VEMU INSTITUTE OF TECHNOLOGY

(Approved by AICTE, New Delhi & Affiliated to JNTUA, Anantapur)
Accredited by NAAC & NBA , Recognized by UGC 12(B) & 2(F)
P.Kotha Kota – 517 112, Pakala (M), Chittoor Dist., ANDHRA PRADESH, INDIA

20A05304 – WEB APPLICATION DEVELOPMENT

Nageswara Rao P

ASSOCIATE PROFESSOR, DEPT.OF CSE

Module-1:

HTML: What is a browser? What is HTML? Elements and Tags, Basic HTML5 structure, Metadata, <title>, Adding favicon, Comments, headings.

Task: Create a Basic HTML document

Web Browser:

A web browser is a client machine software that is used to access website and web content available over the Internet or World Wide Web. The most popular types of Web browsers are:

- Internet Explorer
- Mozilla Firefox
- Google Chrome
- Opera
- Apple Safari
- UC Browser
- Netscape Navigator
- Mosaic Web Browser
- Microsoft Edge Browser

A software application used for accessing the information on the World Wide Web is called a Web Browser. The web browser **retrieves the data from a web server** when a user requests any information and then shows the webpage on the user's computer.

HTML:

- ✓ HTML stands for Hyper Text Markup Language.
- ✓ HTML is used to create web pages and web applications.
- ✓ HTML is widely used language on the web.
- ✓ We can create a static website by HTML only.
- ✓ Technically, HTML is a Markup language rather than a programming language.

HTML is an acronym which stands for **Hyper Text Markup Language** which is used for creating web pages and web applications. HTML is derived from SGML (Standard Generalized Markup Language), which is a parent for all other markup languages.

Hyper Text: HyperText simply means "Text within Text." A text has a link within it, is a hypertext. Whenever you click on a link which brings you to a new webpage, you have clicked on a hypertext. HyperText is a way to link two or more web pages (HTML documents) with each other.

Markup language: A markup language is a computer language that is used to apply layout and formatting conventions to a text document. Markup language makes text more interactive and dynamic. It can turn text into images, tables, links, etc.

Web Page: A web page is a document which is commonly written in HTML and translated by a web browser. A web page can be identified by entering an URL. A Web page can be of the static or dynamic type. **With the help of HTML only, we can create static web pages**.

Hence, HTML is a markup language which is used for creating attractive web pages with the help of styling, and which looks in a nice format on a web browser. An HTML document is made of many HTML tags and each HTML tag contains different content.

Example 1.1:

```
<!DOCTYPE>
<html>
<head>
<title>Web page title</title>
</head>
<body>
<h1>Write Your First Heading</h1>
<p>Write Your First Paragraph. </p>
</body>
</html>
```

Description of HTML Example

<!DOCTYPE>: It defines the document type or it instruct the browser about the version of HTML.

<html >: This tag informs the browser that it is an HTML document. Text between html tag describes the web document. It is a container for all other elements of HTML except <!DOCTYPE>

<head>: It should be the first element inside the <html> element, which contains the metadata (information about the document). It must be closed before the body tag opens.

<title>: As its name suggested, it is used to add title of that HTML page which appears at the top of the browser window. It must be placed inside the head tag and should close immediately. (Optional)

<body>: Text between body tag describes the body content of the page that is visible to the end user. This tag contains the main content of the HTML document.

<h1> : Text between <h1> tag describes the first level heading of the webpage.

<p> : Text between <p> tag describes the paragraph of the webpage.

Brief History of HTML

In the late 1980's , a physicist, Tim Berners-Lee who was a contractor at CERN, proposed a system for CERN researchers. In 1989, he wrote a memo proposing an internet-based hypertext system.

Tim Berners-Lee is known as the father of HTML. The first available description of HTML was a document called "HTML Tags" proposed by Tim in late 1991. The latest version of HTML is HTML5,

HTML Versions

HTML 1.0: The first version of HTML was 1.0, which was the barebones version of HTML language, and it was released in1991.

HTML 2.0: This was the next version which was released in 1995, and it was standard language version for website design. HTML 2.0 was able to support extra features such as form-based file upload, form elements such as text box, option button, etc.

HTML 3.2: HTML 3.2 version was published by W3C in early 1997. This version was capable of creating tables and providing support for extra options for form elements. It can also support a web page with complex mathematical equations. It became an official standard for any browser till January 1997. Today it is practically supported by most of the browsers.

HTML 4.01: HTML 4.01 version was released on December 1999, and it is a very stable version of HTML language. This version is the current official standard, and it provides added support for stylesheets (CSS) and scripting ability for various multimedia elements.

HTML5: HTML5 is the newest version of HyperText Markup language. The first draft of this version was announced in January 2008. There are two major organizations one is W3C (World Wide Web Consortium), and another one is WHATWG (Web Hypertext Application Technology Working Group) which are involved in the development of HTML 5 version, and still, it is under development.

Features of HTML

1) It is a very easy and simple language. It can be easily understood and modified.

2) It is very easy to make an effective presentation with HTML because it has a lot of formatting tags.

3) It is a markup language, so it provides a flexible way to design web pages along with the text.

4) It facilitates programmers to add a link on the web pages (by html anchor tag), so it enhances the interest of browsing of the user.

5) It is platform-independent because it can be displayed on any platform like Windows, Linux, and Macintosh, etc.

6) It facilitates the programmer to add Graphics, Videos, and Sound to the web pages which makes it more attractive and interactive.

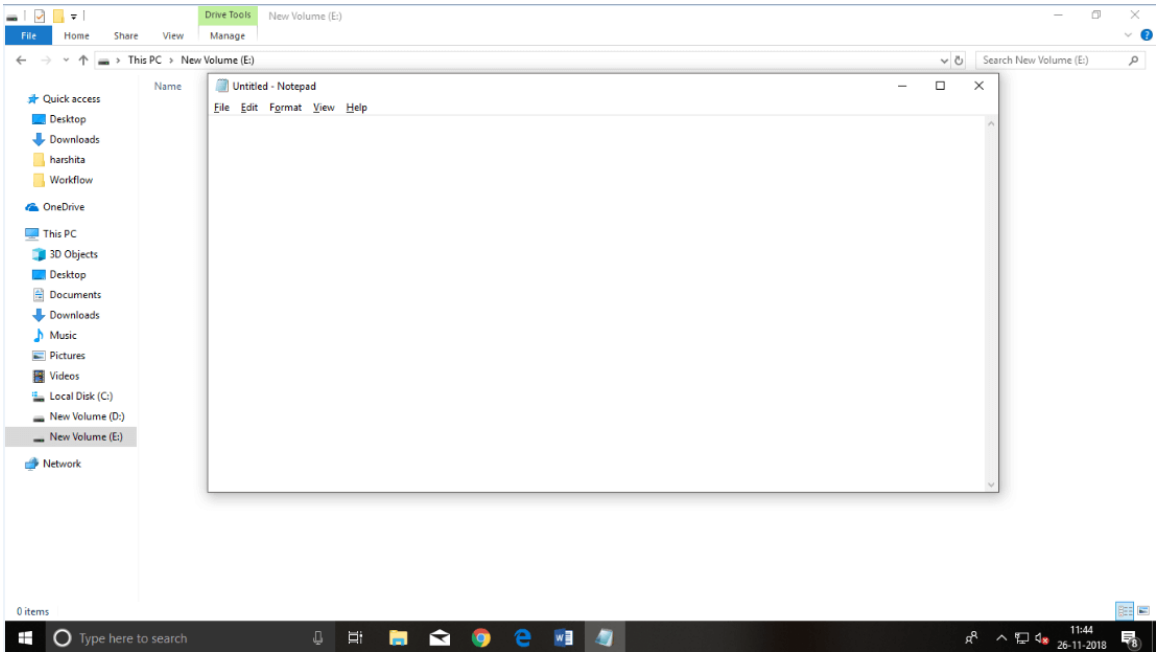
7) HTML is a case-insensitive language, which means we can use tags either in lower-case or upper-case.

HTML text Editors

- An HTML file is a text file, so to create an HTML file we can use any text editors.

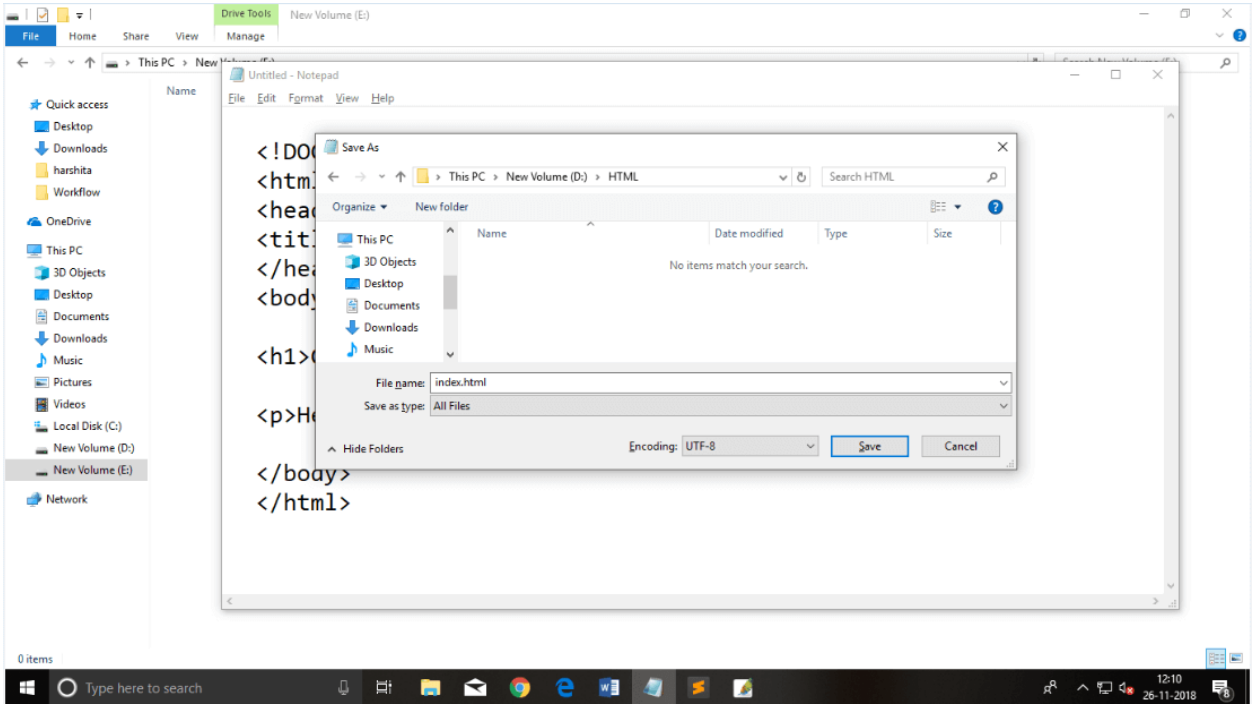
- Text editors are the programs which allow editing in a written text, hence to create a web page we need to write our code in some text editor.
- There are various types of text editors available which you can directly download, but for a beginner, the best text editor is Notepad (Windows) or TextEdit (Mac).
- After learning the basics, you can easily use other professional text editors which are, **Notepad++, Sublime Text, Vim, etc**.

HTML code with Notepad.

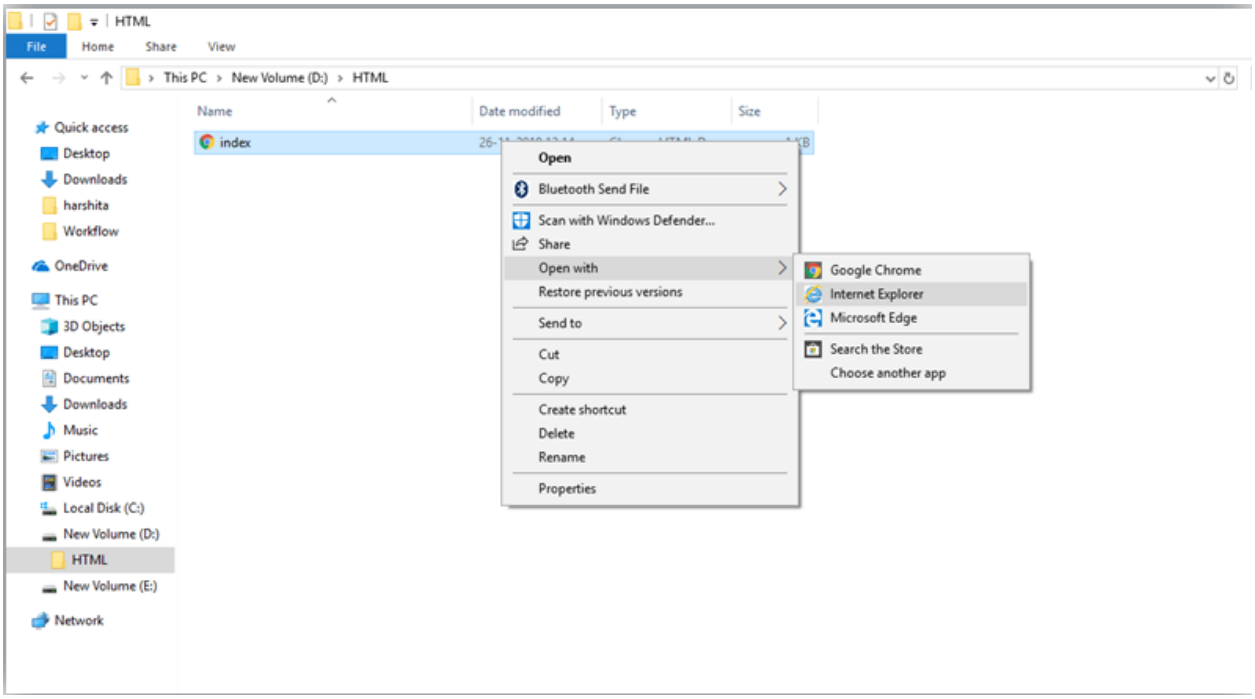


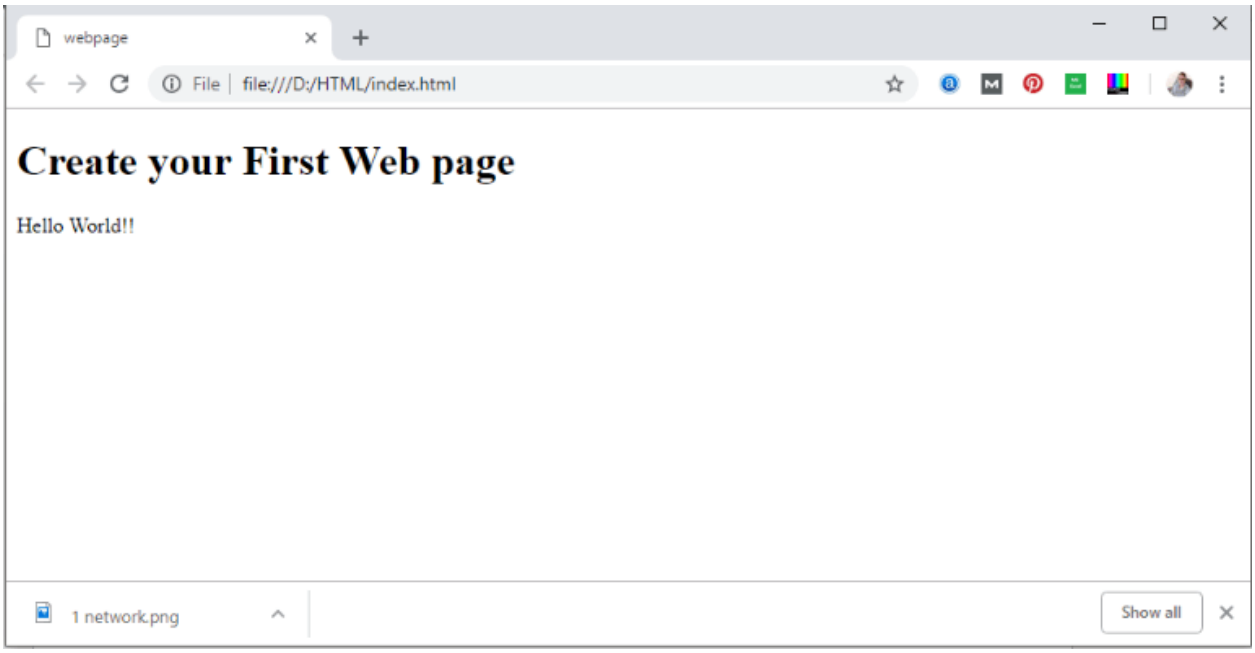
Step 2: Write code in HTML

Step 3: Save the HTML file with .htm or .html extension.



Step 4: Open the HTML page in your web browser.



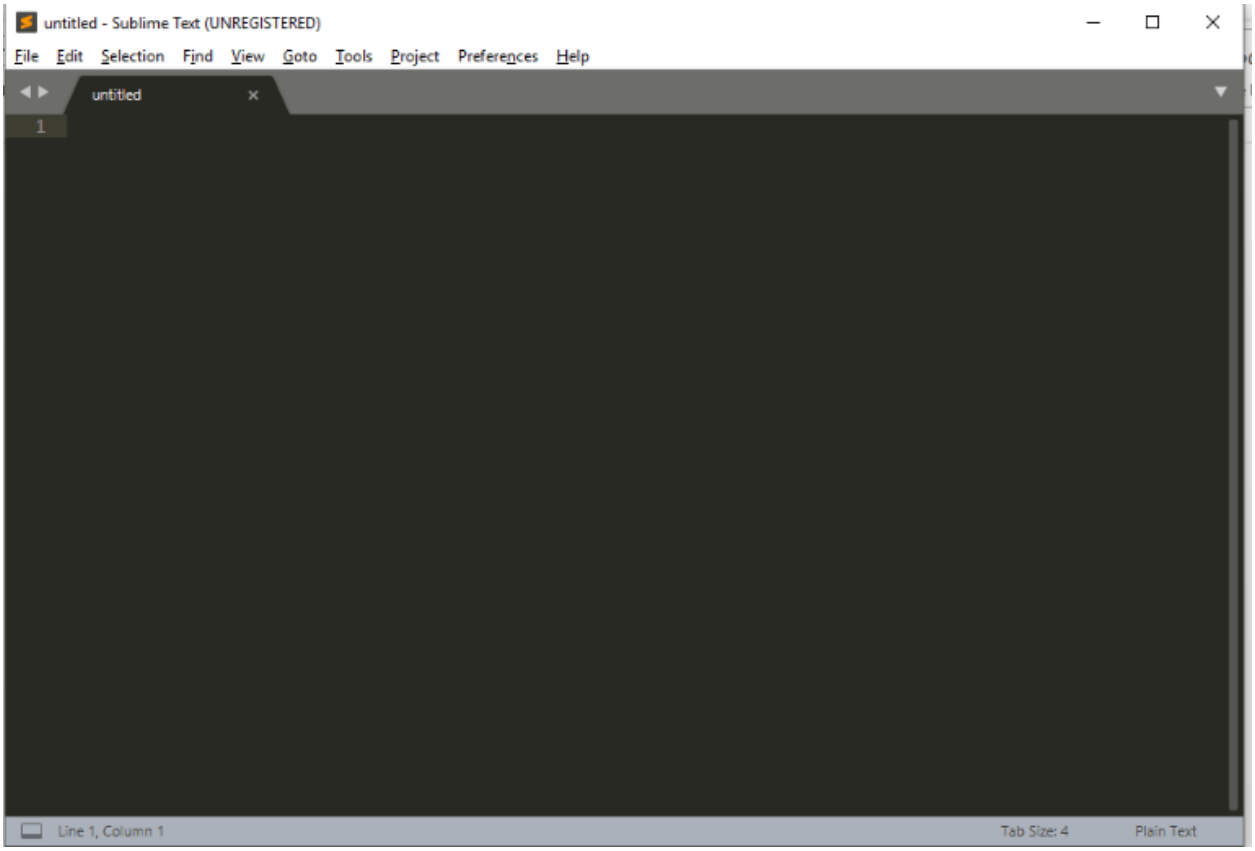


HTML code with Sublime Text-editor.

When you will learn the basics of HTML, then you can use some professional text editors, which will help you to write an efficient and fast code. So, to use **Sublime Text editors**, first it needs to download and install from internet. You can easily download it from this **https://www.sublimetext.com/download link** and can install in your PC. When installation of Sublime text editor done then you can follow the simple steps to use it:

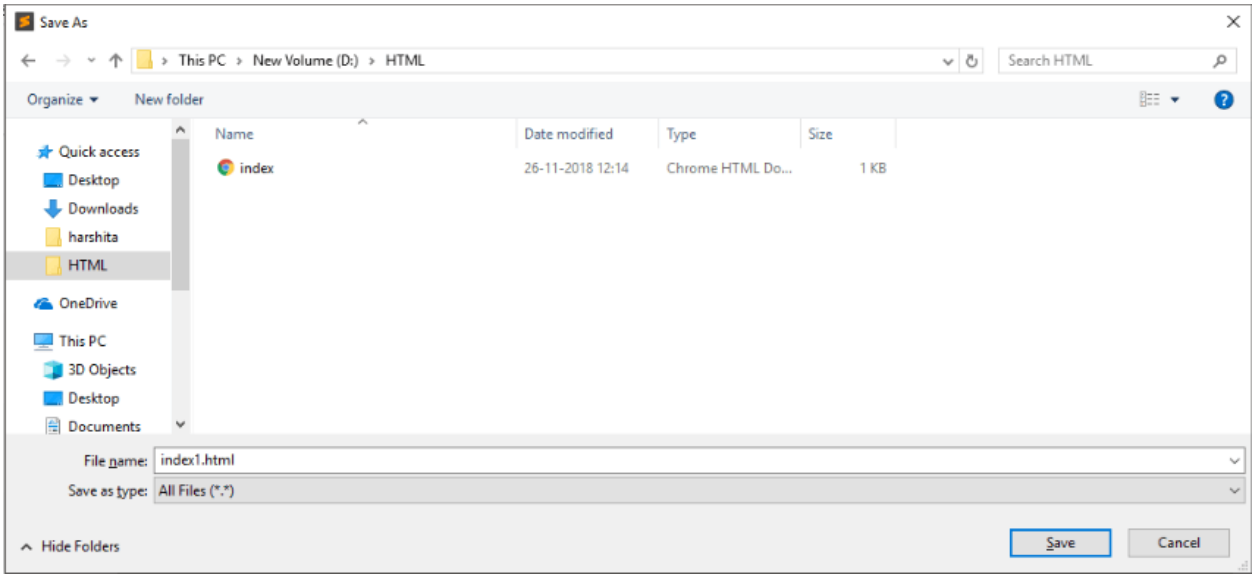
Step 1: Open Sublime Text editor (Windows 8 or 10):

To open Sublime Text editor, go to **Start screen --> type Sublime Text--> Open** it. To open a new page press **CTRL+N.**

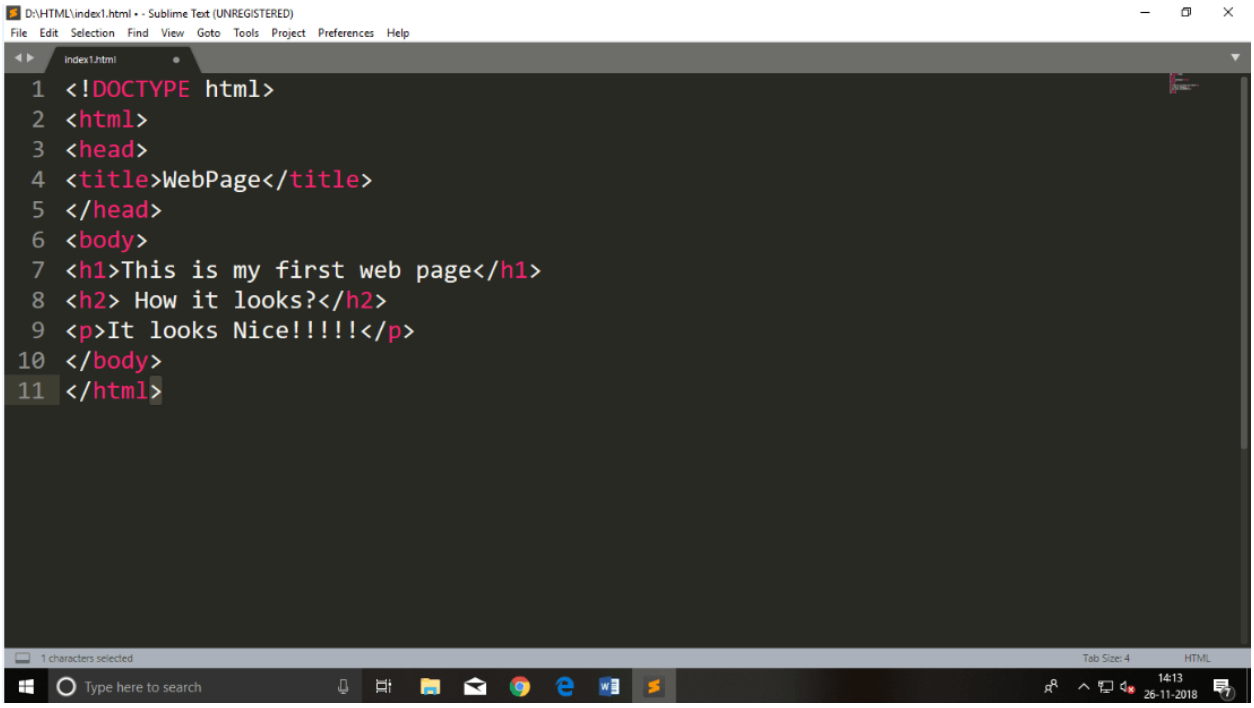


Step 2: Save the page before writing any code.

To save your page in Sublime Text press Ctrl+S or go to File option --> save, to save a file use extension .htm or .html. We recommend to save the file first then write the code because after saving the page sublime text editor will give you suggestions to write code.

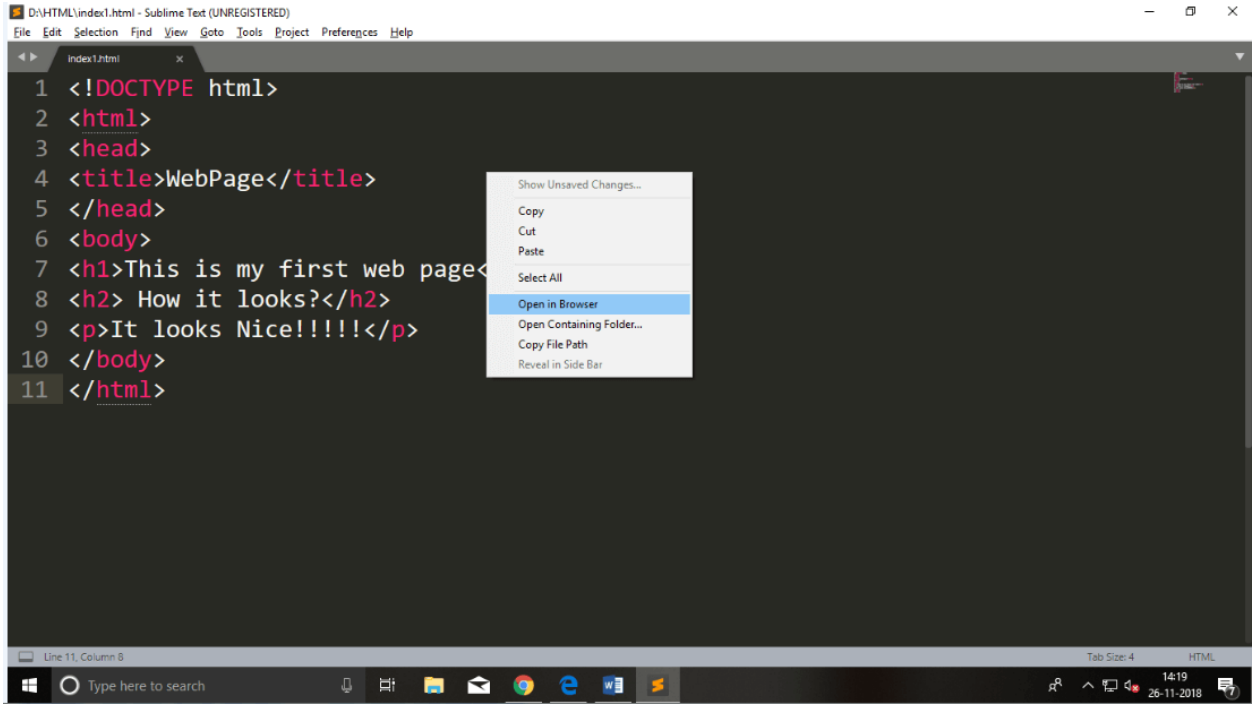


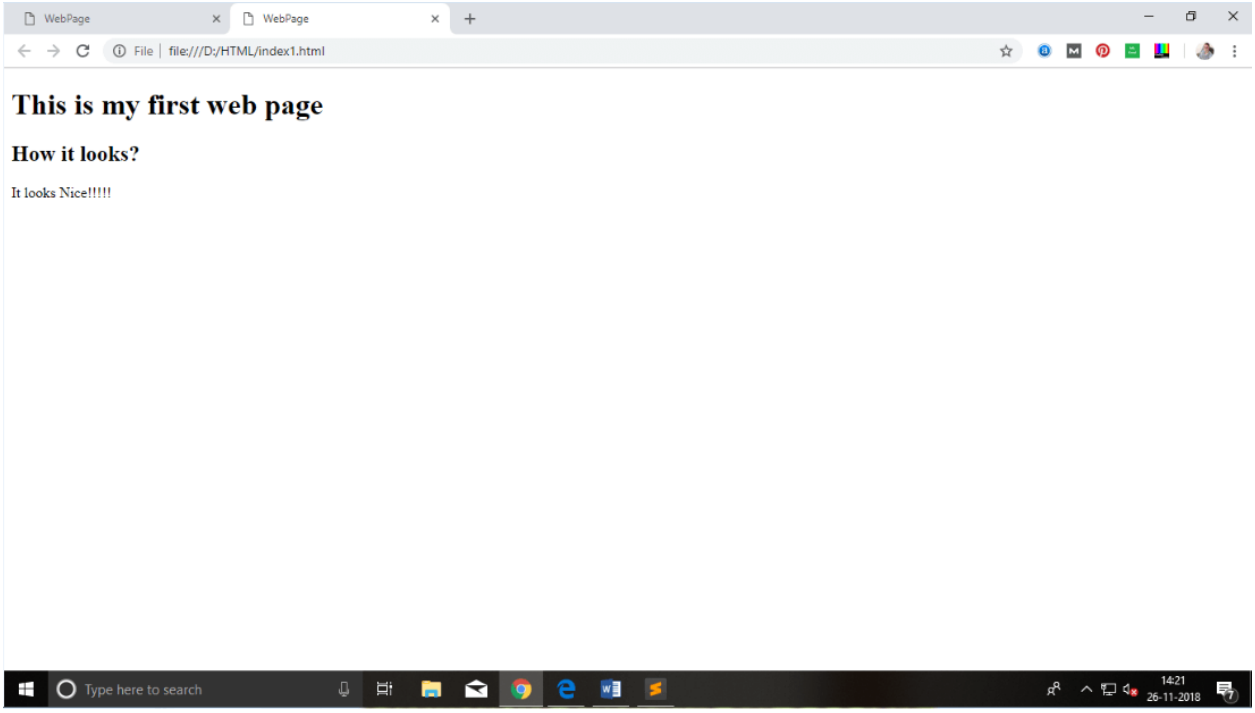
Step 3: Write the code in Sublime Text editor



Step 4: Open the HTML page in your Browser

To execute or open this page in Web browser just right click by mouse on sublime text page and click on Open in Browser.





Building blocks of HTML

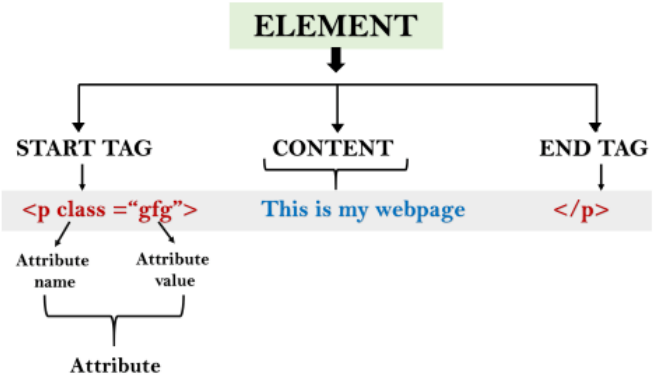
An HTML document consist of its basic building blocks which are:

- **Tags:** An HTML tag surrounds the content and apply meaning to it. It is written between < and > brackets.
- **Attribute:** An attribute in HTML provides extra information about the element, and it is applied within the start tag. An HTML attribute contains two fields: name & value.

