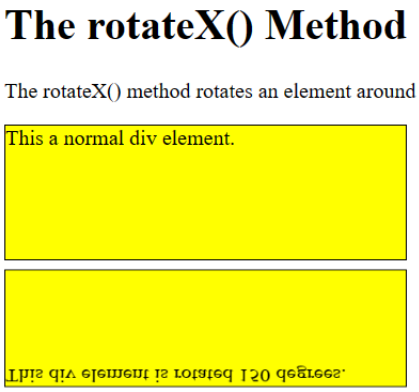
```
#myDiv {
  transform: rotateX(150deg);
}
```

Code:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
}
#myDiv {
  transform: rotateX(150deg);
}
</style>
</head>
<body>
<h1>The rotateX() Method</h1>
<p>The rotateX() method rotates an element around its X-axis at a given
degree.</p>
<div>
This a normal div element.
</div>
<div id="myDiv">
This div element is rotated 150 degrees.
</div>
</body>
```

```
</html>
```

Output:



The rotateY() Method

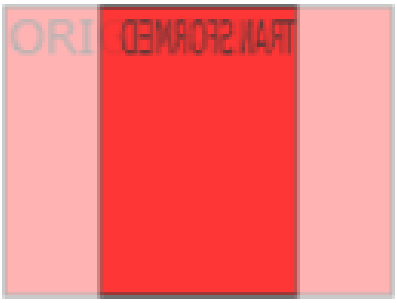
The rotateY() method rotates an element around its Y-axis at a given degree:

Example

```
#myDiv {
  transform: rotateY(150deg);
}
```

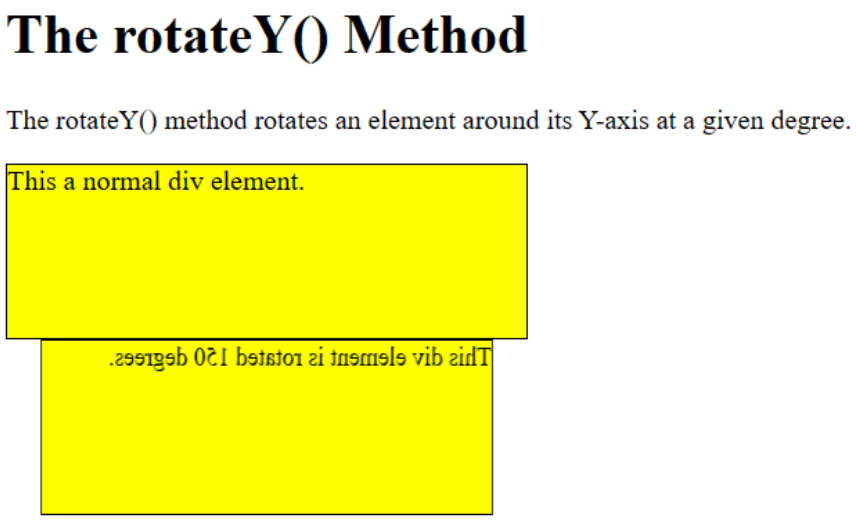
Code:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
}

#myDiv {
  transform: rotateY(150deg);
}
</style>
```



```
</head>
<body>

<h1>The rotateY() Method</h1>

<p>The rotateY() method rotates an element around its Y-axis at a given degree.</p>

<div>
This a normal div element.
</div>

<div id="myDiv">
This div element is rotated 150 degrees.
</div>

</body>
</html>
```

Output:



The rotateZ() Method

The rotateZ() method rotates an element around its Z-axis at a given degree:

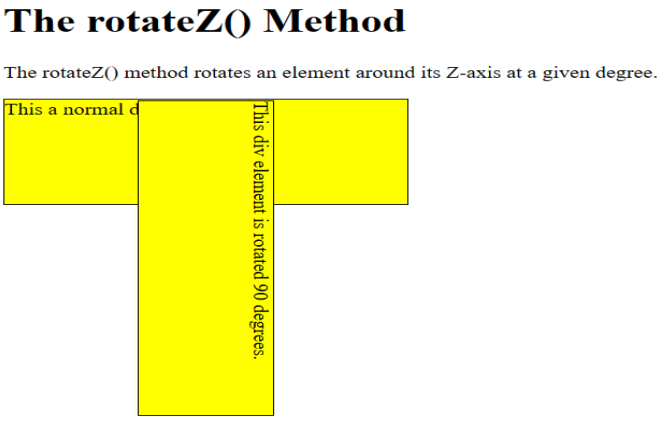
Example

```
#myDiv {
  transform: rotateZ(90deg);
}
```

Code:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
}
#myDiv {
  transform: rotateZ(90deg);
}
</style>
</head>
<body>
<h1>The rotateZ() Method</h1>
<p>The rotateZ() method rotates an element around its Z-axis at a given degree.</p>

<div>
This a normal div element.
</div>
<div id="myDiv">
This div element is rotated 90 degrees.
</div>
</body>
</html>
```

Output:



CSS Transform Properties

Property	Description
transform	Applies a 2D or 3D transformation to an element
transform-origin	Allows you to change the position on transformed elements
transform-style	Specifies how nested elements are rendered in 3D space
perspective	Specifies the perspective on how 3D elements are viewed
perspective-origin	Specifies the bottom position of 3D elements
backface-visibility	Defines whether or not an element should be visible when not facing the screen

CSS 3D Transform Methods

Function	Description
matrix3d (*n,n,n,n,n,n,n,n,n,n,n,n,n,n,n,n*)	Defines a 3D transformation, using a 4x4 matrix of 16 values
translate3d(*x,y,z*)	Defines a 3D translation
translateX(*x*)	Defines a 3D translation, using only the value for the X-axis
translateY(*y*)	Defines a 3D translation, using only the value for the Y-axis
translateZ(*z*)	Defines a 3D translation, using only the value for the Z-axis
scale3d(*x,y,z*)	Defines a 3D scale transformation
scaleX(*x*)	Defines a 3D scale transformation by giving a value for the X-axis
scaleY(*y*)	Defines a 3D scale transformation by giving a value for the Y-axis
scaleZ(*z*)	Defines a 3D scale transformation by giving a value for the Z-axis
rotate3d(*x,y,z,angle*)	Defines a 3D rotation
rotateX(*angle*)	Defines a 3D rotation along the X-axis
rotateY(*angle*)	Defines a 3D rotation along the Y-axis
rotateZ(*angle*)	Defines a 3D rotation along the Z-axis
perspective(*n*)	Defines a perspective view for a 3D transformed element

CSS Animations:

CSS Animations allow you to create dynamic and interactive effects by specifying keyframes and animating CSS properties over time. You can define keyframes using the @keyframes rule and apply animations using the animation property.

Here's an example of applying a CSS animation to a element:

```
div {
  animation-name: slide;
  animation-duration: 2s;
  animation-timing-function: ease;
  animation-delay: 1s;
  animation-iteration-count: infinite;
  animation-direction: alternate;
}

@keyframes slide {
  0% { transform: translateX(0); }
  50% { transform: translateX(200px); }
  100% { transform: translateX(0); }
}
```

In this example, the <**div**> element will slide horizontally back and forth continuously. The animation is defined using the **@keyframes** rule with three keyframes representing the initial position, midpoint, and final position of the animation. Properties like **animation-duration, animation-timing-function, animation-delay, animation-iteration-count**, and **animation-direction** control various aspects of the animation.

You can also animate other CSS properties like opacity, color, width, and more to create a wide range of visual effects.

Defining a Responsive Layout

A responsive layout refers to the design and development approach that aims to create web pages or applications that adapt and respond effectively to various screen sizes, devices, and orientations. The goal is to provide an optimal user experience, regardless of whether the user is accessing the content on a desktop computer, laptop, tablet, or smartphone.
Responsive Web design is the approach that suggests that design and development should respond to the user's behavior and environment based on screen size, platform, and orientation.

The practice consists of a mix of flexible grids and layouts, images, and an intelligent use of CSS media queries. As the user switches from their laptop to iPad, the website should automatically switch to accommodate for resolution, image size and scripting abilities. One may also have to consider the settings on their devices; if they have a VPN for iOS on their iPad, the website should not block the user's access to the page. In other words, the website should have the technology to respond automatically to the user's preferences. This would eliminate the need for a different design and development phase for each new gadget.

Here are some key principles and techniques commonly used in creating responsive layouts:

1 **Fluid Grids:** Instead of fixed pixel-based layouts, responsive designs use fluid grids that are based on relative proportions. This allows elements to resize and reposition dynamically based on the screen size.

2 **Flexible Images and Media:** Images and media elements, such as videos or embedded content, are made responsive by adjusting their size and scaling proportionally to fit different devices.

3 **Media Queries:** CSS media queries are used to apply different styles and layouts based on the characteristics of the user's device, such as screen width, height, and orientation. Media queries enable developers to target specific devices or ranges of screen sizes and adapt the design accordingly.

4 **Breakpoints:** Breakpoints are specific points in the responsive design where the layout and content rearrange or change to fit the screen size better. These breakpoints are often determined based on common device widths or popular screen resolutions.

5 **Mobile-First Approach:** With the increasing dominance of mobile devices, many designers and developers adopt a mobile-first approach. This involves initially designing and developing for smaller screens and then progressively enhancing the layout and features as the screen size increases.

6 **Content Prioritization:** Responsive layouts often involve prioritizing content based on its importance and relevance to the user. This ensures that critical information is readily accessible, even on smaller screens, while less essential content may be rearranged or hidden.

By employing these techniques and principles, a responsive layout can provide a seamless and optimized user experience across a wide range of devices, improving accessibility and user engagement.

The Concept of Responsive Web Design

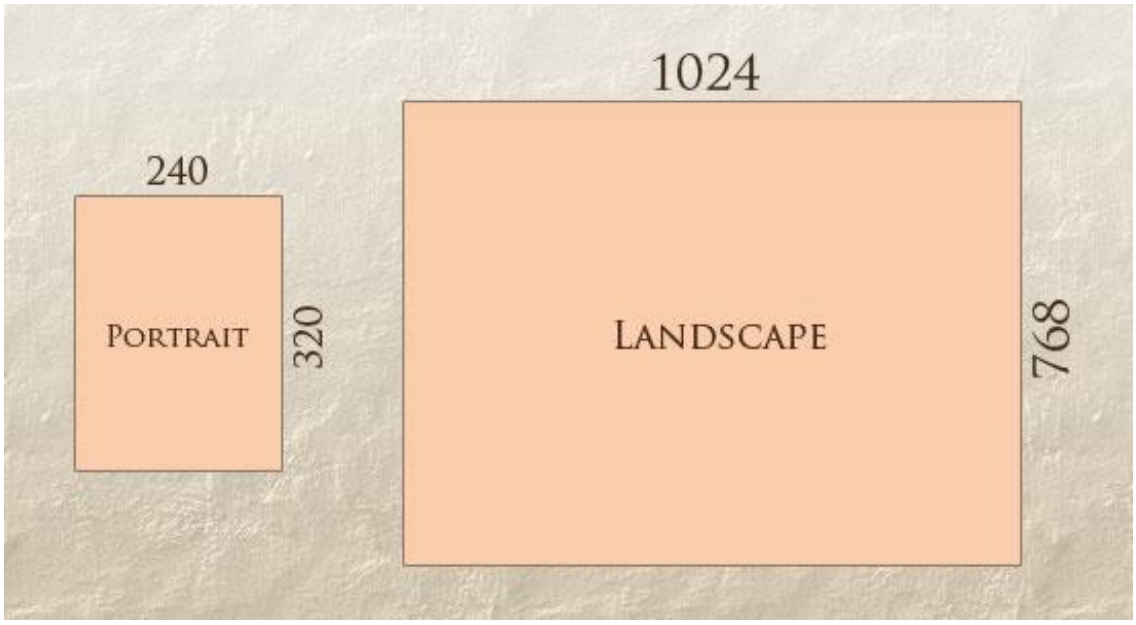
Transplant this discipline onto Web design, and we have a similar yet whole new idea. Why should we create a custom Web design for each group of users; after all, architects don't design a building for each group size and type that passes through it? Like responsive architecture, Web design should automatically adjust. It shouldn't require countless custom-made solutions for each new category of users.

We can't use motion sensors and robotics to accomplish this the way a building would. Responsive Web design requires a more abstract way of thinking. However, some ideas are already being practised: fluid layouts, media queries and scripts that can reformat Web pages and mark up effortlessly (or automatically).

But responsive Web design is not only about adjustable screen resolutions and automatically resizable images but rather about a whole new way of thinking about design. Let's talk about all these features and additional ideas in the making.

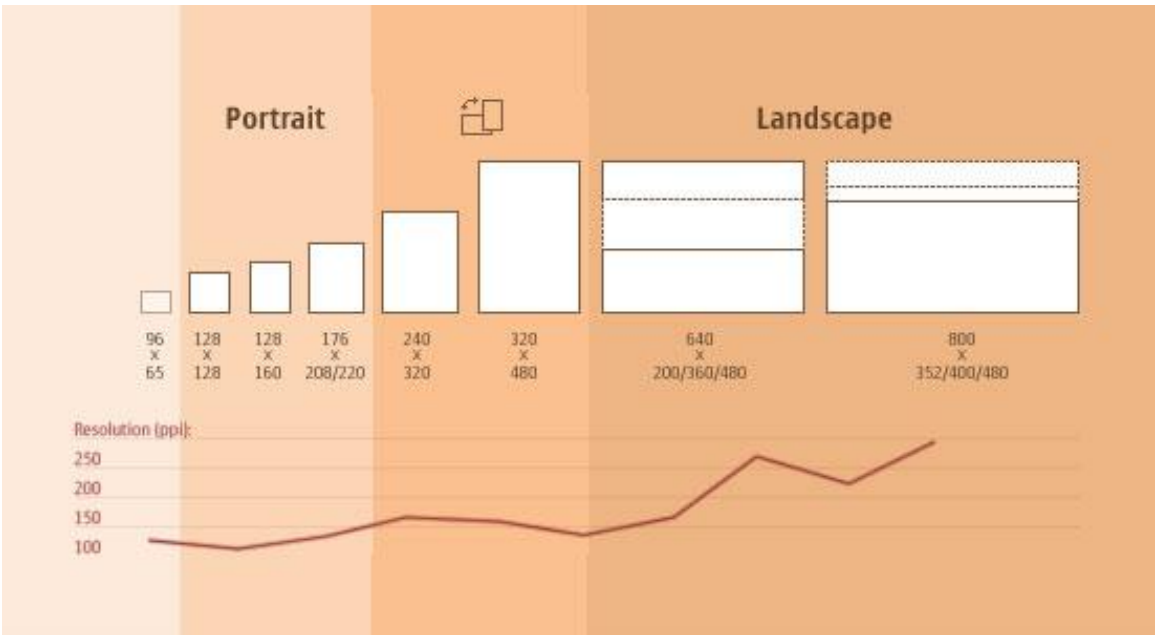
Adjusting Screen Resolution

With more devices come varying screen resolutions, definitions and orientations. New devices with new screen sizes are being developed every day, and each of these devices may be able to handle variations in size, functionality and even color. Some are in landscape, others in portrait, still others even completely square. As we know from the rising popularity of the iPhone, iPad and advanced smartphones, many new devices are able to switch from portrait to



In addition to designing for both landscape and portrait (and enabling those orientations to possibly switch in an instant upon page load), we must consider the hundreds of different screen sizes. Yes, it is possible to group them into major categories, design for each of them, and make each design as flexible as necessary. But that can be overwhelming, and who knows what the usage figures will be in five years? Besides, many users do not maximize their browsers, which itself leaves far too much room for variety among screen sizes.

Morten Hjerde and a few of his colleagues identified statistics on about 400 devices sold



between 2005 and 2008. Below are some of the most common:

Since then even more devices have come out. It's obvious that we can't keep creating custom solutions for each one. So, how do we deal with the situation?

PART OF THE SOLUTION: FLEXIBLE EVERYTHING

A few years ago, when flexible layouts were almost a "luxury" for websites, the only things that were flexible in a design were the layout columns (structural elements) and the text. Images could easily break layouts, and even flexible structural elements broke a layout's form when pushed enough. Flexible designs weren't really that flexible; they could give or take a few hundred pixels, but they often couldn't adjust from a large computer screen to a netbook.

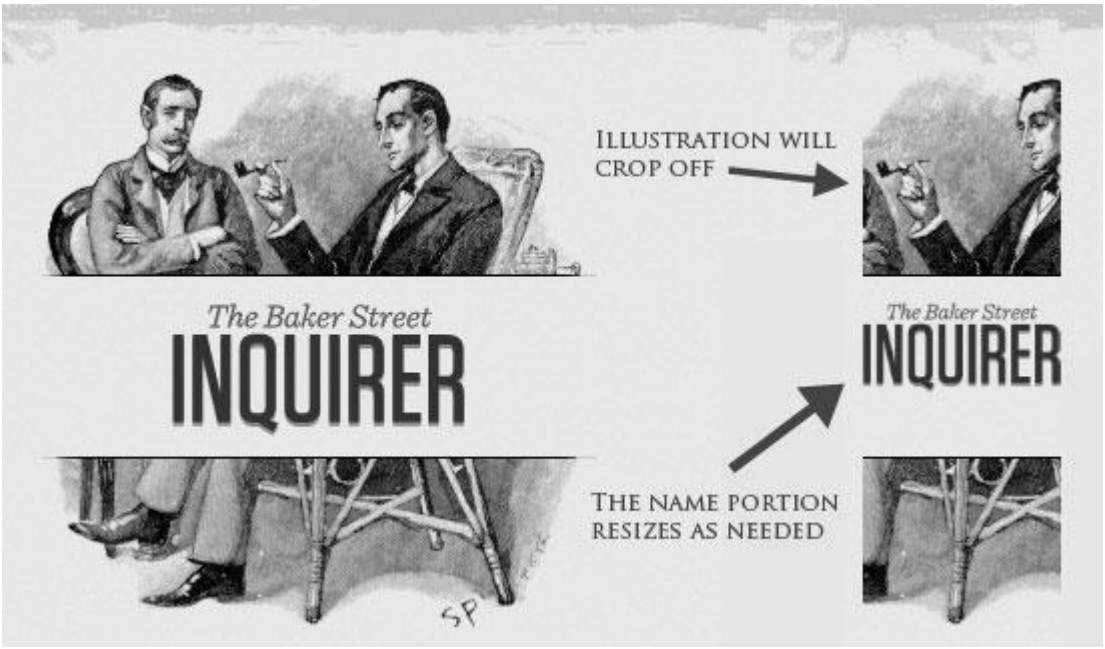
Now we can make things more flexible. Images can be automatically adjusted, and we have workarounds so that layouts never break (although they may become squished and illegible in the process). While it's not a complete fix, the solution gives us far more options. It's perfect for devices that switch from portrait orientation to landscape in an instant or for when users switch from a large computer screen to an iPad.

In Ethan Marcotte's article, he created a sample Web design that features this better flexible layout. The entire design is a lovely mix of fluid grids, fluid images and smart mark-up where needed. Creating fluid grids is fairly common practice, and there are a number of techniques for creating fluid images:

- Hiding and Revealing Portions of Images
- Creating Sliding Composite Images
- Foreground Images That Scale With the Layout

For more information on creating fluid websites, be sure to look at the book “Flexible Web Design: Creating Liquid and Elastic Layouts with CSS” by Zoe Mickley Gillenwater, and download the sample chapter “Creating Flexible Images.” In addition, Zoe provides the following extensive list of tutorials, resources, inspiration and best practices on creating flexible grids and layouts: “Essential Resources for Creating Liquid and Elastic Layouts”.

While from a technical perspective this is all easily possible, it’s not just about plugging these features in and being done. Look at the logo in this design, for example:



Logo example were the image is divided in two: the background, set to be cropped and to maintain its size, and the other image resized proportionally.

If resized too small, the image would appear to be of low quality, but keeping the name of the website visible and not cropping it off was important. So, the image is divided into two: one (of the illustration) set as a background, to be cropped and to maintain its size, and the other (of the name) resized proportionally.

