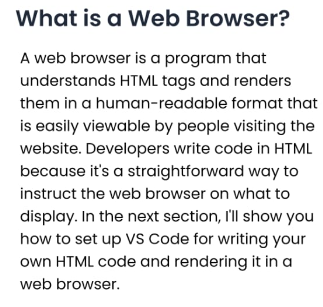
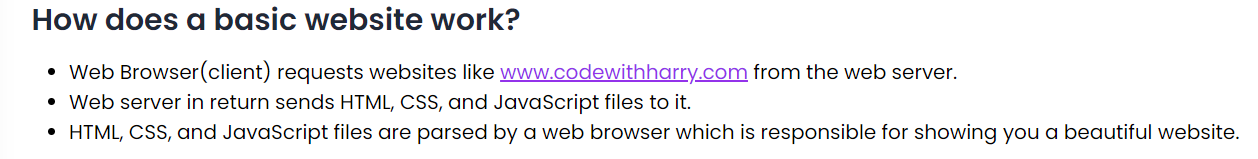
**WEB DEVELOMPENT**

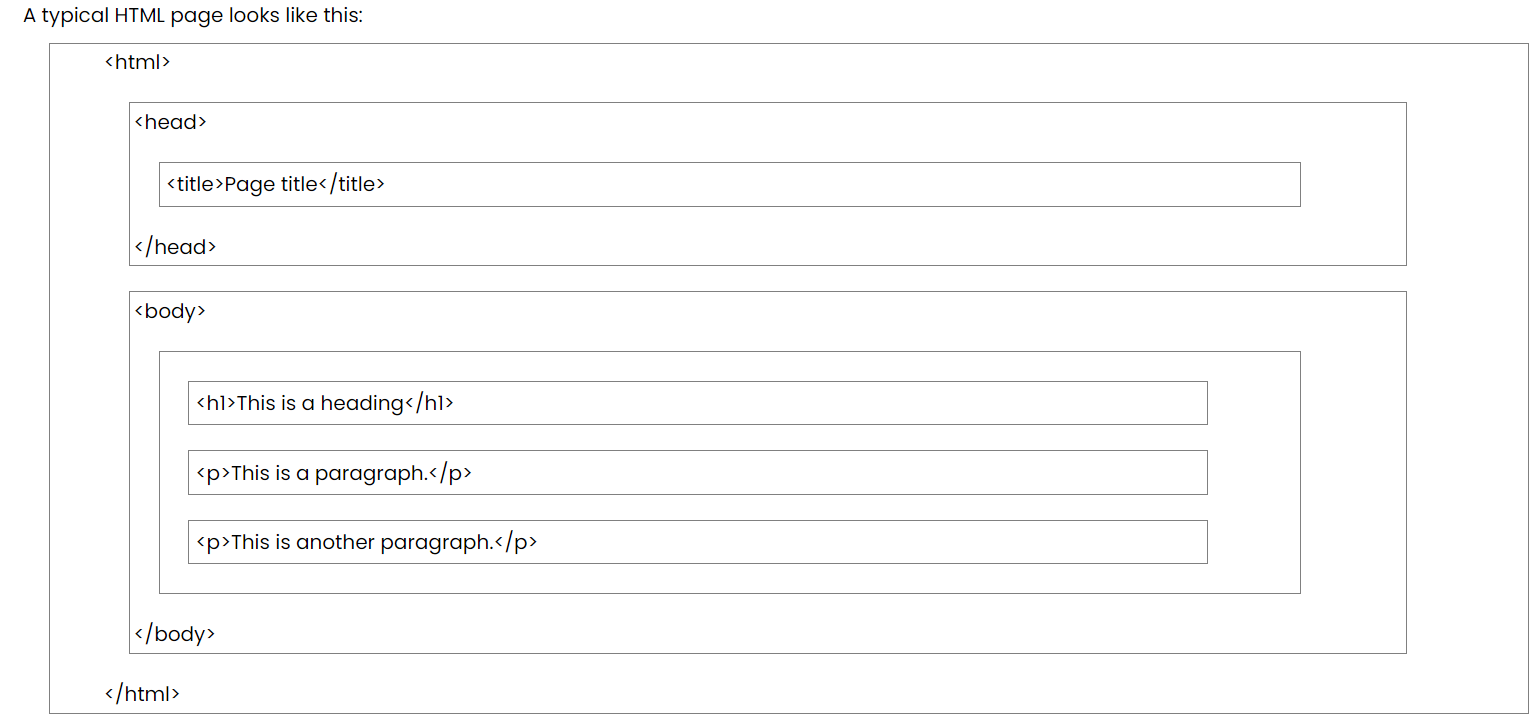
1. WEB BROWSER

When we want to access any information on the internet, we search for it using a web browser. The web browser retrieves the content from web servers, where it is stored in the form of HTML documents.





1. HTML attributes



(HTML attributes are special words used inside the opening tag of an HTML element to control the element's behavior or provide additional information about it) are used to define the characteristics of an HTML element. They are placed within the element's opening tag and consist of two parts: the **name** and the **value**.

* **Name**: Specifies the property for that element.
* **Value**: Sets the value of that property for the element.

There are three main types of HTML attributes:

1. **Core Attributes**: These are basic attributes that can be applied to most HTML elements. Examples include id, class, and style.

Core attributes are some of the most widely used attributes in HTML. There are four main types:

* Id

**ID Attribute**

The ID attribute is used to assign a unique identifier to an HTML element. Each element with an ID has its own unique identity, similar to how each individual has a unique identity. Multiple elements cannot have the same ID.

Example:

<p id="html">This is an HTML tutorial</p>

<p id="python">This is a Python tutorial</p>

In this example, the ID attribute helps to distinguish between two paragraphs by having different values: "html" and "python".

(basically if we want to give color or any attribute to “html” or “python” “only” then “id” can be very useful)

* Class

**Class Attribute**

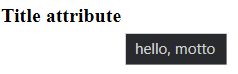
The class attribute is used to associate an HTML element with a particular class, typically for styling or JavaScript manipulation. Unlike the ID attribute, the class attribute is not unique, and multiple elements can share the same class.

* Title

The title attribute provides additional information about an element and is often displayed as a tooltip when the mouse hovers over it.

Example:

<h4 title="hello, motto">Title attribute</h4>



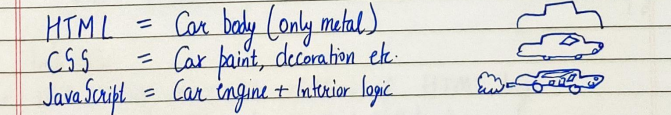
* Style

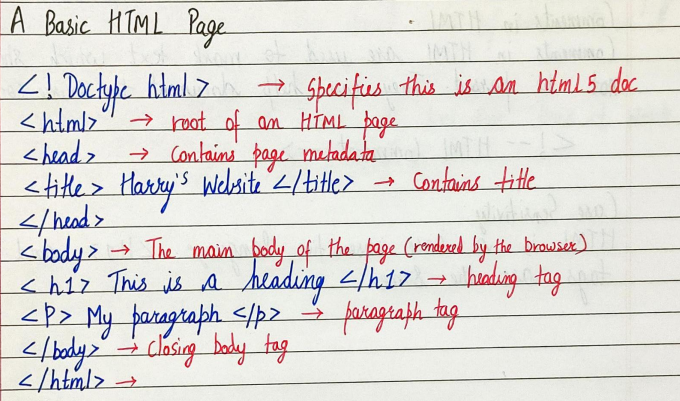
1. **Internationalization Attributes**: These attributes help adapt the document to different languages and regions. Examples include “lang” and “dir”.
2. **Generic Attributes**: These attributes provide additional information about the element but don't necessarily affect its appearance or behavior. Examples include data-\* attributes for storing custom data private to the page or application.
3. UNDERSTANDING OF LANGUAGES

Html file gives the structure to website

Css file gives style to website

Js file gives functioning to website



****

1. Metadata is “data about data,” providing information about the HTML document itself.

It is not displayed on the page, but rather exists within the HTML code.

**Common uses of metadata in HTML:**

* **Page description**: Providing a brief summary of the page’s content to help search engines understand its relevance.

    <meta name="description" content="This is Happy page">

    <!-- page description -->

   <!-- even if we don't provide "description" even then

      the browser will proivde the "description" on is own

      according to our website content -->

* **Keywords**: Specifying important terms or phrases related to the page’s content to aid search engine indexing.
* **Authorship**: Identifying the creator or maintainers of the page.
* **Expiration dates**: Indicating when the page’s content is no longer valid or should be updated.
* **Copyright information**: Specifying licensing or ownership details.

**HTML FILE**

<!DOCTYPE html>

<--

The <!DOCTYPE html> declaration informs the web browser about the HTML version being used.

-->

<html lang="en">

<--

The <html lang="en"> tag defines the document to be in English.

-->

<--  
The <html> tag is the root element that encapsulates all the content on the page.

Like including other HTML tags like <head>, <body>, <title>, and more.

-->

<head>

<--

The <head> tag contains metadata and links to external resources like CSS and JavaScript files.

-->

    <meta charset="UTF-8">

<--

: By specifying UTF-8, you ensure that the text in your HTML document will be correctly displayed, even if it contains special characters or symbols (like accented letters, emojis, etc.)

-->

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

**<!—**

**name="viewport"**: The name attribute specifies that this metadata is related to the **viewport**, which is the visible area of a web page on a device. This area varies between devices, such as between a desktop and a smartphone.

--!>

<!—

**content="width=device-width"**: This sets the width of the viewport to match the device's width.

--!>

<!—

**initial-scale=1.0**: This sets the initial zoom level when the page is first loaded by the browser. An initial scale of 1.0 means that the page will be displayed at the actual size (without zooming in or out).

--!>

    <title>HAPPY'S First Website</title>

    <!-- this is the "words" we see on the tab  -->

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

<!—

The “rel” attribute helps the browser understand the role or purpose of the linked resource. It is required when using <link> to ensure that the relationship between the document and the external resource is properly interpreted.

--!>

<!—

rel="stylesheet" specifies that the linked resource is a stylesheet that will be used to style the HTML page.

Other examples

 rel="icon": Links to a favicon.

 rel="alternate": Links to alternate versions of the document (e.g., in a different language or format).

 rel="author": Links to information about the author of the document.

--!>

    <!-- this statement is important to associate html file with css file -->

</head>

<body>

<--

The <body> tag contains the visible content of the web page. This is where text, images, and other elements go.

-->

    HEY!!

    DIKSHA KHUREJA MAA

    THIS IS MY FIRST WEBSITE.

    <!—

this is the "words" we see when we open website -->

    <script src="script.js"></script>

    <!—

this statement is important to associate html file with js file

-->

<!—

**<script> Tag**:

The <script> tag is used to embed or reference executable JavaScript code within an HTML document.

 The src attribute specifies the location (URL or path) of an external JavaScript file.

 In this example, "script.js" is the file that contains the JavaScript code you want to include.

--!>

</body>

</html>

In the above code,

The placement of every line is syntax

**CSS FILE**

body

{

    background-color: brown;

     /\* website background colour \*/

     color: antiquewhite;

     /\* body font colour \*/

}

**JS FILE**

alert("Welcome To My First Website")

/\*

"alert" refers to the "popup"

when we open a website

\*/

1. BASIC WEBSITE

<!DOCTYPE html>

<html lang="en">

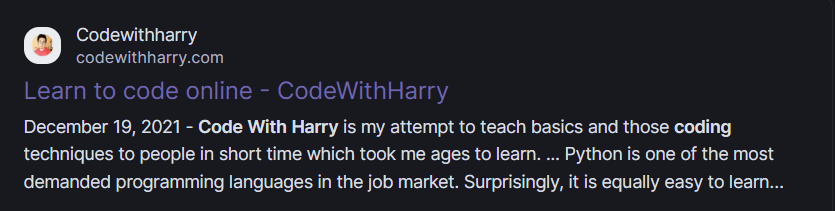
<head>

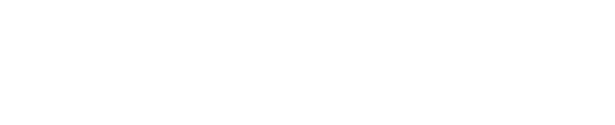
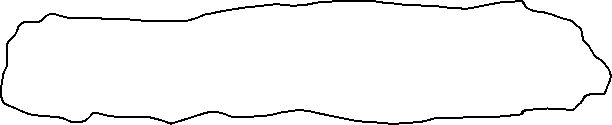
    <meta charset="UTF-8">

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

<meta name="description" content="xyz">

<!-- "description" written on the link  -->





    <title>BOOKMARKS</title>

<!-- "title" written on the browser's tab  -->

</head>

<body>

    <h1>MY BOOKMARKS</h1>

    <!-- "h1" refers to the "biggest heading" -->

    <h2>PRIMARY BOOKMARKS</h2>

    <!-- "h2" refers to the "second biggest heading" -->

    <p>

        <a target="\_blank" href="https://www.google.com">Open Google</a>

<!—

In this code , when we click “open google”

Automatically the mentioned link will be opened (in this case https://www.google.com)

<!—

The” target="\_blank"”is used to specify that the link should open in a **new browser tab** or **window** when clicked, rather than in the same tab or window where the link was clicked.--!>

<!—

The <a> tag in HTML stands for "anchor" and is used to create hyperlinks.