Syntax

<tag name attribute_name= " attr_value"> content </ tag name>

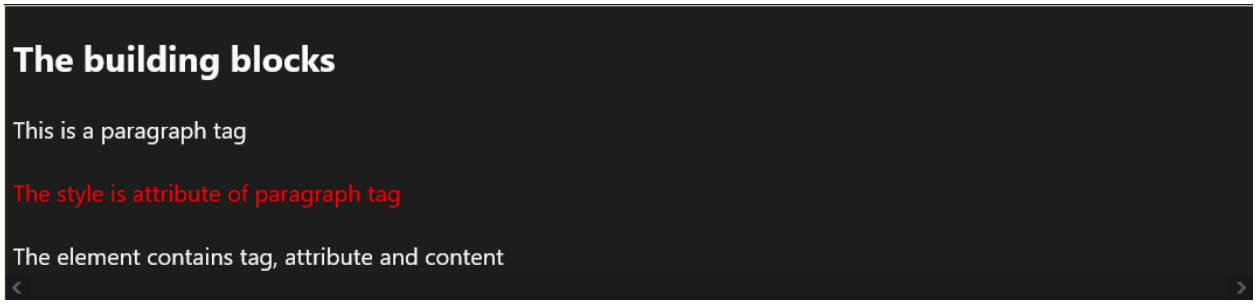
Elements: An HTML element is an individual component of an HTML file. In an HTML file, everything written within tags are termed as HTML elements.



Example 1.2

```
<!DOCTYPE html>
<html>
  <head>
    <title>The basic building blocks of HTML</title>
 </head>
  <body>
      <h2>The building blocks</h2>
      <p>This is a paragraph tag</p>
      <p style="color: red">The style is attribute of paragraph tag</p>
      <span>The element contains tag, attribute and content</span>
  </body>  </html>
```

Output:



HTML Tags

HTML tags are like keywords which defines that how web browser will format and display the content. With the help of tags, a web browser can distinguish between an HTML content and a simple content. HTML tags contain three main parts: opening tag, content and closing tag. But some HTML tags are unclosed tags.

When a web browser reads an HTML document, browser reads it from top to bottom and left to right. HTML tags are used to create HTML documents and render their properties. Each HTML tags have different properties.

An HTML file must have some essential tags so that web browser can differentiate between a simple text and HTML text. You can use as many tags you want as per your code requirement.

- All HTML tags must be enclosed within < > these brackets.
- Every tag in HTML performs different tasks.
- If you have used an open tag <tag>, then you must use a close tag </tag> (except some tags)

Syntax

<tag> content </tag>

HTML Tag Examples

 Paragraph Tag
<h2> Heading Tag </h2>
 Bold Tag
 Italic Tag
<u> Underline Tag</u>

Unclosed HTML Tags

Some HTML tags are not closed, for example br and hr.

**
 Tag**: br stands for break line, it breaks the line of the code.

<hr> Tag: hr stands for Horizontal Rule. This tag is used to put a line across the webpage.

HTML Meta Tags

DOCTYPE, title, link, meta and style

HTML Text Tags

<p>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, , , <abbr>, <acronym>, <address>, <bdo>, <blockquote>, <cite>, <q>, <code>, <ins>, <del>, <dfn>, <kbd>, <pre>, <samp>, <var> and

HTML Link Tags

<a> and <base>

HTML Image and Object Tags

<img>, <area>, <map>, <param> and <object>

HTML List Tags

, <ol>, <li>, <dl>, <dt> and <dd>

HTML Table Tags

table, tr, td, th, tbody, thead, tfoot, col, colgroup and caption

HTML Form Tags

form, input, textarea, select, option, optgroup, button, label, fieldset and legend

---

HTML Scripting Tags

script and noscript

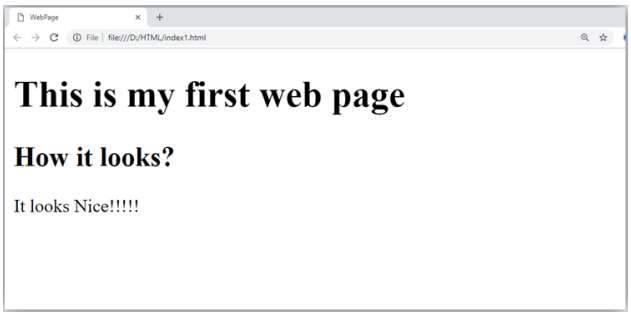
HTML Elements

An HTML file is made of elements. These elements are responsible for creating web pages and define content in that webpage. **An element in HTML usually consists of a start tag <tag name>, close tag </tag name> and content inserted between them**. Technically, an element is a collection of start tag, attributes, end tag, content between them**.**

Example 1.3

```
<!DOCTYPE html>
<html>
<head>
   <title>WebPage</title>
</head>
<body>
  <h1>This is my first web page</h1>
   <h2> How it looks?</h2>
    <p>It looks Nice!!!!!</p>
</body>
</html>
```

Output:



- All the content written between body elements are visible on web page.

Void element: All the elements in HTML do not require to have start tag and end tag, some elements does not have content and end tag such elements are known as Void elements or empty elements. **These elements are also called as unpaired tag**.

**Some Void elements are
 (represents a line break) , <hr>(represents a horizontal line), etc.**

Nested HTML Elements: HTML can be nested, which means an element can contain another element.

Basic HTML5 Structure:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <title></title>
    <meta name="description" content="">
  </head>
<body>
</body>
</html>
```

The format is much simplified compared to the previous standard with especially a minimal doctype.

The DOCTYPE

The document type has been introduced to mark the difference between old browsers which followed the usual format in the 90s and newer browsers that are closer to the HTML specifications 3 then 4 and 5.

The language and the *lang* attribute

The lang attribute is not for browsers, but rather to the processing tools that must understand contents according to their language.

```
<p lang="fr">Citation en Français</p>
```

Head

The tag contains several types of elements:

- Encoding with the meta tag or charset.
- The title of the page.
- Links with the link tag.
- And other indications by metas.

Encoding

The most common tag has the following form:

<meta http-equiv="Content-Type" content="text/html” charset=”utf-8">

It defines the content type, its format that is generally text/html and its encoding, usually the utf8 charset.

Links

Many links can be specified in the header. Some are essential to the browser as the link to a style sheet or the RSS feed, or the favicon.

Favicon

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

Stylesheet

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

RSS or Atom

```
<link rel="alternate" type="application/rss+xml" href="" title="">
```

HTML 5 introduces multiple tags to help represent the usual structure of documents.

<header>

Contains an introduction to a part or the whole page.

<footer>

Contains information that are usually placed at the end of a section. We can put it at the end of a section or page, but also anywhere in the section.

For example it contains a link on the index, which can be placed below the title.

<section>

Sections mark out parts of content. It is then up to the webmaster to associate a style sheet or using them dynamically in scripts.

Very basically, we can frame a section with a border, or separate it from the above by a space.

<hgroup>

Represents the header of a section. The <header> tag may contain at the beginning a <hgroup> tag.

<nav>

This container is intended to enclose a group of links.

<article>

Denotes a typical content that can be found on different pages, or even different sites. This can be a forum post, a newspaper article and this is for tools to extract more easily the content (by separating the unnecessary data such as navigation menus).

<aside>

To delimit something separate to the actual content, and may define a sidebar.

<address>

Contains contact information, eg name of the author.

<mark>

Used to mark a portion of a text, highlight, as the old but more general.

Meta Data:

HTML provides us <meta> tag which describes data about data.

- The <meta> tag defines metadata about an HTML document. Metadata is data (information) about data.

- <meta> tags always go inside the <head> element, and are typically used to specify character set, page description, keywords, author of the document, and viewport settings.
- Metadata will not be displayed on the page, but is machine parsable.
- Metadata is used by browsers (how to display content or reload page), search engines (keywords), and other web services.

Define keywords for search engines:

```
<meta name="keywords" content="HTML, CSS, JavaScript">
```

Define a description of your web page:

```
<meta name="description" content="Free Web tutorials for HTML and CSS">
```

Define the author of a page:

```
<meta name="author" content="John Doe">
```

Refresh document every 30 seconds:

```
<meta http-equiv="refresh" content="30">
```

Describe metadata within an HTML document:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <meta name="description" content="Free Web tutorials">
  <meta name="keywords" content="HTML,CSS,XML,JavaScript">
  <meta name="author" content="John Doe">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<p>All meta information goes in the head section...</p>
</body>
</html>
```

HTML <title> Tag

The <title> tag defines the title of the document. The title must be text-only, and it is shown in the browser's title bar or in the page's tab.

The <title> tag is required in HTML documents!

The contents of a page title is very important for search engine optimization (SEO)! The page title is used by search engine algorithms to decide the order when listing pages in search results.

The <title> element:

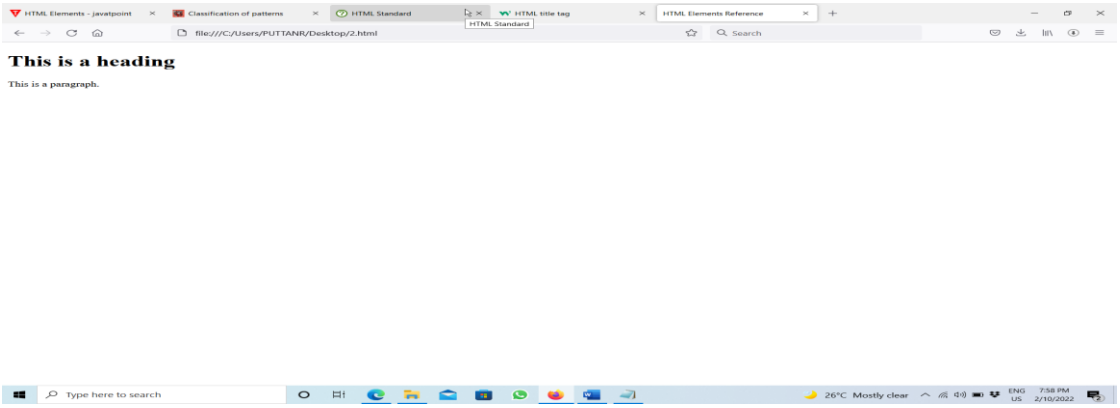
- ✓ defines a title in the browser toolbar
- ✓ provides a title for the page when it is added to favorites
- ✓ displays a title for the page in search-engine results

Example 1.4

```
<!DOCTYPE html>
<html>
<head>
  <title>HTML Elements Reference</title>
</head>
<body>

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

```
</body>
</html>
```

Output:



Adding Favicon:

A **favicon** is a small file containing the one or more icons which are used to represent the website or a blog. It is also known as a tab icon, website icon, URL icon, or a bookmark icon. This icon is actually displayed on the address bar, browser's tab, browser history, bookmark bar, etc. The image of a favicon is in **.ico** file format. There are various file formats, but **.ico** format is supported by all the browsers.

How to Create a Favicon

Following are the steps for creating the favicon:

1. Click on the following URL, to create the favicon: https://www.favicon.cc/
2. Once the favicon is successfully created, we can download the favicon by clicking on the **download favicon** option.
3. 3After the downloading, a favicon with the name **favicon.ico** is available in the file system drive.

How to insert the Favicon in HTML file

1. Open the HTML file. Then use the following syntax to insert the favicon in the HTML file.

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

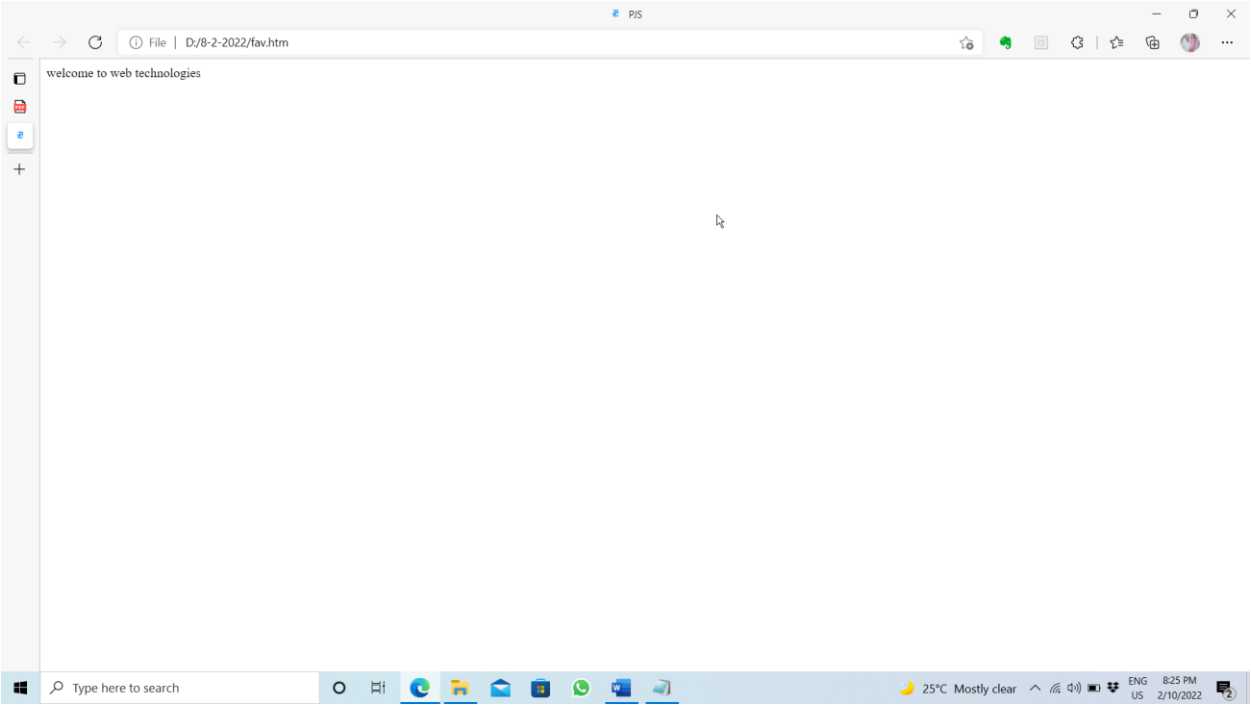
2. We have to use the above syntax in the tag of our html file. Then save the file.

3. Now. Open the HTML file in any browser. We can see the icon on the web page.

Example 1.4

```
<html>
<head>
<link rel="shortcut icon" href="favicon.ico" type="image/x-icon">
</head>
<title> Example of favicon</title>
<body>
<br>
<br>
<p align="center">
<img src="download.jpg">
</p>
</body>
</html>
```

Output:



Comments:

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

HTML Comment Tag

You can add comments to your HTML source by using the following syntax:

<!-- Write your comments here -->

Notice that there is an exclamation point (!) in the start tag, but not in the end tag.

Add Comments

With comments you can place notifications and reminders in your HTML code:

Example

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

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

<!-- Remember to add more information here -->
```

Example 1.5:

```
<!DOCTYPE html>
<html>
<body>
<p>This is a paragraph.</p>
<!-- <p>This is another paragraph </p> -->
<p>This is a paragraph too.</p>
</body>
</html>
```

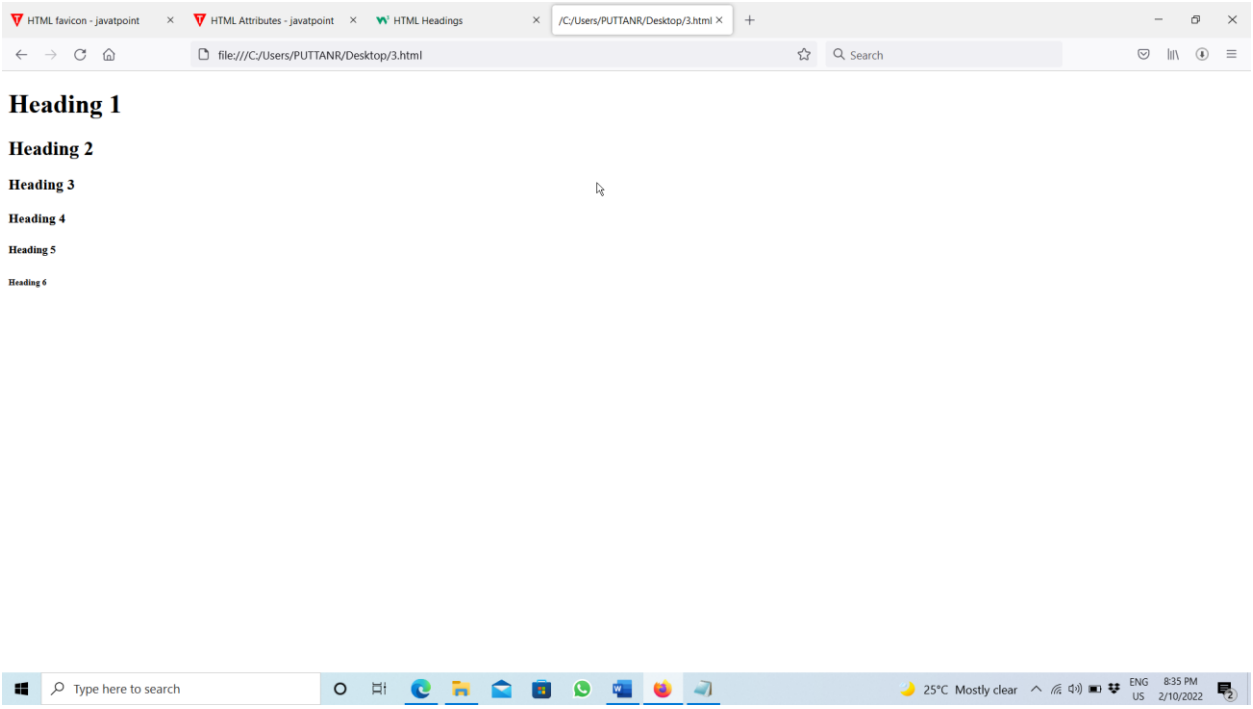
HTML Headings

- HTML headings are titles or subtitles that you want to display on a webpage.
- HTML headings are defined with the <h1> to <h6> tags.
- <h1> defines the most important heading. <h6> defines the least important heading.
- <h1> headings should be used for main headings, followed by <h2> headings, then the less important <h3>, and so on.

Example 1.6:

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

Output:



HTML Formatting Elements

Formatting elements were designed to display special types of text:

<b> - Bold text
 - Important text
<i> - Italic text
 - Emphasized text
<mark> - Marked text
<small> - Smaller text
<del> - Deleted text
<ins> - Inserted text
<sub> - Subscript text
<sup> - Superscript text

HTML Attributes

HTML attributes provide additional information about HTML elements.

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

The href Attribute

The <a> tag defines a hyperlink. The href attribute specifies the URL of the page the link goes to:

Example

```
<a href="http://www.vemu.org">Visit W3Schools</a>
```

The src Attribute

The <img> tag is used to embed an image in an HTML page. The src attribute specifies the path to the image to be displayed:

<img src="img_girl.jpg">

The width and height Attributes

The <img> tag should also contain the width and height attributes, which specifies the width and height of the image (in pixels):

Example

```
<img src="img_girl.jpg" width="500" height="600">
```

The alt Attribute

The required alt attribute for the <img> tag specifies an alternate text for an image, if the image for some reason cannot be displayed.

```
<img src="img_girl.jpg" alt="PJS Photo">
```

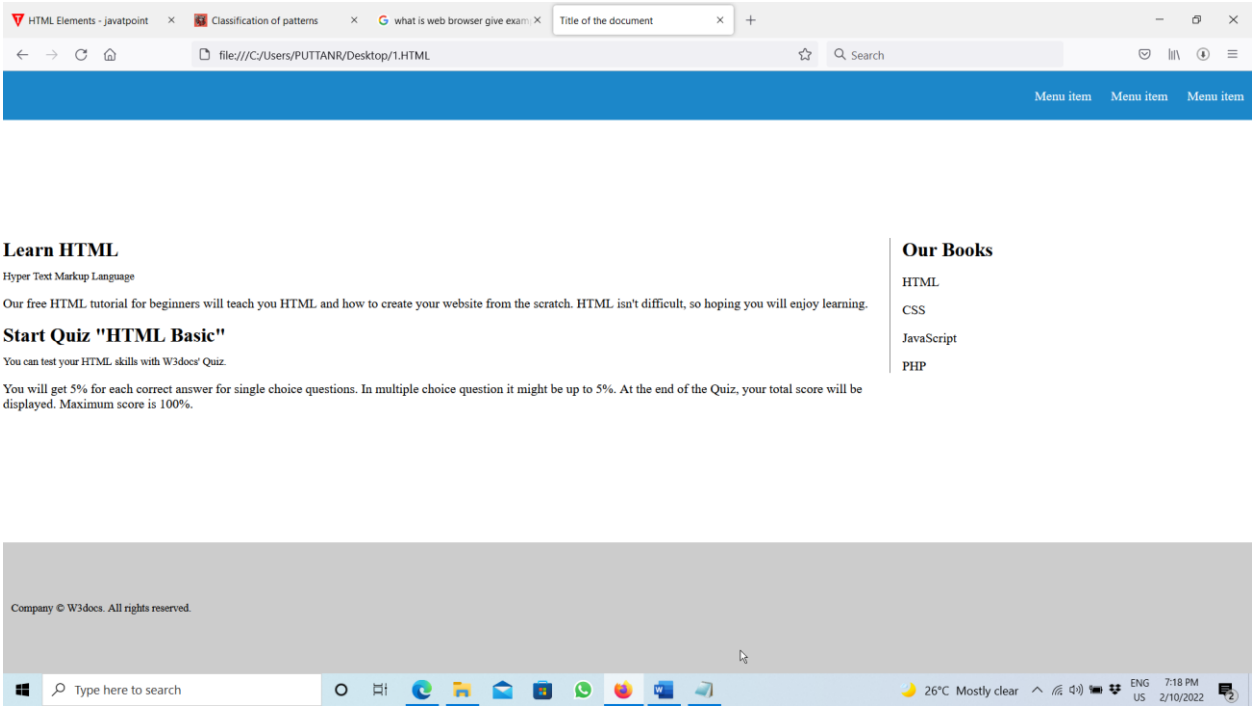
Task: Creating a Basic HTML Document

```
<!DOCTYPE html>
<html>
  <head>
    <title>Title of the document</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <style>
      html,
      body {
        height: 100%;
      }
      body {
        display: flex;
        flex-wrap: wrap;
        margin: 0;
      }
      .header-menu,
      footer {
        display: flex;
        align-items: center;
        width: 100%;
      }
      .header-menu {
        justify-content: flex-end;
        height: 60px;
        background: #1c87c9;
        color: #fff;
      }
      h2 {
        margin: 0 0 8px;
      }
      ul li {
        display: inline-block;
        padding: 0 10px;
        list-style: none;
      }
      aside {
        flex: 0.4;
        height: 165px;
```

```
      padding-left: 15px;
      border-left: 1px solid #666;
    }
    section {
      flex: 1;
      padding-right: 15px;
    }
    footer {
      padding: 0 10px;
      background: #ccc;
    }
  </style>
</head>
<body>
  <header class="header-menu">
    <nav>
      <ul>
        <li>Menu item1</li>
        <li>Menu item2</li>
        <li>Menu item3</li>
      </ul>
    </nav>
  </header>
  <section>
    <article>
      <header>
        <h2>Learn HTML</h2>
        <small>Hyper Text Markup Language</small>
      </header>
      <p>Our free HTML tutorial for beginners will teach you HTML and how to create
your website from the scratch. HTML isn't difficult, so hoping you will enjoy
learning.</p>
    </article>
    <article>
      <header>
        <h2>Start Quiz "HTML Basic"</h2>
        <small>You can test your HTML skills with W3docs' Quiz.</small>
      </header>
      <p>You will get 5% for each correct answer for single choice questions. In
multiple choice question it might be up to 5%. At the end of the Quiz, your total score
will be displayed. Maximum score is 100%.</p>
    </article>
  </section>
  <aside>
    <h2>Our Books</h2>
    <p>HTML</p>
    <p>CSS</p>
    <p>JavaScript</p>
    <p>PHP</p>
  </aside>
  <footer>
    <small>Company © W3docs. All rights reserved. </small>
  </footer>
</body>
```

</html>

Output:



Module - 2:

HTML (continued): Block-Level Elements & Inline Elements, Links (Understand Absolute vs Relative paths), Lists, Images, iframe (embed youtube video)

Task: Create your Profile Page

HTML Block-Level Elements

Every HTML element has a default display value, depending on what type of element it is. There are two display values: block and inline.

Block-level Elements

A block-level element always starts on a new line, and the browsers automatically add some space (a margin) before and after the element.

A block-level element always takes up the full width available (stretches out to the left and right as far as it can).

Two commonly used block elements are: <p> and .

The <p> element defines a paragraph in an HTML document.

The element defines a division or a section in an HTML document.

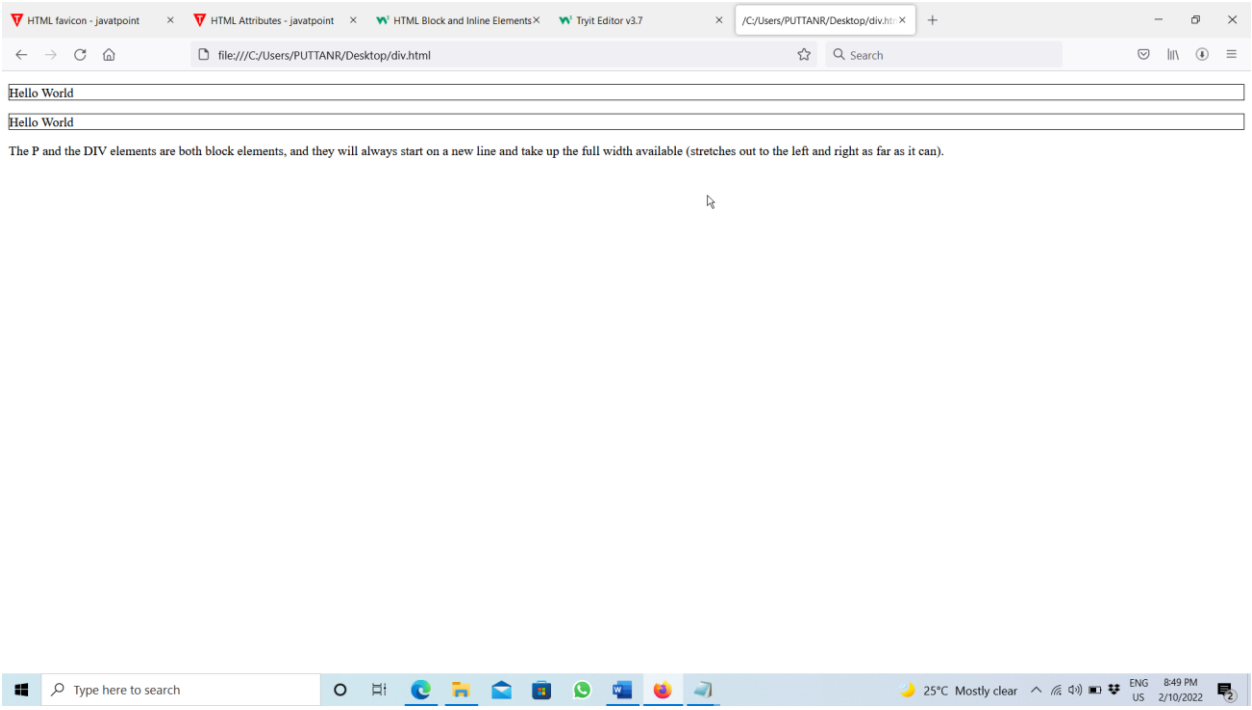
The <p> element is a block-level element.

The element is a block-level element.

Example 2.1

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

<p style="border: 1px solid black">Hello World</p>
<div style="border: 1px solid black">Hello World</div>

<p>The P and the DIV elements are both block elements, and they will always start on a new line and take up the full width available (stretches out to the left and right as far as it can).</p>

</body></html>
```

Output:



Here are the block-level elements in HTML:

<address>	<article>	<aside>	<blockquote>	<canvas>	<dd>		<dl>
<dt>	<fieldset>	<figcaption>	<figure>	<footer>	<form>	<h1>-<h6>	<header>
<hr>		<main>	<nav>	<noscript>		<p>	<pre>
<section>	<table>	<tfoot>		<video>			

Inline Elements

An inline element does not start on a new line.

An inline element only takes up as much width as necessary.

This is a element inside a paragraph.

Example 2.2

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

<p>This is an inline span <span style="border: 1px solid black">Hello World</span> element
inside a paragraph.</p>

<p>The SPAN element is an inline element, and will not start on a new line and only takes up as
much width as necessary.</p>

</body>
</html>
```

Output:

This is an inline span Hello World element inside a paragraph.

The SPAN element is an inline element, and will not start on a new line and only takes up as much width as necessary.

Here are the inline elements in HTML:

<a>	<abbr>	<acronym>	<b>	<bdo>	<big>	 	<button>
<cite>	<code>	<dfn>		<i>	<img>	<input>	<kbd>
<label>	<map>	<object>	<output>	<q>	<samp>	<script>	<select>
<small>			<sub>	<sup>	<textarea>	<time>	<tt>
<var>							

An inline element cannot contain a block-level element!

The Element

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

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

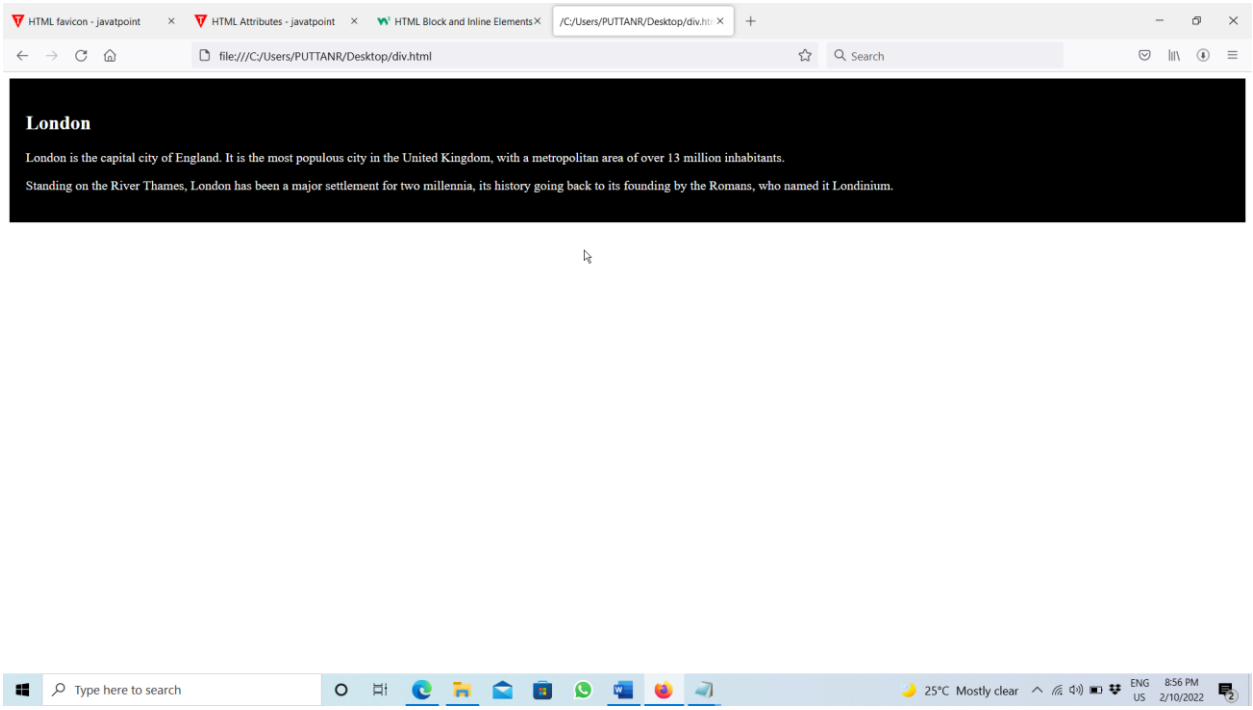
When used together with CSS, the element can be used to style blocks of content

Example 2.3

```
<!DOCTYPE html>
<html>
<body>
<div style="background-color:black;color:white;padding:20px;">
  <h2>London</h2>
```

```
<p>London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>

<p>Standing on the River Thames, London has been a major settlement for two millennia, its history going back to its founding by the Romans, who named it Londinium.</p>

</div>

</body>

</html>
```

Output:



The Element

The element is an inline container used to mark up a part of a text, or a part of a document.

