Best Front-End Notes (HTML, CSS, JS)

HTML

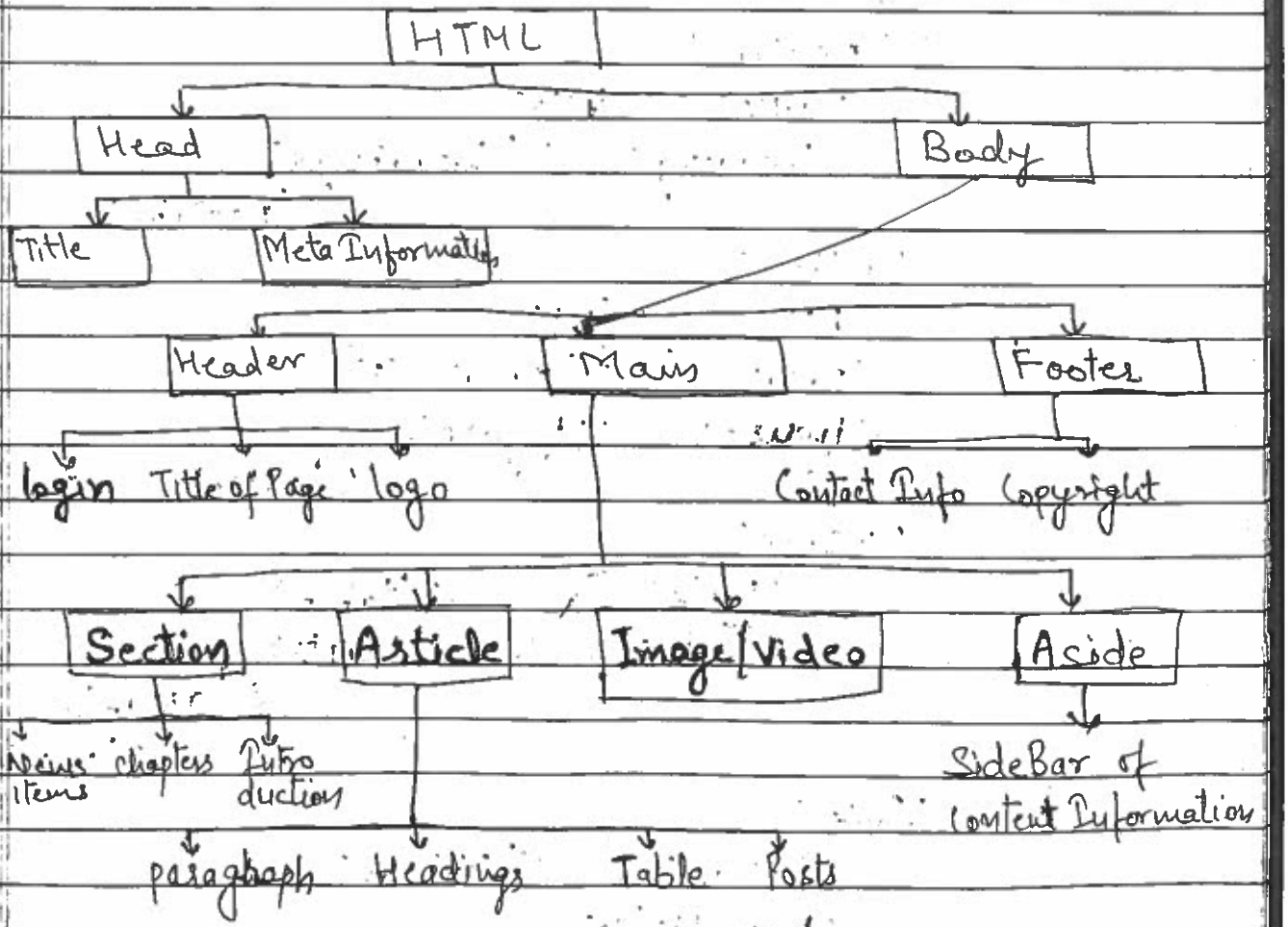
→ HTML :- HyperText Markup Language

In a hypertext document, any word can be "interlinked" to other document.
A text which is used to inter-link with other document.

Markup is what HTML Tags do to the text inside of them; they mark it as a specific type of text.

→ HTML consists of a series of elements/tags"
→ These elements tell the "browser how to display the content."

HTML Page Structure :-



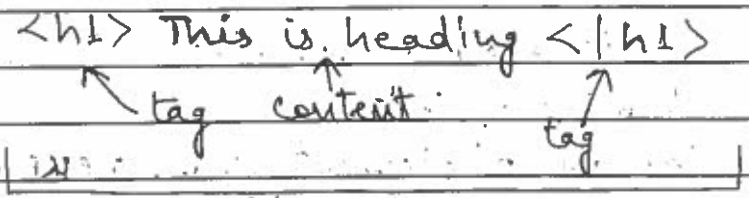
A Simple HTML WebPage :-

```
<!Doctype HTML>
<html>
<head>
<meta>
<Title>  </Title>
                </head>
<body>
<header>
  <img>
  <heading>  </heading>
                </header>
<main>
  <section>
    <lists> </lists>
    <headings> </headings>
                </section>
  <article>
    <h1>  </h1>
    <para>  </para> (<p>)
    <figure>  </figure>
                </article>
  <aside>
    <table>  </table>
    <Hyperlinks> </Hyperlinks>
                </aside>
            </main>
<footer>
  <favicon>  </favicon>
  <hyperlinks>
                </footer>
            </body></html>
```

→ <!DOCTYPE html> = It indicates to the browsers that this is html5 code/document.

HTML Tags :- Tags give meaning to the plain text of HTML.
- → Tags are enclosed with angle braces(<>)
- → Tags can have any number of tags within themselves.
- → Tags start with an "alphabet or underscore (_)"

HTML Elements :- An element is defined by a start tag, some content and an end tag.



→ Empty Element → Some elements doesn't have closing tag. They are called "empty elements" for eg :-

→ <html> :- It is the root element of an HTML Page. It contains all html data and is the starts of an html document.

→ <head> :- It contains the information of the document.

→ <meta> :- It contains more specific information of the document.
for eg :- characters-set, Page Refresh, Viewport

→ <title> :- It provides a title for the HTML document. in the browser's title bar.

→ <body> :→ It contains everything visible on the web Pages, such as Headings, paragraph, Images, tables, Links, lists etc.

→ <header> :- It contains an introduction contents, login details, logo or icon, set of navigational links.

→ <main> :→ It contains section, article., aside etc.

→ <footer> :- It contains an copyright info, contact info related documents., sitemap etc.

→ <P> :- It is a paragraph element and contains blocks of text.
→ When p element ends, next element will appear in next line.

→ Headings :- It is used to mark some content as headings in a HTML webpage.
→ There are 6 levels of headings ←
h1, h2, h3, h4, h5, h6
<h1> :- It is the largest and most important heading.
<h6> :- It is the smallest and least important heading.

→
 :- It defines a single line break.
It is an empty element.

→ Comments :-
Comments are not displayed in the browser.
<!-- Comments -->

→ Attributes :- They provides an additional information about HTML elements.
→ All elements can have attributes.
→ Always specified in the starting tag.
→ Comes in name = "value" pairs

→ Images :- In order to display an image on the webpage, we need to use <img> tag.
It has four main attributes :-

(1) Src :- It specifies the path/URLs of the media.
(2) alt :- It specifies the alternate text for the media.
(3) width & height :- It specifies the width and height of the media in pixels. or in percentage of screen.

for eg :-
<img src = "image.jpg" alt = "image" width = "500" height = "500" >

→ Anchor Tag <a> :- It defines a hyperlink, which is used to link from one page to another.
By default, links will appear as follows :-

(1) An unvisited link is underlined and blue.
(2) A visited link is underlined and purple.
(3) An active link is underlined and red.

An anchor tag <a> has two attributes :-

(1) href = It indicates the link's destination/URL.
(2) target = It indicates where to open the linked documents.

It has four values :-
(i) _self :- opens the link in current window
(ii) _blank :- opens the link in new window.
(iii) _parent :- opens the link in parent frame
(iv) _top :- opens the link in the full body of the window.

for eg :-
```
<a href="URL" target="_self" >click</a>
```

Absolute URLs :- It is used to specify outside links i.e. URL of the web page

for eg :-
```
<a href="https://www.google.com"
   target="_self">....</a>
```

Relative URLs :- It is used to specify local links i.e. inside the root folder.

Four situation arises in this case :-

(1) File is present in the same folder = name of file is provided.
```
<a href="image.jpg">click</a>
```
(2) File is present in the sub folder = name of file with folder name
```
<a href="subfolder/image.jpg">click</a>
```
(3) File is present in the parent folder = use ../ + file name
```
<a href="../image.jpg">click</a>
```
(4) File is present in the subfolder of parent folder = use ../ + folder + file name
```
<a href="../subfolder/image.jpg">click</a>
```

→ Bookmarks :→ Links can be used to create bookmarks, so the reader can jump directly to the specific part of a webpage.

→ Use id attribute to create bookmarks in a webpage and then add the value of id attribute in href attribute of <a> tag to link to the bookmark.

for eg :-

```
<h2 id="abc"> chapter 2 </h2>
<h1> <a href="#abc"> Jump to chapter 2 </a> </h1>
```

→ Image as a Link :- Images can also be used as a link. To do that use <img> tag inside the <a> tag.

for eg :-

```
<a href="URL"> <img src="URL
alt="image"> </a>
```

→ Link to an Email :- Use mailto: inside a href attribute of <a> tag to let user send a mail.

for eg :-

```
<a href="mailto:kumar1908@gmail.com">
send mail </a>
```

→ Lists :- A set of items is called list.
It has 2 types :-

(1) UnOrdered List :- :- A set of items grouped in no particular order.

type = It has type attribute with four values as style = list-style-type :" "
disc(•), circle(○), square(□), none
(Default value).

for eg :- <ul type = "circle">
 Apple
 Banana

(2) Ordered List :- :- A set of items grouped in specific order.

type = Type attribute has five values
"1", "A", "a", "i", "I"
(Default)

Start = Start attribute is used to start the numbering of a list from that start value.

for eg :- <ol type = "1" start = "10">
 Apple
 Banana

(3) Discription Lists :- <dl> :- It is a list of terms and explanation of terms

(4) Discription Term :- <dt> :- used to describle term.

(5) Discription defination <dd> :- used for explanation

<dl> <dt> coffee </dt>
<dd> - Hot drinks </dd> </dl>

→ Tag = It defines a block-level element/ section and used as a container for other HTML elements.
→ By default, browsers place a linebreak
 after and before the element.
→ It is often used together with CSS for layout.

→ Semantic Elements = An element which clearly describe its meaning. and
→ used to define different parts of a webpage.
→ or to structure a webpage in well manner.
There are many semantic elements :-

1) <nav> tag :- Used for navigation bar and place on the top of a webpage.
2) <header> tag :- Used for logo, login details, Main headings.
3) <footer> tag :- place at the end of a webpage and used for personal details, copyright, sitemap etc.
4) <section> tag :- defines a section in a webpage.
5) <main> tag :- used for the main content of a webpage.
6) <figure> tag :- used for image, diagrams, Illustrations.
7) <figcaption> tag :- used to write caption of the figure
for eg :- <figure> <img src="URL">
<figcaption> An Image Caption </figcaption>
</figure>
8) <article> tag :- specifies independant, self contained content.
9) <aside> tag :- It is placed in sidebar and used for some content aside from main content.
10) <summary> tag :- used for a summary of a document.
11) <time> tag :- used to represent specific time with datetime Attribute
12) <mark> tag :- used to highlight a text.
13) <details> tag :- used in summary tag to define details of a topic.

→ Block Elements :→ It always starts on a new line and takes up the full width available on the webpage.
→ It actually blocks the other elements to sit next to it on left or right.
for eg:- <P>, , <h1 to h6>, <nav> etc, <li>, etc

→ Inline Elements :→ It does not start on a new line and take up as much space or width needed to display element.
for eg:- <img>, <time>, <mark>, etc.

→ Text Formatting Elements :→ They are used to format the appearance of the text on your webpage.
There are many text formatting elements:-

1) <b> tag :→ used for bold text. (inline element)
2) <i> tag :→ used for italics text. (")
3) tag :→ used for bold and strong importance text.
4) tag :→ used for emphasized text in italic.
5) <small> tag :→ used to reduce the size of the text.
6) <u> tag :→ used for underlined (_____) text.
7) <ins> tag :→ used for inserted text by underlying (_) the text.
8) <del> tag :→ used for striking through the text. Text
9) <s> tag :→ same as <del> tag.
10) <mark> tag :- used for highlighting a text.
11) <sub> tag :- used for subscripted text. H_2O
12) <sup> tag :- used for superscripted text. $a^2 + b^2$

24) <hr> tag :- used for horizontal line in webpage.

13) <q> tag :⇒ used for a short quotation. (" ")
14) <cite> tag :⇒ similar to tag & used for cited text.
15) <pre> tag :⇒ used to define a text as written in HTML code (extra spaces)
16) <kbd> tag :- used to define keyboard text. [ctrl] + [swift] + [R]
17) <var> tag :- used for variable text. 1/2 x b x h.
18) <code> tag :- used for computer code. push()
19) <abbr> tag :- used for abbreviation or shortform of a text.
It has "title" attribute. Use full form of a text in the value of title attribute.
20) <bdo> tag :- "Bio-Directional Override". Used to change the direction of a text. right to left
It has "dir" attribute and the value is "rtl"
21) <address> tag :- Block element. Used for contact information and text will be in italic.
22) <blockquote> tag :⇒ used for a section that is taken from other source.
Always use "cite" attribute to mention the URL of the source
23) <samp> tag :⇒ used to define sample output of computer program and in "monospace" font.