```
<h1 id="logo"><a href="#"><img src="site/logo.png" alt="The Baker Street Inquirer" /></a></h1>
```

Above, the h1 element holds the illustration as a background, and the image is aligned according to the container’s background (the heading).

This is just one example of the kind of thinking that makes responsive Web design truly effective. But even with smart fixes like this, a layout can become too narrow or short to look right. In the logo example above (although it works), the ideal situation would be to not crop

half of the illustration or to keep the logo from being so small that it becomes illegible and "floats" up.

Flexible Images

One major problem that needs to be solved with responsive Web design is working with images. There are a number of techniques to resize images proportionately, and many are easily done. The most popular option, noted in Ethan Marcotte's article on fluid images but first experimented with by Richard Rutter, is to use CSS's max-width for an easy fix.

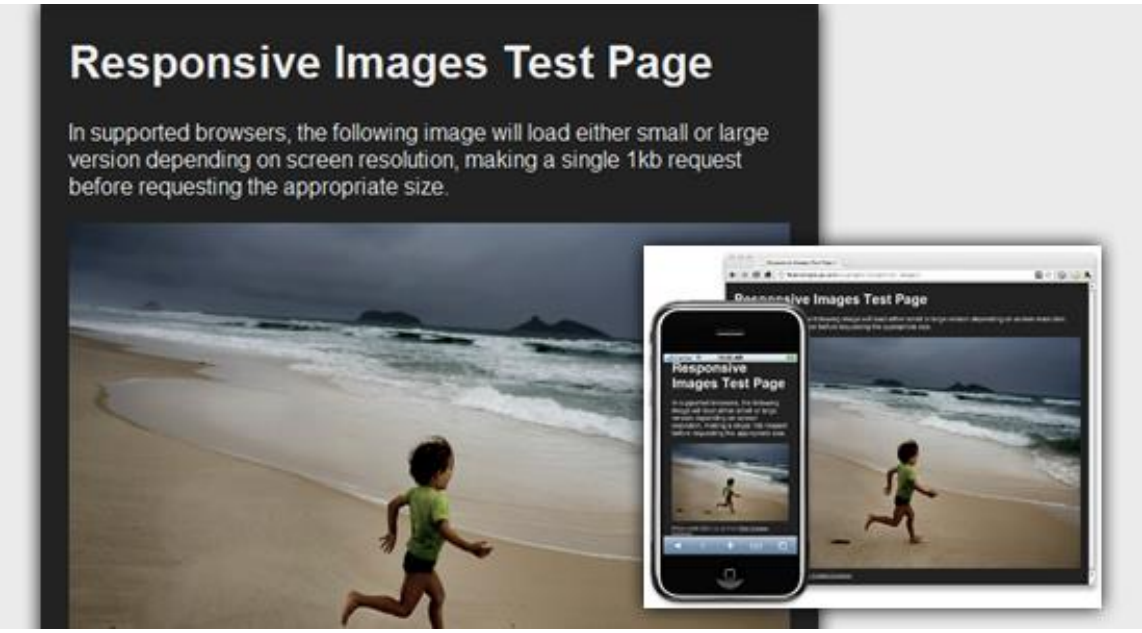
```
img { max-width: 100%; }
```

As long as no other width-based image styles override this rule, every image will load in its original size, unless the viewing area becomes narrower than the image's original width. The maximum width of the image is set to 100% of the screen or browser width, so when that 100% becomes narrower, so does the image. As Jason Grigsby noted, "The idea behind fluid images is that you deliver images at the maximum size they will be used at. You don't declare the height and width in your code, but instead let the browser resize the images as needed while using CSS to guide their relative size". It's a great and simple technique to resize images beautifully.

Note that max-width is not supported in IE, but a good use of width: 100% would solve the problem neatly in an IE-specific style sheet. One more issue is that when an image is resized too small in some older browsers in Windows, the rendering isn't as clear as it ought to be. There is a JavaScript to fix this issue, though, found in Ethan Marcotte's article.

FILAMENT GROUP'S RESPONSIVE IMAGES

This technique, presented by the Filament Group, takes this issue into consideration and not only resizes images proportionately, but shrinks image resolution on smaller devices, so very large images don't waste space unnecessarily on small screens.



This technique requires a few files, all of which are available on Github. First, a JavaScript file (rwd-images.js), the .htaccess file and an image file (rwd.gif). Then, we can use just a bit of HTML to reference both the larger and smaller resolution images: first, the small image, with an .r prefix to clarify that it should be responsive, and then a reference to the bigger image using data-fullsrc.

```
<img src="smallRes.jpg" data-fullsrc="largeRes.jpg">
```

The **data-fullsrc** is a custom HTML5 attribute, defined in the files linked to above. For any screen that is wider than 480 pixels, the larger-resolution image (largeRes.jpg) will load; smaller screens wouldn't need to load the bigger image, and so the smaller image (smallRes.jpg) will load.

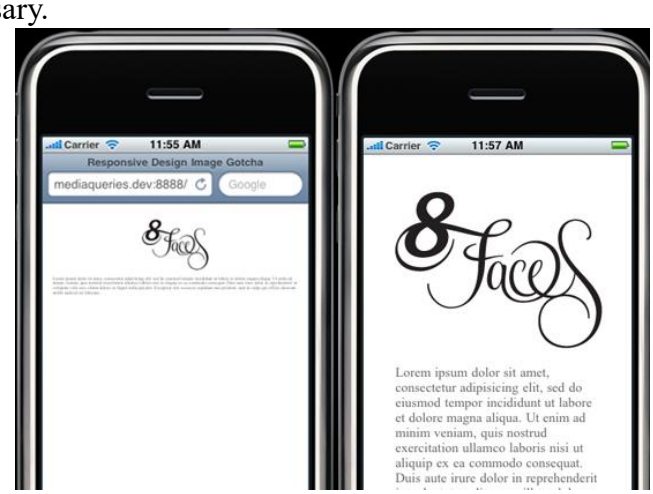
The JavaScript file inserts a base element that allows the page to separate responsive images from others and redirects them as necessary. When the page loads, all files are rewritten to their original forms, and only the large or small images are loaded as necessary. With other techniques, all higher-resolution images would have had to be downloaded, even if the larger versions would never be used. Particularly for websites with a lot of images, this technique can be a great saver of bandwidth and loading time.

This technique is fully supported in modern browsers, such as **IE8+, Safari, Chrome** and Opera, as well as mobile devices that use these same browsers (iPad, iPhone, etc.). Older browsers and Firefox degrade nicely and still resize as one would expect of a responsive image, except that both resolutions are downloaded together, so the end benefit of saving space with this technique is void.

STOP IPHONE SIMULATOR IMAGE RESIZING

One nice thing about the iPhone and iPod Touch is that Web designs automatically rescale to fit the tiny screen. A full-sized design, unless specified otherwise, would just shrink proportionally for the tiny browser, with no need for scrolling or a mobile version. Then, the user could easily zoom in and out as necessary.

There was, however, one issue this simulator created. When responsive Web design took off, many noticed that images were still changing proportionally with the page even if they were specifically made for (or could otherwise fit) the tiny screen. This in turn scaled down text and other elements.



Because this works only with Apple's simulator, we can use an Apple-specific meta tag to fix the problem, placing it below the website's <**head**> section. Thanks to Think Vitamin's article on image resizing, we have the meta tag below:

```
<meta name="viewport" content="width=device-width; initial-scale=1.0">
```

Custom Layout Structure

We may want to change the layout for extreme size changes, either through a separate style sheet or, more efficiently, through a CSS media query. This does not have to be troublesome; most styles can remain the same, while specific style sheets can inherit these styles and move elements around with floats, widths, heights and so on.

For example, we could have one main style sheet (which would also be the default) that would define all the main structural elements, such as #wrapper, #content, #sidebar, #nav, colors, backgrounds and typography. Default flexible widths and floats could also be defined.

If a style sheet made the layout too narrow, short, wide or tall, we could detect that and switch to a new one. This new child style sheet would adopt everything from the default style sheet and then redefine the layout's structure.

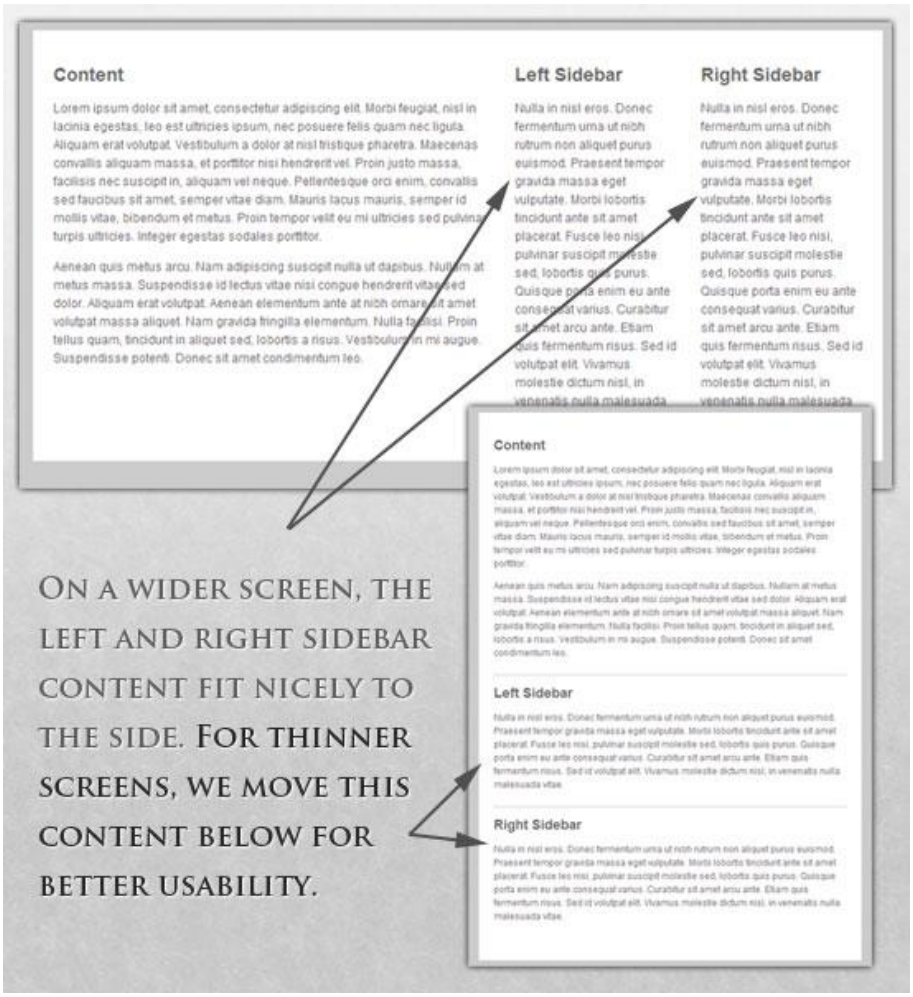
Here is the *style.css* (default) content:

```
/* Default styles that will carry to the child style sheet */
html,body{
  background...
  font...
  color...
}
h1,h2,h3{}
p, blockquote, pre, code, ol, ul{}
/* Structural elements */
#wrapper{
  width: 80%;
  margin: 0 auto;
  background: #fff;
  padding: 20px;
}
#content{
  width: 54%;
  float: left;
  margin-right: 3%;
}
```

```
#sidebar-left{
  width: 20%;
  float: left;
  margin-right: 3%;
}
#sidebar-right{
  width: 20%;
  float: left;
}
```

Here is the *mobile.css* (child) content:

```
#wrapper{
  width: 90%;
}
#content{
  width: 100%;
}
#sidebar-left{
  width: 100%;
  clear: both;
  /* Additional styling for our new layout */
  border-top: 1px solid #ccc;
  margin-top: 20px;
}
#sidebar-right{
  width: 100%;
  clear: both;
  /* Additional styling for our new layout */
  border-top: 1px solid #ccc;
  margin-top: 20px;
}
```



Interpreting Media Queries with CSS.

Media queries in CSS are a powerful tool that allows you to apply different styles and layouts based on the characteristics of the user's device or viewport. They enable you to create responsive designs that adapt to various screen sizes, resolutions, orientations, and even specific features of the device.

To interpret media queries in CSS, you typically use the @media rule followed by a media type or feature expression. Here's the basic syntax:

```
@media mediaType and (mediaFeature) {
 /* CSS rules and styles */
}
```

Let's break down the different components:

1. **@media:** This is the keyword that signifies the start of a media query.

2. **mediaType:** It represents the general category of the device or media being targeted. Some commonly used media types include:

   - all: Applies to all devices and media types.
   - screen: Applies to devices with a screen, such as desktops, laptops, tablets, and smartphones.
   - print: Applies when printing the document.
   - speech: Applies to screen readers and other speech synthesis devices.

3. mediaFeature: It specifies the specific characteristics or conditions that must be met for the styles within the media query to apply. Here are a few examples of media features:

   - width: Specifies the width of the viewport.
   - height: Specifies the height of the viewport.
   - orientation: Determines the orientation of the device (landscape or portrait).
   - aspect-ratio: Represents the aspect ratio of the viewport.
   - resolution: Indicates the pixel density of the output device.

You can combine multiple media features within a media query using logical operators like and, not, and only to create complex conditions. Here's an example:

```
@media screen and (max-width: 768px) and (orientation: landscape) {
 /* CSS rules and styles for screens with a maximum width of 768px in landscape
orientation */
}
```

In the example above, the styles within the media query will only be applied if the device has a screen, the viewport width is 768 pixels or less, and the orientation is landscape.

Implementing Media Queries.

Implementing media queries involves writing CSS rules within the appropriate media query blocks to target specific device characteristics or viewport conditions.

CSS3 supports all of the same media types as CSS 2.1, such as screen, print and handheld, but has added dozens of new media features, including max-width, device-width, orientation and color. New devices made after the release of CSS3 (such as the iPad and Android devices) will definitely support media features. So, calling a media query using CSS3 features to target these devices would work just fine, and it will be ignored if accessed by an older computer browser that does not support CSS3.

n Ethan Marcotte's article, we see an example of a media query in action:

```
<link rel="stylesheet" type="text/css"
 media="screen and (max-device-width: 480px)"
 href="shetland.css" />
```

This media query is fairly self-explanatory: if the browser displays this page on a screen (rather than print, etc.), and if the width of the screen (not necessarily the viewport) is 480 pixels or less, then load shetland.css.

New CSS3 features also include orientation (portrait vs. landscape), device-width, min-device-width and more. Look at "The Orientation Media Query" for more information on setting and restricting widths based on these media query features.

One can create multiple style sheets, as well as basic layout alterations defined to fit ranges of widths — even for landscape vs. portrait orientations. Be sure to look at the section of Ethan Marcotte's article entitled "Meet the media query" for more examples and a more thorough explanation.

Multiple media queries can also be dropped right into a single style sheet, which is the most efficient option when used:

```
/* Smartphones (portrait and landscape) ----------- */
@media only screen
and (min-device-width : 320px)
and (max-device-width : 480px) {
/* Styles */
}
/* Smartphones (landscape) ----------- */
```

```
@media only screen
and (min-width : 321px) {
/* Styles */
}
/* Smartphones (portrait) ----------- */
@media only screen
and (max-width : 320px) {
/* Styles */
}
```

The code above is from a free template for multiple media queries between popular devices by Andy Clark. See the differences between this approach and including different style sheet files in the mark-up as described in his book "Hardboiled Web Design."

CSS3 Media Queries

Above are a few examples of how media queries from CSS 2.1 and CSS3 could work. Let's now look at some specific how-to's for using CSS3 media queries to create responsive Web designs.

The min-width and max-width properties do exactly what they suggest. The min-width property sets a minimum browser or screen width to which a certain set of styles (or separate style sheets) would apply. If anything is below this limit, the style sheet link or styles will be ignored. The max-width property does just the opposite. Anything above the maximum browser or screen width specified would not apply to the respective media query.

The examples below show that we're using the syntax for media queries that could be used all in one style sheet. As mentioned above, the most efficient way to use media queries is to place them all in one CSS style sheet, with the rest of the styles for the website. This way, multiple requests don't have to be made for multiple style sheets.

```
@media screen and (min-width: 600px) {
    .hereIsMyClass {
        width: 30%;
        float: right;
    }
}
```

The class specified in the media query above (hereIsMyClass) will work only if the browser or screen width is above 600 pixels. In other words, this media query will run only if the minimum width is 600 pixels (therefore, 600 pixels or wider).

```
@media screen and (max-width: 600px) {
    .aClassforSmallScreens {
        clear: both;
     font-size: 1.3em;
    }
}
```

Now, with the use of max-width, this media query will apply only to browser or screen widths with a maximum width of 600 pixels or narrower.

While the above min-width and max-width can apply to either screen size or browser width, sometimes we'd like a media query that is relevant to device width specifically. This means that even if a browser or other viewing area is minimized to something smaller, the media query would still apply to the size of the actual device. The min-device-width and max-device-width media query properties are great for targeting certain devices with set dimensions, without applying the same styles to other screen sizes in a browser that mimics the device's size.

```
@media screen and (max-device-width: 480px) {
    .classForiPhoneDisplay {
        font-size: 1.2em;
    }
}
```

```
@media screen and (min-device-width: 768px) {
    .minimumiPadWidth {
        clear: both;
     margin-bottom: 2px solid #ccc;
    }
}
```

For the iPad specifically, there is also a media query property called orientation. The value can be either landscape (horizontal orientation) or portrait (vertical orientation).

```
@media screen and (orientation: landscape) {
    .iPadLandscape {
        width: 30%;
     float: right;
    }
```

```
    }

@media screen and (orientation: portrait) {
    .iPadPortrait {
        clear: both;
    }
}
```

Unfortunately, this property works only on the iPad. When determining the orientation for the iPhone and other devices, the use of max-device-width and min-device-width should do the trick.

There are also many media queries that make sense when combined. For example, the min-width and max-width media queries are combined all the time to set a style specific to a certain range.

```
@media screen and (min-width: 800px) and (max-width: 1200px) {
    .classForaMediumScreen {
        background: #cc0000;
        width: 30%;
        float: right;
    }
}
```

The above code in this media query applies only to screen and browser widths between 800 and 1200 pixels. A good use of this technique is to show certain content or entire sidebars in a layout depending on how much horizontal space is available.

Some designers would also prefer to link to a separate style sheet for certain media queries, which is perfectly fine if the organizational benefits outweigh the efficiency lost. For devices that do not switch orientation or for screens whose browser width cannot be changed manually, using a separate style sheet should be fine.

You might want, for example, to place media queries all in one style sheet (as above) for devices like the iPad. Because such a device can switch from portrait to landscape in an instant, if these two media queries were placed in separate style sheets, the website would have to call each style sheet file every time the user switched orientations. Placing a media query for both the horizontal and vertical orientations of the iPad in the same style sheet file would be far more efficient.

Another example is a flexible design meant for a standard computer screen with a resizable browser. If the browser can be manually resized, placing all variable media queries in one style sheet would be best.

Nevertheless, an organization can be key, and a designer may wish to define media queries in a standard HTML link tag:

