World Wide Web

INTRODUCTION:-

The World Wide Web (WWW) is a technological innovation and a digital heartbeat of our interconnected world. Born from the visionary mind of Sir Tim Berners-Lee in 1989 and realised in 1991, the WWW is a testament to the boundless possibilities that arise when human ingenuity meets the internet's vast potential. Beyond being a network of websites, it is a dynamic ecosystem where information flows freely, transcending geographical and cultural barriers. The WWW weaves humanity's collective knowledge together in its intricate web of hyperlinks, forming a virtual tapestry across continents and disciplines.

As a catalyst for societal transformation, the WWW has become the cornerstone of the Information Age, reshaping the way we learn, communicate, and conduct business. Its features, ranging from the fundamental concept of hyperlinks to the seamless integration of multimedia, create an immersive digital experience that engages users on multiple levels. The universal resource identifiers (URIs) assigned to each digital entity underscore the precision with which the WWW organises and disseminates information, ensuring a structured and accessible digital landscape.

The WWW's journey is more than a technological evolution; it's a narrative of empowerment, where individuals and communities gain unprecedented access to information. This introduction aims to set the stage for a comprehensive exploration of the WWW's features, its profound significance in meeting the diverse needs of our interconnected society, and the riveting historical milestones that have shaped its evolution into the indispensable fabric of the modern digital era.

History of WWW:-

1. **Invention of the Web:**
   - Origin: Sir Tim Berners-Lee's vision for the WWW emerged in 1989 while he was at CERN, reflecting a desire to create a globally interconnected information system.
   - Implementation: The first tangible manifestation of this vision was the launching of the first website (info.cern.ch) in 1991, marking the birth of the WWW.

2. **First Website and Browser:**
   - Milestones: The inauguration of the first website was accompanied by the development of the WorldWideWeb browser (later renamed Nexus) in 1991.
   - Pioneering Access: These milestones laid the groundwork for a new era where individuals could navigate and contribute to the digital realm.

3. **Commercialisation:**
   - Shift: The mid-1990s witnessed the commercialisation of the WWW with the advent of popular browsers like Netscape.
   - Dot-com Boom: This phase catalysed the boom, characterised by an influx of online businesses and investments in internet-related ventures.

4. **Standardisation:**
   - Necessity: As the WWW expanded, the need for standardisation became evident, establishing crucial web technologies like HTML and HTTP.

- W3C: The World Wide Web Consortium (W3C) emerged as a critical player in setting and maintaining web standards, ensuring compatibility and uniformity.

5. **Evolution:**
   - Technological Advancements: The web has evolved dynamically, spurred by technological advancements, giving rise to dynamic, interactive web applications.
   - User Expectations: From static HTML pages to real-time, collaborative platforms, the evolution of the web has been shaped by its users' changing expectations and demands.

Features of WWW:-

1. **Hyperlinking:**
   - Definition: At the heart of the WWW is the concept of hyperlinking, a mechanism that allows users to traverse the digital landscape effortlessly.
   - Functionality: Hyperlinks serve as digital bridges, connecting disparate web pages and creating a tapestry of information users can navigate at will.

2. **Multimedia Content:**
   - Definition: Unlike traditional forms of communication, the WWW is not confined to text; it embraces a rich multimedia tapestry, including images, audio, and video.
   - Versatility: This multimedia integration ensures that information is conveyed through words and a spectrum of sensory experiences.

3. Interactivity:

- Definition: The WWW is not a static repository but a dynamic platform fostering interactivity through forms, comments, and real-time collaboration.
- Engagement: Users are not passive consumers but active participants, contributing to the collective intelligence that defines the web.

4. Universal Resource Identifiers (URIs):

- Definition: Each digital entity on the web is assigned a unique Universal Resource Identifier (URI), commonly known as a URL.
- Uniqueness: URIs provide a precise and globally recognised way to identify and access any resource on the vast network of the WWW.

5. Web Browsers:

- Definition: Web browsers, such as Chrome, Firefox, and Safari, serve as the windows through which users interact with the digital realm of the WWW.
- Accessibility: These browsers translate complex web technologies into user-friendly interfaces, making the web accessible to people around the globe.

6. Search Engines:

- Definition: Search engines, led by giants like Google and Bing, act as the web's gatekeepers, indexing and retrieving information based on user queries.
- Discovery: They play a pivotal role in enabling users to discover relevant content amidst the vast sea of information.

Why WWW was Invented:-

When the World Wide Web (WWW) was invented by Sir Tim Berners-Lee in 1991, it addressed several crucial needs in information sharing and communication. Here are some key reasons why the WWW was invented:

1. **Information Access:** The primary need was to create a system that would make vast amounts of information easily accessible to people. Before the WWW, data was often stored in isolated databases or libraries, making it challenging for individuals to access a wide range of knowledge conveniently.
2. **Interconnected Information:** The WWW aimed to connect information in a way that had not been done before. Hyperlinks allow users to navigate seamlessly from one piece of information to another, creating a web of interconnected data. This structure significantly improved the ease with which users could explore diverse topics.
3. **Universal Access:** The WWW was designed to be universally accessible. It didn't matter where you were in the world; as long as you had an internet connection and a web browser, you could tap into a global information network. This universality was a groundbreaking concept that democratised access to knowledge.
4. **Communication:** Besides accessing information, the WWW was meant to facilitate communication. Tim Berners-Lee envisioned a platform where people could not only consume content but also interact with it and with each other. This laid the groundwork for developing email, forums, and social media.
5. **Standardisation of Protocols:** The WWW introduced standardised protocols like HTTP (Hypertext Transfer Protocol) and HTML (Hypertext Markup Language). These standards ensured that information could be exchanged seamlessly between computers and systems, fostering compatibility and interoperability.
6. **Global Collaboration**: The WWW aimed to enable collaboration on a worldwide scale. Researchers, scientists, and individuals could share their work, findings, and ideas without the limitations imposed by geographical distances. This interconnectedness sparked a new era of collaboration and information exchange.

7. **Easier Publishing:** Before the WWW, publishing content for a broad audience often involved traditional media channels. The WWW democratised publishing, allowing anyone with internet access to create and share content. This democratisation paved the way for a diverse range of voices and perspectives.
8. **Commercial Opportunities:** The WWW invention opened up new business possibilities. It provided a platform for e-commerce, allowing companies to reach customers globally and conduct transactions online. This shift laid the foundation for the digital economy and the rise of online businesses.

Conclusion:-

The World Wide Web (WWW) constitutes a transformative paradigm, transcending its status as mere technology to represent a fundamental shift in how global society accesses information and communicates. Beyond its foundational elements, such as hyperlinks and multimedia integration, the WWW stands as a dynamic and interconnected digital symphony.

Its historical journey, from the inauguration of the first website to the standardisation of essential protocols, reveals a collective commitment to achieving universal accessibility. This intricate digital tapestry binds the world together, creating a space where diverse voices converge and information flows seamlessly, overcoming geographical limitations.

The WWW, extending beyond its role as a technological marvel, is a testament to the collective human endeavour to foster connectivity, communication, and innovation. It mirrors the power of information to shape societies, obliterate barriers, and thrust us into an era where the boundaries between the physical and digital realms are increasingly indistinct.

In essence, comprehending the WWW entails meticulously exploring a global network that has fundamentally redefined how knowledge is acquired, communication is conducted, and the future is envisioned. It is a multifaceted tool that encapsulates the intricacies of our shared human experience in the digital age.

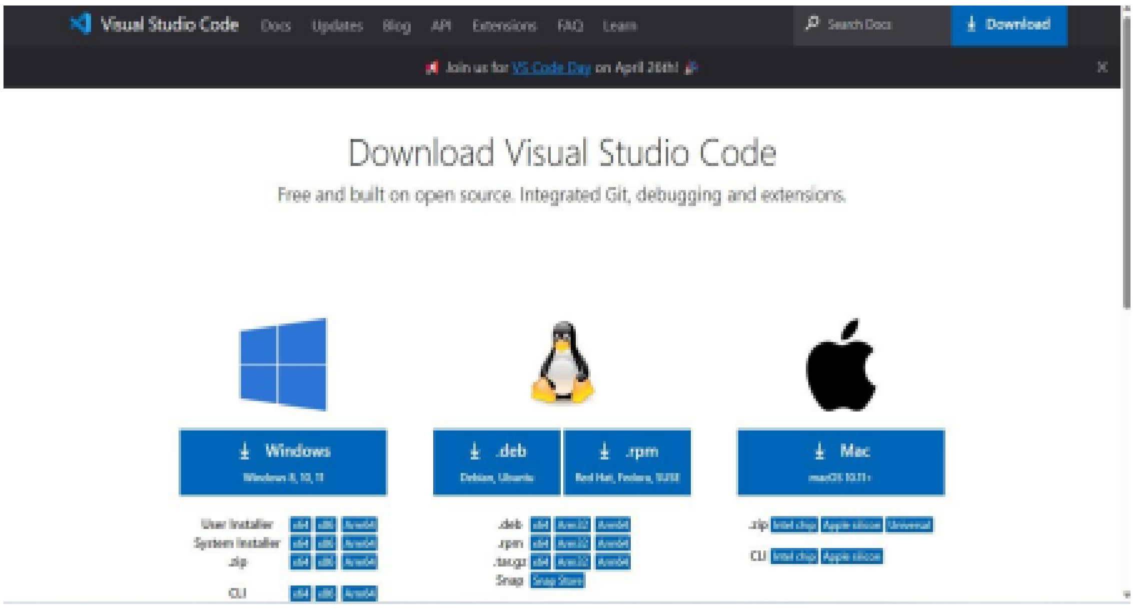
References:

- https://info.cern.ch/
- https://en.wikipedia.org/wiki/World_Wide_Web

Installing VS Code in Windows

To install Vs Code in Windows, follow these simple steps:

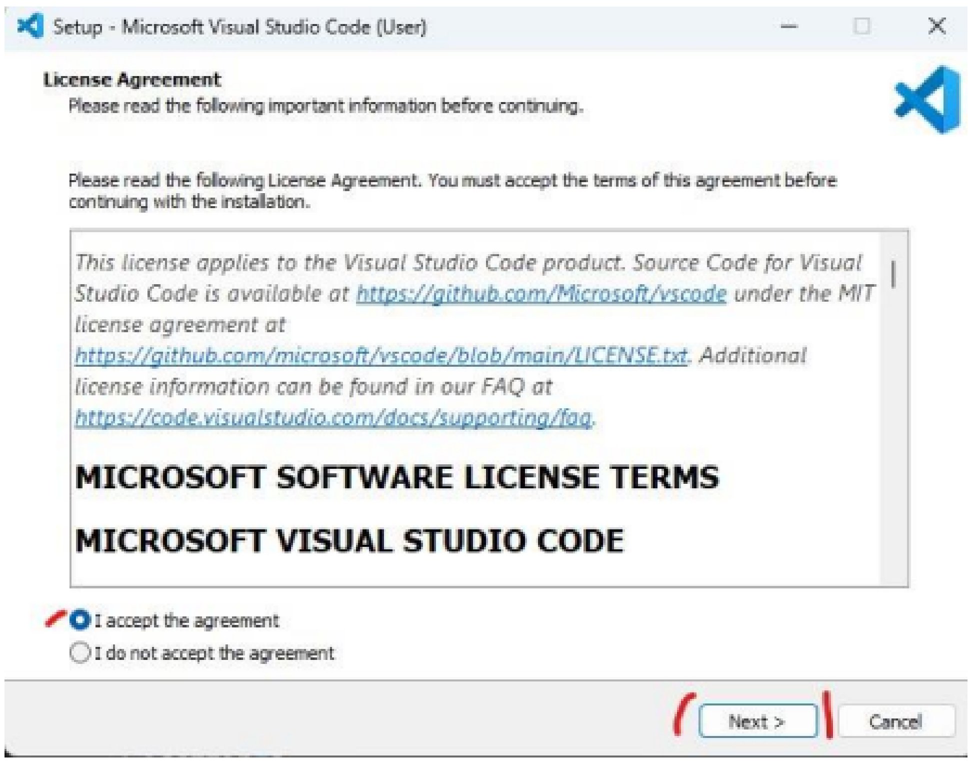
- Open the browser, and Type "VS Code Download, "and click on the first link it shows Click on the following link: https://code.visualstudio.com/download.
- You will be redirected to this page.



Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.

- Click on the download button for Windows. This will download the installer file for Visual Studio Code.
- Once the download is complete, run the installer file by double-clicking.
- You will see the setup wizard for Visual Studio Code. Select I accept the agreement and click on the "Next" button.


Setup - Microsoft Visual Studio Code (User)

License Agreement
Please read the following important information before continuing.

Please read the following License Agreement. You must accept the terms of this agreement before continuing with the installation.

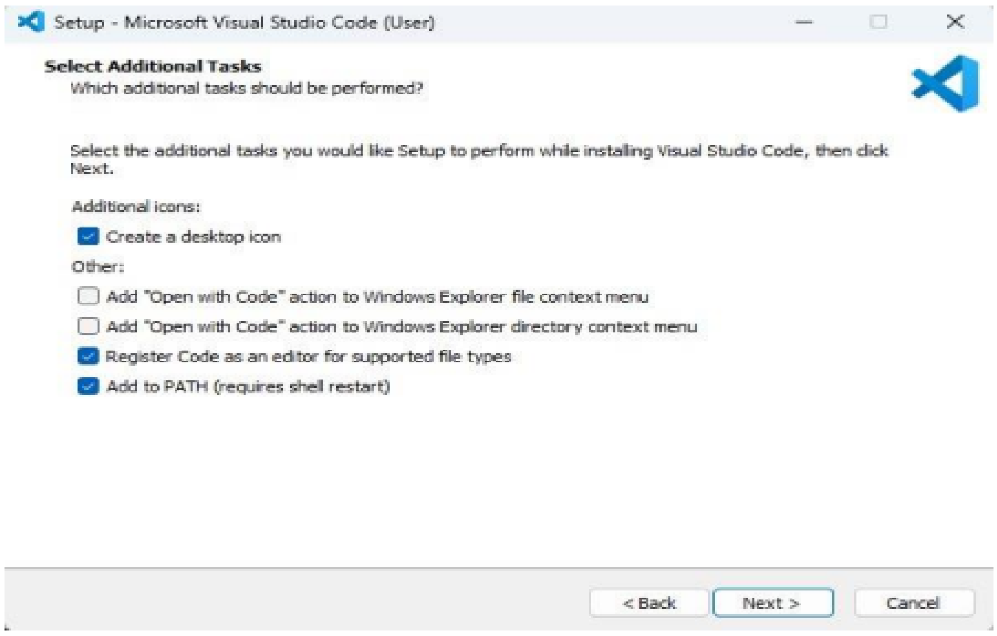
This license applies to the Visual Studio Code product. Source Code for Visual Studio Code is available at https://github.com/Microsoft/vscode under the MIT license agreement at https://github.com/microsoft/vscode/blob/main/LICENSE.txt. Additional license information can be found in our FAQ at https://code.visualstudio.com/docs/supporting/faq.

MICROSOFT SOFTWARE LICENSE TERMS

MICROSOFT VISUAL STUDIO CODE

[✓ I accept the agreement]
I do not accept the agreement

- The next screen will show you the installation location for Visual Studio Code on your computer. You can change this location if you want or leave it as the default location. Click on the "Next" button.
- The next screen will ask you to select the additional tasks you want to perform. You can check or uncheck the options as per your requirements. Check the "Create a desktop icon" to create a desktop shortcut. Click on the "Next" button.


Setup - Microsoft Visual Studio Code (User)

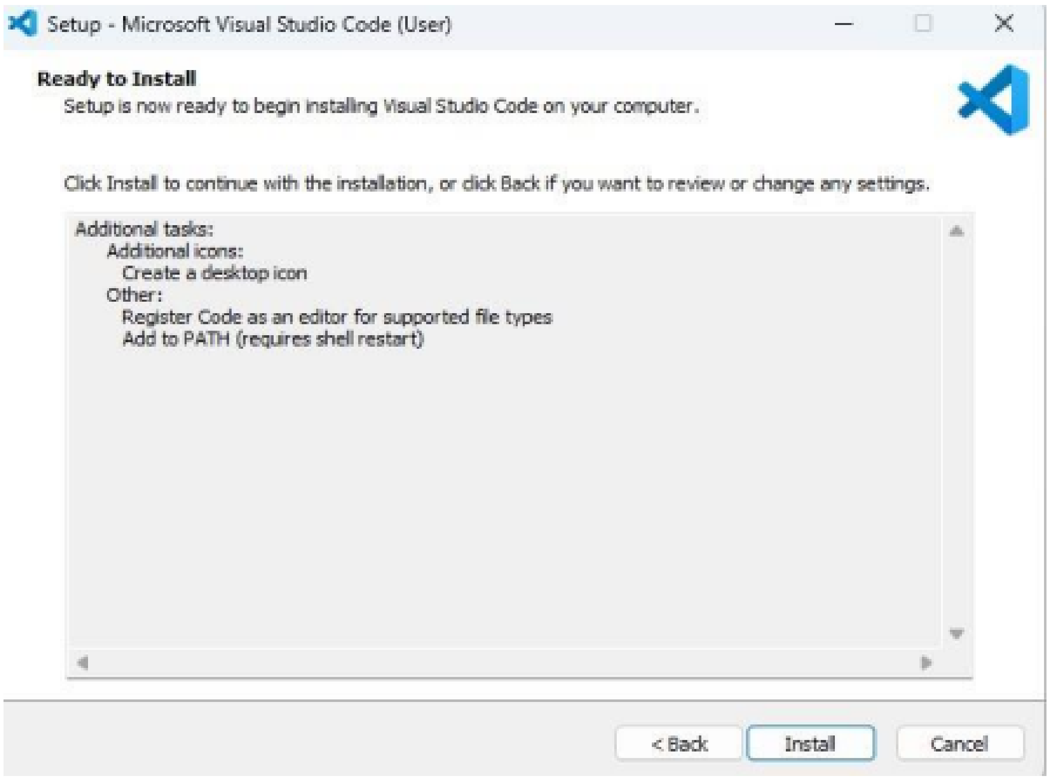
Select Additional Tasks
Which additional tasks should be performed?

Select the additional tasks you would like Setup to perform while installing Visual Studio Code, then click Next.

Additional icons:
- [✓] Create a desktop icon

Other:
- [ ] Add "Open with Code" action to Windows Explorer file context menu
- [ ] Add "Open with Code" action to Windows Explorer directory context menu
- [✓] Register Code as an editor for supported file types
- [✓] Add to PATH (requires shell restart)

- On the next screen, you will see a summary of your selected installation configurations. Click on the "Install" button to start the installation.



