mood-book

WEB PROGRAMMING
UNIT - 1 SCRIPTING

Introduction

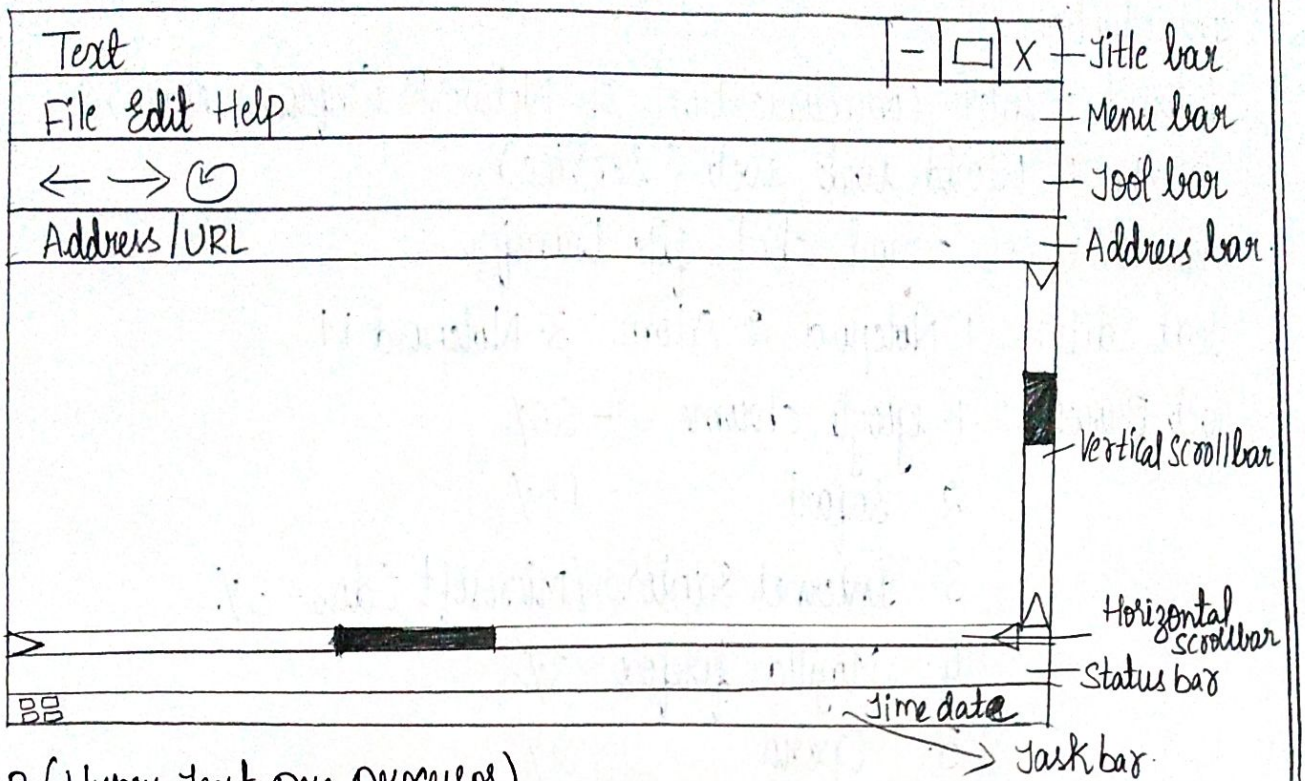
Internet : Inter-communication of Network (infrastructure)
WWW : World wide web (Service)
Other services : Email, chat, file transfer
Text Editor : 1. Notepad 2. Atom 3. Notepad++
Web Browser : 1. google chrome - 65%
2. Safari - 15%
3. Internet Explorer / Microsoft Edge - 5%
4. Mozilla firefox - 3%
5. Opera - 2%
File Extension : html (or) htm
Programming paradigms : Imperative (how to do)
Declarative (what to do)

Scripting Vs Programming

→ Scripting uses Interpretation
Ex: VB script, Java script, PHP, Ruby, perl - - -
→ Programming uses compilation
Ex: C, C++, Java, Python, Cobol

Java Vs JavaScript.

→ JavaScript is developed by Netscape Communications cooperation
→ Initially they used to call JavaScript as LiveScript.
→ SunMicrosystems and Netscape had an agreement JavaScript as complimentary Scripting language for Java.
→ Syntax of JavaScript is similar to syntax of Java.



PHP (Hyper Text pre processor)
It is a server side scripting language.

HTML (Hyper Text Markup Language)

→ It is a markup language.
→ It is case insensitive
→ It is tag-based language.

There are 2 types of tags:

→ Paired tags
→ Singular tags (unpaired/Empty/stand-alone)
→ Tag is used to represent in pointed brackets (< >)
→ paired tag consist of both opening tag and closing tag.

<tagname> opening tag
</tagname> closing tag

Singular tag consists of only opening tag

<tagname> This is the content </tagname>

Content

Element

Attributes are used to customize tags

<tagname>, attribute1 = value1 attribute2 = value2

Structure of html:

```
<!DOCTYPE html>
<HTML>
    <HEAD>
    </HEAD>
    <BODY>
    </BODY>
</HTML>
```

HTML 4.01

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
    "http://www.org/TR/html4/strict.dtd">
```

W3C: world wide web Consortium

DTD: Document Type Definition

Comment: <!-- This is comment -->

Categories of tags:

1. Head tags
2. Text Formatting tags
3. Linked tags
4. List tags
5. Table tags
6. Image and object tags
7. Form tags
8. Script tags

Head Tags:

1. <title>: To define title for html document
2. <style>: To define internal styles.
3. <link>: To attach external style sheets
4. <meta>: To define meta data of html document

5. <Script>:

6. <base> : charset
   href - (hreference) → default URL
   * target → _blank — parent
   — Self — top.

Ex: <!DOCTYPE html>
<HTML>
<HEAD>
<TITLE> my first web page </TITLE>
</HEAD>
</HTML>

Text-Formatting tags:

<h> heading tags from <h1> to <h6>

<h1> heading 1 </h1>
<h2> heading 2 </h2>
<h3> heading 3 </h3>
<h4> heading 4 </h4>
<h5> heading 5 </h5>
<h6> heading 6 </h6>

The following are the units used for specifying size of the font.

- pt (point), 72 points per inch
- px (pixel) — Smallest element on screen, 96 pixel per inch
- em (Element), rem (root Element) — relative units

12pt = 16px

1em = 16px

Ex:

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

 to define paragraph.
browser will add a blank line before and after element automatically.

 text inline container.
<b> bold text - (physical tag)
 Strong text (important text) (logical tag)
<i> Italic tag.
 Emphasized text (logical tag)
<SUP> Super script text E=mc² → $E=mc^2$
<SUb> Sub script text
<u> underlined text
 Division container
<pre> pre-formatted text.