--!>

<!—

A **hyperlink** (or simply "link") is a reference in a web document that allows users to click and navigate from one location to another, either within the same document or to an entirely different document or web page.

--!>

        <!-- "href" is used to provide reference(using link) to a website  -->

    </p>

    <p>

        <a target="\_blank" href="https://www.facebook.com">Open facebook</a>

    </p>

    <p>

        <a target="\_blank" href="https://www.quora.com">Open quora>

    </p>

    <p>

        <a target="\_blank" href="https://www.codewithharry.com">Open codewithharry</a>

    </p>

<!-- inline "css" through "style attribute" -->

<p>

        Lorem, ipsum dolor sit amet consectetur adipisicing elit. Id, tempora!

        <!--

        "dummy code" using "lorem(number of words)"

        to make a "template"

        -->

    </p>

    <h2>SECONDARY BOOKMARKS</h2>

</body>

NOTE: href,rel,styles all are attributes to give

To give properties to “tags”

1. BASIC WEBSITE INFO

 1 column

 2 columns



</html>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>IMAGES</title>

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

<!—to add “css file” --!>

</head>

<body>

    <img width="500" height="500" src="image.png" alt="LAL SAI">

    <br>

    <!-- line break  -->

    <!--

    "img" is used to display an image

    "width" tells us about the width of image

    "height" tells us about the height of an image

    [if we attribute either of them (width or height) the other will set automatically]

    "src" is the image to be displayed(SAME FOR VIDEO)

    "alt" is used for the time when for some reason

    image couldn't be displayed then the text mentioned in "alt"

    will be shown

     -->

     <table>

            <caption>Student Details</caption>

            <!--

            "caption" is basically "title" to "table"

            similarly "header" and "footer" bhi hote hai

            -->

        <tr>

            <!-- "tr"->"table row" -->

          <th>Name</th>

          <th>Designation</th>

          <th>Fav Lang</th>

          <!-- "th"->"table heading" -->

        </tr>

        <tr>

          <td colspan="2">Happy</td>

          <!--

          "colspan" simply means how many "columns" a "data" will take

          similarly "rowspan"

          -->

          <td rowspan="6">Programmer</td>

          <td>Sindhi</td>

          <!-- "td"->"table data" -->

        </tr>

      </table>

      <ul type="square">

        <!-- "unordered list" -->

        <!-- "square" for bullets to be in square shape -->

           <li>Happy</li>

           <li>Pammu</li>

      </ul>

      <ol type="A">

        <!-- "ordered list" -->

        <!-- "A" for "CAPITAL ALPHABETS" for bullets  -->

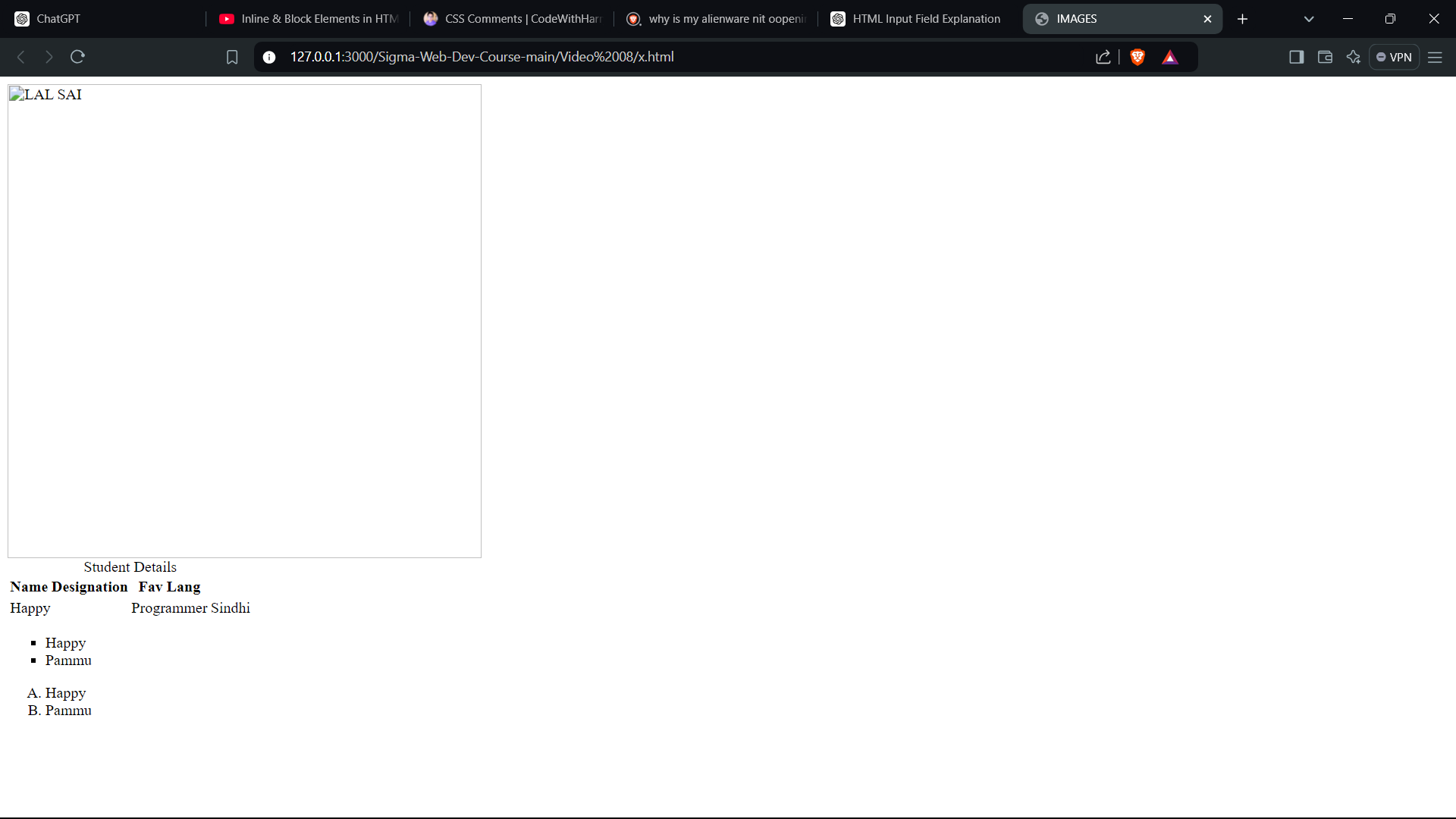
           <li>Happy</li>

           <li>Pammu</li>

      </ol>

</body>

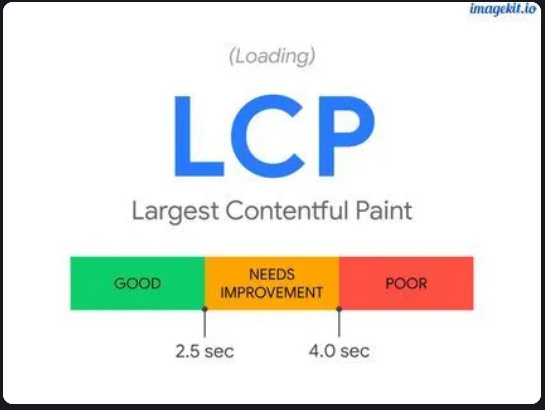
</html>



1. WEB VITALS

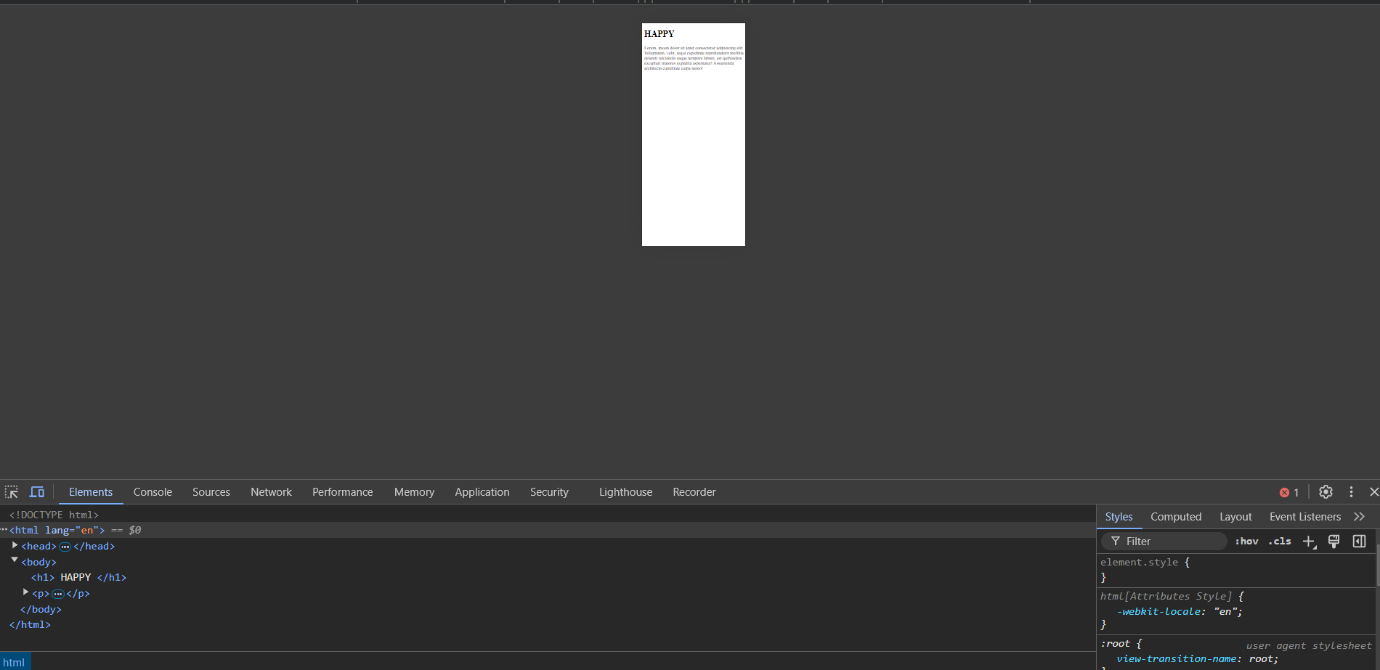
Cumulative Layout Shift (CLS) is a  metric that measures the visual stability of a webpage (basically when we click on some option in website then does the content stays intact)

LCP is **a Core Web Vital metric that measures the time it takes for the largest content element (typically an image or text block) to become visible in the viewport**.



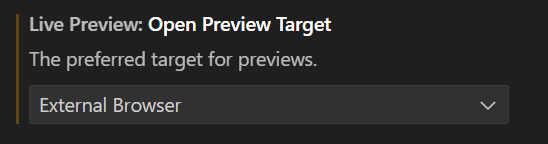
In Web Development, FID stands for **First Input Delay**, a metric used to measure the responsiveness of a webpage. It represents the time between a user’s initial interaction with a website (e.g., clicking a button or link) and the browser’s response to that interaction.

NOTE: Web dev good website for web development





Widget for viewing the website in any view



Opening the website directly in browser (vice versa)

1. FORMS

HTML forms are essential for collecting user input on web pages.