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

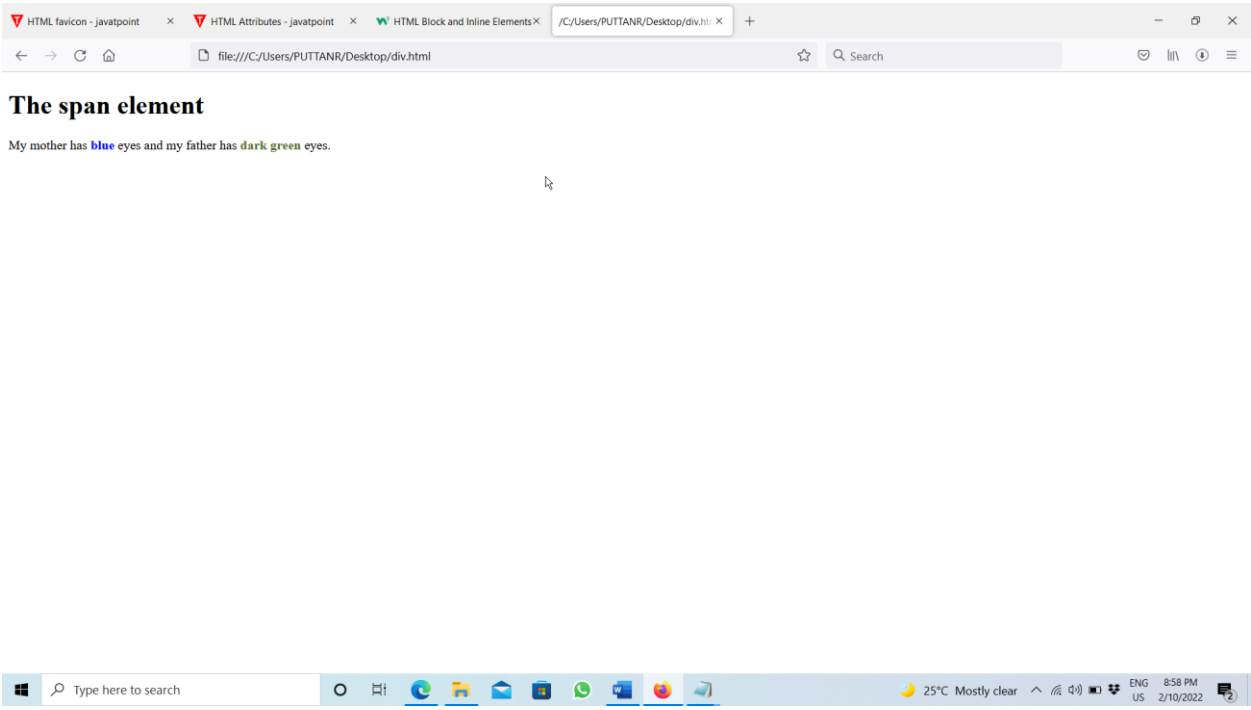
When used together with CSS, the element can be used to style parts of the text:

Example 2.4

```
<!DOCTYPE html>

<html>

<body>

<h1>The span element</h1>
```

```
<p>My mother has <span style="color:blue;font-weight:bold">blue</span> eyes and
my father has <span style="color:darkolivegreen;font-weight:bold">dark green</span>
eyes.</p>

</body>

</html>
```

Output:



Links (Understand Absolute paths & Relative paths)

- Links are found in nearly all web pages. Links allow users to click their way from page to page.
- HTML Links – Hyperlinks.
- You can click on a link and jump to another document.
- When you move the mouse over a link, the mouse arrow will turn into a little hand.

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

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

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

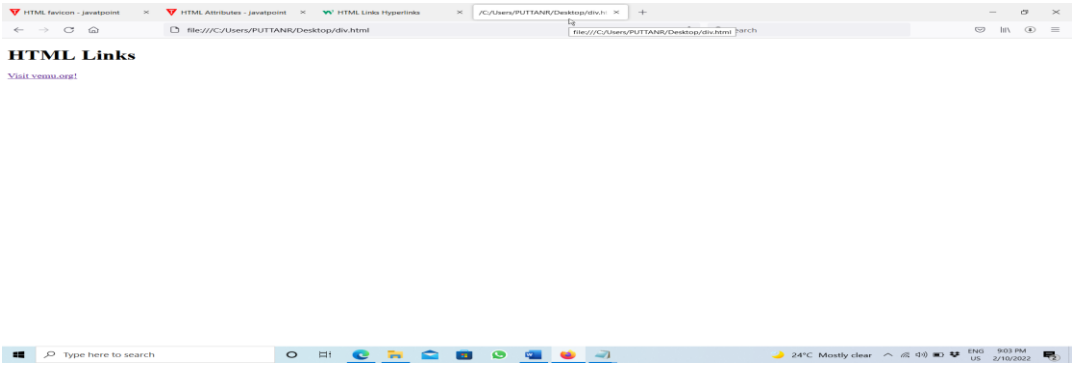
The *link text* is the part that will be visible to the reader.

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

Example 2.5

```
<!DOCTYPE html>
<html>
<body>
<h1>HTML Links</h1>
<p><a href="https://www.vemu.org/">Visit vemu.org!</a></p>
</body>
</html>
```

Output:



By default, links will appear as follows in all browsers:

- An unvisited link is underlined and blue
- A visited link is underlined and purple
- An active link is underlined and red

The target Attribute

By default, the linked page will be displayed in the current browser window. To change this, you must specify another target for the link.

The target attribute specifies where to open the linked document.

The target attribute can have one of the following values:

- _self - Default. Opens the document in the same window/tab as it was clicked
- _blank - Opens the document in a new window or tab
- _parent - Opens the document in the parent frame
- _top - Opens the document in the full body of the window

Example 2.6

```
<!DOCTYPE html>
```

```
<html>
<body>
<h2>The target Attribute</h2>
<a href="https://www.vemu.org/" target="_blank">Visit Vemu Institute!</a>
<p>If target="_blank", the link will open in a new browser window or tab.</p>
</body>
</html>
```

Absolute Path vs. Relative Path

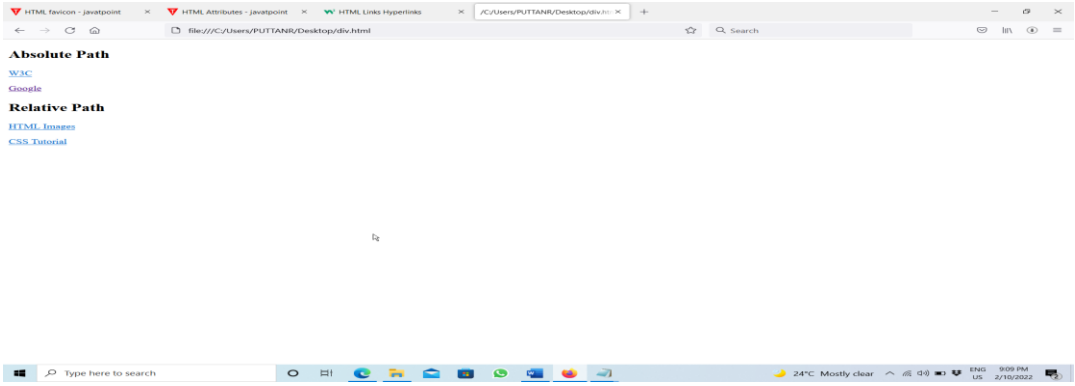
Both examples above are using an **absolute path** (a full web address) in the href attribute.

A local link (a link to a page within the same website) is specified with a **relative path** (without the "https://www" part):

Example 2.7

```
<!DOCTYPE html>
<html>
<body>
<h2>Absolute Path</h2>
<p><a href="https://www.w3.org/">W3C</a></p>
<p><a href="https://www.google.com/">Google</a></p>
<h2>Relative Path</h2>
<p><a href="html_images.asp">HTML Images</a></p>
<p><a href="/css/default.asp">CSS Tutorial</a></p>
</body>
</html>
```

Output:



Providing Hyperlink to an Image:

To use an image as a link, just put the <img> tag inside the <a> tag:

Example 2.8

```
<!DOCTYPE html>

<html>

<body>

<h2>Image as a Link</h2>

<p>The image below is a link. Try to click on it.</p>

<a href="default.asp"><img src="smiley.gif" alt="HTML tutorial"
style="width:42px;height:42px;"></a>

</body>

</html>
```

LISTS:

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

Example

An unordered HTML list:

- Item
- Item
- Item
- Item

An ordered HTML list:

1. First item
2. Second item
3. Third item
4. Fourth item

There are 3 types of lists available in HTML.

They are

(i) Unordered List
(ii) Ordered List
(iii) Definition List

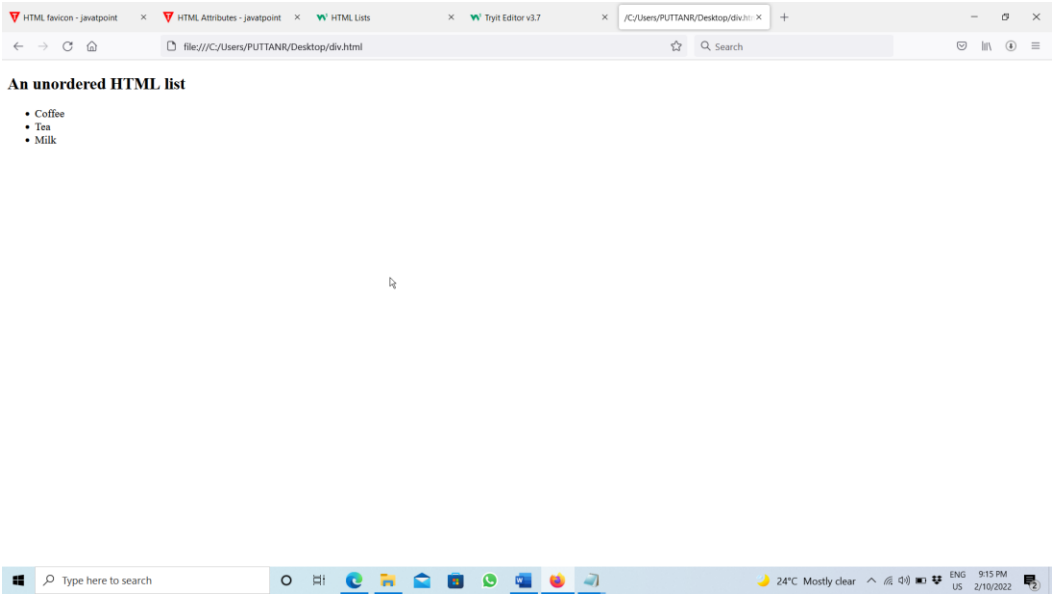
Unordered List:

An unordered list starts with the <ul> tag. Each list item starts with the <li> tag. The list items will be marked with bullets (small black circles) by default:

Example 2.9

```
<!DOCTYPE html>
<html>
<body>
<h2>An unordered HTML list</h2>
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
</body>
</html>
```

Output:



Unordered List with Type Attribute: -

All the information is displayed in a web page with bullets like circle.

Example 2.10

```
<!DOCTYPE html>
<html>
<body>
<h2>An unordered HTML list</h2>
<ul type="circle">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>
</body>
</html>
```

Ordered Lists:

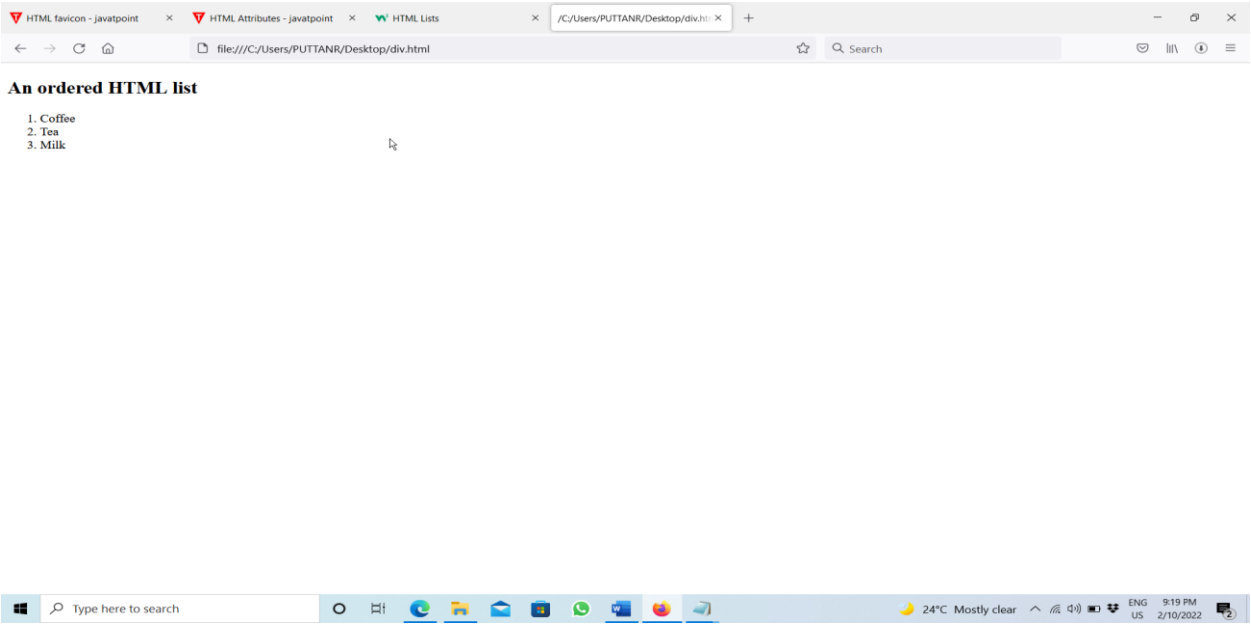
An ordered list starts with the <ol> tag. Each list item starts with the <li> tag.

The list items will be marked with numbers by default:

Example 2.11

```
<!DOCTYPE html>
<html>
<body>
<h2>An ordered HTML list</h2>
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
/body></html>
```

Output:



Ordered Lists with type attribute

All the information is displayed in a webpage by applying numerical numbers like 1,2,3 …… or i, ii, iii…. Or I, II, III… Or A, B, C……or a,b,c…

Example 2.11

```
<!DOCTYPE html>
<html>
<body>
<h2>An ordered HTML list</h2>
<ol type="A">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
</body>
</html>
```

Example 2.12

```
<!DOCTYPE html>
<html>
<body>
<h2>An ordered HTML list</h2>
<ol type="a">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
</body>
</html>
```

Example 2.13

```
<!DOCTYPE html>
<html>
<body>
<h2>An ordered HTML list</h2>
<ol type="i">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ol>
</body>
</html>
```

Example 2.14

```
<!DOCTYPE html>
<html>
<body>
<h2>An ordered HTML list</h2>
<ol type="I">
```

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

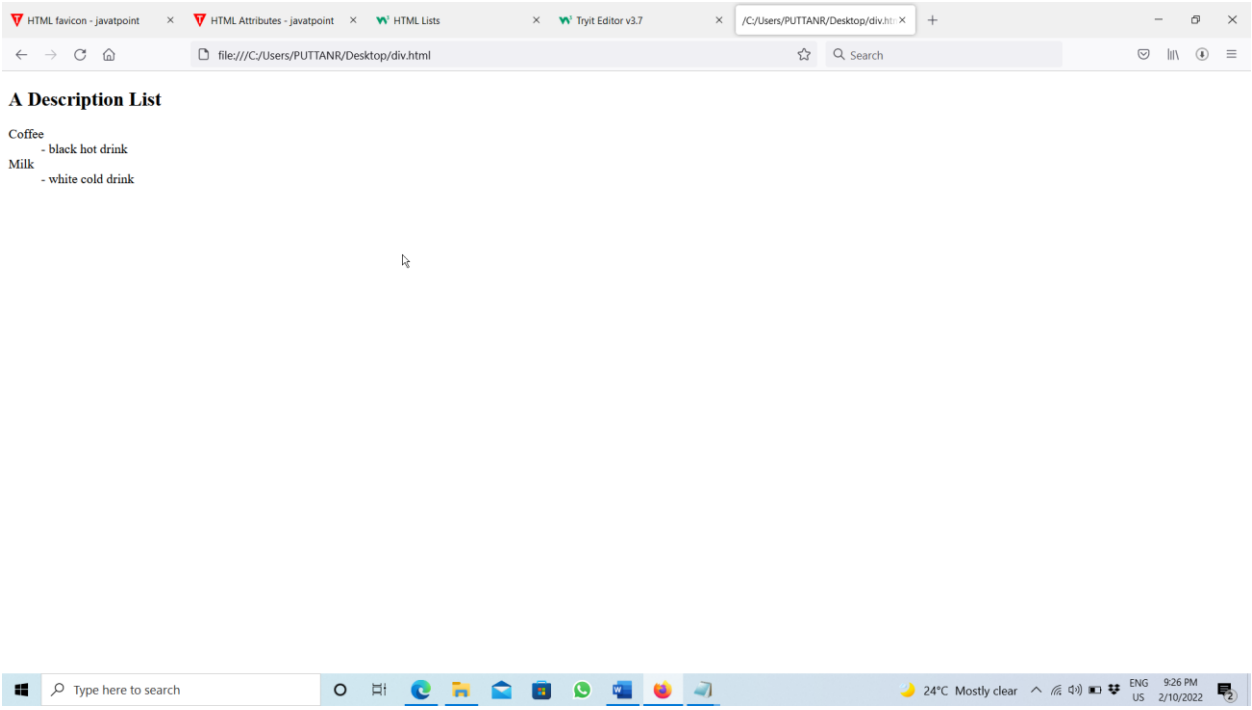
Definition Lists:

- HTML also supports description lists.
- A description list is a list of terms, with a description of each term.
- The <dl> tag defines the description list, the <dt> tag defines the term (name), and the <dd> tag describes each term:

Example 2.15

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

Output:



Images:-

HTML img tag is used to display image on the web page. HTML img tag is an empty tag that contains attributes only, closing tags are not used in HTML image element.

Attributes of HTML img tag

The src and alt are important attributes of HTML img tag. All attributes of HTML image tag are given below.

1) src

It is a necessary attribute that describes the source or path of the image. It instructs the browser where to look for the image on the server.

The location of image may be on the same directory or another server.

2) alt

The alt attribute defines an alternate text for the image, if it can't be displayed. The value of the alt attribute describe the image in words. The alt attribute is considered good for SEO prospective.

3) width

It is an optional attribute which is used to specify the width to display the image. It is not recommended now. You should apply CSS in place of width attribute.

4) height

It h3 the height of the image. The HTML height attribute also supports iframe, image and object elements. It is not recommended now. You should apply CSS in place of height attribute.

Use of height and width attribute with img tag

If we want to give some height and width to display image according to our requirement, then we can set it with height and width attributes of image.

Example 2.16

```
<!DOCTYPE html>
<html>
<head>
    <title>Image tag</title>
</head>
<body>
<h2>HTML image example with height and width</h2>
<img src="d://animal.jpg" height="180" width="300" alt="animal image">
</body>
</html>
```

iframe (embed youtube video):-

- **An HTML iframe is used to display a web page within a web page.**
- **The HTML <iframe> tag specifies an inline frame.**
- **An inline frame is used to embed another document within the current HTML document.**

Example 2.17

```
<!DOCTYPE html>
<html>
<body>
<h2>HTML Iframes</h2>
<p>You can use the height and width attributes to specify the size of the iframe:</p>
<iframe src="demo_iframe.htm" height="200" width="300" title="Iframe Example"></iframe>
```

```
</body>
</html>
```

Task : Create Your Profile Page

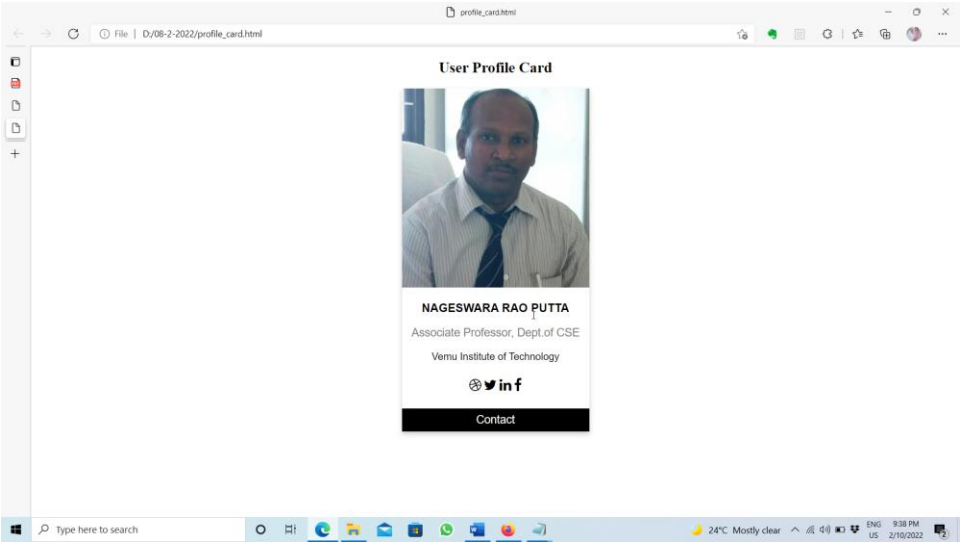
Source Code:

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">
<style>
.card {
  box-shadow: 0 4px 8px 0 rgba(0, 0, 0, 0.2);
  max-width: 300px;
  margin: auto;
  text-align: center;
  font-family: arial;
}

.title {
  color: grey;
  font-size: 18px;
}

button {
  border: none;
  outline: 0;
  display: inline-block;
  padding: 8px;
  color: white;
```

```
  background-color: #000;
  text-align: center;
  cursor: pointer;
  width: 100%;
  font-size: 18px;
}

a {
  text-decoration: none;
  font-size: 22px;
  color: black;
}

button:hover, a:hover {
  opacity: 0.7;
}
</style>
</head>
<body>

<h2 style="text-align:center">User Profile Card</h2>

<div class="card">
  <img src="pnr.jpg" alt="P Nageswara Rao" style="width:100%">
  <h3>NAGESWARA RAO PUTTA</h3>
  <p class="title">Associate Professor, Dept.of CSE</p>
  <p>Vemu Institute of Technology</p>
  <div style="margin: 24px 0;">
    <a href="#"><i class="fa fa-dribbble"></i></a>
    <a href="#"><i class="fa fa-twitter"></i></a>
```

```
    <a href="#"><i class="fa fa-linkedin"></i></a>
    <a href="#"><i class="fa fa-facebook"></i></a>
  </div>
  <p><button>Contact</button></p>
</div></body></html>
```

Output:



Module - 3:

HTML (continued): Tables: <table>, <tr>, <th>, <td>, Attributes for each Table element

Task: Create a Class Timetable (to merge rows/columns, use rowspan/colspan)

HTML Tables:-

- HTML table tag is used to display data in tabular form (row * column). There can be many columns in a row.
- We can create a table to display data in tabular form, using <table> element, with the help of <tr> , <td>, and <th> elements.
- In Each table, table row is defined by <tr> tag, table header is defined by <th>, and table data is defined by <td> tags.

- HTML tables are used to manage the layout of the page e.g. header section, navigation bar, body content, footer section etc. But it is recommended to use div tag over table to manage the layout of the page

The HTML tables allow web authors to arrange data like text, images, links, other tables, etc. into rows and columns of cells.

The HTML tables are created using the <table> tag in which the <tr> tag is used to create table rows and <td> tag is used to create data cells. The elements under <td> are regular and left aligned by default

HTML Table Tags

Tag	Description
<table>	It defines a table.
<tr>	It defines a row in a table.
<th>	It defines a header cell in a table.
<td>	It defines a cell in a table.
<caption>	It defines the table caption.
<colgroup>	It specifies a group of one or more columns in a table for formatting.
<col>	It is used with <colgroup> element to specify column properties for each column.
<tbody>	It is used to group the body content in a table.
<thead>	It is used to group the header content in a table.
<tfooter>	It is used to group the footer content in a table.

Example 3.1

```
<!DOCTYPE html>
<html>
  <head>
    <title>HTML Table</title>
  </head>
        <body>
    <table border = "1" width = "100%">
        <tr>
         <td>
           <table border = "1" width = "100%">
             <tr>
```

```
            <th>Name</th>
            <th>Salary</th>
          </tr>
          <tr>
            <td>Ramesh Raman</td>
            <td>5000</td>
          </tr>
          <tr>
            <td>Shabbir Hussein</td>
            <td>7000</td>
          </tr>
        </table>
      </td>
    </tr>
    </table>
  </body>
</html>
```

Output:

This will produce the following result –

Name	Salary
Ramesh Raman	5000
Shabbir Hussein	7000

Table Caption

You can add a caption that serves as a heading for the entire table.

Monthly savings	
Month	**Savings**
January	$100
February	$50

To add a caption to a table, use the <caption> tag:

Source Code:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid black;
  border-collapse: collapse;
}
th, td {
  padding: 5px;
  text-align: left;
}
</style>
</head>
<body>

<h2>Table Caption</h2>
<p>To add a caption to a table, use the caption tag.</p>

<table style="width:100%">
  <caption>Monthly savings</caption>
  <tr>
    <th>Month</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>January</td>
    <td>$100</td>
  </tr>
  <tr>
    <td>February</td>
    <td>$50</td>
  </tr>
</table>

</body>
</html>
```

HTML Table Padding & Spacing

HTML tables can adjust the padding inside the cells, and also the space between the cells.

With Padding		
hello	hello	hello

hello	hello	hello
hello	hello	hello

With Spacing		
hello	hello	hello
hello	hello	hello
hello	hello	hello

Cell Padding

Cell padding is the space between the cell edges and the cell content.

By default, the padding is set to 0.

Cell Spacing

Cell spacing is the space between each cell.

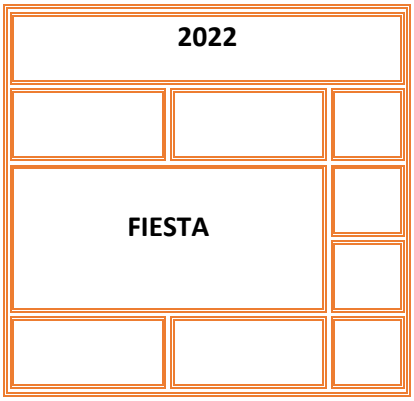
By default, the space is set to 2 pixels.

Colspan & Rowspan

HTML tables can have cells that spans over multiple rows and/or columns.

NAME		

APRIL		



Colspan

To make a cell span over multiple columns, use the colspan attribute:

Example 3.2

```
<!DOCTYPE html>
<html>
<head>
<title>Colspan</title>
</head>
<body>

<h2>Cell that spans two columns</h2>
<p>To make a cell span more than one column, use the colspan attribute. </p>

<table style="width:100%">
  <tr>
    <th colspan="2">Name</th>
    <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td>
    <td>Smith</td>
    <td>43</td>
  </tr>
  <tr>
    <td>Eve</td>
    <td>Jackson</td>
    <td>57</td>
  </tr>
</table>
</body>
</html>
```

Output:

Cell that spans two columns

To make a cell span more than one column, use the colspan attribute.

Name		Age
Jill	Smith	43
Eve	Jackson	57

Rowspan

To make a cell span over multiple rows, use the rowspan attribute:

Example 3.3

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<h2>Cell that spans two rows</h2>
<p>To make a cell span more than one row, use the rowspan attribute.</p>
<table style="width:100%">
  <tr>
    <th>Name</th>
    <td>Jill</td>
  </tr>
  <tr>
    <th rowspan="2">Phone</th>
    <td>555-1234</td>
  </tr>
  <tr>
    <td>555-8745</td>
  </tr>
</table>
</body>
</html>
```

Output:

Cell that spans two rows

To make a cell span more than one row, use the rowspan attribute.

Name	Jill
Phone	555-1234
	555-8745

Task: Create Your Class Time Table

Source Code:

A Table is an arrangement of rows and columns. Anyone can create a table by knowing the basics of HTML (HyperText Markup Language). A table is defined by using <table> tag in HTML.

Steps to Create a Table:

1. Create a <html> tag.
2. Create a table using the tags <table></table>.
3. Create rows in the table using <tr>This is the row tag</tr>.
4. Insert the data into rows using <td> Table Data</td> tags.
5. Close the table tag.
6. Close the html tag </html>.

This is the basic Time table created in HTML without the usage of font color and background colors.

Example:

```
<!DOCTYPE html>
<html>

<body>
```

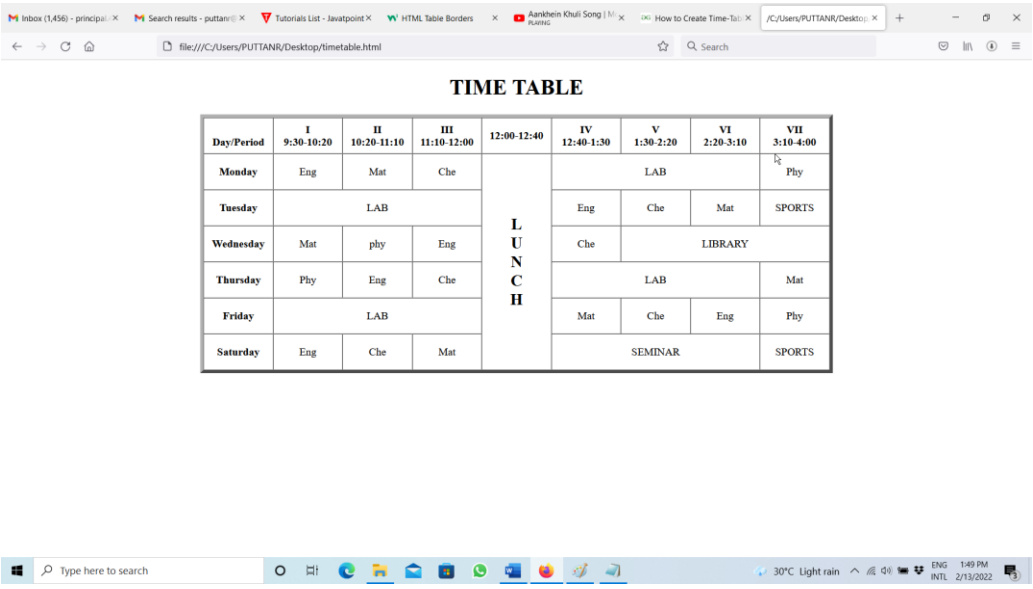
```
<h1>TIME TABLE</h1>
<table border="5" cellspacing="0" align="center">
   <!--<caption>Timetable</caption>-->
   <tr>
      <td align="center" height="50"
         width="100"><br>
         <b>Day/Period</b></br>
      </td>
      <td align="center" height="50"
         width="100">
         <b>I<br>9:30-10:20</b>
      </td>
      <td align="center" height="50"
         width="100">
         <b>II<br>10:20-11:10</b>
      </td>
      <td align="center" height="50"
         width="100">
         <b>III<br>11:10-12:00</b>
      </td>
      <td align="center" height="50"
         width="100">
         <b>12:00-12:40</b>
      </td>
      <td align="center" height="50"
         width="100">
         <b>IV<br>12:40-1:30</b>
      </td>
      <td align="center" height="50"
         width="100">
         <b>V<br>1:30-2:20</b>
      </td>
      <td align="center" height="50"
```

```
            width="100">
            <b>VI<br>2:20-3:10</b>
        </td>
        <td align="center" height="50"
            width="100">
            <b>VII<br>3:10-4:00</b>
        </td>
    </tr>
    <tr>
        <td align="center" height="50">
            <b>Monday</b></td>
        <td align="center" height="50">Eng</td>
        <td align="center" height="50">Mat</td>
        <td align="center" height="50">Che</td>
        <td rowspan="6" align="center" height="50">
            <h2>L<br>U<br>N<br>C<br>H</h2>
        </td>
        <td colspan="3" align="center"
            height="50">LAB</td>
        <td align="center" height="50">Phy</td>
    </tr>
    <tr>
        <td align="center" height="50">
            <b>Tuesday</b>
        </td>
        <td colspan="3" align="center"
            height="50">LAB
        </td>
        <td align="center" height="50">Eng</td>
        <td align="center" height="50">Che</td>
        <td align="center" height="50">Mat</td>
        <td align="center" height="50">SPORTS</td>
    </tr>
```

```
<tr>
    <td align="center" height="50">
        <b>Wednesday</b>
    </td>
    <td align="center" height="50">Mat</td>
    <td align="center" height="50">phy</td>
    <td align="center" height="50">Eng</td>
    <td align="center" height="50">Che</td>
    <td colspan="3" align="center"
        height="50">LIBRARY
    </td>
</tr>
<tr>
    <td align="center" height="50">
        <b>Thursday</b>
    </td>
    <td align="center" height="50">Phy</td>
    <td align="center" height="50">Eng</td>
    <td align="center" height="50">Che</td>
    <td colspan="3" align="center"
        height="50">LAB
    </td>
    <td align="center" height="50">Mat</td>
</tr>
<tr>
    <td align="center" height="50">
        <b>Friday</b>
    </td>
    <td colspan="3" align="center"
        height="50">LAB
    </td>
    <td align="center" height="50">Mat</td>
    <td align="center" height="50">Che</td>
```

```
            <td align="center" height="50">Eng</td>
            <td align="center" height="50">Phy</td>
        </tr>
        <tr>
            <td align="center" height="50">
                <b>Saturday</b>
            </td>
            <td align="center" height="50">Eng</td>
            <td align="center" height="50">Che</td>
            <td align="center" height="50">Mat</td>
            <td colspan="3" align="center"
                height="50">SEMINAR
            </td>
            <td align="center" height="50">SPORTS</td>
        </tr>
    </table>
</body>