```
<pre> line1
      line2
      line3
</pre>
```

<small> Smaller text.
<mark> marked text / highlighted text
 this is <mark> Marked Text </mark>
→ highlight color will be Yellow.
<marque> moving text / scrolling text.
<marque> this is moving text </marque>
<ins> Inserted Text
<del> deleted text
Color of rose is <del> blue </del><ins>red</ins>
o/p: color of rose is red

Link tags:
<a> Anchor tag
href (Hyperlink reference) URL
target.

There are the three states for hyperlink:
Unvisited (blue)
Visited (purple)
Active (red)

 google

List Tags:

 To define unordered list
 list item
 To define ordered list

Start
type 1 (default)
a
A
i
I

Eg:
```
<ul>
  <li>C</li>
  <li>Java</li>
  <li>Python</li>
</ul>
```

Olp:
- C
- Java
- Python

<ol start=3 type="a">	default if 'i'
olp: c	3 iii
d	4 iv
e	5 v

Eg:
```
<ol>
  <li>C</li>
  <li>Java</li>
  <li>Python</li>
</ol>
```

Olp:
1. C
2. Java
3. Python

<dl> to define definition list
<dt> to define definition term
<dd> Definition description

Eg:
```
<dl>
  <dt>C</dt>
  <dd>About C</dd>
  <dt>Java</dt>
  <dd>About Java</dd>
</dl>
```

Olp:
C
About C
Java.
About Java.

Table Tags:

<table> To define a table
<tr> Table row
<th> Table header
<caption> To define caption for a table.
<thead> To group head contents
<tbody> To group body contents
<tfoot> To group footer contents

Ex:
```
<!DOCTYPE html>
<html>
<body>
<table>
<caption> Student Table </caption>
<tr>
<th> Name </th>
<th> Age </th>
</tr>
<tr>
<td> abc </td>
<td> 20 </td>
</tr>
</table>
</body>
</html>
```

Op:

Student Table

Name	Age
abc	20

Image and Object tags:

<img> To embed image in html document

- src (source) - path/URL of image

  file extensions :- (i) .jpeg - (Joint photographic Experts group)
  (ii) .gif - (graphics Interchange format)
  (iii) .png - (portable Network graphics)

- alt (Alternative text)
- height
- width

Ex: `<img src="filename.jpeg" alt="Rose flower">`

<figure> To define figure (image + caption)

<figcaption> To define caption for a figure.

Ex:
```
<figure>
<img src="filename.jpeg" alt="Rose flower">
<figcaption> Fig:1 Rose flower </figcaption>
</figure>
```

<Object> To define a container for an external resource (document, audio, video, etc ---)

- data - filename
- height
- width

Eg: <Object data = "Filename.mp3" height = "200" width = "200">
</Object>

<audio> To embed audio content

- src
- Controls

audio file extensions
- mp3 (MPEG layer-3) Moving pictures Expert group
- wav (wave file format)
- ogg (ogg vorbis Compressed audio format)

Eg: i) <audio src = "Filename.mp3" Controls>
</audio>

ii) <audio controls>
<source src = "Filename.mp3" type = "audio/mp3">
<source src = "Filename.ogg" type = "audio/ogg">
</audio>

<video> To embed video content

- src
- width
- height
- Controls

file extensions
- mp4 (MPEG layer-4)
- WebM (Web Media file)
- Ogg (Ogg theora Video format)

Eg: i) <video src = "Filename.mp4" width = "200" height = "200" controls>
</video>

(ii)
```
<video controls>
  <source src="Filename.mp4" type="video/mp4">
  <source src="Filename.ogg" type="video/ogg">
</video>
```

<iframe> To define inline frame

- src
- width
- height
- title

Eg:
```
<iframe src="Filename.html" width="200" height="200">
</iframe>
```

CSS3 (cascading style sheets level 3)

It is used to style html document

Syntax:

```
h1 { color: blue ; fontsize: 10px }
```

(h1: Selector; color: blue; fontsize: 10px: Declaration; color: property; blue: value)

There are 3 types of CSS

1. In-line styles - one element
2. Internal styles - one webpage
3. External styles - one website

Inline styles:

It is used to apply styles for single element. In this style attribute is used to specify styles.

Ex:
```
<!DOCTYPE html>
<html>
  <body>
    <p style="font-size:30px; color=blue">
       This is my paragraph
    </p>
  </body>
</html>
```

Internal styles:
It is used to apply styles for a webpage.
In this <style> tag is used in <head> tag to specify styles.

```
<style>
   type text/css.
```

Ex:
```
<!DOCTYPE html>
<html>
  <head>
    <style>
       p{font-size:30px; color:blue}
    </style>
  </head>
  <body>
     <p> This is my paragraph </p>
  </body>
</html>
```

There are 5 types of selectors
1. Element selector
2. Id selector
3. class selector
4. Universal selector
5. group selector.

1. Element Selector: In this tag name is used as selector.
Ex: (prev Ex)

2. ID selector: In this unique ID is defined for a single tag.
Ex:
```
<!DOCTYPE html>
<html>
 <head>
  <style># para{ font-size: 30px; color: blue} </style>
 </head>
 <body>
    <p id = "para"> This is my paragraph </p>
 </body>
</html>
```

3. Class selector: In this class name is defined and it can be used for any tag
Eg:
```
<!DOCTYPE html>
<html>
  <head>
   <style>
     .para { font-size: 30px; color: blue }
   </style>
   </head>
    <body>
      <p class = "para"> This is my paragraph </p>
    </body>
</html>
```

4. Universal Selector: This is used to apply styles for all the tags.

Eg:
```
<!DOCTYPE html>
<html>
  <head>
    <style>
      * { font-size: 30px; color: blue }
    </style>
  </head>
  <body>
    <p> This is my paragraph </p>
  </body>
</html>
```

5. Group Selector: It is used to apply same styles for different tags.

Ex:
```
<!DOCTYPE html>
<html>
  <head>
    <style>
      P, td { font-size: 30px; color: blue }
    </style>
  </head>
  <body>
    <p> This is my paragraph </p>
  </body>
</html>
```

External Styles:

It is used to apply styles for (multiple webpages) entire website.

It is used with <link> tag under <head> tag.

<link> (i) rel "stylesheet"
(ii) href Filename/path.css } most often used attributes

Eg: External.html

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="filename.css">
  </head>
  <body>
    <p> This is my paragraph </p>
  </body>
</html>
```

filename.css

```
p {color: blue; font-size: 30px}
```

CSS properties:

i) Text properties: text-align: left
right
center

ii) Font properties:
Font family: times new roman
verdana
georgia.
Font-size: px
em

iii) Table properties:
border: thickness style color.

border-style : solid
dotted
dashed
double
insert
groove
outset

border collapse : color: name/Hex/rgb
width
height
padding

Ex: color : black
color : #000000
color : rgb(0,0,0)

Other properties : background-color
foreground-color

background-image : url(path)

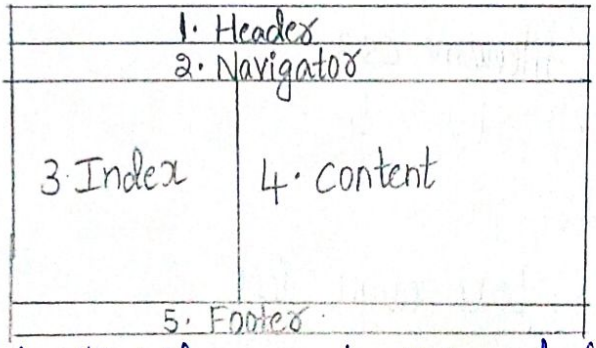
background-attachment : scroll
fixed

background-repeat : no-repeat
repeat

background-position : center
left
right

Layout Management:

Table Layout:




```
<body>
 <table style="width:100%">
   <tr>
    <td colspan=2 style="height:40px; background-color:red">
    =
    </td>
   </tr>
   <tr>
    <td colspan=2 style="height:40px; background-color:grey">
    =
    </td>
   </tr>
   <tr style="height:400px">
    <td style="width:25%; background-color:blue">
    =
    </td>
```

```
    <td style = "background-color: green">
      ==
    </td>
  </tr>
  <tr>
    <td colspan=2 style = "height:40px; background-color: Yellow">
    </td>
  </tr>
</table>
</body>
```

Division Layout:

```
<head>
 <style>
    .wrap{width:100%}
    .header{height:40px; background-color:red}
    .nav{height:40px; background-color:grey}
    .ind{width:25%; height:400px; float:left; background-color:
                                                        blue}
    .cont{height:400px; width:75%; float:right; background
                                                -color:green}
    .foot{height:40px; background-color:yellow}
    .clear{clear:both}
 </style>
</head>
<body>
  <div class = "wrap">
    <div class = "header">
      ==
    </div>
    <div class = "nav">
      ==
    </div>
```

```
<div>
  <div class = "ind">
    ==
  </div>
  <div class = "cont">
    ==
  </div>
  <div class = "clear"> </div>
 </div>
 <div class = "foot">
    ==
 </div>
</div>
</body>
```

Forms: The <form> element is used to create forms with different user input fields.