They enable users to submit data, which can be processed, stored, or returned by a server.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <meta name="description" content="This is Happy page">

    <!-- page description -->

    <!--

      even if we don't provide "description" even then

      the browser will provide the "description" on is own

      according to our website content

     -->

    <title>HAPPY</title>

</head>

<body>

    <h1>Form to apply for Sigma Web Development</h1>

    <form action="post">

        <!--

        the <form> element is used to create a “form” that allows users to input data and submit it to a server.

        The action attribute within the <form> tag specifies the URL where the form data should be sent when the form is submitted.

        The method attribute specifies how the data should be sent. The value "post" for the method attribute means that the form data will be sent to the server as part of the HTTP request body. This is different from the "get" method, where data is appended to the URL in the form of query parameters.

            -->

        <div>

            <label for="username">Enter your Username</label>

            <!--

This creates a label(The <label> tag is used to define a label for an input element, such as a “text field”(label in this case) that is associated with the form control with the id="username". The text "Enter your Username:" is displayed as the label.

             In the context of HTML, a "label" is a user interface element that provides a description or identifier for a form control, such as a text box, checkbox, radio button, or other input elements. The purpose of a label is to inform users what data they are expected to input or what a particular control represents.

             -->

<!—

The “for” attribute links the “label” to an “input element” by matching the “for” attribute's value with the “id” of the “input element”

In this case, for="username" means that this label is associated with the input element that has id="username".

--!>

<!—

“enter your username”

This is the text that appears as the label next to or above the input field. It gives the user a clear indication of what they should enter in the associated input field.

--!>

            <input type="text" id="username" name="username" placeholder="Enter your username">

            <!--

            The "type" attribute specifies the type of input control (e.g., text, password, checkbox).

            -->

<!—

The “id” attribute provides a “unique identifier” for this input field.

(matlab ki “input field” mein dal kya raha hai /kis baarein mein raha hai

--!>

<!—

It defines the “name of the data” sent to the server, so the value entered by the user will be associated with this name in the form submission.

(matlab ki jo bhi data enter hua hai , uss “data” ko naam dediya hai

Ab kabhi vo “data” ko mention karna hoga ya usko uske naam se bulana hoga toh jo “name=username” kara hai vo uska naam hai (in this case “username”

--!>

<!—

 The placeholder attribute provides a hint or instruction to the user about what to enter in the input field.

 This text is displayed inside the input field when it is empty and disappears when the user starts typing.

--!>

        </div>

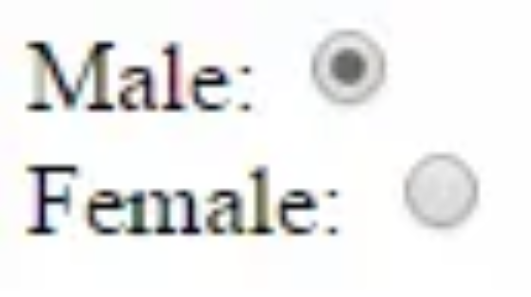
        <!—

The <div> element in HTML is a block-level container used to group and organize content

-->

        <div>

         <input type="radio" id="male" name="gender" value="male">



RADIO BUTTONS

            <!—

The “id” attribute provides a unique identifier for this specific radio button. It allows you to associate a <label> with this radio button for better accessibility and user interaction.

-->

<!—

The “name” attribute groups this radio button with other radio buttons that share the same name.

-->

            The text input type is the most basic form of input and is widely used for collecting simple text data.

            <!—

The “value” attribute specifies the value that will be sent to the server if this radio button is selected when the form is submitted. In this case, if the "male" radio button is selected, the value "male" will be submitted.

-->

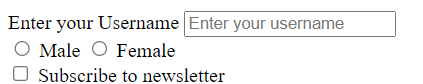
The "placeholder" attribute provides a hint to the user about what to enter.

            The “password input type” is similar to the text type but hides the characters entered by the user for security reasons.

            "Radio" buttons are used when you want the user to select only one option from a set of choices.

            -->

            <label for="male">Male</label>





            <input type="radio" id="female" name="gender" value="female">

            <label for="female">Female</label>

        </div>

        <div>

            <input type="checkbox" id="subscribe" name="subscribe" value="yes">

            <!—

Checkboxes allow the user to select multiple options from a set

-->

            <label for="subscribe">Subscribe to newsletter</label>

        </div>

        <div>

            <label for="comment">enter your comment</label>

            <!--

             The text area element is used when you need multiline text input from the user. This is particularly useful for comments, reviews, or any other type of input where the length is unpredictable.

             -->

            <br>

            <textarea name="comment" rows="4" cols="50">

          </textarea>

        </div>

        <div>

            <select name="fruits">

                <!--

                 The select element creates a dropdown menu for the user. It is useful when you have a predefined list of options for the user to choose from.

                  -->

                <option value="apple">Apple</option>

                <option value="banana">Banana</option>

                <option value="cherry">Cherry</option>

            </select>

            <!-- autofocus use karna left hai 12:00 -->

        </div>

    </form>

</body>

</html>

<!--

**Quick Quiz:**

    Without using br tag, write a vertically aligned form asking for name, city and pincode of a user.

    Everyone must comment

 -->

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Document</title>

</head>

<body>

    <form action="post">

        <!--

         as such "form" hatane ya daalne se website ki "UI" par koi difference nahi aayega but "form" keyword use karne se it enables users to submit data, which can be processed, stored, or returned by a server.

        -->

        <div>

            <label for="name">enter your name</label>

            <!-- id->"name" -->

            <!-- the information written on the website will be "enter your name" -->

            <input type="text" id="name" name="name" placeholder="Enter your name" >

            <!-- in the box the text which will be acting as a placeholder will be "Enter your name" -->

        </div>

        <div>

            <label for="city">enter your city name</label>

            <input type="text" id="city" name="name" placeholder="Enter your city name" >

        </div>

    </form>

</body>

</html>

1. ID AND CLASSES



<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>ID AND CLASSES</title>

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

</head>

<body>

    <div id="firstdiv" class="red bg-yellow">First</div>

    <!--

    The “class” attribute lets you give the same name to “multiple HTML elements.” That way, you can easily change their look or behavior all at once. “Classes” are “not unique” and can be assigned to “multiple elements”. They are generally used for applying the same styles or behaviors to a group of elements.

    -->

    <div id="seconddiv" >Second</div>

    <!--

     An “ID” is an attribute, a unique identifier assigned to only one HTML element within a page. It is often used for unique styling and JavaScript manipulations.

    -->

    <span class="red"></span>

</body>

</html>

<!—

kisi bhi link ke aage agar “#” likh kar ID name likh denge toh direct vo khulega

 -->

.red

{

color: red;

}

/\*

In CSS, elements with classes are selected using a dot (.) before the class name.

\*/

.bg-yellow

{

    background-color: yellow;

}

#firstdiv

/\*

 elements with IDs are selected using a hash (#) symbol before the ID, and

\*/

{

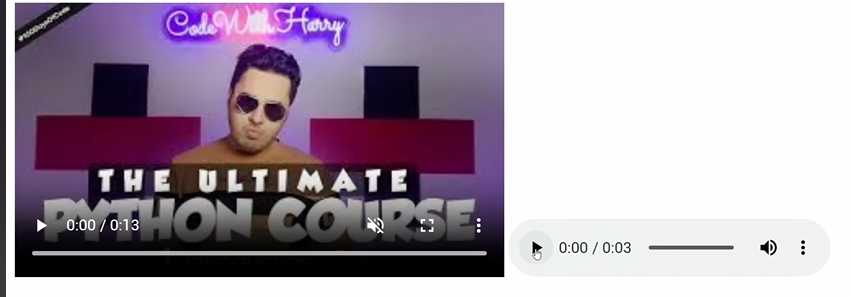
color: aliceblue;

}

1. AUDIO AND VIDEO





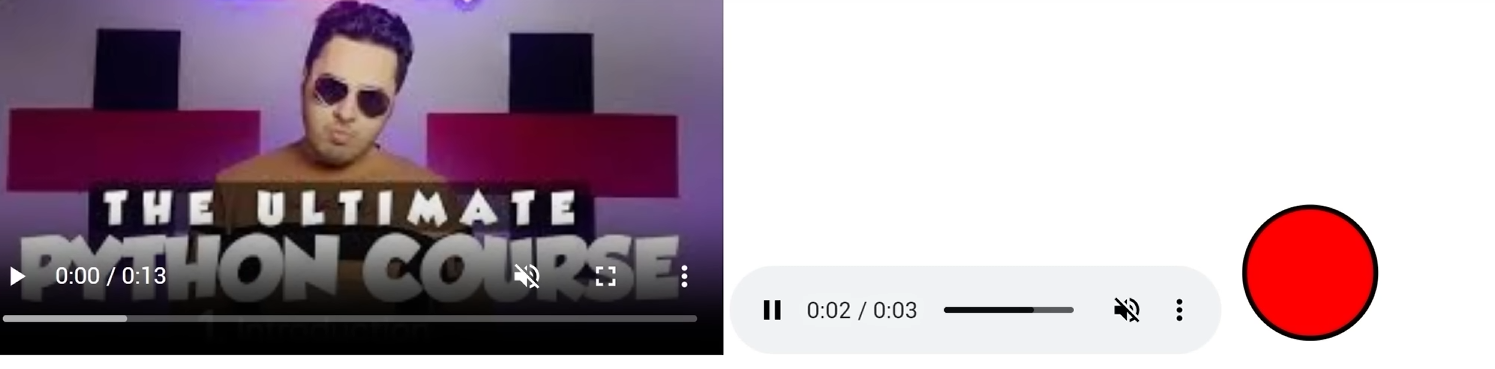


Using “audio”

* 1. **preload**: Specifies if and how the audio should be loaded when the page loads ('auto', 'metadata', 'none').

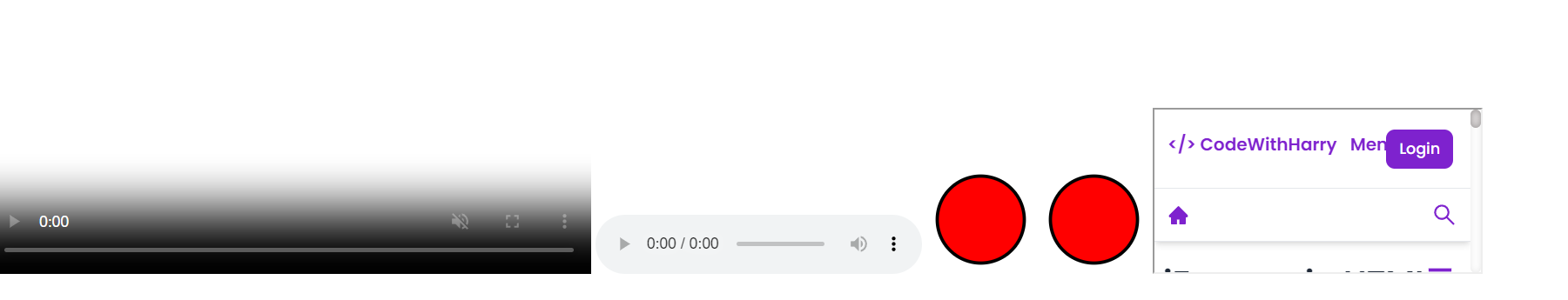
The "preload" attribute can have the following values:

* none: This is the default value. It indicates that the browser should not preload the audio file at all. The audio file will only start downloading when the user initiates playback.
* metadata: This value tells the browser to preload only the metadata of the audio file, such as its duration and basic information about the audio. This can be useful if you want to display the audio duration to the user without fully loading the audio data.
* auto: This value instructs the browser to preload the entire audio file as much as possible without delaying the loading of other important page content. The browser will try to load the audio file in the background so that it's ready to play when the user decides to start it.





