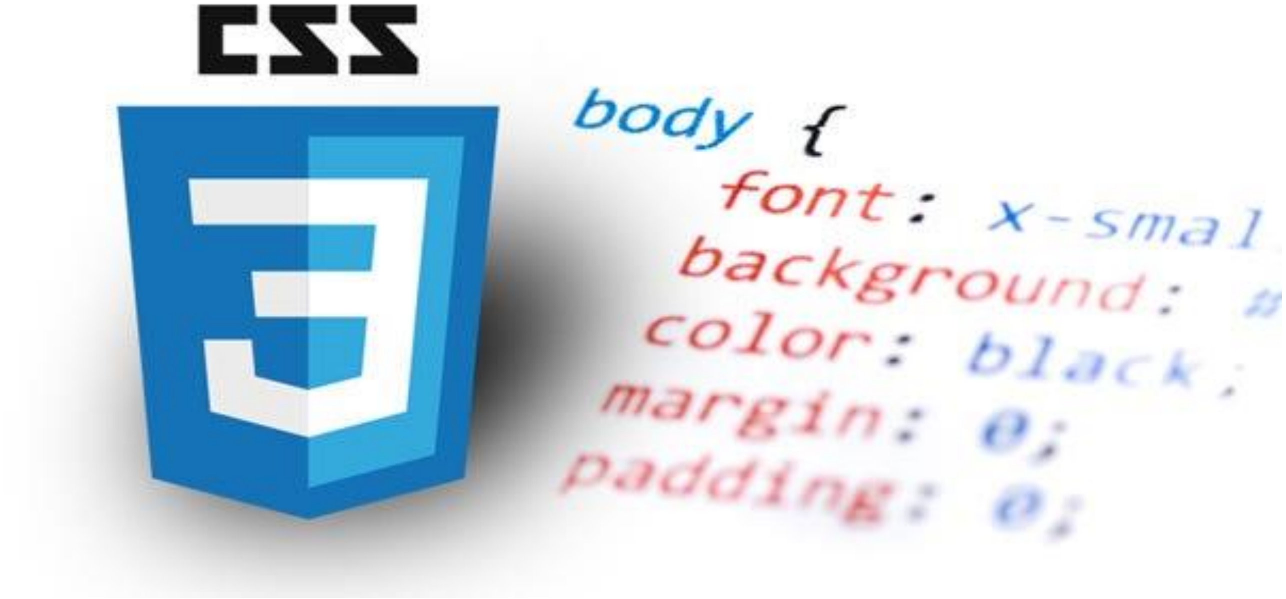
የሥራና ክህሎት ሚኒስቴር
Ministry of Labor and Skills

Web Development and Database Administration Level-III

Based on November 2023, Curriculum Version II



Module Title: - Developing Cascading style sheets

Module code: EIS WDDBA3 M05 1123

Nominal duration: 90 Hour

Prepared by: Ministry of Labor and Skill

November, 2023
Addis Ababa, Ethiopia

Table of Content

ACKNOWLEDGMENT

Ministry of Labor and Skills wish to extend thanks and appreciation to the many representatives of TVET instructors and respective industry experts who donated their time and expertise to the development of this Teaching, Training and Learning Materials (TTLM).

ACRONYM

CJK	Chinese, Japanese, Korean
CSS	Cascading Style Sheets
DOM	Document Object Model
GUI	Graphical User Interface
HSL	Hue, Saturation, Lightness
HSLA	Hue, Saturation, Lightness, Alpha
HTML	Hypertext Markup Language
LAB	Lightness, A-axis, B-axis
LTR	Left-to-Right
PHP	Hypertext Pre-processor
RGB	Red, Green, Blue
RGBA	Red, Green, Blue, Alpha
RTL	Right-to-Left
SASS	Syntactically Awesome Stylesheets
SEO	Search Engine Optimization
URL	Uniform Resource Locator
UX	User Experience
WCAG	Web Content Accessibility Guidelines
SVG	-Scalable Vector Graphic

INTRODUCTION TO THE MODULE

In web development and database administration filed; Develop Cascaded style sheets helps to achieve the performance outcomes, skills and knowledge required to use web authoring tools to create, modify and test CSS in a website.

This module is designed to meet the industry requirement under web development and database administration occupational standard, particularly for the unit of competency: Develop Cascaded style sheets

This module covers the units:

- Foundations of CSS and User-Centric Design
- Web page creations using CSS and HTML
- web page and CSS Validation and testing

Learning Objective of the Module

At the end of the module the trainee will be able to:

- Understand the fundamentals of CSS and essential design principles
- Demonstrate method of positioning and arranging document elements
- Understand the concept of style sheets and their application across multiple pages
- solve designing challenges different screen sizes
- Explore techniques for handling browser-specific styling challenges
- Emphasize the importance of adhering to industry standards in CSS
- Address browser-specific differences to enhance website accessibility

Module Instruction

For effective use these modules trainees are expected to follow the following module instruction:

1. Read the information written in each unit
2. Accomplish the Self-checks at the end of each unit
3. Perform Operation Sheets which were provided at the end of units

UNIT ONE: FOUNDATIONS OF CSS AND USER-CENTRIC DESIGN

4. Do the “LAP test” giver at the end of each unit

This unit is developed to provide you the necessary information regarding the following content coverage and topics:

- Introduction to CSS and basic design principles
- User requirements for style
- Development of CSS to Match User Requirements

This unit will also assist you to attain the learning outcomes stated in the cover page. Specifically, upon completion of this learning guide, you will be able to:

- gather and analyse user preferences for styling
- prioritize and incorporate user preferences into the design process
- understand and apply basic design principles, such as layout, color, and typography
- apply CSS concepts to create simple styles for web elements
- Ability to address challenges in translating user requirements into effective CSS styles

1.1. Introduction to CSS and Basic Design Principles

CSS is a stylesheet language used to control the presentation and layout of HTML or XML documents. It allows for the separation of content and design, enabling consistent and visually appealing web pages. Basic design principles encompass styling elements, layout structuring, and maintaining a cohesive visual aesthetic. This foundational knowledge is essential for creating well-designed and user-friendly websites.

- **Advantages and areas of application of CSS**

CSS (Cascading Style Sheets) offers advantages such as enhanced website consistency, easier maintenance, and improved page loading speed. It provides a separation of content and design, making styling efficient. CSS is widely applied in web development for creating responsive layouts, customizing visual elements, and ensuring a seamless user experience across various devices. Its versatility and adaptability make it a cornerstone technology for modern web design.

1.2. USER REQUIREMENTS FOR CSS STYLE

User requirements are the specific needs and wants of a user that must be taken into account during the web design quality assurance process. The user requirements gathering process includes surveys, interviews, focus groups, and more. Once you determine user requirements and set an acceptance criteria checklist for each, you should put them in a website requirements document. Those will help you with the UX design process by specifying what the visitor should see on the website, how it should work, and any special considerations that need to be taken into account.

- **Accessibility guidelines for UX design process**

Website accessibility guidelines WCAG are a set of best practices that help people with disabilities use the internet. These guidelines help web designers to make websites easier to use, by ensuring they are accessible to everyone. Those guidelines are sorted into the following categories:

- **Operable**

You need to make sure that every task is performable on both keyboard and mouse, that users have enough time to complete a task.

- **Perceivable**

You need to ensure that all users can easily perceive your content. Provide text alternatives for videos and images, plan on implementing audio alternatives for textual content, and ensure it can be consumed by visitors who use screen readers.

- **Understandable**

Maximizing the chances that visitors will be able to make sense of your content is vital in web design. You can do it by avoiding complex words and instructions, explaining error messages and interactive website design elements in detail..

- **Robust**

Through quality assurance, you need to ensure that your website is accessible from all the different browsers and devices people use to explore the internet

1.3. DEVELOPMENT OF CUSS TO MATCH USER REQUIREMENTS

Before you start writing user requirements for a website design you need to deeply understand your target audience.

- **Build trust with customers**

As people mostly buy products and services from businesses they trust, the e-Commerce development process needs to be focused on building a website that invokes trustworthiness.

First impressions are lasting impressions, and there are several ways in which web strategists can secure positive ones:

- **Mirror the words of your target audience**

Imitating the tone of your target audience can enormously increase a website's conversion rates.

- **Evoke emotions with visuals**

Powerful way to connect with the target audience is by including photos and videos that evoke trust across your landing pages.

- **Useful and creative content**

Most users will come to your website with one goal – to find useful information. To improve customer retention for your business, you need to ensure that only high-quality content ends up on your website.

- **Make it easy to navigate**

Before starting a user experience design process, it is crucial to put yourself in the website visitor's shoes.

You need to ensure that the navigation menu is simple to understand and placed exactly where users expect it to be.

- **Horizontal navigation bar**

Probably the most popular navigation menu type, that is usually located at the top of the page, just below the header. The horizontal navigation bar goes in the left-to-right direction.

- **Vertical sidebar navigation menu**

This type of menu appears on the right or left side and usually stretches from the top to the bottom of the page

- **Dropdown navigation menu**
  A dropdown navigation menu is a common user interface element used in websites and applications to organize and display a list of options in a hierarchical manner.
- **Hamburger navigation menu**

This is a type of navigation menu that appears when you hover your mouse or click over a "hamburger" button. It is the most popular menu type for mobile website design because it occupies the least space until the user clicks on it.

- **Footer navigation menu**

One of the website design elements that you shouldn't overlook when creating a web development checklist is the footer navigation menu. As its name suggests, it appears at the bottom of your web pages.

As more and more people are using smartphones and tablets to browse the internet, your web development team needs to ensure the mobile-friendliness of the website during the design process.

SELF-CHECK 1

Part-I: Choose the correct answer

1. 1. What is CSS?

   a) CSS is a style sheet language

   b) CSS is designed to separate the presentation and content, including layout, colors, and fonts

   c) CSS is the language used to style the HTML documents

   d) All of the mentioned

2. What is the primary role of CSS in a web page?

   a) Managing server requests

   b) Controlling page behavior

   c) Styling and formatting content

   d) Handling user interactions

3. Why is it important to consider user requirements when designing the style of a website?

   a) To increase server performance

   b) To align with the latest design trends

   c) To improve user experience

   d) To reduce development time

4. Which method is effective for gathering user requirements for website style?

   a) Conducting usability testing

   b) Analyzing server logs

   c) Writing complex code

   d) Ignoring user feedback

5. What does responsive design aim to achieve in web development?

a) Fixed layout on all devices

b) Consistent design across browsers

c) Adaptation to different screen sizes

d) Minimizing the use of CSS

Part-II: Answer the following questions accordingly

1) What is the significance of the term "Cascading" in Cascading Style Sheets?
2) Explain the concept of specificity in CSS and how it affects the application of styles.
3) Why is it important to consider user requirements when designing the style of a website?
4) How can color psychology be applied in CSS to evoke specific emotions or reactions from users?
5) How can CSS be developed to match specific user requirements?
6) Discuss the role of user testing in refining CSS to meet user requirements.

Part-II: Say true or false

1. CSS stands for "Central Style Sheet."
2. In CSS, the box model consists of margin, border, padding, and content.
3. User requirements play a minimal role in determining the styling of a website.
4. Understanding user preferences and needs is essential for creating a user-centric design.
5. CSS is static and cannot be adjusted based on user feedback.
6. Iterative development and feedback loops are important in aligning CSS with user requirements.

UNIT TWO: WEB PAGE CREATIONS USING CSS AND HTML

This unit is developed to provide you the necessary information regarding the following content coverage and topics:

- Style elements of a web page
- Page layout creation with CSS
- Positioning document elements with CSS
- Style sheets application on multiple pages on a website
- Web pages creation for varied screen resolutions
- Application layering for the Desired Design

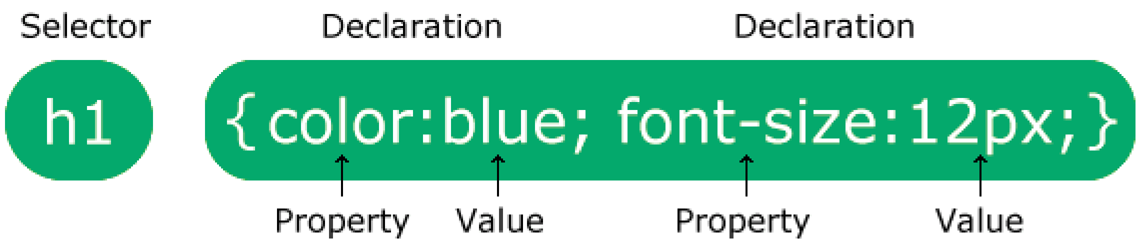
This unit will also assist you to attain the learning outcomes stated in the cover page. Specifically, upon completion of this learning guide, you will be able to:

- Understand the fundamentals of CSS and essential design principles
- Understand how to gather user preferences and requirements for effective styling
- Apply learned principles to develop CSS that aligns with user preferences
- follow principles of layout design using CSS
- apply methods of positioning and arrange document elements
- explain concept of style sheets and their application across multiple pages
- solve challenges of designing for different screen sizes
- Explore techniques for handling browser-specific styling challenges
- Emphasize the importance of adhering to industry standards in CSS
- Address browser-specific differences to enhance website accessibility

2.1. STYLE ELEMENTS OF A WEB PAGE

- **CSS Syntax**

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts –



The selector points to the HTML element you want to style. Example: <h1> or <table> etc.

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.

Property − A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color, border etc.

Value − Values are assigned to properties. For example, color property can have value either red or #F1F1F1 etc.

Fig: 2.1. Some CSS styles

- ➢ **The Type Selectors**
  - ✓ **The Universal Selectors**

Is used to select and apply styles to all elements on a web page.

```
* { color: #000000; }
```

- ✓ **The Descendant Selectors**

Suppose you want to apply a style rule to a particular element only when it lies inside a particular element.

```
ul em {color: #000000;}
```

- ✓ **The Class Selectors**

You can define style rules based on the class attribute of the elements. All the elements having that class will be formatted according to the defined rule.

```
.black {color: #000000; }
```

- ✓ **The ID Selectors**

You can define style rules based on the id attribute of the elements. All the elements having that id will be formatted according to the defined rule.

```
#black {color: #000000; }
h1#black { color: #000000; }
```

- ✓ **The Child Selectors(>)**

You have seen the descendant selectors. There is one more type of selector, which is very similar to descendants but have different functionality. Consider the following example −

```
body > p {color: #000000; }
```

- ✓ **Adjacent Sibling Selector (+)**

The adjacent sibling selector is used to select an element that is directly after another specific element.

- ✓ **General Sibling Selector (~)**

The general sibling selector selects all elements that are next siblings of a specified element.

- ✓ **The Attribute Selectors**

You can also apply styles to HTML elements with particular attributes.

```
input[type = "text"] {color: #000000; }
```

- ✓ **Grouping Selectors**

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

- **Multiple Style Rules**

You may need to define multiple style rules for a single element. You can define these rules to combine multiple properties and corresponding values into a single block as defined in the fg example :

```
h1{color: #36C; font-weight: normal; text-transform: lowercase;}
```

- **Add CSS to a Webpage**

There are four ways to associate styles with your HTML document. Most commonly used methods are inline CSS and External CSS.

A. Embedded CSS - The <style> Element

You can put your CSS rules into an HTML document using the <style> element. This tag is placed inside the <head>...</head> tags.

```
<html>
<head>
<style type = "text/css" media = "all">
body {  background-color: linen;}
</style>
</head>
<body>
</body>
</html>
```

B. Inline CSS - The style Attribute

You can use style attribute of any HTML element to define style rules. These rules will be applied to that element only.

```
<element style = "...style rules....">
<html>
```

```
<head>
</head>
<body>
<h1 style = "color:#36C;">
This is inline CSS
</h1>
</body>
</html>
```

C. External CSS - The <link> Element

An external style sheet is a separate text file with .css extension. You define all the Style rules within this text file and then you can include this file in any HTML document using <link> element.

