24/12/24

Web Technology.

UNIT-1 :

HTML common tags & Java Script.

HTML common tags - Images, Lists, Tables, Forms, Frames, Graphics, Media, APIS.

Cascading Style sheet.

Java Script - Overview of Java Script, Primitives, Operations, expressions, Control statements, Arrays, Functions, Pattern matching using regular expressions

Introduction to HTML :

i) HTML is the standard markup language for creating webpages. It is not a programming language.

ii) HTML stands for HyperText Markup language.

iii) Markup language is set of tags (< >)

iv) HTML describes the structure of webpage.

v) HTML consists of a series of elements.

vi) HTML elements tell the browser how to display the content

vii) HTML elements label pieces of content such as "This is a heading", "This is a paragraph", "This is a link", etc.

viii) HTML file is always saved with list of .htm or .html

ix) HTML programs always start with <html>

x) Structure of html : <!DOC TYPE HTML>

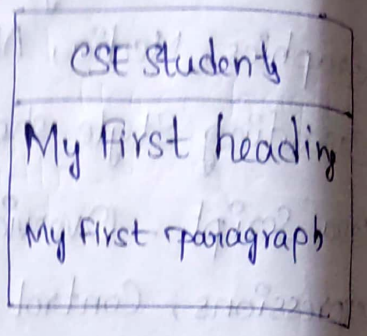
<html>
<head> . . . </head>
<body>
. . .
</body>
</html>

</body>
</html>

Ex:

```
<!DOCTYPE html>
<html>
<head>
<title> CSE Students </title>
</head>
<Body>
<h1> My First heading </h1>
<p> My First paragraph </p>
</body>
</html>
```

o/p:



⇒ <!DOCTYPE html> declares define that this document is an html document.

⇒ The html element is the root element of an html page. <html>

⇒ <head> The head element contain meta information about the html page

⇒ <title> The title element specifies a title for html page. (which is shown in the browsers title bar (or) in the pages tab)

⇒ <body> The body element defines the documents body and is a container for all the visible content such as heading, paragraph, images, hyperlinks, Tables, etc.

⇒ <h1> The h1 element defines a large heading

⇒ <p> The p element defines a paragraph.

26/12/24

HTML History :

The early days of the world wide web there have been many versions of html.

Year	Version
1989	Tim Berneus lee invented www
1991	Tim Berneus lee invented HTML
1993	Dave Raggett drafted HTML+
1995	HTML working group defined HTML 2.0
1997	W3C recommendation HTML 3.2
1999	W3C recommendation HTML 4.01
2000	W3C recommendation HTML 1.0
2008	WHAT WG HTML5 1st public draft
2012	WHAT WG HTML5 living standard
2014	W3C recommendation HTML5
2016	W3C Candidate recommendation HTML5.
2017	W3C recommendation HTML5.1 2nd edition
2017	W3C recommendation HTML 5.2

Images :- HTML.

Images can improve the design and the apperance of a webpage.

Syntax : <img src = "url" alt = "alternate text">

⇒ The HTML <img> tag is used to embed an image in a webpage

⇒ Images are not technically inserted into a webpage, images are linked to webpages.

⇒ The image tag is empty, it contains attributes only & <img> doesn't have a closing tag.

⇒ The <img> tag has 2 required attributes (1) scr - it specifies

the path to the image (or) alt - it specifies an alternate text for the image

Ex: <html>
<body>
<h2> HTML Image </h2>
<img src = "img_flower.jpg" alt = "Flower" height = "600"
width = "500">
</body>
</html>

⇒ Image size width & height we can use the style attribute to specify the width and height of an image

Ex: <img src = "img_girl.jpg" alt = "Girl in Flower"
style = "width: 500px; height: 600 px;">

⇒ The height & width attributes always define the width & height of the image in pixel.

⇒ Width & height are style :

The width & height & style attributes are all valid in HTML.

The style attributes is prevents the style sheets from changing the size of images

```
<html> <head>
<style>
 img {
      width : 100%;
     }
</style > </head>
<body>
```

<img src = "html5.gif" alt = "HTML5 Icon" width = "128" height = "128" >

<img src = "html5.gif" alt = "HTML5 Icon" style = "width: 128 px ; height : 128 px ; " >

</body>

</html>

Images in Another folder:

If we have images in a subfolder, we must include the folder name in src attribute.

Ex : <img src = "/images/ html5.gif" alt = "HTML5 Icon" style = "width : 128 px ; height : 128 px ; " >

Images on another server / website:

* Some websites point to an image on another server.
* To point to an image on another server, we must specify an full URL in the src attribute.

Ex : <img src = "https://www.w3schools.com/images/ w3schools-green.jpg" alt = "w3schools.com" >

Image floating:

It uses the CSS float property to let the image float to the right or to the left of a text.

Ex : <img src = "smiley.gif" alt = "Smiley face" Style = "float : right ; width : 42 Px ; height : 42 Px ;" >

The image will float to the right of text

 <img src = "smiley.gif" alt = "Smiley face" Style = "float : left ; width : 42 Px ; height : 42 Px ;" >

The image will float to the left of the text

Background Image:

* To add a background image on an HTML element, use the HTML Style attribute & the CSS background image property.