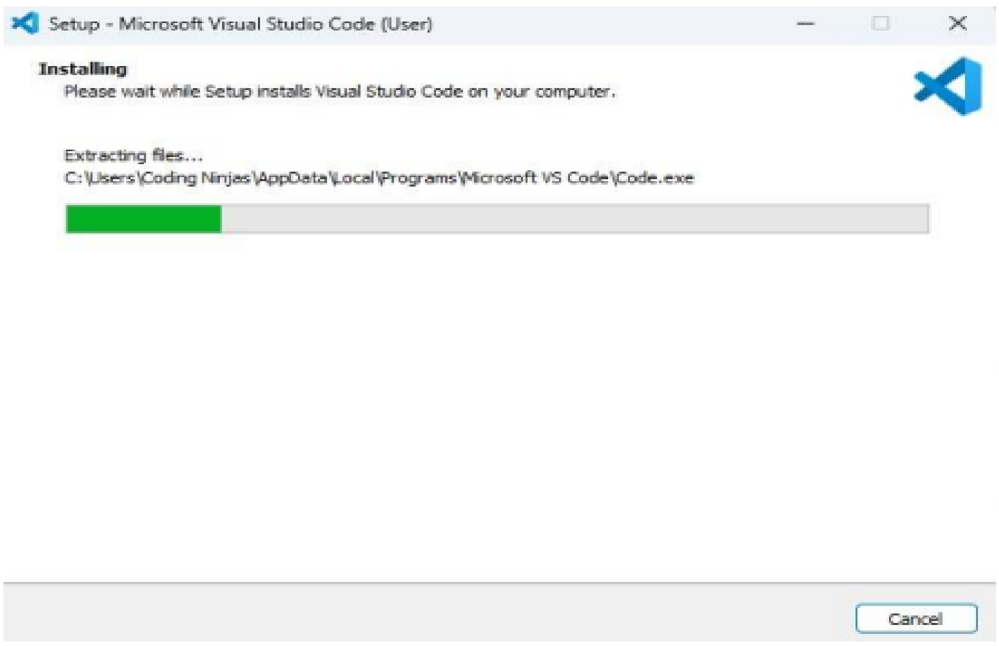
Setup - Microsoft Visual Studio Code (User)

Ready to Install
Setup is now ready to begin installing Visual Studio Code on your computer.

Click Install to continue with the installation, or click Back if you want to review or change any settings.

```
Additional tasks:
    Additional icons:
      Create a desktop icon
    Other:
      Register Code as an editor for supported file types
      Add to PATH (requires shell restart)
```

- Wait for the installation to complete. This may take a few minutes.

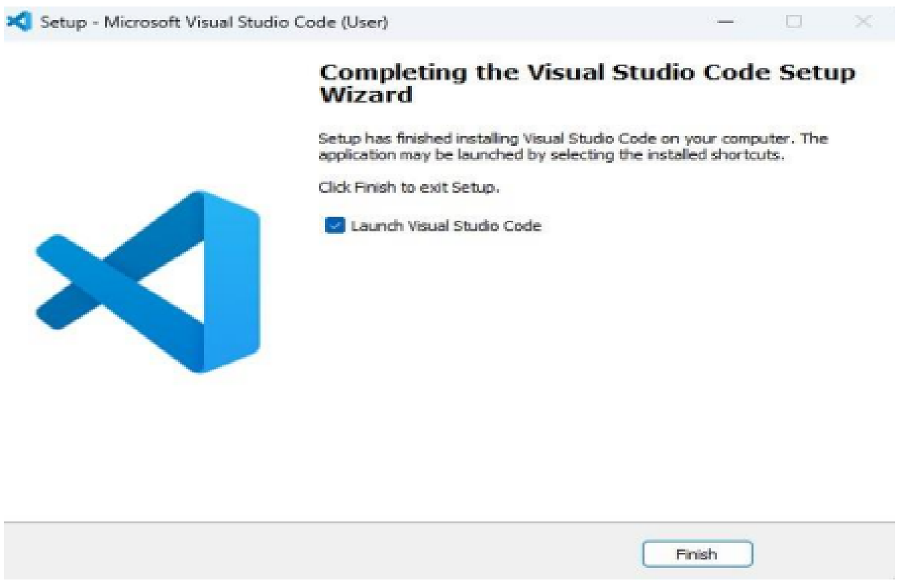


Setup - Microsoft Visual Studio Code (User)

Installing
Please wait while Setup installs Visual Studio Code on your computer.

Extracting files...
C:\Users\Coding Ninjas\AppData\Local\Programs\Microsoft VS Code\Code.exe

- Once the installation is complete, click on the "Finish" button.


Setup - Microsoft Visual Studio Code (User)

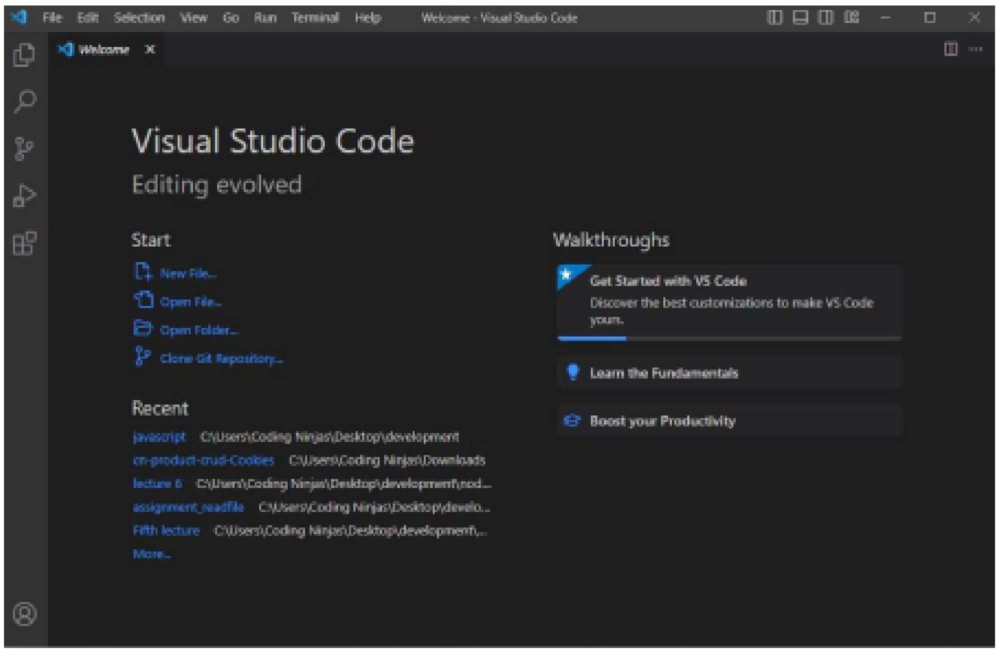
Completing the Visual Studio Code Setup Wizard

Setup has finished installing Visual Studio Code on your computer. The application may be launched by selecting the installed shortcuts.

Click Finish to exit Setup.

☑ Launch Visual Studio Code

- Visual Studio Code is now installed on your Windows computer. You can launch it from the Start menu or the desktop shortcut.


File
Edit
Selection
View
Go
Run
Terminal
Help
Welcome - Visual Studio Code
Welcome
Visual Studio Code
Editing evolved
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Recent
javascript C:\Users\Coding Ninjas\Desktop\development
cn-product-crud-Cookies C:\Users\Coding Ninjas\Downloads
lecture 6 C:\Users\Coding Ninjas\Desktop\development\nod...
assignment_readfile C:\Users\Coding Ninjas\Desktop\develo...
Fifth lecture C:\Users\Coding Ninjas\Desktop\development\...
More...
Walkthroughs
Get Started with VS Code
Discover the best customizations to make VS Code yours.
Learn the Fundamentals
Boost your Productivity

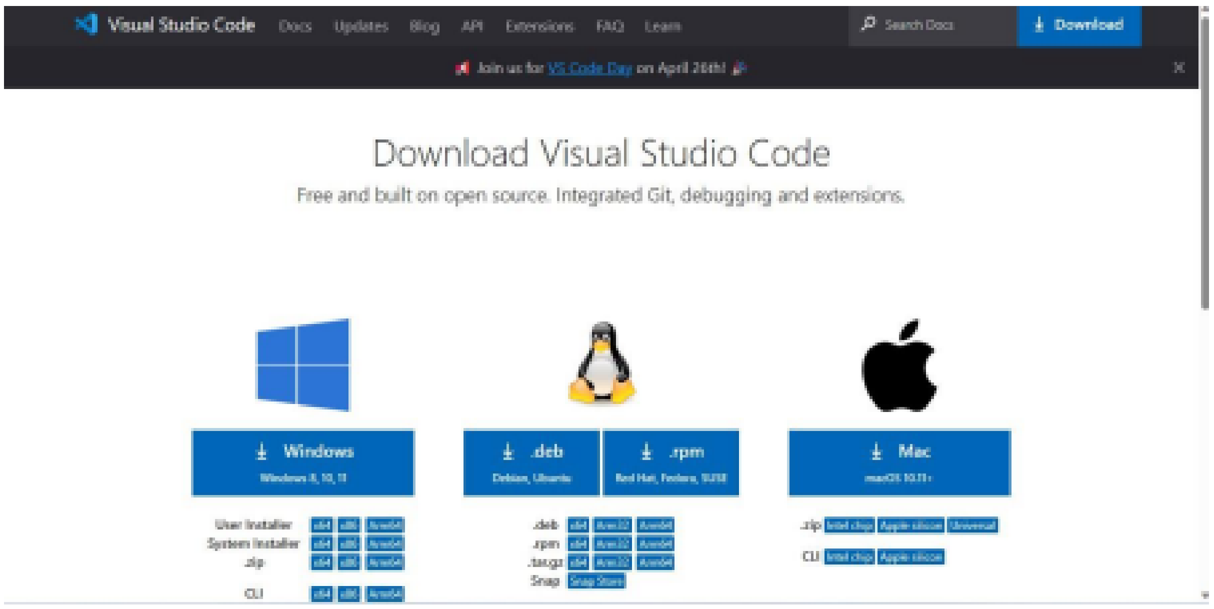
Installing VS Code on Mac

To install VS Code on Mac, follow these simple steps:

Go to the official Visual Studio Code website:

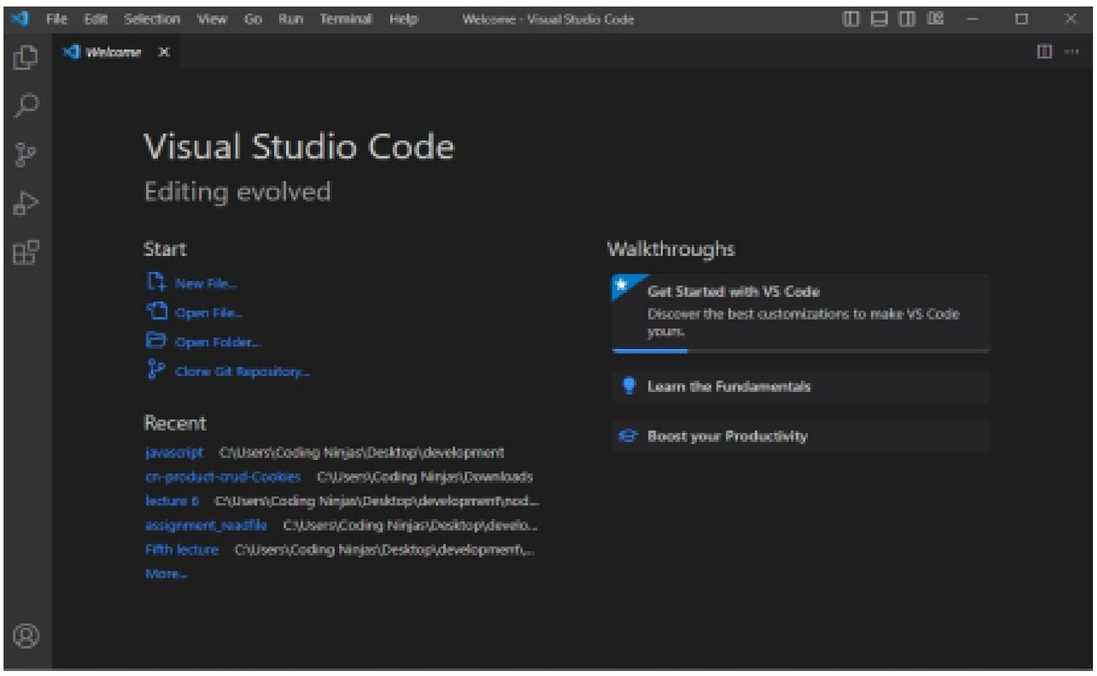
https://code.visualstudio.com/download.

You will be redirected to this page.



- Click on the "Download for Mac" button. This will start the download of the Visual Studio Code disk image file (.dmg).
- Once the download is complete, open the downloaded file (usually located in your Downloads folder). You can do this by double-clicking on the .dmg file.
- A new window will open with the Visual Studio Code application icon and a shortcut to the Applications folder. Drag the Visual Studio Code icon to the Applications folder to install it on your Mac. You can also drag the shortcut to your Dock if you want easy access to the application.
- After the installation is complete, you can open Visual Studio Code from the Applications folder. You can launch it from Spotlight Search by pressing Cmd + Space and typing "Visual Studio Code" in the search bar.

- If you are prompted with a warning that the application is from an unidentified developer, you can right-click on the Visual Studio Code icon and select "Open" to bypass the warning. You only need to do this the first time you open the application.
- Once Visual Studio Code is open, you can start customizing it to your preferences and begin working on your projects!


File
Edit
Selection
View
Go
Run
Terminal
Help
Welcome - Visual Studio Code
Welcome
Visual Studio Code
Editing evolved
Start
New File...
Open File...
Open Folder...
Clone Git Repository...
Recent
javascript C:\Users\Coding Ninjas\Desktop\development
cn-product-crud-Cookies C:\Users\Coding Ninjas\Downloads
lecture 6 C:\Users\Coding Ninjas\Desktop\development\nod...
assignment_readfile C:\Users\Coding Ninjas\Desktop\develo...
Fifth lecture C:\Users\Coding Ninjas\Desktop\development\...
More...
Walkthroughs
Get Started with VS Code
Discover the best customizations to make VS Code yours.
Learn the Fundamentals
Boost your Productivity

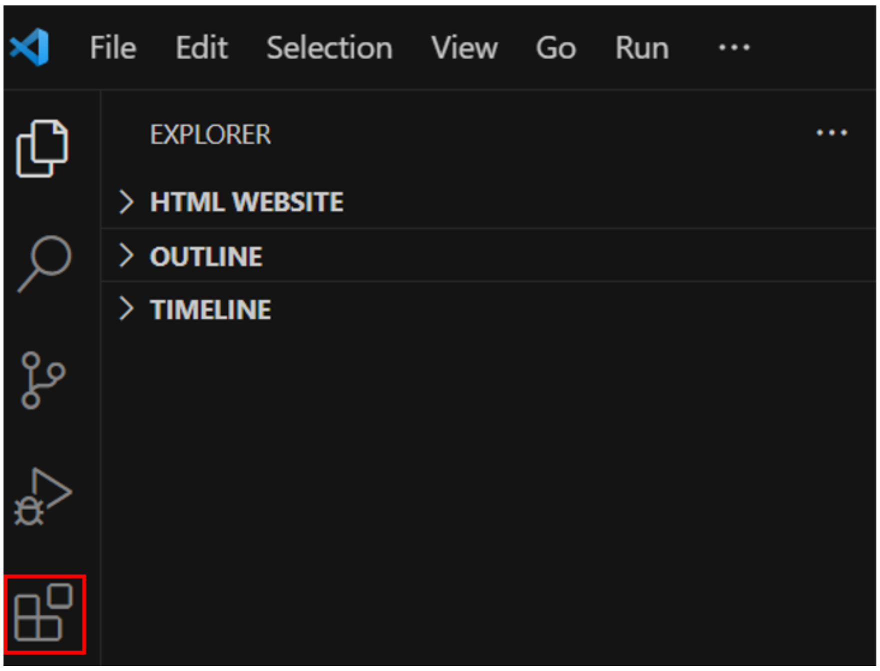
VS Code Extensions

To install VS Code extensions, follow these simple steps:

1. **Live Server:**

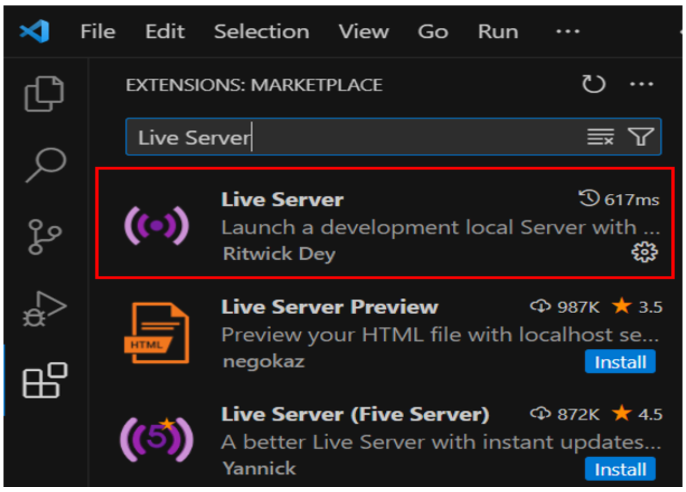
   a. Open Visual Studio Code:
      - Launch VS Code on your computer.

   b. Go to Extensions:
      - Click on the Extensions icon in the Activity Bar on the side of the window or use the shortcut Ctrl + Shift + X.



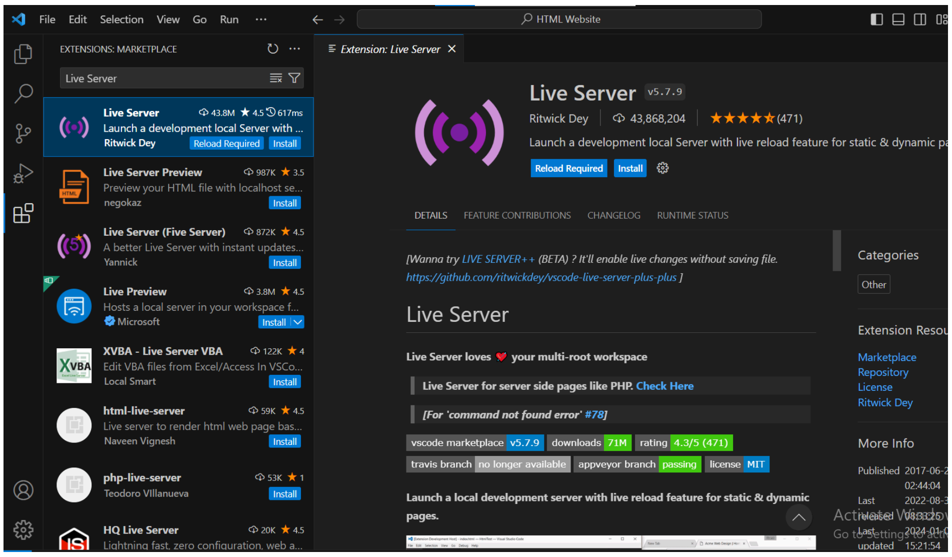
c. Search for Live Server:

- In the Extensions view, search for "Live Server" in the search bar.


File
Edit
Selection
View
Go
Run
EXTENSIONS: MARKETPLACE
Live Server
Live Server
617ms
Launch a development local Server with ...
Ritwick Dey
Live Server Preview
987K 3.5
Preview your HTML file with localhost se...
negokaz
Install
Live Server (Five Server)
872K 4.5
A better Live Server with instant updates...
Yannick
Install

d. Install Live Server:

- Find the "Live Server" extension in the search results.
- Click the "Install" button next to the extension.


