UI FULL STACK WEBDEVELOPMENT NOTES

INTRODUCTION:

A full stack web developer is a person who can develop both **client** and **server** software.

In addition to mastering HTML and CSS, he/she also knows how to:

- Program a **browser** (like using JavaScript, jQuery, Angular, or Vue)
- Program a **server** (like using PHP, ASP, Python, or Node)
- Program a **database** (like using SQL, SQLite, or MongoDB)

Front End:

HTML4&5
CSS2&3
JAVA SCRIPTS
ADVANCED JAVA SCRIPTS
JQUERY
ECMA 6
BOOTSTRAP 5

BACK END:

NODE JS
EXPRESS JS
SOCKET IO

DATA BASE:

MANGO DB

Frontend:

It only defines UI which a user or a client will unable to access the data in the database.

HTML (Hyper Text Markup Language):

HTML is the language used to create the websites you visit every day, and it provides a logical way to structure content for websites. We can also define HTML as the language that helps in creating the backbone of any website.
Below mentioned are the basic HTML tags which divides the whole document into various parts like head, body etc.

- Every HTML document begins with a HTML document tag. Although this is not mandatory but it is a good convention to start the document with this below mentioned tag:
  <!DOCTYPE html>

- **<html>:** Every HTML code must be enclosed between basic HTML tags. It begins with **<html>** and ends with **</html>** tag.
- **<head>:** The head tag comes next which contains all the header information of the web page or document like the title of the page and other miscellaneous information. These information's are enclosed within head tag which opens with **<head>** and ends with **</head>**. The contents will of this tag will be explained in the later sections of course.
- **<title>:** We can mention the title of a web page using the **<title>** tag. This is a header information and hence mentioned within the header tags. The tag begins with **<title>** and ends with **</title>**
- **<body>:** Next step is the most important of all the tags we have learned so far. The body tag contains the actual body of the page which will be visible to all the users. This opens with **<body>** and ends with **</body>**. Every content enclosed within this tag will be shown on the web page be it writings or images or audios or videos or even links. We will see later in the section how using various tags we may insert mentioned contents into our web pages.

CSS (Cascading Style Sheets):

CSS is used to stylize the HTML contents present on a website. This includes modifying the page colour, font-family, font-size, element positioning and more.
There are three types of CSS:

1. In a separate file (external)
2. At the top of a web page document (internal)
3. Right next to the text it decorates (inline)

External Style Sheets:

Separate files having CSS instructions with the file extension **(.css)**. The main advantage of using external style sheet is that you can change the whole website's style at once, without rewriting or modifying the style tag every page. Thus, saving a lot of time and energy. However, the external style sheet must be linked into the HTML file by using the tag between for making it work.

Internal Styles:

Placed at the top of each web page document before any of the content is listed. The internal style CSS codes are written between the head tags in the of the HTML file itself. Internal styles are very easy to find and they are given the second highest priority, next to the external style sheets.

Inline Styles:

Placed right where you need them, next to the text or graphics you wish to decorate. The inline styles can be inserted in the middle of the HTML code. This gives the real freedom to specify each web page element, however, can make the maintenance work of the website difficult.

JavaScript:

JavaScript and Java are completely different languages, both in concept and design, and both have no correlation with each other. Java is an Object-Oriented Programming (OOP) language created by James Gosling of Sun Microsystems. JavaScript is a scripting language and was originally known as Live Script. JavaScript is used in front-end development whereas Java is used for back-end development in web development.

ECMA 6:

ES6 stands for ECMAScript 6. ECMAScript was created to standardize JavaScript, and ES6 is the 6th version of ECMAScript, it was published in 2015, and is also known as ECMAScript 2015.

Bootstrap:

Bootstrap is the most popular CSS Framework for developing responsive and mobile-first Websites.

jQuery:

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API (Application Programming Interface) that works across a multitude of browsers.

Back End:

Back-end Development refers to the server side of development where you are primarily focused on how the site works. ... This type of web development usually consists of three parts: a server, an application, and a database. Code written by back-end developers is what communicates the database information to the browser.

Node.JS:

Node. Js is a JavaScript run-time environment built on Chrome's V8 JavaScript engine. It comes with an http module that provides a set of functions and classes for building a HTTP server. For this basic HTTP server, we will also be using file system, path and URL, all of which are native Node.

Express.JS:

Express. Js is a free and open-source web application framework for Node. Js. It is used for designing and building web applications quickly and easily.4 | P a g e Data Base (DB): A database is an organized collection of structured information, or data, typically stored electronically in a computer system. A database is usually controlled by a database management system (DBMS). ... The data can then be easily accessed, managed, modified, updated, controlled, and organized.

Mongo DB:

Mongo DB is a document database with the scalability and flexibility that you want with the querying and indexing that you need

DATE:14/6/2021

HIPERTEXT MARKUP LANGUAGE:

A Predefined language using which we could be able to add content and sources to the field. It comes with the predefined set of html tags used to add different types of content accordingly.

Some of html tags:

html -> Tag used to hold the complete content of page
body -> To hold the actual content of page
title -> Holds the page title
p -> Paragraph tag used to hold multi line text content
br -> Break tag to add a single line break
div -> Div tag to hold block content
span -> Inline element to hold content in same line
table -> To render content in row and column way
ul/ol -> To render content in ordered or unordered way
li -> List
h1 to h6 -> Heading tags to render content in the form of heading
Etc.

HTML ATTRIBUTES:

We can add extra information to the html elements through attributes. The attributes can be both predefined or user defined.

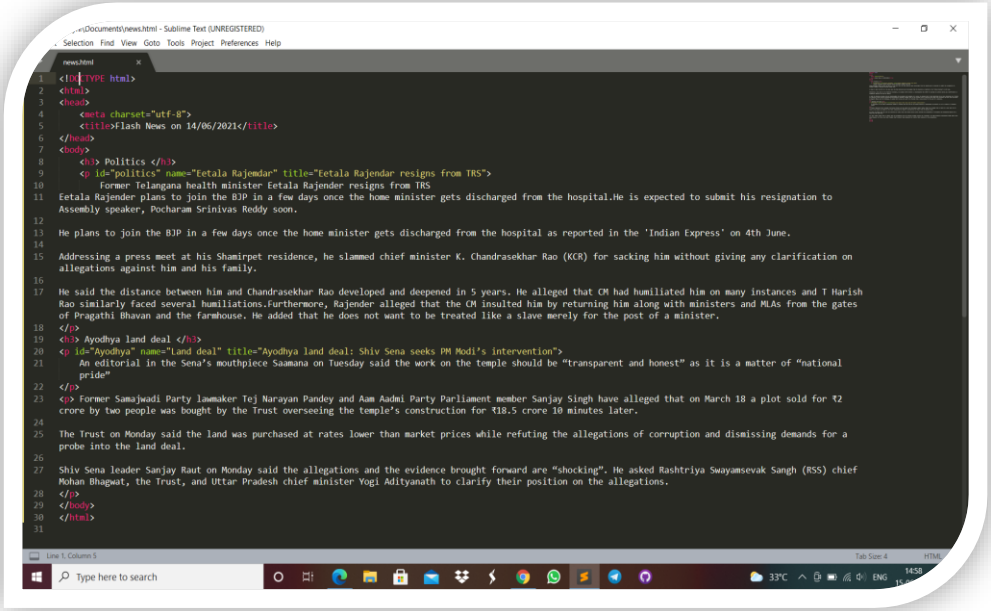
Following are the some of predefined attributes that can be added to the elements:

id -> Using which we can add unique reference to elements
name -> Name value can be added to elements
class -> Using which we can add single/multiple CSS classes to elements
style -> To add single or multiline CSS properties to elements
alt -> Using which we can alternate text content to img tag

title -> using which we can add title to any html elements.

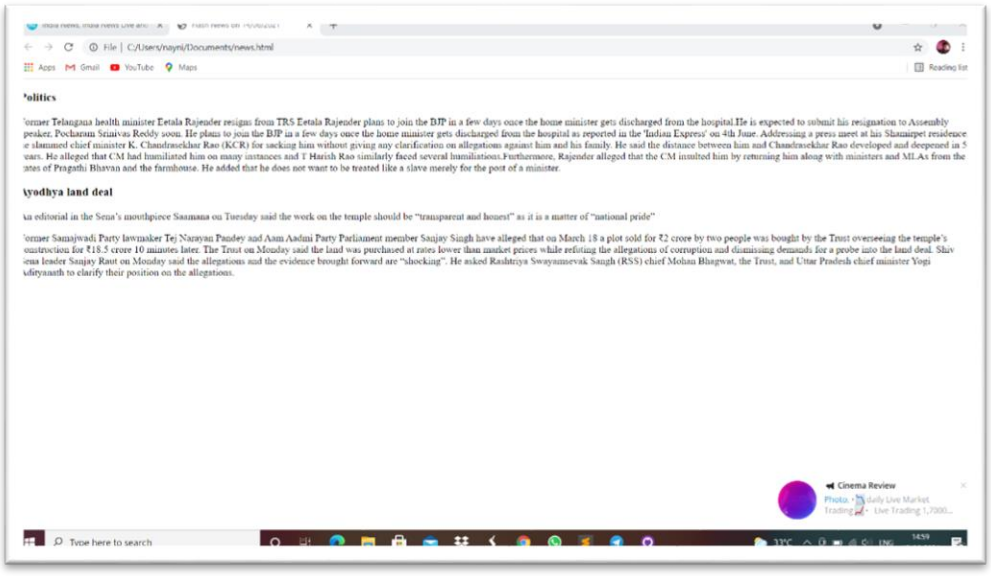
Etc.

Sample Webpage Program Using HTML attributes:



```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Flash News on 14/06/2021</title>
</head>
<body>
    <h1> Politics </h1>
    <p id="politics" name="Eetala Rajemdar" title="Eetala Rajendar resigns from TRS">
        Former Telangana health minister Eetala Rajender resigns from TRS
Eetala Rajender plans to join the BJP in a few days once the home minister gets discharged from the hospital.He is expected to submit his resignation to
Assembly speaker, Pocharam Srinivas Reddy soon.

He plans to join the BJP in a few days once the home minister gets discharged from the hospital as reported in the 'Indian Express' on 4th June.

Addressing a press meet at his Shamirpet residence, he slammed chief minister K. Chandrasekhar Rao (KCR) for sacking him without giving any clarification on
allegations against him and his family.

He said the distance between him and Chandrasekhar Rao developed and deepened in 5 years. He alleged that CM had humiliated him on many instances and T Harish
Rao similarly faced several humiliations.Furthermore, Rajender alleged that the CM insulted him by returning him along with ministers and MLAs from the gates
of Pragathi Bhavan and the farmhouse. He added that he does not want to be treated like a slave merely for the post of a minister.
    </p>
    <h3> Ayodhya land deal </h3>
    <p id="Ayodhya" name="Land deal" title="Ayodhya land deal: Shiv Sena seeks PM Modi's intervention">
        An editorial in the Sena's mouthpiece Saamana on Tuesday said the work on the temple should be "transparent and honest" as it is a matter of "national
        pride"
    </p>
    <p> Former Samajwadi Party lawmaker Tej Narayan Pandey and Aam Aadmi Party Parliament member Sanjay Singh have alleged that on March 18 a plot sold for ₹2
crore by two people was bought by the Trust overseeing the temple's construction for ₹18.5 crore 10 minutes later.

The Trust on Monday said the land was purchased at rates lower than market prices while refuting the allegations of corruption and dismissing demands for a
probe into the land deal.

Shiv Sena leader Sanjay Raut on Monday said the allegations and the evidence brought forward are "shocking". He asked Rashtriya Swayamsevak Sangh (RSS) chief
Mohan Bhagwat, the Trust, and Uttar Pradesh chief minister Yogi Adityanath to clarify their position on the allegations.
    </p>
</body>
</html>
```

OUTPUT:



DATE:16/6/2021

Today class is taken about GITHUB account:

Steps to be followed to install this GitHub account

/*

Install Git on Windows

Navigate to the latest Git for Windows installer and download the latest version.

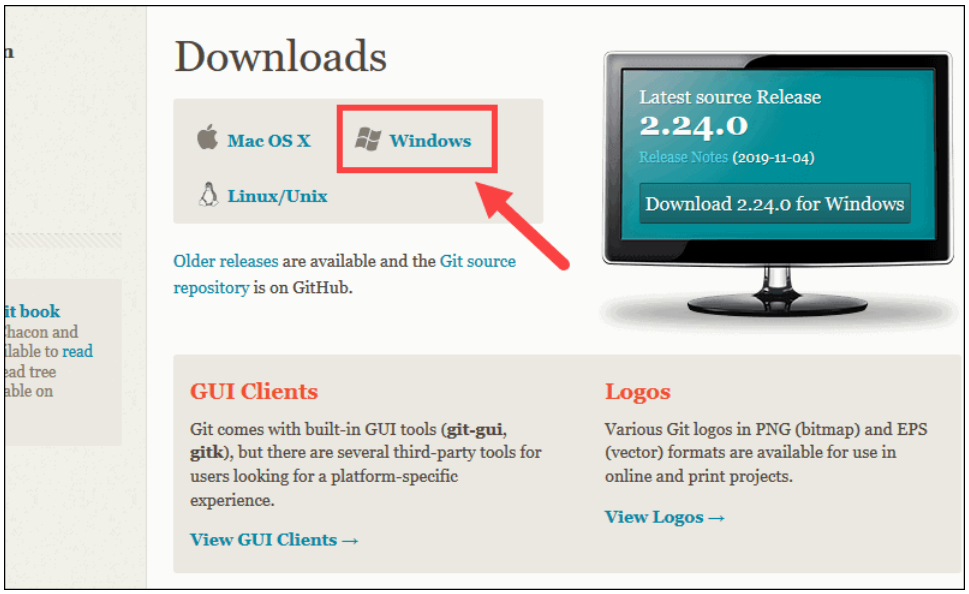
Once the installer has started, follow the instructions as provided in the **Git Setup** wizard screen until the installation is complete. Open the windows command prompt (or **Git Bash** if you selected not to use the standard Git Windows Command Prompt during the Git installation).

Type git version to verify Git was installed.

*/

1.Browse to the official Git website: https://git-scm.com/downloads

2. Click the download link for Windows and allow the download to complete.



3. Browse to the download location (or use the download shortcut in your browser). Double-click the file to extract and launch the installer.

4. If you want you can allow the app to make changes to your device by clicking **Yes** on the User Account Control dialog that opens.

5. The installer will ask you for an installation location. Leave the default, unless you have reason to change it, and click **Next**.

6. A component selection screen will appear. Leave the defaults unless you have a specific need to change them and click **Next**.

7. The installer will offer to create a start menu folder. Simply click **Next**.

8. Select a text editor you'd like to use with Git. Use the drop-down menu to select Notepad++ (or whichever text editor you prefer) and click **Next**.

After all the necessary steps taken (After installation process) Sign up to GitHub account using your Email-id and password (necessary details asked at the time of signup).

After that process has been completed open the GitHub Desktop on your pc such that you can clone the codes to your pc by using the other person repositories.

Steps followed to clone the codes from other repositories (Durgaprasad sir repository):

1.Open the GitHub Desktop on your pc.

2.Sign in to GitHub account (use your registered email-id and password)

3.Click on file on top left corner and click on the clone repository option available there.

4.After opening that option u can find the name of repository from which you want to clone the data available. Click on that.

5.Choose the path where you want to clone the data (Better to choose windows D or E).

6.Click on the clone option.

PUSH AND PULL THE CODES:

❖ **What Does git push Do?**

git push updates the remote branch with local commits. It is one of the four commands in Git that prompts interaction with the remote repository. You can also think of git push as *update* or *publish*.

By default, git push only updates the corresponding branch on the remote. So, if you are checked out to the master branch when you execute git push, then only the master branch will be updated. It's always a good idea to use git status to see what branch you are on before pushing to the remote.

❖ **Common usages and options for git push**

- git push -f: Force a push that would otherwise be blocked, usually because it will delete or overwrite existing commits *(Use with caution!)*
- git push -u origin [branch]: Useful when pushing a new branch, this creates an upstream tracking branch with a lasting relationship to your local branch
- git push --all: Push all branches
- git push --tags: Publish tags that aren't yet in the remote repository

❖ **What Does git pull Do?**

git pull

git pull is one of the 4 remote operations within Git. Without running git pull, your local repository will never be updated with changes from the remote. git pull should be used every day you interact with a repository with a remote, at the minimum. That's why git pull is one of the most used Git commands.

▪ **git pull and git fetch**

git pull, a combination of git fetch + git merge, updates some parts of your local repository with changes from the remote repository. To understand what is and isn't affected by git pull, you need to first understand the concept of remote tracking branches. When you clone a repository, you clone one working branch, master, and all of the remote tracking branches. git fetch updates the remote tracking branches. git merge will update your current branch with any new commits on the remote tracking branch.

git pull is the most common way to update your repository.

➢ **Common usages and options for git pull**

- git pull: Update your local working branch with commits from the remote, *and* update all remote tracking branches.
- git pull --rebase: Update your local working branch with commits from the remote, but rewrite history so any local commits occur after all new commits coming from the remote, avoiding a merge commit.
- git pull --force: This option allows you to force a fetch of a specific remote tracking branch when using the <refspec> option that would otherwise not be fetched due to conflicts. To force Git to overwrite your current branch to match the remote tracking branch, read below about using git reset.
- git pull --all: Fetch *all* remotes - this is handy if you are working on a fork or in another use case with multiple remotes.

REFERENCE: https://github.com/git-guides/git-clone

Some commands which relate to repository structure:

- ❖ git add // transfers your project from working directory// to staging area.
- ❖ git commit // transfers your project from staging area to // Local Repository.
- ❖ git push// transfers project from local to central repository.// (requires internet)

/*For Easy Reference:

1. Find a project you want to contribute to
2. Fork it
3. Clone it to your local system
4. Make a new branch
5. Make your changes
6. Push it back to your repo
7. Click the **Compare & pull request** button
8. Click **Create pull request** to open a new pull request*/

Date:18/06/2021

CSS:

Cascading Style Sheets (CSS) is used to format the layout of a webpage. With CSS, you can control the color, font, the size of text, the spacing between elements, how elements are positioned and laid out, what background images or background colours are to be used, different displays for different devices and screen sizes

CSS Syntax:

A CSS comprises style rules that are interpreted by the browser and then applied to the corresponding elements in your

document.
A style rule set consists of a selector and declaration block.

Selector -- h1Declaration -- {color:pink;font size:12px;}

/*Not Spoken Topic

CSS SELECTORS:

CSS selectors are used to "find" (or select) HTML elements based on their element name, id, class, attribute**.**

There are six types of selectors in CSS:

1.THE UNIVERSAL SELECTORS: Rather than selecting elements of a specific type, the universal selector quite simply matches the name of any element type.

Example:

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

OUTPUT:

Hello World!

These paragraphs are styled with CSS.

2.THE ELEMENT SELECTOR: The element selector selects elements based on the element name. You can select all p elements on a page like this.

Example:

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

OUTPUT:

Every paragraph will be affected by the style.

Me too!

And me!

3. THE DESCENDANT SELECTOR: Suppose you want to apply a style rule to a particular element only when it lies inside a particular element. As given in the following example, the style rule will apply to the em element only when it lies inside the ul tag.

Example:

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

OUTPUT:

Hello World!

These paragraphs are styled with CSS.

4. THE ID SELECTOR: The id selector uses the id attribute of an HTML element to select a specific element. The id of an element should be unique within a page, so the id selector is used to select one unique element!To select an element with a specific id, write a hash (#) character, followed by the id of the element.The style rule below will be applied to the HTML element with id=" para1":

5. THE CLASS SELECTORS: The class selector selects elements with a specific class attribute.To select elements with a specific class, write a period (.) character, followed by the name of the class.

Example:

*** This paragraph refers to two classes. ***

OUTPUT:

This heading will not be affected

This paragraph will be red and center-aligned.

This paragraph will be red, center-aligned, and in a large font-size.

6. GROUPING SELECTORS: If you have elements with the same style definitions,

***h1** {
text-align: center;
color: blue;
}
h2 {
text-align: center;
color: blue;
}
p {
text-align: center;

color: blue;
} *
it will be better to group the selectors, to minimize the code. To group selectors, separate each selector with a comma.

```
*h1, h2, p {
text-align: center;
color: red;
}*
```

OUTPUT:

Not Spoken Topic*/

Applications of CSS:

- **CSS saves time** - You can write CSS once and then reuse same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many Web pages as you want.
- **Pages load faster** - If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So less code means faster download times.
- **Easy maintenance** - To make a global change, simply change the style, and all elements in all the web pages will be updated automatically.
- **Superior styles to HTML** - CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
- **Multiple Device Compatibility** - Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cell phones or for printing.
- **Global web standards** - Now HTML attributes are being deprecated and it is being recommended to use CSS. So its a good idea to start using CSS in all the HTML pages to make them compatible to future browsers.

Styles can be added in three following ways:
Inline, Internal, and External.

1.Inline CSS:

For inline styles, we use the style attribute of the HTML tags. CSS is passed as a string to the style attribute which adds the styles to the tags.

For example: If we want to make our web page background pink with inline CSS, we can write something like this:

<body style=" background-color: pink"></body>

2.Internal CSS

For Internal CSS, a style tag is used in which all the styles related to the webpage are added. This style tag is added in the head tag of the page so that the styles are added even before the HTML document is rendered.

Example:

```
<head>
  <style>
        body {
            background-color: pink;
      }
</style>
</head>
```

3.External CSS

In external CSS we use a separate file with a .css extension where we write all our styles. This CSS file can be used by multiple webpages by using a link tag which is added under the head tag.

Example:

1.First, we will add all our styles to our CSS file, for our example, it will be styles.css

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

2.Then we will link this CSS file to our html file as follows:

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

★ **CSS Colors, Fonts and Sizes:**

The CSS color property defines the text color to be used.

The CSS font-family property defines the font to be used.

The CSS font-size property defines the text size to be used.

EXAMPLE PROGRAM FOR CSS:

```
<html>
    <head>
        <title>
            News Page
        </title>
        <style type="text/css">
            .newsTitle {
                background: red;
                color: green;
                width: 300px;
            }
            .news_content {
                background: pink;
                font-size: 20px;
            }
        </style>
    </HEad>
    <BODY>
        <!--
            This page is to demonstrate css classes and tags...


        -->

        <h3 class="newsTitle">New Item 2:</h3>
            <p id="specialItem2" class="news_content" storytype="education"  dataContnet="newsscontentlist">
                Shops, malls and restaurants in
                Delhi
                will open from today as Covid numbers in the national capital drop to a three-month low. Shops will be open seven days a week instead of the
                current odd-even system. Chief Minister Arvind Kejriwal, however, said this will be on trial basis for a week and strict action will be taken
                if the Covid numbers rise.Shop timings will remain the same, from 10 am to 8 pm, Mr Kejriwal said.  Restaurants -- which were open only for
                takeaways and home deliveries -- can now have diners but with only 50 per cent of seating capacity.
            </p>

        <h3 class="newsTitle">New Item 3:</h3>
            <p style="background: pink;font-size:20px;">
                Shops, malls and restaurants in Delhi will open from today as Covid numbers in the national capital drop to a three-month low. Shops will be
                open seven days a week instead of the current odd-even system. Chief Minister Arvind Kejriwal, however, said this will be on trial basis for a
                week and strict action will be taken if the Covid numbers rise.Shop timings will remain the same, from 10 am to 8 pm, Mr Kejriwal said.
                Restaurants -- which were open only for takeaways and home deliveries -- can now have diners but with only 50 per cent of seating capacity.
```

```
            </p>
        <h3 class="newsTitle">New Item 4:</h3>
            <p style="background: pink;font-size:20px;">
                Shops, malls and restaurants in Delhi will open from today as Covid numbers in the national capital drop to a three-month low. Shops will be
                open seven days a week instead of the current odd-even system. Chief Minister Arvind Kejriwal, however, said this will be on trial basis for a
                week and strict action will be taken if the Covid numbers rise.Shop timings will remain the same, from 10 am to 8 pm, Mr Kejriwal said.
                Restaurants -- which were open only for takeaways and home deliveries -- can now have diners but with only 50 per cent of seating capacity.
            </p>
    </body>
</html>
```

OUTPUT:

News Item 2:

Shops, malls and restaurants in Delhi will open from today as Covid numbers in the national capital drop to a three-month low. Shops will be open seven days a week instead of the current odd-even system. Chief Minister Arvind Kejriwal, however, said this will be on trial basis for a week and strict action will be taken if the Covid numbers rise.Shop timings will remain the same, from 10 am to 8 pm, Mr Kejriwal said. Restaurants -- which were open only for takeaways and home deliveries -- can now have diners but with only 50 per cent of seating capacity.

News Item 3:

Shops, malls and restaurants in Delhi will open from today as Covid numbers in the national capital drop to a three-month low. Shops will be open seven days a week instead of the current odd-even system. Chief Minister Arvind Kejriwal, however, said this will be on trial basis for a week and strict action will be taken if the Covid numbers rise.Shop timings will remain the same, from 10 am to 8 pm, Mr Kejriwal said. Restaurants -- which were open only for takeaways and home deliveries -- can now have diners but with only 50 per cent of seating capacity.

News Item 4:

Shops, malls and restaurants in Delhi will open from today as Covid numbers in the national capital drop to a three-month low. Shops will be open seven days a week instead of the current odd-even system. Chief Minister Arvind Kejriwal, however, said this will be on trial basis for a week and strict action will be taken if the Covid numbers rise.Shop timings will remain the same, from 10 am to 8 pm, Mr Kejriwal said. Restaurants -- which were open only for takeaways and home deliveries -- can now have diners but with only 50 per cent of seating capacity.

Date: 21/06/2021

CSS CLASSES:

A concept of grouping all the required CSS properties as an individual block (under the {}) assigning a user defined name to it is called a CSS classes.

1.Every CSS class name should always start with. Operator

2.in a single page we can define any number of CSS classes

3. style is a predefined tag capable of holding any number of CSS classes.

4. class is a predefined attribute using which we could be able to inject one or multiple CSS classes to a single html element.

5. once a CSS class been defined it can be used any number of times to any number of elements.

syntax:

```
<style>
    .classname {
    ....
    ....
    .... /Set of CSS property classes
}
</style>
```

Topic: COLORS IN HTML: -

HTML color codes are hexadecimal triplets representing the colours red, green, and blue (#RRGGBB).

For example, in the color red, the color code is #FF0000, which is '255' red, '0' green, and '0' blue. These color codes can change the color of the background, text, and tables on a web page.

Here are some of the color codes mentioned for reference:

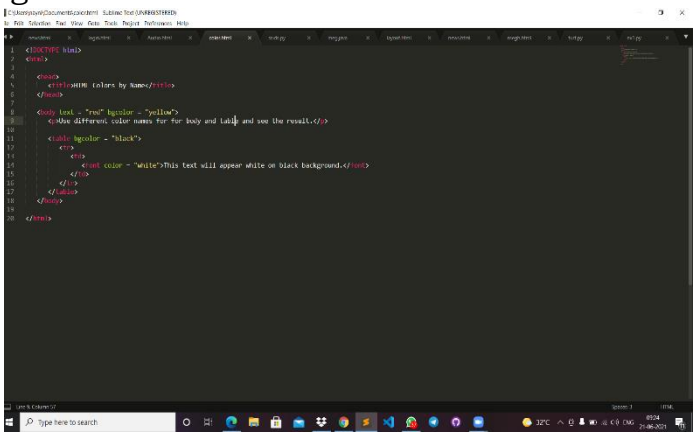
Color Name	Color Code	Color Name	Color Code
Red	#FF0000	White	#FFFFFF
Cyan	#00FFFF	Silver	#C0C0C0
Blue	#0000FF	Gray or Grey	#808080
DarkBlue	#00008B	Black	#000000
LightBlue	#ADD8E6	Orange	#FFA500
Purple	#800080	Brown	#A52A2A
Yellow	#FFFF00	Maroon	#800000
Lime	#00FF00	Green	#008000
Magenta	#FF00FF	Olive	#808000
Pink	#FFC0CB	Aquamarine	#7FFD4

The <body> tag has following attributes which can be used to set different colours −

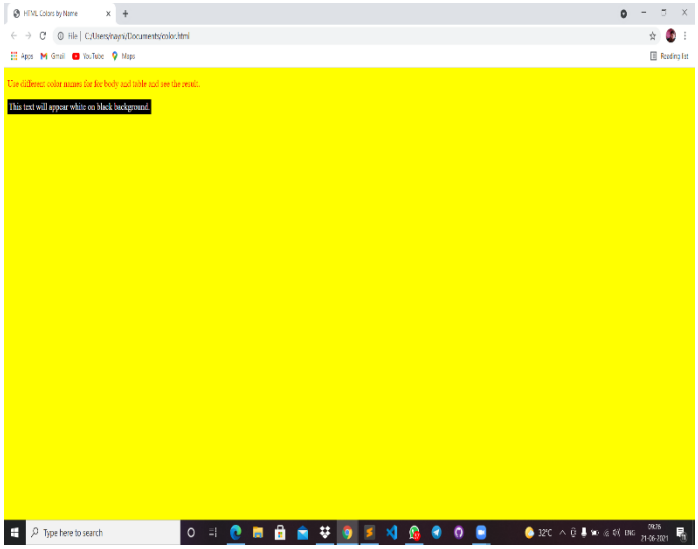
- **bgcolor** − sets a color for the background of the page.

- **text** – sets a color for the body text.
- **alink** – sets a color for active links or selected links.
- **link** – sets a color for linked text.
- **vlink** – sets a color for *visited links* – that is, for linked text that you have already clicked on

Example program:

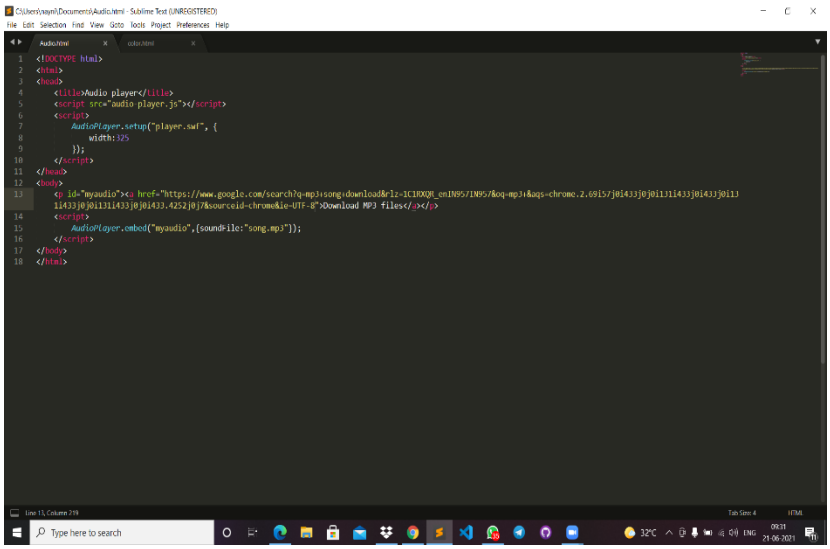


OUTPUT:

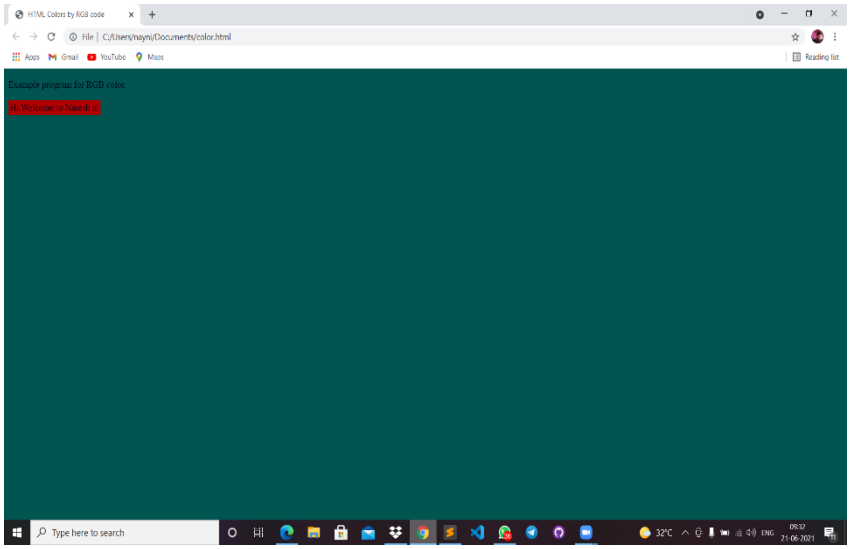


HTML Colors - RGB Values

This color value is specified using the **rgb( )** property. This property takes three values, one each for red, green, and blue. The value can be an integer between 0 and 255 or a percentage.



OUTPUT:



COLOR:HASHCODES:-

Hex color code contains the symbol **"#"** followed by 6 alphabets or numbers. Numbers are in the hexadecimal numeric system.**00** value range of the code represents the lowest intensity of color on the other hand **FF** value range of code represents the highest intensity.

- The **white color** is a mixture of the three primary colours at full intensity representing the Hex color code **#FFFFFF**.
- The **black color** is a mixture of the three primary colours at the lowest intensity representing the color code **#000000**.
  EXAMPLE: - color: #FFFFFF

Note: if anytime all 6 characters are same, we can give 3 characters.

/*Reference – geeksforgeeks, Tutorial point, W3schools*/

ASCII code:

ASCII stands for the "American Standard Code for Information Interchange". ASCII is a 7-bit character set containing 128 characters. It contains the numbers from 0-9, the upper- and lower-case English letters from A to Z, and some special characters. The character sets used in modern computers, in HTML, and on the Internet, are all based on ASCII.

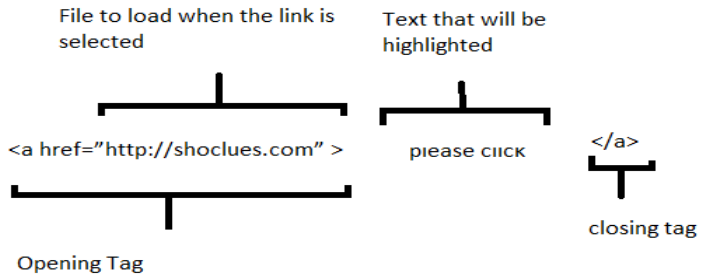
Also let us discuss about some of the anchor tags:

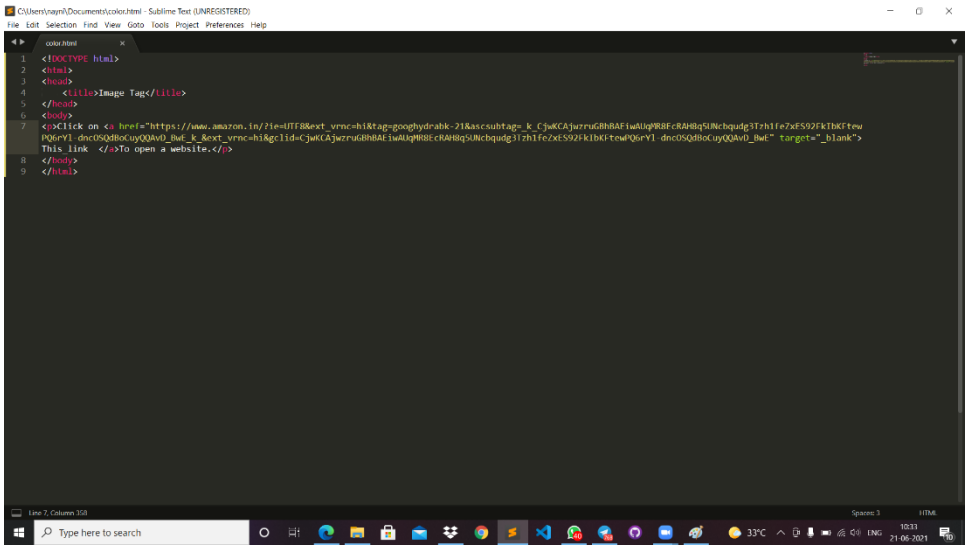
Anchor tag:

The **HTML anchor tag** defines *a hyperlink that links one page to another page*. It can create hyperlink to other web page as well as files, location, or any URL. The "href" attribute is the most important attribute of the HTML a tag. and which links to destination page or URL. use to make links like third party page like amazon, flipkart etc...

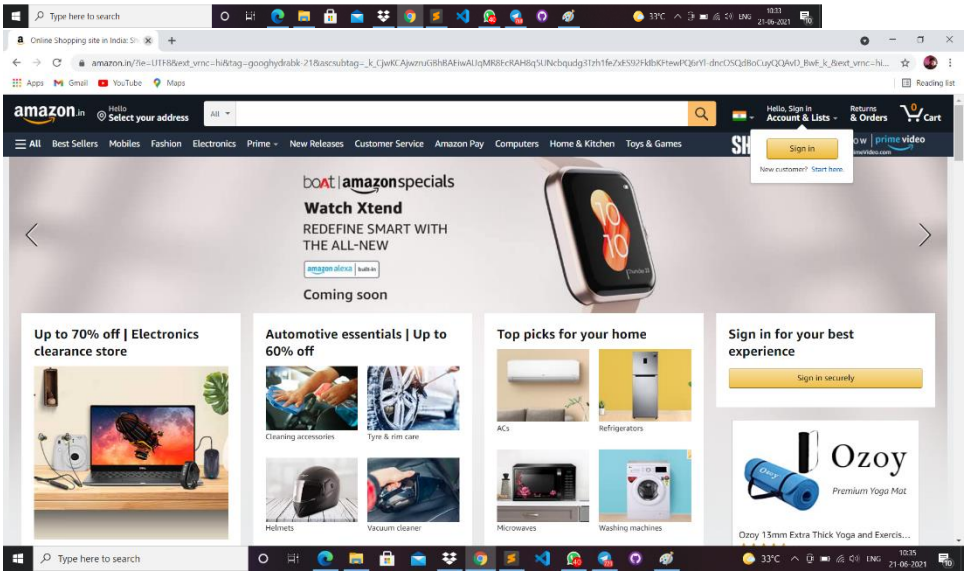
NOTE: - Href is used for third party page opening external and internal web page.

please click





OUTPUT:



Pseudo classes:

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

Target attribute:

The target attribute specifies where to open the linked document. target=" _blank" open in very blank page.