Ex: <P Style = "background-image : url ('img-girl.jpg');

* We can also specify the background image in the <style> element, in the <head> section.

Ex:
```
<html>
<head>
<style>
body (or) P {
    background-image : url('img-girl.jpg');
}
</style>
</head>
<body>
<h3> Girl is very beautiful </h3>
</body>
</html>
```

27/12/24

Lists:

HTML list allow web developers to group a set of related items (or) It is the collection of one (or) more items. HTML supports 3 types of lists.

(1) Unordered list (bullets)
(2) Ordered list (Numbers)
(3) Definition list

(1) Unordered list :

⇒ An Unordered list start with an <ul> tag. Each list item starts with the <li> tag.

⇒ The list item will be marked with bullets by default. (small black circle)

```
<body>
<h2> An Unordered HTML List </h2>
<ul type = "circle" >
<li> coffee </li>
<li> Tea </li>
<li> Milk </li>
</ul>
</body>
```

o/p :

- coffee
- Tea
- Milk

(2) Ordered list :

⇒ An Ordered list starts with the <ol> tag.

⇒ Each list item starts with <li> tag.

⇒ The list items will be marked with numbers

```
<body>
<h2> An ordered list HTML List </h2>
<ol type = "numbers" >
<li> coffee </li>
<li> Tea </li>
<li> Milk </li>
</ol> </body>
```

o/p :

1. Coffee
2. Tea
3. Milk.

(3) Definition list :

⇒ A definition is not a list of items, this is a list of terms and explaination of term. &

⇒ A definition list start with <dl> tag

⇒ Each definition list term start with <dt> tag

⇒ Each definition list definition start with <DD> tag.

Syntax:
```
<DL>
  <DT> Text </DT>
  <DD> text </DD>
  <DT> text </DT>
</DL>
```

eg:
```
<dl>
  <dt> coffee </dt>
  <dd> - Black hot drink </dd>
  <dt> Milk </dt>
  <dd> - White cold drink </dd>
</dl>
```

O/P:
coffee
- Black hot dri
Milk
- white cold drink

28/12/24

HTML Table:

HTML Tables allows web developers to arrange data into rows & columns (OR) Tables can be defined as the intersection of rows & columns.

⇒ A Table in HTML consisty of table cells inside rows & columns

eg:
```
<html>
<style>
table, th, td {
    border : 1Px Solid black;
}
</style>
<body>
< table Style = "width :100 %" >
<tr>
<th> Company </th>
```

```
<th> contact </th>
<th> country </th>
</tr>
<tr>
<td> Infosys </td>
<td> Anusha </td>
<td> Germany </td>
</tr>
<tr>
<td> Infosys </td>
<td> Alekhya </td>
<td> India </td>
</tr>
</table>
</body>
</html>
```

O/P:

company	contact	country
Infosys	Anusha	Germany
Infosys	Alekhya	India

Table cells:

⇒ Each table cell is defined by a <td> tag & a </td> tag

⇒ td stands for table data

⇒ Every thing between <td> & </td> are content of the table cell.

Eg:
```
<body>
<table style = "width : 100%">
<tr>
<td> Email </td>
<td> Tobias </td>
<td> Linus </td>
</tr>
</table>
</body>
```

O/P:

Email	Tobias	Linus

Table rows:

→ Each table row starts with a <tr> tag & end with a </tr> tag

→ <tr> stands for table row.

eg:

```
<body>
<table style="width: 100%">
<tr>
<td> email </td>
<td> Google </td>
<td> whatsapp </td>
</tr>
<tr>
<td> India </td>
<td> Israel </td>
<td> Russia </td>
</tr>
</table>
</body>
```

o/p:

email	Google	whatsapp
India	Israel	Russia

Table header:

→ Sometimes we want your cells to be table header cells

→ In those causes use the <th> tag instead of <td> tag

→ <th> stands for table header

eg:

```
<body>
<table style = "width : 100%">
<tr>
<th> P1 </th>
<th> P2 </th>
<th> P3 </th>
</tr>
<tr>
<td> Pen </td>
<td> Pencil </td>
```

```
<td> Ink </td>
</tr>
</table>
</body>
```

30/12/24

o/p:

P1	P2	P3
Pen	Pencil	Ink

HTML FORMS:

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

The <form> element:

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

Syntax:
```
<form>
    .
    .
  form elements
    .
    .
</form>
```

⇒ The <form> element is a container for different types of input elements such as text field, check boxes, radio buttons, submit buttons, etc...

The <input> element:

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

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

Ex:

TYPE	Description
<input type="text">	Displays a single line text input field
<input type="radio">	Displays a radio button (for selecting one of many choices)
<input type="checkbox">	Displays a checkbox (for selecting zero or more of many choices)

<input type="submit"> - Displays a submit button (for submitting the form)

<input type="button"> - Displays a clickable button

Text fields:

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

The <input type is text & value of this text field " " that means blank is displayed initially & we can enter text of our choice into it

There is a size parameter which allows us to enter some size of text field. Some other parameter can be name which indicates name of text field.