File
Edit
Selection
View
Go
Run
HTML Website
EXTENSIONS: MARKETPLACE
Live Server
Live Server 43.8M 4.5 617ms
Launch a development local Server with ...
Ritwick Dey
Reload Required
Install
Live Server Preview 987K 3.5
Preview your HTML file with localhost se...
negokaz
Install
Live Server (Five Server) 872K 4.5
A better Live Server with instant updates...
Yannick
Install
Live Preview 3.8M 4.5
Hosts a local server in your workspace f...
Microsoft
Install
XVBA - Live Server VBA 122K 4
Edit VBA files from Excel/Access In VSCo...
Local Smart
Install
html-live-server 59K 4.5
Live server to render html web page bas...
Naveen Vignesh
Install
php-live-server 53K 1
Teodoro VIllanueva
Install
HQ Live Server 20K 4.5
Lightning fast, zero configuration, web a...
Extension: Live Server
Live Server v5.7.9
Ritwick Dey
43,868,204
(471)
Launch a development local Server with live reload feature for static & dynamic pa
Reload Required
Install
DETAILS
FEATURE CONTRIBUTIONS
CHANGELOG
RUNTIME STATUS
Categories
Other
Extension Resou
Marketplace
Repository
License
Ritwick Dey
More Info
Published 2017-06-2 02:44:04
Last released 2022-08-3 08:33:25
Last updated 2024-01-0 15:21:54
Activate Windo
Go to Settings to acti

[Wanna try LIVE SERVER++ (BETA) ? It'll enable live changes without saving file. https://github.com/ritwickdey/vscode-live-server-plus-plus]

Live Server

Live Server loves ❤️ your multi-root workspace

> **Live Server for server side pages like PHP. Check Here**

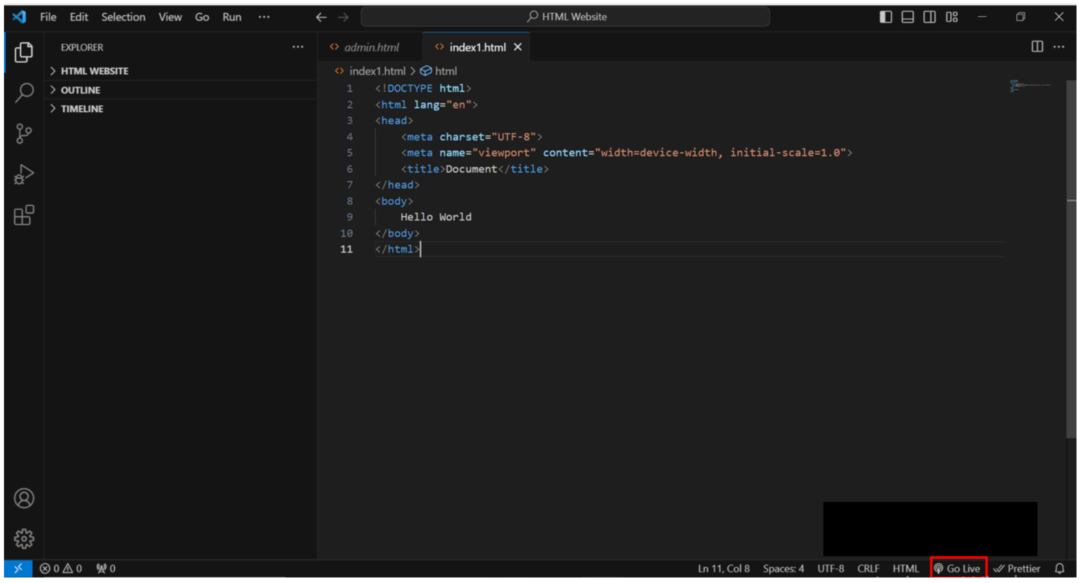
> ***[For 'command not found error' #78]***

vscode marketplace v5.7.9 | downloads 71M | rating 4.3/5 (471)
travis branch no longer available | appveyor branch passing | license MIT

Launch a local development server with live reload feature for static & dynamic pages.

e- Start Live Server:

- Open an HTML file in your project.
- Click on the “Go Live” option available down below to start the Live server.
- This will launch a live development server, and your default browser will open with the live version of your page.


File Edit Selection View Go Run ···
HTML Website
EXPLORER
HTML WEBSITE
OUTLINE
TIMELINE
admin.html
index1.html
index1.html > html
Ln 11, Col 8 Spaces: 4 UTF-8 CRLF HTML Go Live Prettier

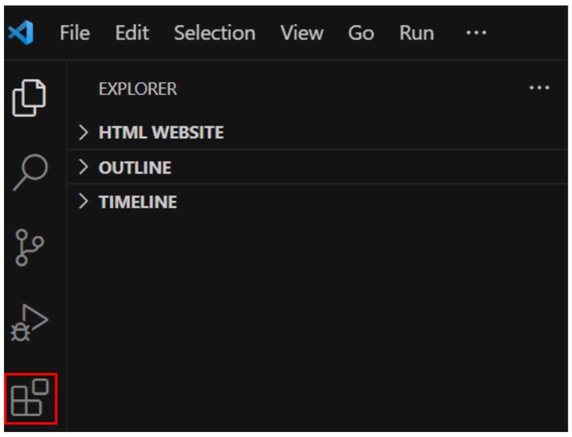
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    Hello World
</body>
</html>
```

[redacted]

2. Install Prettier Extension:

a. Go to Extensions:

- Click on the Extensions icon in the Activity Bar on the side of the window or use the shortcut Ctrl + Shift + X.



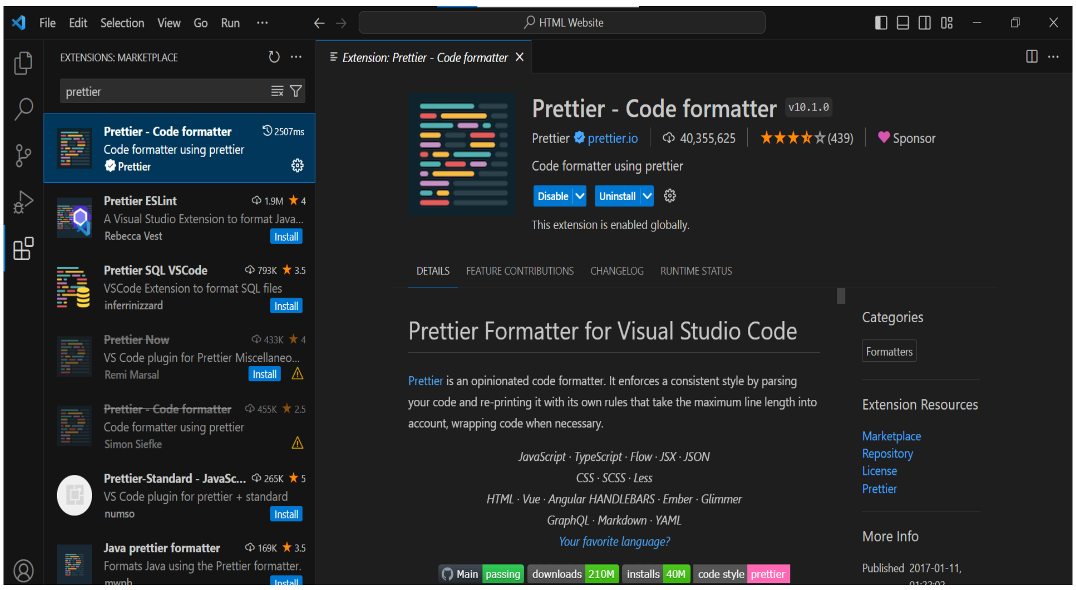
b. Search for Prettier:

- In the Extensions view, search for "Prettier" in the search bar.

c. Install Prettier:

- Find the "Prettier - Code formatter" extension in the search results.

- Click the "Install" button next to the extension.



Prettier - Code formatter 2507ms
Code formatter using prettier
Prettier

Prettier ESLint 1.9M ★ 4
A Visual Studio Extension to format Java...
Rebecca Vest

Prettier SQL VSCode 793K ★ 3.5
VSCode Extension to format SQL files
inferrinizzard

~~Prettier Now~~ 433K ★ 4
VS Code plugin for Prettier Miscellaneo...
Remi Marsal

~~Prettier - Code formatter~~ 455K ★ 2.5
Code formatter using prettier
Simon Siefke

Prettier-Standard - JavaSc... 265K ★ 5
VS Code plugin for prettier + standard
numso

Java prettier formatter 169K ★ 3.5
Formats Java using the Prettier formatter.

Prettier - Code formatter v10.1.0
Prettier prettier.io | 40,355,625 | ★★★★☆ (439) | Sponsor
Code formatter using prettier
This extension is enabled globally.

Prettier Formatter for Visual Studio Code

Prettier is an opinionated code formatter. It enforces a consistent style by parsing your code and re-printing it with its own rules that take the maximum line length into account, wrapping code when necessary.

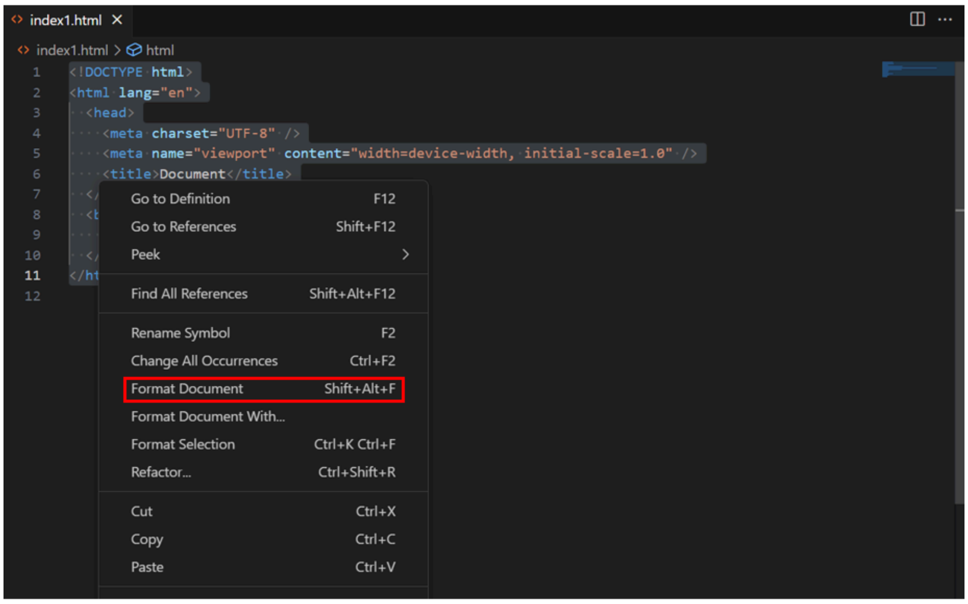
JavaScript · TypeScript · Flow · JSX · JSON
CSS · SCSS · Less
HTML · Vue · Angular HANDLEBARS · Ember · Glimmer
GraphQL · Markdown · YAML
Your favorite language?

Main passing | downloads 210M | installs 40M | code style prettier

Published 2017-01-11,

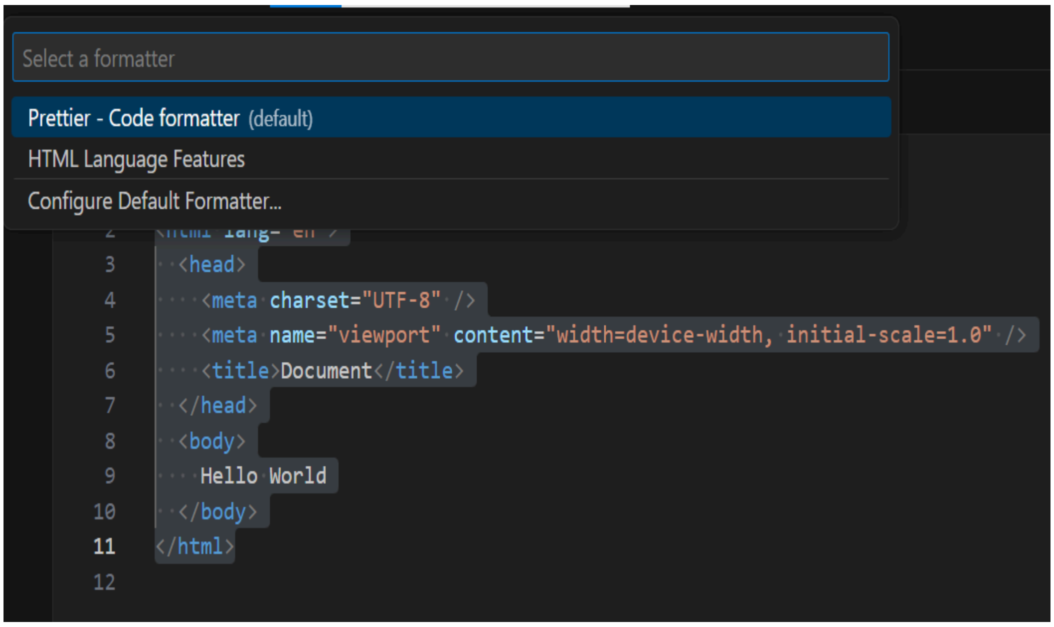
d. Format Code with Prettier:

- Open any HTML file Right-click and select "Format Document" or use the shortcut Shift + Alt + F.
- Prettier will format your code according to its rules.



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

If prompted to choose a formatter, select “Prettier”.


Select a formatter
Prettier - Code formatter (default)
HTML Language Features
Configure Default Formatter...

