Web Design & Development with WordPress

Introduction

Welcome to *"Web Design & Development with WordPress"*! This lecture sheet series is designed to guide you through the entire process of creating stunning and functional websites using WordPress. Whether you are a beginner looking to build your first website or an experienced developer aiming to refine your skills, these lecture sheets will serve as a comprehensive resource.

Why Learn WordPress?

WordPress is the world's most popular Content Management System (CMS), powering over 40% of all websites on the internet. Its flexibility, ease of use, and vast ecosystem make it the ideal platform for creating anything from personal blogs to complex e-commerce websites.

Who Is This Lecture Sheet For?

This lecture sheet series is crafted for:

- **Beginners** who want to learn how to design and develop websites using WordPress from scratch.
- **Freelancers** aiming to offer WordPress services to clients.
- **Business Owners** who wish to manage and maintain their own websites.
- **Developers** seeking to expand their skills and explore advanced WordPress features.

What You Will Learn

Throughout these lecture sheets, you will learn:

Web Technologies

- HTML: Structure of web pages
- CSS: Styling and layout design
- JavaScript: Adding interactivity and dynamic content
- jQuery: Simplifying JavaScript operations
- Bootstrap: Building responsive and mobile-first designs

Backend Development

- PHP: Server-side scripting and dynamic content management
- MySQL: Database management and data storage

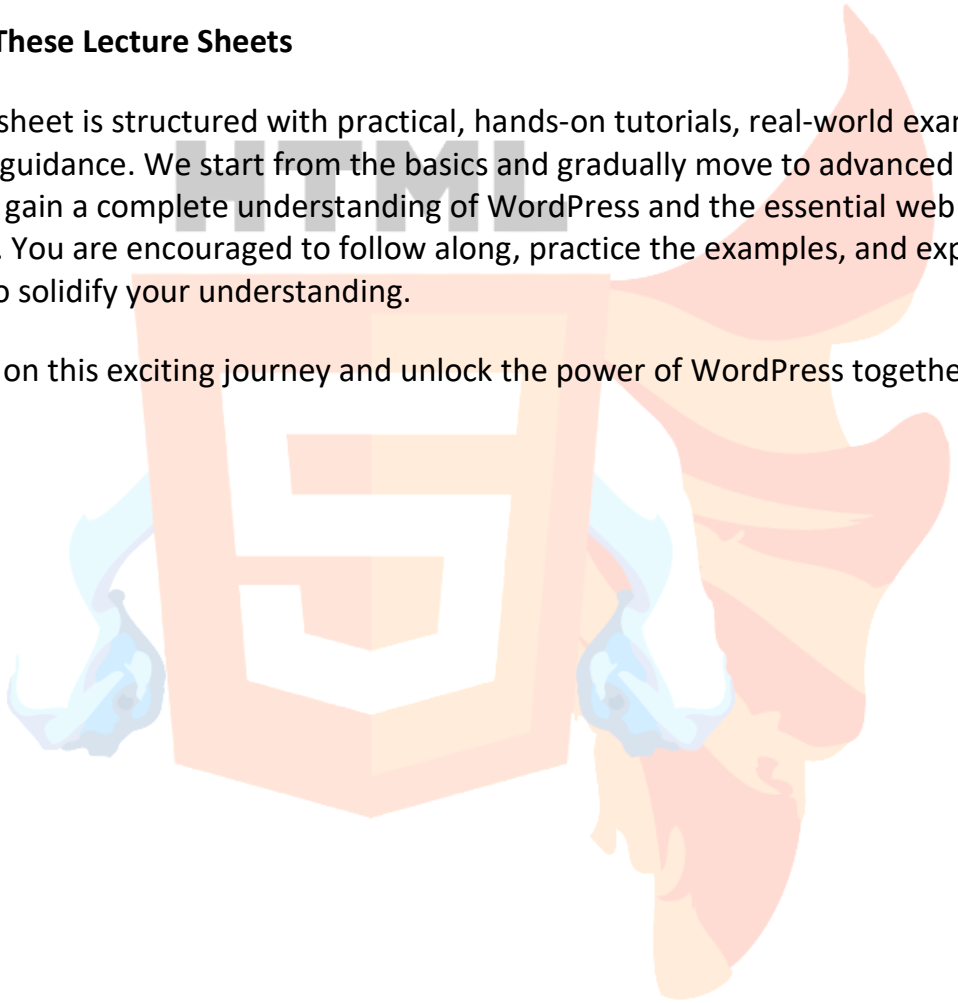
WordPress Development

- The basics of WordPress installation and setup
- Customizing themes and creating child themes
- Utilizing plugins to enhance website functionality
- Building responsive and SEO-friendly websites
- Managing content and optimizing performance
- Advanced techniques for customizing and extending WordPress
- Developing custom themes and plugins
- Security best practices and performance optimization

How to Use These Lecture Sheets

Each lecture sheet is structured with practical, hands-on tutorials, real-world examples, and step-by-step guidance. We start from the basics and gradually move to advanced topics, ensuring you gain a complete understanding of WordPress and the essential web technologies. You are encouraged to follow along, practice the examples, and explore beyond the lessons to solidify your understanding.

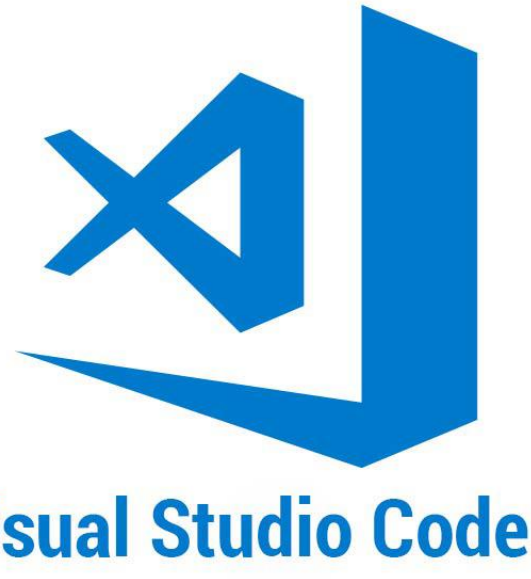
Let's embark on this exciting journey and unlock the power of WordPress together!



By successfully completing this course, you will be able to work effectively in both online and offline marketplaces.

About Lecture Sheet

In this lecture sheet, we will learn all the essential details of HTML (HyperText Markup Language), from basic concepts to advanced techniques. HTML is the foundation of web development, allowing you to structure web pages with text, images, links, tables, forms, and multimedia elements. This lecture sheet explains each topic step by step, making it easy for beginners to understand. You will explore HTML tags, attributes, page layout, and form handling. By practicing the provided examples, you will gain the skills to create fully functional web pages with confidence.



Lecture 1

Introduction to HTML & Basic Structure

Introduction to HTML

HTML (HyperText Markup Language) was created by **Tim Berners-Lee** in **1991** to share and link documents on the World Wide Web. The first version, **HTML 1.0**, was simple and focused on basic web structure. Over time, it evolved to **HTML5**, supporting multimedia, responsive design, and advanced web applications.

Why Use HTML?

HTML (HyperText Markup Language) is essential for creating and structuring web pages. It provides the foundation for displaying content like text, images, videos, and links on the internet. HTML is easy to learn, compatible with all browsers, and works seamlessly with CSS for styling and JavaScript for interactivity. It's the backbone of every website, allowing developers to build user-friendly and responsive web pages. Whether you're creating a simple webpage or a complex application, HTML is a fundamental tool for web development.

Basic Structure

```
<!DOCTYPE html>
<html>
    <head>
        <title>Page Title</title>
    </head>
<body>

    <h1>My First Heading</h1>
    <p>My first paragraph.</p>

</body>
</html>
```

HTML Editors

HTML Editors are software tools used to write, edit, and manage HTML code efficiently. They come with features like syntax highlighting, code completion, and error checking to make the coding process easier. There are two types of HTML editors: Text Editors and WYSIWYG (What You See Is What You Get) Editors.

1.Text Editors:

- Examples: Notepad++, Sublime Text, Visual Studio Code
- These are lightweight and provide flexibility for writing clean code. They often offer advanced features like extensions and plugins to improve productivity.

2.WYSIWYG Editors:

- Examples: Adobe Dreamweaver, Microsoft Expression Web
- These editors allow users to design web pages visually without writing code. They generate HTML code automatically as users design the page.

Using an HTML editor makes coding more efficient, especially for beginners and professionals who need to handle large projects.

HTML Elements

The HTML element is everything from the start tag to the end tag:

<tagname>Content goes here...</tagname>

Examples of some HTML elements:

```
<h1>This is element</h1>
<p>This is another element.</p>
```

HTML Attributes

HTML attributes provide additional information about HTML elements.

- All HTML elements can have attributes
- Attributes provide additional information about elements
- Attributes are always specified in the start tag
- Attributes usually come in name/value pairs like: name="value"

```
<h1 style="color: red;">My First Heading</h1>
```

style *is the attribute in this case*

HTML Headings

HTML headings are titles or subtitles that you want to display on a webpage

HTML headings are defined with the <h1> to <h6> tags.

<h1> defines the most important heading. <h6> defines the least important heading.

Code

```
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<h4>Heading 4</h4>
<h5>Heading 5</h5>
<h6>Heading 6</h6>
```

Output

Heading 1

Heading 2

Heading 3

Heading 4

Heading 5

Heading 6

HTML Paragraphs

A paragraph always starts on a new line, and is usually a block of text.

The HTML <p> element defines a paragraph

A paragraph always starts on a new line, and browsers automatically add some white space (a margin) before and after a paragraph.

```
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
```

HTML Styles

The HTML **style** attribute is used to add styles to an element, such as color, font, size, and more.

```
<h1 style="color: red;">My First Heading</h1>
```

Output - My First Heading

HTML Comments

"HTML comments are not displayed in the browser, but they help document your HTML source code. You can add comments to your HTML source using the following syntax:"

This version maintains the original meaning but improves the flow slightly. Let me know if you'd like more tweaks!

```
<!-- This is a comment -->
   <p>This is a paragraph.</p>
<!-- Remember to add more information here -->
```

Homework/Practice Task

1. What does the abbreviation HTML stand for, and why is it important in web development?
2. Name three popular HTML editors and describe their key features.
3. What is the purpose of the <!DOCTYPE> declaration in an HTML document?
4. Can you explain the basic structure of an HTML document? What are the roles of the <html>, <head>, and <body> tags?
5. What are HTML elements, and how are they defined in a webpage? Can you provide an example?
6. How do attributes enhance HTML elements, and what are some common attributes you can use in HTML tags?
7. What is the difference between <h1> and <h6> HTML tags? How are they used in a webpage?
8. How do you add a paragraph of text in HTML? Provide an example of the syntax.
9. How do you add comments in HTML? Why is it important to use comments in your code?
10. What is the purpose of the <head> section in an HTML document, and what are some common elements that go inside it?

Lecture 2

Styling and Multimedia Elements, Links, Lists

HTML Colors (Named, RGB, HEX)

HTML colors are specified with predefined color names, or with RGB, HEX, HSL, RGBA, or HSLA values.

```
<h1 style="color:Tomato;">Hello World</h1>
<p style="color:DodgerBlue;">Lorem ipsum...</p>
<p style="color:MediumSeaGreen;">Ut wisi enim...</p>
```

HTML Images (Embedding & Attributes)

Images can improve the design and the appearance of a web page.

The HTML <img> tag is used to embed an image in a web page.

Images are not technically inserted into a web page; images are linked to web pages. The <img> tag creates a holding space for the referenced image.

The <img> tag is empty, it contains attributes only, and does not have a closing tag.

The <img> tag has two required attributes:

- src - Specifies the path to the image
- alt - Specifies an alternate text for the image

Syntax

```
<img src="url" alt="alternatetext">
```

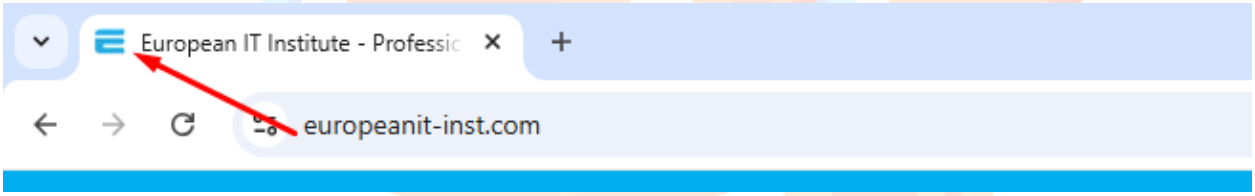
HTML Favicon (Adding Website Icon)

You can use any image you like as your favicon.

Tip: A favicon is a small image, so it should be a simple image with high contrast.

```
<!DOCTYPE html>
<html>
<head>
  <title>My Page Title</title>
  <link rel="icon" type="image/x-icon" href="/images/favicon.ico">
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

A favicon image is displayed to the left of the page title in the browser tab, like this:



HTML Audio (Embedding Sound)

The HTML <audio> element is used to play an audio file on a web page.

HTML Audio - How It Works

The controls attribute adds audio controls, like play, pause, and volume.

The <source> element allows you to specify alternative audio files which the browser may choose from. The browser will use the first recognized format.

The text between the <audio> and </audio> tags will only be displayed in browsers that do not support the <audio> element.

```
<audio controls autoplay>
  <source src="horse.ogg" type="audio/ogg">
  <source src="horse.mp3" type="audio/mpeg">
Your browser does not support the audio element.
</audio>
```

HTML Video (Embedding Videos)

The HTML <video> element is used to show a video on a web page.

How it Works

The controls attribute adds video controls, like play, pause, and volume.

It is a good idea to always include width and height attributes. If height and width are not set, the page might flicker while the video loads.

The <source> element allows you to specify alternative video files which the browser may choose from. The browser will use the first recognized format.

The text between the <video> and </video> tags will only be displayed in browsers that do not support the <video> element.

To start a video automatically, use the autoplay attribute:

```
<video width="320" height="240" autoplay muted>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>
```

HTML Links (Internal, External, Target)

Links are found on nearly all web pages and allow users to navigate from page to page.

HTML links are hyperlinks.

You can click on a link and jump to another document.

When you move the mouse over a link, the mouse arrow will turn into a little hand.

The HTML <a> tag defines a hyperlink. It has the following syntax:

```
<a href="url">link text</a>
```

The most important attribute of the <a> element is the href attribute, which indicates the link's destination.

The link text is the part that will be visible to the reader.

Clicking on the link text, will send the reader to the specified URL address.

```
<a href="https://europeanit-inst.com/">Visit europeanit-inst.com!</a>
```

Use target="_blank" to open the linked document in a new browser window or tab:

```
<a href="https://europeanit-inst.com/" target="_blank">Visit europeanit-inst.com!</a>
```

HTML Lists

HTML lists allow web developers to group a set of related items in lists.

Unordered HTML List

An unordered list starts with the <ul> tag. Each list item starts with the <li> tag.

The list items will be marked with bullets (small black circles) by default:

```
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
```

Ordered HTML List

An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.

The list items will be marked with numbers by default:

```
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
```

Homework/Practice Task

1. What is the difference between internal and external links in HTML? Provide examples.
2. How do you create an email link in HTML? Write the syntax.
3. What does the target="_blank" attribute do in an HTML link?
4. How do you link to a specific section of the same webpage using HTML?
5. What is the difference between ordered, unordered, and definition lists in HTML?
6. Write the HTML code to create a numbered list with three items.
7. How do you create a nested list using HTML? Provide an example.
8. How do you apply inline CSS to change the background color of a webpage?
9. What are the three ways to define colors in HTML? Provide examples for each (Named, RGB, HEX).
10. What is the purpose of the alt attribute in the <img> tag? Why is it important?
11. Write the HTML code to display an image with width=300px and height=200px.
12. How do you add a favicon to a webpage? Provide the correct syntax.
13. Write the HTML code to embed an audio file that plays automatically.
14. How do you embed a video in HTML and set it to autoplay?
15. What is the purpose of the href attribute in the <a> tag, and how is it used?

Lecture 3

Tables, Forms & Input Handling, Advanced HTML Concepts

HTML Tables

HTML tables allow web developers to arrange data into rows and columns.

Example

Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy

Define an HTML Table

A table in HTML consists of table cells inside rows and columns.

```
<table>
  <tr>
    <th>Company</th>
    <th>Contact</th>
    <th>Country</th>
  </tr>
  <tr>
    <td>Alfreds Futterkiste</td>
    <td>Maria Anders</td>
    <td>Germany</td>
  </tr>
  <tr>
    <td>Centro comercial Moctezuma</td>
    <td>Francisco Chang</td>
    <td>Mexico</td>
  </tr>
</table>
```

HTML Forms

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

```
<form>
.
form elements
.
</form>
```

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.

Here are some 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

HTML Input Types

Here are the different input types you can use in HTML:

- <input type="button">
- <input type="checkbox">
- <input type="color">
- <input type="date">
- <input type="datetime-local">
- <input type="email">
- <input type="file">
- <input type="hidden">
- <input type="image">
- <input type="month">
- <input type="number">
- <input type="password">
- <input type="radio">
- <input type="range">
- <input type="reset">
- <input type="search">
- <input type="submit">
- <input type="tel">
- <input type="text">
- <input type="time">

- `<input type="url">`
- `<input type="week">`

HTML Div

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

The element has no required attributes, but style, class and id are common.

The element is often used to group sections of a web page together.

```
<div>
  <h2>London</h2>
  <p>London is the capital city of England.</p>
  <p>London has over 9 million inhabitants.</p>
</div>
```

HTML class Attribute

The HTML class attribute is used to specify a class for an HTML element.

Multiple HTML elements can share the same class.

The class attribute is often used to point to a class name in a style sheet. It can also be used by a JavaScript to access and manipulate elements with the specific class name.

In the following example we have three elements with a class attribute with the value of "city". All of the three elements will be styled equally according to the .city style definition in the head section:

```html
<!DOCTYPE html>
<html>
<head>
<style>
.city {
  background-color: tomato;
  color: white;
  border: 2px solid black;
  margin: 20px;
  padding: 20px;
}
</style>
</head>
<body>

<div class="city">
  <h2>London</h2>
  <p>London is the capital of England.</p>
</div>

<div class="city">
  <h2>Paris</h2>
  <p>Paris is the capital of France.</p>
</div>

<div class="city">
  <h2>Tokyo</h2>
  <p>Tokyo is the capital of Japan.</p>
</div>

</body>
</html>
```

HTML id Attribute

The id attribute specifies a unique id for an HTML element. The value of the id attribute must be unique within the HTML document.

The id attribute is used to point to a specific style declaration in a style sheet. It is also used by JavaScript to access and manipulate the element with the specific id.

The syntax for id is: write a hash character (#), followed by an id name. Then, define the CSS properties within curly braces {}.

In the following example we have an <h1> element that points to the id name "myHeader". This <h1> element will be styled according to the #myHeader style definition in the head section:

```
<!DOCTYPE html>
<html>
<head>
<style>
#myHeader {
  background-color: lightblue;
  color: black;
  padding: 40px;
  text-align: center;
}
</style>
</head>
<body>

<h1 id="myHeader">My Header</h1>

</body>
</html>
```

HTML Iframes

An HTML iframe is used to display a web page within a web page.

The HTML <iframe> tag specifies an inline frame.

An inline frame is used to embed another document within the current HTML document.

```
<iframe src="url" title="description"></iframe>
```

HTML File Paths

A file path describes the location of a file in a web site's folder structure.

Path	Description
<img src="picture.jpg">	The "picture.jpg" file is located in the same folder as the current page
<img src="images/picture.jpg">	The "picture.jpg" file is located in the images folder in the current folder
<img src="/images/picture.jpg">	The "picture.jpg" file is located in the images folder at the root of the current web
<img src="../picture.jpg">	The "picture.jpg" file is located in the folder one level up from the current folder

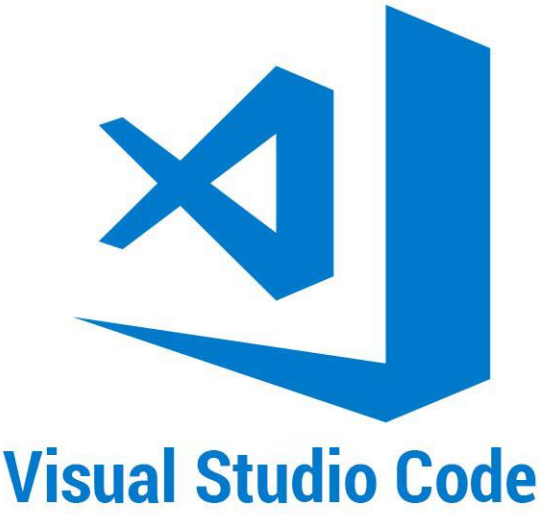
Homework/Practice Task

1. What is the purpose of using HTML tables?
2. What are the main components of an HTML table?
3. Which HTML tags are used to create rows and columns in a table?
4. What is the purpose of the <form> element in HTML?
5. Which attribute in the <input> element determines how the input will be displayed?
6. What is the difference between <input type="submit"> and <input type="button">?
7. Name five different input types supported by the HTML <input> element.
8. Which input type is used to allow users to select a date?
9. How do you create a password input field using HTML?
10. What is the purpose of the HTML element?
11. Which attributes are commonly used with the element?
12. How can a element help organize the structure of a web page?
13. What is the purpose of the HTML class attribute?
14. Can multiple HTML elements share the same class? Explain.
15. How do you select an HTML element with a specific class in CSS?

Web Design & Development with WordPress
CSS

About Lecture Sheet

CSS (Cascading Style Sheets) is a fundamental web design tool used to style and layout HTML elements. It allows developers to control the presentation of a webpage, including the design, colors, fonts, spacing, and overall layout. CSS separates the content of a website (HTML) from its design, making it easier to manage and update. In this lecture sheet, you will learn CSS from basic to advanced concepts. Initially, CSS may seem simple, but as you progress, you can explore more advanced features like Flexbox, CSS Grid, animations, transitions, and media queries for responsive design. Flexbox and Grid help in building complex layouts with ease, while media queries make websites responsive by adjusting the layout according to different screen sizes. Advanced CSS techniques like custom properties (variables) and pseudo-classes enhance flexibility and interactivity. Understanding CSS from basics to advanced concepts is essential for any web designer to create visually appealing, user-friendly, and efficient websites.



Lecture 4

CSS Introduction, CSS Syntax, CSS Selectors
How To Add, CSS Comments, CSS Colors, CSS Backgrounds

CSS Introduction

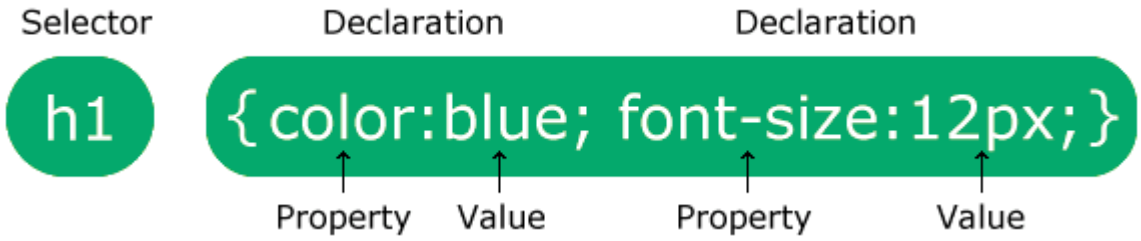
CSS is the language we use to style a Web page

What is CSS?

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

CSS Syntax

A CSS rule consists of a selector and a declaration block.



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

In this example all <p> elements will be center-aligned, with a red text color:

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

Example Explained

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

CSS Selectors

A CSS selector selects the HTML element(s) you want to style.

We can divide CSS selectors into five categories:

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

The CSS element Selector

The element selector selects HTML elements based on the element name.
Here, all <p> elements on the page will be center-aligned, with a red text color:

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

The CSS id Selector

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

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

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

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

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

The CSS class Selector

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

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

In this example all HTML elements with class="center" will be red and center-aligned:

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


The CSS Universal Selector

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

The CSS rule below will affect every HTML element on the page:

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

The CSS Grouping Selector

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

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

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

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

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

How To Add CSS

Three Ways to Insert CSS
There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS

External CSS

With an external style sheet, you can change the look of an entire website by changing just one file!
Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

External styles are defined within the <link> element, inside the <head> section of an HTML page:

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

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

</body>
</html>
```

An external style sheet can be written in any text editor, and must be saved with a .css extension.

The external .css file should not contain any HTML tags.

Here is how the "mystyle.css" file looks:

"mystyle.css"

```css
body {
  background-color: lightblue;
}

h1 {
  color: navy;
  margin-left: 20px;
}
```

Internal CSS

An internal style sheet may be used if one single HTML page has a unique style.

The internal style is defined inside the <style> element, inside the head section.

Internal styles are defined within the <style> element, inside the <head> section of an HTML page:

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

h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
<body>

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

</body>
</html>
```

Inline CSS

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

Inline styles are defined within the "style" attribute of the relevant element:

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```

CSS Comments

CSS comments are not displayed in the browser, but they can help document your source code.

Comments are used to explain the code, and may help when you edit the source code at a later date.

Comments are ignored by browsers.

A CSS comment is placed inside the <style> element, and starts with /* and ends with */:

```
/* This is a single-line comment */
p {
  color: red;
}

/* This is
a multi-line
comment */

p {
  color: red;
}
```

CSS Color

Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.

In CSS, a color can be specified by using a predefined color name:

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>
<p style="background-color:Tomato;">Lorem ipsum...</p>
```

3 Digit HEX Value

Sometimes you will see a 3-digit hex code in the CSS source.

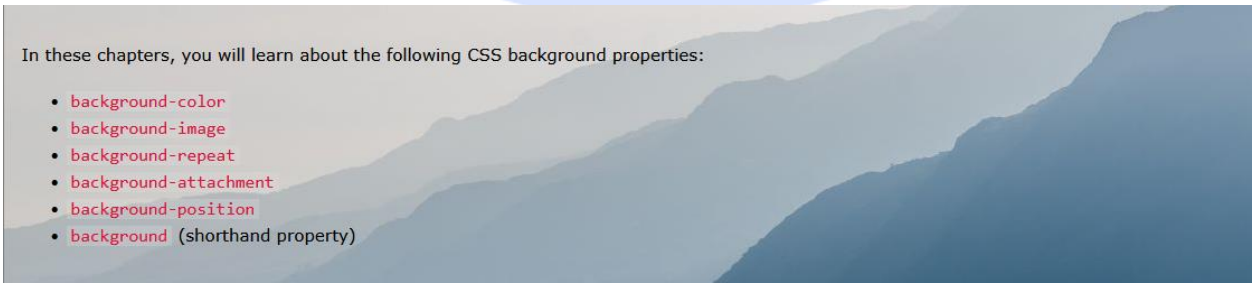
The 3-digit hex code is a shorthand for some 6-digit hex codes.

The 3-digit hex code has the following form:

```
body {
  background-color: #fc9; /* same as #ffcc99 */
}

h1 {
  color: #f0f; /* same as #ff00ff */
}

p {
  color: #b58; /* same as #bb5588 */
}
```

CSS Backgrounds

The CSS background properties are used to add background effects for elements.



CSS background-color

The background-color property specifies the background color of an element.

The background color of a page is set like this:

```
body {
  background-color: lightblue;
}
```

Here, the <h1>, <p>, and elements will have different background colors:

```
h1 {
  background-color: green;
}

div {
  background-color: lightblue;
}

p {
  background-color: yellow;
}
```

CSS Background Image

The background-image property specifies an image to use as the background of an element.

By default, the image is repeated so it covers the entire element.

```
body {
  background-image: url("paper.gif");
}
```

The background image can also be set for specific elements, like the <p> element:

```
p {
  background-image: url("paper.gif");
}
```

CSS Background Image Repeat

By default, the background-image property repeats an image both horizontally and vertically.

Some images should be repeated only horizontally or vertically, or they will look strange, like this:

```
body {
  background-image: url("gradient_bg.png");
}
```

If the image above is repeated only horizontally (background-repeat: repeat-x;), the background will look better:

```
body {
  background-image: url("gradient_bg.png");
  background-repeat: repeat-x;
}
```

Tip: To repeat an image vertically, set background-repeat: repeat-y;

CSS background-repeat: no-repeat

Showing the background image only once is also specified by the background-repeat property:

Show the background image only once:

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
}
```

CSS background-position

The background-position property is used to specify the position of the background image.

Position the background image in the top-right corner:

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
}
```

CSS background-attachment

The background-attachment property specifies whether the background image should scroll or be fixed (will not scroll with the rest of the page):

Specify that the background image should be fixed:

```css
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  background-attachment: fixed;
}
```

Specify that the background image should scroll with the rest of the page:

```css
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  background-attachment: scroll;
}
```

Homework/Practice Task

1. What does **CSS** stand for, and what is its primary purpose?
2. Explain the structure of a **CSS rule**. What are the main components?
3. How does **CSS** help in managing the layout of multiple web pages?
4. What is a **CSS selector**, and why is it important?
5. What are the five main categories of **CSS selectors**?
6. How do you select an HTML element by its **ID** in CSS? Provide an example.
7. What is the **universal selector** in CSS, and how is it used?
8. How can you group multiple selectors to apply the same style? Give an example.
9. What are the **three ways** to add CSS to a web page?
10. Describe the differences between **external**, **internal**, and **inline** CSS.
11. Where do you place the <link> element when using an **external** CSS file?
12. How do you add **comments** in CSS, and why are they useful?
13. List the different ways to specify **colors** in CSS.
14. What is a **3-digit HEX** value, and how is it different from a 6-digit HEX value?
15. What does the **background-image** property do, and how can you prevent the image from repeating?