Password:

It works exactly as text but content is not display on screen & instead '*' is used.

Check boxes:

It is simple component which is used to portation when we want to make some selection from several options.

Radio button:

This form component is used to indicate selection from several choices using <input type="radio">. We can place this on web page this component allows us to make only one selection at a time.

Button:

There are 2 types of buttons that can be created in HTML. One is submit & other is Reset / Cancel.

In various parameters of submit buttons are:
name - denotes name of submit button.
value - writing some text on button
align - it specifies alignment of button.

Text area: Text field is a form component which allows us to enter single line text. What if you want multiple lines text we must use text area field.

Syntax: <textarea cols="n" rows="n" name="string">

* Rows denotes total no. of rows in text area. Col represents total no. of columns in text area.

Name: Denotes name of text area which can be utilized for handling that component for specific purpose.

Menus: HTML allows us to have top down menu on webpage so that desired selection can be made. The parameter <select> is for menu component & <option> parameter is for selecting values to the option of drop down menu. We can make some specific option selected by using selected.

Ex: form.html

```
<form>
<h1 align="center"> Email- Registration form </h1>
<hr width="100%">
Name : <input type="text" size=30 value=" "> <br>
Age : <input type="text" size=30 value=" "> <br>
UId : <input type="text" size=30 value=" "> <br>
pswd : <input type="password" size=30 value=" "> <br>
gender : <input type="radio"> male & nbsp (non breaking space)
```

<input type="radio"> female

Interested Fields :
<input type = "check box"> IT Professional
<input type = "check box"> Manager
<input type = "check box"> Other
country :
<select size = 1 >
<option selected > India </option>
<option> UK </option>
<option> US </option>
</select>
Address :
<text area cols = 50 row = 10 >
</text area >
<input type = "submit" value = "submit" align = "center">

<input type = "submit" value = "cancel" align = "center" >
</form>
</body>
</html>

Servers { ASP - Active Server Page
JSP - Java Server Page
CGI - Common Gateway Interface

2/1/25

HTML form attributes:

Attributes of <form> tag

① action
② Method
③ Enctype (Encryption type)

1) Action: It is used to determine where to send data. If specify URL (Uniform Resource Location) to which form data will be submitted.

→ we would specify URL of a program on a server/an email

eg: <form action = "data.jsp">

2) Method: It is an attribute of form tag.

→ It determines how form data will be submitted.

⇒ The two options of this attributes is "get" & "Post".

get: i) It appends the form data into the URL.

ii) Server received URL + data

3) We are getting the data from the server.

4) Here no. URL is required.

post: i) It sends the data seperately.

ii) It is considered as the preferred option.

3) Enctype:

⇒ It specify the format of the data being submitted.

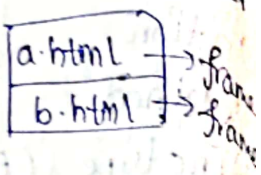
⇒ It specify an encoding protocol known as Multipurpose Internet Mail Extension (MIME).

⇒ MIME ensure the data, doesn't become corrupted when transmit across the internet

<form action = "data.asp" method = "post" Enctype = "plain/text">

HTML Frames: (Divide browsers)

→ It can display one (or) more than one html doc in the same browser window.



→ Each html doc is called frame & each frame is independent of others.

→ <frameset> tag is used to divide browser window.

→ <body> tag is not required.

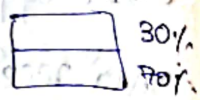
Attributes of Frameset Tag:

→ Rows
→ Columns
→ Frame border
→ Bordercolor
→ Name

Rows: It divides browser window row wise

a.html

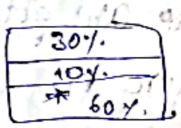
```
<html>
<frameset rows = "30%, 70%">
```



(or)

```
<frame set rows = "30%, 10%, *">
```

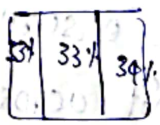
(* → rest %)




```
</frameset>
</html>
```

Columns: It divides browser window column wise

```
<html>
<frameset cols = "33%, 33%, 34%">
  <frame src = "a.html">
  <frame src = "b.html">
  <frame src = "c.html">
</frameset>
</html>
```



⇒ Canvas element shows 4 elements. They are:

① A red rectangle
② A gradient rectangle
③ A multicolor rectangle
④ A multicolor text

⇒ The canvas element is only a container for graphics we must use javascript to actually draw the graphics.

⇒ Canvas has several methods for drawing.

⇒ Paths, boxes, circles, text & adding images.

⇒ Canvas is supported by all browsers [Opera, Google chrome, Internet explorer, Mozilla firefox]

Eg: A canvas is a rectangular area on a HTML page by default a canvas has no border & no content.

```
<canvas id = "mycanvas" width="200" height="100" >
</canvas>
```

⇒ It specify an id attribute [to be referred to in a script] & a width & height attribute to define the size of the canvas.

