REAL WEB DEV 2024 Guide 👍

My strategy to being a noob software developer (Easy)

Class Contents for the next 30 days :

Using an IDE (Types and preferred)

Working with extensions

Working with Tags, Elements & Attributes

Document Structures : Text Formatting & Semantics.

Design concepts: Links & Images

HTML 5 Semantic Fillings

CSS tags and uses

Selectors

Styling text, color and background

Layouts

Responsive Designing : Introduction to responsive designing

Advanced CSS

- Creating your first static website (Project)

Addition:

- Using vscode
- using Github
- Learning to use Github with vscode

By the end of this 30 day s yall would have learnt the basics of software development and understood the basics of what it at least takes to start off your journey in frontend development (and got your very own static website)

It is a vast sector and a lot of information is to be grasped so if you the type to consistently want to learn, then software development is for you

(this program is free for the first class but will change as the time progresses to learn more but for now...

just study bruhh💀 💀)

Using an IDE

1. Commonly Used IDEs:

- **Visual Studio Code (VSCode)**: A popular, free code editor with extensions for various languages and tools.

- **WebStorm**: A powerful IDE for modern JavaScript development.

- **Atom**: A hackable text editor for the 21st century.

- **Sublime Text**: A sophisticated text editor for code, markup, and prose.

3. Common Shortcuts in VSCode:

- **Open Command Palette**: `Ctrl+Shift+P` or `Cmd+Shift+P` on Mac.

- **Toggle Terminal**: `Ctrl+`` or `Cmd+`` on Mac.

- **Find Files**: `Ctrl+P` or `Cmd+P` on Mac.

- **Go to Definition**: `F12` or `Cmd+Click` on Mac.

Working with Extensions

1. Commonly Used Extensions in VSCode:

- **ESLint**: Integrates ESLint into VSCode.

- **Prettier**: An opinionated code formatter.

- **Live Server**: Launch a local development server with live reload feature.

- **Bracket Pair Colorizer**: Colorizes matching brackets.

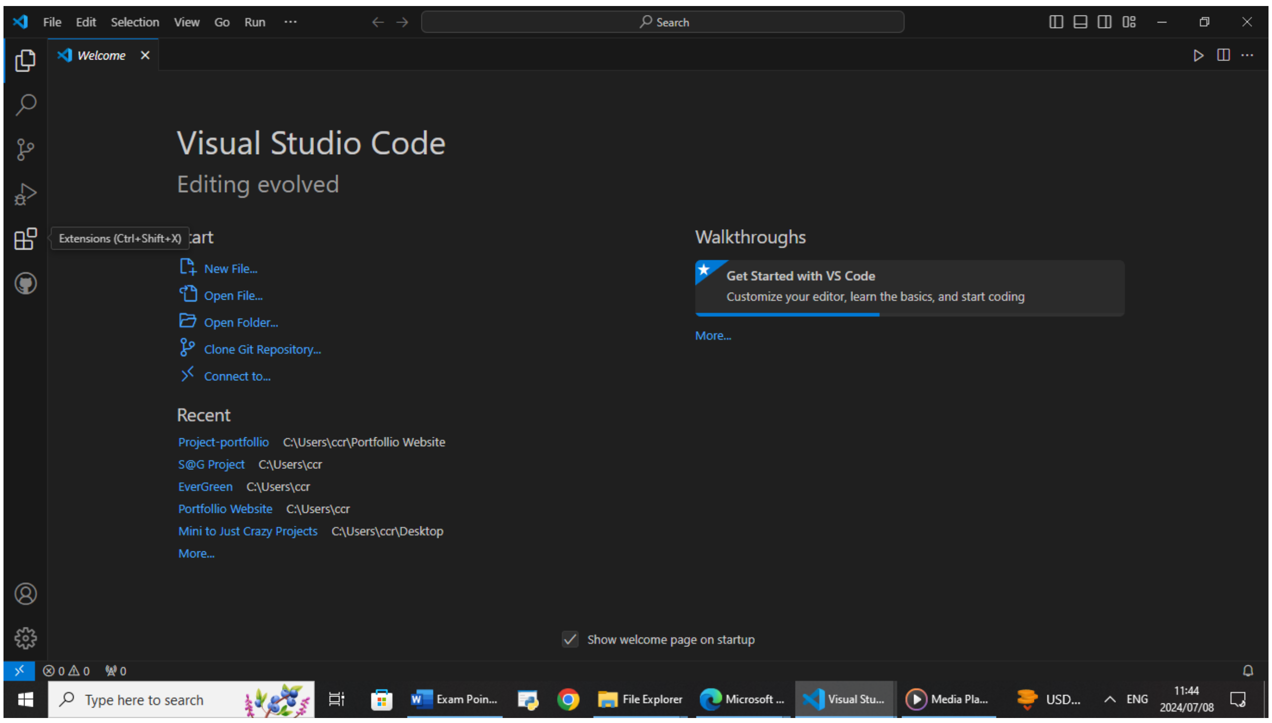
- **GitLens**: Supercharges the Git capabilities built into VSCode.