Lecture 5

CSS Borders, CSS Margins, CSS Padding
CSS Height/Width, CSS Box Model, CSS Outline, CSS Text

CSS Borders

The CSS border properties allow you to specify the style, width, and color of an element's border.

CSS Border Style

The border-style property specifies what kind of border to display.

The following values are allowed:

- dotted - Defines a dotted border
- dashed - Defines a dashed border
- solid - Defines a solid border
- double - Defines a double border
- groove - Defines a 3D grooved border. The effect depends on the border-color value
- ridge - Defines a 3D ridged border. The effect depends on the border-color value
- inset - Defines a 3D inset border. The effect depends on the border-color value
- outset - Defines a 3D outset border. The effect depends on the border-color value
- none - Defines no border
- hidden - Defines a hidden border

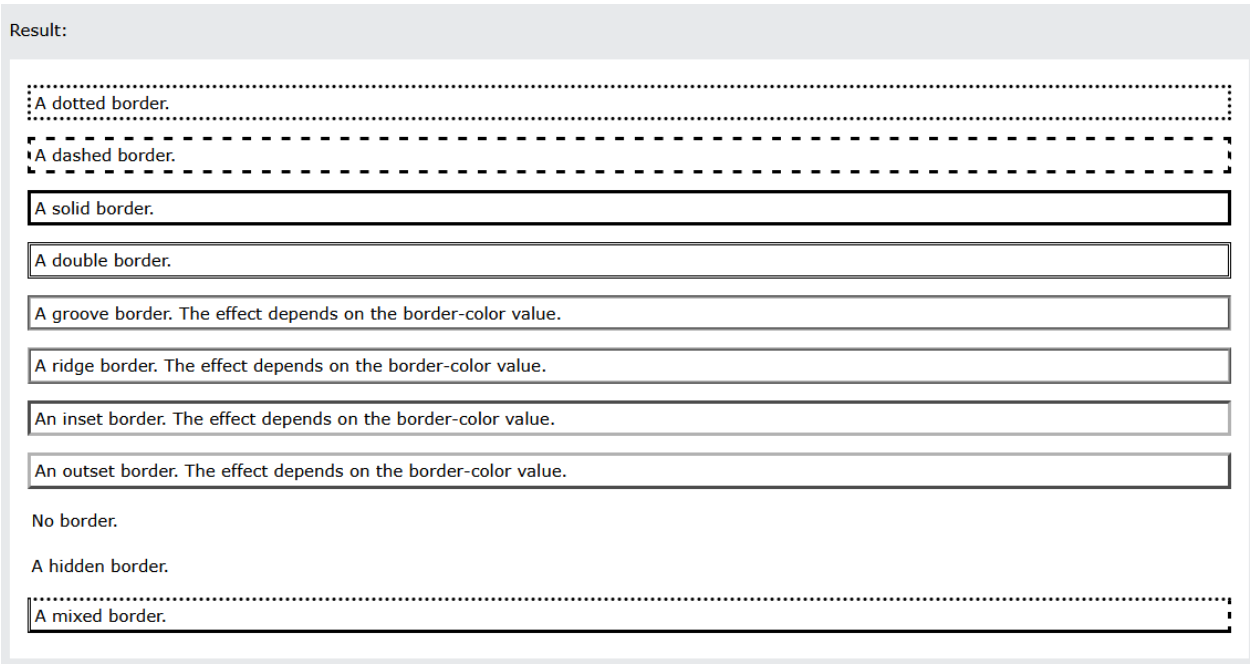
The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border).

Example

Demonstration of the different border styles:

```
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
```

Result:



CSS Border Width

The border-width property specifies the width of the four borders.

The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick:

Demonstration of the different border widths:

```
p.one {
  border-style: solid;
```

```
  border-width: 5px;
}

p.two {
  border-style: solid;
  border-width: medium;
}

p.three {
  border-style: dotted;
  border-width: 2px;
}

p.four {
  border-style: dotted;
  border-width: thick;
}
```

Result:

5px border-width

medium border-width

2px border-width

thick border-width

CSS Margins

Margins are used to create space around elements, outside of any defined borders.
The CSS margin properties are used to create space around elements, outside of any defined borders.

With CSS, you have full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom, and left).

Margin - Individual Sides
CSS has properties for specifying the margin for each side of an element:

- margin-top

- margin-right
- margin-bottom
- margin-left

All the margin properties can have the following values:

- auto - the browser calculates the margin
- length - specifies a margin in px, pt, cm, etc.
- % - specifies a margin in % of the width of the containing element
- inherit - specifies that the margin should be inherited from the parent element

Tip: Negative values are allowed.

Example

Set different margins for all four sides of a <p> element:

```
p {
  margin-top: 100px;
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 80px;
}
```

CSS Padding

Padding is used to create space around an element's content, inside of any defined borders.
The CSS padding properties are used to generate space around an element's content, inside of any defined borders.

With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).

Padding - Individual Sides

CSS has properties for specifying the padding for each side of an element:

- padding-top
- padding-right
- padding-bottom
- padding-left

All the padding properties can have the following values:

- length - specifies a padding in px, pt, cm, etc.
- % - specifies a padding in % of the width of the containing element
- inherit - specifies that the padding should be inherited from the parent element

Note: Negative values are not allowed.

Example
Set different padding for all four sides of a element:

```
div {
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
}
```

CSS Height, Width and Max-width

The CSS height and width properties are used to set the height and width of an element.

The CSS max-width property is used to set the maximum width of an element.

CSS Setting height and width

The height and width properties are used to set the height and width of an element.

The height and width properties do not include padding, borders, or margins. It sets the height/width of the area inside the padding, border, and margin of the element.

CSS height and width Values

The height and width properties may have the following values:

- auto - This is default. The browser calculates the height and width
- length - Defines the height/width in px, cm, etc.
- % - Defines the height/width in percent of the containing block
- initial - Sets the height/width to its default value
- inherit - The height/width will be inherited from its parent value

CSS height and width Examples

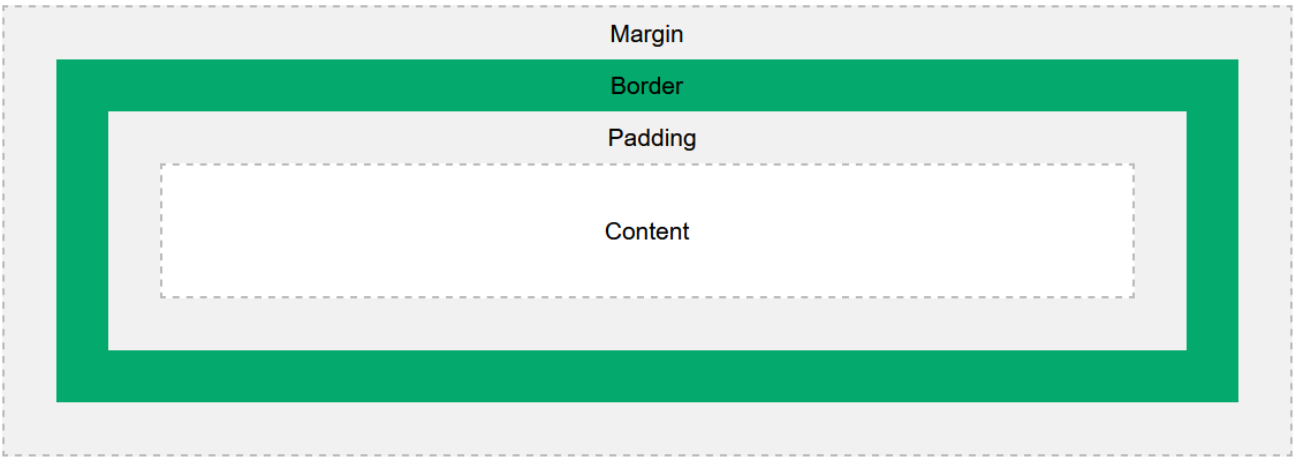
This element has a height of 200 pixels and a width of 50%

```
div {
  height: 200px;
  width: 50%;
  background-color: powderblue;
}
```

CSS Box Model

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: content, padding, borders and margins. The image below illustrates the box model:



Explanation of the different parts:

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

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

CSS Outline

An outline is a line drawn outside the element's border.

Demonstration of the different outline styles:

```
p.dotted {outline-style: dotted;}
p.dashed {outline-style: dashed;}
p.solid {outline-style: solid;}
p.double {outline-style: double;}
p.groove {outline-style: groove;}
p.ridge {outline-style: ridge;}
p.inset {outline-style: inset;}
p.outset {outline-style: outset;}
```

Result:

A dotted outline.

A dashed outline.

A solid outline.

A double outline.

A groove outline. The effect depends on the outline-color value.

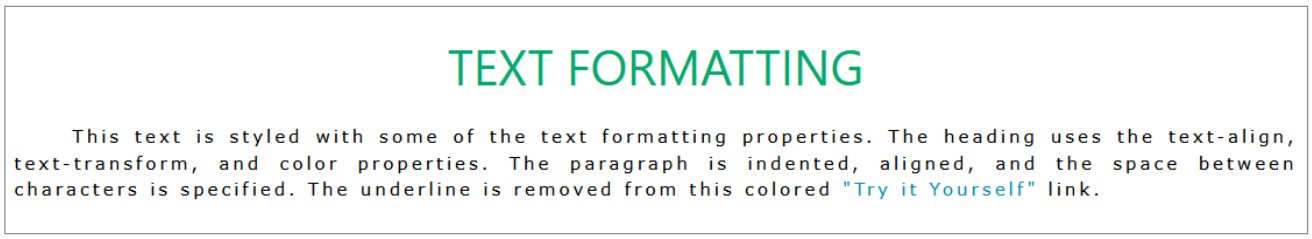
A ridge outline. The effect depends on the outline-color value.

An inset outline. The effect depends on the outline-color value.

An outset outline. The effect depends on the outline-color value.

CSS Text

CSS has a lot of properties for formatting text.



Text Color

The color property is used to set the color of the text. The color is specified by:

CSS Text Alignment

Text Alignment and Text Direction

In this chapter you will learn about the following properties:

- text-align
- text-align-last
- direction
- unicode-bidi
- vertical-align

Text Alignment

The text-align property is used to set the horizontal alignment of a text.
A text can be left or right aligned, centered, or justified.

The following example shows center aligned, and left and right aligned text (left alignment is default if text direction is left-to-right, and right alignment is default if text direction is right-to-left):

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

h2 {
  text-align: left;
}

h3 {
  text-align: right;
}
```

Output

When the text-align property is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers):

```
div {
  text-align: justify;
}
```

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown

printer took a galley of type and scrambled it to make a type specimen book. It has

CSS Text Decoration

In this chapter you will learn about the following properties:

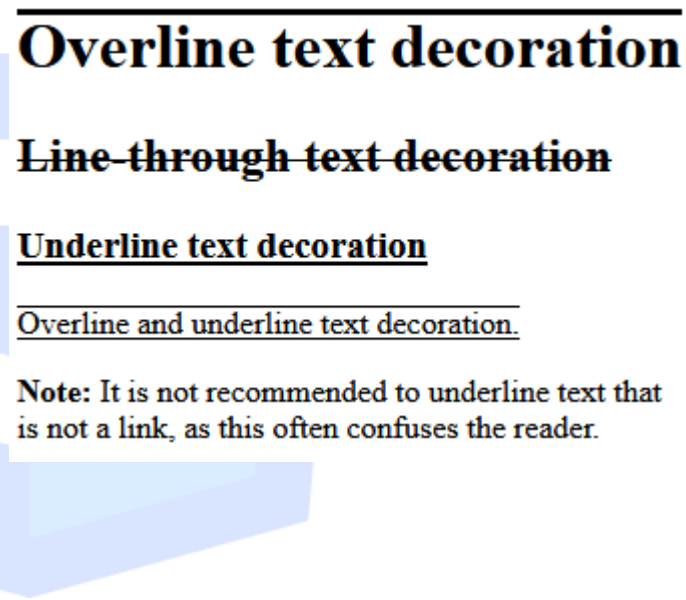
- text-decoration-line
- text-decoration-color
- text-decoration-style
- text-decoration-thickness
- text-decoration

Add a Decoration Line to Text

The text-decoration-line property is used to add a decoration line to text.

Tip: You can combine more than one value, like overline and underline to display lines both over and under a text.

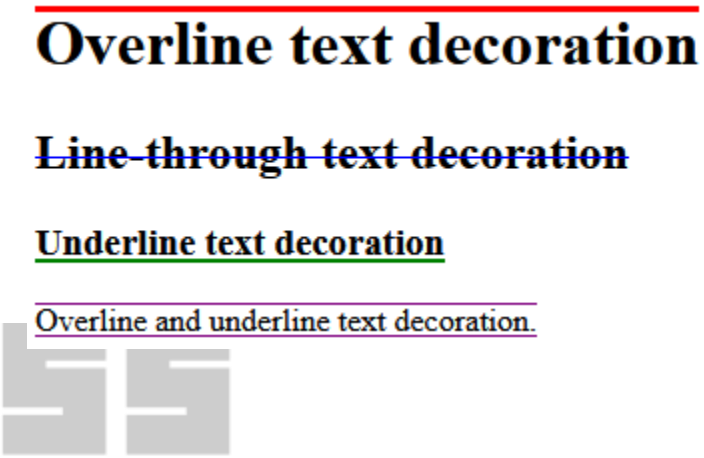
```
h1 {
  text-decoration-line: overline;
}

h2 {
  text-decoration-line: line-through;
}

h3 {
  text-decoration-line: underline;
}

p {
  text-decoration-line: overline underline;
}
```



Specify a Color for the Decoration Line

The text-decoration-color property is used to set the color of the decoration line.

```
h1 {
  text-decoration-line: overline;
  text-decoration-color: red;
}

h2 {
  text-decoration-line: line-through;
  text-decoration-color: blue;
}

h3 {
  text-decoration-line: underline;
  text-decoration-color: green;
}

p {
  text-decoration-line: overline underline;
  text-decoration-color: purple;
}
```



CSS Text Transformation

The text-transform property is used to specify uppercase and lowercase letters in a text.

It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word:

Example

```
p.uppercase {
  text-transform: uppercase;
}

p.lowercase {
  text-transform: lowercase;
}

p.capitalize {
  text-transform: capitalize;
}
```

Text Spacing

In this chapter you will learn about the following properties:

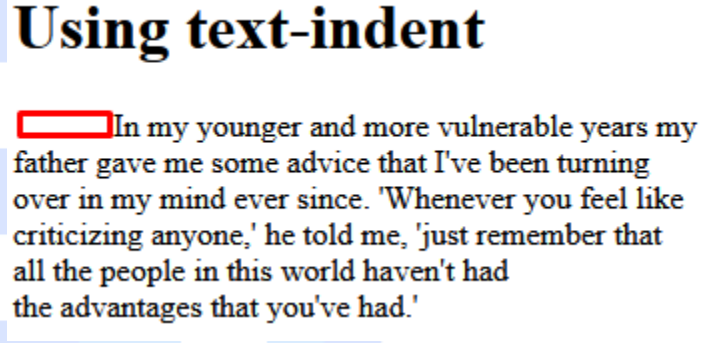
- text-indent
- letter-spacing
- line-height
- word-spacing
- white-space

Text Indentation

The text-indent property is used to specify the indentation of the first line of a text:

Example

```
p {
  text-indent: 50px;
}
```



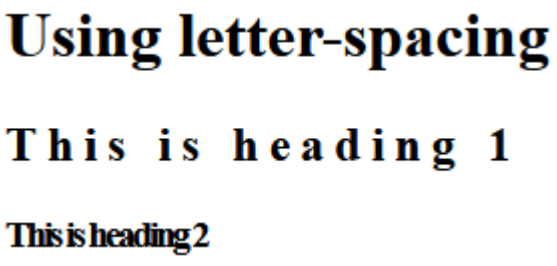
Letter Spacing

The letter-spacing property is used to specify the space between the characters in a text.

The following example demonstrates how to increase or decrease the space between characters:

E***xample***

```
h1 {
  letter-spacing: 5px;
}

h2 {
  letter-spacing: -2px;
}
```



Text Shadow

The text-shadow property adds shadow to text.

In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px):

Example

```
h1 {
  text-shadow: 2px 2px;
}
```

Text-shadow effect!

Homework/Practice Task

1. **Border Styles**
   Create a with the following properties:
   - Width: 300px, Height: 150px
   - Solid border on the top, dashed on the right, dotted on the bottom, and double on the left.
   - Border color: blue.
2. **Border Width**
   Write CSS to create three paragraphs (<p>) with:
   - First paragraph: solid border with a width of 10px.
   - Second paragraph: dotted border with a medium width.
   - Third paragraph: groove border with a thick width.
3. **Margins**
   Create a that has the following margin properties:
   - 50px top margin
   - 20px right margin
   - 30px bottom margin
   - auto left margin (center horizontally).
4. **Padding**
   Write CSS to add **different padding** for all four sides of a box:
   - Top: 15px, Right: 25px, Bottom: 35px, Left: 45px.
5. **Height, Width, and Max-Width**
   Create a responsive <section> with:
   - Height: 400px
   - Width: 80%
   - Maximum Width: 1200px
   - Background color: lightgreen.
6. **CSS Box Model**
   Create a **box** with the following properties:
   - Content width: 300px

- Padding: 20px
- Border: 5px solid black
- Margin: 30px.

7. **Outline**
   Create a paragraph with:
   - dashed outline of 4px
   - Outline color: red.
8. **Text Alignment**
   Create three paragraphs with the following alignments:
   - First paragraph: left aligned (default)
   - Second paragraph: center aligned
   - Third paragraph: right aligned.
9. **Text Decoration**
   Write CSS to create a heading (<h1>) with:
   - underline and overline decoration
   - Decoration color: purple
   - Decoration thickness: 3px.
10. **Text Shadow**
    Create a <p> with the text "Hello European IT" and apply the following shadow:

- Horizontal offset: 3px
- Vertical offset: 3px
- Shadow color: gray.

Lecture 6

CSS Fonts, CSS Icons, CSS Display, CSS Position, Z-index

CSS Fonts

Choosing the right font has a huge impact on how the readers experience a website.

The right font can create a strong identity for your brand.

Using a font that is easy to read is important. The font adds value to your text. It is also important to choose the correct color and text size for the font.

How to add Google Font

✅ Method 1: Using <link> in HTML

1. Go to **Google Fonts**.
2. Choose a font and click **"Select"** or **"Get Font"**.
3. Copy the <link> code.
4. Paste it inside the <head> section of your HTML file.

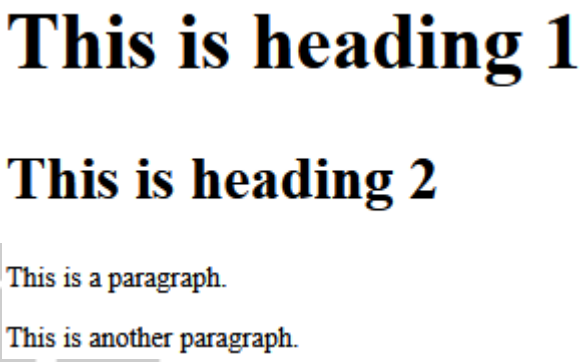
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Google Fonts Example</title>

    <!-- Google Font Link -->
    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap" rel="stylesheet">

    <style>
        body {
            font-family: 'Poppins', sans-serif;
        }
    </style>

</head>
<body>
    <h1>Hello, European IT!</h1>
</body>
</html>
```

CSS Font Size

The font-size property sets the size of the text.

Example

```
h1 {
  font-size: 40px;
}

h2 {
  font-size: 30px;
}

p {
  font-size: 14px;
}
```

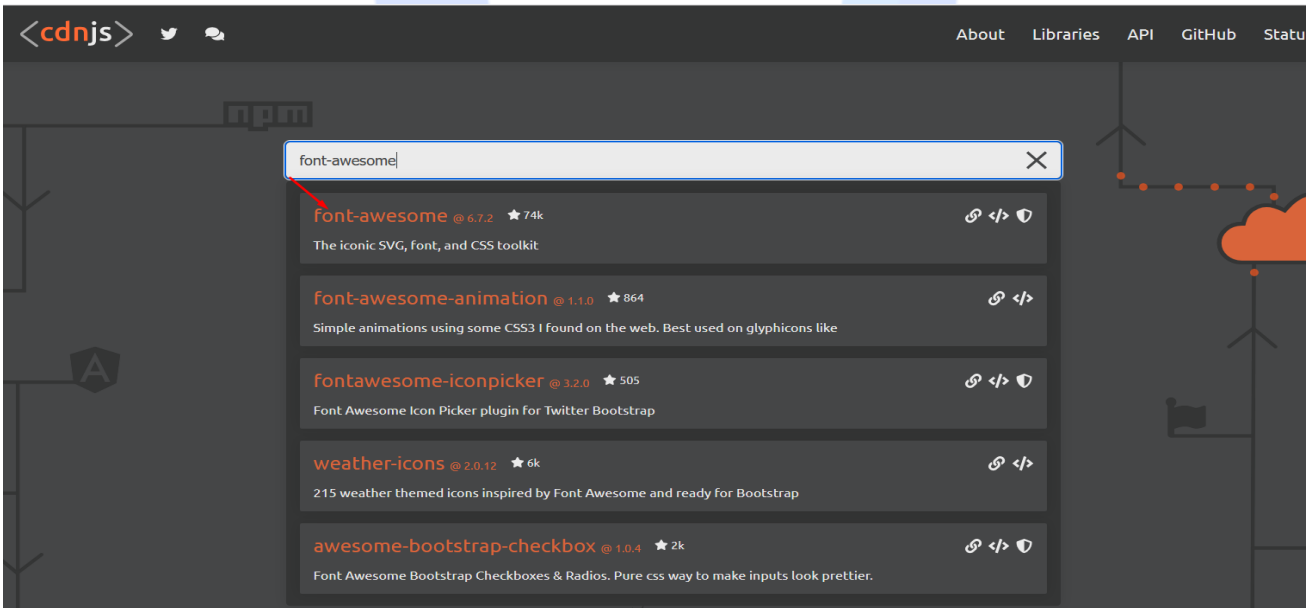


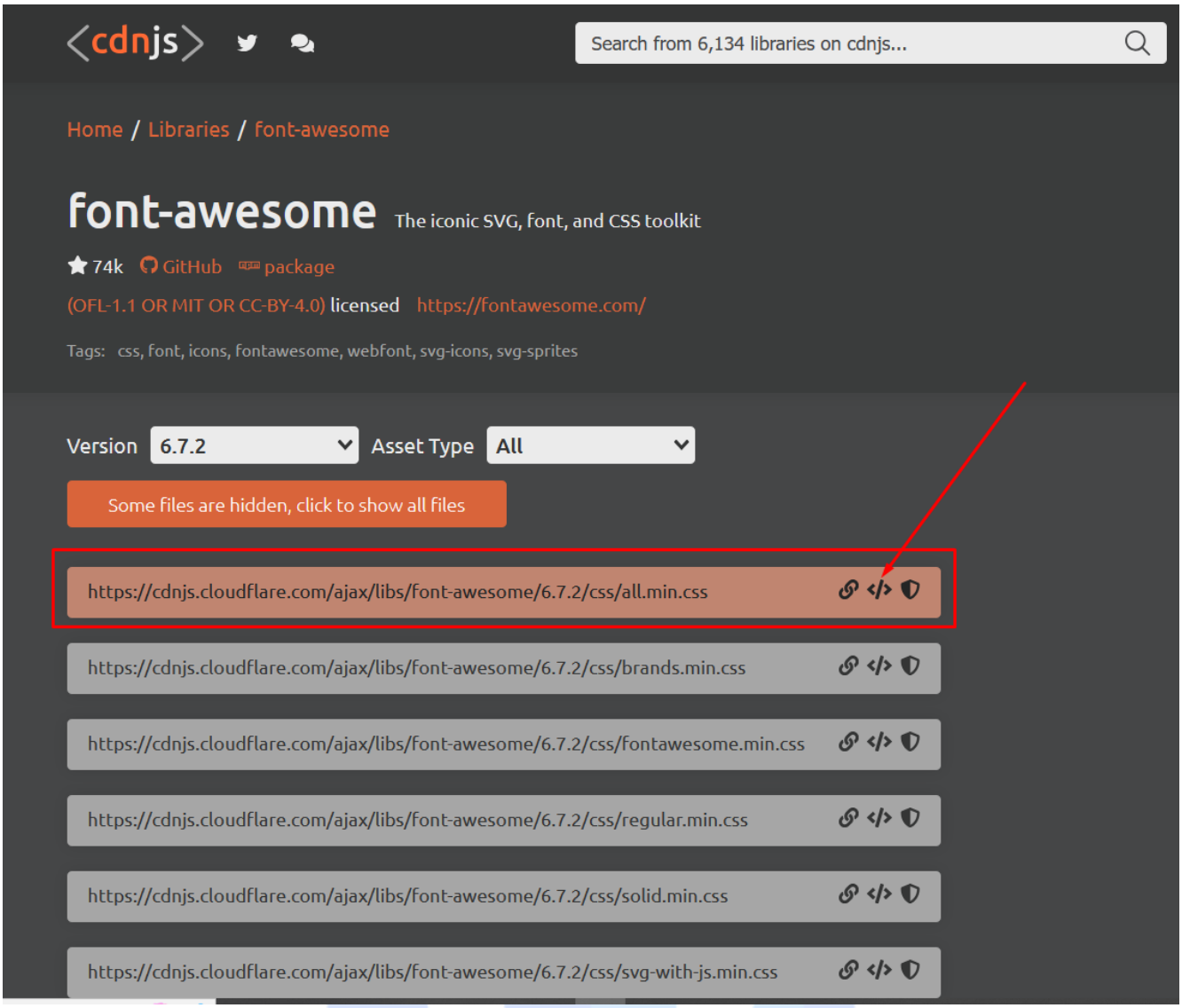
CSS Icons

Icons can easily be added to your HTML page, by using an icon library.

How To Add Icons

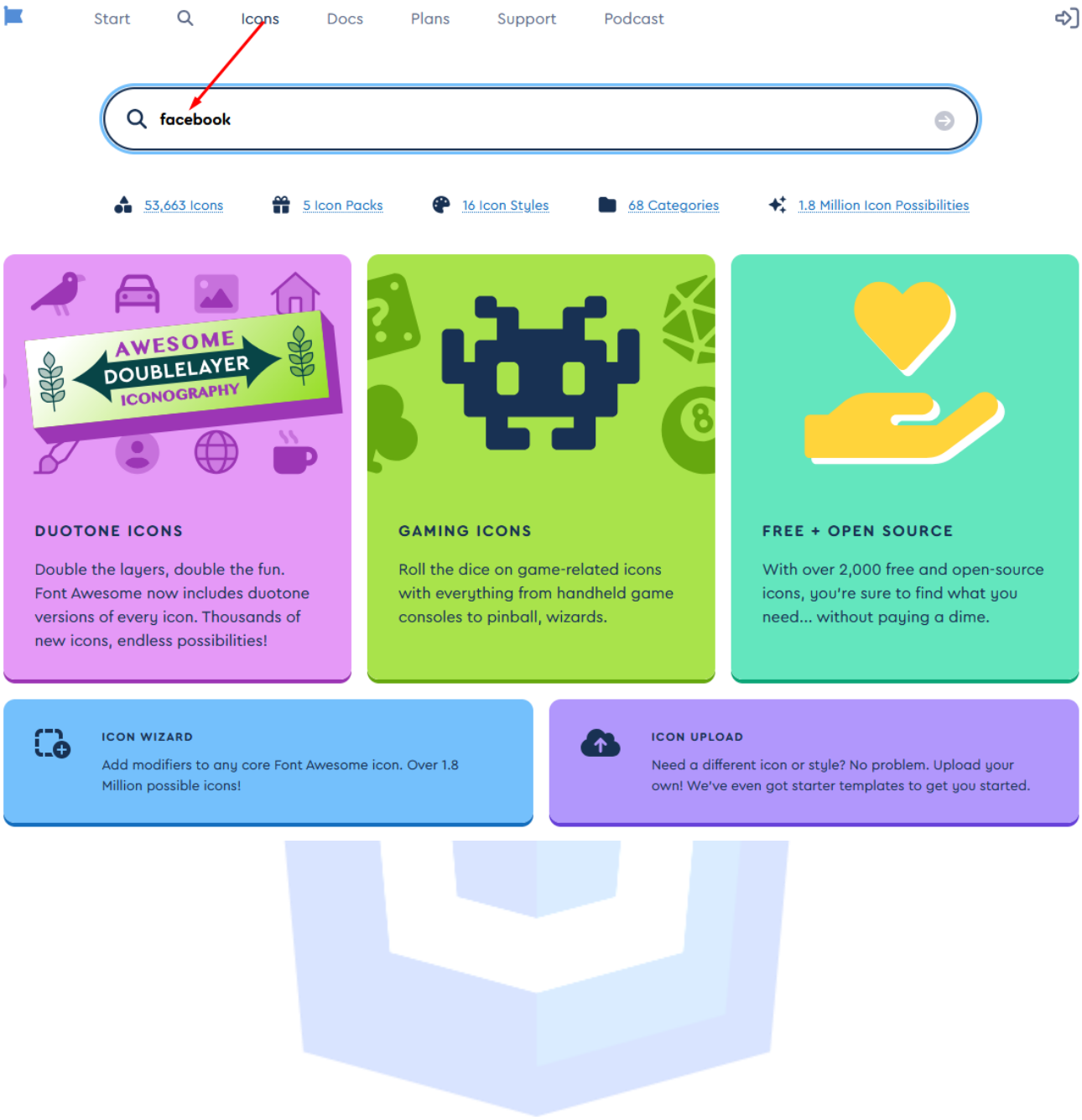
Go to the cdnjs.com website and search for Font Awesome in the search bar. Then, copy the link and paste it into the <head> section."