```
<link rel="stylesheet" media="screen and (max-width: 600px)"
href="small.css" />
<link rel="stylesheet" media="screen and (min-width: 600px)"
href="large.css" />
<link rel="stylesheet" media="print" href="print.css" />
```

Applying a Responsive Approach to a Webpage

To apply a responsive approach to a webpage, you can follow these general steps:

- Design with Mobile-First in Mind: Start by designing your webpage for smaller screens first, considering the constraints of mobile devices. This approach ensures that your design focuses on essential content and functionality.

- Use Fluid Grids and Flexible Layouts: Instead of fixed-width layouts, employ fluid grids that adapt to different screen sizes. Use relative units like percentages and ems rather than pixels for sizing elements. This allows them to adjust proportionally as the viewport changes.

- Responsive Typography: Set font sizes, line heights, and other typographic properties using relative units. This enables the text to resize appropriately based on the viewport size.

- Responsive Images and Media: Use CSS techniques like max-width: 100% on images to ensure they scale down proportionally on smaller screens. For media elements like videos or embedded content, utilize responsive techniques or libraries to adapt their size and behavior based on the viewport.

- Media Queries: Utilize CSS media queries to define specific styles and layouts for different viewport sizes or device characteristics. Adjust the design as needed to provide an optimal user experience across various devices.

- Breakpoints: Identify breakpoints or specific screen sizes where the design needs to adapt significantly. Determine these breakpoints based on common device widths or popular screen resolutions. Use media queries to apply specific styles at these breakpoints to ensure a smooth transition between different layouts.

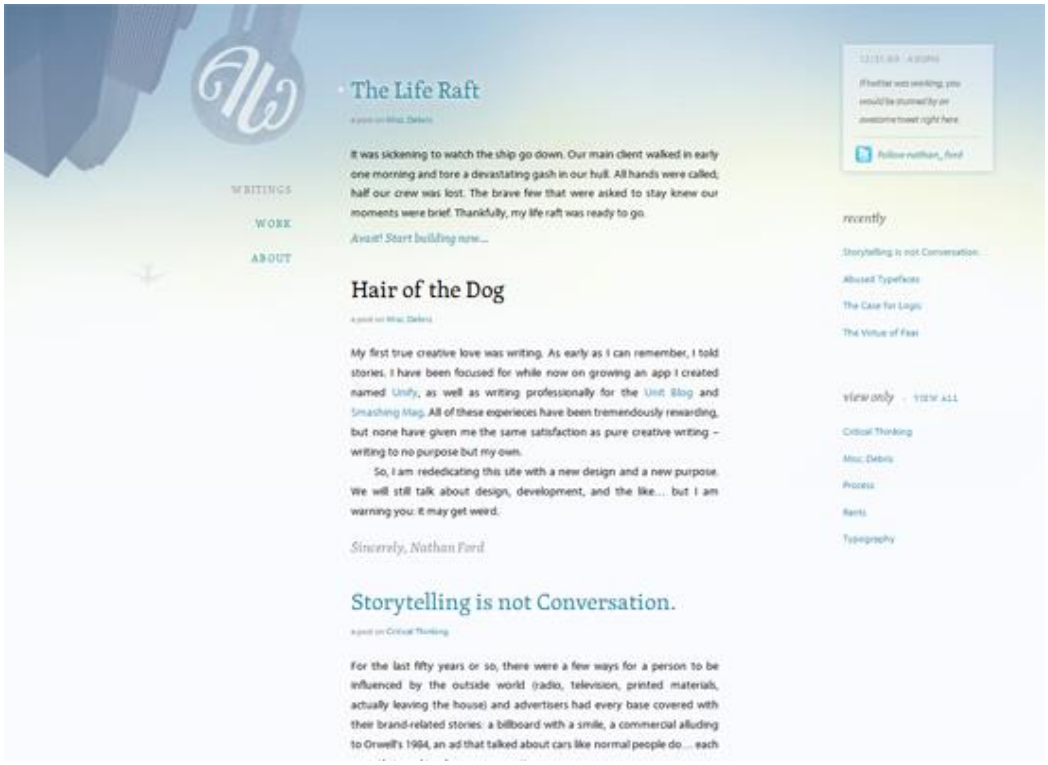
- Content Prioritization: Prioritize your content based on its importance and relevance. Make sure critical information is easily accessible on smaller screens. Consider rearranging or hiding less essential content to improve the user experience on mobile devices.

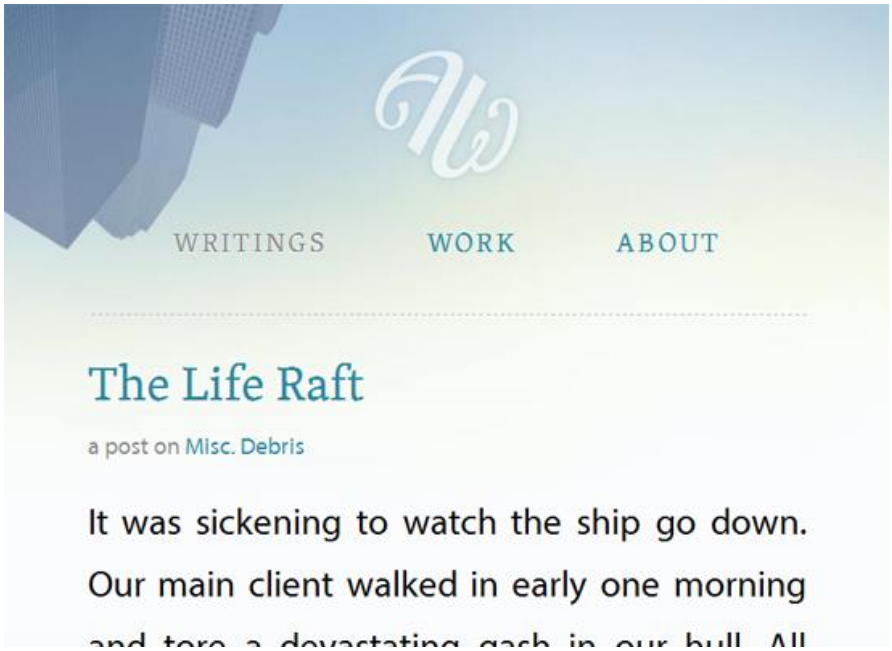
- Test and Iterate: Regularly test your webpage on different devices, screen sizes, and orientations. Make adjustments as necessary to ensure a consistent and optimized experience across various platforms. This iterative process helps fine-tune your responsive design.

Remember, implementing a responsive approach requires a combination of design principles, CSS techniques, and thorough testing. By focusing on flexibility, adaptability, and prioritizing user experience, you can create a webpage that provides an optimal viewing experience across a wide range of devices and screen sizes.

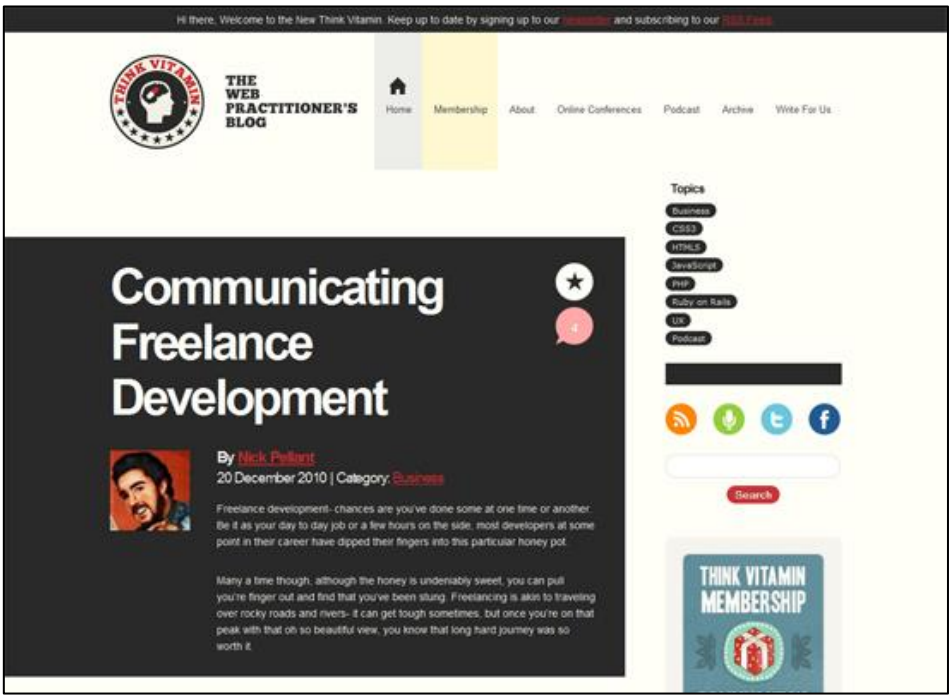
Below we have a few examples of responsive Web design in practice today. For many of these websites, there is more variation in structure and style than is shown in the pairs of screenshots provided. Many have several solutions for a variety of browsers, and some even adjust elements dynamically in size without the need for specific browser dimensions. Visit each of these, and adjust your browser size or change devices to see them in action.

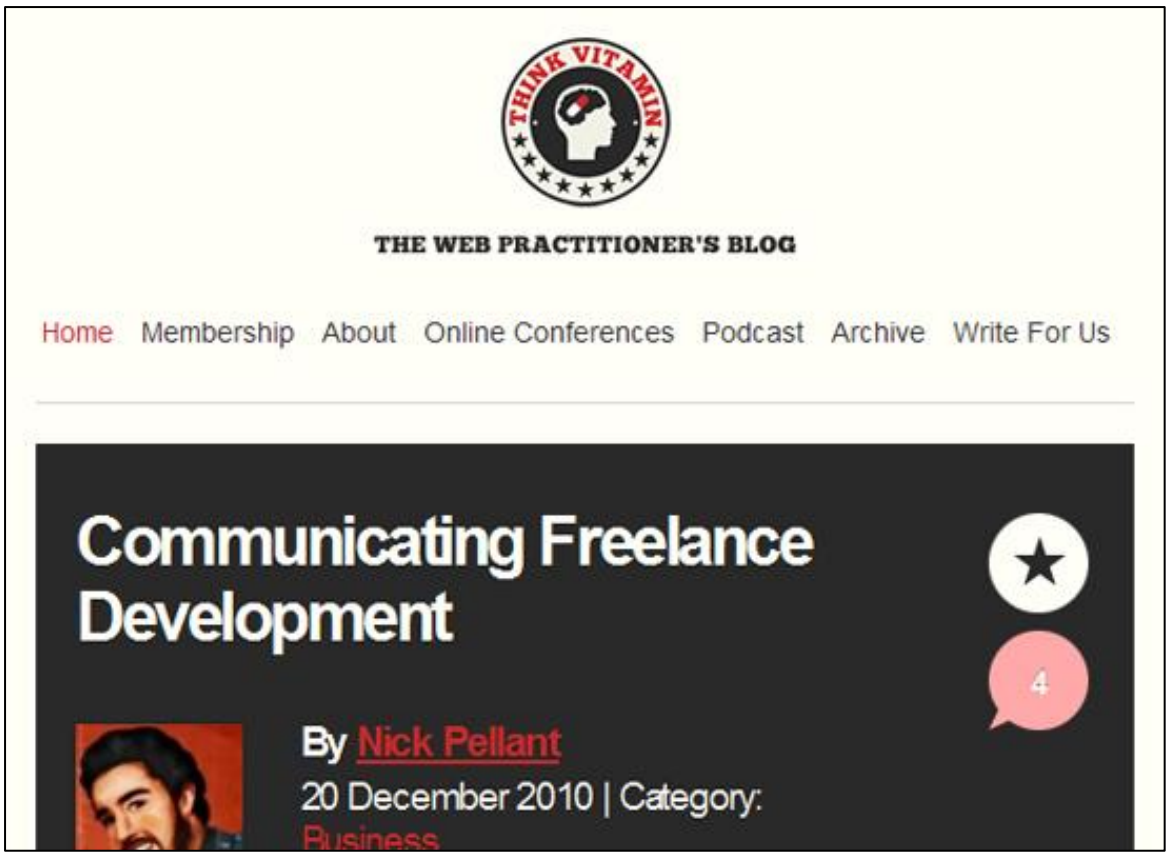
Art Equals Work is a simple yet great example of responsive Web design. The first screenshot below is the view from a standard computer screen dimension. The website is flexible with browser widths by traditional standars, but once the browser gets too narrow or is otherwise switched to a device with a smaller screen, then the layout switches to a more readable and user-friendly format. The sidebar disappears, navigation goes to the top, and text is enlarged for easy and simple vertical reading.





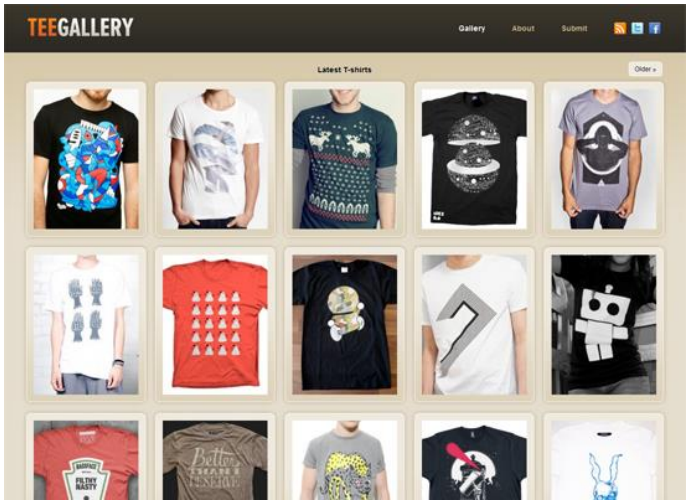
With Think Vitamin, we see a similar approach. When the sidebar and top bar are removed on a smaller screen or browser, the navigation simplifies and moves directly above the content, as does the logo. The logo keeps its general look yet is modified for a more vertical orientation, with the tagline below the main icon. The white space around the content on larger screens is also more spacious and interesting, whereas it is simplified for practical purposes on smaller screens.

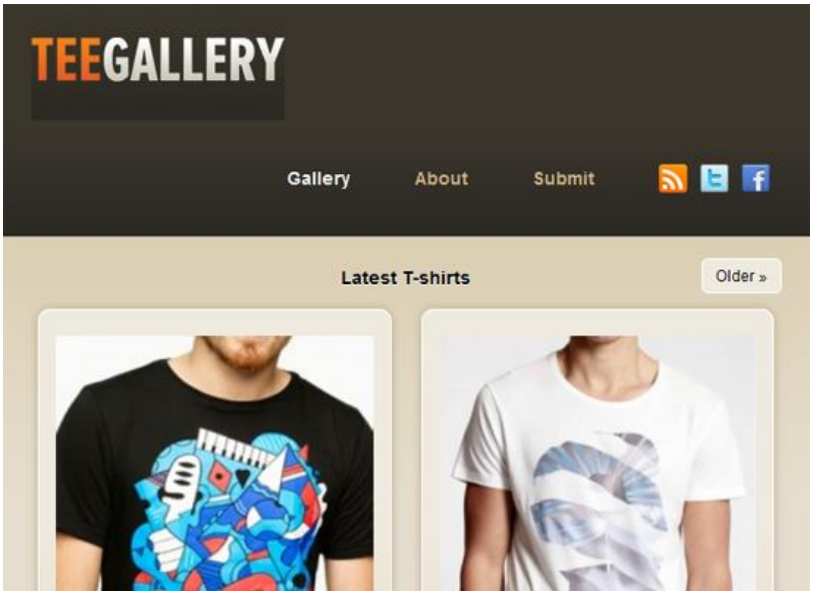




With four relatively content-heavy columns, it's easy to see how the content here could easily be squished when viewed on smaller devices. Because of the easy organized columns, though, we can also collapse them quite simply when needed, and we can stack them vertically when the space doesn't allow for a reasonable horizontal span. When the browser is minimized or the user is on a smaller device, the columns first collapse into two and then into one. Likewise, the horizontal lines for break points also change in width, without changing the size or style of each line's title text.

As one can see, the main navigation here is the simple layout of t-shirt designs, spanning both vertically and horizontally across the screen. The columns collapse and move below as the browser or screen gets smaller. This happens at each break point when the layout is stressed, but in between the breakpoints, the images change proportionally in size. This maintains balance in the design, while ensuring that any images (which are essential to the website) don't get so small that they become unusable.





Interpreting CSS Grid

CSS Grid is a powerful layout system in CSS that allows you to create complex grid-based layouts for web pages. When interpreting CSS Grid, there are several key concepts and properties to consider:

- **Container:** CSS Grid requires a container element as the parent for the grid items. The container is defined using the display: grid; property. It establishes the grid formatting context for its child elements.

- **Grid Tracks:** CSS Grid divides the container into horizontal and vertical tracks. Tracks are the columns and rows of the grid. You can define the size and number of tracks using grid-template-columns and grid-template-rows. You can specify sizes using fixed values (pixels, percentages) or flexible units (fr).

- **Grid Items:** Grid items are the child elements of the grid container. They are placed within the grid tracks and can span multiple tracks. To position grid items, you can use properties like grid-column-start, grid-column-end, grid-row-start, and grid-row-end. The shorthand property grid-column and grid-row can also be used for simplicity.

- **Grid Areas:** Grid areas are rectangular areas within the grid layout that can contain grid items. You can assign names to grid areas using the grid-template-areas property. By specifying the area names for grid items, you can easily position them within the grid.

- **Grid Lines:** Grid lines are the dividing lines between the tracks. They can be referenced using numerical indices or named lines. For example, you can refer to the first column using grid-column-start: 1; or to a named line using grid-column-start: start.

- **Grid Template:** The grid template is a shorthand way to define the number of columns and rows and their sizes and alignment. For example, grid-template: 1fr 2fr / 100px 1fr; creates two rows (1fr and 2fr) and two columns (100px and 1fr).

- **Grid Gap:** Grid gap refers to the space between horizontal and vertical grid tracks. You can control the size of the gap using properties like grid-gap, grid-column-gap, and grid-row-gap.

Defining CSS Grid Containers

To define a CSS Grid container, you need to follow these steps:

- Select the HTML element that will serve as the container for your grid layout. This could be a <**div**>, a <**section**>, or any other suitable element.

- Apply the **display**: **grid**; property to the selected container element. This sets the container as a grid formatting context and allows you to define grid-related properties.

- Optionally, specify the size and alignment of the grid tracks. You can use the **grid-template-columns** property to define the number, size, and alignment of the columns, and the grid-template-rows property to define the number, size, and alignment of the rows. Here's an example:

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr; /* Three columns with equal width */
  grid-template-rows: 100px auto; /* First row with fixed height, second row takes remaining space */
}
```

In the above example, the container element is selected with the class .**container**. It is set as a grid using **display: grid**;. The **grid-template-columns** property creates three columns with an equal width of 1fr each. The **grid-template-rows** property sets the height of the first row to 100px and allows the second row to take the remaining space (auto).

- If needed, define named grid areas using the grid-template-areas property. This allows you to create a visual map of your grid layout. You can assign names to different areas and then assign those names to grid items. Here's an example:

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-template-rows: auto;
  grid-template-areas:
    "header header"
    "sidebar main"
    "footer footer";
}
```

In this example, the grid areas are defined using the grid-template-areas property. The areas are named "header", "sidebar", "main", and "footer". You can then assign these area names to specific grid items.

- Place the grid items within the container by using properties like grid-column-start, grid-column-end, grid-row-start, and grid-row-end. You can also use shorthand properties like grid-column and grid-row. Here's an example:

```
.item {
  grid-column: 1 / 3; /* Item spans from column 1 to 3 */
  grid-row: 2; /* Item is placed in row 2 */
}
```

In this example, the .**item** class represents a grid item. The **grid-column** property is set to **1 / 3**, which means the item spans from column 1 to column 3. The **grid-row** property is set to 2, placing the item in row 2.

Identifying CSS Grid Items

Identifying CSS Grid items involves selecting and applying styles to specific elements within the grid layout. Here's how you can identify and work with CSS Grid items:

- Select Grid Items: Identify the elements that you want to target as grid items within the grid container. These elements will be direct children of the grid container element.
- Apply Grid Item Styles: To apply styles specifically to grid items, you can use the element selector or a class selector along with additional CSS properties. Here's an example:

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-template-rows: 100px auto;
}

.item {
  background-color: lightblue;
  padding: 10px;
}
```

In this example, the .container class represents the grid container, and the .item class is applied to the grid items. The .item class has specific styles like background-color and padding applied to it.

- Positioning Grid Items: To position grid items within the grid layout, you can use properties like grid-column-start, grid-column-end, grid-row-start, grid-row-end, or the shorthand properties grid-column and grid-row. These properties define where the grid item should start and end in terms of grid lines. For example:

```
.item {
  grid-column-start: 1;
  grid-column-end: 3;
  grid-row-start: 2;
  grid-row-end: 3;
}
```

In this case, the grid item with the .item class starts at column line 1, ends at column line 3, starts at row line 2, and ends at row line 3.

- Applying Grid Area Names: If you have defined named grid areas using the grid-template-areas property on the container, you can assign these area names to specific grid items. This provides an alternative way to position grid items. Here's an example:

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-template-rows: 100px auto;
  grid-template-areas:
    "header header"
    "sidebar main"
    "footer footer";
}
.item {
  grid-area: main;
}
```

In this example, the grid item with the .item class is assigned the main area using the grid-area property. This positions the item within the defined main area of the grid layout.

Applying CSS Grid

To apply CSS Grid to a web page layout, you need to follow these steps:

- Identify the Container: Select the HTML element that will serve as the container for your grid layout. This element will be the parent for all grid items. It could be a , a <section>, or any other suitable element.

- Apply Grid Display: Add the display: grid; property to the container element. This sets the container as a grid formatting context and enables the use of grid-related properties. For example:

```
.container {
  display: grid;
}
```

- Define Grid Template Columns and Rows: Specify the size and number of columns and rows in the grid layout using the grid-template-columns and grid-template-rows properties. You can use fixed values (pixels, percentages) or flexible units (fr) to define the sizes. For example:

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr; /* Three columns with equal width
*/
  grid-template-rows: auto 200px; /* First row with auto height, second
row with fixed height of 200px */
}
```

- Position Grid Items: Place the grid items within the container using properties like grid-column-start, grid-column-end, grid-row-start, and grid-row-end. You can also use shorthand properties like grid-column and grid-row for simplicity. For example:

```
.item {
  grid-column-start: 1;
  grid-column-end: 3;
  grid-row-start: 2;
  grid-row-end: 4;
}
```

In this example, the. item class represents a grid item. It starts at the first-column line, ends at the third-column line, starts at the second-row line, and ends at the fourth-row line.