Installing Extensions

- Open the Extensions view by clicking on the Extensions icon in the Activity Bar on the side of the window or by pressing `Ctrl+Shift+X`.

- Search for the extension you want to install.

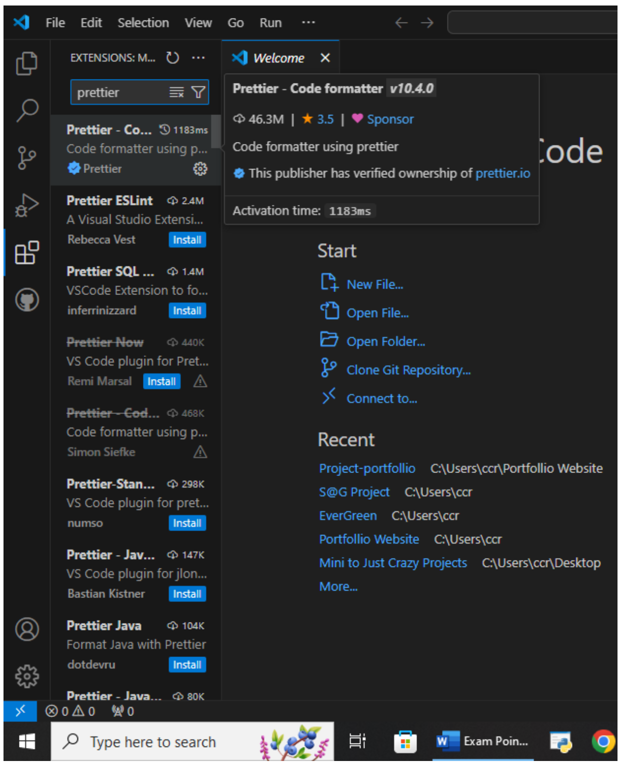
- Click the **Install** button.



Just a few major extensions you might need so heres the few you will start with :

- **Prettier**
- **Live Server**
- **Live Preview**

(just search it on the top icon and search each title, then install so that you can start with the process)



Working with Tags, Elements & Attributes

Tag	Description	Common Attributes	Example
<html>	Root element of an HTML document	lang, dir	<html lang="en">
<head>	Container for metadata (not displayed on the page)		<head>
<title>	Sets the title of the document		<title>My Page</title>
<body>	Container for the content of the document	background, bgcolor, text, link, vlink, alink	<body>
<h1> to <h6>	Headings, with <h1> being the highest level	id, class, style	<h1>Heading 1</h1>
<p>	Paragraph	id, class, style, align	<p>This is a paragraph.</p>
<a>	Anchor (hyperlink)	href, target, title, rel	<a href="https://example.com">Link</a>
<img>	Image	src, alt, height, width	<img src="image.jpg" alt="Description">
<ul>	Unordered list	id, class, style	<ul><li>Item 1</li><li>Item 2</li></ul>
<ol>	Ordered list	id, class, style, type, start, reversed	<ol><li>Item 1</li><li>Item 2</li></ol>
<li>	List item	id, class, style, value	<li>List item</li>