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

D. Imported CSS - @import Rule

@import is used to import an external stylesheet in a manner similar to the <link> element. Here is the generic syntax of @import rule.

```
<head>
@import "URL";
@import url("URL");
</head>
```

- **CSS Rules Overriding**

We have discussed four ways to include style sheet rules in an HTML document. Here is the rule to override any Style Sheet Rule.

Any inline style sheet takes highest priority. So, it will override any rule defined in <style>...</style> tags or rules defined in any external style sheet file.

Any rule defined in <style>...</style> tags will override rules defined in any external style sheet file.

Any rule defined in external style sheet file takes lowest priority, and rules defined in this file will be applied only when above two rules are not applicable.

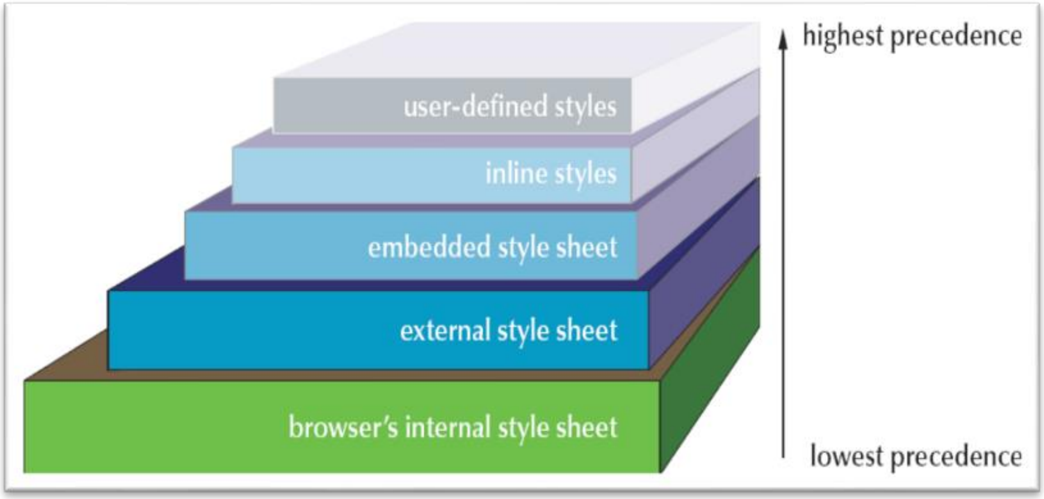


Fig: 2.2. Cascading Order (Precedence)

- **CSS Comments**

Many times, you may need to put additional comments in your stylesheet blocks. So, it is very easy to comment any part in style sheet. You can simple put your comments inside /*.....this is a comment in style sheet.....*/.

```
/* This is a single-line comment */
/* This is a
multi-line
comment */
```

- **CSS – Units**

Values and units, in CSS, are significant as they determine the size, proportions, and positioning

A) Absolute Length Units

These units are categorized as fixed-length units, which means that lengths specified with absolute units maintain an exact, unchanged size on the screen.

Example:

Unit	Description	Equivalent value	Example
px	Pixels	1px = 1/96th of 1in	font-size: 15px;

B) Relative Length Units

Are relative to something else, perhaps the size of the parent element's font, or the size of the viewport.

Table:2.1. Relative Length Units

Unit	Description	Example
rem	Relative to font-size of the root element.	font-size: 2rem;
vh	It is relative to the height of the viewport. 1vh = 1% or 1/100 of the height of the viewport.	font-size: 4vh;
vw	It is relative to the width of the viewport. 1vw = 1% or 1/100 of the width of viewport.	width: 4vw;
vmin	It is relative to the smaller dimension of the viewport. 1vmin = 1% or 1/100 of the viewport's smaller dimension.	width: 4vmin;

- **CSS – Colors**

CSS uses color values to specify a color. Typically, these are used to set a color either for the foreground of an element (i.e. its text) or else for the background of the element. They can also be used to affect the color of borders and other decorative effects.

- **Named Colors**

A CSS named color is the name of a color, such as red, blue, black, or light-green.

- **RGB Hexadecimal Colors**

A hexadecimal is a 6 digit representation of a color. The first two digits (RR) represent a red value, the next two are a green value (GG), and the last are the blue value (BB).

Each hexadecimal code will be preceded by a pound or hash sign '#'. Following are the examples of hexadecimal notation.

border: 1px solid #ff0099;

- **RGB Values**

This color value is specified using the **rgb( )** property.

It takes three values, one each for red, green, and blue.

The value can be an integer between 0 and 255 or a percentage.

Color:RGB(20,50,255)

- **HSL Values**

This color value is specified using the hsl() function.

HSL stands for hue, saturation and lightness.

Hue is represented in degrees (0-360), saturation and lightness are represented as percentages (0% - 100%).

Color:hsl(0,0%,50%)

- **currentcolor keyword**

The currentcolor keyword takes the value of the **color** property of an element. It can be passed to any other styling property using the keyword currentcolor.

div{ color: red; border: 5px solid currentcolor; }

- **CSS – Backgrounds**

This section teaches you how to set backgrounds of various HTML elements. You can set the following background properties of an element

The **background-color** property is used to set the background color of an element.

The **background-image** property is used to set the background image of an element.

The **background-repeat** property is used to control the repetition of an image in the background.

The **background-position** property is used to control the position of an image in the background.

The **background-attachment** property is used to control the scrolling of an image in the background.

The **background** property is used as a shorthand to specify a number of other background properties.

- **CSS – Fonts**

The **font-family** property is used to change the face of a font.

The **font-style** property is used to make a font italic or oblique.

The **font-variant** property is used to create a small-caps effect.

The **font-weight** property is used to increase or decrease how bold or light a font appears.

The **font-size** property is used to increase or decrease the size of a font.

The **font** property is used as shorthand to specify a number of other font properties.

- **HTML Forms**

An HTML form is used to collect user input. The user input is most often sent to a server for processingThe

<form> Element

The HTML <form> element is used to create an HTML form for user input:

<form>

/*

form elements

*/

</form>

The <form> element is a container for different types of input elements, such as: text fields, checkboxes, radio buttons, submit buttons, etc.

The <input> Element

The HTML <input> element is the most used form element.

An <input> element can be displayed in many ways, depending on the type attribute.

examples:

Type	Description
<input type="text">	Displays a single-line text input field
<input type="radio">	Displays a radio button (for selecting one of many choices)
<input type="checkbox">	Displays a checkbox (for selecting zero or more of many choices)
<input type="submit">	Displays a submit button (for submitting the form)
<input type="button">	Displays a clickable button

- **Text Fields**

The <input type="text"> defines a single-line input field for text input.

Example

A form with input fields for text:

```
<html>

<body>

<h2>Text input fields</h2>
```

```
   <form>

<label for="fname">First name:</label><br>

 <input type="text" id="fname" name="fname" value="John"><br>

 <label for="lname">Last name:</label><br>

 <input type="text" id="lname" name="lname" value="Doe">

</form>
```

```


</body>
```

```
</html>
```

- **The <label> Element**

Notice the use of the <label> element in the example above.

The <label> tag defines a label for many form elements.

The <label> element is useful for screen-reader users, because the screen-reader will read out loud the label when the user focuses on the input element.

- **Radio Buttons**

The <input type="radio"> defines a radio button.

Radio buttons let a user select ONE of a limited number of choices.

Example

A form with radio buttons:

```
<!DOCTYPE html>
<html>
<body>
<h2>Radio Buttons</h2>
<p>Choose your favorite Web language:</p>
<form>
  <input type="radio" id="html" name="fav_language" value="HTML">
  <label for="html">HTML</label><br>
  <input type="radio" id="css" name="fav_language" value="CSS">
  <label for="css">CSS</label><br>
  <input type="radio" id="javascript" name="fav_language" value="JavaScript">
  <label for="javascript">JavaScript</label>
</form>
</body>
</html>
```

- **Checkboxes**

  The <input type="checkbox"> defines a **checkbox**.

  Checkboxes let a user select ZERO or MORE options of a limited number of choices.

  Example

  A form with checkboxes:

```
<!DOCTYPE html>
```

```
<html>
<body>
<h2>Checkboxes</h2>
<p>The <strong>input type="checkbox"</strong> defines a checkbox:</p>
<form action="/action_page.php">
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
  <label for="vehicle1"> I have a bike</label><br>
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
  <label for="vehicle2"> I have a car</label><br>
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
  <label for="vehicle3"> I have a boat</label><br><br>
  <input type="submit" value="Submit">
</form>
</body>
</html>
```

- **Submit Button**

The <input type="submit"> defines a button for submitting the form data to a form-handler.

The form-handler is typically a file on the server with a script for processing input data.

The form-handler is specified in the form's action attribute.

Example

A form with a submit button:

```
<!DOCTYPE html>
<html>
<body>
<h2>HTML Forms</h2>
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
<p>If you click the "Submit" button, the form-data will be sent to a page called "/action_page.php".</p>
</body>
</html>
```

- **HTML Div Element**

The <div> element is used as a container for other HTML elements.

The <div> element is by default a block element, meaning that it takes all available width, and comes with line breaks before and after.

Example

A <div> element takes up all available width:

```
<!DOCTYPE html>
<html>
<style>
div {  background-color: #FFF4A3;}
</style>
<body>
<h1>HTML DIV Example</h1>
Lorem Ipsum <div>I am a div</div> dolor sitamet.
<p>The yellow background is added to demonstrate the footprint of the DIV element.</p>
</body>
</html>
Example2
<!DOCTYPE html>
<html>
<style>
div {  background-color: #FFF4A3;}
</style>
<body>
```

```
<h1>HTML DIV Example</h1>
<div>
  <h2>London</h2>
  <p>London is the capital city of England.</p>
  <p>London has over 13 million inhabitants.</p>
</div>
<p>The yellow background is added to demonstrate the footprint of the DIV element.</p>
</body>
</html>
```

- ✓ **Center align a <div> element**

  If you have a <div> element that is not 100% wide, and you want to center-align it, set the CSS margin property to auto.

  ```
  <style>
  div {width:300px; margin:auto;}
  </style>
  ```

- ✓ **Multiple <div> elements**

  You can have many <div> containers on the same page.

- ✓ **Aligning <div> elements side by side**

  When building web pages, you often want to have two or more <div> elements side by side, like this:

  There are different methods for aligning elements side by side, all include some CSS styling. We will look at the most common methods:

  Float: The CSS float property is used for positioning and formatting content and allow elements float next to each other instead of on top of each other.

  Inline-block: If you change the <div> element's display property from block to inline-block, the <div> elements will no longer add a line break before and after, and will be displayed side by side instead of on top of each other.

  Flex: The CSS Flexbox Layout Module was introduced to make it easier to design flexible responsive layout structure without using float or positioning.

  To make the CSS flex method work, surround the <div> elements with another <div> element and give it the status as a flex container.

Grid: The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

Sounds almost the same as flex, but has the ability to define more than one row and position each row individually.

The CSS grid method requires that you surround the <div> elements with another <div> element and give the status as a grid container, and you must specify the width of each column.

- **HTML <fieldset> and <legend>Tag**

The <fieldset> tag is used to group related elements in a form.

The <fieldset> tag draws a box around the related elements.

Example

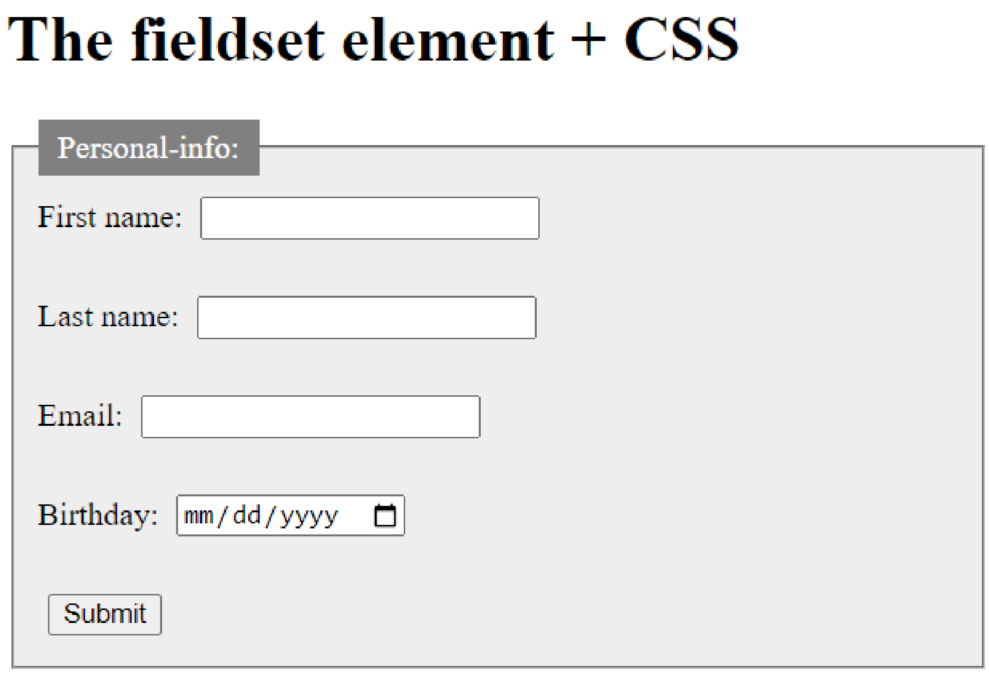
Use CSS to style <fieldset> and <legend>:

```
<!DOCTYPE html>

<html>

<head>

<style>

fieldset { background-color: #eeeeee;}

legend { background-color: gray; color: white;  padding: 5px 10px;}

input { margin: 5px;}

</style>

</head>

<body>
```

```
<h1>The fieldset element + CSS</h1>

<form action="/action_page.php">

 <fieldset>
```

```
  <legend>Personal-info:</legend>

  <label for="fname">First name:</label>

  <input type="text" id="fname" name="fname"><br><br>

  <label for="lname">Last name:</label>

  <input type="text" id="lname" name="lname"><br><br>

  <label for="email">Email:</label>

  <input type="email" id="email" name="email"><br><br>

  <label for="birthday">Birthday:</label>

  <input type="date" id="birthday" name="birthday"><br><br>

  <input type="submit" value="Submit">

 </fieldset>

</form>

</body>

</html>
```

Output:



The fieldset element + CSS

Personal-info:

First name:

Last name:

Email:

Birthday: mm/dd/yyyy

- **CSS – Text**

A text refers to a piece of written or printed information in the form of words or characters that can be read and understood. Texts can include content such as books, articles, emails, messages, web pages, etc.

➢ **Text Color**

Altering the color of the text can add visual interest or align with a specific design scheme.

The CSS **color** property is used to set the color of the text. The possible values for this property are as follows:

➢ **Text Alignment**

The position or placement of text on a page is called its alignment. The text is aligned based on the left and right margins of the page.

✓ **left**: Aligned with the left-margin.
✓ **right**: Aligned with the right-margin.
✓ **center**: Aligned at the center of the page.
✓ **justify**: Aligned with both the margins.

- ✓ **justify-all**: Same as justify, making the last line justified as well.
- ➢ **Text Direction**

Text direction refers to the orientation of text characters within a document or element. It determines whether text should be displayed from left to right (LTR) or right to left (RTL) based on the writing system used.

- ➢ **Text Decoration**

The CSS **text-decoration** property helps in adding extra decoration to the text, such as, adding a line (underline, strikethrough, overline) and color, style and thickness to the line.