→ Special Characters or HTML Entities :-
To display special characters in webpage, we use entities that "begins" with ampersand (&) and "ends" with a semicolon (;) and in between is hexcode or entity name
for eg :-
(1) Character Entities :-   (space), < (<), ' (')
(2) Diacritical Entities :- à (à), ô (ô)
(3) Mathematical Symbols :- ∑ or ∑ (∑)
(4) Some other Entities :- ← (←) &hearts (♥)

→ Table :→ Tables are used to show the "tubular data"

1) <table> tag :- It is used to "create a table".
2) <tr> tag :→ It represents the "rows of a table".
3) <td> tag :→ It represents the "data of cells" in table-rows.
4) <th> tag :→ It represents the "table-header".
5) <caption> tag :→ It is used just after the <table> tag to mention the "heading of the table".

Attributes

→ <table> tag has "border attribute" to mention the "border of the table and each cells" of borders table.

→ Value of Border is "numeric (1,2,3...)" to mention the "thickness of border"

→ <th> tag has two attribute :-

(1) colspan :- You can have a header that spans over two or more columns

Name		Age
X	Y	19
A	B	20

(2) rowspan :- It is used to spans over two or more rows.

Name	X	Y
Age	11	21
	12	22

3 * Styling of table will be discuss in CSS.

HTML Forms :→ Forms are used to collect different kinds of user input.
It has many elements and attributes :-

→ <form> tag :- It allows user to enter information like textfields, textarea fields, checkboxes, drop down menus, radio and submit button.
It has 5 attributes :-

1) Action :→ It is used to send user data to the server when user submits a button.
for eg :-
<form action = "default.php"> </form>

2) Method :→ It specifies the HTTP method to be used when submitting the data.
It has 2 values :- get and post.
→ get value is used to show the form data in the address bar of new window.
→ post value is used to not to show the form data in the address bar of new window.

3) Autocomplete :→ It has ON & OFF values.
When it is ON, the browser auto complete the values based on the values that the user has entered before.

4) Novalidate :→ It has no values. It specifies that the values not to be validated when submitted.

5) Target :→ It is used to specifies where to display the received response after submitting form
It has 4 values :- _blank, _self, _parent, _top.
for eg :-
<form action = "default.php" autocomplete = "ON" novalidate method = "get" target = "_self"> </form>

→ HTML <form> element has 10 elements :-

1) <label> :→ It is used to "mention the name of detail" that has to be enter by user.
→ It has "for attribute" whose value is same to the "id attribute" of the "corresponding tag".
When 'user clicks' on <label> tag (name of detail) it will focus on the 'textfield' respected of name.

2) <input> :→ It specifies an "input field" where user can "enter data."
It has many attributes & covered in next chapter

3) <select> :→ It defines a "drop-down list."
It has 4 attributes :→
→ id attribute :- It has same value of for attribute of their <label> tag.
→ name attribute :- This "name is send to the server", mostly same as id attribute
→ Size attribute :- It "specifies the number of visible values/options."
→ multiple attribute :→ It has no value.
"It allows to select multiple options"

4) <option> :→ It always comes after the <select> tag.
It "defines an option that can be selected."
It has 2 attributes
→ value attribute :→ It has "name of the option will send to the server"
→ Selected attribute :→ It has no values. and defines a pre-selected option.

5) <textarea> :→ It defines a multiple line input field.
It has 3 attributes :-

→ name attribute :→ Used to send this name to the server
→ rows & cols :→ Used to specify the number of rows & columns

It can have id attribute to connect with its label.

6) <button> :→ It uses for clickable button.
It has a "type attribute" whose value is submit.
2 Attributes :- { Autofocus and onclick = "alert(' ')" }

7) <optgroup> :→ Defines a group of related options in drop-down list.

It is used in <select> tag and before <option> tag.
It has "label attribute" to define the groups in drop-down list.

8) <fieldset> :→ It is used to group related data in a form
it draws a box around a group of related data.

9) <legend> :→ It defines a caption for <fieldset> tag
It comes just after the <fieldset> tag

10) <datalist> :→ It is similar to <select> tag with addition of we can type a option which is not in list.

It has id attribute of same value to the list attribute of <input> tag.

for eg :-

```
<label for="car-select"> Select a car </label>
<input type="text" list="cars">
<datalist id="cars"> <option value="BMW"> </datalist></input>
```

→ HTML <input> Element :→ This element can be displayed in many ways depending on type attribute.

Values of type attribute :-

1) text :→ defines a single line text input field.

2) radio :→ defines a "small circle radio button to select one option."

→ With radio value of type attribute of <input> tag we have to use id, name & value attribute

for eg :-

```
<label for="lang"> Fav language </label>
<input type="radio" id="lang"
    name = "Fav lang" Value = "HTML "><br>
<input type="radio" id="lang" name = "Fav lang"
    value = "CSS"> CSS <br>
```

→ name attribute should have the "same" value in all "input tag of radio type"

3) checkbox :→ defines a square checkbox to select one or more option.

It also use id, name & value attribute in input tag

→ name attribute should have the "different" value in all "input tag of checkbox type"

4) range :→ defines a "slides to select number value"

It has min, max, step attribute with "range type <input tag"

→ min = Min. range value in Numbers
→ max = Max. range value in Numbers
→ Step = No of steps to change the slider value

5) date :→ is used to choose a date from calender. It has min & max attributes with date input.
→ min = Y-M-d or YYYY-MM-DD date will start from min.
→ max = YYYY-MM-dd date will be before max date.

6) month :→ specifies a month and year

7) time :→ specifies a time. It can have min and max attribute with time value value attribute to set default time value.

8) datetime-local :→ specifies a date and time, in no timeZone

9) color :→ is used select a color.

10) password :→ is used to take password from user. It can have required attribute.

11) week :- is used to select a week and year.

12) email :- is used to take email iuput data.

13) url :- is used to take URL address data.

14) file :- is used to browse a file or choosefile button.

15) reset :- is used to reset the form values

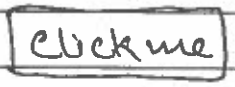
16) number :- is used to take numeric data input. It can have min & max attribute to restrict the numbers.

17) tel :- defines some space for a telephone No. It can have placeholder attribute whose value is "telephone number format."

18) Search :- defines a "rectangle search space".

19) Submit :- defines a Submit button. It can have value attribute to "change the name of button."

for eg :-



20) image :→ displays an "image as submit button". It has four attributes
→ Src = URL of the image. (path)
→ alt = alternative text when image is not seen.
→ width = width of an image.
→ height = length of an image.

HTML <input> element's Attributes :→

1) type :- We have studied type attributes & its values.

2) id :- species the unique id to the input whose value should same as for attribute of label tag.

3) name :- Used to "submit the name of data to the server"

4) value :- → reset & submit :- to change text on button.
→ text & password :- to set default values
→ checkbox, option, radio & button :- to defines value that is sent to server on submit.

6) readonly :- input field space can not be modified.

5) disabled :- input field space will be uneditable.

7) size :- specifies the visible width of the input field space in numbers (like 50)
It works with text, ~~password~~, URL, email, tel, search and password.

8) maxlength :- specifies maximum no. of characters.

9) min & max :- specifies min. & max. values.
It works with number, range, datetime, month, date, time and week.

10) required :- space must be filled before submit.

11) autofocus :- field space get focused on the page loads

12) width & height :- specifies the width & height of image button

13) multiple :- use to enter more than one value.

14) placeholder :- describe the format of expected value.

15) steps :- Use to fix the number of steps. It works with numbers, date-time, range, date, week, month.

16) autocomplete :- use to autocomplete the input field.

17) list :- It is an attribute of ~~autocomplete~~ element. Its value should be same as of id attribute of <datalist> "pre-defined options in <input>"

→ HTML Multimedia :→

Sound, music, videos, animations and images are multimedia on the webpage.

1) <audio> tag :→ It is used to play an audio file on the webpage.

It has 3 attributes and 1 element as follows :-

→ controls :→ use to play, pause and stop.

→ muted :- use to mute an audio by default.

* <source> tag :→ It is used to provide the "path and type of audio" to browser.

It has 2 attributes :-

→ src :→ used to provide the path/URL of an audio.

type :→ used to provide the type of an audio.

Ex:-
```
<audio controls muted>
<source src="audio.mp3" type="audio/mp3">
</audio>
```

2) <video> tag :→ It is used to play a video file on the webpage

It has 4 attributes :-

→ controls :- use to play, pause and stop a video

→ width & height :- defines the width & height of a video

→ Autoplay :- enables an autoplay on a video

→ It can have <source> element to describe the type and path of a video.

→ Similar to <audio> tag.

(Youtube Video)

3) <iframe> tag :→ It is used display a webpage within a page web" for eg. Youtube.
It has 4 attributes :→

→ src :→ used to specify the URL path of a webpage
→ title :→ used to specify the description of iframe.
→ width & height :→ defines the width & height of iframe

HTML <head> tag :→ It is "container of data" about a document (webpage) and placed b/w <html> and before <body> tag.
It has 5 elements as follows :-

1) <title> tag :- It defines the title of a webpage in browser's title bar."

2) <style> tag :- It is used for "internal CSS" of a page. Single

3) <Link> tag :- It defines the "relation b/w current webpage and external resource" (External CSS).
It has 2 attributes :-
→ href :→ "location/path of external resource (css)".
→ rel :→ name of external resource (stylesheet).

4) <script> tag :→ It is used to define "Javascripts"
It can have src attribute to tell the path/location.

5) <base> tag :→ It is used to tell the "base URL and base target for all relative URLs" in a page with the help of href and target attribute.

6.) <meta> tag :→ It is used to define the "metadata of a webpage." It is important for SEO."

It has 4 attributes :-

1) name :→ specify the name of the metadata. It has four values.
→ author :→ name of the author of a webpage.
→ description :→ description of a webpage.
→ Keywords :→ comma seprated keywords for SEO.
→ viewport :→ specifies the control of viewport on different devices.

* name attributes always comes with content attribute

ii) content :→ gives the "value associated with name attributes".

foreg :-

<meta name="author" content="Dinesh">

iii) charset :- defines the "charater encoding for the Page"

Value = UTF-8

iv) http-equiv :- used to "refresh the page" in specified time in content attribute. (in seconds)

foreg :-

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

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

HTML Favicon :- An icon used in the title bar of the browser.

→ Use <link> tag to add icon on title bar.
It has 3 attributes :-

→ href = 'location/path of the icon image'
→ type = 'type of the icon/image.'
→ rel = relation of the icon. (rel = "icon")

CSS

→ CSS :→ Cascading style sheet

→ It describes how HTML elements are to be displayed on screen.

→ CSS can control the layout of multiple web pages all at once.

→ It is used to add color, fonts, layouts, etc.

→ Syntax for CSS :→

selector { properties ; }

(2)

(1) → Comments in CSS :→ use /*comments*/ to comment.

→ Adding CSS to HTML page :-

1) inline CSS :→ Inline CSS is directly applied to the HTML element using style attribute. It is used to apply CSS for a single element.

for eg :-

```
<h1 style="color:red; font-size:40px">
    inline CSS</h1>
```

2) internal CSS :→ Internal CSS is applied using <style> element inside the <head> element. It is used to apply css to "all the same elements at once."

for eg :-

```
<style>
    h1 { color: red;
         font-size: 40px;
    }
</style>
```

3) External CSS:- It is similar to internal stylesheet(CSS), but it is applied using a complete different file with ".css" extension. and it is linked to HTML using <link> element.

for eg:-

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

(2) → Selectors :→

Selectors are used to point HTML element which we want to style."

1) Element Selector :- It selects all elements with same name."

Syntax :- element name { properties; }

2) Class Selector :- Class is attribute used in element's opening tag. We can have multiple values of class separated by space.

→ The class selector selects all elements with same class attribute. Use dot(.) & class.

for eg:-

HTML= `<p class="abc">I am para.</p>`
`<h1 class="abc xyz">I am heading</h1>`

CSS=
```
.abc { color: red; }
.xyz { font-size: 40px; }
```

→ It will change the color of both <p> and <h1> element because both have same class("abc") attribute

→ font-size will change <h1> element as it has "xyz" class attribute.

3) Id selectors :→ It selects only one element whose has a specific id attribut
→ To select element with id attribute use hash (#) and id name.
→ There can be only one id in HTML tag. This id is unique in a HTML page.

for eg :-
HTML= <h1 id="one"> Heading1 </h1>
<h1 id="two"> Heading2 </h2>

CSS= Element Selector h1 { color: red; }
Id selector #one { font-size: 20px; }
#two { font-size: 40px; }

→ Here color of both <h1> will be red as we have selected <h1> element.
→ id="one" is applied only to first <h1> so the font-size of first <h1> will be 20px.
→ id="two" is applied only to second <h1> so the font-size of second <h2> will be 40px.

4) Grouping Selectors :→ We can apply CSS to multiple elements at once with combination of tag, class-name or id separated by comma.

for eg :-
p, h1, div { properties }

#id, .class, span { properties }

☆ Preference Order :- inline > id selector > class selector > element selector > property defined in last

5) Nesting Selector :→ It is used to apply CSS to the HTML element which is inside the specific HTML element.

→ To use this, use space b/w selectors such that the sequence represents a hierarchy, starting from top.

for eg:-

HTML code

```
<ol type="A">
    <li> Mango <li>
    <li> Banana <li>
</ol>
<ul type="square">
    <li> Appli <li>
    <li> Guhava <li>
</ul>
```

CSS

```
ol li { color: red;
        font-size: 40px; }
```

Here CSS will apply "only to those <li> whose are inside <ol> element."

6) Chaining Selector :→ We can have same class for multiple elements and we want to apply CSS to one/some of them, then use chaining selectors.

→ To do this "use the element name with class name without any space."

for eg :-

```
h1.class-name { properties; }
```

(4)

→ CSS colors :→ It is used to set the colors of element's content and its decoration.

It can be specified in 6 ways :-

1) By name :- plain and solid color.
{color : red;}

2) By rgb :→ Combination of red, green and blue.
{color : rgb(000, 000, 000);}
0 to 255

3) By rgba :→ Same as rgb with "opacity (0.0-1)"

4) By hsl :→ Hue - a degree on the color
0 = Red, 120 = green, 240 = blue
Saturation - in %age. 0% = light gray 100% = full color
Lightness - in %age. 0% is black 100% = white

5) By hsla :→ a - opacity in 0, 0.1 -- 0.9, 1 value
{color : hsla(255, 0, 0, 0.6);} Red color

6) By hex :→ hexadecimal code of 6 digits.
Starts with #(hash).