Tag	Description	Common Attributes	Example
<div>	Division or section	id, class, style	<div class="container">Content</div>
<span>	Inline container for text	id, class, style	<span style="color:red;">Red text</span>
<table>	Table	id, class, style, border, cellpadding, cellspacing	<table><tr><td>Data</td></tr></table>
<tr>	Table row	id, class, style	<tr><td>Row</td></tr>
<td>	Table data cell	id, class, style, colspan, rowspan	<td>Cell</td>
<th>	Table header cell	id, class, style, scope, colspan, rowspan	<th>Header</th>
<form>	Form for user input	id, class, style, action, method, enctype, target	<form action="/submit" method="post"></form>
<input>	Input field	type, name, value, id, class, style, placeholder	<input type="text" name="username">
<button>	Button	type, name, value, id, class, style, disabled	<button type="submit">Submit</button>
<label>	Label for an input element	for, id, class, style	<label for="username">Username:</label>
<textarea>	Multi-line text input field	name, id, class, style, rows, cols, placeholder	<textarea name="message" rows="4" cols="50"></textarea>
<select>	Drop-down list	name, id, class, style,	<select name="options"><option>Option 1</option></select>

Tag	Description	Common Attributes	Example
		multiple, size	
<option>	Option in a drop-down list	value, selected, id, class, style	<option value="1">Option 1</option>
<link>	Link to an external resource	href, rel, type, media, hreflang, sizes	<link rel="stylesheet" href="styles.css">
<script>	Script for client-side JavaScript	src, type, async, defer	<script src="script.js"></script>
<meta>	Metadata about the HTML document	charset, name, content, http-equiv	<meta charset="UTF-8">
<style>	Style definitions	type, media	<style>body { font-family: Arial; }</style>
<header>	Header section of a document or section	id, class, style	<header>Header content</header>
<footer>	Footer section of a document or section	id, class, style	<footer>Footer content</footer>
<nav>	Navigation links	id, class, style	<nav><a href="#">Home</a></nav>
<section>	Section of a document	id, class, style	<section>Section content</section>
<article>	Independent piece of content	id, class, style	<article>Article content</article>
<aside>	Content indirectly related to the main content	id, class, style	<aside>Aside content</aside>
<main>	Main content of the document	id, class, style	<main>Main content</main>
<figure>	Figure with optional caption	id, class, style	<figure><img src="image.jpg" alt="Image"><figcaption>Caption</figcaption></figure>
<figcaption>	Caption for a <figure> element	id, class, style	<figcaption>Caption</figcaption>
<blockquote>	Block quotation	cite, id, class, style	<blockquote cite="https://example.com">Quote</blockquote>
<q>	Inline quotation	cite, id, class, style	<q cite="https://example.com">Quote</q>

Tag	Description	Common Attributes	Example
<code>	Inline code snippet	id, class, style	<code>inline code</code>
<pre>	Preformatted text	id, class, style	<pre>Preformatted text</pre>
<em>	Emphasized text	id, class, style	<em>Emphasized</em>
<strong>	Strongly emphasized text	id, class, style	<strong>Strong</strong>
<br>	Line break	id, class, style	<br>
<hr>	Horizontal rule	id, class, style	<hr>