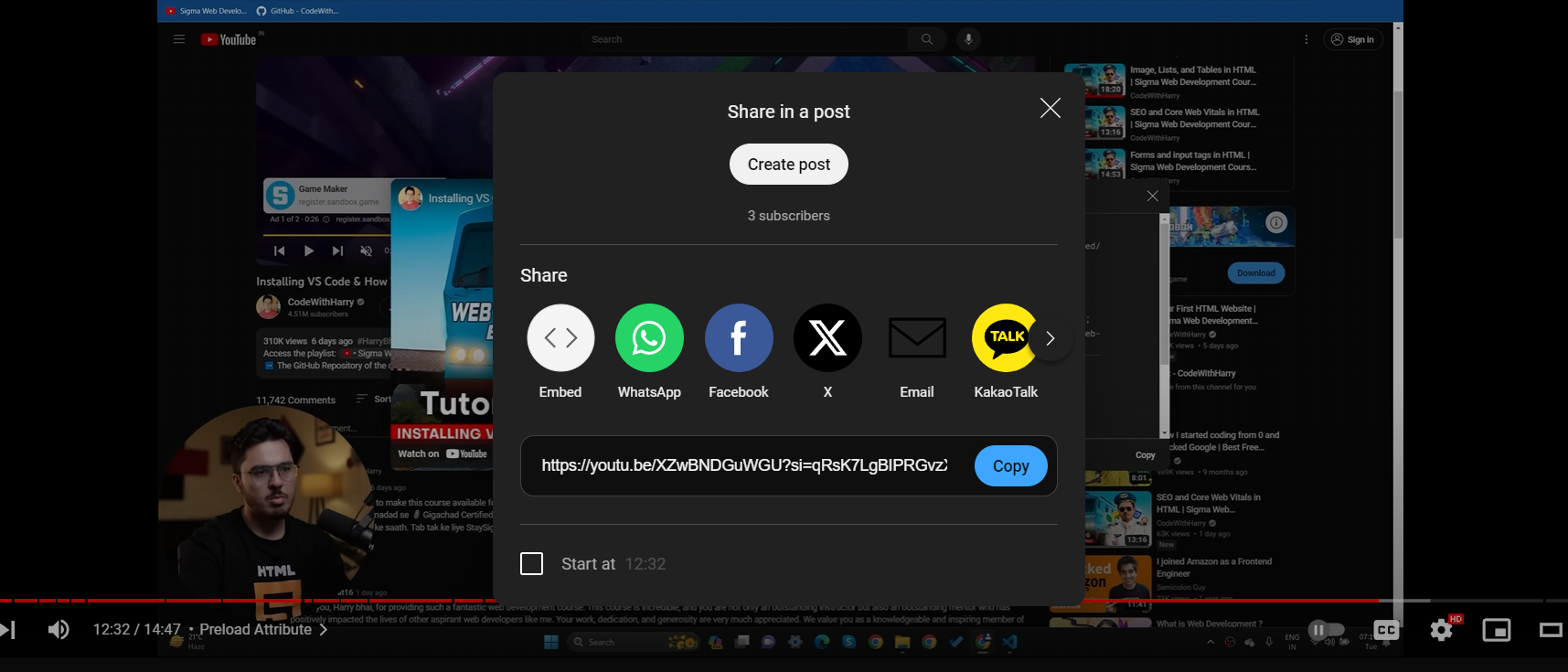
* 1. S.V.G can be created in another file as well and can be imported using “img”alt





IFRAME

* 1. iFrames offer a variety of use-cases:
* **Content Isolation**: iFrames allow you to isolate third-party content, which can improve security.
* **Modularity**: Easily embed external plugins, widgets, or content.
* **Resource Separation**: Content within an iFrame can load separately from the rest of the page.





EMBED CODE FOR YOUTUBE(IFRAME CODE)

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Document</title>

</head>

<body>

    <video src="video.mp4" width="555" height="255" controls autoplay loop muted poster="download.jpg"></video>

    <!--

    The <video> tag is used to embed video files in an HTML document. It supports multiple attributes to control the video playback

     -->

        <!--

        src: Specifies the path to the audio/video file.

        controls: Adds audio controls, like play, pause, and volume.

        autoplay: Automatically starts playing the audio when the page loads.

        loop: Repeats the audio once it ends.

        muted: Mutes the audio by default.

        poster:"download.jpg" is like a thumbnail to the video

        -->

        <audio src="audio.mp3" controls></audio>

<svg  xmlns="http://www.w3.org/2000/svg" height="100" width="100">

    <circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="red" />

</svg>

        <svg height="100" width="100">

            <circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="red" />

<!--

**<circle>**: This is an SVG element that defines a circle. It requires a center point and a radius to draw the circle.

 **cy="50"**: This specifies the y-coordinate of the center of the circle. Here, the center is 50 units along the y-axis.

 **r="40"**: This specifies the radius of the circle. The circle has a radius of 40 units.

 **stroke="black"**: This sets the color of the circle's outline. In this case, the stroke (outline) is black.

 **stroke-width="3"**: This defines the width of the circle's outline. The outline is 3 units thick.

 **fill="red"**: This sets the fill color of the circle. The inside of the circle is filled with red.

-->

        </svg>

        <!--

         Scalable Vector Graphics (SVG) has become an indispensable part of modern web development. SVG enables developers to create high-quality, scalable graphics that look crisp at any size or resolution

        -->

        <!--

         SVG stands for Scalable Vector Graphics. Unlike raster images like PNGs or JPGs, SVGs are not pixel-based. They're composed of vectors—mathematical formulas that describe shapes, paths, and fills. This means SVGs can be resized without losing quality.

        -->

         <img src="img.svg" alt="My SVG image">

         <!--

         used to import "svg file"

         -->

         <iframe src="https://www.codewithharry.com/tutorial/html-iframes/" width="width" height="height"></iframe>

         <!--

         iFrames, or Inline Frames, are an integral part of modern web development. They allow you to embed another HTML page(or youtube video) within your current page.

         -->

<!—

Embed karne ke liye link (upar ss mein bataya hai kaise lena hai)

-->

         <!--

         An iFrame is an HTML element that enables an inline frame for the embedding of external content. Essentially, you can load another web page within a designated area of your current webpage.

         -->

</body>

</html>

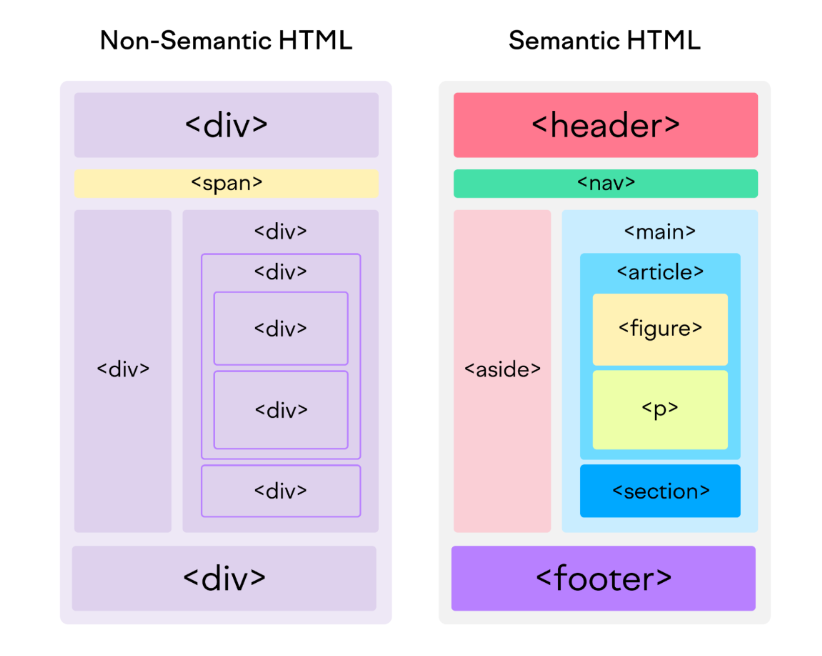
**NOTE: Ensure that the video.mp4 file is in the same directory as your HTML file. If it is in a different folder, you need to specify the correct path.**

**src="path/to/video.mp4">video1</video>**

1. SEMANTIC TAGS

Semantic tags add meaning to your HTML. They tell both the browser and the developer what kind of content is being presented.

They enhance SEO, improve accessibility, and make your code easier to read and maintain.



<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Semantic Tags</title>

</head>

<body>

    <header>

        <!-- <header>: Used to represent the top section of a web page, often containing headings, logos, and navigation. -->

        <nav>

        <!-- <nav>: Signifies a navigation menu on a web page. -->

            <ul>

                <li>Home</li>

<!-- <li> means “list item”

                <li>About</li>

                <li>Contact</li>

            </ul>

        </nav>

    </header>

    <main>

        <h1>What are Semantic Tags</h1>

    </main>

    <footer>

    <!-- <footer>: Represents the footer of a web page, usually containing copyright information and contact details. -->

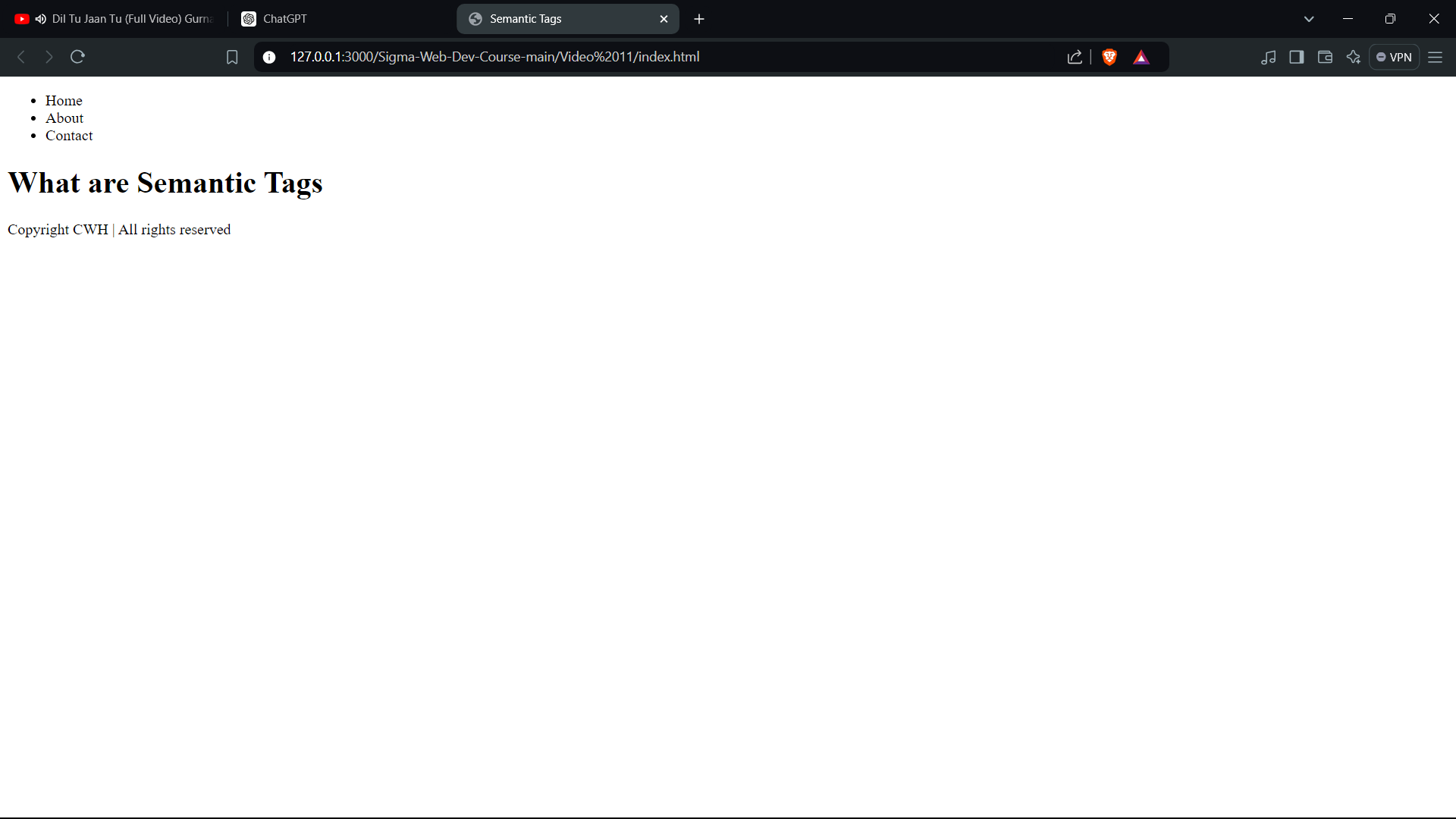
        Copyright CWH | All rights reserved

    </footer>

    <!-- <img src="semantic-tags.png" alt=""> -->

</body>

</html>



1. HTML ENTITIES

HTML entities are a crucial part of HTML markup language. They enable you to display characters that are reserved in HTML or that aren't readily available on the keyboard.

HTML entities are used to represent special characters in a format that the browser can understand. They start with an ampersand (&) and end with a semicolon (;).