- Customise Grid Gaps: Optionally, adjust the space between grid tracks (columns and rows) using grid-column-gap, grid-row-gap, or the shorthand property grid-gap. For example:

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-template-rows: auto;
  grid-column-gap: 20px; /* Adds a 20px gap between columns */
  grid-row-gap: 10px; /* Adds a 10px gap between rows */
}
```

- Add Grid Item Styling: Apply specific styles to the grid items using class selectors or element selectors. You can customize the appearance, positioning, and other properties of the grid items. For example:

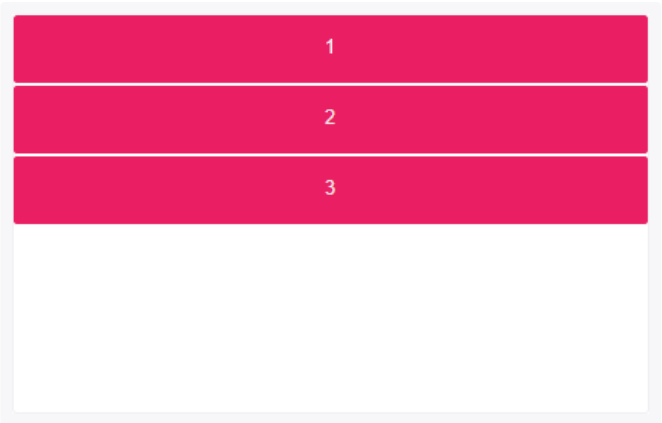
```
.item {
  background-color: lightblue;
  padding: 10px;
}
```

In this example, the **.item** class is applied to the grid items and has styles like background color and padding applied to it.

By following these steps, you can apply CSS Grid to create flexible and powerful grid layouts for your web page. Remember to adjust the grid template, position the grid items, and customize styles according to your design requirements.

Grid Container
Create a grid container by setting the display property with a value of grid or inline-grid. All direct children of grid containers become grid items.

display: grid
Grid items are placed in rows by default and span the full width of the grid container.



display: inline-grid

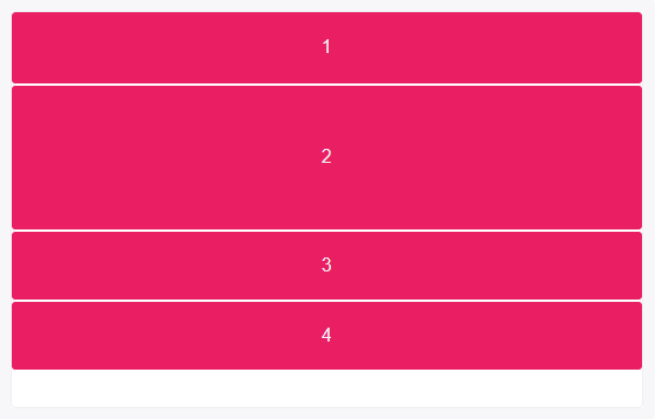
Explicit Grid

Explicitly set a grid by creating columns and rows with the grid-template-columns and grid-template-rows properties.
grid-template-rows: 50px 100px

A row track is created for each value specified for grid-template-rows. Track size values can be any non-negative, length value (px, %, em, etc.)

Items 1 and 2 have fixed heights of 50px and 100px.

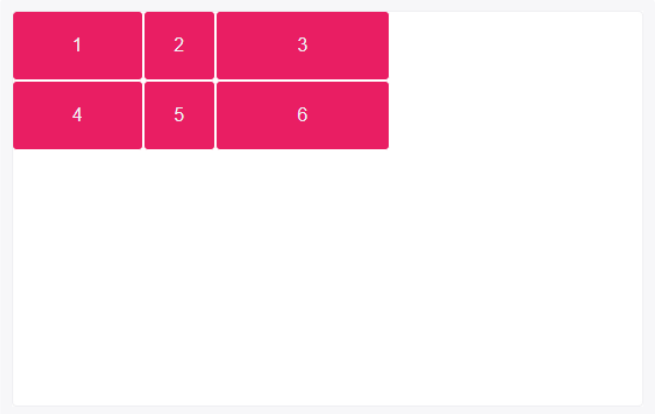
Because only 2 row tracks were defined, heights of items 3 and 4 are defined by the contents of each.



grid-template-columns: 90px 50px 120px
Like rows, a column track is created for each value specified for grid-template-columns.

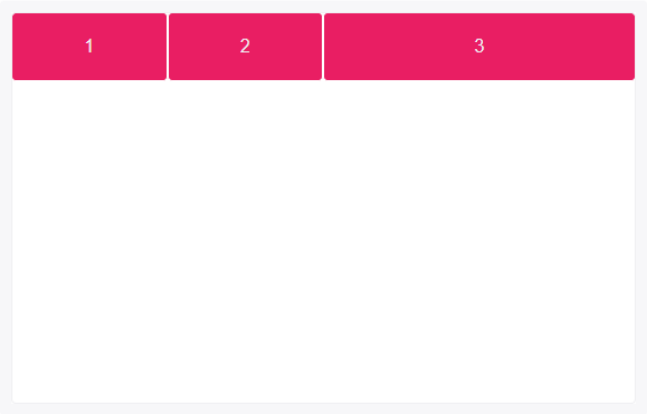
Items 4, 5 and 6 were placed on a new row track because only 3 column track sizes were defined; and because they were placed in column tracks 1, 2 and 3, their column sizes are equal to items 1, 2 and 3.

Grid items 1, 2 and 3 have fixed widths of 90px, 50px and 120px respectively.



grid-template-columns: 1fr 1fr 2fr
The fr unit helps create flexible grid tracks. It represents a fraction of the available space in the grid container (works like Flexbox's unitless values).

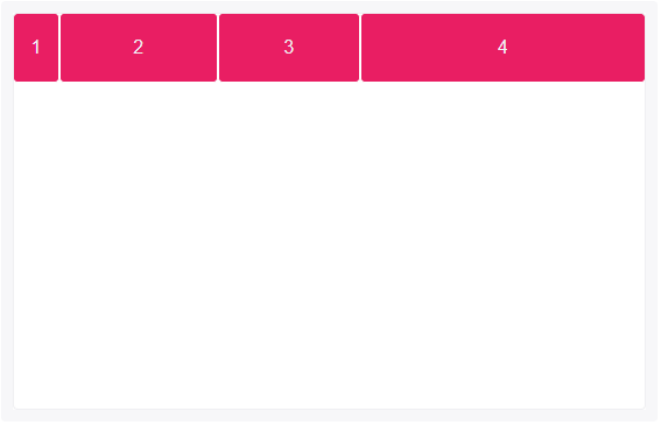
In this example, items 1 and 2 take up the first two (of four) sections while item 3 takes up the last two.



```
grid-template-columns: 3rem 25% 1fr 2fr
```

fr is calculated based on the remaining space when combined with other length values.

In this example, 3rem and 25% would be subtracted from the available space before the size of fr is calculated:
1fr = ((width of grid) - (3rem) - (25% of width of grid)) / 3



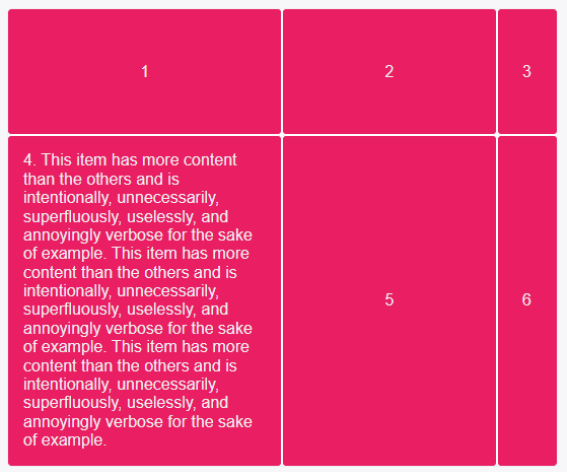
Minimum and Maximum Grid Track Sizes

Tracks sizes can be defined to have a minimum and/or maximum size with the minmax() function

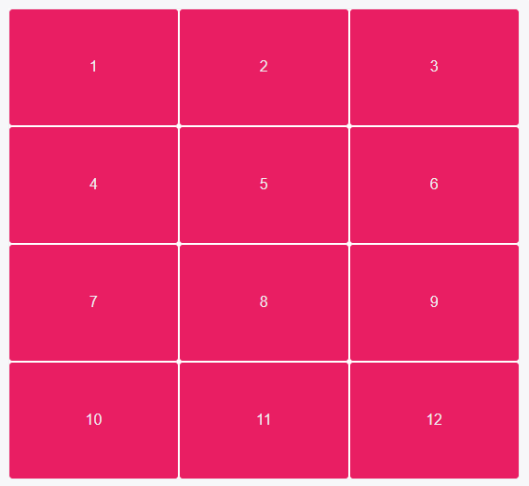
```
grid-template-rows: minmax(100px, auto);
grid-template-columns: minmax(auto, 50%) 1fr 3em;
```

The minmax() function accepts 2 arguments: the first is the minimum size of the track and the second the maximum size. Alongside length values, the values can also be auto, which allows the track to grow/stretch based on the size of the content.

In this example, the first row track is set to have a minimum height of 100px, but its maximum size of auto will allow the row track to grow it the content is taller than 100px.



The first column track has a minimum size of auto, but its maximum size of 50% will prevent it from getting no wider than 50% of the grid container width.



Repeating Grid Tracks

Define repeating grid tracks using the repeat() notation. This is useful for grids with items with equal sizes or many items.

