Html

<!DOCTYPE html>

<html>

<head>

<title> ---title of the document>

<meta> --- page description,character set

<link> used to link other files like CSS

<style> Internal Css

<script> Java script

<body>

<h1> headings

<p> paragraph

<img> embed the image

<a> Hyperlink

<div> division/section

<ul>,<ol><li> ordered or unordered lists

<table> can create table

<span> group inline elements

Semantic Elements in HTML

Many web sites contain HTML code like: <div id="nav"> <div class="header"> <div id="footer"> to indicate navigation, header, and footer.

In HTML there are some semantic elements that can be used to define different parts of a web page:

* <article> --- self-contained content.
* <aside> --- It is aside from the content
* <details>
* <figcaption>
* <figure>
* <footer> -- footer of the document
* <header>
* <main>
* <mark>
* <nav> -- nav bar contains navigation links
* <section> -- defines a section in a document
* <summary>
* <time>

<Table>

<form> used to take users information(textfield,submit,radio,checkboxes)

<frames> --- An HTML iframe is used to display a web page within a web page.

<div> - division/ block (used with id/class)

**Understanding CSS selectors, properties, and values**

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

---- Type Selector(Tag Selector): It targets the elements by their tag ex: <p>

---- Class Selector : It targets the elements by their class attribute ex: class = “min”

In css : .min{ }

Id selector : It targets the elements by their id ex: id=”header”

In css: #header { }

Attribute selectors : select elements based on an attribute or attribute value

Type=”text”

Pseudo-class selectors (select elements based on a certain state)

Ex: a:hover{

}

Pseudo-elements selectors (select and style a part of an element)

P::first-letter{

font-weight: bold;

}

**Implementing CSS box model**

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

div {  
 width: 320px;  
 height: 50px;  
 padding: 10px;  
 border: 5px solid gray;  
 margin: 0;  
}

320px (width of content area)  
+ 20px (left padding + right padding)  
+ 10px (left border + right border)  
**= 350px (total width)** 50px (height of content area)  
+ 20px (top padding + bottom padding)  
+ 10px (top border + bottom border)

**= 80px (total height)**

<!DOCTYPEhtml>

<htmllang="en">

<head>

<metacharset="UTF-8">

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

<title>Document</title>

<style>

.main

{

font-size:30px;

font-weight:bold;

Text-align:center;

}

.class

{

margin-left:100px;

border:50pxsolid Purple;

width:300px;

height:200px;

text-align:center;

padding:50px;

}

.class1

{

font-size:40px;

font-weight:bold;

color:black;

margin-top:80px;

margin-left: 10px;

background-color:purple;

}

.class2

{

font-size:20px;

font-weight:bold;

background-color:white;

margin-top: 50px;

}

</style>

</head>

<body>

<divclass = "main">CSS Box-Model Property</div>

<divclass = "class">

<divclass = "class1"> We are learning</div>

<divclass = "class2">CSS BoX Model</div>

</div>

</body>

</html>

**Working with CSS layouts: flexbox and grid**

**Flexbox:**

The flexbox or flexible box layout module (usually referred to as flexbox) is a one-dimensional layout model for distributing space between items and includes numerous alignment capabilities.

FlexBox:

----------- Aligning items either horizontally or vertically with in a container.

<!DOCTYPEhtml>

<htmllang="en">

<head>

<metacharset="UTF-8">

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

<title>Document</title>

<style>

.flex-container{

display: flex;

padding: 30px;

margin: 30px;

height: 300px;

width: 300px;

border: 2pxsolid black;

justify-content: space-evenly;

align-items: center;

}

.flex1-container{

display: inline-flex;

height: 200px;

width: 400px;

padding: 30px;

margin: 30px;

background-color: brown;

justify-content: space-evenly;

}

</style>

</head>

<body>

<divclass="flex-container">

<div>one</div>

<div>two</div>

<div>three</div>

</div>

<divclass="flex1-container">

<div>1</div>

<div>2</div>

<div>3</div>

</div>

</body>

</html>

**Grid Layout**

Grid Layout is two dimensional , we can work in both row and coloum.