**Common HTML Entities**

&lt; for <

&gt; for >

&amp; for &

&nbsp; for a non-breaking space

It is often used when you want to keep things

together on the same line.

&copy; for ©

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>Document</title>

</head>

<body>

    3>4

    this is the code for paragraph &lt;p&gt;&lt;/p&gt;

    <pre>

      <!-- line breaks and spaces are preserved exactly-->

        <p>This is a para</p>

        this is after few new lines           and some more

    </pre>

   Copyright &copy;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;codewithharry

   <blockquote cite="source-url">

    <!--

     The <blockquote> tag in HTML is used for displaying a block of text that is quoted from another source. It is typically indented from the rest of the text to indicate that it is a quotation.

     You can also use the cite attribute within the <blockquote> tag to reference the source of the quote.

    -->

    Quotation text here.

  </blockquote>

<h1>This is html boilerplate code</h1>

  <pre><code>

<!--

    The <code> tag in HTML is used to define a piece of computer code

    making it easier to distinguish from regular text. This tag is often used to display short snippets of code inline within paragraphs.

    For larger blocks of code, the <pre> tag is often combined with the <code> tag to maintain formatting and spacing:

  -->

&lt;!DOCTYPE html&gt;

&lt;html lang="en"&gt;

&lt;head&gt;

    &lt;meta charset="UTF-8"&gt;

    &lt;meta name="viewport" content="width=device-width, initial-scale=1.0"&gt;

    &lt;title&gt;Document&lt;/title&gt;

&lt;/head&gt;

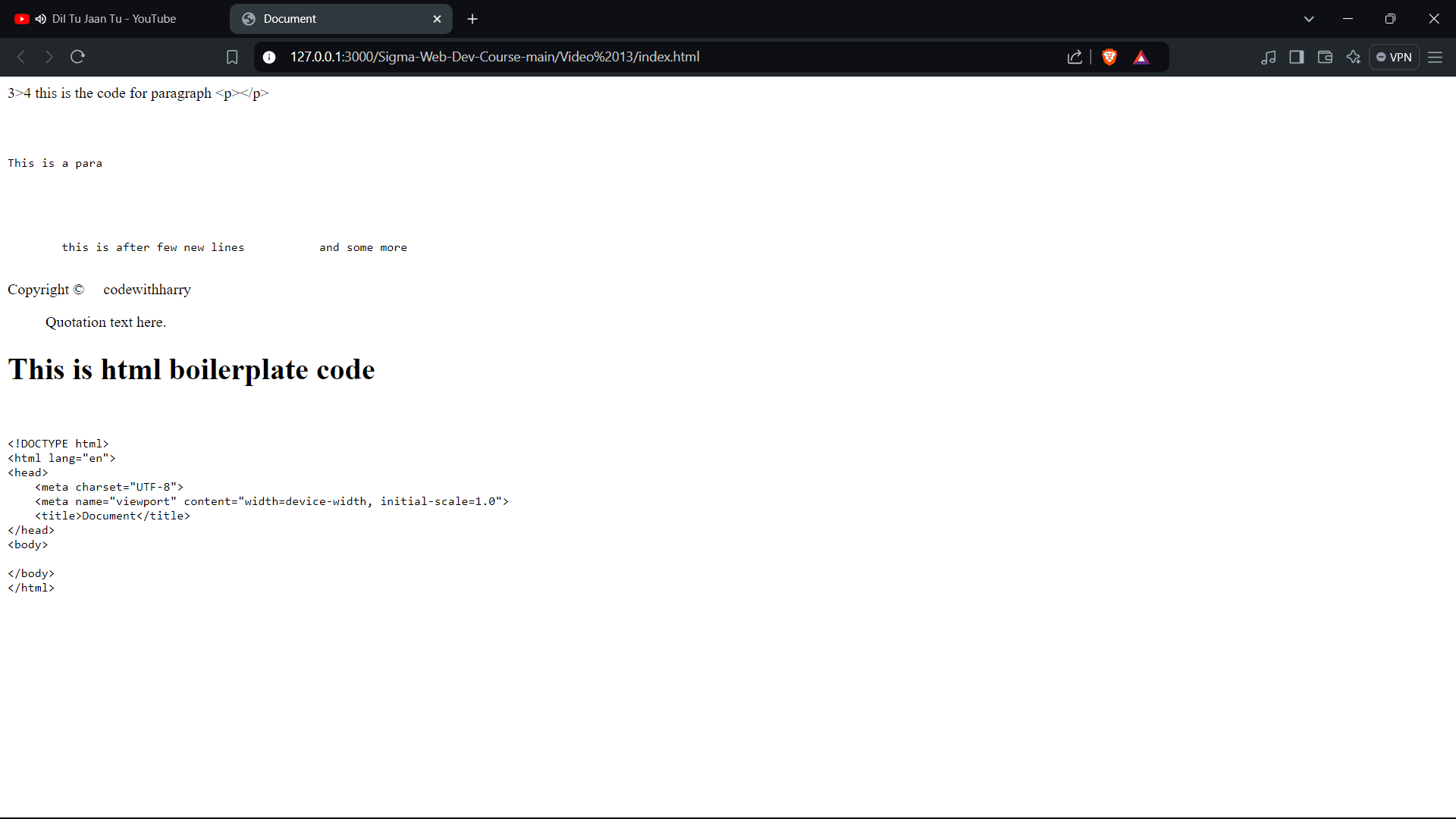
&lt;body&gt;

&lt;/body&gt;

&lt;/html&gt;

  </code></pre>

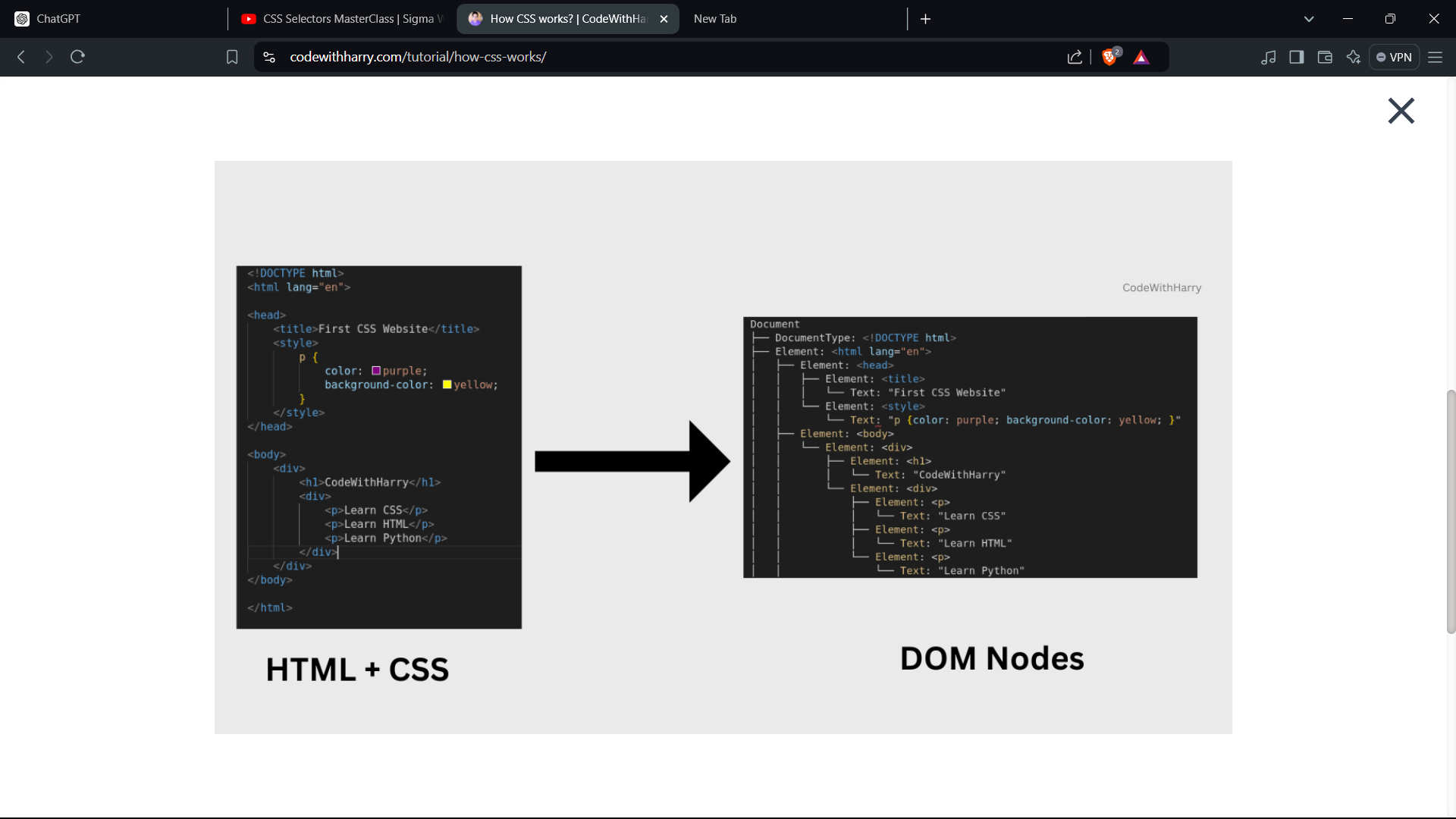
</body>



1. **How CSS Works?**

The following steps will help us understand more about CSS working :

* The user types the URL and clicks enter.
* The browser makes a fetch request to the server.
* HTML is fetched from the server.
* HTML is converted into a DOM. In the DOM, each tag is considered a **node**.
* The browser fetches all the related files and assets that are linked to that HTML, such as external CSS, fonts, images, etc.
* The browser then parses the CSS and groups it based on the selectors, which can be tags.
* Each CSS is attached to its respective node. In this phase, CSS gets attached to its respective node. This is called a **render tree**.
* The render tree is the well-structured, well-arranged **DOM** node that will appear on the screen.
* The well-structured, custom-designed website is presented on the screen. This is called **painting**.
* A DOM is like a tree structure representation of all the tags and elements on the page. Each part of a web page, like headings, paragraphs, images, buttons, etc., will be part of the tree.
* You can think of it as a blueprint for a web page that web browsers use to understand and display web content.
* Consider the below example,

The tags are converted into nodes. Each node establishes a parent-child relationship between each other. To be precise, **Document Object Model(DOM)** is a sort of API that represents and interacts with HTML documents.

In the example:

h2

{

color: blue;

}

* **h2**: h2 is the selector.
* **color**: It's the property.
* **blue:** The property value.