Underlining text:

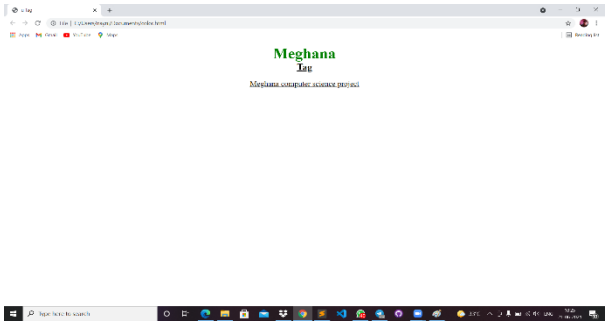
The <u> tag in HTML stands for underline and it's used to underline the text enclosed within the <u> tag. This tag is generally used to underline misspelled words. This tag requires a starting as well as ending tag.

EXAMPLE:

```
<html>
```

```
<head>
  <title>u Tag</title>
  <style>
    body {
      text-align:center;
    }
    .meg {
      font-size:40px;
      font-weight:bold;
      color:green;
    }
    .project {
      font-size:25px;
      font-weight:bold;
    }
    p {
      font-size:20px;
    }
  </style>
</head>
<body>
  <div class = "meg">Meghana </div>
  <div class = "project"><u> Tag</div>
  <p>Meghana <u>computer science</u>
            project</p>
</body>
</html>
```

OUTPUT:

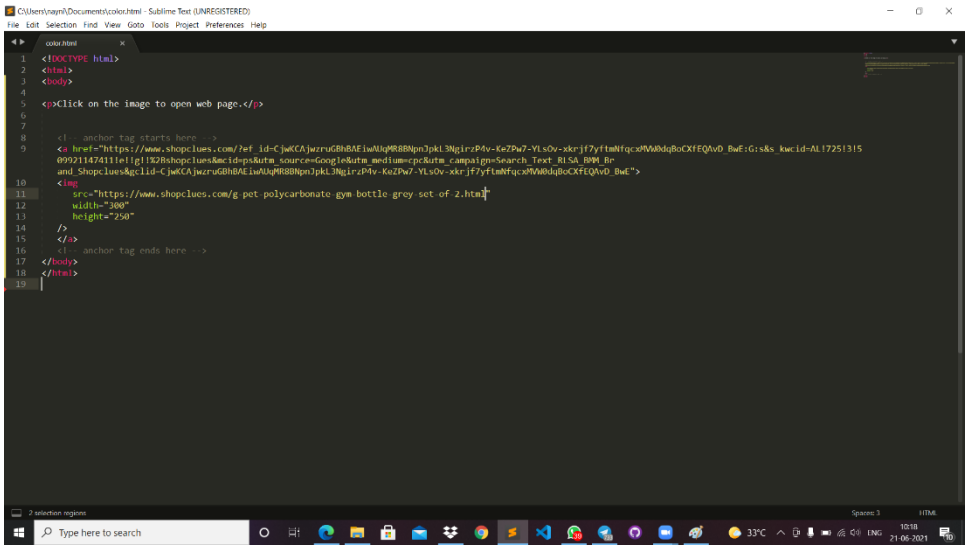


Adding images to web page:

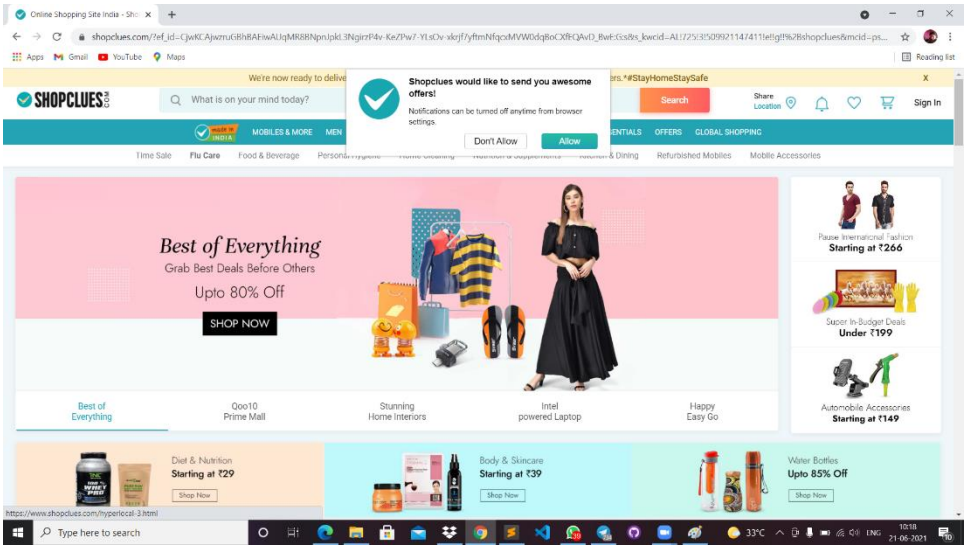
img tag: src attribute:

Syntax: <img src=”copy img link &add “

EXAMPLE:



OUTPUT: -



DATE: 22/06/2021

GitHub:

GitHub is an independent platform for full stack users basically helpful for cloning or pushing or pulling the data origin from the repositories URL address.

What is cloning? How it's used?

Cloning a GitHub repository creates a local copy of the remote repository. This allows you to make all of your edits locally rather than directly in the source files of the origin repository.

/*Note: U have to have access to get the code from the repository member if its private repository. That is, they have to add us a collaborator. */

Steps for cloning:

After getting the access as a collaborator:

1.Open the file explorer and create a separate folder in D or E Drive (UIClasses file as we are cloning the data for UI classes from Durgaprasad sir).

2.Then Open GitHub desktop (which is installed) and check whether the collaborator repository (Durgaprasad sir repository) is available or not.

3.If available click on that choose the path (the folder you have created(UIClasses folder in D or E Drive)). Then simply click on the clone option shown on the GitHub Desktop.

What is push and pull origins?

Push origin:

Used to push the assignment after fetching the data

Pull origin:

Used to pull the origins or files or codes of others (If any changes were made and resubmitted again) by other collaborators.

Submitting an assignment in GitHub: -

open GitHub account and first fetch the origin. After fetching you will get a notification to pull the origin then click on that so that what you have submitted gets saved in the GitHub. Then see on down left corner ull see a comment window keep the title as Assignment1_studentdetails and click on commit to master.

Again, ull see option on top like pull the origin do that. After all these processes completed open GitHub in chrome browser ull see Durgaprasad sir folder click on that open 7:30Am batch see whether Ur name is registered there or simplest method is open file explorer and go to D:drive and click on uiclasses(or the name ull saved) then open 7:30 Am and click on assignment folder created check Ur name is shown or not.

/*Avoid giving spaces between the file you are saving*/

/*This will be useful for all windows 64-bit compilers*/

DATE: 23/06/2021

Gmail: webdev.prasad@gmail.com

Assignment 1 – Batch 7Am (DOM)

Body: file location

Notes:

Different ways of injecting CSS to an element:

1. Adding Inline CSS through style Attribute. Use. (dot) operator
2. Creating a class, injecting through class attribute.
3. Creating a class through id of an element. Use # (hash) operator

4. Creating a CSS class using the tag name itself. No need to make use of any operators.

- Creating a class through id of an element:

If there is an element for which we are adding CSS and there is id attribute already injected then we can define a class through id of the element using # operator.

DOM Structure (DOCUMENT OBJECT MODEL): -

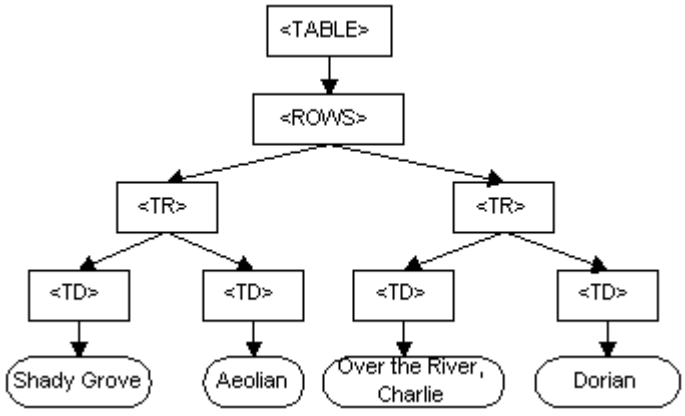
The Document Object Model (DOM) is a programming API for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated. In the DOM specification, the term "document" is used in the broad sense - increasingly, XML is being used as a way of representing many different kinds of information that may be stored in diverse systems, and much of this would traditionally be seen as data rather than as documents. Nevertheless, XML presents this data as documents, and the DOM may be used to manage this data.

→ For Every web page browser dynamically creates a DOM structure which indicates the tree structure of current webpage with all the elements.

→ DOM structure specifies the relationship between the elements, list of elements participated within the page, attributes and its corresponding values etc.

- Example:

→



DEBUGGER TOOL:

Every browser by default comes with a default debugger tool using which we could be able to debug any webpage running under the browser.

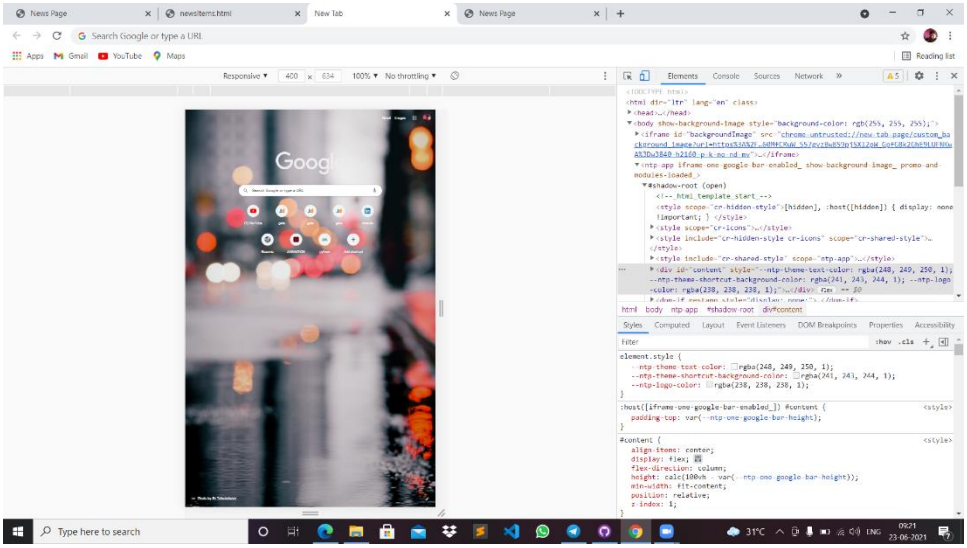
1. It shows the DOM structure of the current webpage.
2. It provides an option to create a dynamic HTML element while the page is running.

3. It provides an option to edit, delete, update any HTML element and its corresponding attributes while the page is running.
4. Any CSS property of any HTML element can be edited, deleted, add a new property can be done through debugging tool.
5. It also provides an option to explore list of resources been loaded in the current webpage.
6. It provides an option to explore list of resources getting loaded through network or server (images, js files, CSS files, XHR requests etc.)

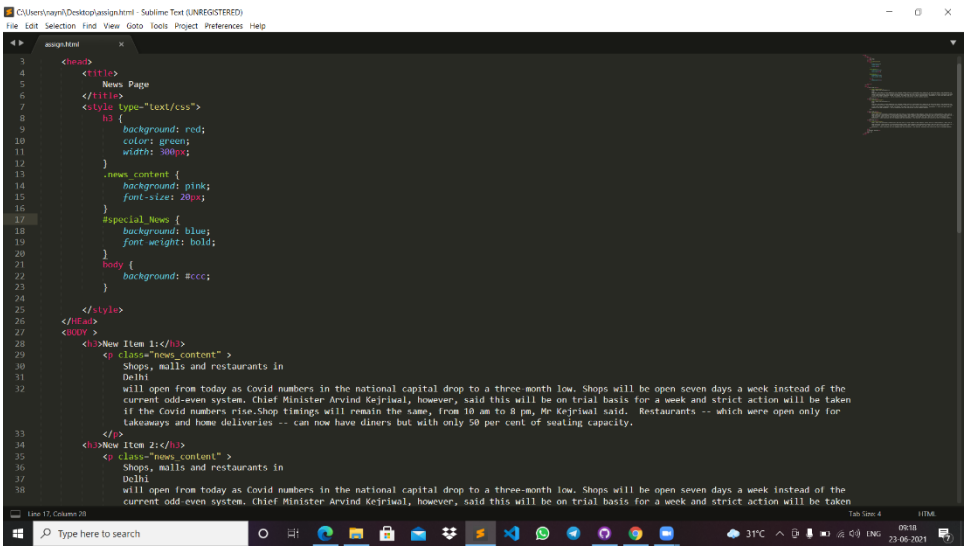
/* It is a temporary changeable tool*/

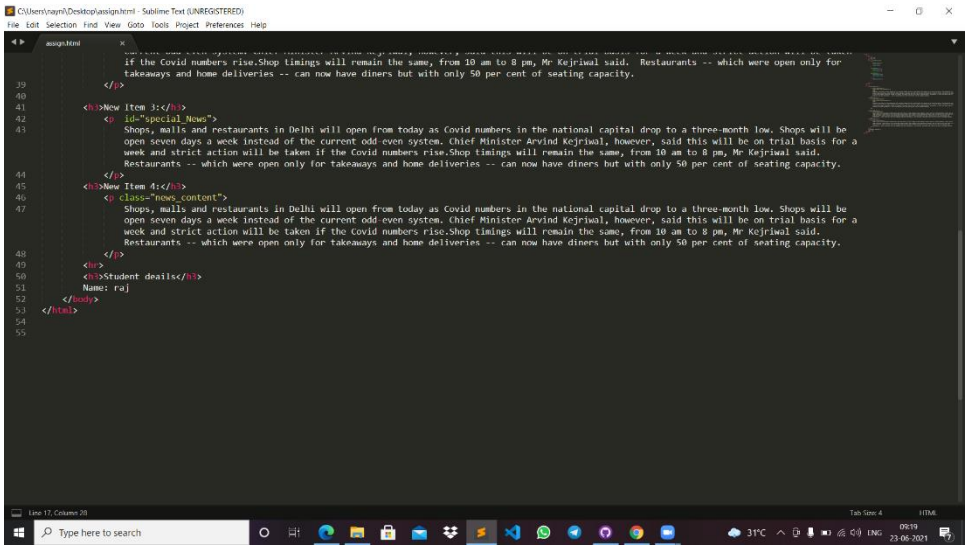
/* It is one of very important tool for UI development*/

Example of Debugging window:

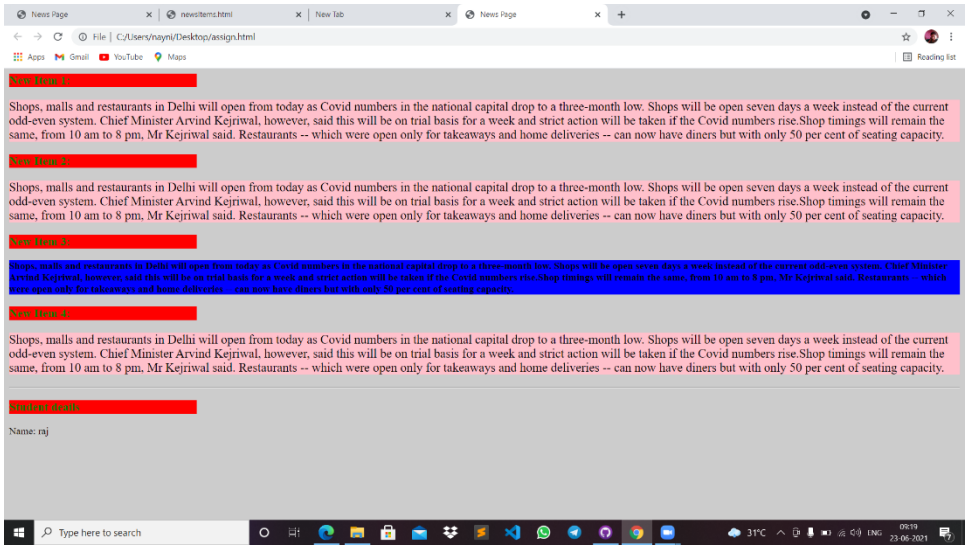


TODAY'S EXAMPLE:





OUTPUT:



Assignment: Practice with debugging tool.

DATE: 24/06/2021

Debugger tool:

Debugging is the process of finding and fixing errors within a script. All modern browsers and most other environments support debugging tools – a special UI in developer tools that makes debugging much easier. It also allows to trace the code step by step to see what exactly is going on.

Mainly debugging tool consists of 8 parts:

1.Elements

Add attribute — add a new attribute to the selected element
Edit attribute — edit an attribute, available only when you click on the attribute
Edit as HTML — by picking this one, you can edit the whole element; also, useful to copy part of an element that you want to use somewhere else

Copy:

Copy outer HTML — copy tag, including a tag itself and a child element
Copy selector — copy of a CSS selector (div > span > #id)
Copy XPath — copy of an XPath //*[@id=" answer11208745–20"]/div/div [3]/time, more in further reading
Cut element — cuts element
Copy element — copy element and children's elements
Hide element — hide element temporarily by adding display: none; (cmd + H / ctrl+H)
Delete element — delete element and children's elements, can be reversed by cmd + z
Expand all — expand all nodes
Collapse all — collapse all nodes
: active — set element in active state*
: hover — set element in hover state*
: focus — set element in focus state*
: visited — set element in visited state*
Scroll into view — gets you immediately to the selected element on web page

2.Resources

3.Network

4.Sources

The Sources panel has 3 parts:

The File Navigator pane lists HTML, JavaScript, CSS and other files, including images that are attached to the page. Chrome extensions may appear here too.
The Code Editor pane shows the source code.
The JavaScript Debugging pane is for debugging,

5.Timeline

6.Profile

7.Audits

8.Console

If we press Esc, then a console opens below. We can type commands there and press Enter to execute.

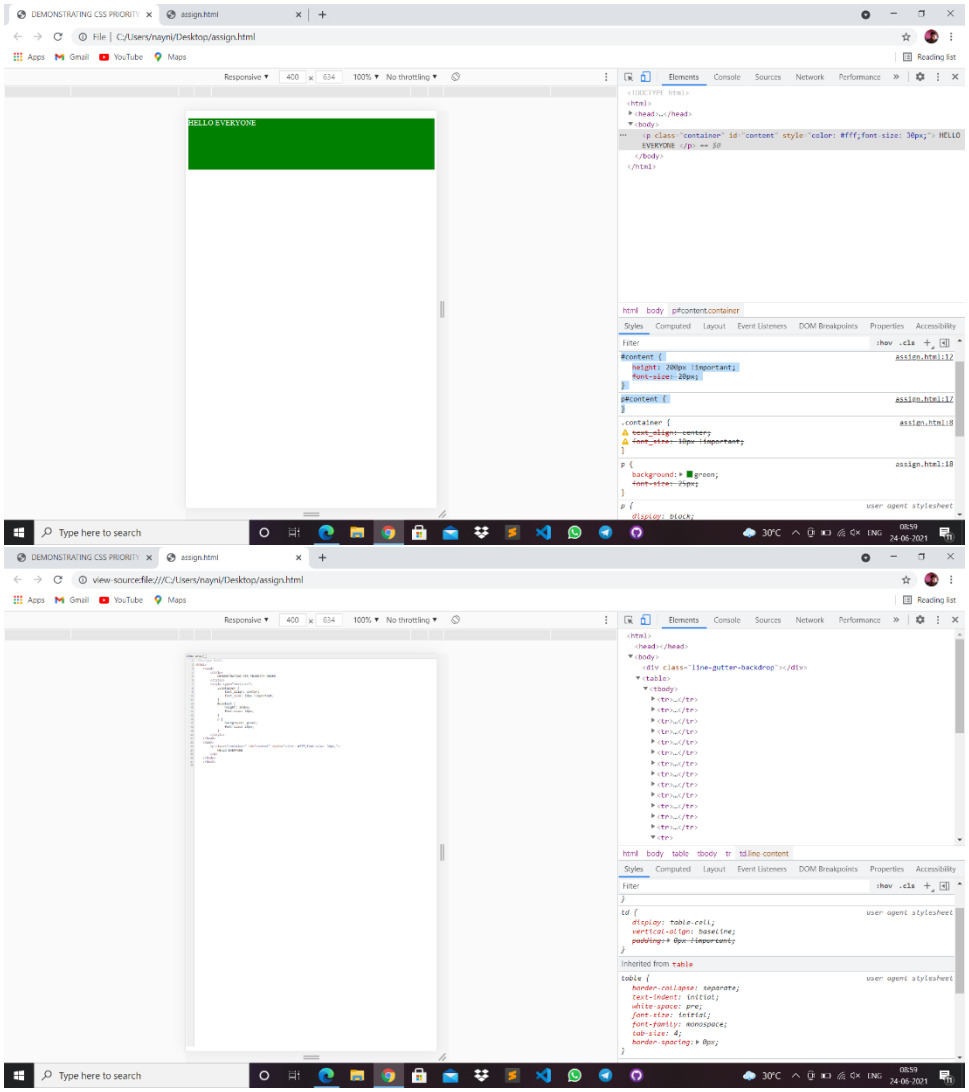
After a statement is executed, its result is shown below.

For example, here 1+2 results in 3, and hello("debugger") returns nothing, so the result is undefined:

EXAMPLE:

```
<!Doctype html>
<html>
<head>
<title>
DEMONSTRATING CSS PRIORITY ORDER
</title>
<style type="text/css">
.container {
text_align: center;
font_size: 10px !important;
}
#content {
height: 200px !important;
font-size: 20px;
}
p {
background: green;
font-size: 25px;
}
</style>
</head>
<body>
<p class="container" id="content" style="color: #fff; font-size: 30px;">
HELLO EVERYONE
</p>
</body>
</html>
```

OUTPUT:



CSS priorities:

As we have the 4 different ways to apply CSS properties for any HTML element, if the same CSS property being applied to a single element using al the 4 different ways with different values, browser by default follows the below priority order.

1. Among all the 4 different ways CSS been applied inline (through style attribute) will always takes higher priority in order.
2. CSS been applied through id based takes second priority in order.
3. CSS been applied through class takes third priority in order.
4. CSS applied through tag-based class takes least priority in order.

Note: - Irrespective of above default priority order any CSS property being added with! important will always takes highest priority in order.

DATE:25/06/2021

Interview Questions

❖ Inline and block level elements:

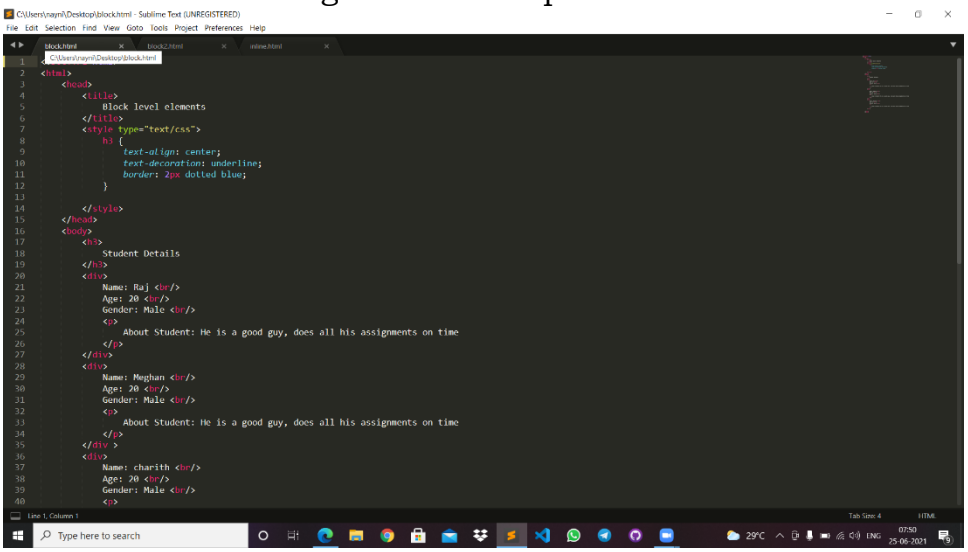
All the html elements are been categorised into two types:

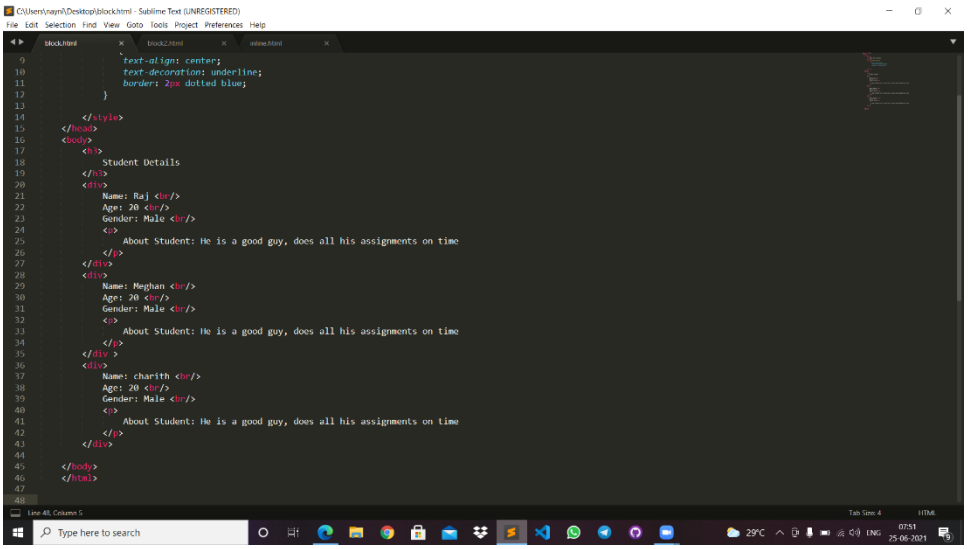
1.Block level element

2.Inline element

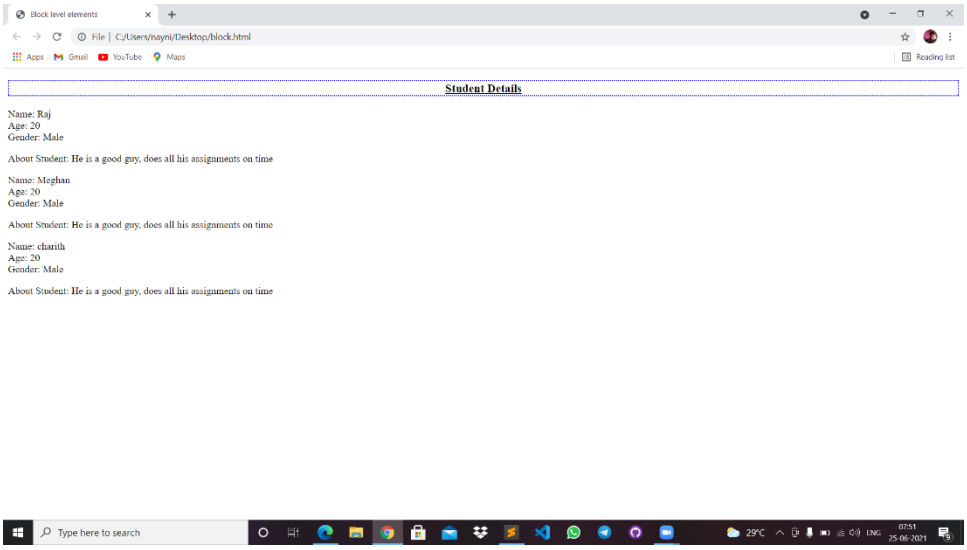
1.Block level element: Any HTML element which comes under the block level category holds the following properties:

- It occupies 100% width of its container by default
- Even though it occupies 100% width we can still control the dimensions of block level elements through CSS width and height properties.
- While getting rendered with in the page by default it comes to new line and gets rendered.
- The element which is following a block level element also automatically comes to a new line and gets rendered.
- Block level elements are mainly used to hold group of relative items as lightly individual block.
- tag is best example for block level element.



- 

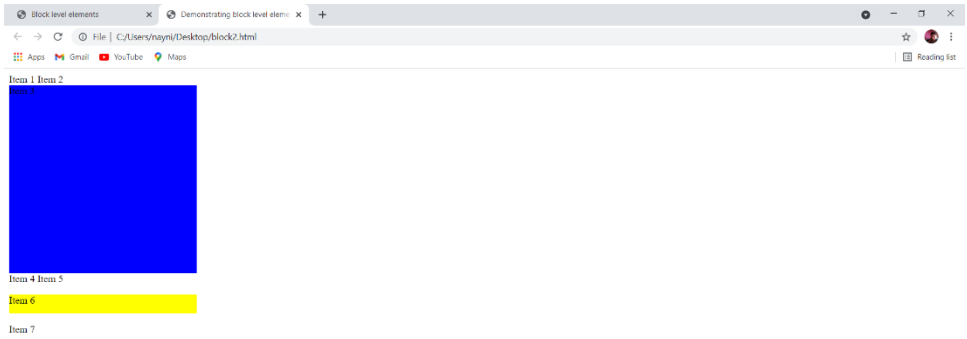
OUTPUT:



Example 2:

```
<!DOCTYPE html>
<html>
    <head>
        <title>
                Demonstrating block level elements
        </title>
        <style type="text/css">
                .container {
                    background: blue;
                    width: 300px;
                    height: 300px;
                }
                p {
                    background: yellow;
                    width: 300px;
                    height: 30px;
                }
        </style>
    </head>
    <body>
        Item 1
        Item 2
        <div class="container"> Item 3</div>
        Item 4
        Item 5
        <p>Item 6</p>
        Item 7
    </body>
    </html>
```

OUTPUT:



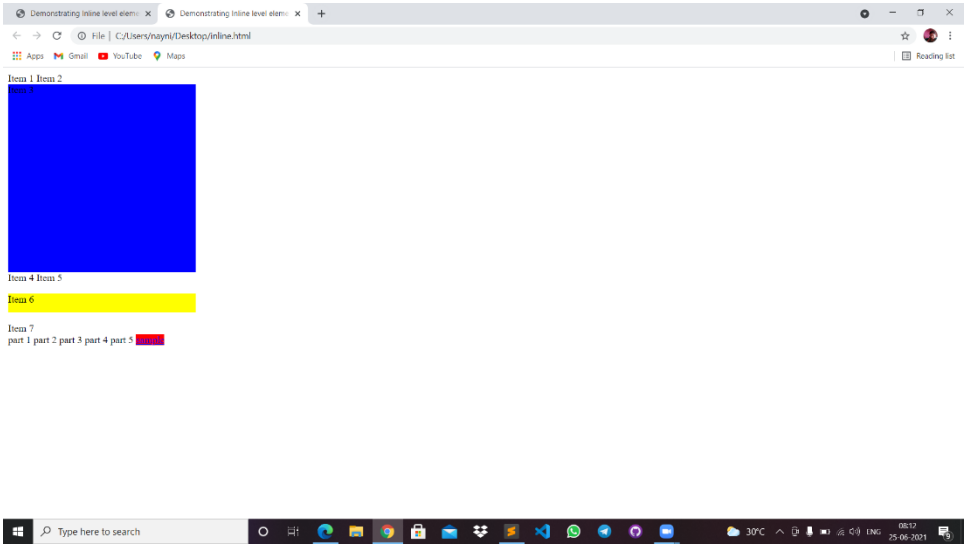
2.Inline element: Any HTML element comes under inline category holds following properties:

- While rendering on the page all the inline elements try to render within the same line.
- Inline element always occupies the width within the container based on the content it is holding.
- We cannot control the dimensions of inline element.
- ‘span’ tag is the best example for inline element.

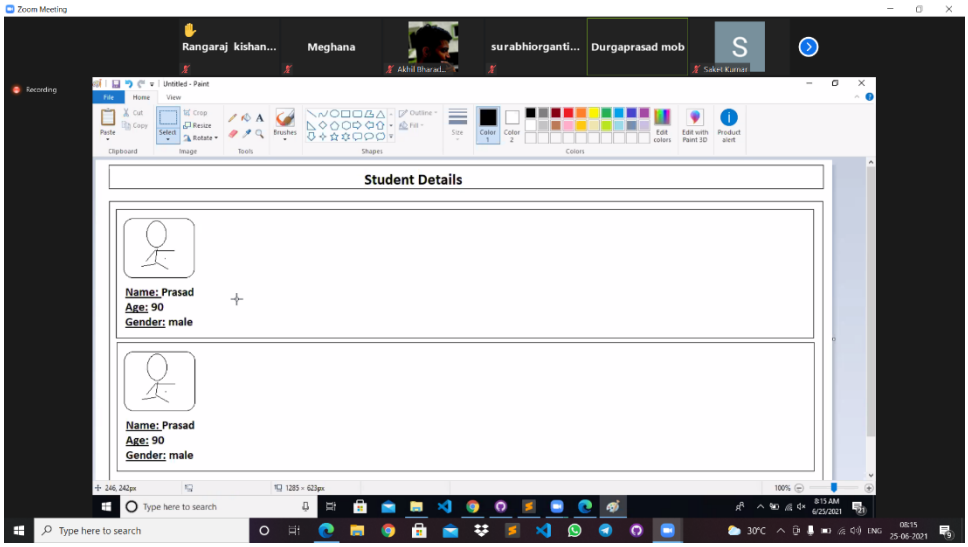
/* Any element in inline block doesn’t occupy new line*/

```
<!DOCTYPE html>
<html>
    <head>
        <title>
                Demonstrating inline level elements
        </title>
        <style type="text/css">
                .container {
                    background: blue;
                    width: 300px;
                    height: 300px;
                }
                p {
                    background: yellow;
                    width: 300px;
                    height: 30px;
                }
                a {
                    background: red;
                    width: 600px;
                }
        </style>
    </head>
    <body>
        Item 1
        Item 2
        <div class="container"> Item 3</div>
        Item 4
        Item 5
        <p>Item 6</p>
        Item 7
        <br/>
        part 1
        part 2
        <span class="part 3">part 3</span>
        part 4
        part 5
        <a href="http://abc.com">sample</a>
    </body>
    </html>
```

OUTPUT:

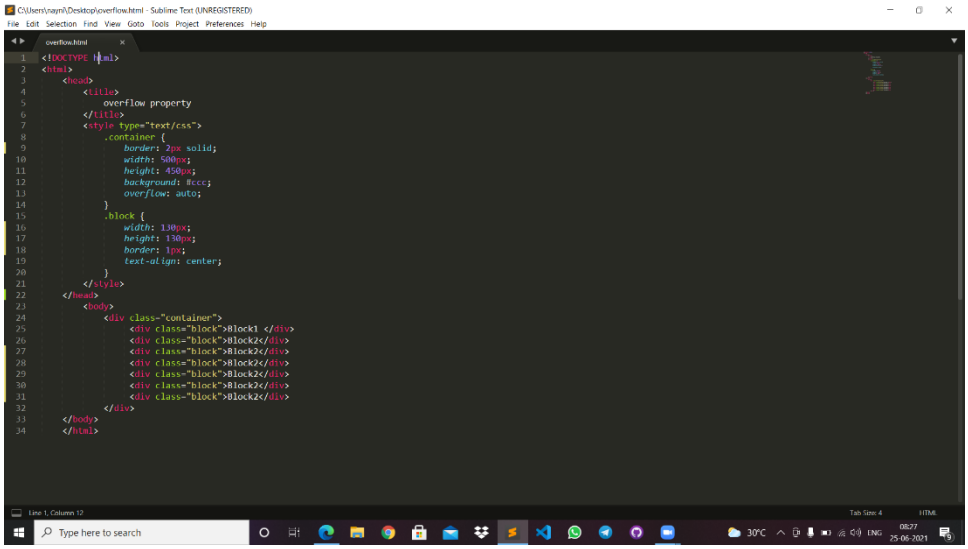


ASSIGNMENT-2:

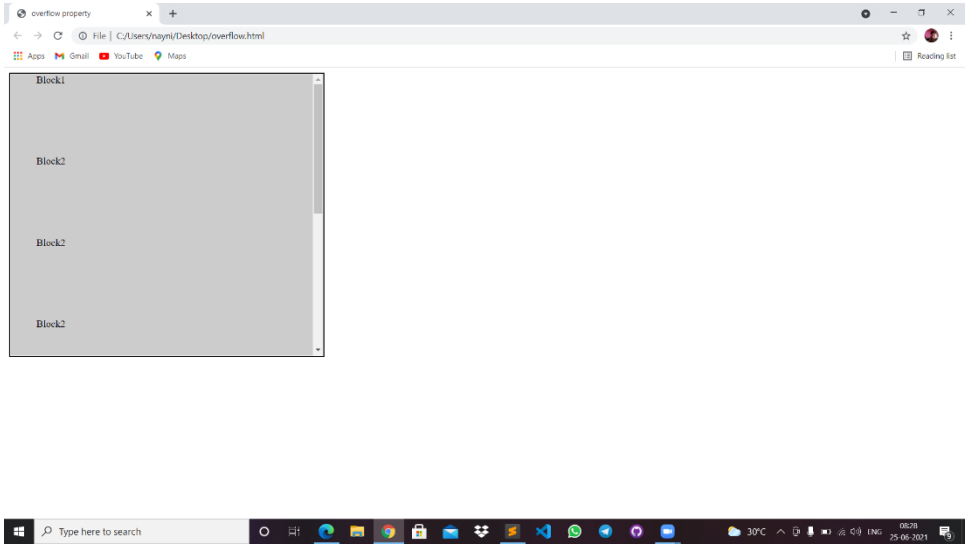


Minimum of 10 students

Overflow:



OUTPUT:



Span Tag:

- The span tag is an inline container used to mark up to a part of text, or a part of document.
- It is easily styled by the CSS or manipulated with JavaScript using the class or id attribute.
- The tag is much like the element, but is a block-level element and is an inline element.