{color : #00 00 00 ;} ~~Colors~~ Black color

Red Green Blue

Values = 00 to FF

{color : #FFFFFF ;} ~~Black~~ White color

(4) → CSS units :→ It expressed the size of length.

for eg :- height, width, margin, padding, fontsize, lineheight

1) Absolute :- There are fixed length values and not recommended."

values :- px, cm, mm, in(inch), pt(points)

{font-size = 16px;} 1px = 1/96 inch

2) Relative :- specify a length relative to another length property."

It has 6 values :-

(i) % percentage :- relative to the parent element.

for eg :

HTML
```
<div id="div1">
    <div id="div2"></div></div>
```

CSS
```
#div1{ height : 100px;
       width : 200px; }
#div2{ height : 50%.      ⇒ 50% of 100px = 50px
       width : 50%. }
```

(ii) vw & vh :- relative to the window's width and height

1 vh = 1% window's height

100 vw = 100% window's width

(iii) rem :- relative to the root element (<html>).

By default length of <html> is 16px.

∴ 1 rem = 16 px

2 rem = 32 px

(iv) em :→ relative to the font-size of parent element.

∴ If font-size of parent element is 20px then '1em = 20px'.

) → CSS Borders :- It is used to set element's border. It has 4 properties :-

(i) border-style :- solid, dotted, dashed ... etc.

top right bottom

(ii) border-width :- Specifies the thickness of border.

(iii) border-color :- specifies the color of the border.

(iv) border-radius :- used for adding rounded border.

☆ We can apply border properties to all 4 sides.

for eg :- property : top right bottom left
= border-width : 1px 2px 3px 4px

⇒ Observe the border drawn above.

☆ Border :- shorthand property of width, style & color.

border : 2px solid black

17) overflow-wrap :- used to wrap the big letters
break-word.

16) Vertical-align - used to set the vertical alignment of the text. text-top, text-bottom, baseline
useful in display: inline-block

⑥ → CSS Fonts & Texts :- It is used to change the style. text-font.

1) font-size :- specifies the size of the text.

2) font-style :- specifies the style of the text. values are normal, italic, oblique etc.

3) font-weight :- specifies the bold text. bold, bolder, 100-900

4) font-family :- specifies the font of the text. Times New Roman, Serif etc.

5) font-varient :- changes all lowercase letters into uppercase letters. (small-caps)

6) color :- use to set the color of the text.

7) text-align :- used to set the alignment of the text. left, right, center, justify.

8) text-align-last :- used to set the last line of text.

9) line-height :- used to set the space between lines.

10) word-spacing :- used to set the space between words.

11) text-transform :- used to set uppercase, lowercase & capitalize.

12) text-decoration :- used to set the underline to the text. color, thickness, style of the underline.

13) direction :- used to set the direction of the text. rtl

14) unicode-bidi : bidi-override ; ये याद करलो ऐसे ही

15) text-indent :- used to set from where the first line start. values in pixels.

(8)

→ CSS Background :→ used to specify the background of an element.

1) background-color :- used to set the background-color of an element.

1.1) opacity :- 0.1 to 1. controls the transparency of bg color.

2) background-image :- used to set the background image of an element.

{background-image : url("image.jpg") ; }

3) background-repeat :- By default, a bg image is placed at the top left corner. and repeated both vertical & horizontal.

values :- no-repeat, repeat, repeat-x, repeat-y

4) background-position :- used to set the position of background image.

values :- left, right, center, top, bottom.

5) background-size :- Used to set the size of bg image.

values :- Xpx Ypx, cover, contain, auto

(9)

6) background-attachment :- Used to set whether a background image scrolls with the page or is fixed.

values :- scroll, fixed, local.

7) background :- It is the shorthand property of

color image position repeat attachment

(8) → CSS Margin :- "Space outside the border."
Border has four sides :- top, right, left, bottom.
Values :- (1) auto for horizontally center an element.

Margin is the shorthand property of :-
margin-top, margin-right, margin-bottom, margin-left.

for eg :-
{margin : 25px ;} 25px for all sides.
{margin : 25px 35px ;} 25px for top & bottom
35px for right & left.
{margin : 25px 35px 45px ;} 25px for top
35px for right & left
45px for bottom
{margin : 25px 35px 45px 55px ;}
top right bottom left

★ "Vertical margin of two elements will be collapsed and not added." (Larger of two will be considered)
"Horizontal margin of two element's will will be added."

(9) → CSS Padding :- "Space outside the ~~border~~ element and inside the border."

→ Values are same as the margin.

padding-top, padding-right, padding-bottom, padding-left.

Padding will increase the width of an element so → property so the width of the element will not increase.

★ → Use box-sizing : border-box

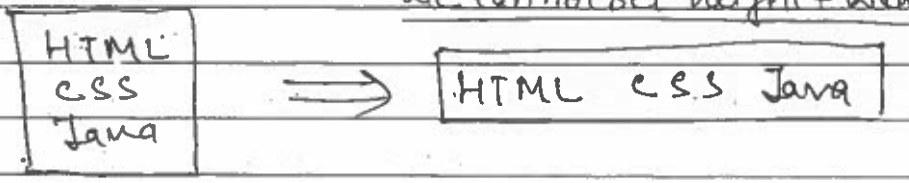
(10) → CSS Display :- It specifies how an element is displayed (Block or Inline)

It has following values :-

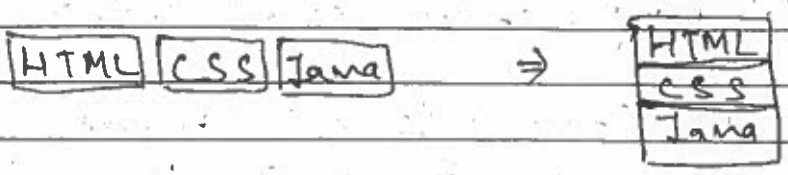
1) display : none :→ It will remove the element and does not take the space in web page.

1.1) visibility : hidden :→ It will remove an element and but that element will take space in webpage.

2) display : inline :→ It is used to convert block element into inline element. We cannot set height & width of inline element.



3) display : block :→ used to convert inline element into block element.



4) display : inline-block :→ used to convert block element into inline element. But here we set the height and width of inline element.

5) ~~display : list-item~~

(11)

(11) → CSS Position :- It is used to set the position of an element.

1) Static :- By default position of an element.

2) relative :- An element's position is relative to its normal/by default position. Now you can change the position of an element (using left, right, top & bottom) from its original position.
* It will create the white space in the webpage.

3) absolute :- position is relative to its parent element. Use left, right, top & bottom to change the position.
* It will not create the white space in the webpage.

* To use this, first we need to change the position of its parent element to relative.

4) fixed :- use to stick the position of an element to the window. Maintains top, Bottom, right & left.

5) sticky :- It is a combination of relative and fixed. First an element will behave like a fixed position upto the height of its parent element. and after scrolling the webpage it will behave like a relative position.
* It will work only if you have the left, right, top & bottom value.

→ CSS Float :- It is used for positioning an element inside a container. (14)

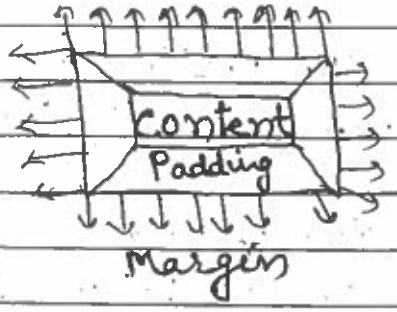
1) left :- The element floats to the left of its container.

2) right :- The element floats to the right of its container.

3) none :- By default position

The CSS clear property specifies what should happen with the element that is next to a floating element.
value should be same as float.

CSS Box Model :- It is a box around HTML element. It contains an element's content, padding, border, margin.



(15)

Imp :- When you set the height and width of an element, you just set height and width of the content. To cal. height & width of an element :-

Height :- Top + bottom padding + Top + bottom Margin + Top & bottom border + height of the content

Width :- Same as height

(14) → CSS Height and Width :-

It is used to set height and width of block element's to a specific size.

→ The size will be fixed and this creates a "problem in smaller devices" and the browser gets a "scrollbar to scroll through the element". To overcome this use max-width property.

1) Min/Max Width :-

To set min/max width, first set the width of an element in relative units (%, em, rem). It actually makes an element resizable with min and max. width.

Min-width :- It specifies the min. width that the element can have.

Max-width :- It specifies the max. width that the element can have.

2) Min/Max Height :- It specifies the min and max height that the element can have.

(15) → CSS Overflow :- It specifies what happens if the content's height and width is larger than element's height and width.

1) visible :- By default value.

2) hidden :- The content that fits inside the box is visible and the overflow content is hidden.

3) Scroll :- It will add a scroll-bar in a box and content will be visible through scrolling in a box.

4) auto :- If content overflows a scroll-bar will gets added in a box.

5) overflow-X :- used to add scrollbar horizontally.

6) overflow-Y :- used to add vertical scroll-bar.

16) → CSS Advance Selectors :-

1) Nesting/Descendant Selector :- To select an element which is inside the specific/parent element.

for eg:- ——— (1)
<P> ——— </P> ——— (2)
<P> ——— </P> ——— (3)
<section> ——— (4)
<P> ——— </P> ——— (5)
</section> ——— (6)
 ——— (7)

→ div P { css: } It selects all <P> elements which is inside the element.
div section P{css:}

2) Child Selector :- To select an element who is direct child of specific/parent element.

→ div > P { css; } <P> is direct child of line 2 & 3 and remember <P> inside <section> is not direct child of . (line-5)

3) Adjacent sibling selector :- To select an element who comes immediate after an specific element.

for eg:-
```
<div>
<P> ——— <P>
</div>
<p> ——— <H1>
<h5> ——— <h1>
<P> ——— <P>
```
① ② ③ ④ ⑤ ⑥

→ div + h1 { css; } It selects line-4 <h1> element as it comes immediate after the

4) General sibling selector :- To select all elements comes after an specific element.

→ div ~ h1 { css; } It selects line-4 & 5 all <h1> elements who comes after element.

(17) CSS Attribute Selectors!-

1) element [attribute] :- To select an element who has a specific attribute specified in square braces.

for eg:-
```
<img src="image.jpg" alt="image">
```
→ img [src] { css } It selects <img> element which has "src" attribute.

2) element [attr = "value"] :- To select an element who has a specific attribute and its value.

→ img [src = "image.jpg"] It select an element <img> which has "src" attribute and its value is "image.jpg"

3) element[attr^="value"] :-

<img src="cat.jpg" alt="image of cat">

→ img[alt^="image"]{css;} It selects an element who has specific attribute and its ~~value~~ starting value.

4) element[attr$="value"] :- It selects an element who has an specific attribute and its ending value

→ img[alt$="cat"] :- "cat" is ending/last value of "src" attribute.

5) element[attr~="value"] :- It selects an element who has an specific attribute and its value should contain the specified value.

→ img[alt~="of"]{css;} :-
alt attribute and its value contains [of].
image of cat.

Not Img
6) element[attr|="value"] :- It selects an element whose specific attribute & its value contains hyphen (-).

(17) CSS Pseudo-Class Selectors :-

1) :first-child :- It selects every specified element which is first child of its parent element.

foreg :-
 — ——— (1)
 — ——— (2)
 — ——— (3)
 ——— (4)
 ——— (5)
 — ——— (6)
 — ——— (7)
 ——— (8)

li : first-child { CSS; } :- It selects line-2 & 7. As they are first child <li> element of their parent element.

2) : last-child :- It selects every specified element which is last child of its parent element.

li : last-child { CSS; } :-
It selects line 4 & 8. As they are last child <li> of their parent element.

3) : nth-child(n) :- It selects every specified element which is nth-child of its parent element.

li : nth-child(2) { CSS; } :- It selects line 3 & 8. As they are 2nd child element of their parent element.

4) : nth-of-type(n) :- It selects the nth child specific element.

for eg :-
 —— —— 2
 —— —— 3
 —— —— 4
 —— 5
 —— 6
 —— —— 7
 —— —— 8
 —— 9

→ li : nth-of-type(2) {css;} :- It selects line 3 & 8. As they are 2nd child <li> element.

5) : nth-last-child(n) :- It selects nth last child element of its parent element.

→ li : nth-last-child(2) {css;} :- It selects line 3 4 7. As they are 2nd last child <li> element of its parent element

6) : nth-last-of-type(n) :- It selects the nth last child specific element.

→ li : nth-last-of-type(2) {css;} :- It selects. As they are 2nd last child of <li> element.

7) :only-child :- It selects an element which is only child element of its parent element.

→ h1 : only-child :- It selects <h1) element if its the only child element of its parent.

8) :only-of-type :- It selects an element if this type of element is only one in its parent element.

for eg :-

 —— 1
<h1> —— </h1> —— 2
 —— —— 3
 —— —— 4
 —— 5

→ h1 : only-of-type {css;} :- It selects line 2. As <h1> type element is only one in element.

9) :first-of-type :- It selects the first element of specific type in its parent element.

→ p : first-of-type {css;} :- It selects line 3. As is first only element of type element in its parent element .

10) :last-of-type :- It selects the last element of specific type in its parent element.

→ p : last-of-type {css;} :- It selects line 4. As is last element of type element in its parent element .

11) : empty :- It selects an element only if its content is empty. (18)

for eg :-

⇒ div : empty :- The content of element is empty.

12) : not( ) :- It selects all elements except the specified in brackets. You can specify id, class.

for eg :-
 Hello —— 1
 Hello2 —— 2

⇒ p:not(#abc) {css;} :- It select line 1. Line 2 is not selected as its attribute id is given in brackets

13) : hover :- It use to change the style of specified element on hovering the mouse to it.

⇒ a:hover {css;} :- When you hover the mouse on link, its style will change

14) : link :- It is used to select the unvisited links. (19)

15) : visited :- It is used to select the visited links.

16) : active :- It is used to select the active links.

17) : target :- It is used to create an animation effects on links

(18) → CSS Pseudo-Element Selector (::) :-

1) ::first-line :- It used to select the first line of element

2) ::first-letter :- It is used to select the first letter of element.