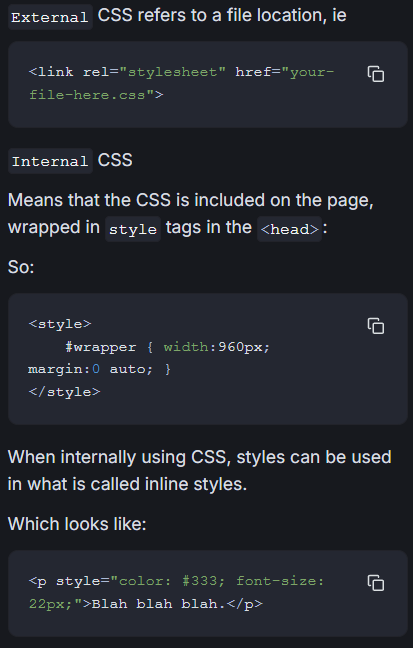
1. CSS

There are three different ways to add CSS to an HTML page, which are:

[Inline CSS](https://www.codewithharry.com/tutorial/ways-to-add-css/#inline)

[Internal CSS](https://www.codewithharry.com/tutorial/ways-to-add-css/#internal)

[External CSS](https://www.codewithharry.com/tutorial/ways-to-add-css/#external)

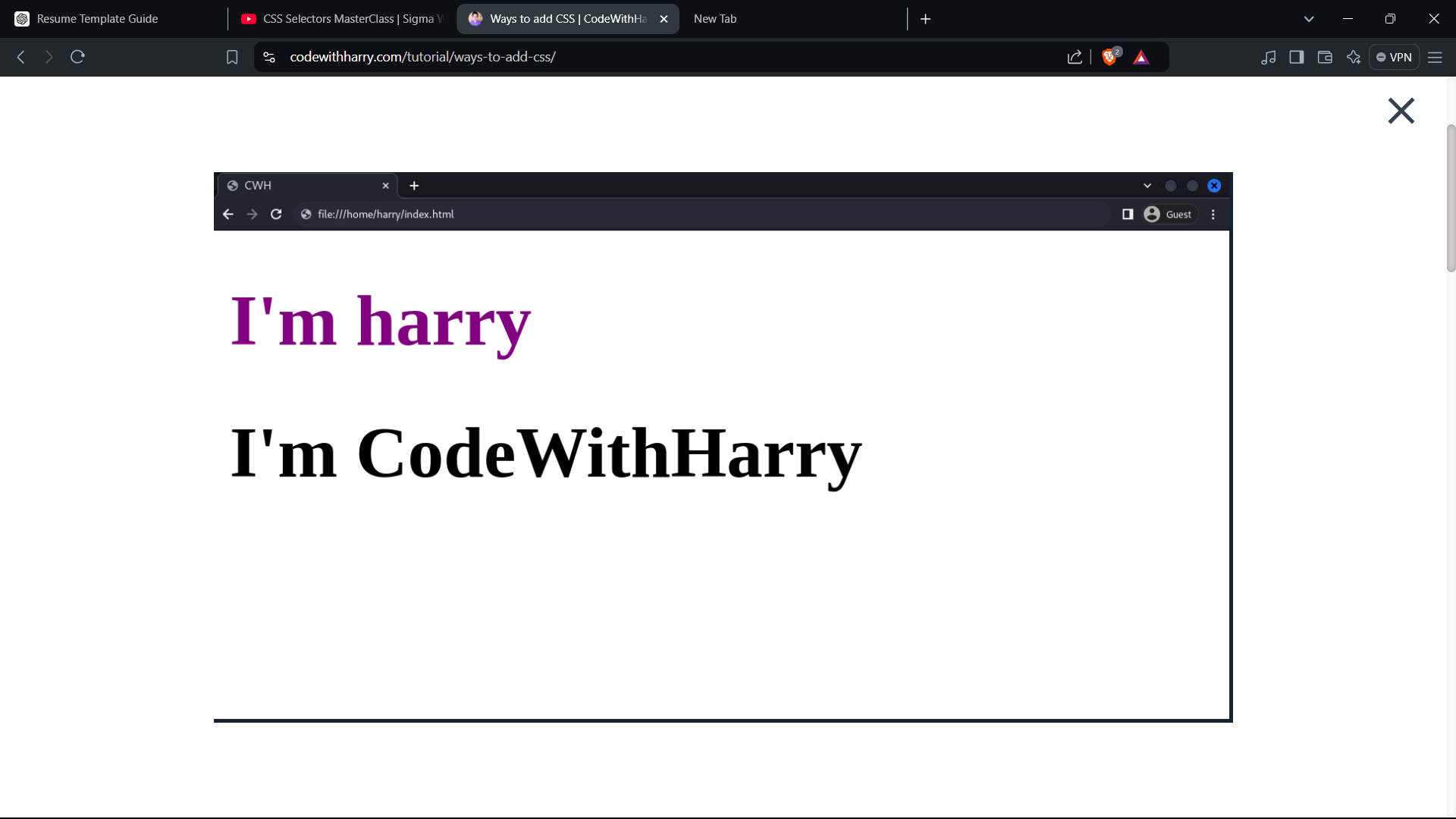


Inline CSS is used to add custom properties to specific elements. The added style will only reflect on that particular element only.

To use inline CSS, Insert the "style" attribute within the HTML element's opening tag.

EXAMPLE:

<h1 style="color: purple;">I'm harry</h1>

<h2>I'm CodeWithHarry</h2>

Note: The downside of using inline CSS is, that once the project complexity increases, it will become difficult to manage the styles of each and individual elements.

**Note: The <style> block should always be in the <head> sec**

* External CSS works similarly to internal CSS but with a twist. Instead of adding the styles within the HTML file, we create a separate file with **.css**extension. This file will hold all the styling details. Then, we link this file to the HTML page, giving it the instructions on how to look.

There is a new **<link>**tag in the **head section,**and this **link** tag has **rel**and**href**properties**.**

The following points will explain each keyword's meaning**:**

* **<link>**: This tag is used to create links between different resources, like stylesheets, fonts, and more. In our case, we are using a link tag to link the **CSS** file with the **HTML** file.
* **rel="stylesheet"**: **rel**stands for **relationship,** this defines the type of relationship between the HTML document and the linked resource. When set to "stylesheet", it specifies that the linked resource is a stylesheet that will be used to style the HTML content.
* **href="style.css"** : The **href** attribute stands for "**hypertext reference**." It specifies the path or URL to the external resource we want to link. In this case, it's the path to the external CSS file called "style.css".

<html lang="en">

<head>

<title>CodeWithHarry</title>

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

</head>

<body>

<p>I'm harry, from CodeWithHarry</p>

<p>I'm a Developer and founder of CodeWithHarry.com</p>

</body>

</html>

This approach enables to use of the same CSS to multiple HTML files, wherever the same custom style is required.

This is helpful when we have to maintain consistency on our web pages and want to use the same CSS styles across multiple pages.

**Note:**The precedence is Inline CSS > Internal CSS > External CSS. If we define the same property with different property values in three different ways, the element will have the property value of inline CSS.

CSS selectors allow us to choose specific elements and apply styles to them. Suppose we want to add a custom style to only a specific tag(s). There, We can make use of CSS selector.

There are different types of CSS selectors, which are as follows:

* [Universal Selector](https://www.codewithharry.com/tutorial/CSS-selectors/#universal)

Universal selector represented by "**\***" targets all the HTML elements on the page.

The syntax of Universal Selector is as follows:

\*

{

property : value;

}

<html>

<head>

<style>

\* {

color: purple;

text-align: center;

}

</style>

</head>

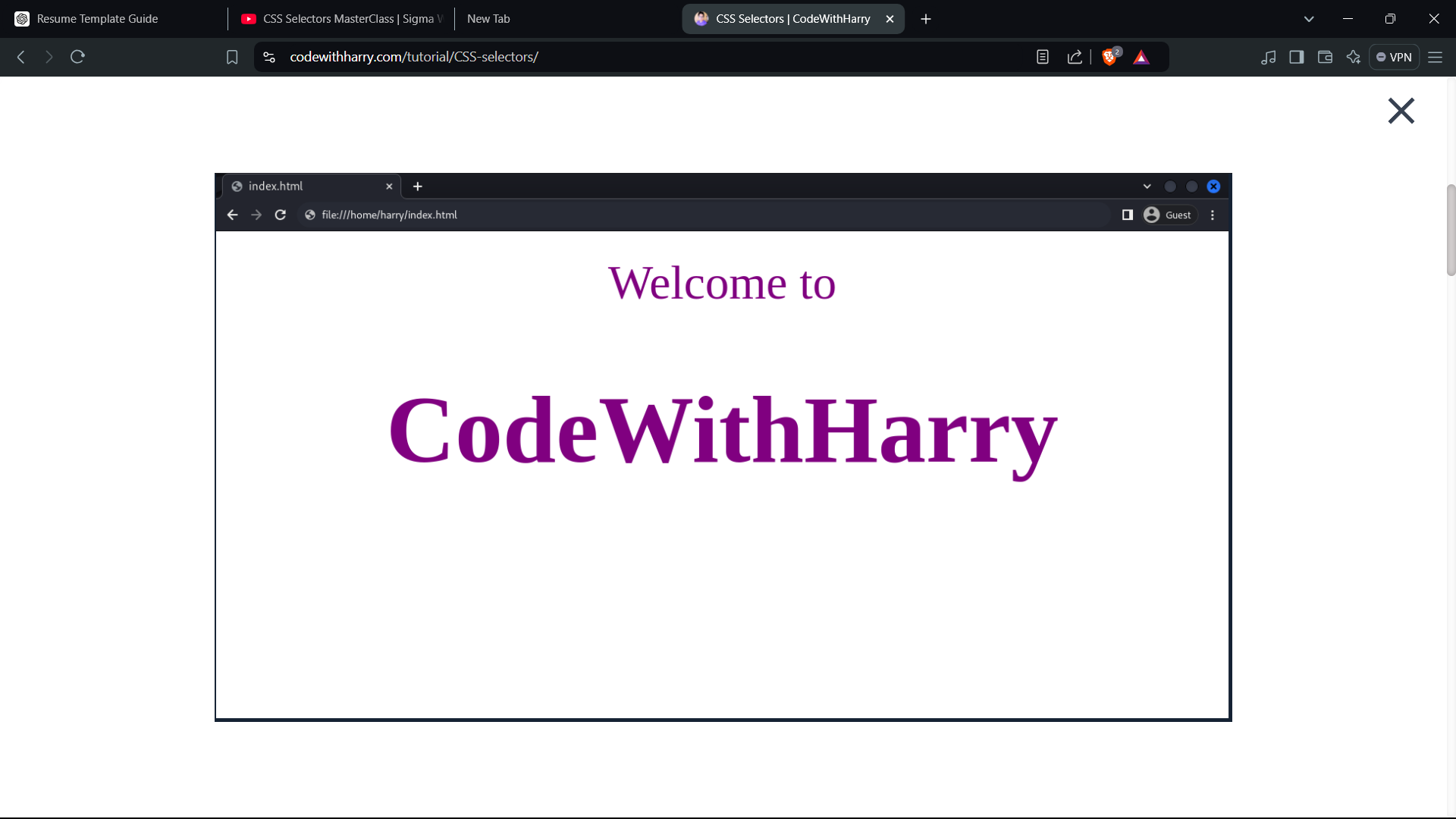
<body>

<p>Welcome to </p>

<h1>CodeWithHarry</h1>

</body>

</html>



Notice, Irrespective of the tag, the style will be applied to all the elements and tags.

* [Element Selector](https://www.codewithharry.com/tutorial/CSS-selectors/#element) **(Type Selector)**

The element selector selects the target element based on the specific type. Suppose you want to underline all the <p> tags; in this case, the element selector will be the best choice.

The syntax of Element Selector is as follows:

p

{

property : value;

}

A selector can be any HTML tag. Here, we have considered the p tag.

<html>

<head>

<title>CSS</title>

<style>

P

{

text-decoration: underline;

}

</style>

</head>

<body>

<h1>CodeWithHarry</h1>

<h2>we offer: </h2>

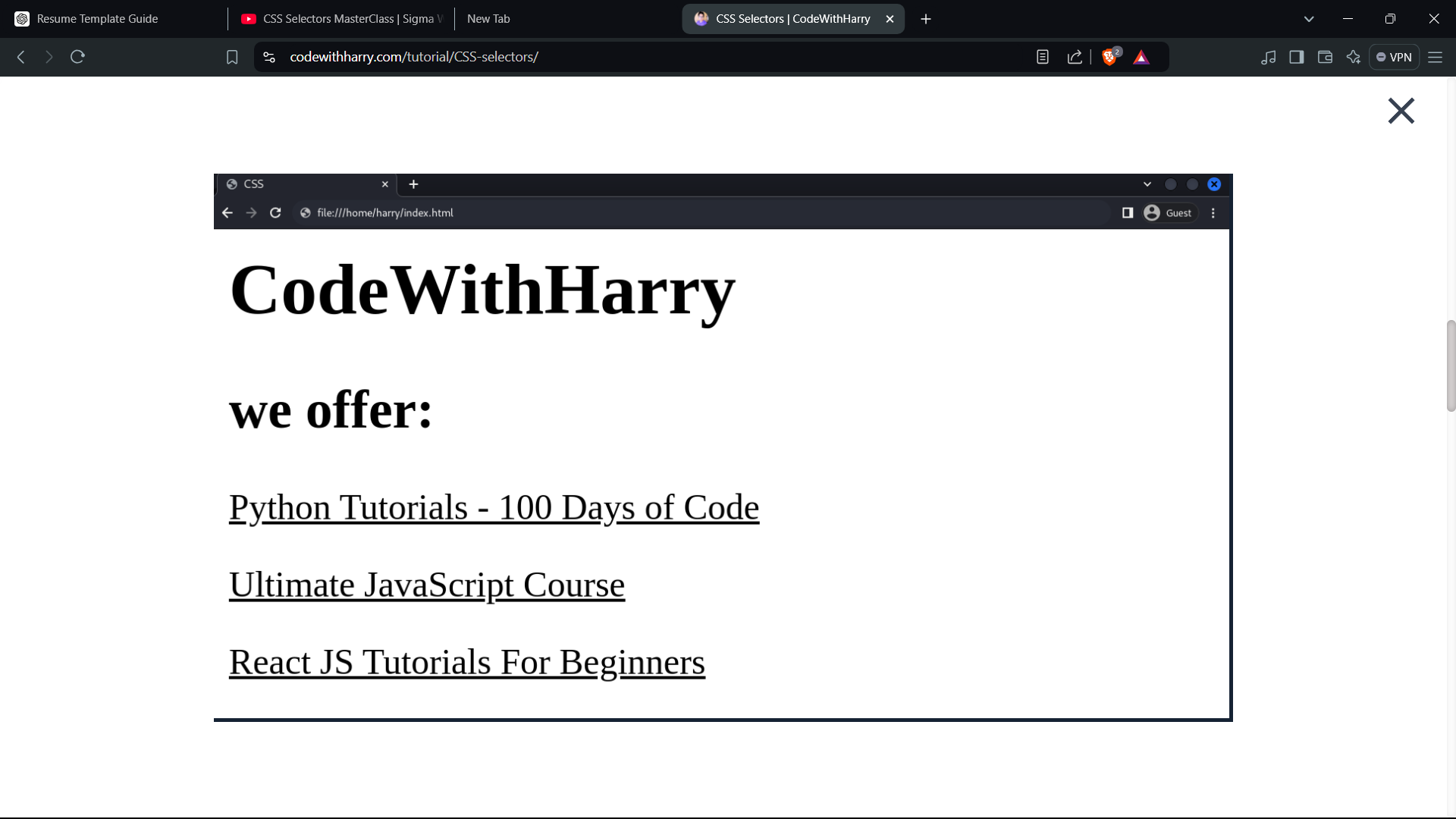
<p>Python Tutorials - 100 Days of Code</p>