⇒ To add a border use "style" attribute.

```
<canvas id = "mycanvas" width="200" height="100"
style = "border : 5px solid green;" >
Your browser doesn't support HTML canvas tag.
</canvas>
```

O/P:

Add a java script:

After creating the rectangular canvas area, we must add a java script to do the drawing

```
<canvas id = "mycanvas" width: "200" height = "100"
    Style = "border : 1Px Solid green ;" >
```

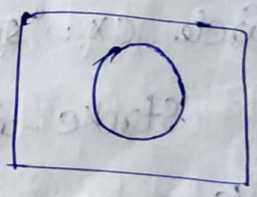
O/P:

```
<script>
var c = document.getElementById("mycanvas");
varctx = c.getcontent.("2d");
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
</script></body></html>
```

Draw a circle :

```
<canvas id = "my canvas" coldth = "200" height = "100"
        Style : "border : 1Px Solid red;" >
Your browser doesn't support, the HTML canvas tag
</canvas>
<script>
var c = document.getElementById("my canvas");
var ctx = c.getcontext("2d");
ctx.beginpath();
ctx.arc(95,50,40,0,2*mathPI);
ctx.stroke();
</script>
</body>
</html>
```

o/p:



Draw a Text :

```
<script>
varc = document.getelement ById("My Canvas");
varctx = c.getcontant("2d");
ctx.font = "30px Arial";
ctx.fillText("Hello world", 10,50);
</script)
```

o/p :

Helloworld

Alignment change

HTML SVG:

⇒ It is used to define graphics for the webpage.
⇒ It defines vector based graphics in XML, which can be directly embedded in HTML Page.
⇒ Each element & attribute in ~~such~~ SVG file can be animated.
⇒ SVG can be implemented without using Javascript also.
⇒ SVG elements supports google Chrome (4.0), internet Explorer (9.0), Mozilla Firefox (3.0), Opera (9.1).
⇒ Suppose if your browser not having latest version then this SVG not support.

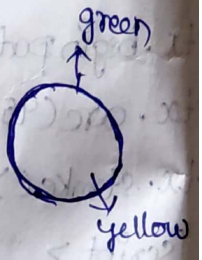
The <SVG> element:

⇒ The HTML <SVG> element is a container for SVG graphics.
⇒ SVG has several methods for drawing paths, boxes, circles, text & graphic images.

① SVG circle:

```
<body>
<svg width="100" height="100">
<circle cx="50" cy="50" r="40" stroke="green"
    stroke-width="4" fill="yellow" />
</svg>
```

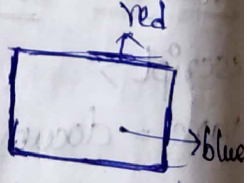
O/P:



② SVG Rectangle:

```
<svg width="100" height="120">
<rect x="10" y="10" width="200" height="100"
  stroke="red" stroke-width="6" fill="blue" />
Sorry ur brower doesn't support inline SVG
</svg>
```

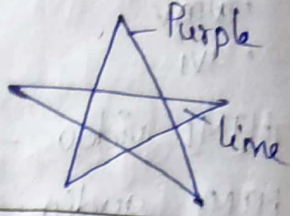
O/P:



③ SVG Star:

```
<svg width = "300" height = "200">
<Polygon points = "100, 10, 40, 198, 190, 78, 10, 78, 160, 198"
    style = "fill : lime ; stroke : Purple ; stroke_width : 5;
    fill-rule : even odd ;" />
</svg>
```

or:



SVG	canvas
① Resolution independent	① Resolution dependent
② Support for event handlers	② No support for event handler
③ Good text rendering capabilities	③ Poor text rendering capabilities.
④ Slow rendering if complex	④ We can save the resulting image as .png & .jpg
⑤ Not suited for game applications	⑤ Well suited for graphic intensive games.

4/01/2025

Media:

Multimedia on the web is sound, music, videos, movies and animations. Multimedia comes in many different formats like images, music, sound, videos, records, films, animations, etc.

It contain multimedia elements of different types & formats. The first webbrowser had support for text only limited to a single font in a single color.

Later came browsers with support to colors, fonts, images & multimedia. Multimedia elements (like audio or video) are stored in media files. The multimedia files have

formats & different extensions like :

- wav
- mp3
- mp4
- mpg
- wmv
- avi

HTML video → <video> <source> </video>
HTML audio → <audio> <source> </audio>
HTML Pluging → <object>, <embed>
HTML Youtube → <iframe>

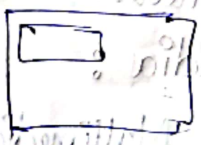
HTML video :

autoplay → video plays automatically

⇒ The HTML video element is used to show a video on a web page.

```
<body>
<video width="320" height="240" controls>
<source src="movie.mp4" type="video/mp4">
</video>
</body>
```

o/p :



⇒ The "control" attribute add video controls like play, pause, volume and it always include width & height attributes.

→ If height & width are not set the page height flicker while the videos loads.

⇒ The source element allows you to specify alternate video files which the browser may choose from. the

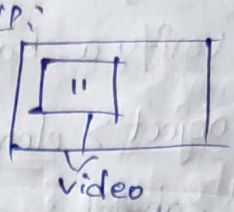
⇒ The text between the <video> & </video> tags will only be displayed in browsers that don't support the video element

HTML video autoplay :

→ To start a video automatically use the autoplay attribute.

```
<body>
<video width = "320" height = "240" autofile >
<source src = "movie.mp4" type = "video/mp4" >
</video>
</body>
```

o/p:



HTML audio :

→ The HTML audio element is used to play an audio file on a webpage.