NOTE: The tag also supports the Global attribute in HTML. And also supports the Event attribute.

Div Tag:

- ★ The div tag is known as Division tag. The div tag is used in HTML to make divisions of content in the web page like (text, images, header, footer, navigation bar, etc).
- ★ Div tag has both open( ) and closing ( ) tag and it is mandatory to close the tag.
- ★ The Div is the most usable tag in web development because it helps us to separate out data in the web page.
- ★ we can create a particular section for particular data or function in the web pages.

Properties	Div Tag	Span Tag
Elements Types	Block-Level	Inline
Space/Width	Contain Whole Width Available	Takes only required Width
Examples	Headings, Paragraph, form	Attribute, image
Uses	Web-layout	container for soome text
Attributes	Not required,with common css, class	Not required,with common css, class

CSS Overflow: -

The CSS overflow property controls what happens to content that is too big to fit into an area.

The overflow property has the following values:

- Visible - Default. The overflow is not clipped. The content renders outside the element's box
- div {
  width: 200px;

```
  height:                    50px;
  background-color:          #eee;
  overflow:                  visible;
}
```

- hidden - The overflow is clipped, and the rest of the content will be invisible

```
div                          {
  overflow:                  hidden;
}
```

- scroll - The overflow is clipped, and a scrollbar is added to see the rest of the content

```
div                          {
  overflow:                  scroll;
}
```

- auto - Similar to scroll, but it adds scrollbars only when necessary

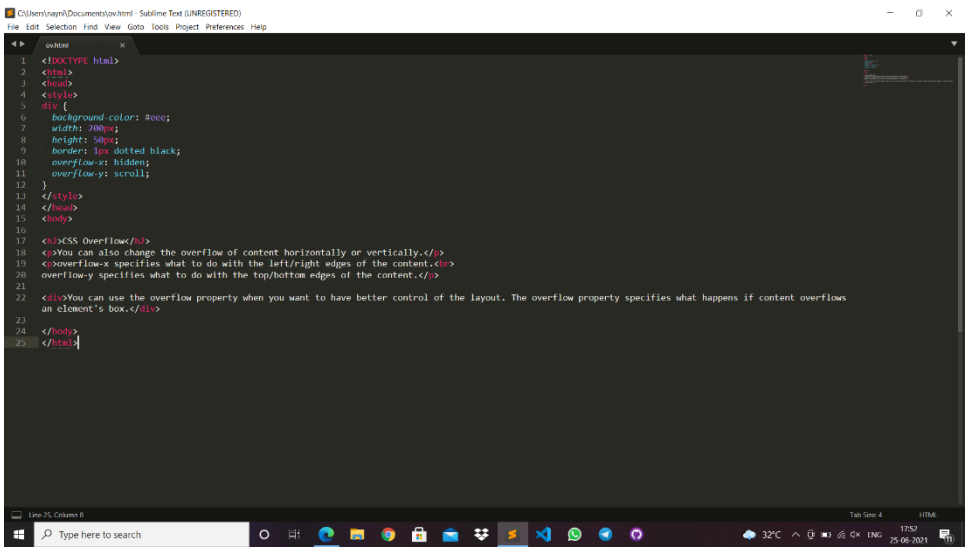
```
div                          {
  overflow:                  auto;
}
```

Overflow-x and Overflow-y:

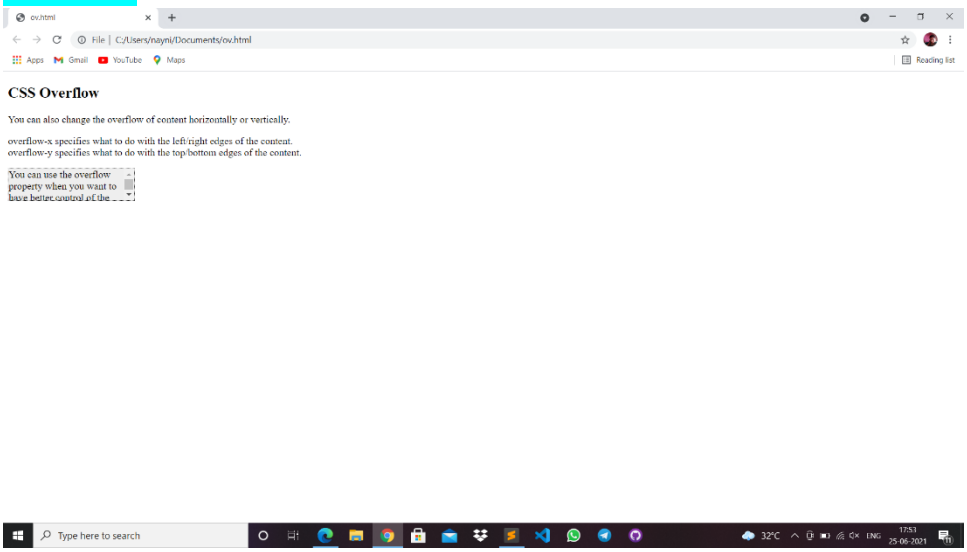
The Overflow-x and Overflow-y properties specifies whether to change the overflow of content just horizontally or vertically (or both):

1. Overflow-x specifies what to do with the left/right edges of the content.
2. Overflow-y specifies what to do with the top/bottom edges of the content.

EXAMPLE:



OUTPUT:



DATE:26/06/2021

CSS PADDING AND MARGIN properties:

By default, while the elements getting rendered on the page, they try to occupy very next line of the previous element without maintaining any space. In order to add space or gap between or within the elements we make use of the CSS property called as 'Padding' or 'Margin'.

CSS 'Padding' Property: -

Through padding property, we could able to add space or gap between the border and content of the container.

→ Syntax: padding: 10px; Adds padding of 10px to all directions (top,left,right and bottom)

→ Padding: 5px 10px; Adds padding of 5px to top and bottom, 10px to left and right.

→ Padding: 5px 6px 10px 15px;

Top right Bottom left

→ Padding-top: 10px;

→ Padding-left: 35px;

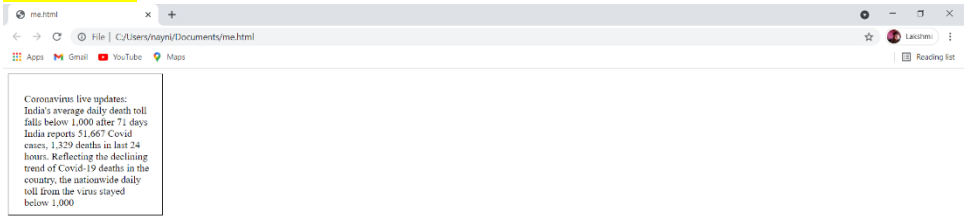
→ Padding-right: 10px;

→ Padding-bottom: 10px;

EXAMPLE: -

```
<style type="text/css">
    div.special2
    {
        width:200px;
        border-style: solid;
        border-width:thin;
        border-color:#000;
        padding:30px 20px 10px 25px;
    }
</style>
<div class="special2">
    Coronavirus live updates: India's average daily death toll falls below 1,000 after 71 days
India reports 51,667 Covid cases, 1,329 deaths in last 24 hours. Reflecting the declining trend of Covid 19 deaths in the country, the nationwide daily toll
from the virus stayed below 1,000
</div>
```

OUTPUT:



CSS 'Margin' property:

Through which we could be able to add space or gap above the border of container.

→ Syntax: margin: 10px; Adds padding of 10px to all directions (top,left,right and bottom)

→ margin: 5px 10px; Adds padding of 5px to top and bottom, 10px to left and right.

→ margin: 5px 6px 10px 15px;

Top right Bottom left

→ margin-top: 10px;

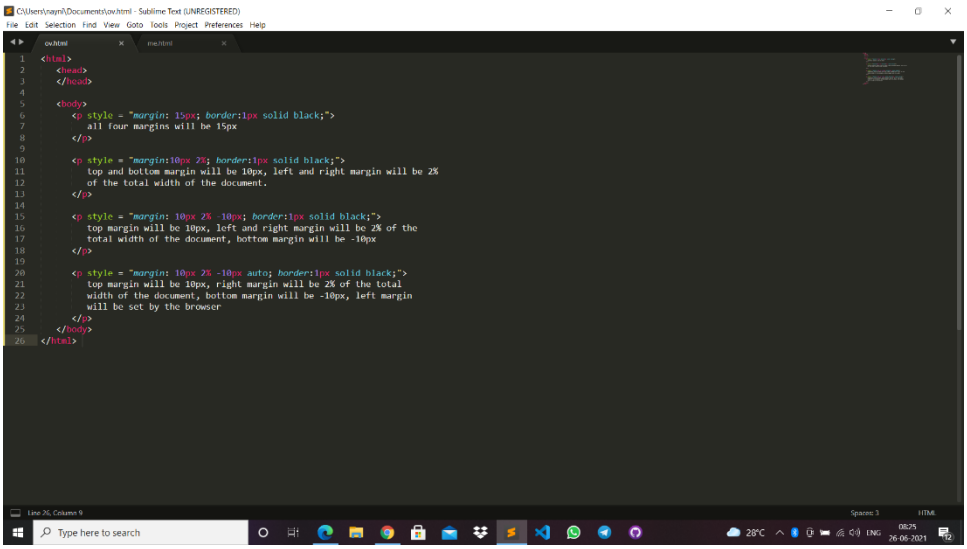
→ margin-left: 35px;

→ margin-right: 10px;

→ margin-bottom: 10px;

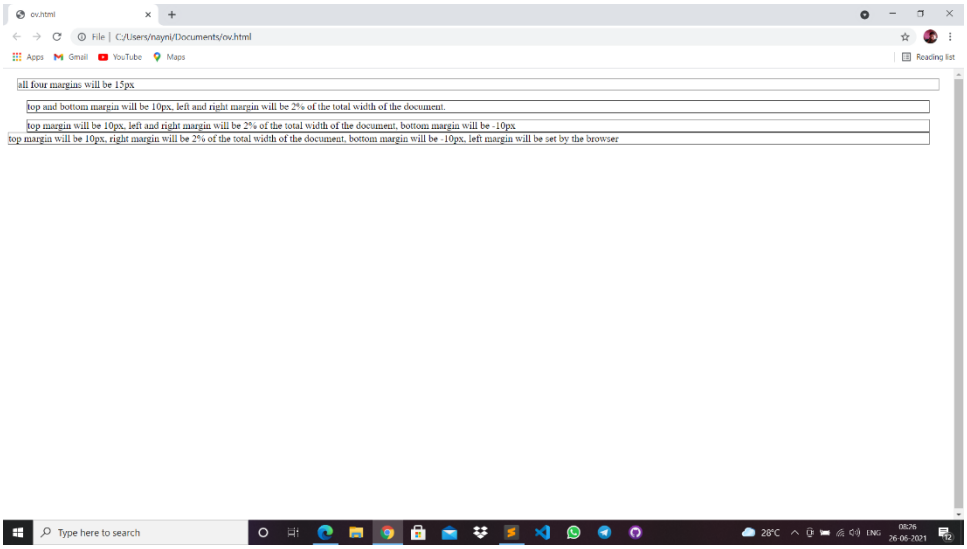
→ Margin:10px auto; Adds 10px of margin for top and bottom, moves the element to the center of container horizontal.

EXAMPLE:



```
<html>
   <head>
   </head>

   <body>
      <p style = "margin: 15px; border:1px solid black;">
         all four margins will be 15px
      </p>

      <p style = "margin:10px 2%; border:1px solid black;">
         top and bottom margin will be 10px, left and right margin will be 2%
         of the total width of the document.
      </p>

      <p style = "margin: 10px 2% -10px; border:1px solid black;">
         top margin will be 10px, left and right margin will be 2% of the
         total width of the document, bottom margin will be -10px
      </p>

      <p style = "margin: 10px 2% -10px auto; border:1px solid black;">
         top margin will be 10px, right margin will be 2% of the total
         width of the document, bottom margin will be -10px, left margin
         will be set by the browser
      </p>
   </body>
</html>
```

OUTPUT:

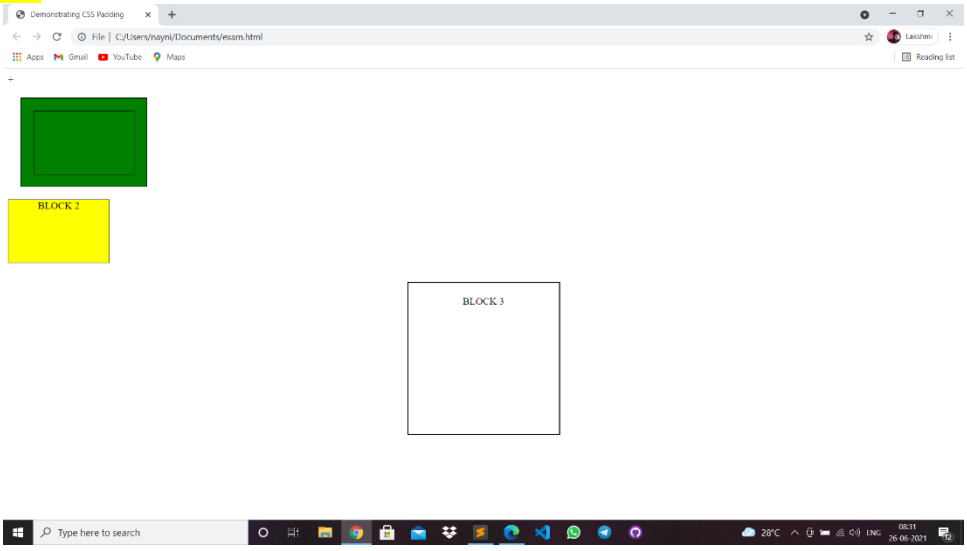


TODAY'S EXAMPLE:

```
<!DOCTYPE html>
<html>
    <head>
        <title>
            Demonstrating CSS Padding
        </title>
        <style type="text/css">


            .block {
                border: 1px solid;
                width: 150px;
                height: 100px;
                text-align: center;
            }

            .block1 {
                background: green;
                padding: 20px;
                margin: 20px;
            }
            .block2 {
                background: yellow;
            }
            .block3 {
                margin: 30px auto;
                border: 2px solid;
                text-align: center;
                width: 200px ;
                padding: 20px;
                height: 200px;
            }
        </style>
    </head>
    <body>
        <div class="block block1">
            <div class="block"></div>
        </div>
        <div class="block block2">BLOCK 2</div>
```

```
            padding: 20px;
            margin: 20px;
        }
        .block2 {
            background: yellow;
        }
        .block3 {
            margin: 30px auto;
            border: 2px solid;
            text-align: center;
            width: 200px ;
            padding: 20px;
            height: 200px;
        }
    </style>
</head>
<body>
    <div class="block block1">
        <div class="block"></div>
    </div>
    <div class="block block2">BLOCK 2</div>

    <div class="block3">BLOCK 3</div>
</body>
</html>
```

OUTPUT:



DATE:28/06/2021

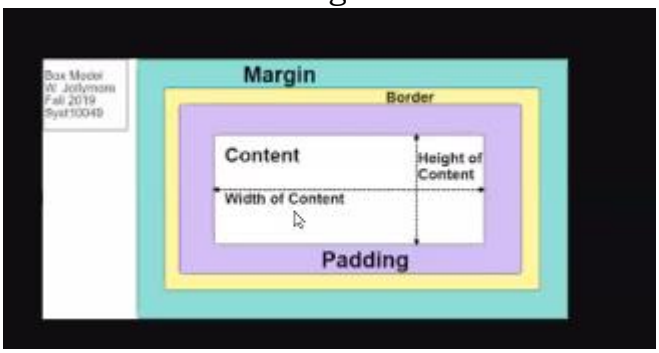
CSS Box Model:

When an element gets rendered on a page, browser calculates the actual dimensions of the existing element on the page, accordingly it renders the new DOM element.

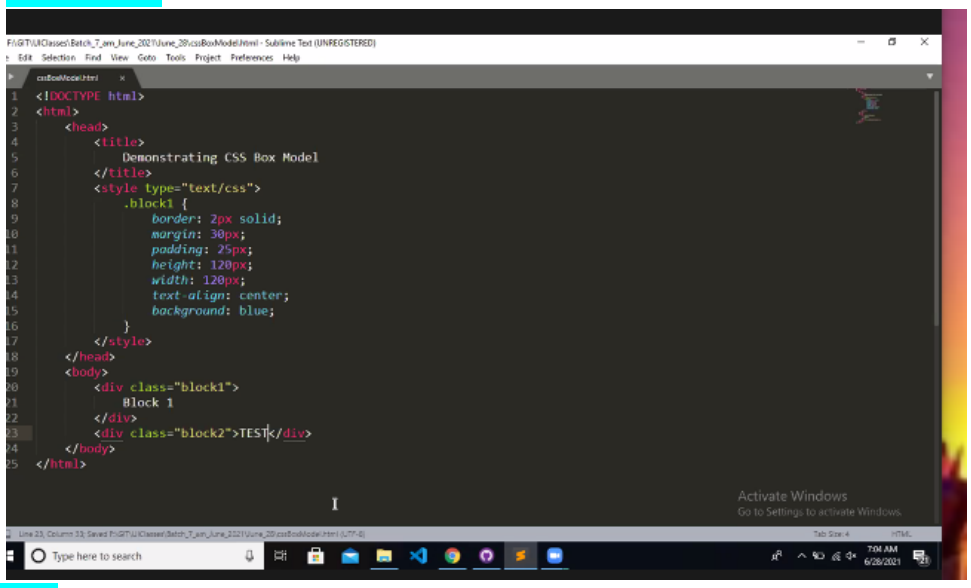
While calculating the dimensions of an element, it follows Box model feature.

In CSS box model property while calculating the actual dimensions of an element, it considers the following CSS properties:

- The total border space been occupied.
- The total margin space is occupied
- The total padding space is occupied.
- Actual width and height of Dom Element.

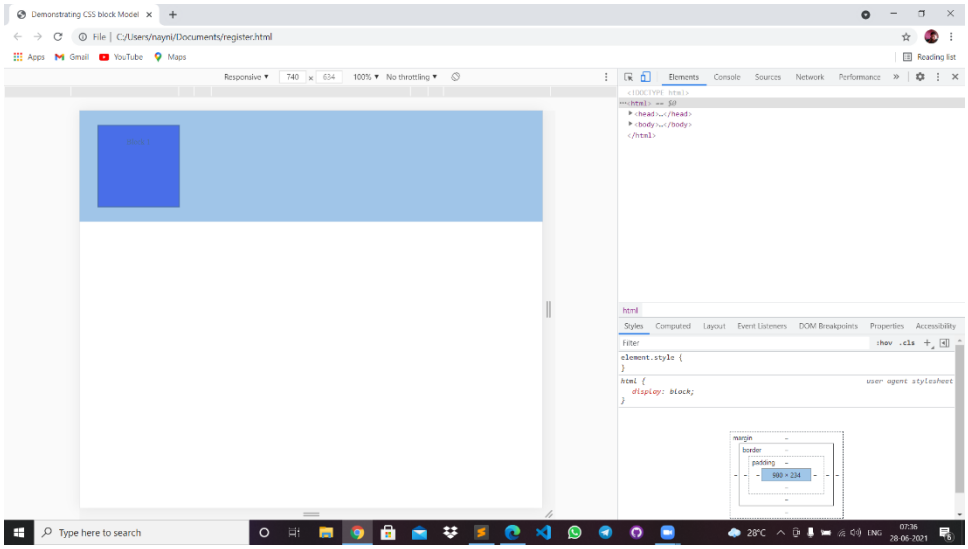


EXAMPLE:

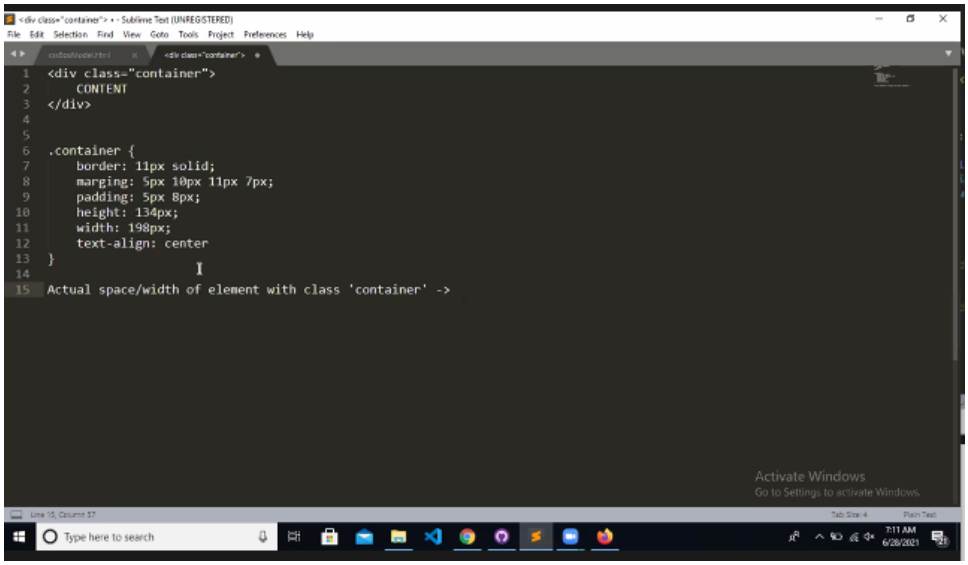


```
<!DOCTYPE html>
<html>
    <head>
        <title>
            Demonstrating CSS Box Model
        </title>
        <style type="text/css">
            .block1 {
                border: 2px solid;
                margin: 30px;
                padding: 25px;
                height: 120px;
                width: 120px;
                text-align: center;
                background: blue;
            }
        </style>
    </head>
    <body>
        <div class="block1">
            Block 1
        </div>
        <div class="block2">TEST</div>
    </body>
</html>
```

OUTPUT: -



Sample Example to find width: -



Width of element: - 11+11+7+10+8+8+198=253

HTML UL tags and OL tags:

The two predefined HTML tags using which we could able to group relative items as an individual block.

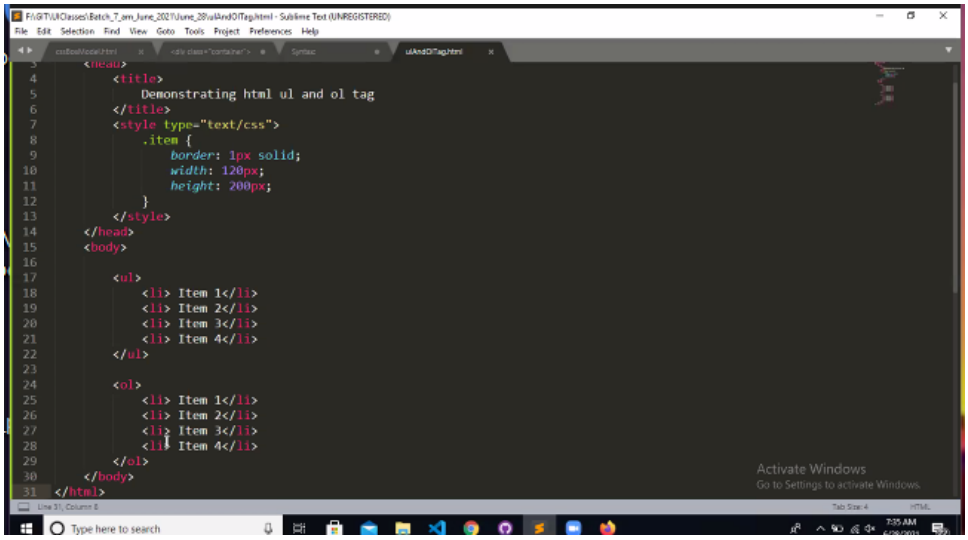
'ul' and ' ol 'tag renders the element almost same where the only difference is ul tag (unordered list) renders the element with default bullet symbols, Ol tag (ordered list) render the element with default serial number.

'List-style' is the CSS property through which we can control the type of list style getting rendered for both ul and ol tag.

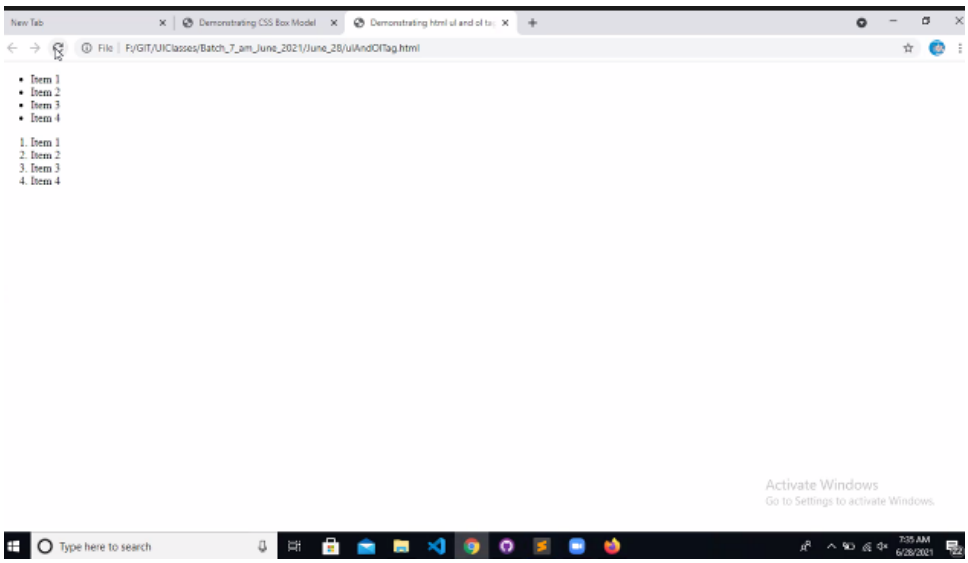
Syntax:

```
<ul>
<li>item1</li>
<li>item2</li>
<li>item3</li>
.....
</ul>
<ol>
<li>item1</li>
<li>item2</li>
<li>item3</li>
.....
</ol>
```

Example: -



OUTPUT: -

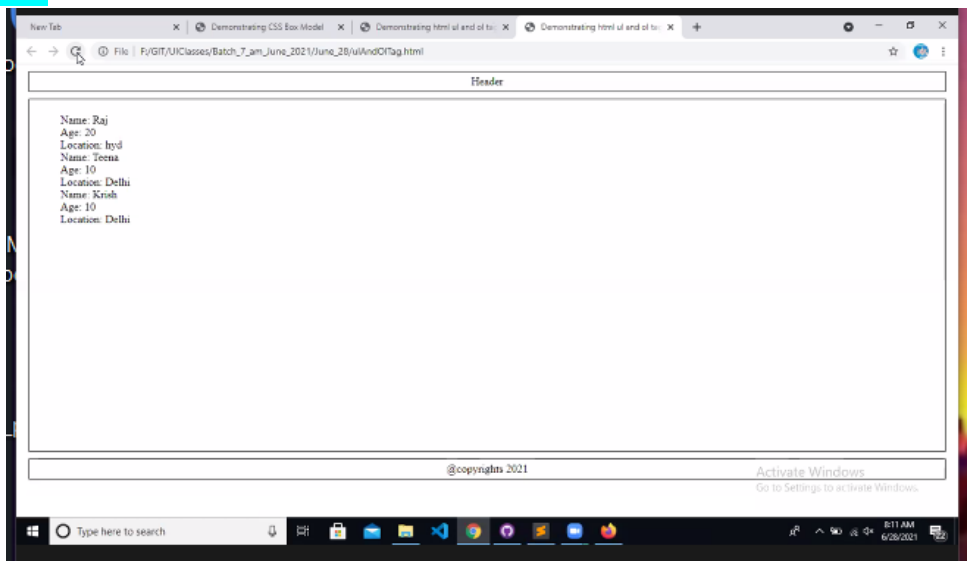


Sample Example: -

```
            .container {
                height: 500px;
            }
            .detailsConntent {
                text-align: left;
            }
        </style>
    </head>
    <body>
        <div class="header">Header</div>
        <div class="container">
            <div class="detailsConntent">
                <div>
                    <div>Name: Raj</div>
                    <div>Age: 20</div>
                    <div>Location: hyd</div>
                </div>

                <div>
                    <div>Name: Teena</div>
                    <div>Age: 10</div>
                    <div>Location: Delhi</div>
                </div>

                <div>
                    <div>Name: Krish</div>
                    <div>Age: 10</div>
                    <div>Location: Delhi</div>
```

```
    </head>
    <body>
        <div class="header">Header</div>
        <div class="container">
            <ol class="detailsConntent">
                <li>
                    <div>Name: Raj</div>
                    <div>Age: 20</div>
                    <div>Location: hyd</div>
                </li>

                <li>
                    <div>Name: Teena</div>
                    <div>Age: 10</div>
                    <div>Location: Delhi</div>
                </li>

                <li>
                    <div>Name: Krish</div>
                    <div>Age: 10</div>
                    <div>Location: Delhi</div>
                </li>
            </ol>
        </div>
        <div class="footer">@copyrights 2021</div>

    </body>
</html>
```

OUTPUT:



DATE: 29/06/2021

Background-Property:

→ Using the background attribute in the HTML code of your pages, you can reach really attractive color effects for your web presence. You can use that attribute in two basic ways - through the Cascading Style Sheets (CSS).

→ The background-position property is used to define the initial position of the background image. By default, the background image is placed on the top-left of the webpage.

→ You can set the following positions:

- center
- top
- bottom
- left
- Right

Properties of Background image:

Property	Description
background	Sets all the background properties in one declaration
background-attachment	Sets whether a background image is fixed or scrolls with the rest of the page
background-clip	Specifies the painting area of the background
background-color	Sets the background color of an element
background-image	Sets the background image for an element
background-origin	Specifies where the background image(s) is/are positioned
background-position	Sets the starting position of a background image
background-repeat	Sets how a background image will be repeated
background-size	Specifies the size of the background image(s)

Reference:-https://www.ntchosting.com/encyclopedia/scripting-and-programming/background/

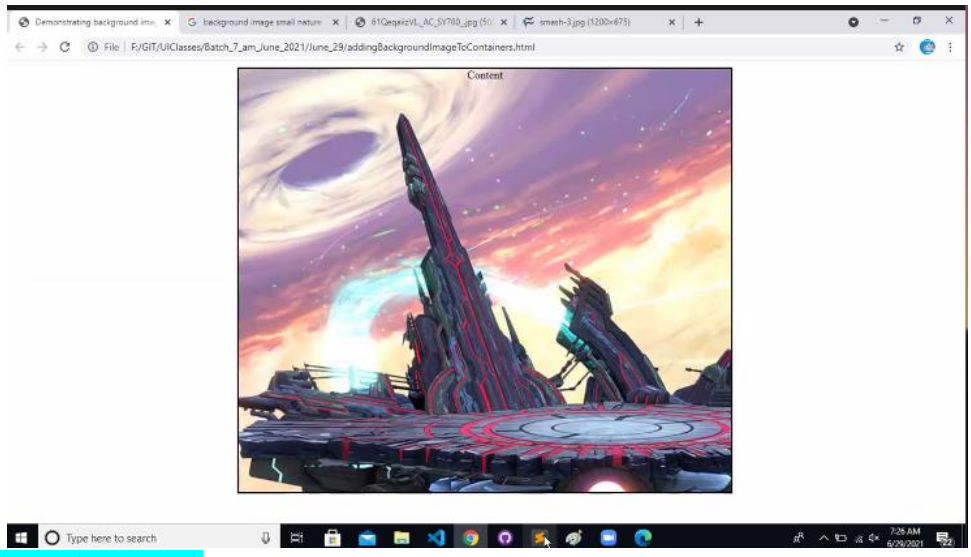
Background-image property:

Using this property, you can set an image as a background for either a separate element on your site or for the whole site.

Example:

```
<!DOCTYPE html>
<html>
    <head>
        <title>
            Demonstrating background images
        </title>
        <style type="text/css">
            .container {
                border: 2px solid;
                text-align: center;
                width: 700px;
                height: 600px;
                margin: 10px auto;
                background-image: url('https://images-na.ssl-images-amazon.com/images/I/61QeqaiizVL._AC_SY780_.jpg');
                background-repeat: no-repeat;

            }
        </style>
    </head>
    <body>
        <div class="container">
            Content
        </div>
    </body>
</html>
```

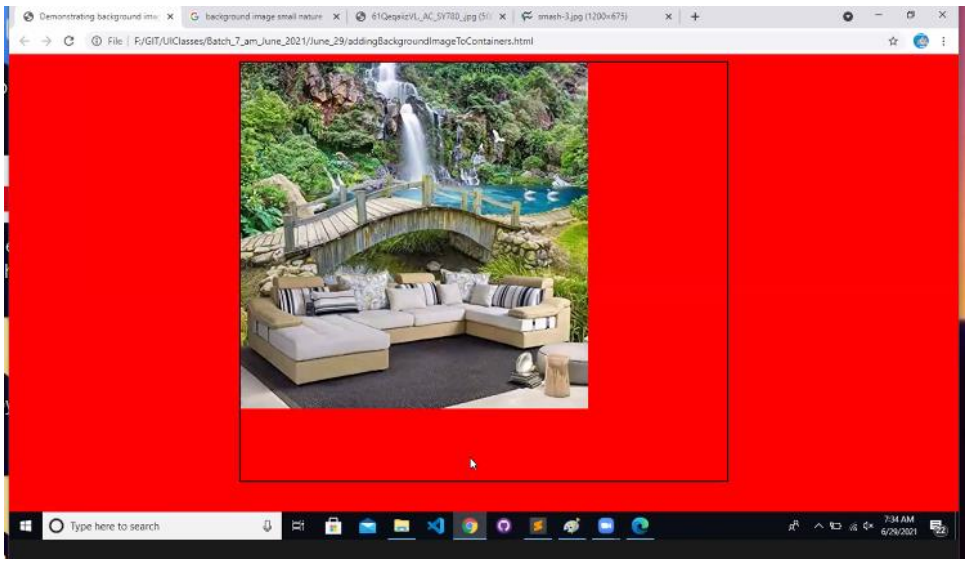
OUTPUT:



To Cover full Background:

```
<!DOCTYPE html>
<html>
    <head>
        <title>
            Demonstrating background images
        </title>
        <style type="text/css">
            .container {
                border: 2px solid;
                text-align: center;
                width: 700px;
                height: 600px;
                margin: 10px auto;
                background-image: url('https://images-na.ssl-images-amazon.com/images/I/61QeqaiizVL._AC_SY780_.jpg');
                background-repeat: no-repeat;

            }
    </head>
    <body>
        <div class="container">
            Content
        </div>
    </body>
</html>
```

OUTPUT:

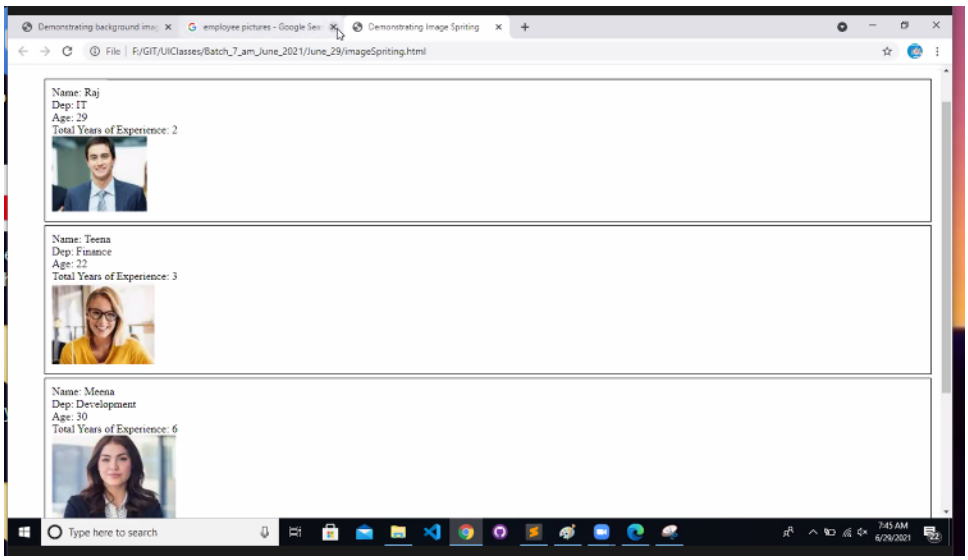


Employee details using image property:

```
<!DOCTYPE html>
<html>
    <head>
        <title>
            Demonstrating Image Spriting
        </title>
        <style type="text/css">
            h3 {
                text-align: center;
                padding: 10px;
            }
            li {
                list-style: none;
                border: 1px solid;
                margin: 5px;
                padding: 10px;
            }
        </style>
    </head>
    <body>
        <h3>Employee Details</h3>
        <ol>
            <li>
                <div>Name: Raj</div>
                <div>Dep: IT</div>
                <div>Age: 29</div>
                <div>Total Years of Experience: 2</div>
                <img src="images/emp1.png" />
            </li>
            <li>
                <div>Name: Teena</div>
```

```
        <body>
            <h3>Employee Details</h3>
            <ol>
                <li>
                    <div>Name: Raj</div>
                    <div>Dep: IT</div>
                    <div>Age: 29</div>
                    <div>Total Years of Experience: 2</div>
                    <img src="images/emp1.png" />
                </li>
                <li>
                    <div>Name: Teena</div>
                    <div>Dep: Finance</div>
                    <div>Age: 22</div>
                    <div>Total Years of Experience: 3</div>
                    <img src="images/emp2.png" />
                </li>
                <li>
                    <div>Name: Meena</div>
                    <div>Dep: Development</div>
                    <div>Age: 30</div>
                    <div>Total Years of Experience: 6</div>
                    <img src="images/emp3.png" />
                </li>
                <li>
                    <div>Name: Krish</div>
                    <div>Dep: Testing/QA</div>
                    <div>Age: 32</div>
                    <div>Total Years of Experience: 7</div>
                    <img src="images/emp4.png" />
                </li>
```

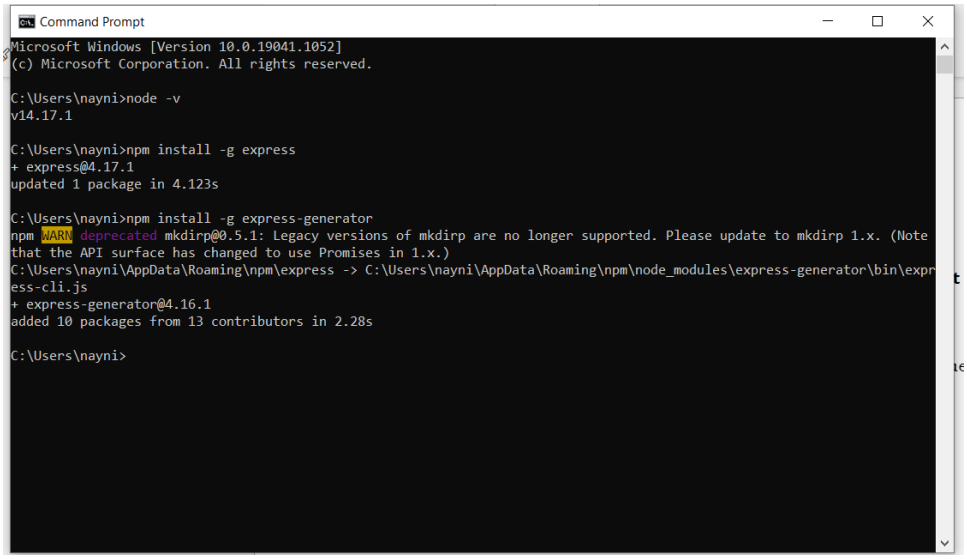
OUTPUT:



To Upload the Image in Database it takes only three steps: -

- Make a HTML form to upload the image
- Connect to the database and store image
- Displaying the Image

https://nodejs.org/en/
Install Nodejs for windows (14.17.1 LTS Mostly recommended).
Open new command prompt and node –v
Create a server:
To do this we need to install a new server npm(node package manager)
cmd: npm install –g express
Then give command: npm install –g express-generator



DATE:30/06/2021

create a server:

Switch to drive where you want to create the server folder.
Step:1 Change folder into Batch code and do the following steps.