```
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    Hello World
  </body>
</html>
```

Now, you have both the Live Server and Prettier extensions installed and configured in Visual Studio Code.

VS Code Shortcuts

General

Ctrl+Shift+P, F1	Show Command Palette
Ctrl+P	Quick Open, Go to File…
Ctrl+Shift+N	New window/instance
Ctrl+Shift+W	Close window/instance
Ctrl+,	User Settings
Ctrl+K Ctrl+S	Keyboard Shortcuts

Basic editing

Ctrl+X	Cut line (empty selection)
Ctrl+C	Copy line (empty selection)
Alt+ ↑ / ↓	Move line up/down
Shift+Alt + ↓ / ↑	Copy line up/down
Ctrl+Shift+K	Delete line
Ctrl+Enter	Insert line below
Ctrl+Shift+Enter	Insert line above
Ctrl+Shift+	Jump to matching bracket
Ctrl+] / [	Indent/outdent line
Home / End	Go to beginning/end of line
Ctrl+Home	Go to beginning of file
Ctrl+End	Go to end of file
Ctrl+↑ / ↓	Scroll line up/down
Alt+PgUp / PgDn	Scroll page up/down
Ctrl+Shift+[	Fold (collapse) region
Ctrl+Shift+]	Unfold (uncollapse) region

Ctrl+K Ctrl+[	Fold (collapse) all subregions
Ctrl+K Ctrl+]	Unfold (uncollapse) all subregions
Ctrl+K Ctrl+0	Fold (collapse) all regions
Ctrl+K Ctrl+J	Unfold (uncollapse) all regions
Ctrl+K Ctrl+C	Add line comment
Ctrl+K Ctrl+U	Remove line comment
Ctrl+/	Toggle line comment
Shift+Alt+A	Toggle block comment
Alt+Z	Toggle word wrap

Navigation

Ctrl+T	Show all Symbols
Ctrl+G	Go to Line...
Ctrl+P	Go to File...
Ctrl+Shift+O	Go to Symbol...
Ctrl+Shift+M	Show Problems panel
F8	Go to next error or warning
Shift+F8	Go to previous error or warning
Ctrl+Shift+Tab	Navigate editor group history
Alt+ ← / →	Go back / forward

Search and replace

Ctrl+F	Find
Ctrl+H	Replace
F3 / Shift+F3	Find next/previous
Alt+Enter	Select all occurences of Find match
Ctrl+D	Add selection to next Find match
Ctrl+K Ctrl+D	Move last selection to next Find match
Alt+C / R / W	Toggle case-sensitive / regex / whole word

Multi-cursor and selection

Alt+Click	Insert cursor
Ctrl+Alt+ ↑ / ↓	Insert cursor above / below
Ctrl+U	Undo last cursor operation
Shift+Alt+I	Insert cursor at end of each line selected
Ctrl+L	Select current line
Ctrl+Shift+L	Select all occurrences of current selection
Ctrl+F2	Select all occurrences of current word
Shift+Alt+→	Expand selection
Shift+Alt+←	Shrink selection
Shift+Alt + (drag mouse)	Column (box) selection
Ctrl+Shift+Alt + (arrow key)	Column (box) selection
Ctrl+Shift+Alt +PgUp/PgDn	Column (box) selection page up/down

Rich languages editing

Ctrl+Space, Ctrl+I	Trigger suggestion
Ctrl+Shift+Space	Trigger parameter hints
Shift+Alt+F	Format document
Ctrl+K Ctrl+F	Format selection
F12	Go to Definition
Alt+F12	Peek Definition
Ctrl+K F12	Open Definition to the side
Ctrl+.	Quick Fix
Shift+F12	Show References
F2	Rename Symbol
Ctrl+K Ctrl+X	Trim trailing whitespace
Ctrl+K M	Change file language

Editor management

Ctrl+F4, Ctrl+W	Close editor
Ctrl+K F	Close folder
Ctrl+	Split editor
Ctrl+ 1 / 2 / 3	Focus into 1st, 2nd or 3rd editor group
Ctrl+K Ctrl+ ←/→	Focus into previous/next editor group
Ctrl+Shift+PgUp / PgDn	Move editor left/right
Ctrl+K ← / →	Move active editor group

File management

Ctrl+N	New File
Ctrl+O	Open File...
Ctrl+S	Save
Ctrl+Shift+S	Save As...
Ctrl+K S	Save All
Ctrl+F4	Close
Ctrl+K Ctrl+W	Close All
Ctrl+Shift+T	Reopen closed editor
Ctrl+K Enter	Keep preview mode editor open
Ctrl+Tab	Open next
Ctrl+Shift+Tab	Open previous
Ctrl+K P	Copy path of active file
Ctrl+K R	Reveal active file in Explorer
Ctrl+K O	Show active file in new window/instance

Display

F11	Toggle full screen
Shift+Alt+0	Toggle editor layout (horizontal/vertical)
Ctrl+ = / -	Zoom in/out
Ctrl+B	Toggle Sidebar visibility
Ctrl+Shift+E	Show Explorer / Toggle focus
Ctrl+Shift+F	Show Search
Ctrl+Shift+G	Show Source Control
Ctrl+Shift+D	Show Debug
Ctrl+Shift+X	Show Extensions
Ctrl+Shift+H	Replace in files
Ctrl+Shift+J	Toggle Search details
Ctrl+Shift+U	Show Output panel
Ctrl+Shift+V	Open Markdown preview
Ctrl+K V	Open Markdown preview to the side
Ctrl+K Z	Zen Mode (Esc Esc to exit)

Debug

F9	Toggle breakpoint
F5	Start/Continue
Shift+F5	Stop
F11 / Shift+F11	Step into/out
F10	Step over
Ctrl+K Ctrl+I	Show hover

Integrated terminal

Ctrl+`	Show integrated terminal
Ctrl+Shift+`	Create new terminal
Ctrl+C	Copy selection
Ctrl+V	Paste into active terminal
Ctrl+↑ / ↓	Scroll up/down
Shift+PgUp / PgDn	Scroll page up/down
Ctrl+Home / End	Scroll to top/bottom

Getting Started with Basics

Welcome to the web development course!

- You will learn all aspects of front-end web development. Starting from the very basics to the advanced.
- The best part of this journey is you'll get hands-on experience as you'll be doing some projects along the course, which will reinforce and strengthen your skills.
- We'll learn how to structure our webpage, design it and add interactivity to it. HTML, CSS, and JS

This course is divided into two modules:

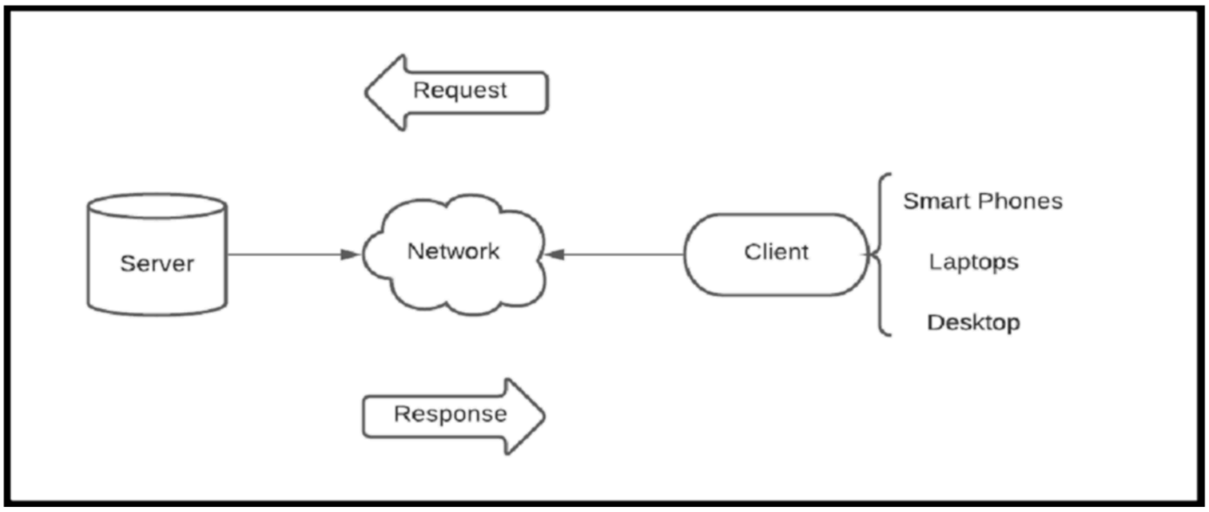
1. **Front-end Web Development** – Here you will understand the principles of user interface design and user experience.
   You'll construct new, appealing, responsive websites with HTML, CSS, JavaScript, jQuery, and Bootstrap. This part of the course focuses on core concepts and lays a strong programming foundation.

2. **Back-end Web Development** – In this section, you'll learn how to build and maintain the technologies that power the administrative side of websites.
   It's a deep dive that will help you understand a web application's operation with databases to evolve simple, static websites into dynamic, database-driven web applications. You'll go through a wide range of topics like NodeJS, event handling, file uploads, social authentication, deployment, etc.

Client-Server Architecture

Like in the outside world, the Client means any person or organization that uses a particular service. Similarly, in the world of computer networks, a client is a host (any computer or a device that is connected to a computer network) that receives some service from certain service providers (Servers).

As the word suggests, a Server is a remote computer that provides some service/information to its clients. So, a client requests some information, and the Server responds by serving it to the Client. Servers store and retrieve the information from the databases. The client can be any device like a mobile phone, laptop, desktop, etc.



Front end and Back end

There are two faces of a website: front-end and back-end. These are the most popular andcrucial topics in web development.

Front-end:

- The front-end is the part of a website with which the users interact directly. It's also called the ‘client side’ of a website.
- It includes everything the users see: text, colours, styles, images, buttons, menus, etc.
  HTML, CSS, and JavaScript are generally used for front-end development.
  Front-end developers build the front-end part of a website.
  Being a front-end developer, you will have to ensure that your websites look good on all devices for a smooth user experience.

- There are many front-end frameworks and libraries which developers use, like React.js, jQuery, Angular.js, etc.

Backend:

- The back end is the server side of a website.
- A back-end web developer uses it to store and manage data and make sure that the client side of the website works without any issues.
- The website's users or clients cannot see and interact with the back end of a website.The backend part of the software is abstracted from the users.
- Every line of code you write for the backend will be used on the frontend side anyways.
- In simple words, we can say that everything that happens in the background can be credited to the backend.
- Like frontend, there are many backend libraries that developers use like Express, Rails, Django, etc.

Static Vs Dynamic Websites:

- A static website is a website that displays the same content to each user, and it’s usually written using simple HTML and CSS.
- Whereas a dynamic website displays different content to different users depending on user data and preferences and provides user interaction. In addition to HTML, making dynamic websites require advanced technologies, frameworks, and databases to store user data.
- Usually, when static websites run on a browser, the content shown is the same for every person accessing the website. An excellent example of a static website would be a simple Blog page.
- A dynamic website on the other hand is more functional. Here the users are required to interact with the website as user interaction plays a big role in

dynamic websites.

- As the website is dynamic, the content shown on the website will vary according to specific users. The information shown on the page will not be the same as Facebook, where the content is relevant and related to the specific user.

What happens when you visit a website?

Every website has its IP address, using which we can access a particular website. But, as humans, are not good at remembering numbers, we use domain names, which a user-friendly way to access a website's IP address. As the user enters the URL of the website, the browser sends a request for the domain name of the website. The DNS (Domain Name Server), which is like a phonebook of the internet, connects domain names with IP addresses and responds with the web server's IP address. Then the browser sends an HTTP request to the web server's IP (which is provided by DNS). The Server sends over the website's necessary files, which are then correctly rendered by the browser and displayed as a website.

What is DNS?

- The Domain Name System (DNS) is the phonebook of the Internet. E.g. when you want to call your friend, search for the friend's name in the phone directory and call them, but in an actual call on their mobile number. Similarly, Domain Name System (DNS) does this same process but for domain names and IP addresses.

- Humans access information online through domain names, like google.com. Web browsers interact through Internet Protocol (IP) addresses. DNS translates domain names to IP addresses so browsers can load Internet resources.

What happens when you enter google.com in the web browser?

Steps:

- Initial Typing: When you start typing google.com in the browser, the browser will start looking for your browser cache first if the content is fresh and present in the cache display the same.

- URL Parsing: If not, the browser checks if the IP address of the URL is present in the browser cache if not then requests the OS to do a DNS lookup using UDP to get the corresponding IP address of the URL from the DNS server to establish a new TCP connection.

- A new TCP connection is set between the browser and server.

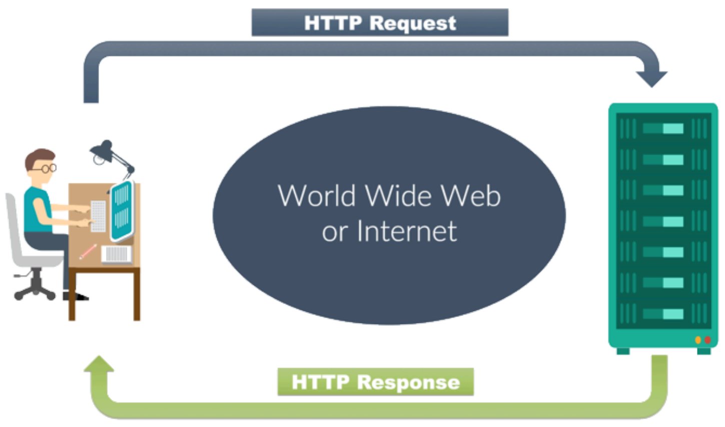
- An HTTP request is sent to the server using the TCP connection.

- The web servers running on the Servers handle the incoming HTTP request and send HTTP responses.

- The browser processes the HTTP response sent by the server and may close the TCP connection or reuse the same for future requests.

- If the response data is cacheable then browsers cache the same.

- Browser decodes the response and renders the content.



What is HTTP?

- HTTP is a protocol that allows the fetching of resources, such as HTML documents.
- It is the foundation of any data exchange on the Web, and it is a client-server protocol. Your server will receive requests from the browser that follows HTTP.
- It then responds with an HTTP response that all browsers can parse.

What is a web browser, and how does it work?

The web browser is an application that provides access to the web server, sends a network request to the URL, obtains resources, and interactively represents them.

E.g., Google Chrome, Firefox, Safari, Internet Explorer and Opera.

Conclusion:

In conclusion, embarking on the web development course provides a comprehensive journey through the fundamentals and advanced aspects of front-end and back-end web development. The course structure, divided into two modules, ensures a well-rounded learning experience. In the front-end module, participants dive into user interface design principles, honing their skills in creating visually appealing and responsive websites using HTML, CSS, JavaScript, jQuery, and Bootstrap. The subsequent back-end module delves into the intricacies of web application functionality, covering technologies such as NodeJS, event handling, file uploads, social authentication, deployment, and more. The hands-on approach, involving project work, promises to reinforce and solidify the acquired skills, making this course a valuable resource for individuals looking to build a strong foundation in web development.

Exploring essential concepts like client-server architecture, the distinction between front-end and back-end development, and the contrast between static and dynamic websites, the course sets the stage for a holistic understanding of web development. Participants gain insights into the client-server relationship, understanding how browsers interact with servers through protocols like HTTP and how DNS plays a pivotal role in translating user-friendly domain names into IP addresses. The notes on the client-server architecture highlight the integral role of servers in serving information to clients, shaping the foundation of the internet. Overall, the course provides a comprehensive introduction to the world of web development, demystifying key concepts and empowering learners to navigate the dynamic landscape of creating interactive and engaging web applications.

References:

- https://www.cloudflare.com/en-gb/learning/dns/what-is-dns/
- https://www.cloudflare.com/en-gb/learning/ddos/glossary/hypertext-transfer-protocol-http/
- https://developer.mozilla.org/en-US/docs/Web/Performance/How_browsers_work

Getting Started with Basics

History of Web

Tim Berners-Lee created HTML in late 1991 but didn't release it officially. He published it later in 1995 as HTML 2.0. The idea behind creating the Web was to create a service that helps everyone communicate, share, and receive information. Then came HTML 4, which served as a major version of HTML. HTML has evolved very much and received various updates since its creation. With each version, the creation of web pages got
easier and more stylish.

HTML is generally used to design the basic **structure** of the web pages, which will be improved using other technologies like CSS and JavaScript. **CSS** controls the **looks**, feel, layout and formatting, whereas **JavaScript** is used to control different elements' **behaviour** on the page.

What is HTML:

HTML, or Hypertext Markup Language, serves as the backbone of the World Wide Web, providing a standardized method for creating and structuring content on web pages. It is a markup language that uses a system of tags to define and organize various elements, such as headings, paragraphs, links, images, forms, and more. These tags instruct web browsers on how to present and display content to users. HTML plays a crucial role in web development, facilitating the creation of visually appealing, well-organized, and interactive websites.

HTML uses a system of tags, which are enclosed in angle brackets (< >), to define different components and attributes of a webpage. At its core, HTML is a markup language, not a programming language, as it doesn't possess the logical and computational capabilities of languages like JavaScript or Python. Instead, HTML focuses on defining the structure and semantics of a document. Each HTML tag has a specific purpose, and by combining these tags, developers can create a hierarchy of elements that form the structure of a webpage.

HTML lays the foundation for web development, working in tandem with cascading style sheets (CSS) for styling and presentation, and JavaScript for interactivity. As the cornerstone of the web, HTML empowers developers to create structured, accessible, and user-friendly content that can be seamlessly viewed across various devices and platforms. Understanding HTML is a fundamental step for anyone entering the field of web development.

History of HTML:

The history of HTML can be traced back to 1989 when Sir Tim Berners-Lee, a British computer scientist, envisioned the World Wide Web. The first version, HTML 1.0, was introduced in 1991, laying the foundation with basic tags for text formatting. HTML 2.0, released in 1995, expanded its capabilities by introducing features like tables. HTML 3.2, in 1997, further evolved by adding support for scripting languages, paving the way for more dynamic web pages. HTML 4.01, arriving in 1999, marked a significant milestone with advanced features such as stylesheets and scripting capabilities, enhancing the overall web development experience.

The subsequent introduction of XHTML in 2000 aimed to align HTML with XML standards, emphasizing the importance of a more structured and rigorous syntax. However, HTML5, finalized in 2014, emerged as a game-changer. HTML5 not only addressed the shortcomings of its predecessors but also introduced groundbreaking features, including native support for audio and video, improved semantics, and enhanced support for creating responsive and interactive websites. HTML5 has since become the standard for modern web development, shaping the way websites are built and experienced across a variety of devices.

Versions of HTML:

- HTML 1.0 (1991): The inaugural version, introducing basic text formatting tags.
- HTML 2.0 (1995): Expanded features to include tables, enhancing the layout possibilities for developers.
- HTML 3.2 (1997): Added support for scripting languages, enabling the development of more interactive web pages.
- HTML 4.01 (1999): Significantly advanced web development with features like stylesheets and scripting capabilities.

- XHTML (2000): Aimed to align HTML with XML standards, emphasizing a stricter and more structured syntax.
- HTML5 (2014): The latest and most comprehensive version, revolutionizing web

  development with native multimedia support, improved semantics, and enhanced capabilities for creating modern and responsive websites. HTML5 has become the industry standard, shaping the landscape of contemporary web development practices.

HTML Boilerplate:

HTML boilerplate refers to a standardized and basic structure of HTML code that serves as a starting point for building a web page. It is essentially a template that includes the essential elements required for every HTML document. Creating an HTML boilerplate is a common practice among web developers to ensure consistency, save time, and adhere to best practices. Here's a breakdown of the key components of an HTML boilerplate:

Document Type Declaration (<!DOCTYPE html>): This declaration specifies the version of HTML that the document follows. In modern web development, <!DOCTYPE html> is used to indicate HTML5.

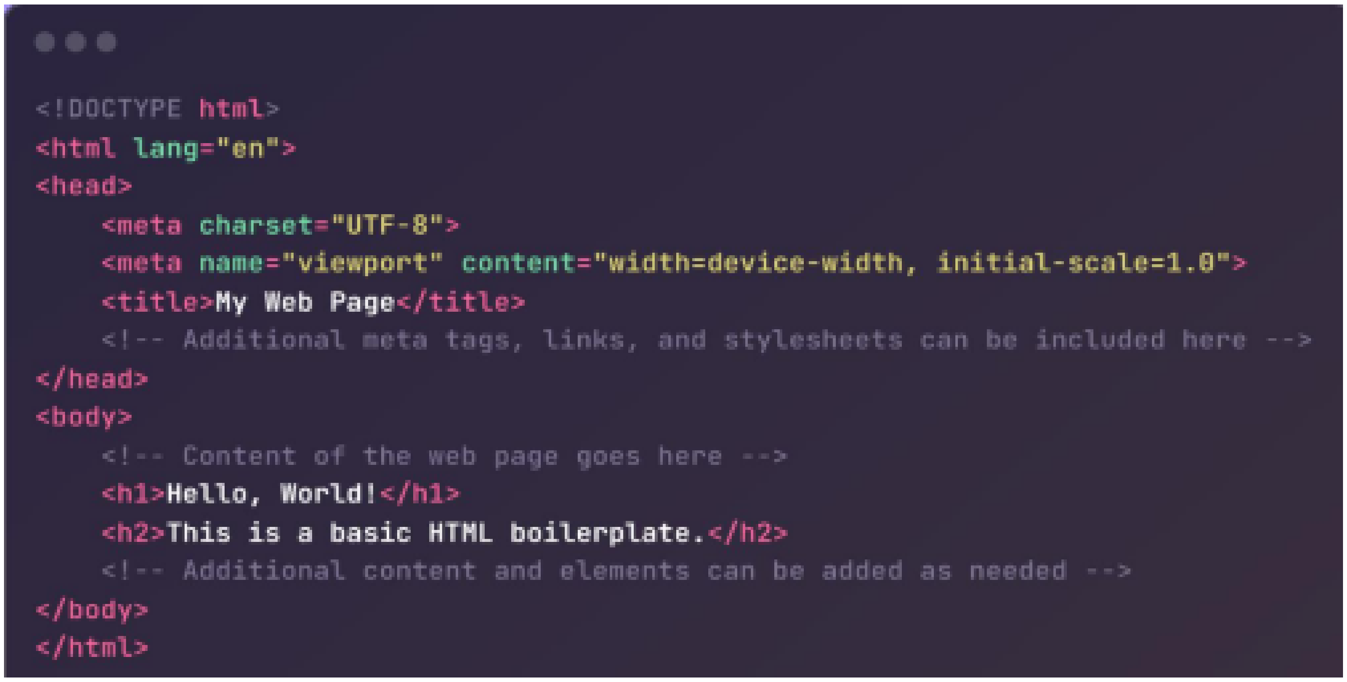
HTML Opening and Closing Tags (<html>...</html>): The root element of an HTML document. All other elements are nested within these tags.

Head Section (<head>...</head>): This section contains meta-information about the HTML document, such as the title of the page, links to external stylesheets or scripts, and other metadata.

Meta Charset Tag (<meta charset="utf-8">): This meta tag declares the character encoding for the document, ensuring proper interpretation of the text.

Title Tag (<title>...</title>): This tag defines the title of the HTML document, which is displayed in the browser's title bar or tab.

Body Section (<body>...</body>): The main content of the HTML document resides within these tags. Elements like headings, paragraphs, images, links, and other visible content are placed here.



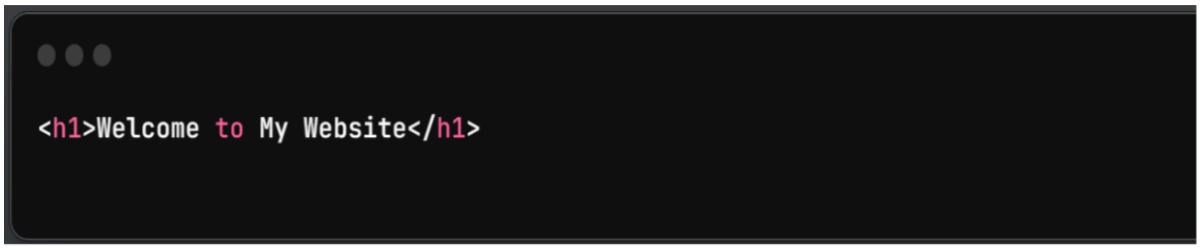
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>My Web Page</title>
    <!-- Additional meta tags, links, and stylesheets can be included here -->
</head>
<body>
    <!-- Content of the web page goes here -->
    <h1>Hello, World!</h1>
    <h2>This is a basic HTML boilerplate.</h2>
    <!-- Additional content and elements can be added as needed -->
</body>
</html>
```

H1 and H2 Tags:

HTML <h1> and <h2> tags are part of the heading elements used to define headings or titles within a webpage. These tags play a crucial role in structuring content and indicating the hierarchy of information.

<h1> Tag: -

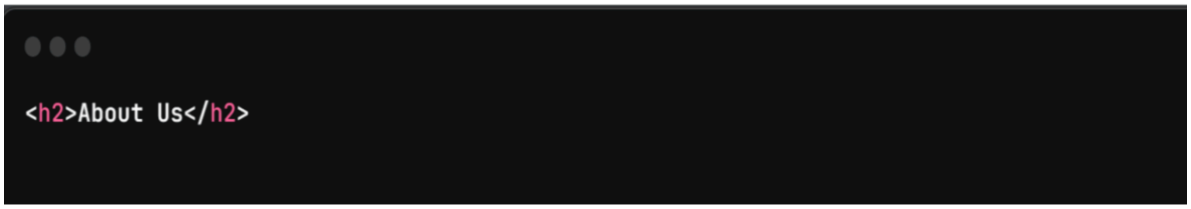
- The <h1> tag is the highest-level heading in HTML and signifies the main heading or title of a webpage.
- It carries significant semantic weight, representing the primary topic or purpose of the document.
- There should be only one <h1> tag per page, reflecting the most important heading on that page.



```
<h1>Welcome to My Website</h1>
```

<h2> Tag:

- The <h2> tag is a lower-level heading that comes after <h1> in the hierarchy.
- It is used to create subheadings or section titles within the content of the webpage.
- Multiple <h2> tags can be employed to organize and structure the page effectively.