<p>Ultimate JavaScript Course</p>

<p>React JS Tutorials For Beginners</p>

</body>

</html>



Note: Element selector is not recommended as the same tag can be used multiple times in the document. So, overlapping rules can occur in the stylesheet.

* [Id Selector](https://www.codewithharry.com/tutorial/CSS-selectors/#id)

The ID selector targets the elements based on the specific ID. It is written with the hash **“**#**”** character followed by the ID name in the style sheet.

The syntax of ID Selector is as follows:

#ID

{

property : value;

}

<html>

<head>

<style>

#title {

text-align: center;

color: red;

}

</style>

</head>

<body>

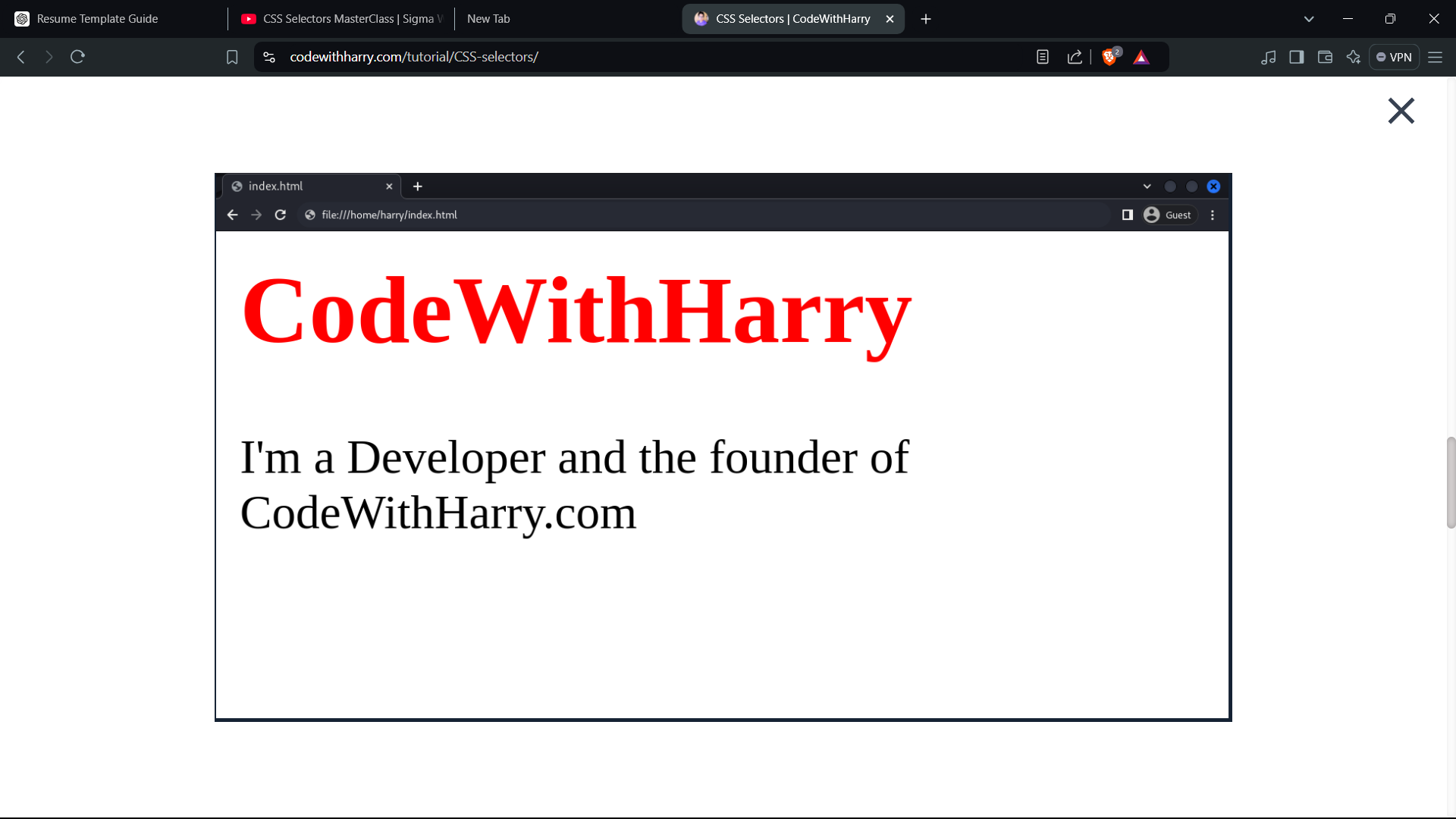
<h1 id="title">CodeWithHarry</h1>

<p>I'm a Developer and the founder of CodeWithHarry.com</p>

</body>

</html>

In the style block, the selector  **#title,** will only target the HTML element having an ID of "title".



Notice, how the property color: red is only applied to <h1> tag.

* [Class Selector](https://www.codewithharry.com/tutorial/CSS-selectors/#class)

The class selector does the same job as the id selector, a class selector helps group various types of elements. Suppose, we want to give a custom style to a specific group of elements. In this case, the class selector is the best option.

It is written with the period **“.”** character followed by the class name in the style sheet.

The syntax of Class Selector is as follows:

.class {

property : value;

}

<html>

<head>

<title>CSS</title>

<style>

.red {

color: red;

}

</style>

</head>

<body>

<p>This is simple p tag</p>

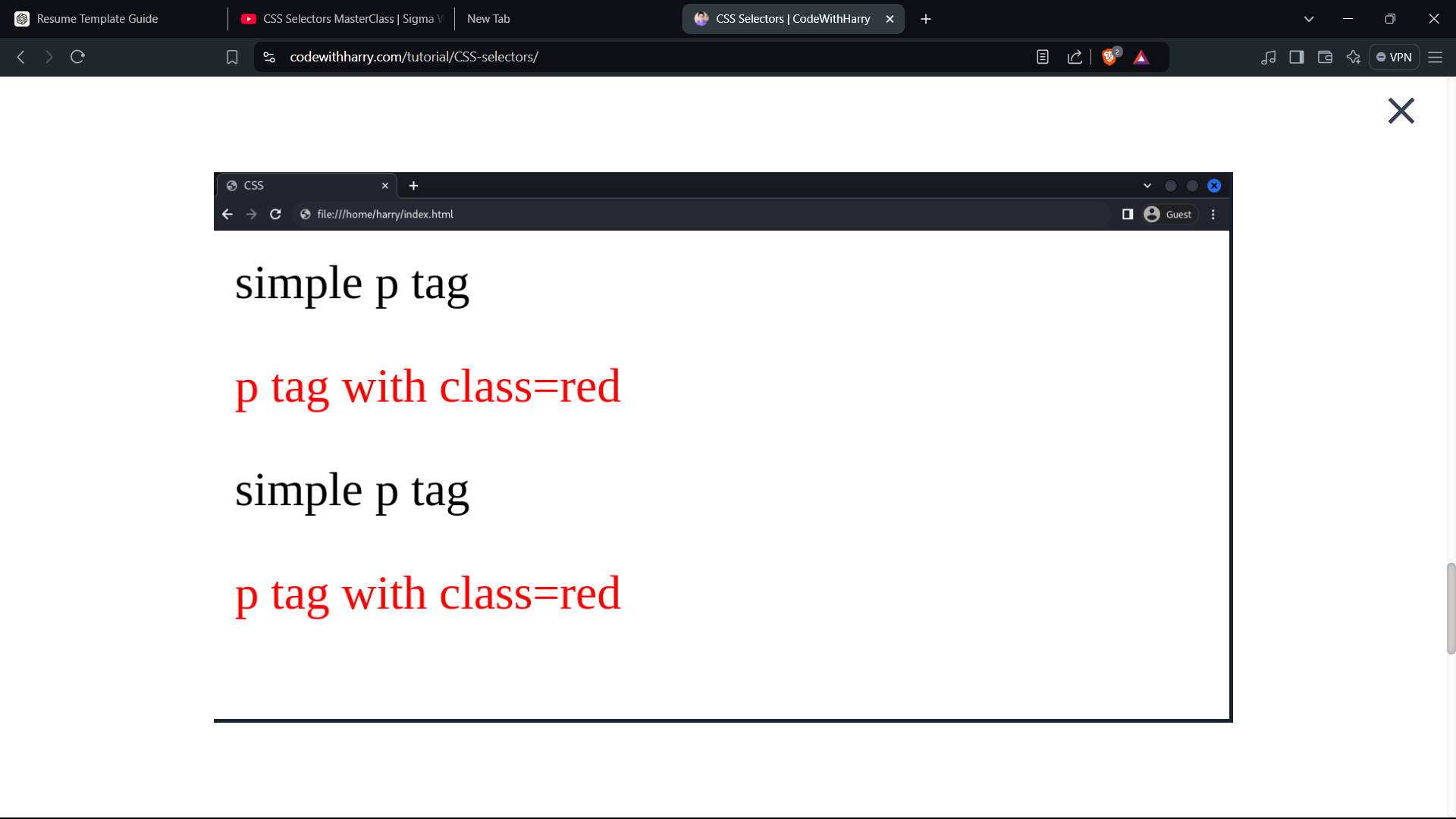
<p class="red">This p tag has class red</p>

<p>This is simple p tag</p>

<p class="red">This p tag has class red</p>

</body>

</html>



Note: The class selector helps in grouping two or more elements.

* [Group Selector](https://www.codewithharry.com/tutorial/CSS-selectors/#group)

The group selector is used to minimise the code. Commas **","** are used to separate each selector in a grouping. This reduces the number of lines of code. The code also looks clean.