- The commands are:

npm install express
npm install express-generator
go the drive/folder where u want to create server - with in command prompt
express server name

e.g.:

express batch_7am....
go inside folder
cd batch_7am....
npm install
open app.js -> add code for making server to listing at given port no.
npm start -> start Ur server
the npm install express

npm install express-generator

---

go the drive/folder where u want to create server - with in commandprmpt

express server name

e.g.:

express batch_7am....

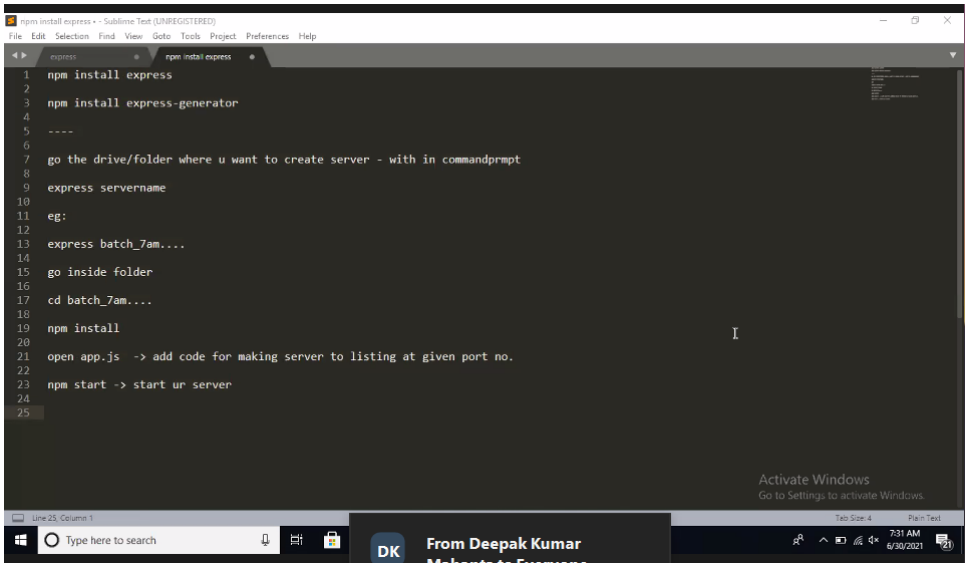
go inside folder

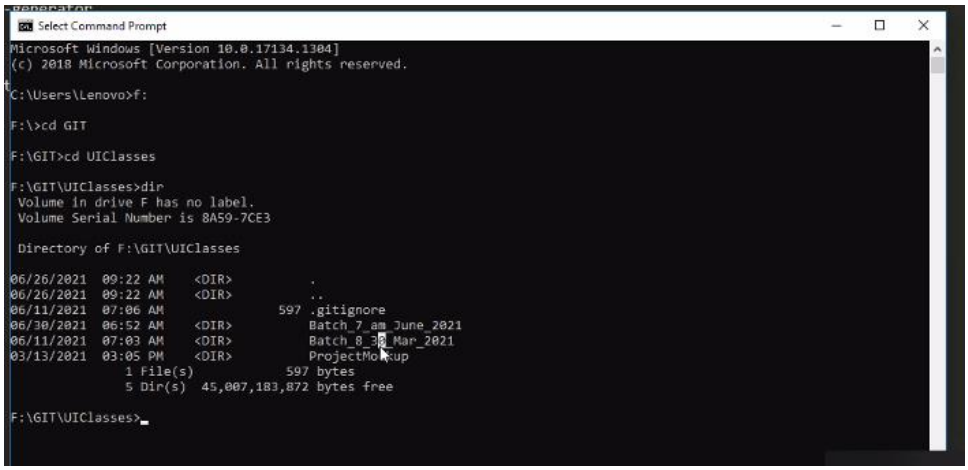
cd batch_7am....

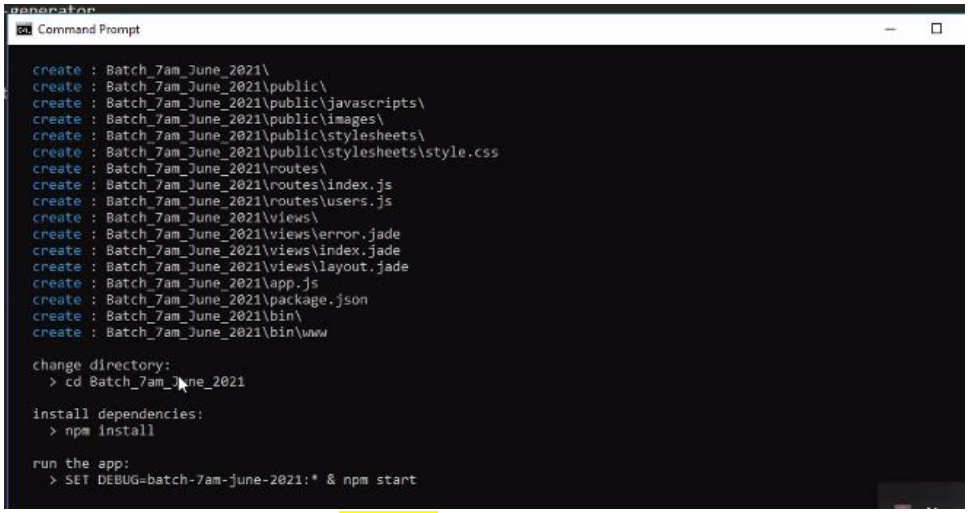
npm install

open app.js -> add code for making server to listing at given port no.

npm start -> start Ur server

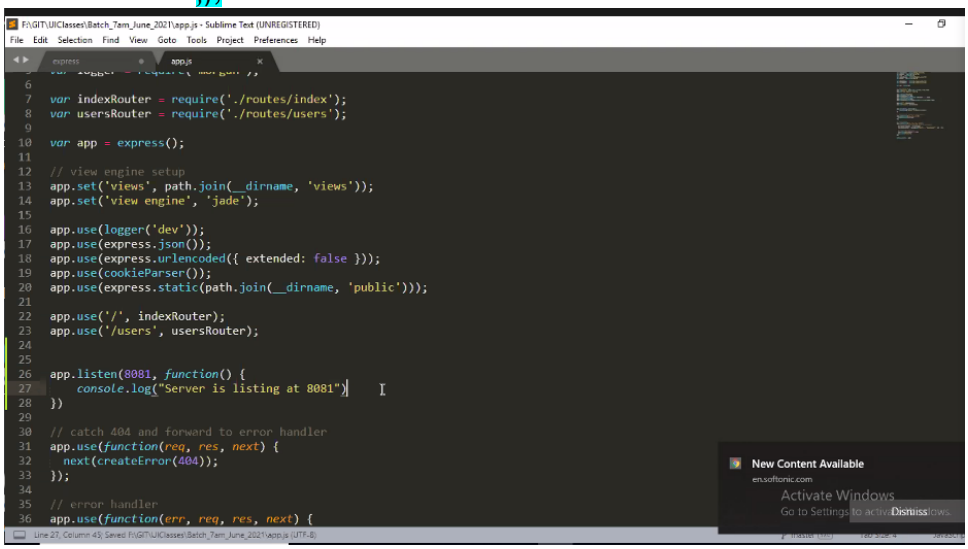




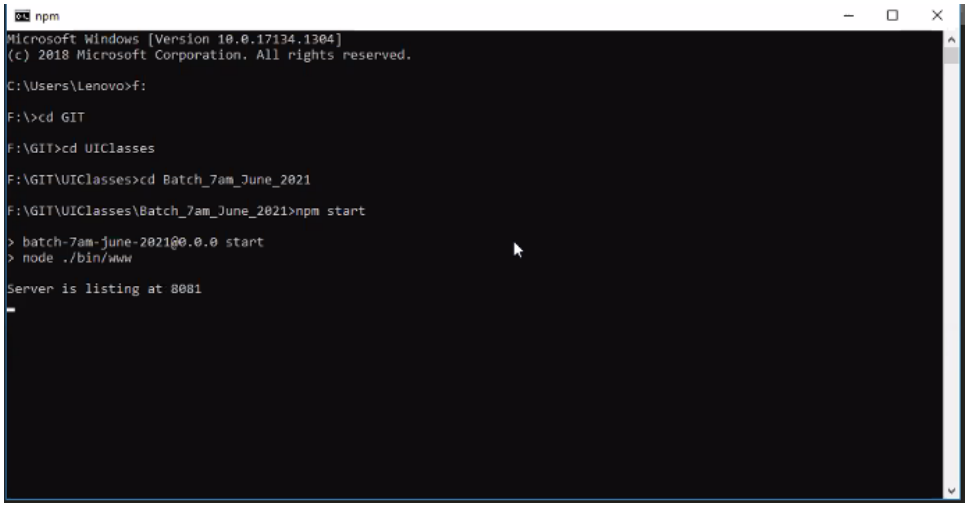


Step:6 Now open **app.js** file in Batch_7am_June_2021 folder which was created by us and at line no 24 give the following command.

```
app.listen(8081, function () {
console.log ("Server is listing at 8081");
});
```



Step:7 So now we need to check the server by giving following command in command prompt or simply open in browser and check using this link http://localhost:8081/

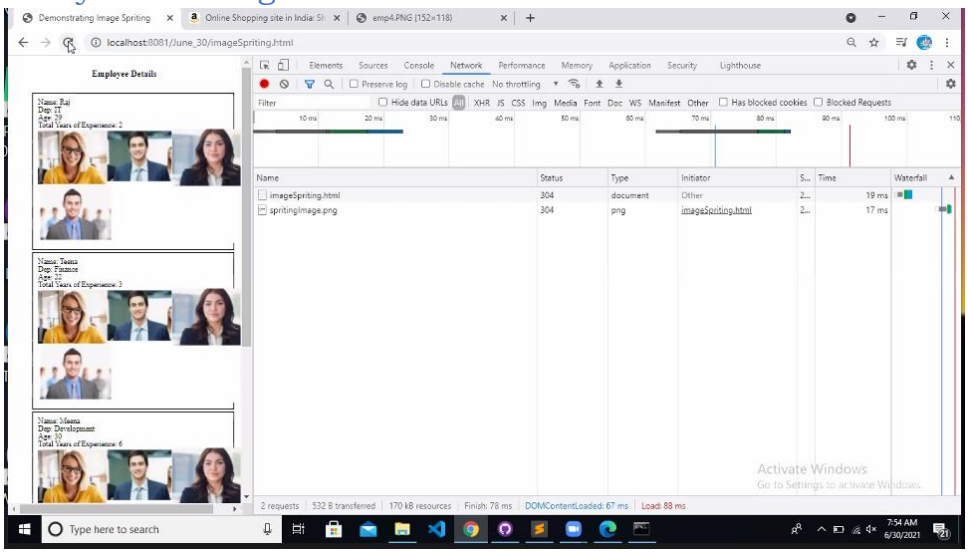


Step:8 Now open public folder and clear the folders available there and simply copy them under public folder.

ImageSpriting

- The concept of merging all the static images to form a single image, through using background position property we show only the required image is called image spiriting.
- Through image spiriting we increase the performance of the page by loading all the static images in a single call

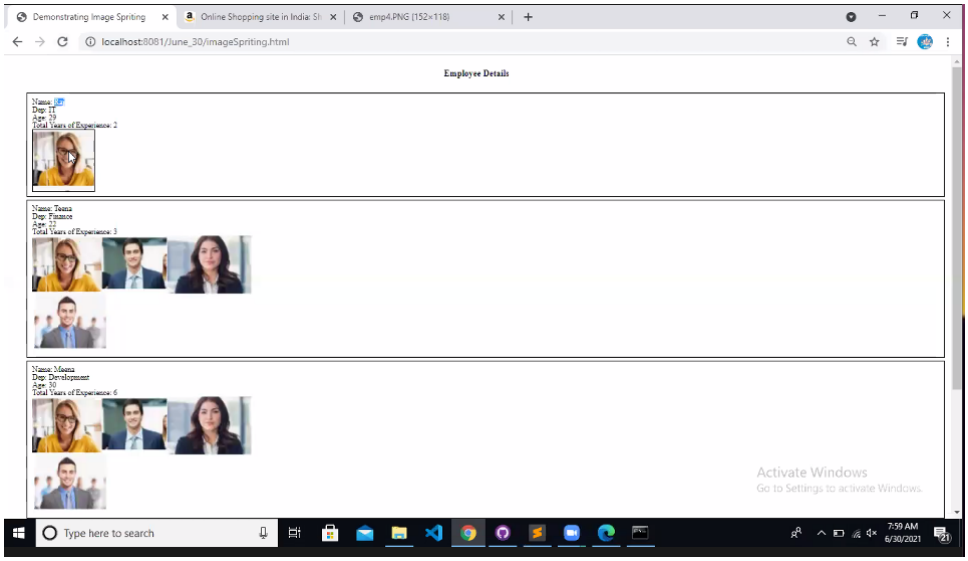
Note: Image spiriting is only recommended for static images but not for dynamic images



TO LOCK AND MOVE THE IMAGES IN BACKGROUND:

```
            li {
                list-style: none;
                border: 1px solid;
                margin: 5px;
                padding: 10px;
            }
            .imgBlock {
                width: 130px;
                height: 130px;
                border: 1px solid;
                background-image: url('images/spritingImage.png');
            }
        </style>
    </head>
    <body>
        <h3>Employee Details</h3>
        <ol>
            <li>
                <div>Name: Raj</div>
                <div>Dep: IT</div>
                <div>Age: 29</div>
                <div>Total Years of Experience: 2</div>
                <div class="imgBlock"></div>
            </li>
            <li>
                <div>Name: Teena</div>
                <div>Dep: Finance</div>
                <div>Age: 22</div>
                <div>Total Years of Experience: 3</div>
                <img src="images/spritingImage.png" />
            </li>
```

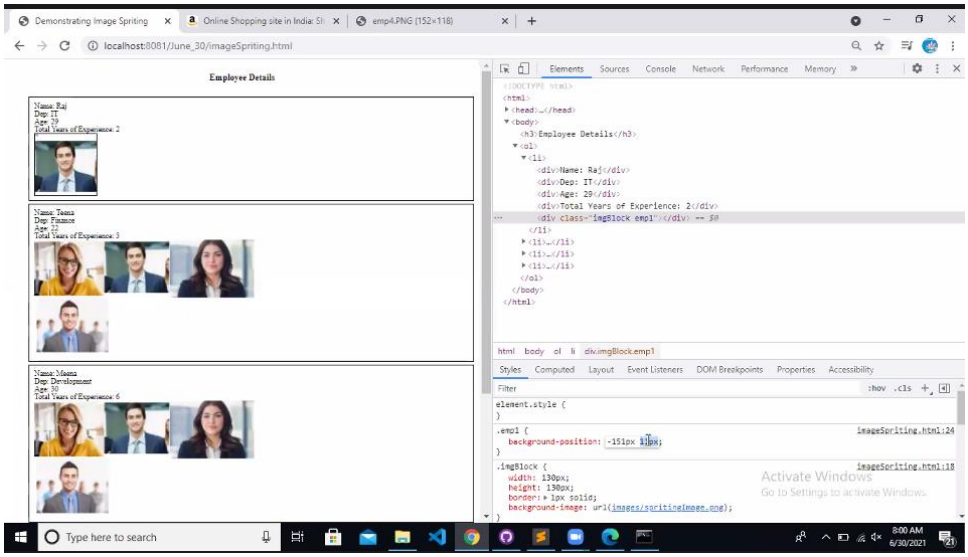
OUTPUT:



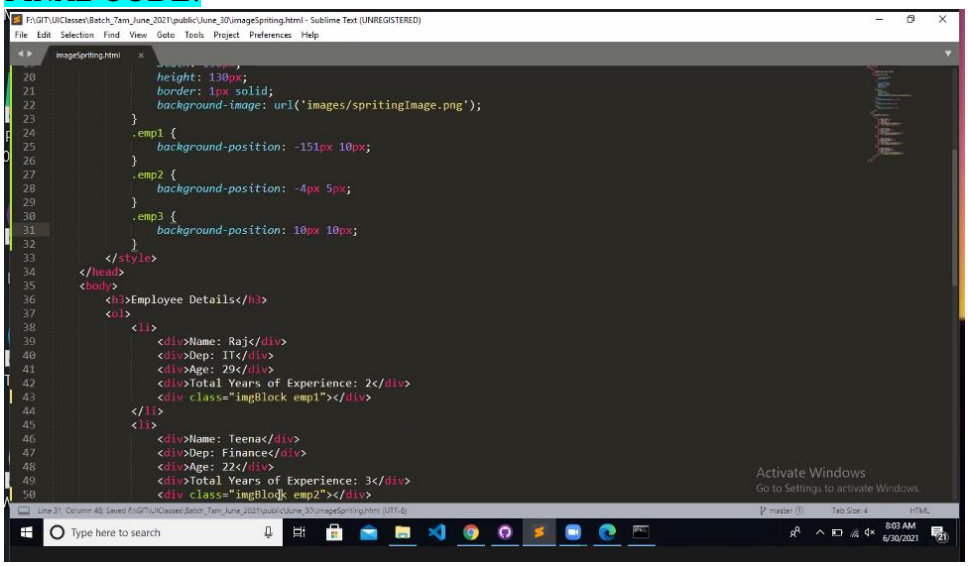
MOVING THE IMAGE:

```
            li {
                list-style: none;
                border: 1px solid;
                margin: 5px;
                padding: 10px;
            }
            .imgBlock {
                width: 130px;
                height: 130px;
                border: 1px solid;
                background-image: url('images/spritingImage.png');
            }
            .emp1 {
                background-position: 10px 10px;
            }
        </style>
    </head>
    <body>
        <h3>Employee Details</h3>
        <ol>
            <li>
                <div>Name: Raj</div>
                <div>Dep: IT</div>
                <div>Age: 29</div>
                <div>Total Years of Experience: 2</div>
                <div class="imgBlock emp1"></div>
            </li>
            <li>
                <div>Name: Teena</div>
                <div>Dep: Finance</div>
                <div>Age: 22</div>
```

OUTPUT:

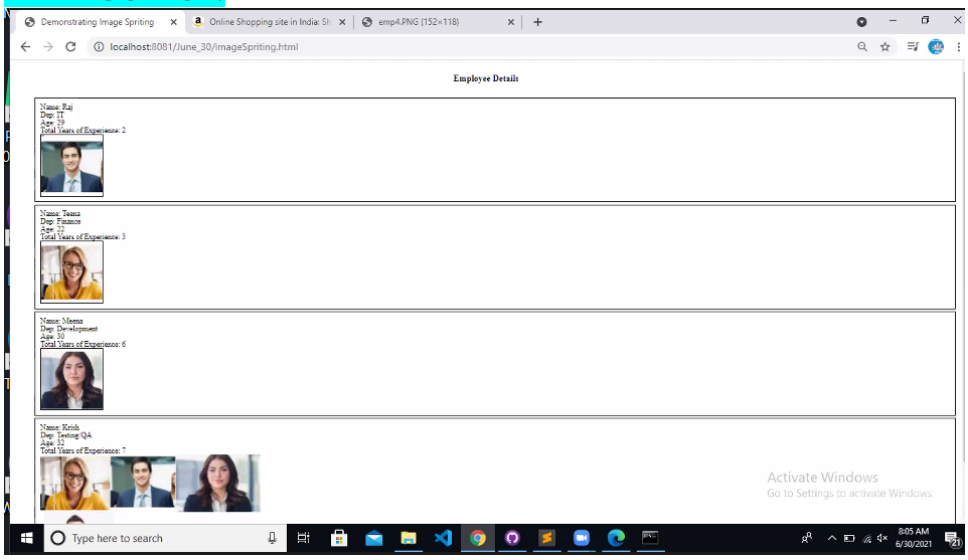


FINAL CODE:



```
                <div>Name: Teena</div>
                <div>Dep: Finance</div>
                <div>Age: 22</div>
                <div>Total Years of Experience: 3</div>
                <div class="imgBlock emp2"></div>
            </li>
            <li>
                <div>Name: Meena</div>
                <div>Dep: Development</div>
                <div>Age: 30</div>
                <div>Total Years of Experience: 6</div>
                <div class="imgBlock emp3"></div>
            </li>
            <li>
                <div>Name: Krish</div>
                <div>Dep: Testing/QA</div>
                <div>Age: 32</div>
                <div>Total Years of Experience: 7</div>
                <img src="images/spritingImage.png" width="150px"  height="150px"/>
            </li>
        </ol>

    </body>
</html>
```

FINAL OUTPUT:



DATE: 1/07/2021

CSS Positions:

Till now in order to move the DOM elements to a specified position we were using padding and margin properties, while using padding or margin properties to move the DOM elements, it is not actually moving the DOM element but it is increasing its dimensions.

In order to actually move the DOM element to a required position without increasing its dimensions we use the following CSS properties:

→ Top
→ Left
→ Right

→ Bottom

- Not every DOM element is capable of considering the above 4 CSS properties but the DOM elements which are positioned can only consider the above properties.
- CSS 'position' is the property through which we could be able to control the position of any DOM element. Following are possible values the position attribute takes:
  - → Static
  - → Relative
  - → Absolute
  - → Fixed
  - → Sticky

❖ Element with position static:

- Every DOM element by default holds the static position which indicates the DOM element cannot be moved to any position from its default (it will not consider the properties top left right and bottom).

★ Syntax:

Position: static;

❖ Element with position Relative:

- Any DOM element with position relative holds the following properties:
  - ➢ It is capable of moving to any required position within the page (it considers top left right and bottom properties).
  - ➢ While moving to a new position it never loses space been occupied on load of the page.
  - ➢ While moving to a new position it always moves relevant to its default position.
  - ★ Syntax:

    position: relative;
  - ★ EXAMPLE:

❖ Element with position Absolute:

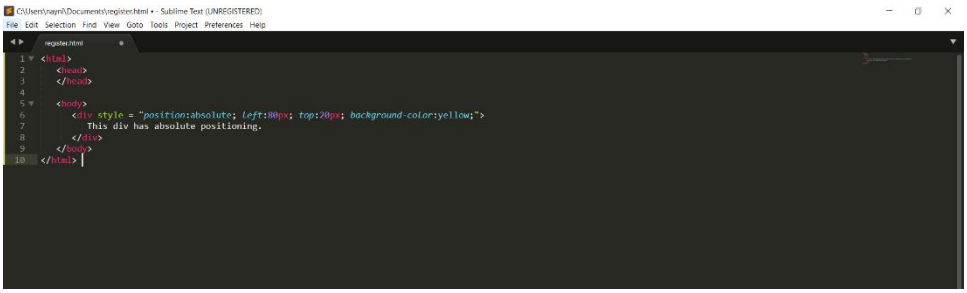
- Any DOM element with position absolute holds the following properties:
    - It is capable of moving to any required position (considers top left right and bottom properties)
    - Element with position absolute will automatically lose the default space it occupies on the page (loses space within x-y axis).
    - While moving to a new position it always moves relevant to its parent's position.
    - While depending on the parent position it only depends on parent whose position value is non static.
    - If its immediate parent doesn't hold the position non-static, it traverses to its grandparents until it finds a parent with position on static.

NOTE: ELEMENT WITH POSITION ABSOLUTE AUTOMATICALLY JUMPS FROM DEFAULT X-Y AXIS TO Z-AXIS.

★ Syntax:

position: absolute;

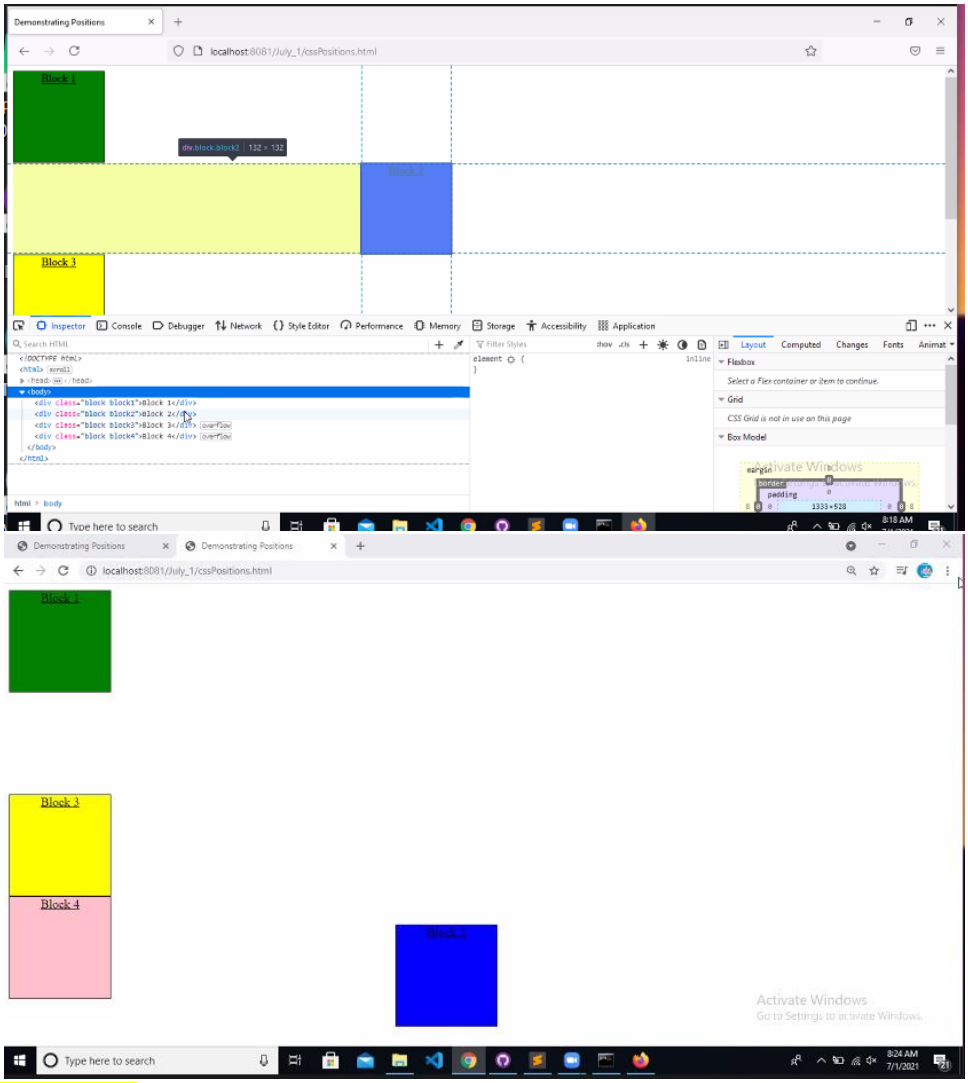
★ EXAMPLE:



TODAYS'EXAMPLE FOR RELATIVE:

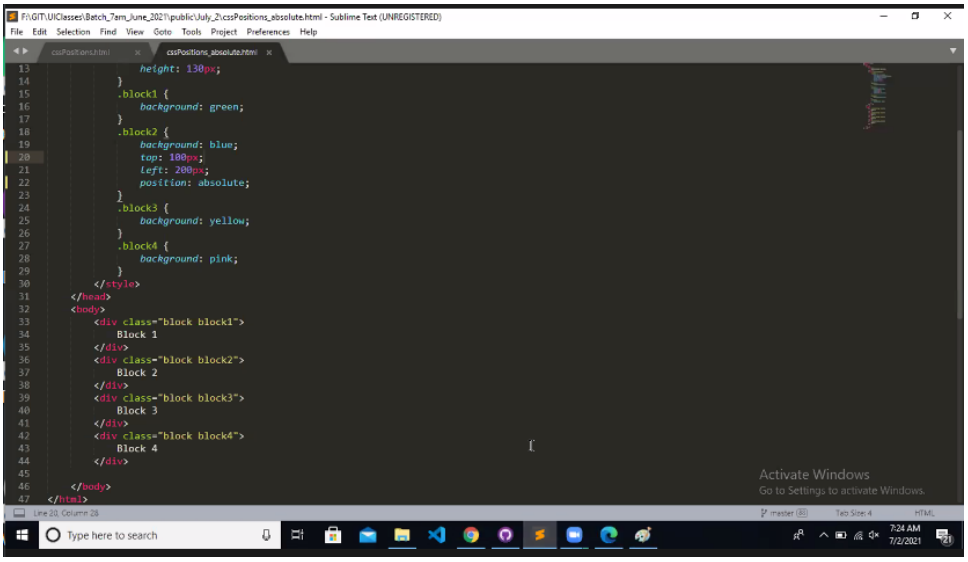
```
            }
            .block1 {
                background: green;
            }
            .block2 {
                background: blue;
                left: 500px;
                top: 300px;
                position: relative;
            }
            .block3 {
                background: yellow;
            }
            .block4 {
                background: pink;
            }
        </style>
    </head>
    <body>
        <div class="block block1">
            Block 1
        </div>
        <div class="block block2">
            Block 2
        </div>
        <div class="block block3">
            Block 3
        </div>
        <div class="block block4">
            Block 4
        </div>
```

OUTPUT:



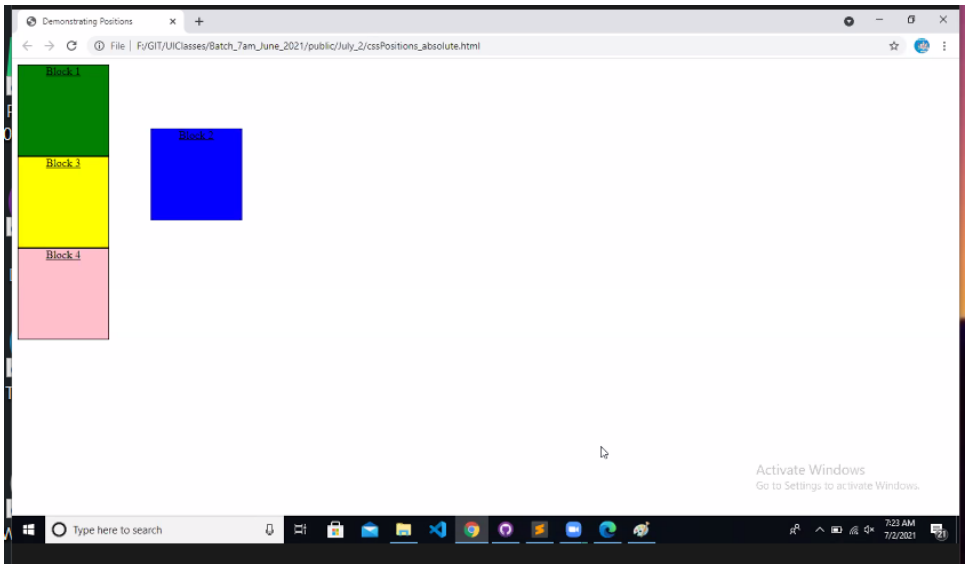
DATE: 2/07/2021

EXAMPLE FOR ABSOLUTE:

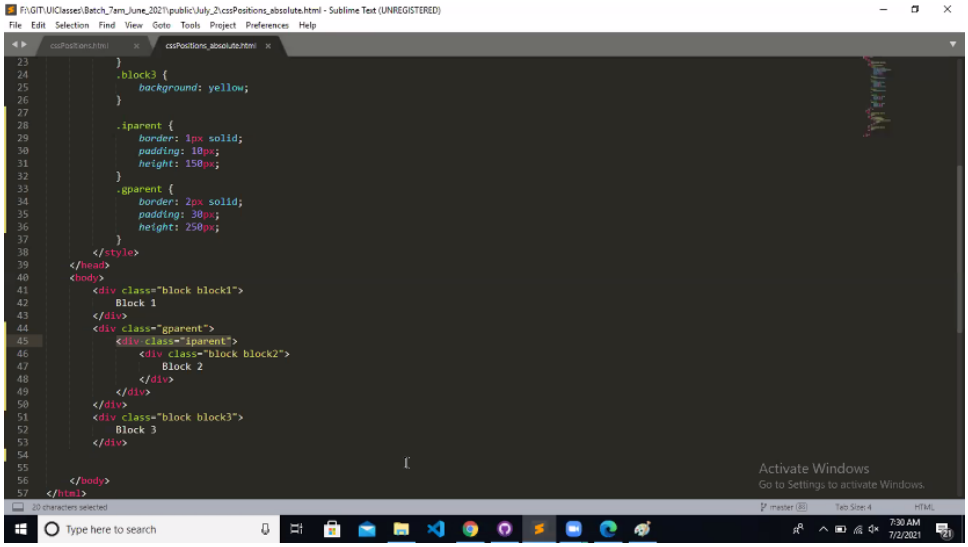


```
            height: 130px;
        }
        .block1 {
            background: green;
        }
        .block2 {
            background: blue;
            top: 100px;
            left: 200px;
            position: absolute;
        }
        .block3 {
            background: yellow;
        }
        .block4 {
            background: pink;
        }
    </style>
</head>
<body>
    <div class="block block1">
        Block 1
    </div>
    <div class="block block2">
        Block 2
    </div>
    <div class="block block3">
        Block 3
    </div>
    <div class="block block4">
        Block 4
    </div>

</body>
</html>
```

OUTPUT:

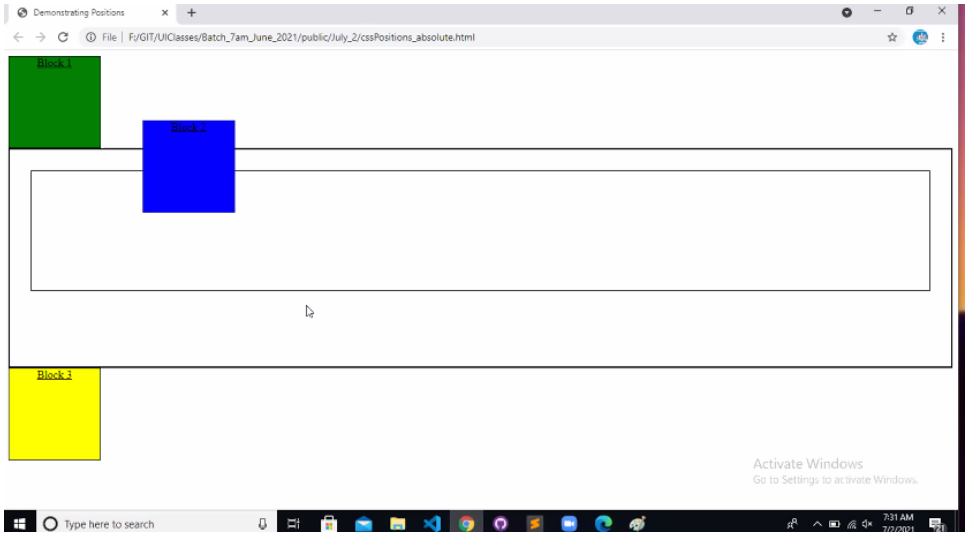


EXAMPLE-2:

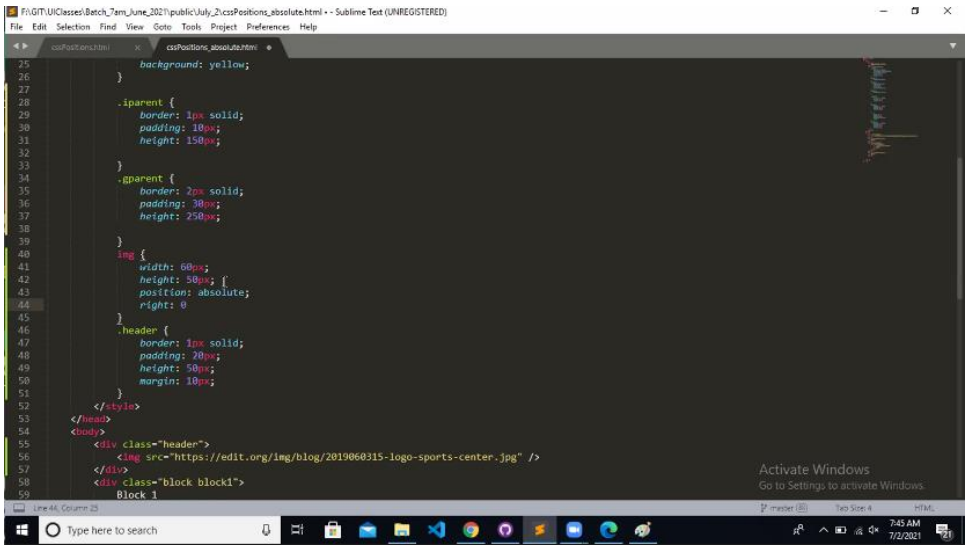


```
        }
        .block3 {
            background: yellow;
        }

        .iparent {
            border: 1px solid;
            padding: 10px;
            height: 150px;
        }
        .gparent {
            border: 2px solid;
            padding: 30px;
            height: 250px;
        }
    </style>
</head>
<body>
    <div class="block block1">
        Block 1
    </div>
    <div class="gparent">
        <div class="iparent">
            <div class="block block2">
                Block 2
            </div>
        </div>
    </div>
    <div class="block block3">
        Block 3
    </div>

</body>
</html>
```

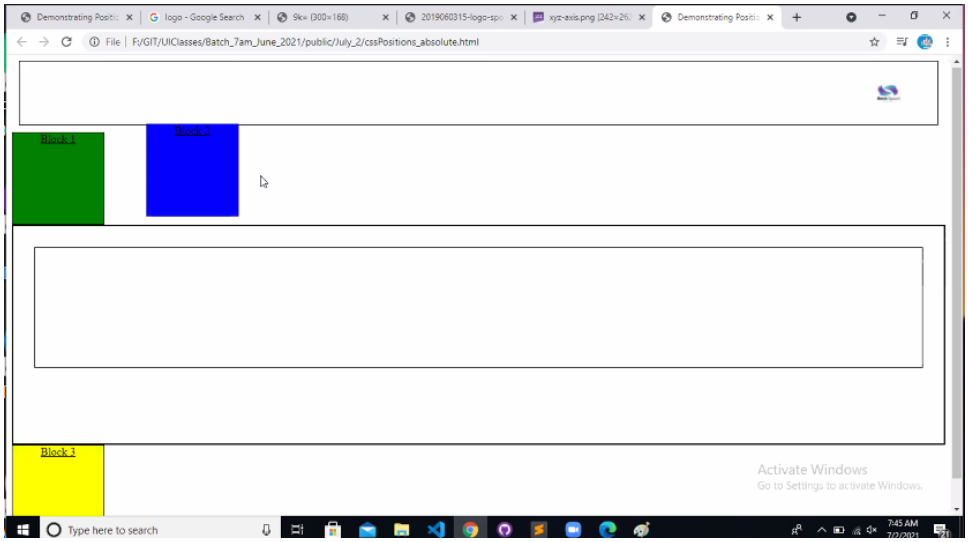
OUTPUT:



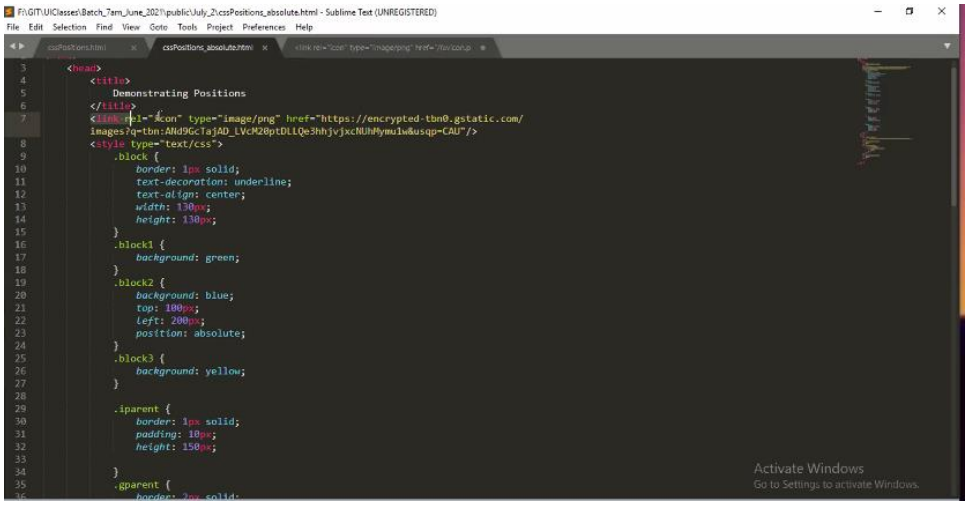
PLACING LOGO IN OUR WEB PAGE:



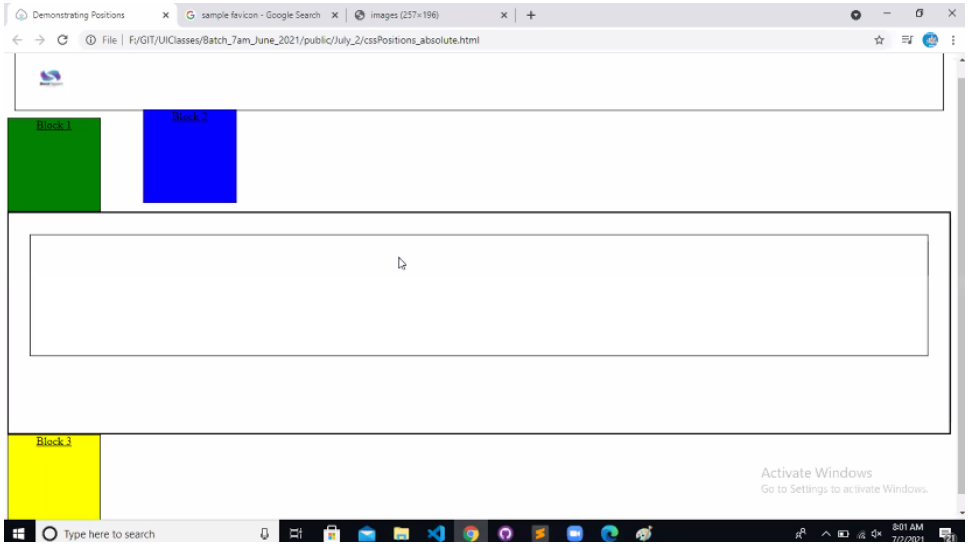
OUTPUT:



PLACING AN ICON IN THE BROWSER:



OUTPUT:

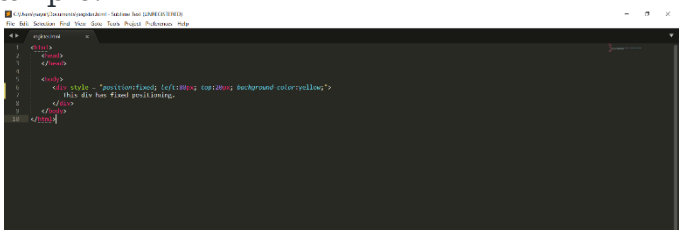


- ❖ Element with position Fixed:
  - Any DOM element with position fixed is almost like an element with position absolute where the only difference is once the element gets its fixed position it doesn't even move from its original position even while scroll.

    - ★ Syntax:

      Position: fixed;

    - ★ Example:



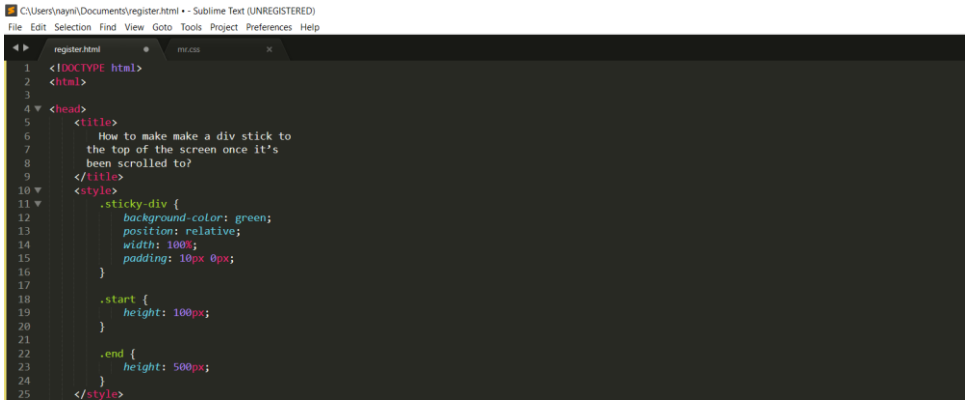
❖ Element with position sticky:

- Any DOM element with position Sticky is almost like element with position relative where the only difference is once positions values been given(top,left,right,bottom) if we tried to scroll the elements out of its view port, it automatically turns to fixed position and doesn't gets scrolled.

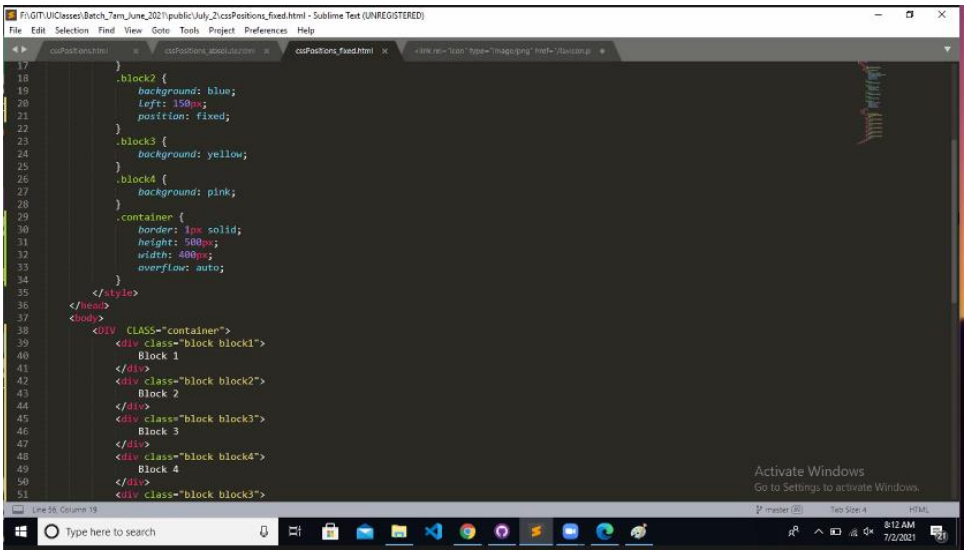
★ Syntax:

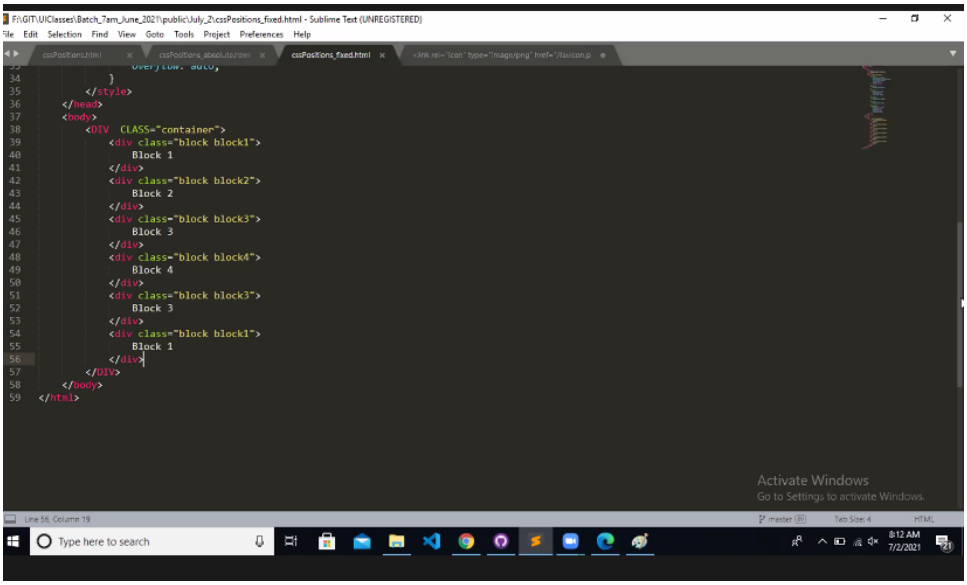
Position: sticky;

★ Example:



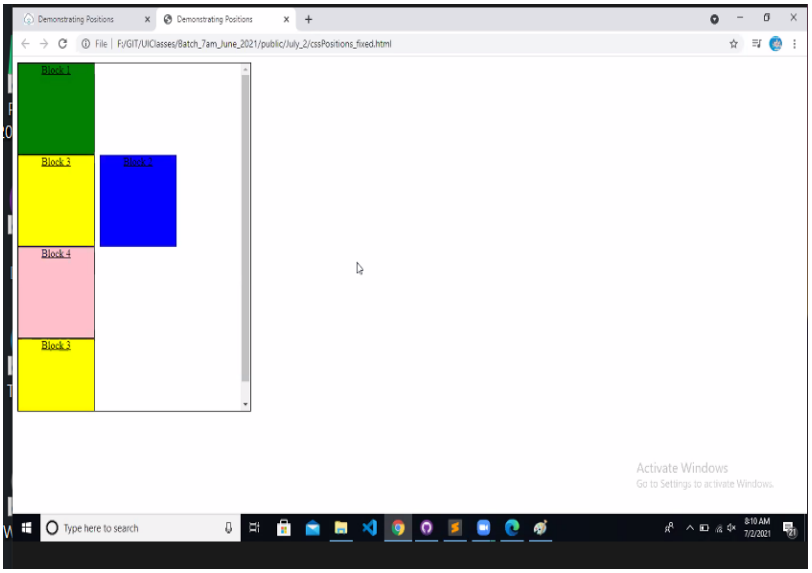
TODAY'S EXAMPLE FOR FIXED:

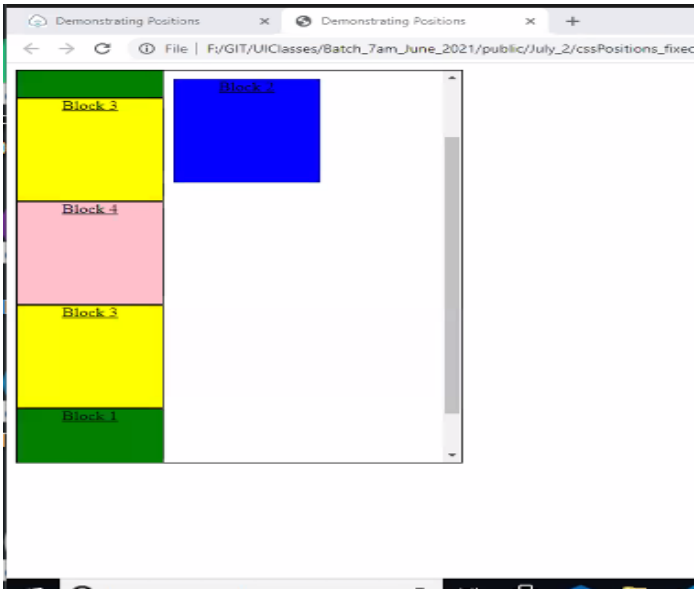




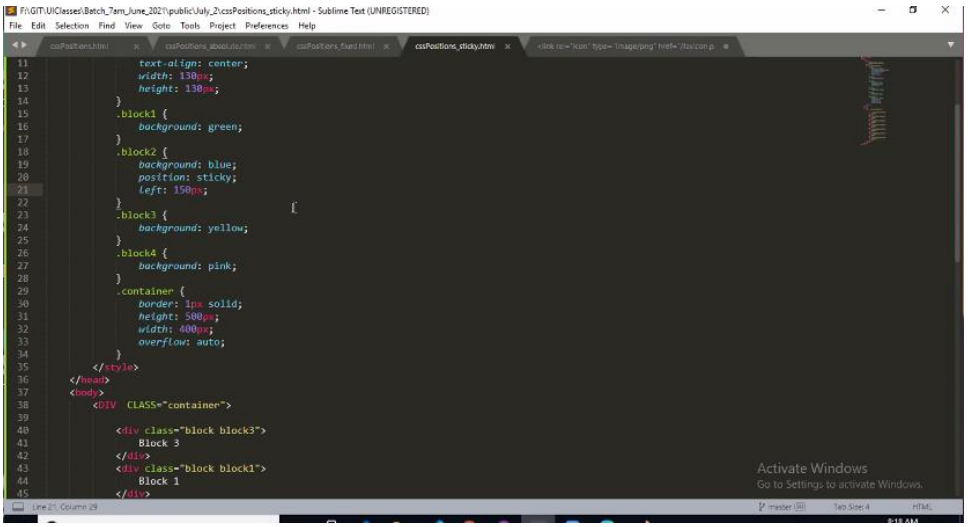
```
                }
            </style>
        </head>
        <body>
            <DIV  CLASS="container">
                <div class="block block1">
                    Block 1
                </div>
                <div class="block block2">
                    Block 2
                </div>
                <div class="block block3">
                    Block 3
                </div>
                <div class="block block4">
                    Block 4
                </div>
                <div class="block block3">
                    Block 3
                </div>
                <div class="block block1">
                    Block 1
                </div>
            </DIV>
        </body>
    </html>
```

OUTPUT:

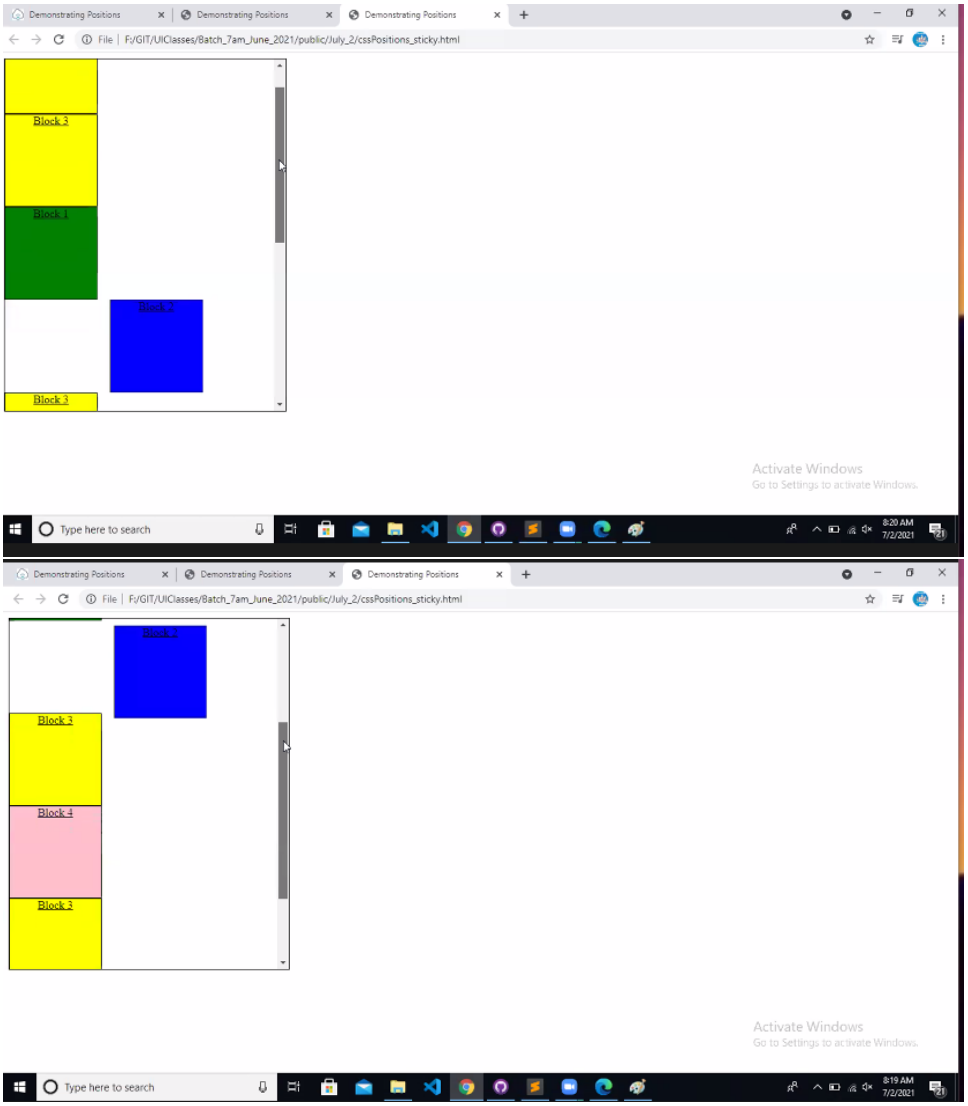




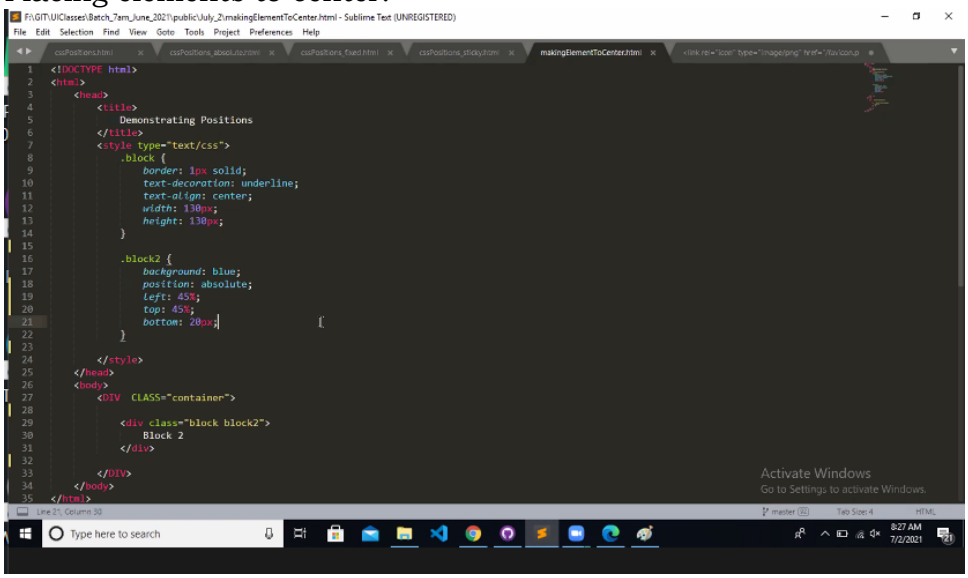
TODAY'S EXAMPLE FOR STICKY:



OUTPUT:



Placing elements to center:



DATE: 3/7/2021

CSS Z-Index Property:

The elements which are falling under z-axis, there is a chance for multiple elements override each other while rendering on the page.

While the elements are overriding, we can control the rendering order through CSS Z-Index property.

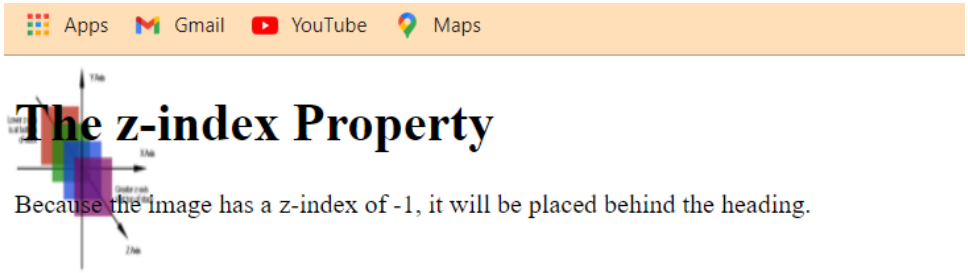
Z-Index a CSS property takes a positive number as a value, through which we can specify the priority order, the element with higher priority order always renders on the top.

NOTE: Z-Index property can only be applied to elements which fall under Z-axis (Elements with position non-static).

EXAMPLE:

```
<head>
<style>
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
}
</style>
</head>
<body>
<h1>The z-index Property</h1>
<imgsrc="http://www.vanseodesign.com/blog/wp-content/uploads/2009/11/z-index.png"
width="100" height="140">
<p>Because the image has a z-index of -1, it will be placed behind the heading. </p>
</body>
</html>
```

OUTPUT:



CSS 'FLOAT' PROPERTY:

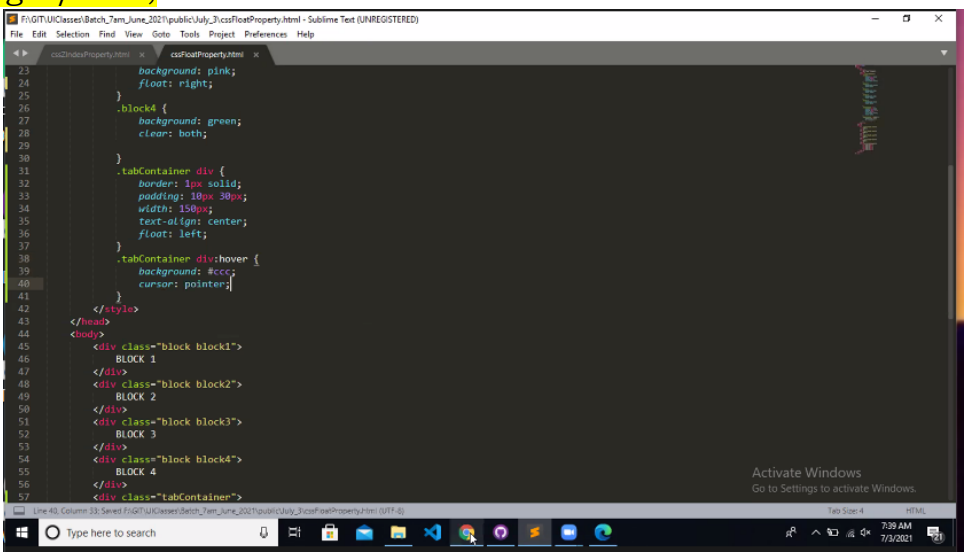
By default, while the elements getting rendered on the page, they try to render from the left top corner/position of the container, the block level elements fall under the new line, inline elements get rendered within the same line. Using CSS 'float' property we can make the elements to get rendered either to the right side or left side of the container.

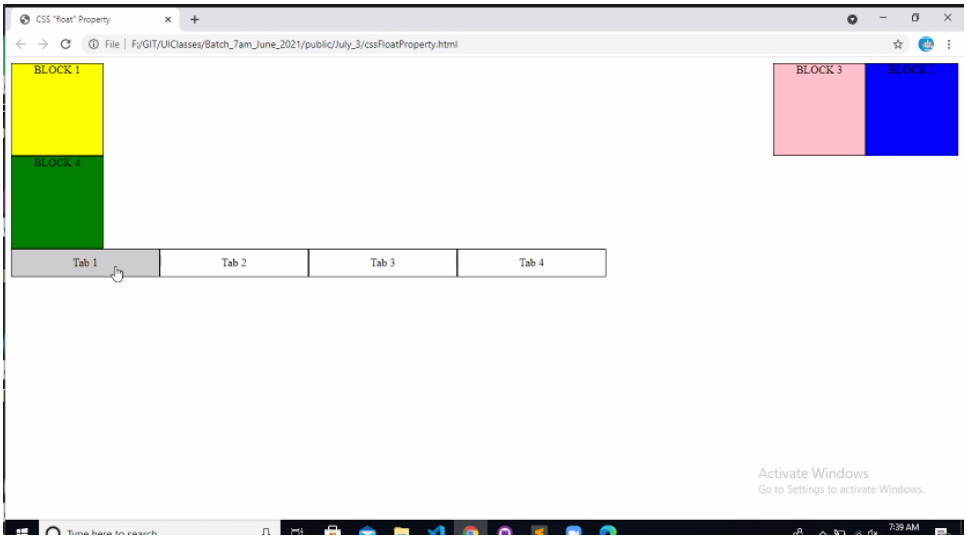
Following are the possible values a float property takes:

float: left/right;

1. Element with float left property will render to complete left side of the container, element with float right property will render to complete right side of the container.
2. Multiple continuous DOM elements applied with CSS float property will always tries to render within the same line irrelevant of block or inline type of elements.
3. If an element changes its default direction through float property, the element which is following it will also try to render in the same line.
4. To make the element not to follow previous element direction we make use of CSS property: "clear".
    i. Syntax:

Clear: left/right/both;





HTML TABLES:

following are predefined tables supported in html using which we could be able to render any data in the form of row and column wise.

- table - > Holds the complete table content
- thead -> To hold table header
- tbody -> holds the body of table
- th -> table header cell
- tr -> hold row of table
- td-> holds table data cell
- tfoot -> Holds footer of table

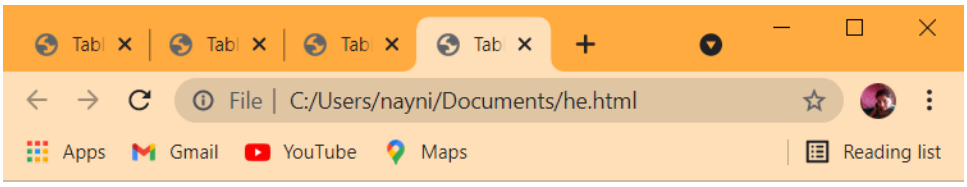
```
<table>
    <thead>
        <tr>
            <th>heading1</th>
            <th>heading 2</th>
            <th>....</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>..</td>
            <td>...</td>
        </tr>
        <tr>
            ...
        </tr>
        ...
    </tbody>
    <tfoot>
        <tr>
            <td>...</td>
        </tr>
        <tr>...</tr>
    </tfoot>
</table>
```

EXAMPLE-1:

```
<!DOCTYPE html>
        <html>
            <head>
              <title>Tables</title>
              <table border="1">
```

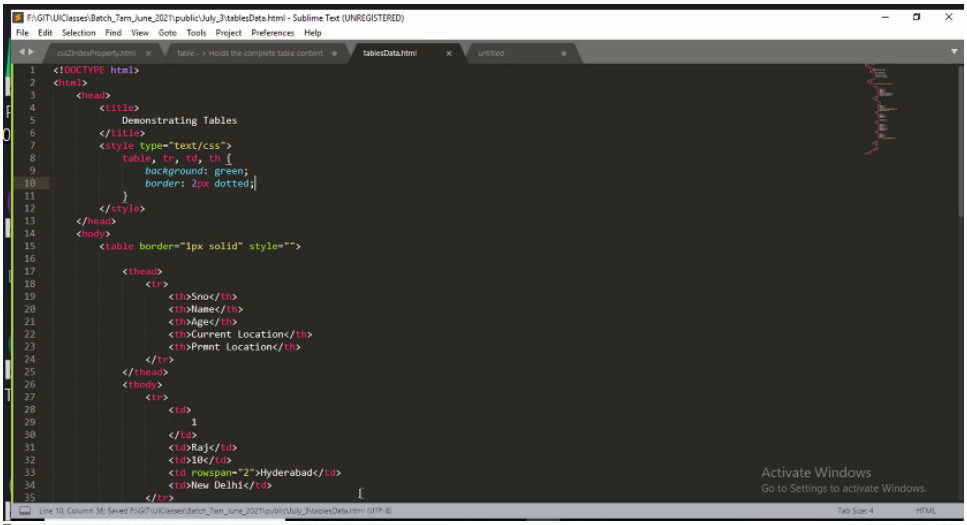
```
            <tr>
              <th>Name</th>
              <td>Meghana</td>
              <td>Ravi</td>
              <td>Sreenu</td>
            </tr>
            <tr>
              <th>Height</th>
              <td>5'4"</td>
              <td>6'5"</td>
              <td>5'1"</td>
            </tr>
            <tr>
              <th>Age</th>
              <td>22</td>
              <td>25</td>
              <td>19</td>
            </tr>
            <tr>
              <th>Place</th>
              <td>Singapore</td>
              <td>Vizag</td>
              <td>rajasthan</td>
            </tr>
          </table>
    </head>
    </html>
```

OUTPUT:

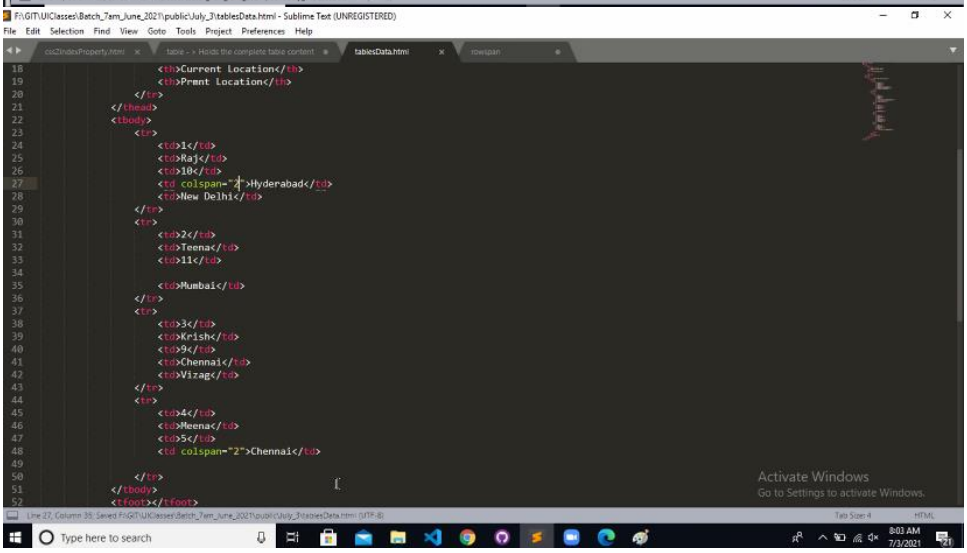


Name	Meghana	Ravi	Sreenu
Height	5'4"	6'5"	5'1"
Age	22	25	19
Place	Singapore	Vizag	rajasthan

EXAMPLE-2:



```
<!DOCTYPE html>
<html>
    <head>
        <title>
            Demonstrating Tables
        </title>
        <style type="text/css">
            table, tr, td, th {
                background: green;
                border: 2px dotted;
            }
        </style>
    </head>
    <body>
        <table border="1px solid" style="">

            <thead>
                <tr>
                    <th>Sno</th>
                    <th>Name</th>
                    <th>Age</th>
                    <th>Current Location</th>
                    <th>Prmnt Location</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>
                        1
                    </td>
                    <td>Raj</td>
                    <td>10</td>
                    <td rowspan="2">Hyderabad</td>
                    <td>New Delhi</td>
                </tr>
```



```
                    <th>Current Location</th>
                    <th>Prmnt Location</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td>1</td>
                    <td>Raj</td>
                    <td>10</td>
                    <td colspan="2">Hyderabad</td>
                    <td>New Delhi</td>
                </tr>
                <tr>
                    <td>2</td>
                    <td>Teena</td>
                    <td>11</td>

                    <td>Mumbai</td>
                </tr>
                <tr>
                    <td>3</td>
                    <td>Krish</td>
                    <td>9</td>
                    <td>Chennai</td>
                    <td>Vizag</td>
                </tr>
                <tr>
                    <td>4</td>
                    <td>Meena</td>
                    <td>5</td>
                    <td colspan="2">Chennai</td>

                </tr>
            </tbody>
            <tfoot></tfoot>
```

OUTPUT:

Sno	Name	Age	Current Location	Prmnt Location
1	Raj	10	Hyderabad	New Delhi
2	Teena	11		Mumbai
3	Krish	9	Chennai	Vizag
4	Meena	5	Chennai	

DATE: 6/7/2021

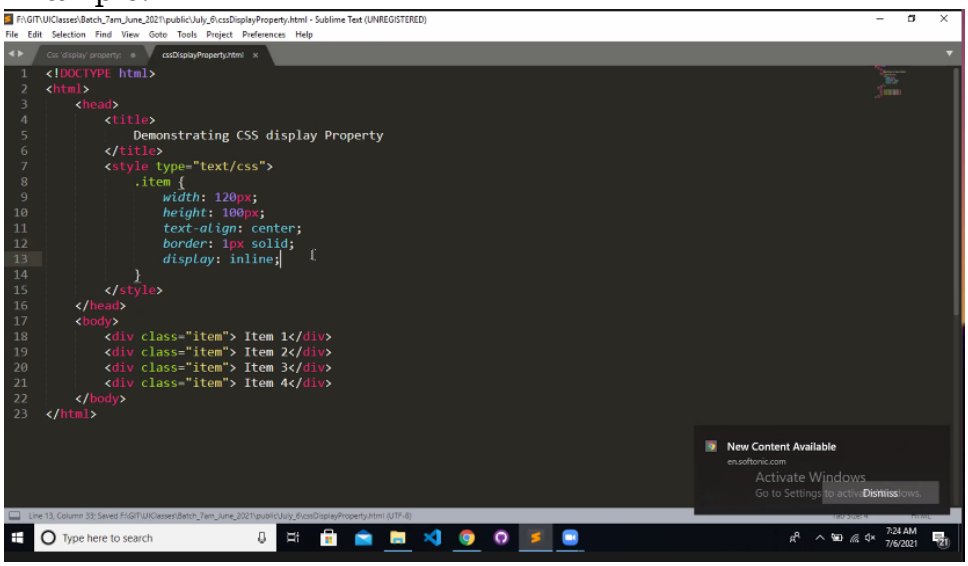
CSS ‘display’ property:

‘display’ is a CSS property through which we could be able to change the default rendering type of any DOM element.