```
grid-template-rows: repeat(4, 100px);
grid-template-columns: repeat(3, 1fr);
```

The repeat() notation accepts 2 arguments: the first represents the number of times the defined tracks should repeat, and the second is the track definition.

2.4 Saving and Executing a Website

Once you've designed and styled your website using CSS and CSS frameworks, the next important steps are saving and executing your project. This information sheet will guide you through the essential processes of saving your work and making your website live for users to access.

Why Saving and Execution Matter:

1. Preservation of Work: Saving your project ensures that all your design and code efforts are preserved. It prevents the loss of data due to unexpected issues.

2. Testing and Debugging: Execution is necessary to test your website and identify any errors or issues that may have arisen during the design process.

3. User Access: Execution makes your website accessible to users on the internet. It's the final step in the web development process.

Steps to Save Your Website:

1. Organize Files: Ensure all your HTML, CSS, JavaScript, and image files are organized in a designated project folder.

2. Save Code Files: Save your HTML file with the ".html" extension and CSS files with the ".css" extension. Use meaningful file names for easy identification.

3. Backup Regularly: Regularly create backups of your project folder to prevent data loss.

Executing Your Website:

1. Web Hosting: To make your website live, you need web hosting. Choose a reliable web hosting provider and set up your hosting account.

2. Domain Name: If you have a custom domain name, configure it to point to your web hosting server.

3. Upload Files: Use FTP (File Transfer Protocol) or a hosting control panel to upload your HTML, CSS, JavaScript, and image files to the hosting server.

4. Testing: Access your website using a web browser to test its functionality and appearance. Check for any issues and debug as needed.

5. Launch: Once you're satisfied with the testing results, officially launch your website for public access.

Self-Check Sheet 2 Design the website using cascading style sheets (CSS)

1 What does CSS stand for?

2 What is the purpose of CSS?

3 What are the three ways to include CSS in an HTML document?

4 What is the difference between class and ID selectors in CSS?

5 How can you select all <a> (anchor) elements within a specific element?

6 How do you select an element with the ID "my-element" using CSS?

7 What is the CSS box model?

8 How can you center a block-level element horizontally in CSS?

9 How can you apply multiple CSS classes to an element?

10 How do you specify a font-family in CSS?

11 What is the purpose of the display property in CSS?

12 How can you apply a background image to an element in CSS?

13 What is the difference between margin and padding in CSS?

14 What is the purpose of media queries in CSS?

15 How can you create a CSS animation?

16 What is the purpose of the float property in CSS?

17 How can you hide an element in CSS?

18 What is the difference between em and rem units in CSS?

19 What is a responsive layout?

Answer Key 2 Design the website using cascading style sheets (CSS)

1 CSS stands for Cascading Style Sheets.

2 The purpose of CSS is to describe the presentation and formatting of a document written in HTML or XML, including elements such as layout, colors, fonts, and animations.

3 Inline CSS, internal/embedded CSS, and external CSS.

4 Class selectors target elements based on a specific class attribute value, while ID selectors target elements based on a specific ID attribute value. Class selectors can be used on multiple elements, while ID selectors should be unique within the document.

5 By using the descendant selector, div a

6 By using the ID selector: #my-element

7 The CSS box model is a concept that describes how elements are displayed and spaced, including properties such as content, padding, border, and margin.

8 Setting its left and right margins to auto and giving it a specific width.

9 By adding multiple class names to the class attribute, separated by spaces.

10 By using the font-family property. For example: font-family: Arial, sans-serif;

11 The display property defines how an element is displayed on the page. It can be used to change the default behavior of elements, such as making them block-level or inline.

12 By using the background-image property and specifying the URL of the image. For example: background-image: url("image.jpg");

13 Margin is the space outside an element, creating a gap between it and other elements, while padding is the space within an element, creating a gap between the element's content and its border.

14 Media queries allow you to apply different styles based on different conditions, such as screen size, device orientation, or printing.

15 Using the @keyframes rule to define the animation steps and apply the animation to an element using the animation property.

16 The float property positions elements to the left or right, allowing text and other elements to wrap around them.

17 Using the **display: none**; property, or by setting the **visibility**: **hidden; property**.

18 **em** units are relative to the parent element's font size, while **rem** units are relative to the root (html) element's font size.

19 A responsive layout is a design approach that adapts to different screen sizes and devices to provide an optimal user experience.

Job Sheet 2 Implementing CSS for Web Layout and Design

Job Name: Implementing CSS for Web Layout and Design

Procedure:

Step 1: Review the project brief and design specifications to understand the layout and visual elements required for the website. Identify the key components that need styling, such as headers, navigation menus, content sections, forms, and footers.

Step 2: Set up a new or existing CSS file to write the style rules for the web layout.

Step 3: Integrate CSS Files

Step 4: Implement CSS for Layout

Step 5: Applying CSS Box Model and Positioning

Step 6: Define font families, sizes, weights, and styles for headings, paragraphs, and other text elements. Set color, line height, and letter spacing to improve readability and visual appeal.

Step 7: Apply CSS styles to images, buttons, links, and other visual elements to enhance appearance and interaction.

Specification Sheet 2 Implementing CSS for Web Layout and Design

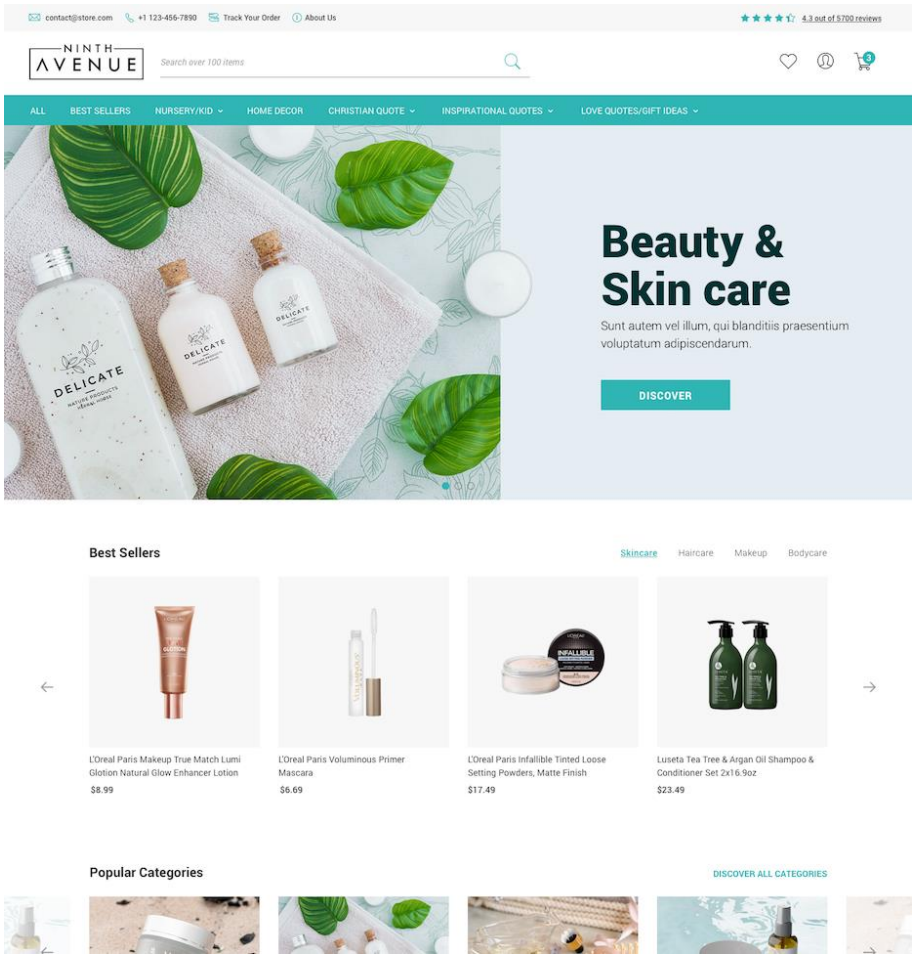
Necessary tools and equipment

Sl. No	Name of Tools & Equipment	Specification	Unit	Quantity
1	Computer	Minimum Corei3 with 4 GB RAM	No.	1
2	Code Editor	Any	No.	1
3	Web Browser	Any	No.	1

Other Specifications

1. Organize the CSS file by grouping related styles for ease of maintenance.
2. Link the CSS file(s) to the corresponding HTML file(s) by using the <link> tag in the document's <head> section.
3. Use CSS properties like display, float, flexbox, or grid to structure the content.
4. Use the CSS box model (margins, borders, padding, and content).

Create the CSS for the below HTML template. Or you can use any similar design.



Learning Outcome 3: Enhance website using CSS framework

Assessment Criteria	1. Framework is collected and configured with website. 2. HTML and CSS framework are integrated. 3. CSS framework is customized using CSS as per requirements. 4. Web site is saved and executed.
Conditions and Resources	1. Applicable tools, utensils, and equipment as prescribed by competency standards. 2. Supply materials 3. Relevant ingredients 4. CBLM related to the learning outcome. 5. Instructions, job sheets, activity sheets, and standard operating procedures 6. Personal protective equipment 7. Module/reference
Contents	1. Collecting and Configuring the Framework for Your Website 2. Integrating HTML and CSS Frameworks into Your Web Project 3. Customizing the CSS Framework to Meet Design Requirements 4. Saving and Executing Your Website: Making It Live for Users
Training Methods	1. CBLM 2. Handouts 3. Books, Manuals 4. Module/ Reference 5. Paper 6. Pen
Assessment Methods	1. Written Test 2. Demonstration 3. Oral Questioning

Learning Experience 3: Enhance website using CSS framework

You must perform the learning steps below to achieve the objectives stated in this learning guide. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

Learning Steps	Resources specific instructions
1. Students will ask the instructor about design styles with CSS and CSS framework.	1. The instructor will provide the learning materials for enhance website using CSS framework.
2. Read the **Information sheet/s**	2. Information Sheet No 1: Enhance website using CSS framework. ▪ Collecting and Configuring the Framework ▪ Integrating HTML and CSS Frameworks ▪ Customizing the CSS Framework to Meet Design Requirements ▪ Saving and Executing Website
3. Complete the **Self-Checks & Answer key sheets.**	3. Self-Check No: 3 Enhance website using CSS framework. Answer key No: 3 Enhance website using CSS framework.
4. Read the **Job/ Task sheet** and **Specification Sheet**	4. Job/ task sheet and specification sheet Job Sheet No: 3 Enhance website using CSS framework. Specification Sheet: 3 Enhance website using CSS framework.

Information Sheet 3 Enhance website using CSS framework

Learning Objective: After completing this information sheet, the learners will be able to Enhance website using CSS framework.

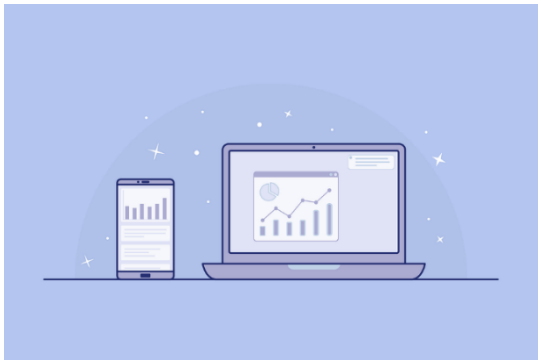
3.1 Collecting and Configuring the Framework

What Are Front-End Frameworks?

Front-end frameworks are essential for web developers as they help them create user interfaces (UIs) more efficiently and effectively. Essentially, front-end frameworks are sets of pre-written code and tools that developers use to build the UI of a website. These frameworks consist of HTML, CSS, and JavaScript components that developers can use to create responsive layouts, dynamic user interfaces, and interactive web applications.

Front-end frameworks provide a foundation for developers, allowing them to focus on building unique features and functionality to make their websites stand out. Moreover, these frameworks also help ensure that the website's UI is consistent across different browsers and devices, making it easier for users to navigate and use. Web front-end frameworks also come with various pre-built templates, modules, and plugins, which developers can use to speed up their development process. And, this is particularly useful when creating complex web applications that require a lot of different components.

Why Is a Front-End Framework Crucial?



A front-end framework is a crucial tool for developers because it speeds up the development process, provides consistency in the code, and is designed to be responsive.

Without a front-end framework, developers would have to start from scratch each time they create a new project. Furthermore, it can be time-consuming and tedious.

A front-end framework, on the other hand, provides a structure that allows developers to focus on the functionality of the application rather than the underlying code.

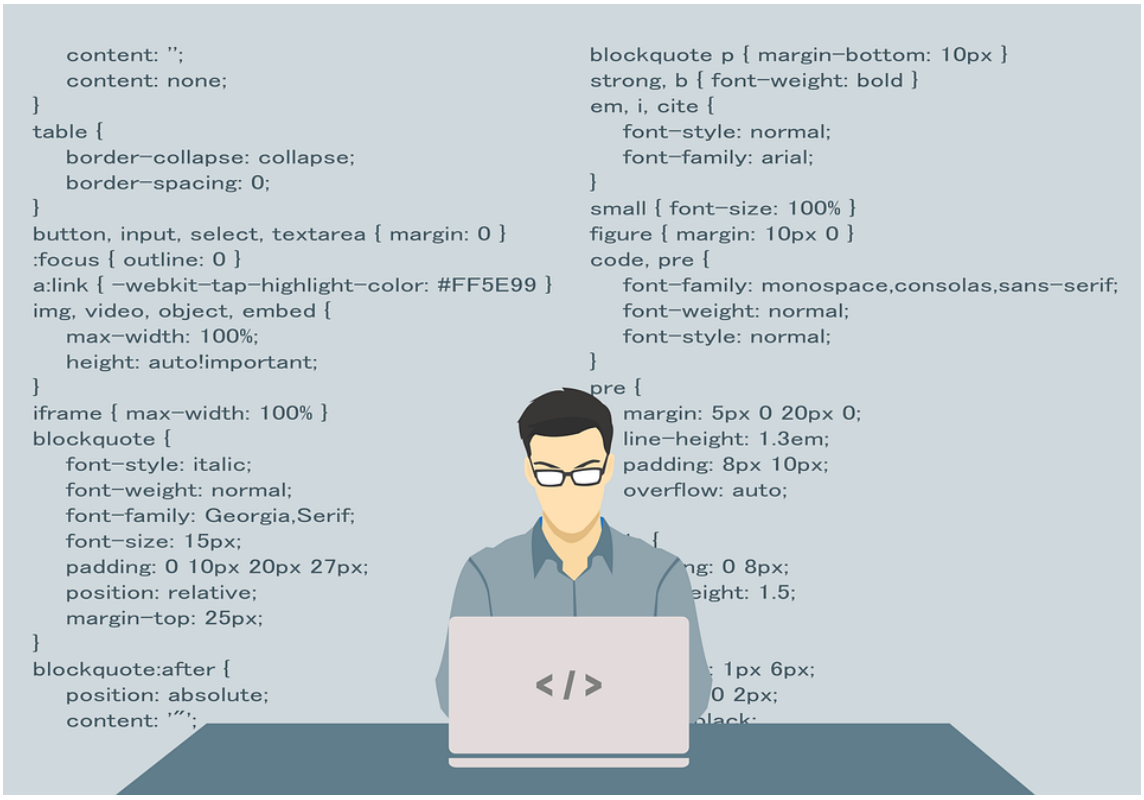
This consistency ensures the code is easy to maintain and any updates or changes can be made quickly and efficiently.

Additionally, front-end frameworks are designed to be responsive, meaning the application will work well on different devices and screen sizes. Hence, providing a better user experience.

Why Use Frontend Frameworks?

You must know the front-end frameworks determine every element's behavior, design, logic, structure, and animation. Here are some reasons showing the importance of front-end development frameworks.

- Accelerated Development Speed
  Front-end frameworks offer pre-built components and libraries, accelerating development speed by reducing manual coding efforts.
- Low Code
  With front-end frameworks, developers can build applications with minimal coding as the frameworks offer pre-designed UI components.
- Easy Maintenance
  Since front-end frameworks use standardized coding practices, it becomes easier to maintain and update the codebase, leading to reduced maintenance efforts.
- Two-Way Data Binding
  Two-way data binding in front-end frameworks allows for seamless communication between the model and view, reducing the need for manual updates.



What Are the Benefits of the Front-End Framework?

Here are some benefits that every developer must know before using the front-end development framework.

- Responsive Features and Applications

Front-end frameworks provide a wide range of responsive design elements and tools. Moreover, they allow developers to create websites and applications that automatically adjust to different screen sizes and devices.

- Cross Browser Support

Front-end frameworks offer cross-browser compatibility. Web applications built using these frameworks work seamlessly across different browsers. It also prevents glitches or errors.

- User Driven Outcomes

Front-end frameworks focus on user-centric design principles. Hence, offering a range of pre-built UI components and templates that improve user experience. Moreover, it also helps developers create user-friendly applications quickly and efficiently.

Selecting the Bootstrap Front-end Framework

When selecting a front-end framework, Bootstrap is a popular and versatile choice that provides a solid foundation for building responsive and mobile-first websites. Here are some factors to consider when selecting Bootstrap as your front-end framework:

- **Documentation and Community:** Bootstrap has excellent documentation, including guides, examples, and a comprehensive list of components and utilities. It also has a large and active community, which means you can find support, tutorials, and additional resources easily.

- **Customization**: Bootstrap offers extensive customization options. You can choose to include the entire framework or only select components, making it adaptable to your project's specific needs. You can also use Bootstrap's customization tool to modify the default styles and design.

- **Responsive Design:** One of the main advantages of Bootstrap is its built-in support for responsive web design. It provides a responsive grid system and responsive utility classes that help create layouts that adapt to different screen sizes and devices.

- **Component Library:** Bootstrap comes with a wide range of pre-built components, such as navigation bars, buttons, forms, modals, and carousels. These components can save development time and ensure consistency across your website.

- **Browser Compatibility:** Bootstrap is designed to work well across different browsers, ensuring a consistent user experience. It includes built-in polyfills for older browsers and focuses on compatibility with popular browsers, making it a reliable choice.

- **Third-Party Integration:** Bootstrap plays well with other tools and libraries, allowing you to easily integrate it with JavaScript frameworks like React or Angular. There are

also numerous third-party plugins and extensions available for extending Bootstrap's functionality.

- **Performance:** Bootstrap is optimized for performance, with a focus on lightweight CSS and JavaScript. However, keep in mind that including the entire framework can result in larger file sizes, so consider selectively including only the components you need to minimize the impact on performance.

- **Learning Curve:** While Bootstrap is relatively easy to get started with, it does have a learning curve, especially if you want to take full advantage of its customization options and advanced features. Familiarity with HTML, CSS, and JavaScript is recommended.

Ultimately, the choice of a front-end framework depends on your project requirements, familiarity with the technology stack, and personal preferences. Bootstrap is widely used and offers a robust set of features, making it a reliable choice for many web development projects.

Bootstrap

Bootstrap is a popular open-source front-end framework developed by Twitter. It provides a collection of pre-built HTML, CSS, and JavaScript components and a responsive grid system that helps developers quickly create modern and mobile-friendly websites and web applications.

Here are some reasons why you might choose to use Bootstrap:

- **Responsive Design**: One of the main benefits of using Bootstrap is its built-in support for responsive web design. It includes a responsive grid system that allows you to create flexible layouts that automatically adjust to different screen sizes and devices. This saves time and effort in building separate versions of your site for desktop and mobile devices.

- **Time Efficiency:** Bootstrap offers a comprehensive set of ready-to-use components, such as navigation bars, buttons, forms, modals, and more. These components are well-tested and provide consistent styling and behavior across different browsers. By leveraging these pre-built components, you can save development time and focus on customizing them to suit your specific needs.

- **Time Efficiency:** Bootstrap offers a comprehensive set of ready-to-use components, such as navigation bars, buttons, forms, modals, and more. These components are well-tested and provide consistent styling and behavior across different browsers. By leveraging these pre-built components, you can save development time and focus on customizing them to suit your specific needs.

- **Consistency and Cross-Browser Compatibility:** Bootstrap provides a consistent and unified look and feel across different browsers and devices. It takes care of handling browser compatibility issues and provides built-in CSS resets and JavaScript polyfills for older browsers. This ensures that your website looks and functions consistently for all users.

- **Customization Options:** While Bootstrap comes with a default set of styles, it also offers extensive customization options. You can use Bootstrap's customization tool to select specific components, modify the default colors, typography, and other design elements to match your project's branding. This flexibility allows you to create unique designs while still leveraging Bootstrap's underlying structure and functionality.

- **Community and Support:** Bootstrap has a large and active community of developers. This means you can find plenty of resources, tutorials, and documentation to help you get started and solve any issues you encounter. The community also contributes additional themes, templates, and plugins that extend Bootstrap's functionality and offer further customization options.

- **Mobile-First Approach:** Bootstrap follows a mobile-first approach, meaning that it prioritizes the design and development of websites for mobile devices. This approach ensures that your website is optimized for mobile users, who make up a significant portion of internet traffic. By using Bootstrap, you can easily create responsive and mobile-friendly designs without the need for extensive coding.

Understanding the Fundamentals of Bootstrap

What is Bootstrap?

Bootstrap is a popular open-source frontend framework that provides a collection of CSS styles, JavaScript components, and responsive grid layouts to simplify and accelerate web development. Here are the key fundamentals of Bootstrap:

- **Responsive Grid System:** Bootstrap's grid system is a fundamental feature that helps create responsive layouts. It is based on a 12-column grid, allowing developers to divide the web page content into rows and columns. The grid automatically adjusts the layout and column stacking based on the device screen size, ensuring a responsive design across various devices.

- **CSS Styles and Components:** Bootstrap includes a comprehensive set of CSS classes and styles that can be applied to HTML elements. These styles provide consistent typography, buttons, forms, alerts, navigation menus, and other UI

components. By utilizing the predefined classes, developers can easily style and customize the appearance of their web pages.

- **JavaScript Components and Plugins: Bootstrap** offers a wide range of JavaScript components and plugins that enhance the functionality and interactivity of websites. These components include modals, carousels, dropdowns, tooltips, tabs, accordions, and more. By incorporating these components, developers can add advanced features to their websites without having to write complex JavaScript code from scratch.

- **Customizable Themes: Bootstrap** allows developers to customize the look and feel of their websites using themes. It provides a default theme, but developers can easily modify variables and styles to create their own custom themes. This feature enables consistent branding and visual coherence across the website.

- **Cross-Browser Compatibility:** Bootstrap is designed to work seamlessly across different web browsers, ensuring consistent rendering and behavior. It takes care of browser compatibility issues by providing CSS resets and utilizing modern web standards. This allows developers to build websites that function correctly on major browsers without needing extensive testing and adjustments.

- **Documentation and Community:** Bootstrap offers thorough and well-organized documentation that guides developers through the framework's features and usage. The documentation includes examples, code snippets, and explanations, making it easier for developers to understand and implement Bootstrap in their projects. Additionally, Bootstrap has a large community of developers who actively contribute, provide support, and share resources, which further enhances the learning experience.

- **Mobile-First Approach:** Bootstrap follows a mobile-first approach to responsive design. This means that the framework prioritizes designing and optimizing for mobile devices first and then progressively enhances the experience for larger screens. The default CSS styles and components in Bootstrap are optimized for mobile devices, ensuring a smooth and user-friendly experience on smartphones and tablets.

By understanding and utilizing these fundamental aspects of Bootstrap, developers can leverage the framework's features, reduce development time, and create responsive and visually appealing websites with ease.

Why should We use Bootstrap?

Bootstrap offers a collection of CSS classes "out of the box" that allow developers to quickly create pages that scale to screens of different sizes. Developers can also decorate common UI components such as dialog boxes, buttons, forms, and tables with these classes.

Bootstrap supports the latest, stable releases of all major browsers and platforms.

Getting Started

We start with a skeleton HTML page. Nothing fancy here.

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-
scale=1">
  <title>Bootstrap Demo</title>
 </head>
  <body></body>
 </html>
```

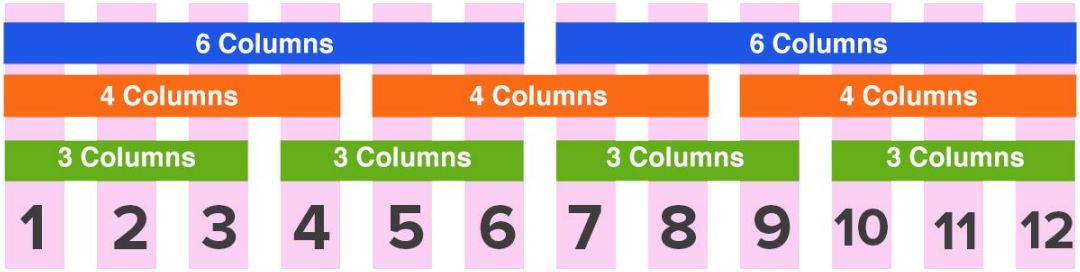
We then add the current stable version of Bootstrap CSS inside our <head> element.

```
<!DOCTYPE html>
<html lang="en">
 <head>
  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-
scale=1">
  <title>Bootstrap Demo</title>
  <!-- Bootstrap CSS -->
```

```
  <link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap
.min.css" rel="stylesheet">
 </head>
 <body></body>
</html>
```

The Grid System

Bootstrap layouts are based on a 12-column grid. We use CSS classes to specify the width of each element. Each HTML element may take up to 12 columns of space.



Grid classes are based on screen size:

- xs = Less than 768 pixels
- sm = 768-991 pixels
- md = 992-1199 pixels
- lg = Greater than or equal to 1200 pixels

Suppose we want our page layout to consist of a main content area that takes up most of the page width but also include room for a sidebar on the left.

We can use the provided CSS classes to accomplish this:

```
<div id="side-bar" class="col-md-2">
 <ul>
  <li>Sidebar item 1</li>
  <li>Sidebar item 2</li>
  <li>Sidebar item 3</li>
 </ul>
</div>
<div id="main-content" class="col-md-10">
 <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor
```

```
 incididunt ut labore et dolore magna aliqua. Ut enim ad minim
veniam, quis nostrud
 exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat.</p>
</div>
```

We specify 2 columns for the sidebar and 10 columns for the content area. We want these values to add up to 12 because the grid system is 12 columns.
Grid classes apply to screen size we specify and larger. So when we apply the col-md-2 class to the sidebar element, it will take up 2 columns on medium and large screens.

Containers and Rows

A container is a class that defines the page size based on the viewport width. This is typically applied to one div element which contains the entire contents of the page.
Rows may be placed within a .container (fixed-width) or .container-fluid (full-width) for proper alignment and padding.

Use rows (.row) to create horizontal groups of columns.

```
<body>
 <div class="container">
  <div class="row">Row 1</div>
  <div class="row">Row 2</div>
 </div>
</body>
```

Toggle Element Visibility

We can use the hidden-xx or visible-xx classes to toggle visibility of elements based on screen size:
Hide sidebar on small screens

```
<div id="side-bar" class="col-md-2 hidden-sm">
```

Show logo on large screens

```
<img alt="Logo" class="visible-lg" src="logo.png">
```

Glyphicons

These are icons bundled with Bootstrap that can be added to the page using a span element. A list of available glyphicons can be found here. We can insert these icons anywhere we would normally put in a span not just buttons (table headings, paragraphs, lists, etc).

We could make a download button more visually appealing with a glyphicon:

```
<button class="btn btn-lg btn-primary">
  <span class="glyphicon glyphicon-download-alt"></span> Download
</button>
```

Customization

A common criticism of Bootstrap is that all sites end up looking the same. However, it is easy to customize the styles shipped with Bootstrap so we can have a unique look that conforms to company branding.

Change Default Font

Let's say we didn't like the default font-family of Helvetica, Arial, sans-serif and instead wanted to use Roboto from Google Fonts instead. Here's how we could do that:

- Add CSS for Roboto font

  ```
  <link href="https://fonts.googleapis.com/css?family=Roboto" rel="stylesheet">
  ```

- Custom stylesheet added after Bootstrap CSS

  ```
  body { font-family: 'Roboto', sans-serif; }
  ```

Change Button Style

The same process can be used to modify colors, button shapes, and anything else defined in the Bootstrap CSS file.


```
.btn { border-radius: 20px; }
```


```
.btn { border-radius: 0px; }
```

Themes

We can download themes created by others to skin the look as well.

The Bootstrap project provides a theme that changes the appearance of buttons and other components for a less flat look.

Themes can be added by adding the appropriate CSS file after the Bootstrap CSS.

```
<link
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap
-theme.min.css" rel="stylesheet">
```

Components

Bootstrap has many other components which I'll be covering in a subsequent article.

3.2 Integrating HTML and CSS Frameworks Integrating Bootstrap into web projects involves a few steps to get started. Here's a general guide on how to integrate Bootstrap into your web projects:

- **Include Bootstrap Files:** Once downloaded, extract the Bootstrap files and locate the CSS and JavaScript files. Copy the necessary files and place them in your project directory. You typically need to include the Bootstrap CSS file (bootstrap.css or bootstrap.min.css) in the <**head**> section of your HTML file and the Bootstrap JavaScript file (bootstrap.js or bootstrap.min.js) before the closing </body> tag.

- **Add Required Dependencies:** Bootstrap requires jQuery, a popular JavaScript library, to work properly. Make sure to include jQuery in your project as well. You can either download jQuery from its official website or include it via a Content Delivery Network (CDN) by adding the following line in the <head> section of your HTML file:

  ```
  <script src="https://code.jquery.com/jquery-{version}.min.js"></script>
  ```

  Replace {version} with the desired jQuery version (e.g., 3.6.0).

- **Customize Bootstrap:** If desired, you can customize the Bootstrap framework to match your project's design and branding. Modify the variables and styles in the Bootstrap source files (if you downloaded the Sass version) or create custom CSS rules that override the default Bootstrap styles. This allows you to create a unique visual appearance for your website.

- **Test and Iterate:** After integrating Bootstrap, test your web project across different devices and screen sizes to ensure the responsive behavior is working as expected. Make any necessary adjustments and iterate on the design and functionality based on user feedback or requirements.

Websites that were built with a lot of CSS and JavaScript can now be built with a few lines of code using Bootstrap. Bootstrap comprises of mainly three components:

- CSS
- Fonts
- Javascript

The bootstrap can be used in 2 ways:

- Using Bootstrap CDN Link.
- By downloading the Bootstrap file.

We can easily get the resources for both approaches from the official website. Let’s begin the discussion with the first approach.

Method 1: Using CDN links – This method of installing Bootstrap is fairly easy but it requires a stable internet connection. It is highly recommended that you follow this method.

Step 1: Goto getbootstrap and click Getting Started. There you will find the below CDN links.

Step 2: Copy the links & paste it inside the <head> tag.

CSS link:

```
<link
href=”https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css”
rel=”stylesheet” integrity=”sha384-
F3w7mX95PdgyTmZZMECAngseQB83DfGTowi0iMjiWaeVhAn4FJkqJByhZMI3
AhiU” crossorigin=”anonymous”>
```

JavaScript link:

```
<script
src=”https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/js/bootstrap.bundle.min.js”
integrity=”sha384-
/bQdsTh/da6pkI1MST/rWKFNjaCP5gBSY4sEBT38Q/9RBh9AH40zEOg7Hlq2TH
RZ” crossorigin=”anonymous”></script>
```

Step 3: After completing the above steps, the code will be like the following:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8" />
    <meta name="viewport" content=
        "width=device-width, initial-scale=1" />
    <!-- Bootstrap CSS -->
    <link href=
"https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css"
        rel="stylesheet" integrity=
"sha384-
F3w7mX95PdgyTmZZMECAngseQB83DfGTowi0iMjiWaeVhAn4FJkqJB
yhZMI3AhiU"
        crossorigin="anonymous" />
</head>
<body>
    <h1>Hello, world!</h1>
    <div>
        You're learning Bootstrap
        on Geeksforgeeks.org
    </div>
    <!-- Optional JavaScript; choose one of the two! -->
    <!-- Option 1: Bootstrap Bundle with Popper -->
    <script src=
"https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/js/bootstrap.bundle.min.js
"
        integrity=
"sha384-
/bQdsTh/da6pkI1MST/rWKFNjaCP5gBSY4sEBT38Q/9RBh9AH40zEOg7
Hlq2THRZ"
        crossorigin="anonymous">
    </script>
    <!-- Option 2: Separate Popper and Bootstrap JS -->
```

```
        <!--
        <script src=
"https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.3/dist/umd/popper.min.js
"
                integrity=
"sha384-
W8fXfP3gkOKtndU4JGtKDvXbO53Wy8SZCQHczT5FMiiqmQfUpWbYd
Til/SxwZgAN"
                crossorigin="anonymous">
        </script>

        <script src=
"https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/js/bootstrap.min.js"
                integrity=
"sha384-
skAcpIdS7UcVUC05LJ9Dxay8AXcDYfBJqt1CJ85S/CFujBsIzCIv+l9liuY
LaMQ/"
                crossorigin="anonymous">
        </script> -->
</body>
</html>
```

At this stage, we have completed the installation process & we can now start to implement the logic.

Example: This example illustrates the use of the Bootstrap CDN link, in order to use the Bootstrap with the HTML document.