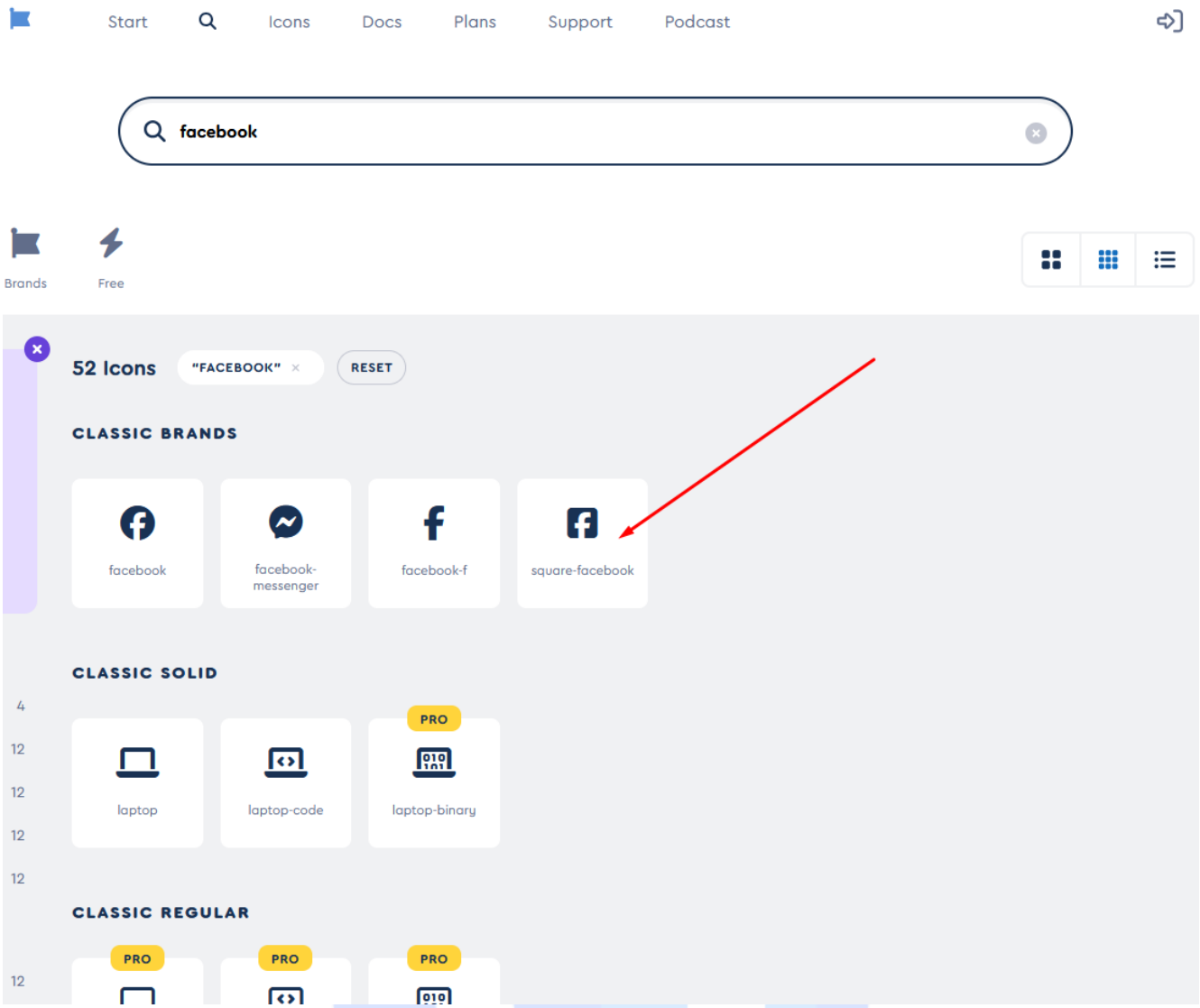




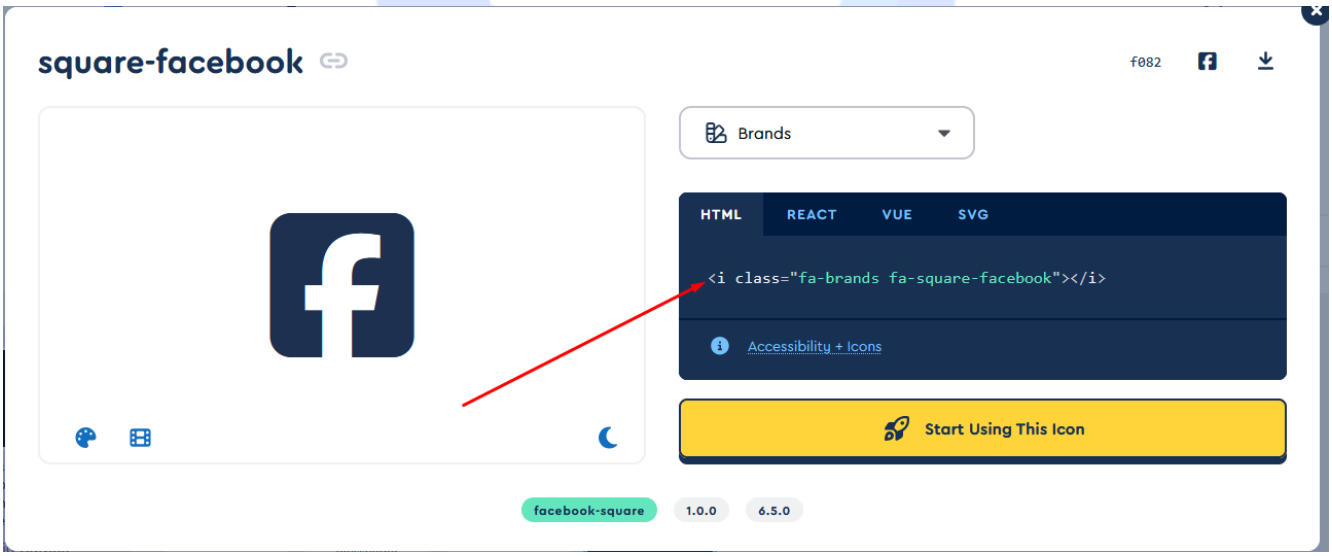

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Google Fonts Example</title>

    <!-- Google Font Link -->
    <link href="https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap" rel="stylesheet">

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css"
    integrity="sha512-Evv84Mr4kqVGRNSgIGL/F/aIDqQb7xQ2vcrdIwxfjThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9ABhg=="
    crossorigin="anonymous" referrerpolicy="no-referrer" />

    <style>
        body {
            font-family: 'Poppins', sans-serif;
        }
    </style>

</head>
<body>
    <h1>Hello, European IT!</h1>
</body>
</html>
```

Now go to fontawesome.com, click on the 'Icons' menu, and then search using the search bar.





Click on your preferred icon, copy the HTML code, and paste it into your website.



```html
<html lang="en">
<head>

    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css"
    integrity="sha512-Evv84Mr4kqVGRNSgIGL/F/aIDqQb7xQ2vcrdIwxfjThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9ABhg=="
    crossorigin="anonymous" referrerpolicy="no-referrer" />

    <style>
        body {
            font-family: 'Poppins', sans-serif;
        }
    </style>

</head>
<body>
    <h1>Hello, European IT!</h1>

    <i class="fa-brands fa-square-facebook"></i>

</body>
</html>
```

Hello, European IT!

CSS Layout - The display Property

The display property is the most important CSS property for controlling layout.

The display property is used to specify how an element is shown on a web page.

Every HTML element has a default display value, depending on what type of element it is. The default display value for most elements is block or inline.

The display property is used to change the default display behavior of HTML elements.

Block-level Elements

A block-level element ALWAYS starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The element is a block-level element.

Examples of block-level elements:

-
- <h1> - <h6>
- <p>
- <form>
- <header>
- <footer>
- <section>

Inline Elements

An inline element DOES NOT start on a new line and only takes up as much width as necessary.

This is an inline element inside a paragraph.

Examples of inline elements:

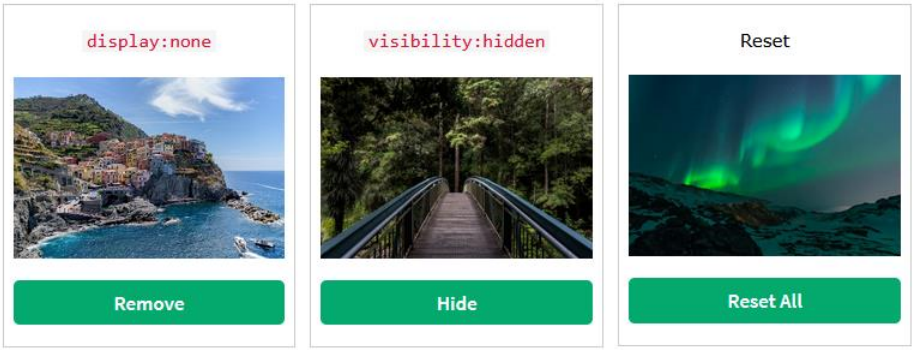
-
- <a>
- <img>

The display Property Values

The display property has many values:

Value	Description
inline	Displays an element as an inline element
block	Displays an element as a block element
contents	Makes the container disappear, making the child elements children of the element the next level up in the DOM
flex	Displays an element as a block-level flex container
grid	Displays an element as a block-level grid container
inline-block	Displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values
inline-flex	Displays an element as an inline-level flex container
inline-grid	Displays an element as an inline-level grid container
inline-table	The element is displayed as an inline-level table
list-item	Let the element behave like a element
run-in	Displays an element as either block or inline, depending on context
table	Let the element behave like a <table> element
table-caption	Let the element behave like a <caption> element
table-column-group	Let the element behave like a <colgroup> element
table-header-group	Let the element behave like a <thead> element
table-footer-group	Let the element behave like a <tfoot> element
table-row-group	Let the element behave like a <tbody> element
table-cell	Let the element behave like a <td> element
table-column	Let the element behave like a <col> element
table-row	Let the element behave like a <tr> element
none	The element is completely removed
initial	Sets this property to its default value
inherit	Inherits this property from its parent element

Hide an Element - display:none or visibility:hidden?



Hiding an element can be done by setting the `display` property to `none`. The element will be hidden, and the page will be displayed as if the element is not there:

CSS Position

The position property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

There are five different position values:

- static
- relative
- fixed
- absolute
- sticky

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

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:

This element has position: static;

Here is the CSS that is used:

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

position: relative;

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

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

This element has position: relative;

Here is the CSS that is used:

```
div.relative {
  position: relative;
  left: 30px;
  border: 3px solid #73AD21;
}
```

position: fixed;

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

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

Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:

```
div.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
```

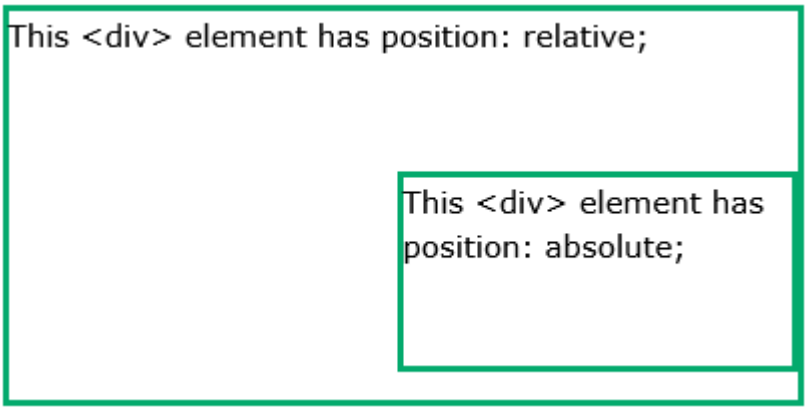
position: absolute;

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

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

Note: Absolute positioned elements are removed from the normal flow, and can overlap elements.

Here is a simple example:



Here is the CSS that is used:

```css
div.relative {
  position: relative;
  width: 400px;
  height: 200px;
  border: 3px solid #73AD21;
}

div.absolute {
  position: absolute;
  top: 80px;
  right: 0;
  width: 200px;
  height: 100px;
  border: 3px solid #73AD21;
}
```

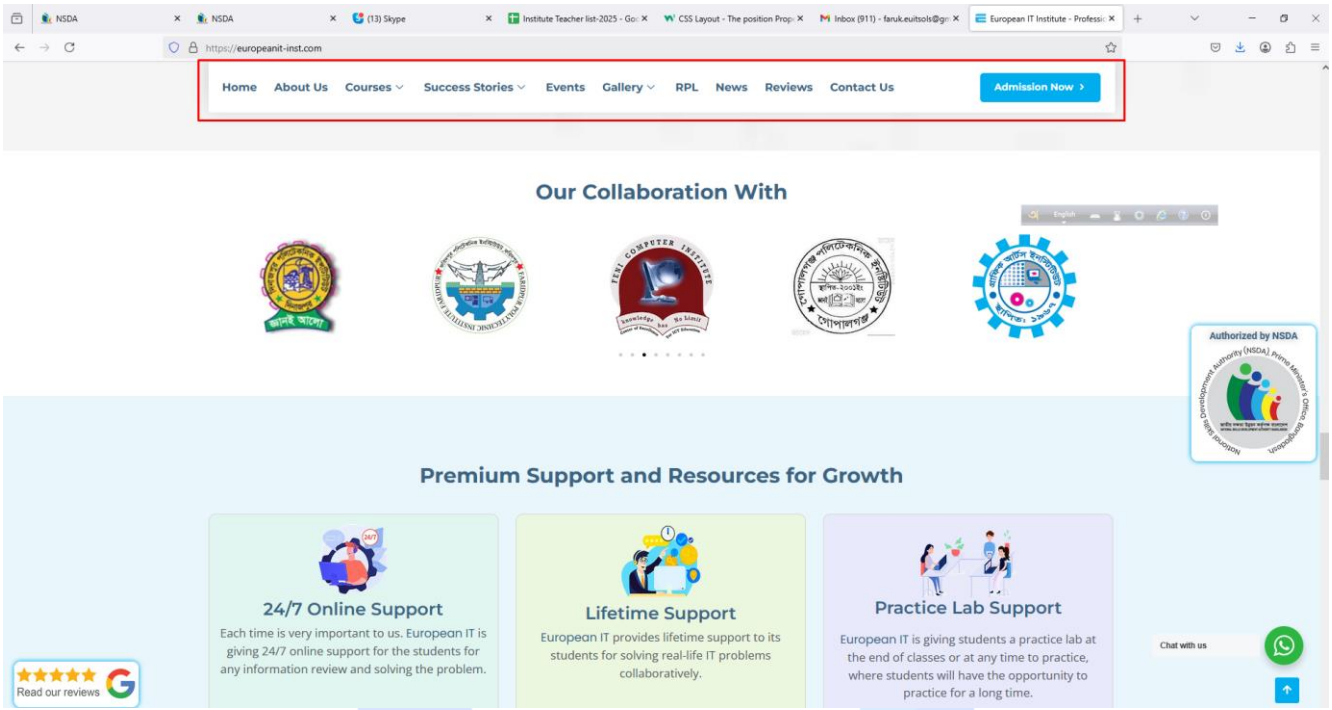
position: sticky;

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

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

Note: You must specify at least one of top, right, bottom or left for sticky positioning to work.

```css
div.sticky {
  position: sticky;
  top: 0;
  background-color: green;
  border: 2px solid #4CAF50;
}
```

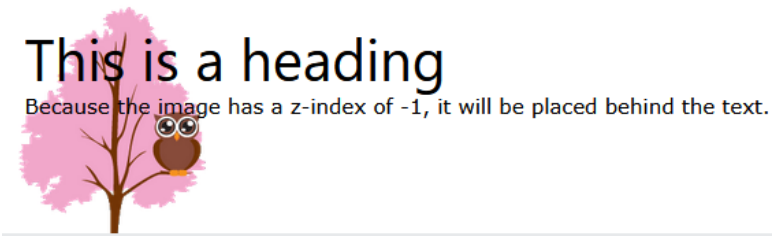


The z-index Property

When elements are positioned, they can overlap other elements.

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

An element can have a positive or negative stack order:




```css
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
}
```

Homework/Practice Task

1. What is the difference between block-level and inline elements? Provide two examples of each.
2. Explain the purpose of the position: relative; property and how it affects the placement of an element.
3. What is the z-index property, and how does it affect overlapping elements on a webpage?
4. Describe the difference between position: absolute; and position: fixed; with examples.
5. What are the steps to add Google Fonts to an HTML document using the <link> method?
   - Google Font Integration:
6. Add the "Roboto" font from Google Fonts to your webpage and apply it to all paragraph (<p>) elements.
   - Icon Implementation:
7. Integrate Font Awesome into your webpage and display a "home" icon.
   - CSS Layout - Display Property:
8. Create a webpage where:

   A element is displayed as a block.

   An <a> element is displayed as an inline-block.
   - CSS Positioning:
9. Create a fixed navigation bar at the top of the page using position: fixed;. Ensure it remains visible while scrolling.
   - z-index Demonstration:
10. Create two overlapping elements where the top element is placed behind the bottom one using z-index.

Web Design & Development with WordPress
Project 1 Using HTML & CSS

About Lecture Sheet

In this project, we will design and develop a website named "Silver Hawk" using HTML and CSS, focusing on converting a PSD design into a fully responsive webpage. Through this project, we will learn essential techniques such as integrating Google Fonts, using Font Awesome icons, and applying advanced CSS properties like display, position, and z-index. By the end of this project, you will have a comprehensive understanding of converting PSD to HTML while mastering key CSS concepts, including block and inline elements, layout management, and positioning methods to create professional and visually appealing web pages.

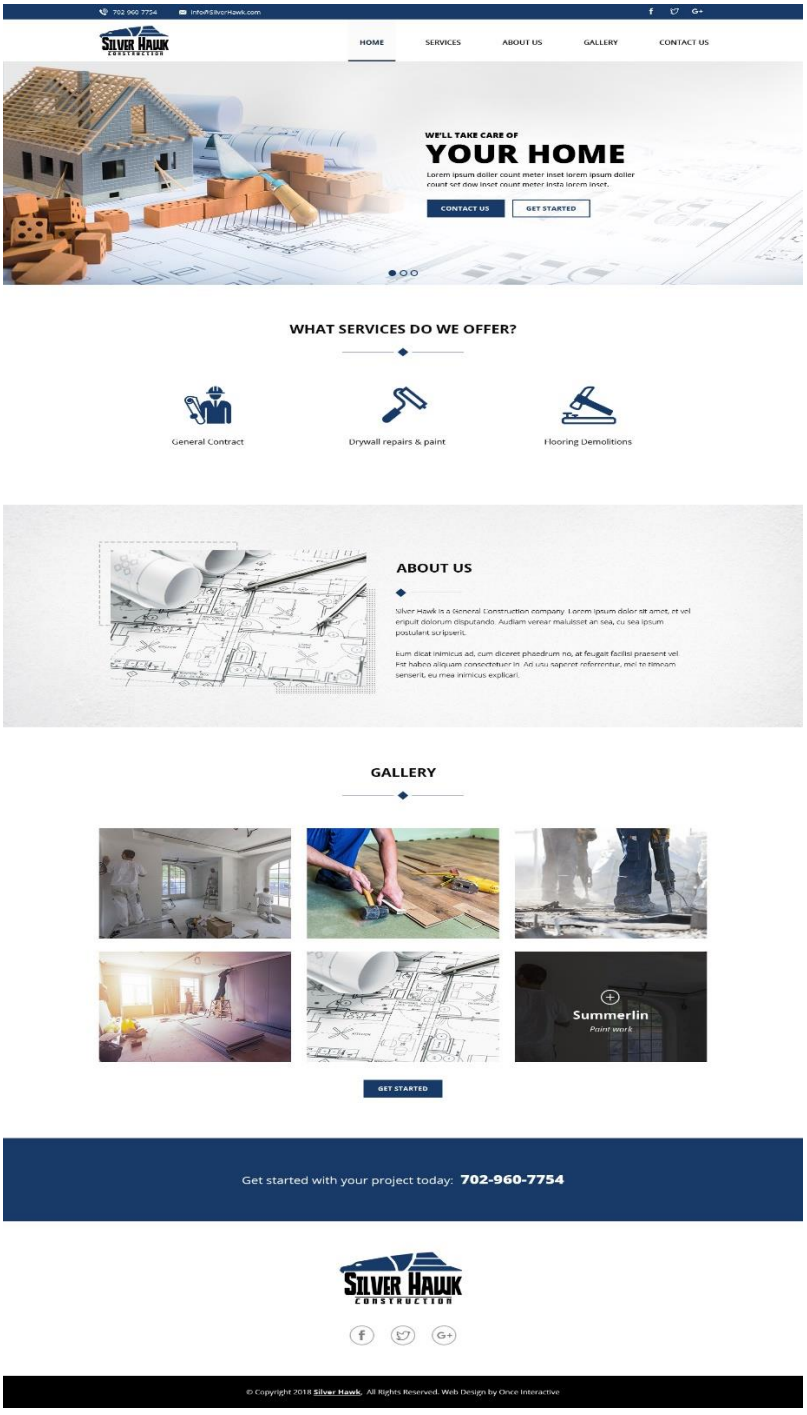
Lecture 7

Project Name: Silver Hawk

Technology: HTML & CSS | Tools: Notepad++

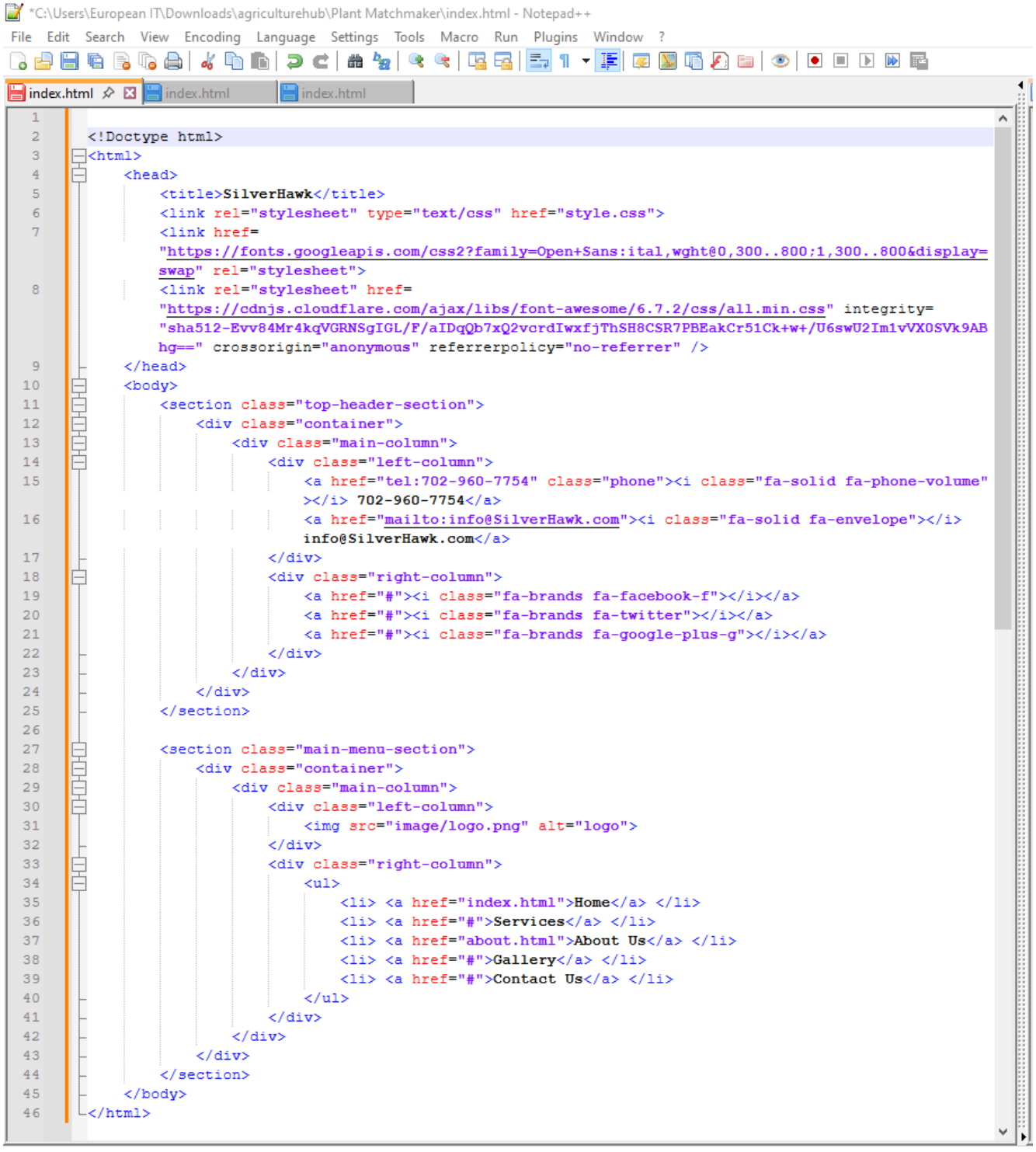
Click & Download the PSD File - https://shorturl.at/Z3Orq

PSD Layout



Open Notepad++ or VS Code and start coding.

HTML Code




```
<!Doctype html>
<html>
    <head>
        <title>SilverHawk</title>
        <link rel="stylesheet" type="text/css" href="style.css">
        <link href=
        "https://fonts.googleapis.com/css2?family=Open+Sans:ital,wght@0,300..800;1,300..800&display=
        swap" rel="stylesheet">
        <link rel="stylesheet" href=
        "https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.7.2/css/all.min.css" integrity=
        "sha512-Evv84Mr4kqVGRNSgIGL/F/aIDqQb7xQ2vcrdIwxfjThSH8CSR7PBEakCr51Ck+w+/U6swU2Im1vVX0SVk9AB
        hg==" crossorigin="anonymous" referrerpolicy="no-referrer" />
    </head>
    <body>
        <section class="top-header-section">
            <div class="container">
                <div class="main-column">
                    <div class="left-column">
                        <a href="tel:702-960-7754" class="phone"><i class="fa-solid fa-phone-volume"
                        ></i> 702-960-7754</a>
                        <a href="mailto:info@SilverHawk.com"><i class="fa-solid fa-envelope"></i>
                        info@SilverHawk.com</a>
                    </div>
                    <div class="right-column">
                        <a href="#"><i class="fa-brands fa-facebook-f"></i></a>
                        <a href="#"><i class="fa-brands fa-twitter"></i></a>
                        <a href="#"><i class="fa-brands fa-google-plus-g"></i></a>
                    </div>
                </div>
            </div>
        </section>

        <section class="main-menu-section">
            <div class="container">
                <div class="main-column">
                    <div class="left-column">
                        <img src="image/logo.png" alt="logo">
                    </div>
                    <div class="right-column">
                        <ul>
                            <li> <a href="index.html">Home</a> </li>
                            <li> <a href="#">Services</a> </li>
                            <li> <a href="about.html">About Us</a> </li>
                            <li> <a href="#">Gallery</a> </li>
                            <li> <a href="#">Contact Us</a> </li>
                        </ul>
                    </div>
                </div>
            </div>
        </section>
    </body>
</html>
```

CSS Code

style.css

```css
.container{
    max-width:1140px;
    margin:0 auto;

}
*{
    font-family: "Open Sans", serif;
    margin:0;
    padding:0;
}

.top-header-section .main-column{
    display: grid;
    grid-template-columns:1fr 1fr;
    column-gap:5px;
}

.top-header-section {
    background: #173a69;
    padding-top: 6px;
    padding-bottom: 6px;
}

.top-header-section a{
    color:#fff;
    text-decoration: none;
    font-size:13px;
}

.top-header-section .phone{
    margin-right:40px;
}

.top-header-section .left-column i{
    padding-right:5px;
}

.top-header-section .right-column{
    text-align:right;
}

.top-header-section .fa-twitter{
    padding-left:10px;
    padding-right:10px;
}
```

style.css

```css
.main-menu-section .main-column{
    display: grid;
    grid-template-columns: 1fr 5fr;
    column-gap: 10px;
    align-items: center;
}

.main-menu-section li{
    list-style: none;
    display: inline-block;
    padding-right:50px;
}

.main-menu-section li:last-child{
    padding-right:0;
}

.main-menu-section a{
    text-decoration: none;
    color:#000000;
    font-weight:600;
    text-transform: uppercase;
}

.main-menu-section .right-column{
    text-align: right;
}

.main-menu-section{
    padding-top: 10px;
    padding-bottom: 10px;
}
```

Output

Lecture 8

Project Name: Silver Hawk

Technology: HTML & CSS | Tools: Notepad++

HTML Code

Slider Section