<form> • action
• method get (default) URL
• target post attached to body (secured)

→ <input> element is used to specify different form input fields.

<input> • type → text, password, radio, checkbox, submit, reset, button, file, email, date

• name • disabled
• value • readonly
• size • id.

→ <label> element is used to create label.

<label> • for

Note: To bind label with input, id attribute of <input> and for attribute of <label> should be same.

Eg:
```
<!DOCTYPE html>
<html>
<body>
<h4> Sign up </h4>
<form action="/actionpage.php">
<label for="fname"> Firstname: </label>
<input type="text" id="fname"> <br>
<label for="lname"> Lastname: </label>
<input type="text"> id="lname"> <br> <br>
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Input type checkbox: It is used to create checkboxes

Eg:
```
<!DOCTYPE html>
<html>
<body>
<h4> programming languages: </h4>
<form action="/actionpage.php">
<input type="checkbox" name="chkb1" value="c" id="cb1">
<label for="cb1"> c </label> <br>
<input type="checkbox" name="chkb2" value="c++" id="cb2">
<label for="cb2"> c++ </label> <br>
<input type="checkbox" name="chkb3" value="Java" id="cb3">
<label for="cb3"> Java </label>
</form>
</body>
</html>
```

Input type radio : It is used to create radio button

Eg:
```
<!DOCTYPE html>
<html>
<body>
<h4> gender : </h4>
<form action = "/actionpage.php">
<input type = "radio" name = "rb" value = "male" id = "rb1">
<label for = "rb1"> Male </label>
<input type = "radio" name = "rb" value = "female" id = "rb2">
<label for = "rb2"> Female </label>
</form>
</body>
</html>
```

Dropdown list:

<Select> tag is used to create drop-down list

- name
- id
- size (visibility)
- multiple

<option> tag is used to specify list item.

- value
- selected
- disabled.

Eg:
```
<!DOCTYPE html>
<html>
<body>
<form action = "/actionpage.php">
<label for = "dl"> State : </label>
<Select name = "states" id = "dl">
```

```
<option selected disabled> choose your state </option>
<option value = "TS"> Telangana </option>
<option value = "AP"> Andhrapradesh </option>
<option value = "AM"> Assam </option>
</select>
</form>
</body>
</html>
```

Input type date: It is used to create date in the form.

```
Eg: <label for = "dob"> Date of birth: </label>
    <input type = "date" name = "date of birth" id = "dob">
```

Input type button: It is used to create clickable button.

```
Eg: <input type = "button" name = "my button" onclick = js function()
        value = "DSB">
```

→ <button> tag is also used to create clickable button.

- type - button, submit, reset
- name
- value
- onclick

It is a paired tag, content can be defined.

```
Eg: <button type = "button" name = "mybutton" value = "DSB"
        onclick = jsfunction()>

    <img src = "image.jpg" alt = "my image">

    </button>
```

<textarea> tag is used to create multiline textbox.

- rows (no. of lines Visibility)
- columns (width)

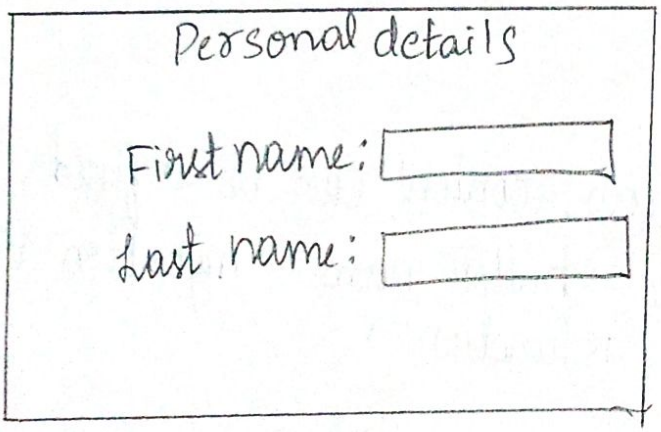
Eg: <textarea rows = "10" cols = "30">
</textarea>

<fieldset> tag is used to group input fields.

<legend> tag is used to specify caption for fieldset.

Eg:

```
<fieldset>
    <legend> personal details </legend>
    <label for = "fname"> First name: </label>
    <input type = "text" name = "first name" id = "fname"><br>
    <label for = "lname"> Last name: </label>
    <input type = "text" name = "Last name" id = "lname">
</fieldset>
```



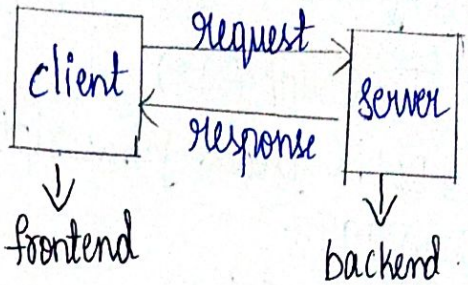
Scripting

Client Side Vs Server Side → Both are independent

Client Side → less secure

Server Side → more secure

Basic client server architecture



→ The script which is written on client is client side & it runs on web browser.

→ Every web browser provides a Engine to run the scripting

→ It has JS as default script.

Eg: Chrome - V8 (engine) → 70% users use it

firefox - Spyder monkey (engine)

Safari - webkit (engine) (or) JavaScript Core webkit

Edge - Chakra (engine) → Script permission should be enable manually.

Script enable permission is given automatically

Server-side

→ The script which is written on server is called Server-side & it runs on Server-machine.

→ The scripting changes according to the applications.

Eg: PHP, Ruby, Perl, Python - - -

→ Mostly PHP is in use

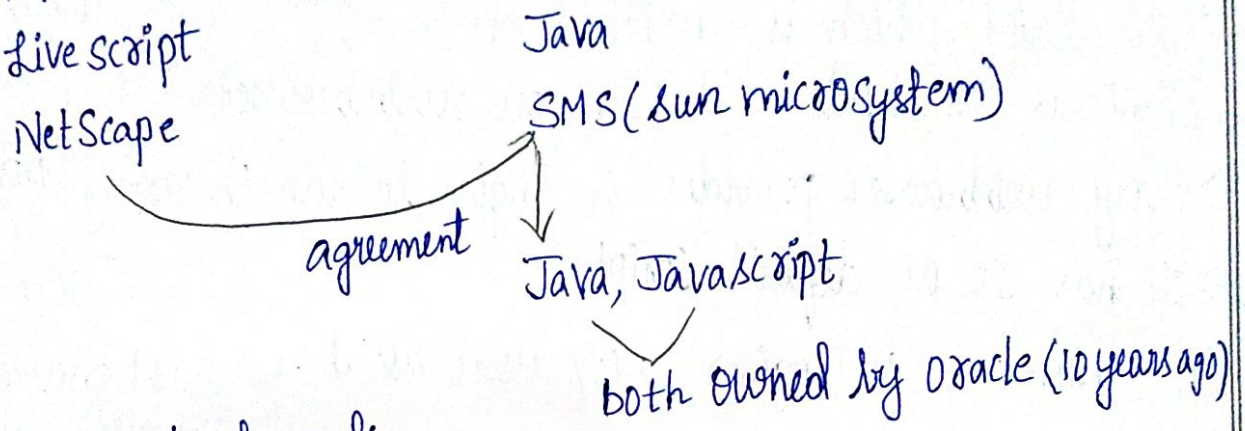
Why Scripting

→ client-side scripting is used for creating Interacting pages & Validation

→ Server-side scripting is used for generate dynamic response since each client response in different way. So this dynamic response is called as "tailor-made response"

JavaScript (JS)

→ Both java & javascript (livescript at start) came into existence in 1995.



→ Java is trademark

1. Javascript (JS) is a light weight, interpreted, object-oriented programming language (also used for scripting).
   Java is strongly-typed (give core importance to datatype, checks datatypes)

→ JS is lightweight since it is weakly typed.
(no datatype usage)
(they are generic)

HTML → content , CSS → styling & Layout , JS → behaviour.

2. HTML defines content of the page.
3. CSS defines styling and Layout of the page.
4. Javascript defines behaviour of the page.

5. JS is prototype-based programming, It is a kind of Object-oriented programming.

6. Prototype means an object, it defines behaviour and it can be reused.

Where to use <script> tag : head & body (can write in multiple) → for functions we use scripting in head
↓ end of body - to speed up loading

What is document in html :
White space where html components marked up on browser.

7. In HTML, JS is inserted in <script> tag.

8. <script> tag can be written in both <head> & <body> tags.

<script> → default JS
type text/javascript (script type mostly JS)
src filename.js (external js file to attach)

Note: Paranthesis ( ) - method/function
no paranthesis - property.

9. External JS file (filename.js) can be attached to the HTML document using "src" attribute of <script> tag.

10. It is possible to write multiple script tags within a document.

11. Usually <script> tag is written at the end of the body to improve/reduce loading time (since script is interpreted the page is loaded fastly)

12. Javascript follows syntax of C & Java programming languages.

Input/output

- innerHTML (property) element (display content on element)
- document.write() → document (display content on document)
- document.writeln() → document (next line) - special reqⁿ.
- window.alert() → Alert dialog box (display at top after message)
- window.confirm() → confirm dialog box (has true or false) Ok/cancel
- window.prompt() → input

innerHTML:

→ To display output (Content) in a HTML element, (Tag)
'innerHTML' property is closed
(info b/w opening & closing tag)

getElement (Lower camel case) GretElement (upper camel case)

Ex:

```
<!DOCTYPE html>
  <html>
    <body>
      <p id = "para"></p>
      <script>
          document.getElementById("para").innerHTML="My paragraph";
      </script>
    </body>
  </html>
```

→ used to fetch paragraph by id

Semicolon is mandatory in <script> tag.

document.write() & document.writeln()

→ To display output on the document "document.write()" (or) "document.writeln()" is used.

2 situations

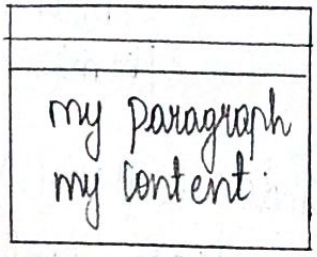
i) While loading (script is executing)
↓
write() can execute so no problem

ii) After loading - script is executing
↓
write() if executes then data erases.

Ex: <!DOCTYPE html> (while loading)

```
<html>
  <body>
    <P> My paragraph </P>
    <script>
      document.write("my content");
    </script>
  </body>
</html>
```

Output:



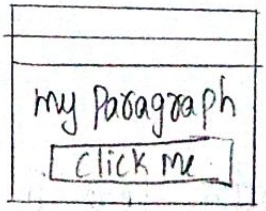
Note: for next line ⇒ write("content" + "
")
↓
for writeln "(ln)" is appended automatically.

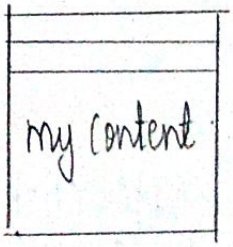
Ex: (After loading)

```
<!DOCTYPE html>
<html>
  <body>
    <P> My paragraph </P>
    <button type="button" onclick="document.write("my content")">
      Click me
    </button>
  </body>
</html>
```

Output:

After pressing button ⟹





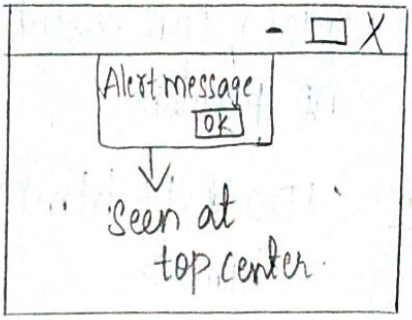
Note: If "document.Write()" is executed after loading the page, it erases the previous content and string specified in the method will be marked up.

window.alert()

To display alert message "window.alert()" is used

Eg:
```
<!DOCTYPE html>
<html>
<body>
<script>
window.alert("Alert Message");
</script>
</body>
</html>
```

Output:



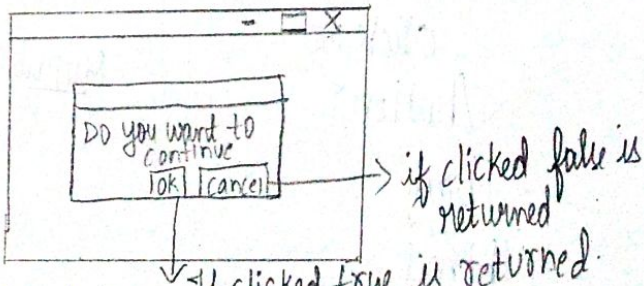
after clicking on 'OK' button, nothing is returned.

window.confirm()

To display confirm message "window.confirm()" is used.

Eg:
```
<!DOCTYPE html>
<html>
<body>
<script>
window.confirm("Do you want to Continue");
</script>
</body>
</html>
```

Output:



→ After clicking on 'OK' button 'true' is returned, for 'cancel' button 'false' is returned.

Window.prompt( )

→ To read input "window.prompt( )" is used.

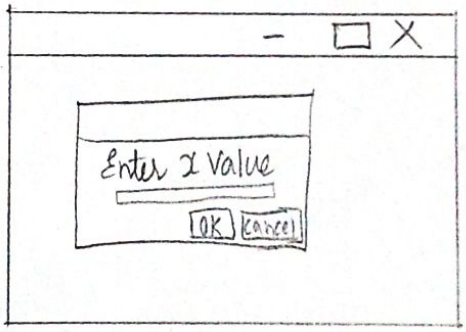
→ It takes 2 parameters i.e one is compulsory parameter & other is optional parameter

Syntax: prompt(string, Default string)
↳ optional.

Eg: <!DOCTYPE html> (execute)