</html>
```

Output:



TIME TABLE

Day/Period	I 9:30-10:20	II 10:20-11:10	III 11:10-12:00	12:00-12:40	IV 12:40-1:30	V 1:30-2:20	VI 2:20-3:10	VII 3:10-4:00
Monday	Eng	Mat	Che	LUNCH	LAB			Phy
Tuesday	LAB				Eng	Che	Mat	SPORTS
Wednesday	Mat	phy	Eng		Che	LIBRARY		
Thursday	Phy	Eng	Che		LAB			Mat
Friday	LAB				Mat	Che	Eng	Phy
Saturday	Eng	Che	Mat		SEMINAR			SPORTS

Module - 4:

HTML (continued): Form Elements: <input>, <select>, <textarea>, <button>, Attributes for each Form element

Task: Create a Student Hostel Application Form

HTML Forms

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

The <form> Element

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

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

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

Form Attributes

The Action Attribute

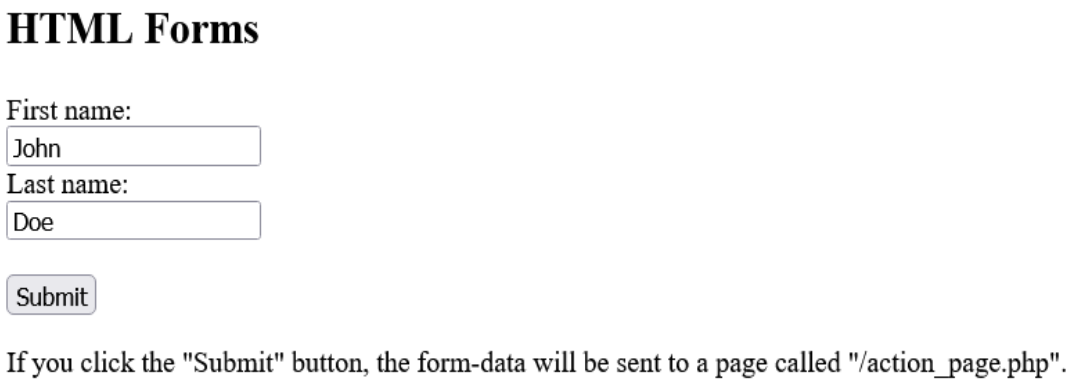
- The action attribute defines the action to be performed when the form is submitted.
- Usually, the form data is sent to a file on the server when the user clicks on the submit button.

Example 4.0

```
<!DOCTYPE html>
<html>
<body>
<h2>HTML Forms</h2>
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
```

```
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
<p>If you click the "Submit" button, the form-data will be sent to a page called "/action_page.php".</p>
</body>
</html>
```

Output:



The Target Attribute

- The target attribute specifies where to display the response that is received after submitting the form.

The target attribute can have one of the following values:

Value	Description
_blank	The response is displayed in a new window or tab
_self	The response is displayed in the current window
_parent	The response is displayed in the parent frame
_top	The response is displayed in the full body of the window
framename	The response is displayed in a named iframe

The default value is _self which means that the response will open in the current window.

Example 4.0(b)

```
<!DOCTYPE html>
```

```
<html>
<body>
<h2>The form target attribute</h2>
<p>When submitting this form, the result will be opened in a new browser
tab:</p>
<form action="/action_page.php" target="_blank">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
</body>
</html>
```

Output:

The form target attribute

When submitting this form, the result will be opened in a new browser tab:

First name:
John
Last name:
Doe

Submit

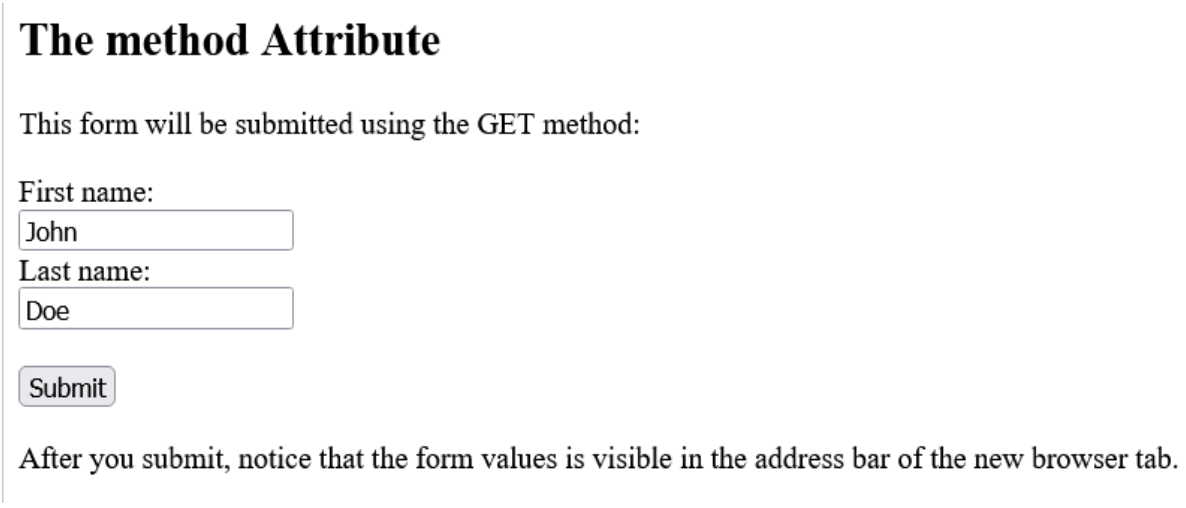
The Method Attribute

- The method attribute specifies the HTTP method to be used when submitting the form data.
- The form-data can be sent as URL variables (with method="get") or as HTTP post transaction (with method="post").
- The default HTTP method when submitting form data is GET.

Example 4.0(c)

```
<!DOCTYPE html>
<html>
<body>
<h2>The method Attribute</h2>
<p>This form will be submitted using the GET method:</p>
<form action="/action_page.php" target="_blank" method="get">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>

<p>After you submit, notice that the form values is visible in the address bar of the new browser tab.</p>
</body>
</html>
```

Output:

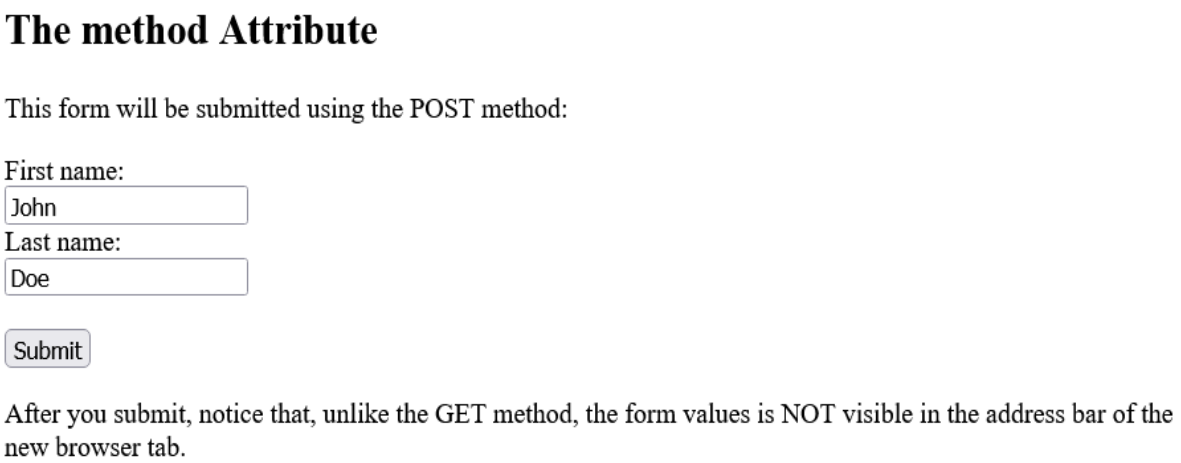


Example 4.0(d)

```
<!DOCTYPE html>
<html>
<body>
<h2>The method Attribute</h2>
```

```
<p>This form will be submitted using the POST method:</p>
<form action="/action_page.php" target="_blank" method="post">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
<p>After you submit, notice that, unlike the GET method, the form values
is NOT visible in the address bar of the new browser tab.</p>
</body>
</html>
```

Output:



GET	POST
➢ Appends the form data to the URL, in name/value pairs ➢ NEVER use GET to send sensitive data! (the submitted	➢ Appends the form data inside the body of the HTTP request (the submitted form data is not shown in the URL)

form data is visible in the URL!) ➢ The length of a URL is limited (2048 characters) ➢ Useful for form submissions where a user wants to bookmark the result ➢ GET is good for non-secure data, like query strings in Google	➢ POST has no size limitations, and can be used to send large amounts of data. ➢ Form submissions with POST cannot be bookmarked

Differentiate GET vs POST Methods of Form Elements

The HTML <form> Elements

The HTML `<form>` element can contain one or more of the following form elements:

- <input>
- <label>
- <select>
- <textarea>
- <button>
- <fieldset>
- <legend>
- <datalist>
- <output>
- <option>
- <optgroup>

The <input> Element

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

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

Here are some examples:

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

Text Fields

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

Example 4.1

```
<!DOCTYPE html>
<html>
<body>
<h2>Text input fields</h2>
<form>
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe">
</form>
<p>Note that the form itself is not visible.</p>
<p>Also note that the default width of text input fields is 20 characters.</p>
</body>
</html>
```

Output:

Text input fields

First name:
John
Last name:
Doe

Note that the form itself is not visible.

Also note that the default width of text input fields is 20 characters.

The <label> Element

- ✓ Notice the use of the <label> element in the example above.
- ✓ The <label> tag defines a label for many form elements.
- ✓ The <label> element is useful for screen-reader users, because the screen-reader will read out loud the label when the user focus on the input element.
- ✓ The for attribute of the <label> tag should be equal to the id attribute of the <input> element to bind them together.

Radio Buttons

- ➢ The <input type="radio"> defines a radio button.
- ➢ Radio buttons let a user select ONE of a limited number of choices.

Example 4.2

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

```
  <label for="javascript">JavaScript</label>
</form>
</body>
</html>
```

Output:

Checkboxes

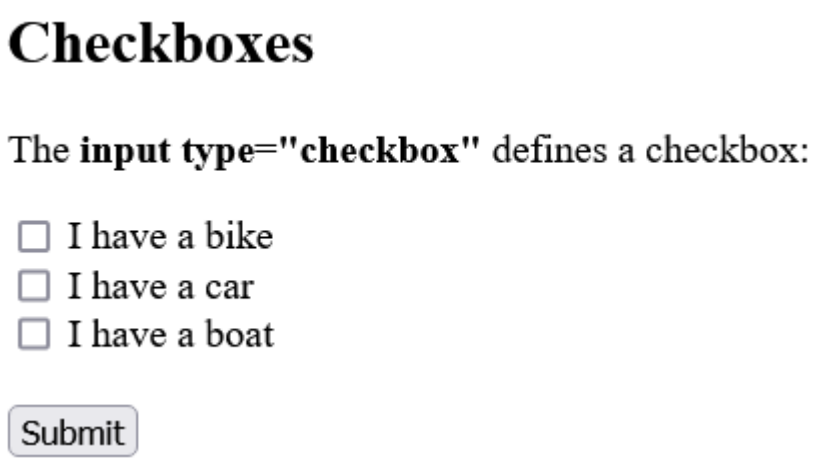
- The <input type="checkbox"> defines a checkbox.
- Checkboxes let a user select ZERO or MORE options of a limited number of choices.

Example 4.3

```
<!DOCTYPE html>
<html>
<body>
<h2>Checkboxes</h2>
<p>The <strong>input type="checkbox"</strong> defines a checkbox:</p>

<form action="/action_page.php">
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
  <label for="vehicle1"> I have a bike</label><br>
  <input type="checkbox" id="vehicle2" name="vehicle2" value="Car">
  <label for="vehicle2"> I have a car</label><br>
  <input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">
  <label for="vehicle3"> I have a boat</label><br><br>
  <input type="submit" value="Submit">
```

```
</form>
</body>
</html>
```

Output:



The Submit Button

- The <input type="submit"> defines a button for submitting the form data to a form-handler.
- The form-handler is typically a file on the server with a script for processing input data.
- The form-handler is specified in the form's action attribute.

Example 4.4

```
<!DOCTYPE html>
<html>
<body>
<h2>HTML Forms</h2>
<form action="/action_page.php">
  <label for="fname">First name:</label><br>
  <input type="text" id="fname" name="fname" value="John"><br>
  <label for="lname">Last name:</label><br>
  <input type="text" id="lname" name="lname" value="Doe"><br><br>
  <input type="submit" value="Submit">
</form>
```

```
<p>If you click the "Submit" button, the form-data will be sent to a page called "/action_page.php".</p>

</body>

</html>
```

Output:

HTML Forms

First name:
John
Last name:
Doe

Submit

If you click the "Submit" button, the form-data will be sent to a page called "/action_page.php".

The Name Attribute for <input>

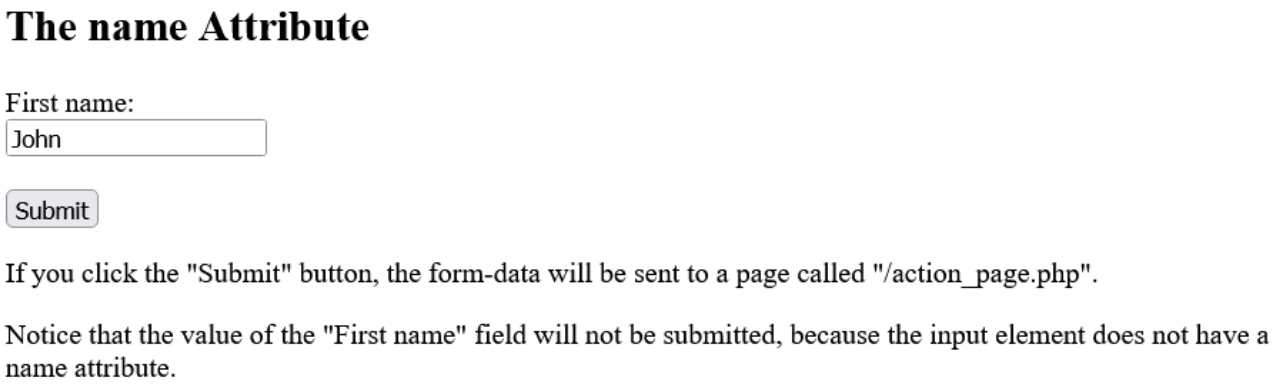
- Notice that each input field must have a name attribute to be submitted.
- If the name attribute is omitted, the value of the input field will not be sent at all.

Example 4.5

```
<!DOCTYPE html>

<html>

<body>

<h2>The name Attribute</h2>

<form action="/action_page.php">

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

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

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

</form>

<p>If you click the "Submit" button, the form-data will be sent to a page called "/action_page.php".</p>


<p>Notice that the value of the "First name" field will not be submitted, because the input element does not have a name attribute.</p>
```

```
</body>
</html>
```

Output:



The <select> Element

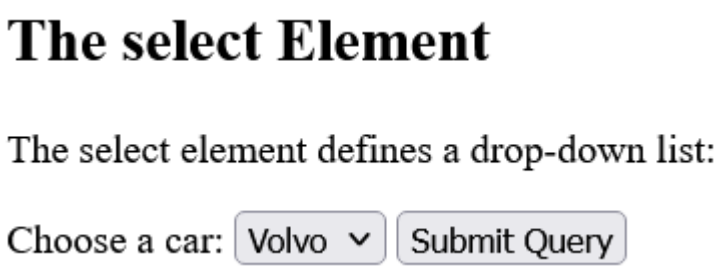
The <select> element defines a drop-down list:

Example 4.6

```
<!DOCTYPE html>
<html>
<body>
<h2>The select Element</h2>
<p>The select element defines a drop-down list:</p>
<form action="/action_page.php">
  <label for="cars">Choose a car:</label>
  <select id="cars" name="cars">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
  </select>
  <input type="submit">
</form>
</body>
```

```
</html>
```

Output:



The <option> elements define an option that can be selected.

- ✓ By default, the first item in the drop-down list is selected.
- ✓ To define a pre-selected option, add the selected attribute to the option:

Example 4.7

```
<!DOCTYPE html>
    <html>
    <body>
    <h2>Pre-selected Option</h2>
    <p>You can preselect an option with the selected attribute:</p>
    <form action="/action_page.php">
      <label for="cars">Choose a car:</label>
      <select id="cars" name="cars">
        <option value="volvo">Volvo</option>
        <option value="saab">Saab</option>
        <option value="fiat" selected>Fiat</option>
        <option value="audi">Audi</option>
      </select>
      <input type="submit">
    </form>
    </body>
    </html>
```

Output:

Pre-selected Option

You can preselect an option with the selected attribute:

Choose a car: Fiat Submit Query

Visible Values:

Use the size attribute to specify the number of visible values:

Example 4.8

```
<!DOCTYPE html>
<html>
<body>
<h2>Visible Option Values</h2>
<p>Use the size attribute to specify the number of visible values.</p>
<form action="/action_page.php">
  <label for="cars">Choose a car:</label>
  <select id="cars" name="cars" size="3">
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
  </select><br><br>
  <input type="submit">
</form>
</body>
</html>
```

Output:

Visible Option Values

Use the size attribute to specify the number of visible values.

Choose a car: Volvo Saab Fiat

Submit Query

Allow Multiple Selections:

- **Use the multiple attributes to allow the user to select more than one value:**

Example 4.9

```
<!DOCTYPE html>
<html>
<body>
<h2>Allow Multiple Selections</h2>
<p>Use the multiple attribute to allow the user to select more than one value.</p>
<form action="/action_page.php">
  <label for="cars">Choose a car:</label>
  <select id="cars" name="cars" size="4" multiple>
    <option value="volvo">Volvo</option>
    <option value="saab">Saab</option>
    <option value="fiat">Fiat</option>
    <option value="audi">Audi</option>
  </select><br><br>
  <input type="submit">
</form>
<p>Hold down the Ctrl (windows) / Command (Mac) button to select multiple options.</p>
```

```
</body>
</html>
```

Output:

Allow Multiple Selections

Use the multiple attribute to allow the user to select more than one value.

Choose a car: Volvo Saab Fiat Audi

Submit Query

Hold down the Ctrl (windows) / Command (Mac) button to select multiple options.

The <textarea> Element

- The <textarea> element defines a multi-line input field (a text area):

Textarea

The textarea element defines a multi-line input field.

The cat was playing in the garden.

Submit Query

- The rows attribute specifies the visible number of lines in a text area.
- The cols attribute specifies the visible width of a text area.

The <button> Element

The <button> element defines a clickable button:

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

```
<body>
<h2>The button Element</h2>
<button type="button" onclick="alert('Hello World!')">Click Me!</button>
</body>
</html>
```

Output:

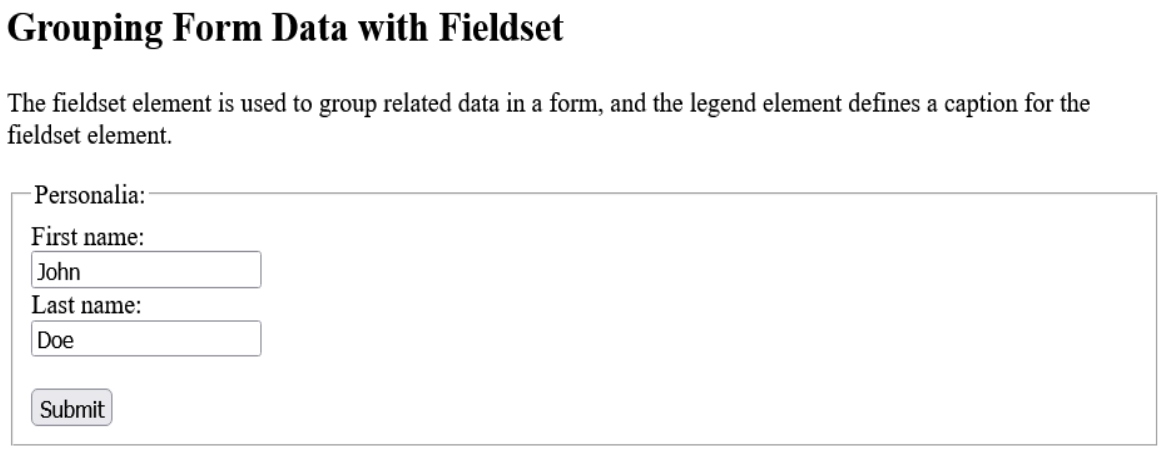
The <fieldset> and <legend> Elements

- The <fieldset> element is used to group related data in a form.
- The <legend> element defines a caption for the <fieldset> element.

```
<!DOCTYPE html>
<html>
<body>
<h2>Grouping Form Data with Fieldset</h2>
<p>The fieldset element is used to group related data in a form, and the legend element defines a caption for the fieldset element.</p>
<form action="/action_page.php">
  <fieldset>
    <legend>Personalia:</legend>
    <label for="fname">First name:</label><br>
    <input type="text" id="fname" name="fname" value="John"><br>
    <label for="lname">Last name:</label><br>
    <input type="text" id="lname" name="lname" value="Doe"><br><br>
    <input type="submit" value="Submit">
  </fieldset>
</form>
</body>
```

```
</html>
```

Output:



The <datalist> Element

- The <datalist> element specifies a list of pre-defined options for an <input> element.
- Users will see a drop-down list of the pre-defined options as they input data.
- The list attribute of the <input> element, must refer to the id attribute of the <datalist> element.

```
<!DOCTYPE html>
<html>
<body>
<h2>The datalist Element</h2>
<p>The datalist element specifies a list of pre-defined options for an input element.</p>
<form action="/action_page.php">
  <input list="browsers" name="browser">
  <datalist id="browsers">
    <option value="Internet Explorer">
    <option value="Firefox">
    <option value="Chrome">
    <option value="Opera">
    <option value="Safari">
  </datalist>
```

```
  <input type="submit">
</form>
<p><b>Note:</b> The datalist tag is not supported in Safari prior version 12.1.</p>
</body>
</html>
```

Output:

The datalist Element

The datalist element specifies a list of pre-defined options for an input element.

Submit Query

Note: The datalist tag is not supported in Safari prior version 12.1.

The <output> Element

The <output> element represents the result of a calculation (like one performed by a script).

```
<!DOCTYPE html>
<html>
<body>
<h2>The output Element</h2>
<p>The output element represents the result of a calculation.</p>
<form action="/action_page.php"
oninput="x.value=parseInt(a.value)+parseInt(b.value)">
  0
  <input type="range" id="a" name="a" value="50">
  100 +
  <input type="number" id="b" name="b" value="50">
  =
  <output name="x" for="a b"></output>
  <br><br>
```

```
  <input type="submit">
</form>
<p><strong>Note:</strong> The output element is not supported in Edge prior version 13.</p>
</body>
</html>
```

Output:

The output Element

The output element represents the result of a calculation.

0 100 + 50 =

Submit Query

Note: The output element is not supported in Edge prior version 13.

HTML Input Types

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

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

- <input type="tel">
- <input type="text">
- <input type="time">
- <input type="url">
- <input type="week">

Input Type Password

<input type="password"> defines a password field:

```
<!DOCTYPE html>

<html>

<body>

<h2>Password field</h2>

<p>The <strong>input type="password"</strong> defines a password field:</p>


<form action="/action_page.php">

  <label for="username">Username:</label><br>

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

  <label for="pwd">Password:</label><br>

  <input type="password" id="pwd" name="pwd"><br><br>

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

</form>

<p>The characters in a password field are masked (shown as asterisks or circles).</p>

</body>

</html>
```

Output:

Password field

The **input type="password"** defines a password field:

Username:

Password:

Submit

The characters in a password field are masked (shown as asterisks or circles).

Input Type Reset

<input type="reset"> defines a reset button that will reset all form values to their default values:

```
<!DOCTYPE html>

<html>

<body>

<h2>Reset Button</h2>

<p>The <strong>input type="reset"</strong> defines a reset button that resets all form values to their default values:</p>

<form action="/action_page.php">

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

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

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

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

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

  <input type="reset">

</form>

<p>If you change the input values and then click the "Reset" button, the form-data will be reset to the default values.</p>

</body>

</html>
```

Output:

Reset Button

The **input type="reset"** defines a reset button that resets all form values to their default values:

First name:
John
Last name:
Doe

Submit Reset

If you change the input values and then click the "Reset" button, the form-data will be reset to the default values.

Input Type Color

The <input type="color"> is used for input fields that should contain a color.

Depending on browser support, a color picker can show up in the input field.

```
<!DOCTYPE html>

<html>

<body>

<h2>Show a Color Picker</h2>

<p>The <strong>input type="color"</strong> is used for input fields that should contain a color.</p>

<form action="/action_page.php">

  <label for="favcolor">Select your favorite color:</label>

  <input type="color" id="favcolor" name="favcolor" value="#ff0000">

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

</form>

<p><b>Note:</b> type="color" is not supported in Internet Explorer 11 or Safari 9.1 (or earlier).</p>

</body>

</html>
```

Output:

Show a Color Picker

The **input type="color"** is used for input fields that should contain a color.

Select your favorite color: Submit

Note: type="color" is not supported in Internet Explorer 11 or Safari 9.1 (or earlier).

Input Type Date

The <input type="date"> is used for input fields that should contain a date.

Depending on browser support, a date picker can show up in the input field.

```
<!DOCTYPE html>
<html>
<body>
<h2>Date Field</h2>
<p>The <strong>input type="date"</strong> is used for input fields that should contain a date.</p>
<form action="/action_page.php">
  <label for="birthday">Birthday:</label>
  <input type="date" id="birthday" name="birthday">
  <input type="submit" value="Submit">
</form>
<p><strong>Note:</strong> type="date" is not supported in Internet Explorer 11 or prior Safari 14.1.</p>
</body>
</html>
```

```
<!DOCTYPE html>
<html>
<body>
<h2>Date Field</h2>
```

```
<p>The <strong>input type="date"</strong> is used for input fields that should contain a date.</p>

<form action="/action_page.php">

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

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

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

</form>

<p><strong>Note:</strong> type="date" is not supported in Internet Explorer 11 or prior Safari 14.1.</p>

</body>

</html>
```

Output

Date Field

The **input type="date"** is used for input fields that should contain a date.

Birthday: mm / dd / yyyy Submit

Note: type="date" is not supported in Internet Explorer 11 or prior Safari 14.1.

Input Type Email

- The <input type="email"> is used for input fields that should contain an e-mail address.
- Depending on browser support, the e-mail address can be automatically validated when submitted.
- Some smartphones recognize the email type, and add ".com" to the keyboard to match email input.

```
<!DOCTYPE html>

<html>

<body>

<h2>Email Field</h2>

<p>The <strong>input type="email"</strong> is used for input fields that should contain an e-mail address:</p>


<form action="/action_page.php">
```

```
  <label for="email">Enter your email:</label>
  <input type="email" id="email" name="email">
  <input type="submit" value="Submit">
</form>
</body>
</html>
```

Output:

Email Field

The **input type="email"** is used for input fields that should contain an e-mail address:

Enter your email: [ ] Submit

Input Type File

The <input type="file"> defines a file-select field and a "Browse" button for file uploads.

```
<!DOCTYPE html>
<html>
<body>
<h1>File upload</h1>
<p>Show a file-select field which allows a file to be chosen for upload:</p>
<form action="/action_page.php">
  <label for="myfile">Select a file:</label>
  <input type="file" id="myfile" name="myfile"><br><br>
  <input type="submit" value="Submit">
</form>
</body>
</html>
```

Output:

File upload

Show a file-select field which allows a file to be chosen for upload:

Select a file: Browse... No file selected.

Submit

Input Type Hidden

- The <input type="hidden"> defines a hidden input field (not visible to a user).
- A hidden field let web developers include data that cannot be seen or modified by users when a form is submitted.
- A hidden field often stores what database record that needs to be updated when the form is submitted.

```
<!DOCTYPE html>
<html>
<body>
<h1>A Hidden Field (look in source code)</h1>
<form action="/action_page.php">
  <label for="fname">First name:</label>
  <input type="text" id="fname" name="fname"><br><br>
  <input type="hidden" id="custId" name="custId" value="3487">
  <input type="submit" value="Submit">
</form>
<p><strong>Note:</strong> The hidden field is not shown to the user, but the data is sent when the form is submitted.</p>
</body>
</html>
```

Output:

A Hidden Field (look in source code)

First name:

Submit

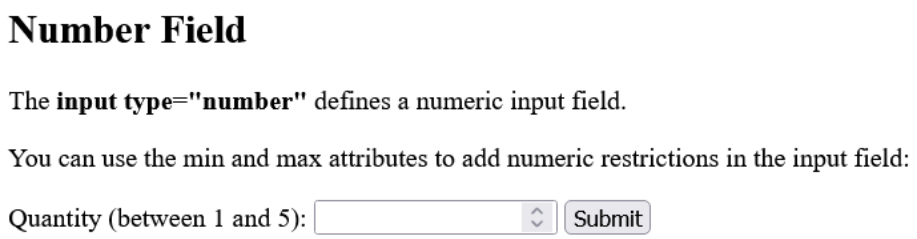
Note: The hidden field is not shown to the user, but the data is sent when the form is submitted

Input Type Number

- The <input type="number"> defines a numeric input field.
- You can also set restrictions on what numbers are accepted.
- The following example displays a numeric input field, where you can enter a value from 1 to 5:

```
<!DOCTYPE html>
<html>
<body>
<h2>Number Field</h2>
<p>The <strong>input type="number"</strong> defines a numeric input field.</p>
<p>You can use the min and max attributes to add numeric restrictions in the input field:</p>
<form action="/action_page.php">
  <label for="quantity">Quantity (between 1 and 5):</label>
  <input type="number" id="quantity" name="quantity" min="1" max="5">
  <input type="submit" value="Submit">
</form>
</body>
</html>
```

Output:



Input Type Search

The <input type="search"> is used for search fields (a search field behaves like a regular text field).

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

<h2>Search Field</h2>
```

```
<p>The <strong>input type="search"</strong> is used for search fields (behaves like a regular text field):</p>

<form action="/action_page.php">

  <label for="gsearch">Search Google:</label>

  <input type="search" id="gsearch" name="gsearch">

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

</form>

/body>

</html>
```

Output:

Search Field

The **input type="search"** is used for search fields (behaves like a regular text field):

Search Google: Submit

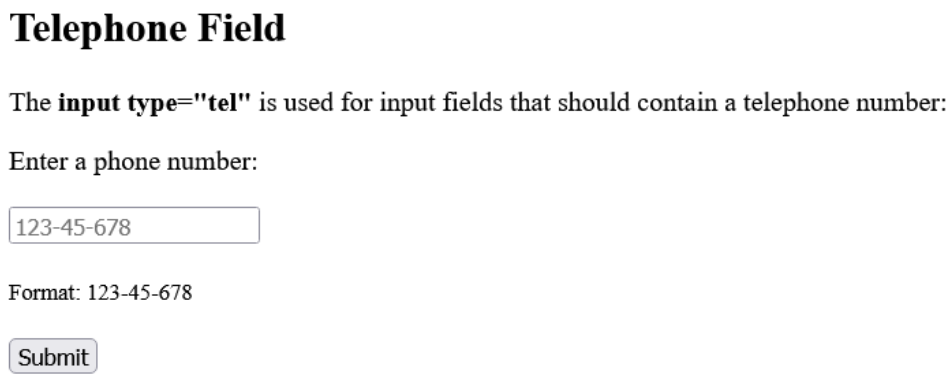
Input Type Tel

The <input type="tel"> is used for input fields that should contain a telephone number.

```
 <!DOCTYPE html>

<html>

<body>

<h2>Telephone Field</h2>

<p>The <strong>input type="tel"</strong> is used for input fields that should contain a telephone number:</p>

<form action="/action_page.php">

  <label for="phone">Enter a phone number:</label><br><br>

  <input type="tel" id="phone" name="phone" placeholder="123-45-678" pattern="[0-9]{3}-[0-9]{2}-[0-9]{3}" required><br><br>

  <small>Format: 123-45-678</small><br><br>

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

</form>

</body>
```

```
</html>
```

Output:



Input Type Url

The <input type="url"> is used for input fields that should contain a URL address.