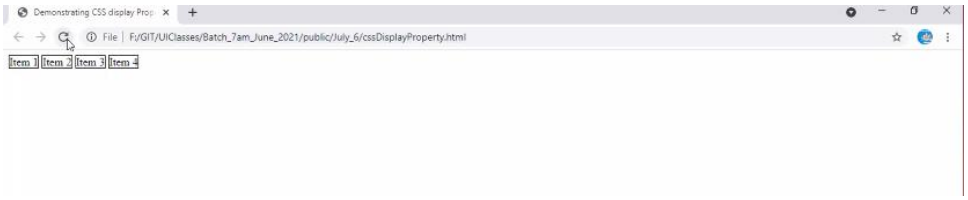
Following are the possible values:

- display: block; -> makes the Dom element to render like a block level element.
- display: inline; -> makes any Dom element to render like an inline element.
- display: inline-block -> Makes any Dom element to render like a block level element, but occupies in same line as like inline element.
- display: none; -> makes element to not be shown on page. (It still exists in Dom structure)
- display: flex -> A css3 property to render flexible items on page.
- Etc…

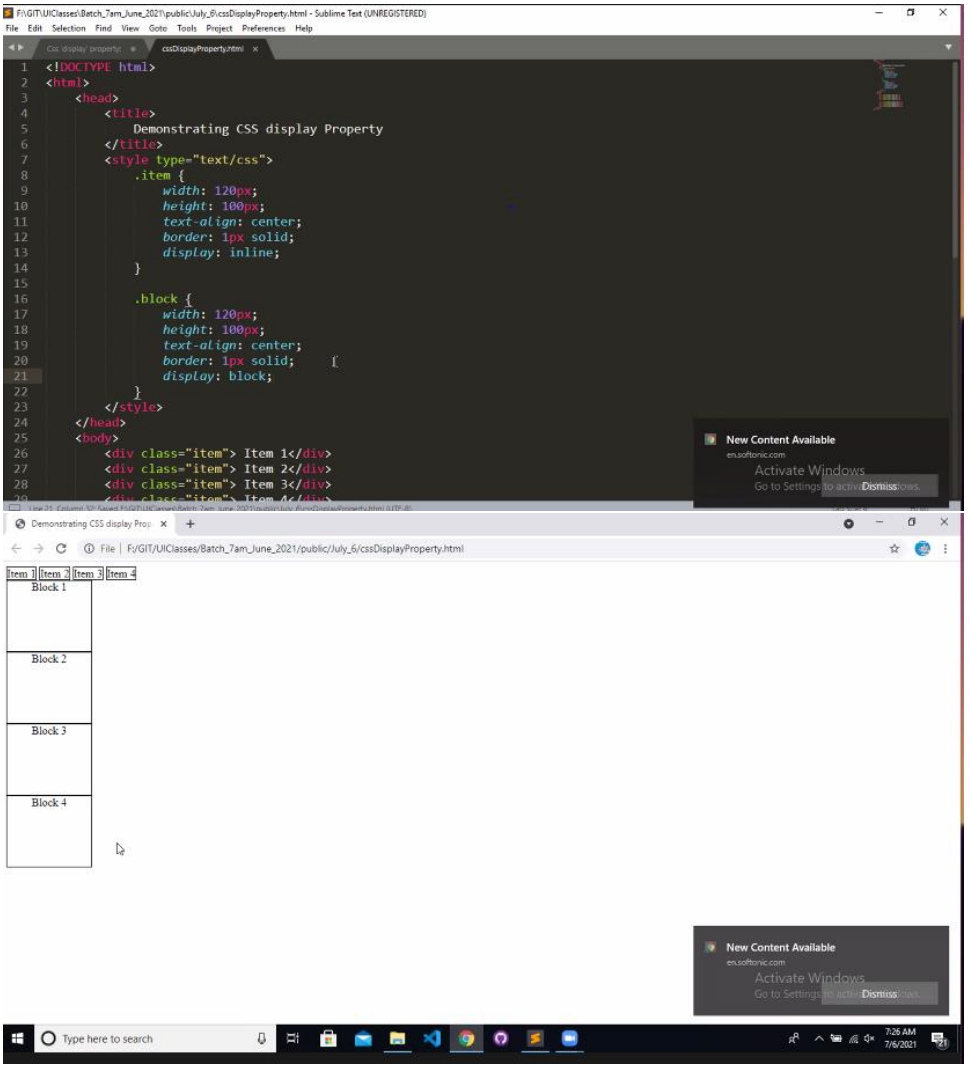
Example:



OUTPUT:



- Display Block:



- Difference between display 'none' and visibility 'hidden' properties:

  Both CSS properties are used to make the DOM element to be not visible on the page where the only difference is display 'none' makes the element to be not visible on the page, it doesn't even occupy its own space.

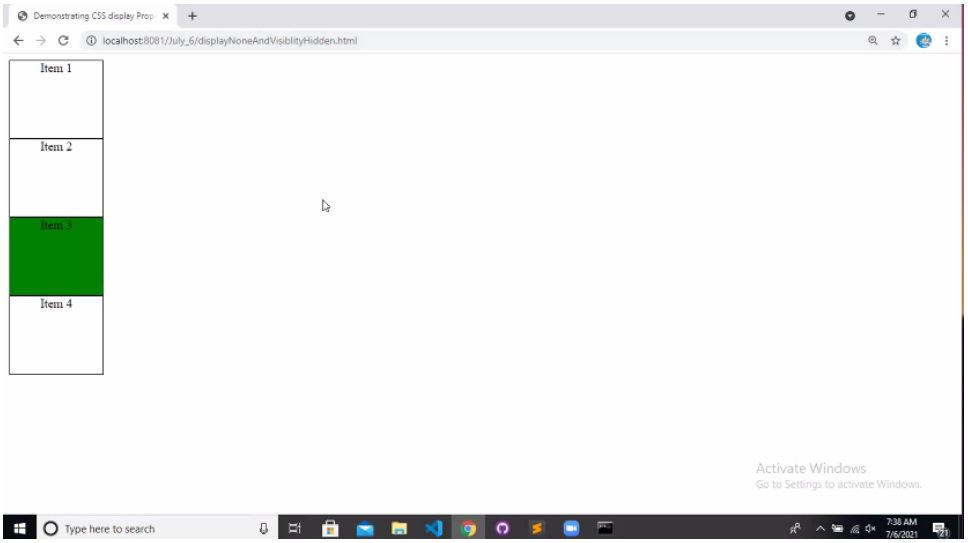
  Visibility 'hidden' also makes the element to be not shown on the page but it still occupies its own space with in the page.

  Note: Both visibility hidden and display 'none' properties makes element to be not visible on the page but the element still exists within the DOM structure.

Example:

```
<!DOCTYPE html>
<html>
    <head>
        <title>
            Demonstrating CSS display Property
        </title>
        <style type="text/css">
            .item, .item3 {
                width: 120px;
                height: 100px;
                text-align: center;
                border: 1px solid;
            }

            .item3 {
                background: green;
            }
        </style>
    </head>
    <body>
        <div class="item"> Item 1</div>
        <div class="item"> Item 2</div>
        <div class="item3"> Item 3</div>
        <div class="item"> Item 4</div>
    </body>
</html>
```

OUTPUT:



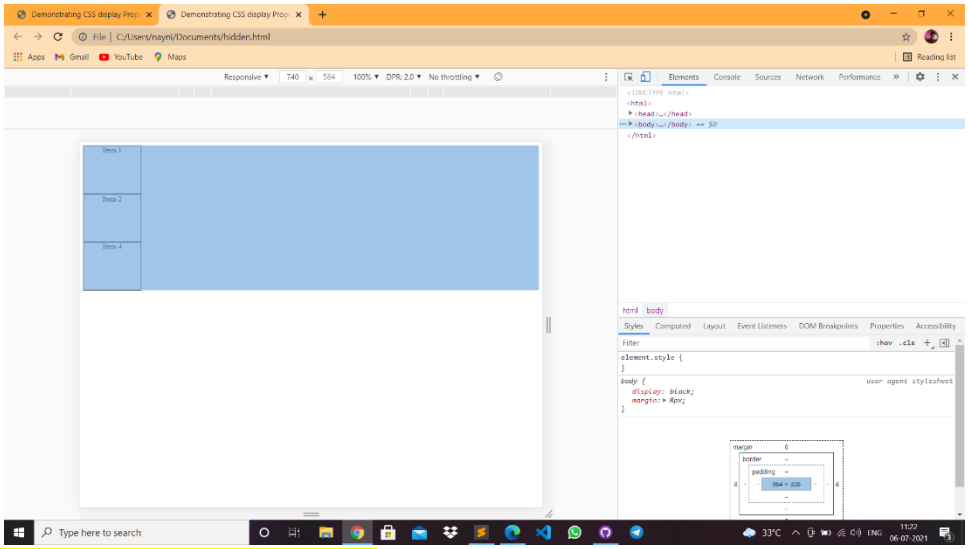
Now if we want to hide item-3 we use following properties:
Display none;
Visibility hidden;

Example for display: none;

```
<!DOCTYPE html>
<html>
<head>
<title>
Demonstrating CSS display Property
</title>
<style type="text/css">
.item,. item3 {
width: 120px;
height: 100px;
```

```
text-align: center;
border: 1px solid;
}

. item3 {
background: green;
display: none;
}

</style>
</head>
<body>
<div class="item"> Item 1</div>
<div class="item"> Item 2</div>
        <div class="item3">Item 3</div>
<div class="item"> Item 4</div>
</body>
</html>
```

OUTPUT:

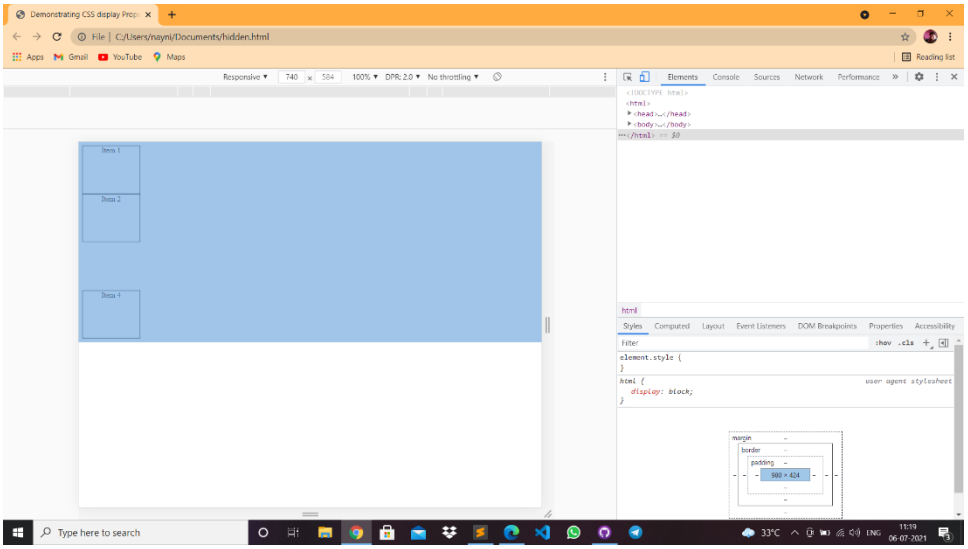


Example for visibility: hidden;

```
<!DOCTYPE html>
<html>
<head>
<title>
Demonstrating CSS display Property
</title>
<style type="text/css">
.item,. item3 {
width: 120px;
```

```
height: 100px;
text-align: center;
border: 1px solid;
}

. item3 {
background: green;
/*display: none; */
visibility: hidden;
}
</style>
</head>
<body>
<div class="item"> Item 1</div>
<div class="item"> Item 2</div>
<div class="item3"> Item 3</div>
<div class="item"> Item 4</div>
</body>
</html>
```

OUTPUT:



CSS pseudo classes: Following are the predefined pseudo classes been supported using which we could able to apply the CSS on elements not on load of the page but based on current state of the element.

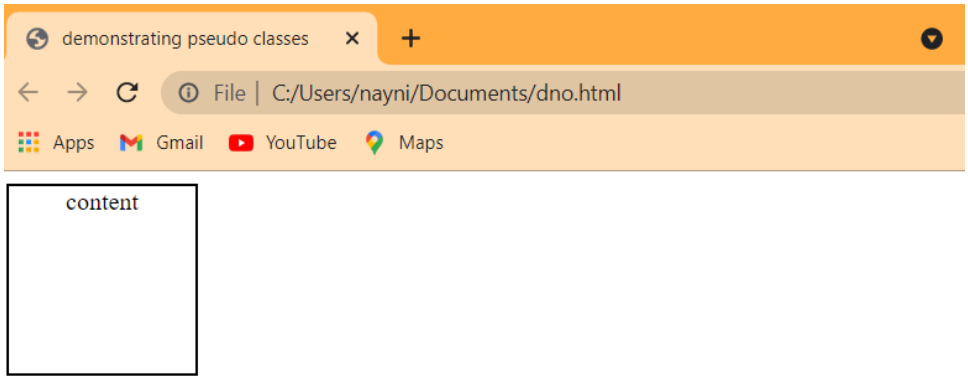
- **: hover -> applies set of css when there is hover**
- **: empty -> Applied to any element which doesn't have child elements**
- **: disabled -> To element with disable state**
- **: enabled -> To element with enable state**

- **: active -> Selects any active link**
- **: checked -> selects checkbox element with checked state**
- **: focus -> Selects element when it's in focus state**
- **: first-child -> element which is in first child state.**
- **: last-child -> element which is in last child state.**
- **div: first-of-type -> to select every div element which is the first div of its parent.**
- **a: link -> selects unvisited links**
- **a: visited -> Selects visited links**
- **nth-child (2) -> selects element in 2nd position**
- **p:only-child - selects p tag which is only child of its element**
- **etc.**

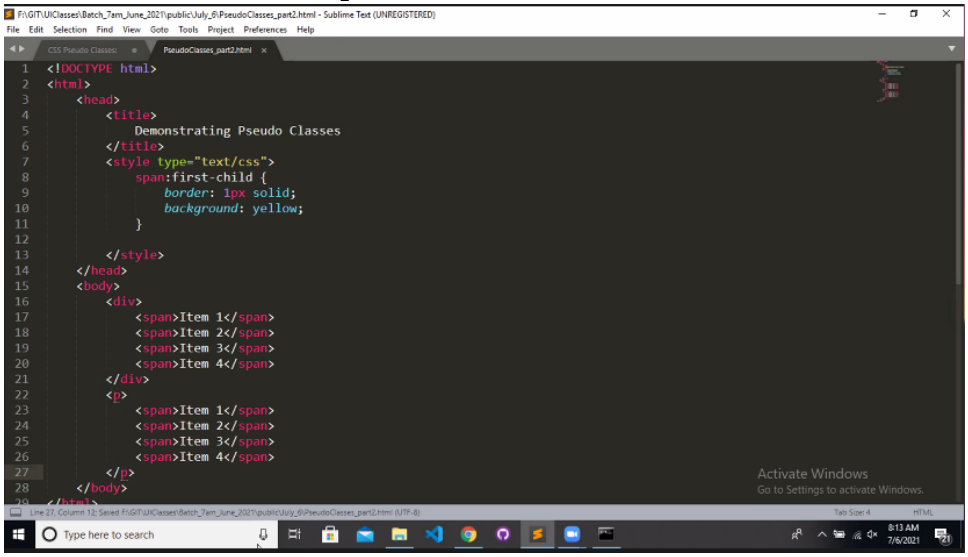
Example:

```
<!DOCTYPE html>
        <html>
        <head>
        <title>
        demonstrating pseudo classes
        </title>
        <style type="text/css">
        .container {
        border: 2px solid;
        width: 120px;
        height: 120px;
        text-align: center;
        }
        .container: hover {
        background: green;
        font-size: 20px
        }

        </style>
        </head>
        <body>
        <div class="container">
        content
        </div>
        </body>
        </html>
```

Output:

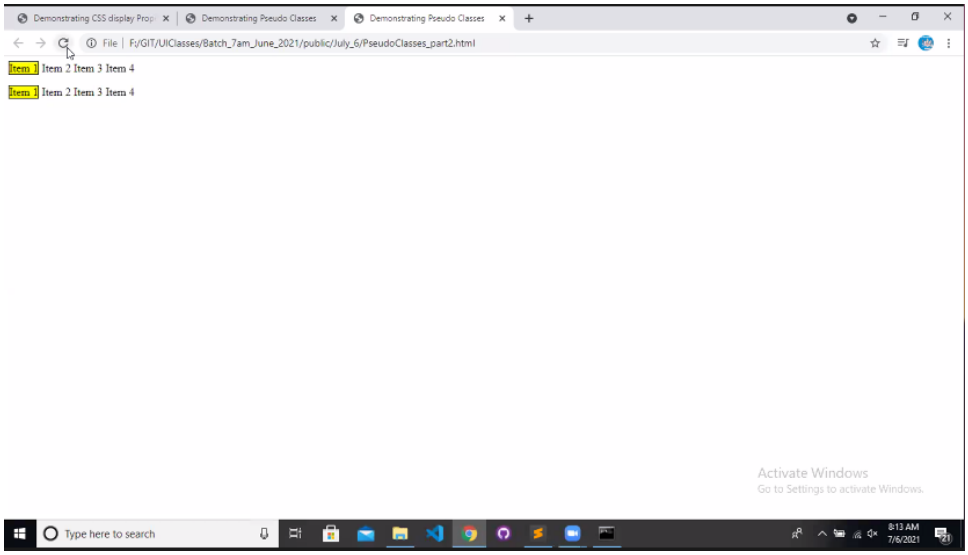


Pseudo Classes example 2:



```
<!DOCTYPE html>
<html>
    <head>
        <title>
            Demonstrating Pseudo Classes
        </title>
        <style type="text/css">
            span:first-child {
                border: 1px solid;
                background: yellow;
            }

        </style>
    </head>
    <body>
        <div>
            <span>Item 1</span>
            <span>Item 2</span>
            <span>Item 3</span>
            <span>Item 4</span>
        </div>
        <p>
            <span>Item 1</span>
            <span>Item 2</span>
            <span>Item 3</span>
            <span>Item 4</span>
        </p>
    </body>
</html>
```

Output:

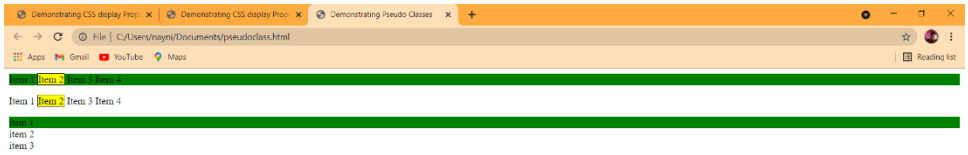


Example 3 (for middle i.e.., nth value)

```
<!DOCTYPE html>
<html>
<head>
<title>
Demonstrating Pseudo Classes
</title>
<style type="text/css">
span: nth-child (2) {
border: 1px solid;
background: yellow;
}

div: first-of-type {
background: green;
}
</style>
</head>
<body>
<div>
<span>Item 1</span>
<span>Item 2</span>
<span>Item 3</span>
<span>Item 4</span>
</div>
<p>
<span>Item 1</span>
<span>Item 2</span>
<span>Item 3</span>
```

```
<span>Item 4</span>
</p>

<div>
<div>item 1</div>
<div>item 2</div>
<div>item 3</div>
</div>
</body>
</html>
```

Output:

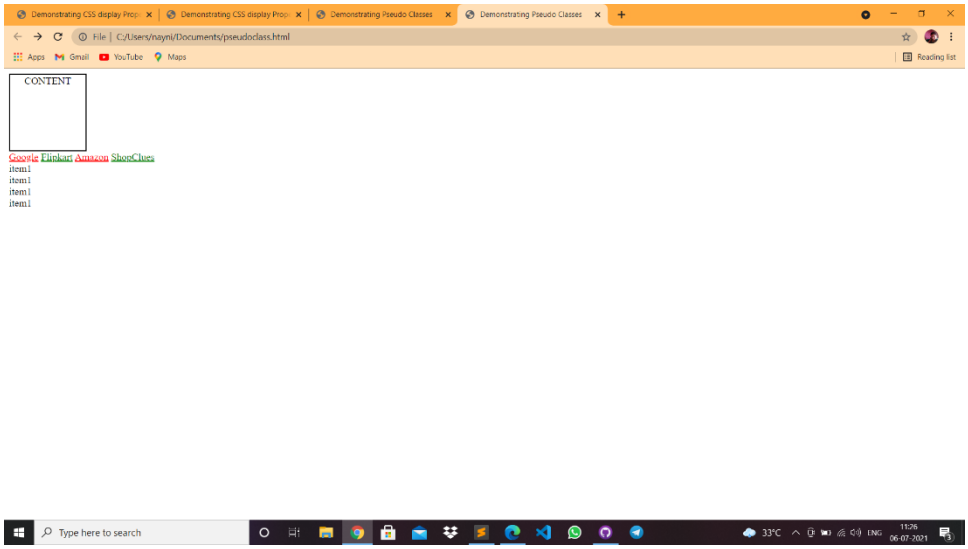


Highlighting the image or links in pseudo-Classes:

```
<!DOCTYPE html>
                    <html>
                    <head>
            <title>
            Demonstrating Pseudo Classes
            </title>
            <style type="text/css">
            .container {
            border: 2px solid;
            width: 120px;
            height: 120px;
            text-align: center;

            }
```

```
.container: hover {
background: green;
font-size: 20px;
}
a: link {
color: green;
}
a: visited {
color: red;
}
a: active {
background: red;
}
a: focus {
background: yellow;
}
div: empty {
border: 2px solid;
}
</style>
</head>
<body>
<div class="container">
CONTENT
</div>
<a href="http:www.google.com">Google</a>
<a href="http:www.flipkart.com">Flipkart</a>
<a href="http:www.amazon.com">Amazon</a>
<a href="http:www.shopclues.com">Shop Clues</a>
<div>
<div>item1</div>
<div>item1</div>
<div>
<span>item1</span>
</div>
<div>item1</div>
</div>
</body>
</html>
```

Output:



DATE: 7/07/2021

Pseudo Elements:

Following are the predefined pseudo elements been supported using which we could able to apply CSS for not to the complete element but partially to the content of the element.

- ::after -> Add any content after selected element.
  - It inserts the content after the content of the element. It is used to add something after the specific part of an element. Generally, it is used with the content property.
  - It also allows us to add regular strings or images after the content.
- ::before -> Add any content before the selected element.
  - It allows us to add something before the element's content. It is used to add something before the specific part of an element. Generally, it is used with the **content** property.
  - We can also add the regular strings or images before the content using this pseudo-element.
- ::first-letter ->Only gets applied to first letter of the container.
  - It affects the first letter of the text. It can be applied only to block-level elements. Instead of supporting all CSS properties
- ::first-line ->Only gets applied to first line in content.
  - It is similar to the ::first-**letter** pseudo-element, but it affects the entire line. It adds the special effects to the first line of the text.
- ::marker -> to set CSS for makers of list items.

- ::selection -> Selects the part/portion of element content which is selected.
    - It is used to style the part of an element that is selected by the user. We can use the following CSS properties with it:
    - **color.**
    - **background-color.**
    - Other properties include **cursor, outline,** etc.

Sample Definition: CSS pseudo-elements are used to add special effects to some selectors. You do not need to use JavaScript or any other script to use those effects. A simple syntax of pseudo-element is as follows – selector: pseudo-element {property: value}.

Example:

```
<!DOCTYPE html>
<html>
<head>
<title>
Demonstrating Pseudo Elements
</title>
<style type="text/css">
p::first-letter {
font-size: 25px;
color: green;
font-weight: bold;
}
p::first-line {
background: yellow;
}
p::selection {
background: red;
color: green;
}
p::after {
content: 'Continue reading from main page';
color: blue;
text-decoration: underline;
}
.test::marker {
color: blue;
```

```
font-size: 20px;
}
</style>
</head>
<body>
<ul>
<li class="test">
item 1
</li>
<li>
item 2
</li>
<li>
item 3
</li>
</ul>
<h3>
News Item 1:
</h3>
<p>
```

In a historic move, a new 'Ministry of Cooperation' has been created by the Narendra Modi Government to realise the vision of 'Shakar se Samriddhi' (prosperity from cooperation).

Government sources told CNN-News18 that the ministry would provide a separate administrative, legal and policy framework to strengthen the cooperative movement in the country. It will also help deepen cooperatives as a true people-based movement reaching upto the grassroots.

The creation of a new ministry comes a day ahead of the much-awaited cabinet reshuffle.

"In our country, a co-operative based economic development model is very relevant where each member works with a spirit of responsibility," a source said.

```
</p>
<h3>
News Item 1:
</h3>
<p>
```

In a historic move, a new 'Ministry of Cooperation' has been created by the Narendra Modi Government to realise the vision of 'Shakar se Samriddhi' (prosperity from cooperation).

Government sources told CNN-News18 that the ministry would provide a separate administrative, legal and policy framework to strengthen the cooperative movement in the country. It will also help deepen cooperatives as a true people-based movement reaching up to the grassroots.

The creation of a new ministry comes a day ahead of the much-awaited cabinet reshuffle.

"In our country, a co-operative based economic development model is very relevant where each member works with a spirit of responsibility," a source said.

<h3>
News Item 1:
</h3>

In a historic move, a new 'Ministry of Cooperation' has been created by the Narendra Modi Government to realise the vision of 'Shakar se Samriddhi' (prosperity from cooperation).

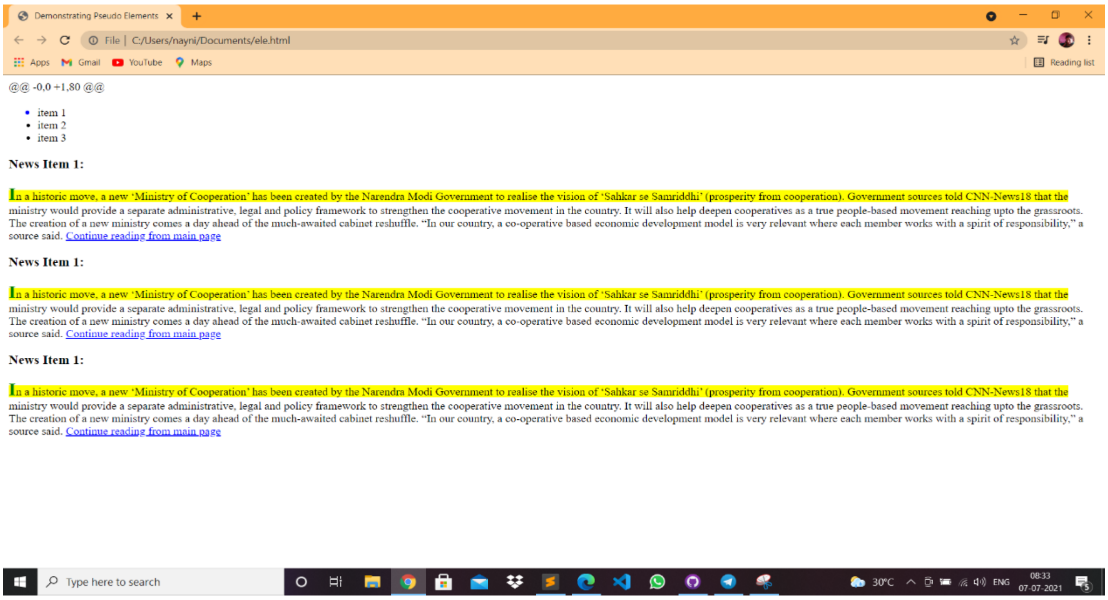
Government sources told CNN-News18 that the ministry would provide a separate administrative, legal and policy framework to strengthen the cooperative movement in the country. It will also help deepen cooperatives as a true people-based movement reaching up to the grassroots.

The creation of a new ministry comes a day ahead of the much-awaited cabinet reshuffle.

"In our country, a co-operative based economic development model is very relevant where each member works with a spirit of responsibility," a source said.

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

Output:



DATE:8/7/2021

- CSS 'Opacity' property: While elements get override each other we could able to control the transparency level of elements through CSS 'opacity' property.
- It takes a value between 0-1.
  Example:

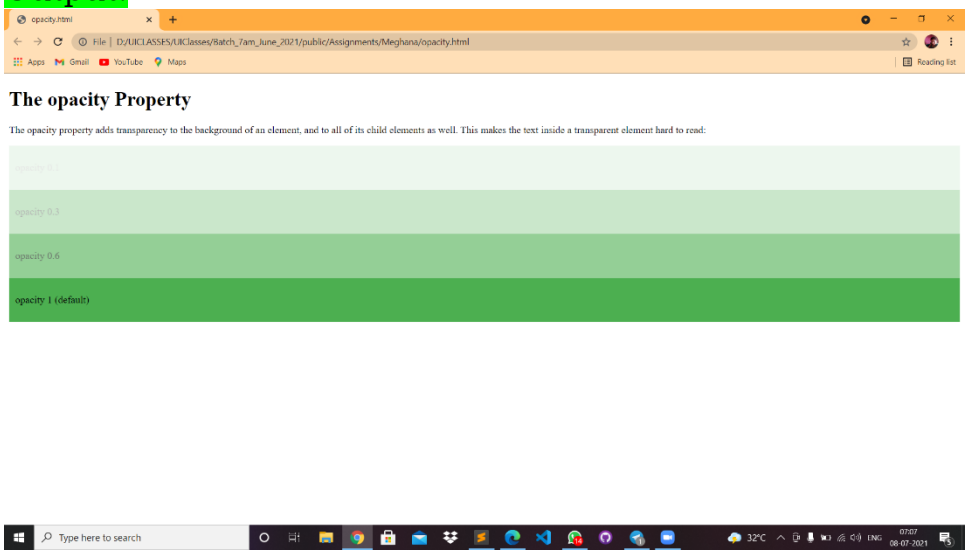
opacity: 0.3;

Example program:

```
<!DOCTYPE html>
                        <html>
                            <head>
                                <style>
                            div {
                              background-color: #4CAF50;
                              padding: 10px;
                            }

                            div. First {
                              opacity: 0.1;
```

```
}

div. Second {
  opacity: 0.3;
}

div. Third {
  opacity: 0.6;
}
</style>
</head>
<body>

<h1>The opacity Property</h1>

<p>The opacity property adds transparency to the background of an element, and to all of its child elements as well. This makes the text inside a transparent element hard to read:</p>

<div class="first"><p>opacity 0.1</p></div>
<div class="second"><p>opacity 0.3</p></div>
<div class="third"><p>opacity 0.6</p></div>
<div><p>opacity 1 (default)</p></div>

</body>
</html>
```

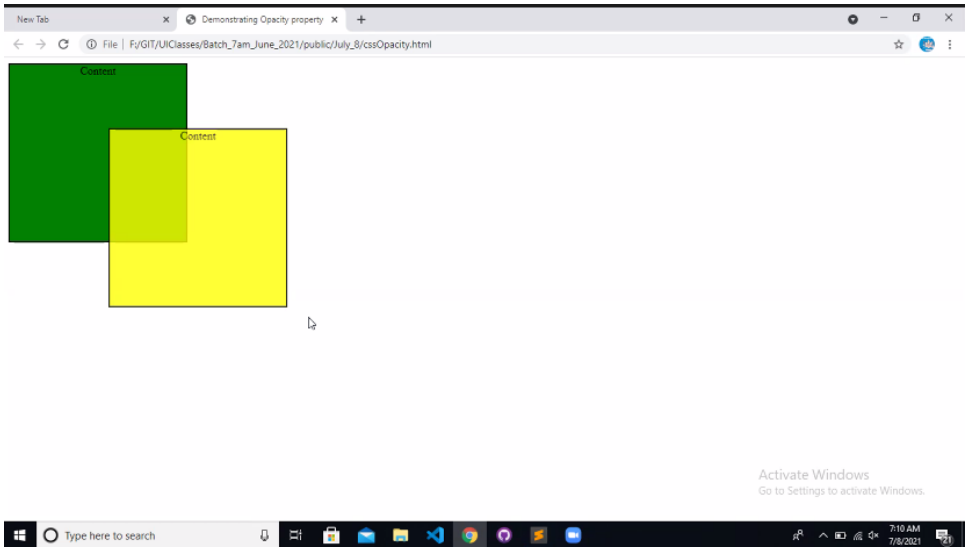
Output:



Example-2:

```
<!DOCTYPE html>
<html>
    <head>
        <title>
            Demonstrating Opacity property
        </title>
        <style type="text/css">
            .container {
                width: 250px;
                height: 250px;
                border: 2px solid;
                text-align: center;
            }
            .container1 {
                background: green;
            }
            .container2 {
                background: yellow;
                position: absolute;
                left: 150px;
                top: 100px;
                opacity: 0.8;
            }
        </style>
    </head>
    <body>
        <div class="container container1">Content</div>
        <div class="container container2">Content</div>
    </body>
</html>
```

Output:



HTML 'Input' Elements:

Following are the predefined HTML elements supported using which we could able to read different types of data from the user.

<input type="text" > --> To read the type of content.
<input type=" password" > --> To read sensitive data.
<input type=" radio" > --> creates a radio button.
<input type=" checkbox" > --> creates a checkbox.
<input type=" button" > --> creates a button.

textarea -->creates a multiline text container.

<select>
<option>one</option>
<option>two</option>
</select> --> creates a dropdown with multiple options.
Etc.