```
<html>
  <body>
    <script>
        window.prompt("Enter x value");
    </script>
  </body>
</html>
```

Output:



→ It is used less often since we don't read the input most of the time since want to write script for existing element.

→ After clicking on 'OK' button, input string is returned & for cancel button null is returned.

→ It is mostly used for strings.

Variables: It is a container which stores a value with a name.

→ Variable is a container for storing data.

→ In JavaScript, variables can be declared using any one of the following keywords.

- Var
- let
- Const

→ 'let' & 'const' are introduced in the year 2015.

→ These two keywords supports block scope { }, cannot be redeclared.

→ Whereas, variables declared using 'var' keyword won't support block scope (supports global & local/function scope), can be redeclared.

→ In javascript there are 3 possible scopes.

1. global scope
2. local/function scope.
3. Block scope ({ }) → block representation

→ The range of statements in which a variable is visible/accessible is called as scope.

→ The amount of time in which a variable holds it's value is called as life-time.

→ After declaring a variable lifetime starts, lifetime ends for a global variable after closing web browser for local variable after executing a function, for block variable after executing a block.

→ If a variable is declared & not defined it stores undefined.

i.e var x; // declaration

x=100; // definition.

Ex: ① Var x = 10;
⋮
Var x = 100;
⇒ Var allows redeclaration

② let x = 10;
⋮
let x = 100; X
→ let doesn't allow redeclaration

③ Var x = 10;
≡
{
Var x = 100;
≡
}
x ? 100
→ Var doesn't support block scope so it takes x = 100 as redeclaration

④ let x = 10;
{
let x = 100;
≡
}
x ? 10
→ Block scope completes & again x = 10 since global.

→ Multiple variables can be declared in single declaration
Eg: Var x = 10, y = 20, z = 30;

→ Declaration can span in multiple lines
Eg: Var x = 10,
y = 20,
z = 30;

→ Keyword is a reserved word which have predefined meaning

Comments

→ Single line comments are represented using '//'

→ Multiline comments are represented using "/* ---- */"

Identifiers:

→ Identifier is a user-defined name created for user-defined things (variables, constants, function, objectclasses, -- etc)
→ Identifier name is constructed using (a-z)(0-9)(_)&($)
(Space not allowed)
→ It shouldn't start with a digit
→ Keywords shouldn't be considered as Identifier names

Operators

→ In JavaScript, there are 6 categories of operators:

1) Arithmetic operators (+, -, /, %, *, ++, --)
2) Relational operators
3) Logical operators
4) Assignment operators
5) Bitwise operators
6) Special operators.

1) Arithmetic operators:

+, -, *, /, %, ++, --

2) Relational operators

== equal to - loose equality (compares only value)
!=
<
<=
>
<=
=== identical - strict equality (compares both value & type)
!== not identical

→ number is promoted to string then compared.

Ex: ① 5 == "5" - true
(5 → string)

② 5 === "5" - false
number string

③ 5!="5" - false ④ 5!=="5" - true.

3) Logical operators
&& logical AND (2 parameters) → 2T's - T } binary operator
|| logical OR (2 parameters) → 2F's - F }
! logical NOT (1 parameter) → T|F, F|T } unary operator

4) Assignment operator
= is equal to
op= shorthand assignment
Eg: x = x+10;
x+=10;

5) Bitwise operators
& bitwise AND
| bitwise OR
~ bitwise NOT
^ bitwise X-OR
<< shift left
>> shift right
>>> shift right with zeros

<<: Syntax a << n
a → operand
n → no. of positions to shift left.

→ we can deal with +ve & -ve
→ fill LSB with '0' in empty space while left shift
→ each left shift ⇒ value multiplied by '2' for both +ve & -ve.
→ '-ve' is stored in 2's complement format.

>>:

Syntax a>>n

→ fill MSB with '0' if +ve, '1' if -ve

→ each right shift ⇒ value is divided by '2' for both +ve & -ve

Eg 3>>1 = 1

>>>:

Syntax a>>>n +ve } fill MSB with
-ve } 0

Eg: a = -4

4 = | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |

11111011
+ 111

11111100

a>>>1

| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | ⇒ | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | ⇒ 126 //

Special operators:

→ In Js we have

?: - Conditional operator (Ternary operator)

,

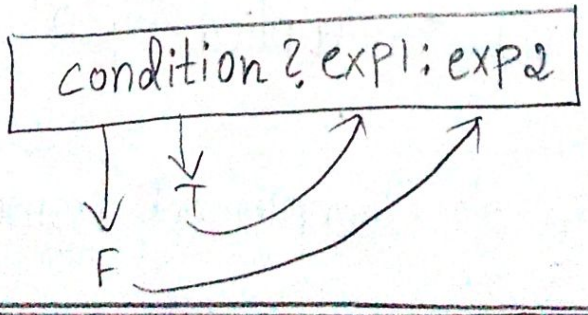
new

typeof

instanceof

?:

Syntax:



,: comma operator evaluates expressions from left to right last operand value is assigned to the variable.

```
x = 10
x = (x++, x)
      ||
      ||
      ||
```

new: new operator is used to create an instance (object) (it is similar to java)

Ex:

```
const obj = new student("abc", "AIML", 1234);
```

obj/instance — class — properties of class student

typeof: typeof operator is used to know the datatype of a variable/value

```
x = 10
```

Ex: typeof x //Number.
typeof(x)
typeof "hello" // string

instanceof: instanceof operator is used to check whether an identifier name is object of class or not.

Eg: Obj instanceof student // True or False.

Obj → identifier name
student → class.

Datatypes

JavaScript has following datatype they are.

1. String
2. number - can store upto $(2^{53}-1)$
3. boolean
4. Object
5. function
6. Symbol ES(2015) → (European Computer Manufacturer's Association Script
   EIMAS → ES → define rules for scripting
7. BigInt (2020) → $(2^{53}-1)$ >
8. null → Internationally no value.
9. undefined → having no value.

→ every variable is associated with a datatype.

→ we don't use these for declaring variable

→ we use var to declare value

let

const

ex: var x;
typeof x // undefined (since we defined variable)

→ If any variable is undefined then it's type is also undefined

ex: var x = 10;
typeof x // number
typeof "hello" // string

→ If we intentionally assign "null" to a variable, then its type is null

→ BigInt has a number which is greater than $2^{53}-1$ size it is denoted by appending 'n' at end (introduced in 2020)

ex: 9112345689123456 7n = let x

(or)

let x = BigInt(9112345689123456 7)

typeof x // bigint

→ Symbol is introduced 2015, it is used to define unique (↓ no value) identifier & provide optional description used to avoid duplication.

ex: `Var x = Symbol("id")` → 'x' is unique wherever we use 'x' it's unique. (→ capital)

Note: `Symbol() == Symbol()` // false (↳ always false)

→ object is similar to dictionaries in Python declared in curly braces { } & has key: value pair seperated by commas(,)

→ boolean - true or false.

→ number - int & float less than $2^{53}-1$ size.

→ string - associated with " " & ' '

→ A single quote should be enclosed with double quotes & vice versa

Eg: 'Hello "world"' (or) "Hello 'world'" (✓) 'Hello 'world'' (✗)

→ function - we can declare parameters in function.

Eg: function func()

typeof fun // function

→ null & undefined are not same (or) not identical

`null === undefined` // false

Control statements: used to control flow of execution

1. Selection statements 2. loop statements 3. other statements related to loops.

① Selection statements: Selection statement meant to select plot based on condition (syntax is similar to c)

if - simple block of statements

if-else - 2 options

multiway selection:
- else if - It is a ladder (used for multiple conditions)
- switch - we have integral expression, based on that exp we perform case, if any one of case is true it's executed else if goes to default. we use break for each & every case.

② loop statements:

Pretest loop:

for - for(declaration; condition; inc/dec)

for in

for of

Post test loop:

while - while(cond) -> we have inc & dec as last statements in block.

do-while -
```
do
{
}while(cond);
```

for in:

```
for (key in object)
{
}
```

Eg:

```
<!DOCTYPE html>
<html>
<body>
<p id = "para"></p>
<script>
const x = [10,20,30];
let text = " ";
for(i in x)
{ text += x[i] + "<br>";
}
document.getElementById("para").innerhtml = text;
</script>
</body>
</html>
```

o/p: 10
20
30

for of:

```
for(variable of iterable)
{ =
}
```

Eg:

```
<!DOCTYPE html>
<html>
<body>
<p id = "para"></p>
<script>
const x = [10,20,30];
let text = " ";
for(i of x)
{ text += i + "<br>";
}
document.getElementById("para").innerHTML=text;
</script>
</body>
</html>
```

o/p: 10
20
30

③ Other

Break: used to come out of a loop.

Continue: used to skip the next iteration.

* Functions: It is a block of code used to perform a particular task

Syntax: function name (parameters) → valid identifier name
{
statements → function definition
}

→ In JS function has no function declaration

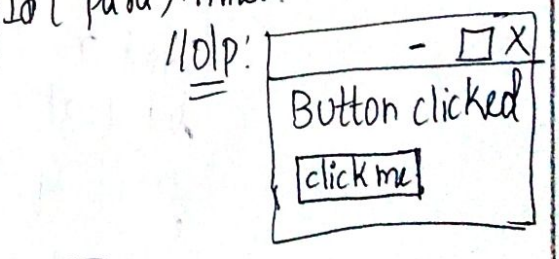
→ In Javascript a function can be invoked in 3 ways

1) If an event occurs
2) Invoked in JS 3) Invoked Automatically (self invoked)

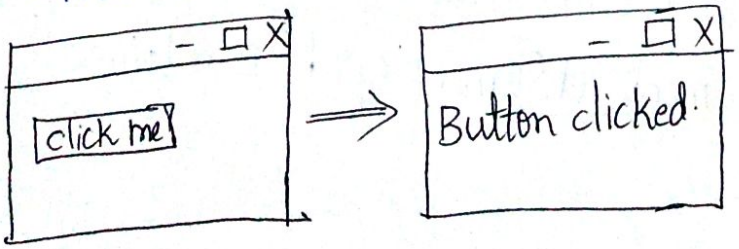
1) If an event occurs (on click event)