```
<section class="slider-section">
    <div class="slider">
        <div class="item">
            <img src="image/slider.jpg">
        </div>
    </div>
</section>
```

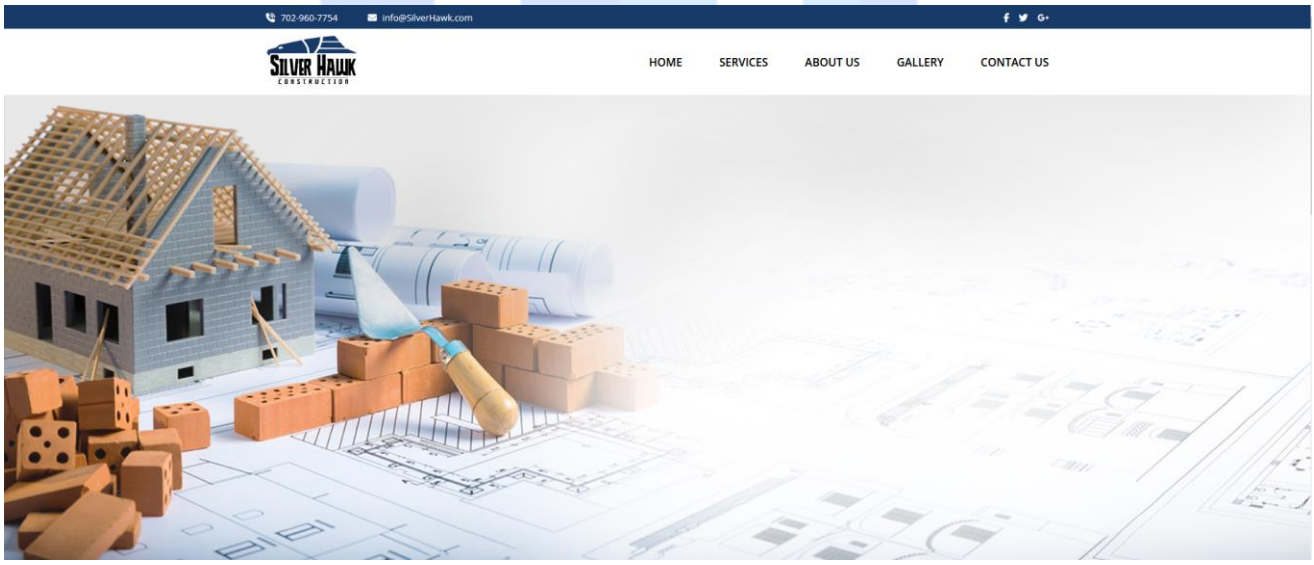
CSS Code

Slider section

```
.slider-section img{
    width:100%;
}
```

Output

Slider section



HTML Code

Service Section

```
<section class="service-section">
    <div class="container">
        <h2>What services do WE offer?</h2>
        <img src="image/heading-icon.png">
        <div class="main-column">

            <div class="flex-column">
                <img src="image/general.png">
                <p>General Contract</p>
            </div>
            <div class="flex-column">
                <img src="image/dry.png">
                <p>Drywall repairs & paint</p>
            </div>
            <div class="flex-column">
                <img src="image/general.png">
                <p>General Contract</p>
            </div>
        </div>
    </div>
</section>
```

CSS Code

Service Section

```
.service-section .main-column{
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    text-align: center;
    padding-top: 80px;
}
.service-section{
    text-align: center;
    padding-top:80px;
    padding-bottom:80px;
}
.service-section h2{
    font-size: 28px;
    color: #090909;
    text-transform: uppercase;
    padding-bottom: 20px;
}
```

Output

Service Section

HTML Code

About Section

```
<section class="about-section">
    <div class="container">
        <div class="main-column">
            <div class="flex-column image-column">
                <img src="image/about.png">
            </div>
            <div class="flex-column">
                <h2>About Us</h2>
                <img src="image/about-icon.png">
                <p>Silver Hawk is a General Construction company. Lorem ipsum dolor sit amet, et vel eripuit dolorum disputando. Audiam verear maluisset an sea, cu sea ipsum postulant scripserit. </p>
                <p>Eum dicat inimicus ad, cum diceret phaedrum no, at feugait facilisi praesent vel. Est habeo aliquam consectetuer in. Ad usu saperet referrentur, mel te timeam senserit, eu mea inimicus explicari.</p>
            </div>
        </div>
    </div>
</section>
```

CSS Code

About Section

```css
.about-section .main-column {
    display: grid;
    grid-template-columns: 1fr 1fr;
    column-gap: 40px;
    align-items: center;
}
.about-section .image-column img{
    width:100%;
}
.about-section p {
    padding-bottom: 40px;
    line-height: 25px;
    padding-top: 20px;
}
.about-section p:last-child{
    padding-bottom:0;
    padding-top:0;
}
.about-section h2{
    font-size: 28px;
  color: #090909;
  text-transform: uppercase;
  padding-bottom: 20px;
}

.about-section{
    background: url(image/about-bg.jpg);
    background-repeat: no-repeat;
    background-size:100%;
    padding:80px 0px;
}
```

Output

About Section

Lecture 9

Project Name: Silver Hawk Page: Home

Technology: HTML & CSS | Tools: Notepad++/VS Code

HTML Code

Gallery Section

```
<section class="gallery-section">
    <h2>Gallery</h2>
    <div class="container">
        <div class="main-column">
            <div class="flex-column">
                <img src="image/gallery.jpg">
                <div class="caption">
                    <img src="image/plus-icon.png">
                    <h3>Summerlin</h3>
                    <p>Paint work</p>
                </div>
            </div>
            <div class="flex-column">
                <img src="image/gallery.jpg">
                <div class="caption">
                    <img src="image/plus-icon.png">
                    <h3>Summerlin</h3>
                    <p>Paint work</p>
                </div>
            </div>
            <div class="flex-column">
                <img src="image/gallery.jpg">
                <div class="caption">
                    <img src="image/plus-icon.png">
                    <h3>Summerlin</h3>
                    <p>Paint work</p>
                </div>
            </div>
            <div class="flex-column">
                <img src="image/gallery.jpg">
                <div class="caption">
                    <img src="image/plus-icon.png">
                    <h3>Summerlin</h3>
                    <p>Paint work</p>
                </div>
            </div>
            <div class="flex-column">
                <img src="image/gallery.jpg">
                <div class="caption">
                    <img src="image/plus-icon.png">
                    <h3>Summerlin</h3>
                    <p>Paint work</p>
                </div>
            </div>
            <div class="flex-column">
                <img src="image/gallery.jpg">
                <div class="caption">
                    <img src="image/plus-icon.png">
                    <h3>Summerlin</h3>
                    <p>Paint work</p>
                </div>
            </div>
        </div>
        <a href="#">Get Started</a>
    </div>
</section>
```

CSS Code

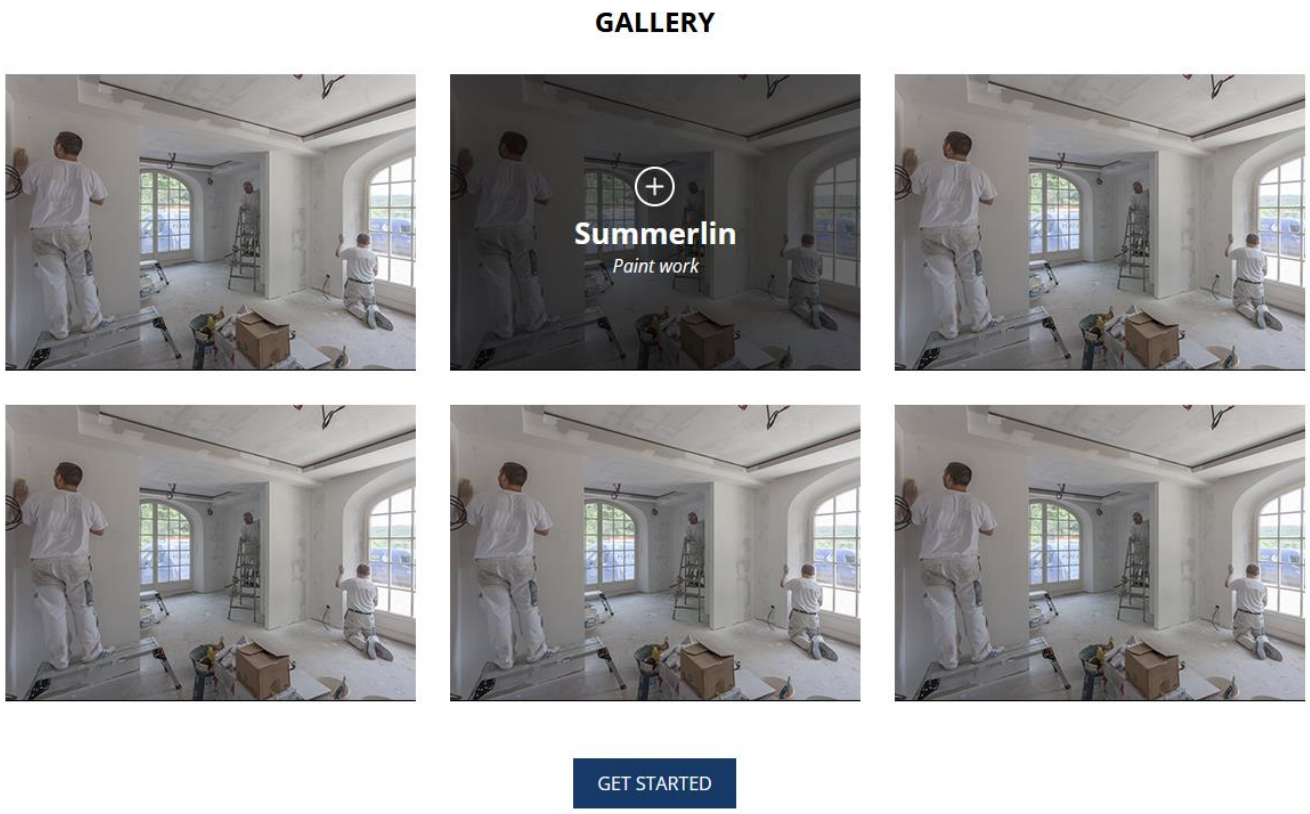
Gallery Section

```css
.gallery-section{
    padding-top: 90px;
    padding-bottom:90px;
}

.gallery-section .main-column{
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    column-gap:30px;
    row-gap:30px;
}

.gallery-section h2{
    text-align: center;
    text-transform: uppercase;
    padding-bottom:30px;
}

.gallery-section img{
    width:100%;
    display:block;
}

.gallery-section .caption img{
    width: auto;
    display: inline-block;
}

.gallery-section .flex-column{
    position:relative;
}

.gallery-section .caption{
    position: absolute;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  text-align: center;
  z-index:1;
  opacity:0;
  transition:.5s;
}

.gallery-section .flex-column:hover .capti
    opacity:1;
}

.gallery-section h3{
    color: #fff;
  font-size: 26px;
}
```

```css
.gallery-section p{
    font-size: 16px;
  color: #fff;
  font-style: italic;
}
.gallery-section .flex-column::after {
    content: '';
    position: absolute;
    background: #000;
    width: 100%;
    height: 100%;
    top: 0;
    opacity: 0;
    transition:.5s;
    left:0;
}
.gallery-section .flex-column:hover:aft
    opacity:.6;
}
.gallery-section a{
    text-decoration: none;
  background: #173a69;
  color: #fff;
  text-transform: uppercase;
  padding: 10px 20px;
  display: inline-block;
  margin-top: 50px;
  border:1px solid #173a69;
}
.gallery-section a:hover{
    background: transparent;
    color:#173a69;
}
.gallery-section .container{
    text-align: center;
}
```

Output

Gallery Section



HTML Code

Get started, Footer & Socket Section

```
<section class="getstarted-section">
    <div class="container">
        <p>Get started with your project today: <a href="tel:702-960-7754"> 702-960-7754 </a></p>
    </div>
</section>

<section class="footer">
    <div class="container">
        <img src="image/footer-logo.png">
        <div class="social-icon">
            <a href="#"><i class="fa-brands fa-facebook-f"></i></a>
            <a href="#"><i class="fa-brands fa-twitter"></i></a>
            <a href="#"><i class="fa-brands fa-google-plus-g"></i></a>
        </div>
    </div>
</section>

<section class="socket">
    <div class="container">
        <p>® Copyright 2018 <a href="#">Silver Hawk</a>,  All Rights Reserved. Web Design by Once Interactive
        </p>
    </div>
</section>
```

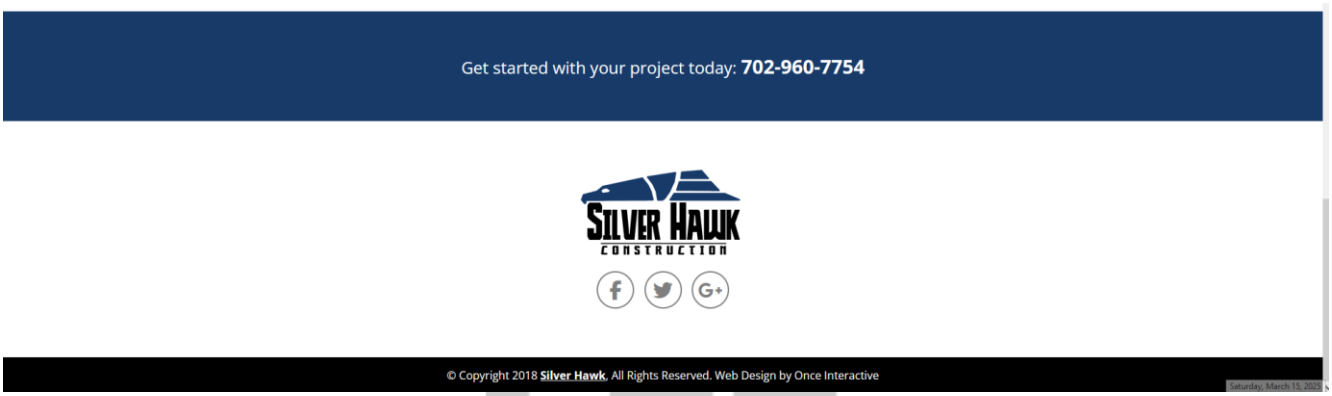
CSS Code

Get started, Footer & Socket Section

```
.getstarted-section{
    background: #173a69;
    text-align: center;
    padding-top:60px;
    padding-bottom:60px;
}
.getstarted-section p{
    color: #fff;
    font-size:24px;
}
.getstarted-section a{
    text-decoration: none;
  color: #fff;
  font-weight: 700;
  font-size: 28px;
}
.footer{
    text-align: center;
    padding-top:70px;
    padding-bottom:70px;
}
.footer i{
    color: #7f7f7f;
  font-size: 28px;
  border: 2px solid;
  height: 50px;
  width: 50px;
  border-radius: 50%;
  line-height: 50px;
}
.footer .fa-twitter{
    margin-left:10px;
    margin-right:10px;
}
.socket{
    background:#000;
    padding-bottom:15px;
    padding-top:15px;
}
.socket p{
    color: #fff;
    text-align: center;
}
.socket a{
    color: #fff;
  font-weight: 700;
}
```

Output

Get started, Footer & Socket Section



Homework/Practice Task

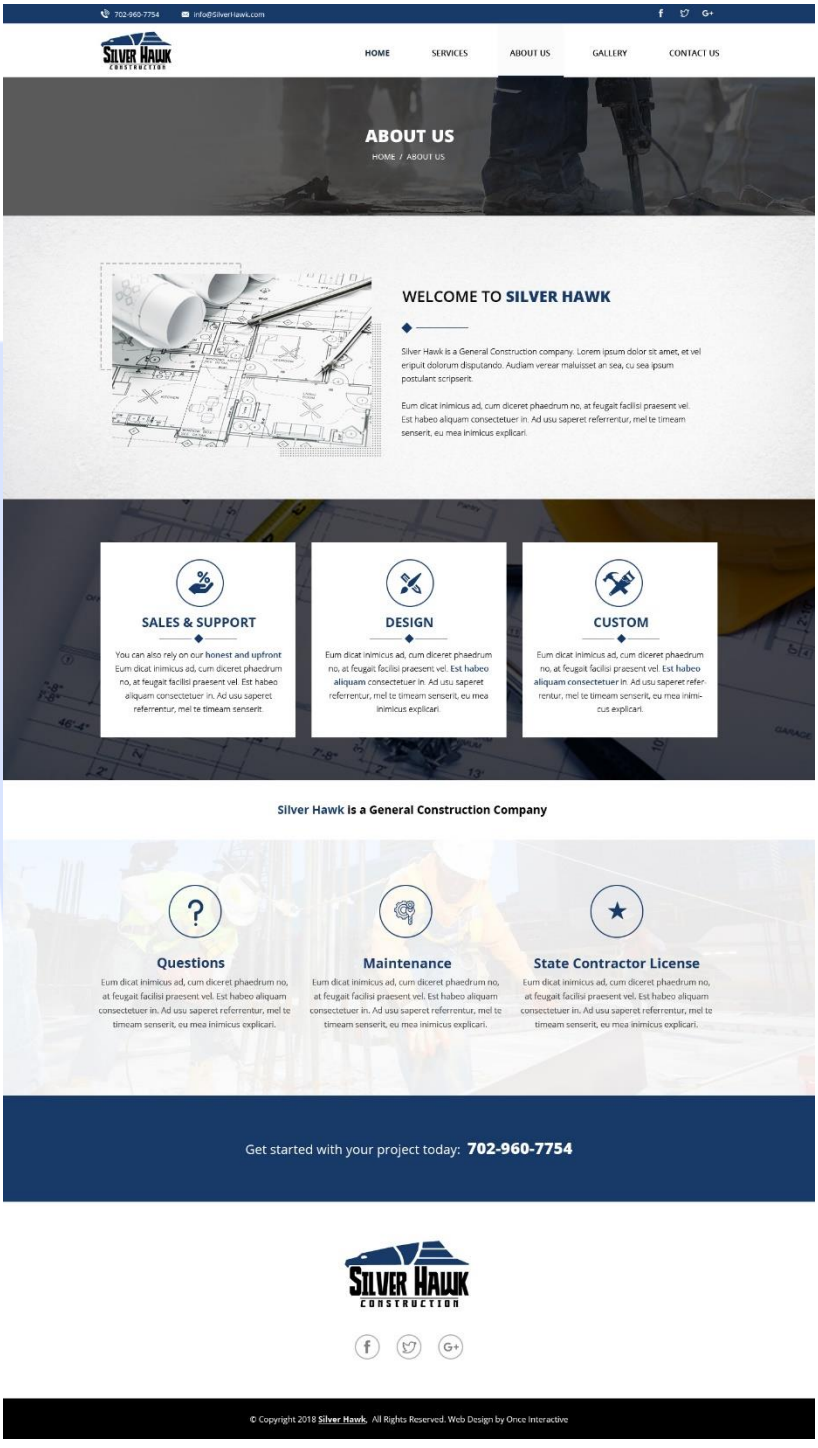
- ❖ Design Home page using HTML & CSS

Lecture 10

Project Name: Silver Hawk | Page: About Us

Technology: HTML & CSS | Tools: Notepad++/VS Code

About Us Page. Click and Download the PSD File: https://shorturl.at/BVAEU



HTML Code

Banner Section

```
<section class="inner-banner-section">
    <h1>About Us</h1>
    <p>Home  /  About Us</p>
</section>
```

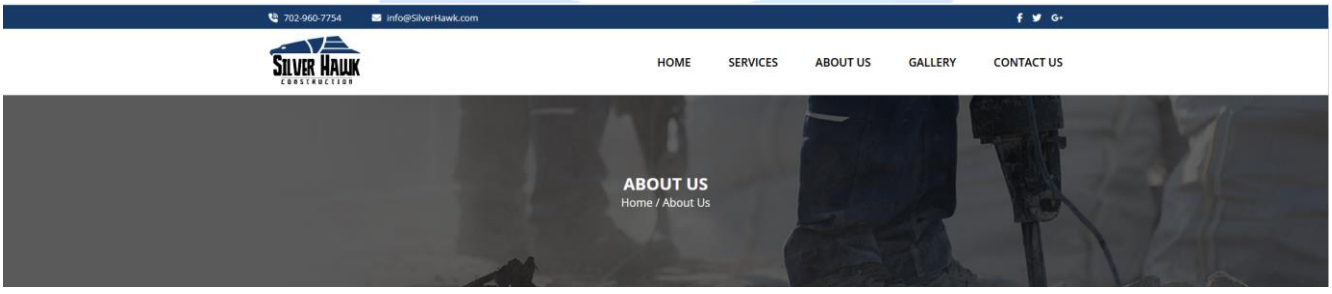
CSS Code

Banner Section

```
.inner-banner-section {
    background: url(image/about-banner.jpg);
    background-repeat: no-repeat;
    background-size: cover;
    text-align: center;
    padding-top: 110px;
    padding-bottom: 110px;
}

.inner-banner-section h1{
    color:#fff;
    text-transform: uppercase;
}

.inner-banner-section p{
    color: #fff;
}
```

Output

Banner Section



HTML Code

About Section

```
<section class="about-section">
    <div class="container">
        <div class="main-column">
            <div class="flex-column image-column">
                <img src="image/about.png">
            </div>
            <div class="flex-column">
                <h2>Welcome to <span>Silver Hawk</span></h2>
                <img src="image/about-icon.png">
                <p>Silver Hawk is a General Construction company. Lorem ipsum dolor sit amet, et vel eripuit
                dolorum disputando. Audiam verear maluisset an sea, cu sea ipsum postulant scripserit. </p>
                <p>Eum dicat inimicus ad, cum diceret phaedrum no, at feugait facilisi praesent vel. Est habeo
                aliquam consectetuer in. Ad usu saperet referrentur, mel te timeam senserit, eu mea inimicus
                explicari.</p>
            </div>
        </div>
    </div>
</section>
```

CSS Code

About Section

```
.about-section h2 span{
    color:green;
}
```

Output

About Section

HTML Code

Sales & Support Section

```html
<section class="support-section">
    <div class="container">
        <div class="main-column">
            <div class="flex-column">
                <img src="image/suppoert-icon.png">
                <h2>Sales & Support</h2>
                <img src="image/heading-icon.png">
                <p>You can also rely on our honest and upfront Eum dicat inimicus ad, cum diceret phaedrum no,
                at feugait facilisi praesent vel. Est habeo aliquam consectetuer in. Ad usu saperet
                referrentur, mel te timeam senserit.</p>
            </div>
            <div class="flex-column">
                <img src="image/suppoert-icon.png">
                <h2>Sales & Support</h2>
                <img src="image/heading-icon.png">
                <p>You can also rely on our honest and upfront Eum dicat inimicus ad, cum diceret phaedrum no,
                at feugait facilisi praesent vel. Est habeo aliquam consectetuer in. Ad usu saperet
                referrentur, mel te timeam senserit.</p>
            </div>
            <div class="flex-column">
                <img src="image/suppoert-icon.png">
                <h2>Sales & Support</h2>
                <img src="image/heading-icon.png">
                <p>You can also rely on our honest and upfront Eum dicat inimicus ad, cum diceret phaedrum no,
                at feugait facilisi praesent vel. Est habeo aliquam consectetuer in. Ad usu saperet
                referrentur, mel te timeam senserit.</p>
            </div>
        </div>
    </div>
</section>
```

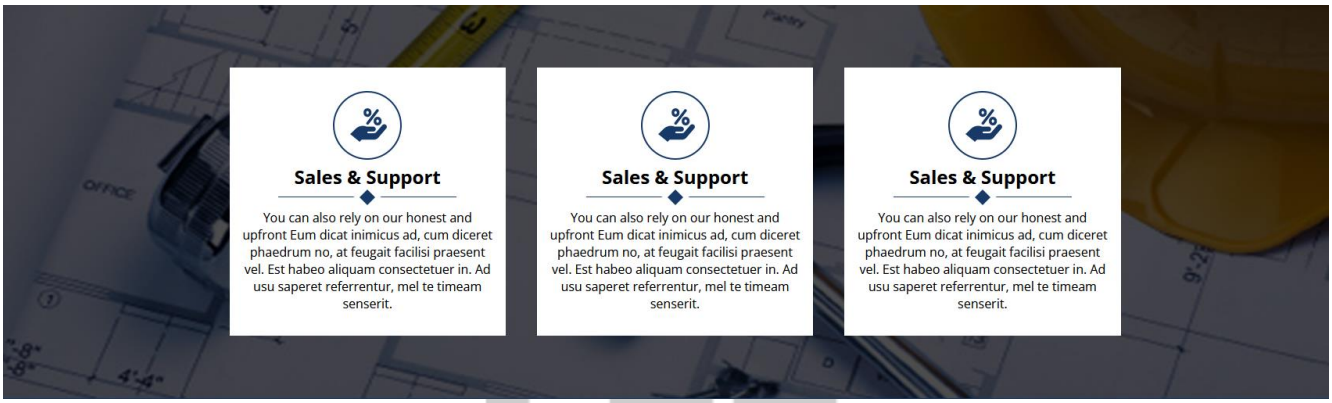
CSS Code

Sales & Support Section

```css
.support-section .main-column{
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    column-gap:40px;
    text-align: center;
}

.support-section{
    background: url(image/support-bg.jpg);
    background-repeat: no-repeat;
    background-size: cover;
    padding-top:80px;
    padding-bottom:80px;
}

.support-section .flex-column{
    background: #fff;
  padding: 30px 15px;
}
```

Output

Sales & Support Section



HTML Code

Construction Company Section

```
<section class='questions_section'>
    <div class="section_heading">
        <div class="container">
            <h2>Silver Hawk is a General Construction Company</h2>

        </div>
    </div>

    <div class="section_item">
        <div class="container">
            <div class='item'>
            <img src='image/questions_icon.png'/>
                <h2>Questions</h2>
                <P>Eum dicat inimicus ad, cum diceret phaedrum no, at feugait facilisi praesent vel. Est habeo
                aliquam consectetuer in. Ad usu saperet referrentur, mel te timeam senserit, eu mea inimicus
                explicari.</P>
            </div>
            <div class='item'>
            <img src='image/questions_icon.png'/>
                <h2>Questions</h2>
                <P>Eum dicat inimicus ad, cum diceret phaedrum no, at feugait facilisi praesent vel. Est habeo
                aliquam consectetuer in. Ad usu saperet referrentur, mel te timeam senserit, eu mea inimicus
                explicari.</P>
            </div>

            <div class='item'>
            <img src='image/questions_icon.png'/>
                <h2>Questions</h2>
                <P>Eum dicat inimicus ad, cum diceret phaedrum no, at feugait facilisi praesent vel. Est habeo
                aliquam consectetuer in. Ad usu saperet referrentur, mel te timeam senserit, eu mea inimicus
                explicari.</P>
            </div>



        </div>
    </div>


</section>
```

CSS Code

Construction Company Section

```
.questions_section .section_heading .container{
    text-align:center;
    padding:50px 0;
}
.questions_section .section_item{
    background-image:url('image/div_bg.png');
    background-repeat:no-repeat;
    background-size:cover;
}

.questions_section .section_item .container {
    display:grid;
    grid-template-columns:1fr 1fr 1fr;
    padding:80px 0;
    gap:30px;
}
.questions_section .section_item .container .item {
    text-align:center;
}
```

Output

Construction Company Section

Homework/Practice Task

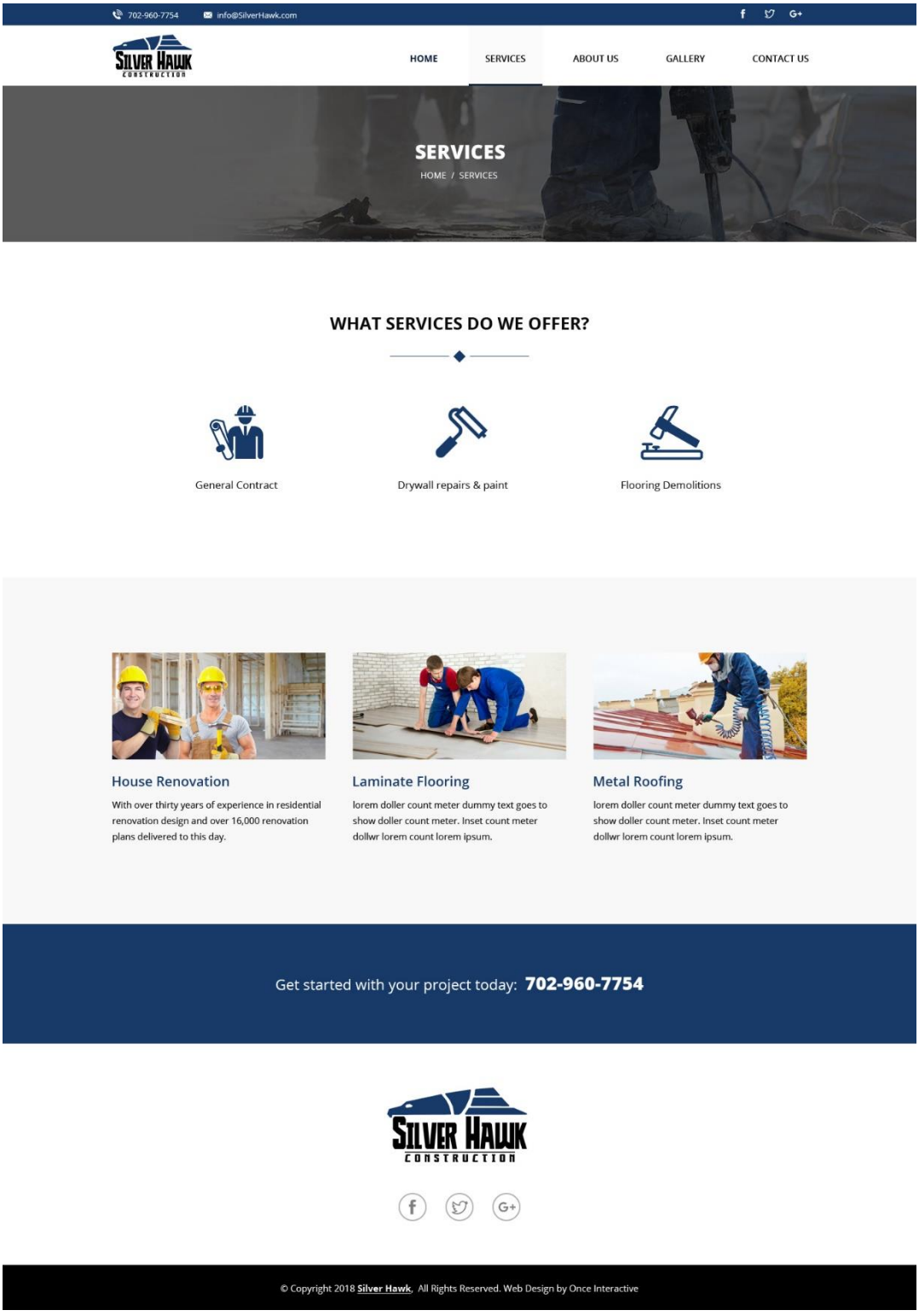
- ❖ Design About page using HTML & CSS

Lecture 11

Project Name: Silver Hawk | Page: Service & Gallery

Technology: HTML & CSS | Tools: Notepad++/VS Code

Service Page. Click and Download the PSD File: https://shorturl.at/NMltp



HTML Code

Service page