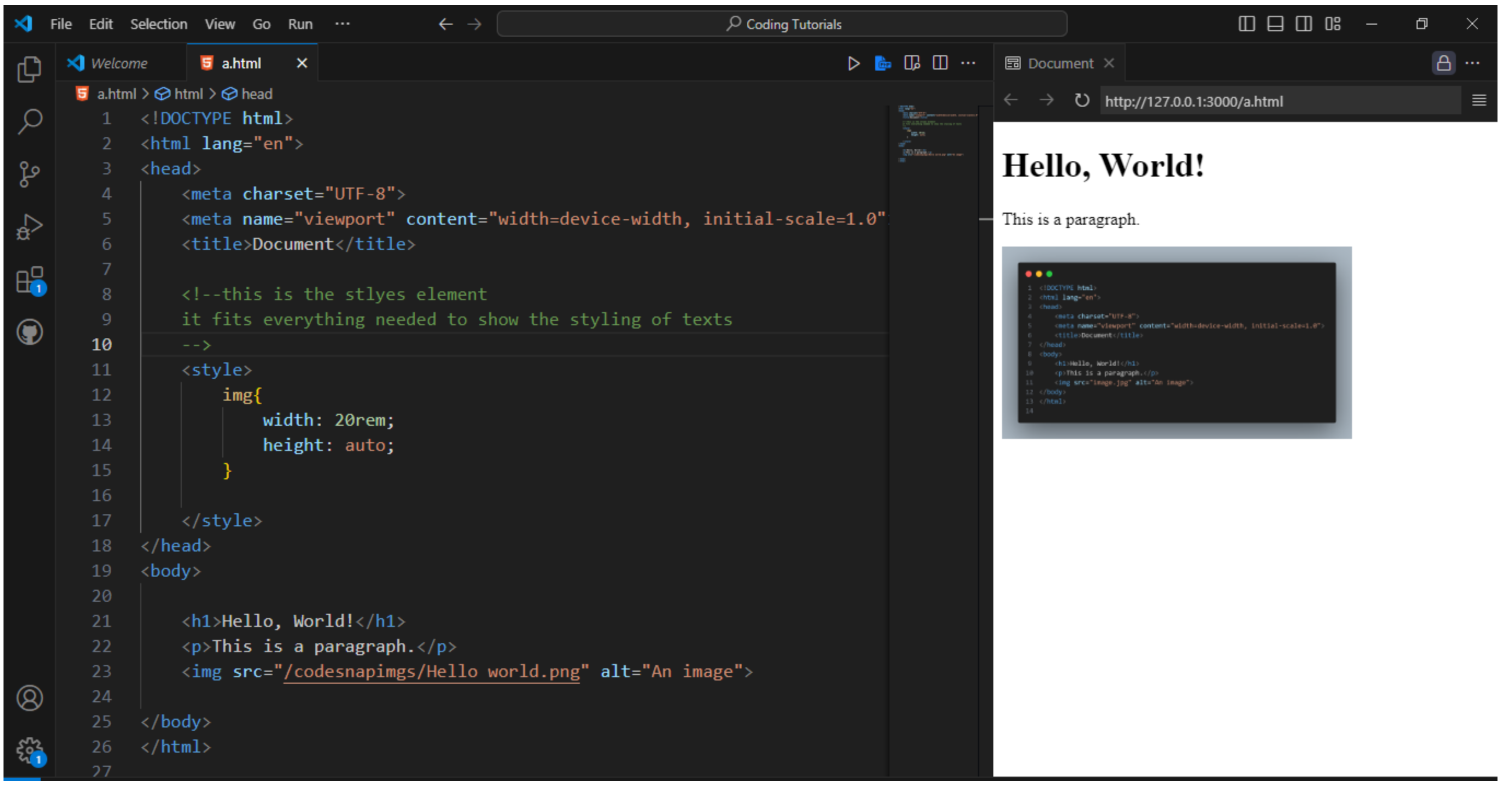
HTML Basics

- **Tags**: Keywords surrounded by angle brackets, e.g., `<div>`.

- **Elements**: Consist of an opening tag, content, and a closing tag, e.g., `<div>Hello World</div>`.

- **Attributes**: Provide additional information about elements, e.g., `<img src="image.jpg" alt="Description">`.

Example 1


File Edit Selection View Go Run ···
Coding Tutorials
Welcome
a.html
a.html > html > head
Document
http://127.0.0.1:3000/a.html

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0"
    <title>Document</title>

    <!--this is the stlyes element
    it fits everything needed to show the styling of texts
    -->
    <style>
        img{
            width: 20rem;
            height: auto;
        }

    </style>
</head>
<body>

    <h1>Hello, World!</h1>
    <p>This is a paragraph.</p>
    <img src="/codesnapimgs/Hello world.png" alt="An image">

</body>
</html>
```

```
Hello, World!

This is a paragraph.
```

Summary

- **IDE**: Use VSCode for features like syntax highlighting, code completion, and integrated terminal.

- **Extensions**: Install extensions like ESLint, Prettier, and Live Server to enhance your coding experience.

- **HTML & React**: Understand the basic tags, elements, and attributes. Use JSX in React to write HTML-like code and manage state and props for dynamic data handling.

These notes should help you get started with using an IDE, extensions, and understanding the basics of HTML and React.

Document Structures: Text Formatting & Semantics

Text Formatting

HTML provides several tags for text formatting, such as:

- `<strong>` for strong importance (usually bold)
- `<em>` for emphasized text (usually italic)
- `<mark>` for highlighted or marked text
- `<del>` for deleted text
- `<ins>` for inserted text

Example:

```html
<p>This is <strong>important</strong> information.</p>

<p>Here is some <em>emphasized</em> text.</p>
```

Semantics

Semantic HTML tags provide meaning to the content, aiding accessibility and SEO:

- `<header>`, `<footer>`, `<nav>` for structural sections
- `<article>`, `<section>`, `<aside>` for content sections
- `<main>` for the main content area
- `<figure>` and `<figcaption>` for images and their captions

Example:

```html
<header>

  <h1>Website Header</h1>

  <nav>

    <ul>

      <li><a href="#">Home</a></li>

      <li><a href="#">About</a></li>
    </ul>
  </nav>
</header>

<article>
  <h2>Article Title</h2>
  <p>Article content goes here.</p>
</article>

<aside>
  <h3>Related Links</h3>
  <ul>
    <li><a href="#">Link 1</a></li>
    <li><a href="#">Link 2</a></li>
  </ul>
</aside>

<footer>
  <p>&copy; 2024 Your Website</p>
</footer>
```

HTML5 introduces semantic elements to better structure web pages:

- `<header>`, `<footer>`, `<nav>`: For header, footer, and navigation sections.
- `<article>`, `<section>`, `<aside>`: For content sections and sidebars.
- `<main>`: For the main content of the page.
- `<figure>` and `<figcaption>`: For images and their captions.

Design Concepts: Links & Images

Links

Creating links with `<a>` tag:

html

```
<a href="https://example.com">Visit Example</a>
```

Adding target and title attributes:

```
<a href="https://example.com" target="_blank" title="Visit Example in a new tab">Visit
Example</a>
```

Images

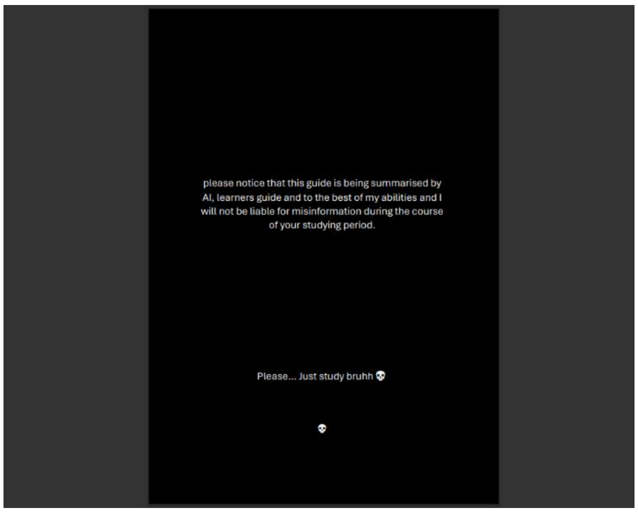
Displaying images with `<img>` tag:

```
<img src="image.jpg" alt="Description of the image">
```

Using width and height attributes (optional):

```
<img src="image.jpg" alt="Description" width="300" height="200">
```

Exercise:



please notice that this guide is being summarised by AI, learners guide and to the best of my abilities and I will not be liable for misinformation during the course of your studying period.

Please... Just study bruhh 💀

💀

These examples and descriptions should help in understanding how to structure HTML documents effectively using text formatting, semantic elements, links, and images. Let me know if you need further explanations or examples!

CSS (Cascading Style Sheets)

CSS (Cascading Style Sheets) is used to style the appearance of HTML elements on a web page. Here's an overview of CSS tags, their uses, and selectors:

CSS Tags and Uses:

1. Universal Selector (`*`)

Applies styles to all elements on the page.

```
 css
* {
  margin: 0;
  padding: 0;
}
```

2. Type Selector

Targets all instances of a specific HTML tag.

```css
h1 {
  font-size: 24px;
}
```

3. Class Selector (`.class`)

Targets elements with a specific class attribute.

```css
.highlight {
  background-color: yellow;
```

}

4. ID Selector (`#id`)

Targets a single element with a specific ID attribute.

```css
#main-content {
   width: 80%;
}
```

5. Attribute Selector (`[attribute=value]`)

Targets elements with a specific attribute and value.