Example-1:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Student Registration Form</title>
    <style>
        * {
            font-family: Arial, Helvetica, sans-serif;
        }

        input {
            padding: 10px;
            margin: 10px;
        }

        button {
            margin: 10px;
        }
        form {
            margin: 10px;
            padding: 10px 20px;
            border: rgb(93, 1, 180) solid 2px;
        }
        h1 {
            text-decoration: underline;
            color: blueviolet;
        }
        a {
            text-decoration: none;
        }
    </style>
</head>

<body>
```

```
   <center>
      <div>
         <img
src="C:\Users\nayni\Desktop\jr_files\html\stude\srus.jpg"
alt="Registration" />
      </div>
      <h1>STUDENT REGISTRATION FORM</h1>
   </center>
   <form action="./thankYou.html" style="background-color: blueviolet;
color: white;">
      <table>
         <tr>
            <td>First Name:</td>
            <td><input type="text" maxlength="30" placeholder="Enter
first name" /> (max 30 characters a-z A-z)
            </td>
         </tr>
         <tr>
            <td>Last Name:</td>
            <td><input type="text" maxlength="30" placeholder="Enter
last name" /> (max 30 characters a-z A-z)
            </td>
         </tr>
         <tr>
            <td>Date of Birth</td>
            <td>
               <select name="" id="">
                  <option value="null">Day:</option>
                  <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="10">10</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>
```

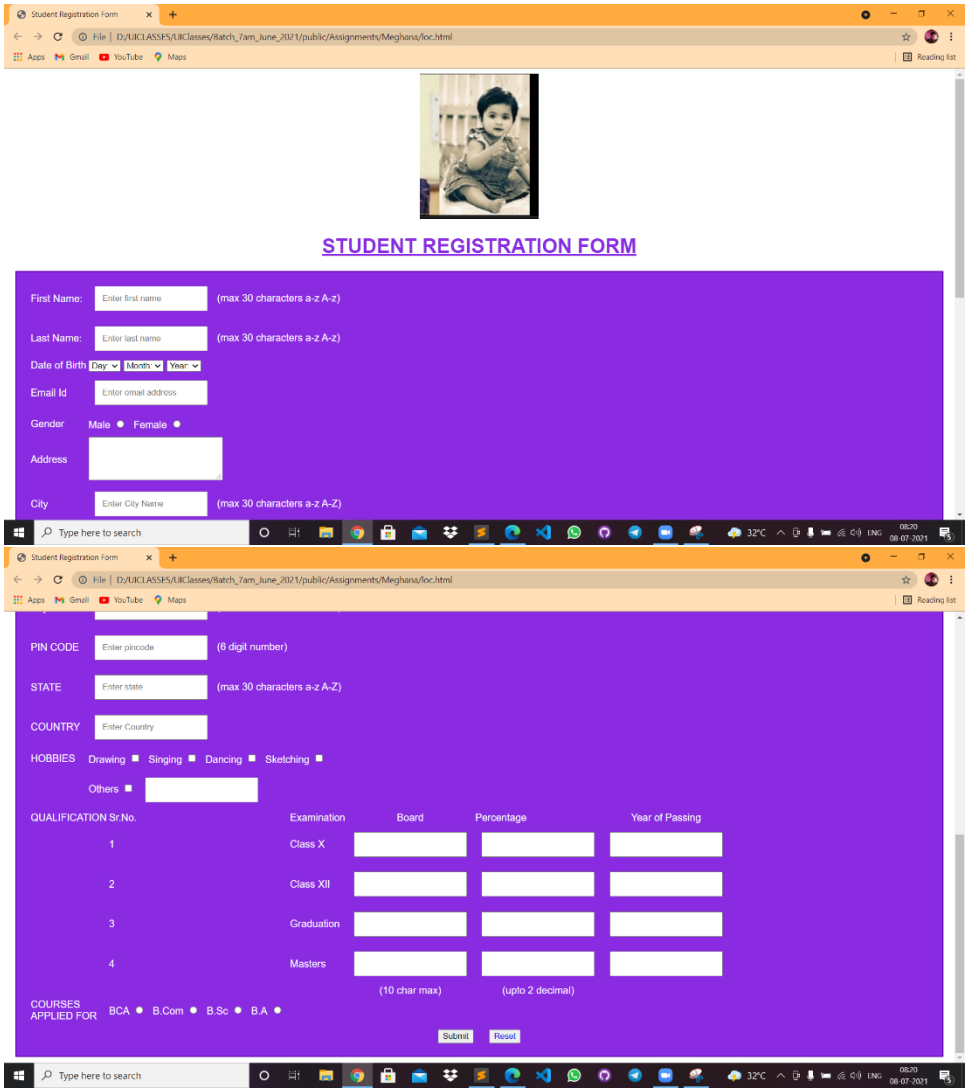
```
            <option value="18">18</option>
            <option value="19">19</option>
            <option value="20">20</option>
            <option value="21">21</option>
            <option value="22">22</option>
            <option value="23">23</option>
            <option value="24">24</option>
            <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 name="Month" id="">
            <option value="null">Month:</option>
            <option value="Jan">Jan</option>
            <option value="Feb">Feb</option>
            <option value="Mar">Mar</option>
            <option value="Apr">Apr</option>
            <option value="May">May</option>
            <option value="Jun">Jun</option>
            <option value="Jul">Jul</option>
            <option value="Aug">Aug</option>
            <option value="Sept">Sept</option>
            <option value="Oct">Oct</option>
            <option value="Nov">Nov</option>
            <option value="Dec">Dec</option>
         </select>
         <select name="" id="">
            <option value="null">Year:</option>
            <option value="2020">2020</option>
            <option value="2019">2019</option>
            <option value="2018">2018</option>
            <option value="2017">2017</option>
            <option value="2016">2016</option>
            <option value="2015">2015</option>
            <option value="2014">2014</option>
            <option value="2013">2013</option>
            <option value="2012">2012</option>
         </select>
      </td>
   </tr>
   <tr>
```

```
            <td>Email Id</td>
            <td><input type="email" name="" id="" placeholder="Enter
email address"></td>
        </tr>
        <tr>
            <td>Gender</td>
            <td>
                <label for="male">Male</label><input type="radio"
name="gender" id="male">
                <label for="female">Female</label><input type="radio"
name="gender" id="female">
            </td>
        </tr>
        <tr>
            <td>Address</td>
            <td>
                <textarea name="" id="" cols="30" rows="4"></textarea>
            </td>
        </tr>
        <tr>
            <td>City</td>
            <td><input type="text" name="City" id="" maxlength="30"
placeholder="Enter City Name"> (max 30
                characters a-z A-Z) </td>
        </tr>
        <tr>
            <td>PIN CODE</td>
            <td><input type="number" name="" id="" maxlength="6"
placeholder="Enter pincode"> (6-digit number)
            </td>
        </tr>
        <tr>
            <td>STATE</td>
            <td><input type="text" name="" id="" placeholder="Enter
state" maxlength="30"> (max 30 characters
                a-z
                A-Z) </td>
        </tr>
        <tr>
            <td>COUNTRY</td>
            <td><input type="text" name="" id="" placeholder="Enter
Country"></td>
        </tr>
        <tr>
            <td>HOBBIES</td>
```

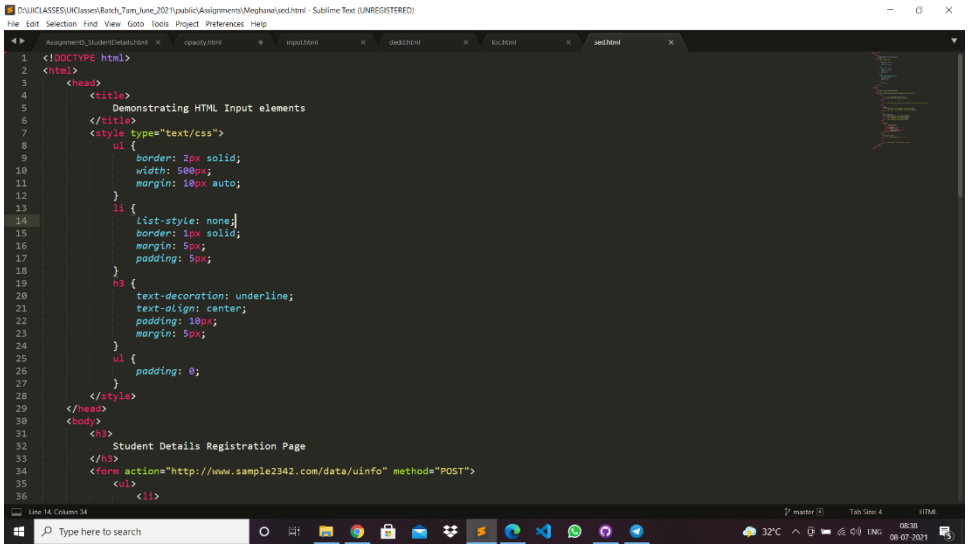
```
            <td>
               <label                    for="drawing">Drawing</label><input
type="checkbox" name="hobbies" id="drawing">
               <label                      for="singing">Singing</label><input
type="checkbox" name="hobbies" id="singing">
               <label                    for="dancing">Dancing</label><input
type="checkbox" name="hobbies" id="dancing">
               <label               for="sketching">Sketching</label><input
type="checkbox" name="hobbies" id="sketching">
            </td>
         </tr>
         <tr>
            <td></td>
            <td><label for="other">Others</label><input type="checkbox"
name="hobbies" id="other"><input type="text"
                  name="" id=""></td>
         </tr>
      </table>
      <table>
         <tr>
            <td>QUALIFICATION</td>
            <td>Sr.No.</td>
            <td>Examination</td>
            <td><center>Board</center></td>
            <td>Percentage</center></td>
            <td><center>Year of Passing</center></td>
         </tr>
         <tr>
            <td rowspan="4"></td>
            <td>1</td>
            <td>Class X</td>
            <td><input type="text" name="" id="" maxlength="10"></td>
            <td><input type="text" name="" id=""></td>
            <td><input type="text" name="" id=""></td>
         </tr>
         <tr>
            <td>2</td>
            <td>Class XII</td>
            <td><input type="text" name="" id="" maxlength="10"></td>
            <td><input type="text" name="" id=""></td>
            <td><input type="text" name="" id=""></td>
         </tr>
         <tr>
            <td>3</td>
            <td>Graduation</td>
```

```
            <td><input type="text" name="" id="" maxlength="10"></td>
            <td><input type="text" name="" id=""></td>
            <td><input type="text" name="" id=""></td>
        </tr>
        <tr>
            <td>4</td>
            <td>Masters</td>
            <td><input type="text" name="" id="" maxlength="10"></td>
            <td><input type="text" name="" id=""></td>
            <td><input type="text" name="" id=""></td>
        </tr>
        <tr>
            <td colspan="3"></td>
            <td><center>(10 char max) </center></td>
            <td><center>(up to 2 decimal) </center></td>
        </tr>
        <tr>
            <td>COURSES <br />APPLIED FOR</td>
            <td>
                <label for="bca">BCA</label><input type="radio"
name="COURSES" id="bca">
                <label for="BCom">BCom</label><input type="radio"
name="COURSES" id="BCom">
                <label for="bsc">B.Sc.</label><input type="radio"
name="COURSES" id="bsc">
                <label for="ba">B. A</label><input type="radio"
name="COURSES" id="ba">
            </td>
        </tr>
    </table>
    <center>
        <div>
            <button type="submit">Submit</button>
     <button type="button"><a
href="./index.html">Reset</a></button>
        </div>
    </center>
  </form>
</body>
</html>
```

Output:



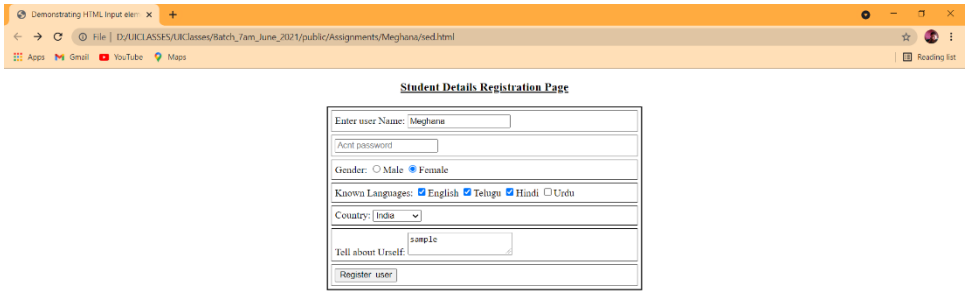
TODAY'S EXAMPLE:

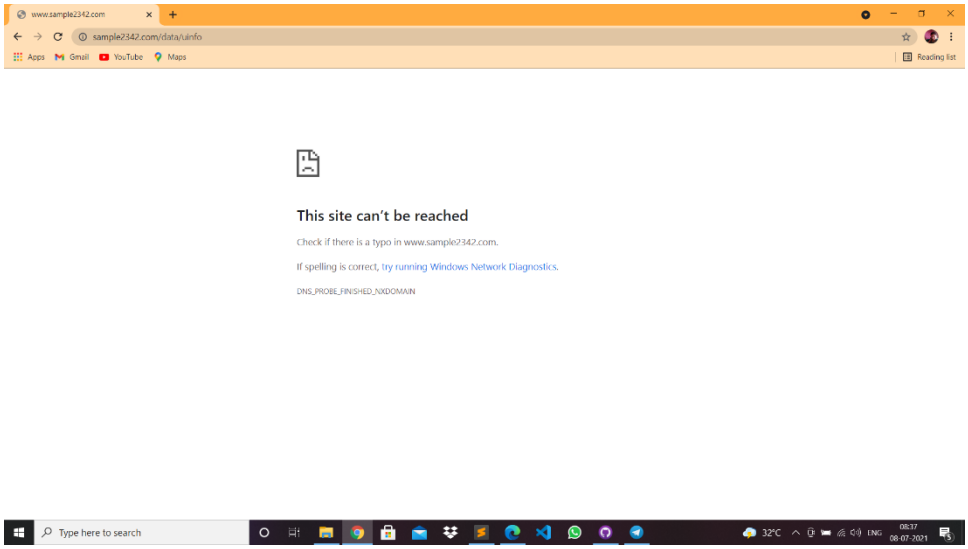


```
                    <label for="uname">Enter user Name:</label>
                    <input type="text" name="uname" required>
                </li>
                <li>

                    <input name="upwd" type="password" maxlength="5" placeholder="Acnt password">
                </li>
                <li>
                    Gender:
                    <input type="radio" name="ugender" value="male">Male
                    <input type="radio" name="ugender" value="female">Female
                </li>

                <li>
                    Known Languages:
                    <input type="checkbox" name="lng_eng">English
                    <input type="checkbox" name="lng_tel">Telugu
                    <input type="checkbox" name="lng_hin">Hindi
                    <input type="checkbox" name="lng_urd">Urdu
                </li>
                <li>
                    Country:
                    <select name="ucountry">
                        <option></option>
                        <option>India</option>
                        <option>USA</option>
                        <option selected>Japan</option>
                        <option>Singapor</option>
                    </select>
                </li>
                <li>
                    Tell about Urself:
                    <textarea name="aboutUser"></textarea>
                </li>

                <li>
```

```
                    <input type="submit" value="Register  user">
                </li>
            </ul>
        </form>
    </body>
</html>
```

OUTPUT:





DATE: 9/7/2021

HTML form tag:

A predefined tag using which we could able to send user input data to server. It takes following mandatory attributes.

. Method

. Action

*Method attribute through which we could specify the type of communication while sending or receiving the data.

*Action attribute through which we specify the path of the server to which communication should happen.

```
<form action=" path/url of server "method=" GET/POST">
...
... // set of input elements.
</form>
```

❖ Types of communications:

While communicating with a server, it could be either secure on non-secure type of communication.

1.non-secured type of communication (GET): In this type of communication, data will be sent to the server by appending to the URL as query parameters

E.g:

http://www.sample.com/data/user/info?uname=test&age=20&gender=male

2.Secured communication (POST): In this type of communication the data will be sent to the server by adding it to the request header, which is not exposed to the end user.

*Any time we send sensitive data to server we use post type of communication.

❖ JAVASCRIPT:

★ JavaScript (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document.

★ JavaScript was first known as Live Script, but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java.

♦ Advantages of JavaScript

× The merits of using JavaScript are:

 Less server interaction: You can validate user input before sending the page off to the server. This saves server traffic, which means less load on your server.

 Immediate feedback to the visitors: They don't have to wait for a page reload to see if they have forgotten to enter something.

 Increased interactivity: You can create interfaces that react when the user hovers over them with a mouse or activates them via the keyboard.

 Richer interfaces: You can use JavaScript to include such items as drag and-drop.

❖ Applications of JavaScript:

× Client-side validation,

× Dynamic drop-down menus,

× Displaying date and time,

- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks etc.

Syntax of JavaScript:

```
<script ...>
        JavaScript code
      </script>
```

EXAMPLE PROGRAM:

```
<html>
  <body>
    <script language = "javascript" type = "text/javascript">
      <!--document.write("Hello World!")// -->
    </script>
      </body>
</html>
```

The Structure of JavaScript:

When you include any JavaScript code in an HTML document (apart from using the <script> tag), you must follow the few conventions:

1. HTML standards prior to HTML5 required that <script> tag to be placed between <head> and </head> tags at the starting of the document.
2. Unlike HTML which uses the <! --- comment tag ---> commits inside Javascript code use the // symbol at the start of the line.

So, as I was always saying we use <script> tag let us know what is this <script> tag:

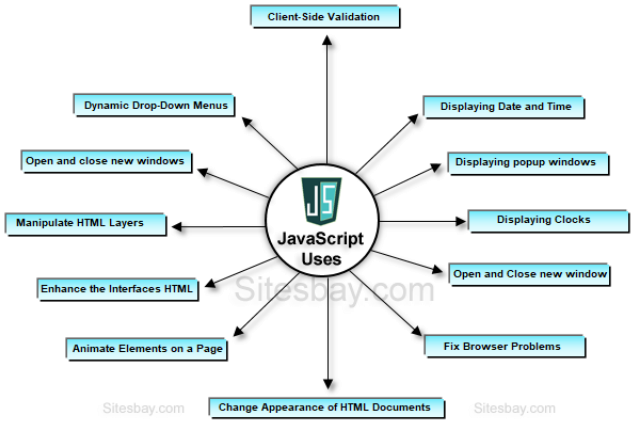
The <script> tag:

The <script> tag is used to include a JavaScript script in a webpage.

- The HTML <script> tag is used to define a client-side script (JavaScript).
- The <script> element either contains script statements, or it points to an external script file through the src attribute.
- Common uses for JavaScript are image manipulation, form validation, and dynamic changes of content.
- To select an HTML element, JavaScript most often uses the document.getElementById() method.
- This JavaScript example writes "Hello JavaScript!" into an HTML element with id="demo":
- <script>
  document.getElementById("demo"). inner HTML = "Hello JavaScript!";
  </script>

Why we use JavaScript:
JavaScript is mainly used for web-based applications and web browsers. But JavaScript is also used beyond the Web in software, servers and embedded hardware controls. Here are some basic things JavaScript is used for:

1. Adding interactive behaviour to web pages
2. Creating web and mobile apps
3. Building web servers and developing server applications
4. Game Development.



Date: 12/07/2021
Datatypes in JavaScript:

Datatypes is an important concept in JavaScript that will be able to operate on variables. The datatype specifies the type of data that a variable can hold. In JavaScript while declaring a variable it is not required to specify a datatype which declares the variables.

"VAR" is a pre-defined keyword through which we could declare the variables in JavaScript.
Basically, there are seven datatypes in JavaScript:

$ Numbers
  - A=90
  - B=9.4

$ String
  - Code = 'a'
  - Name = 'India'

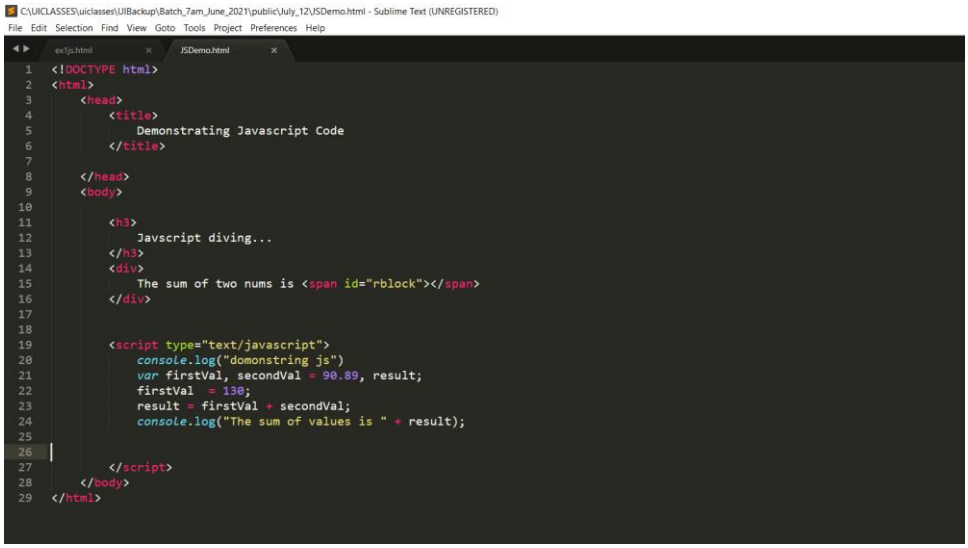
$ Boolean

- IsFeePaid = true;
- Isfemale = false;

$ Undefined
$ Null
$ function
$ Object

*In side JavaScript we need not decide what type of data to store and how much memory it has to be occupied.
Ex: var a --> declaring a variable a capable of data
var b --> variable b capable of data
var c
a=4.5 -->number or a = “Hello” --> string
*In JavaScript any line we are declaring should end with the semicolon. It indicated that is the end of the code.

Example:

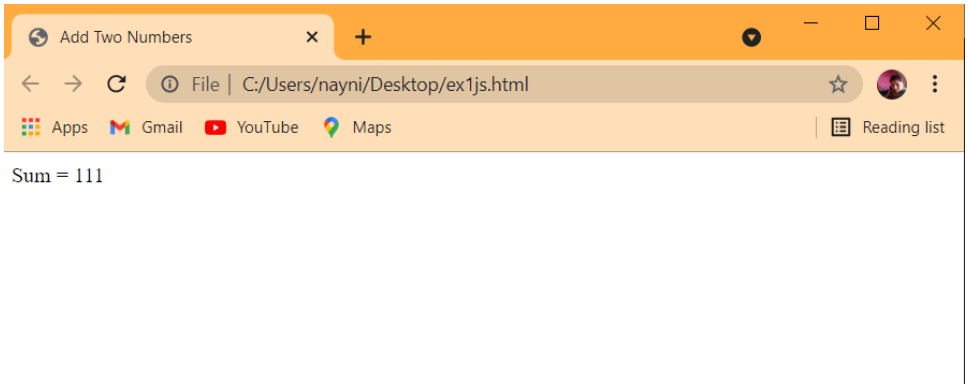


Example JavaScript program for adding two numbers:

```
<!DOCTYPE html>
        <html>
            <head>
            <title>Add Two Numbers</title>
            <script>
              var m = 91;
              var a = 20;
              var sum = m + a;
              document. Write ("Sum = " + sum);
            </script>
```

```
        </head>
        <body>

        </body>
    </html>
```

Output:

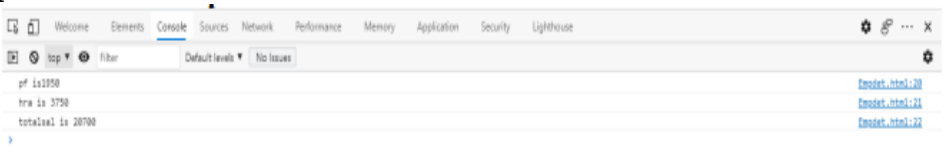


Date: 13/07/2021

Write a program for employee details like sname,sage,basic sal,totsal,hra,pf and department and calculate the total salary?

```
<script type=”text/javascript”>
var sname=”Meghana”;
var sage=22;
var gender =” Female”;
var dept=” CSE”;
var basicsal=15000;

var totalsal,hra,pf;
pf=13/100*basicsal;
hra=25/100*basicsal;
totalsal = hra + basicsal + pf;

console.log (“pf is “+ pf);
console.log (“hra is “+ hra);
console.log (“totalsal is “+ totalsal);
```

</script>



Type of Method:

A predefined method being supported in JavaScript using which we could able to get type of data a variable is holding. It takes a variable as a parameter and returns, the type of data the variable is holding.
Syntax: typeof(Variable_Name);

Ex: var num = 10;
Var name= 'Meghana';
console.log(typeof(age));
console.log(typeof(name));

If any variable is not initialized it will be defined as undefined. It takes the variable parameter and returns the type of data the variable is holding.

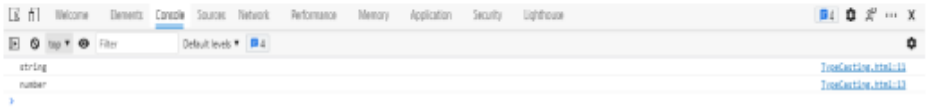
Type Casting in JavaScript:

Typecasting, or type conversion, is a method of changing an entity from one datatype to another. It is used in computer programming to ensure variables are correctly processed by a function.

An example of typecasting is converting an integer to a string.

Example:

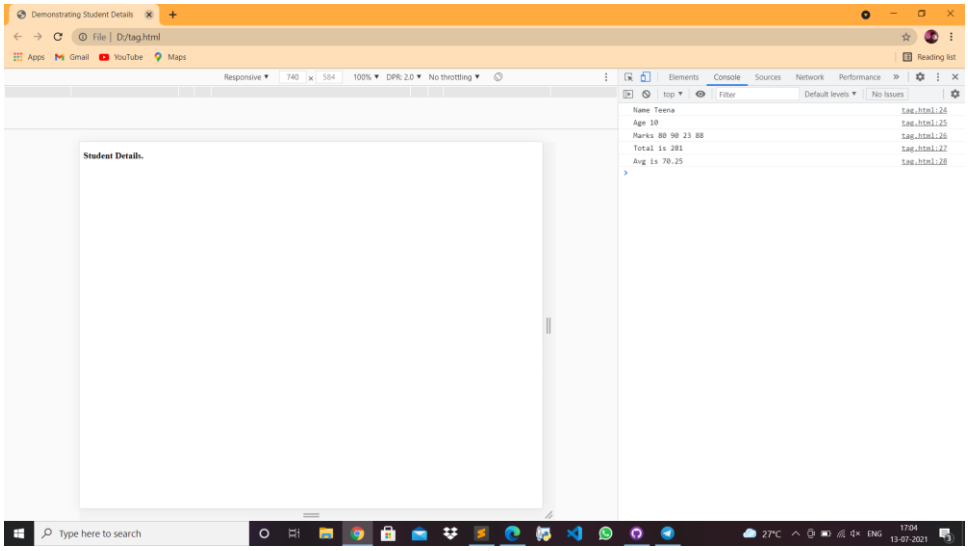
```
<script type="text/javascript">
        var num = 10;
         Var name= 'Meghana';
         console.log(typeof(num));
         console.log(typeof(name));
        </script>
```



TODAYS EXAMPLE STUDENT DETAILS:

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

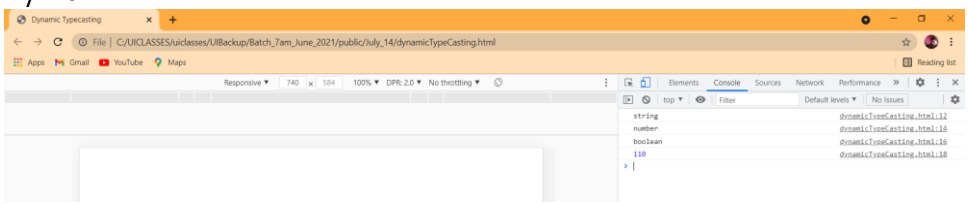
```
<title>
Demonstrating Student Details
</title>
</head>
<body>
<h3>
Student Details.
</h3>
<script type="text/javascript">

var sname = "Teena";
var sage = 10;
var gender = 'Female';
var m1 = 80;
var m2 = 90, m3 = 23, m4 = 88;
var total, avg;

total = m1 + m2 + m3 + m4;

avg = total / 4;
console.log ("Name " + sname);
console.log ("Age " + sage);
console.log ("Marks " + m1 + ' ' + m2 + " " + m3 + ' ' + m4);
console.log ("Total is " + total);
console.log ("Avg is " + avg);

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



Date: 14/07/2021

Type Casting:
The process of converting datatype of a variable into another is called type casting.
This is classified into two types:
ParseInt () - Takes any datatype value and returns the number.
Parse float () - Takes any datatype value and returns the decimal number.
Example:

```
<!DOCTYPE html>
<html>
<head>
<title>
Dynamic Typecasting
</title>
</head>
<body>
<script type="text/javascript">
var z = 80
var a = "30";
console.log(typeof(a));
a = 90;
console.log(typeof(a));
a = true;
console.log(typeof(a));
console.log (30 + z);

var c = 10 + z

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

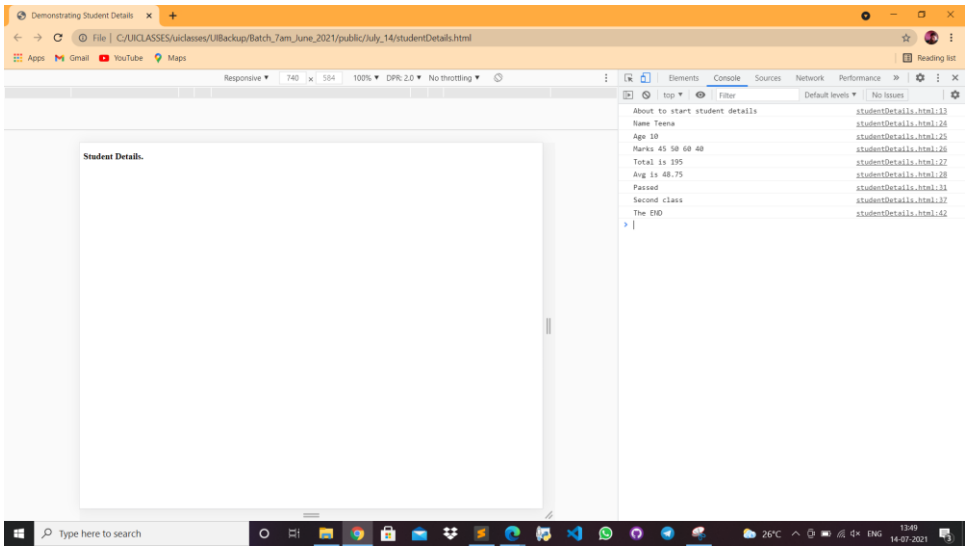


Today's topic: Dynamic Type casting:
JavaScript internally implements and supports a feature called dynamic type casting in which it automatically converts the datatype of a variable based on type of data it is actually holding.

STUDENT DETAILS EXAMPLE:

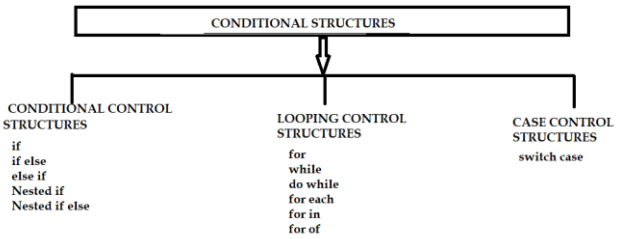
```
<!DOCTYPE html>
<html>
<head>
<title>
Demonstrating Student Details
</title>
</head>
<body>
<h3>
Student Details.
</h3>
<script type="text/javascript">
console.log ("About to start student details")
var sname = "Teena";
var sage = 10;
var gender = 'Female';
var m1 = 45;
var m2 = 50, m3 = 60, m4 = 40;
var total, avg;

total = m1 + m2 + m3 + m4;

avg = total / 4;
console.log ("Name " + sname);
console.log ("Age " + sage);
console.log ("Marks " + m1 + ' ' + m2 + " " + m3 + ' ' + m4);
console.log ("Total is " + total);
console.log ("Avg is " + avg);

if (avg >= 40) {
console.log("Passed");
if (avg >= 90) {
console.log ("you got Distinction")
} else if(avg >= 60) {
console.log ("you got first class");
} else {
console.log ("Second class");
}
} else {
console.log("Failed");
}
console.log ("The END");


</script>
```

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



Control structures:
Following are the set of control structures being supported through which we could able to control the sequence of execution flow in an application.

1.Conditional Control Structures.
2.Looping Control Structures.
3.Case Control Structures.



1.Conditional Control Structures:

- æ IF Condition: The **if** statement is the fundamental control statement that allows JavaScript to make decisions and execute statements conditionally.
- æ Syntax:

```
if (condition) {
// block of code to be executed if the condition is true
```

```
}
```

æ Else: We use the else statement to specify a block of code to be executed if the condition is false.

æ Syntax:

```
if (condition) {
// block of code to be executed if the condition is true
}
else {
// block of code to be executed if the condition is false
}
```

æ Else if Statement: Use the else if statement to specify a new condition if the first condition is false.

æ Syntax:

```
if (condition1) {
// block of code to be executed if the condition1 is true.
}
else if (condition2) {
// block of code to be executed if the condition1 is false and condition 2 is true.
}
else {
// block of code to be executed if the condition1 is false and condition 2 is False.
}
```

æ Nested if: A nested if is an if statement that is the target of another if or else. Nested if statements mean an if statement inside an if statement. JavaScript allows us to nest if statements within if statements. i.e., we can place an if statement inside another if statement.

æ Syntax:

```
if (condition1) {
// block of code to be executed if the condition1 is true
} if (condition2) {
// block of code to be executed if the condition 2 is true.
}
```

Date: 15/07/2021

2.Looping control structures:

Looping control structures has been supported in Javascript takes the set of instructions and repeat them for multiple times. JavaScript supports different kinds of loops:

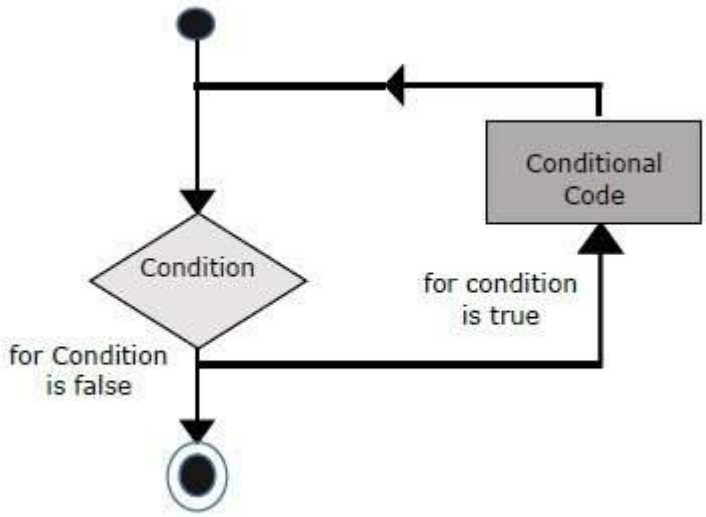
- For - loops through a block of code a number of times
- for/in - loops through the properties of an object
- for/off- loops through the values of an iterable object
- while - loops through a block of code while a specified condition is true
- do/while - also loops through a block of code while a specified condition is true

- For Loop: Loops through a block of code a number of times.
  - The '**for**' loop is the most compact form of looping. It includes the following three important parts −
    - The **loop initialization** where we initialize our counter to a starting value. The initialization statement is executed before the loop begins.
    - The **test statement** which will test if a given condition is true or not. If the condition is true, then the code given inside the loop will be executed, otherwise the control will come out of the loop.
    - The **iteration statement** where you can increase or decrease your counter.

Syntax:

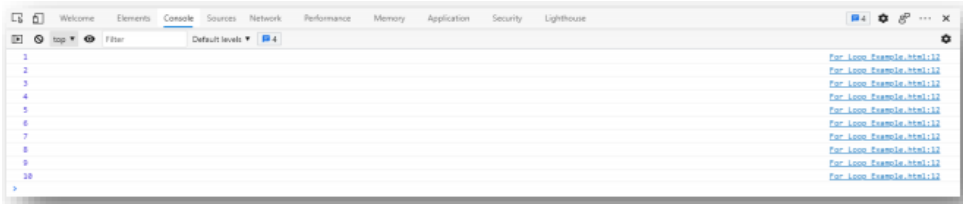
```
for (initialization, <condition>, increment/decrement)
{// set of lines to be Executed
}
```

- The flow chart of a for loop in JavaScript would be as follows −



Ex:

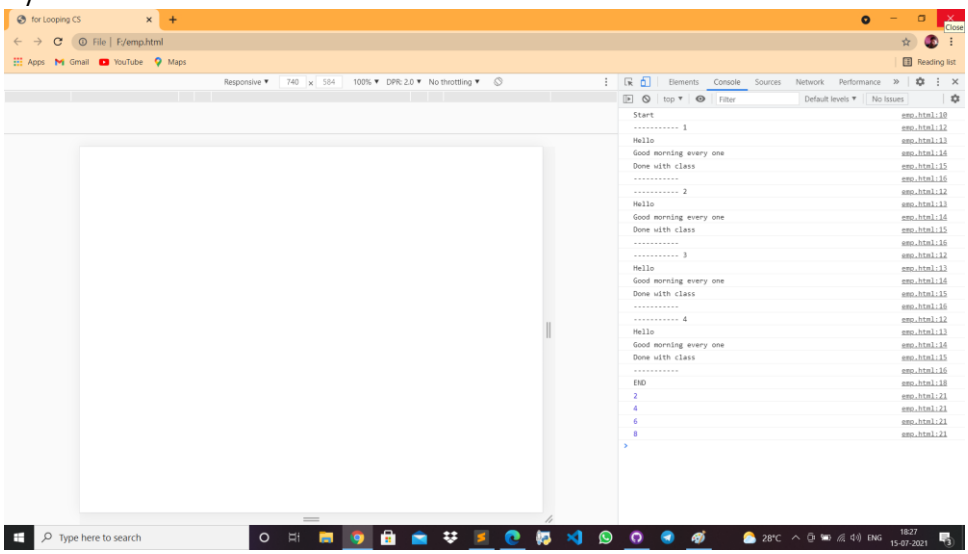
```
<script type=”text/javascript”>
                Var i
                Var j=10;
                For (i=1; i<=j; i++) {
                Console.log(i);
                }
                </script>
```



TODAY’S EXAMPLE:

```
<!DOCTYPE html>
<html>
<head>
<title>
for Looping CS
</title>
</head>
<body>
<script type="text/javascript">
console.log("Start")
for (var i = 1; i <= 4; i++) {
console.log ("---------- " + i)
console.log ("Hello ");
```

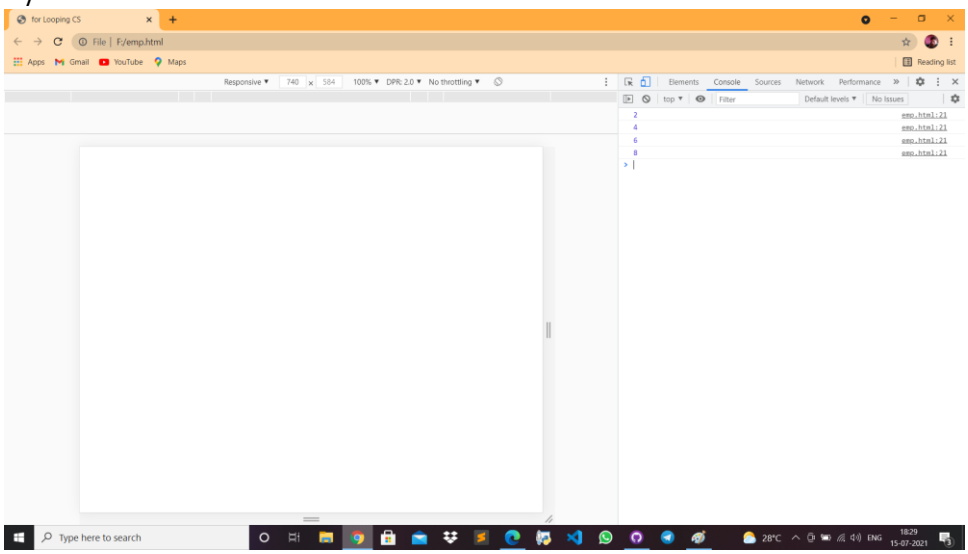
```
console.log ("Good morning every one");
console.log ("Done with class");
console.log ("-----------");
}
console.log("END");
for (var i = 1; i <= 9; i++) {
if ( i % 2 == 0) {
console.log(i);
}
}

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



EXAMPLE-2:

```
<!DOCTYPE html>
<html>
<head>
<title>
for Looping CS
</title>
</head>
<body>
<script type="text/javascript">
/*console.log("Start")
for (var i = 1; i <= 4; i++) {
console.log ("----------- " + i)
console.log ("Hello ");
console.log ("Good morning every one");
console.log ("Done with class");
```

```
console.log ("-----------");
}
console.log("END"); */
for (var i = 1; i <= 9; i++) {
if ( i % 2 == 0) {
console.log(i);
}
}

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



Assignment:
Emp details – age, name, dept, basic Sal, gender
Cal pf, total Sal, hra

Also calculate tax value
male emp – total Sal -> 2lakh -> tax percent is 15%
male emp – total Sal -> 1lakh -> tax percent is 10%
male emp – total Sal -> 50k -> tax percent is 5%
Female emp – total Sal -> 2lakh -> tax percent is 10%
Female emp – total Sal -> 1lakh -> tax percent is 5%
Female emp – total Sal -> 50k -> tax percent is 0%

Difference between '=', '==' and '===' operators:

- Equal to (=) is an assignment operator, which sets the variable on the left of the = to the value of the expression that is on its right. This operator assigns lvalue to rvalue.
- For example, writing a=10 is fine. If we write 10=10, 'a' = 10 or 'a' = 'a', it will result in a reference error.

- Double equals (==) are a comparison operator, which transforms the operands having the same type before comparison. So, when you compare string with a number, JavaScript converts any string to a number. An empty string is always converts to zero. A string with no numeric value is converts to NaN (Not a Number), which returns false.
- Note: == operator while comparing values only checks for value equality but not type equality

- === (Triple equals) is a strict equality comparison operator in JavaScript, which returns false for the values which are not of a similar type. This operator performs type casting for equality. If we compare 2 with "2" using ===, then it will return a false value.
- Note: === it is almost like '==' operator used to compare value equality where the only difference is while comparing values it not just checks for value equality but it also checks for type equality of data

Date: 17/07/2021

Write a program to show the list of odd numbers between 59-11.