```
<section class="inner-banner-section">
    <h2>Services</h2>
    <p><a href='index.html'>Home</a>  /  Services</p>
</section>
<section class="service-section">
    <div class="container">
       <h2>What services do WE offer? </h2>
       <img src="image/headin-incon.png">
        <div class="main-column">
            <div class="flex-column">
                <img src="image/service-icon.png">
                <p>General Contract</p>
            </div>
            <div class="flex-column">
                <img src="image/service-icon.png">
                <p>General Contract</p>
            </div>
            <div class="flex-column">
                <img src="image/service-icon.png">
                <p>General Contract</p>
            </div>
        </div>
    </div>
</section>

<section class="blog_section">
    <div class="container">
        <div class="blog">
            <img src='image/blog.png'/>
            <h3>House Renovation</h3>
            <p>With over thirty years of experience in residential renovation design and over 16,000 renovation
            plans delivered to this day.</p>

        </div>
        <div class="blog">
            <img src='image/blog.png'/>
            <h3>House Renovation</h3>
            <p>With over thirty years of experience in residential renovation design and over 16,000 renovation
            plans delivered to this day.</p>

        </div>
        <div class="blog">
            <img src='image/blog.png'/>
            <h3>House Renovation</h3>
            <p>With over thirty years of experience in residential renovation design and over 16,000 renovation
            plans delivered to this day.</p>

        </div>


    </div>

</section>
```

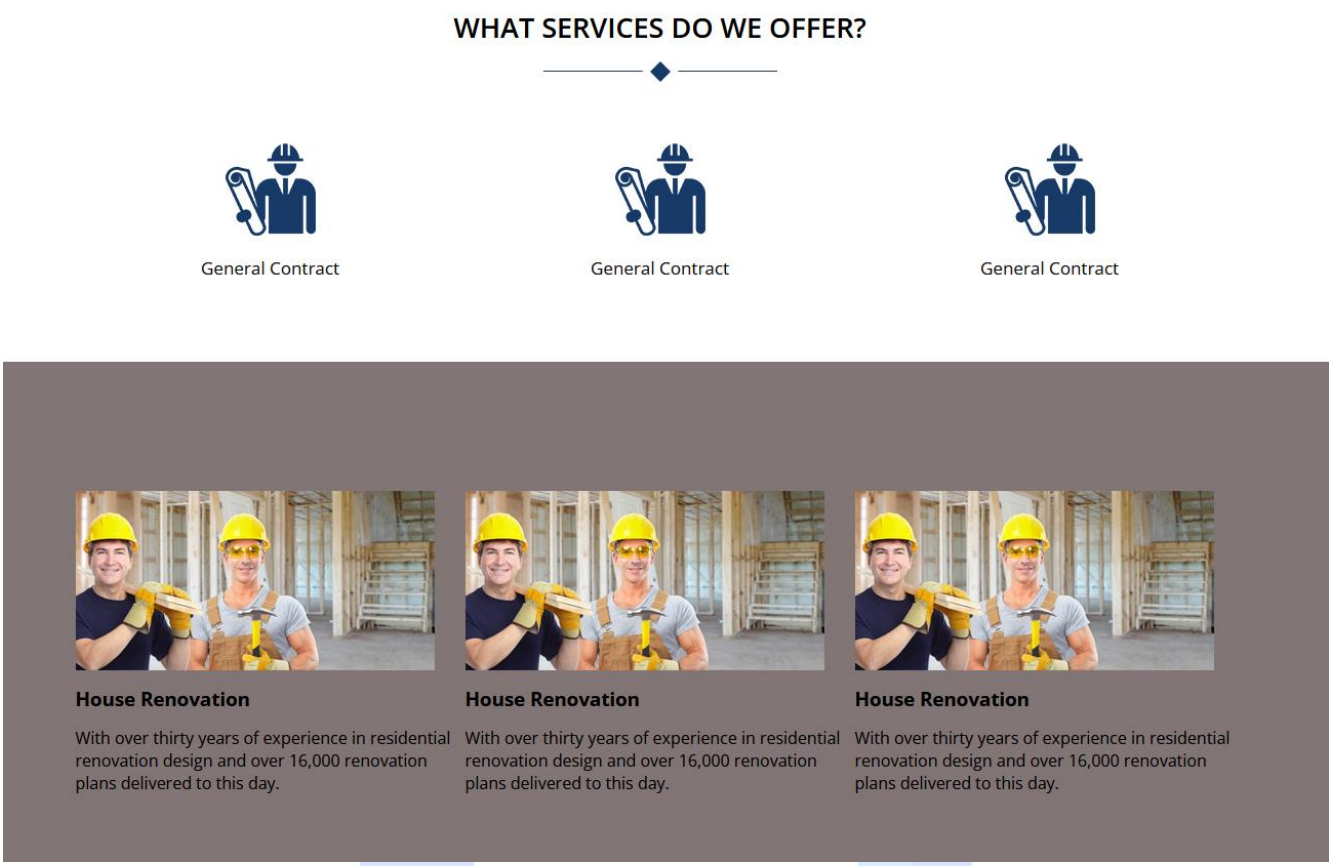
CSS Code

Service page

```
/*----Services Page style--------*/
.blog_section{
    background:#817575;
    padding:126px 0;
}


.blog_section .container{
    display:grid;
    grid-template-columns:1fr 1fr 1fr;


}
.blog_section .container .blog h3{
    margin:15px 0;
}
```

Output

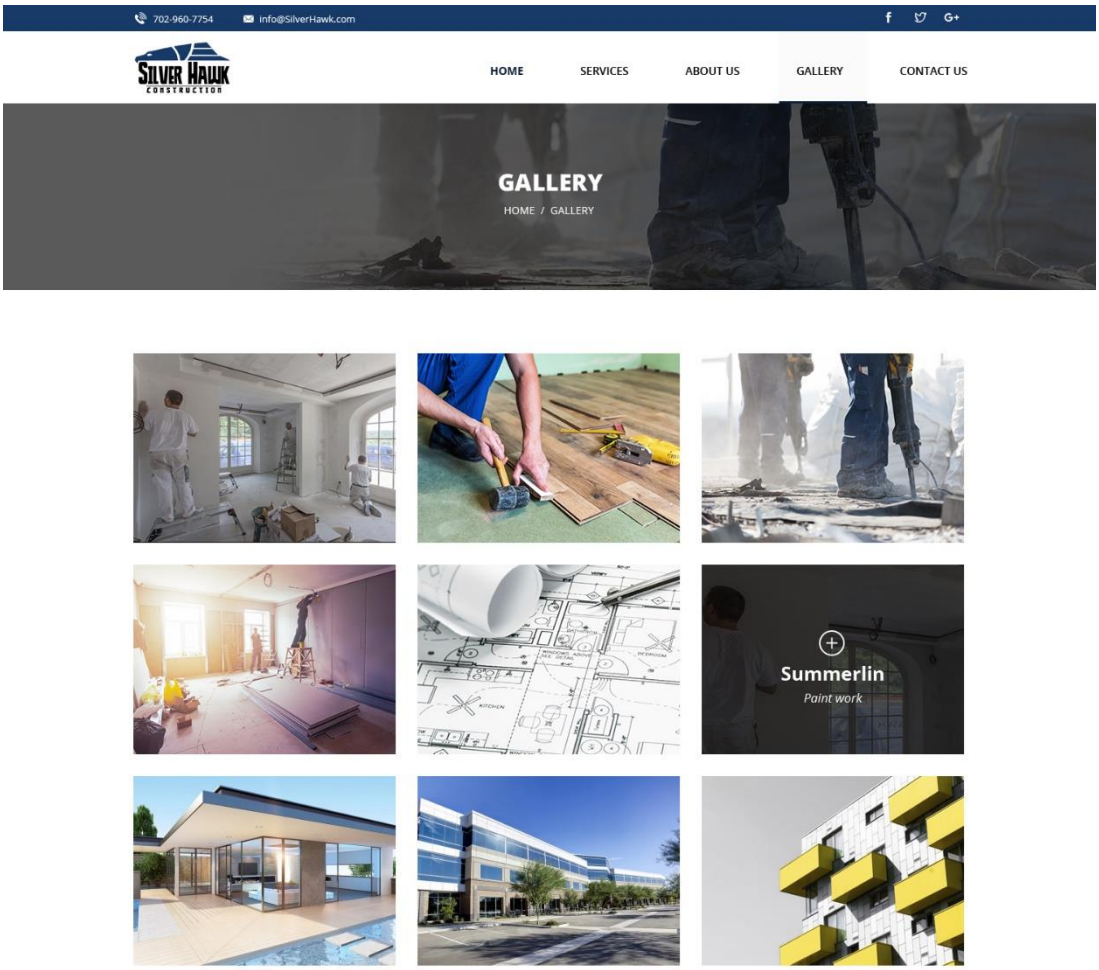
Service page



Homework/Practice Task

- Design Service page using HTML & CSS

Gallery Page. Click and Download the PSD File: https://shorturl.at/QA5jj



Homework/Practice Task

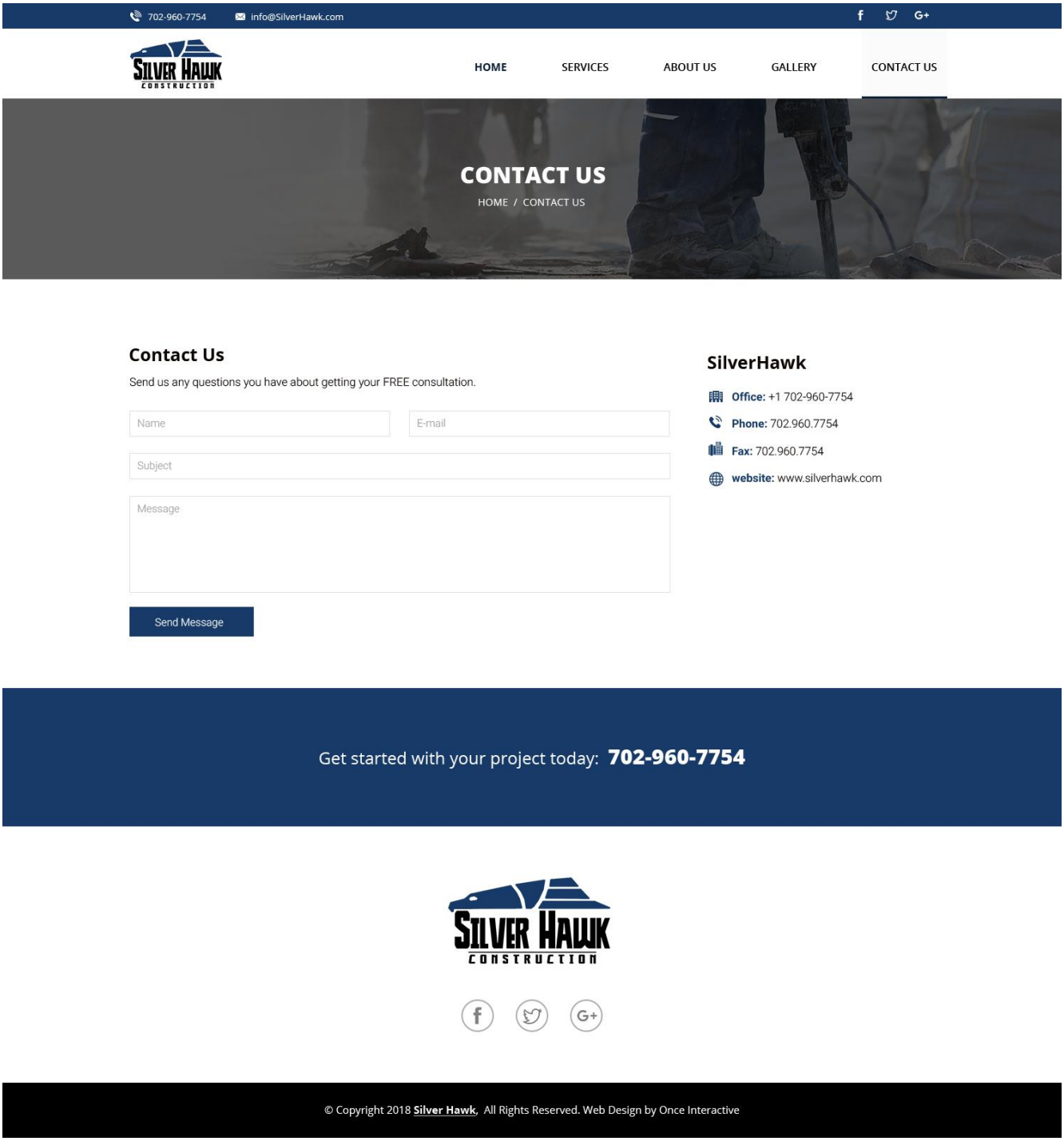
- Design Gallery page using HTML & CSS

Lecture 12

Project Name: Silver Hawk | Page: Contact

Technology: HTML & CSS | Tools: Notepad++/VS Code

Gallery Page. Click and Download the PSD File: https://shorturl.at/T5U9g



HTMl Code

Contact Section

```
<section class="contact-page-section">
    <div class="container">
        <div class="main-column">
            <div class="left-column">
                <h2>Contact Us</h2>
                <p>Send us any questions you have about getting your FREE consultation. </p>
                <form>
                    <div class="name_email">
                        <input type="text" placeholder="name">
                        <input type="email" placeholder="Email">
                    </div>
                    <input type="text" placeholder="subject">
                    <textarea placeholder="message"></textarea>
                    <input type="submit" value="Send Message" class="submit-btn">
                </form>
            </div>
            <div class="right-column">
                <h2>SilverHawk</h2>
                <ul>
                    <li class="phone"><a href="tel:+1 702-960-7754"><strong>Office:</strong>+1 702-960-7754
                    </a></li>
                    <li><a href="tel:+1 702-960-7754"><strong>Office:</strong>+1 702-960-7754 </a></li>
                    <li class="email"><a href="tel:+1 702-960-7754"><strong>Office:</strong>+1 702-960-7754
                    </a></li>
                    <li class="fax"><a href="tel:+1 702-960-7754"><strong>Office:</strong>+1 702-960-7754 </a></li>
                    <li><a href="tel:+1 702-960-7754"><strong>Office:</strong>+1 702-960-7754 </a></li>
                </ul>
            </div>
        </div>
    </div>
</section>
```

CSS Code

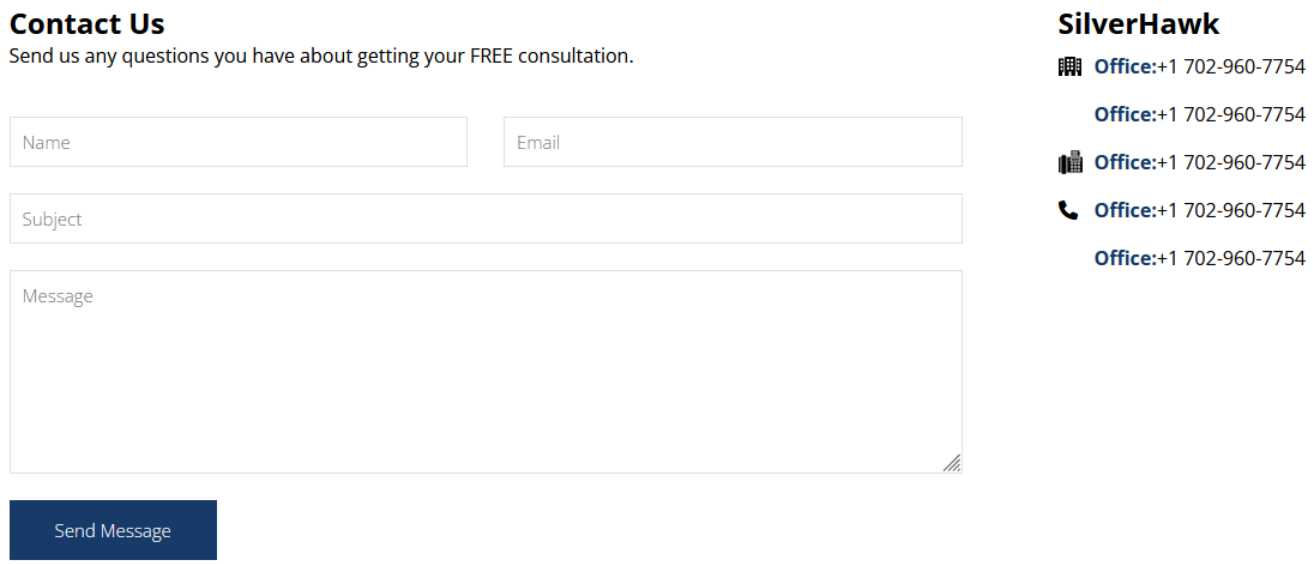
Contact Section

```css
.contact-page-section .main-column {
    display: grid;
    grid-template-columns: 3fr 1fr;
    column-gap: 80px;
}

.contact-page-section .name_email{
    display: grid;
    grid-template-columns: 1fr 1fr;
    column-gap: 30px;
    margin-top:40px;
}

.contact-page-section input,
.contact-page-section textarea {
    display: block;
    width: 100%;
    box-sizing: border-box;
    border: 1px solid #e1e1e1;
    padding: 10px 10px;
    text-transform: capitalize;
    font-weight: 300;
    font-size: 15px;
    margin-bottom: 22px;
    font-family: "Open Sans", sans-serif;

}



.contact-page-section textarea {
    height:170px;
}


.contact-page-section .submit-btn {
    width: auto;
    background: #173a69;
    color: #fff;
    font-size: 15px;
    font-weight: 300;
    padding: 13px 36px;
    cursor: pointer;
    border:2px solid #173a69;
}

.contact-page-section .submit-btn:hover{
    background: transparent;
    color: #173a69;
}
```

```css
.contact-page-section a{
    font-size: 16px;
  color: #0c0603;
  text-decoration: none;
}

.contact-page-section strong{
    color: #173a69;
}

.contact-page-section ul {
    list-style: none;
    line-height: 40px;
}

.contact-page-section li {
    position: relative;
    padding-left: 30px;
}

.contact-page-section .phone::before {
    content: '';
    position: absolute;
    background: url(image/Forma1.png);
    height: 16px;
    width: 19px;
    background-size: 100%;
    left: 0;
    top: 50%;
    transform: translateY(-50%);
}

.contact-page-section .email::before {
    content: '';
    position: absolute;
    background: url(image/L0001.png);
    height: 20px;
    width: 20px;
    background-size: 100%;
    left: 0;
    top: 50%;
    transform: translateY(-50%);
}


.contact-page-section .fax::before {
    content: '\f095';
    position: absolute;
    left: 0;
    top: 50%;
    transform: translateY(-50%);
    font-family: fontawesome;
}
```

Output

Contact section



Homework/Practice Task

- Design Contact page using HTML & CSS

Lecture 13

How to Create Fiverr Account

🌟 What You Must Know Before Creating a Fiverr Account

🔰 Introduction

Creating a Fiverr account is your first step into the world of online freelancing. Fiverr is a powerful platform where you can showcase your skills, connect with global clients, and earn a steady income. But before jumping in, it's important to prepare yourself with the right mindset, strategy, and tools. Setting up your account properly from the beginning can make a big difference in how successful you become on the platform.

Whether you're a WordPress expert, a designer, or a developer, the following checklist will help you get started the right way:

✅ 1. Know What Service You'll Offer

- Choose your **niche** (e.g., WordPress development, Elementor design, WooCommerce setup).
- Specialize in what you're good at and have real experience in (you already have this in WordPress).
- Think about 2–3 related services you can offer as **separate gigs**.

✅ 2. Prepare Portfolio Samples

- You need **at least 3–5 solid samples** of past work to show clients.
- If you don't have client work, create sample projects (homepages, e-commerce, landing pages, etc.).
- Use tools like Figma, screenshots, or live links to show your work.

✅ 3. Decide on Username and Email

- Choose a **professional username** (easy to remember and relevant if possible).
- Use a dedicated email for freelancing.

✅ 4. Write Down Your Skills and Tags

- Fiverr asks for your **skills**, **tags**, and **gig metadata**.
- Make a list of keywords (like “Elementor expert”, “WordPress bug fix”, “WooCommerce setup”).

✅ 5. Prepare Gig Descriptions

- Write **clear, benefit-focused descriptions** for each service.
- Mention what you offer, what the client will get, and why you’re the right choice.

✅ 6. Plan Pricing Strategy

- Fiverr allows **3 packages**: Basic, Standard, and Premium.
- Start with **competitive pricing** (not too low, but affordable).
- Offer **extras** (like faster delivery, extra revisions, etc.) to increase value.

✅ 7. Have a Good Profile Image and Gig Thumbnails

- Use a **professional photo** of yourself.
- Create attractive **gig images or thumbnails** using Canva or Photoshop.
- Optional: Add a short **intro video** (boosts trust).

✅ 8. Setup Payment Method

- Fiverr pays via **Payoneer**, **Bank Transfer**, or **PayPal**.
- Make sure you have at least one method ready (Payoneer is popular in Bangladesh).

✅ 9. Understand Fiverr Rules

- Don’t ask clients to communicate outside Fiverr.
- Don’t copy other sellers' content.
- Be responsive and polite in messages.

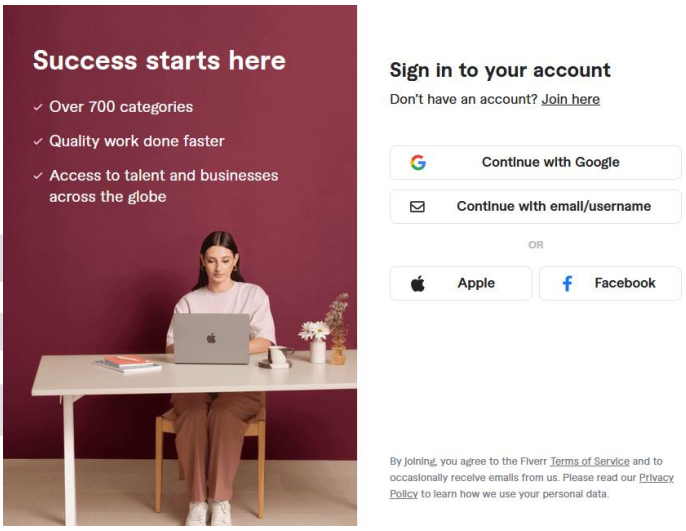
✅ 10. Be Ready to Respond Quickly

- Fiverr ranks you based on **response time and delivery**.
- Keep the app on your phone for quick replies.

How to Create Fiverr Account?

Step 1: Go to Fiverr Website

Visit https://www.fiverr.com and click on 'Join' at the top-right corner.



Step 2: Sign Up for Fiverr

You can sign up with Google, Facebook, Apple, or email. Choose 'Continue with Email' for professional use.

Step 3: Set a Username & Password

Choose a professional username (e.g., webfaruk). Create a strong password and click 'Join'. Confirm via email.

Step 4: Switch to Seller Account

After verifying, click your profile picture > 'Become a Seller' > click through Fiverr intro screens.

Step 5: Set Up Your Seller Profile

Add professional info, profile photo, description, skills, education, and link accounts like GitHub or StackOverflow.

Step 6: Create Your First Gig

Title example: 'I will create a responsive WordPress website using Elementor'. Category: Website Development. Add search tags, pricing, gig description, requirements (like content/logo), and upload gig images or a short video.

Final Step: Publish Gig

Click 'Publish'. Now your Fiverr profile is ready! Start promoting your gig to get your first order.

Lecture 14

JS Introduction, JS Where To,
JS Output,JS Statements

JavaScript

- JavaScript is the world's most popular programming language.
- JavaScript is the programming language of the Web.
- JavaScript is easy to learn.
- This lecture sheet will teach you JavaScript from basic to advanced.

What is JavaScript?

- JavaScript is the programming language of the web.
- It can update and change both HTML and CSS.
- It can calculate, manipulate and validate data.

Why Study JavaScript?

JavaScript is one of the 3 languages all web developers must learn:

1. HTML to define the content of web pages
2. CSS to specify the layout of web pages
3. JavaScript to program the behavior of web pages

JavaScript Where To

In HTML, JavaScript code is inserted between <script> and </script> tags.

```
<script>
document.getElementById("demo").innerHTML = "My First JavaScript";
</script>
```

JavaScript in <head> or <body>

You can place any number of scripts in an HTML document.
Scripts can be placed in the <body>, or in the <head> section of an HTML page, or in both.

JavaScript in <head>

In this example, a JavaScript function is placed in the <head> section of an HTML page.
The function is invoked (called) when a button is clicked:

```
<!DOCTYPE html>
<html>
<head>
<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>
</head>
<body>

<h2>Demo JavaScript in Head</h2>

<p id="demo">A Paragraph</p>
<button type="button" onclick="myFunction()">Try it</button>

</body>
</html>
```

JavaScript in <body>

In this example, a JavaScript function is placed in the <body> section of an HTML page.
The function is invoked (called) when a button is clicked:

```
<!DOCTYPE html>
<html>
<body>

<h2>Demo JavaScript in Body</h2>

<p id="demo">A Paragraph</p>

<button type="button" onclick="myFunction()">Try it</button>

<script>
function myFunction() {
  document.getElementById("demo").innerHTML = "Paragraph changed.";
}
</script>

</body>
</html>
```

External JavaScript

Scripts can also be placed in external files:
To use an external script, put the name of the script file in the src (source) attribute of a <script> tag:
<script src="myScript.js"></script>

JavaScript Output

JavaScript Display Possibilities
JavaScript can "display" data in different ways:

- Writing into an HTML element, using innerHTML or innerText.
- Writing into the HTML output using document.write().
- Writing into an alert box, using window.alert().
- Writing into the browser console, using console.log().

Using innerHTML

To access an HTML element, you can use the document.getElementById(id) method.
Use the id attribute to identify the HTML element.
Then use the innerHTML property to change the HTML content of the HTML element:

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My First Paragraph</p>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "<h2>Hello World</h2>";
</script>

</body>
</html>
```

Using innerText

To access an HTML element, use the document.getElementById(id) method.
Then use the innerText property to change the inner text of the HTML element:

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My First Paragraph</p>

<p id="demo"></p>

<script>
document.getElementById("demo").innerText = "Hello World";
</script>

</body>
</html>
```

Using document.write()

For testing purposes, it is convenient to use document.write():

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
document.write(5 + 6);
</script>

</body>
</html>
```

Using window.alert()

You can use an alert box to display data:

```
<!DOCTYPE html>
<html>
<body>

<h1>My First Web Page</h1>
<p>My first paragraph.</p>

<script>
window.alert(5 + 6);
</script>

</body>
</html>
```

Using console.log()

For debugging purposes, you can call the console.log() method in the browser to display data.

```
<!DOCTYPE html>
<html>
<body>

<script>
console.log(5 + 6);
</script>

</body>
</html>
```

JavaScript Statements

JavaScript statements are composed of:
Values, Operators, Expressions, Keywords, and Comments.
This statement tells the browser to write "Hello Dolly." inside an HTML element with id="demo":

Example

```
let x, y, z;   // Statement 1
x = 5;         // Statement 2
y = 6;         // Statement 3
z = x + y;     // Statement 4


document.getElementById("demo").innerHTML = "Hello Dolly.";
```

Semicolons ;

Semicolons separate JavaScript statements.

Add a semicolon at the end of each executable statement:

```
let a, b, c;  // Declare 3 variables
a = 5;        // Assign the value 5 to a
b = 6;        // Assign the value 6 to b
c = a + b;    // Assign the sum of a and b to c
```

When separated by semicolons, multiple statements on one line are allowed:

a = 5; b = 6; c = a + b;

JavaScript White Space

JavaScript ignores multiple spaces. You can add white space to your script to make it more readable.

The following lines are equivalent:

```
let person = "Hege";
let person="Hege";
```

A good practice is to put spaces around operators (= + - * /):

```
let x = y + z;
```

JavaScript Keywords

JavaScript statements often start with a keyword to identify the JavaScript action to be performed.

Keyword	Description
var	Declares a variable
let	Declares a block variable
const	Declares a block constant
if	Marks a block of statements to be executed on a condition
switch	Marks a block of statements to be executed in different cases
for	Marks a block of statements to be executed in a loop
function	Declares a function
return	Exits a function
try	Implements error handling to a block of statements

Homework/Practice Task

1. Write a short note (3-5 sentences) explaining what JavaScript is and why it is important in web development.
2. Create an HTML file where you place a <script> inside the <head> section that shows an alert box saying “Hello from the head!”.
3. Create another HTML file where the JavaScript function is placed in the <body> section and displays a message in an alert box when a button is clicked.
4. Create two files:
   index.html
   script.js
   In script.js, write a simple alert("This is from external JS file!"); and link it properly in index.html.
5. Create an HTML file that uses all four output methods to display different messages:
   - innerHTML
   - document.write()
   - window.alert()
   - console.log()
6. Create a paragraph with an id in HTML. Then use JavaScript to change the content of that paragraph using innerHTML.
7. Create a heading (<h2>) with an id. Write JavaScript to change its text using innerText.
8. Write a script that declares three variables x, y, and z, assigns values to x and y, and calculates the sum in z. Then display the result using innerHTML.
9. Write a JavaScript example showing multiple statements on one line using semicolons. Then rewrite the same code in a cleaner, more readable format using white space.
10. List 5 JavaScript keywords you learned from the content, and write a sentence using each one in a simple script.