```
<!DOCTYPE html>
<html>
<body>
//<p id="para"></p>
<button type="button" onclick="myfunction()">
Click Me
</button>
<script>
function myfunction()
{
document.write("Button clicked");
//document.getElementById("para").innerHTML="Button clicked";
}
</script>
</body>
</html>
```

//O/P:



O/P:



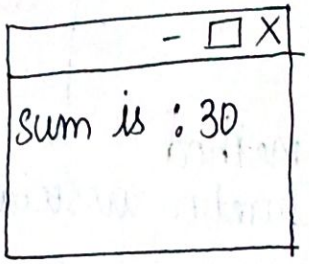
→ no double quotes for variable name

2) Invoked in JS

① <!DOCTYPE html> (body)

```
<html>
 <body>
  <p id = "para"></p>
  <script>
    let a = 10, b = 20, c;
     c = add function (a,b);
     document. get Element By Id ("para"). inner HTML = "sum is:" + c;
     function add function (x, y)
      {
       return (x+y);
      }
  </script>
 </body>
</html>
```

O/p:

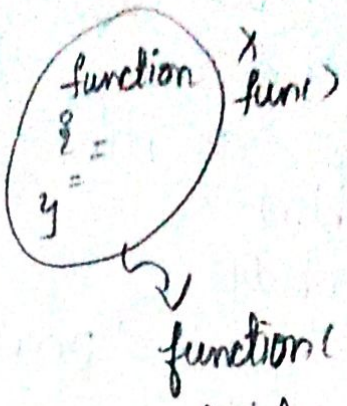


② (head)

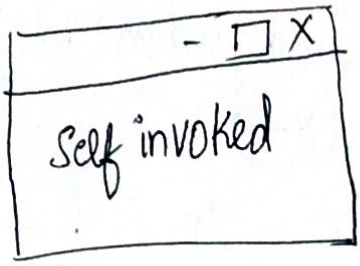
```
<!DOCTYPE html>
 <html>
  <head>
   <script>
     function addfunction (x, y)
     { return (x+y);
     }
   </script>
  </head>
   <body> <p id = "para"></p>
          <script>
             let a = 10, b = 20, c;
               c = addfunction (a, b);
             document. get Element By Id ("para"). inner HTML = c;
          </script>
   </body>
 </html>
```

3) Invoked Automatically (self invoked)

```
<!DOCTYPE html>
<html>
<body>
<script>
(function ()
{ document.write("self invoked"); // directly functioncall by putting whole function as a unit in function name
})();
</script>
</body>
</html>
```



o/p:



Function Vs Method

Standalone function associated with object then it is method.

JavaScript objects (similar to dictionaries)

→ In JavaScript, collection of Key: Value pairs that are enclosed in curly braces { } is called as object

→ Each key-value pair is property of an object

→ object can also be defined as collection of properties and methods

→ property can be accessed using anyone of the following notations

ObjectName.properotyName (mostly used)

(or)

ObjectName["PropertyName"]

→ Method can be accessed using the following notations.

ObjectName.methodName()

Ex: const student = {fname : "abc",
lname : "xyz",
meth : function() {
}
};

Student.fname
Student["fname"]
Student.meth()

→ If a function is associated with an object, then it is called as method

function → standalone

method → Function with object

Arrays

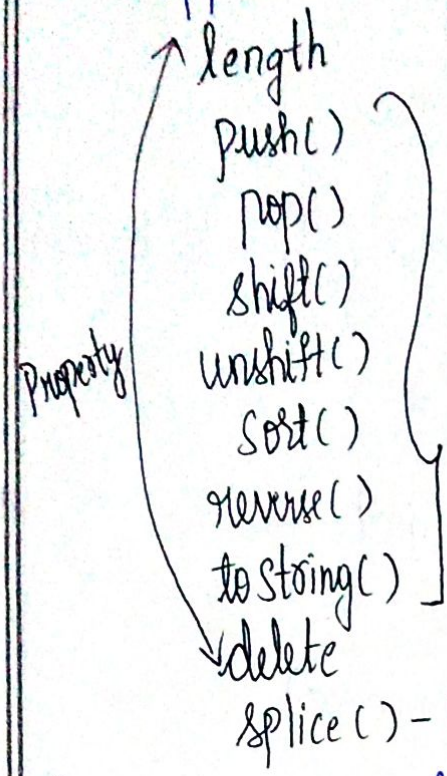
→ Similar to list in python (can be heterogeneous)
→ It is treated as object
→ Collection of elements is called as an array
→ In JavaScript, array can be defined in 2 ways.
1. array literal
2. array object

Ex: const arr = [10,20,30]; //array literal
const arr = new Array(10,20,30); // array object

→ In JavaScript array is treated as an object
typeof arr //object

→ To know whether the object is array or not, we can use any one of the following
1) arr instanceof Array // true - then array object
2) Array.isArray("arr") // true - array
↳ class ↳ method.

→ JavaScript provides several properties & methods, those can be applied on arrays.



1) length: It gives length of array / no. of elements in an array

Ex: const arr = [10,20,30];
arr.length; //3 o/p

2) push(): It is used to add an element at end of an array

Ex: arr.push(40); //[10,20,30,40]

3) pop(): It removes last element from an array

Ex: arr.pop(); //[10,20,30]
returns 40

4) shift(): It pop's out 1st element in an array

Ex: arr.shift(); //[20,30]
returns 10

5) unshift(): It will push element into 1st position of array.

Ex: arr.shift(10); //[10,20,30]

6) sort(): Arrange elements in array in ascending order

Ex: arr = [20,10,30];
arr.sort(); //[10,20,30]

7) reverse(): It arranges all elements of array in reverse order.

Ex: arr = [20,10,30];
arr.reverse(); //[30,10,20]

(or)

arr = [10,20,30];
arr.reverse(); //[30,20,10]

8) delete: It is used to delete element at required index but it leaves a blackhole i.e value to removed but it's place remains as undefined.

Ex: delete arr[10]; //[20,30]
10 X → Blackhole.
arr[0]; //undefined (∵ 10 deleted)

→ arr = [10, 20, 30]
0 1 2

if we assign beyond index i.e
arr[5] = 40 //[10, 20, 30, 40]
0 1 2 5

then in index 3 & 4 undefined is stored i.e
arr[3] = undefined
arr[4] = undefined

↓
to avoid blackhole we can use

Splice(): it is used to add/or remove elements to/from an array

Syntax: splice(index, howmany, element1, element2, ----)

Ex: ① only add elements

```
const arr = [10,20,30];
arr.splice(2, 0, 40, 50)
```

(2 → index before we add; 0 → only add then zero)

o/p: [10, 20, 40, 50, 30]

② only removing

```
arr = [10,20,30]
arr.splice(1,1) // [10,30]
arr.splice(1) // [10]
```

③ Both remove & add.

```
arr = [10,20,30]
arr.splice(1,1,40,50) // [10,40,50,30]
```

↳ first removes & adds.

→ toString(): It is used to convert all the elements of an array into a single string

Ex:

```
const arr = [10,20,30];
arr.toString() // "10,20,30"
```

⇓ (or)