```css
input[type="text"] {
   border: 1px solid #ccc;
}
```

Uses of CSS

- **Styling**: Control the visual presentation of HTML elements (e.g., colors, fonts, margins).

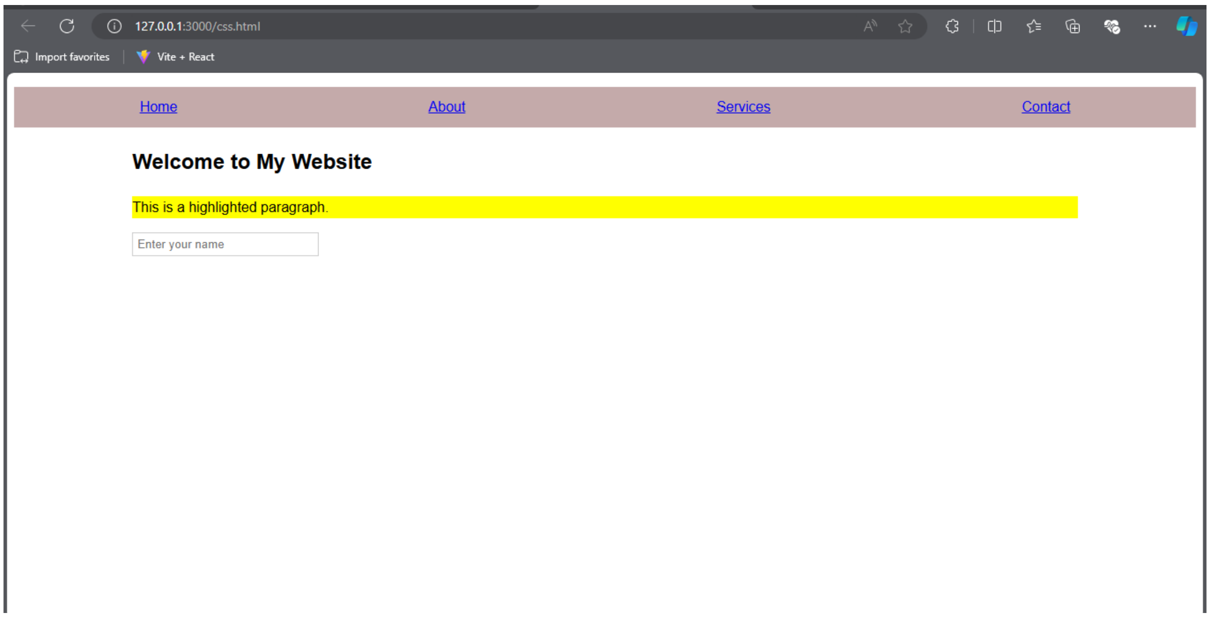
- **Layout**: Define the layout of a web page (e.g., positioning, grid systems, flexbox).

- **Responsive Design**: Adapt styles based on the device or screen size.

- **Animations and Transitions**: Add motion and interactivity to elements.

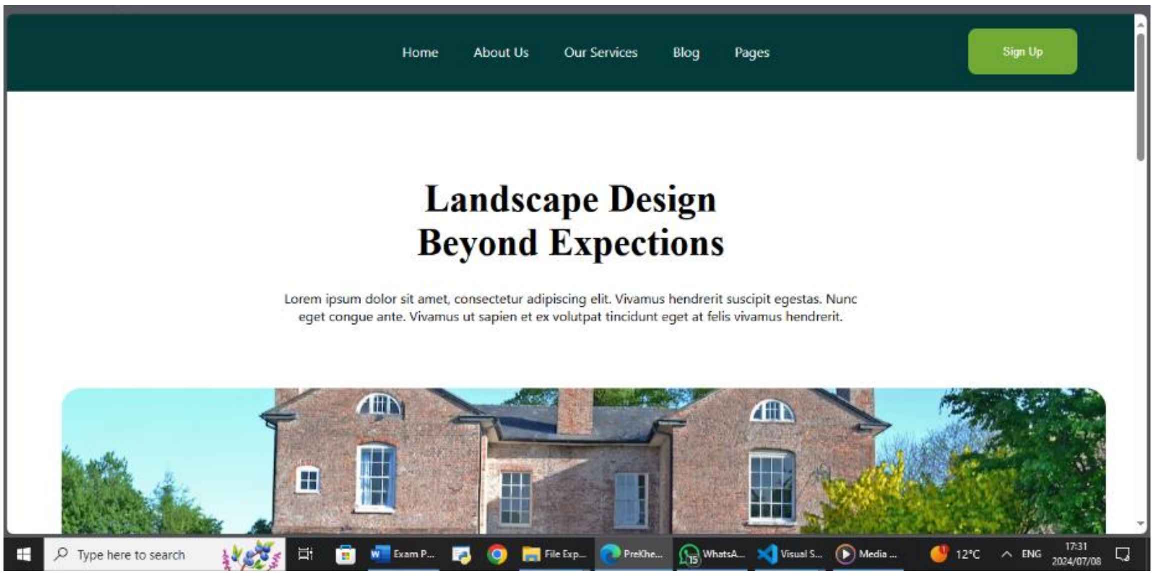
- **Accessibility**: Enhance accessibility by adjusting styles for readability and usability.

Example (design is css.html)



Welcome to My Website

This is a highlighted paragraph.



Landscape Design Beyond Expections

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vivamus hendrerit suscipit egestas. Nunc eget congue ante. Vivamus ut sapien et ex volutpat tincidunt eget at felis vivamus hendrerit.

In this example, CSS is used to style various elements on the page using different selectors and properties. Each selector targets specific elements or groups of elements based on their type, class, ID, attributes, or relationships with other elements.

Responsive Design

Responsive Design is an approach to web design that ensures a web page looks good on all devices and screen sizes. It involves creating flexible layouts and using CSS media queries to adjust styles based on the device characteristics like screen width, height, orientation, etc. Here’s an introduction to responsive designing:

Principles of Responsive Design

1. **Flexible Grid Layouts**: Use relative units like percentages or `em` instead of fixed units (`px`) for widths and heights to allow elements to resize based on the screen size.

2. **Media Queries**: CSS3 media queries allow you to apply different styles based on the characteristics of the device. They typically check for the width of the viewport and adjust styles accordingly.

   Example of a media query:

   ```css
   /* For screens smaller than 768px wide */
   @media (max-width: 768px) {
     .container {
       width: 100%;
       padding: 10px;
     }
   }