```
<script type="text/javascript">
   for (var i =59; i>=11; i--)
   {
      if (i%2! =0)
      {
      console.log ("odd number +i");
   }
   }
</script>
```

Write a program to show the list of even numbers between 1-100.

```
<script type="text/javascript">
   for (var i =1; i>=100; i--)
   {
      if (i%2 ==0)
      {sum = sum + i;
console.log (sum is "+sum);}
      </script>
```

Write a program to show the given the number is prime or not.

```
<script type="text/javascript">

        function checkPrime() {

      var n, i, flag = true;

       n = document.myform.n.value;
      n = parseInt(n)
      for (i = 2; i <= n - 1; i++)
          if (n % i == 0) {
              flag = false;
              break;
          }

      // Check and display output
      if (flag == true)
          console.log (n + " is prime");
      else
          console.log (n + " is not prime");
    }
</script>
```

ASSIGNMENT:

Write a program to find sum of all the prime numbers between 1 –100
Write a program to show the table of a given number.

Date: 19/07/2021

- While loop control structure: It is almost like for loop control structure used to iterate instructions more than once where the only difference is while loop only takes the condition initialization, increment or decrement can be placed anywhere with the code.

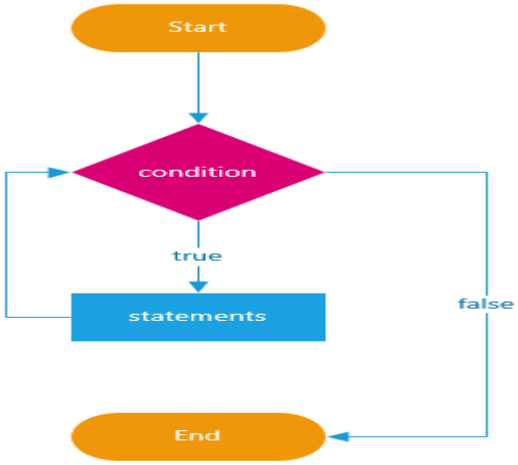
Syntax:
While (<condition>) {
……
……
…… // set of instructions gets executed for multiple times, until the provided condition is unsatisfied.

- ¶ The while statement evaluates the expression before each iteration of the loop.

¶ If the expression evaluates to true, the while statement executes the statement. If the expression evaluates to false, execution continues with the statement after the while loop.

¶ The while loop evaluates the expression before each iteration; therefore, the while loop is known as a pre-test loop. For this reason, it is possible that the statement inside the while loop is never executed.

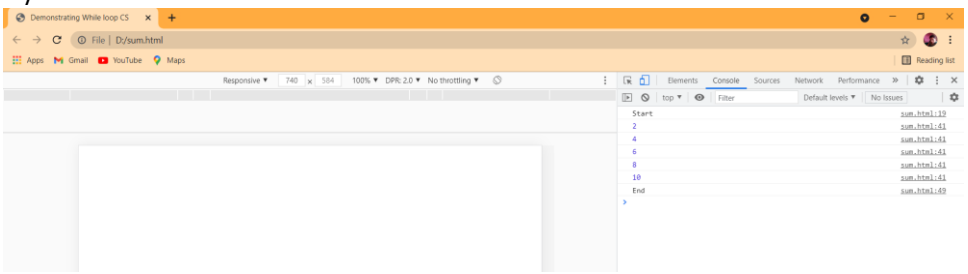
¶ The following flowchart illustrates the while loop statement:



Example:

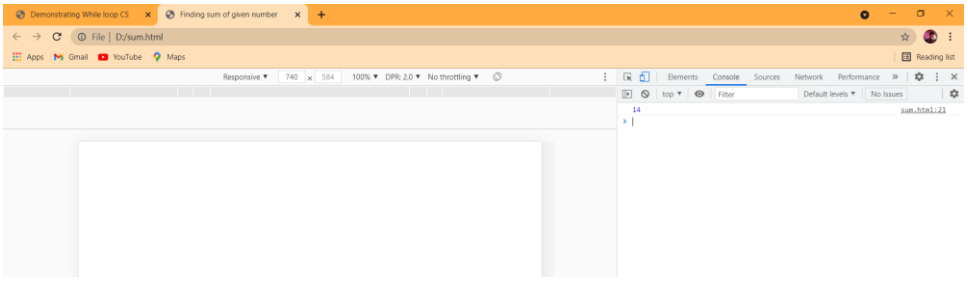
```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>
Demonstrating While loop CS
</title>
</head>
<body>
<script type="text/javascript">
console.log("Start");
/*for (var i = 1; i <= 10; i++) {

if (i % 2 == 0) {
console.log(i);
}
}*/

var i = 1;
while ( i <= 10) {
```

```
if (i % 2 == 0) {
console.log(i);
}
i++;
}
console.log("End")
</script>
</body>
</html>
```



SUM OF GIVEN NUMBERS:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>
Finding sum of given number
</title>
</head>
<body>
<script type="text/javascript">

var n = 2345;
var rem, sum = 0;
while (n > 0) {
rem = n % 10;
sum = sum + rem;
n = n / 10;
n = parseInt(n);

};
console.log(sum)
</script>
</body>
</html>
```



ASSIGNMENTS:

1. Write a program to find out the lucky number of a given number.
2. Write a program to find out the reverse of a given number.
3. Write a program to check whether given number is palindrome number or not.
4. Write a program to list out all the palindrome numbers between 300-21.
5. Write a program to list out reverse of numbers between 32-950.
6. Write a program to check whether given number is Armstrong number or not.

Date: 20/07/2021

- Prompt Method: This method is used to display a dialog box, which would require a person to type in input.
- The JavaScript prompt () method is commonly used for prompting the user for input before entering a web page.
- [redacted]

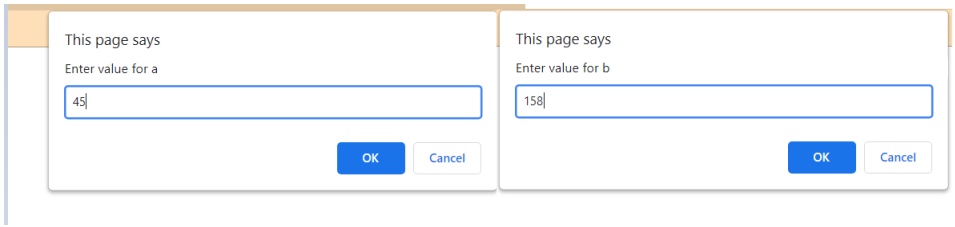
PROMPT WINDOW

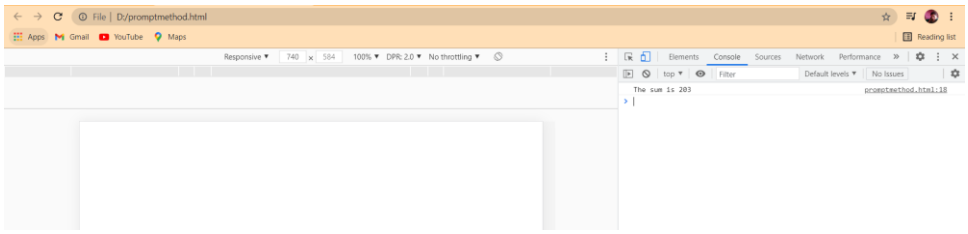
Note: Once a prompt box appears, the user has to click either ok or cancel after entering the input. However, do not use this method excessively, as it prevents access to parts of the webpage until submitted, making the user experience less smooth.

Example:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>
Demonstrating Prompt Method
</title>
```

```
</head>
<body>
<script type="text/javascript">
var a = prompt ("Enter value for a");
var b = prompt ("Enter value for b");
a = parseInt(a);
b = parseInt(b);
var result = a + b;
console.log ("The sum is " + result);
</script>
</body>
</html>
```

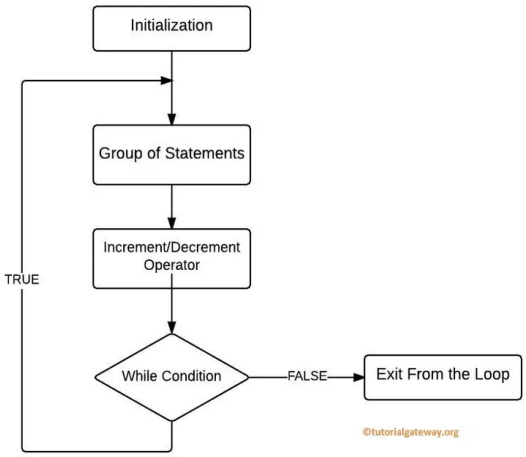




DO-WHILE LOOP:

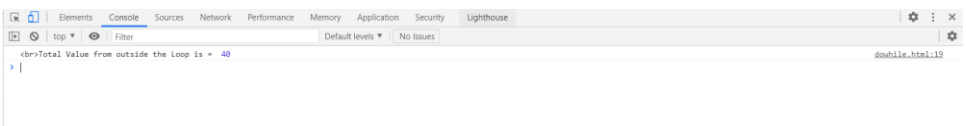
- The do-while creates a loop that executes a specified statement until the test condition evaluates to false. The condition is evaluated after executing the statement, resulting in the specified statement executing at least once.
- It is almost like a while loop control structure use to repeat a set of instructions for more than once where the only difference is while loop until execute the instructions only when the provided condition is true, whereas do-while executes set of instructions at least for once irrelevant of where the provided where the condition given is true or false.
- Syntax:

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



Example:

```
<!DOCTYPE html>
<html>
<head>
   <title> JavaScript Do While Loop </title>
</head>

<body>
   <h1> JavaScript Do While </h1>
<script>
   var number = 6, total=0;
   do
   {
      total = total + number;
      document.write("<br\>Number = " + number);
      document.write("<br\>Total Value is = " + total);
      number++;
   }
while (number <= 10);

   console.log("<br\>Total Value from outside the Loop is = ", total);
</script>
</body>
</html>
```



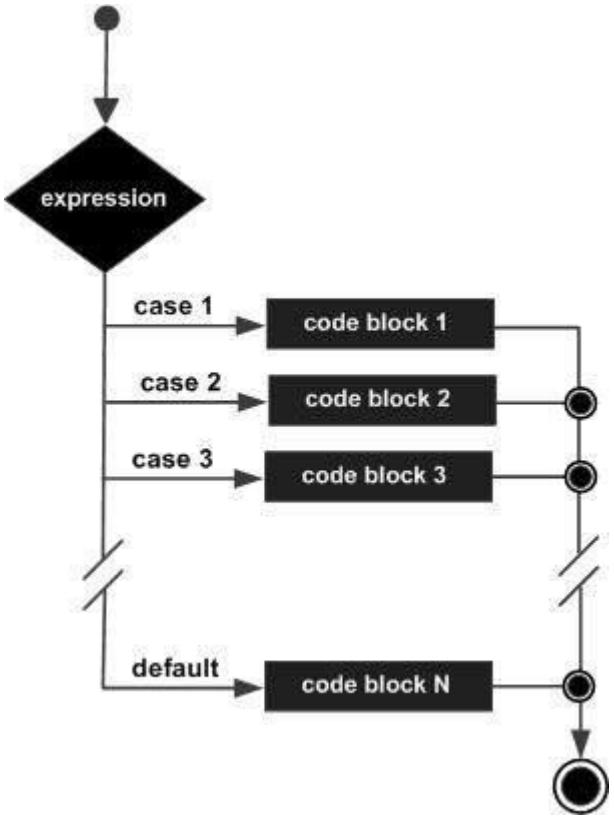
Date: 21/07/2021

Variable Hosting:

- The var statement declares a function-scoped or globally-scoped variable, optionally initializing it to a value. The var declarations, wherever they occur, are processed before any code is executed. This is called hosting.
- In JavaScript, Hoisting is the default behaviour of moving all the declarations at the top of the scope before code execution. Basically, it gives us an advantage that no matter where functions and variables are declared, they are moved to the top of their scope regardless of whether their scope is global or local. It allows us to call functions before even writing them in our code.

Case Control Statements--->Switch Case:

- It takes multiple cases within it and invokes corresponding particular case based on the value that is been passed.
- The switch statement evaluates an expression. The value of the expression is then compared with the values of each case in the structure. If there is a match, the associated block of code is executed.
- The switch statement is often used together with a break or a default keyword (or both). These are both optional:
- The break keyword breaks out of the switch block. This will stop the execution of more execution of code and/or case testing inside the block. If break is omitted, the next code block in the switch statement is executed.
- The default keyword specifies some code to run if there is no case match. There can only be one default keyword in a switch. Although this is optional, it is recommended that you use it, as it takes care of unexpected cases



Syntax:
Switch (<value>) {
case <value 1>:
……
break;
case <value 2>:
……
break;
……
……
default :
……// gets executed automatically if none of cases matches
}

Example:

```
<script type = "text/javascript">

<! -- var grade = 'A';

document.write("Entering switch block<br />");

switch (grade)
```

```
{
case 'A': document.write("Good job<br />");
case 'B': document.write("Pretty good<br />");
case 'C': document.write("Passed<br />");
case 'D': document.write("Not so good<br />");
case 'F': document.write("Failed<br />");
default: document.write("Unknown grade<br />")
}
document.write("Exiting switch block");
//--> </script>
```

.....
.....

ARRAYS:

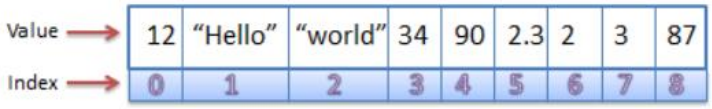
- A predefined data structure through which we could be able to hold group of multiple values within it.
- In JavaScript arrays are heterogeneous data structure capable of holding different types of data.
- Array is a single variable that is used to store different elements.
- It is often used when we want to store list of elements and access them by a single variable.
- '[]' braces are always used to represent arrays in any language.
- All the values within the array will automatically assigned a numeric index value "ZERO.

Here are the two different ways we can create an array in JavaScript:

- Var a= new array () // Dynamic allocation.

(or)

- Var a = []; // Empty Array
  Var a = [25,25,25,5]; // Declaring an array with value



JavaScript Array

Example: A sample example for 2-D element of Array:
var a = [[1, 2, 1, 24], [8, 11, 9, 4], [7, 0, 7, 27], [7, 4, 28, 14], [3, 10, 26, 7]];

```
for (var i in a)
{
  console.log ("row " + i);
  for (var j in a[i])
    {
     console.log (" " + a[i][j]);
    }
}
```

Date: 24/07/2021

Assignment:

- Write a program to find out all the even numbers with in n-array?
- Write a program to find Sum of odd numbers within array?
- Write a program to list out the prime numbers within given array?
- Write a program to find out the palindrome numbers within the given array?
- Write a program to find largest and smallest value from an array?
- Write a program to find the second largest value from an array?
- Write a program to take an array and sort array in both ascending and descending order?
- Write a program to find out total number of times a given value is getting repeated in an array?

→ **Pre-defined Methods in arrays:**

- The strength of JavaScript arrays lies in the array methods. Array methods are functions built-in to JavaScript that we can apply to our arrays
- Each method has a unique function that performs a change or calculation to our array

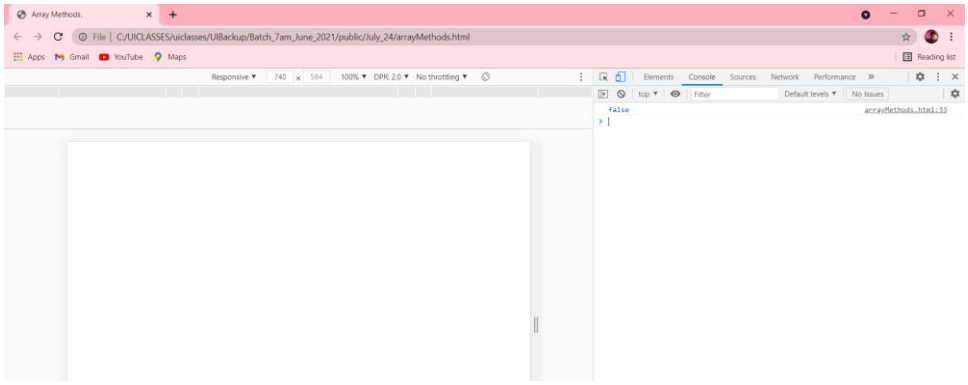
- The pre-defined methods in javascript are given below that basically supports the javascript which can be applied on arrays:

    - length – It is a property which returns the total number of values in an array.
    - push(<values>) - This method is used to insert single or multiple values to an array from its right direction
    - pop () - This method is used to delete the single value from an array from its right direction.
    - shift () - This method is used to delete a single value from an array from its left direction.
    - unshift (<values>) - This method is used to insert single or multiple values to an array from its left direction
    - splice (start position, <no of values to delete>, <optional values to be inserted>) - This splice method is used to insert or delete the values in an array.
    - join () - This method is used to merge all the values of an array to a single value with or without Seperator.
    - includes () - This method checks if an array contains the specified element

Example:

```
<script type="text/javascript">
var details = [34, 34, 78,90, 56, 89, 100, 102];
/*console.log(details);
console.log(details.length);
details.push(100);
console.log(details);
details.unshift(101);
console.log(details);
//-----
details.pop();
console.log(details);
details.shift();
console.log(details);
details.splice(4, 0, 201, 202, 304);
console.log(details);

console.log(details.join(''))
```

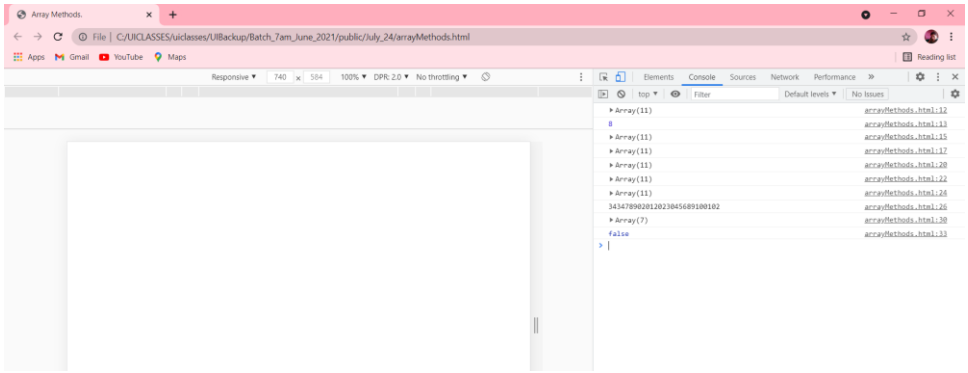
```
var a = [45, 67, 90];
var b = [34, 11, 22,33];
var c = a.concat(b);
console.log(c) */

var details = [34, 56, 89, 90, 100];
console.log(details.includes(99))
</script>
```



Example-2:

```
<script type="text/javascript">
var details = [34, 34, 78,90, 56, 89, 100, 102];
console.log(details);
console.log(details.length);
details.push(100);
console.log(details);
details.unshift(101);
console.log(details);
//-----
details.pop();
console.log(details);
details.shift();
console.log(details);
details.splice(4, 0, 201, 202, 304);
console.log(details);

console.log(details.join(''))
var a = [45, 67, 90];
var b = [34, 11, 22,33];
```

```
var c = a.concat(b);
console.log(c)

var details = [34, 56, 89, 90, 100];
console.log(details.includes(99))
</script>
```



Date: 26/07/2021

Pre-defined Methods on Strings:

→ Every string can be created in javascript internally gets created in the form of an array.

→ Following are the pre-defined methods by default supported in javascript which can be applied on the strings:

- length – This is a method that returns the total number of characters within that string including spaces.
- charAt(<index>) - This method returns the characters at provided index position.
- charCodeAt(<index>) - This method returns the ASCII value of the character present at provided index position.
- SubStr(index, length of substring) - This method returns the substring from the given string.
- replace ("what", "with") - This method is used to replace the string parts with required strings.
- split (<" optional Seperator">) - This method is used to split the main string in the form of the array.
- includes ("<string>") - This method checks whether the string that is provided is a part of main string, and returns true/false

- indexof ("string") - This method returns at what the index provided substring exists in main string.
- tolowercase() - This method converts the provided string to lower case format.
- touppercase() - This method converts the provided string into upper case format.
- slice(startindex, endindex) - This method returns a substring from one index to another index.

JAVASCRIPT - "DATE" OBJECT: -

- JavaScript **Date** objects represent a single moment in time in a platform-independent format. Date objects contain a Number that represents milliseconds.
- The Date object is a datatype built into the JavaScript language. Date objects are created with the **new Date ()** as shown below.
- Once a Date object is created, a number of methods allow you to operate on it. Most methods simply allow you to get and set the year, month, day, hour, minute, second, and millisecond fields of the object, using either local time or UTC (universal, or GMT) time.
- "Date" is a pre-defined class supported in javascript using which we could able to work with systems current date or user defined custom data.

  - Syntax:

    var date = new Date (); // creates a reference of systems current date.

    var data = new Date (<user specified date>); // creates a reference of user specified date.

- Date (): It returns presents day's date and time.
- getDate(): It returns the day for the specified date (1-31)
- getDay(): It returns the day of the week for the specified date (0-6).
- getFullYear(): It returns the year of the specified date.
- getYear(): This method returns the year in the specified date.

- getHours(): It returns the hour in a specified date (0-24).

- getMilliseconds(): It returns the milliseconds in the specified date (0-999).
- get Minutes (): It returns the minutes in the specified date (0-59).

- getMonth(): It returns the month in the specified date. This also find the month (0-11).

- getSeconds(): This method returns the seconds in the specified date (0-59)

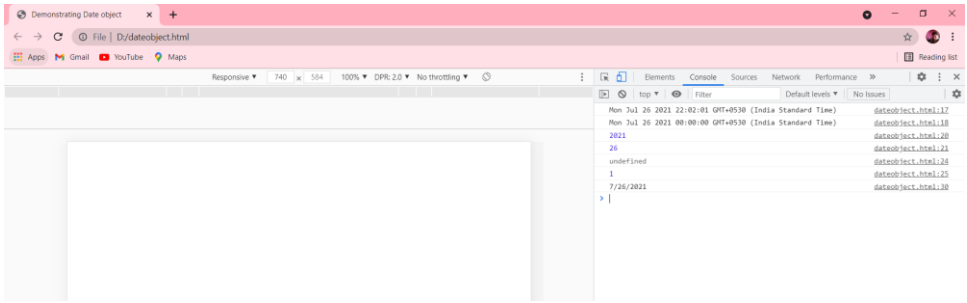
- getTime(): This method returns the date in terms of numeric value as milliseconds.

- setDate(): This method sets the day of the month for a specified date.

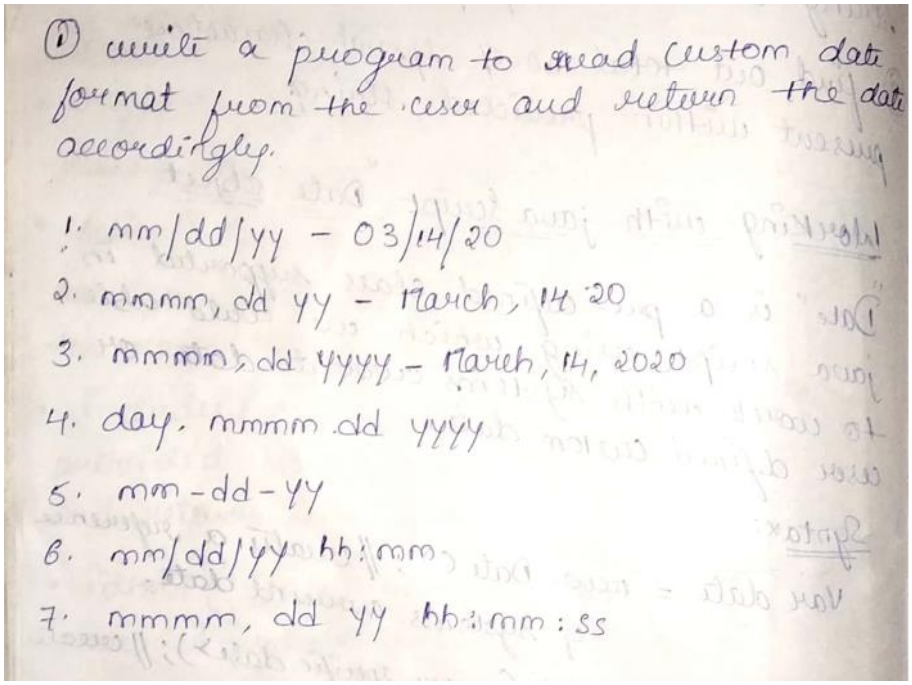
- setFullYear(): This method sets the full year for a specified date.

Example:

```
<script type="text/javascript">
var myDate = new Date ();
var mydob = new Date (" July 26 2021 ")
var months = ['Jan', "Feb", "March", "April", ".."];
var days = ["Sun", "Mond", ".."]
console.log(myDate);
console.log(mydob);
console.log(myDate.getFullYear());
console.log(myDate.getDate());
var monthIndex = myDate.getMonth();
var monthName = months[monthIndex]
console.log(monthName);
console.log(myDate.getDay());
// date = mm/dd/yyyy
                var customDate = (myDate.getMonth() + 1) + '/' +
                myDate.getDate() + '/' + myDate.getFullYear();
console.log(customDate);
</script>
```



ASSIGNMENT:



Date: 27/07/2021

JavaScript functions:

- The set of instructions binded as a module with in "{}" assigned with a user defined custom name which can be reused anywhere within the page is called a function.
- In JavaScript 'function' is a predefined keyword using which we can define a function. [until ECMA6].
- In a single page we can define any number of functions in JavaScript. Once the function ben defined it can be invoked any number of times through the function name.
- While defining the function we name it through function keyword, while calling we simply use function name.

- Local variables and global variables: Any variable which is declared outside of a module will become global variable can be accessible anywhere within the application.
- Global variable can be accessible and modifiable anywhere within the page. Local variable can't be accessed outside the module it can be accessed within the module.
    - Ex:

```
var a = 100 //global variable
Function test ()
{
var a = 101; //local variable
}
```

    - Accessing private data of a function outside of it: Through passing parameters while calling a function.
- Returning value from called function to calling function.
- Both passing parameters and returning values.
    - Passing parameters while calling a function: While calling a function within another function we could able to pass local-variables of calling function to called function.
- While calling a function we can pass any number of parameters.
- While defining a function we could able to catch the values being passed while calling a function, using temporary variables.
- The parameters being passed while calling a function are called as actual parameters whereas the temporary parameter declared at function definition are called formal parameters.
- While passing formal parameters at function definition we don't need to specify or declare those parameters we simply specify the variable names.
- Returning a value from the called function to calling function:
- "Arguments" Keyword:
    - A predefined keyword supported in JavaScript and by default available in every function which holds all the parameter values been passed to that function in the form of an array.
- In JavaScript it is mandatory to catch the parameters in function definition while matching function call with function definition it

only matches function name but never looks for type of parameter or number of parameters being passed.

- Irrelevant of whether the parameters been catched or not, arguments keyword gets created in every function and holds all the parameters

- JSON Data Structure:
  - → JSON is derived as JavaScript Object Notation.
  - → It is predefined data structure being supported in JavaScript by default. It is used to not just hold the data but, holds the data with the extra information of particular data.
  - → The JSON Data gets stored in the format of key and value based.
  - → Ex: key: value, Key2: value2;
  - → All the keys within the JSON Object should be only string data type, whereas the corresponding value could be of any JavaScript supported Data Type.
  - → While accessing data from object we make use of the combination of object name and corresponding key.

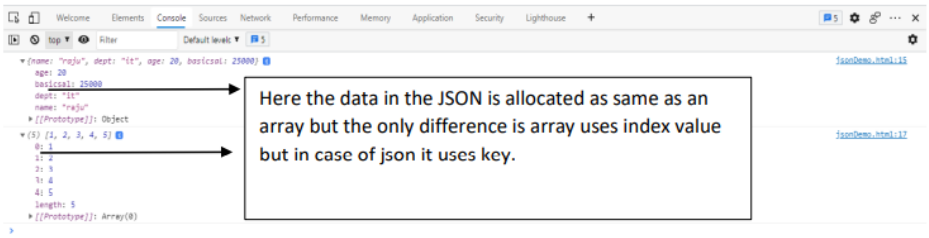
```
ne wrap
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title></title>
    </head>
    <body>
        <script type="text/javascript">
            var empdetails = {
                'name' : 'raju',
                'dept' : 'it',
                'age' : 20,
                'basicsal' : 25000,
            }
            empdetails.hra = 15/100 * empdetails.basicsal;
            empdetails.pf = 25/100 * empdetails.basicsal;
            empdetails.total = empdetails.basicsal + empdetails.hra + empdetails.pf;
            if (empdetails.total > 20000) {
                empdetails.tax = empdetails.total / 15;
            } else {
                empdetails.tax = 0;
            }
            console.log("name" + empdetails.name);
            console.log("dept" + empdetails.dept);
            console.log("age" + empdetails.age);
            console.log("basic salary" + empdetails.basicsal);
            console.log("hra is " + empdetails.hra);
            console.log("pf is" + empdetails.pf);
            console.log("total is" + empdetails.total);
            console.log("tax is :" + empdetails.tax);
        </script>
    </body>
</html>
```

Memory allocation of JSON object:

Internally JSON object gets memory allocated same as like an array where the only difference is, the key of the data is assigned as index value by default.

Ex:

```
<script type="text/javascript">
    var empdetails = {
        'name' : 'raju',
        'dept' : 'it',
        'age' : 20,
        'basicsal' : 25000,
    }
    console.log(empdetails)
    var a = [1, 2, 3, 4, 5];
    console.log(a);
</script>
```



- Anonymous function: Any function which is defined without a name is called an anonymous function. In case we want to define a function, it will be invoked only once within the function do not waste the memory by giving a name to it, we can simply create anonymous function.

- Syntax: function () {
  }

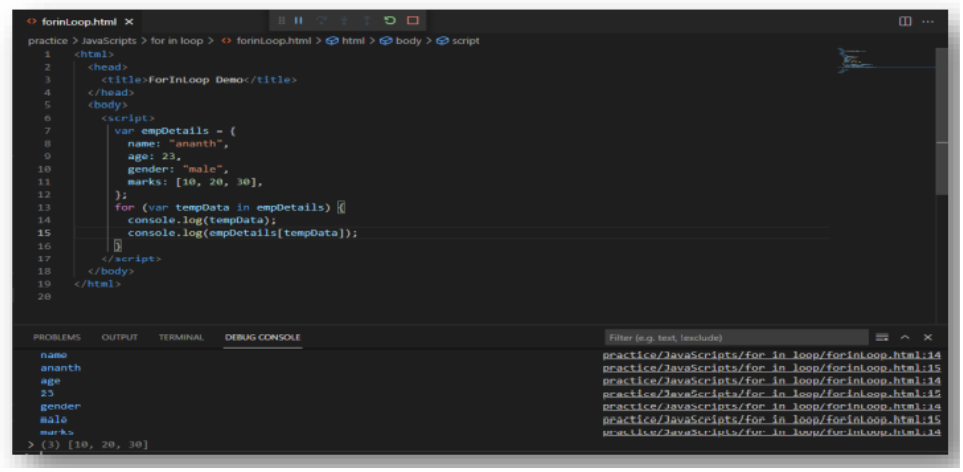
- thisKeyword: A predefined Key word been supported in JavaScript which refers the current object data. In order to access objects data within its corresponding methods, we can access either by using object name or by using this keyword.

Ex:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8" />
    <title></title>
  </head>
  <body>
    <script type="text/javascript">
      var empdetails = {
        name: "raju",
        dept: "it",
        age: 20,
        basicsal: 25000,
        empsal: function () {
          this.hra = (15 / 100) * this.basicsal;
          console.log("HRA is : " + this.hra);
        },
      };
      empdetails.empsal();
    </script>
```

- for in loop Control Structure: A Predefined looping Control Structure through which we could able to iterate over a JSON object.

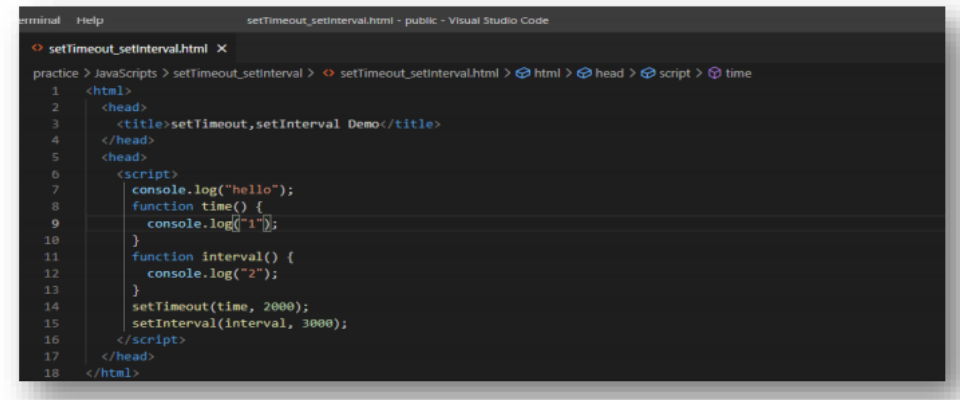
Ex:



JavaScript 'setTimeout' and 'setInterval' methods:

→ A two predefined methods been supported in JavaScript using which we could able to invoke set of instructions not just when the controller reaches to it but, executes the set of instructions only after the provided interval time.

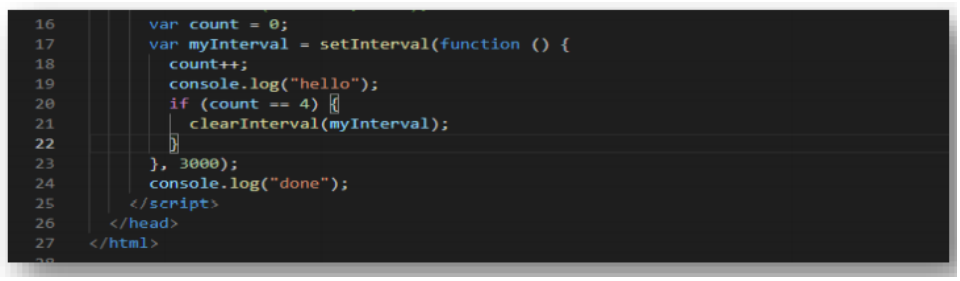
- Syntax: setTimeout(<callbackmethod,<delayvalue>);



- setInterval and setTimeout always works the same where the only difference is setTimeout invokes the function only once after the provided interval time whereas setInterval method invokes the call-back method repeatedly with a gap of provided interval time.

- Note: clearInterval () is a predefined method supported through which we could able to stop a running interval job.

Ex:



```
        var count = 0;
        var myInterval = setInterval(function () {
          count++;
          console.log("hello");
          if (count == 4) {
            clearInterval(myInterval);
          }
        }, 3000);
        console.log("done");
      </script>
    </head>
  </html>
```