The HTML audio element :

To play an audio file in HTML use the audio element

```
<body>
<audio controls>
<source src = "music.mp3" type = "audio/mpeg" >
</audio>
</body>
```

o/p:

HTML audio autoplay :

To start an audio file automatically to use the auto play attribute

```
<body>
<audio controls autoplay/autoplaymuted >
<source src = "music.mp3" type = "audio/mpeg">
</audio>
</body>
```

HTML Plug-ins :

Plug-ins were designed to be used for many different purposes.

→ To run java applets.
→ To run microsoft active x controls.
→ To display flash movies
→ To display maps.
→ To scan for viruses
→ To verify a bank id.

The <object> element :

→ The <object> element is supported by all browsers.
→ The <object> element defines an embedded obj within an HTML doc.
→ It was designed to embed plug-ins (like java applets, PDF readers & Flash Players) in webpages but can also be used to include HTML in HTML.

```
<body>
<h1> Hello welcome to my channel </h1>
<object width = "100%" height = "500px" data = "v.html">
</object>
</body>
```

The <embed> element :

→ The <embed> element is supported in all major browsers.
→ The <embed> element also defines an embedded obj within an HTML doc.
→ Web browsers have supported the <embed> element for a long time. However, it has not been apart of the HTML specification HTML 5.

```
<body>
<h1> Hello </h1>
<embed src = "1.jpg">
</body>
```

HTML youtube :

→ The easiest way to play videos in HTML, is to use youtube.

→ Converting videos to diff formats can be difficult & time consuming

→ An easier solution is to let youtube play the video's in your webpage.

Displaying a youtube video in HTML :

To play your video on a webpage, do the following :

* Upload the video to youtube.
* Take a note of the video id
* Define an <iframe> element in your webpage.
* Let src attribute point to the video URL.
* Use the width & height attributes to specify the dimensions of the player.
* Add any other parameters to the URL (see below)

```
<body>
<iframe width="420" height="345"
        src="https://www.youtube.com/embed/
             tgbNymZ7vqY">
</iframe>
</body>
```

Youtube autoplay + Mute :

We can let your video start playing auto vertically when a user visits the page, by adding autoplay=1 to the youtube URL.

Add mute = 1 after autoplay=1 to let your video start playing automatically (but muted).

<iframe width="420" height="345" src="https://www.youtube.com/embed/tgbNYM27VqY?autoplay=1&mute=1">
</iframe>

Youtube loop:

Add playlist = videoID & loop=1 to bt your video loop for even.

loop = 0 (default) - The video will play only once.

loop = 1 The video will loop (for even)

<iframe width="420" height="345" src="https://www.youtube.com/embed/tgbNYM27VqY?playlist=tgbNYM27VqY&loop=1">
</iframe>

videoID

Youtube controls:

Add controls = 0 to NOT display controls in the video player.

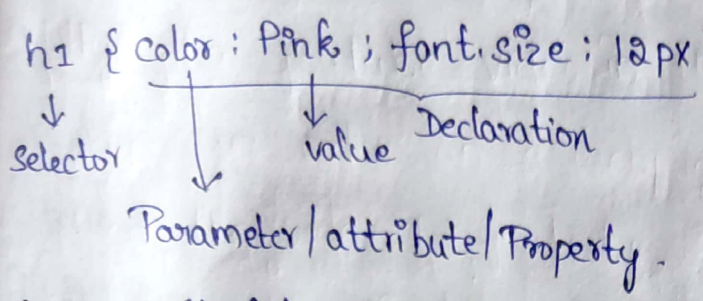
Controls = 0 - Player controls doesn't display

Controls = 1 - (default) - Player control is displayed.

<iframe width="420" height="315" src="https://www.youtube.com/embeded/tgbNYM27VqY?controls=0">
</iframe>

Cascading Style Sheet [CSS]:

⇒ It help us to specify presentation of elements on webpage.
⇒ CSS is a simple mechanism for adding style to web documents.
⇒ External Style Sheet can solve a lot work & then are stored in CSS files.
⇒ A CSS rule has 2 main parts. They are
① Selector ② Declaration.



Selector : It determines the elements to which rule is applied.
Declaration : It specify the exact property value to be applied to the element.

There are 3 ways to insert CSS. They are :
① External Style Sheet
② Internal Style Sheet
③ Inline Style sheet

External SS : This is one type of Style sheet which is used to applied styles to our web pages.
⇒ Here we write to seperate program & we will save it as .CSS extension.
⇒ When we want to link external SS then we have to use <link> tag, which is to be written in head section.
⇒ <link> tag tells that browsers some file must be linked to page.
rel → This stylesheet tells the browser that this linked thing a SS.
href → Path denotes the "Path name of Stylesheet file".
type → Text & CSS tells the browser that what it is reading is text i.e. effected by CSS.

Internal ss : This is also called as embeded ss.