3) ::selection :- It is used to change the style of selected content in the webpage. (* select something in the website to copy.)

4) ::marker :- It is used to select the markers in list-items.

Imp 5) ::after :- It is used to add content(text) after an element. Must use content property to add text.

→ /* p::after { content = "Remember"; } */

Imp 6) ::before :- It is used to add content(text) before an element. Must add content property to add text.

→ /* p::before { content = "Remember"; } */

(19) → CSS Pseudo-class on Form selectors :-

1) :focus :→ works with <input> tag of <form> element. It is used to change the style of <input> element only when a user clicks on an element on the webpage

input:focus { CSS; }

2) :checked :→ works with input[type="checkbox/radio"] (20)
It is used to change the style of selected option's icon.

input[type="checkbox/radio"]:checked {css;}

3) :disabled :- used to change the style of disabled input.

input:disabled {css;}

4) :enabled :→ used to change the style of enabled input. <input> elements are by default enabled.

input:enabled {css;}

5) :required :→ used to change the style of <input> element which is required.

input:required {css;}

6) :optional :→ used to change the style of <input> element which is optional. (21)

input:optional {css;}

7) :in-range :→ works with <input type="number">. used to change the style of range.

input[type="number"]:in-range {css;}

8) :read-only :→ used to change the style of <input> element is read-only.

input:read-only {css;}

(20) → CSS Transition :→ It is used to change the style of ~~an element~~ smoothly when a user hover on the element.

It takes four values :-

(1) transition-property :→ on which property you want to apply transition.

(2) transition-duration :→ specify the time need to complete the transition in seconds.

(3) transition-timing-function :- specifies the speed of transition effect.

value :- ease, linear, ease-in, ease-out, ease-in-out, steps(number, end).

(4) transition-delay :- specifies the delay time in seconds.

for eg :-

```
transition: height 2s ease 2s;
```

(21) → CSS Flex :- It is a property of display. It makes easier to design flexible and responsive layout of webpage.

★ display:flex; property is used for outer or container. It comes with 5 more properties which also used on container.

1) flex-direction :- defines the direction of inner elements of container :-

row(→), row-reverse(←), column(↓), column-reverse(↑)

2) flex-wrap :- used to wrap inner elements of the container. (रपेट-ता)
wrap (⇉), wrap-reverse (⇇), no-wrap (default)

3) flex-flow :- shorthand of flex-direction & flex-wrap

4) justify-content :- used to align the inner elements of container in X-axis or horizontally.
center, flex-start, flex-end, space-around, space-between, space-evenly.

5) align-items :- used to align the inner elements of container in Y-axis or vertical.
flex-start, flex-end, center, baseline, stretch (default)

(25)

→ Properties of inner elements of flex-container

1) order :- It defines the order (in number) in which inner elements are placed in a container.

2) flex-grow :- It is used to increase the width of inner elements with respect to outer or container.

for eg :-
If a container has 500px width. and it has 2 inner div with 200px width. Then both the div will increase its width to 250 px. We can give value in number also. from 0 to any numbers.

जितने element की value ज्यादा होगी उसकी width ज्यादा होगी

3) flex-shrink :- It is used decrease the width of inner elements w.r.t outer container. Values will be given in numbers. 0 to any number.
जिस element की value ज्यादा होगी उसकी width ज्यादा घटेगी।

4) align-self :- It is used to align the individual inner element of the container.
flex-end, flex-start, center, baseline.

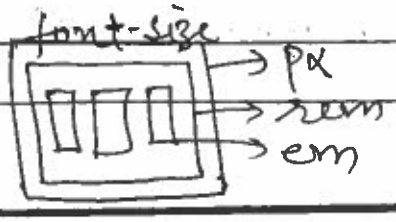
5) flex-basis :- It specifies the default length (width & height) of an individual element inside a container.
auto, number in px or % relative to other element

(22) → CSS Responsive Design :-
Responsive design means a website should adjust its layout according to the screen size of device. e.g. :- mobile, tablet, laptop / desktop.

1) Measurement units :→ We must use relative units (%, vw/vh, rem, em) to make our website responsive as fluid layout.

* Percentage and vw/vh are commonly used to set the length/dimension (height & width) of an element. But we should always use min-width, max-width, min-height and max-height

* rem and em are used for font-size of a text.



2) Viewport meta tag :→ viewport means screen of device.

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

It allows you to control the width and scaling of the viewport so that it's sized correctly on all devices.

3) Media-Queries :- It is used to make responsive website by changing the CSS style according to media type and condition.

→ Media Types :- (i) All (ii) Screen
(iii) print (iv) speech

→ conditions :- (i) max-width (ii) min-width
(iii) max-height (iv) min-height

Syntax :-

@media media type and (condition) { element
{ css; } }

for eg :-

```
@media only screen and (max-width: 480px)
{ body { width: 100%; } }
```

* The view of a website will change in mobile, tablet, desktop based on screen media type and its conditions of breakpoint. (specifies the device's width).

(23) → CSS Important Properties :-

1) user-select :- It can be used in any element. It allows/stops a user to copy the content.
values :- auto, none, text, all

2) quotes :- It is used with <q> tag only to change the sign of quotations (" ")
for eg :-
q { quotes : "$"; }

3) z-index :- It is used to display an element over another element. Values in numbers.
An element which has higher value will display over an element who has lesser value.

4) Box-shadow and text-shadow :- It is used to add shadow effect to the box of an element and the text.
values :- inset, h-shadow, v-shadow, blur, spread, color, inset → (To make box-shadow inside the box), px. ← (Thickness of box-shadow)

5) box-sizing :- It is used to include the padding and border in an element's height and width
value :- border-box

6) object-fit :- It is used to make an image responsive.
values :- contain, cover. (Similar to bg-size property)

7) object-position :- It is used to set the starting position of an image.
values :- top, bottom, left, right, center

8) Gradient :- It is used with background property to set two or more colors in background. (25)

Values :- (i) linear-gradient (color1, color2, ...);
(ii) radial-gradient (color1, color2 size of color in %px);
(iii) repeatedly-gradient (color1, color2, ---);
(iv) repeatedly-gradient (color1, color2 ---);

24) → CSS Transform 2D :-

It is used to rotate, scale, skew and translate an element.

1) translate() :- used to change the position of an element to given X-axis & Y axis.

2) rotate() :- used to rotate an element from an origin at a given degree value.

3) scale() :- used to increase/decrease the height/width of an element at given times.
for eg :- scaleX() and scaleY() also.
transform : scale(2, 0.5);

4) skew() :- used to skew (तेढा करना) an element at given degree along X axis and Y-axis. Also we have skewX() & skewY().

transform : skew(20deg, 30deg);

5) matrix() :- shorthand of all transform 2-D properties.

matrix(scaleX, skewX, skewX, scaleY, translateX, translateY)

(2.5) CSS Transform 3D :-

1) translate3d() :- used to change the position of an element at given X axis, Y axis and Z axis.
translateX(), translateY(), translateZ().

2) rotateX() :- used to rotate an element along X-axis. ↕
rotateY() :- used to rotate an element along Y-axis. ↔
rotateZ() :- used to rotate an element along Z-axis. ↙
similar to rotate() in transform 2D.

3) transform-origin :- used to change the origin at given position.
values :- left top, left bottom, right top, right bottom.

4) perspective() :- used for 3D view of element at given distance in px.

5) scale3d() :- used to increase/decrease the size of an element along X axis, Y axis and Z axis.
scaleX(), scaleY(), scaleZ()

6) perspective-origin :- used to change the origin of view at given position. for eg:-
left center, right top, right center etc.

7) transform-style :- used in parent element to give the 3D effect.
value :- preserve-3d.

8) Backface-visibility :- used to show/hide the backface of an element.
values :- hidden, visible (default).

(26) CSS Animation :- It allows an element to change the style from one to another at multiple times.

* ~~Animation - Name~~ To use an animation we must create an animation and give them a name.

Process to create an Animation :-

@ Keyframes :- It is used to create an animation and give them a name.

for eg :-

```
@keyframes increase {        → Animation name
    from {css;}
    to {css;}
}
```

* From and To are two breakpoints to change the CSS style from 0% to 100%.

* We can give more breakpoints to change the css style according to breakpoints.

for eg :-

```
@keyframes increase {
    0% {css;}
    25% {css;}
    50% {css;}
    100% {css;}
}
```

1) animation-name :- used to give the animation-name. We can also give more than one animation name.

2) animation-duration :- time to complete the animation.

3) animation-timing-function :- speed to complete an animation
values :- linear, ease, ease-in, ease-out, ease-in-out, steps(4, end).

4) animation-delay :- animation will start after this time.

5) animation-iteration-count :- number of times to play an animation.
values :- 1, 2, 3 --- infinite.

6) animation-direction :- direction of an animation.
values :- normal, reverse, alternate.

7) animation-play-state :- used to pause an animation at any situation. (for eg :- hover).

8) animation-fill-mode :- not so useful. see YahooBaba youtube video.

⁂ Animate.css :- This website is useful to use pre-built animations in your CSS file.

JavaScript

→ JavaScript is a web based scripting language for client and server side.

→ JavaScript used in "event" based dynamic website.

→ It is a case-sensitive and object oriented language.

→ JavaScript is developed by Brandon Eich in 1995 and published in 1997.

→ Initial name of JavaScript was Mocha, then LiveScript and finally JavaScript.

JavaScript is also known as ECMAScript.

JavaScript mainly used for:- (1)

1) Client-Side Validation.
2) Dynamic drop down menu.
3) Displaying date, time and clocks.
4) Pop-up windows and dialog boxes.
5) Audio/Vedio player. (2)
6) Animated sliders.
7) Zoom effect.
8) Charts and Graphs.
9) Maps.

→ JavaScript can be placed in both <head> and <body> using <script> tag.

→ A computer program is a list of statements / instructions that has to be one by one executed by a computer.

JavaScript statements are made of :-

(i) Variables and its values.
(ii) operators and condition.
(iii) expressions and keywords.
(iv) Comments.

→ Always use semicolons(;) after every statement.

for eg :- `var firstName = "Dinesh" ;`

- `var` → Keyword
- `firstName` → Variable Name
- `=` → operator
- `"Dinesh"` → value
- `;` → Semi colons

(1) → Keywords :- They are pre-defined words in JS and cannot use in declaring any name.

for eg :- var, let, const, If, If-else, switch, return, for ...

(2) → Variables :- They are used to store data / values. Var, let and const are three keywords use to declare an variable.

for eg :

```
var rollNo = 16 ;
```

Declared with Var Keyword

→ Here rollNo is a variable name, declared with var keyword.

→ An equal sign(=) is used assign values to variables.

1) Var :-

→ A variable declared with Var keyword can again declare with var keyword and assign new value. And new value will take place of old value.

for eg:- Var a = 16; variable a is declared with Var keyword & value 16 is assign to it.

Var a = "Dinesh"; variable a is again declared with Var keyword & value 10 is assign to it. (2)

a = true; variable a is again defined and value is boolean (true) assign to it.

2) let :- A variable declared with let keyword cannot be declared again but you can define it again.

for eg:- let a = 16; variable a is declared with let keyword & value 16 is assign to it.

let a = "Dinesh"; Error / you can't do this (3)

a = "Dinesh"; You can do this. Here variable a is not declare again. Only value Dinesh is assign to it.

3) Const :- A variable declared with const keyword can not be declare and define again.

for eg:- const a = "Dinesh"; variable a is declared with const keyword & value Dinesh is assign to it.

const a = "Mukesh"; // Error // You can't do this.
a = 10;

(2) → Block-Scope :-

→ Variable declared with var keyword inside a block { } can be accessed from outside a blocks.

for eg:-

```
var x = 5;
{ var x = 6; }
```

o/p : x = 6
Var is accessible in a blocks { }

→ Variable declared with let & const keywords inside a block { } can not be accessed from outside a blocks { } and it is only accessible inside a block { }

(3) → Data Types :- Data type is the type of value assigned to a variable.

There are 6 primitive and 1 non-primitive datatypes.

1) Numbers :- Any integer and decimal numbers. Also Infinity, -infinity and NaN.

NaN :- Not a number

2) String :- It means text characters. And anything declare with in quotes(" ") will be treated as a string.
foreg:- "Dinesh" "10" "Dinesh10"

3) Boolean :- true and false are bollean values but without quotes(" ") otherwise they will be treated as a string.

4) Null :- null value. This means that there is an absence of a value.
Datatype is an Object.

5) Undefined :- A variable declared without a data.
foreg: var a;

6) Empty :- Only quotes(" ") are declared without any data. It will be considered as a string.

7) Objects :- We will (study) it after some chapters. It has full chapters. Non-primitive. Page-92

(4) → typeof :- It is used find the type of data/value.
foreg: type of(" ") ⟶ string
type of(NaN) ⟶ Number
type of(12-"abc") ⟶ Number
typeof(true) ⟶ Boolean

(5) Operators :- They are used to perform operations on numbers.

1) Arithmetic Operators :- normal mathematical operators

1) '+' addition → used to add to values / variables.
2) '-' Subtraction → used to subtract
3) '*' multiplication → used to multiply
4) '/' divison → used to divide
5) '%' modulus → used to get the remainder.
6) '**' exponential → to raise the power. user. x=5 ** 3 | 5^3

2) Assignment Operators :- used to assign value to variables

1) '=' Assign → used to assign a value to variable
2) '+=' → used to first add & then assign
   var x = 10;
   x += 5; first 5 is added to x and
   o/p : x = 15, then 15 is assign to x
3) '-=' → used to first subtract & then assign
4) '/=' → first divide & then assign
5) '%=' → first find the modulus and then assign that value to the variable.
6) '**=' → first raise the power of second value to first value and assign that value.

3) Logical Operators :- It is used to compare the two conditions.

1) "&&" Logical and → If both conditions are true then True
2) "||" Logical or → If any one condition is true then True
3) "!" Logical not → Used with only one condition. It will show the opposite result of that one condition

4) Comparison Operators :- Used to compare two variables or their values.