Lecture 15

JS Syntax, JS Comments, JS Variables, JS Let,
JS Const, JS Operators

JavaScript Syntax

JavaScript syntax is the set of rules, how JavaScript programs are constructed:

```
// How to create variables:
var x;
let y;

// How to use variables:
x = 5;
y = 6;
let z = x + y;
```

JavaScript Values

The JavaScript syntax defines two types of values:

- Fixed values
- Variable values

Fixed values are called **Literals**.

Variable values are called **Variables**.

JavaScript Literals

The two most important syntax rules for fixed values are:

1. **Numbers** are written with or without decimals:

```
10.50

1001
```

2. **Strings** are text, written within double or single quotes:

```
"John Doe"

'John Doe'
```

JavaScript Comments

JavaScript comments can be used to explain JavaScript code, and to make it more readable.
JavaScript comments can also be used to prevent execution, when testing alternative code.

Single Line Comments

Single line comments start with //.
Any text between // and the end of the line will be ignored by JavaScript (will not be executed).

```
// Change heading:
document.getElementById("myH").innerHTML = "My First Page";

// Change paragraph:
document.getElementById("myP").innerHTML = "My first paragraph.";
```

Multi-line Comments

Multi-line comments start with /* and end with */.

Any text between /* and */ will be ignored by JavaScript.

```
/*
The code below will change
the heading with id = "myH"
and the paragraph with id = "myP"
in my web page:
*/
document.getElementById("myH").innerHTML = "My First Page";
document.getElementById("myP").innerHTML = "My first paragraph.";
```

JavaScript Variables

Variables are Containers for Storing Data
JavaScript Variables can be declared in 4 ways:

- Automatically
- Using var
- Using let
- Using const

In this first example, x, y, and z are undeclared variables.

They are automatically declared when first used:

```
x = 5;
y = 6;
z = x + y;
```

Example using var

```
var x = 5;
var y = 6;
var z = x + y;
```

Note

The var keyword was used in all JavaScript code from 1995 to 2015.

The let and const keywords were added to JavaScript in 2015.

The var keyword should only be used in code written for older browsers.

Example using let

```
let x = 5;
let y = 6;
let z = x + y;
```

Example using const

```
const x = 5;
const y = 6;
const z = x + y;
```

Mixed Example

```
const price1 = 5;
const price2 = 6;
let total = price1 + price2;
```

The two variables price1 and price2 are declared with the const keyword.

These are constant values and cannot be changed.

The variable total is declared with the let keyword.

The value total can be changed.

When to Use var, let, or const?

1. Always declare variables

2. Always use const if the value should not be changed

3. Always use const if the type should not be changed (Arrays and Objects)

4. Only use let if you can't use const

5. Only use var if you MUST support old browsers.

JavaScript Let

The let keyword was introduced in ES6 (2015)

Variables declared with let have **Block Scope**

Variables declared with let must be **Declared** before use

Variables declared with let cannot be **Redeclared** in the same scope

Block Scope

Before ES6 (2015), JavaScript did not have **Block Scope**.

JavaScript had **Global Scope** and **Function Scope**.

ES6 introduced the two new JavaScript keywords: let and const.

These two keywords provided **Block Scope** in JavaScript:

Example

Variables declared inside a { } block cannot be accessed from outside the block:

```
{
  let x = 2;
}
// x can NOT be used here
```

Global Scope

Variables declared with the var always have **Global Scope**.

Variables declared with the var keyword can NOT have block scope:

Example

Variables declared with var inside a { } block can be accessed from outside the block:

```
{
  var x = 2;
}
// x CAN be used here
```

JavaScript Const

The const keyword was introduced in ES6 (2015)

Variables defined with const cannot be **Redeclared**

Variables defined with const cannot be **Reassigned**

Variables defined with const have **Block Scope**

Cannot be Reassigned

A variable defined with the const keyword cannot be reassigned:

```
const PI = 3.141592653589793;
PI = 3.14;      // This will give an error
PI = PI + 10;   // This will also give an error
```

JavaScript Operators

Javascript operators are used to perform different types of mathematical and logical computations.

Examples:

The Assignment Operator = assigns values

The Addition Operator + adds values

The Multiplication Operator * multiplies values

The Comparison Operator > compares values

Assignment Examples

```
let x = 10;
```

JavaScript Addition

The Addition Operator (+) adds numbers:

```
let x = 5;
let y = 2;
let z = x + y;
```

JavaScript Multiplication

The Multiplication Operator (*) multiplies numbers:

```
let x = 5;
let y = 2;
let z = x * y;
```

Types of JavaScript Operators

There are different types of JavaScript operators:

- Arithmetic Operators
- Assignment Operators
- Comparison Operators
- String Operators
- Logical Operators
- Bitwise Operators
- Ternary Operators
- Type Operators

Arithmetic Operators Example

```
let a = 3;
let x = (100 + 50) * a;
```

JavaScript Assignment Operators

```
let x = 10;
x += 5;
```

JavaScript Comparison Operators

Operator	Description
==	equal to
===	equal value and equal type
!=	not equal
!==	not equal value or not equal type
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to
?	ternary operator

JavaScript String Comparison
All the comparison operators above can also be used on strings:

```
let text1 = "A";
let text2 = "B";
let result = text1 < text2;
```

JavaScript String Addition

The + can also be used to add (concatenate) strings:

```
let text1 = "John";
let text2 = "Doe";
let text3 = text1 + " " + text2;
```

JavaScript Logical Operators

Operator	Description
&&	logical and
||	logical or
!	logical not

JavaScript Bitwise Operators

Bit operators work on 32 bits numbers.

Any numeric operand in the operation is converted into a 32 bit number. The result is converted back to a JavaScript number.

Operator	Description	Example	Same as	Result	Decimal
&	AND	5 & 1	0101 & 0001	0001	1
|	OR	5 | 1	0101 | 0001	0101	5
~	NOT	~ 5	~0101	1010	10
^	XOR	5 ^ 1	0101 ^ 0001	0100	4
<<	left shift	5 << 1	0101 << 1	1010	10
>>	right shift	5 >> 1	0101 >> 1	0010	2
>>>	unsigned right shift	5 >>> 1	0101 >>> 1	0010	2

Homework/Practice Task

1. Create a JavaScript file where you define:

- A number variable with a decimal
- A number variable without a decimal
- A string using double quotes
- A string using single quotes
  Then display each one using console.log().

2. Create a JavaScript program that contains:

- At least 2 single-line comments
- 1 multi-line comment
  Make the comments describe what each part of the code is doing.

3. Write a script where you declare variables using var, let, and const.
   Display the values using document.write() or console.log().
   Try to change the value of a const variable and observe the result (don't worry about the error—it's expected!).

4. Write an HTML file with embedded JavaScript that:
   Declares a variable with let inside a block ({})
   Declares a variable with var inside the same block
   Try to access both variables outside the block and display the result using console.log().

5. Create a script that:
   Declares two constants for price1 and price2
   Declares a total variable using let
   Calculates and displays the total price using innerHTML

Lecture 16

JS Data Types, JS Functions, JS Objects

JS Data Types

JavaScript has 8 Datatypes

- String
- Number
- Bigint
- Boolean
- Undefined
- Null
- Symbol
- Object

The Object Datatype

The object data type can contain both built-in objects, and user defined objects:
Built-in object types can be:
objects, arrays, dates, maps, sets, intarrays, floatarrays, promises, and more.

```
// Numbers:
let length = 16;
let weight = 7.5;

// Strings:
let color = "Yellow";
let lastName = "Johnson";

// Booleans
let x = true;
let y = false;

// Object:
const person = {firstName:"John", lastName:"Doe"};

// Array object:
const cars = ["Saab", "Volvo", "BMW"];

// Date object:
const date = new Date("2022-03-25");
```

Note

A JavaScript variable can hold any type of data.

The Concept of Data Types

In programming, data types is an important concept.

To be able to operate on variables, it is important to know something about the type.

Without data types, a computer cannot safely solve this:

```
let x = 16 + "Volvo";
```

Does it make any sense to add "Volvo" to sixteen? Will it produce an error or will it produce a result?
JavaScript will treat the example above as:

```
let x = "16" + "Volvo";
```

Note

When adding a number and a string, JavaScript will treat the number as a string.

Example

```
let x = 16 + "Volvo";
```

```
let x = "Volvo" + 16;
```

JavaScript evaluates expressions from left to right. Different sequences can produce different results:

JavaScript

```
let x = 16 + 4 + "Volvo";
```

Result

```
20Volvo
```

JS Function

A JavaScript function is a reusable block of code designed to perform a specific task. You can define a function once and then call it whenever you need it.

◈ Syntax

```
function functionName(parameters) {
  // code to be executed
}
```

- **functionName** – The name of the function.
- **parameters** – Optional. Input values the function accepts (like variables).
- The code inside the curly braces {} runs when the function is called.

◈ Example 1: Basic Function

```
function greet(name) {
  console.log("Hello, " + name + "!");
}

greet("Faruk"); // Output: Hello, Faruk!
```

◈ Explanation:

- **greet** is the function name.
- **name** is the parameter.
- When you call **greet("Faruk")**, it prints "Hello, Faruk!".

◈ Example 2: Function with Return Value

```
function add(a, b) {
  return a + b;
}

let result = add(5, 7);
console.log(result); // Output: 12
```

◈ Explanation:

- add takes two parameters: a and b.
- It returns the sum using return a + b.

◈ Types of Functions in JS

1. Function Declaration:

```
function sayHi() {
  console.log("Hi!");
}
```

2. Function Expression:

```
const sayHi = function() {
  console.log("Hi!");
};
```

3. Arrow Function:

```
const sayHi = () => {
  console.log("Hi!");
};
```

JavaScript Object?

A JavaScript object is a collection of key-value pairs used to store related data and functions. It's like a real-world object: a car has properties like color, model, and functions like start or drive.

◈ Syntax

```
const objectName = {
  key1: value1,
  key2: value2,
  key3: function() {
    // method
  }
};
```

◈ Example: Person Object

```
const person = {
  firstName: "Faruk",
  lastName: "Ahamed",
  age: 30,
  fullName: function() {
    return this.firstName + " " + this.lastName;
  }
};

console.log(person.firstName);          // Output: Faruk
console.log(person["age"]);             // Output: 30
console.log(person.fullName());         // Output: Faruk Ahamed
```

◈ Explanation:

- `firstName`, `lastName`, and `age` are **properties**.
- `fullName` is a **method** (function inside an object).
- `this` refers to the **current object**.

◈ Accessing Object Data

- Dot notation: `person.age`
- Bracket notation: `person["age"]`

◈ Adding / Updating / Deleting

```
person.country = "Bangladesh";      // Add
person.age = 31;                    // Update
delete person.lastName;             // Delete
```

◈ Object Inside Object

```
const student = {
  name: "Rafi",
  address: {
    city: "Dhaka",
    zip: 1205
  }
};

console.log(student.address.city); // Output: Dhaka
```

◈ Array of Objects

```
const users = [
  { name: "Alice", age: 25 },
  { name: "Bob", age: 30 }
];

console.log(users[0].name); // Output: Alice
```

Homework/Practice Task

Task 1: Identify Data Types

Create 8 variables using different JavaScript data types (String, Number, BigInt, Boolean, Undefined, Null, Symbol, and Object) and use typeof to display their types in the

Task 2: String + Number

Write a script where you add a number to a string and observe the output. Then, reverse the order and compare the results. Example:

```
let result1 = "Volvo" + 16;
let result2 = 16 + "Volvo";
console.log(result1);
console.log(result2);
```

Task 3: Create a Basic Function

Write a function called welcomeUser(name) that prints "Welcome, [name]!" to the console.

Call the function with your own name.

Task 4: Function that Returns a Value

Create a function called multiply(a, b) that returns the multiplication of two numbers. Call the function with different values and display the result.

Task 5: Function Types Practice

Write 3 different functions that do the same thing (e.g., say "Hello!") using:

- Function Declaration
- Function Expressio
- Arrow Function

Compare their syntax.

Lecture 17

JS Array

JavaScript Array

A JavaScript array is a special variable that can hold multiple values in a single variable. Instead of creating separate variables for each value:

```
let fruit1 = "Apple";
let fruit2 = "Banana";
let fruit3 = "Mango";
```

You can use an array:

```
let fruits = ["Apple", "Banana", "Mango"];
```

◈ Array Syntax

```
let arrayName = [item1, item2, item3, ...];
```

Example:

```
let colors = ["Red", "Green", "Blue"];
```

◈ Array Indexing

- Array indexes start at 0.
- To access the first element: arrayName[0]

Example:

```
let cars = ["BMW", "Volvo", "Tesla"];
console.log(cars[0]); // Output: BMW
console.log(cars[2]); // Output: Tesla
```

◈ Array Length

You can get the total number of items using .length.

```
console.log(cars.length); // Output: 3
```

◈ Modifying Array Elements

```
cars[1] = "Toyota";
console.log(cars); // Output: ["BMW", "Toyota", "Tesla"]
```

◈ Adding Elements

1. **To the End** – `.push()`

```
cars.push("Honda");
```

2. To the Beginning – .unshift()

```
cars.unshift("Ford");
```

◈ Removing Elements

1. From the End – .pop()

```
cars.pop(); // removes last element
```

2. From the Beginning – .shift()

```
cars.shift(); // removes first element
```

◈ Loop Through an Array

```
for (let i = 0; i < cars.length; i++) {
  console.log(cars[i]);
}
```

Or using forEach:

```
cars.forEach(function(car) {
  console.log(car);
});
```

◈ Array Methods (Most Common)

Method	Description
`push()`	Adds element to end
`pop()`	Removes element from end
`shift()`	Removes element from beginning
`unshift()`	Adds element to beginning
`length`	Returns the number of elements
`concat()`	Merges two arrays
`slice()`	Returns a portion of the array
`splice()`	Adds/removes elements at a given index
`indexOf()`	Finds the index of an element
`includes()`	Checks if an element exists
`sort()`	Sorts the array alphabetically
`reverse()`	Reverses the order of elements

◈ Example: Full Use

```
let students = ["Rafi", "Asha", "Nayeem"];

students.push("Sadia");          // Add to end
students.unshift("Tanvir");      // Add to start
students.pop();                  // Remove from end
students.splice(1, 1, "Mahi");   // Replace 1 element at index 1

console.log(students); // Output: ["Tanvir", "Mahi", "Nayeem"]
```

◈ Array of Objects (Bonus)

```
let users = [
  { name: "Faruk", age: 28 },
  { name: "Sabbir", age: 30 }
];

console.log(users[0].name); // Output: Faruk
```

◈ Summary

- ✅ Arrays can hold multiple values
- ✅ Elements can be added, removed, or changed
- ✅ Use loops or methods like forEach to work with arrays
- ✅ Arrays are powerful for data handling in JavaScript

Homework/Practice Task

1. Create an array called countries with at least 5 country names.
Print:

- The first country
- The last country
- The total number of countries

2. Create an array called animals = ["Cat", "Dog", "Elephant"].
   Then:

   - Change "Dog" to "Lion"
   - Add "Tiger" to the end
   - Remove the first element

   Print the updated array.

3. Create an empty array called fruits.

   - Do the following using array methods:
   - Add "Apple", "Banana", and "Mango" to the array
   - Remove the last fruit
   - Add "Orange" to the beginning
   - Print the final array

4. Create an array numbers = [10, 20, 30, 40, 50].
   Use a for loop to print each number multiplied by 2.

5. Create an array of names: ["Sami", "Fahim", "Rima"].
   Use .forEach() to print a greeting like:

   Hello, Sami!
   Hello, Fahim!
   Hello, Rima!

Lecture 18

JS Loop

JavaScript Loop

A loop in JavaScript is used to repeat a block of code multiple times, either for a specific number of times or until a condition is met.
Loops help automate repetitive tasks like iterating through arrays, processing lists, or repeating calculations.

◈ Types of Loops in JavaScript

JavaScript supports several loop types:

1. **for**
2. **while**
3. **do...while**
4. **for...in**
5. **for...of**

◈ 1. for Loop

Used when you know how many times to run the loop.

◈ Syntax:

```
for (initialization; condition; increment) {
  // code to be executed
}
```

◈ Example:

```
for (let i = 1; i <= 5; i++) {
  console.log("Count:", i);
}
```

◈ 2. while Loop

Used when the number of iterations is unknown, and you want to run a loop while a condition is true.

◈ Syntax:

```
while (condition) {
  // code to be executed
}
```

◈ Example:

```
let i = 1;
while (i <= 5) {
  console.log("While count:", i);
  i++;
}
```

◈ 3. do...while Loop

Similar to **while**, but the code block runs at least once before checking the condition.

◈ Syntax:

```
do {
  // code to be executed
} while (condition);
```

◈ Example:

```
let i = 1;
do {
  console.log("Do While count:", i);
  i++;
} while (i <= 5);
```

◈ 4. for...in Loop

Used to loop through the properties of an object.

◈ Example:

```
const student = { name: "Faruk", age: 28, course: "Web Dev" };

for (let key in student) {
  console.log(key + ": " + student[key]);
}
```

◈ 5. for...of Loop

Used to loop through the values of an iterable (like an array or string).

◈ Example:

```
const colors = ["Red", "Green", "Blue"];

for (let color of colors) {
  console.log(color);
}
```

◈ Break and Continue

- break: exits the loop entirely.
- continue: skips the current iteration and goes to the next one.

◈ Example (Break)

```
for (let i = 1; i <= 10; i++) {
  if (i === 5) break;
  console.log(i);
}
```

◈ Example (Continue):

```
for (let i = 1; i <= 5; i++) {
  if (i === 3) continue;
  console.log(i);
}
```

◈ Summary Table

Loop Type	Use Case
`for`	Fixed number of iterations
`while`	Loop until condition becomes false
`do...while`	Always runs once, then checks
`for...in`	Loop through object keys
`for...of`	Loop through values of iterable

🧩 Real-Life Usage Examples

- Looping through a list of products
- Displaying students in a table
- Counting clicks or action
- Processing form input fields
- Repeating animations

Homework/Practice Task

1. Write a for loop that prints numbers from 1 to 10 in the console.
2. Use a for loop to print only even numbers (2, 4, 6, ...) from 1 to 20.
3. Use a while loop to calculate the sum of numbers from 1 to 10.
4. Example output: Sum = 55
5. Write a do...while loop that prints:

```
Count is: 1
Count is: 2
...
Count is: 5
```

Lecture 19

JavaScript DOM

◈ What is the DOM?

DOM stands for **Document Object Model**.
It is a **programming interface for web documents**.

When a web page is loaded, the browser creates a **DOM of the page**, which represents the page as a **tree structure** where each node is an object representing a part of the document (HTML elements, attributes, text, etc.).

◈ Why Use DOM in JavaScript?

With JavaScript and the DOM, you can:

- Access and change HTML elements
- Modify content and styles
- Add or remove elements dynamically
- Respond to user events (clicks, typing, etc.)

◈ DOM Structure Example

```html
<!DOCTYPE html>
<html>
  <head><title>My Page</title></head>
  <body>
    <h1 id="title">Hello, DOM!</h1>
    <p class="info">Learning DOM manipulation.</p>
    <button onclick="changeText()">Click Me</button>
  </body>
</html>
```

DOM Tree Representation:

```
html
 ├── head
 │    └── title
 └── body
      ├── h1 (id="title")
      ├── p (class="info")
      └── button
```

🔷 1. Accessing Elements

✅ By ID

```
let heading = document.getElementById("title");
console.log(heading.innerText);
```

✅ By Class Name

```
let info = document.getElementsByClassName("info")[0];
console.log(info.innerText);
```

✅ By Tag Name

```
let paragraphs = document.getElementsByTagName("p");
console.log(paragraphs.length);
```

✅ By Query Selector

```
let heading = document.querySelector("#title");
let info = document.querySelector(".info");
```

🔷 2. Changing Content

✅ innerText

```
document.getElementById("title").innerText = "Welcome to DOM!";
```

✅ innerHTML

```
document.querySelector(".info").innerHTML = "<strong>New content</strong>";
```

🔷 3. Changing Style

```
document.getElementById("title").style.color = "blue";
document.querySelector(".info").style.fontSize = "20px";
```

🔷 4. Changing Attributes

```
document.querySelector("img").src = "newimage.jpg";
```

🔷 5. Creating and Appending Elements

```
let newPara = document.createElement("p");
newPara.innerText = "I was added with JavaScript!";
document.body.appendChild(newPara);
```

🔷 6. Removing Elements

```
let oldPara = document.querySelector("p");
document.body.removeChild(oldPara);
```

🔷 7. DOM Events

You can make pages interactive using events.

✅ Example: Button Click

HTML:

```
<button onclick="showMessage()">Click Me</button>
<p id="msg"></p>
```

JavaScript:

```
function showMessage() {
  document.getElementById("msg").innerText = "You clicked the button!";
}
```

🔷 Summary Table

Task	Method/Property
Access by ID	`getElementById()`
Access by Class	`getElementsByClassName()`
Change Text	`element.innerText`
Change HTML	`element.innerHTML`
Change Style	`element.style.property`
Add Element	`createElement()` + `appendChild()`
Remove Element	`removeChild()`
Add Click Functionality	`onclick` or `addEventListener()`

Lecture 20

JavaScript DOM

✅ 1. Change Heading Text

```
<!DOCTYPE html>
<html>
<head>
    <title>Task 1: Change Heading</title>
</head>
<body>
    <h1 id="mainHeading">Original Heading</h1>
    <button onclick="changeHeading()">Change Heading</button>

    <script>
        function changeHeading() {
            document.getElementById("mainHeading").innerText = "DOM is Powerful!";
        }
    </script>
</body>
</html>
```

✅ 2. Change Style on Button Click

```
<!DOCTYPE html>
<html>
<head>
    <title>Task 2: Change Style</title>
</head>
<body>
    <p id="demo">This paragraph will change style.</p>
    <button onclick="changeStyle()">Change Style</button>

    <script>
        function changeStyle() {
            const para = document.getElementById("demo");
            para.style.color = "red";
            para.style.backgroundColor = "yellow";
        }
    </script>
</body>
</html>
```

✅ 3. Show/Hide Paragraph

```
<!DOCTYPE html>
<html>
<head>
    <title>Task 3: Toggle Paragraph</title>
</head>
<body>
    <p id="myPara">This paragraph can be toggled.</p>
    <button onclick="togglePara()">Toggle Paragraph</button>

    <script>
        function togglePara() {
            const para = document.getElementById("myPara");
            if (para.style.display === "none") {
                para.style.display = "block";
            } else {
                para.style.display = "none";
            }
        }
    </script>
</body>
</html>
```

✅ 4. Get Input Value

```
<!DOCTYPE html>
<html>
<head>
    <title>Task 4: Show Input</title>
</head>
<body>
    <input type="text" id="userInput" placeholder="Enter something">
    <button onclick="showInput()">Show Input</button>
    <p id="result"></p>

    <script>
        function showInput() {
            const input = document.getElementById("userInput").value;
            document.getElementById("result").innerText = input;
        }
    </script>
</body>
</html>
```

✅ 5. Change Image Source

```
<!DOCTYPE html>
<html>
<head>
    <title>Task 5: Change Image</title>
</head>
<body>
    <img id="myImage" src="img1.jpg" alt="Sample image" width="200">
    <button onclick="changeImage()">Change Image</button>

    <script>
        function changeImage() {
            document.getElementById("myImage").src = "img2.jpg";
        }
    </script>
</body>
</html>
```

✅ 6. Add List Item Dynamically

```
<!DOCTYPE html>
<html>
<head>
    <title>Task 6: Add List Item</title>
</head>
<body>
    <ul id="myList">
        <li>First item</li>
        <li>Second item</li>
    </ul>
    <button onclick="addItem()">Add Item</button>

    <script>
        function addItem() {
            const li = document.createElement("li");
            li.innerText = "New Item";
            document.getElementById("myList").appendChild(li);
        }
    </script>
</body>
</html>
```

✅ 7. Count Button Clicks

```
<!DOCTYPE html>
<html>
<head>
    <title>Task 7: Count Clicks</title>
</head>
<body>
    <button onclick="countClicks()">Click Me</button>
    <p id="clicks">Clicked 0 times</p>

    <script>
        let count = 0;
        function countClicks() {
            count++;
            document.getElementById("clicks").innerText = "Clicked " + count + " times";
        }
    </script>
</body>
</html>
```

✅ 8. Change Background Color

```
<!DOCTYPE html>
<html>
<head>
    <title>Task 8: Change Background</title>
</head>
<body>
    <button onclick="changeBG()">Change Background</button>

    <script>
        function changeBG() {
            const colors = ["lightblue", "lightgreen", "lightpink", "lightgray"];
            const random = Math.floor(Math.random() * colors.length);
            document.body.style.backgroundColor = colors[random];
        }
    </script>
</body>
</html>
```

✅ 9. Create and Remove Elements

```
<!DOCTYPE html>
<html>
<head>
    <title>Task 9: Add/Remove Paragraph</title>
</head>
<body>
    <div id="container"></div>
    <button onclick="addPara()">Add Paragraph</button>
    <button onclick="removePara()">Remove Paragraph</button>

    <script>
        function addPara() {
            const p = document.createElement("p");
            p.innerText = "New paragraph created.";
            p.id = "dynamicPara";
            document.getElementById("container").appendChild(p);
        }

        function removePara() {
            const p = document.getElementById("dynamicPara");
            if (p) p.remove();
        }
    </script>
</body>
</html>
```

✅ 10. Loop Through Elements and Highlight

```
<!DOCTYPE html>
<html>
<head>
    <title>Task 10: Highlight Items</title>
    <style>
        .item {
            margin: 5px;
            padding: 5px;
            border: 1px solid #ccc;
        }
    </style>
</head>
<body>
    <div class="item">Item 1</div>
    <div class="item">Item 2</div>
    <div class="item">Item 3</div>
    <button onclick="highlightItems()">Highlight Items</button>

    <script>
        function highlightItems() {
            const items = document.querySelectorAll(".item");
            items.forEach(function(item) {
                item.style.backgroundColor = "lightgreen";
            });
        }
    </script>
</body>
</html>
```

Lecture 21

jQuery, Owl Carousel Slider, SlickNav

◈ What is jQuery?

jQuery is a fast, small, and feature-rich JavaScript library. It simplifies HTML document traversing, event handling, animation, and Ajax with an easy-to-use syntax.

◈ Why Use jQuery?

- Simplifies JavaScript code
- Cross-browser compatible
- Easy DOM manipulation
- Built-in animation and effects
- Ajax support

How to Use Owl Carousel Slider

✅ Step 1: Include Required Files

Add these links in your <head> and before your closing </body> tag:

```
<!-- CSS -->
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/OwlCarousel2/2.3.4/assets/owl.carousel.min.css" />
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/OwlCarousel2/2.3.4/assets/owl.theme.default.min.
css" />

<!-- jQuery -->
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<!-- Owl Carousel JS -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/OwlCarousel2/2.3.4/owl.carousel.min.js"></script>
```

✅ Step 2: HTML Structure

Create a container with .owl-carousel class and add your items inside:

html

```
<div class="owl-carousel">
  <div><img src="img1.jpg" alt="Image 1"></div>
  <div><img src="img2.jpg" alt="Image 2"></div>
  <div><img src="img3.jpg" alt="Image 3"></div>
</div>
```

✅ Step 3: Initialize Owl Carousel

Use jQuery to activate the slider:

html

```
<script>
  $(document).ready(function(){
    $(".owl-carousel").owlCarousel({
      loop: true,
      margin: 10,
      nav: true,
      autoplay: true,
      autoplayTimeout: 3000,
      responsive:{
        0:{ items:1 },
        600:{ items:2 },
        1000:{ items:3 }
      }
    });
  });
</script>
```

✅ Optional Custom Styling

Add some basic CSS for layout (if needed):

html

```
<style>
  .owl-carousel div {
    text-align: center;
    padding: 10px;
  }
  .owl-carousel img {
    width: 100%;
    border-radius: 10px;
  }
</style>
```

📱 How to Add SlickNav (Mobile Menu)

✅ Step 1: Include Required Files

Add these in your HTML <head> or before closing </body> tag:

```
<!-- SlickNav CSS -->
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/SlickNav/1.0.10/slicknav.min.css" />

<!-- jQuery (Required) -->
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<!-- SlickNav JS -->
<script src="https://cdnjs.cloudflare.com/ajax/libs/SlickNav/1.0.10/jquery.slicknav.min.js"></script>
```

✅ Step 2: Create a Navigation Menu

Your regular desktop menu should look like this:

html

```
<ul id="menu">
  <li><a href="#">Home</a></li>
  <li><a href="#">About</a></li>
  <li><a href="#">Services</a></li>
  <li><a href="#">Contact</a></li>
</ul>
```

✅ Step 3: Initialize SlickNav

Add this script at the bottom of your page (after jQuery and SlickNav):

```html
<script>
  $(document).ready(function(){
    $('#menu').slicknav();
  });
</script>
```

✅ Optional Styling (CSS)
You can hide the original menu and show SlickNav on mobile with custom styles if needed.

```html
<style>
  /* Optional: Hide the default menu on small screens */
  @media (max-width: 768px) {
    #menu {
      display: none;
    }
  }
</style>
```

✅ Customize SlickNav (Optional Settings)
You can customize the menu behavior like this:

```javascript
$('#menu').slicknav({
  label: '☰ Menu',
  duration: 400,
  prependTo: 'body'
});
```

Lecture 22

Bootstrap

Introduction to Bootstrap

🎯 Learning Objectives

By the end of this lecture, students will be able to:

- Understand what Bootstrap is
- Use Bootstrap for responsive web design
- Apply Bootstrap components and utilities
- Build a responsive layout using Bootstrap grid system

🧠 1. What is Bootstrap?