- ✓ **text-decoration-line**: Sets the type of decoration line (underline, strikethrough or overline).
- ✓ **text-decoration-color**: Sets the color to the text decoration.
- ✓ **text-decoration-style**: Sets the style to the text decoration (dotted, dashed, solid, wavy, double, etc.)
- ✓ **text-decoration-thickness**: Sets the thickness to the text decoration.
- ➢ **Text Emphasis**

CSS provides a property to apply emphasis marks on a block of text using the property **text-emphasis**. These marks are typically used to highlight specific content or to indicate pronunciation or stress in certain languages.

It allows the designer to apply emphasis to individual characters or phrase within a block of text.

It is a shorthand for **text-emphasis-style** and **text-emphasis-color**.

- ➢ **Text Indentation**

Indentation is the space between the margin and the first line of text. Proper indentation enhances the readability and clarity of text on a page.

CSS also provides a property to set the text indentation and that is **text-indent**. The following values can be passed to this property:

- ✓ **length**: Any specific length {pixels (px), points (pt), ems (em), etc}. Default value is 0.

- ✓ **percentage (%)**: The value passed is in relation to the percentage of the width of the parent element.
- ✓ **each-line**: Affects the first line of a block of text along with each line after a forced line break.
- ✓ **hanging**: Indentation is inverted, except for the first line.
- ✓ **initial**: Sets the text-indent to its default value.
- ✓ **inherit**: Allows to inherit the text-indent value from its parent element.

➢ **Text Letter Spacing**

The CSS property **letter-spacing** is used to adjust the space between the letters of a text.

Following are the possible values that this property can have:

- ✓ **normal**: Default value and represents the normal amount of space between letters.
- ✓ **length**: Any specific length {pixels (px), points (pt), ems (em), or percentages (%)}.
- ✓ **initial**: Sets the letter-spacing to its default value.
- ✓ **inherit**: Allows to inherit the letter-spacing value from its parent element.

➢ **Text Word Spacing**

CSS provides property to adjust the spacing between the words in a piece of text, just like letter spacing. The property to adjust the space between words is **word-spacing**.

This property can take any of the following values:

- ✓ **normal**: Default value and represents the normal amount of space between words.
- ✓ **length**: Any specific length {pixels (px), points (pt), ems (em), or percentages (%)}.
- ✓ **initial**: Sets the word-spacing to its default value.
- ✓ **inherit**: Allows to inherit the word-spacing value from its parent element.

➢ **Text Shadow**

The text-shadow property is used to add a shadow effect to text. It allows you to specify the color, offset, blur-radius, and spread-radius of the shadow. Following is the syntax of text-shadow:

/* offset-x | offset-y | blur-radius | color */

text-shadow: 1px 1px 2px black;

- **Text Line Break**

CSS provides property **line-break** that is useful in determining how to break lines in a block of text.

- **Text Word Break**

The CSS **word-break** property in CSS is used to specify how words should be broken or wrapped in case they exceed the available width of an element.

- **Images**

Images play an important role in any webpage. Though it is not recommended to include a lot of images, but it is still important to use good images wherever required.

CSS plays a good role to control image display. You can set the following image properties using CSS.

The **border** property is used to set the width of an image border.

The **height** property is used to set the height of an image.

The **width** property is used to set the width of an image.

The **-moz-opacity** property is used to set the opacity of an image.

- **Links**

In web development, a "link" generally refers to an HTML element that allows the user to navigate from one web page to another or to another resource, such as a style sheet, an image, or a script. The HTML **<a>** (anchor) element is the most common way to create links.

The **:link** signifies unvisited hyperlinks.

The **:visited** signifies visited hyperlinks.

The **:hover** signifies an element that currently has the user's mouse pointer hovering over it.

The **:active** signifies an element on which the user is currently clicking.

Usually, all these properties are kept in the header part of the HTML document.

Remember a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective. Also, a:active MUST come after a:hover

- **Tables**

A table is an HTML element used to display data in a structured format with rows and columns. It is created using the <table> tag in HTML and can be styled using CSS properties.

Following is a simple HTML table which has been styled using CSS :

```
<html>
<head>
<style>
table {font-family: Arial, Helvetica, sans-serif;border-collapse: collapse; width: 100%;}
td, th {border: 1px solid #ddd; padding: 8px;}
tr:nth-child(even){background-color: #f2f2f2;}
tr:hover {background-color: #ddd;}
th {padding-top: 12px; padding-bottom: 12px;text-align: left; background-color: #40a944; color: white;
</style>
</head>
<body>
<h1>CSS Styled Table</h1>
<table>
<tr>
```

```
<th>Name</th>
<th>Address</th>
<th>Country</th>
</tr>
<tr>
<td>Nuha Ali</td>
<td>My Home Bhooja</td>
<td>India</td>
</tr>
<tr>
<td>Zara Ali</td>
<td>Students Roosters</td>
<td>England</td>
</tr>
</table>
</body>
</html>
```

- **Properties of a Table**

You can set the following CSS properties of a table

The **border-collapse** specifies whether the browser should control the appearance of the adjacent borders that touch each other or whether each cell should maintain its style.

The **border-spacing** specifies the width that should appear between table cells.

The **caption-side** specifies where the table caption should be displayed (top or bottom).

The **empty-cells** specifies whether the border should be shown if a cell is empty.

The **table-layout** allows browsers to speed up layout of a table by using the first width properties it comes across for the rest of a column rather than having to load the whole table before rendering it.

The **width** and **height** properties set the height and width of a table.

The **text-align** property sets the horizontal alignment of the text content within table cells.

The **border** property can be used to set border to table and its cells.

The **background-color** property can be applied to the table itself, the table cells, or table rows.

The **font-size**, **font-family**, **font-weight**, **font-color** etc style the table font.

Collapse Table Border using CSS

The property **border-collapse** ensures that borders between table cells collapse into a single border, creating a cleaner look. Property **border-collapse** can have values **collapse** or **separate (default)**.

- **Borders**

The border of an HTML element is simply one or more lines that surround the content and padding of an element. Every border has three aspects: its width, or thickness; its style, or appearance; and its color.

The CSS **border** properties allow you to specify how the border of the box representing an element should look. There are three properties of a border

border-style - Specifies whether a border should be solid, dashed line, double line, or one of the other possible values.

border-color - Specifies the color of a border. The default value is the foreground color of the element and if element is blank, then color of the parent element

border-width - Specifies the width of a border. The default value is medium.

Example:

<p style="border-style: none;">No border.</p>

<p style="border-style: hidden;">Hidden border.</p>

<p style="border-style: dotted;">Dotted border.</p>

- **Rounded Borders**

We can use **border-radius** property to add rounded borders to an element.

Syntax

P{ border: 2px solid #40a944; border-radius: 5px;}

- **Margins**

The **margin** property is used to set the margin space around an element. Margins are the space outside the border of an element. The **margin** property can be set for individual sides (top, right, bottom, left) or as a shorthand property for all sides at once.

- **Margins - Single-Side Properties**

CSS provides four separate properties to set margins for left, right, top and bottom for an element.

margin-bottom, margin-top, margin-left and margin-right

Following example demonstrates how we can set different margins around h1 elements:

h1 {margin-top: 20px; margin-right:40px; margin-bottom:10px;margin-left:0px; background-color: #eee;**}**

- **Margins - Set Three Values**

We can set margin at three values as **margin: 20px 40px 10px**, in this case top margin will be 20px, right and left margins will be 40px and bottom margin will be 10px. Following is the example. You should try to compare the output with previous example:

h1 {margin: 20px 40px 10px; background-color: #eee;}

- **Margins - Set Two Values:**

We can set margin at two values as **margin: 20px 40px**, in this case top and bottom margins will be 20px, right and left margins will be 40px.

```
h1 {margin: 20px 40px; background-color: #eee;}
```

We have seen that setting a single value for margin, applies margin equally to all the sides - top, right, bottom and left. You can check very first example to understand this case.

- **Margins - Mix up Units**

CSS allows to mix up the types of length value you use while specifying different margins in a shorthand property.

- **CSS Negative Margins**

CSS allows to specify negative margins for an element. This will cause the element's box to stick out of its parent or to overlap other elements.

- **Margins - Percentages**

It is very much possible to set percentage values for the margins of an element. Percentage margins values are computed in relation to the width of the parent element's content area, so they can change if the parent element's width changes.

- **Margins - Inline Elements**
  Margins can also be applied to inline elements but top and bottom margins do not have any effect on the line height of these nonreplaced elements and these margins are always transparent.
- **Margins - Auto Value**

In order to center an element inside its parent, use **margin: 0 auto** as shown in the following example:

```
<style>

div {width:600px; border:1px dotted;}

h1 {margin:0 auto;}

</style>
```

- **Lists**

Lists are useful as they present the information in a structured and organized manner. Lists improve the readability and comprehension of content on a web page. So, if the content is listed, it is easy to follow.

Lists are commonly used to display items, steps, options, or any other type of related information that should be presented sequentially or in a group.

➢ **Ordered List**

Ordered lists are used when the items need to be presented in a specific order marked with numbers or letters. Following is the syntax to create HTML ordered lists:

```
<ol>

<li>First</li>

<li>Second</li>

<li>Third</li>

</ol>
```

➢ **Unordered List**

Unordered lists are used when the items need to be presented in a specific order marked with simple bullets.

✓ **CSS Lists - Properties**

CSS provides a set of properties that can be applied to any list, which are as follows:

The **list-style-type** allows you to control the shape or appearance of the list markers (bullet points).

The **list-style-position** allows to specify the position of the list-item markers.

The **list-style-image** specifies an image for the marker rather than a bullet point or number.

The **list-style** serves as shorthand property for to manage the markers.

CSS Lists - Marker Types

CSS allows you to control the shape of the list markers (bullet points). The following example shows how CSS property **list-style-type** sets different markers for the ordered and unordered lists. The same property can be used for both ordered and unordered lists.

You can check CSS property **list-style-type** detail to check all the possible marker's types for the lists.

- **CSS Lists - Image Marker**

You might prefer to use an image as an item list marker. The CSS **list-style-image** property can be used to to specify an image as an item list marker.

You can use your own bullet style. If no image is found, then default bullets are printed. Following is an example to show the usage of **list-style-image** property.

```
ul {list-style-image: url('/images/icon-bullet.png');}
```

- **CSS Lists - Marker Position**

The CSS **list-style-position** property indicates whether the marker should appear inside or outside of the box containing the bullet points. It can have one of the following values −

none- No marker is displayed, and the list items are without any bullet points or numbers.

Inside- The marker is positioned inside the content flow, meaning it appears within the content box of the list item.

Outside- The default value. The marker is positioned outside the content flow, appearing to the left of the content box of the list item.

Inherit- is used to inherit the **list-style-position** value from its parent element.

Following is an example to show the usage of **list-style-position** property.

```
<style>

ul.a {list-style-position: outside;}

ul.b {list-style-position: inside;}

</style>
```

- **Lists - Controlled Counting**

Some time we might want to count differently on an ordered list — e.g., starting from a number other than 1, or counting backwards, or counting in steps of more than 1.

There are following three CSS attributes which helps in controlling the list numbering.

<start> - allows you to start the list counting from a number other than 1.

<reversed> - starts the list counting reverse or down instead of up.

Example:

```
<html>
<body>
<h2>CSS Lists - Controlled Counting</h2>
<h3>start attribute</h3>
<ol start="4">
<li>Java.</li>
<li>HTML.</li>
<li>CSS.</li>
<li>React.</li>
</ol>
<h3>reverse attribute</h3>
<ol reversed>
<li>Java.</li>
<li>HTML.</li>
<li>CSS.</li>
<li>React.</li>
</ol>
</body>
</html>
```

Lists - Colors

We can make lists look more stylish and colorful using CSS styling as demonstrated in the following example. As we see any styling added to the <ul> or <ol> tag will affect the entire list, whereas the addition to the individual <li> tag will affect only the items of the corresponding list.

Example:

ul li {background: olive; color:white; padding:10px; font-size:20px; margin:10px;}

- **Paddings**

The CSS padding property is used to specify the space between the content of an element and its borders. This article will teach you CSS padding property and its constituent properties.

The value of CSS padding should be either a length, a percentage, or the word inherit. A padding property does not allow negative values. If the value is inherit, it will have the same padding as its parent element.

Example: h2{padding: 5px; background-color: #eee;}

CSS Paddings - Single-side Properties

CSS provides four separate properties to set padding for top, right, bottom, and left for an element. These properties are:

padding-top,padding-right,padding-bottom and padding-left

Following example shows how different padding properties can be set around an h2 element.

h2 {padding-top: 20px; padding-right:40px;padding-bottom:10px; padding-left:0px;

background-color: #eee;

- **Cursor**

The CSS **cursor** property determines the appearance of the mouse cursor when hovering over an element to which this property is applied. This property is only applicable in environments with mouse and cursor functionality. It's main purpose is to improve usability by visually representing certain functions.

- **Possible Cursor Values**

The **cursor** property can have following values:

<url>: (optional) You have the flexibility to use a series of **url()** values separated by commas, with each **url()** pointing to an image file.

<x> <y>: (optional) You have the option to specify **x** and **y** coordinates that define the hotspot of the cursor and specify the exact position within the cursor image that the cursor points to.

<Keyword>:A keyword value is required that specifies either the cursor type to use or the alternate cursor to use if none of the specified symbols can be loaded.

Keyword Based Cursor Values

The following table lists the available keywords.

Keyword	Example	Description
Auto	↖	The displayed cursor is determined by the user agent based on the current context. This is the default property that the browser uses to define the cursor.
default	↖	The default cursor, which varies depending on the platform, is usually displayed as an arrow.
pointer	👆	The displayed cursor is pointer cursor, showing the cursor serves as an indicator pointing to a hyperlink.

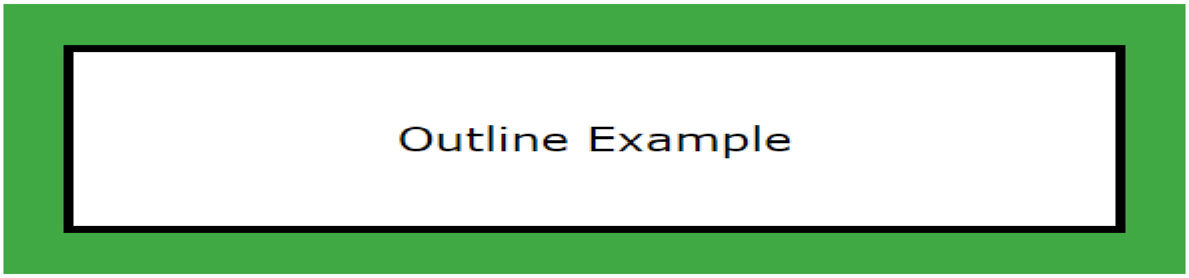
- **Outlines**

CSS defines a special sort of element decoration called an outline which is drawn outside the element's border. CSS outlines are very similar to borders, but there are few major differences as well

An outline does not take up space.

Outlines do not have to be rectangular.

Example: Following is rectangle having a black border of 5px and green outline of 30px.



This tutorial will teach us how to set different properties associated with outlines. CSS alows us to control an outline width, its color, style etc.

- **Outline Color**

The **outline-color** property is used to specify the color of the outline. Its value should either be a color name, a hex color, or an RGB value, as with the color and border-color properties.