1) "==" → used to check value of two variables.
2) === → used to checks value and datatypes of two variables

eg:- var x = 5;
var y = "5";
(x == y) true
(x === y) false; 5 is number & "5" is String

3) != not equal to
4) < less than
5) > greater than
6) >= greater than and equal to
7) <= less than and equal to=

5) Bitwise Operators :- values based on binary Numbers

1) "&" AND
2) "|" OR
3) ~ NOT
4) "^" XOR
5) "<<" leftShift
6) ">>" right Shift
7) ">>>" unsigned right shift

Watch Youtube YahooBaba.

6) Ternary Operator :- we will study after few chapters in details Page No 77

(condition) ? true statement : false statement;

7) Increment / Decrement Operators :-

1) Pre-increment (++) :- var x = 5 ;
++x ; x=6, added before the o/p
o/p : x = 6

2) Post-increment (++) :- var x=5;
x++; x=5, not added.
o/p : x = 5 Here value is added.

3) Pre-decrement (--) :- var x=5;
--x; x=4, subtracted before the o/p
o/p : x=4;

4) Post-decrement (--) :- var x=5;
x--; x=5 not subtracted
o/p : x = 5; Here value is subtracted

8) String operators :-

Concatenate "+" :- It is used to add / concatenate two strings.

* If you put a number in quotes (" "), then the number and rest of numbers will be treated as a string.

for eg : var x = "10"; y = "5"; x+y+x; o/p = 105.

var x = "10" + 5 ; o/p = 105
var x = "Dinesh"; x+ = "Rao" ; o/p = Dinesh Rao
var x = "Dinesh" + 16 ; o/p = Dinesh16
var x = 10 + 5 + "2" + 3 ; o/p = 1523

6) → Conditions :- A condition is used to run different sets of statements based on the situation.

1) If :- A set of statements will execute/run only if the condition is true.

Syntax :-

```
if (condition)
        { statement; }
```

for eg :-

```
var x = 10;
```

(i) `if (x > 0)`
`{ console.log("x is positive"); }`

(ii) `if (x >= 10 && x <= 15)`
`{ console.log("x is eligible"); }`

2) If-else :- If a condition is true, execute/run a first blocks of statements.
→ If a condition is false, execute/run a second blocks of statements.

Syntax :-

```
If (condition)
        { statements; }
else { statements; }
```

for eg :-

```
var x = 18;
if (x >= 18) { console.log("eligible"); }
else { console.log("not eligible"); }
```

3) If - else if :- If a condition is true, execute/run a first block of statements.
→ If a condition is false, check for second condition. Now if second condition is true, execute/run a second block of statements and if second condition is false, check for third condition and do same process till the true condition.
→ If all conditions are false, execute/run else part.

Syntax :-

```
if (condition) { statements ; }
else if (condition) { statements ; }
else if (condition) { statements ; }
else { statements ; }
```

4) Nested - If :- It means this else part is also contains if - else condition.

Syntax :-

```
if (condition)
        { statements ; }
else { if (condition) { statements ; }
        else { statements ; }
}
```

for eg :- If was x = 10;

```
If (x >= 10) { false ; }
else { If (x < 10) { "false" ; }
        else { " true" ; }
}
```

5) Switch-Case :- It is similar to if-else if. (7)

Syntax :- switch (expression/variable/true) {
case condition : statements ;
break;
case condition : statements ;
break;
default :
statements ; }

(8)

→ Expression means any value
→ Variable means name of variable
→ True means run the switch-case in any condition
→ Break is used to come out of the switch-case when a condition is true.
→ default is similar of else. If any of cases are not true execute the default statement.

for eg :- var age = 18 ;
switch (age/true) {
case (age >= 15 && age <= 20) : "eligible" ;
break;
case (age <= 15 && age > 20) : "not eligible" ;
break;
default : "enter the valid age" ;
}

(7) → Ternary - operator :- It is similar like If-else condition. (? : ;)

(condition) ? "true statement" : "false statement";

→ If a condition is true then true statement will execute.
→ If a condition is false then false statement will execute.

(8) → Loops :- It is used execute/run a statement many times according to condition.

1) While-loop :- var i = 0; → Variable declaration
while (i < 10) → condition
{ console.log(i); → statement
i = i++; } ← increment to go out of loop.

2) do-while loop :-

var i = 0; → Variable Declaration
do { console.log(i); → Statement
i = i++; } → Increment to go out of loop.
while (i < 10); → condition.

3) for-loop :-

for (var i = 0, i < 10, i++) { statement } ;

Variable declaration — condition — Increment to go out of loop.

1) → **Functions :-** A function is a set of statements made to performed a specific task.

→ A function is created so that one can use it many times as needed.

→ You don't have to write that same set of statements again and again. You can just use the name of function.

Syntax :-

```
function functionName (VariableName / Arguments)
{ statements;
  return; }
```

→ Keyword to declare a function

→ functionName is the name of function you gave.

→ Arguments/VariableName means name of variables you will use inside the block { }.

→ return Keyword is used to store the result of statements in functionName.

Ex:- A Program to find the factorial of a number

```
function factorial (number){
    var fact = 1;
    for(var i=1; i<=number; i++){
        fact = fact*i; }
    return fact; }
```

→ Function-call :- It means how to use a function in other place of code.

→ In previous example we have factorial(number) function, if you want to print the result of this function, then

factorial(5);
o/p = 120

→ If you want to store the result of this function in any variable, then

var result = factorial(6);
document.write(result); o/p = 720.

⚠ You can call/use this function anywhere in a code by using its name.

→ Variable length Arguments :- If a function is used/called by less/more variable than required, than.

foreg:-
```
function sum(a, b){
    console.log(a, b);
    return a+b; }
```
→ var sumNum = sum(4);
o/p :- NaN

→ var sumNum = sum(4, 5, 6);
o/p :- 9

b is undefined so 4 + undefined = NaN

6 will be ignored.

→ Function-Hoisting & Variable Hoisting :-

i) Variable Hoisting :-

```
console.log(x);
var x = 10;
```

o/p = Undefined.

because the above code will be treated as :-

```
var x;
console.log(x);
x = 10;
```

(ii) Function Hoisting :- Function declarations are Hoisted.

```
demo();
function demo(){
    console.log("Hoisted");
}
```

o/p :- Hoisted.

Funtion call/use before its decaration is possible.

(iii) Variable - Hoisting in function :-

```
function demo(){
    console.log(x);
    var x = 10; }
demo();
```

o/p :- Undefined.

⇒

```
function demo(){
    var x;
    console.log(x);
    x = 10; }
demo();
```

→ Function - Scope :→ Scope is the area in code where a variable is accessible.

1) Global - scope :→ A variable is declared outside the blocks { } is accessible any where in a code.

for eg :-

```
var x = 'name';
console.log(x);
o/p: name
```

variable declared outside of { } blocks.

Var Keyword has a global scope and can be accessed by any anywhere in a code. It means variable declared with var keyword can be accessible inside as well as outside of the blocks { }.

2) local-scope :- A variable declared inside the block { } is accessible only inside the blocks { } area.

for eg :-

(i)
```
let x = "Global";
function demo(){
    let x = "local";
    console.log(x); }
demo();
console.log(x);
```

o/p :- local
Global

let & const Keyword has a local scope & can be accessed only inside the block area { }.

(ii)
```
function demo(){
    let x = "local";
    console.log(x); }
demo();
console.log(x);
```

o/p ⇒ local
error.

3) function-scope :- A variable declared with any keyword (var, let, const) is accessible only inside the block { } of function.

for eg:-

```
function demo() {
    var x = "scope";
    console.log(x); }
demo();
console.log(x);
```

o/p => scope error

x is inside the function block { }.

→ Function in function :-

A scope of a function which declared inside a function is local.

for eg:

```
function outer() {
    var x = 10;
    function inner() {
        console.log(x); }
    inner();
} outer();
```

o/p = 10.

* You cannot call/use a inner function outside of the outer function.

Examples of function & variable scopes :-

i)

```
var name = "global";
function outer(){
    var name = "outer";
    function inner(){
        var name = "inner";
        console.log(name);
    }
    inner();
    console.log(name);
} outer();
console.log(name);
```

o/p = inner
o/p = outer
o/p = global

ii)

```
var name = "global";
function outer(){
    var name = "outer";
    function inner(){
        console.log(name);
    }
    inner();
    console.log(name);
}
outer();
console.log(name);
```

o/p = outer
o/p = outer
o/p = global

iii)

```
var name = "global";
function outer(){
    function inner(){
        console.log(name);}
    inner;
    console.log(name);}
outer; console.log(name);
```

o/p = global
o/p = global
o/p = global.

→ Function-Expression :- It is used to assign the function to a variable

(i) for eg:- Named function :-

```
var factorial = function fact(n){
    var ans = 1;
    for(var i=0; i<=n; i++){
        ans = ans * i; }
    return ans; };
console.log(factorial(5));
```

o/p : 120.

Now you can't call/use this fact() function without its variable (factorial). It will show an error.

(ii) for eg:- Anonymous function :- → Name of function is not available

```
var factorial = function (n){
    var ans = 1;
    for(var i=0; i<=n; i++){
        ans = ans * i; }
    return ans; }
console.log(factorial(4));
```

o/p = 24.

☆ You can use/call a function inside a function in case of recursion.

Callback Function /

→ Passing Function as Arguement :- We can pass a function as function name to the arguement of another function.

for eg:-

```
function fact(n){
var ans = 1;
for(var i=0; i<=n; i++){
    ans = ans * i; }
return ans; }

function ncr(n, r, fact){
    return fact(n)/(fact(n)*(fact(n-r)));
    }

ncr(5, 2, fact);
```

o/p => 10

→ Arrow Function :- Shorter syntax of function.

Before:
```
var print = function(){
    return "Hello"; }
console.log(print());   o/p: Hello
```

After:
```
var print = () => Hello;
console.log(print());   o/p: Hello.
```

If a function has only one statement, you can remove return and curly brackets.

→
```
var sum = (a, b) => a+b;
console.log(sum(2,3));   o/p: 5.
```

☆ → Arrow function is used inside another function to create Closure. (Closure in Page - 114)

(10) → Arrays:- It is list/collection of data/value used to store multiple values in one variable.

→ Each value has index number attached to it, used to access the values. Index starts from 0.

→ Syntax:-

Keyword ArrayName = [value1, value2, ...];

or

Keyword ArrayName = new Array(1,2,3,...);

for eg:-

var arr = [1,2,3,4,5];

or var arr = new Array(1,2,3,4,5);

→ You can also create an Empty array and then provide the values.

```
var arr = [ ];
arr[0] = 1;
arr[1] = 2;
arr[2] = 3;
```

→ How to get output of an Array:-

i) console.log(arr[1]); o/p = 2. — Accessing one value from Array

ii) console.log(arr); o/p = 1,2,3. — Accessing full Array

iii) arr[1] = 5;
console.log(arr[1]); o/p = 5. — Replacing the value at index in array

P.T.O.

→ Array Length :- It is used to find the total number of items / values in an Array.

for eg:
```
var cars = ["BMW", "HONDA", "SWIFT"];
console.log(cars.length);
o/p = 3
```

arrayName.length

iv) `console.log(arr.length-1);`
o/p = 2.

Total length = 3
Index starts from 0. So the 0, 1, 2 has values 1, 2, 3.

→ If you access the array outside it's range ie. invalid index, then the o/p is undefined.

for eg: `console.log(arr[5]);`
o/p : undefined.

→ How to recognise an array :-

typeof is used to find the data type of a variable, but for an array, it will return a object.

i) ArrayName.isArray() :- returns true or false.

for eg: `console.log(arr.isArray());` → true

ii) instanceof :- returns true or false

for eg: `console.log(arr instanceof Array)` → true

→ Array Methods/functions :-

(1) pop() :- It removes the last item from an array and returns that element.

for eg:-

```
var arr = [1, 2, 3, 4, 5];
arr.pop();
```

o/p → 5 and length of array is = 4.

2) shift() :- It removes the first item from an array and shifts all items to left position.

for eg:-

```
arr.shift();
```

o/p = 1 and arr = [2, 3, 4, 5];

3) push() :- It adds the new element at end of an array and returns the last position where an item is added.

for eg:-

```
arr.push(6);
```

o/p: 5 and arr = [2, 3, 4, 5, 6];

4) unshift() :- It adds a new item at the beginning of an array and shifts all items to right position.

for eg:-

```
arr.unshift(1);
```

o/p:- 6 and arr = [1, 2, 3, 4, 5, 6];

→ It returns the length of an array after adding an item.

5) delete arrayName[index] :- It deletes an items from an array at defined index and leaves "empty/undefined" hole in an array.

for eg:-

delete arr[2];

o/p:- arr = [1, 2, empty, 4, 5, 6]; → Index
0 1 2 3 4 5
1 2 3 4 5 6 → Position

6) concat( ) :- It is used to add/concate 2 or more arrays in one array.

for eg:-

var arr1 = [0, 1, 2, 3];
var arr2 = [4, 5, 6];

arr1.concat(arr2);

o/p: arr1 = [0, 1, 2, 3, 4, 5, 6];

7) splice (Index, how many delete, items/values) :- It adds new item to an array at defined index and deletes the defined number of items from an array.

for eg:- 0 1 2 3 4 → Index
var arr = [1, 2, 3, 4, 5];
arr.splice(1, 0, 6, 7);
o/p: arr = [1, 6, 7, 2, 3, 4, 5];
0 1 2 3 4 5 6 → Index

☆ → You can use splice() to delete items without leaving empty/undefined holes.

for eg: arr.splice(1, 2); → Don't give the item parameter.

o/p: arr = [1, 2, 3, 4, 5];

8) reverse() :- It is used to reverse the items of an array.

eg:-
```
var arr = [1, 2, 3, 4, 5];
arr.reverse();
```
o/p: arr = [5, 4, 3, 2, 1]

9) toString() :- It is used to return the array in the form of a string. (comma seprated).

eg:-
```
arr.toString();
```
o/p: "5, 4, 3, 2, 1"

10) join('seprater') :- Similar to (tostring) but here you can define the seprater

eg:-
```
arr.join('*');
```
o/p: "5 * 4 * 3 * 2 * 1"

11) indexOf(element, fromIndex) :- It is used to find the first index of an item.

eg:-
```
var arr = [0, 1, 2, 8, 4, 6, 9, 3];
arr.indexOf(8, 0);
```
o/p: 3

12) sort() :- It is used to sort an array of string items

eg:-
```
var arr = ["Banana", "Mango", "Apple"];
arr.sort();
```
o/p => arr = ["Apple", "Banana", "Mango"].