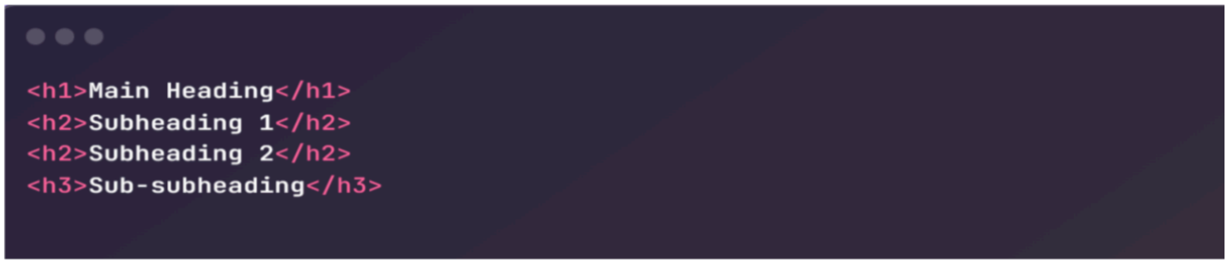


```
<h2>About Us</h2>
```

Heading Hierarchy:

- HTML heading tags follow a hierarchy from <h1> to <h6, where <h1> is the highest and <h6> is the lowest.
- It is crucial to maintain a logical and sequential order in using heading tags to ensure a proper document structure.

Example of Hierarchy:



```
<h1>Main Heading</h1>
<h2>Subheading 1</h2>
<h2>Subheading 2</h2>
<h3>Sub-subheading</h3>
```

Semantic Meaning:

Heading tags in HTML provide semantic meaning to content, aiding both accessibility and search engine optimization.

They assist screen readers in understanding the structure of a document, making it more accessible to users with disabilities.

Search engines use heading tags to comprehend the content hierarchy, influencing page ranking.

We will be going to learn in detail about these tags in the next Lecture.

Conclusion:

In summary, delving into the fundamentals of web development reveals a rich history of HTML, initiated by Tim Berners-Lee in the early 1990s. From the foundational HTML 1.0 to the transformative HTML5, each version has contributed to simplifying web page creation and enhancing stylistic possibilities. HTML, as the backbone of the World Wide Web, collaborates with CSS and JavaScript to structure, style, and add interactivity to web content. The journey through HTML's evolution and its role in web development is complemented by exploring various resources for in-depth understanding.

Moving forward, the exploration extends to HTML boilerplate, a standardized template streamlining the creation of web pages. The HTML boilerplate comprises essential elements like the document type declaration, HTML tags, head and body sections, and meta information. This structured foundation aligns with best practices in web development, ensuring consistency and efficiency. The narrative then shifts to the significance of heading elements (<h1> and <h2>) in HTML, emphasizing their hierarchical structure, semantic meaning, and pivotal role in accessibility and SEO. The conclusion underscores the promise of continued learning in upcoming lectures, solidifying the foundation for comprehensive understanding and proficiency in web development.

References:

- https://webfoundation.org/about/vision/history-of-the-web/
- https://www.sitesbay.com/css/css-history
- https://auth0.com/blog/a-brief-history-of-javascript/

HTML Boilerplate

What is a boilerplate?

Boilerplate code in HTML is required to set up the basic web page structure. This
web page structure is the same for all web pages in general.

```
<html>
  <head>
    <title></title>
  </head>
  <body>
     Body Content
  </body>
</html>
```

Automatic boilerplate generation in different Text Editors

Text editors can produce HTML boilerplate for us. The basic boilerplate generated (as shown above) will always stay the same across all text meditors.

Although different text editors may produce some extra code as well apart from the basic boilerplate as shown above. Below is the boilerplate which appears on

VS 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>

</body>
</html>
```

1. **<!DOCTYPE html>:** This declaration defines the document type and version of HTML being used. In this case, it's HTML5.

2. **<html lang="en">:** This opening tag indicates the beginning of the HTML document. The lang attribute is set to "en" (English), indicating the primary language of the document. It helps browsers and search engines understand the language of the content.

3. **<head>:** This section contains meta-information about the HTML documents, such as character set, viewport settings, and title.

   - **<meta charset="UTF-8">:** Specifies the character encoding for the document. UTF-8 is a widely used character encoding that supports a broad range of characters.

   - **<meta name="viewport"content="width=device-width, initial-scale=1.0">:** Configures the viewport settings for responsive web design. It ensures that the width of the page is set to the device's screen width, and the initial zoom level is set to 1.0.

   - **<title>Document</title>:** Sets the title of the HTML document, which is typically displayed in the browser's title bar or tab.

4. **<body>:** This is the main content of the HTML document. You would place your actual webpage content within the opening and closing <body> tag.

Conclusion

In summary, the provided HTML boilerplate sets up the basic structure of an HTML5 document, specifying the document type, language, character encoding, viewport settings, and title. The actual content of the webpage would be placed within the <body> section.

References

- https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/The_head_metadata_in_HTML

Text Formatting Tags

Text formatting tags in HTML control the appearance of text on a web page. These tags allow you to apply various styles and effects to the text content. Here are some commonly used text formatting tags in HTML:

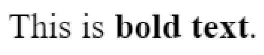
1. **<b>: Defines bold text.**

   The <b> tag makes the enclosed text bold without implying any specific importance or emphasis. It is a presentational tag and doesn't convey any additional meaning to the text. It's often better to use <strong> for semantic importance.

   Example:

   ```
   <p>This is <b>bold text</b>.</p>
   ```

   Output:

   This is **bold text**.

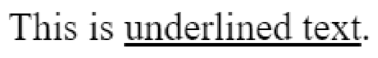
2. **<u>: Defines underlined text.**

   The <u> tag underlines the enclosed text. However, it's worth noting that underlining is not commonly used for general text on the web because underlined text is often associated with hyperlinks. Instead, CSS is usually employed to style text.

   Example:

   ```
   <p>This is <u>underlined text</u>.</p>
   ```

   Output:

   This is underlined text.

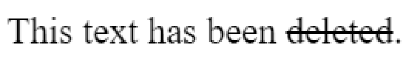
3. <del>: Defines text that has been deleted from a document.

The <del> tag is used to represent text that has been deleted or removed from the document. Browsers typically render this text with a strikethrough.

Example:

```
<p>This text has been <del>deleted</del>.</p>
```

Output:



This text has been ~~deleted~~.

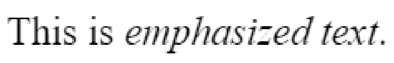
4. <em>: Defines emphasised text

The <em> tag emphasises text, typically rendering it in italics. The enclosed the text should be emphasised, conveying importance or stress.

Example:

```
<p>This is <em>emphasized text</em>.</p>
```

Output:



This is *emphasized text*.

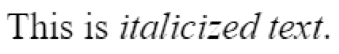
5. <i>: Defines a part of a text in an alternate voice or mood.

The <i> tag represents text in an alternate voice or mood. It is often rendered in italics, but like <b>, it doesn't provide any semantic meaning to the enclosed text.

Example:

```
<p>This is <i>italicized text</i>.</p>
```

Output:


This is *italicized text.*

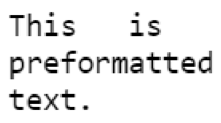
6. <pre>: Defines preformatted text

The <pre> tag defines preformatted text. Text within a <pre> element is typically displayed in a fixed-width font, and whitespace is preserved, including line breaks and spaces.

Example:

```
<pre>
This  is
preformatted
text.
</pre>
```

Output:


```
This    is
preformatted
text.
```

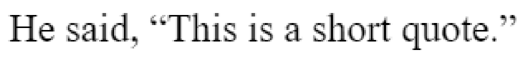
7. <q>: Defines a short quotation

The <q> tag defines a short inline quotation. Browsers typically surround the text with double quotation marks.

Example:

```
<p>He said, <q>This is a short quote.</q></p>
```

Output:


He said, “This is a short quote.”

8. <strong>: Defines important text

The <strong> tag defines text with strong importance. It is often rendered as bold, but its primary purpose is to convey semantic meaning, indicating that the text is of particular significance.
Example:

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

Output:

This is **important text**.

9. <sub>: Defines subscripted text

The <sub> tag defines subscripted text. It is commonly used for chemical formulas, mathematical expressions, or footnotes.

Example:

```html
<p>This is H<sub>2</sub>O (water).</p>
```

Output:

This is H_2O (water).

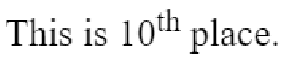
10. <sup>: Defines superscripted text

The <sup> tag is used to define superscripted text. It is commonly used for footnotes, exponents, and other instances where text should appear above the baseline.

Example:

```html
<p>This is 10<sup>th</sup> place.</p>
```

Output:



This is 10th place.

CONCLUSION:

In conclusion, the text formatting tags in HTML serve the purpose of structuring and styling text content on a web page. Each tag has its specific function and visual effect, contributing to the overall presentation of the information. Using these tags thoughtfully is crucial, considering their presentational aspects and semantic meaning.

The <b>, <i>, <u>, <strong>, and <em> tags offer ways to emphasise or highlight text, with <strong> and <em> carrying semantic importance. The <sub> and <sup> tags provide options for subscript and superscript text, useful in scientific and mathematical contexts. The <del> and <ins> tags indicate deleted and inserted text, respectively, providing a way to show document revisions. The <q> tag is handy for marking short inline quotations, while the <pre> tag preserves whitespace for preformatted text.

While these tags contribute to text formatting, it's advisable to use semantic tags like <strong>, <em>, <sub>, and <sup> for better accessibility and search engine optimisation. Additionally, CSS is commonly employed for more extensive styling, separating presentation from content.

REFERENCES:

Explore more about these tags by going through the links below:

1. https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/HTML_text_fundamentals#emphasis_and_importance
2. https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/Advanced_text_formatting#quotations

HTML ENTITIES

Entities are special characters represented by their entity names or entity numbers. These entities are used to display reserved characters in HTML without causing parsing errors.

Here are some commonly used HTML entities:

1. &lt; - Less Than (<):

This entity is used to represent the less than sign ("<") in HTML.
It is primarily used to begin an HTML tag and is essential for the correct structure of HTML documents.
Example:

```
<p>This is an example of using the &lt; symbol in HTML.</p>
```

Output:

```
This is an example of using the < symbol in HTML.
```

2. &gt; - Greater Than (>):

Represents the greater than sign (">") in HTML.
It is used to end an HTML tag and plays a crucial role in defining the structure of HTML elements.
Example:

```
<a href="https://www.example.com">Visit our website &gt;&gt;</a>
```

Output:

```
Visit our website >>
```

3. &amp; - Ampersand (&):

Represents the ampersand symbol ("&") in HTML.
Used to escape the character "&" to ensure it is not confused with the beginning of an entity.
Example:

```
<p>AT&amp;T is a well-known telecommunications company.</p>
```

Output:

```
AT&T is a well-known telecommunications company.
```

4. &quot; - Double Quotation Mark ("):

Represents the double quotation mark (") in HTML.
Used to include attribute values within double quotes in HTML tags.
Example:

```
<input type="text" placeholder="Enter your &quot;Name&quot;">
```

Output:

```
Input field with placeholder: Enter your "Name"
```

5. &apos; - Single Quotation Mark (Apostrophe) ('):

Represents the single quotation mark (apostrophe) in HTML.
Though HTML typically uses double quotes for attribute values, this entity can be used for consistency or when attributes are written inside single quotes.
Example:

```
<p title='This is an example with a single &apos; quote'>Hover over me</p>
```

Output:

```
Hover over me (with a tooltip: This is an example with a single ' quote)
```

6. &nbsp; - Non-Breaking Space:

Represents a non-breaking space in HTML.
It prevents browsers from collapsing consecutive spaces into a single space, ensuring that multiple spaces are rendered as intended.
Example:

```
<p>This&nbsp;word&nbsp;won't&nbsp;break&nbsp;apart.</p>
```

Output:

```
This word won't break apart.
```

7. &copy; - Copyright Symbol (©):

Represents the copyright symbol in HTML.
It is commonly used to indicate copyright ownership.
Example:

```
<p>&copy; 2023 My Company. All rights reserved.</p>
```

Output:

```
(c) 2023 My Company. All rights reserved.
```

8. &reg; - Registered Trademark Symbol (®):

Represents the registered trademark symbol in HTML.
It is used to indicate that a trademark is officially registered with the appropriate authorities.
Example:

```
<p>Our product is a registered trademark&reg; of XYZ Corporation.</p>
```

Output:

```
Our product is a registered trademark® of XYZ Corporation.
```

9. &trade; - Trademark Symbol (™):

Represents the trademark symbol in HTML.
It is used to indicate that a particular word, symbol, or logo is a trademark.
Example:

```
<p>The brand name&trade; signifies quality and innovation.</p>
```

Output:

```
The brand name™ signifies quality and innovation.
```

CONCLUSION:

In conclusion, HTML entities play a crucial role in ensuring the proper representation of special characters and symbols within HTML documents. They are used to escape reserved characters that have specific meanings in HTML, such as <, >, and &, allowing them to be displayed correctly without disrupting the structure of the document.

HTML entities are used to represent characters with special meanings in HTML, preventing them from being interpreted as part of the HTML structure.

In practical terms, understanding and correctly using HTML entities contribute to the proper display and interpretation of content in web development, ensuring that reserved characters and symbols are handled appropriately within HTML documents.

REFERENCES:

- https://developer.mozilla.org/en-US/docs/Glossary/Entity

FUNDAMENTALS OF HTML

Comments in HTML

Generally, developers use comments to **explain their code to other developers** or mark something important that needs editing. Comments are **ignored** by the browser and **hence won't be seen on the webpage or other content.**

You can write a comment by placing the comment text between **<! > tags.**

For example:

```
<!-- This is a comment -->
```

Comments can't be nested, which means a comment can't be put inside another comment.

TAGS

Tags are used to represent HTML elements. These can be seen as keywords that define how a web browser will format and display the website content.

- Tags **define** all document elements, i.e. they give meaning to the plain HTML text.
- Two characters < and > surround HTML tags (They are called **angle brackets)**.
- The name of the tag can either begin with an **alphabet** or an **underscore (_)**.
- The element contents are displayed between the **start** and **end tags**
- Tags that have an **opening** and **closing** can have **any number of tags**

within them.

- The **<H1>** and **<h1>** tags in HTML have the **same meaning,** i.e. tags are **not case-sensitive.**
- The HTML tags are **usually available in pairs,** i.e. opening and closing (it's the same, with the tag name '/' at the beginning) tag.

Eg: <html> and </html> is a tag that comes in pairs and <hr> does not have a closing tag.

	Tag	**Description**
1	**<!DOCTYPE html>**	Specifies that HTML version 5 is used to create the web page
2	**<html> </html>**	Root container for all other HTML elements of the web page (including the head tag)
3	**<head> </head>**	The <head> element is a container for metadata (data about data) Metadata typically defines the document title, character set, styles, scripts, and other meta information.
4	**<title> </title>**	Provides the title of the document and is displayed in the tab in the browser
5	**<body></body>**	Contains all of the elements visible on the web page

NOTE: *There are also "self-closing" tags, whereby a br tag, for eg., will look like "**<br/>**" instead of simply "**<br>***

EXTRA:

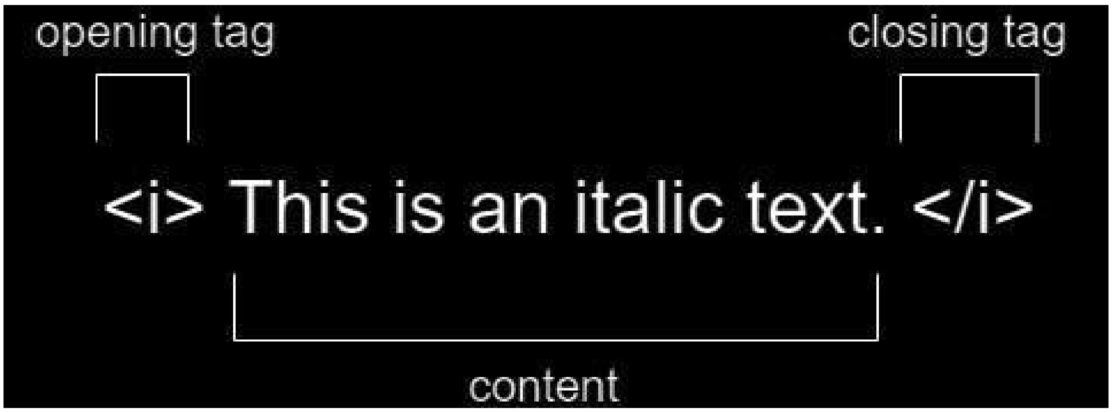
To get the list of all valid tags in HTML5, visit:

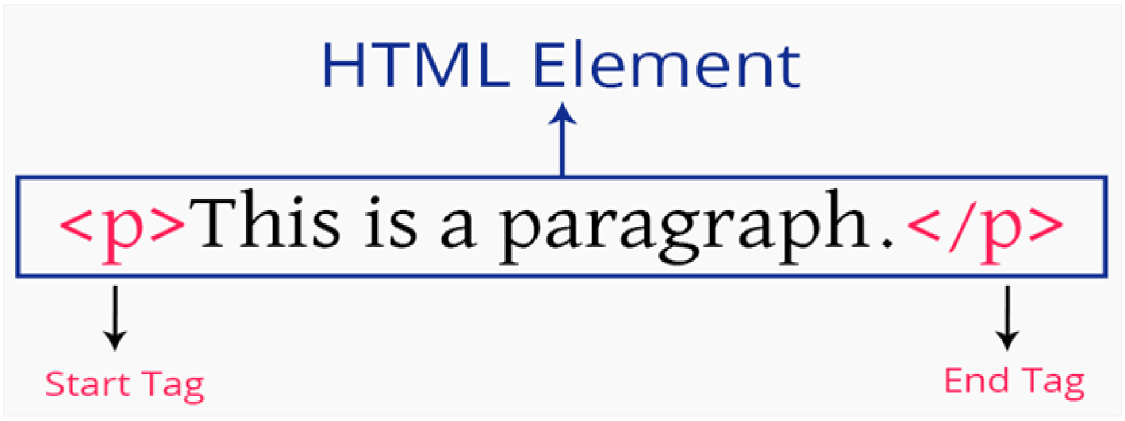
https://developer.mozilla.org/en-US/docs/Web/HTML/Element

These can be explored whenever required while making a website.

HTML ELEMENTS

Elements are the things that make up the web page. Tags define the beginning and end of HTML elements. A web page can be seen as a collection of HTML elements.





The basic elements used till now have been briefly described below

	HTML Element	**Description**
1	**<p> CONTENT </p>**	Paragraph tag
2	**<h1> CONTENT </h1>**	Heading tag
3	** **	Break tag - to enter into a new line
4	**<hr>**	Horizontal ruler

Paragraphs

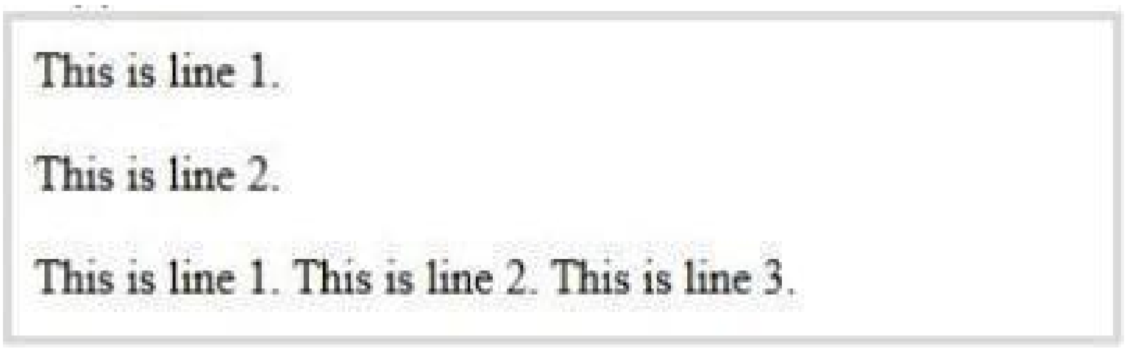
Paragraphs are **blocks of text** separated from each other by some space.

They are defined using the **<p>** and **</p>** tags. When the p element ends, the next element appears in the next line.

E.g.: here's a sample of code for <p> tag:

```
<!DOCTYPE html>
<html>
  <head>
    <title>p tag</title>
  </head>
  <body>
    <p>This is line 1.</p>
    <p>This is line 2.</p>
    <!-- trying to format the text wiathout using p-tag -->
    This is line 1. This is line 2. This is line 3.
  </body>
</html>
```

It appears on a web browser like this:


This is line 1.

This is line 2.

This is line 1. This is line 2. This is line 3.

NOTE: *When formatting without a p-tag, new lines are appended on the current line. This happens because the* ***spacing of text doesn't matter to the browser.***

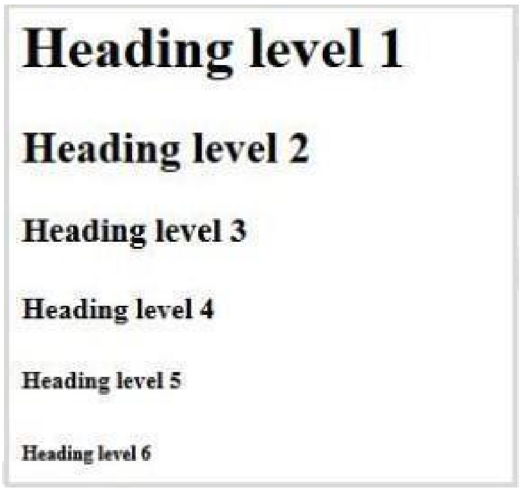
Headings

These are HTML tags that are used to indicate that some content should be treated as **headings**. The headings are divided into six levels: **h1, h2, h3, h4, h5**, and **h6.** Among them, **h1** is the **highest-level** heading and **h6** is the lowest-level heading.

E.g.: here's a sample of code for Heading tags

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Heading Levels</title>
  </head>
  <body>
    <h1>Heading level 1 </h1>
    <h2>Heading level 2 </h2>
    <h3>Heading level 3 </h3>
    <h4>Heading level 4 </h4>
    <h5>Heading level 5 </h5>
    <h6>Heading level 6 </h5>
  </body>
</html>
```

Output:



```
Heading level 1
Heading level 2
Heading level 3
Heading level 4
Heading level 5
Heading level 6
```

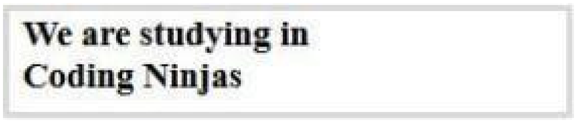
BR Tag

<br> tag can be used to make a **single line split** between the contents of the tab. This means that when this tag is used between a single line, the **contents after this tag will pass to the next line**. Do not use it to allow space between a block of elements (eg., paragraph and heading).

E.g.

```
<h3>We are studying in<br/>Coding Ninjas</h3>
```

The output is as follows:



We are studying in
Coding Ninjas

HR Tag

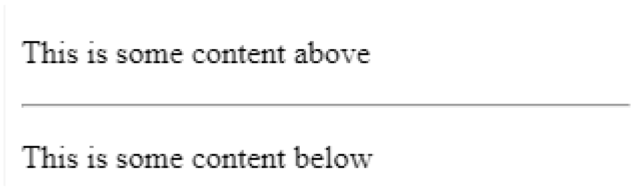
The <hr> tag in HTML, short for "horizontal rule," is a simple and effective element used to create a horizontal line or divider on a web page. This tag is self-closing, meaning it doesn't require a corresponding closing tag. Historically, the <hr> tag was employed to indicate a thematic break or a visual division between different sections of a document.

The basic syntax of the <hr> tag is uncomplicated: <hr>. When used in HTML, it produces a horizontal line that spans the width of its containing element. While it has attributes like align, width, and color, these are often considered deprecated in HTML5, and styling is usually accomplished through CSS for better control over appearance.

```
<!DOCTYPE html>
<html>
    <head>
      <title>Hello, World !< /title>
    </head>
    <body>
      <p>This is some content above</p>
      <hr>
```

```
        <p>This is some content below</p>
    </body>
</html>
```

Output: -


This is some content above

This is some content below

LISTS

Lists are used to **group different pieces of information** so that they are **easily linked** and **easy to read**.

Lists help construct a **well-structured**, more open, and **easy-to-maintain** document from a structural standpoint.

There are three types of lists to pick from: **ordered**, **unordered**, and **description lists**.

Unordered Lists

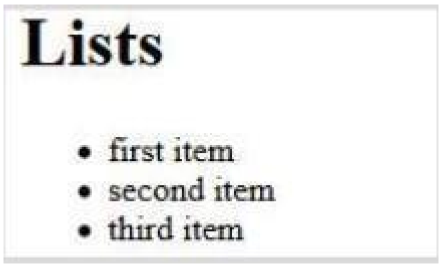
It's used to group a group of similar objects that aren't arranged in any specific order. Where the counting of objects isn't necessary, unordered lists are used.

Bullets are used by default to separate the items. They are defined using the <ul> tag.

Example:

```
<!DOCTYPE html>
<html>
 <head>
   <title>Unordered Lists</title>
 </head>
 <body>
   <h1>Lists</h1>
   <ul>
     <li>first item</li>
     <li>second item</li>
     <li>third item</li>
   </ul>
 </body>
</html>
```

The output is as follows:



```
Lists

  • first item
  • second item
  • third item
```

HTML provides an interesting feature to change the style of the list item marker. There are 4 types of **styles in unordered lists:**

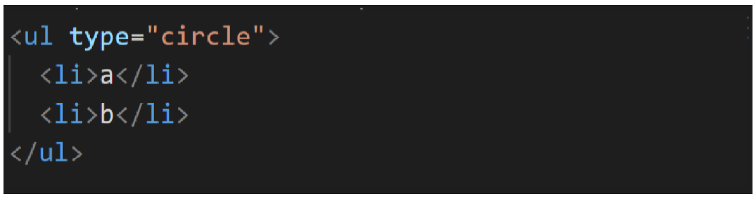
● ***type="disc"*** - Sets the list item marker to a bullet (default).

○ ***type="circle"*** - Sets the list item marker to a circle.

■ ***type="square"*** - Sets the list item marker to a square.

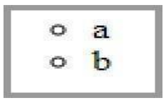
type="none" - The list items will not be marked.

For example, to create a list with ***type=circle:***



```
<ul type="circle">
  <li>a</li>
  <li>b</li>
</ul>
```

Output: -



```
○ a
○ b
```

***NOTE**: The above styles can be produced by using the '**type'** attribute.*

Ordered Lists

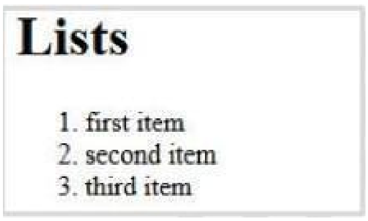
It is used in a certain order to group several related items.

When the n**umbering of items is necessary**, ordered lists are used. By default, **numerical numbers follow** the items.

They are defined using the **<ol>** tag. Eg:

```
<html>
  <head>
    <title>Ordered Lists</title>
  </head>
  <body>
    <h1>Lists</h1>
    <ol>
        <li>first item</li>
        <li>second item</li>
        <li>third item</li>
    </ol>
  </body>
</html>
```

The output is as follows:



```
Lists

  1. first item
  2. second item
  3. third item
```

Similarly, like the unordered lists, there are also different types of ways to number the ordered lists using the **'type'** attribute:

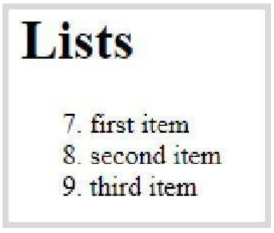
1. **type= "1"** - the numbering will contain **numbers** (default)
2. **type= "A"** - the numbering will contain **uppercase letters**
3. **type= "a"** - the numbering will contain **lowercase letters**

4. **type= "I"** - the numbering will contain **uppercase Roman numbers**
5. **type= "i"** - the numbering will contain **lowercase Roman numbers**

Now, what if you want to change the starting numbering of the lists?

HTML has got the solution for it: the **'start'** attribute. So, if we change <ol> to

<ol start="7">, you will now see the output as



```
Lists

  7. first item
  8. second item
  9. third item
```

Description Lists

A list of definitions is **not the same** as a list of items. This is a **collection of items with an explanation.**

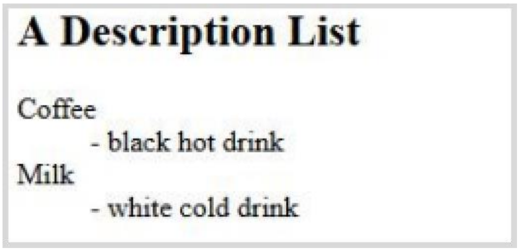
Tag	Description
<dl> tag	to **start** a definition list.
<dt> tag	to **begin** each definition - list term.
<dd> tag	to **begin each definition - list definition.**

In comparison to ordered and unordered lists, description lists are **very specific in their application** and thus are **rarely used**. However, whenever a structure such as a list of terms and their descriptions is required, description lists are ideal.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>Description Lists</title>
</head>
<body>
<h2>Description List</h2>
<dl>
<dt>Coffee</dt>
<dd>- black hot drink</dd>
<dt>Milk</dt>
<dd>- white cold drink</dd>
</dl>
</body>
</html>
```

The output is as follows:



```
A Description List

Coffee
        - black hot drink
Milk
        - white cold drink
```

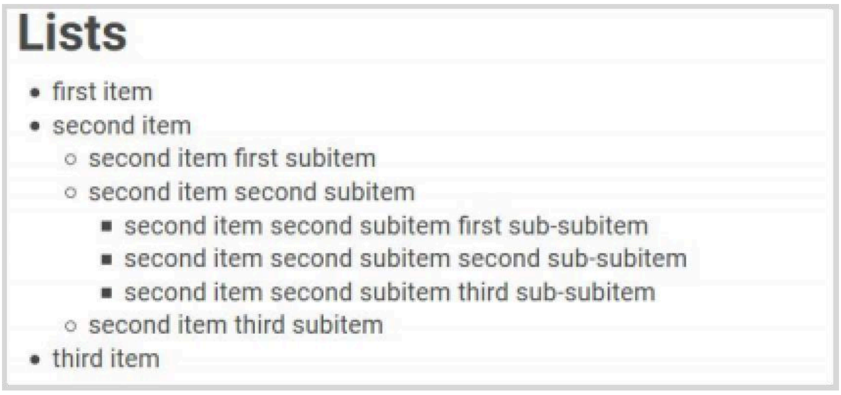
NESTING ELEMENTS

HTML elements can be nested i.e. **elements can contain elements.** All HTML documents consist of nested HTML elements

E.g.:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Hello, World !< /title>
    <link rel="stylesheet" href="styles.css" />
  </head>
  <body>
     <ul>
      <li>first item</li>
      <li>second item
        <ul type="circle">
          <li>second item first subitem</li>
          <li>second item second subitem
             <ul type="square">
                <li>second item second subitem first sub-subitem</li>
                <li>second item second subitem second sub-subitem</li>
                <li>second item second subitem third sub-subitem</li>
             </ul>
          </li>
         <li>second item third subitem</li>|
       </ul>
      </li>
      <li>third item</li>
     </ul>
 </body>
</html>
```

This will give the output as



```
Lists
• first item
• second item
    ○ second item first subitem
    ○ second item second subitem
        ▪ second item second subitem first sub-subitem
        ▪ second item second subitem second sub-subitem
        ▪ second item second subitem third sub-subitem
    ○ second item third subitem
• third item
```

***NOTE**: There is no limitation to the depth of nested lists. Although it is true for all paired/container tags, we should be careful in nesting elements inside each other and should only do something meaningful.*

CONCLUSION:

In conclusion, the notes provide a foundational understanding of essential HTML concepts. The focus begins with the role of comments in HTML, serving as a means for developers to explain their code or highlight important elements for future editing. Notably, these comments remain invisible on web pages as browsers disregard them during rendering.

Moving on to tags, the notes articulate their significance as representations of HTML elements, dictating the formatting and display of content within a webpage. Tags, enclosed in angle brackets, offer a structured approach to organizing and presenting information. The case-insensitive nature of tag names and the necessity for paired opening and closing tags are emphasized.

The discussion extends to boilerplate tags, emphasizing the crucial role of **<DOCTYPE html>**, **<html>**, **<head>**, **<title>**, and **<body>** in establishing the foundational structure of HTML documents. The **<head>** section is highlighted for its role as a container for metadata, while **<title>** influences the document title displayed in the browser tab. The **<body>** tag encapsulates all elements visible on the webpage.

The discussion extends to boilerplate tags, emphasizing the crucial role of <DOCTYPE html>, <html>, <head>, <title>, and <body> in establishing the foundational structure of HTML documents. The <head> section is highlighted for its role as a container for metadata, while<title> influences the document title displayed in the browser tab. The <body>tag encapsulates all elements visible on the webpage.

The exploration of HTML elements delves into commonly used tags such as **<p>**, **<h1>**, **<br>**, and **<hr>**. These tags play key roles in defining paragraphs, headings, line breaks, and horizontal rules, respectively. The practical examples enhance the understanding of how these elements are employed in real-world scenarios.

Additionally, the notes comprehensively cover lists, differentiating between unordered (**<ul>**), ordered (**<ol>**), and description (**<dl>**) lists. Styling options for list items, including types and numbering, are elucidated. The notes also touch upon the concept of nesting elements, highlighting that HTML elements can be nested within each other, facilitating a structured and hierarchical organization of content.

Overall, this set of notes serves as a valuable introductory resource for individuals embarking on HTML-based web development. The detailed explanations, coupled with practical examples, contribute to a well-rounded comprehension of HTML fundamentals, laying a solid foundation for further exploration and learning in web development.

REFERENCES:

a. https://developer.mozilla.org/en-US/docs/Learn/HTML/Introduction_to_HTML/HTML_text_fundamentals

b. https://developer.mozilla.org/en-US/docs/Web/HTML/Element

More on HTML

<img> tag

The <img> tag in HTML embeds images into a web page. It's an empty or self-closing tag, meaning it doesn't have a closing tag. Here's an overview of its attributes:

Required Attributes:

- **‘src’**: Specifies the URL or path to the image file. This attribute is mandatory.

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

Optional Attributes:

- ‘**alt**’: Provides alternative text for the image. It's essential for accessibility and SEO. It describes the image if it fails to load or for disabled users.

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

- **‘width’** and ‘**height**’: Set the width and height of the image, respectively, in pixels. These attributes are optional but can help browsers reserve space for the image before it loads, preventing layout shifts.

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

- **‘title’**: Adds a tooltip when the user hovers over the image. It's not mandatory but can be helpful.

```
<img src="image.jpg" alt="Description" title="Image Title">
```

- **‘loading’**: Determines how the browser should load the image. Options include ‘**lazy’**, ‘**eager’**, and ‘**auto’**. lazy postpones loading until the image is near the viewport.

```
<img src="image.jpg" alt="Description" loading="lazy">
```

- ‘**decoding’**: Specifies the decoding method of the image. Options include async, sync, and auto. It controls the image decoding process.

```
<img src="image.jpg" alt="Description" decoding="async">
```

- **‘srcset’** and ‘**sizes**: For responsive images, srcset allows specifying multiple image sources with different resolutions, while sizes help the browser understand the image's display size.

```
<img src="image.jpg" alt="Description" srcset="image-1x.jpg 1x, image-2x.jpg 2x" sizes="(max-width: 600px) 200px, 50vw">
```

These attributes provide flexibility in displaying images while considering performance, accessibility, and responsiveness across various devices and screen sizes.

Relative and Absolute Paths

In web development, paths refer to the location of files or resources within a directory structure. There are two main types of paths: relative paths and absolute paths.

Relative paths: Relative paths specify the location of a file about the current page. They don’t include the full URL but rather the path from the current directory.

Types of Relative Paths:

1. **Relative to the Current Directory:** If the file you want to link to is in the same directory as the current HTML file, you can reference it directly.

```
<img src="image.jpg" alt="Image">
```

2. **Relative to the Parent Directory:** Use '../' to move up one directory level. For instance, if the file is one level up from the current directory:

```
<img src="../images/image.jpg" alt="Image">
```

3. **Relative to Child Directories:** If the file is located in a subdirectory of the current directory, indicate the path accordingly:

```
<img src="assets/image.jpg" alt="Image">
```

Absolute Paths: Absolute paths define the full URL or path from the root directory to the file. They include the protocol (e.g., http:// or https://) and the complete address.

Types of Absolute Paths:

1. **URL-Based Absolute Paths:** Specify the full URL to the file or resource:

```
<img src="https://www.example.com/images/image.jpg" alt="Image">
```

2. **Server-Based Absolute Paths:** Provide the full path from the root directory of the server:

```
<img src="/images/image.jpg" alt="Image">
```

Factors to Consider:

1. **Portability**: Relative paths are more portable as they adjust automatically if the site structure changes.
2. **Ease of Use**: Absolute paths might be easier to implement for resources hosted on other domains or servers.
3. **Maintenance**: Relative paths can simplify maintenance within a website's directory structure.

When choosing between relative and absolute paths, consider the context of your project, ease of maintenance, and the flexibility needed for future modifications to your website's structure or location of resources.

<video> tag

The <video> tag in HTML is used to embed video content into a web page. It lets you play videos directly within the browser without needing third-party plugins. Here's an overview of its usage and attributes:

Basic Structure:

```
<video controls width="400" height="300">
  <source src="video.mp4" type="video/mp4">
  Your browser does not support the video tag.
</video>
```

Attributes of the <video> Tag:

1. **controls**: Adds playback controls (play, pause, volume, etc.) to the video player.

```
<video controls>
```

2. **width** and **height**: Sets the width and height of the video display area in pixels.

```
<video width="400" height="300">
```

3. **src**: Specifies the URL or path to the video file. Multiple sources can be provided inside <source> tags to support different formats and browsers.

```
<source src="video.mp4" type="video/mp4">
<source src=" video.webm" type="video/webm">
```

4. **type**: Defines the MIME type of the video file. It helps the browser determine which video format to use.
    a. For MP4: **type="video/mp4"**
    b. For WebM: **type="video/webm**"
    c. Other formats have different MIME types.

5. **autoplay**: Automatically starts playing the video when the page loads. Use it cautiously, as autoplaying content can be intrusive.

```
<video autoplay>
```

6. **loop**: Repeats the video playback once it reaches the end.

```
<video loop>
```

7. **muted**: Starts the video muted by default.

```
<video muted>
```

8. **poster**: Displays an image while the video is loading or before it's played. It's a fallback option if the video takes time to load.

```
<video poster="video-poster.jpg">
```

9. **Preload**: Specifies whether the video should be loaded when the page loads ("auto"), whether only metadata should be loaded ("metadata"), or whether the video should not be preloaded ("none").

```
<video preload="auto">
```

These attributes control how the video is displayed, played, and interacted with on the webpage, making the <video> tag versatile for incorporating video content into your site.