Bootstrap is a free and open-source front-end framework used to develop responsive and mobile-first websites using:

- HTML
- CSS
- JavaScript

🔍 Features:

- Pre-designed components
- Responsive grid system
- Powerful utility classes
- JavaScript plugins (modals, carousels, etc.)

📄 2. How to Include Bootstrap

🧱 Using CDN (Recommended for Beginners):

```
<!-- Bootstrap CSS -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

<!-- Bootstrap JS -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
```

📦 3. Bootstrap Grid System

Bootstrap uses a 12-column grid system.

📄 Basic Grid Example:

```
<div class="container">
  <div class="row">
    <div class="col-md-6">Column 1</div>
    <div class="col-md-6">Column 2</div>
  </div>
</div>
```

🔢 Column Sizes:

- `col-` (Extra small: <576px)
- `col-sm-` (≥576px)
- `col-md-` (≥768px)
- `col-lg-` (≥992px)
- `col-xl-` (≥1200px)

🧩 4. Common Bootstrap Components

✅ Buttons

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

✅ Navbar

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Brand</a>
</nav>
```

✅ Cards

```
<div class="card" style="width: 18rem;">
  <div class="card-body">
    <h5 class="card-title">Card Title</h5>
    <p class="card-text">Some content.</p>
  </div>
</div>
```

✅ Alerts

```
<div class="alert alert-success" role="alert">
  This is a success alert!
</div>
```

🧰 5. Bootstrap Utility Classes

Utility classes help you apply styling quickly.

Class	Purpose
`m-3`	Margin
`p-2`	Padding
`text-center`	Center text
`bg-primary`	Background color
`d-flex`	Flex display

🖌️ 6. Responsive Behavior

Bootstrap components and layout automatically adjust for:

- Mobiles
- Tablets
- Laptops
- Desktops

Use responsive classes like:
d-none d-md-block (hide on small screens, show on medium+)

✅ 7. Assignment

Task: Create a responsive landing page using Bootstrap
Requirements:

- Header with navigation bar
- Banner section with text and button
- 3-column service section (responsive)
- Footer with contact info

Use Bootstrap components like:

- Navbar
- Grid system
- Cards
- Buttons
- Alerts (optional)

Lecture 22

Responsive website using media query

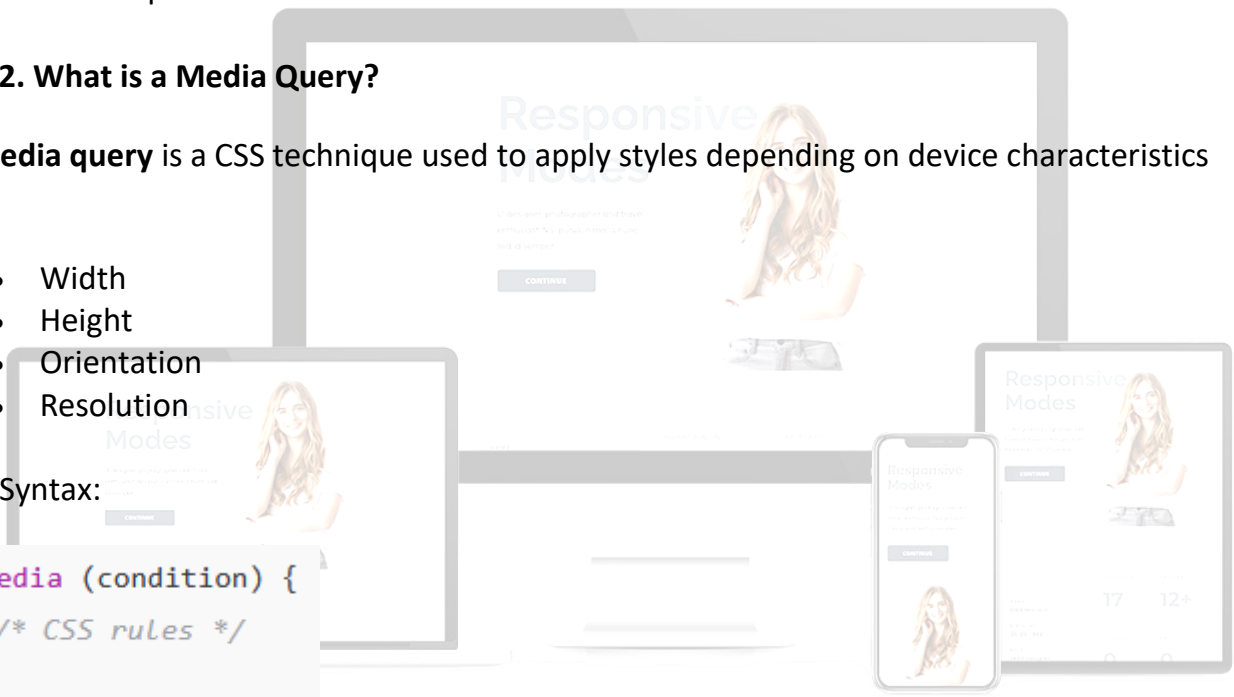
🧠 1. What is Responsive Web Design?

Responsive Web Design is a web development approach that ensures a website looks good and functions well on all devices (desktops, tablets, and smartphones).

🔍 Key Characteristics:

- Fluid grid layout
- Flexible images
- Media queries

💠 2. What is a Media Query?



A **media query** is a CSS technique used to apply styles depending on device characteristics like:

- Width
- Height
- Orientation
- Resolution

📝 Syntax:

```
@media (condition) {
  /* CSS rules */
}
```

🖌️ 3. Common Media Query Breakpoints

Device	Width Range
Extra Small (Mobile)	0 - 575px
Small (Tablet)	576px - 767px
Medium (Tablet/Laptop)	768px - 991px
Large (Desktop)	992px - 1199px
Extra Large (Large Desktop)	1200px+

📄 4. Media Query Examples

▨ Example 1: Mobile First


```css
/* Base styles for mobile */
body {
  font-size: 14px;
}

/* Tablet */
@media (min-width: 768px) {
  body {
    font-size: 16px;
  }
}

/* Desktop */
@media (min-width: 992px) {
  body {
    font-size: 18px;
  }
}
```

▨ Example 2: Change layout

```css
.container {
  display: flex;
  flex-direction: column;
}

@media (min-width: 768px) {
  .container {
    flex-direction: row;
  }
}
```

🧰 5. Tips for Writing Media Queries

- Use mobile-first approach (min-width)
- Avoid fixed widths — use %, vw, or rem
- Combine media queries for performanc
- Test on real devices and using browser dev tools

🛠️ 6. Tools for Testing Responsiveness

- Chrome DevTools → Toggle device toolbar
- Firefox Responsive Design Mode
- Online tools: Responsinator, Am I Responsive?

✅ 7. Assignment

Task: Create a simple webpage layout with a header, sidebar, content area, and footer. Make the layout responsive using media queries.

Lecture 23

Bootstrap

Introduction to Bootstrap

🎯 Learning Objectives

By the end of this lecture, students will be able to:

- Understand what Bootstrap is
- Use Bootstrap for responsive web design
- Apply Bootstrap components and utilities
- Build a responsive layout using Bootstrap grid system

🧠 1. What is Bootstrap?

Bootstrap is a free and open-source front-end framework used to develop responsive and mobile-first websites using:

- HTML
- CSS
- JavaScript

🔍 Features:

- Pre-designed components
- Responsive grid system
- Powerful utility classes
- JavaScript plugins (modals, carousels, etc.)

📄 2. How to Include Bootstrap

🧱 Using CDN (Recommended for Beginners):

```
<!-- Bootstrap CSS -->
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">

<!-- Bootstrap JS -->
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
```

📦 3. Bootstrap Grid System

Bootstrap uses a 12-column grid system.

📄 Basic Grid Example:

```
<div class="container">
  <div class="row">
    <div class="col-md-6">Column 1</div>
    <div class="col-md-6">Column 2</div>
  </div>
</div>
```

🔢 Column Sizes:

- `col-` (Extra small: <576px)
- `col-sm-` (≥576px)
- `col-md-` (≥768px)
- `col-lg-` (≥992px)
- `col-xl-` (≥1200px)

🧩 4. Common Bootstrap Components

✅ Buttons

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

✅ Navbar

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Brand</a>
</nav>
```

✅ Cards

```
<div class="card" style="width: 18rem;">
  <div class="card-body">
    <h5 class="card-title">Card Title</h5>
    <p class="card-text">Some content.</p>
  </div>
</div>
```

✅ Alerts

```
<div class="alert alert-success" role="alert">
  This is a success alert!
</div>
```

🧰 5. Bootstrap Utility Classes

Utility classes help you apply styling quickly.

Class	Purpose
`m-3`	Margin
`p-2`	Padding
`text-center`	Center text
`bg-primary`	Background color
`d-flex`	Flex display

🖌️ 6. Responsive Behavior

Bootstrap components and layout automatically adjust for:

- Mobiles
- Tablets
- Laptops
- Desktops

Use responsive classes like:
d-none d-md-block (hide on small screens, show on medium+)

✅ 7. Assignment

Task: Create a responsive landing page using Bootstrap
Requirements:

- Header with navigation bar
- Banner section with text and button
- 3-column service section (responsive)
- Footer with contact info

Use Bootstrap components like:

- Navbar
- Grid system
- Cards
- Buttons
- Alerts (optional)

Lecture 24

PHP Introduction, PHP Install, PHP Syntax, PHP Comments
PHP Variables, PHP Data Types,

Learn PHP

PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.
PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.

What is PHP?

- PHP is an acronym for "PHP: Hypertext Preprocessor"
- PHP is a widely-used, open source scripting language
- PHP scripts are executed on the server
- PHP is free to download and use

What is a PHP File?

- PHP files can contain text, HTML, CSS, JavaScript, and PHP code
- PHP code is executed on the server, and the result is returned to the browser as plain HTML
- PHP files have extension ".php"

What Can PHP Do?

- PHP can generate dynamic page content
- PHP can create, open, read, write, delete, and close files on the server
- PHP can collect form data
- PHP can send and receive cookies
- PHP can add, delete, modify data in your database
- PHP can be used to control user-access
- PHP can encrypt data

PHP Installation

What Do I Need?
To start using PHP, you can:

- Find a web host with PHP and MySQL support
- Install a web server on your own PC, and then install PHP and MySQL

Use a Web Host With PHP Support

If your server has activated support for PHP you do not need to do anything.
Just create some .php files, place them in your web directory, and the server will automatically parse them for you.
You do not need to compile anything or install any extra tools.
Because PHP is free, most web hosts offer PHP support.

Set Up PHP on Your Own PC

However, if your server does not support PHP, you must:

- install a web server
- install PHP
- install a database, such as MySQL

PHP Syntax

A PHP script is executed on the server, and the plain HTML result is sent back to the browser.

Basic PHP Syntax
A PHP script can be placed anywhere in the document.
A PHP script starts with <?php and ends with ?>:

```
<?php
// PHP code goes here
?>
```

PHP Comments

A comment in PHP code is a line that is not executed as a part of the program. Its only purpose is to be read by someone who is looking at the code.

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

# This is also a single-line comment

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

PHP Variables

Creating (Declaring) PHP Variables

In PHP, a variable starts with the $ sign, followed by the name of the variable:

```
$x = 5;
$y = "John";
```

In the example above, the variable $x will hold the value 5, and the variable $y will hold the value "John".

A variable can have a short name (like $x and $y) or a more descriptive name ($age, $carname, $total_volume).

Rules for PHP variables:

- A variable starts with the $ sign, followed by the name of the variable
- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive ($age and $AGE are two different variables)

Remember that PHP variable names are case-sensitive!

Output Variables

The PHP echo statement is often used to output data to the screen.

The following example will show how to output text and a variable:

```
$txt = "W3Schools.com";
echo "I love $txt!";
```

The following example will produce the same output as the example above:

```
$txt = "W3Schools.com";
echo "I love " . $txt . "!";
```

The following example will output the sum of two variables:

```
$x = 5;
$y = 4;
echo $x + $y;
```

Variable Types

PHP has no command for declaring a variable, and the data type depends on the value of the variable.

```php
$x = 5;       // $x is an integer
$y = "John"; // $y is a string
echo $x;
echo $y;
```

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

Get the Type

To get the data type of a variable, use the var_dump() function.

The var_dump() function returns the data type and the value:

```php
$x = 5;
var_dump($x);
```

See what var_dump() returns for other data types:

```php
<!DOCTYPE html>
<html>
<body>

<pre>

<?php
var_dump(5);
var_dump("John");
var_dump(3.14);
var_dump(true);
var_dump([2, 3, 56]);
var_dump(NULL);
?>

</pre>

</body>
</html>
```

```
int(5)
string(4) "John"
float(3.14)
bool(true)
array(3) {
  [0]=>
  int(2)
  [1]=>
  int(3)
  [2]=>
  int(56)
}
NULL
```

Assign String to a Variable

Assigning a string to a variable is done with the variable name followed by an equal sign and the string:

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = "John";
echo $x;
?>

</body>
</html>
```

John

Assign Multiple Values

You can assign the same value to multiple variables in one line:

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = $y = $z = "Fruit";

echo $x;
echo $y;
echo $z;

?>

</body>
</html>
```

FruitFruitFruit

PHP Variables Scope

In PHP, variables can be declared anywhere in the script.

The scope of a variable is the part of the script where the variable can be referenced/used.

PHP has three different variable scopes:

- local
- global
- static

Global and Local Scope

A variable declared outside a function has a GLOBAL SCOPE and can only be accessed outside a function:

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = 5; // global scope

function myTest() {
  // using x inside this function will generate an
error
  echo "<p>Variable x inside function is: $x</p>";
}
myTest();

echo "<p>Variable x outside function is: $x</p>";
?>

</body>
</html>
```

Variable x inside function is:

Variable x outside function is: 5

A variable declared within a function has a LOCAL SCOPE and can only be accessed within that function:

```php
<!DOCTYPE html>
<html>
<body>

<?php
function myTest() {
  $x = 5; // local scope
  echo "<p>Variable x inside function is: $x</p>";
}
myTest();

// using x outside the function will generate an
error
echo "<p>Variable x outside function is: $x</p>";
?>

</body>
</html>
```

Variable x inside function is: 5

Variable x outside function is:

PHP The global Keyword

The global keyword is used to access a global variable from within a function.
To do this, use the global keyword before the variables (inside the function):

```php
<!DOCTYPE html>
<html>
<body>

<?php
$x = 5;
$y = 10;

function myTest() {
  global $x, $y;
  $y = $x + $y;
}

myTest();  // run function
echo $y; // output the new value for variable $y
?>

</body>
</html>
```

15

PHP also stores all global variables in an array called $GLOBALS[index]. The index holds the name of the variable. This array is also accessible from within functions and can be used to update global variables directly.

The example above can be rewritten like this:

```php
<!DOCTYPE html>
<html>
<body>

<?php
$x = 5;
$y = 10;

function myTest() {
  $GLOBALS['y'] = $GLOBALS['x'] + $GLOBALS['y'];
}

myTest();
echo $y;
?>

</body>
</html>
```

```
15
```

PHP The static Keyword

Normally, when a function is completed/executed, all of its variables are deleted. However, sometimes we want a local variable NOT to be deleted. We need it for a further job.
To do this, use the static keyword when you first declare the variable:

```php
<!DOCTYPE html>
<html>
<body>

<?php
function myTest() {
  static $x = 0;
  echo $x;
  $x++;
}

myTest();
echo "<br>";
myTest();
echo "<br>";
myTest();
?>

</body>
</html>
```

```
0
1
2
```

PHP Data Types
Variables can store data of different types, and different data types can do different things.

PHP supports the following data types:

- String
- Integer
- Float (floating point numbers - also called double)
- Boolean
- Array
- Object
- NULL
- Resource

Getting the Data Type
You can get the data type of any object by using the var_dump() function.
The var_dump() function returns the data type and the value:

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = 5;
var_dump($x);
?>

</body>
</html>
```

```
int(5)
```

PHP String
A string is a sequence of characters, like "Hello world!".
A string can be any text inside quotes. You can use single or double quotes:

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = "Hello world!";
$y = 'Hello world!';

var_dump($x);
echo "<br>";
var_dump($y);
?>

</body>
</html>
```

```
string(12) "Hello world!"
string(12) "Hello world!"
```

PHP Integer
An integer data type is a non-decimal number between -2,147,483,648 and 2,147,483,647.

Rules for integers:

- An integer must have at least one digit
- An integer must not have a decimal point
- An integer can be either positive or negative
- Integers can be specified in: decimal (base 10), hexadecimal (base 16), octal (base 8), or binary (base 2) notation

In the following example $x is an integer. The PHP var_dump() function returns the data type and value:

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = 5985;
var_dump($x);
?>

</body>
</html>
```

```
int(5985)
```

PHP Float

A float (floating point number) is a number with a decimal point or a number in exponential form.

In the following example $x is a float. The PHP var_dump() function returns the data type and value:

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = 10.365;
var_dump($x);
?>

</body>
</html>
```

```
float(10.365)
```

PHP Boolean

A Boolean represents two possible states: TRUE or FALSE.

```
<!DOCTYPE html>
<html>
<body>

<?php
$x = true;
var_dump($x);
?>

</body>
</html>
```

```
bool(true)
```

PHP Array

An array stores multiple values in one single variable.

In the following example $cars is an array. The PHP var_dump() function returns the data type and value:

```
<!DOCTYPE html>
<html>
<body>

<?php
$cars = array("Volvo","BMW","Toyota");
var_dump($cars);
?>

</body>
</html>
```

```
array(3) {
  [0]=>
  string(5) "Volvo"
  [1]=>
  string(3) "BMW"
  [2]=>
  string(6) "Toyota"
}
```

Homework/Practice Task

1. Write a PHP script that prints your name, age, and profession using echo.
2. Declare a variable of each data type (string, int, float, bool, array) and use var_dump() to display type and value.
3. Use variables and string interpolation to output a formatted sentence.

   Expected Output:

   ```pgsql
   Hello, my name is Faruk and I love coding in PHP!
   ```

4. Create a variable outside a function and try to use it inside without the global keyword. Then fix it using the global keyword.
5. Create a function that uses a static variable to count how many times it has been called.

   Expected Output:

   ```bash
   This function has been called 1 times.
   This function has been called 2 times.
   ...
   ```

Lecture 25

PHP Operators, Conditional Statements, Switch Statement
PHP Loops, PHP Functions, PHP Arrays

PHP Operators

Operators are used to perform operations on variables and values.

PHP divides the operators in the following groups:

- Arithmetic operators
- Assignment operators
- Comparison operators
- Increment/Decrement operators
- Logical operators
- String operators
- Array operators
- Conditional assignment operators

PHP Arithmetic Operators

The PHP arithmetic operators are used with numeric values to perform common arithmetical operations, such as addition, subtraction, multiplication etc

Operator	Name	Example	Result
+	Addition	$x + $y	Sum of $x and $y
-	Subtraction	$x - $y	Difference of $x and $y
*	Multiplication	$x * $y	Product of $x and $y
/	Division	$x / $y	Quotient of $x and $y
%	Modulus	$x % $y	Remainder of $x divided by $y
**	Exponentiation	$x ** $y	Result of raising $x to the $y'th power

PHP Assignment Operators

The PHP assignment operators are used with numeric values to write a value to a variable.

The basic assignment operator in PHP is "=". It means that the left operand gets set to the value of the assignment expression on the right.

Assignment	Same as...	Description
x = y	x = y	The left operand gets set to the value of the expression on the right
x += y	x = x + y	Addition
x -= y	x = x - y	Subtraction
x *= y	x = x * y	Multiplication
x /= y	x = x / y	Division
x %= y	x = x % y	Modulus

PHP Comparison Operators

The PHP comparison operators are used to compare two values (number or string):

Operator	Name	Example	Result
==	Equal	$x == $y	Returns true if $x is equal to $y
===	Identical	$x === $y	Returns true if $x is equal to $y, and they are of the same type
!=	Not equal	$x != $y	Returns true if $x is not equal to $y
<>	Not equal	$x <> $y	Returns true if $x is not equal to $y
!==	Not identical	$x !== $y	Returns true if $x is not equal to $y, or they are not of the same type
>	Greater than	$x > $y	Returns true if $x is greater than $y
<	Less than	$x < $y	Returns true if $x is less than $y
>=	Greater than or equal to	$x >= $y	Returns true if $x is greater than or equal to $y
<=	Less than or equal to	$x <= $y	Returns true if $x is less than or equal to $y
<=>	Spaceship	$x <=> $y	Returns an integer less than, equal to, or greater than zero, depending on if $x is less than, equal to, or greater than $y. Introduced in PHP 7.

PHP Increment / Decrement Operators

The PHP increment operators are used to increment a variable's value.
The PHP decrement operators are used to decrement a variable's value.

Operator	Same as...	Description
++$x	Pre-increment	Increments $x by one, then returns $x
$x++	Post-increment	Returns $x, then increments $x by one
--$x	Pre-decrement	Decrements $x by one, then returns $x
$x--	Post-decrement	Returns $x, then decrements $x by one

PHP Logical Operators

The PHP logical operators are used to combine conditional statements.

Operator	Name	Example	Result
and	And	$x and $y	True if both $x and $y are true
or	Or	$x or $y	True if either $x or $y is true
xor	Xor	$x xor $y	True if either $x or $y is true, but not both
&&	And	$x && $y	True if both $x and $y are true
||	Or	$x || $y	True if either $x or $y is true
!	Not	!$x	True if $x is not true

PHP String Operators

PHP has two operators that are specially designed for strings.

Operator	Name	Example	Result
.	Concatenation	$txt1 . $txt2	Concatenation of $txt1 and $txt2
.=	Concatenation assignment	$txt1 .= $txt2	Appends $txt2 to $txt1

PHP if Statements

Conditional statements are used to perform different actions based on different conditions.

PHP Conditional Statements

Very often when you write code, you want to perform different actions for different conditions. You can use conditional statements in your code to do this.

In PHP we have the following conditional statements:

- if statement - executes some code if one condition is true
- if...else statement - executes some code if a condition is true and another code if that condition is false
- if...elseif...else statement - executes different codes for more than two conditions
- switch statement - selects one of many blocks of code to be executed

PHP - The if Statement

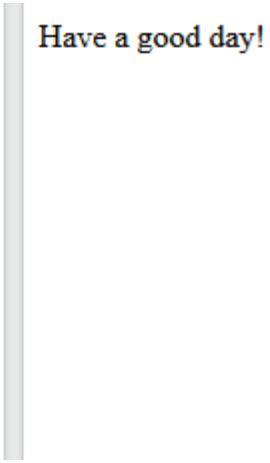
The if statement executes some code if one condition is true.

Syntax

```
if (condition) {
  // code to be executed if condition is true;
}
```

Output "Have a good day!" if 5 is larger than 3:

```
<!DOCTYPE html>
<html>
<body>

<?php
if (5 > 3) {
  echo "Have a good day!";
}
?>

</body>
</html>
```



We can also use variables in the if statement:
Output "Have a good day!" if $t is less than 20:

```
<!DOCTYPE html>
<html>
<body>

<?php
$t = 14;

if ($t < 20) {
  echo "Have a good day!";
}
?>

</body>
</html>
```

Have a good day!

PHP if Operators

Comparison Operators

If statements usually contain conditions that compare two values.

Check if $t is equal to 14:

```
<!DOCTYPE html>
<html>
<body>

<?php
$t = 14;

if ($t == 14) {
  echo "Have a good day!";
}
?>

</body>
</html>
```

Have a good day!

To compare two values, we need to use a comparison operator.
Here are the PHP comparison operators to use in if statements:

Operator	Name	Result
==	Equal	Returns true if the values are equal
===	Identical	Returns true if the values and data types are identical
!=	Not equal	Returns true if the values are not equal
<>	Not equal	Returns true if the values are not equal
!==	Not identical	Returns true if the values or data types are not identical
>	Greater than	Returns true if the first value is greater than the second value
<	Less than	Returns true if the first value is less than the second value
>=	Greater than or equal to	Returns true if the first value is greater than, or equal to, the second value
<=	Less than or equal to	Returns true if the first value is less than, or equal to, the second value

Logical Operators
To check more than one condition, we can use logical operators, like the && operator:

Check if $a is greater than $b, AND if $a is less than $c:

```
<!DOCTYPE html>
<html>
<body>

<?php
$a = 200;
$b = 33;
$c = 500;

if ($a > $b && $a < $c ) {
  echo "Both conditions are true";
}
?>

</body>
</html>
```

Both conditions are true

Here are the PHP logical operators to use in if statements:

Operator	Name	Description
and	And	True if both conditions are true
&&	And	True if both conditions are true
or	Or	True if either condition is true
||	Or	True if either condition is true
xor	Xor	True if either condition is true, but not both
!	Not	True if condition is not true

We can compare as many conditions as we like in one if statement:
Check if $a is either 2, 3, 4, 5, 6, or 7:

```
<!DOCTYPE html>
<html>
<body>

<?php
$a = 5;

if ($a == 2 || $a == 3 || $a == 4 || $a == 5 || $a == 6 || $a == 7) {
  echo "$a is a number between 2 and 7";
}
?>

</body>
</html>
```

5 is a number between 2 and 7

PHP - The if...else Statement

The if...else statement executes some code if a condition is true and another code if that condition is false.

Syntax

```
if (condition) {
  // code to be executed if condition is true;
} else {
  // code to be executed if condition is false;
}
```

Output "Have a good day!" if the current time is less than 20, and "Have a good night!" otherwise:

```php
<!DOCTYPE html>
<html>
<body>

<?php
$t = date("H");

if ($t < "20") {
  echo "Have a good day!";
} else {
  echo "Have a good night!";
}
?>

</body>
</html>
```

Have a good day!

PHP - The if...elseif...else Statement

The if...elseif...else statement executes different codes for more than two conditions.

Syntax

```php
if (condition) {
  code to be executed if this condition is true;
} elseif (condition) {
  // code to be executed if first condition is false and this condition is true;
} else {
  // code to be executed if all conditions are false;
}
```

Output "Have a good morning!" if the current time is less than 10, and "Have a good day!" if the current time is less than 20. Otherwise it will output "Have a good night!":

```php
<!DOCTYPE html>
<html>
<body>

<?php
$t = date("H");
echo "<p>The hour (of the server) is " . $t;
echo ", and will give the following message:</p>";

if ($t < "10") {
  echo "Have a good morning!";
} elseif ($t < "20") {
  echo "Have a good day!";
} else {
  echo "Have a good night!";
}
?>

</body>
</html>
```

The hour (of the server) is 10, and will give the following message:
Have a good day!

PHP Shorthand if Statements

To write shorter code, you can write if statements on one line.

One-line if statement:

```
<!DOCTYPE html>
<html>
<body>

<?php
$a = 5;

if ($a < 10) $b = "Hello";

echo $b
?>

</body>
</html>
```

Hello

Short Hand If...Else

if...else statements can also be written in one line, but the syntax is a bit different.

One-line if...else statement:

```
<!DOCTYPE html>
<html>
<body>

<?php
$a = 13;

$b = $a < 10 ? "Hello" : "Good Bye";

echo $b;
?>

</body>
</html>
```

Good Bye

PHP Nested if Statement

You can have if statements inside if statements, this is called nested if statements.

```
<!DOCTYPE html>
<html>
<body>

<?php
$a = 13;

if ($a > 10) {
  echo "Above 10";
  if ($a > 20) {
    echo " and also above 20";
  } else {
    echo " but not above 20";
  }
}
?>

</body>
</html>
```

Above 10 but not above 20

PHP switch Statement

The switch statement is used to perform different actions based on different conditions.
Use the switch statement to select one of many blocks of code to be executed.

Syntax

```
switch (expression) {
  case label1:
    //code block
    break;
  case label2:
    //code block;
    break;
  case label3:
    //code block
    break;
  default:
    //code block
}
```

This is how it works:

- The expression is evaluated once
- The value of the expression is compared with the values of each case
- If there is a match, the associated block of code is executed
- The break keyword breaks out of the switch block
- The default code block is executed if there is no match

```php
<!DOCTYPE html>
<html>
<body>

<?php
$favcolor = "red";

switch ($favcolor) {
  case "red":
    echo "Your favorite color is red!";
    break;
  case "blue":
    echo "Your favorite color is blue!";
    break;
  case "green":
    echo "Your favorite color is green!";
    break;
  default:
    echo "Your favorite color is neither red, blue, nor green!";
}
?>

</body>
</html>
```

Your favorite color is red!

The break Keyword

When PHP reaches a break keyword, it breaks out of the switch block.
This will stop the execution of more code, and no more cases are tested.
The last block does not need a break, the block breaks (ends) there anyway.

```
<!DOCTYPE html>
<html>
<body>

<?php
$favcolor = "red";

switch ($favcolor) {
  case "red":
    echo "Your favorite color is red!";
  case "blue":
    echo "Your favorite color is blue!";
    break;
  case "green":
    echo "Your favorite color is green!";
    break;
  default:
    echo "Your favorite color is neither red, blue, nor green!";
}
?>

</body>
</html>
```

Your favorite color is red!Your favorite color is blue!

The default Keyword

The default keyword specifies the code to run if there is no case match:
If no cases get a match, the default block is executed:

```
<!DOCTYPE html>
<html>
<body>

<?php
$d = 4;

switch ($d) {
  case 6:
    echo "Today is Saturday";
    break;
  case 0:
    echo "Today is Sunday";
    break;
  default:
    echo "Looking forward to the Weekend";
}
?>

</body>
</html>
```

Looking forward to the Weekend

Putting the default block elsewhere than at the end of the switch block is allowed, but not recommended.