⇒ This is also same as external ss but the difference is here we wont seperate program instead, we simply write all style sheets in head section itself.

Inline ss : An inline ss style can be used, if a unique style to be applied to one single occurance of an attribute

⇒ To use inline styles use the style attribute can contain any css property.

Program :

css.html :

```
<html>
<title> frame 1 </title>
<frame set rows = "20%, *">
<frame src = "fill.html" name = "frame 1">
<frameset cols = "20%, *">
<frame src = "fill 2.html" name = "frame 2">
<frame src = "fill 3.html" name = "frame 3">
</frameset>
</frameset> </html>
```

fill 1.html :

```
<html>
<body bgcolor = "pink">
<h1 align = "center" text = "red">
<marquee align = "light">
Introduction to css </marquee> </h1>
</body>
</html>
```

fill2.html :

```
<html>
<body bgcolor = "Pink" >
<P align = "left">
<a href = "external.html" target = "frame3" > external css
</a></P>
<P align = "left" >
<a href = "inline.html" target = "frame3"> inline css </a>
</P>
</body>
</html>
```

fill3.html :

```
<html>
<body bgcolor = "Pink">
</body>
</html>
```

23/01/25

Expressions : It is any unit of code that can be evaluated to value is an expression.

Ex: var x, y, z;
z = x + y;

Types of expressions :

* Arithmetic expression
* String expression
* Logical expression
* Primary expression
* Left hand side expression
* Assignment expression
* Expression with side effects

"function call is an exp where normal if-else/for loops are not expressions."

Arithmetic exp :

This evaluate to a numeric / any value through arithmetic operators is comes under arithmetic exp.

eg: Here '10' is an exp that is evaluated to the numeric value '10'.

String exp: String exp's are exp's that evaluate to a string

Ex: 'hello';

'hello' + 'world'; //evaluates to string

o/p : 'hello world'

```
var fname = "Kuchala";
var lastname = "Sree";
var age = 16;
var S = "This is" + fname + " " + lastname + ", I'm" + age +
        "years old";
document.writeln(S);
```

O/P : This is Kuchala Sree, I'm 16 years old.

Logical exp: Exp's that evaluate to the Boolean value. True or False are considered to be logical exp's. This exp often involve the usage of logical operators are :

&& (AND), || (OR) & ! (NOT).

eg: 10 > 9 ; // evaluates to boolean value true.

10 > 20 ; // evaluates to boolean value false.

10 > 9 && 10 > 20 // evaluates to boolean value false.
T F

Primary exp's: It refer to stand alone exp's such as literal values, certain key words & variable values.

eg: var a = 100;

Lefthand-side exp's : It also known as left values, left-hand side exp's are those that can appear on the left side of an assignment exp.

ex: var p;
p = 100;

Assignment exp : When exp's use the '=' operator to assign a value to a variable. It is called an assignment exp.

eg: avg = 55;

Exp's with side effects : Exp's with side effects are those that result in a change of a side effect such as setting or modifying the value of a variable through the assignment operator '=', function cal, incr or dec value of a variable.

eg: sum = 20; sum++;

Conditional control statements | Decision making Control statements | Branching Control statements | Selection control statements :

⇒ Conditional control statements works based on a condition.

⇒ To represent the condition we mainly use relational & logical operators. &, |, ! <, >, <=, >=, !=, ==

⇒ Conditional control statements are classified into 4 types.

① If stmt ② if-else stmt ③ else-if ladder | nested if else | if-else-if stmts

④ Switch.

1] simple if stmt :

→ Simple if stmt is the most simple decision making stmt.

→ Simple if is used, if there is only one condition.

Syn : if (condition)
{
stm-block;
}

eg: <html>
<head>
<script>
age = parseInt (prompt ("Enter age:"));
if (age >18)
document.write ("<h1> Eligible for voting");
</script> </head>
<body> </body> </html>

if-else stmt:

if-else is used when there are 2 conditions.

Syn: if (condition)
{
True block stmt;
}
else
{
false block stmt;
}
Statement-x;

eg: <html>
<head>
<script>
age = parseInt (prompt ("Enter age"));
if (age >18)
document.write ("<h1> Eligible for voting");
else
document.write ("<h1> Not eligible for voting");
</script> </head>
<body> </body> </html>

else-if ladder (or) restricted if else (or) if-else-if stmt:

If there are more than 2 conditions then else-if ladder will be used.

Syn:

```
if (cond 1)
Stmt - Block 1 ;
else if (cond 2)
Stmt - Block 2 ;
else if (cond 3)
 Stm - Block 3 ;
 - - -
 - - -
 - - -
else
  Stmt - Block n ;
Statement - X ;
```

Ex: <head>

```
<script>
a = parseInt (Prompt ("Enter first No."));
b = parse Int (Prompt ("Enter second No."));
op = Prompt ("Enter an operator:");
if (op == "+")
 res = a+b;
else if (op == "-")
res = a-b;
else if (op == "*")
 res = a*b;
else if (op == "/")
 res = a/b;
else if (op == "%")
res = a%b;
document.write ("<h1> result is :" + res + "</h1>");
</script> </head>
```

Switch: Switch is mainly used when we have a no. of choices & we may need to execute a diff task for each choice.