<audio> tag

The <audio> tag in HTML is used to embed audio files directly into a web page. It allows for the inclusion of audio content without relying on third-party plugins. The <source> tag is used within the <audio> tag to provide multiple audio sources in different formats for compatibility across browsers.

Attributes of the <audio> Tag:

1. **controls**: Adds playback controls (play, pause, volume, etc.) to the audio player.

```
<audio controls>
```

2. **src**: Specifies the URL or path to the audio file. Multiple sources can be provided inside <source> tags to support different formats and browsers.

```
<source src="audio.mp3" type="audio/mpeg">
<source src="audio.ogg" type="audio/ogg">
```

3. **type**: Defines the MIME type of the audio file. It helps the browser determine which audio format to use.
    a. For MP3: type="audio/mpeg"
    b. For Ogg Vorbis: type="audio/ogg"
    c. Other formats have different MIME types.

4. **autoplay**: Automatically starts playing the audio when the page loads. Similar to video autoplay, use this sparingly.

```
<audio autoplay>
```

5. **loop**: Repeats the audio playback once it reaches the end.

```
<audio loop>
```

6. **muted**: Starts the audio muted by default.

```
<audio muted>
```

7. **preload**: Specifies whether the audio should be loaded when the page loads ("auto"), only metadata should be loaded ("metadata"), or the audio should not be preloaded ("none").

```
<audio preload="auto">
```

These attributes allow for customisation of audio playback and provide options to ensure compatibility with different browsers by offering alternative audio formats using the <source> tag inside the <audio> element.

<iframe> tag

The <iframe> (short for inline frame) tag in HTML is used to embed another document within the current HTML document. It allows you to display content from another source inside the current page, like a webpage, video, map, or any other HTML document.

Attributes of the <iframe> Tag:

1. **src**: Specifies the document URL or resource to be embedded within the iframe.

```
<iframe src="https://www.example.com"></iframe>
```

2. **width** and **height**: Sets the width and height of the iframe's display area in pixels.

```
<iframe width="600" height="400"></iframe>
```

3. **title**: Provides a title or name for the embedded content. It's good practice to include this attribute for accessibility reasons.

```
<iframe title="Embedded Content"></iframe>
```

4. **sandbox**: Adds a security sandbox to the iframe, restricting what the embedded content can do (such as preventing scripts or forms from running).

```
<iframe sandbox></iframe>
```

5. **allow**: Specifies a list of permissions for the embedded content, such as allowing or restricting specific features like fullscreen mode or autoplay for videos.

```
<iframe allow="fullscreen"></iframe>
```

6. **loading**: Determines how the iframe content should be loaded. Options include eager (load immediately) and lazy (load when near the viewport).

```
<iframe loading="lazy"></iframe>
```

7. **frameborder**: Specifies whether to display a border around the iframe. 0 means no border, while 1 means a border should be displayed.

```
<iframe frameborder="0"></iframe>
```

The **<iframe>** tag is commonly used to embed content from other sources into a webpage while providing a segregated environment for that content. Using it responsibly is crucial, considering security and user experience aspects.

<a> tag

The anchor tag, **<a>**, in HTML, is used to create hyperlinks to other web pages, files, locations within the same page, or content on the internet. It's the foundation for navigating the web by linking various resources.

Attributes of the <a> Tag:

1. **href**: The most crucial attribute; it specifies the destination URL or the location to which the link points.

```
<a href="https://www.example.com">Link Text</a>
```

2. **target**: **Specifies where the linked document should open**. Common values include _self (default, opens in the same window/tab), _blank (opens in a new window/tab), _parent, and _top.

```
<a href="https://www.example.com" target="_blank">Link Text</a>
```

3. **download**: Forces the browser to download the linked file instead of navigating to it when clicked. It's used for links to downloadable resources.

```
<a href="file.pdf" download>Download PDF</a>
```

4. **rel**: Describes the relationship between the current and linked documents. For example, rel="nofollow" instructs search engines not to follow the link for indexing.

```
<a href="https://www.example.com" rel="nofollow">Link Text</a>
```

5. **title**: Adds a tooltip that appears when the user hovers over the link.

```
<a href="https://www.example.com" title="Go to Example">Link Text</a>
```

6. **aria-label**: Provides an accessible label for screen readers to describe the purpose of the link.

```
<a href="https://www.example.com" aria-label="Go to Example">Link Text</a>
```

7. **role**: Specifies the role of the link for accessibility purposes. For instance, role="button" links look and behave like a button.

```
<a href="https://www.example.com" role="button">Link Text</a>
```

The **<a>** tag is fundamental for creating navigable content on the web. It allows users to move between different web pages and resources, making it a core element in HTML for building interconnected web experiences.

iframe with anchor Tag:

An iframe can be used as the target frame for a link.
The target attribute of the link must refer to the name attribute of the iframe:

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

<h2>Iframe -Target for a Link</h2>

<iframe
src=" "
name="iframe_a"
height="300px"
width="100%"
title="Iframe Example">
</iframe>

<p>
<a
href="https://wwwcodingninjas.com"
target="iframe_a"
>
W3Schools.com
</a>
</p>

<p>When the target attribute of a link matches the name of an iframe, the link will open in the iframe.</p>

</body>
</html>
```

You have to leave the src attribute empty in the iframe tag.
Put the link of any website and location in the href attribute of the anchor tag.

<table> tag

The <table> tag in HTML is used to create tables, allowing you to display data in rows and columns. Tables consist of rows (<tr>) and cells (<td> or <th>) organised within them.

Tags and Components within the Table:

1. **<table>**: The container tag for the entire table.
2. **<tr>**: Stands for "table row." Used to define a row in the table.
3. **<th>**: Stands for "table header cell." Used to define headers within a table. Typically placed within the <tr> element to denote header cells for each column or row.
4. **<td>**: Stands for "table data cell." Used to define regular cells within a table (non-header cells). These cells contain the actual data.

Attributes of the <table> Tag:

1. **border**: Specifies the border width of the table. Deprecated in HTML5; CSS is recommended for styling.

```
<table border="1">
```

2. **width**: Sets the width of the table. Can be specified in pixels or as a percentage.

```
<table width="100%">
```

3. **cellpadding**: Adds space inside the cells of the table. Deprecated in HTML5; CSS is recommended for styling.

```
<table cellpadding="5">
```

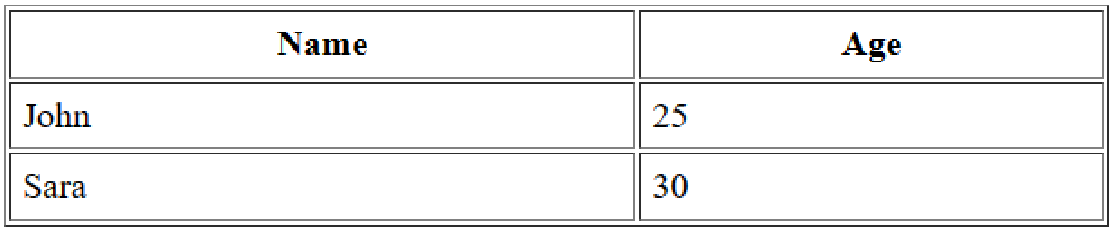
4. **cellspacing**: Specifies the space between cells in the table. Deprecated in HTML5; CSS is recommended for styling.

```
<table cellspacing="2">
```

Example:

```
<table border="1" width="100%" cellpadding="5" cellspacing="2">
 <tr>
  <th>Name</th>
  <th>Age</th>
 </tr>
 <tr>
  <td>John</td>
  <td>25</td>
 </tr>
 <tr>
  <td>Sara</td>
  <td>30</td>
 </tr>
</table>
```

The output is as follows:-



Name	Age
John	25
Sara	30

Tables are used for organising and presenting tabular data in a structured format. However, in modern web design, they're often replaced with CSS frameworks and other layout methods for better responsiveness and accessibility.

5. **rowspan and colspan:**
In HTML, the <table> element is used to create tables, and the rowspan and colspan attributes are used to define the span of a cell in terms of rows and columns, respectively. Here are some notes on rowspan and colspan:

rowspan Attribute:

The rowspan attribute is used to specify how many rows a cell should span.

It is applied to a <td> (table data) or <th> (table header) element within a <tr> (table row) element.

The value of rowspan is an integer representing the number of rows the cell should span.

When a cell has a rowspan greater than 1, it occupies multiple rows in the table.

Example:

```
<table border="1">
  <tr>
    <td rowspan="2">Cell 1</td>
    <td>Cell 2</td>
    <td>Cell 3</td>
  </tr>
  <tr>
    <td>Cell 4</td>
    <td>Cell 5</td>
  </tr>
</table>
```

In this example, the first cell (Cell 1) spans two rows.

colspan Attribute:

The colspan attribute is used to specify how many columns a cell should span.

Like rowspan, it is applied to a <td> or <th> element.

The value of colspan is an integer representing the number of columns the cell should span.

When a cell has a colspan greater than 1, it occupies multiple columns in the table.

Example:

```
<table border="1">
  <tr>
    <td>Cell 1</td>
    <td colspan="2">Cell 2</td>
    <td>Cell 3</td>
  </tr>
</table>
```

In this example, the second cell (Cell 2) spans two columns.

Combining rowspan and colspan:

You can combine both rowspan and colspan attributes in a single cell to create a cell that spans both rows and columns.

Example:

```
<table border="1">
  <tr>
    <td rowspan="2" colspan="2">Cell 1</td>
    <td>Cell 2</td>
  </tr>
  <tr>
    <td>Cell 3</td>
  </tr>
</table>
```

In this example, the first cell (Cell 1) spans two rows and two columns.

These attributes are useful for creating more complex table layouts and merging cells to accommodate different data structures in tables.

CONCLUSION:

In conclusion, HTML tags like **<img>, <video>, <audio>, <source>, <iframe>, <a>,** and **<table>** offer incredible versatility in web design and content presentation. From embedding multimedia elements, and creating hyperlinks, to organizing data into tables, these tags provide the foundation for building engaging, accessible, and interactive web experiences.
Understanding their attributes and usage empowers developers to craft visually compelling, multimedia-rich, and structurally organized web pages, catering to diverse user needs and preferences. As web technologies evolve, these tags remain fundamental tools, facilitating the creation of immersive and dynamic online content across various devices and platforms.

REFERENCES:

Explore more about these tags by going through the links below

1. <img> tag
2. <video> tag
3. <audio> tag
4. <iframe> tag
5. <a> tag
6. <table> tag

HTML Input Attributes

Here's an overview of various attributes that can be used with HTML input elements:

Common Attributes:

- **readonly**: Specifies that the input field is read-only and cannot be modified by the user.

```
<input type="text" readonly value="Read-only text">
```

- **disabled**: Disables the input field, preventing user interaction.

```
<input type="text" disabled>
```

- **size:** Sets the visible width of the input field in characters.

```
<input type="text" size="30">
```

- **maxlength:** Specifies the maximum number of characters in the input field.

```
<input type="text" maxlength="50">
```

- **min and max:** Defines numeric input fields' minimum and maximum values.

```
<input type="number" min="0" max="100">
```

- **multiple:** Allows users to select multiple options in a <select> dropdown.

```
<select multiple>
  <option value="option1">Option 1</option>
  <option value="option2">Option 2</option>
</select>
```

Validation and Constraints:

- **pattern:** Specifies a regular expression pattern for validating the input.

```
<input type="text" pattern="[A-Za-z]{3}">
```

- **placeholder:** Provides a short hint that describes the expected value of the input field

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

- **required:** Specifies that the input field must be filled out before submitting the form.

```
<input type="text" required>
```

- **step:** Specifies the step increment or jump between values in a numeric input.

```
<input type="number" step="5">
```

Other Attributes:

- **autofocus:** Automatically focuses the input field when the page loads.

```
<input type="text" autofocus>
```

- **height and width:** Defines the height and width of the input field, mainly used with **<input type="image">.**

```
<input type="image" src="button.png" height="50" width="100">
```

- **list:** Associates the input field with a **<datalist>** element, providing predefined options.

```
<input type="text" list="browsers">
<datalist id="browsers">
  <option value="Chrome">
  <option value="Firefox">
</datalist>
```

- **autocomplete:**
  The autocomplete attribute in HTML specifies whether an input field should have autocomplete enabled.
  When autocomplete is enabled, the browser can suggest and automatically fill values based on previously entered data by the user.

  Values of the autocomplete Attribute
  1. on:
     Enables autocomplete.
     The browser will attempt to fill the input field based on previously entered values.
     This is the default behaviour if the autocomplete attribute is not specified.
  2. off:
     Disables autocomplete.
     The browser will not suggest or fill in values for the input field.
  3. name, email, username, password, new-password, current-password, one-time-code, etc.:
     These are specific values that give the browser more context about the expected input.
     Helps the browser provide more accurate and relevant autocomplete suggestions.

     Example Usage:

```
<form action="/submit">
  <label for="name">Name:</label>
  <input type="text" id="name" name="name" autocomplete="name">

  <label for="email">Email:</label>
  <input type="email" id="email" name="email"
autocomplete="email">

  <label for="username">Username:</label>
  <input type="text" id="username" name="username"
autocomplete="username">

  <label for="password">Password:</label>
  <input type="password" id="password" name="password"
autocomplete="new-password">

  <input type="submit" value="Submit">
</form>
```

These attributes control the appearance, behaviour, and validation of HTML input elements, contributing to a more interactive and user-friendly form experience on web pages.

Conclusion:

In conclusion, HTML input attributes play a crucial role in enhancing the functionality and user experience of web forms. The common attributes such as readonly, disabled, size, maxlength, min, and max provide control over user interaction, input size, and numeric constraints. The validation and constraint attributes like pattern, placeholder, required, and step contribute to data integrity and user guidance. Additionally, other attributes like autofocus, height and width, and list add further customization options. By leveraging these attributes, web developers can create more interactive and user-friendly forms, ultimately improving the overall usability of their websites.

References:

1. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input#attributes

Forms

<form> tag

The <form> tag in HTML is a container used to create an interactive form on a web page. Forms are crucial for user input, allowing visitors to submit data to a server for processing. The <form> tag encompasses various form elements like text fields, checkboxes, radio buttons, and more.

Basic Structure:

```
<form action="/submit" method="post">
<!-- Form elements go here -->
</form>
```

Attributes of the <form> Tag:

- **‘action’**: Specifies the URL or script to which the form data should be submitted. It indicates where the form data will be processed.

```
<form action="/submit" method="post">
```

- ‘**method’**: Defines the HTTP method to send form data to the server. Typical values are "**get**" and "**post**".

```
<form action="/submit" method="post">
```

- ‘**enctype’**: Specifies the encoding type for form data when using the post method. Common values include "**application/x-www-form-urlencoded**" and "**multipart/form-data**". This is often used when dealing with file uploads.

```
<form action="/submit" method="post" enctype="multipart/form-data">
```

- **'target'**: Determines where the form response should be displayed. Values include _blank (opens in a new window or tab), _self (default, opens in the same window), _parent, and _top.

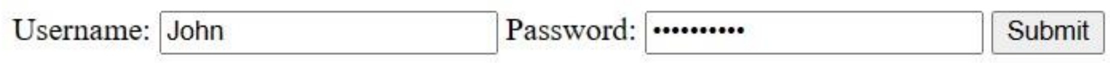
```
<form action="/submit" method="post" target="_blank">
```

Example:

```
<form action="/submit" method="post" enctype="multipart/form-data">
  <label for="username">Username:</label>
  <input type="text" id="username" name="username" required>

  <label for="password">Password:</label>
  <input type="password" id="password" name="password" required>

  <input type="submit" value="Submit"> </form>
```


Username: John Password: •••••••••• Submit

In this example, the form collects a username and password, and when submitted, the data is sent to the server-side script specified in the action attribute. The method is set to "post" for more secure data transmission, and the enctype is specified for potential file uploads. The required attribute ensures that users must fill in the specified fields before submitting the form.

- **'novalidate'**: The novalidate attribute is used to disable the default browser validation of form fields. When this attribute is present on the <form> tag, it prevents the browser from validating the form before submission. This can be useful when you want to perform custom validation on the server side or using JavaScript..

```
<form action="/submit" method="post" novalidate>
  <!-- form fields go here --> </form>
```

- **‘autocomplete’**: The autocomplete attribute is used to control whether a browser should automatically complete form fields or not. It is applied to individual form elements or the <form> tag itself. The values it can take are "**on**" or "**off**".

    - **autocomplete="on"**: This is the default behaviour, where the browser may store and automatically fill in form data based on the user's previous input

```
<form action="/submit" method="post" autocomplete="on">
<!-- form fields go here -->
</form>
```

    - **autocomplete="off"**: This disables the browser's autocomplete feature, preventing it from suggesting or filling in values for form fields.

```
<form action="/submit" method="post" autocomplete="off">
<!-- form fields go here -->
</form>
```

Elements of the <form> tag:

- **‘input’**: The <input> tag is used to create various types of form controls, such as text fields, checkboxes, radio buttons, and more. The specific type is specified using the type attribute.

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

- **‘label’:** The <**label**> tag associates a text label with a form control, enhancing accessibility and user experience. It is often used with form elements like <**input**>, <**select**>, and <**textarea**>.

```
<label for="username">Username:</label>
<input type="text" id="username" name="username">
```

- **‘<select> and <option>’:** The <select> tag creates a dropdown list, and <option> tags define the options within the list.

```
<select name="cars">
<option value="volvo">Volvo</option>
<option value="saab">Saab</option>
<option value="mercedes">Mercedes</option>
<option value="audi">Audi</option>
</select>
```

- **‘<textare>’:** The <**textarea**> tag is used to create a multiline text input field, typically for longer text entries or comments.

```
<textarea name="message" rows="4" cols="50">Enter your message here...
</textarea>
```

- **<button>:** The <button> tag creates a clickable button, which can be used for various purposes such as form submission or triggering JavaScript functions.

```
<button type="submit">Submit</button>
```

- **<fieldset> and <legend>:** The <fieldset> tag groups related form elements together, and <legend> provides a caption or title for the <fieldset>.

```
<fieldset>
<legend>Personal Information</legend>
<!-- Form elements go here -->
</fieldset>
```

- **<datalist> and <option>:** The <datalist> tag contains a list of <option> elements used to provide suggestions or autocompletion for an <input> field

```
<input list="browsers" name="browser" id="browser">
<datalist id="browsers">
  <option value="Chrome">
  <option value="Firefox">
  <option value="Safari">
  <option value="Edge">
  <option value="Opera"> </datalist>
```

- **<optput>:** The <output> tag is used to represent the result of a calculation or user action.

```
<form oninput="result.value=parseInt(a.value)+parseInt(b.value)">
  <input type="range" id="a" value="50">+
  <input type="number" id="b" value="50">
  =<output name="result" for="a b">100</output> </form>
```

- **<optgroup>:** The <optgroup> tag groups <option> elements within a <select> dropdown, providing a way to organize and structure the options.