```
<!DOCTYPE html>
<html lang="en">
<head>
        <!-- Required meta tags -->
    <meta charset="utf-8" />
    <meta name="viewport"
        content="width=device-width,
                initial-scale=1" />

    <!-- Bootstrap CSS -->
    <link href=
"https://cdn.jsdelivr.net/npm/bootstrap@5.1.1/dist/css/bootstrap.min.css"
        rel="stylesheet"
        integrity=
"sha384-
F3w7mX95PdgyTmZZMECAngseQB83DfGTowi0iMjiWaeVhAn4FJkqJB
yhZMI3AhiU"
        crossorigin="anonymous"/>
    <title>Welcome to GeeksforGeeks</title>
</head>
```

```
<body>
        <h1>GeeksforGeeks</h1>
        <h3>Bootstrap Button</h3>
        <hr />
        <button type="button"
                class="btn btn-primary">Primary
        </button>
        <button type="button"
                class="btn btn-secondary">Secondary
        </button>
        <button type="button"
                class="btn btn-success">Success
        </button>
        <button type="button"
                class="btn btn-danger">Danger
        </button>
        <button type="button"
                class="btn btn-warning">Warning
        </button>
        <button type="button"
                class="btn btn-info">Info
        </button>
        <button type="button"
                class="btn btn-light">Light
        </button>
        <button type="button"
                class="btn btn-dark">Dark
        </button>
        <button type="button"
                class="btn btn-link">Link
        </button>
</body>
</html>
```

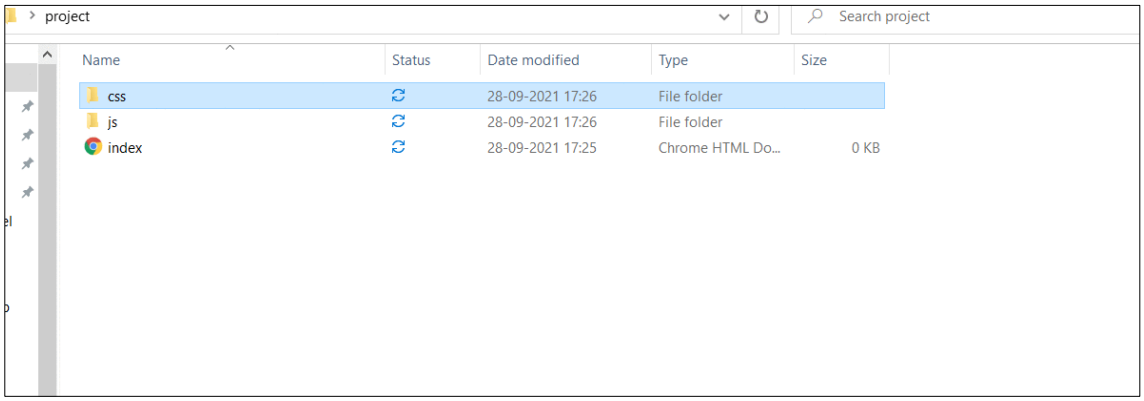
Output:

Method 2: By downloading Bootstrap – This method of installing bootstrap is also easy but it can work offline (doesn't require an internet connection) but it might not work for some browsers.

Step 1: Goto getbootstrap and click Getting Started. Click on the Download Bootstrap button and download the compiled CSS and JS.

Step 2: A .zip file would get downloaded. Extract it and go into the distribution folder. You would see 2 folders named CSS and JS. You can make your HTML file there and then you must paste these links in their respective sections. Under CSS files the most important files to be used are bootstrap and bootstrap.min. Under JS files, the most important are bootstrap.min.js and bootstrap.js.

Step 3: Make a separate project folder and create an HTML file. Under the folder, copy the extracted files downloaded from bootstrap. Under the head tag of the HTML file, the CSS needs to be linked. The jQuery downloaded should also be copied under the JS file. Make sure that under the project file, the downloaded files and HTML page must be present in that folder.



Step 4: After completing the above steps, the final code will look like the following code example. The final code after saving files under the same folder and adding links under the head and body tag respectively.

```
<!DOCTYPE html>
<html lang="en">

<head>
        <!-- Required meta tags -->
        <meta charset="utf-8" />
        <meta name="viewport"
                content="width=device-width,
                                initial-scale=1" />

        <link rel="stylesheet"
```

```
                type="text/css"
                href="css/bootstrap.css" />
</head>

<body>
        <h1>Welcome to gfg</h1>
        <script type="text/javascript"
                        href="js/jquery.js">
        </script>
        <script type="text/javascript"
                        href="js/bootstrap.min.js">
        </script>
</body>
</html>
```

Example: In the example, it can be observed that the downloaded files from bootstrap are included under the head and body section. Now the bootstrap classes can directly be used. As it is downloaded, thus no need for an internet connection required to load classes of bootstrap.

```
<!DOCTYPE html>
<html lang="en">
<head>
        <!-- Required meta tags -->
        <meta charset="utf-8" />
        <meta name="viewport" content=
                "width=device-width, initial-scale=1" />
        <link rel="stylesheet" type="text/css"
                href="css/bootstrap.css" />
</head>

<body>
        <h1>Welcome to GeeksforGeeks</h1>
        <div class="mb-3">
                <label for="exampleFormControlInput1"
                        class="form-label">
                        Email address
                </label>
                <input type="email" class="form-control"
                        id="exampleFormControlInput1"
                        placeholder="name@example.com" />
        </div>
        <div class="mb-3">
```

```
                <label for="exampleFormControlTextarea1"
                        class="form-label">
                        Example textarea
                </label>
                <textarea class="form-control"
                        id="exampleFormControlTextarea1" rows="3">
        </textarea>
        </div>
        <script type="text/javascript"
                href="js/jquery.js">
        </script>

        <script type="text/javascript"
                href="js/bootstrap.min.js">
        </script>
</body>
</html>
```

Output:

3.3 Customizing the CSS Framework to Meet Design Requirements

Utilizing Bootstrap Components

Bootstrap is a popular front-end framework that provides a collection of pre-designed components and CSS classes to help you build responsive and visually appealing web applications. Here's a guide on how to utilize Bootstrap components effectively:

Layout Components: Bootstrap provides layout components that help you structure your page, such as containers, grids, and responsive breakpoints. Use the container class to create a centered container for your content, and the row class to define rows within the container. Inside each row, you can use columns (e.g., col-md-6) to create a responsive grid system.

```
<div class="container">
 <div class="row">
  <div class="col-md-6">
   <!-- Content for the first column -->
  </div>
  <div class="col-md-6">
   <!-- Content for the second column -->
  </div>
 </div>
</div>
```

Typography

Typography plays a crucial role in web design as it affects the readability, user experience, and overall visual appeal of a website. Bootstrap provides a set of typography classes that you can utilize to style your text. Here are some commonly used Bootstrap typography classes:

- Headings: Bootstrap offers classes for headings (h1 to h6) that you can use to apply consistent styles across your headings. The classes are named from h1 to h6, where h1 represents the largest heading size and h6 represents the smallest.

```
<h1 class="display-1">Heading 1</h1>
<h2 class="display-2">Heading 2</h2>
<h3 class="display-3">Heading 3</h3>
<h4 class="display-4">Heading 4</h4>
<h5 class="display-5">Heading 5</h5>
<h6 class="display-6">Heading 6</h6>
```

- Text Colors: Bootstrap provides text color classes to change the color of your text. You can use classes like text-primary, text-secondary, text-success, text-danger, text-warning, text-info, text-light, and text-dark to apply different colors.

```
<p class="text-primary">This text is in a primary color.</p>
<p class="text-success">This text is in a success color.</p>
<p class="text-danger">This text is in a danger color.</p>
```

- Text Alignment: Bootstrap offers classes to align your text. You can use classes like text-left, text-center, and text-right to align your text to the left, center, or right respectively.

```
<p class="text-left">This text is aligned to the left.</p>
<p class="text-center">This text is centered.</p>
<p class="text-right">This text is aligned to the right.</p>
```

- Font Weights: Bootstrap provides classes to change the font weight of your text. You can use classes like fw-light, fw-normal, fw-bold, and fw-bolder to apply different font weights.

```
<p class="fw-light">This text has a light font weight.</p>
<p class="fw-normal">This text has a normal font weight.</p>
<p class="fw-bold">This text has a bold font weight.</p>
```

- Text Transformations: Bootstrap offers classes to transform your text. You can use classes like text-lowercase to convert the text to lowercase, text-uppercase to convert it to uppercase, and text-capitalize to capitalize the first letter of each word.

```
<p class="text-lowercase">this text is in lowercase.</p>
<p class="text-uppercase">this text is in uppercase.</p>
<p class="text-capitalize">this text is capitalized.</p>
```

These are some of the key Bootstrap typography classes you can use to style your text. Remember to combine these classes with other Bootstrap components and your custom CSS to create visually appealing and readable typography on your web pages.

Forms

Bootstrap provides a comprehensive set of CSS classes and components to style and enhance the functionality of forms. Here's how you can utilize Bootstrap for forms:

- Form Structure: Use the <form> element to wrap your form content. Bootstrap provides a grid system to create responsive form layouts. You can use the row class to define rows and the col-* classes to create columns for form elements.

```
<form>
 <div class="row">
  <div class="col-md-6">
   <!-- First column of the form -->
  </div>
  <div class="col-md-6">
   <!-- Second column of the form -->
  </div>
 </div>
</form>
```

- Form Controls: Bootstrap offers classes to style various form controls, such as inputs, selects, checkboxes, and radio buttons. Apply the form-control class to input elements to make them appear as block-level inputs with consistent styling.

```
<input type="text" class="form-control" placeholder="Name">
<select class="form-control">
 <option>Option 1</option>
 <option>Option 2</option>
</select>
```

- Form Layout: You can use Bootstrap's grid system to create responsive form layouts. Use the form-group class to group form controls together and apply consistent spacing. You can also use the form-label class for labels.

```
<div class="form-group">
 <label class="form-label" for="name">Name</label>
 <input type="text" class="form-control" id="name" placeholder="Enter your name">
</div>
```

- Checkboxes and Radio Buttons: Bootstrap provides styling for checkboxes and radio buttons. Use the form-check class for the container and the form-check-input class for the input element. Add a label with the form-check-label class to associate with the input.

```
<div class="form-check">
 <input class="form-check-input" type="checkbox" id="agree">
 <label class="form-check-label" for="agree">Agree to terms and conditions</label>
</div>
```

- Form Text: Bootstrap offers classes to style form help text and validation messages. Use the form-text class for descriptive text or hints associated with form controls. You can also use the text-danger class for displaying error messages.

```
<div class="form-group">
 <label class="form-label" for="email">Email</label>
 <input type="email" class="form-control" id="email" placeholder="Enter your email">
 <small class="form-text text-muted">We'll never share your email with anyone else.</small>
</div>
```

- Form Buttons: Bootstrap provides classes to style form buttons. Use the btn class for a basic button style, and additional classes like btn-primary, btn-secondary, etc., for different color variations.

```
<button type="submit" class="btn btn-primary">Submit</button>
<button type="button" class="btn btn-secondary">Cancel</button>
```

These are some of the key Bootstrap classes and components you can utilize for forms. Bootstrap offers more advanced features like input sizing, form validation, input groups, and more. Make sure to refer to the official Bootstrap documentation for a comprehensive list of form-related components and their usage: Bootstrap Forms Documentation

Buttons

Buttons are an essential element in web design for user interaction and call-to-action purposes. Bootstrap provides a variety of button styles and options to choose from. Here's how you can utilize Bootstrap for buttons:

- **Basic Buttons:** To create a basic button, use the btn class along with an optional contextual class such as ***btn-primary, btn-secondary, btn-success, btn-danger, btn-warning, btn-info***, or ***btn-light***. These contextual classes provide different colors for your buttons.

```
<button class="btn btn-primary">Primary Button</button>
<button class="btn btn-secondary">Secondary Button</button>
<button class="btn btn-success">Success Button</button>
<button class="btn btn-danger">Danger Button</button>
<button class="btn btn-warning">Warning Button</button>
<button class="btn btn-info">Info Button</button>
<button class="btn btn-light">Light Button</button>
```

- **Outline Buttons:** If you prefer a button with an outline instead of a filled background, you can add the btn-outline-* class along with the contextual class.

```
<button class="btn btn-outline-primary">Primary Button</button>
<button class="btn btn-outline-secondary">Secondary Button</button>
<button class="btn btn-outline-success">Success Button</button>
<button class="btn btn-outline-danger">Danger Button</button>
<button class="btn btn-outline-warning">Warning Button</button>
<button class="btn btn-outline-info">Info Button</button>
<button class="btn btn-outline-light">Light Button</button>
```

- **Button Sizes:** Bootstrap provides classes to adjust the size of your buttons. You can use btn-lg for a larger button, btn-sm for a smaller button, or btn-block to make a button span the full width of its container.

```
<button class="btn btn-primary btn-lg">Large Button</button>
<button class="btn btn-primary">Default Button</button>
<button class="btn btn-primary btn-sm">Small Button</button>
<button class="btn btn-primary btn-block">Block Button</button>
```

- **Disabled Buttons:** To disable a button, add the disabled attribute to the <button> element. You can also add the disabled class to give it a disabled appearance.

```
<button class="btn btn-primary" disabled>Disabled Button</button>
<button class="btn btn-primary disabled">Disabled Button</button>
```

- **Button with Icons:** You can include icons within your buttons using Bootstrap's icon library. Add the <i> element with the relevant icon class within the button.

```
<button class="btn btn-primary"><i class="bi bi-heart"></i> Like</button>
<button class="btn btn-danger"><i class="bi bi-trash"></i> Delete</button>
```

- **Button Groups:** Bootstrap allows you to group buttons together using the btn-group class. This is useful when you want to create a set of related buttons that behave as a single unit.

```
<div class="btn-group" role="group" aria-label="Button Group">
<button class="btn btn-primary">Button 1</button>
<button class="btn btn-primary">Button 2</button>
<button class="btn btn-primary">Button 3</button>
</div>
```

These are some of the common ways to utilize Bootstrap for buttons. Bootstrap also provides additional features such as dropdown buttons, split buttons, button loading states, and more. Explore the official Bootstrap documentation for a comprehensive understanding of Bootstrap buttons: Bootstrap Buttons Documentation

Table:

Tables are a powerful way to organize and present data in a structured format. Bootstrap offers a variety of CSS classes and components to style and enhance tables. Here's how you can utilize Bootstrap for tables:

- **Basic Table Structure:** Start by creating a <table> element and apply the table class to it. Use the <thead>, <tbody>, and <tfoot> elements to structure your table content. You can use the thead-dark or thead-light class to set the header's background color.

```
<table class="table">
  <thead class="thead-dark">
    <tr>
      <th>#</th>
      <th>Name</th>
      <th>Email</th>
```

```
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>1</td>
      <td>John Doe</td>
      <td>john@example.com</td>
    </tr>
    <tr>
      <td>2</td>
      <td>Jane Smith</td>
      <td>jane@example.com</td>
    </tr>
  </tbody>
</table>
```

- **Striped Rows:** To add zebra-striping to your table rows, apply the table-striped class to the <table> element.

```
<table class="table table-striped">
  <!-- Table content -->
</table>
```

Bordered Table: If you want to add borders to your table cells, use the table-bordered class.

```
<table class="table table-bordered">
  <!-- Table content -->
</table>
```

- **Hoverable Rows:** To highlight table rows when hovering over them, apply the table-hover class.

```
<table class="table table-hover">
  <!-- Table content -->
</table>
```

- Responsive Table: When working with tables on small screens, you can make them horizontally scrollable by wrapping the <table> element with a and applying the table-responsive class.

```
<div class="table-responsive">
  <table class="table">
    <!-- Table content -->
  </table>
</div>
```

- Contextual Classes: Bootstrap provides contextual classes to highlight specific rows or cells in a table. Use classes like table-primary, table-success, table-danger, table-warning, table-info, or table-light on table rows or cells to apply different background colors.

```
<table class="table">
  <tbody>
    <tr class="table-primary">
      <td>1</td>
      <td>John Doe</td>
      <td>john@example.com</td>
    </tr>
    <tr class="table-success">
      <td>2</td>
      <td>Jane Smith</td>
      <td>jane@example.com</td>
    </tr>
  </tbody>
</table>
```

These are some of the key Bootstrap classes and components you can utilize for tables. Bootstrap also offers additional features like table sizing, table head options, table captions, and more. Make sure to refer to the official Bootstrap documentation for a comprehensive list of table-related components and their usage: Bootstrap Tables Documentation

a) Navigation

Navigation is a critical component of web design that allows users to navigate through different sections and pages of a website. Bootstrap provides several navigation components and classes to create responsive and visually appealing navigation menus. Here's how you can utilize Bootstrap for navigation:

- **Navbar:** The Bootstrap Navbar component is used to create a responsive navigation bar at the top of the page. It automatically collapses into a mobile-friendly menu on smaller screens. Here's an example of a basic navbar:

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container">
    <a class="navbar-brand" href="#">Logo</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse"
data-target="#navbarNav" aria-controls="navbarNav" aria-
expanded="false" aria-label="Toggle navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
```

```
  <div class="collapse navbar-collapse" id="navbarNav">
    <ul class="navbar-nav ml-auto">
      <li class="nav-item active">
        <a class="nav-link" href="#">Home</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">About</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Services</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Contact</a>
      </li>
    </ul>
  </div>
 </div>
</nav>
```

- **Dropdown Menu**: Bootstrap allows you to create dropdown menus within the navbar or any other container element. Use the dropdown class along with the dropdown-toggle class to create a dropdown trigger. Add the dropdown-menu class to the dropdown content.

```
<li class="nav-item dropdown">
 <a class="nav-link dropdown-toggle" href="#" id="navbarDropdown" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">
  Dropdown
 </a>
 <div class="dropdown-menu" aria-labelledby="navbarDropdown">
  <a class="dropdown-item" href="#">Action</a>
  <a class="dropdown-item" href="#">Another Action</a>
  <div class="dropdown-divider"></div>
  <a class="dropdown-item" href="#">Something Else Here</a>
 </div>
</li>
```

- **Navs and Pills:** Bootstrap provides the nav and nav-pills classes to create horizontal navigation menus or pill-style navigation menus. You can use the nav-link class for the menu items.

```
<ul class="nav nav-tabs">
 <li class="nav-item">
```

```
  <a class="nav-link active" href="#">Home</a>
 </li>
 <li class="nav-item">
  <a class="nav-link" href="#">About</a>
 </li>
 <li class="nav-item">
  <a class="nav-link" href="#">Services</a>
 </li>
 <li class="nav-item">
  <a class="nav-link" href="#">Contact</a>
 </li>
</ul>
```

- **Breadcrumbs:** Breadcrumbs help users understand their current location within a website's hierarchy. Use the breadcrumb class along with the breadcrumb-item class for each breadcrumb item.

```
<nav aria-label="breadcrumb">
 <ol class="breadcrumb">
  <li class="breadcrumb-item"><a href="#">Home</a></li>
  <li class="breadcrumb-item"><a href="#">Category</a></li>
  <li class="breadcrumb-item active" aria-current="page">Product</li>
 </ol>
</nav>
```

- **Pagination**: For paginated content, Bootstrap offers the pagination class to create a pagination component. Use the page-item class for each page item and the page-link class for the link inside.

```
<nav aria-label="Page navigation">
 <ul class="pagination">
  <li class="page-item">
   <a class="page-link" href="#" aria-label="Previous">
    <span aria-hidden="true">&laquo;</span>
   </a>
  </li>
  <li class="page-item"><a class="page-link" href="#">1</a></li>
  <li class="page-item"><a class="page-link" href="#">2</a></li>
  <li class="page-item"><a class="page-link" href="#">3</a></li>
  <li class="page-item">
   <a class="page-link" href="#" aria-label="Next">
    <span aria-hidden="true">&raquo;</span>
   </a>
  </li>
 </ul>
```

```
</nav>
```