3. **Flexible Images and Media**: Use `max-width: 100%;` for images and videos to ensure they scale within their parent containers without exceeding the viewport width.
   ```

Example:

~~~css

img {

    max-width: 100%;

    height: auto;

}

4. **Viewport Meta Tag**: Ensure proper scaling and viewport dimensions with the viewport meta tag in the `<head>` of your HTML:

```html

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

### Benefits of Responsive Design

- **Improved User Experience**: Ensures usability and readability across devices, reducing the need for zooming or horizontal scrolling.

- **SEO Benefits**: Responsive sites tend to perform better in search rankings as they provide a better user experience and are preferred by search engines.

- **Cost-Effective**: Instead of creating separate websites or apps for different devices, responsive design adapts to various screen sizes with a single codebase.

- **Future-Proofing**: With the diversity of devices continually expanding, responsive design helps ensure your site remains accessible and functional on new devices.
~~~

Example of Responsive Design

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Responsive Design Example</title>
    <style>
        /* Basic styles */
        body {
            font-family: Arial, sans-serif;
            line-height: 1.6;
            margin: 0;
            padding: 0;
        }

        .container {
            width: 80%;
            margin: 0 auto;
            padding: 20px;
            background-color: #f0f0f0;
        }

        /* Media query for smaller screens */
        @media (max-width: 768px) {
            .container {
                width: 100%;
            padding: 10px;
          }
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Welcome to Responsive Design</h1>
        <p>This page demonstrates the basics of responsive design using CSS media
queries.</p>
        <img src="example.jpg" alt="Example Image">
    </div>
</body>
</html>
```

In this example, the `.container` element adjusts its width and padding using a media query when the screen width is 768 pixels or less, ensuring a better layout on smaller devices.

Responsive design is crucial for creating websites that are accessible and user-friendly across a wide range of devices, from smartphones and tablets to desktop computers. It involves understanding how to use flexible layouts, media queries, and viewport settings to ensure a consistent and enjoyable user experience on any device.