Name - Example red, blue or green

HEX - Example #ff0000

RGB - Example rgb(255,0,0)

HSL - Example hsl(0, 100%, 50%)

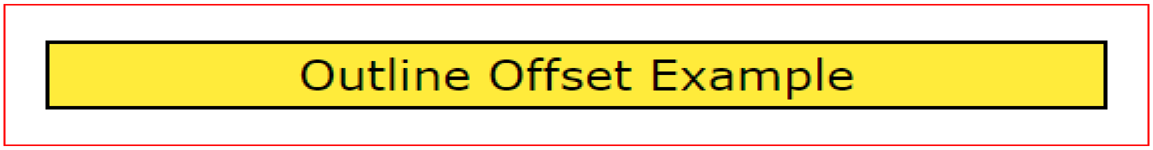
Example:

<p style = "outline-width:thick; outline-style:solid; outline-color:rgb(255,200,0); padding:5px;">Outline color set with RGB code "rgb(255,200,0)".</p>

- **Outline Offset**

The **outline-offset** property is used to specify the space between an outline and the border edge of an element. The space between the outline and the element is transparent.

The following example show an outline 20px outside the border of the element:



Outline Offset Example

The above example shows that the space between an element's border and its outline is transparent.

Example:

<p style = "border:1px solid #000; outline:1px solid red; outline-offset:20px;margin:25px">Outline offset 20px;</p>

- **Outline vs Border**

The two obvious differences are that outlines cannot have a **hidden** style where as a borders can be hidden second an outline can have **auto** style whereas border can't have it. Following table illustrates more differences between outline and border:

Table: 2.2 outline and border comparison

Outline	**Border**
Outline is a non-rectangular shape that surrounds an element, usually with a solid color.	Border is a rectangular shape that is drawn around the content of an element, can be solid, dashed, or dotted, and can have different colors and sizes.
It does not take up any space in the layout and does not affect the size or position of the element.	It affects the size and position of the element, as it adds width to the element.
It is typically used to highlight or emphasize an element, such as when an element is focused or activated.	It can be used for various purposes, such as separating elements, creating boxes, and adding visual emphasis.
It can be created using the outline property in CSS.	It can be created using the border property in CSS.

- **The CSS Box Model**

All HTML elements can be considered as boxes. In CSS, the term "box model" is used when talking about design and layout.

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

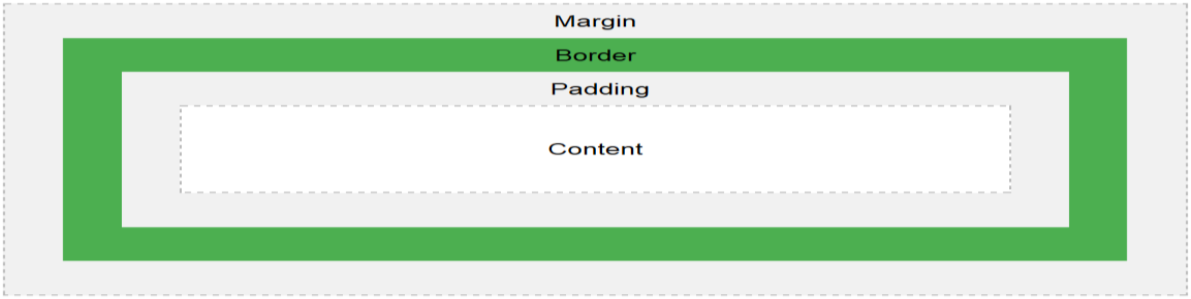


Fig: 2.2. Box model

- **Height & Width (Dimension)**

The dimensions of HTML elements is often specified with CSS width and height properties and we can use these properties to set the dimension of the elements.

CSS also provides properties like max-width, min-width, max-height and min-height to set the maximum/minimum width and height of an element.

The height and width properties allow you to set the height and width of an element. These properties can hold following values:

length: The height and width of an element can be of any unit of length (px, pt, em, in, etc.)

percentage (%): The value can be passed as a percentage value, which is in percent of the containing block.

auto: Browser calculates the height and width of the element. It is the default value.

initial: Sets the value of height and width to its default value.

inherit: The value of height and width is inherited from its parent value.

The height and width properties do not add anything to the layout of the element i.e they do not include padding, margin or borders. They set the height and width of the area inside the padding, border, and margin of the element.

- ➢ **Max-Height**

CSS can limit the maximum height of an element using max-height property. This property allows to specify maximum height of an element. The value of the max-height property can be:

none: No maximum height value is set. This is the default value.

length: Sets the maximum height in terms of length units (px, pt, em, in, etc.)

percentage (%): The value is relative to the percent of containing block.

initial: Sets the value of height and width to its default value.

inherit: The value is inherited from its parent value.

- **min-height**

CSS can limit the minimum height of an element using **min-height** property. This property allows to specify minimum height of an element. It specifies the smallest height that an element can have, ensuring that it will never shrink below that value. The value of the **min-height** property can be:

length: Sets the minimum height in terms of length units (px, pt, em, in, etc.)

percentage (%): The value is relative to the percent of containing block.

initial: Sets the value of height and width to its default value.

inherit: The value is inherited from its parent value.

The minimum height will be applied, when the content is smaller than the minimum height. And when the content is larger than the minimum height, the value of min-height has no effect on the element.

- **max-width**

CSS can limit the maximum width of an element using **max-width** property. This property allows to specify maximum width of an element. The value of the **max-width** property can be:

none: No maximum width value is set. This is the default value.

length: Sets the maximum width in terms of length units (px, pt, em, in, etc.)

percentage (%): The value is relative to the percent of containing block.

initial: Sets the value of height and width to its default value.

inherit: The value is inherited from its parent value.

The max-width value overrides the value of width property. If the content within the element is larger than the specified max-width, it will automatically adjust the height of the element to accommodate the content within the element. If the content within the element is smaller than the specified max-width, the max-width property has no effect.

- **min-width**

CSS can limit the minimum width of an element using min-width property. This property allows to specify minimum width of an element. It specifies the smallest width that an element can have, ensuring that it will never shrink below that value. The value of the min-width property can be:

- **length**: Sets the minimum width in terms of length units (px, pt, em, in, etc.)
- **percentage (%)**: The value is relative to the percent of containing block.
- **initial**: Sets the value of height and width to its default value.
- **inherit**: The value is inherited from its parent value.

If the content with the element is larger than the min-width, the min-width property has no effect but if the content with the element is smaller than the specified min-width, the minimum width will be applied.

- **Scrollbars**

There may be a case when an element's content might be larger than the amount of space allocated to it. For example, given width and height properties do not allow enough room to accommodate the content of the element.

Table: 2.3. Scrol1bar values

Sr.No.	Value & Description
1	**Visible**:Allows the content to overflow the borders of its containing element.
2	**Hidden**:The content of the nested element is simply cut off at the border of the containing element and no scrollbars is visible.
3	**Scroll**: The size of the containing element does not change, but the scrollbars are added to allow the user to scroll to see the content.

- **Display**
  - **Display:Inline**

    An element with a display property set to inline will not start on a new line and it will take up the remaining/available screen width. It just takes up the space such an element would normally take.Because of this, you can't set the width and height of an element that has a display of inline, becuase it does not take up the whole screen width.

    Some elements are inline by default, like <span>, <a>, <i>, and <img>.
  - **display: block**

    An element that has the display property set to block starts on a new line and takes up the available screen width.

    You can specify the width and height properties for such elements. Examples of elements that are at block-level by default are <div>, <section>, <p>, and lots more.

Inline vs block vs inline-block

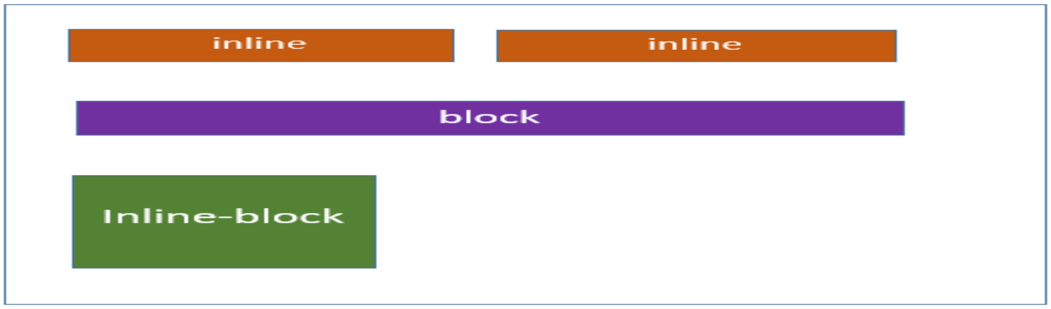
Difference between display: inline, display: block and display: inline-block:

Table: 2.3. Inline Block and inline-block comparison

Inline	block	inline-block
The element is displayed on the same line.	The element is displayed on a new line.	The element is displayed on the same line.
It does not take up the full width of the container.	It takes up the full width of the container.	It does not take up the full width of the container.
It does not have a margin or	It has a margin and	It has a margin and padding

padding by default.	padding by default.	by default.

Following diagram shows the different layout behavior of **inline**, **block**, and **inline-block** elements:



- ✓ **Navigation Links Using Inline Block**

The inline-block property is used to create horizontal navigation menus or lists, where each navigation item is displayed as a block-level element, but remains inline with other items.

- ✓ **Images And Text Using Inline Block**

The inline-block property causes the image and span to be displayed on the same line, allowing them to be aligned horizontally within the block.

- **Visibility**

The CSS visibility property allows you to show or hide an element without changing the layout of a document, while hidden elements take up space.

The **visibility** property can take any of the following values:

Visible − The element is visible.

hidden − The element is hidden, but it still takes up space on the page.

collapse − Remove table rows, columns, column groups, and row groups from a table. **collapse** has the same meaning as hidden if it is used on non-table elements.

initial − Sets the visibility of an element to its default value.

inherit − Inherits the visibility property from its parent element

- **Overflow**

The overflow property determines how content which overflows its element's content area should be handled.

Possible Values

visible − Overflowing content should be displayed.

hidden − Overflowing content should not be displayed.

scroll − Overflowing content should not be displayed, but the user agent should provide some means of accessing the hidden content (e.g., a set of scrollbars).

auto − The behavior caused by this value is dependent on the browser.

- **Resize Elements**

CSS resize is a property that allows users to adjust the size of an element, either vertically, horizontally, both, or none, based on the specified value.

Resize property adds a handle at the bottom-right corner of an element on a webpage. This handle allows users to click and drag to change the size of an elements, making it larger or smaller according to their preference.

none − No user-controllable method for resizing an element is possible. This is default value.

vertical − User can resize an element in the vertical direction.

horizontal − User can resize an element in the horizontal direction.

both − User can resize an element both horizontally and vertically.

block − User can resize an element in the block direction (either horizontally or vertically, depending on the writing-mode and direction value).

inline − User can resize an element in the inline direction (either horizontally or vertically, depending on the writing-mode and direction value).

- **Quotes**

The CSS quotes property sets how the browser should render quotation marks that are automatically added to the HTML <q> element

- **Hover**

The CSS: hover pseudo-class is used to target an element when the user hovers over it with the mouse cursor.

CSS Syntax

:hover{…}

- **CSS Hover - Link Color Change**

Following is an example of changing the background color of a link when mouse cursor comes over it.

- **Hover - Background Color Change**

Here is an example of changing the background color of a button when mouse cursor comes over it:

- **Hover - Change Cursor Shape**

Here is an example of changing the cursor shape when mouse cursor comes over the button:

- **Hover - Change Border Shape**

Here is an example, where border of the link is changing on hover:

- **CSS - Opacity**

The CSS **opacity** property controls the transparency of an element. Opacity determines how much of a hidden element's content is visible.

number – It represented as a decimal number from 0 to 1.

percentage – It represented as a percentage from 0% to 100%.

Following table showing the different opacity values:

Table: 2.4. Opacity value

Value	Description
0	The element is fully transparent and not visible.
0.5	The element is half transparent.
1	The element is fully opaque and visible.

- **Navigation bar**

A navigation bar is a section of a graphical user interface (GUI) that helps users navigate through a website, app, or other software. It is essential for users to quickly and easily navigate to the content they are looking for.

The navigation bar can be horizontal or vertical, that contains links to important pages or features. Navbars can also contain other elements, such as the logo of the website or app, search bar, or social media icons. Navbars can be styled using CSS to change their appearance.

- **CSS Horizontal Navbar**

Following example shows a horizontal navigation bar, which is the most common type of navigation bar displayed across the top of a web page as shown below

```
<html>
<head>
<style>
ul {
background-color: #28bf17;
overflow: hidden;
list-style-type: none;
margin: 0;
padding: 0;
}
li {
float: left;
}
li a {
display: block;color: #f2f2f2;text-align: center;padding: 10px;text-decoration: none;font-size: 17px;}
li a:hover {background-color: #dd9ede;color: black;}
.active-link {background-color: #f53319;color: white;}
</style>
</head>
<body>
<ul>
<li><a href="#" class="active-link">Tutorialspoint</a></li>
<li><a href="#">Home</a></li>
<li><a href="#">Articles</a></li>
<li><a href="#">Courses</a></li>
<li><a href="#">eBook</a></li>
</ul>
<h1>Welcome to TutorialsPoint</h1>
<h3>Simple Easy Learning at your fingertips</h3>
</body>
</html>
```

- **CSS Vertical Navbar**

A vertical navigation bar is also known as a sidebar menu. It is typically positioned on the left or right side of the screen.

Example

```
<html>
<head>
<style>
ul {background-color: #28bf17;list-style-type: none;margin: 0;padding: 0;width: 200px;}
li {text-align: center;}
li a {display: block;color: #f2f2f2;text-align: center;padding: 10px;text-decoration: none;font-size: 17px;}
li a:hover {background-color: #dd9ede;color: black;}
.active-link {background-color: #f53319;color: white;}
</style>
</head>
<body>
<ul>
<li><a href="#" class="active-link">Tutorialspoint</a></li>
<li><a href="#">Home</a></li>
<li><a href="#">Articles</a></li>
<li><a href="#">Courses</a></li>
<li><a href="#">eBook</a></li>
</ul>
</body>
</html>
```

- **CSS Dropdown Navbar**

A dropdown navbar is a navigation bar with a drop-down menu that appears when a user hovers over a link. Dropdown menus are a way to show a list of related items in a small space.

Example

```
<html>
<head>
<style>
.navbar {background-color: #28bf17;overflow: hidden;}
.navbar a {display: block;float: left;color: #f2f2f2;text-align: center;padding: 10px;
text-decoration: none;font-size: 15px;}
.navbar a:hover {background-color: #dd9ede;color: black;}
.active-link {background-color: #f53319;color: white;}
.dropdown {float: left;}
.dropdown .dropbtn {border: none;color: #f2f2f2;padding: 10px;background-color:
#28bf17;}
.dropdown-content {display: none;position: absolute;background-color: #c7385a;
min-width: 120px;}
.dropdown-content a {float: none;color: #f2f2f2;padding: 10px;display: block;
text-align: left;}
.dropdown-content a:hover {background-color: #dd9ede;color: black;}
.dropdown:hover .dropdown-content {display: block;}
</style>
</head>
<body>
<nav class="navbar">
<a href="#" class="active-link">Tutorialspoint</a>
<a href="#">Home</a>
<div class="dropdown">
<button class="dropbtn">Articles</button>
<div class="dropdown-content">
<a href="#">HTML</a>
<a href="#">CSS</a>
```

```
<a href="#">Bootstrap</a>
</div>
</div>
<a href="#">Courses</a>
<a href="#">eBook</a>
</nav>
<h1>Welcome to TutorialsPoint</h1>
<h3>Simple Easy Learning at your fingertips</h3>
</body>
</html>
```

- **CSS Fixed Navbar**

A fixed navbar is a navigation bar that sticks to the top of the screen when the user scrolls down the page. To make a navbar fixed, you can use the **position: fixed** property.