Syn:

→ int | float | char | string

```
Switch (exp)
{
case value 1: stmt block 1;
              break;
case value 2 : stmt - block 2;
              break;
   :  :
default : default - block ;
          break;
}
statement - x;
```

Ex:

```
<head>
<script>
a = parseInt ( Prompt ("Enter first no."));
b = parseInt ( Prompt ("Enter second no."));
op = prompt ("Enter an operator");
switch (op)
{
case "+" : res = a+b;
           break;
case "-" : res = a-b;
           break;
case "*" : res = a*b;
           break;
case "/" : res = a/b ;
           break;
}
document . write ("<h1> result is :" + res + "</h1>");
</script> </head>
```

Looping control stmts (or) Iterative control stmts :

Looping control stmts are used to execute a set of stmts repeatedly.

Types of looping control stmts:

* for loop * while loop * do-while loop

for loop : It is mainly used to execute a set of stmts repeatedly

syn: for (initialization ; condition ; inc/dec)
{
stmt-block;
}
stmt-x;

Initialization :- It means assigning a value to the variable & for initialization "=" operator is used.

Ex: i=1; A=10; total=100;

Condition :- Relational & logical operators are mainly used for representing conditions.

ex: a>10; B==10; (a<b) && (a<c)

Inc/Dec :- Inc & Dec operators are used for updating the value of variable. Ex: a++, a--

Ex :
```
<html>
  <head>
    <script>
      var i;
      for(i=1; i<=10; i++)
      document.write("<h1>i=" +i+ "</h1>");
    </script>
  </head>
  <body> </body>
</html>
```

o/p : i=1
i=2
:
:
i=10

while loop: It is used to execute a set of stmts repeatedly

Syn:
```
while (condition)
{
stmt-block;
}
stmt-x;
```

Ex:
```
<script>
var i;
i=1;
while (i<=5)
{
document.write("<h1>i=" +i+ "</h1>");
i++;
}
</script>
```

O/P: i=1
i=2
i=3
i=4
i=5

do-while loop: It is used to execute a set of stmts repeatedly

Syn:
```
initialization
do
{
stmt-block;
}
while(cond);
stmt-x;
```

Ex:
```
<script>
var i;
i=1;
do
{
document.write("<h1>i=" +i+ "</h1>");
i++;
}
while (i<=7);
</script>
```

O/P: i=1
i=2
i=3
i=4
i=5
i=6
i=7

Arrays in Java script :

→ An array is a group of similar data types items that can share a common value (OR)

An array represents a group of elements of same data type

→ An array index begins from zero.

Creating an array : An array can be declared in 3 ways

① Syn : var array-name = new Array( ) ;
→ constructor (method)

Ex : var ax = new Array ( ) ;

② var array-name = new Array (list of values) ;
↓ to create values dynamically

Ex : var ax = new Array ("Nehal", "Venu", "Sai") ;

③ Var array-name = [list of values] ;

Ex : var ax = ["Nehal", "Venu", "Sai"] ;

Adding elements to an array :

Syn : Array_name [index Position] = value ;

Ex : robo[0] = "Nehal" ;
robo[1] = "Venu" ;

Accessing Array Elements :

The elements in the array are accessed through their index.

Syn : Array_name [index Position]

Ex : robo[0]
robo[1]

Ex :
```
<html>
<head><title> Array </title></head>
<body>
<script language = "Javascript">
var robo = [10,20,30,40,50];
```

```
for (var i=0 ; i<5 ; i++)
for (var i=0 ; i<robo.length ; i++)
{
document.write(robo[i]);
}
</script></body></html>
```

r[0]	r[1]	r[2]	r[3]	r[4]
10	20	30	40	50
0	1	2	3	4

O/P: 10 20 30 40 50

Functions in Javascript:

Function is a self contained block of stmts which can perform a specified task [OR]

→ Function can be defined as a collection of stmts which perform a given task.

→ Functions are mainly useful for reusability & modular programming

" In java script there is no need of any fun declaration i.e. fun

Function definition:

Functions are defined using the function keyword followed by function-name

Syn:
```
function function_name (Parameter1, P2, ... Pn)
{
  Body;
}
```

Function call: A function is called using the function name followed by a list of parameters seperated by commas enclosed in paranthesis terminates with semicolon.

Syn: function function-name (P1, P2, ... Pn);

Ex: ①
```
<html>
<head>
<title> Sample fun </title></head>
<body>
<script language="javascript">
```

```
function robo()
{
  document.writeln("Hai");
}
robo();
</script>
</body></html>
```

②

```
<html>
<head>
<title>Sample function</title>
<script language="Javascript">
function robo()
{
 document.writeln("Hai");
}
</script></head>
<body>
<script language="Java script">
robo();
</script></body></html>
```

script

Java Built-in functions:

⇒ Function is a reusable code-block that will be executed whenever it is called

⇒ Function is a great time save.

⇒ It is used for performing repetative task where, you can call the same fun multiple times to get the same effect

⇒ It allows code reusability

⇒ Java script provides no.of built-in functions.