13) sort(function(a,b){return a-b}) :- It is used to sort an array of numbers items. (Ascending Order)

for eg:- var arr = [10, 5, 9, 8, 1, 3];
arr.sort(function(a,b){return a-b});
o/p: arr = [1, 3, 5, 8, 9, 10].

14) sort(function(a,b){return b-a}) :- used to sort an array in descending order.

for eg:- arr.sort(function(a,b){return b-a});
o/p: arr = [10, 9, 8, 5, 3, 1]

→ Array Iteration :- It is used to work on ~~each~~ every array item

1) For loop :- used to iterate over all items of an array.

```
var arr = [0, 1, 2, 3];
for (var i=0; i<arr.length; i++){
    console.log(arr[i] * 2); }
```

o/p = 0, 2, 4, 6

2) forEach :- used to print all items and their index of array

Same name for all

```
arr.forEach(function print(items, index){
    console.log(index + " " + items); })
```

o/p

```
0 : 0
1 : 1
2 : 2
3 : 3
```

```
for (i of arr){ console.log(i); }
```

o/p = 0, 1, 2, 3.

3) for of :- similar to for loop.

(11) → Objects :- Is objects are a collection of properties in a key-value pair.

→ Each property is in key: value pair seprated by colon (:).

→ The value can contain any type of data :- number, boolean, string, null, array, function and other object also.

for eg :-

```
                                    Key : value
var obj = { fname : "Dinesh",
string —   lname : "Kumar",
number —   age : 50,
array —    classes : [10, 12, "Btech"],
objects —  marks : { english : 70,
                     hindi : 50, },
function — fullname : function () {
                      return this.fname
                      + this.lname }
          };
```

or

```
var obj = new Object({ ' ', ' ' });
```

→ Creating and Accessing Object Properties :-

1) using dot(.) :- If key is a string use dot(.) operator

for eg :-

Accessing :- obj.age ; o/p = 50.

Creating new Property :- obj.semester = 2;

2) using square bracket [ ] :- If key is not a string use square brackets

foreg:- var persons = { 1 : "Dinesh",
2 : "Mukesh",
3 : "Mahinish", };

Accessing :- persons["2"]; o/p = Mukesh.

Creating new property :- persons["4"] = "Himanya"; o/p = Himanya.

→ Deleting Property :- To delete any property from object.

~~delete["persons.2"]~~ delete persons["2"];

or

~~delete.lname;~~ delete obj.lname;

→ Changing old property :- To existing property change

persons["2"] = "Jyoti";

or

obj.lname = "Rao";

→ Objects inside an Object :- When a value of object property is another object.

obj.marks.english; o/p = 70
obj.marks.english = 80; o/p = 80
delete obj.marks.english;

or obj["marks"]["english"];

or obj.marks = {Math : 90}; To add new property

→ function inside Object :- When a function is the value of an object.

```
obj.function();
```
or
```
obj["function()"];
```

(12)

→ Array inside Object :- When an array is the value of an object.

```
obj.classes[0];    o/p = 10
```

[0] is index of an array used to access the values of object key's array.

→ Iterating Objects :- It is used traverse all the keys of an object.

1) for-in loop :-
```
var obj = { name : "Dinesh",
            age : 50,
            class : 12 };
for (var key in obj) {
    console.log(key, obj[key]);
}
```

o/p

name : Dinesh, age : 50, class : 12,	var key is denoting key of object properties.

2) Object.keys(objectName);
3) Object.getOwnPropertyNames(objectName);

Above both function are pre-defined and used to get the key name of object properties.

(12) → Timing-Events :- It allows the execution of the specific code at specific time interval. They are pre-defined (Global) methods/event in the DOM window object.

1) setTimeout() :- used to execute a piece of code after a certain amount of time.
- The function is passed as the parameter.

for eg :-
```
function sayHello(){
    console.log("Hello");}
```
sayHello - function

```
var hello = setTimeout(sayHello, 1000);
```
var: variable; setTimeout: Time Event function; sayHello: function; 1000: Time (After how many seconds this function execute)

- The above sayHello function will execute after 1000ms / 1sec of page load.

2) setInterval() :- used to execute a piece of code repeatedly with fixed time interval.

for eg :-
```
var hello = setInterval(sayHello, 1000);
```

- The above sayHello function will execute repeatedly after 1000ms / 1sec of time interval.

3) clearTimeout() :- used to cancel the setTimeout() event/method.

- This event is used in the function, which is used as the parameter of setTimeOut().

for eg:-

```
function sayHello(){
    console.log("Hello");
    clearTimeout(hello);
}
var hello = setTimeout(sayHello, 1000);
```

this variable is used in clearTimeout()

4) clearInterval() :- used to cancell the repeated time action established by setInterval().

for eg:-

```
var count = 0;
function timer(){
    if(count == 10){
        console.log("Time up");
        clearInterval(id);
        return;}
    console.log(count);
    count++;}
var id = setInterval(timer, 1000);
```

same variable

- The above timer() function will execute for 10sec at 1sec time interval and after 10sec it will automatically cancel.

(13) → DOM :- Document Object Model

- It is an API that represents and interacts with HTML document.
- API is an easy way by which you can use code written by somebody else.
- DOM is used to accessing / editing elements in an HTML page as adding new elements to the page will become quite easy and you don't have to write everything from scratch.

- DOM represents a object of HTML document with logical Tree.
- In DOM, <HTML> tag is root node, then <head> and <body> tags are it's children. Like this we have tags inside another tags as their children. Each branch of tree ends in a node and each node is a object. DOM is used to access these nodes (objects) in HTML document.

→ Methods to find HTML elements :-

1) document.getElementById("id") :-
used to find an HTML element of a specific ID

2) document.getElementByClassName("class") :-
used to find all HTML elements of the same class name. It will return the array of all elements of same class. By using array Index, we can trigger any element of array.

3) document.getElementByTagName("Tag") :-

Used to find all HTML elements of same Tag name.
It will return the array of elements of same Tag name and you can access the individual element using Index of array.

4) document.querySelectorAll("#id/.class") :-

Used to find all HTML elements whose class/id name match.
It will also return the array of elements.

for eg :-
document.querySelectorAll("#abc")[2];

5). Predefined Objects :-

(i) document.body;
(ii) document.head;
(iii) document.title;
(iv) document.anchors;
(v) document.images;
(vi) document.URL;
(vii) document.domain;
(viii) document.forms;
(ix) document.write();

→ Methods to Get & Set the content of HTML element :-

- The content of an element can be Text, text with HTML tags and its attributes name and attributes value. (style = "color : cyan").
  name = value

1) element.innerHTML :- used to get and set the full HTML content (Text with HTML Tags).

for eg :-

→ var element = document.getElementByTagName("H1");

It will give the ~~HTML content~~ all 'H1' element in array.

→ element[0].innerHTML ;

It will give the full Text with HTML Tag of 1st H1 tag.

→ element[0].innerHTML = "<h1> Hello </h1>".
o/p = Hello.

It will change the ~~value~~ Text of H1 tag.

2) element.innerText :- Used to get and set the content of HTML element. (without Tags).

for eg :-

→ var element = document.querySelectorAll(".abc")
[0].innerText ;

It will give the Text of 1st element whose class is abc.

→ element = "<H1> Hello </H1>" ;
o/p = <H1> Hello </H1>

Always use innerText.

3) element.attributes :- Used to get all the attributes name of specified element.

It will return the array of all attributes.

for eg :-

→ var element = document.getElementByTag name("H1").attributes;

→ console.log(element);

It will returns the list of all attributes of H1 element in form of array.

4) element.getAttribute("AttributeName") :-

Used to get the value of attribute. ('id="abc") of specified Tag.

Attribute name Attribute Value.

for eg :-

→ var element = document.element.getAttributes("id");

→ console.log('element');

o/p = 'abc'

5) element.getAttribute("Attribute Name", "New Value") :-

Used to change the value of attribute of specified Tag.

for eg :- → var a = document.element.getAttribute ("id", "xyz");

→ console.log(a);

o/p = 'xyz'

6) element.removeAttribute("Attribute name") :-

Used to remove / delete the specified attribute of specified element.

eg:- → var a = document.element.removeAttribute("id");
→ console.log(a)
o/p: null.

→ DOM Create Methods:-

1) To create a new HTML element through Js DOM:-

var a = document.createElement("p");

2) To create a Text node in HTML through Js DOM :-

var b = document.createTextNode("Any Text");

3) To create a comment in HTML through Js DOM :-

var d = document.createComment("This is comment");

DOM Append Methods:-

Using create methods we have created element, textnode and comment. Now we have to put these into HTML page.

1) appendChild():- using eg 1 & 2; a.appendChild(b);

Now the textnode is appended to new element(p).

document.element.appendChild(a);

Now the new element is appended as the last child element of specified element.

2) insertBefore(new element, triggered element) :-

If you dont want append a new element as the as last child element of a specified element, use this.

for eg:-

```
<div id="div1">
  <p id="p1"> This is first para <P>
  <p id="p2"> This is second para <P>
</div>
<script>
var element = document.createElement("p");
var node = document.createTextNode("Hello");
  element.appendChild(node);
var old = document.getElementById("div1");
var child = document.getElementById("p2");
  old.insertBefore(element, child);
</script>
```

O/P

This is first para.
Hello
This is second Para.

selectedElement.insertBefore(newelement, Triggered element);

selected element के अन्दर किस element से पहले नया element डालना है उस element का नाम।

नया element

किसी element में आपको नया element डालना (append करना) है।

→ Create and Append Together Methods :-

1) insertAdjacentHTML( ) :- used to create both element and text together and then append in selected element. We have 4 positions where to append new element.

i) BeforeBegin :- Selected element ke pehle new element add hoga
ii) Before End :-
iii) After Begin :-
iv) After End :-

for eg :- var element = document.getElementById("div1");
var new = <h1> This is new element with Text </h1>

element.insertAdjacentHTML(AfterEnd, new);

↓ Selected element — Position — new element with text.

2) insertAdjacentText( ) :- used to create and append new text only.

for eg :-
var element = document.getElementById("div1");
var new = "Hello I am a Text";
element.insertAdjacentText(BeforeEnd, new);

3) insertAdjacentElement( ) :- used to create and append new element only.

for eg :-
var element = document.getElementById("div1");
var new = document.createElement("p");
element.insertAdjacentElement(BeforeBegin, new);

→ DOM Remove Methods :-

Used to remove or replace the old element.

1) `document.getElementById("d1").remove();`

2) removeChild() :- used to remove the child element of selected element.

foreg :-

```
var element= document.getElementById("div1");
var new = document.getElementById("p1");
element.removeChild(new);
```

3) replaceChild() :- used to replace the old child element with new child element of selected element.

foreg :-

~~var element = getElementById(~~

New Element) var element = document.createElement("p");
New Text) var text = document.createTextNode("Hello");
New Text in New element) element.appendChild(text);
Selected Element) var main = document.getElementById("div1");
old child element) var old = document.getElementById("p1");

main.replaceChild(element, old);

Selected Element — new element — old element

(14) → Events :- An HTML event is something that the browser does or something a user does.

An event can be triggered by the user action. for eg:- clicking a mouse button or tapping keyboard.

Event is an attribute permofed according to their HTML element.

It mostly comes with Js code (function)

1) addEventListener() :- It has 2 parameters.

(i) eventName :- Name of the event (action).

(ii) function() :- this function will be called when the above event triggers.

for eg:-

```
<button addEventListener('click', function() {
        alert("Button is clicked");});>
    Click me </button>
```

When a user clicks on the above button, alert ("function will call"); .....

2) Mouse Events :- An event/action happened with mouse.

i) onclick :- when mouse click on element.

for eg:-

```
<input type="text" value="firstName"
    onclick="function()">
```

ii) onmouseover :- When the cursor of the mouse comes over the element.

iii) onmouseout :- When the cursor of the mouse leaves an element.

iv) onmousedown :- When the mouse button is pressed over the element.

v) onmouseup :- When the mouse button is released over the element.

vi) onmousemove :- When the mouse movement takes place.

3) KeyBoard Events :-

i) onkeydown :- When the user press Keyboard key down

ii) onkeyup :- When the user release keyboard key up

for eg:- <input type = "text" onkeyup = "function() {document.write("Hello User");}">

Press a key and release inside the input box, you will get an o/p Hello User.

iii) onkeypress :- When the user press any key over the screen.

for eg:-

<input type = "text" onkeypress = "function(event) {console.log("KeyPressed", event.KeyCode)}">

★ KeyCode :- is used to find out which key is pressed.

4) Form Events:-

i) onfocus :- when the user focuses on at element.

ii) onsubmit:- when the user submits the form

iii) onblur :- when the focus is away from an element.

iv) onchange :- when the user changes the value of a form element.

5) Window | Document Events :-

i) onload :- when the browser finishes loading of the page.

ii) onunload :- when the user leaves the current webpage.

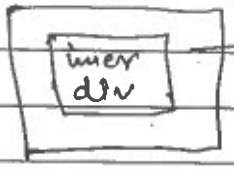
iii) onresize :- when the user resizes the window of the browser.

iv) onscroll :- when a user scrolls a page in browser or scrolls an element's content.

for eg:- <body onscroll = "window.alert('page scrolled')">

→ Propagation of Events :-

- Suppose we have one div and one more div inside the main div.



- Now if we have an event on outer div and another event on inner div, then clicking on inner div both inner event and outer event will triggered.
- So if you to stop to outer event inside the inner event you should use stopPropagation method

StopPropagation() :- It will stop outer events only inside the inner events.

for eg :- <button onClick = "function(event) { window.alert("Button is clicked"); event.stopPropagation; }></button>

(15) → <Script> Tag :- It is used to placed the JavaScript file inside the HTML code.