These are some of the key Bootstrap navigation components and classes you can utilize. Bootstrap also provides additional features like sticky navigation, justified navigation, vertical navigation, and more. Make sure to refer to the official Bootstrap documentation for a comprehensive list of navigation-related components and their usage: Bootstrap Navigation Documentation.

b) Modals

Modals are a popular UI component used to display additional content or interact with the user without navigating to a new page. Bootstrap provides a simple and customizable way to create modals. Here's how you can utilize Bootstrap for modals:

- **Modal Structure**: Start by creating a element with the modal class. Inside the modal, include a with the modal-dialog class, which represents the modal's main container. Within the dialog, add a with the modal-content class to hold the modal's content.

```
<div class="modal" id="myModal">
  <div class="modal-dialog">
    <div class="modal-content">
      <!-- Modal content goes here -->
    </div>
  </div>
</div>
```

- **Modal Trigger:** To open the modal, you need a trigger element such as a button or a link. Add the data-toggle attribute with a value of "modal" to the trigger element, and specify the target modal's ID using the data-target attribute.

```
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#myModal">
  Open Modal
</button>
```

Modal Content: Inside the modal-content div, you can include the modal's header, body, and footer. Use the modal-header, modal-body, and modal-footer classes for these sections.

```
<div class="modal" id="myModal">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title">Modal Title</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
```

```
      </button>
    </div>
    <div class="modal-body">
      <p>Modal content goes here...</p>
    </div>
    <div class="modal-footer">
      <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
      <button type="button" class="btn btn-primary">Save Changes</button>
    </div>
   </div>
  </div>
</div>
```

- **Modal Options:** Bootstrap provides various options to customize the behavior and appearance of modals. You can add the fade class to create a fade-in animation for the modal. Additionally, you can use the backdrop and keyboard attributes to control the behavior of the modal's backdrop and close the modal when pressing the Esc key.

```
<div class="modal fade" id="myModal" data-backdrop="static" data-keyboard="false">
  <!-- Modal content -->
</div>
```

- JavaScript Events: If you want to programmatically control the modal, Bootstrap provides JavaScript events that you can hook into. For example, you can use the show.bs.modal event to perform actions when the modal is about to be shown, or the hidden.bs.modal event to trigger actions after the modal is closed.

```
$('#myModal').on('show.bs.modal', function (event) {
  // Perform actions before the modal is shown
});
$('#myModal').on('hidden.bs.modal', function (event) {
  // Perform actions after the modal is closed
});
```

These are the basic steps to create a modal using Bootstrap. You can further customize the appearance, add form inputs, or incorporate dynamic content within the modal based on your specific requirements. For more advanced options and details, refer to the official Bootstrap documentation on modals: [Bootstrap Modals Documentation

a) Image carousels

Image carousels, also known as sliders or slideshows, are a popular way to display a series of images or content in a rotating manner. Bootstrap provides a Carousel component that makes it easy to create image carousels with built-in navigation controls. Here's how you can utilize Bootstrap for image carousels:

- **Carousel Structure**: Start by creating a element with the carousel class. Inside the carousel, add a with the carousel-inner class to hold the carousel items (images or content). Finally, include navigation controls such as previous and next buttons.

```
<div id="myCarousel" class="carousel slide" data-ride="carousel">
  <!-- Indicators -->
  <ol class="carousel-indicators">
    <li data-target="#myCarousel" data-slide-to="0" class="active"></li>
    <li data-target="#myCarousel" data-slide-to="1"></li>
    <li data-target="#myCarousel" data-slide-to="2"></li>
  </ol>
    <!-- Slides -->
  <div class="carousel-inner">
    <div class="carousel-item active">
      <img src="image1.jpg" alt="Image 1">
      <div class="carousel-caption">
        <h3>Image 1</h3>
        <p>Description of Image 1</p>
      </div>
    </div>
    <div class="carousel-item">
      <img src="image2.jpg" alt="Image 2">
      <div class="carousel-caption">
        <h3>Image 2</h3>
        <p>Description of Image 2</p>
      </div>
    </div>
    <div class="carousel-item">
      <img src="image3.jpg" alt="Image 3">
      <div class="carousel-caption">
        <h3>Image 3</h3>
        <p>Description of Image 3</p>
      </div>
    </div>
  </div>
  <!-- Navigation Controls -->
```

```
 <a class="carousel-control-prev" href="#myCarousel" role="button" data-slide="prev">
   <span class="carousel-control-prev-icon" aria-hidden="true"></span>
   <span class="sr-only">Previous</span>
 </a>
 <a class="carousel-control-next" href="#myCarousel" role="button" data-slide="next">
   <span class="carousel-control-next-icon" aria-hidden="true"></span>
   <span class="sr-only">Next</span>
 </a>
</div>
```

- **Carousel Indicators:** The <ol> element with the class carousel-indicators creates the carousel indicators (small dots) that represent each slide. Use the data-target attribute to specify the ID of the carousel, and the data-slide-to attribute to indicate the index of each slide. Make sure to add the active class to the first indicator to mark it as the active slide.

- **Carousel Slides:** The element with the class carousel-item represents each slide in the carousel. Add the active class to the first slide to make it the initial visible slide. Inside each slide, include an <img> tag with the src attribute pointing to the image file, and an optional with the carousel-caption class to display captions or descriptions for the images.

- **Navigation Controls:** The <a> elements with the classes carousel-control-prev and carousel-control-next create the previous and next navigation buttons, respectively. Use the data-slide attribute to specify the desired action ("prev" or "next"). Inside each <a> element, you can add a element with the respective carousel-control-prev-icon or carousel-control-next-icon class to display the navigation icons.

- **Optional:** Autoplay and Timing: By default, the Bootstrap carousel does not autoplay the slides. If you want the carousel to automatically transition between slides, you can add the data-ride="carousel" attribute to the main element. Additionally, you can control the timing of slide transitions using the data-interval attribute in milliseconds.

```
<div id="myCarousel" class="carousel slide" data-ride="carousel" data-interval="5000">
 <!-- Carousel structure -->
</div>
```

These are the basic steps to create an image carousel using Bootstrap. You can further customize the appearance, add more slides, change the transition effects, or incorporate additional features based on your specific requirements. For more advanced options

and details, refer to the official Bootstrap documentation on carousels: Bootstrap Carousel Documentation

3.4 Saving and Executing Website

Creating Mockups with Bootstrap

Bootstrap can be a great tool for creating mockups due to its extensive collection of pre-designed components and responsive grid system. Here are some steps to help you create mockups using Bootstrap:

- **Identify Mockup Requirements:** Start by understanding the requirements and goals of your mockup. Consider the layout, content, and functionality you want to demonstrate.
- **Plan the Structure:** Use the Bootstrap grid system to plan the layout of your mockup. Determine the number of columns and their sizes based on the content you want to include. You can use the container, row, and col classes to create the desired structure.
- **Select Components:** Identify the Bootstrap components you want to use in your mockup. Components such as navigation bars, headers, cards, forms, and buttons can be useful for demonstrating various sections of your design. Refer to the Bootstrap documentation to explore the available components and their usage.
- **Build the Mockup:** Begin building your mockup by adding the necessary HTML structure and applying Bootstrap classes to the elements. Utilize the predefined classes for typography, buttons, forms, tables, and other components to achieve the desired visual appearance.
- **Customize the Styling:** Customize the default Bootstrap styling to match your mockup's **Add Interactivity:** design. You can override the default CSS properties by adding custom CSS classes or modifying the existing classes. This allows you to create a unique look and feel for your mockup.
- If your mockup requires interactive elements such as dropdowns, modals, or carousels, incorporate the appropriate Bootstrap JavaScript plugins and add the necessary JavaScript code. You can use JavaScript/jQuery to handle events, perform actions, and showcase the functionality of your mockup.
- **Test Responsiveness:** Ensure that your mockup is responsive by testing it on different devices and screen sizes. Bootstrap's responsive grid system will automatically adjust the layout based on the device width, but it's important to verify that your mockup looks and functions well on various screen sizes.
- **Iterate and Refine:** Iterate on your mockup, gather feedback, and refine your design as necessary. Use the flexibility of Bootstrap to make adjustments and improvements to your mockup based on user feedback and requirements.

Remember, Bootstrap is a powerful framework that provides a solid foundation for creating mockups quickly. However, it's essential to customize and extend Bootstrap to suit your specific design needs and branding guidelines.

By following these steps, you can leverage Bootstrap's components, responsive grid system, and customization options to create visually appealing and interactive mockups.

Testing and Presenting Mockups

Testing and presenting your mockups is an important step in the design process to gather feedback, validate your design decisions, and ensure the effectiveness of your solution. Here are some suggestions for testing and presenting your Bootstrap mockups:

- **User Testing:** Conduct user testing sessions with a diverse group of users who represent your target audience. Provide them with specific tasks or scenarios to accomplish using your mockup. Observe their interactions, gather feedback, and note any usability issues or areas for improvement. User testing can be done in-person or remotely using screen sharing tools or prototyping platforms.
- **Clickable Prototypes:** Convert your mockups into clickable prototypes using tools like InVision, Figma, or Adobe XD. Link the different screens or components together to create a navigable prototype that simulates user interactions and flows. This allows stakeholders and users to experience the mockup in an interactive way and provides a more realistic representation of the final product.
- **Presentation Deck:** Prepare a presentation deck to showcase your mockups and design decisions. Use slides to introduce the project goals, user needs, and design considerations. Include screenshots or interactive prototype previews to visually demonstrate your mockups. Explain the functionality, user flow, and key features of your design. This presentation can be used for internal design reviews, stakeholder meetings, or client presentations.
- **Contextualize the Mockups:** Provide context for your mockups by explaining the problem you're addressing, the user pain points you're solving, and the benefits your design brings. Help stakeholders and clients understand how your mockups align with project objectives and user needs. Use real-world examples or scenarios to illustrate the value and impact of your design.
- **Gather Feedback:** Actively seek feedback from stakeholders, clients, and users. Encourage open and constructive discussions to understand their perspectives, concerns, and suggestions. Capture feedback in a structured manner, such as through feedback forms or collaborative tools like Google Docs or project management platforms. Consolidate and analyze the feedback to identify common themes and areas of improvement.

- **Iterate Based on Feedback:** Incorporate the feedback received into your mockups. Iterate on your design, making necessary adjustments and refinements. Address any usability issues, improve visual elements, and align the mockups with the feedback and insights gathered. Repeat the testing and presentation process if required to validate the effectiveness of your updated mockups.
- **Document Design Decisions:** Maintain a record of the design decisions you made during the mockup creation process. Document the rationale behind your design choices, including the usability principles, user research insights, and feedback considerations. This documentation serves as a reference and helps in explaining your design decisions to stakeholders and team members.
- **Collaborative Tools:** Use collaborative tools to share your mockups and gather feedback from distributed teams or clients. Platforms like InVision, Figma, or Miro allow for real-time collaboration, annotations, and discussions on specific design elements. This promotes effective communication and facilitates remote collaboration.
- **Usability Testing:** Usability testing involves observing users as they interact with your mockups to identify any usability issues or areas of improvement. It helps you understand how users navigate through your design, where they encounter difficulties, and how they interpret and interact with different components. Usability testing can be conducted through moderated sessions, where a facilitator guides users through tasks, or unmoderated sessions, where users perform tasks independently. Consider recording the sessions for later analysis and to capture valuable insights.
- **A/B Testing:** A/B testing (also known as split testing) involves creating two or more versions of your mockups and testing them with different groups of users to determine which version performs better. This testing method allows you to compare different design variations, such as layout, color scheme, or wording, and gather data-driven insights to inform your design decisions. A/B testing can be conducted using specialized tools or platforms that track user interactions and measure key metrics.
- **Remote Testing:** Remote testing allows you to gather feedback and insights from users who may not be physically present in the same location as you. You can conduct remote usability testing sessions using screen-sharing tools or remote user testing platforms. Remote testing offers flexibility and allows you to reach a wider audience, including users from different geographical locations.
- **Contextual Inquiry:** Contextual inquiry involves observing users in their natural environment as they interact with your mockups. By observing users in their real-world context, you can gain a deeper understanding of their needs, behaviors, and pain points. This method helps you gather rich qualitative data and uncover insights that may not emerge in a controlled testing environment. Contextual inquiry can

be conducted through on-site visits, remote video calls, or ethnographic research methods.

- **Design Critiques:** Conduct design critiques with your team or stakeholders to gather feedback and generate discussions around your mockups. Design critiques involve presenting your mockups to a group of people who provide constructive feedback, identify potential issues, and suggest improvements. This collaborative approach allows you to leverage diverse perspectives and tap into the collective expertise of the participants. Consider establishing clear guidelines for the critique session to ensure a focused and productive discussion.

- **Demonstrating Interactivity:** When presenting your mockups, emphasize the interactive elements and functionality to showcase how users would engage with your design. Use your clickable prototypes or live demos to walk through different user flows and demonstrate key interactions. This helps stakeholders and clients visualize the user experience and provides a more immersive understanding of your design concept.

- **User Surveys:** Conduct surveys to collect quantitative and qualitative feedback from a larger sample of users. Surveys allow you to gather insights on specific aspects of your mockups, such as visual appeal, ease of use, or overall satisfaction. Online survey tools like Google Forms or SurveyMonkey can be utilized to create and distribute surveys. Consider including both closed-ended questions (e.g., Likert scale rating) and open-ended questions to capture detailed feedback and suggestions.
- **Visual Design Presentations:** When presenting your mockups, pay attention to visual design aspects such as typography, color scheme, and overall aesthetics. Explain your design choices, including the rationale behind your color palette, typography selection, and visual hierarchy. Present your mockups in a visually appealing and engaging manner to effectively communicate your design vision.

Self-Check Sheet 3 Enhance website using CSS framework

Multiple Choice Questions (MCQs)

1. What are front-end frameworks primarily used for?
   a. Backend development
   b. Database management
   c. User interface (UI) development
   d. Server configuration

2. Why are front-end frameworks crucial for web developers?
   a. They simplify server-side scripting.
   b. They provide database management tools.
   c. They help create responsive and interactive user interfaces.
   d. They optimize server performance.

3. Which of the following is NOT a benefit of using front-end frameworks?
   a. Accelerated development speed
   b. Low code requirements
   c. Increased server security
   d. Easy maintenance

4. What does the Bootstrap front-end framework prioritize in its design approach?
   a. Desktop-first design
   b. Mobile-first design
   c. Tablet-first design
   d. Cross-browser compatibility

5. What is the purpose of Bootstrap's grid system?
   a. To create responsive layouts
   b. To manage server resources
   c. To generate database queries
   d. To optimize SEO performance

6. What is the purpose of conducting user testing when presenting mockups?
   a. To gather quantitative feedback
   b. To validate design decisions
   c. To create clickable prototypes
   d. To document design decisions

7. Which of the following tools can be used to convert Bootstrap mockups into clickable prototypes?
   a. Bootstrap documentation
   b. Google Docs
   c. InVision
   d. Adobe Photoshop

8. What is the primary benefit of A/B testing in the mockup testing process?
   a. Gathering qualitative insights
   b. Validating design principles
   c. Comparing design variations
   d. Conducting usability testing

9. In design critiques, what is the main purpose of presenting mockups to a group of people?
   a. To finalize the design
   b. To identify potential issues
   c. To create clickable prototypes
   d. To gather quantitative feedback

10. What is the purpose of emphasizing interactive elements when presenting mockups?
   a. To showcase visual design
   b. To engage stakeholders
   c. To gather user feedback
   d. To highlight functionality

Short Questions

1. Explain the importance of a mobile-first design approach in front-end development.

2. How does Bootstrap's grid system work, and why is it useful in web development?

3. Describe the role of typography in web design and how Bootstrap facilitates typography styling.

4. What are some common use cases for buttons in web design, and how does Bootstrap simplify button creation?

5. What is a modal, and how can it enhance user interactions on a website? Provide an example of when you might use a modal.

6. What is the key advantage of using Bootstrap for creating mockups?

7. How can you ensure that your Bootstrap mockup is responsive, and why is this important?

8. What is the purpose of documenting design decisions during the mockup creation process?

9. What is the difference between usability testing and A/B testing in the context of mockup testing?

10. Why is it important to contextualize mockups when presenting them to stakeholders and clients?

Answer Key 3 Enhance website using CSS framework

Multiple Choice Questions (MCQs)

1. c. User interface (UI) development
2. c. They help create responsive and interactive user interfaces.
3. c. Increased server security
4. b. Mobile-first design
5. a. To create responsive layouts
6. b. To validate design decisions
7. c. InVision
8. c. Comparing design variations
9. b. To identify potential issues
10. d. To highlight functionality

Short Questions

1. A mobile-first design approach prioritizes designing for mobile devices before considering larger screens. It's important because:
   - Mobile usage is widespread, so starting with mobile ensures a better user experience for a larger audience.
   - It encourages simplicity and essential content, which can improve overall design.
   - It aligns with Google's mobile-first indexing, which affects search engine rankings.

2. Bootstrap's grid system is based on a 12-column layout. It works by dividing the webpage into rows and columns, where content is placed. It's useful because:
   - It provides a responsive layout structure that automatically adjusts to different screen sizes.
   - It simplifies creating complex layouts without extensive custom CSS.
   - It's adaptable and saves development time, making it ideal for rapid prototyping.

3. Typography in web design involves selecting fonts, sizes, spacing, and styling for text content. Bootstrap facilitates this by offering predefined typography classes that can be applied to HTML elements. For example, you can use `class="h1"` to create a heading with the appropriate styling. This consistency ensures a polished and harmonious look for your text content.

4. Buttons are used in web design for various actions like submitting forms, triggering navigation, or interacting with content. Bootstrap simplifies button creation by providing predefined button styles, like primary, secondary, and success, that can be easily applied to HTML elements using classes like `btn` and `btn-primary`. This consistency ensures that buttons are visually appealing and align with the overall design.

5. A modal is a dialog box or popup window that appears on top of the main content to capture user input or display additional information. Modals enhance user interactions by focusing attention and providing a clear call to action. For example, you might use a modal for a "Sign Up" form on a website. When users click the "Sign Up" button, a modal appears, prompting them to fill in their details without navigating away from the current page. This enhances user engagement and simplifies the registration process.

6. The key advantage of using Bootstrap for creating mockups is its extensive collection of pre-designed components and a responsive grid system, which allows for rapid mockup development.

7. You can ensure that your Bootstrap mockup is responsive by testing it on different devices and screen sizes. This is important because it ensures that your mockup looks and functions well on various devices, providing a consistent user experience. Bootstrap's responsive grid system automatically adjusts the layout based on the device width.

8. Documenting design decisions helps provide a clear rationale for the choices made in the mockup, including usability principles, user research insights, and feedback considerations. It serves as a reference and aids in explaining design decisions to stakeholders and team members.

9. Usability testing involves observing users as they interact with mockups to identify usability issues and gather qualitative feedback. A/B testing, on the other hand, compares multiple variations of mockups with different design elements to determine which performs better based on quantitative data.

10. Contextualizing mockups by explaining the problem they address, user pain points they solve, and their alignment with project objectives helps stakeholders and clients understand the value and impact of the design. It provides a broader perspective and justifies design choices.

Job Sheet 3 Implementing CSS framework for Web Layout and Design

Job Name: Implementing CSS framework for Web Layout and Design

Procedure:

Step 1: Review the project brief and design specifications to understand the layout and visual elements required for the website. Identify the key components that need styling, such as headers, navigation menus, content sections, forms, and footers.

Step 2: Set up a new or existing CSS file to write the style rules for the web layout—import external CSS framework for the project.

Step 3: Integrate CSS Files

Step 4: Implement CSS for Layout

Step 5: Define font families, sizes, weights, and styles for headings, paragraphs, and other text elements. Set color, line height, and letter spacing to improve readability and visual appeal.

Step 6: Apply CSS styles to images, buttons, links, and other visual elements to enhance appearance and interaction.