Fixed Vertical Navbar

The following example demonstrates how the **position: fixed** property can be used to create a fixed vertical navbar, which ensures that the navbar stays on the left side of the screen, even when the user scrolls down the page.

```
<html>
<head>
<style>
ul {position: fixed;background-color: #28bf17;list-style-type: none;margin: 0;
padding: 0;width: 200px;height: 100%;}
li {text-align: center;border-bottom: 2px solid #f013c8;}
li a {display: block;color: #f2f2f2;text-align: center;padding: 10px;
text-decoration: none;font-size: 17px;}
li a:hover {background-color: #dd9ede;color: black;}
```

```
.active-link {background-color: #f53319;color: white;}
.heading {padding-top: 170px;text-align: center;background-color: #e6e451;
padding-bottom: 300px}
</style>
</head>
<body>
<ul>
<li><a href="#" class="active-link">Tutorialspoint</a></li>
<li><a href="#">Home</a></li>
<li><a href="#">Articles</a></li>
<li><a href="#">Courses</a></li>
<li><a href="#">eBook</a></li>
</ul>
<div  class="heading">
<h1>Welcome to TutorialsPoint</h1>
<h2>Tutorialspoint CSS Fixed Vertical Navbar</h2>
<h3>Simple Easy Learning at your fingertips</h3>
</div>
</body>
</html>
```

- **Image Gallery**

CSS image gallery is a collection of images that is displayed using CSS. CSS can be used to control the layout of the images, their size, spacing, and other visual properties.

CSS image galleries are commonly used on websites to display products, portfolios, or other visual content in a visually appealing way.

The following example shows a simple image gallery layout that is displayed in a row −

```
<html>
```

```
<head>
<style>
.image-gallery {display: flex;flex-flow: row wrap;justify-content: space-between;
align-items: center;}
.image-gallery img {width: 25%;height: 250px;}
</style>
</head>
<body>
<div class="image-gallery">
<img src="images/red-flower.jpg" alt="Red Flower">
<img src="images/white-flower.jpg" alt="White Flower">
<img src="images/orange-flower.jpg" alt="Orange Flower">
</div>
</body>
</html>
```

- **Multi Background**

CSS Multi background property is used to add one or more images at a time without HTML code, We can add images as per our requirement. A sample syntax of multi background images is as follows.

```
#multibackground {
background-image: url(/css/images/logo.png), url(/css/images/border.png);
background-position: left top, left top;
background-repeat: no-repeat, repeat;
padding: 75px;
}
```

Table: 2.5. Background property values

Sr.No.	Value & Description
1	**Background**:Used to setting all the background image properties in one section
2	**background-clip**:Used to declare the painting area of the background
3	**background-image**:Used to specify the background image
4	**background-origin**:Used to specify position of the background images
5	**background-size**:Used to specify size of the background images

Example:

```
<html>
<head>
<style>
#multibackground {background-image: url(/css/images/logo.png),
url(/css/images/border.png);
background-position: left top, left top;
background-repeat: no-repeat, repeat;
padding: 75px;
}
</style>
</head>
<body>
<div id = "multibackground">
<h1>www.tmptc.com</h1>
<p>
```

```
Entoto Polytechnic College (EPTC) the former Taferi Mekonen School was
established in 1917 E.C as a primary school and worked as complementary school.
The college has today’s name and task starting from 1994 E.C. The college
provides vocational training in 11 occupational sectors namely: fine arts and
aesthetic, automotive, manufacturing, construction, electrical electronics,
information and communication technology, textile leather and garment,
woodwork, business, surveying, drafting, and urban and land, business, and urban
agriculture.        </p>
</div>
</body>
</html>
```

- **Size of Multi background**

Multi background property is accepted to add different sizes for different images.A sample syntax is as shown below −

```
#multibackground {
background:     url(/css/imalges/logo.png)     left     top     no-repeat,
url(/css/images/boarder.png) right bottom no-repeat, url(/css/images/css.gif) left
top repeat;
background-size: 50px, 130px, auto;
}
```

As shown above an example, each image is having specific sizes as 50px, 130px and auto size.

- **Gradients**

What is Gradients?

Gradients displays the combination of two or more colors.

Types of gradients

Linear Gradients(down/up/left/right/diagonally)

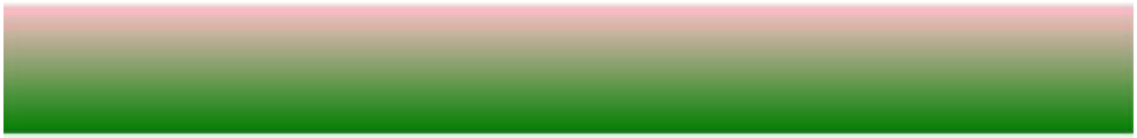
Radial Gradients

Linear gradients are used to arrange two or more colors in linear formats like top to bottom.

Top to bottom

```
<html>
<head>
<style>
#grad1 {
height: 100px;
background: -webkit-linear-gradient(pink,green);
background: -o-linear-gradient(pink,green);
background: -moz-linear-gradient(pink,green);
background: linear-gradient(pink,green);
}
</style>
</head>
<body>
<div id = "grad1"></div>
</body>
</html>
```

Output:



Left to right-example

background: linear-gradient(to right, red , blue);

Diagonal:example

Diagonal starts at top left and right button.

background: -webkit-linear-gradient(left top, red , blue);

background: -o-linear-gradient(bottom right, red, blue);

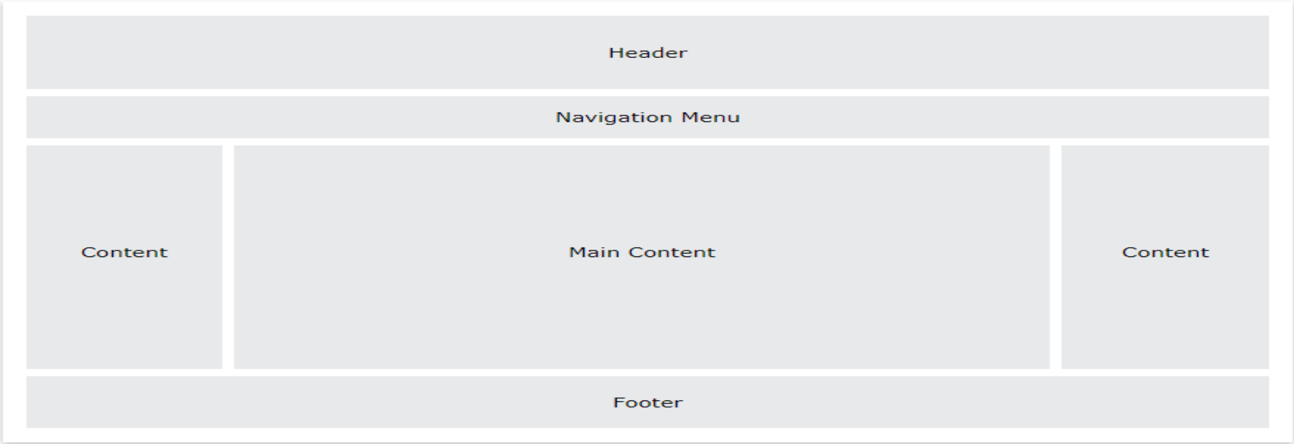
Multi color

background: -webkit-linear-gradient(red, orange, yellow, red, blue, green,pink);

2.2. PAGE LAYOUT CREATION WITH CSS

2.2.1. CSS Website Layout

A website is often divided into headers, menus, content and footer:



There are tons of different layout designs to choose from. However, the structure above, is one of the most common, and we will take a closer look at it in this tutorial.

- **Header**

A header is usually located at the top of the website (or right below a top navigation menu). It often contains a logo or the website name:

- **Navigation Bar**

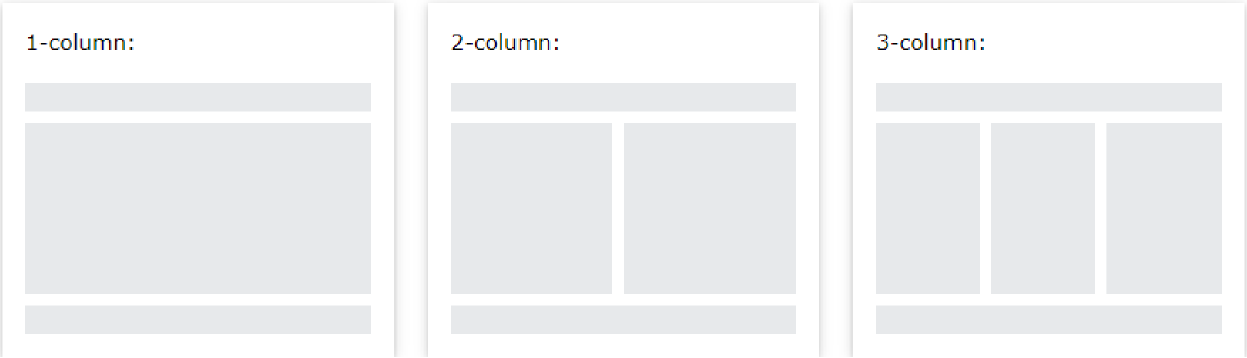
A navigation bar contains a list of links to help visitors navigating through your website:

The layout in this section, often depends on the target users. The most common layout is one (or combining them) of the following:

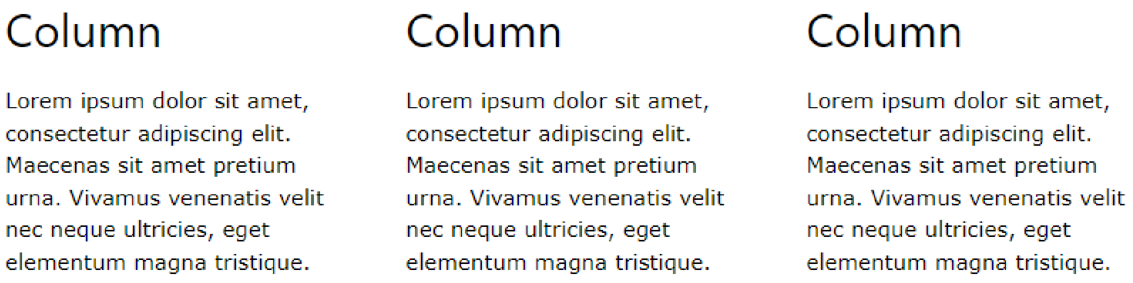
1-column (often used for mobile browsers)

2-column (often used for tablets and laptops)

3-column layout (only used for desktops)



We will create a 3-column layout, and change it to a 1-column layout on smaller screens:



Column

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique.

Column

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique.

Column

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas sit amet pretium urna. Vivamus venenatis velit nec neque ultricies, eget elementum magna tristique.

- **Unequal Columns**

The main content is the biggest and the most important part of your site.

It is common with **unequal** column widths, so that most of the space is reserved for the main content. The side content (if any) is often used as an alternative navigation or to specify information relevant to the main content.

- **Footer**

The footer is placed at the bottom of your page. It often contains information like copyright and contact info:

2.3. POSITIONING DOCUMENT ELEMENTS WITH CSS

I. 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:

Example

```
div. Static {
   position: static;
   border: 3px solid #73AD21;
}
```

II. Relative Positioning

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.

Move Left - Use a negative value for left.

Move Right - Use a positive value for left.

Move Up - Use a negative value for top.

Move Down - Use a positive value for top.

NOTE − You can use bottom or right values as well in the same way as top and left.

Example

```
<html>
<head>
</head>
<body>
<div style = "position:relative; left:80px; top:2px; background-color:yellow;">
This div has relative positioning.
</div>
</body>
</html>
```

III. Absolute Positioning

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.

Move Left - Use a negative value for left.

Move Right - Use a positive value for left.

Move Up - Use a negative value for top.

Move Down - Use a positive value for top.

NOTE − You can use bottom or right values as well in the same way as top and left.

Example

```
<html>
```

```
<head>
</head>
<body>
<div style = "position:absolute; left:80px; top:20px; background-color:yellow;">
This div has absolute positioning.
</div>
</body>
</html>
```

IV. Fixed Positioning

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.

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

Move Left - Use a negative value for left.

Move Right - Use a positive value for left.

Move Up - Use a negative value for top.

Move Down - Use a positive value for top.

NOTE − you can use bottom or right values as well in the same way as top and left.

Example

```
<html>
<head>
</head>
<body>
<div style = "position:fixed; left:80px; top:20px; background-color:yellow;">
```

```
This div has fixed positioning.
</div>
</body>
</html>
```

2.4. Style sheets application on multiple pages on a website

The <link> element can be used to include an external stylesheet file in your HTML document.

An external style sheet is a separate text file with **.css** extension. You define all the Style rules within this text file and then you can include this file in any HTML document using <link> element.

Here is the generic syntax of including external CSS file

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

Imported CSS - @import Rule

@import is used to import an external stylesheet in a manner similar to the <link> element. Here is the generic syntax of @import rule.

```
<head>
@import "URL";
</head>
```

Here URL is the URL of the style sheet file having style rules. You can use another syntax as well –

```
<head>
@import url("URL");
</head>
```

Example

Following is the example showing you how to import a style sheet file into HTML document

```
<head>
@import "mystyle.css";
</head>
```

2.5. WEB PAGES CREATION FOR VARIED SCREEN RESOLUTIONS

2.5.1. Responsive Web Design

Responsive web design provides an optimal experience, easy reading and easy navigation with a minimum of resizing on different devices such as desktops, mobiles and tabs).

Below image shows the responsive structure of web pages.