Depending on browser support, the url field can be automatically validated when submitted.

```
<!DOCTYPE html>
<html>
<body>
<h1>Display a URL Input Field</h1>
<p>The <strong>input type="url"</strong> is used for input fields that should contain a URL address:</p>
<form action="/action_page.php">
  <label for="homepage">Add your homepage:</label>
  <input type="url" id="homepage" name="homepage">
  <input type="submit" value="Submit">
</form>
</body>
</html>
```

Output:

Task: Create a Student Hostel Application Form

Source Code:

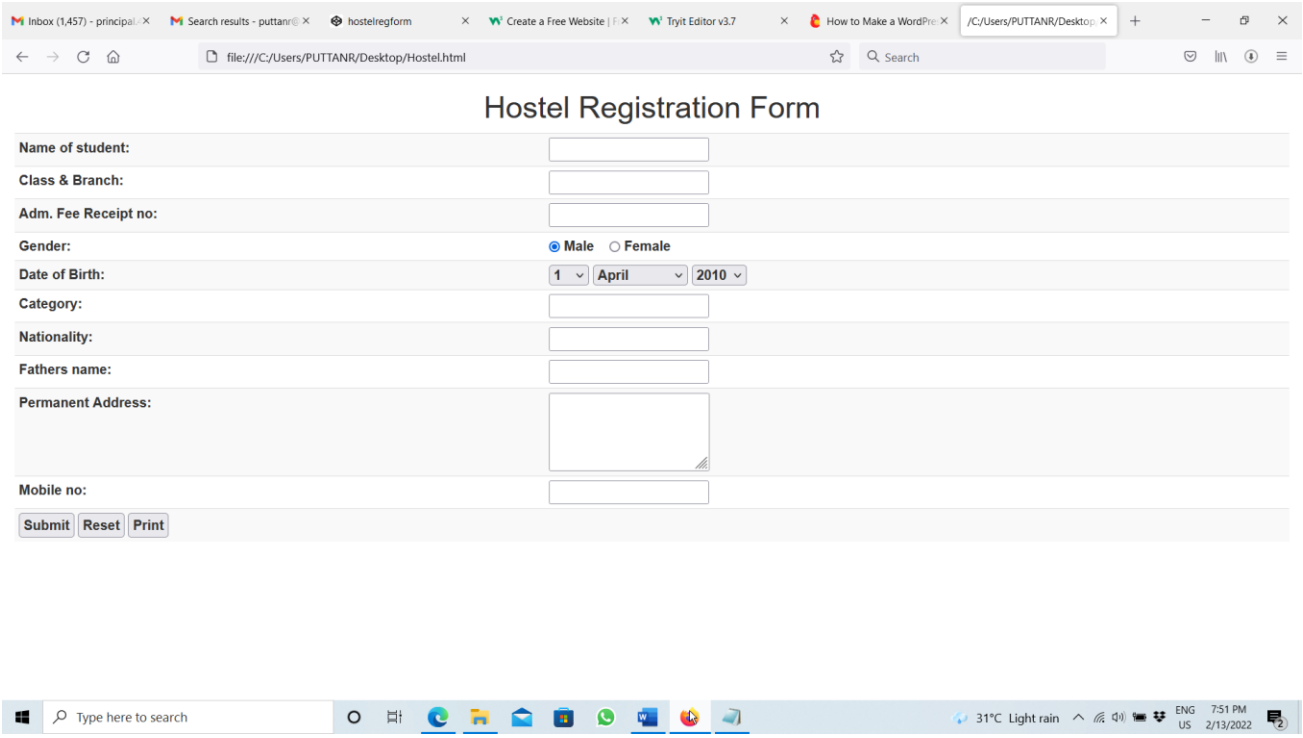
```
<!DOCTYPE HTML>

<html>

<head>

<title>Hostel Application Form</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>

<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></scri
pt>

<body>

<div class="container-fluid">

<h1 align="center">Hostel Registration Form</h1>

<!--<div id="s0">

    <img src="http://www.kgcas.ac.in/images/banner2.jpg" height="50" width="80">

    <img src="http://www.spvp.edu.in/SBPP/assets/img/portfolio/facilities-big-2.jpg"
height="50" width="80" id="img1">

    <img src="https://campushunt.in/photogallery/Vit-college-of-Engg-and-Tech-
cateen-344.jpg" height="50" width="80" id="img2">

</div>-->

<div id="s1">

        <form method="post">


        <table class="table table-striped table-condensed">

        <tr>

        <th align="left" class="txt1">Name of student:</th>

        <th align="left"><input type="text"></th></tr>
```

```
<tr><th align="left">Class & Branch:</th> <th align="left"><input type="text"
></th></tr>

<tr><th align="left">Adm. Fee Receipt no:</th> <th align="left"><input type="text"
></th></tr>

<tr><th align="left">Gender:</th><th align="left"><input type="radio"
name="gender" value="m" checked> Male   

<input type="radio" name="gender" value="f"> Female</th></tr>

<tr><th align="left">Date of Birth:</th> <th align="left"><select id="sl1">

<option value="1">1</option>

<option value="2">2</option>

<option value="3">3</option>

<option value="4">4</option>

<option value="5">5</option>

<option value="6">6</option>

<option value="7">7</option>

<option value="8">8</option>

<option value="9">9</option>

<option value="11">11</option>

<option value="12">12</option>

<option value="13">13</option>

<option value="14">14</option>

<option value="15">15</option>

<option value="16">16</option>

<option value="17">17</option>

<option value="18">18</option>

<option value="19">19</option>

<option value="20">20</option>

<option value="21">21</option>

<option value="22">22</option>

<option value="23">23</option>

<option value="24">24</option>
```

```
<option value="25">25</option>
<option value="26">26</option>
<option value="27">27</option>
<option value="28">28</option>
<option value="29">29</option>
<option value="30">30</option>
<option value="31">31</option>
     </select>
    <select id="sl2">
    <option value="Jan">January</option>
    <option value="Feb">February</option>
    <option value="Mar">March</option>
    <option value="Apr" selected>April</option>
    <option value="May">May</option>
<option value="June">June</option>
<option value="July">July</option>
<option value="August">August</option>
<option value="September">September</option>
<option value="October">October</option>
<option value="November">November</option>
<option value="December">December</option>
     </select>
    <select id="sl3">
    <option value="2010">2010</option>
    <option value="2011">2011</option>
    <option value="2012">2012</option>
    <option value="2013">2013</option>
    <option value="2014">2014</option>
    <option value="2015">2015</option>
    <option value="2016">2016</option>
```

```
    <option value="2017">2017</option>
<option value="2018">2018</option>
<option value="2019">2019</option>
<option value="2020">2020</option>
<option value="2021">2021</option>
<option value="2022">2022</option>
    </select></th></tr>
    <tr><th align="left">Category:</th> <th align="left"><input type="text" ></th></tr>
    <tr><th align="left">Nationality:</th><th align="left"> <input type="text"
></th></tr>
    <tr><th align="left">Fathers name:</th><th align="left"><input type="text"
></th></tr>
    <tr><th align="left">Permanent Address:</th><th align="left"> <textarea rows="3"
cols="18"></textarea></th></tr>
    <tr><th align="left">Mobile no:</th><th align="left"><input type="text"
></th></tr>
    <tr><th colspan="2"><input type="submit" id="btn1" value="Submit">
    <input type="reset" value="Reset" id="btn2"> <input type="submit" value="Print"
id="btn3"></th></tr>
        </table>
        </form>
</div>
</div>
</body>
</html>
```

Output:



Module - 5:

Cascading Style Sheets (CSS): CSS Properties, Types of CSS, Selectors, box model, Pseudo-elements, z-index

Task: Make the Hostel Application Form designed in Module -4 beautiful using CSS (add colors, backgrounds, change font properties, borders, etc.)

Cascading Style Sheets (CSS)

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

Example 5.1

```
<!DOCTYPE html>

<html>

<head>

<style type="text/css">
```

```
body {
  background-color: lightblue;
}
h1 {
  color: white;
  text-align: center;
}
p {
  font-family: verdana;
  font-size: 20px;
}
</style>
</head>
<body>
<h1>My First CSS Example</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

Output

CSS Solved a Big Problem

- HTML was NEVER intended to contain tags for formatting a web page!
- HTML was created to describe the content of a web page, like:

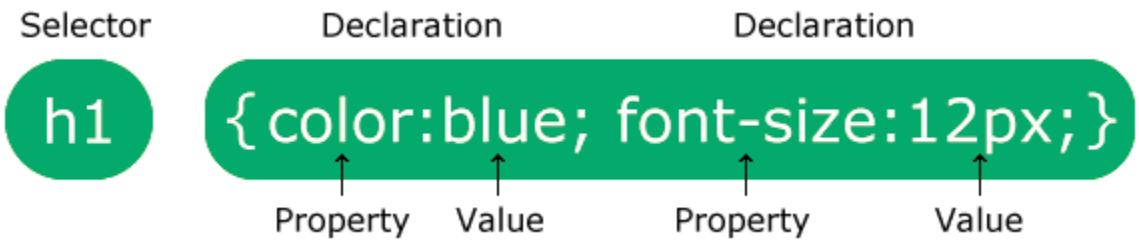
  <h1>This is a heading</h1>

This is a paragraph.

- When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.
- To solve this problem, the World Wide Web Consortium (W3C) created CSS.
- CSS removed the style formatting from the HTML page!

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

CSS Syntax



- The selector points to the HTML element you want to style.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property name and a value, separated by a colon.
- Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

```
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
p {
  color: red;
  text-align: center;
}
</style>
</head>
<body>
<p>Hello World!</p>
<p>These paragraphs are styled with CSS.</p>
</body>
</html>
```

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

Types of CSS

When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.

There are 3 ways to insert CSS

1. inline CSS
2. internal CSS or Embedded CSS
3. external CSS

Inline CSS

- An inline style may be used to apply a unique style for a single element.
- To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

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

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

Output:

This is a heading

This is a paragraph.

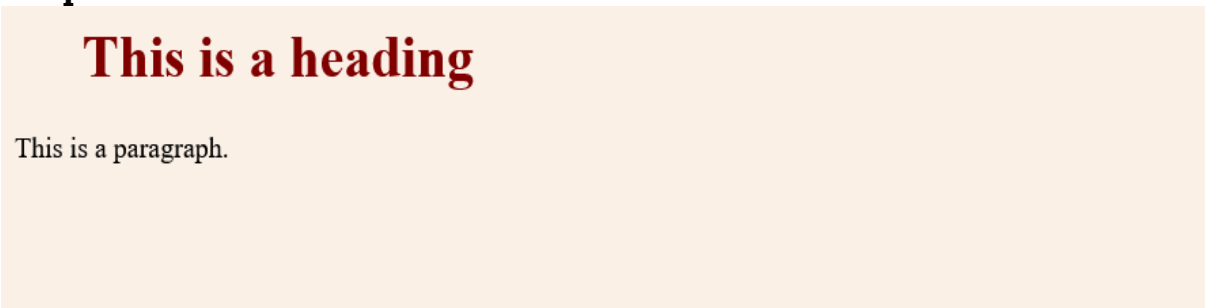
Internal CSS

- An internal style sheet may be used if one single HTML page has a unique style.
- The internal style is defined inside the <style> element, inside the head section.

```
<!DOCTYPE html>
<html>
<head>
<style type="text/css">
```

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

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

</body>
</html>
```

Output



External CSS

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

Task: Make the Hostel Application Form designed in Module -4 beautiful using CSS (add colors, backgrounds, change font properties, borders, etc.)

Hostel.html

```
<!DOCTYPE HTML>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
```

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

<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.0/jquery.min.js"></script>

<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></scri
pt>

</head>

<body>

<div class="container-fluid">

<h1 align="center">Hostel Registration Form</h1>

<!--<div id="s0">

   <img src="http://www.kgcas.ac.in/images/banner2.jpg" height="50" width="80">

   <img src="http://www.spvp.edu.in/SBPP/assets/img/portfolio/facilities-big-2.jpg"
height="50" width="80" id="img1">

   <img src="https://campushunt.in/photogallery/kg-college-of-arts-and-science-
cateen-344.jpg" height="50" width="80" id="img2">

</div>-->

<div id="s1">

        <form method="post">

        <table class="table table-striped table-condensed">

        <tr>

        <th align="left" class="txt1">Name of student:</th>

        <th align="left"><input type="text"></th></tr>

   <tr><th align="left">Class & Branch:</th> <th align="left"><input type="text"
></th></tr>

   <tr><th align="left">Adm. Fee Receipt no:</th> <th align="left"><input type="text"
></th></tr>

   <tr><th align="left">Gender:</th><th align="left"><input type="radio"
name="gender" value="m" checked> Male   

   <input type="radio" name="gender" value="f"> Female</th></tr>

   <tr><th align="left">Date of Birth:</th> <th align="left"><select id="sl1">

   <option value="1">1</option>

   <option value="2">2</option>
```

```
<option value="3">3</option>
<option value="4">4</option>
<option value="5">5</option>
    <option value="6">6</option>
    <option value="7">7</option>
    <option value="8">8</option>
    <option value="9">9</option>
    <option value="11">11</option>
    <option value="12">12</option>
    <option value="13">13</option>
    <option value="14">14</option>
    <option value="15">15</option>
    <option value="16">16</option>
    <option value="17">17</option>
    <option value="18">18</option>
    <option value="19">19</option>
    <option value="20">20</option>
    <option value="21">21</option>
    <option value="22">22</option>
    <option value="23">23</option>
    <option value="24">24</option>
    <option value="25">25</option>
    <option value="26">26</option>
    <option value="27">27</option>
    <option value="28">28</option>
    <option value="29">29</option>
    <option value="30">30</option>
    <option value="31">31</option>
 </select>
<select id="sl2">
```

```
    <option value="Jan">January</option>
    <option value="Feb">February</option>
    <option value="Mar">March</option>
    <option value="Apr" selected>April</option>
    <option value="May">May</option>
  <option value="June">June</option>
  <option value="July">July</option>
  <option value="August">August</option>
  <option value="September">September</option>
  <option value="October">October</option>
  <option value="November">November</option>
<option value="December">December</option>
     </select>
    <select id="sl3">
    <option value="2010">2010</option>
    <option value="2011">2011</option>
    <option value="2012">2012</option>
    <option value="2013">2013</option>
    <option value="2014">2014</option>
    <option value="2015">2015</option>
    <option value="2016">2016</option>
    <option value="2017">2017</option>
    <option value="2018">2018</option>
   <option value="2019">2019</option>
   <option value="2020">2020</option>
   <option value="2021">2021</option>
   <option value="2022">2022</option>
    </select></th></tr>
    <tr><th align="left">Category:</th> <th align="left"><input type="text" ></th></tr>
```

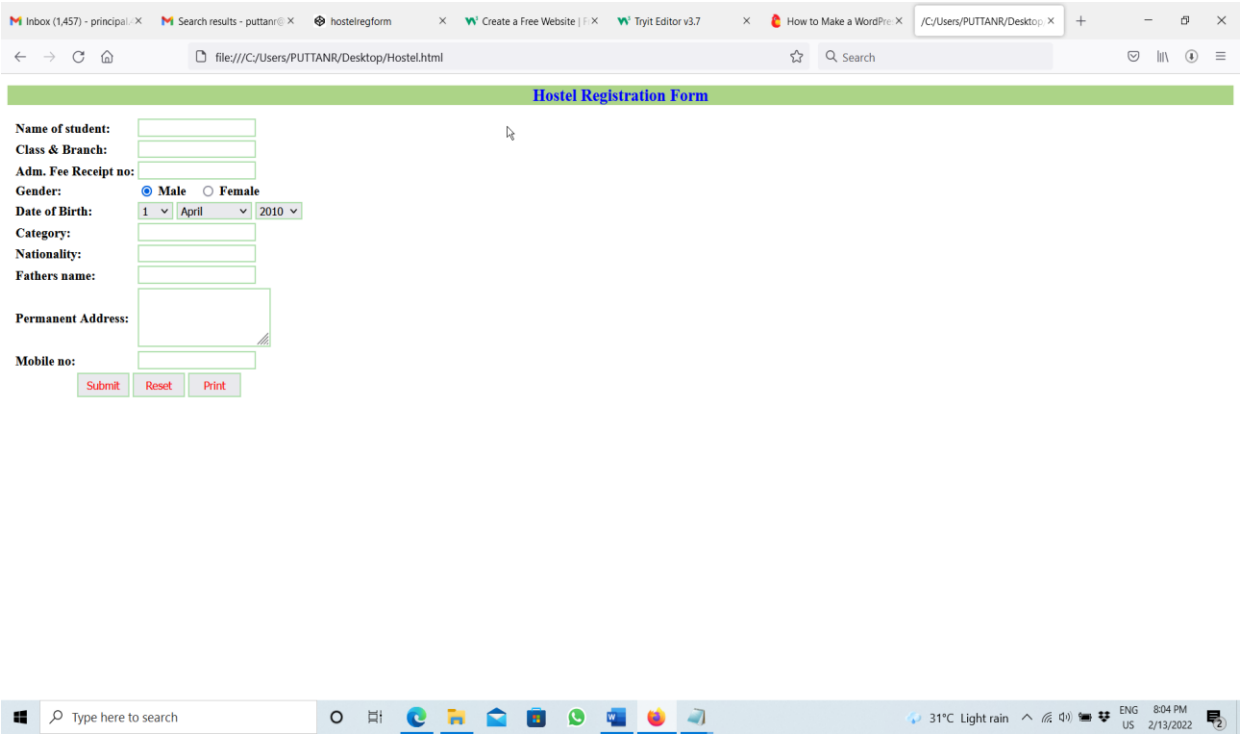
```
    <tr><th align="left">Nationality:</th><th align="left"> <input type="text"
></th></tr>

    <tr><th align="left">Fathers name:</th><th align="left"><input type="text"
></th></tr>

    <tr><th align="left">Permanent Address:</th><th align="left"> <textarea rows="3"
cols="18"></textarea></th></tr>

    <tr><th align="left">Mobile no:</th><th align="left"><input type="text"
></th></tr>

    <tr><th colspan="2"><input type="submit" id="btn1" value="Submit">

    <input type="reset" value="Reset" id="btn2"> <input type="submit" value="Print"
id="btn3"></th></tr>

        </table>

        </form>

</div>

</div>

</body>

</html>
```

Hostel.css

```
div.container-fluid h1
{
  color:blue;
  background-color:#acd487;
  font-family:Gill Sans MT;
  font-size:20px;
  }
div.container-fluid div#s1
{
  position:absolute;
  top:50px;
  left:15px;
}
div.container-fluid div#s0 img
{
  border:2px dashed blue;
}
div.container-fluid div#s0 img#img1
{
  position:absolute;
  top:115px;
  left:15px;
```

```
}
div.container-fluid div#s0 img#img2
{
  position:absolute;
  top:180px;
  left:15px;
}
div.container-fluid div#s1 input#btn1
{
padding:5px;
width:65px;
}
div.container-fluid div#s1 input#btn2
{
padding:5px;
width:65px;
}
div.container-fluid div#s1 input#btn3
{
padding:5px;
width:65px;
}
div.container-fluid div#s1 input
{
  border:2px solid #aedfac;
  color:red;
}
div.container-fluid div#s1 textarea
{
  border:2px solid #aedfac;
}
div.container-fluid div#s1 select
{
  border:2px solid #aedfac;
}
```

Output:



Module - 6:

Bootstrap – CSS Framework: Layouts (Containers, Grid System), Forms, Other Components

Task: Style the Hostel Application Form designed in Module -5 still more beautiful using Bootstrap CSS (Re-size browser and check how the webpage displays in mobile resolution)

1. What is Bootstrap?

Bootstrap is a free front-end framework for faster and easier web development

Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins

Bootstrap also gives you the ability to easily create responsive designs.

2. What is Responsive Web Design?

Responsive web design is about creating web sites which automatically adjust themselves to look good on all devices, from small phones to large desktops.

3. Advantages of using Bootstrap?

Easy to use: Anybody with just basic knowledge of HTML and CSS can start using Bootstrap

Responsive features: Bootstrap's responsive CSS adjusts to phones, tablets, and desktops

Mobile-first approach: In Bootstrap, mobile-first styles are part of the core framework

4. How to add Bootstrap CSS to a HTML page?

Three steps to be followed when we want to add Bootstrap to any HTML page.

a. Add the HTML5 doctype

Bootstrap 5 uses HTML elements and CSS properties that require the HTML5 doctype.

Always include the HTML5 doctype at the beginning of the page, along with the lang attribute

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Bootstrap 5 Example</title>
    <meta charset="utf-8">
  </head>
</html>
```

b. Bootstrap 5 is mobile-first

Bootstrap 5 is designed to be responsive to mobile devices. Mobile-first styles are part of the core framework.

To ensure proper rendering and touch zooming, add the following <meta> tag inside the <head> element:

```
<meta name="viewport" content="width=device-width, initial-scale=1">
```

c. Link Bootstrap CSS to HTML in head section

```
<!-- Latest compiled and minified CSS -->
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"
rel="stylesheet">


<!-- Latest compiled JavaScript -->
```

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js
"></script>

5. How to apply bootstrap CSS styles to HTML tags?

Using Classes we apply bootstrap CSS styles to html tags. There are style classes defined in the bootstrap for each HTML element such as table, forms, text.

Example:

h1 Bootstrap heading

In the above example, we applied “h1” class from bootstrap to element. Similarly, there are several styles defined for each HTML element. All these bootstrap classes can be applied to html elements as per our requirement.

6. What are other style classes defined in bootstrap framework?

There are style classes defined for each HTML tag. Like “table” class for <table> tag, “form-control” class for <input> tag etc. We can find classes for each element in W3Schools.com bootstrap tutorials.

7. How do we apply margins and paddings using Bootstrap framework?

We have Style classes defined to apply margins and paddings for HTML tags. Class “m” will be used and for padding class “p” will be used. We should specify margins or paddings using units instead of pixels in Bootstrap.

For Example:

Hello world

The above code shows, 5 units of margin applied to paragraph element. Just like CSS, we have classes for margin-top (“mt”), margin-bottom (“mb”) and so on.

We follow similar mechanism for paddings also. We use class “p” to apply padding to html elements.

Hello world

The above code shows, 5 units of padding applied to paragraph element. Just like CSS, we have classes for padding-top (“pt”), padding-bottom (“pb”) and so on.

Examples:

Hello world // margin top 5

Hello world // margin left(start) 5

Hello world // padding top 5

Hello world //padding bottom 5

<p class=”pb-5 mt-3”>Hello world</p> // padding bot 5, margin top 3

8. How do we create sections in HTML Webpage using Bootstrap to make it a responsive page?

Bootstrap has grid system feature, using which we can divide our web page into sections and make our webpage responsive as well.

Bootstrap's grid system is built with flexbox and allows up to 12 columns across the page.

If you do not want to use all 12 columns individually, you can group the columns together to create wider columns.

The grid system is responsive, and the columns will re-arrange automatically depending on the screen size.

Make sure that the sum adds up to 12 or fewer (it is not required that you use all 12 available columns).

Grid Classes

The Bootstrap 5 grid system has six classes:

- .col- (extra small devices - screen width less than 576px)
- .col-sm- (small devices - screen width equal to or greater than 576px)
- .col-md- (medium devices - screen width equal to or greater than 768px)
- .col-lg- (large devices - screen width equal to or greater than 992px)
- .col-xl- (xlarge devices - screen width equal to or greater than 1200px)
- .col-xxl- (xxlarge devices - screen width equal to or greater than 1400px)

The classes above can be combined to create more dynamic and flexible layouts.

Tip: Each class scales up, so if you want to set the same widths for sm and md, you only need to specify sm.

Example 1:

```
<div class="row">
  <div class="col">.col</div>
  <div class="col">.col</div>
```

```
  <div class="col">.col</div>
</div>
```

The above example shows how to create three equal-width columns, on all devices and screen widths.

Example 2:

```
<div class="row">
  <div class="col-sm-3">.col-sm-3</div>
  <div class="col-sm-3">.col-sm-3</div>
  <div class="col-sm-3">.col-sm-3</div>
  <div class="col-sm-3">.col-sm-3</div>
</div>
```

The following example shows how to create four equal-width columns starting at tablets and scaling to extra large desktops. On mobile phones or screens that are less than 576px wide, the columns will automatically stack on top of each other:

Example 3:

```
<div class="row">
  <div class="col-sm-4">.col-sm-4</div>
  <div class="col-sm-8">.col-sm-8</div>
</div>
```

The above example shows how to get two various-width columns starting at tablets and scaling to large extra desktops.

Module – 7:

HTTP & Browser Developer Tools: Understand HTTP Headers (Request & Response Headers), URL & its Anatomy, Developer Tools: Element/Inspector, Console, Network, Sources, Performance, Application Storage.

HTTP – Hyper Text Transfer Protocol

HTTP is a transfer protocol used by the World Wide Web to retrieve information from distributed servers. The HTTP model is extremely simple; the client establishes a connection to the remote server, then issues a request. The server then processes the request, returns a response, and closes the connection.

Requests

The request format for HTTP is quite simple. The first line specifies an object, together with the name of an object to apply the method to. The most commonly used method is "GET", which ask the server to send a copy of the object to the client.

The client can also send a series of optional headers; these headers are in RFC-822 format. The most common headers are "Accept", which tells the server which object types the client can handle, and "User-Agent", which gives the implementation name of the client.

Request syntax

```
<METHOD> <URI> "HTTP/1.0" <crlf>
{<Header>: <Value> <crlf>}*
<crlf>
```

Example

```
GET  /index.html HTTP/1.0
Accept: text/plain
Accept: text/html
Accept: */*
User-Agent:  Yet Another User Agent
```

Responses

The response format is also quite simple. Responses start with a status line indicating which version of HTTP the server is running, together with a result code and an optional message.

This is followed by a series of optional object headers; the most important of these are "Content-Type", which describes the type of the object being returned, and "Content-Length", which indicates the length.The headers are teminated by an empty line.

The server now sends any requested data. After the data have been sent, the server drops the connection.

Response syntax

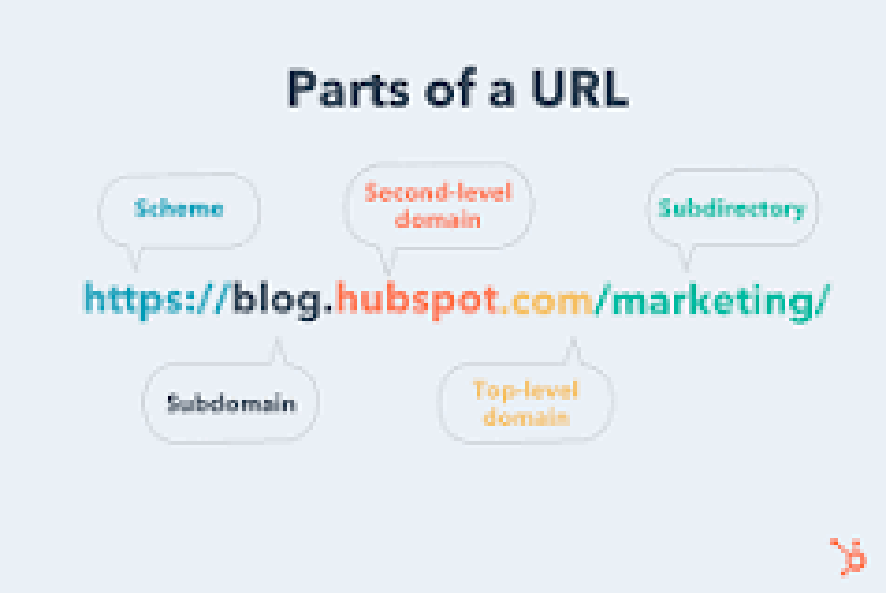
```
"HTTP/1.0" <result-code> [<message>]  <crlf>
{<Header>: <Value> <crlf>}*
<crlf>
```

Example

```
HTTP/1.0 200 OK
Server: MDMA/0.1
MIME-version: 1.0
Content-type: text/html
Last-Modified: Thu Jul  7 00:25:33 1994
Content-Length: 2003

<title>MDMA - Multithreaded Daemon for Multimedia Access</title>
<hr>
....
```

```
<hr>
<h2> MDMA - The speed daemon </h2>
<hr>

[Connection closed by foreign host]
```

What is the anatomy of a URL?



A URL consists of five parts: **the scheme, subdomain, top-level domain, second-level domain, and subdirectory**.

A URL, or uniform resource locator, is an address that helps your web browser locate a specific webpage, picture, file, or other resource.

Your browser takes the address, translates the domain name to the IP address of the server, and the rest of the URL shows the path to the specific file on that server.

A URL leads to a specific file or page, while a domain name is the general "address" to the whole website or server.

A URI (or a Uniform Resource Identifier) is different from a URL in that it refers to the unique ID of a file or resource, but not necessarily the means through which you can access it.

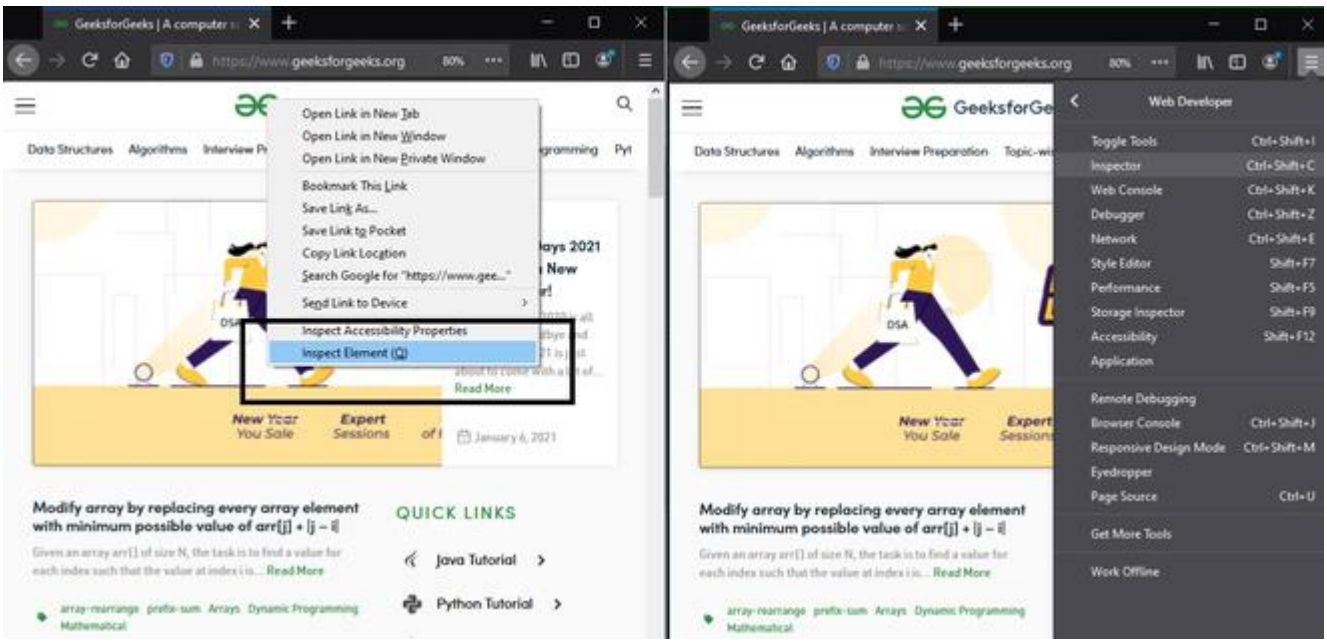
"A Uniform Resource Identifier (URI) is a compact sequence of characters that identifies an abstract or physical resource."

Browser Developer Tools

Every web-developer needs some basic set of tools for understanding the underlying structure of the code and enables us to inspect the web content. Developer tools are built directly into the browser. These are the tools that are browser dependent. Most of these tools are common among various browsers and do a range of things, from inspecting elements of a currently-loaded HTML, CSS, and JavaScript. With developer tools, we can directly interact with the source code that is fetched into the client side of our system.

How to open DevTools in the browser

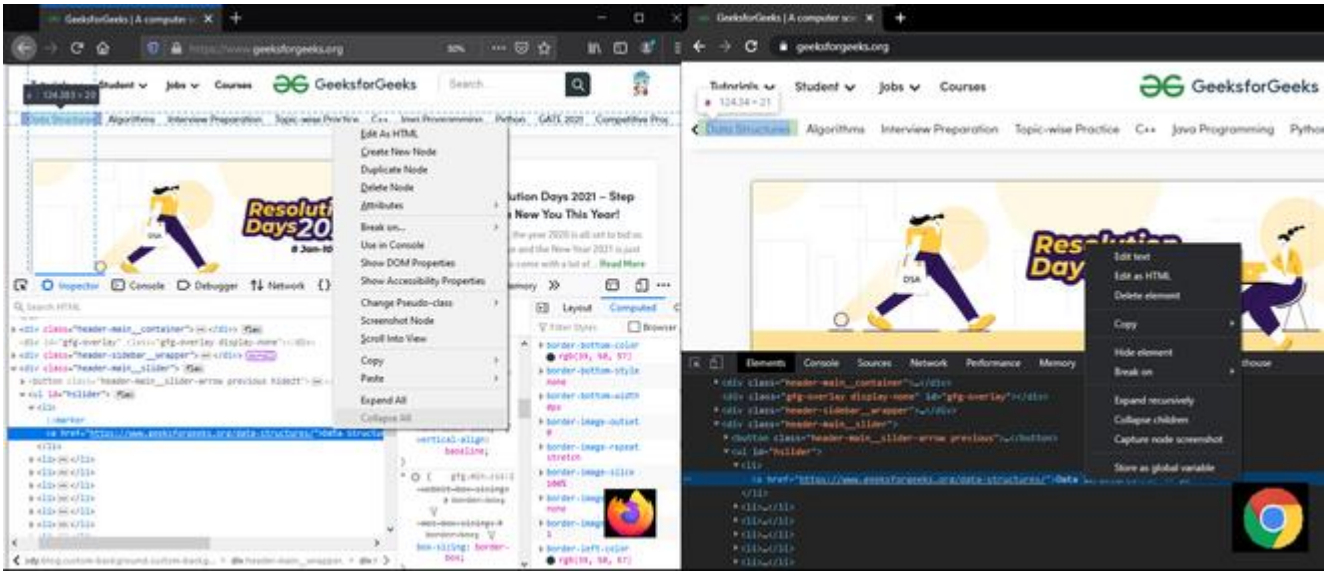
There are many ways to open the Browser Developer Tools.



- To access the DOM or CSS of the webpage, right-click the desired element on the page and select Inspect. Or press Command+Option+C (Mac), Control+Shift+C/I (Windows, Linux, Chrome OS), or press F12 (Internet Explorer, Edge).

1. Inspector

The Inspector tool allows you to see the HTML, CSS of the webpage that you are currently inspecting. With it, you can check what CSS is applied to each element on the page. It also allows you to preview instant changes to the HTML and CSS which are reflected live in the browser. These changes are not permanent and are reset once you refresh the browser window.