```
<!DOCTYPE html>
<html>
<body>

<?php
$d = 4;

switch ($d) {
  default:
    echo "Looking forward to the Weekend";
    break;
  case 6:
    echo "Today is Saturday";
    break;
  case 0:
    echo "Today is Sunday";
}
?>

</body>
</html>
```

Looking forward to the Weekend

Common Code Blocks

If you want multiple cases to use the same code block, you can specify the cases like this:

```
<!DOCTYPE html>
<html>
<body>

<?php
$d = 3;

switch ($d) {
  case 1:
  case 2:
  case 3:
  case 4:
  case 5:
    echo "The week feels so long!";
    break;
  case 6:
  case 0:
    echo "Weekends are the best!";
    break;
  default:
    echo "Something went wrong";
}
?>

</body>
</html>
```

The week feels so long!

Lecture 26

MySQL

Introduction to MySQL

Definition: MySQL is an open-source relational database management system (RDBMS) based on Structured Query Language (SQL).

Use Cases: Web applications, data warehousing, logging applications, e-commerce, and more

Key Concepts

- Database: Collection of related data organized in tables.
- Table: Structure with rows and columns storing data entries.
- Row (Record): Single data item in a table.
- Column (Field): Attribute describing the record.
- Primary Key: Unique identifier for a table row.
- Foreign Key: Reference to a primary key in another table.

MySQL Architecture

1. **Client Layer**: Applications communicate via connectors/drivers.
2. **Server Layer**:
    - **Connection Manager**
    - **Authentication & Security**
3. **Storage Engine Layer**:
    - **InnoDB** (transactional)
    - **MyISAM** (non-transactional)
4. **Operating System**: File system access

Installation & Setup

1. **Download & Install**: Download from official site or use package managers (apt, yum).
2. **Basic Configuration**: my.cnf for tuning parameters.
3. **Secure Installation**: mysql_secure_installation.
4. **Accessing MySQL**

    mysql -u root -p

Basic SQL Commands
Creating & Selecting Databases

```
CREATE DATABASE school;
USE school;
```

Table Operations

```
CREATE TABLE students (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  age INT,
  class VARCHAR(50)
);
DROP TABLE students;
```

Data Manipulation
Insert

```
INSERT INTO students (name, age, class)
VALUES ('Alice', 14, '8th Grade');
```

Select:

```
SELECT * FROM students;
SELECT name, age FROM students WHERE age > 12;
```

Update:

```
UPDATE students SET class = '9th Grade' WHERE id = 1;
```

Delete

```
DELETE FROM students WHERE id = 1;
```

Lecture 27 & 28

CRUD Project

Here's a complete CRUD (Create, Read, Update, Delete) project using procedural PHP and MySQL with 6 data fields. This example will manage a simple "Products" database.

1. Database Setup

First, let's create the MySQL database and table:

```sql
CREATE DATABASE php_crud;
USE php_crud;

CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    price DECIMAL(10,2) NOT NULL,
    quantity INT NOT NULL,
    category VARCHAR(50),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

INSERT INTO products (name, description, price, quantity, category) VALUES
('Laptop', 'High performance laptop with 16GB RAM', 999.99, 10, 'Electronics'),
('Smartphone', 'Latest smartphone with great camera', 699.99, 15, 'Electronics'),
('Headphones', 'Noise cancelling wireless headphones', 199.99, 20, 'Accessories'),
('Desk', 'Ergonomic office desk', 249.99, 5, 'Furniture'),
('Notebook', 'Premium quality notebook', 9.99, 50, 'Stationery'),
('Coffee Mug', 'Ceramic coffee mug with logo', 12.99, 30, 'Kitchen');
```

2. Database Connection (db.php)

```php
<?php
$host = 'localhost';
$dbname = 'php_crud';
$username = 'root';
$password = '';

try {
    $conn = new PDO("mysql:host=$host;dbname=$dbname", $username, $password);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch(PDOException $e) {
    echo "Connection failed: " . $e->getMessage();
}
?>
```

3. Create (create.php)

```php
<?php
include 'db.php';

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $name = $_POST['name'];
    $description = $_POST['description'];
    $price = $_POST['price'];
    $quantity = $_POST['quantity'];
    $category = $_POST['category'];

    try {
        $stmt = $conn->prepare("INSERT INTO products (name, description, price, quantity, catego
ry) VALUES (?, ?, ?, ?, ?)");
        $stmt->execute([$name, $description, $price, $quantity, $category]);

        header("Location: index.php");
        exit();
    } catch(PDOException $e) {
        echo "Error: " . $e->getMessage();
    }
}
?>
```

```html
<!DOCTYPE html>
<html>
<head>
    <title>Create Product</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="s
ylesheet">
</head>
<body>
    <div class="container mt-5">
        <h2>Create New Product</h2>
        <form method="post">
            <div class="mb-3">
                <label class="form-label">Name</label>
                <input type="text" class="form-control" name="name" required>
            </div>
            <div class="mb-3">
                <label class="form-label">Description</label>
                <textarea class="form-control" name="description"></textarea>
            </div>
            <div class="mb-3">
                <label class="form-label">Price</label>
                <input type="number" step="0.01" class="form-control" name="price" required>
            </div>
            <div class="mb-3">
                <label class="form-label">Quantity</label>
                <input type="number" class="form-control" name="quantity" required>
            </div>
            <div class="mb-3">
                <label class="form-label">Category</label>
                <input type="text" class="form-control" name="category">
            </div>
            <button type="submit" class="btn btn-primary">Submit</button>
            <a href="index.php" class="btn btn-secondary">Cancel</a>
        </form>
    </div>
</body>
</html>
```

4. Read (index.php)

```php
<?php
include 'db.php';

try {
    $stmt = $conn->prepare("SELECT * FROM products ORDER BY created_at DESC");
    $stmt->execute();
    $products = $stmt->fetchAll(PDO::FETCH_ASSOC);
} catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
?>

<!DOCTYPE html>
<html>
<head>
    <title>Product List</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
    <div class="container mt-5">
        <h2>Product List</h2>
        <a href="create.php" class="btn btn-success mb-3">Add New Product</a>

        <table class="table table-striped">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Name</th>
                    <th>Description</th>
                    <th>Price</th>
                    <th>Quantity</th>
                    <th>Category</th>
                    <th>Actions</th>
                </tr>
            </thead>
            <tbody>
                <?php foreach ($products as $product): ?>
                <tr>
                    <td><?php echo $product['id']; ?></td>
                    <td><?php echo htmlspecialchars($product['name']); ?></td>
                    <td><?php echo htmlspecialchars($product['description']); ?></td>
                    <td>$<?php echo number_format($product['price'], 2); ?></td>
                    <td><?php echo $product['quantity']; ?></td>
                    <td><?php echo htmlspecialchars($product['category']); ?></td>
                    <td>
                        <a href="edit.php?id=<?php echo $product['id']; ?>" class="btn btn-sm btn-warning">Edit</a>
                        <a href="delete.php?id=<?php echo $product['id']; ?>" class="btn btn-sm btn-danger" onclick="return confirm
                        ('Are you sure?')">Delete</a>
                    </td>
                </tr>
                <?php endforeach; ?>
            </tbody>
        </table>
    </div>
</body>
</html>
```

5. Update (edit.php)

```php
<?php
include 'db.php';

if (!isset($_GET['id'])) {
    header("Location: index.php");
    exit();
}

$id = $_GET['id'];

try {
    $stmt = $conn->prepare("SELECT * FROM products WHERE id = ?");
    $stmt->execute([$id]);
    $product = $stmt->fetch(PDO::FETCH_ASSOC);

    if (!$product) {
        header("Location: index.php");
        exit();
    }
} catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}

if ($_SERVER['REQUEST_METHOD'] == 'POST') {
    $name = $_POST['name'];
    $description = $_POST['description'];
    $price = $_POST['price'];
    $quantity = $_POST['quantity'];
    $category = $_POST['category'];

    try {
        $stmt = $conn->prepare("UPDATE products SET name = ?, description = ?, price = ?, quantity = ?, category = ? WHERE id = ?");
        $stmt->execute([$name, $description, $price, $quantity, $category, $id]);

        header("Location: index.php");
        exit();
    } catch(PDOException $e) {
        echo "Error: " . $e->getMessage();
    }
}
?>
```

```php
<!DOCTYPE html>
<html>
<head>
    <title>Edit Product</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
    <div class="container mt-5">
        <h2>Edit Product</h2>
        <form method="post">
            <div class="mb-3">
                <label class="form-label">Name</label>
                <input type="text" class="form-control" name="name" value="<?php echo htmlspecialchars($product['name']); ?>" required>
            </div>
            <div class="mb-3">
                <label class="form-label">Description</label>
                <textarea class="form-control" name="description"><?php echo htmlspecialchars($product['description']); ?></textarea>
            </div>
            <div class="mb-3">
                <label class="form-label">Price</label>
                <input type="number" step="0.01" class="form-control" name="price" value="<?php echo $product['price']; ?>" required>
            </div>
            <div class="mb-3">
                <label class="form-label">Quantity</label>
                <input type="number" class="form-control" name="quantity" value="<?php echo $product['quantity']; ?>" required>
            </div>
            <div class="mb-3">
                <label class="form-label">Category</label>
                <input type="text" class="form-control" name="category" value="<?php echo htmlspecialchars($product['category']); ?>">
            </div>
            <button type="submit" class="btn btn-primary">Update</button>
            <a href="index.php" class="btn btn-secondary">Cancel</a>
        </form>
    </div>
</body>
</html>
```

6. Delete (delete.php)

```php
<?php
include 'db.php';

if (!isset($_GET['id'])) {
    header("Location: index.php");
    exit();
}

$id = $_GET['id'];

try {
    $stmt = $conn->prepare("DELETE FROM products WHERE id = ?");
    $stmt->execute([$id]);

    header("Location: index.php");
    exit();
} catch(PDOException $e) {
    echo "Error: " . $e->getMessage();
}
?>
```

How to Use This Project

1. Create the database and table using the SQL provided
2. Save all the PHP files in your web server's root directory (like htdocs or www)
3. Access the application through your browser (http://localhost/index.php)
4. You can:
    - View all products
    - Add new products
    - Edit existing products
    - Delete products

Lecture 29

WordPress

Introduction to WordPress

1. What is WordPress?

- Open-source CMS (Content Management System)
- Powers ~43% of all websites (2023)
- Two versions:
  - **WordPress.org** (self-hosted, full control)
  - **WordPress.com** (hosted, limited features)

2. Why Use WordPress?

- Easy to use (no coding required)
- Highly customizable (themes & plugins)
- SEO-friendly
- Large community support

3. WordPress vs. Other CMS

Feature	WordPress	Joomla	Drupal
Ease of Use	★★★★★	★★★	★★
Extensibility	★★★★★	★★★	★★★★
Learning Curve	Low	Medium	High

Installing WordPress

1.Prerequisites

- Domain Name (e.g., yourwebsite.com)
- Web Hosting (e.g., Bluehost, SiteGround)
- FTP Client (FileZilla, Cyberduck)
- Text Editor (VS Code, Sublime Text)

2. Installation Methods

Method 1: One-Click Install (Recommended for Beginners)

- Log in to your hosting (e.g., cPanel).
- Find "Softaculous Apps Installer" or "WordPress Installer".
- Click Install Now and follow the setup wizard.
- Set Admin Email, Username, Password

Method 2: Manual Install (Advanced)

- **Download WordPress** from wordpress.org.
- **Upload files** via FTP to /public_html/.
- **Create a MySQL Database** (via cPanel > MySQL Databases).
- Run installation (yourdomain.com/wp-admin/install.php)

Method 3: Local Installation (For Testing)

- Use **XAMPP/WAMP** (Windows) or **MAMP** (Mac).
- Install WordPress in localhost.

WordPress Dashboard Overview
Key Sections

1. **Dashboard Home** – Overview of site activity.
2. **Posts** – Manage blog posts.
3. **Pages** – Create static pages (Home, About, Contact).
4. **Media** – Upload images, videos, PDFs.
5. **Appearance** – Themes, Customizer, Menus.
6. **Plugins** – Extend functionality.
7. **Users** – Manage admins, authors, subscribers.
8. **Settings** – General, Permalinks, Reading.

First Steps After Installation
Change Site Title & Tagline (Settings > General).
Set Permalinks (Settings > Permalinks > Post Name).
Delete Default Plugins/Themes (e.g., "Hello Dolly" plugin).
Install Essential Plugins (e.g., Yoast SEO, Elementor)

Working with Themes

What is a WordPress Theme?

- Controls design & layout of your site.
- Free vs. Premium themes (e.g., Astra, Divi, OceanWP).

Installing a Theme

- Go to Appearance > Themes > Add New.
- Search for a theme (e.g., "Astra").
- Click Install then Activate

Customizing a Theme

- Use **WordPress Customizer** (Appearance > Customize).
- Adjust:
    - **Colors & Fonts**
    - **Header & Footer**
    - **Homepage Settings** (Static or Blog)

Creating Content
Posts vs. Pages

Feature	Posts	Pages
Dynamic (Blog)	✅	❌
Static (Home)	❌	✅
Categories/Tags	✅	❌

Adding a New Post

- Go to Posts > Add New.
- Enter Title & Content (use Gutenberg editor).
- Assign Categories & Tags.
- Set Featured Image.
- Publish or Schedule

Extending WordPress with Plugins
What Are Plugins?

- Add new features (SEO, Security, Forms).
- Over 60,000+ free plugins in the WordPress repository.

Must-Have Plugins

Plugin	Purpose
Yoast SEO	Optimize for search engines
Elementor	Drag-and-drop page builder
WooCommerce	E-commerce store
WP Super Cache	Speed up site
Wordfence	Security & firewall

Installing a Plugin

- Go to Plugins > Add New.
- Search for a plugin (e.g., "Yoast SEO").
- Click Install Now then Activate.

WordPress Security & Maintenance

Basic Security Practices

- Use **strong passwords**.
- Install **Wordfence/Sucuri**.
- Keep **WordPress, themes, plugins updated**.
- Enable **two-factor authentication (2FA)**

Backing Up Your Site

- Use UpdraftPlus (free backup plugin).
- Store backups on Google Drive/Dropbox.

Optimizing Speed

- Use **caching plugins** (WP Rocket, W3 Total Cache).
- Optimize images (**Smush** plugin).
- Use a **CDN** (Cloudflare).

Lecture 30

Elementor

Elementor is the world's leading drag-and-drop WordPress page builder, empowering over 18 million websites globally to create professional, responsive, and visually stunning websites without coding

What is Elementor?

Elementor is a **visual website builder** for WordPress that replaces the default editor with an intuitive **frontend drag-and-drop interface**. It allows users to design:

- **Pages** (Home, About, Contact)
- **Blog layouts**
- **WooCommerce stores**
- **Headers, footers, and popups**
- **Custom theme templates**

Unlike traditional WordPress editing, Elementor provides **real-time previews**, eliminating the need to switch between backend and frontend views

Key Features of Elementor

A. Drag & Drop Editor
- Build pages visually by dragging widgets (text, images, buttons, etc.).
- **100+ widgets** (free & Pro) for advanced layouts 1.

B. 300+ Designer-Made Templates
- Pre-built **full website kits**, landing pages, and popups.
- **Section-based templates** for faster design 1.

C. Responsive Design Controls
- Custom breakpoints for **desktop, tablet, and mobile**.
- Hide/show elements per device 1.

D. Theme Builder (Pro Feature)
- Design **headers, footers, single posts, and WooCommerce pages** without theme restrictions 2.

E. WooCommerce Builder (Pro Feature)
- Customize **product pages, cart, checkout, and account pages** 3.

F. Dynamic Content & Integrations
- Connect with **ACF, Toolset, and CRM tools** (Mailchimp, HubSpot) 2.

G. Performance Optimizations
- **Reduced DOM output, lazy loading, and faster font loading** for speed

Elementor Free vs. Pro

Feature	Free Version	Pro Version
Widgets	40+	100+
Theme Builder	❌	✅
Popup Builder	❌	✅
WooCommerce	❌	✅
Dynamic Content	❌	✅
Form Builder	❌	✅
Global Styling	Limited	Full Control

Pricing (Pro Plans):

- **Essential ($59/year)** – 1 site, basic Pro features.
- **Advanced Solo ($79/year)** – 1 site, full Pro tools.
- **Agency ($399/year)** – 1,000 sites

How to Install Elementor

Free Version:

1. Go to **WordPress Dashboard → Plugins → Add New**.
2. Search for **"Elementor"** → Install & Activate.

Pro Version:

1. Purchase **Elementor Pro** from elementor.com.
2. Upload the **ZIP file** via **Plugins → Add New → Upload Plugin**.
3. Activate the license under **Elementor → License**

How to Use Elementor

1. Create/Edit a Page: Click "Edit with Elementor" on any page.
2. Add Sections & Widgets: Drag elements like headings, buttons, and images.
3. Customize Styling: Adjust fonts, colors, and spacing in real-time.
4. Save & Publish: Click "Update" to apply changes

Pro Tip: Use Elementor Templates to speed up workflow (e.g., pre-made About Us pages)

Who Should Use Elementor?

- **Beginners:** No coding needed.
- **Freelancers & Agencies:** Faster client site builds.
- **E-commerce Owners:** Custom WooCommerce designs.
- **Developers:** Extend with custom CSS/JS

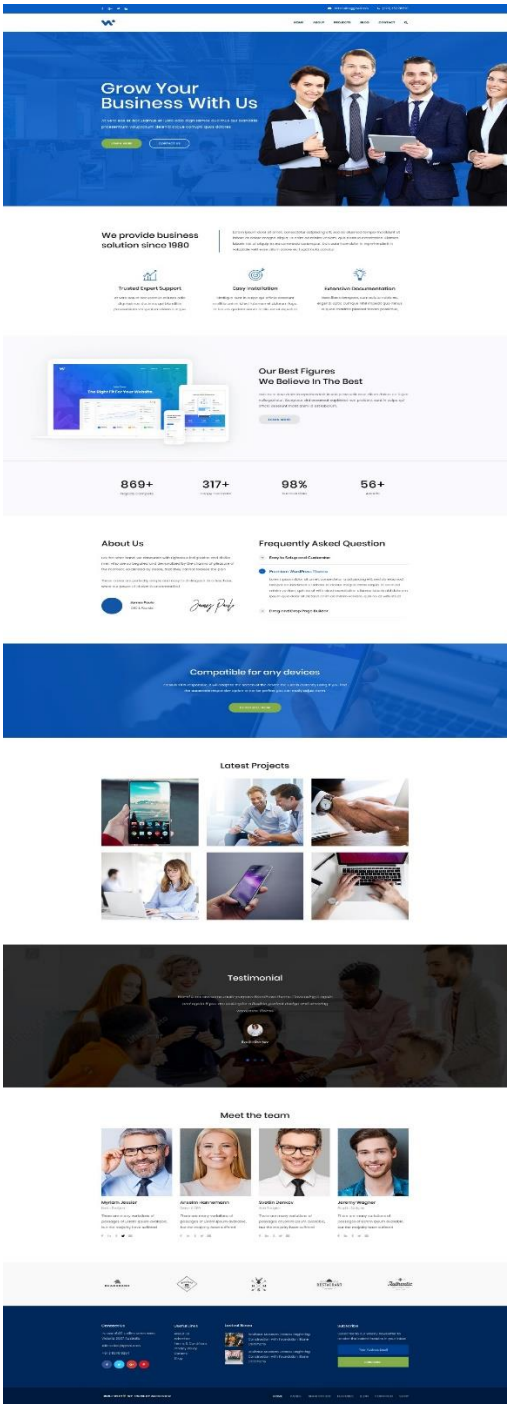
*Elementor is the **#1 WordPress builder** for a reason—its **ease of use, flexibility, and powerful features** make it ideal for **any website project**.*

Lecture 31 & 32

Elementor- Project 1, Home Page Design

We’re really thrilled to be developing this project with Elementor! Click the link below to download the project files.

PSD Link - https://shorturl.at/2uhWs

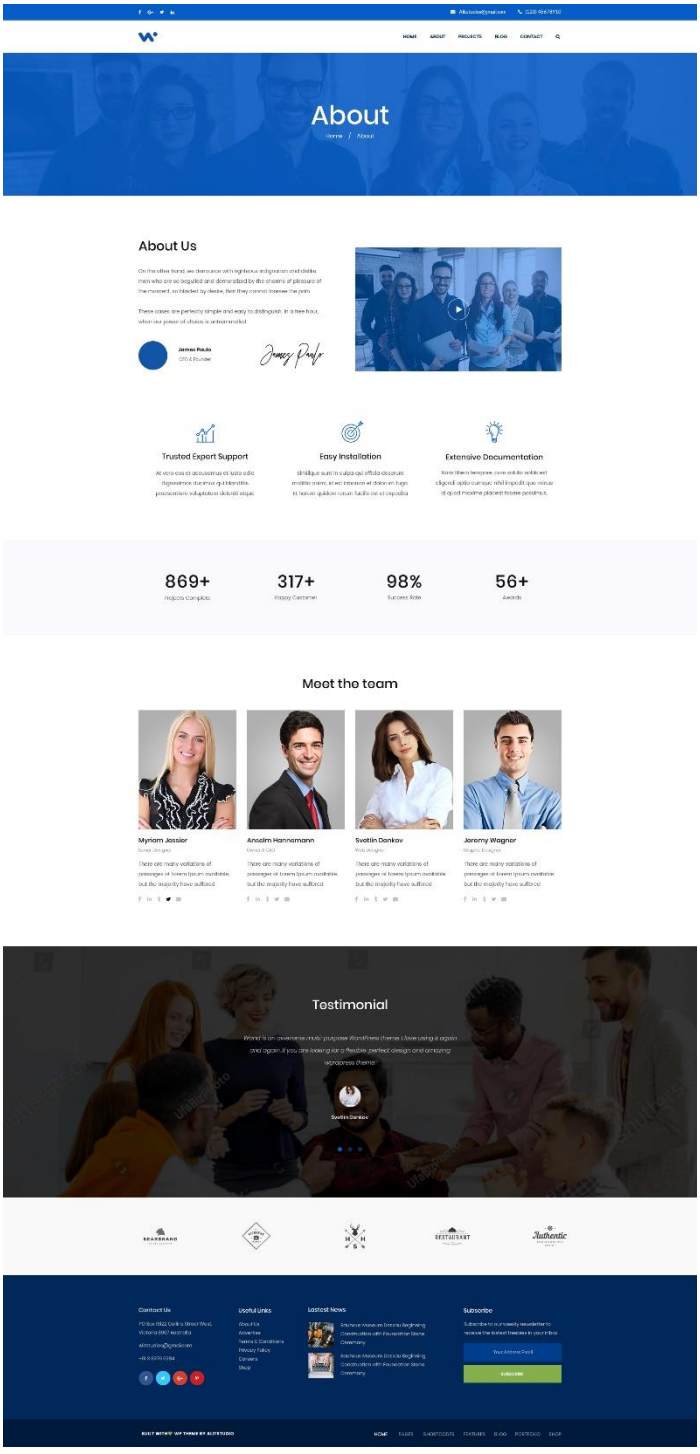


Lecture 33

Elementor- Project 1, About Page Design

Design About Page using Elementor

About Page PSD File - https://shorturl.at/xXMNb

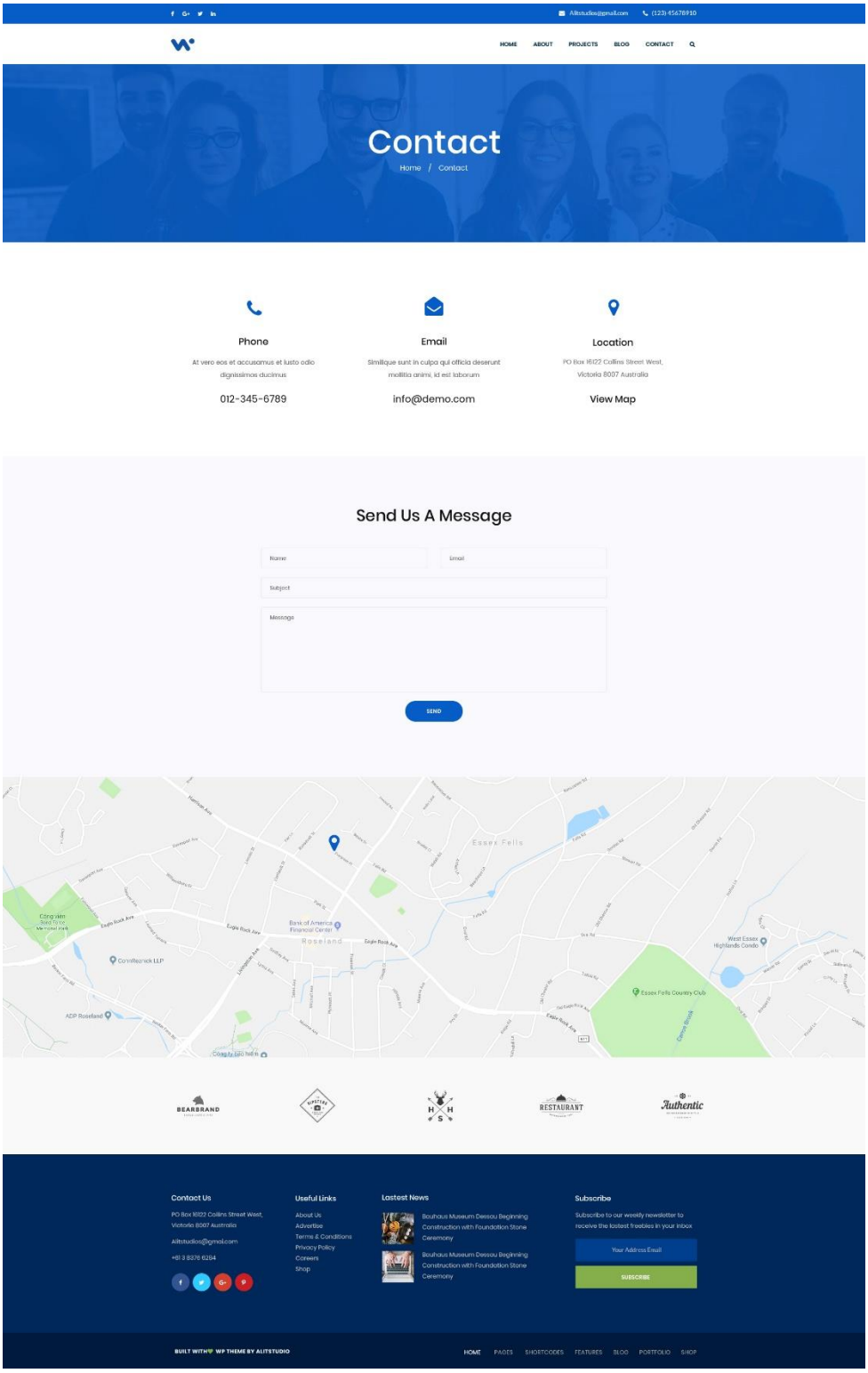


Lecture 34

Elementor- Project 1, Contact Page

Design Contact Page using Elementor

PSD File - https://shorturl.at/RfVnw



Lecture 35-40

Elementor- Project 2, Woocommerce Project

Create a WooCommerce website using the latest design (collect the design files from our software farm), and ensure all core WooCommerce functionality is included in the project.

Here's an overview of the core functionality that WooCommerce brings to a WordPress site:

1. **Product Management**
    - Simple, grouped, external/affiliate, and variable products
    - Inventory tracking (stock levels, backorders, low-stock notifications)
    - Digital (downloadable) and virtual product support
2. **Shopping Cart & Checkout**
    - AJAX "add to cart" and persistent cart across devices
    - Guest checkout or require user registration
    - Address auto-complete and validation
3. **Payment Gateways**
    - Built-in support for PayPal, Stripe, bank transfers, and checks
    - Hundreds of additional gateways via free or premium extensions
    - Tokenization for saved cards and one-click payments
4. **Shipping & Tax**
    - Flat, free, or real-time carrier rates (e.g. USPS, FedEx, DHL)
    - Per-item, per-class, or per-order shipping rules
    - Automatic tax calculation based on customer location
5. **Coupons & Discounts**
    - Fixed cart discounts, percentage discounts, free shipping coupons
    - Usage limits (per coupon, per user) and expiry dates
    - Advanced promotions via extensions (BOGO, dynamic pricing)
6. **Order Management**
    - Custom order statuses (Processing, On Hold, Completed, Refunded, etc.)
    - Email notifications to admin and customers at each status change
    - Manual order creation and editing in the admin area
7. **Reporting & Analytics**
    - Sales by date, product, category, coupon, or customer
    - Customer reports (new vs. returning customers)
    - Exportable CSV reports
8. **Extensibility & Integration**
    - Thousands of free & premium extensions for subscriptions, bookings, memberships, and more
    - REST API for headless implementations or mobile apps
    - Hooks (actions & filters) for custom development
9. **Marketing & SEO**
    - Integration with Mailchimp, HubSpot, and other email platforms

- Built-in SEO friendliness inherited from WordPress (pretty permalinks, schema markup)
- Product reviews and ratings

10. **Localization & Multi-Currency**
    - Fully translatable strings (supports WPML, Polylang, etc.)
    - Extensions for multi-currency pricing and conversion
11. **Security & Compliance**
    - PCI-compliant when using hosted gateways
    - GDPR-compliance tools (data export, anonymization)
    - Regular updates and large community scrutiny
12. **Mobile & PWA Support**
    - Responsive design out-of-the-box
    - Compatible with WooCommerce mobile apps (iOS/Android)
    - Extensions available for progressive web app functionality

Together, these features make WooCommerce a flexible, scalable e-commerce solution that can be adapted—from a simple one-product shop to a full-scale marketplace.