Specification Sheet 3 Implementing CSS framework for Web Layout and Design

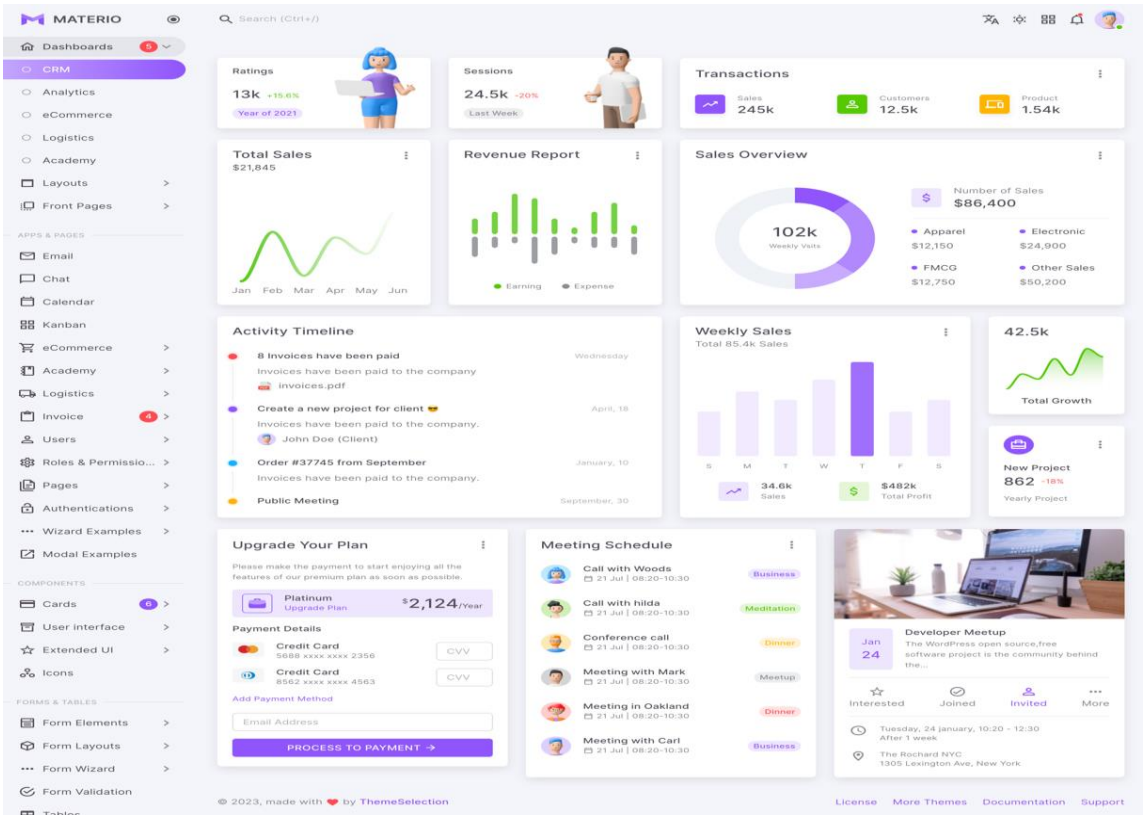
Necessary tools and equipment

Sl. No	Name of Tools & Equipment	Specification	Unit	Quantity
1.	Computer	Minimum Corei3 with 4 GB RAM	No.	1
2.	Code Editor	Any	No.	1
3.	Web Browser	Any	No.	1
4.	CSS Framework	Bootstrap	No.	1

Other Specifications

1. Organize the CSS file by grouping related styles for ease of maintenance.
2. Link the CSS file(s) to the corresponding HTML file(s) by using the <link> tag in the document's <head> section.
3. Use CSS properties like display, float, flexbox, or grid to structure the content.

Create the CSS for the below HTML template. Or you can use any similar design.



Learning Outcome 4: Test and confirm the website

Assessment Criteria	1. The website is tested to ensure functionality and errors are corrected per standard operating procedure. 2. The website is opened with common browsers and checked for accessibility, readability, legibility, and presentation in accordance with client requirements. 3. The website is evaluated for fitness in terms of the purpose, target audience and specifications of client requirements.
Conditions and Resources	1. Applicable tools, utensils, and equipment as prescribed by competency standards. 2. Supply materials 3. Relevant ingredients 4. CBLM related to the learning outcome. 5. Instructions, job sheets, activity sheets, and standard operating procedures 6. Personal protective equipment 7. Module/reference
Contents	1. Testing Website Functionality and Correcting Errors 2. Checking Website Compatibility and Accessibility 3. Evaluating Website Fitness for Purpose and Client Requirements
Training Methods	1. CBLM 2. Handouts 3. Books, Manuals 4. Module/ Reference 5. Paper 6. Pen
Assessment Methods	1. Written Test 2. Demonstration 3. Oral Questioning

Learning Experience 4: Test and confirm the website

You must perform the learning steps below to achieve the objectives stated in this learning guide. Beside each step are the resources or special instructions you will use to accomplish the corresponding activity.

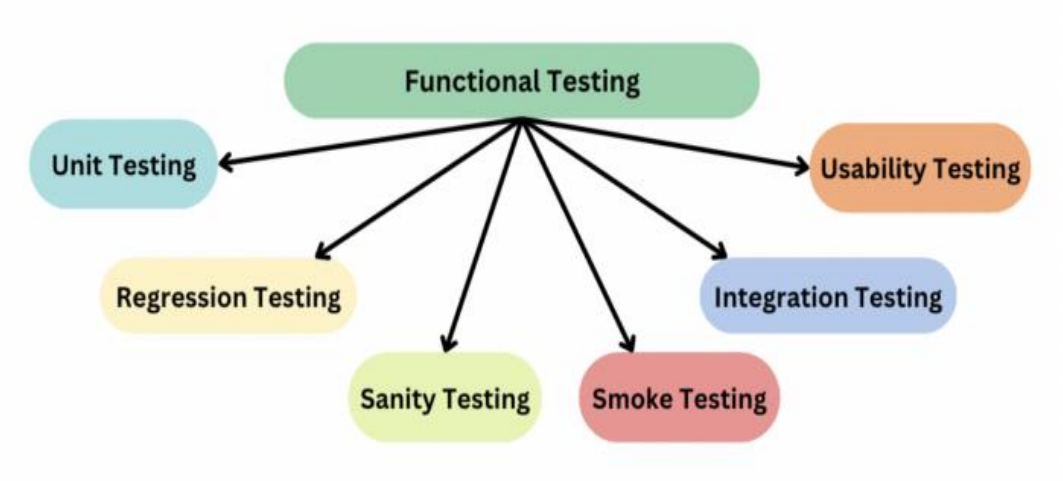
Learning Steps	Resources specific instructions
1. Students will ask the instructor about design styles with CSS and CSS framework	1. The instructor will provide the learning materials for test and confirm the website.
2. Read the **Information sheet/s**	2. Information Sheet No 4: Test and confirm the website. ▪ Testing Website Functionality and Correcting Error ▪ Checking Website Compatibility and Accessibility ▪ Evaluating Website Fitness for Purpose and Client Requirements
3. Complete the **Self-Checks & Answer key sheets.**	3. Self-Check No 4: Test and confirm the website. Answer key No 4: Test and confirm the website.
4. Read the **Job/ Task sheet** and **Specification Sheet**	4. Job/ task sheet and specification sheet Job Sheet No 4: Test and confirm the website. Specification Sheet 4: Test and confirm the website.

Information Sheet 4 Test and confirm the website

Learning Objective: After completing this information sheet, the learners will be able to test and confirm the website.

4.1 Testing Website Functionality and Correcting Error
4.2 Checking Website Compatibility and Accessibility
4.3 Evaluating Website Fitness for Purpose and Client Requirements

4.1 Testing Website Functionality and Correcting Error



Functional testing is a critical phase in web development that ensures a website's interactive elements and features function as intended. This type of testing evaluates the website's behavior, including its forms, buttons, navigation menus, and any dynamic content. The primary goal is to confirm that user interactions are smooth and that all features work consistently across various web browsers and devices.

- **Functional Testing:** Before anything else, it's vital to guarantee that your website functions as intended. This involves testing interactive elements like forms, buttons, navigation menus, and dynamic content. Ensure that user interactions are smooth and all features work across different browsers.
- **Error Identification:** It's natural for errors to occur during web development. These errors can range from broken links to misaligned elements and JavaScript bugs. Systematically identify and document these errors so that you can correct them efficiently.
- **Validation:** Use W3C Markup Service and CSS validation tools to check if your HTML and CSS code follows industry standards. Valid code is less prone to errors and functions better across various browsers.

Key Objectives of Functional Testing:

- **Verification of Functionality:** The core objective of functional testing is to verify that every interactive element on the website performs its intended function without errors or unexpected behavior. This includes features such as contact forms, login systems, search functionality, and interactive widgets.
- **Browser Compatibility:** Functional testing ensures that the website functions correctly on different web browsers like Google Chrome, Mozilla Firefox, Apple Safari, Microsoft Edge, and others. It's crucial because various browsers may interpret code differently, potentially leading to discrepancies in functionality.
- **Device Responsiveness:** Websites must be responsive and adaptive to different devices, including desktop computers, laptops, tablets, and smartphones. Functional testing assesses whether the website's interactive elements work seamlessly across various screen sizes and resolutions.

Key Steps in Functional Testing:

- **Test Planning:** Begin by creating a comprehensive test plan that outlines the scope of functional testing. Identify all interactive elements, features, and user flows that need to be tested. Specify the browsers and devices to be included in the testing process.
- **Test Case Design:** Develop detailed test cases for each interactive element and feature. Test cases should include step-by-step instructions for testing, expected outcomes, and criteria for success. Consider both positive and negative scenarios (expected user interactions) (error conditions).
- **Execution:** Execute the test cases on different browsers and devices as specified in the test plan. During testing, interact with the website as a user would and carefully observe the behavior of each element. Document any deviations from expected functionality.
- **Issue Reporting:** If any issues or defects are identified during testing, report them in a structured manner. Include detailed information about the issue, steps to reproduce it, the browser and device where it was encountered, and its impact on the user experience.
- **Regression Testing:** After issues are resolved, perform regression testing to ensure that fixes haven't introduced new problems. This step helps maintain the website's functionality as new features or changes are implemented.
- **Cross-browser Testing:** Test the website on various browsers, paying attention to differences in rendering and functionality. Use browser developer tools to debug and diagnose issues specific to particular browsers.
- **Device Testing:** Verify that the website functions correctly on different devices with varying screen sizes and resolutions. Emulators and real devices can be used to simulate mobile and tablet experiences.
- **Usability Testing:** As part of functional testing, assess the overall user experience. Ensure that the website's interactive elements are intuitive and user-friendly. Gather feedback on navigation, responsiveness, and overall satisfaction.

- **Accessibility Testing:** Functional testing should also consider accessibility. Ensure that interactive elements are accessible to users with disabilities, including those who rely on screen readers and keyboard navigation.

4.2 Checking Website Compatibility and Accessibility

- **Cross-browser Compatibility:** Test your website on multiple web browsers such as Chrome, Firefox, Safari, and Internet Explorer. Ensure that the website appears and functions consistently across these platforms. Use browser developer tools to identify and fix any compatibility issues.

  Why it's Important: Web users employ various browsers to access the internet, and each browser may interpret website code slightly differently. Ensuring cross-browser compatibility guarantees that your website looks and performs consistently, regardless of the browser used.

  Testing Methodology:

  - **Browser Selection:** Choose a set of commonly used web browsers, such as Google Chrome, Mozilla Firefox, Apple Safari, and Microsoft Edge, for testing.
  - **Browser Developer Tools:** Utilize browser developer tools to inspect and debug your website on each browser. These tools provide insights into rendering issues, JavaScript errors, and layout problems.
  - **Visual Inspection:** Manually inspect your website on different browsers to identify any visual discrepancies or layout issues. Pay attention to the alignment of elements, font rendering, and overall design consistency.
  - **Responsive Design:** Verify that your website adapts gracefully to different screen sizes, from large desktop monitors to small mobile devices. This responsiveness is critical for providing a seamless user experience.
  - **Why it's Important:** With the proliferation of various devices, your website must be responsive. Responsive design ensures that your site adapts gracefully to different screen sizes, providing an optimal user experience on desktop computers, laptops, tablets, and smartphones.

  Testing Methodology:

  - **Device Emulation:** Use device emulators or responsive design testing tools to simulate various screen sizes and resolutions. Verify that your website's layout adjusts appropriately.
  - **Manual Testing:** Physically test your website on various devices and screen sizes. Interact with the site to confirm that all interactive elements, such as buttons and forms, remain usable and accessible.
  - **Accessibility Testing:** Make sure your website is accessible to people with disabilities. Test with screen readers and keyboard navigation to ensure all users can

navigate and interact with your content. Address any accessibility issues that you discover.

- **Why it's Important:** Web accessibility is about making your website usable for everyone, including individuals with disabilities. Ensuring accessibility is a legal requirement in many regions and a moral imperative.

Testing Methodology:

- **Screen Reader Testing:** Use screen reader software (e.g., JAWS, NVDA, VoiceOver) to navigate your website. Ensure that all content, including text, images, and interactive elements, is accessible through screen readers.
- **Keyboard Navigation:** Test your website's functionality using only a keyboard. Ensure that all interactive elements can be accessed and used without a mouse.
- **Alt Text for Images:** Verify that images have appropriate alternative text (alt text) that describes their content or purpose.
- **Semantic HTML:** Use semantic HTML elements (e.g., headings, lists, labels) to structure content logically. Ensure that form fields have associated labels.
- **Colour Contrast:** Check the colour contrast of text and background elements to ensure readability, especially for individuals with visual impairments.
- **Testing Tools:** Utilize accessibility testing tools and browser extensions (e.g., Axe, WAVE) to automate some aspects of accessibility testing and identify issues.

4.3 Evaluating Website Fitness for Purpose and Client Requirements

- **Client Requirements:** Review the initial client requirements and project objectives. Ensure that your website aligns with these requirements. Ensure all requested features are implemented, and the design matches the client's expectations.
- **Usability Testing:** Conduct usability testing with potential users or stakeholders. Gather feedback on the website's ease of use, clarity of content, and overall user experience. Make improvements based on this feedback.
- **Performance Evaluation:** Assess the website's loading speed and overall performance. Use tools like Google PageSpeed Insights to identify areas for improvement. A fast-loading website is essential for retaining visitors.
- **Content Review:** Ensure that all content, including text, images, and multimedia elements, is accurate, up-to-date, and relevant to the website's purpose. Broken or outdated content can negatively impact the user experience.

An example Testing Case/Scenario:

XYZ Corporation, a global technology company, is revamping its corporate website to showcase its products, services, and corporate social responsibility initiatives. The company aims to reach a diverse audience, including potential clients, partners, and individuals with disabilities. The project team is committed to ensuring the website's compatibility across browsers and accessibility to all users.

1. **Cross-browser Compatibility Testing:**

   To address cross-browser compatibility, the project team selects four major web browsers for testing: Google Chrome, Mozilla Firefox, Apple Safari, and Microsoft Edge. They use browser developer tools and manual inspection to identify and resolve issues.
   Challenge: During testing, the team discovers that certain CSS styles are not rendering correctly in Internet Explorer (IE), an older browser still used by some corporate clients.
   Solution: The developers apply conditional CSS statements to target IE and resolve the rendering issues specifically. They also optimise JavaScript code to ensure compatibility with IE.

2. **Responsive Design Testing:**

   The team recognises the importance of responsive design for their global audience using various devices. They employ both device emulators and physical devices for testing.
   Challenge: Due to overcrowding, the website's navigation menu becomes difficult to use on smaller screens, such as smartphones and tablets.
   Solution: The designers implement a responsive navigation menu that collapses into a toggleable menu icon on smaller screens. This enhancement improves usability on all devices.

3. **Accessibility Testing:**

   To ensure accessibility, the team conducts thorough testing using screen readers, keyboard navigation, and accessibility testing tools.
   Challenge: Some images on the website lack alternative text, making them inaccessible to users relying on screen readers.
   Solution: The content team adds appropriate alt text to images throughout the website, ensuring they are adequately described for screen reader users.
   Challenge: Form fields on certain pages lack clear and associated labels, making it challenging for screen reader users to understand their purpose.
   Solution: The developers add descriptive labels to form fields, enhancing their accessibility.
   Challenge: The color contrast of text on some pages does not meet accessibility guidelines, potentially causing readability issues for users with visual impairments.
   Solution: The designers adjust the color scheme to meet minimum contrast requirements, improving text readability.

Self-Check Sheet 4 Test and confirm the website

1. What is the primary goal of functional testing in web development?
    - A. Checking website compatibility
    - B. Ensuring cross-browser compatibility
    - C. Verifying that interactive elements work as intended
    - D. Evaluating website fitness for purpose

2. During functional testing, what is the purpose of regression testing?
    - A. To identify errors systematically
    - B. To ensure cross-browser compatibility
    - C. To check responsiveness on mobile devices
    - D. To confirm that fixes haven't introduced new issues

3. Why is validation of HTML and CSS code essential in functional testing?
    - A. It ensures cross-browser compatibility.
    - B. It identifies visual discrepancies.
    - C. Valid code is less prone to errors.
    - D. It improves website performance.

4. What is the purpose of cross-browser compatibility testing?
    - A. To ensure that the website is accessible
    - B. To verify that the website functions as intended
    - C. To confirm that the website appears consistent on different browsers
    - D. To evaluate the website's loading speed

5. Why is responsive design testing important for a website?
    - A. To check for broken links
    - B. To ensure accessibility
    - C. To verify functionality
    - D. To adapt to different screen sizes

6. What is the main goal of accessibility testing?
    - A. To verify functionality
    - B. To ensure cross-browser compatibility
    - C. To make the website usable for people with disabilities
    - D. To assess website performance

7. In the context of website evaluation, what should you do to ensure the website aligns with client requirements?
    - A. Conduct usability testing
    - B. Review client feedback
    - C. Assess the loading speed
    - D. Review initial client requirements

8. What is the purpose of usability testing during website evaluation?
    - A. To assess loading speed
    - B. To gather feedback on the user experience
    - C. To identify broken links
    - D. To ensure cross-browser compatibility

9. Why is performance evaluation important for a website?
    - A. To ensure accessibility
    - B. To verify functionality
    - C. To assess loading speed and overall performance
    - D. To check for broken links

10. What should be reviewed to ensure that website content is accurate and relevant?
    - A. Browser compatibility
    - B. Code validation
    - C. Content alignment
    - D. Client feedback

Answer Sheet 4 Test and confirm the website

1. C. Verifying that interactive elements work as intended
2. D. To confirm that fixes haven't introduced new issues
3. C. Valid code is less prone to errors.
4. C. To confirm that the website appears consistent on different browsers
5. D. To adapt to different screen sizes
6. C. To make the website usable for people with disabilities
7. D. Review initial client requirements
8. B. To gather feedback on the user experience
9. C. To assess loading speed and overall performance
10. C. Content alignment

Job Sheet 4 Test Website Cross-browser compatibility

Job Name: Test Website Cross-browser compatibility.

Procedure:

1. Choose a set of commonly used web browsers.
2. Manually inspect your website on different browsers to identify visual discrepancies or layout issues.
3. Verify that your website adapts gracefully to different screen sizes, from large desktop monitors to small mobile devices.
4. Utilise browser developer tools to inspect and debug your website on each browser.
5. List all/any identified issues.
6. Fix issues, if any.

Specification Sheet 4 Test Website Cross-browser compatibility

Necessary tools and equipment

Sl. No	Name of Tools & Equipment	Specification	Unit	Quantity
1	Computer	Minimum Corei3 with 4 GB RAM	No.	1
2	Code Editor	Any	No.	1
3	Web Browser for Testing	Google Chrome, Mozilla Firefox, Apple Safari, and Microsoft Edge	No.	1

Other Specifications

1. Use the Website developed in Job 2 & 3 for reference.

Review of Competency

Below is your assessment rating for module **Design Styled with CSS and CSS Framework**

Assessment of Performance Criteria	Yes	No
The purpose and intended audience of the website are identified.		
The design requirements and constraints are identified.		
A conceptual design is developed.		
Necessary software installed as per requirement.		
Web layout is selected as per design requirement.		
Web layout is designed using CSS as per client's requirements.		
HTML and CSS file is integrated as required.		
Web site is saved and executed.		
Framework is collected and configured with website.		
HTML and CSS framework are integrated.		
CSS framework is customized using CSS as per requirements.		
Web site is saved and executed.		
The website is tested to ensure functionality and errors are corrected per standard operating procedure.		
The website is opened with common browsers and checked for accessibility, readability, legibility, and presentation in accordance with client requirements.		
The website is evaluated for fitness in terms of the purpose, target audience and specifications of client requirements.		

I now feel ready to undertake my formal competency assessment.

Signed:

Date:

Development of CBLM:

The Competency Based Learning Material (CBLM) of '**Design styles with CSS and CSS Framework' (**Occupation: Web Design and Development for Freelancing, Level-3**)** for National Skills Certificate is developed by NSDA with the assistance of SIMEC System, ECF consultancy & SIMEC Institute JV (Joint Venture Firm) in the month of June 2023 under the contract number of package SD-9A dated 07th May 2023.

Sl No.	Name & Address	Designation	Contact number
1	Khondoker Ali Asgor Pavel	Writer	01711 873 008
2	Nasirn Khondakar	Editor	01912 464 747
3	Md. Amir Hossain	Co-Ordinator	01631 670 445
4	Mahbub Ul Huda	Reviewer	01735 490 491