```
<select>
  <optgroup label="Fruits">
    <option value="apple">Apple</option>
    <option value="banana">Banana</option>
  </optgroup>
  <optgroup label="Vegetables">
    <option value="carrot">Carrot</option>
    <option value="broccoli">Broccoli</option>
  </optgroup> </select>
```

These HTML tags play crucial roles in creating interactive and user-friendly forms on web pages. They provide a variety of input methods and customisation options to cater to different types of user interactions.

HTML Input Types

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

- **<input type="button">**: Creates a clickable button, typically used in conjunction with JavaScript for custom actions.
- **<input type="checkbox">**: Creates a checkbox for selecting multiple options in a group.
- **<input type="color">:** Provides a colour picker interface for selecting a colour.
- **<input type="date">:** Generates a date input field, allowing users to select a date from a calendar.
- **<input type="datetime-local">:** Allows users to input a date and time in the local time zone.
- **<input type="email">:** Creates a text box specifically for email input, with built-in validation for email addresses.
- **<input type="hidden">:** Hides the input value from the user interface but allows it to be submitted with the form.
- **<input type="image">:** Creates an image as a submit button. When clicked, it sends the coordinates of the click location to the server.
- **<input type="month">:** Allows users to input a month and year.
- **<input type="number">:** Creates a numeric input field, restricting input to numeric values.
- **<input type="password">:** Generates a password input field, obscuring the entered text.
- **<input type="radio">:** Creates a radio button for selecting a single option from a group.

- **<input type="range">:** Creates a slider control, allowing users to select a value within a specified range.

- **<input type="reset">:** Resets all form elements to their default values.

- **<input type="search">:** Generates a search input field.

- **<input type="submit">:** Creates a submit button for submitting form data to the server.

- **<input type="tel">:** Allows users to input a telephone number.

- **<input type="text">:** Generates a single-line text input field.

- **<input type="time">:** Allows users to input a time value.

- **<input type="url">:** Creates a text box for inputting a URL, with built-in validation for URLs.

- **<input type="week">:** Allows users to input a week and year.

Each input type serves a specific purpose, providing a wide range of options for collecting user input in web forms.

File Input

The <**input**> element with **type**="**file**" is used to create a file input field, allowing users to select and upload files from their devices. The associated attributes -**accept**, -**capture**, -**id**, and -**name** are not standard HTML attributes for the <input> element, but they might be used in certain contexts or frameworks.

Here is the explanation of the standard and commonly used attributes for the <input type="file"> element:

1. **accept**: Specifies the types of files the file input should accept. It is a comma-separated list of file type specifiers, such as file extensions or MIME types. This attribute can help filter the types of files that users can select.

```
<input type="file" accept=".pdf, .doc, .docx">
```

2. **capture**: Used in mobile devices to specify the device's camera or file picker. It can take values like "**user**" (for the front-facing camera), "**environment**" (for the rear-facing camera), or "**filesystem**" (for file pickers).

```
<input type="file" capture="user">
```

3. **id**: Specifies a unique identifier for the input element. This can be useful for associating the input with a label or for targeting the element with JavaScript or CSS.

```
<input type="file" id="fileInput">
```

4. **name**: Specifies the input element's name, which is sent to the server when the form is submitted. This attribute is crucial for identifying the input field in server-side processing.

```
<input type="file" name="fileUpload">
```

Button Tag

The <button> tag in HTML is used to create a clickable button on a webpage. It can contain text, images, or other HTML elements, and it is often used to trigger JavaScript functions, submit forms, or handle user interactions.

Attributes of the <button> Tag:

1. **type**: Specifies the type of button. The most common values are
   a. "**button**" (default): A regular button.
   b. "**submit**": Submits the form data.
   c. "**reset**": Resets the form fields to their default values.

```
<button type="submit">Submit</button>
```

2. **name**: Assigns a name to the button, which is sent to the server when the form is submitted. Useful when multiple buttons exist in the same form.

```
<button type="submit" name="submitButton">Submit</button>
```

3. **value**: Assigns a value to the button, which is also sent to the server when the form is submitted. Useful when multiple buttons submit the same form.

```
<button type="submit" value="submit">Submit</button>
```

4. **disabled:** Disables the button, preventing user interaction. The button appears grayed out.

```
<button type="button" disabled>Disabled Button</button>
```

5. **onclick:** Specifies a JavaScript function to be executed when the button is clicked.

```
<button type="button" onclick="myFunction()">Click me</button>
```

6. **autofocus:** Automatically focuses the button when the page loads.

```
<button type="button" autofocus>Click me</button>
```

HTML Input Form Attributes

1. **formaction**: Specifies the URL to which the form data should be submitted when the user clicks a submit button. This attribute can be applied to <input type="submit">, <button type="submit">, and <button formaction="..."> elements.

   Note: This attribute overrides the action attribute of the <form> element.

```
<form action="/default" method="post">
<input type="submit" formaction="/custom">
</form>
```

2. **formenctype**: Specifies the encoding type for form data when the form is submitted. It can take values such as "**application/x-www-form-urlencoded**", "**multipart/form-data**", or "**text/plain**".
   Note: This attribute overrides the enctype attribute of the <form> element.

```
<form action="/submit" method="post" enctype="multipart/form-data">
<input type="submit" formenctype="text/plain">
</form>
```

3. **formmethod**: Specifies the HTTP method to be used when the form is submitted. It can take values like "get" or "post".
   Note: This attribute overrides the method attribute of the <form> element.

```
<form action="/submit" method="post">
 <input type="submit" formmethod="get"> </form>
```

4. **formtarget**: Specifies the target window or frame in which the form response should be displayed. It can take values like "blank", "self", "parent", or "top".
   Note: This attribute overrides the target attribute of the <form> element.

```
<form action="/submit" method="post">
<input type="submit" formtarget="_blank">
</form>
```

5. **formnovalidate**: When present, this boolean attribute disables form validation before submission. It can be used to submit the form even if some fields do not meet validation criteria.

```
<form action="/submit" method="post">
<input type="submit" formnovalidate>
</form>
```

CONCLUSION:

In conclusion, the attributes and elements discussed in the context of HTML forms provide developers with a powerful toolkit to create interactive and customisable user interfaces. Starting with the <form> tag as the foundational element, various attributes such as action, method, and enctype define how form data is submitted and processed on the server. Attributes like autocomplete, novalidate, and target offer control over the form's behaviour and user experience.

Further customisation is achieved through input elements with diverse types such as text, number, date, and file, each enriched with attributes like readonly, disabled, and placeholder. Validation is enhanced using attributes like required, maxlength, and pattern. Additionally, formaction, formenctype, formmethod, and formtarget allow fine-tuning form behavior on a per-element basis, offering flexibility in form submission and response handling. The novalidate attribute, whether applied to the entire form or individual elements, provides an option to override default browser validation.

Collectively, these HTML form elements and attributes empower developers to create forms that are not only visually appealing but also functional, accessible, and tailored to specific requirements. As web technologies continue to evolve, these tools remain fundamental for crafting seamless and user-friendly interactions on the web.

REFERENCES:

Explore these tags more by going through the links below.

1. Form Tag
2. Form Input Types
3. Input Attributes
4. Form Elements

Semantics

Block Vs Inline Elements

In HTML, elements are categorized as either block-level or inline elements. The distinction between block and inline elements has implications for their default behaviour in terms of layout and how they interact with other elements

Block Elements	Inline Elements
They use the entire webpage width, effectively blocking any other elements from being placed on the left or the right side.	They only take up as much width as necessary to show the element's contents and after that, other elements can be made to match the inline element.
Block elements always start on a new line.	Inline elements do not start from a new line.
Examples: <div>, <p>, <h1>, <h6>, <nav>, etc.	Examples:<b>, <span>, <img> etc

Div Tag

Definition: The <div> (division) element is a fundamental container in HTML used to group and structure other HTML elements. It does not inherently provide any styling or semantic meaning but is a flexible tool for layout and organisation.

Example:

```html
<div id="main-content">
  <p>This is the main content area.</p>
</div>
```

Useful Attributes:

- **id:** Assigns a unique identifier to the <div> for styling or scripting purposes.
  Example:

```
<div id="main-content">...</div>
```

Use Case: Employed to create webpage sections, group related elements, or apply styling to a specific content block.

Span Tag

Definition: The <span> element is an inline container that applies styles or scripting to a specific text portion within a block-level element. Unlike <div>, it doesn't introduce a new line and is typically used for small, inline styling.
Example:

```
<p>This is <span style="color: red;">highlighted</span> text.</p>
```

Useful Attributes:

- **id:** Assigns a unique identifier to the <span> for styling or scripting purposes.
  Example:

```
<p>This is <span id="highlighted-text">highlighted</span> text.</p>
```

Use Case: Useful for applying styles to a specific word or phrase within a larger text block.

Semantic vs Non-Semantic Elements

- Semantic elements clearly define the content they carry. Tags used to enclose them are called semantic tags. For example, a paragraph element, <p>CONTENT</p> tells that its content is a paragraph.

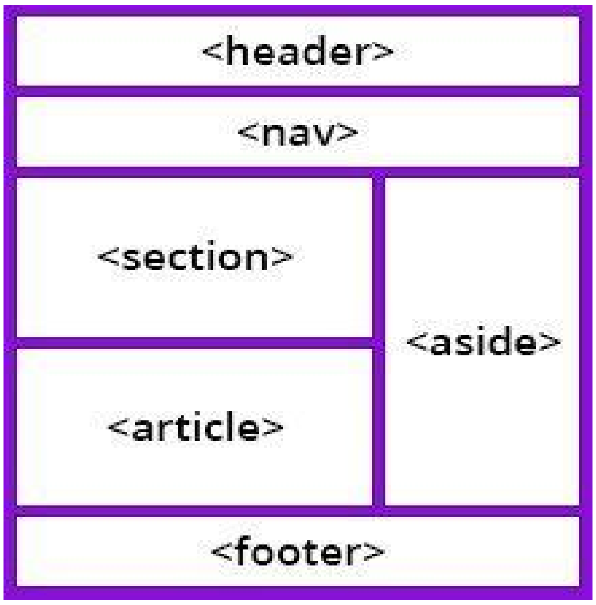
- Some of the semantic elements are:
  article, aside, details, figcaption, figure, footer, header, main, mark, nav, section, etc.

- Non-semantic elements don't tell any information about the content that they carry.

- Tags used to enclose these are called non-semantic tags. For example, a div element, <div>CONTENT</div> can carry any content (headings, paragraphs, links, etc) but the content as a whole has no defined semantics. Hence div is a non-semantic element.

  Note: Semantically correct HTML helps search engines, screen readers, and other user devices determine the significance and context of web content.

Semantic Tags: Building Blocks of HTML Structure



Header Tag

Definition: The <header> element represents a header section in HTML. It can contain headings (<h1> to <h6>), logos, navigation links, and other introductory content. It typically appears at the top of a page or a section.
Example:

```
<header>
  <h1>Your Website</h1>
  <nav>
    <ul>
      <li><a href="#">Home</a></li>
      <li><a href="#">About</a></li>
      <!-- More navigation items -->
    </ul>
  </nav>
</header>
```

Useful Attributes:

- **id:** Assigns a unique identifier to the <header> for linking or styling.
- **role:** Specifies the role of the <header> for accessibility.

  Example:

```
<header id="site-header" role="banner">
  <h1>Your Website</h1>
</header>
```

Use Case: Defines a webpage's introductory or navigational part.

Footer Tag

Definition: The <footer> element represents a footer section in HTML. It contains information about the document, copyright information, author details, or links to related resources.
Example:

```
<footer>
  <p>&copy; 2024 Your Name. All rights reserved.</p>
</footer>
```

Useful Attributes:

❖ **id:** Assigns a unique identifier to the <footer> for linking or styling.

❖ **role:** Specifies the role of the <footer> for accessibility.

Example:

```
<footer id="site-footer" role="contentinfo">
  &copy; 2024 Your Name
</footer>
```

Use Case: Provides concluding information or additional links at the bottom of a webpage.

● Main Tag

Definition: The <main> element represents the main content of the document, excluding headers, footers, or sidebars. It helps in improving the document's accessibility and structure.

Example:

```
<main>
  <h2>Main Content</h2>
  <p>This is the main content of the webpage.</p>
</main>
```

Useful Attributes:

❖ **id:** Assigns a unique identifier to the <main> for linking or styling.

Example:

```
<main id="main-content">
  <h2>Main Content</h2>
  <p>This is the main content of the webpage.</p>
</main>
```

Use Case: Centralizes the primary content of a webpage, enhancing semantic HTML structure.

- ## Nav Tag

Definition: The <nav> element defines a set of navigation links. It's commonly employed to create menus, lists of links, or other navigation structures.
Example:

```
<nav>
  <ul>
    <li><a href="#">Home</a></li>
    <li><a href="#">Products</a></li>
    <!-- Additional navigation items -->
  </ul>
</nav>
```

Useful Attributes:

❖ **id:** Assigns a unique identifier to the <nav> for linking or styling.
Example:

```
<nav id="main-navigation">
  <ul>
    <li><a href="#">Home</a></li>
    <!-- Additional navigation items -->
  </ul>
</nav>
```

Use Case: Group navigation links aid site navigation and accessibility.

- ## Section Tag

Definition: The <section> element represents a generic section of content within a document. It helps in organising content and is often used for thematic grouping.

Example:

```
<section>
  <h2>Section Title</h2>
  <p>This is the content of the section.</p>
</section>
```

Useful Attributes:

❖ **id:** Assigns a unique identifier to the <section> for linking or styling.
Example:

```
<section id="intro-section">
  <h2>Introduction</h2>
  <p>Welcome to our website!</p>
</section>
```

Use Case: Encloses related content to enhance document structure and readability.

- ## Article Tag

Definition: The <article> element represents self-contained content that can be distributed and reused independently. It's suitable for articles, blog posts, comments, or content that can stand alone.

Example:

```
<article>
  <h3>Article Title</h3>
  <p>This is the content of the article.</p>
</article>
```

Useful Attributes:

- **id:** Assigns a unique identifier to the <article> for linking or styling.
  Example:

```
<article id="blog-post">
  <h3>Blog Post Title</h3>
  <p>Content of the blog post goes here.</p>
</article>
```

Use Case: Encapsulates content meant to be distributable and reusable, improving document structure.

- ## Aside Tag

Definition: The <aside> element marks content tangentially related to the content around it. It's often used for sidebars, pull quotes, or information boxes.
Example:

```
<aside>
  <p>Related information goes here.</p>
</aside>
```

Useful Attributes:

- **id:** Assigns a unique identifier to the <aside> for linking or styling.
  Example:

```
<aside id="sidebar">
  <p>Related information goes here.</p>
</aside>
```

Use Case: Contains content that is not the main focus but complements the primary content.

● Details Tag

Definition: The <details> element is used to create a disclosure widget from which the user can obtain additional information. It's often used in conjunction with the <summary> element.
Example:

```
<details>
  <summary>Click to reveal more information</summary>
  <p>Additional details go here.</p>
</details>
```

Useful Attributes:

❖ **id:** Assigns a unique identifier to the <details> for linking or styling.

❖ **open:** Specifies that the <details> element should be open (visible) by default.
Example:

```
<details id="additional-info" open>
  <summary>Click to reveal more information</summary>
  <p>Additional details go here.</p>
</details>
```

Use Case: Creates an expandable/collapsible section for additional details.

● Summary Tag

Definition: The <summary> element is used as the visible heading for a <details> element. It provides a label or title for the content hidden inside a disclosure widget.
Example:

```
<details>
  <summary>Click me</summary>
  <p>Content revealed when clicked.</p>
</details>
```

Useful Attributes:

❖ id: Assigns a unique identifier to the <summary> for linking or styling.
Example:

```
<details>
  <summary id="reveal-info">Click me</summary>
  <p>Content revealed when clicked.</p>
</details>
```

Use Case: Provides a concise label for the content within a disclosure widget.

Additional Elements

● Favicon

Definition: A favicon is a small icon displayed in the browser tab or next to the URL. It is linked using the <link> tag in the <head> section.
Example:

```
<link rel="icon" href="favicon.ico" type="image/x-icon">
```

Useful Attributes:

❖ **rel:** Specifies the relationship between the current document and the linked favicon.
❖ **href:** Specifies the location (URL) of the favicon file.
❖ **type:** Specifies the MIME type of the linked file.
Example:

```
<link rel="icon" href="favicon.png" type="image/png">
```

Use Case: Enhances the website's branding and provides a recognisable icon in browser tabs.

- ## Meta Tag

Definition: The <meta> tag is used to provide metadata about the HTML document, such as character set, description, keywords, and viewport settings.
Example:

```
<meta charset="UTF-8">
<meta name="description" content="A brief description of the webpage.">
```

Useful Attributes:

- **charset:** Specifies the character encoding for the HTML document.
- **name:** Defines the name of a metadata property.
- **content:** Provides the value for the metadata property.
  Example:

```
<meta charset="UTF-8">
<meta name="description" content="A brief description of the webpage.">
```

Use Case: Offers additional information about the document for browsers and search engines.

- ## Code Tag

Definition: The <code> tag is used to define a piece of code within the text. It's commonly used to represent code snippets, variable names, or inline code.
Example:

```
<p>This is a <code>console.log('Hello, World!');</code> within a paragraph.</p>
```

Use Case: Highlights and stylizes code within the text, improving readability.

- ## Figure Tag

Definition: The <figure> element is used to encapsulate media content, such as images or videos, along with their captions. It provides a semantic container for multimedia content and is particularly useful when you want to associate a caption with an image or a video.
Example:

```
<figure>
  <img src="image.jpg" alt="Description">
  <figcaption>Caption goes here</figcaption>
</figure>
```

Useful Attributes:

❖ **id:** Assigns a unique identifier to the <figure> for linking or styling.

Example:

```
<figure id="image-container">
  <img src="image.jpg" alt="Description">
  <figcaption>Caption goes here</figcaption>
</figure>
```

Use Case: Group and semantically associate an image or video with its corresponding caption.

Figcaption Tag

Definition: The <figcaption> element is used to provide a caption or description for the content inside a <figure> element. It allows you to add context or additional information related to the media content within the <figure>.

Example:

```
<figure>
 <img src="image.jpg" alt="Description">
 <figcaption>This is an informative caption.</figcaption>
</figure>
```

Useful Attributes:

❖ **id:** Assigns a unique identifier to the <figcaption> for linking or styling.

Example:

```
<figure>
      <figcaption id="image-caption">This is an informative caption.</figcaption>
</figure>
```

Use Case: Enhances accessibility and provides context by adding captions to images or videos within a <figure> element.

Conclusion:

The fundamental HTML elements and tags are crucial for building well-structured and semantically meaningful web pages. This comprehensive guide overviews various HTML elements commonly used in web development, their attributes and practical examples.

Starting with the building blocks of structure, we explored tags like <div> and <span>, which offer flexibility in organising and styling content. The <header> and <footer> tags define the header and footer sections, contributing to a page's structure and accessibility. The <main> tag centralises the primary content, enhancing document structure.

Navigation is streamlined with the <nav> tag, while thematic grouping is achieved through the <section> tag. The <article> tag encapsulates independent and distributable content, while <aside> complements the main content with related information. The <details> and <summary> tags provide a convenient way to create expandable and collapsible sections.

Additionally, we explored elements like <figure> and <figcaption>, designed for media content and associated captions, contributing to a more accessible and visually appealing presentation.

Including attributes such as id, role, open, and specific examples allows for customisation, styling, scripting, and improved accessibility. The <code> tag, illustrated with a JavaScript snippet, showcases how to highlight and stylise code within text for better readability.

Whether you are a beginner or looking to reinforce your HTML knowledge, this guide is a valuable resource for mastering essential HTML elements and tags, facilitating the creation of well-structured and semantically rich web content. As you delve deeper into web development, the principles outlined here will form the foundation for building dynamic and visually appealing websites.

References

1. https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_flow_layout/Block_and_inline_layout_in_normal_flow
2. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/div
3. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/span
4. https://developer.mozilla.org/en-US/docs/Glossary/Semantics#semantics_in_html
5. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/meta
6. https://developer.mozilla.org/en-US/docs/Web/HTML/Element/code