<!DOCTYPEhtml>

<htmllang="en">

<head>

<metacharset="UTF-8">

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

<title>Document</title>

<style>

.wrapper{

display: grid;

grid-template-columns: repeat(5, 1fr);

grid-gap: 30px;

}

</style>

</head>

<body>

<divclass="wrapper">

<div>One</div>

<div>Two</div>

<div>Three</div>

<div>Four</div>

<div>Five</div>

</div>

</body>

</html>

**Working with CSS3 transitions and animations**

CSS Transitions

CSS transitions allows you to change property values smoothly, over a given duration.

<!DOCTYPEhtml>

<htmllang="en">

<head>

<metacharset="UTF-8">

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

<title>Document</title>

<style>

div{

transition: background-color 0.5sease; }

div:hover{

background-color: red;

}

</style>

</head>

<body>

<div> jsvhdskjhsdkhfds</div>

</body>

</html>

## **CSS Animations**

An animation lets an element gradually change from one style to another.

You can change as many CSS properties you want, as many times as you want.

To use CSS animation, you must first specify some keyframes for the animation.

Keyframes hold what styles the element will have at certain times.

<!DOCTYPEhtml>

<htmllang="en">

<head>

<metacharset="UTF-8">

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

<title>Document</title>

<style>

@keyframesexample {

from {background-color: red;}

to {background-color: yellow;}

}

/\* The element to apply the animation to \*/

div {

width: 100px;

height: 100px;

background-color: red;

animation-name: example;

animation-duration: 4s;

}

</style>

</head>

<body>

<divexample="animation"> hjdsgcdsjhcgdjshgcs</div>

</body>

</html>

**Understanding responsive design with CSS3 media queries**

Media queries in CSS3 extended the CSS2 media types idea: Instead of looking for a type of device, they look at the capability of the device.

Media queries can be used to check many things, such as:

* width and height of the viewport
* orientation of the viewport (landscape or portrait)
* resolution

Using media queries are a popular technique for delivering a tailored style sheet to desktops, laptops, tablets, and mobile phones (such as iPhone and Android phones).

<!DOCTYPEhtml>

<htmllang="en">

<head>

<metacharset="UTF-8">

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

<title>Document</title>

</head>

<style>

@mediascreen and (max-width: 900px) {

body{

background-color: blue;

}

}

@mediascreen and (max-width: 600px) {

body{

background-color: red;

}

}

</style>

<body>

</body>

</html>

**Working with Bootstrap for responsive design**

Bootstrap is a free, open source front-end development framework for the creation of websites and web apps. Designed to enable responsive development of mobile-first websites, Bootstrap provides a collection of syntax for template designs.

As a framework, Bootstrap includes the basics for responsive web development, so developers only need to insert the code into a pre-defined grid system. The Bootstrap framework is built on Hypertext Markup Language (HTML), cascading style sheets (CSS) and JavaScript. Web developers using Bootstrap can build websites much faster without spending time worrying about basic commands and functions.

Where to Get Bootstrap?

There are two ways to start using Bootstrap on your own web site.

You can:

* Download Bootstrap from getbootstrap.com
* Include Bootstrap from a CDN

**Containers**

Bootstrap also requires a containing element to wrap site contents.

There are two container classes to choose from:

1. The .container class provides a responsive **fixed width container**
2. The .container-fluid class provides a **full width container**, spanning the entire width of the viewport

Assignment 1: Implement a navigation bar using an unordered list with links to different sections of your HTML page. Use CSS to style the list as a horizontal menu and highlight the current page or section.

Assignment 2: Create a simple HTML page that includes the use of headings, paragraphs, and at least two semantic tags like <article> or <section>. Add a table with data of your choice and a form with fields for a user's name, email, and a submit button.

Assignment 3: Apply the CSS box model to ensure that your page content has appropriate margins and padding. Create a layout using div tags and style them to arrange content in a multi-column format using floats or flexbox.

Assignment 4: Enhance the page by adding CSS3 animations to the menu and form elements. Also, use media queries to make the page responsive, ensuring it looks good on both desktop and mobile screen sizes.

Assignment 5: Take the static HTML page from Day 1 and integrate Bootstrap. Refactor the navigation bar, table, and form to use Bootstrap components. Ensure the page is responsive using Bootstrap's grid system."