- <Script> tag can be placed in <head> tag as well as <body> tag.
- Most of the times, <script> tag is placed in the last of HTML document of <body> tag.

(16) → **Strict Mode** :→ Javascript internally eliminates/neglects some silent errors.

- Strict Mode is used to not neglecting the errors internally by javascript.

⇒ You can apply strictmode to the entire script by writing "use strict" at the top whole script.
⇒ Or write it inside the functions to apply it to a particular function only.

(17) → Difference between let / var / const :-

Declaration / Initialisation :-

(i)

var a;	const a;	const a = 10;
console.log(a);	console.log(a);	console.log(a);
a = 10;	o/p :- Error	o/p :- 10
o/p : undifined	a = 10;	
console.log(a);	console.log(a)	
o/p : 10	o/p :- Error	

Hoisting :-

(ii)

console.log(a);	console.log(a)	Similar in let
var a = 10;	const a = 10;	
o/p :- undifined	o/p :- Error	
⇓	It will not treat above code like var	
var a; console.log(a); o/p :- undifined)	It will treat above code like this	

(iii) Scope :-

```
var a = 10;
if (a<=10) {
    var b = 40;}
else { var c = 60;}   // this will not work
console.log(a,b,c);
```

o/p :- 10 40 Undifined
Global Scope of var

```
var a = 10;
if (a<=10) {
    let b = 40;}
else { let c = 60;}
console.log(a,b,c);
```

o/p :- 10 Error
Block Scope of let

→ Always use let keyword in blocks statements and if you don't want to use a let variable outside the blocks.

for eg:-

```
for(var i=0; i<=5; i++){setTimeOut(function(){
    console.log(i);}, 1000);}
```

O/P:- 6 6 6 6 6 6

```
for(let i=0; i<=5; i++){
    setTimeOut(function(i){
    console.log(i);}, 1000);}
```

O/P: 0 1 2 3 4 5

(17) → Execution Context and Lexical Environment :-

(1) Execution Context :— i) Global
ii) Function

i) Global :- The global execution context is created when a javascript <script> first starts to run, and it represents the global scope (window scope) in Javascript.

All variables and functions declared independently have global (window) scope execution context.

ii) Function :- A function (local) execution context is created whenever a function is called (invoked), representing the function's local scope.

All variables and functions declared inside any function have local scope execution.

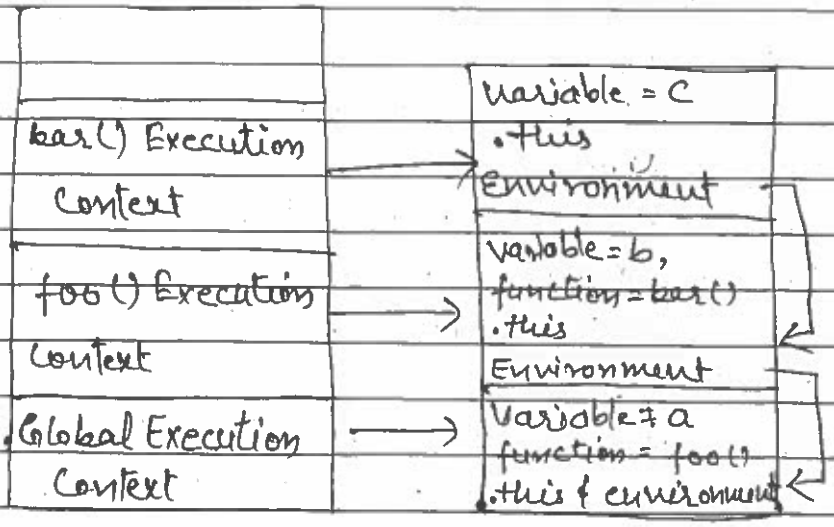
for eg :-
```
var message = "Hello there";
function foo(message) {
    bar(message); }
function bar(message) {
    console.log(message); }
foo(message);
```

→ Initially Execution Context Stack is empty.
→ When this code runs, javascript engine create one global execution context and push it to Execution Context Stack.
→ When function foo is called, Global execution context will pause and javascript engine will create a new Function context in Execution context stack.
→ When function bar is called, function context of foo will pause and new function execution context will be created in stack.
→ After function bar was finished executing it will popped out from the Execution Context Stack and go back to function foo and resume its execution.
→ Same process will applied to the foo until we finished and go back to the Global Execution Context and resume the execution.

② Lexical Environment :- A place where the local variable and function declarations are stored and reference (access) to its outer (parent) environment.

for eg:

```
var a = 'a';
function foo(){
    var b = 'b';
function bar(){
    var c = 'c';
console.log(c);
console.log(b);
console.log(a); }
bar(); }
foo();
```



Execution Context and its Lexical environment.

Whenever a Global and function execution context is created, they create their lexical environment to store their local variables and functions and refrence to outer (parent) environment.

(18) → IIFE :-

Immediately-Invoked Function Expressions pronounced "iffy" is used to executes a function instantly after it's defined.

- To ensure variables are only accessible within the scope of the defined function.

for eg :- (function () {
var fileName = "file1";
console.log(fileName); }) ();

* Suppose you have two or more <script> files in your HTML code and all of them has same variable name then javascript engine will confuse. So to ~~avoid~~ prevent this avoid use global variables.

- Now this IIFE function will invoke (runs) as soon as it is defined.

Let Vs IIFE :- Let has also local scope. It can be useful in preventing the above problem.

for eg :- { let fileName = "file1";
console.log(fileName); }

(19) → **Closure** :- It is a feature in which an inner function has access to the outer function's variables - a scope chain

It has access to its own variables.
It has access to the outer function's variables.
It has access to the global ~~function~~ variables.

A closure is created when an inner function is made accessible from outside of the function that created it. This occurs when outer function returns an inner function. When this happens, the inner function remembers all the variables (and their values) that were in scope at that time.

for eg:-

i)
```
var i = 10;
function outer(){
    var j = 20;
    var inner = function(){
        var k = 30;
        console.log(i, j, k);
    }
    return inner;
}
var inner = outer();
inner();
```

inner function returned

and called outside of outer function

o/p → 10 20 30

inner function remembers all the variables and their values.

ii)

```
var i = 10;
funtion outer(){
  var j = 20;
var inner = function(){
  var K = 30;
  console.log(i, j, K);
  j++;
  K++;
  }
return inner; }
var value = outer();
value(); value();
```

o/p:- 10 20 30
10 21 30

→ It will increased to 21
→ It will not increase as it is called ~~is outer function~~ again.
→ called 2 times
→ 1st time o/p
→ 2nd time o/p

iii)

```
var i = 10;
function outer(){
  var j = 20;
var inner = function(){
  var K = 30;
  console.log(i, j, K);
  i++;
  j++;
  K++; }
return inner;
var value = outer();
value();
var spend = outer();
spend();
```

o/p:- 10 20 30
11 20 30

→ It will increased to 11
→ They will not increased as they were called again so the initial value of their variable will be set again

→ 1st Time o/p
→ 2nd Time o/p

iv)
```
function Adder(x) {
    return function(y) {
        return x+y; }; }

var add = Adder(5);          —(i)
add(2);                      —(ii)
var add = Adder(10);         —(iii)
add(2);                      —(iv)
```

o/p → 7
12.

- In above example, Adder(5) function is called which has variable x with value 5; and a inner function.
- In line (i), variable add store the inner function.
- In line (ii), inner function is called add(2) which has variable y with value 2 and variable x=5 of outer function. This function will return x+y=7.
- In line (iii), variable add again store the inner function with outer function's Adder(10) variable x=10.
- In line (iv), inner function is called add(2) which has variable y=2 and outer function's variable x=10, This function will return x+y=10+2=12.

(20) → "This" Keyword :-

"this" Keyword is used refer an object that is executing in the current code.

- Similarly, every function while executing has a reference to its current execution context, which can be referenced by "this."

- The value of "this" keyword depends by how a function is called.

1) In normal / sloppy Mode :-

→ In global execution context, 'this' keyword refers the global object. i.e. window object.

for eg :-
```
function abc() {
    console.log(this); window.this
} abc();
```
o/p :- window object.

→ When a function is defined as a property of an object, "this" keyword refer that object.

for eg :-
```
var obj = {
    a : 25,
    abc : function() {
        console.log(obj.this);
        console.log(this.a);
    }
};
obj.abc();
```
o/p :- object
25. (this refers the obj).

2) In strict Mode :- If a function is not a property of an object. then in global execution context, "this" will be undefined.

for eg :-
```
function abc() {
    console.log(this); }
abc(); window.abc();
```
o/p :- undefined, windowobject.

3) Call() method :- In strict mode, if a function has global execution context, then you can bound a function to any object using call() method.

for eg:-

```
"use strict";
function demo(){
    console.log(this);
}
var obj = {
    a : 15,
    abc : function(){
        console.log(this);
        console.log(this.a);
    }}
demo();
```

o/p :- undefined.

```
demo.call(obj);
```

o/p :- obj object.

demo function Bound obj object.

```
c = obj.abc();
```

o/p :- obj object
15.

```
demo.call(window);
```

o/p :- window object

demo function Bound to window object

- In case of Arguments :-

```
"use strict";
function demo(a,b){
    console.log(a,b);
    console.log(this);}
var obj = { a : 15};
demo.call(obj, 3, 4);
```

o/p :- 3 4
obj object

4) apply() method :- Similar to call() method.

In case of Arguements :- You have pass the values of arguements in array.

from previous example :-

```
demo.apply(obj, [3,4]);
```

o/p :- 3 4
obj object.

(21) → Constructors :- It is used to create multiple similar objects. It provides a blueprint / structure for objects.

- Constructor are regular functions but the first letter of the function should be capital.
- Objects can be created by calling the constructor function with the "new" Keyword.

for eg :-
```
function Student(name, rollNo, age) {
    this.name = name;
    this.rollNo = rollNo;
    this.age = age; }
```
Constructor :-

Objects :-
```
var student1 = new Student("Jyoti", 24, 20);
var student2 = new Student("Divesh", 12, 30);
console.log(student1.name);
```
o/p = Jyoti.
```
console.log(student2.age);
```
o/p = 30.

Examples to understand Constructor :-

1)
```
function User(name){
    this.name = name;
    this.isAdmin = false;
    return;}
var user = User("Jyoti");
console.log(user);
```
o/p :- undefined.

Because User is called without new. And the User function is not returning anything.

2)
```
function User(name){
    this.isAdmin = false;
}
var user = new User("Jyoti");
console.log(user);
```
o/p :- User{ isAdmin : false }

3)
```
function Vehicle(numWheels, price){
    this.numWheels = numWheels;
    this.price = price;
    this.getPrice = function(){
        return this.price;
    }
}
var vehicle = new Vehicle(2, 50000);
console.log(vehicle.getPrice());
```
o/p :- 50000;

(22) → Prototype :- When you create an object using constructor and new keyword, javascript creates two object. One is main object that you want to create. Second is prototype object of constructor.

- It is used to add functions to the constructor instead of having inside the constructor.

for eg :-

```
function Vehicle(numWheels, price) {
    this.numWheels = numWheels;
    this.price = price; };
Vehicle.prototype.getPrice = function() {
    return this.price; };

var vehicle1 = new Vehicle(2, 5000);
var vehicle2 = new Vehicle(4, 50000);
console.log(vehicle1);
```

o/p :- vehicle1 object { nW : 2, price : 5000}

```
consoleLog(vehicle1.getPrice());
```

o/p :- 5000

- This adds the functions to the prototype object of the constructor.
- It will save the memory storage as only the object which requires this function, will have this property.
- If you want access the prototype of that constructor.

```
Vehicle1.prototype;
```

- Now to go back to the constructor from prototype.

```
Vehicle.prototype.constructor;
```

→ If you want to access the prototype of an object that you created.
Object.getPrototypeOf(vehicle1);

* As all the objects that you have created share same prototype of constructor. So :-

object.getPrototypeOf(vehicle1) === Vehicle.prototype;
o/p :- true.

→ If you want to check that prototype of constructor is same prototype of an object that created, then.

Vehicle.prototype.isPrototypeOf(vehicle1); o/p :- true
Constructor Prototype — object that you created.

→ If you want to check that the specific property belongs to their object constructor or inherited (ಆಸ್ತಿ) from prototype of constructor.

vehicle1.hasOwnProperty('price');
o/p :- true.
vehicle1.hasOwnProperty('getPrice');
o/p :- false.

*** More on Objects :- When you create an object.

var obj = new Object();

JavaScript automatically creates prototype of Object & constructor of object.
All the objects in JavaScript are directly or indirectly inherited from Object.prototype.

(23) → Class :- It is similar to constructor and prototype. Only Syntax is different.

for eg :-
```
class Vehicle {
    constructor(numDoors, price) {
        this.numDoors = numDoors;
        this.price = price;
    }
    getDescription() {
        return this.price + " " + this.numDoors;
    }
}

var vehicle1 = new Vehicle(2, 5000);
console.log(vehicle1.getDescription());
```
o/p :- 5000 2.

1) Class Expression :- It is another way to define a class, which is similar to function expression.
They can be named or unnamed both.

for eg :-
```
var Vehicle = class { };
```
or
```
var Vehicle = class Vehicle { };
```

2) Hoisting :- Class declarations are not hoisted.
If you try to make an object of a class before class definition, then the output will be undifined.

for eg :-
```
var vehicle1 = new Vehicle(2, 5000);
class Vehicle {
}
```
o/p = undefined.