Fig: 2.3 Responsive Web Design

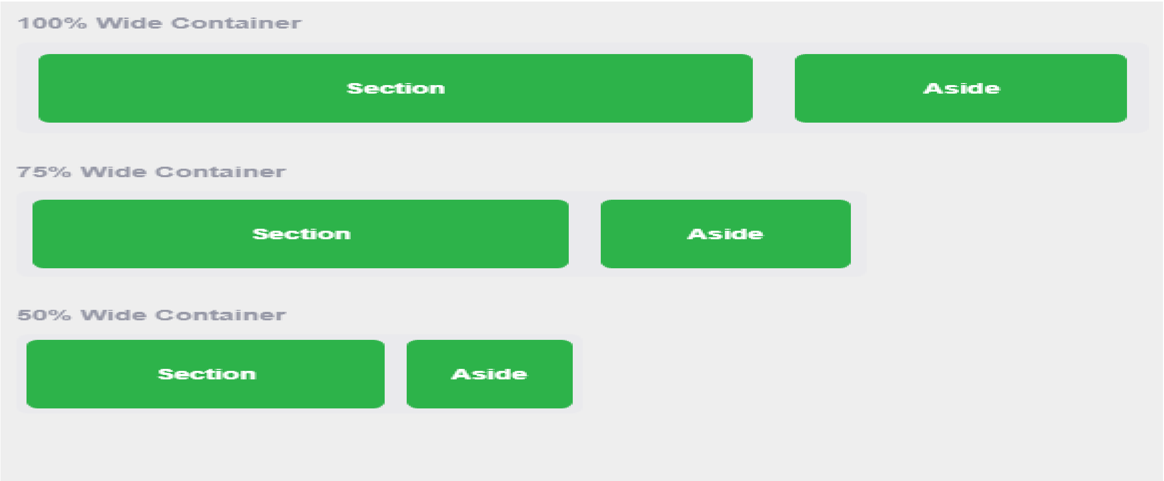
Flexible Grid

```
<html>
<head>
<style>
body {
font: 600 14px/24px "Open Sans",
"HelveticaNeue-Light",
"Helvetica Neue Light",
"Helvetica Neue",
Helvetica, Arial,
"Lucida Grande",
Sans-Serif;
}
h1 {
color: #9799a7;
font-size: 14px;
font-weight: bold;
margin-bottom: 6px;
}
.container:before, .container:after {
content: "";
display: table;
}
.container:after {
clear: both;
}
.container {
background: #eaeaed;
margin-bottom: 24px;
*zoom: 1;
}
.container-75 {
width: 75%;
}
.container-50 {
```

```
margin-bottom: 0;
width: 50%;
}
.container, section, aside {
border-radius: 6px;
}
section, aside {
background: #2db34a;
color: #fff;
margin: 1.858736059%;
padding: 20px 0;
text-align: center;
}
section {
float: left;
width: 63.197026%;
}
aside {
float: right;
width: 29.3680297%;
}
</style>
</head>
<body>
<h1>100% Wide Container</h1>
<div class = "container">
<section>Section</section>
<aside>Aside</aside>
```

```
</div>


<h1>75% Wide Container</h1>
<div class = "container container-75">
<section>Section</section>
<aside>Aside</aside>
</div>
<h1>50% Wide Container</h1>
<div class = "container container-50">
<section>Section</section>
<aside>Aside</aside>
</div>
</body>
</html>
```

Output:



- **Media queries**

Media queries is for different style rules for different size devices such as mobiles, desktops, etc.,

```
<html>
<head>
<style>
body {
background-color: lightpink;
}
@media screen and (max-width: 420px) {
body {background-color: lightblue;}}
</style>
</head>
<body>
<p>
If screen size is less than 420px, then it will show lightblue
color, or else it will show light pink color
</p>
</body>
```

```
</html>
```

2.6. Application layering for the Desired Design

Application layering in the context of web development refers to the organized and structured application of different layers of design elements, styles, and functionality to achieve a desired overall design for a website. This approach helps in creating a modular and maintainable design system. In CSS, the z-index property is used for controlling the stacking order of positioned elements

SELF-CHECK 2

Part-I: Multiple choice

1. What type of CSS is the following code snippet?

```
<h1 style="color: blue ;">A Blue Heading</h1>
```

A. Inline C. External

B. None of the above D. Internal

2. Which HTML tag is used to declare internal CSS?

A. <style> C. <script>

B. None of the above D. <link>

3. How can we select an element with a specific ID in CSS?

A. # C. ^

B. . (dot) D. None of the above

4. How can we write comments in CSS?

A. /* */ C. #

B. // D. All of the above

5. The CSS property used to specify the transparency of an element is?

A. Opacity C. Filter

B. Visibility D. All of the above

6. Which CSS feature is commonly used to create flexible and responsive page layouts in web design?

a) Transitions c) Flexbox

b) Animations d) Transforms

7. What CSS feature is essential for creating responsive web pages that adapt to different screen sizes?

a) Fixed positioning c) Inline styling

b) Media queries d) Float layouts

8. Why might developers use vendor prefixes in CSS?

a) To hide styles from specific browsers

b) To create animations

c) To apply styles universally

d) To address cross-browser compatibility issues

Part-II: Give short answer

1) Explain the importance of page layout in web design, and how can CSS be utilized to create effective page layouts?
2) What are the different CSS positioning properties, and how do they impact the placement of document elements on a web page?
3) In what ways can CSS be used to create responsive web pages that adapt to varied screen resolutions, and why is responsiveness crucial in modern web design?
4) Why might developers need to apply browser-specific rules in CSS, and what are some common techniques for addressing cross-browser compatibility issues?

Part-III: Say True or False

1) In CSS, the z-index property is used for controlling the stacking order of positioned elements.
2) 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.
3) Fixed-width layouts are recommended for accommodating varied screen resolutions.
4) External style sheets are applied to multiple pages using the **<style>** tag within the HTML document.
5) The position: absolute CSS property positions an element relative to its nearest positioned ancestor.
6) CSS Flexbox and CSS Grid are techniques commonly used for page layout creation.
7) CSS is used for styling the structure and layout of a web page.

OPERATION SHEET-2.1

Operation Title: Creating a Page Layout

Purpose: Creating a Page Layout Using CSS Flexbox

Tools and equipment: Browsers computer and text editor

Steps in doing the task

Step1- Create HTML Structure with a container div, and within it, create two div’s for the navigation menu and main content.(index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
    <title>Flexbox Two-Column Layout</title>
</head>
<body>
    <div class="container">
        <nav class="navigation">
            <ul>
                <li><a href="#">Home</a></li>
                <li><a href="#">About</a></li>
                <li><a href="#">Contact</a></li>
            </ul>
        </nav>

        <main class="content">

            <h1>Welcome to My Website</h1>

            <p>This is the main content of the webpage.</p>

        </main>

    </div>

</body>

</html>
```

Step-2 create CSS file

Create a CSS file (styles.css) and Apply Flexbox properties to the container to achieve a two-column layout .

```
/* styles.css */

body {

    margin: 0;

    font-family: Arial, sans-serif;

}

.container {

    display: flex;

}

.navigation {

    flex: 0 0 20%; /* Navigation column takes 20% width */

   background-color: #edc2c2;

    color: #fff;
    padding: 20px;

}

.content {

    flex: 1; /* Content column takes remaining space */

    padding: 20px;

}
```

Step-3 Link External CSS File

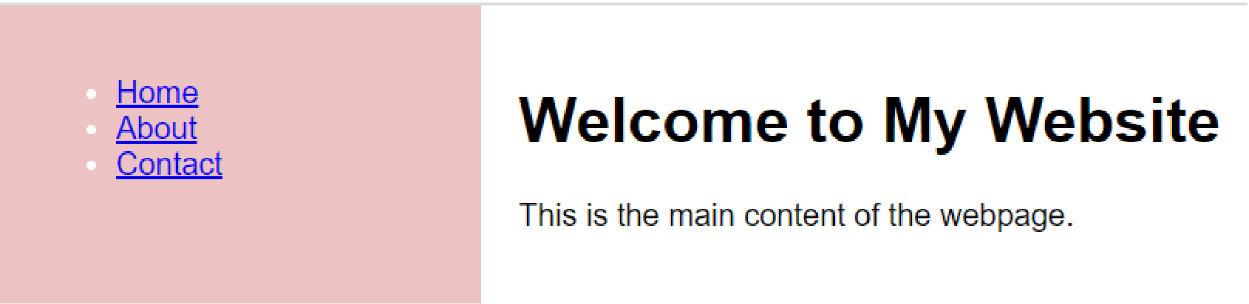
```
    <link rel="stylesheet" href="styles.css">
```

Step-4 test

Open the HTML file in a web browser and check if the two-column layout is achieved.

Adjust the content and styles as needed.

Quality Criteria: The output should be look like this



- Home
- About
- Contact

Welcome to My Website

This is the main content of the webpage.

OPERATION SHEET 2.2

Operation Title: Positioning Document Elements with CSS

Purpose: position a specific element “image “at the center of the page.

Tools and equipment: browsers computer and text editor

Steps in doing the task

Step-1: Create HTML Structure with the element you want to center. For this example, let's center an image.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
    <title>Centering Element Example</title>
</head>
<body>
    <div class="center-container">
        <img src="example-image.jpg" alt="Centered Image">
    </div>
</body>
</html>
```

Step-2 Create a CSS file (styles.css) and link it to your HTML document.

Apply CSS properties to center the element.

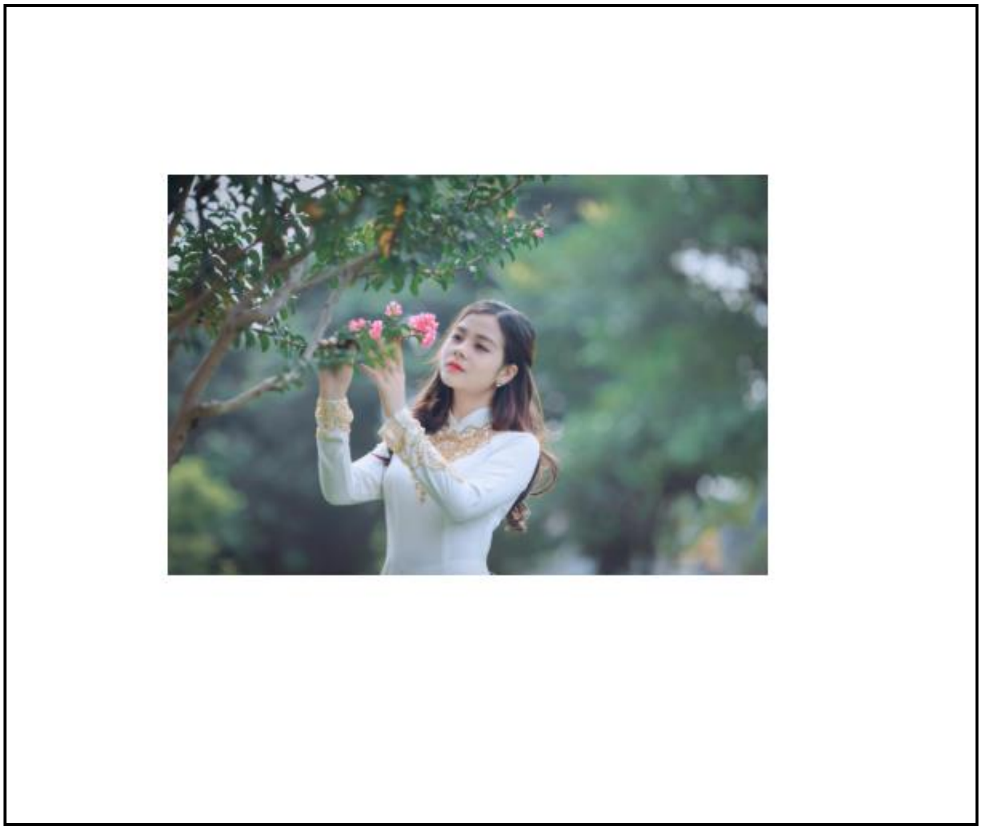
```
/* styles.css */
```

```
body, html {
   height: 100%;
   margin: 0;
   display: flex;
   justify-content: center;
   align-items: center;
}
.center-container {
   text-align: center;
}
.center-container img {
   width: 300px; /* Adjust the width as needed */
   height: auto;}
```

Step-3 Test and Refine

Open the HTML file in a web browser and verify that the image is centered on the page.

Quality Criteria: The output should be look like this



OPERATION SHEET 2.3

Operation Title: Creating Responsive Web Pages

Purpose: Creating Responsive Web Pages

Tools and equipment: computer Browsers and Text editor

Steps in doing the task

Step-1 Create HTML Structure with a header, main content, and footer.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="styles.css">
    <title>Responsive Webpage</title>
</head>
<body>
    <header>
        <h1>My Responsive Website</h1>
    </header>
    <main>
        <!-- Main content goes here -->
    </main>
    <footer>
        <p>&copy; 2023 My Website</p>
    </footer>

</body>

</html>
```

Step-2 Create a CSS file (styles.css) and link it to your HTML document.

Apply responsive design techniques using media queries.

```
/* styles.css */

body {font-family: 'Arial', sans-serif;

    margin: 0;

    padding: 0;}

header, main, footer {

    padding: 20px;

    text-align: center;

}

header {

    background-color: #3498db;

    color: #fff;

}

main { background-color: #ecf0f1;

}

footer {

    background-color: #3498db;

    color: #fff;

}
```

```
/* Responsive Design for Different Screen Sizes */

@media only screen and (max-width: 768px) {

    header, main, footer {

        padding: 10px;

    }

}
```

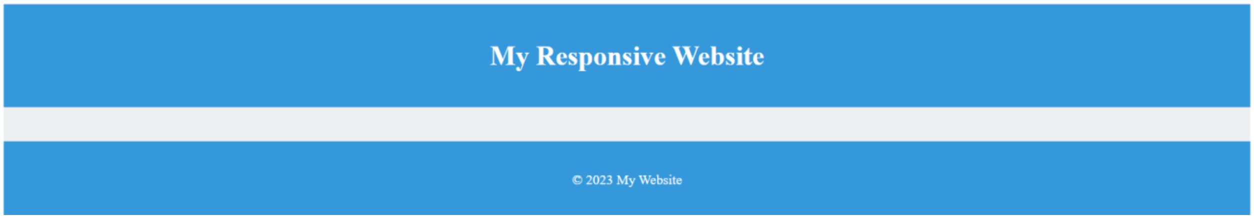
Step-3 Test and Refine

Open the HTML file in a web browser and resize the browser window to test responsiveness.

Ensure that the layout and styling adapt to different screen sizes.

Quality Criteria: The output should be look like this

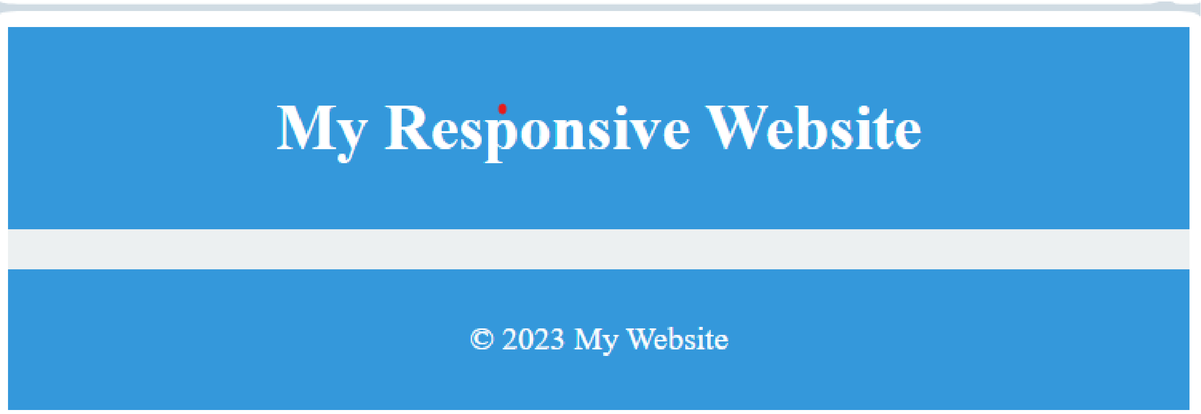
In large screen



My Responsive Website

© 2023 My Website

In small screen



My Responsive Website

© 2023 My Website

OPERATION SHEET-2.4

Operation title: Styling Navigation Menu Links with CSS

Write the HTML and CSS code to achieve these styling requirements. Additionally, create a sample navigation menu with at least three links for testing your styles.

Purpose: styling navigation menu links with CSS is to enhance user experience, improve navigation usability

Tools and Materials: Computer**,** Text editors (VS /Notepad/Notepad ++/Sublime)

Steps in doing the task

1. Create the HTML Structure:

Open your preferred text editor (e.g., Visual Studio Code, Sublime Text, Notepad).

Create a new HTML file **index.html** and add the basic HTML structure.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <link rel="stylesheet" href="styles.css">
   <title>CSS Hover Example</title>
</head>
<body>
  <a href="#" class="nav-link">Home</a>
  <a href="#" class="nav-link">About</a>
   <a href="#" class="nav-link">Contact</a>
</body>
</html>
```

2. Create an external CSS file (styles.css) and apply the following styles

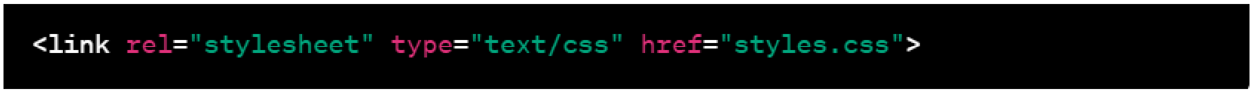
- Set the default color of the links to black.
- When a user visits a link, change the color to green.
- When the user hovers over a link, change the color to yellow.
- When a user clicks on a link, change the color to purple.
- Text decoration: none
- Display: inline block
- Padding 8px 16px

- Set background color: rgb(243, 145, 17)

```
/* styles.css */

/* Set the default color of the links to black */

body{

    background-color: rgb(243, 145, 17);

}

.nav-link {

  color: black;

  text-decoration: none; /* Remove the default underline */

  padding: 8px 16px; /* Add padding for better visual appearance */

  display: inline-block; /* Make links behave like blocks for padding to work */

}

/* Change color when a user visits the link */

.nav-link:visited {

  color: green;

}

/* Change color on hover */

.nav-link:hover {

  color: yellow;

}

/* Change color on click */

.nav-link:active {

  color: purple;
}
```

3. Link External CSS File

In the <head> section of your HTML document, link the external CSS file.