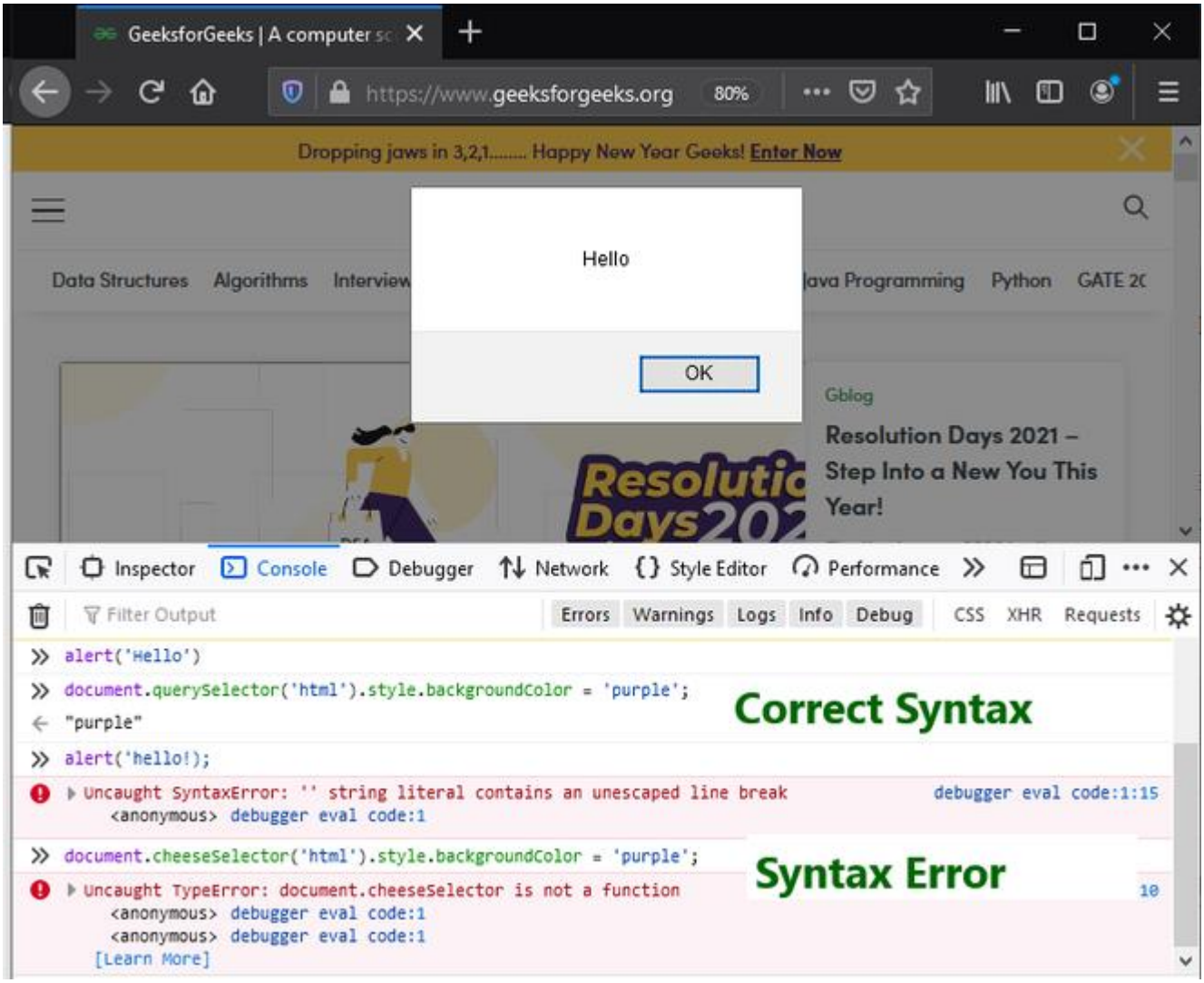
- **Edit as HTML (Add attribute/Edit text):** This allows to change the HTML and see the results in real-time. Most useful for debugging and testing.
- **Create Node:** Create another empty inner division.
- **Duplicate Node:** Create an exact copy of the division, along with the same attributes.
- **Delete Node:** Deletes the current element.
- **Copy/ Copy as HTML:** Copy HTML which is currently selected.
- **Change Pseudo-Class (:hover/:active/:focus):** It forces element states to be toggled on. Enabling us to view the particular styling on the element.
- **Copy CSS Path/ Copy XPath:** Most useful feature of Inspector (feature could be enabled manually) it allows to copy the CSS selector or XPath expression, this selects the current HTML element.

Application:

- Viewing and changing the DOM/ CSS
- Inspect and change HTML Pages.
- Inspect animations
- Find unused CSS.

2. Console

The console is used for debugging JavaScript present in the source code of the webpage. The console window act as our debug window that allows us to handle JavaScript that isn't working as expected. It allows you to either run a code block or single lines of JavaScript against the page which is currently loaded in the browser. The console reports the errors which are encountered by the browser as it tries to execute the code.



- The console recognized the correct syntax and generated an alert message corresponding to the JavaScript Code.
- It also recognized type errors and gave us an error message for the wrong syntax.

Application:

- Viewing logged messages
- Running JavaScript
- Preserve Log
- Group similar elements
- Log XmlHttpRequests
- Preserving live expression

3. Debugger(Firefox)/ Sources(Chrome)

The Sources/Debugger UI panel watches JavaScript code and allows you to set breakpoints and watch the value of variables. Breakpoints are set at the places in the code where we want to pause the execution cycle and debug or identify the problems with the execution. With this panel, you debug the JavaScript.

This panel has three parts:

1. File list(Firefox)/ File Navigator(Chrome): Lists every file associated with the page that we are debugging.

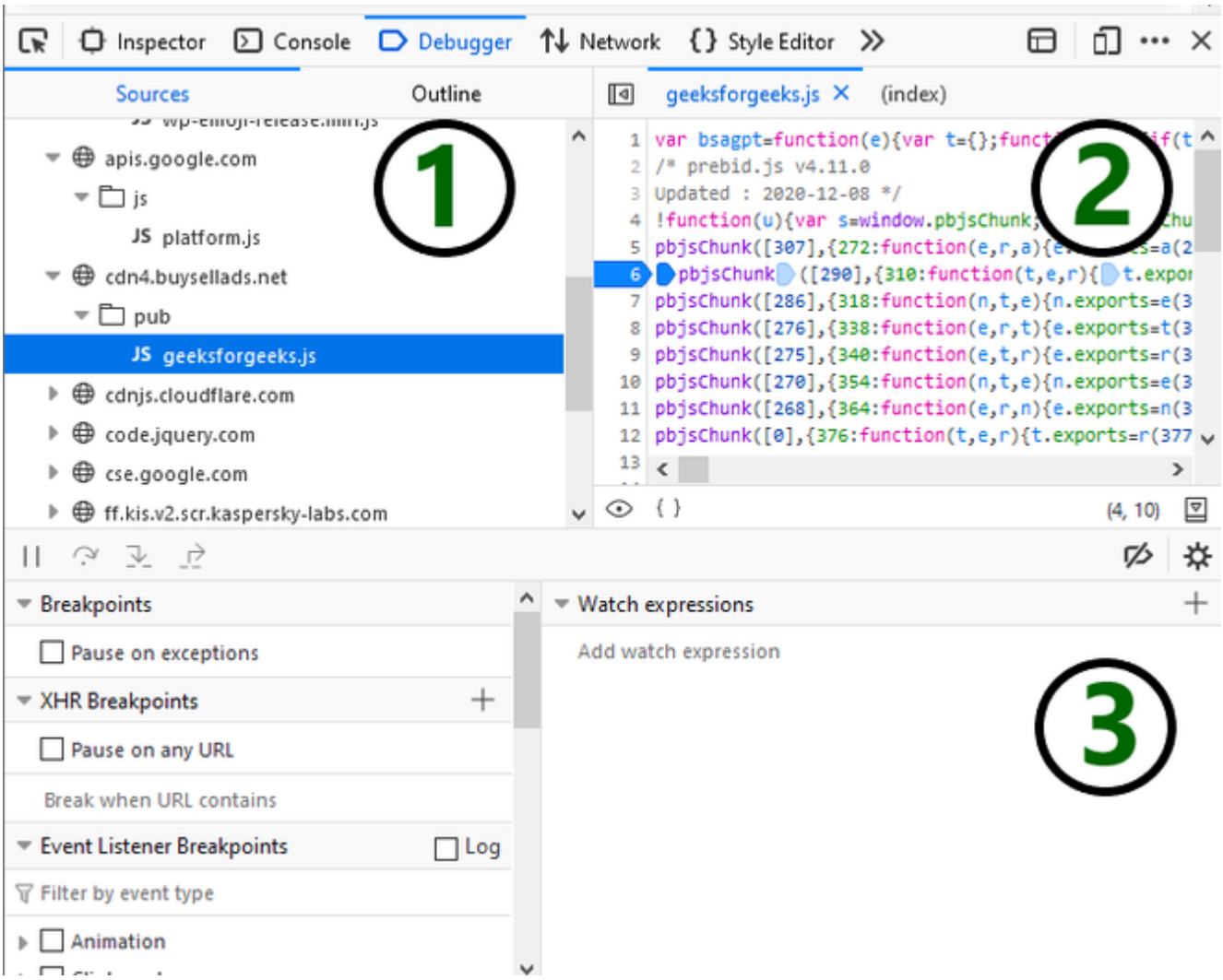
2. Source code(Firefox)/ Code Editor(Chrome): Contents of the selected file is displayed in this panel. Breakpoints could also be set in this panel, where we want to pause execution.

3. Watch expressions and breakpoints: These are the tools for inspecting the JavaScript of the page.

There are two more sections that only appear when the code is running.

4. Call Stack: It shows you what code was executed to get to the current line.

5. Scopes: It shows the values which are visible from a different perspective from within the given code.



Application:

- Pause code with breakpoints.
- Save changes to disk with workspaces.

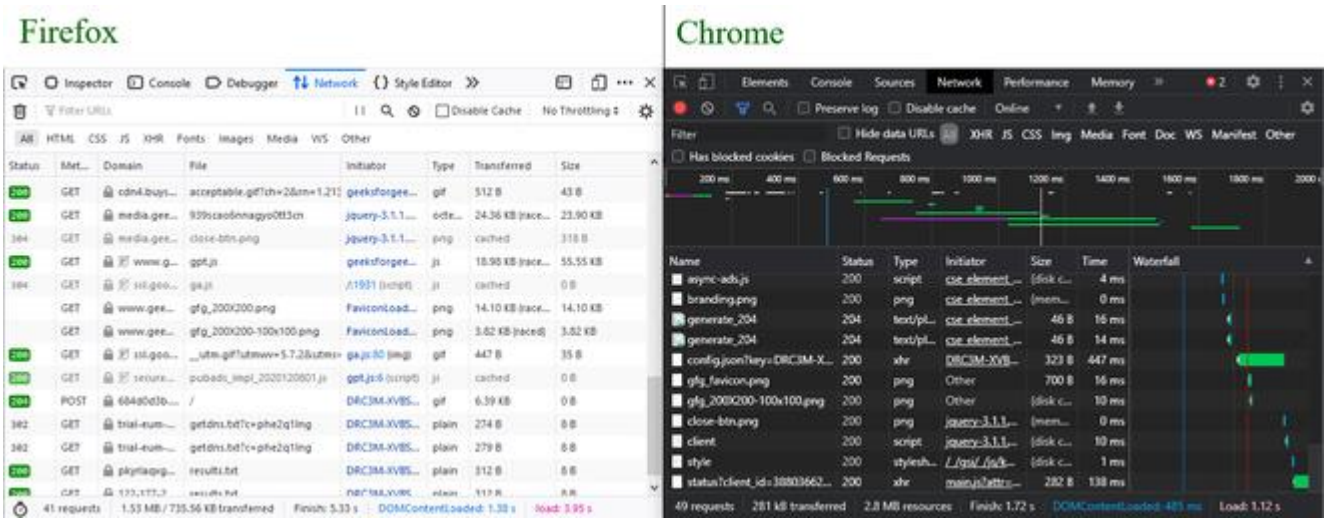
- Run snippets of code from any page.
- Find unused JavaScript.
- Persist changes across page reloads with local overrides.
- Get a JavaScript debugging reference.

3. Network Monitor

A Network panel is used to make sure what all resources that are being downloaded or uploaded are being done as expected. It enables us to view the network requests made when a page is loaded. It tells how long each request takes, and details of each request. The monitor is empty when it is loaded, the logs are created once any action is performed while the monitor is open.

Each row of Network Logs does represent a list of resources. These resources are listed according to the following structure:

1. **Status:** The HTTP response code.
2. **Type**: The resource type.
3. **Initiator:** The Initiator column is a hyperlink that takes you to the source code of the request.
4. **Time:** Time is taken by the request.
5. **Timeline/ Waterfall:** Graphical representation of the various stage of these requests.

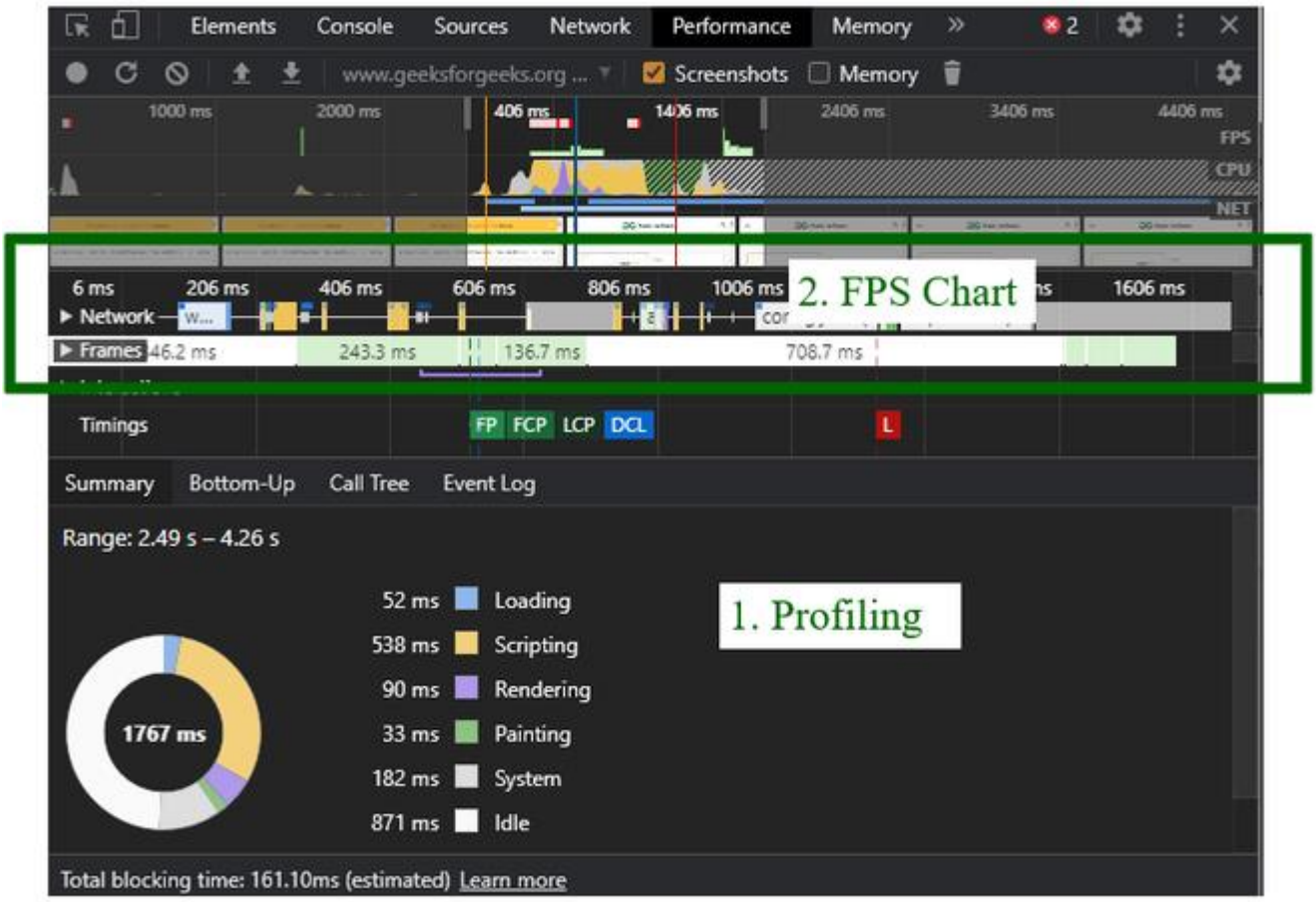


Application:

- Inspecting the properties of an individual resource (HTTP headers, content, size).
- Checking Network request list
- View Network traffic recording
- Creating Performance analysis
- Inspecting web sockets
- Inspecting server-sent events
- To throttle the network speed

4. Performance Tools

Performance is recorded at runtime and tells how the page performs when it is running, as opposed to loading. The tool gives a general insight into how good is the site's general responsiveness. It also measures the JavaScript & layout performance of the site. The tool creates a recording/ profile, of the website over a period of time, the tool is made to run. An overview is created using the RAIL model, listing all the frames of the browser activity which has been done to render the website.



Application:

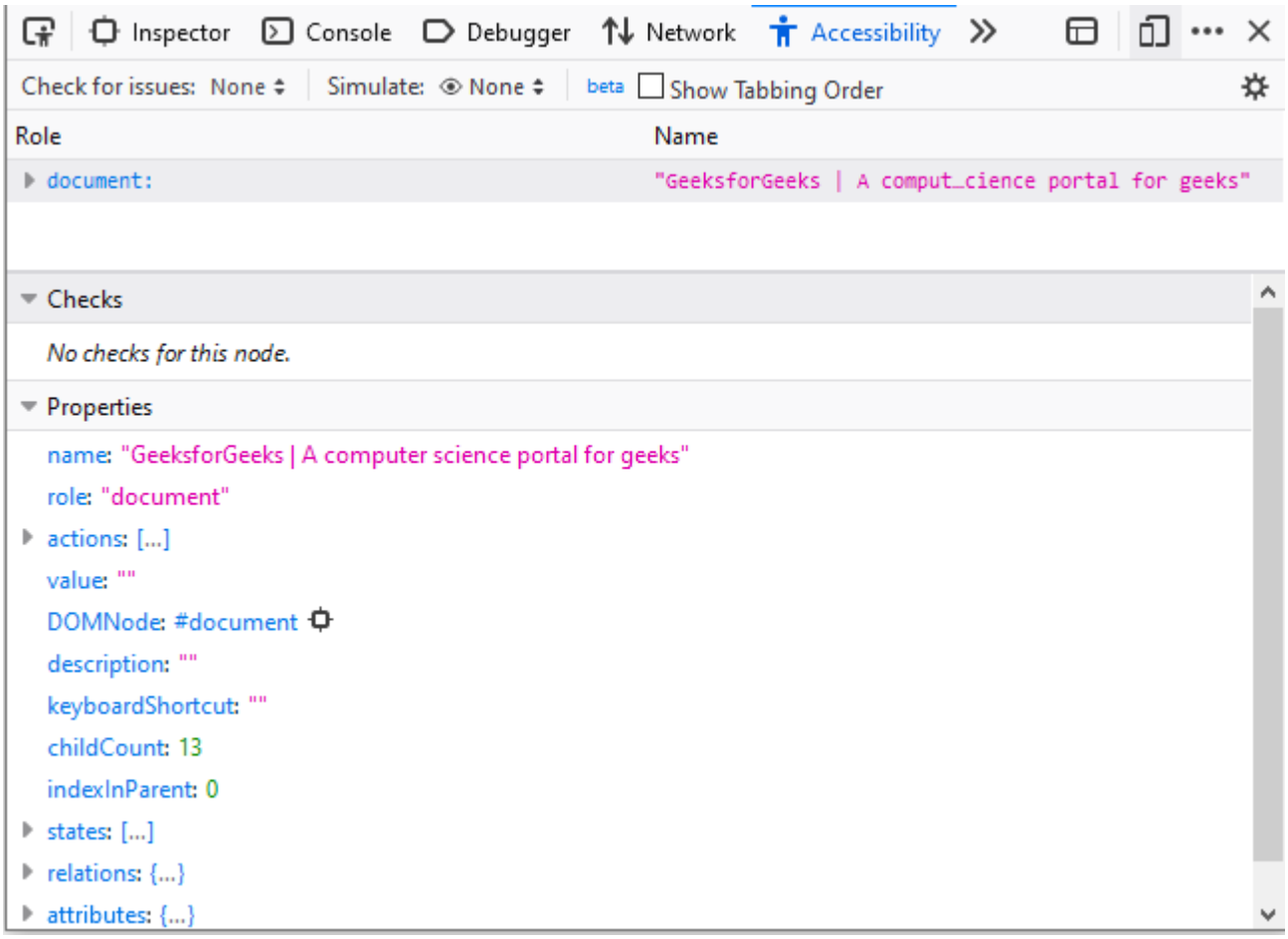
- Simulating a mobile CPU
- Recording runtime performance of our site
- Analyzing frame per second
- Finding the bottleneck capacity
- Animating CSS properties

5. Accessibility Inspector

This tool provides a way to access important information which is generally exposed to the assistive tech stack on the current view page via the accessibility tree present. It allows checking what element is missing or otherwise needs attention.

Accessibility is the practice of creating websites usable by as many of us, as much as possible. This tool tries the best to not prevent anyone from accessing information due to any kind of disability. Disability also includes any feature that a user is not able to use due to the lack of capabilities of the device which

is being used to access the site. This may include the speed of their network connection, or their geographic location or locale.



The accessibility window is present as a tree diagram, representing all the items on the current page. Items having the nested children got these arrows that can be clicked to reveal the nesting level with the children.

Applications:

- Fully keyboard controllable
- Print accessibility tree to JSON
- Show web page tabbing order
- Check for accessibility issues
- Simulate the webpage if viewed by a color-blind person
- Highlighting of UI items
- Check whether the item has the correct role set to it

Task: Analyze various HTTP requests (initiators, timing diagrams, responses) and identify problems if any.

1. What are different HTTP methods available to communicate with a server?

GET, POST, PUT, DELETE methods are the most famous http methods to communicate with server.

a. GET

According to the design of the HTTP specification, GET requests are used only to read data and not change it. So, they are considered safe. That is they can be called without risk of data modification or corruption—calling it once has the same effect as calling it 10 times.

The HTTP GET method is used to read (or retrieve) a representation of a resource. In case of success (or non-error), GET returns a representation in JSON and an HTTP response status code of 200 (OK). In an error case, it most often returns a 404 (NOT FOUND) or 400 (BAD REQUEST).

b. POST

The POST method is most often utilized to create new resources. In particular, it is used to create subordinate resources. That is subordinate to some other (e.g. parent) resource. In other words, when creating a new resource, POST to the parent and the service takes care of associating the new resource with the parent, assigning an ID (new resource URI), etc.

On successful creation, HTTP response code 201 is returned.

Caution:
POST is not a safe operation. Making two identical POST requests will most likely result in two resources containing the same information but with different identifiers.

c. PUT

PUT is most-often utilized for **update** capabilities, PUT-ing to a known resource URI with the request body containing the newly-updated representation of the original resource.

However, PUT can also be used to create a resource in the case where the resource ID is chosen by the client instead of by the server. In other words, if the PUT is to a URI that contains the value of a non-existent resource ID. Again, the request body contains a resource representation. Many feel this is convoluted and confusing. Consequently, this method of creation should be used sparingly, if at all.

Alternatively, use POST to create new resources and provide the client-defined ID in the body representation—presumably to a URI that doesn't include the ID of the resource (see POST below).

d. DELETE

DELETE is pretty easy to understand. It is used to **delete** a resource identified by a URI.

On successful deletion, return HTTP status 200 (OK) along with a response body, perhaps the representation of the deleted item (often demands too much bandwidth), or a wrapped response (see Return Values below). Either that or return HTTP status 204 (NO CONTENT) with no response body. In other words, a 204 status with no body, or the JSEND-style response and HTTP status 200 are the recommended responses.

2. What are the different response codes which you can receive from server during client-server communication?

HTTP response status codes indicate whether a specific HTTP request has been successfully completed. Responses are grouped in five classes:

Informational responses (100–199)
Successful responses (200–299)
Redirection messages (300–399)
Client error responses (400–499)
Server error responses (500–599)

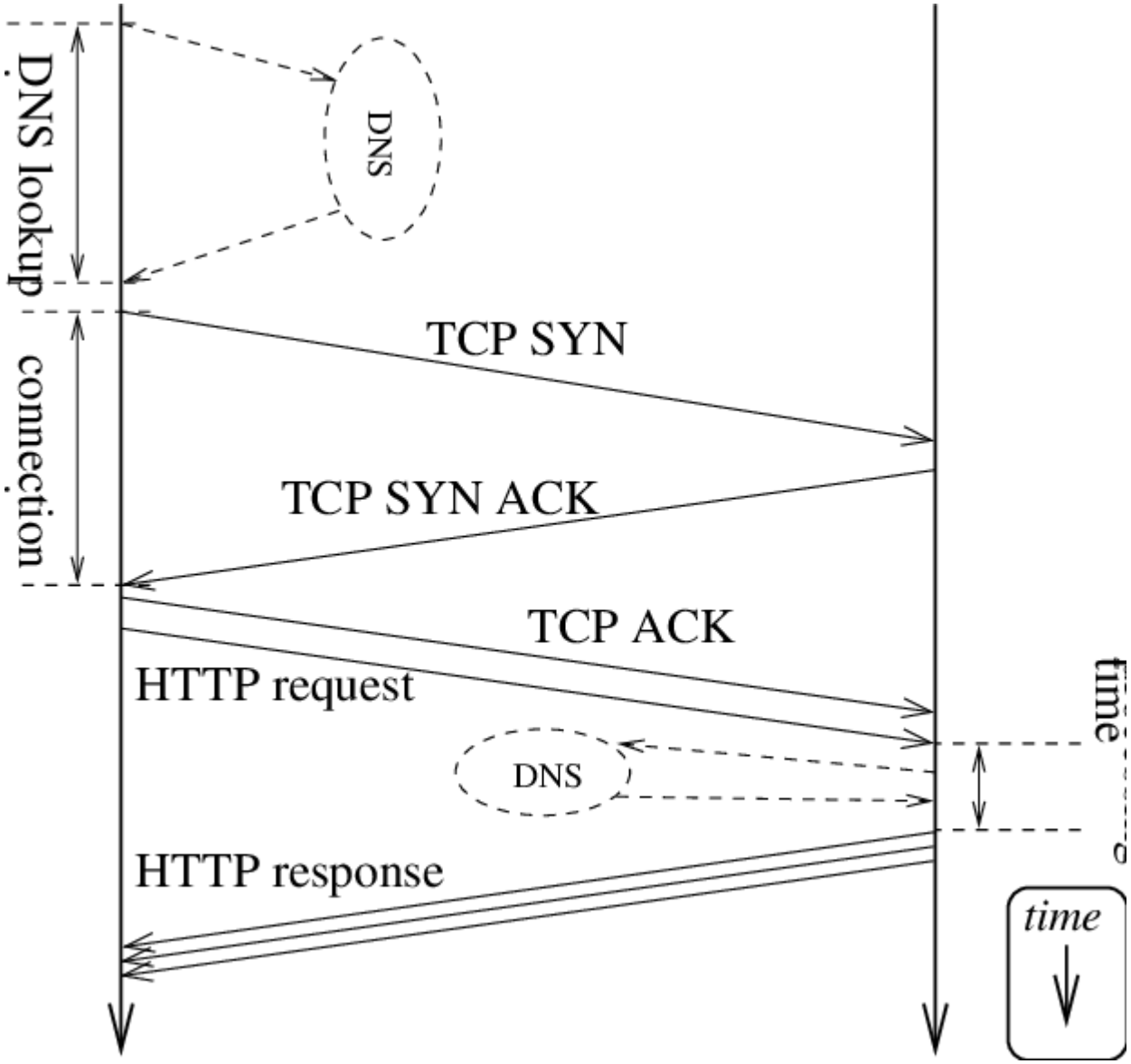
3. What are HTTP headers? Where and When do they get used in client-server communication?

HTTP headers let the client and the server pass additional information with an HTTP request or response. An HTTP header consists of its case-insensitive name followed by a colon (:), then by its value. Whitespace before the value is ignored.

Headers can be grouped according to their contexts:

- Request headers contain more information about the resource to be fetched, or about the client requesting the resource.
- Response headers hold additional information about the response, like its location or about the server providing it.
- Representation headers contain information about the body of the resource, like its MIME type, or encoding/compression applied.
- Payload headers contain representation-independent information about payload data, including content length and the encoding used for transport.

HTTP Timing Diagram



Module – 8:

JavaScript: Variables, Data Types, Operators, Statements, Objects, Functions, Events & Event Listeners, DOM.

Task: Design a simple calculator using JavaScript to perform sum, product, difference, and quotient operations.

Program1:

```
<html>
<head>
</head>
<body>
<script>
function insert(num){
```

```
document.form.textView.value=document.form.textView.value+num;
}
function equal(){
var exp = document.form.textView.value;
if(exp){
document.form.textView.value=eval(document.form.textView.value);
}
}
function c(){
document.form.textView.value=" ";
}
</script>
<center>
<form name="form">
<input type="text" name="textView"/><br>
<input type="button" value="1" onclick="insert(1)"/>
<input type="button" value="2" onclick="insert(2)"/>
<input type="button" value="3" onclick="insert(3)"/>
<input type="button" value="+" onclick="insert('+')"/>
<br><br>
<input type="button" value="4" onclick="insert(4)"/>
<input type="button" value="5" onclick="insert(5)"/>
<input type="button" value="6" onclick="insert(6)"/>
<input type="button" value="-" onclick="insert('-')"/>
<br><br>
<input type="button" value="6" onclick="insert(6)"/>
<input type="button" value="7" onclick="insert(7)"/>
<input type="button" value="8" onclick="insert(8)"/>
<input type="button" value="*" onclick="insert('*')"/>
<br><br>
<input type="button" value="0" onclick="insert(0)"/>
<input type="button" value="/" onclick="insert('/')"/>
<input type="button" value="=" onclick="equal()"/>
<input type="button" value="C" onclick="c()"/>
<br>
</form>
</center>
</body>
</html>
```

Another Design of Simple Calculator

```
!-- Create a simple Program to build the Calculator in JavaScript using with HTML and
CSS web languages. -->
<!DOCTYPE html>
<html lang = "en">
<head>
<title> JavaScript Calculator </title>
```

```
<style>
h1 {
        text-align: center;
        padding: 23px;
        background-color: skyblue;
        color: white;
        }

#clear{
width: 270px;
border: 3px solid gray;
        border-radius: 3px;
        padding: 20px;
        background-color: red;
}

.formstyle
{
width: 300px;
height: 530px;
margin: auto;
border: 3px solid skyblue;
border-radius: 5px;
padding: 20px;
}



input
{
width: 20px;
background-color: green;
color: white;
border: 3px solid gray;
        border-radius: 5px;
        padding: 26px;
        margin: 5px;
        font-size: 15px;
}


#calc{
width: 250px;
border: 5px solid black;
        border-radius: 3px;
        padding: 20px;
        margin: auto;
}
```

```
</style>

</head>
<body>
<h1> Calculator Program in JavaScript </h1>
<div class= "formstyle">
<form name = "form1">

        <!-- This input box shows the button pressed by the user in calculator. -->
  <input id = "calc" type ="text" name = "answer"> <br> <br>
  <!-- Display the calculator button on the screen. -->
  <!-- onclick() function display the number prsses by the user. -->
  <input type = "button" value = "1" onclick = "form1.answer.value += '1' ">
  <input type = "button" value = "2" onclick = "form1.answer.value += '2' ">
  <input type = "button" value = "3" onclick = "form1.answer.value += '3' ">
   <input type = "button" value = "+" onclick = "form1.answer.value += '+' ">
  <br> <br>

  <input type = "button" value = "4" onclick = "form1.answer.value += '4' ">
  <input type = "button" value = "5" onclick = "form1.answer.value += '5' ">
  <input type = "button" value = "6" onclick = "form1.answer.value += '6' ">
  <input type = "button" value = "-" onclick = "form1.answer.value += '-' ">
  <br> <br>

  <input type = "button" value = "7" onclick = "form1.answer.value += '7' ">
  <input type = "button" value = "8" onclick = "form1.answer.value += '8' ">
  <input type = "button" value = "9" onclick = "form1.answer.value += '9' ">
  <input type = "button" value = "*" onclick = "form1.answer.value += '*' ">
  <br> <br>


  <input type = "button" value = "/" onclick = "form1.answer.value += '/' ">
  <input type = "button" value = "0" onclick = "form1.answer.value += '0' ">
    <input type = "button" value = "." onclick = "form1.answer.value += '.' ">
        <!-- When we click on the '=' button, the onclick() shows the sum results on the
calculator screen. -->
  <input type = "button" value = "=" onclick = "form1.answer.value =
eval(form1.answer.value) ">
  <br>
  <!-- Display the Cancel button and erase all data entered by the user. -->
  <input type = "button" value = "Clear All" onclick = "form1.answer.value = ' ' " id=
"clear" >
  <br>

</form>
</div>
</body>
</html>
```

Module - 9:

Dynamic HTML with JavaScript: Manipulate DOM, Error Handling, Promises, async/await, Modules.

Task: Design& develop a Shopping Cart Application with features including Add Products, Update Quantity, Display Price (Sub-Total & Total), Remove items/products from the cart.