Common Built-in functions in Javascript:

Functions	Description
isVar() [Not a number]	* Return true, if the obj is not a number. * Return false, if the obj is a number
ParseFloat (string)	* If the string begins with num, the fun reads through the string until it finds end of the num; * If the string doesn't begin with a num, the fun returns var (not a num)
String (object)	Convert the obj into a string
eval()	Returns the result of evaluating an arithmetic expression

User defined functions:

User defined fun means you can create a function for your own use. You can create yourself according to your need.

Syn:

```
<script>
function function-name (parameter)
{
statements;
return
}
functional call
</script>
```

ex:

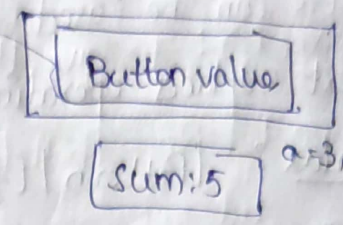
```
function add()
{
var a,b;
var sum=0;
sum = a+b;
```

document.write("Addition:"+sum);

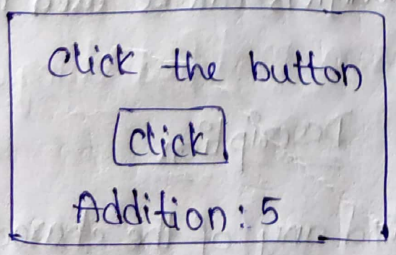
How is a fun executed on an event in a Javascript:

Input type = "button"
onclick = "add()"
value = "button value"
onclick = Event Handler
add() = function name

Ex: <html>
<body>
<script type = "text. javascript">
function add()
{
var a = 2; b = 3;
var sum = 0;
Sum = a+b;
document.write(" Addition : "+sum)
}
</script>
 click the button
<Input type = "button" onclick = "add()"
Value = "click" >
</body>
</html>



O/P:



Pattern matching using Regular expressions: 25/01/25

A regular expression is a sequence of characters that forms a search pattern. The search pattern can be used for text search text to replace operations.

⇒ A regular expression can be single character (or) more complic pattern.

⇒ RE's can be used to perform all types of text search & text replaces operations

Syntax: /Pattern/[flags] RegExpliteral Notation

ex: /hello/

start /^[0-9]{10}$/ → end.

using RE constructor method.

new RegExp ('Pattern', [flags])

eg:
```
<body>
<script>
let re = /hello/ (or)
new RegExp ("hello")
let res = re.test ("hello Javascript")   O/P: True.
console.log (res)
</script>
</body>
```

Regular Expressions modifiers: Modifiers can be used to perform multiline searches.

Ex:

Expression	Description
[abc]	Find any of the character inside the bracket.
[0-9]	Find any of the digits blw the brackets 0 to 9.
(x|y)	Find any of the alternatives blw x & y, seperated with |

Regular Expression Patterns: Meta characters are characters with a special meaning.

Ex:

Meta character	Description
\d	Used to find a digit
\s	Used to find a white space character
\b	Used to find a match at begining or at the end of a word
\uxxxx	Used to find the unicode character specified by the hexadecimal number xxxx

Quantifiers: It defines quantities.

Ex:

Quantifier	Description
n+	Used to match any string that contains atleast one 'n'.
n*	Used to match any string that contains '0' or more occurances.
n?	Used to matches any string that contains '0' or one occurance of 'n'.

Ex:

① Let re = /hello/i; (∵ i = flag, i means incase sensitive)
/i.g (global)

```
let res = re.test("Hello javascript")
console.log(res)
```

O/P: True.

② Let re = /^[0-9]{10}$/

```
let res = re.test("07867 46241")
consde.log(res)
```

O/P: True

③ let re = /^h/

```
let res = re.test("thello")
console.log(res)
```

O/P: False

To test the pattern has 2 methods.

1) test() 2) Exec() (:: exec → executable constructor)

① test():

eg:- let re = /hello/
let res = re.test("hello javascript") o/p: True
console.log(res)

② Exec(): To give the some extra information

eg: ① let re = /hello/g
let res = re.Exec("Hello javascript hello")
console.log(res)
console.log(res.index) o/p: error

② let re = /hello/g
let res = re.Exec("hello javascript hello")
console.log(res)
console.log(res.index) o/p: 0, 17

Using String Methods:

Regular Expressions are often used with the two string methods

① search() ② replace()

⇒ The search() method uses an expression to search for a match and returns the position of the match.

⇒ The replace() method returns a modified string where the pattern is replaced.

Using string search() with R.E:

eg: function myfunction()
{ //input string
var str = "Javascript follows the syntax & structure of the C programs!";

```
// searching string with modifier i
var n = str.search(/Javascript/i);
document.write(n + '<br>');
//searching string without modifier i
var n = str.search(/Javascript/);
document.write(n)
}
myfunction();
```

replace:

```
function myFunction()
{
var str = "Javascript follows the syntax & structure of
          the programming!";
var txt = str.replace(/C programming/i, "C programming
          language");
document.write(txt);
}
myFunction();
```

O/P: Javascript follows the syntax & structure of the C programming Language!