The syntax of Group Selector is as follows:

div, p, a

{

property : value;

}

<html>

<head>

<title>CSS</title>

<style>

h1 {

color: red;

}

p,a {

color: purple;

}

</style>

</head>

<body>

<h1>CodeWithHarry</h1>

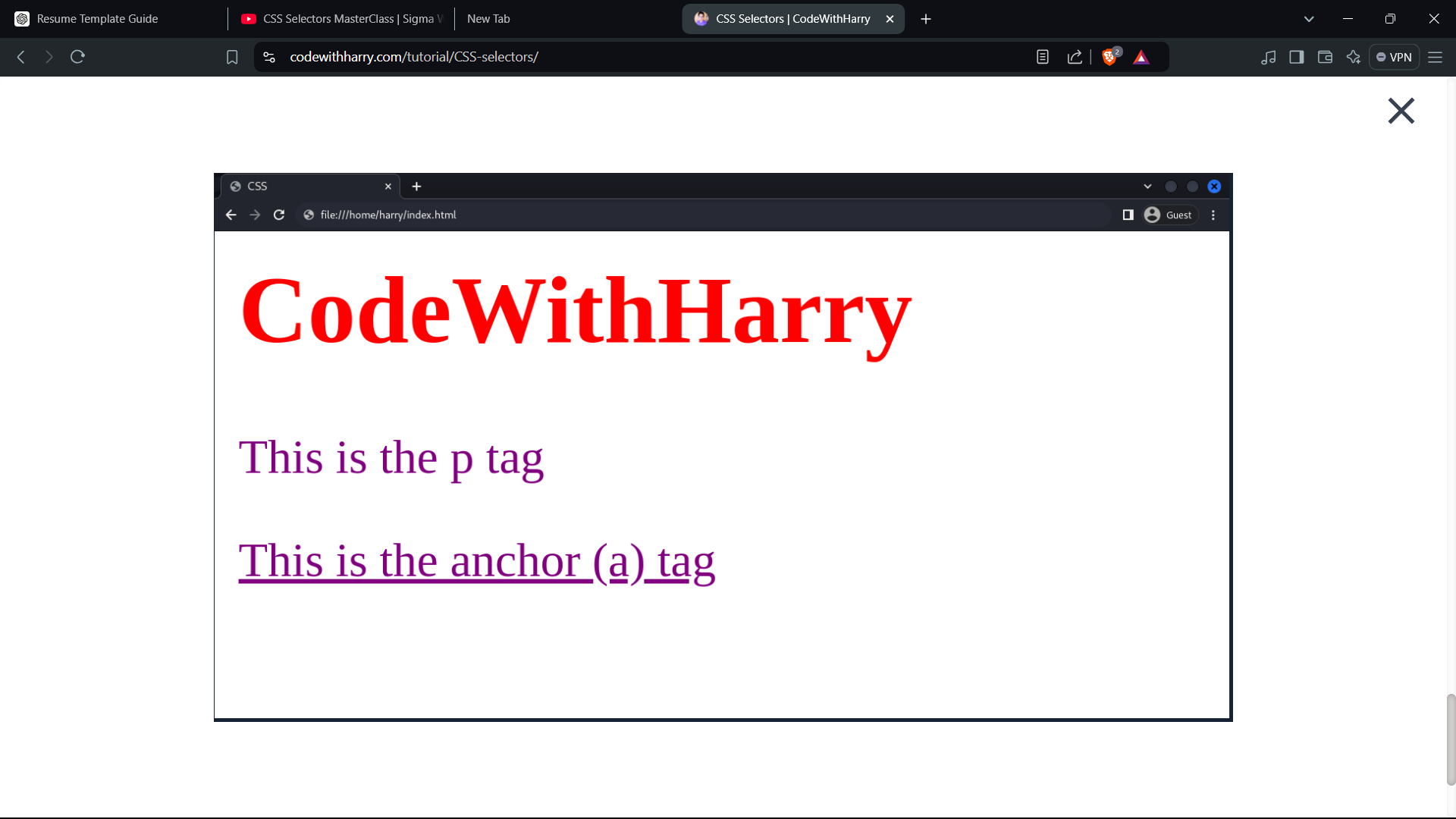
<p>This is the p tag</p>

<a href="#">This is the anchor (a) tag</a>

</body>

</html>

In the <style> block, p and a tag are grouped together so, that both tags will have the same properties.



* Universal Selector(\*): Target the entire page.
* Element Selector: Target a specific element.
* ID Selector(#): Target element with a specific ID.
* Class Selector(.): Target element(s) with the same class.
* Group Selector: Group elements and target them.

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>CSS Selectors</title>

</head>

<body>

    <main class="one">

        <!--

         The <main> tag is an HTML5 semantic element used to designate the dominant content of the <body>. It typically contains the central part of the web page that is directly related to the page's purpose. The <main> element should be unique to the page and should not be used more than once.

         -->

        <p>I am first</p>

        <p>I am second</p>

    </main>

    <div class="red">

        I am a div

        <article>

            <!-- "article" is article -->

            <p>I am a para inside div</p>

        </article>

    </div>

    <div id="green">

        I am another div

    </div>

    <a href="https://www.google.com">Go to Google</a>

    <a href="https://www.facebook2.com">Go to Facebook</a>

    <style>

        /\* Element Selector  \*/

        div{

            color: white;

        }

        /\* Class Selector  \*/

        .red{

            background-color: red;

        }

        /\* Id Selector \*/

        #green{

            background-color: green;

        }

        /\* Child Selectors  \*/

No, you cannot use a child selector (>) to target a class in CSS.

        div > p {

            color: blue;

            background-color:black;

        }

        /\* Descendant Selector  \*/

        /\*

         The descendant selector in CSS is used to select all elements that are descendants of a specified ancestor. A descendant is any element that is nested within another element, no matter how deep the nesting goes.

         \*/

         /\*

         If you only want to target direct children (not all descendants), you would use the child selector (>).

         \*/

        div p {

            color: blue;

            background-color: brown;

        }

        /\* Universal Selector \*/

        \* {

            margin:0;

            padding: 0;

        }

        /\* Pseudo Selector  \*/

        /\*

        A pseudo-class in CSS is a keyword added to selectors that specifies a special state of the selected elements. Pseudo-classes are typically used to style elements based on their state (such as when they're hovered over or clicked) or based on their position within the document (such as the first child in a parent element).

        \*/

        a:visited{

            color: yellow

        }

        a:link {

            color: green;

        }

        a:active{

            background-color: red;

        }

        a:hover{

            background-color: yellow;

        }

        p:first-child

        /\*

"p" ka bhi "first child" (in this case jahan likha hai "I am first"

\*/

        {

            background-color: black;

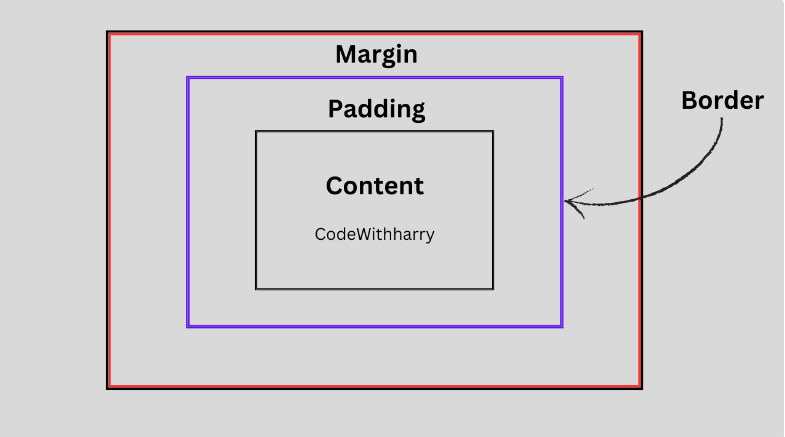
        }

    </style>

</body>

</html>

1. CSS BOX MODEL



innermost component of the box model is the actual content of the element. It can be text, image, video, etc

space between the actual content and the border of the element is the padding.

 border surrounds the content and padding and gives the visual border of the element.

margin is the space outside the element that separates it from other elements in the layout.

**Total Width** = Width + Left Padding + Right Padding + Left Border + Right Border + Left Margin + Right Margin

**Total Height** = Height + Top Padding + Bottom Padding + Top Border + Bottom Border + Top Margin + Bottom Margin

<head>

<style>

p{

width: 200px;

height: 300px;

padding: 15px;

border: 10px solid red;

margin: 5px;

}

</style>

</head>

<body>

<p>CodeWithHarry</p>

</body>

</html>

Here, the total height and width will be represented as

**Total Width** = 200px (width) + 15px (left padding) + 15px (right padding) + 10px (left border) + 10px (right border) + 5px (left margin) + 5px (right margin) = 260px.

**Total Height** = 300px (Height) + 15px (Top Padding) + 15px (Bottom Padding) + 10px (Top Border) + 10px (Bottom Border) + 5px (Top Margin) + 5px (Bottom Margin) = 360px

NOTE: MARGIN COLLAPSE **Margin collapse** is a behavior in CSS where the vertical margins of adjacent block-level elements collapse into a single margin that is equal to the largest of the margins involved. This can occur in a few different scenarios:

**1. Adjacent Siblings:**

When two block-level elements are placed one after the other, their vertical margins collapse into one. For example:

<div style="margin-bottom: 20px;">Element 1</div>

<div style="margin-top: 30px;">Element 2</div>

In this case, the margin between Element 1 and Element 2 will be 30px (the larger of the two margins), not 50px (20px + 30px).

**2. Parent and Child:**

If a block-level element contains another block-level element, and there is no border, padding, or inline content between them, the margin of the child element can collapse with the margin of the parent. For example:

<div style="margin-top: 30px;">

<p style="margin-top: 20px;">Child element</p>

</div>

In this case, the parent and child margins collapse, and the resulting top margin is 30px, not 50px.

**3. Empty Block Elements:**

If a block-level element has no height, padding, or border, and contains no content, its margins will collapse with the margins of adjacent elements.

1. FONTS

// github link to be given

1. CSS SPECIFITY

Order of specifity

**Inline Style > ID Selector > Class or Attribute Selector > Element Selector > Universal Selector**

* Universal Selector: 0
* Element selectors and pseudo-elements: 1
* Class selectors, attribute selectors, and pseudo-classes: 10
* ID selectors: 100
* Inline styles: 1000

If the specifity no. of electors are tie then the latest position selector will give the color

A pseudo-class is used to define a special state of an element, such as when it's being hovered over, clicked, or selected. It acts like a class in CSS, but it’s applied dynamically based on the user's actions or other conditions.

A pseudo-element is used to style a specific part of an element, like the first letter, first line, or inserting content before or after the element.

Harry’s advice: don’t use !important or inline

Instead try to give color using classes only

//github link to be given

1. CSS SIZING UNITS

width of the screen > width of our website

then horizontal scroll bar will appear

px=(inch/196)

**1. Inline Elements:**

* **Takes up only as much width as needed** by its content.
* **Does not start on a new line**: It remains in line with surrounding content (like text).
* **Ignore width and height properties**. You can’t explicitly control their dimensions, as they are sized based on their content.

2. **Block-Level Elements**:

 **Takes up the full width** of its parent container, even if its content doesn’t require that much space.

 **Starts on a new line**: Each block-level element appears on its own line, separating content from the elements before and after it.

When I say that a block-level element (like a <div>)r **"starts on a new line"**, I mean that it forces any content after it to appear below it on the webpage, rather than continuing on the same line.

**Example:**

**Block-Level Element (<div>)**

<div>This is a block-level element.</div>

<p>This is another block of content.</p>

**Output**:

This is a block-level element.

This is another block of content.

In this case, the <div> forces the text following it to appear on a new line, stacking vertically.

**Inline Element (<span>)**

<p>This is an inline element: <span>This is inside a span.</span> See how it stays inline.</p>

**Output**:

This is an inline element: This is inside a span. See how it stays inline.

Here, the <span> element keeps everything on the same line. It does **not** create a break before or after it.

By default, the CSS box model works with box-sizing: content-box, which means:

* The width and height you set for an element only apply to the **content**.
* Any padding, borders, and margins are added on **outside** of the set width/height, making the element larger.

//github link to be given

1. CSS SHADOWS

//github link to be given

1. STYLING LISTS

<https://github.com/HK51104/WEBDEVELOPMENT/blob/main/Sigma-Web-Dev-Course-main/Video%2025/index.html>

1. CSS OVERFLOW

[https://github.com/HK51104/WEBDEVELOPMENT/blob/main/Sigma-Web-Dev-Course-main/Video%2026/index.html](https://github.com/HK51104/WEBDEVELOPMENT/blob/main/Sigma-Web-Dev-Course-main/Video%2026/index.html%09)

1. CSS POSITION PROPERTY

By default the position is static(top,bottom,left,right don’t work)

Exception: transform, filter, or perspective properties can also make an element appear as positioned

<https://github.com/HK51104/WEBDEVELOPMENT/blob/main/Sigma-Web-Dev-Course-main/Video%2028/index.html>

NOTE: try to first adjust margin then border then padding then width in order to bring the content in the right shape and size.

1. CSS VARIABLE

To be done….

1. MEDIA QUERIES

To be done…

1. CSS FLOAT AND CLEAR

GIT ADD .

GIT COMMIT -M “ANY MESSAGE”

Git push