Main file : Index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="author" content="CodeHim">
    <title>Shopping Cart Example</title>

    <!-- Style CSS -->
    <link rel="stylesheet" href="css/style.css">
        <!-- Demo CSS (No need to include it into your project) -->
        <link rel="stylesheet" href="css/demo.css">

  </head>
  <body>
 <header class="intro">
 <h1>Shopping Cart Example</h1>
 <p>A shopping cart system in HTML CSS &amp; JavaScript.</p>


 </header>


 <main>
     <!-- Start DEMO HTML (Use the following code into your project)-->
    <div class="container">
   <h1>Simple JavaScript Shopping Cart</h1>
   <p>Simplified 'add to cart' functionality. Uses Javascript
   and WebStorage API/Cookies to remember cart data converted to JSON format.</p>
   <p>Click 'Empty Cart' button to remove session cookies from browser.</p>
   <div id="alerts"></div>
<div class="productcont">
       <div class="product">
           <h3 class="productname">Product 1</h3>
           <p>Product description excerpt...</p>
           <p class="price">$5.05</p>
           <button class="addtocart">Add To Cart</button>
```

```
        </div>
        <div class="product">
            <h3 class="productname">Product 2</h3>
            <p>Product description excerpt...</p>
            <p class="price">$8.50</p>
            <button class="addtocart">Add To Cart</button>
        </div>
        <div class="product">
            <h3 class="productname">Product 3</h3>
            <p>Product description excerpt...</p>
            <p class="price">$10.50</p>
            <button class="addtocart">Add To Cart</button>
        </div>
        </div>
        <div class="cart-container">
          <h2>Cart</h2>
          <table>
            <thead>
              <tr>
              <th><strong>Product</strong></th>
              <th><strong>Price</strong></th>
            </tr>
            </thead>
            <tbody id="carttable">
            </tbody>
          </table>
          <hr>
          <table id="carttotals">
            <tr>
              <td><strong>Items</strong></td>
              <td><strong>Total</strong></td>
            </tr>
            <tr>
              <td>x <span id="itemsquantity">0</span></td>

              <td>$<span id="total">0</span></td>
            </tr></table>


          <div class="cart-buttons">
            <button id="emptycart">Empty Cart</button>
            <button id="checkout">Checkout</button>
          </div>
        </div>
</div>
<!-- partial -->

    <!-- END DEMO HTML (Happy Coding!)-->
 </main>
```

```
  <footer class="credit">Author: P Nageswara Rao - Distributed By: <a title="Awesome web design code & scripts" href="https://pnrbtech.blogspot.com" target="_blank">PJS</a></footer>

      <script  src="./js/script.js"></script>
  </body>
</html>
```

Supporting files:

Demo.css

```
@import url('https://fonts.googleapis.com/css?family=Roboto&display=swap');
@import url('https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css');
*{ margin: 0; padding: 0;}
body{
  font-family: 'Roboto', sans-serif;
  font-style: normal;
  font-weight: 300;
  font-smoothing: antialiased;
-webkit-font-smoothing: antialiased;
-moz-osx-font-smoothing: grayscale;
  font-size: 15px;
}
.intro{
   background: #fff;
   padding: 60px 30px;
   color: #333;
   margin-bottom: 15px;
   line-height: 1.5;
   text-align: center;
}
.intro h1 {
   font-size: 18pt;
   padding-bottom: 15px;

}
.intro p{
   font-size: 14px;
}

.action{
   text-align: center;
   display: block;
   margin-top: 20px;
}
```

```
a.btn {
  text-decoration: none;
  color: #666;
   border: 2px solid #666;
   padding: 10px 15px;
   display: inline-block;
   margin-left: 5px;
}
a.btn:hover{
   background: #666;
   color: #fff;
    transition: .3s;
-webkit-transition: .3s;
}
.btn:before{
  font-family: FontAwesome;
  font-weight: normal;
  margin-right: 10px;
}
.github:before{content: "\f09b"}
.down:before{content: "\f019"}
.back:before{content:"\f112"}
.credit{
    background: #fff;
    padding: 12px;
    font-size: 9pt;
    text-align: center;
    color: #333;
    margin-top: 40px;

}
.credit span:before{
   font-family: FontAwesome;
   color: #e41b17;
   content: "\f004";


}
.credit a{
   color: #333;
   text-decoration: none;
}
.credit a:hover{
   color: #1DBF73;
}
.credit a:hover:after{
    font-family: FontAwesome;
    content: "\f08e";
    font-size: 9pt;
```

```
    position: absolute;
    margin: 3px;
}
main{
  background: #fff;
  padding:: 20px;

}

article li{
   color: #444;
   font-size: 15px;
   margin-left: 33px;
   line-height: 1.5;
   padding: 5px;
}
article h1,
article h2,
article h3,
article h4,
article p{
   padding: 14px;
   color: #333;
}
article p{
   font-size: 15px;
    line-height: 1.5;
}

@media only screen and (min-width: 720px){
    main{
      max-width: 720px;
      margin-left: auto;
      margin-right: auto;
      padding: 24px;
    }


}

.set-overlayer,
.set-glass,
.set-sticky {
         cursor: pointer;
         height: 45px;
         line-height: 45px;
         padding: 0 15px;
         color: #333;
         font-size: 16px;
```

```
}
.set-overlayer:after,
.set-glass:after,
.to-active:after,
.set-sticky:after {
        font-family: FontAwesome;
        font-size: 18pt;
        position: relative;
        float: right;
}
.set-overlayer:after,
.set-glass:after,
.set-sticky:after {
        content: "\f204";
        transition: .6s;
}

.to-active:after {
        content: "\f205";
        color: #008080;
        transition: .6s;
}
.set-overlayer,
.set-glass,
.set-sticky,
.source,
.theme-tray {
        margin: 10px;
        background: #f2f2f2;
        border-radius: 5px;
        border: 2px solid #f1f1f1;
        box-sizing: border-box;
}
/* Syntax Highlighter*/

pre.prettyprint {
        padding: 15px !important;
   margin: 10px;
        border: 0 !important;
  background: #f2f2f2;
  overflow: auto;
}

.source {
        white-space: pre;
        overflow: auto;
        max-height: 400px;
}
code{
```

```
    border:1px solid #ddd;
    padding: 2px;
    border-radius: 2px;
}
```

Style.css

```
@import url('https://fonts.googleapis.com/css?family=Quicksand:400,700');
*, ::before, ::after { box-sizing: border-box; }
body{
  font-family:'Quicksand', sans-serif;
  text-align:center;
  line-height:1.5em;
/*   background:#E0E4CC; */
 background: #69d2e7;
background: -moz-linear-gradient(-45deg, #69d2e7 0%, #a7dbd8 25%, #e0e4cc 46%, #e0e4cc 54%, #f38630 75%, #fa6900 100%);
background: -webkit-linear-gradient(-45deg, #69d2e7 0%,#a7dbd8 25%,#e0e4cc 46%,#e0e4cc 54%,#f38630 75%,#fa6900 100%);
background: linear-gradient(135deg, #69d2e7 0%,#a7dbd8 25%,#e0e4cc 46%,#e0e4cc 54%,#f38630 75%,#fa6900 100%);
filter: progid:DXImageTransform.Microsoft.gradient( startColorstr='#69d2e7', endColorstr='#fa6900',GradientType=1 );
}
hr {
  border:none;
  background:#E0E4CC;
  height:1px;
/*   width:60%;
  display:block;
  margin-left:0; */
}
h1{
    margin: 12px;
}
.container {
  max-width: 800px;
  margin: 1em auto;
  background:#FFF;
  padding:30px;
  border-radius:5px;
}
.productcont {
  display: flex;
}
.product {
  padding:1em;
  border:1px solid #E0E4CC;
  margin-right:1em;
```

```
  border-radius:5px;
}
.cart-container {
  border:1px solid #E0E4CC;
  border-radius:5px;
  margin-top:1em;
  padding:1em;
}
button, input[type="submit"] {
    border:1px solid #FA6900;
    color:#fff;
    background: #F38630;
    padding:0.6em 1em;
    font-size:1em;
    line-height:1;
    border-radius: 1.2em;
    transition: color 0.2s ease-in-out, background 0.2s ease-in-out, border-color 0.2s
ease-in-out;
}
button:hover, button:focus, button:active, input[type="submit"]:hover,
input[type="submit"]:focus, input[type="submit"]:active {
    background: #A7DBD8;
    border-color:#69D2E7;
    color:#000;
    cursor: pointer;
}
table {
  margin-bottom:1em;
  border-collapse:collapse;
}
table td, table th {
  text-align:left;
  padding:0.25em 1em;
  border-bottom:1px solid #E0E4CC;
}
#carttotals {
  margin-right:0;
  margin-left:auto;
}
.cart-buttons {
  width:auto;
  margin-right:0;
  margin-left:auto;
  display:flex;
  justify-content:flex-end;
  padding:1em 0;
}
#emptycart {
  margin-right:1em;
```

```
}
table td:nth-last-child(1) {
  text-align:right;
}
.message {
  border-width: 1px 0px;
  border-style:solid;
  border-color:#A7DBD8;
  color:#679996;
  padding:0.5em 0;
  margin:1em 0;
}
```

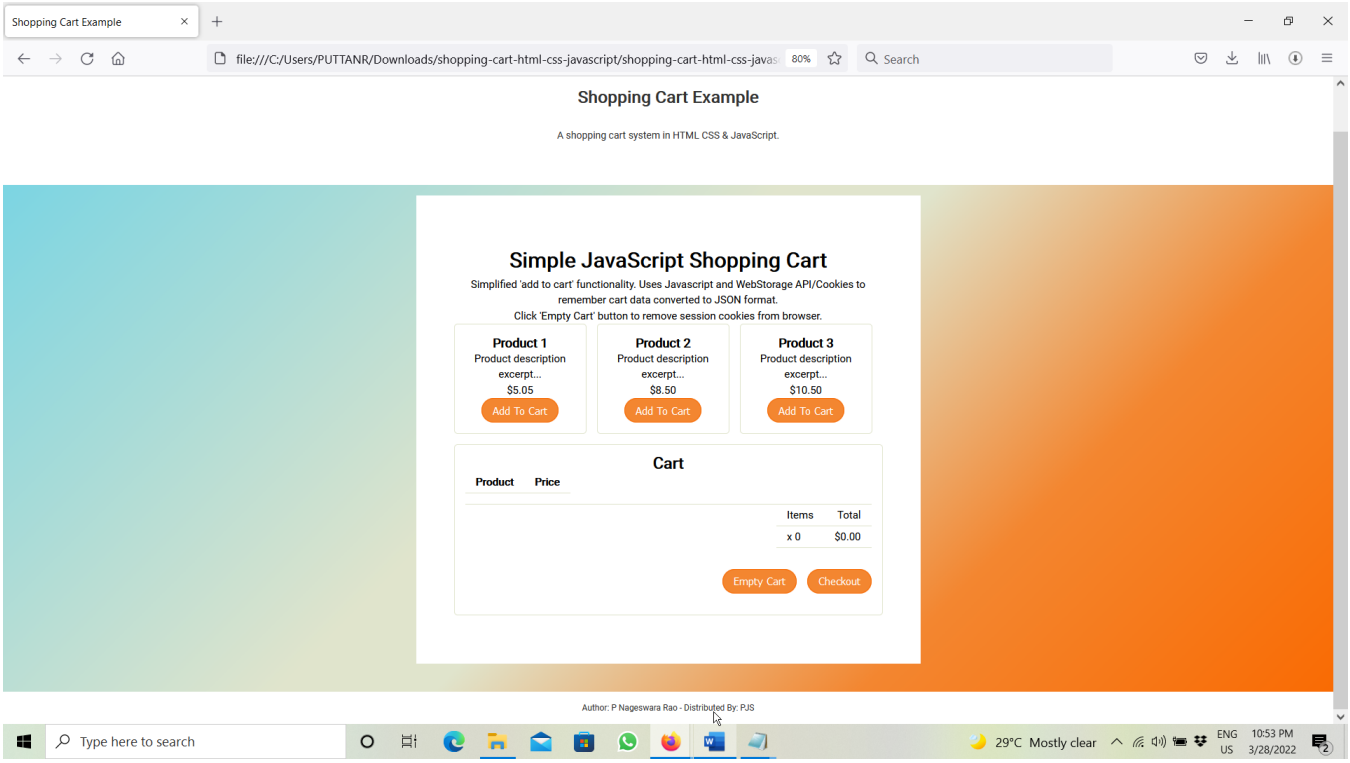
Script.js

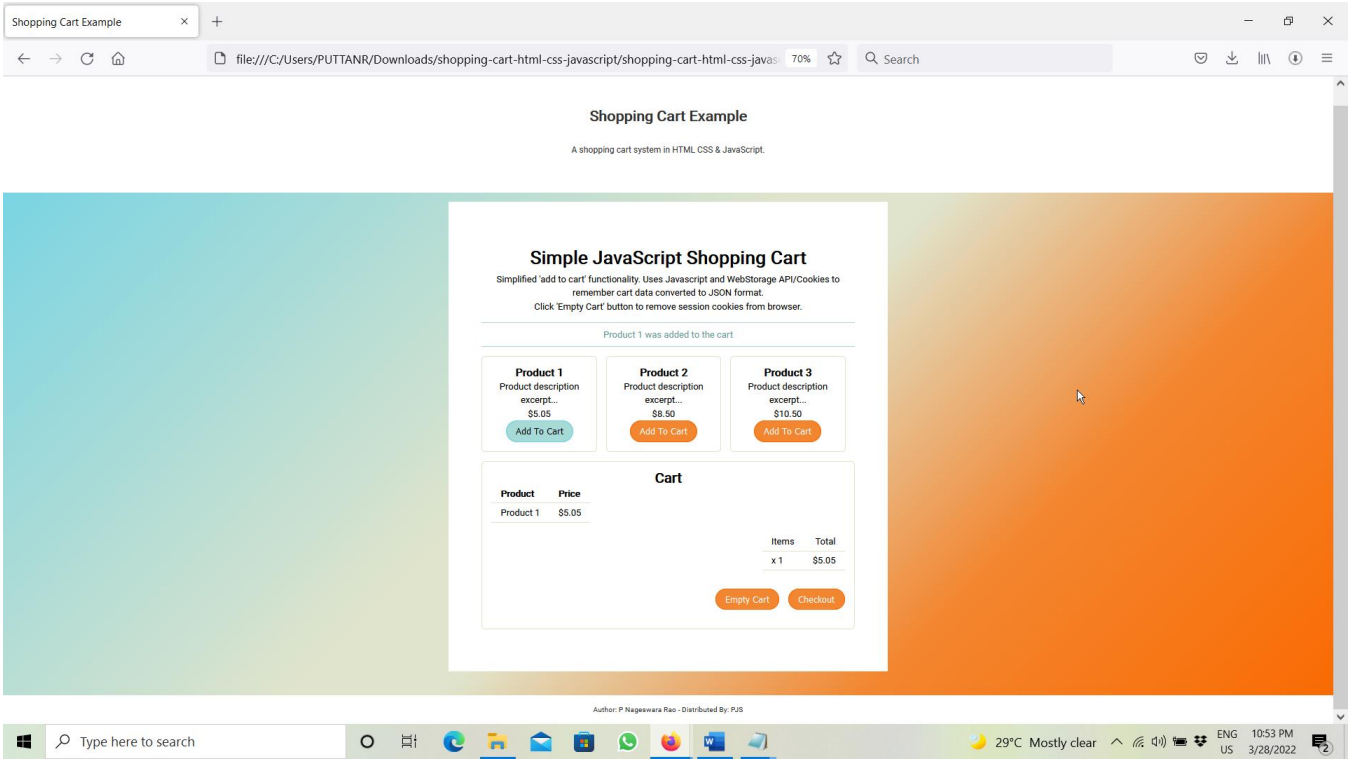
```
/* get cart total from session on load */
updateCartTotal();

/* button event listeners */
document.getElementById("emptycart").addEventListener("click", emptyCart);
var btns = document.getElementsByClassName('addtocart');
for (var i = 0; i < btns.length; i++) {
    btns[i].addEventListener('click', function() {addToCart(this);});
}

/* ADD TO CART functions */

function addToCart(elem) {
    //init
    var sibs = [];
    var getprice;
    var getproductName;
    var cart = [];
     var stringCart;
    //cycles siblings for product info near the add button
    while(elem = elem.previousSibling) {
        if (elem.nodeType === 3) continue; // text node
        if(elem.className == "price"){
            getprice = elem.innerText;
        }
        if (elem.className == "productname") {
            getproductName = elem.innerText;
        }
        sibs.push(elem);
    }
    //create product object
    var product = {
        productname : getproductName,
        price : getprice
    };
```

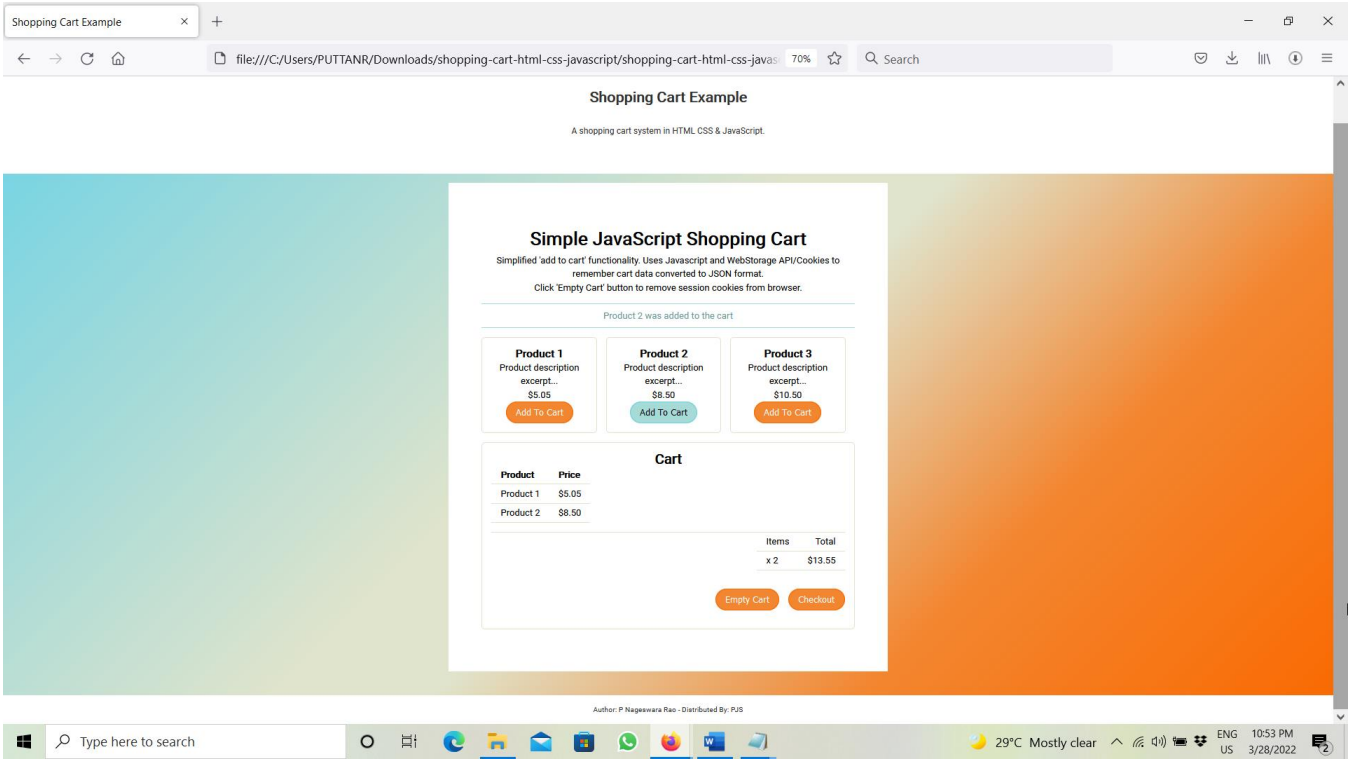
```
    //convert product data to JSON for storage
    var stringProduct = JSON.stringify(product);
    /*send product data to session storage */

    if(!sessionStorage.getItem('cart')){
        //append product JSON object to cart array
        cart.push(stringProduct);
        //cart to JSON
        stringCart = JSON.stringify(cart);
        //create session storage cart item
        sessionStorage.setItem('cart', stringCart);
        addedToCart(getproductName);
        updateCartTotal();
    }
    else {
        //get existing cart data from storage and convert back into array
       cart = JSON.parse(sessionStorage.getItem('cart'));
        //append new product JSON object
        cart.push(stringProduct);
        //cart back to JSON
        stringCart = JSON.stringify(cart);
        //overwrite cart data in sessionstorage
        sessionStorage.setItem('cart', stringCart);
        addedToCart(getproductName);
        updateCartTotal();
    }
}
/* Calculate Cart Total */
function updateCartTotal(){
    //init
    var total = 0;
    var price = 0;
    var items = 0;
    var productname = "";
    var carttable = "";
    if(sessionStorage.getItem('cart')) {
        //get cart data & parse to array
        var cart = JSON.parse(sessionStorage.getItem('cart'));
        //get no of items in cart
        items = cart.length;
        //loop over cart array
        for (var i = 0; i < items; i++){
            //convert each JSON product in array back into object
            var x = JSON.parse(cart[i]);
            //get property value of price
            price = parseFloat(x.price.split('$')[1]);
            productname = x.productname;
            //add price to total
```

```
            carttable += "<tr><td>" + productname + "</td><td>$" + price.toFixed(2) +
"</td></tr>";
            total += price;
        }

    }
    //update total on website HTML
    document.getElementById("total").innerHTML = total.toFixed(2);
    //insert saved products to cart table
    document.getElementById("carttable").innerHTML = carttable;
    //update items in cart on website HTML
    document.getElementById("itemsquantity").innerHTML = items;
}
//user feedback on successful add
function addedToCart(pname) {
  var message = pname + " was added to the cart";
  var alerts = document.getElementById("alerts");
  alerts.innerHTML = message;
  if(!alerts.classList.contains("message")){
     alerts.classList.add("message");
  }
}
/* User Manually empty cart */
function emptyCart() {
    //remove cart session storage object & refresh cart totals
    if(sessionStorage.getItem('cart')){
        sessionStorage.removeItem('cart');
        updateCartTotal();
      //clear message and remove class style
      var alerts = document.getElementById("alerts");
      alerts.innerHTML = "";
      if(alerts.classList.contains("message")){
          alerts.classList.remove("message");
      }
    }
}
```

Output:


Shopping Cart Example
file:///C:/Users/PUTTANR/Downloads/shopping-cart-html-css-javascript/shopping-cart-html-css-javas
80%
Search
Shopping Cart Example
A shopping cart system in HTML CSS & JavaScript.
Simple JavaScript Shopping Cart
Simplified 'add to cart' functionality. Uses Javascript and WebStorage API/Cookies to remember cart data converted to JSON format.
Click 'Empty Cart' button to remove session cookies from browser.
Product 1
Product description excerpt...
$5.05
Add To Cart
Product 2
Product description excerpt...
$8.50
Add To Cart
Product 3
Product description excerpt...
$10.50
Add To Cart
Cart
Product
Price
Items
Total
x 0
$0.00
Empty Cart
Checkout
Author: P Nageswara Rao - Distributed By: PJS
Type here to search
29°C Mostly clear
ENG US
10:53 PM
3/28/2022


Shopping Cart Example
file:///C:/Users/PUTTANR/Downloads/shopping-cart-html-css-javascript/shopping-cart-html-css-javas
70%
Search
Shopping Cart Example
A shopping cart system in HTML CSS & JavaScript.
Simple JavaScript Shopping Cart
Simplified 'add to cart' functionality. Uses Javascript and WebStorage API/Cookies to remember cart data converted to JSON format.
Click 'Empty Cart' button to remove session cookies from browser.
Product 1 was added to the cart
Product 1
Product description excerpt...
$5.05
Add To Cart
Product 2
Product description excerpt...
$8.50
Add To Cart
Product 3
Product description excerpt...
$10.50
Add To Cart
Cart
Product
Price
Product 1
$5.05
Items
Total
x 1
$5.05
Empty Cart
Checkout
Author: P Nageswara Rao - Distributed By: PJS
Type here to search
29°C Mostly clear
ENG US
10:53 PM
3/28/2022



Module - 10:

JQuery - A Javascript Library: Interactions, Widgets, Effects, Utilities, Ajax using JQuery.

Task: Validate all Fields and Submit the Hostel Application Form designed in Module-6 using JQuery

Registration.html

```
<!DOCTYPE html>
<html>
<head>
<title>Registration Form Using jQuery - Demo Preview</title>
<meta name="robots" content="noindex, nofollow">
<!-- Include CSS File Here -->
<link rel="stylesheet" href="css/style.css"/>
<!-- Include JS File Here -->
<script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.11.1/jquery.min.js"></script>
<script type="text/javascript" src="registration.js"></script>
</head>
<body>
<div class="container">
<div class="main">
<form class="form" method="post" action="#">
<h2>Create Registration Form Using jQuery</h2>
<label>Name :</label>
```

```
<input type="text" name="dname" id="name">
<label>Email :</label>
<input type="text" name="demail" id="email">
<label>Password :</label>
<input type="password" name="password" id="password">
<label>Confirm Password :</label>
<input type="password" name="cpassword" id="cpassword">
<input type="button" name="register" id="register" value="Register">
</form>
</div>
</body>
</html>
```

Registration.js

```
$(document).ready(function() {
$("#register").click(function() {
var name = $("#name").val();
var email = $("#email").val();
var password = $("#password").val();
var cpassword = $("#cpassword").val();
if (name == '' || email == '' || password == '' || cpassword == '') {
alert("Please fill all fields...!!!!!!");
} else if ((password.length) < 8) {
alert("Password should atleast 8 character in length...!!!!!!");
} else if (!(password).match(cpassword)) {
alert("Your passwords don't match. Try again?");
} else {
$.post("register.php", {
name1: name,
email1: email,
password1: password
}, function(data) {
if (data == 'You have Successfully Registered.....') {
$("form")[0].reset();
}
alert(data);
});
}
});
});
```

Register.php

```
<?php
$connection = mysql_connect("localhost", "root", ""); // Establishing connection with server..
$db = mysql_select_db("college", $connection); // Selecting Database.
$name=$_POST['name1']; // Fetching Values from URL.
```

```
$email=$_POST['email1'];
$password= sha1($_POST['password1']); // Password Encryption, If you like you can also leave sha1.
// Check if e-mail address syntax is valid or not
$email = filter_var($email, FILTER_SANITIZE_EMAIL); // Sanitizing email(Remove unexpected symbol like <,>,?,#,!, etc.)
if (!filter_var($email, FILTER_VALIDATE_EMAIL)){
echo "Invalid Email.....";
}else{
$result = mysql_query("SELECT * FROM registration WHERE email='$email'");
$data = mysql_num_rows($result);
if(($data)==0){
$query = mysql_query("insert into registration(name, email, password) values ('$name', '$email', '$password')"); // Insert query
if($query){
echo "You have Successfully Registered.....";
}else
{
echo "Error....!!";
}
}else{
echo "This email is already registered, Please try another email...";
}
}
mysql_close ($connection);
?>
```

Module - 11:

Google Charts: Understand the Usage of Pie chart, Bar Chart, Histogram, Area & Line Charts, Gantt Charts.

Pie Chart:

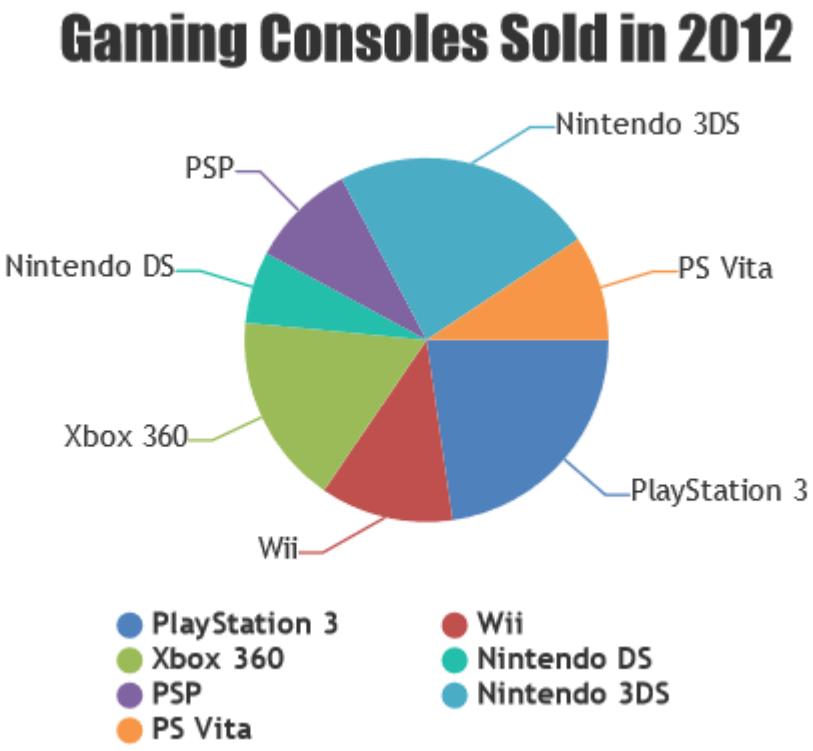
A pie chart is a circular chart divided into sectors, each sector (and consequently its central angle and area), is proportional to the quantity it represents. Together, the sectors create a full disk.

IndexLabels describes each slice of pie chart. It is displayed next to each slice. If indexLabel is not provided, label property is used as indexLabel. If labels are not provided, y value is used as index label.

Pie Chart Specific Properties

Applies To	Attribute	Type	Default	Options/Examples
dataSeries/ dataPoint	exploded	Boolean	false	false, true
dataSeries/ dataPoint	explodeOnClick	Boolean	true	false, true
dataSeries/ dataPoint	startAngle	Number	0	30, -45, 140..
dataSeries	fillOpacity	Number	1 .	2,.4,1 etc
dataPoint	legendText	String	“dataPoint1”	“apple”, “mango”...