3) Inheritance :- It is a concept that acquires the properties from one class to other classes.

for eg :- Continue from eg of class :-

```
class Car extends Vehicle {
    constructor (numWheels) {
        super(2, 5000);
        this.numWheels = numWheels;
    }
    getDescription() {
        super.getDescription();
        console.log("car", this.numWheels);
    }
}
var c = new Car(4);
console.log(c.getDescription());
```

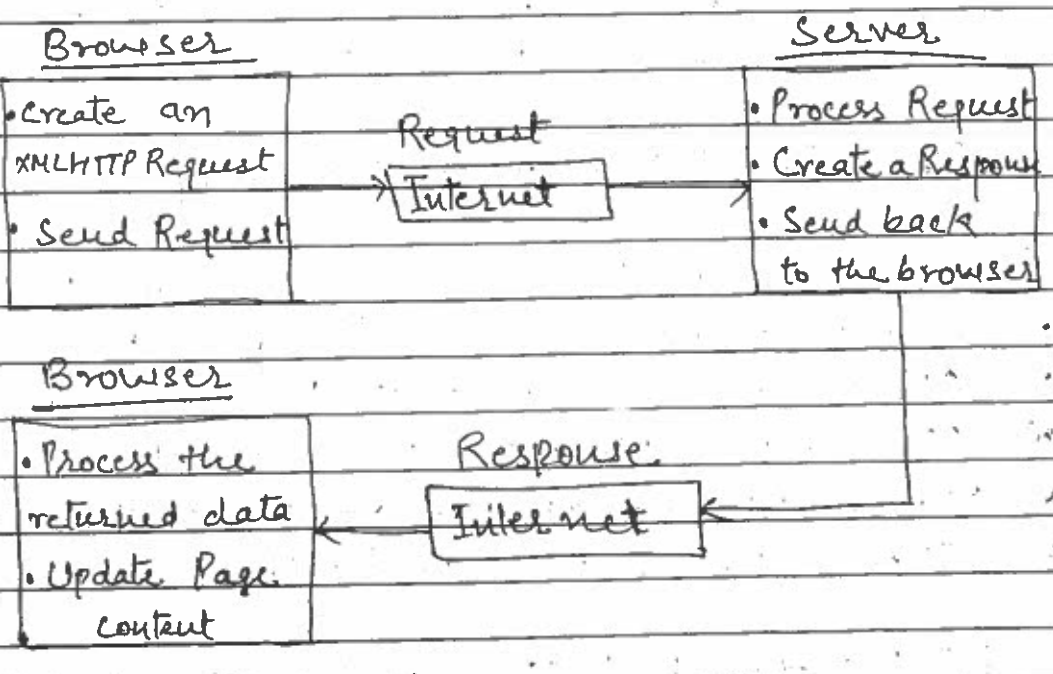
o/p :- 5000 2
car 4

* 'Super' Keyword :- It is used to call the parent class constructor in the base class.
It always comes before the "this" Keyword.

(24) → AJAX :- Asynchronous Javascript and XML.

- It is a technique for creating fast and dynamic webpages.
- It is used to update some parts of a webpage, without reloading the whole page.

1) How AJAX Works :-



Synchronous :- If a synchronous call happens then any further operation on the client gets blocked. ie.
Any further execution of operations stop executing until a response is not received.

Asynchronous :- The client can work on other things and does not need to wait for the response. for eg :- setTimeout(), setInterval().

2) HTTP Requests:- Hyper Text Transfer Protocol.

- It is a protocol used by the world wide web that defines how messages are formatted and transmitted.

- HTTP is a stateless protocol. This means that the server does not require to maintain information/status about every user for the duration of multiple requests.

Steps in HTTP requests processing:-

1) Client requests a connection with the server.
2) Server opens a connection with the client.
3) Client makes a request to the server.
4) The server processes the request.
5) The server sends a response to the client.
6) Client closes the connection.

HTTP Method:- HTTP request method is used to indicate the action to be performed on the data transmitted to the server.

1) GET :- requests for a data.
2) HEAD :- requests for data but without response body.
3) POST :- submits data causing a change in state on the server.
4) PUT :- replaces the specified data with request data.
5) DELETE :- deletes the specified data.
6) PATCH :- apply partial changes to data.
7) CONNECT :-
8) OPTIONS :-
9) Trace :-

3) HTTP Response Code :- It is a status code, that a server response to a browser's request.

1) 1xx :- Informational :- 100 :- continue
101 :- Switching Protocols
102 :- Processing

2) 2xx :- Success :- 200 :- OK
201 :- Created
202 :- Accepted
etc

3) 3xx :- Redirection :- 300 :- Multiple Choices
302 :- Found
304 :- Not Modified.
etc

4) 4xx :- Client Error :- 400 :- Bad Request
401 :- Unauthorized
403 :- Forbidden
404 :- Not Found
409 :- Conflict
etc

5) 5xx :- Server Error :- 500 :- Internal Server Error
502 :- Bad Gateway
503 :- Service Unavailable
etc

4) API :- Application Programming Interface.

- It is a medium that allows two applications to talk to each other.

The communication between the application and the server is all done via API.

The exchange of data is mostly done using JSON format.

Basically, API is a URL provided in AJAX function to fetch (get) the data in JSON format.

So,

1) An application sends data to a server
2) The server retrieves the data and interprets it
3) Some action is performed on the data.
4) Server sends the information back to the application
5) The application interprets the data.
6) The application presents the interpreted information in readable way.

5) JSON :- JavaScript Object Notation

- It is used to exchange data between a browser and a server.
- JSON is used more than XML because it is lighter.
- It is similar to Javascript Objects, It also have "Name" : value pair, but Name must be text/string written in double quotes

JSON Methods :-

1) JSON.parse() :- It is used to convert the JSON string data into a javascript object

eg:- var obj = JSON.parse(text);

2) JSON.stringify() :- It is used to convert the js object into JSON.

eg:- var myJSON = JSON.stringify(myObj);

6) AJAX Request :- Ajax communicates asynchronously with the server using the XMLHTTPRequest object.

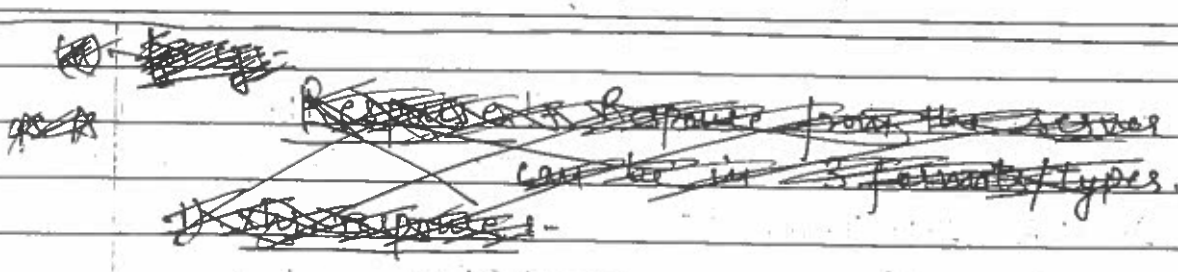
- To send an HTTP request, create XMLHTTPRequest object, open a URL & send the request.

(i) for eg:- <button id="btn">Fetch Image</button>


```
document.getElementById("btn").addEventListner("click",
                                        fetchImage);
function fetchImage(){
    var xhr = new XMLHTTPRequest();
    xhr.onload = function(){
    var responseJSON = JSON.parse(this.response);
    var image = responseJSON.message;
document.getElementById("img").setAttribute("src","image");}
    xhr.open("get","url of image website", true);
    xhr.send();}
```

Process of Last Example :-

1) Select a button element and add click event & function.
2) create the event function.
3) create XMLHTTPRequest object in event function.
4) Now add "load" event and function to xHR object.
5) This.response :- XMLHTTPRequest.response used to collect the response from API(URL) website and convert it into js object.
6) Now response object have message property which contains the Image URL.
7) Add ~~the~~ new attribute (using setAttribute) "src" and 'image URL will be set to the value of src.
8) xhr.open() :- It is used to open a request. It takes 3 parameters.
   i) method :- get/post/head/put/delete etc
   ii) API(URL)
   iii) true :- Asynchronous Request and false :- Synchronous Request
9) xhr.send() :- is used to send a request.



Response :- There can be 3 types of Server Response.

i) Text ii) XML data iii) JSON data

Response is divided into 5 states :-

* .onreadystatechange :- used to check the state of Response.
  0 : Request not initialized.
  1 : Server connection established.
  2 : Request received.
  3 : Processing request.
  4 : Request finished and respons is ready

* .responseText :- If the response is text/string.

* .responseXML :- If the response is XML data.

* Response Status is already discussed.
  200 : OK 404 : Not found.

for eg :-

```
<p id="demo"> Image </p>
<button onclick="textChange"> CLICK </button>
function textChange(){
 var xhr = new XMLHTTPRequest();
 xhr.onreadystatechange = function(){
if( this.response == 4 && this.status == 200){
 document.getElementById("demo").innerText = this.response;
  }
  };
xhr.open("GET","URL",true;);
xhr.send();
  }
```

Not necessary but given to understand

7) AJAX with JQuery :-

JQuery is a shorthand of Javascript. ie Javascript library.

1) $.ajax() :->

```
$.ajax({
    url : 'http://____',
    method : 'get',
    success : function(response) {
        var image = response.message;
        $("#img").attr('src', image);
    });
```

used to add new attribute to selected element →

2) $.get() :- Used only for get method.

```
$("#btn").click(fetchImage);

function fetchImage() {

$.get({'url', function(response){
        var Image = response.message;
    $('#img").attr('src', image); }
    });
}
```

8) AJAX Error Handling :- There can be any reason to get errors while ajax request. To handle this :-

1) xhr.onerror :→ continue to example of page - 129.

```
xhr.onerror = function() {
    console.log("Request Failed");
};
```

Use this before xhr.open();

2) .fail() :- used in $.ajax() continue to example of page - 132

```
.fail(function() {
    console.log("Request Failed");
});
```

3) .fail() :- used in $.get()

```
.fail(function(xhr, textStatus, errorThrown) {
    console.log("Request Failed");
});
```

(25) → Promise :- Promises are used to handle asynchronous http requests and events.

- It is an object takes only one argument which is a callback/anonymous function.
- Callback/anonymous function takes two two arguments, (i) resolve (ii) reject.
- resolve :- If the operations inside the callback functions are perfomed and everything went well then call resolve.
- reject :- If the operations doesnot go well then call reject.

for eg :-

```
var promise = new Promise(
        function(resolve, reject){
  let x = 5;
  let y = 5;
  if(x===y){
                resolve(); }
  else          { reject(); }
});
promise.then(function(){
        console.log("Success");
}).catch(function(){
        console.log("Error");
});
```

★ then() :- When a promise is resolved then() is invoked.

- It may also be defined as a carrier which takes data from promise and further executes it successfully.
- It takes a function as an argument which will execute if a promise is resolved.

★ catch() :- catch() is invoked when a promise is rejected.

- It takes one function as a argument to handle errors or promise rejections.

★★ Promise :- A promise is an object that represents either completion or failure of a user task.

- As the name suggests a promise is either completed (resolve) or rejected.

for eg :-

```
var userLoggedIn = true;

function checkUserLoggedIn() {
var promise = new Promise((resolve,
                reject) =>
setTimeOut(() => {
    if (userLoggedIn) {
        resolve("User logged In");
    }
    else { reject("failed logIn"); }
}, 5000);
```

Arrow function as parameter

```
return promise;
}

checkUserLoggedIn().then((successMsg) =>
        { console.log(successMsg);}).
    catch((error) => { console.log
        (error);}) ;
```

(27.) fetch() :-

It is an alternative method of AJAX and XMLHTTPRequest.

- It is used to insert, update, get/read and delete the data into the server.

Syntax :-

```
fetch(url)
.then((response) => response.json())
.then((data) => {
  // code to handle the response data
    })
.catch((error) => {
// code to handle any errors that occured.
    });
```

- The fetch() function takes a URL as its parameter and sends a "GET request" to that URL to fetch the response.
- It returns a promise that shows the completion or failure of the request.
- In first .then() method JSON data is extracted from the response using response.json().
- Second .then() is used perform further manipulations of the

// JSONPlaceholder.com then go to guide.

28) Async and await :- The keyword "async" before a function makes the function return a promise.

for eg:
```
async function test() {
    return "Hello"; }
```
→ Now this function will return a promise.
→ Promise always returns a successful resolve in .then() and a failure in .catch().

```
test().then((result) => { console.log(result) }).
.catch((error) => { console.log("error") });
```

→ The await keyword can only be used inside an async function.
→ The await keyword makes the function pause the execution and wait for a resolved promise before it continues.

for eg:-
```
async function test() {
        console.log(1);
await console.log(2);
        console.log(3); }
console.log(4);
test();
console.log(5);
```

O/P:
4
1
2
5
3
(After 2, the function will wait to complete the remaining code then come back to 3).

async & await with fetch() :-

```
async function test () {
    const response = await fetch("url");
    const students = await response.json();
    return students;
                }
test().then((result) => {
        console.log(result) })
.catch((error) => {
        console.log("error message") });
```

(29) Try & Catch :-

The try statement allows you to define a block of code to tested for errors while it is being executed.
The catch statements runs, if an error occurs in the try block.

for ex

```
async function test() {
    try {
const response = await fetch("url")
const students = await response.json();
    return students;
        }
    catch (error) {
        console.log(error.message) }
    }
test().then((result) => {
        console.log(result) }
```

(30) Template Literals / strings :-

- It allow you to use strings or embedded expressions in the form of a string.
- They are enclosed with backticks (`${}`) (dollar sign)
- You don't have to use string concatenation or escape characters. (\n).

for eg :-

```
var name = "World";
var lname = "India";
console.log(`Hello, ${name}!
Welcome to ${lname}`);
```

O/P :-

```
Hello World!
Welcome to India.
```

Without template literals :-

```
console.log("Hello" + " " + name \n + "Welcome to" +
            " " + lname);
```

for eg 2 :-

```
let fname = "Dinesh";
let lname = "Kumar";
function fullname(fname, lname) {
  return `${fname} ${lname}`; }
let hello =
  `Hello ${fullname(fname, lname)}`;

Console.log(hello);
```

(31) Rest Operators :- Rest operators used in a function to take an indefinite number of arguements and bundle them in a array.

- This allow us to write functions that can accept a ~~unknown~~ number of arguements, irrespective of the number of parameters defined.

for eg:-

```
function sum(name, ...args) {
    console.log(name);
    let sum = 0;
    for(var i in args) {
        sum += args[i]; }
    console.log(sum); }
sum("Yahoo kaka", 20, 30, 40)
```