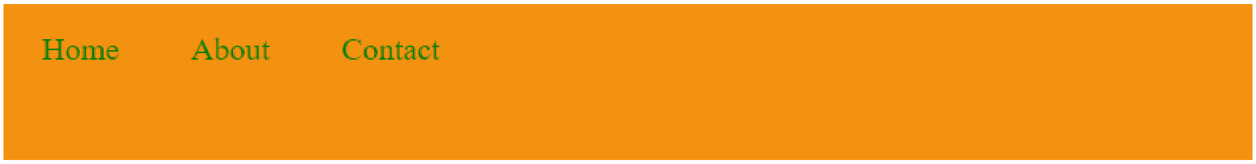


```
<link rel="stylesheet" type="text/css" href="styles.css">
```

4. Test in Browser

Save your HTML and CSS files. Open the HTML file in a web browser to see the individual links with the applied styles.

Quality Criteria: The output should be look like this



OPERATION SHEET-2.5

Operation title: Explore the use of the CSS: hover pseudo-class in different scenarios

Purpose: Change Background Color for Button and Change Border Shape for button

Steps in doing the task

step1: create HTML file (index.html) and Create an HTML button element with the class name "custom-button".

```
<!DOCTYPE html>

<html lang="en">

<head>

   <meta charset="UTF-8">

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

   <link rel="stylesheet" href="styles.css">

   <title>CSS Hover Example</title>

</head>

<body>

   <!-- HTML button element with the class "custom-button" -->

   <button class="custom-button">Hover Me</button>

</body>

</html>
```

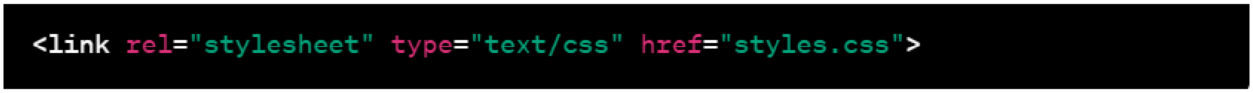
Step2: create css file(styles.css)

```
/* Styling for the custom button */

.custom-button {

   background-color: greenyellow;

   padding: 10px;

   font-size: large;
```

```
    border: none;

    cursor: pointer;

}

/* Hover effect for the custom button */

.custom-button:hover {

    background-color: gold;

}
```

Step3 link HTML to CSS

In the <head> section of your HTML document, link the external CSS file.



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

Steps in doing the task

Step1: create HTML file link.html and (<div>) element with the class name "border-link".

```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <link rel="stylesheet" href="styles.css">

    <title>CSS Hover Example</title>

</head>

<body>
```

```
    <!-- HTML div element with the class "border-link" -->
```

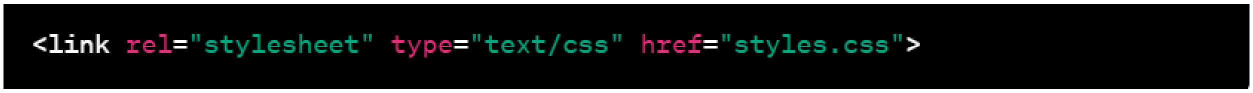
```
    <div class="border-link">Hover Me</div>

</body>

</html>
```

Step2:create CSS file (styles.css) and Apply CSS to set its text color to #40a944, text-transform to uppercase, border to 4px solid #40a944, border-radius to 10px, and display to inline-block. And **hover** pseudo-class to change the text color to #494949, border-radius to 45px, and border color to #494949 when the mouse cursor hovers over the element

```
/* Styling for the div with the class "border-link" */

.border-link {

    color: #40a944;

    text-transform: uppercase;

    border: 4px solid #40a944;

    border-radius: 10px;

    display: inline-block;

    padding: 10px 20px; /* Add padding for better appearance */

    cursor: pointer; /* Add cursor style to indicate clickability */

}

/* Hover effect for the div with the class "border-link" */

.border-link:hover {

    color: #494949;

    border-radius: 45px;
    border-color: #494949;

}
```

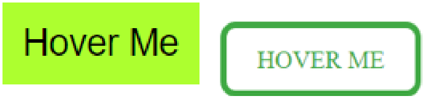
Step3 link HTML to CSS

In the <head> section of your HTML document, link the external CSS file.



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

Quality Criteria: The output should be look like this



OPERATION SHEET 2.6

Operation title: creating webpage layout using HTML and CSS

Purpose: create webpage layout using HTML and CSS

Tool and Equipment: computer browser and text editor

Steps in doing the task

Step1: create Html file index.html and apply the following

- Create an HTML element for the header with the class "header."
- Create an HTML element for the navigation bar with the class "topnav."
- Create HTML elements for three columns with the class "column."
- Create HTML elements for three columns with the class "column."
- Create HTML elements for left, middle, and right columns with the classes "column side," "column middle," and "column side" respectively.
- Create an HTML element for the footer with the class "footer

```
<!DOCTYPE html>
```

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



</head>
<body>
  <!-- Header -->
  <div class="header">
    <h1>Website Header</h1>
    <p>Your website tagline goes here.</p>
  </div>
  <!-- Navigation Bar -->
  <div class="topnav">
    <a href="#">Home</a>
    <a href="#">About</a>
    <a href="#">Services</a>
    <a href="#">Contact</a>
  </div>
  <!-- Content - 3 Column Layout -->
  <div class="row">
    <div class="column">
      <h2>Column 1</h2>
      <p>Some content goes here...</p>
    </div>
    <div class="column">
      <h2>Column 2</h2>
      <p>Some content goes here...</p>
    </div>
    <div class="column">
      <h2>Column 3</h2>
      <p>Some content goes here...</p>
    </div>
  </div>
  <!-- Unequal Columns -->
  <div class="row">
    <div class="column side">
      <h2>Side Column</h2>
      <p>Some content goes here...</p>
    </div>
    <div class="column middle">
      <h2>Main Content</h2>
      <p>Main content goes here...</p>
    </div>
```

```
   <div class="column side">
```

```
      <h2>Side Column</h2>

      <p>Some content goes here...</p>

    </div>

  </div>

  <!-- Footer -->

  <div class="footer">

    <p>&copy; 2023 Your Website Name. All Rights Reserved.</p>

  </div>

</body>

</html>
```

Step2 create CSS file styles.css and apply the following

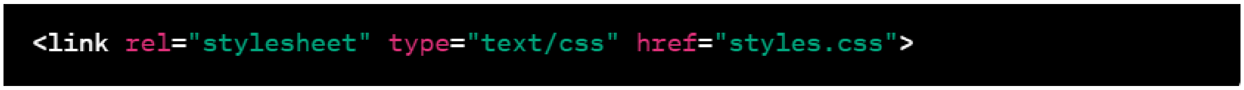
- Apply CSS styles to header give it a background color of #F1F1F1, center-align text, and add padding of 20px.
- Apply CSS styles to the navbar container with overflow: hidden and a background color of #333.
    - Apply CSS styles to navbar links with float, display, color, text-align, padding, and text-decoration.
    - Change the link color on hover (background-color: #ddd, color: black).
- Apply CSS styles to float the columns left and set the width to 33.33%.
- Create a CSS rule to clear floats after the columns.
- Implement responsive design for screens with a width of 600px or less (change column width to 100%).

    - Modify the content section to include unequal column widths.
        - Apply CSS styles to set the widths as follows: left column (25%), middle column (50%), and right column (25%).
        - Implement responsive design to stack the columns on top of each other for screens with a width of 600px or less.
- Apply CSS styles to footer give it a background color of #F1F1F1, center-align text, and add padding of 10px.

```
/* Header */
.header {
  background-color: #F1F1F1;
  text-align: center;
  padding: 20px;
}
/* Navigation Bar */
.topnav {
  overflow: hidden;
  background-color: #333;
}
.topnav a {
  float: left;
  display: block;
  color: #f2f2f2;
  text-align: center;
  padding: 14px 16px;
  text-decoration: none;

}

.topnav a:hover {

  background-color: #ddd;

  color: black;

}

/* Content - 3 Column Layout */

.column {

  float: left;

  width: 33.33%;

}

.row:after {

  content: "";

  display: table;

  clear: both;

}

@media screen and (max-width: 600px) {

  .column {

    width: 100%;

  }

}

/* Unequal Columns */

.column.side {
  width: 25%;
}
.column.middle {
  width: 50%;
}
@media screen and (max-width: 600px) {
  .column.side, .column.middle {
    width: 100%;
  }
}
/* Footer */
.footer {
  background-color: #F1F1F1;
  text-align: center;
  padding: 10px;
}
```

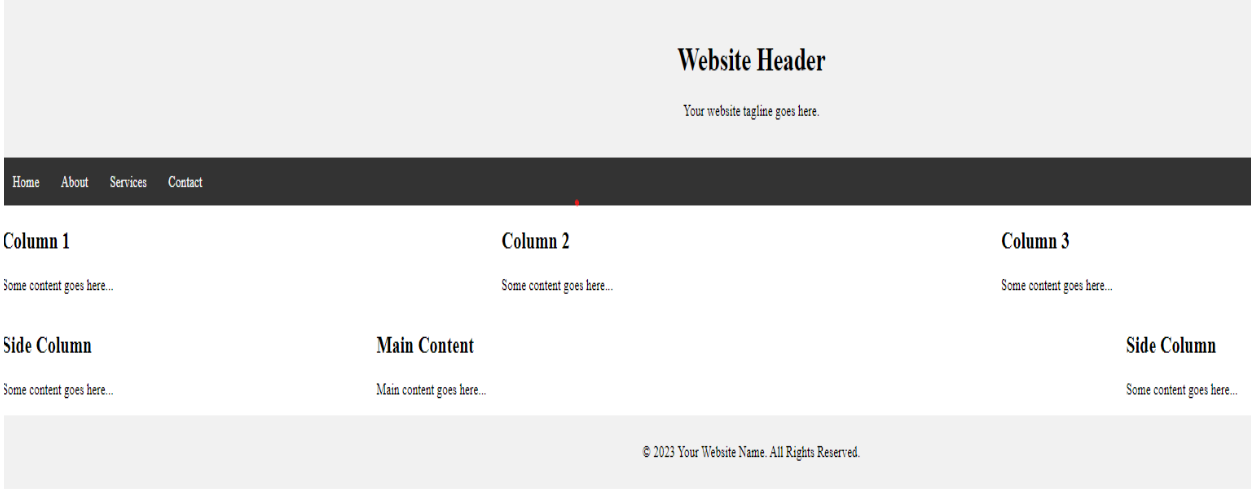
Step3 link html to css

In the <head> section of your HTML document, link the external CSS file.



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

Quality criteria: Output



Website Header

Your website tagline goes here.

Home About Services Contact

Column 1

Some content goes here...

Column 2

Some content goes here...

Column 3

Some content goes here...

Side Column

Some content goes here...

Main Content

Main content goes here...

Side Column

Some content goes here...

© 2023 Your Website Name. All Rights Reserved.

LAP TESTS

Instructions: Given necessary templates, tools and materials you are required to perform the following tasks accordingly.

Task1: Create a Page Layout Using CSS Grid

Task2: Apply Style Sheets across Multiple Pages

Task3: Creating a Page Layout Using CSS Flexbox

Task4: Create a web page for varied screen resolutions

Task5: You are tasked with styling links for a website. Implement the following requirements using CSS:

1) Set the default color of the links to black.
2) When a user visits a link, change the color to green.
3) When the user hovers over a link, change the color to yellow.
4) When a user clicks on a link, change the color to purple.

Task6: Create a registration form?

Inside the form, include input fields for the user to enter information (name, email, and password).

Add labels for each input field using the **<label>** element.

Include a submit button to submit the form.

UNIT THREE: WEB PAGE AND CSS VALIDATION AND TESTING

This unit is developed to provide you the necessary information regarding the following content coverage and topics:

- CSS standard and validation
- Web page evaluation across various browsers
- Rectification of browser

This unit will also assist you to attain the learning outcomes stated in the cover page. Specifically, upon completion of this learning guide, you will be able to:

- Validate CSS against industry standards
- Test a web page in various browsers
- Rectify browser differences to ensure website is accessible
- Explore techniques for handling browser-specific styling challenges
- Emphasize the importance of adhering to industry standards in CSS

3.1. CSS STANDARD AND VALIDATION

Validation of CSS code for standards compliance has some advantages and is no different from HTML validation. A document with CSS code will be considered valid if it has passed the corresponding check and does not contain errors.

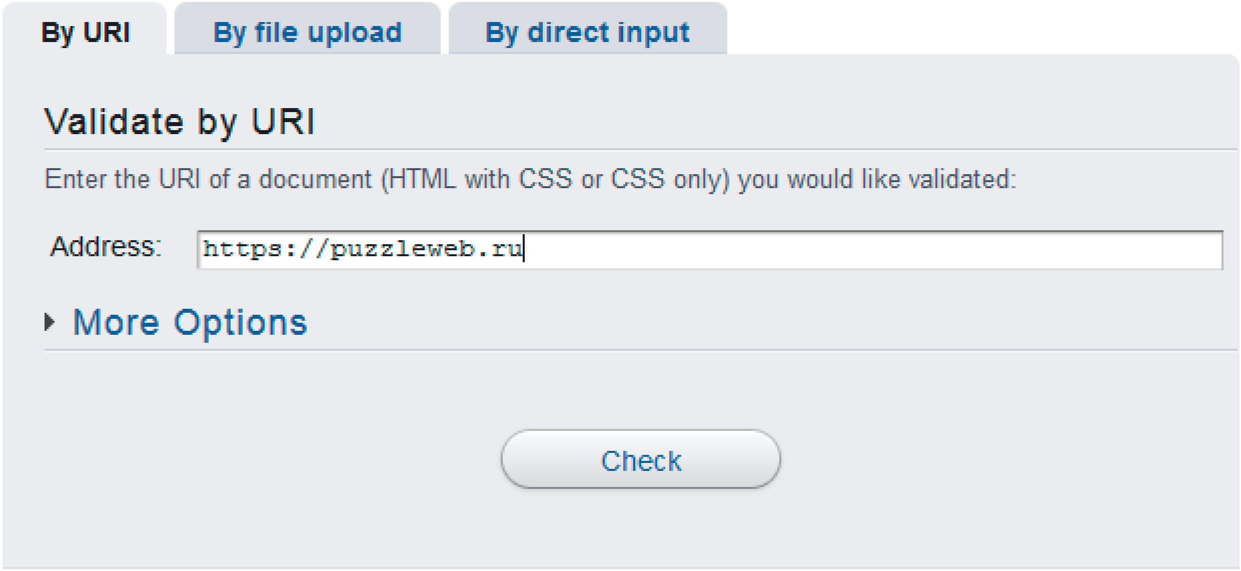
If you use stylesheets predominantly in an external file, this makes it much easier to check your code because you have to check only one file, not each page individually. If you make mistakes when writing style sheet code, you can also use a validator to not look for errors manually.

How to use the validator?

You can use a CSS validator to check CSS code for errors and standards compliance, which provides three ways to check:

URL verification

Just enter the URL of the web page you want to check and click the Check button.



Validate by URI

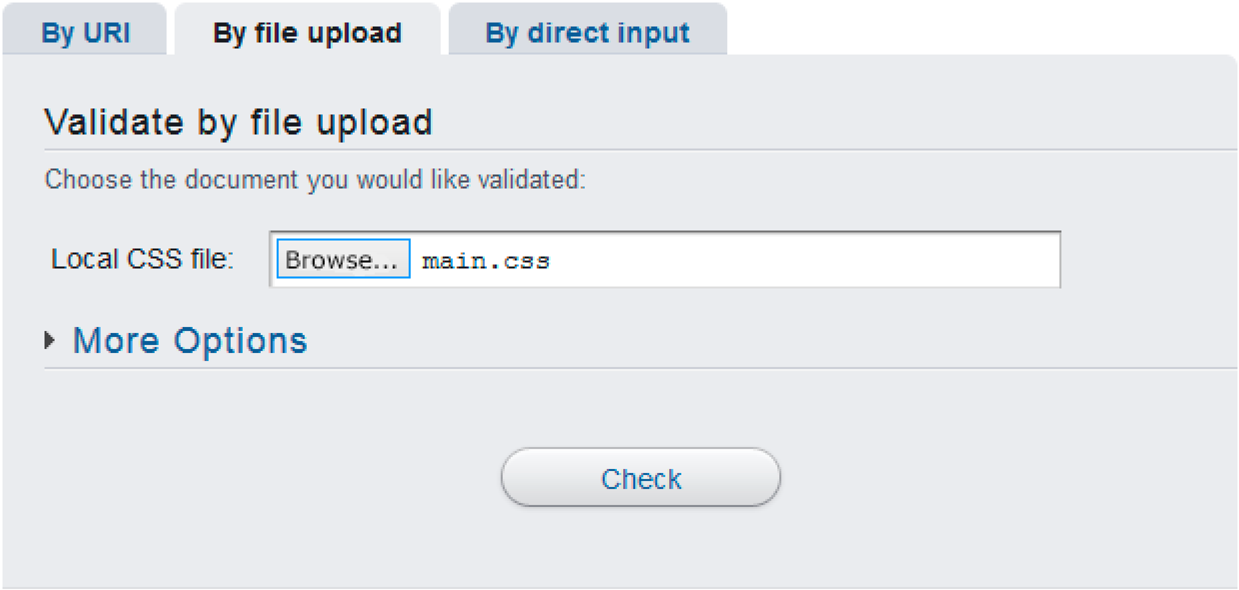
Enter the URI of a document (HTML with CSS or CSS only) you would like validated:

Address: https://puzzleweb.ru

After validating the code with a validator, you will see one of two inscriptions-either that everything was successful and you congratulate with the valid code, or that your CSS code found errors that you are about to fix.

Checking the downloaded file

This method allows you to download the scanned document to the server. To do this, click the Browse button and select the file that you want to check.



Validate by file upload

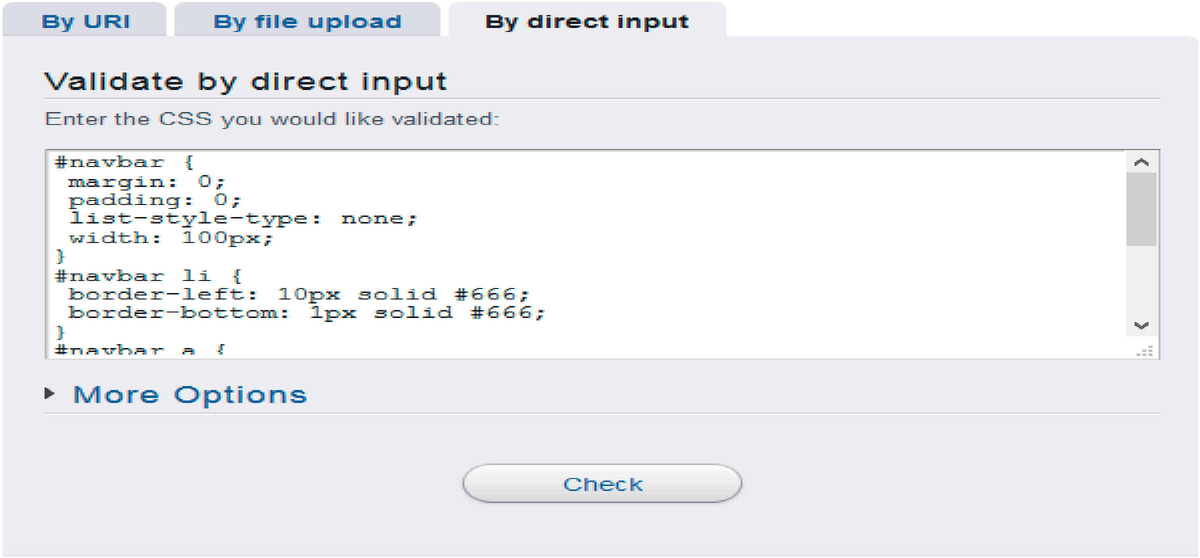
Choose the document you would like validated:

Local CSS file: Browse... main.css

The server will automatically recognize the type of file you are downloading.

Checking pasted code

This method is ideal for validating part of a CSS file. You only need to copy the code into the text box.



Validate by direct input

Enter the CSS you would like validated:

```
#navbar {
 margin: 0;
 padding: 0;
 list-style-type: none;
 width: 100px;
}
#navbar li {
 border-left: 10px solid #666;
 border-bottom: 1px solid #666;
}
#navbar a {
```

3.2. WEB PAGE EVALUATION ACROSS VARIOUS BROWSERS

Browser compatibility or cross-browser testing is a process of evaluating whether the functionality of a web application remains consistent when the end-user visits your web-application from a different browser. It is also used to validate browser compatibility for desktop and web applications.

What Features Are Analyzed in a Browser Test?

1. Basic functionality: To ensure that simple stuff is working on most browser-OS combinations

- All input fields and their validation
- Dialog boxes and menus are working as expected
- Input touch on mobile devices

2. Design: To make sure fonts, styles, images, and layouts are matching requirements that were sent by designers

3. Responsiveness: Verify that design is solid and not broken in some layers after you change the resolution/orientation of a browser

- **How to Choose Browsers?**

There are two easy ways to choose from all the variety of browsers

1. Based on popularity: Choose browsers that can overcome the 5% barrier in global statistics. So, for example, in 2020 you would choose Google Chrome, Safari, Mozilla Firefox, Microsoft Edge, and IE. Regarding IE, it is a bit tricky because there are a lot of versions of this browser. Our recommendation is to support only IE 10, the reason being that IE 8 and 9 are too old and hard to support.

2. Based on the analysis: If you have some analytics tools (Google Analytics, etc.) that are tracking all traffic stats, you will have a full understanding of what your customers are using, and it will be an easy decision to make. If not, you may try to know your customer better by creating some polls, etc. Would it be worth to support all the versions of different browsers? It depends on if you agree that development and testing time will be higher every time. Either way, in the long run, you will have to sacrifice supporting old versions.

The decision on which browsers and platforms to choose depends on the business team and the marketing team.

3.3. RECTIFICATION OF BROWSER

Cross-browser compatibility issues can be a major pain for web developers as different browsers often render web pages differently. However, with proper planning and the use of CSS, these issues can be resolved. In this article, we will explore how to solve cross-browser compatibility issues with CSS.

- **Normalize CSS:** One of the first steps in solving cross-browser compatibility issues is to use a CSS normalization library such as Normalize.css. This library makes sure that default styles are consistent across different browsers, which can help resolve many common compatibility issues.
- **Use vendor prefixes:** Vendor prefixes are special CSS properties and values that are used to ensure that styles are rendered consistently across different browsers. For example, the CSS property "transform" is prefixed with "-webkit-" for Webkit-based browsers like Chrome and Safari, "-moz-" for Mozilla Firefox, and "-ms-" for Microsoft Internet Explorer.

Test on multiple browsers: To ensure that your styles are compatible with different browsers, it is important to test your styles on multiple browsers. This can be done by installing different browsers on your development machine or by using online tools such as Browser Stack or Cross Browser Testing.

- **Use CSS Resets:** CSS Resets are a set of styles that are used to reset the default styles of different elements to a consistent state. This can help resolve compatibility issues by ensuring that the styles applied to an element are the same across different browsers.
- **Avoid browser-specific hacks:** Browser-specific hacks are CSS styles that are specifically designed to work with a particular browser and are often used to resolve compatibility issues. However, these hacks can make your stylesheet difficult to maintain and can also cause compatibility issues with other browsers.
- **Use feature detection:** Feature detection is a technique that is used to determine whether a particular feature is supported by the user's browser. This can be useful in solving compatibility issues by allowing you to apply styles only to browsers that support a particular feature.

- **Use CSS frameworks:** CSS frameworks such as Bootstrap and Foundation can be useful tools in solving cross-browser compatibility issues as they provide a consistent set of styles that are tested across multiple browsers.

SELF-CHECK-3

Part I: Multiple choice

1) What is the purpose of CSS validation?
   A. Enhancing website aesthetics C. Checking code against specifications
   B. Optimizing database performance D. Improving server response time
2) Which tool is commonly used for CSS validation?
   A. Photoshop
   B. W3C CSS Validation Service
   C. Visual Studio Code
   D. Bootstrap
3) Why is testing web pages across different browsers important?
   A. To increase server speed
   B. To ensure a consistent user experience
   C. To reduce the size of CSS files
   D. To enhance SEO ranking

Part II Give short answer

1) Why is it important to adhere to CSS standards?
2) What is CSS validation, and why should web developers validate their CSS code?
3) Name a popular tool for CSS validation and briefly explain how it works.
4) Why is it important to test web pages across different browsers?

Part III Say True or False

1) Cross-browser testing is necessary because all browsers interpret HTML and CSS in the exact same way.
2) Using browser developer tools, developers can identify and fix browser-specific issues in CSS code.
3) Ignoring browser differences is a recommended strategy for handling cross-browser compatibility issues.

OPERATION SHEET-3.1

Operation Title: CSS Validation with W3C CSS Validation Service

Purpose: Validate CSS Code

Required tools and equipment: Internet and computer

Steps

Step-1 Open the W3C CSS Validation Service:

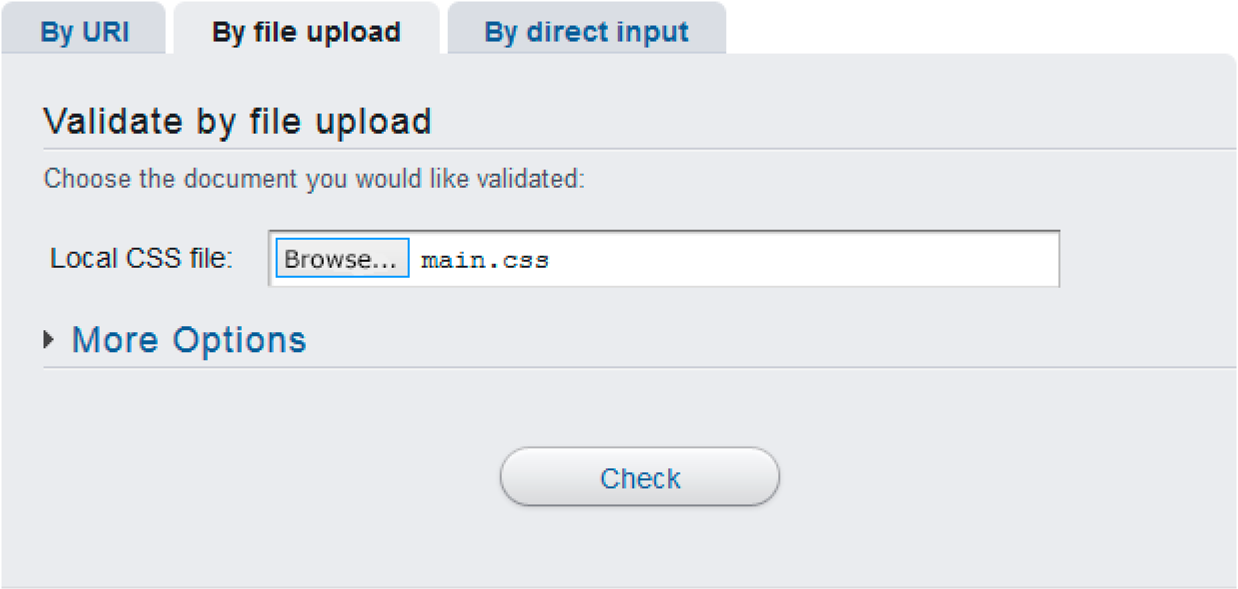
Go to the W3C CSS Validation Service website (https://jigsaw.w3.org/css-validator/).

Step-2 Choose Validation Method:

There are two validation methods: "By URL" or "By File Upload."

If your CSS code is in a file, choose "By File Upload" and upload your CSS file.

If your CSS code is on a webpage, choose "By URL" and enter the URL.



Validate by file upload

Choose the document you would like validated:

Local CSS file: main.css

Step-3 Submit and Validate:

Click the "Check" or "Validate by File Upload" button, depending on your chosen method.

Review the Results:

The validation service will analyze your CSS code and display the results.

Check for any errors or warnings reported. Errors need to be fixed to ensure compliance with CSS standards.

Step-4 Navigate to Error Details:

If there are errors or warnings, click on the error messages to see detailed information about each issue.

Understand the nature of the problems and plan for corrections.

Step-5 Correct the CSS Code:

Open your CSS file using a text editor.

Locate and fix the issues identified by the validation service.

Common issues include syntax errors, unsupported properties, or misspelled values.

Step-6 Revalidate:

After making corrections, go back to the W3C CSS Validation Service.

Repeat the validation process to ensure that all issues have been addressed.

Step-7 Repeat for Each Page:

Quality criteria: Create error free code

LAP Tests

Instructions: Given necessary templates, tools and materials you are required to perform the following tasks accordingly.

Task 1: validate CSS code with W3C CSS Validation Service

Task 2: Cross-Browser Testing Using Browser Developer Tools

REFERENCES

Books

1. Web Design with HTML and CSS Digital Classroom Jeremy Osborn, Jennifer Smith, and the AGI Training Team
2. HTML & CSS: The Complete Reference Thomas A. Powell, fifth edition, 2010
3. Learn to Code HTML & CSS Develop & Style Websites Shay Howe, 2014
4. BEGINNING CSS: CASCADING STYLE SHEETS FOR WEB DESIGN Ian Pouncey Richard York , 2011, Third Edition

URL

https://www.w3schools.com/

https://www.tutorialspoint.com/html/index.htm

DEVELOPER'S PROFILE

No	Name	Qualification	Field of Study	Organization/ Institution	Mobile number	E-mail
1	Frew Atkilt	M-Tech	Network & Information Security	Bishoftu Polytechnic College	0911787374	frew.frikii@gmail.com
2	Gari Lencha	MSc	ICT Managment	Gimbi Polytechnic	0917819599	Garilencha12@gmail.com
3	Kalkidan Daniel	BSc	Computer Science	Entoto Polytechnic	0978336988	kalkidaniel08@gmail.com
4	Solomon Melese	M-Tech	Computer Engineering	M/G /M/Polytechnic College	0918578631	solomonmelese6@gmail.com
5	Tewodros Girma	MSc	Information system	Sheno Polytechnic College	0912068479	girmatewodiros @gmail.com
6	Yohannes Gebeyehu	BSc	Computer Science	Entoto Polytechnic College	0923221273	yohannesgebeyehu73@gmail.com