```
<!DOCTYPE HTML>
<html>
<head>
<script type="text/javascript">
window.onload = function () {
	var chart = new CanvasJS.Chart("chartContainer",
	{
		title:{
			text: "Gaming Consoles Sold in 2012"
		},
		legend: {
			maxWidth: 350,
			itemWidth: 120
		},
		data: [
		{
			type: "pie",
			showInLegend: true,
			legendText: "{indexLabel}",
			dataPoints: [
				{ y: 4181563, indexLabel: "PlayStation 3" },
				{ y: 2175498, indexLabel: "Wii" },
				{ y: 3125844, indexLabel: "Xbox 360" },
				{ y: 1176121, indexLabel: "Nintendo DS"},
				{ y: 1727161, indexLabel: "PSP" },
				{ y: 4303364, indexLabel: "Nintendo 3DS"},
				{ y: 1717786, indexLabel: "PS Vita"}
			]
		}
		]
	});
	chart.render();
}
</script>
<script type="text/javascript" src="https://canvasjs.com/assets/script/canvasjs.min.js"></script>
</head>
<body>
<div id="chartContainer" style="height: 300px; width: 100%;"></div>
</body>
</html>
```



Bar Chart

A bar chart is a chart with rectangular bars with lengths proportional to the values that they represent. A bar Chart is useful for comparing dataPoints in one or more dataSeries.

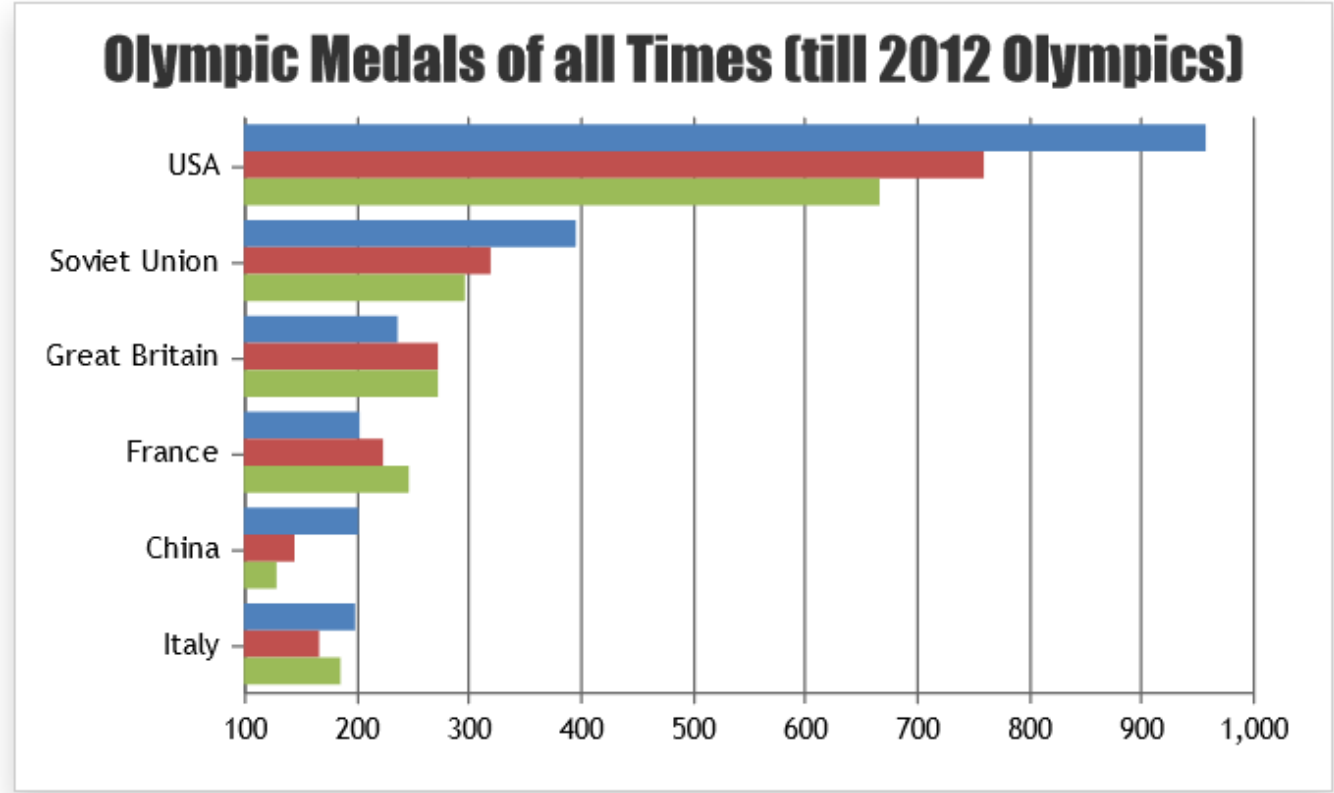
In Bar Chart axisX is Vertical and axisY is Horizontal.

Bar Chart Specific Properties

Applies To	Attribute	Type	Default	Options/Examples
dataSeries	bevelEnabled	Boolean	false	true,false
dataSeries/ dataPoint	indexLabelPlacement	String	“outside”	“inside”,”outside”
dataSeries/ dataPoint	indexLabelOrientation	String	“outside”	“inside”,”outside”
dataSeries	fillOpacity	Number	1	.2,.4,1 etc

```
<!DOCTYPE HTML>
<html>
<head>
  <script type="text/javascript">
  window.onload = function () {
    var chart = new CanvasJS.Chart("chartContainer",
    {
      title:{
        text: "Olympic Medals of all Times (till 2012 Olympics)"
      },
      data: [
      {
        type: "bar",
        dataPoints: [
        { y: 198, label: "Italy"},
        { y: 201, label: "China"},
        { y: 202, label: "France"},
        { y: 236, label: "Great Britain"},
        { y: 395, label: "Soviet Union"},
        { y: 957, label: "USA"}
        ]
      },
      {
        type: "bar",
        dataPoints: [
        { y: 166, label: "Italy"},
        { y: 144, label: "China"},
        { y: 223, label: "France"},
        { y: 272, label: "Great Britain"},
        { y: 319, label: "Soviet Union"},
        { y: 759, label: "USA"}
        ]
      },
      {
        type: "bar",
        dataPoints: [
        { y: 185, label: "Italy"},
        { y: 128, label: "China"},
        { y: 246, label: "France"},
        { y: 272, label: "Great Britain"},
        { y: 296, label: "Soviet Union"},
        { y: 666, label: "USA"}
        ]
      }
      ]
    });

chart.render();
}
```

```
</script>
<script type="text/javascript" src="https://canvasjs.com/assets/script/canvasjs.min.js"></script></head>
<body>
  <div id="chartContainer" style="height: 300px; width: 100%;">
  </div>
</body>
</html>
```

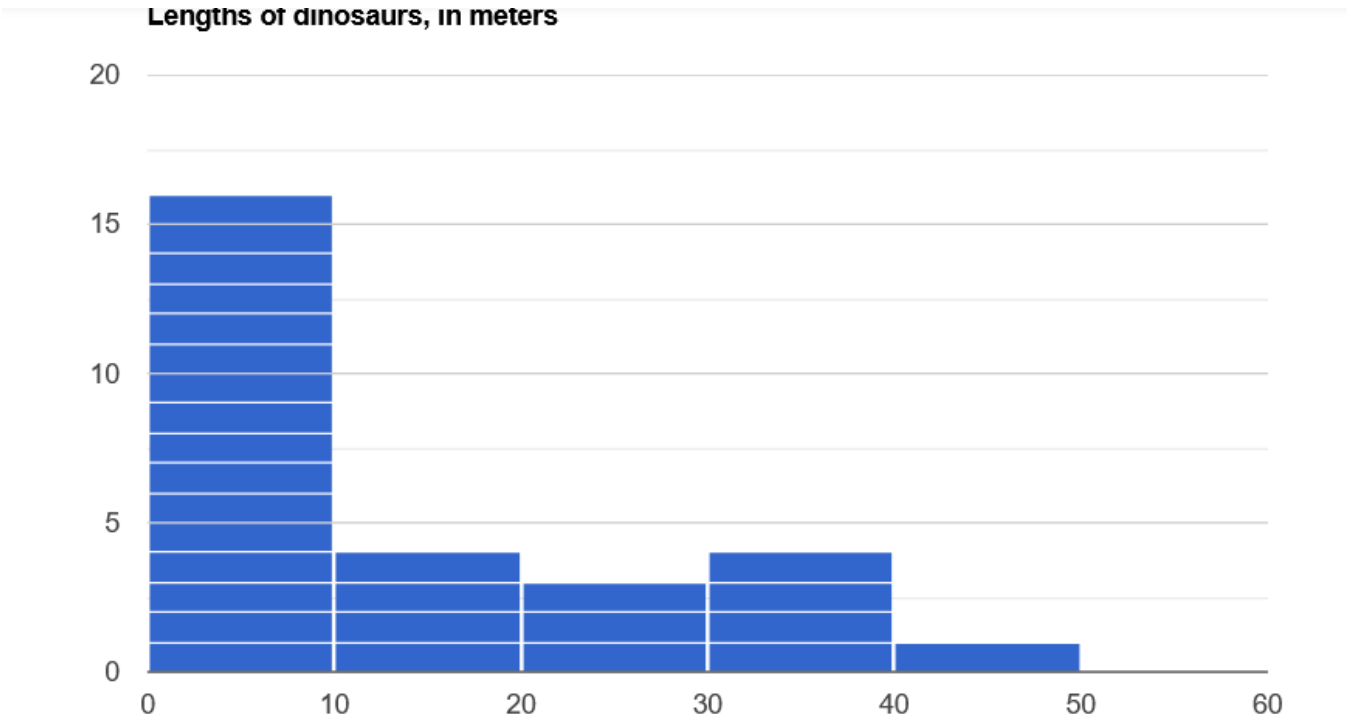


Histogram

A histogram is a chart that groups numeric data into bins, displaying the bins as segmented columns. They're used to depict the distribution of a dataset: how often values fall into ranges.

Google Charts automatically chooses the number of bins for you. All bins are equal width and have a height proportional to the number of data points in the bin.
In other respects, histograms are similar to column charts.

Example
Here's a histogram of dinosaur lengths:



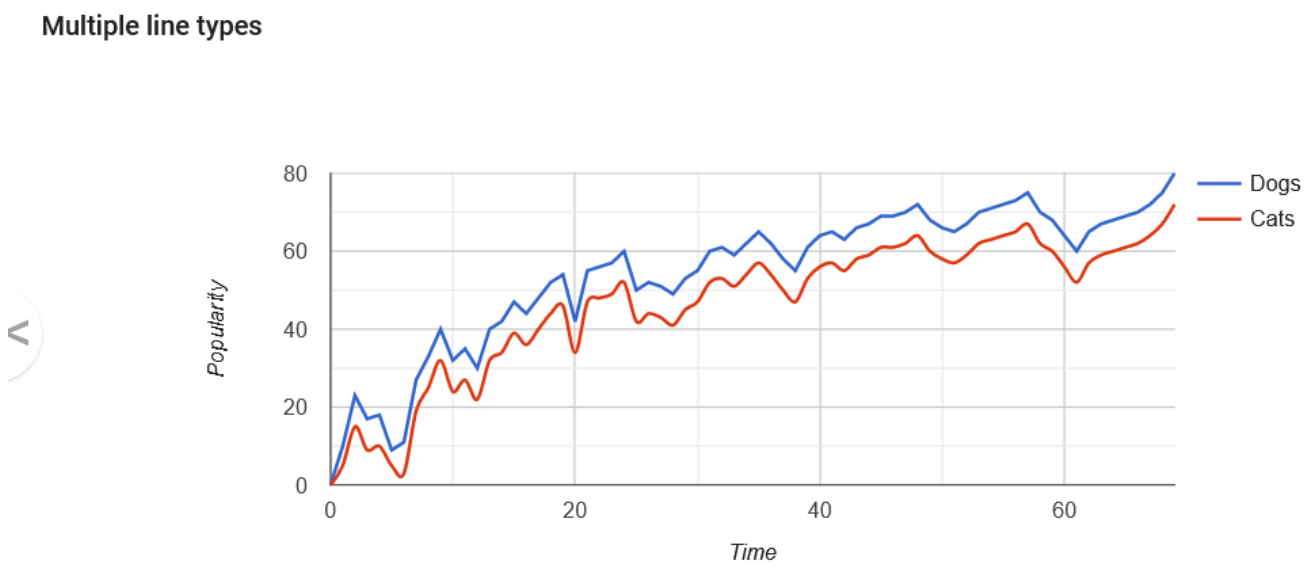
The code to generate this histogram is shown below. After defining the data (here, with google.visualization.arrayToDataTable), the chart is defined with a call to google.visualization.Histogram and drawn with the draw method.

```
<html>
  <head>
    <script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
      google.charts.load("current", {packages:["corechart"]});
      google.charts.setOnLoadCallback(drawChart);
      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Dinosaur', 'Length'],
          ['Acrocanthosaurus (top-spined lizard)', 12.2],
          ['Albertosaurus (Alberta lizard)', 9.1],
          ['Allosaurus (other lizard)', 12.2],
          ['Apatosaurus (deceptive lizard)', 22.9],
          ['Archaeopteryx (ancient wing)', 0.9],
          ['Argentinosaurus (Argentina lizard)', 36.6],
          ['Baryonyx (heavy claws)', 9.1],
          ['Brachiosaurus (arm lizard)', 30.5],
          ['Ceratosaurus (horned lizard)', 6.1],
          ['Coelophysis (hollow form)', 2.7],
          ['Compsognathus (elegant jaw)', 0.9],
          ['Deinonychus (terrible claw)', 2.7],
          ['Diplodocus (double beam)', 27.1],
          ['Dromicelomimus (emu mimic)', 3.4],
```

```
        ['Gallimimus (fowl mimic)', 5.5],
        ['Mamenchisaurus (Mamenchi lizard)', 21.0],
        ['Megalosaurus (big lizard)', 7.9],
        ['Microvenator (small hunter)', 1.2],
        ['Ornithomimus (bird mimic)', 4.6],
        ['Oviraptor (egg robber)', 1.5],
        ['Plateosaurus (flat lizard)', 7.9],
        ['Sauronithoides (narrow-clawed lizard)', 2.0],
        ['Seismosaurus (tremor lizard)', 45.7],
        ['Spinosaurus (spiny lizard)', 12.2],
        ['Supersaurus (super lizard)', 30.5],
        ['Tyrannosaurus (tyrant lizard)', 15.2],
        ['Ultrasaurus (ultra lizard)', 30.5],
        ['Velociraptor (swift robber)', 1.8]]);

      var options = {
        title: 'Lengths of dinosaurs, in meters',
        legend: { position: 'none' },
      };

      var chart = new
google.visualization.Histogram(document.getElementById('chart_div'));
      chart.draw(data, options);
    }
  </script>
 </head>
 <body>
  <div id="chart_div" style="width: 900px; height: 500px;"></div>
 </body>
</html>
```

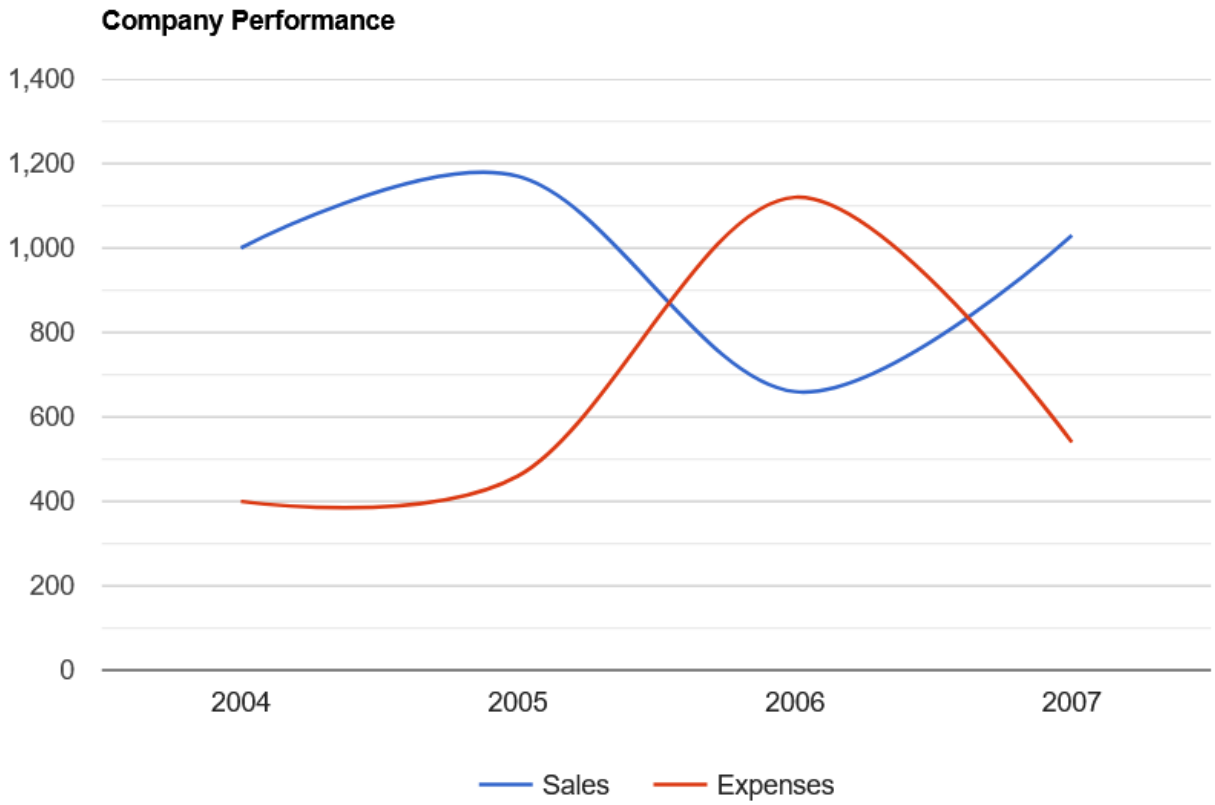
Line Chart:-

A line chart that is rendered within the browser using SVG or VML. Displays tooltips when hovering over points.



Curving the Lines

You can smooth the lines by setting the curveType option to function:



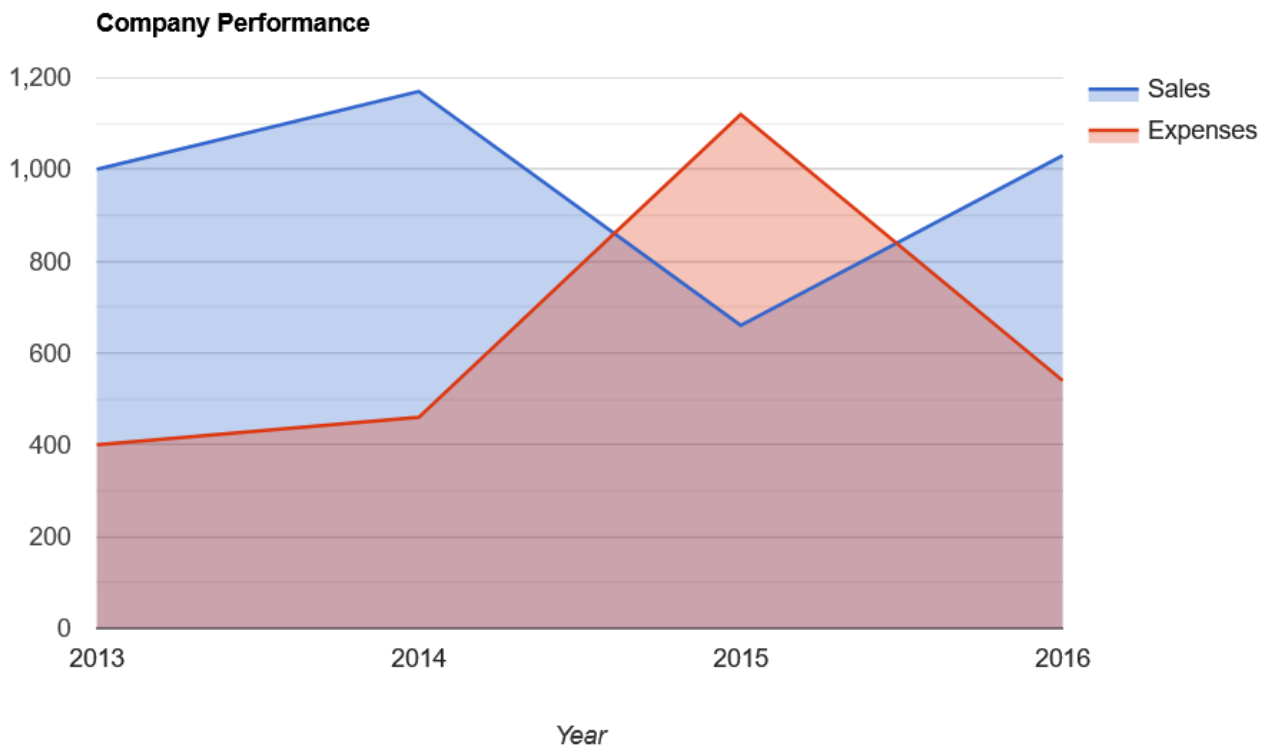

```
  <html>
  <head>
    <script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
      google.charts.load('current', {'packages':['corechart']});
      google.charts.setOnLoadCallback(drawChart);
```

```
      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Year', 'Sales', 'Expenses'],
          ['2004',  1000,      400],
          ['2005',  1170,      460],
          ['2006',  660,       1120],
          ['2007',  1030,      540]
        ]);

        var options = {
          title: 'Company Performance',
          curveType: 'function',
          legend: { position: 'bottom' }
        };

        var chart = new
google.visualization.LineChart(document.getElementById('curve_chart'));

        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <div id="curve_chart" style="width: 900px; height: 500px"></div>
  </body>
</html>
```

Area Chart:-

An area chart that is rendered within the browser using SVG or VML. Displays tips when hovering over points.

Example:



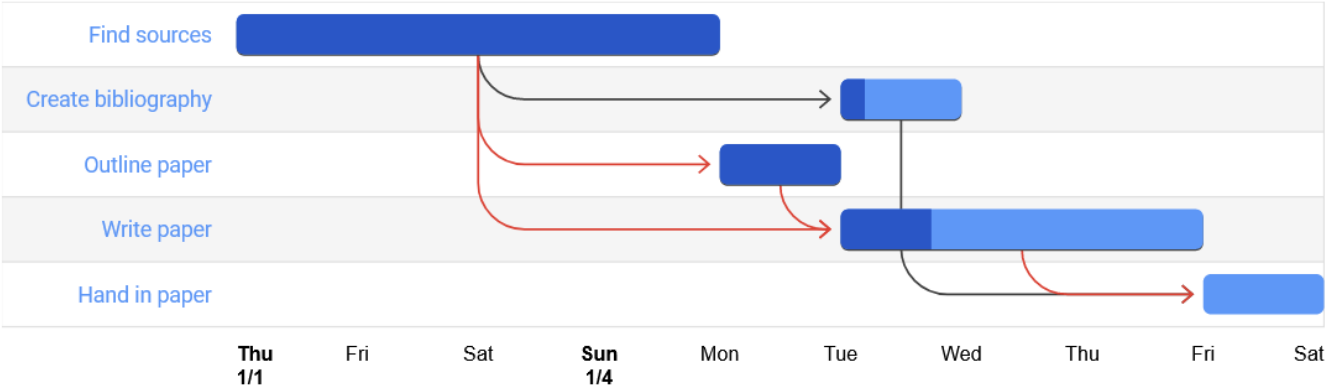

```
<html>
  <head>
    <script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
    <script type="text/javascript">
      google.charts.load('current', {'packages':['corechart']});
      google.charts.setOnLoadCallback(drawChart);

      function drawChart() {
        var data = google.visualization.arrayToDataTable([
          ['Year', 'Sales', 'Expenses'],
          ['2013',  1000,      400],
          ['2014',  1170,      460],
          ['2015',  660,       1120],
          ['2016',  1030,      540]
        ]);

        var options = {
          title: 'Company Performance',
          hAxis: {title: 'Year',  titleTextStyle: {color: '#333'}},
          vAxis: {minValue: 0}
        };

        var chart = new
google.visualization.AreaChart(document.getElementById('chart_div'));
        chart.draw(data, options);
      }
    </script>
  </head>
  <body>
    <div id="chart_div" style="width: 100%; height: 500px;"></div>
```

```
  </body>
</html>
```

Gantt Charts

A Gantt chart is a type of chart that illustrates the breakdown of a project into its component tasks. Google Gantt charts illustrate the start, end, and duration of tasks within a project, as well as any dependencies a task may have. Google Gantt charts are rendered in the browser using SVG. Like all Google charts, Gantt charts display tooltips when the user hovers over the data.

A simple example




```
<html>
<head>
  <script type="text/javascript"
src="https://www.gstatic.com/charts/loader.js"></script>
  <script type="text/javascript">
    google.charts.load('current', {'packages':['gantt']});
    google.charts.setOnLoadCallback(drawChart);

    function daysToMilliseconds(days) {
      return days * 24 * 60 * 60 * 1000;
    }

    function drawChart() {

      var data = new google.visualization.DataTable();
      data.addColumn('string', 'Task ID');
      data.addColumn('string', 'Task Name');
      data.addColumn('date', 'Start Date');
      data.addColumn('date', 'End Date');
      data.addColumn('number', 'Duration');
      data.addColumn('number', 'Percent Complete');
      data.addColumn('string', 'Dependencies');

      data.addRows([
        ['Research', 'Find sources',
         new Date(2015, 0, 1), new Date(2015, 0, 5), null,  100,  null],
        ['Write', 'Write paper',
         null, new Date(2015, 0, 9), daysToMilliseconds(3), 25,
'Research,Outline'],
        ['Cite', 'Create bibliography',
         null, new Date(2015, 0, 7), daysToMilliseconds(1), 20, 'Research'],
```

```
        ['Complete', 'Hand in paper',
         null, new Date(2015, 0, 10), daysToMilliseconds(1), 0, 'Cite,Write'],
        ['Outline', 'Outline paper',
         null, new Date(2015, 0, 6), daysToMilliseconds(1), 100, 'Research']
      ]);

      var options = {
        height: 275
      };

      var chart = new
google.visualization.Gantt(document.getElementById('chart_div'));

      chart.draw(data, options);
    }
  </script>
</head>
<body>
  <div id="chart_div"></div>
</body>
</html>
```

Basic Column Chart

Consider the following tabular data to be rendered in the form of column chart.

Fruit	Sales in first Quarter (Thousand dollars)
Banana	18
Orange	29
Apple	40
Mango	34
Grape	24

Below is how a minimal basic Column Chart would look like. Here are important things to remember

1. Instantiate a new Chart object by sending the ID of div element where the chart is to be rendered. You can also pass DOM element instead of ID
2. Pass all the Chart related "options" to the constructor as the second parameter.
3. Call chart.render() method to render the chart

Chart "options" mainly contains 4 important items.

1. **title** object with its text property set.
2. **dataPoints** – which is an array of all data items to be rendered
3. **dataSeries** – parent of dataPoints that also defines type of chart and other series wide options
4. **data** – array element which is collection of one or more dataSeries objects. Here we have only one dataSeries.

Example

```
<!DOCTYPE HTML>
<html>
<head>
  <script type="text/javascript">
  window.onload = function () {
    var chart = new CanvasJS.Chart("chartContainer", {

      title:{
        text: "Fruits sold in First Quarter"
      },
      data: [//array of dataSeries
        { //dataSeries object

         /*** Change type "column" to "bar", "area", "line" or "pie"***/
         type: "column",
         dataPoints: [
         { label: "banana", y: 18 },
         { label: "orange", y: 29 },
         { label: "apple", y: 40 },
         { label: "mango", y: 34 },
         { label: "grape", y: 24 }
         ]
       }
       ]
    });

    chart.render();
  }
  </script>
  <script type="text/javascript"
src="https://canvasjs.com/assets/script/canvasjs.min.js"></script>
</head>
<body>
  <div id="chartContainer" style="height: 300px; width: 100%;">
  </div>
</body>
</html>
```

Task: Develop an HTML document to illustrate each chart with real-time examples.

```
<!DOCTYPE HTML>
<html>
<head>
<script src="https://canvasjs.com/assets/script/canvasjs.min.js"></script>
<script type="text/javascript">

window.onload = function () {
```

```
        var chart = new CanvasJS.Chart("chartContainer", {
                title:{
                        text: "My First Chart in CanvasJS"
                },
                data: [
                {
                        // Change type to "doughnut", "line", "splineArea", etc.
                        type: "column",
                        dataPoints: [
                                { label: "apple",  y: 10  },
                                { label: "orange", y: 15  },
                                { label: "banana", y: 25  },
                                { label: "mango",  y: 30  },
                                { label: "grape",  y: 28  }
                        ]
                }
                ]
        });
        chart.render();
}
</script>
</head>
<body>
<div id="chartContainer" style="height: 300px; width: 100%;"></div>
</body>
</html>
```

Module - 12:

Open Source CMS (Content Management System): What is a CMS?, Install CMS, Themes, Plugins.

A content management system or CMS is a software that is used to build websites and create content to be published on the internet. Typically, CMS allows you to create a website without writing any code.

In the early days of the web, you needed to know HTML to be able to code a website and publish your content online.

Think of it like driving a car. You don't need to understand the mechanics behind how it all works. Instead, you use a simplified dashboard and pedals to tell the car what to do.

With a powerful CMS platform like WordPress, you can log in to your website dashboard and use a simplified interface to create your web pages, add content, and customize the design.

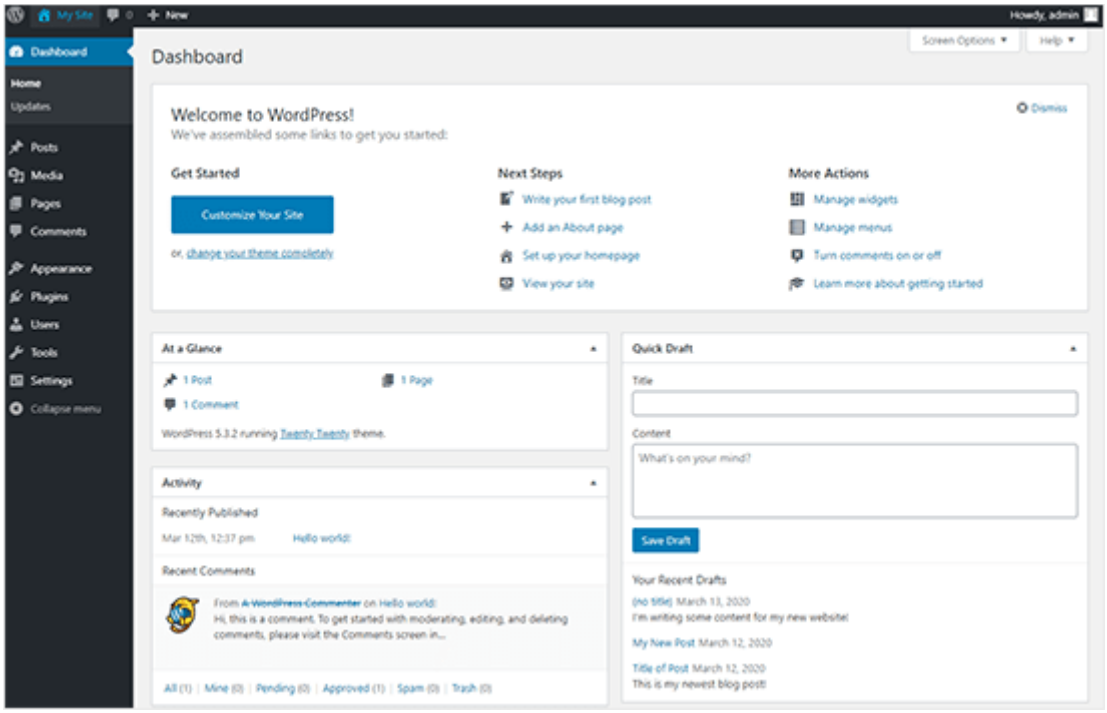
A good content management system allows you to use a simple user interface to create websites.

As the website owner, you can log in to the website dashboard to customize your website. The dashboard can also be called the backend or admin area of a website.

WordPress website dashboard

A good content management system allows you to use a simple user interface to create websites.

As the website owner, you can log in to the website dashboard to customize your website. The dashboard can also be called the backend or admin area of a website.



Themes to Customize the Design

CMS platforms usually let you change the look of your website by selecting a template or theme.

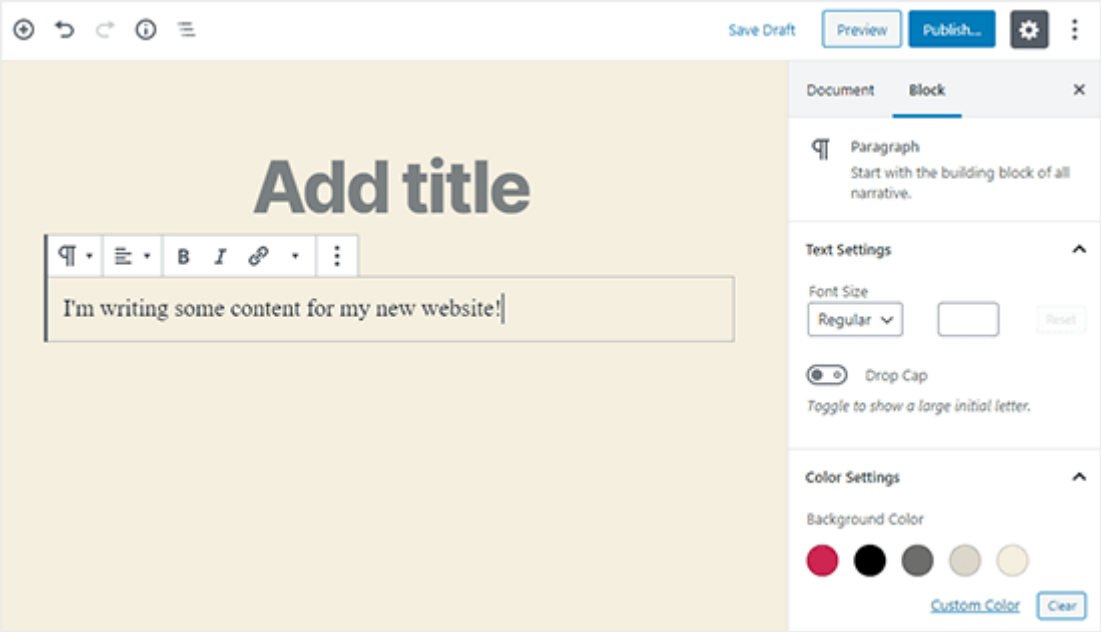
Themes are pre-designed templates that let you change the layout, colors, fonts, and other design aspects of a website.

WordPress has thousands of free and premium themes available. We have created our expert pick of the best WordPress themes.

Content Editor

CMS platforms include a content editor that you can use to create and publish your web pages.

Text editors are sometimes called WYSIWYG editors, which stands for What You See Is What You Get. As you edit the text, you can see exactly what it will look like on your website, instead of looking at code. Think of it like using an editor like Microsoft Word.



the WordPress content editor allows you to publish web content without coding

CMSes also allow you to upload images and other media files to your website.

Plugins and Extensions

Many CMS platforms let you add new features and modify your site in other ways by adding plugins.

Plugins on a website are like apps on a smartphone. You can easily install them to add new features and functions to your website. You can use them to add new features like a contact form or social media integration.

WordPress comes with over 57,000 free plugins. We have created a list of the must have WordPress plugins.

User Management

With a good CMS, you can easily give other people access to publish content on your website.

You can even assign them different user roles to control what they have access to.
Benefits of Using a CMS

CMS software make it easier for beginners to create websites. They open up the internet for non-techy users by allowing them to design their own websites, publish content on the internet, and build online businesses without hiring developers.

Here are just some of the benefits of using a CMS software to build your website.

No hassle publishing: A CMS allows even those without technical skills to publish content, make websites, and edit content using a simple dashboard.

Convenient content scheduling: Content management systems allow site administrators to publish content with a click of a button. You can schedule posts to be published to meet calendar deadlines, business events, or product launches.

Works with any size business: Whether you're making a website for a big name brand, or just starting a business, you can use a CMS. With the right CMS and web hosting package, you can create any type or size of website.

Affordable and easy to control: A CMS allows beginners to manage sites of all sizes without relying on high-priced web developers to perform site maintenance or make routine changes. A content management system allows you to choose who has access to your site.

Ability to Customize: It's easy to change your website design or customize it with your own logo, colors, and styles. You can also use plugins to add more features.

WordPress is the most popular content management system in the world. It powers more than 38% of all websites on the internet.

Task: Develop an E-learning website using any CMS (for example WordPress)