```
arr // "10,20,30"
```

→ `arr.toString() == arr`

EVENTS

→ A state change in a html element is called as event.

→ HTML provides event handler attributes

→ Some of the event handler attributes are

onchange	onmouseout	onkeyup
onclick	onload	⋮
onmouseover	onkeydown	

onchange: It is basically used for drop-down list

Ex:
```
<!DOCTYPE html>
<html>
<head>
<script>
function myfunction()
{ document.getElementById("para").innerHTML =
"you selected:" + document.getElementById
("list").value;
}
</script>
</head>
<body>
<select id = "list" onchange = "myfunction()">
<option> Telangana </option>
<option> Ap </option>
<option> Maharashtra </option>
</select>
<p id = "para"></p>
</body>
</html>
```

onclick: it occurs whenever a button is clicked.

onload: This event occurs while loading the page.
This attribute is used for body tag.

Ex:
```
<!DOCTYPE html>
<html>
<head>
<script>
function myfunction()
{
alert("page is loaded");
}
</script>
</head>
<body onload="myfunction()">
<h1> Heading 1 </h1>
</body>
</html>
```

onkeydown: This event occurs when a key is pressed on keyboard.

Ex:
```
<!DOCTYPE html>
<html>
<head>
<script>
function myfunction()
{
alert("key is pressed");
}
</script>
</head>
<body>
<form>
<input type="text" keydown="myfunction()">
</form>
</body>
</html>
```

onmouseover: it occurs when we move cursor to elements.

→ Onmouseover event occurs when mouse cursor is on the element of the document.

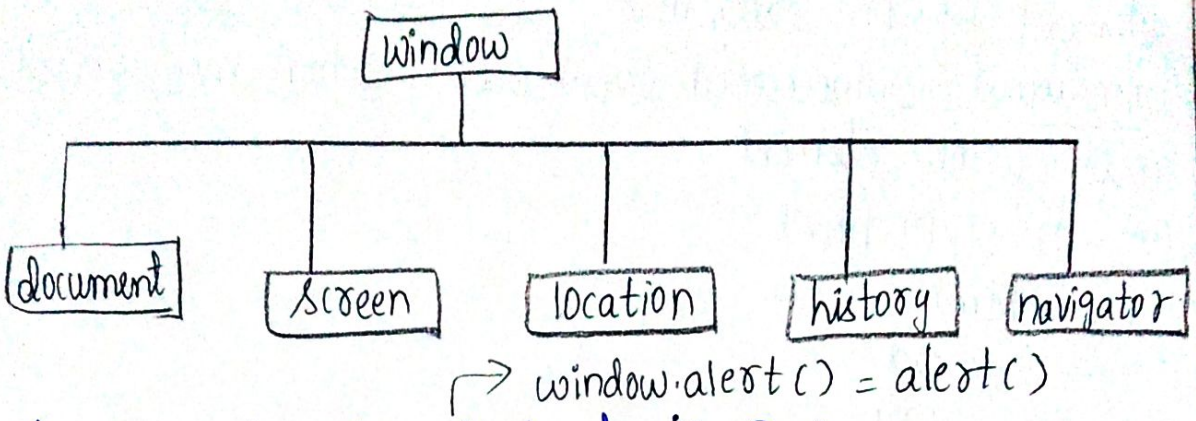
onmouseout: onmouseout event occurs when mousecursor is out of the element.

Ex:
```
<!DOCTYPE html>
<html>
<head>
<script>
function big(x)
{
x.style.height=64px;
x.style.width=64px;
}
function normal(x)
{
x.style.height=32px;
x.style.width=32px;
}
</script>
</head>
<body>
<img src="image.jpeg" alt="sample image" width="32px"
height="32px" onmouseover="big(this)"
onmouseout="normal(this)">

</body>
</html>
```

Browser Object Model (BOM)

→ It allows Javascript to interact with the browser.



↳ window.alert() = alert()

→ window is the default object is BOM

→ There is no specific standard for BOM

→ DOM (Document object model) is subset of BOM
↳ It has some standards

→ window object

Properties & methods:

window.innerHeight
window.innerWidth
window.open() - open url
window.close() - close url.
window.moveTo()
window.resizeTo()
window.alert()
window.confirm()
window.prompt()

→ window.innerHeight & window.innerWidth; are used to get innerHeight (document height) & inner width (document width) respectively.

EX:

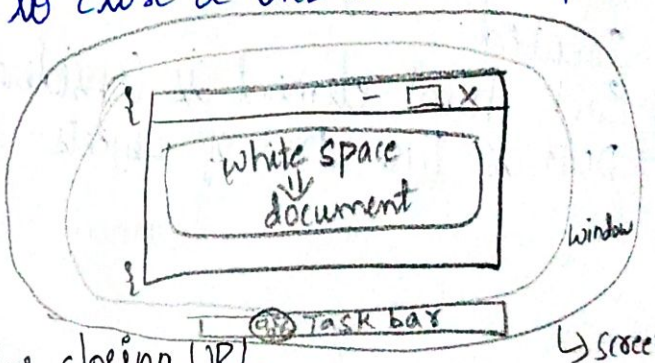
```
<!DOCTYPE html>
<html>
<body>
<p id="para"></p>
<script>
document.getElementById("para").innerHTML = window.innerHeight;
</script>
</body>
</html>
```

window.innerHeight;
↓
for innerWidth value

→ window.open(): it is used to open specific URL
window.open(URL)

Eg: remove para, document.get & write this statement in script.
window.open("www.google.co.in")

→ window.close(): it is used to close a URL which is opened.



Eg: open & close.

```
<!DOCTYPE html>
<html>
<body>
<script>
let u = window.open(URL);
</script>
</body>
</html>
u.close() → for closing
```

u → reference for closing URL

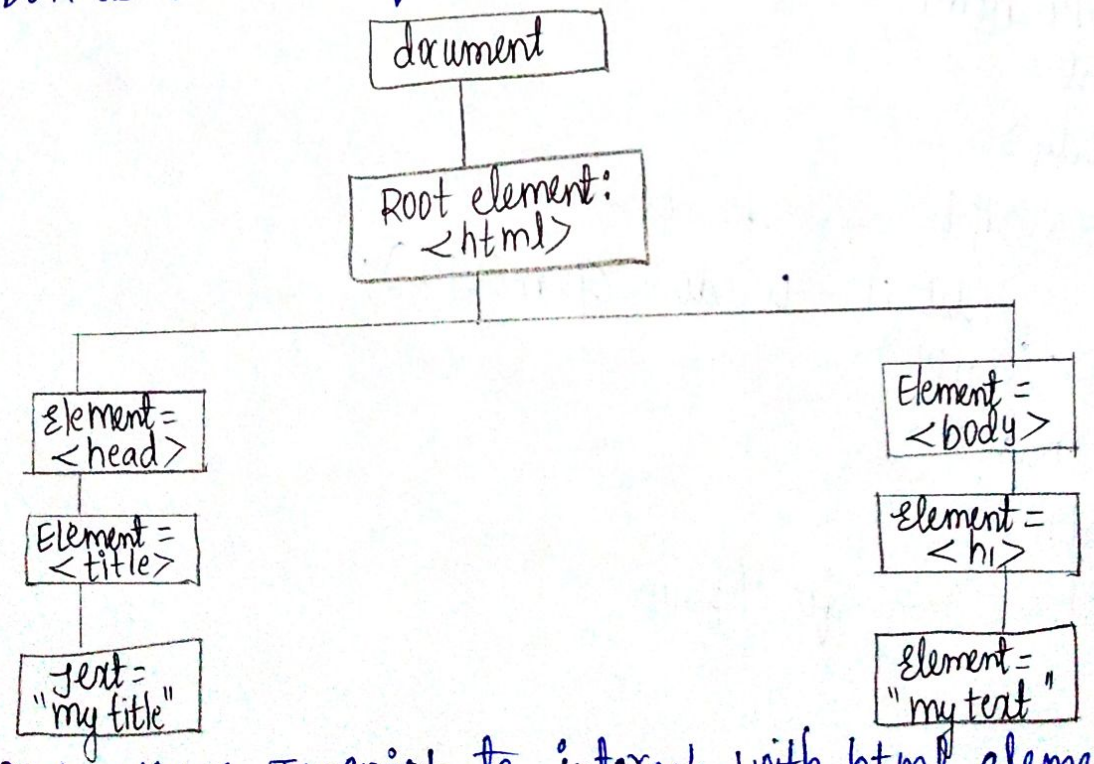
→ Window.moveTo(): is used to move cursor to a particular point.
Window.moveTo(x,y) / basically used in canvas & can directly use moveTo()

Eg: window.moveTo(100,100);

→ window.resizeTo(): is used to change width and height of the element
∴ window.resizeTo(width, height);

Ex: window.resizeTo(100,100);

1. Document object
→ allows us to interact with each & every element
→ After loading the page, Document object model (DOM) is created.
→ Each html element is considered as an object
→ DOM is like tree of objects



→ DOM allows Javascript to interact with html elements

Properties & methods:

document.write()
document.writeln()
document.getElementById()
document.getElementByName()

2. Screen object: It deals with user's screen.

Properties & methods:

in Pixels { screen.height, screen.width } total (complete)
in Pixels { screen.availHeight, screen.availWidth } total - Taskbar i.e excluding Taskbar

in Bits { screen.colorDepth, screen.pixelDepth } - 24|32|64 bits.

Ex:

```
<!DOCTYPE html>
<html>
<body>
<p id = "para"></p>
<script>
document.getElementById("para").innerHTML
= screen.height;
</script>
</body>
</html>
```

screen.height → screen.width / screen.availHeight etc.

3. location object: It deals with url

location.href → complete Url.
location.hostName → domain

location.path → '/'
location.port → port number
location.protocol → http: (or) https:

URL (protocol, domain, port, segment)

port for http-80 & https-443 } we will get no output for these 2 protocols since they are default.

Ex: same example as screen object example.

4. history object: It deals with the history of the browser.

Properties & methods:

→ history.length - no. of URL's present in the history of the browser.
→ history.back() - loads/opens previous URL
→ history.forward() - loads next URL
→ history.go() - goes to specific URL takes 1 parameters i.e value
i.e history.go(-2) → 2nd back URL.

5. Navigator object: It deals with information about the browser
↓ browser

Properties & methods:

→ navigator.cookieEnabled → True|False
→ navigator.languages → en-us
→ navigator.appVersion → version of browser
→ navigator.platform → OS(win32|win64)
→ navigator.javaEnabled() → o/p - True|False

Cookies

→ Server related info is stored in the browser in the form of cookies (for quick access)

→ first browser is netscape navigator, so we call it is a navigator.

HTML 5 canvas

→ canvas is a container for creating graphics on the document

→ HTML 5 provides <canvas> tags to do that

Attributes: id
width
height

→ canvas is a rectangular area by default with no borders.

→ Border can be added to the canvas using border property of CSS

→ we can even draw 2D shapes on canvas.

Ex:

```
<canvas id="my canvas" width="300" height="200" style="border
                                                  :1px solid blue">

</canvas>
```

→ Javascript provides several methods to deal with canvas.

→ getContext() method creates context object i.e facilitates to draw shapes.

→ It takes one parameter value.

getContext(context)

- "2d" } 2D
- "webgl" } 3D (web graphics library)
- "webgl2" }
- "bitmaprenderer" } Replaces canvas content with bitmap image.

→ beginPath() is used to start or reset the path.

→ moveTo() moves cursor to particular point

Syntax: moveTo(x,y) (It is relative to the canvas)

→ Stroke() is used to draw strokes (lines)

→ strokeStyle: property is used to add colour for the stroke.

→ fillStyle: is used to fill the colour within the shape.

→ lineTo(): is used to draw line on the canvas.

Syntax: lineTo(x,y)

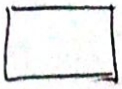
Ex:

```
<!DOCTYPE html>
<html>
<body>
<canvas id="mycanvas" width="300" height="200"
                      style="border: 1px solid">

</canvas>
<script>
        let c = document.getElementById("mycanvas");
        let context = c.getContext("2d");
        context.beginPath();
        context.moveTo(30,30);
        context.lineTo(150,150);
        context.stroke();
</script>
</body>
</html>
```

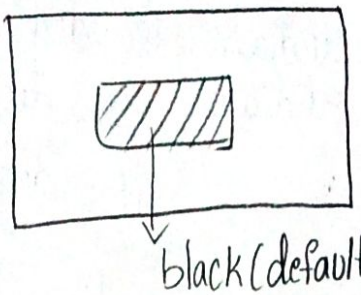
→ rect() - takes 4 parameters
rect(x, y, width, height)

Ex: context.rect(10,10,150,100);

O/p: 

→ fillRect() - fillRect(x, y, width, height)

Ex: context.fillRect(10,10,150,100);

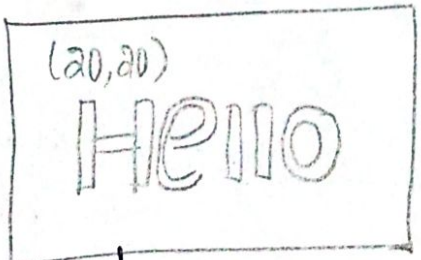


→ context.fillStyle = "red"; → fills with red color

→ strokeText(): strokeText(text, x, y, maxwidth)
↳ (optional)

Ex: context.strokeText("Hello", 20, 20);
context.stroke();

O/p:



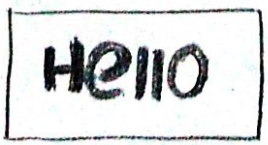
↳ No stroke() required

→ fillText(): fillText(text, x, y, maxwidth)
optional

Ex:

```
context.font = "25px Axial";
context.fillText = ("Hello", 20, 20);
context.stroke();
```

O/p:



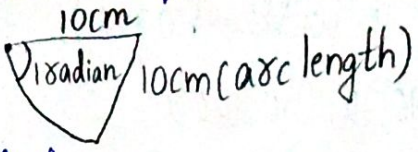
→ arc(): used for drawing circles & arcs.

arc(x, y, radius, sAngle, endAngle, counterclockwise)

sAngle, endAngle → radians

counterclockwise → default value is false.

1 radian = radius equal to arc length



1 radian = 57.3°

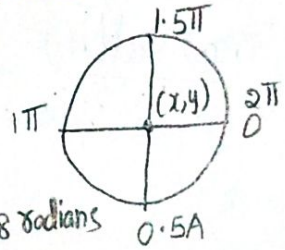
↳ π radians = 180 degrees

1 radian = $\frac{1 \cdot 80}{\pi}$ degrees

= 57.3 degrees.

radian	2π
degrees	360°

1 degree = $\frac{\pi}{180}$ radians = 0.0175 radians.



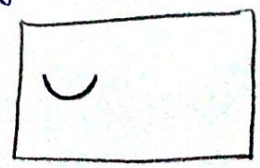
sAngle & eAngle in radians.

Ex:

```
<!DOCTYPE html>
<html>
  <body>
    <canvas id = "my canvas" width = "300" height = "200" style
                                        = "border: 1px solid">

    </canvas>
    <script>
        let c = document.getElementById("my canvas");
        let ctx = c.getContext("2d");
         ctx.beginPath();
         ctx.arc(50, 50, 30, 0, math.PI)
         ctx.strokeStyle = "red";
    </script>
  </body>
</html>
```

O/P:



Animations

```
<!DOCTYPE html>
<html>
<body>
<canvas id = "my canvas" style = "border:1px Solid">
</canvas>
<script>
let c = document.getElementById("my canvas");
let ctx = c.getContext("2d");
c.width = window.innerWidth;
c.height = window.innerHeight;
let x = (Math.random() * window.innerWidth);
let y = (Math.random() * window.innerHeight);
let radius = 30;
let dx = (Math.random() - 0.5) * 2;
let dy = (Math.random() - 0.5) * 2;
function animation()
{ window.requestAnimationFrame(animation);
  ctx.clearRect(0, 0, window.innerWidth, window.
                                    innerHeight);
  if(x+radius > window.innerWidth || x-radius < 0)
  { dx = -dx;
  }
  if(y+radius > window.innerHeight || y-radius < 0)
  { dy = -dy;
  }
  x += dx;
  y += dy;
  ctx.beginPath()
  ctx.arc(x, y, radius, 0, 2 * Math.PI);
  ctx.stroke();
}
animation();
</script>
</body>
</html>
```

FORM VALIDATION:

→ onsubmit (default is true) event handler attribute (submit event for form)
↓
(attached to <form> element)

→ Value property is used in JS to get value of any input field of the form.

→ required attribute is used for input field to make it as mandatory field.

→ forms property of document object is used to access form fields.

EX: checking mandatory form fields.

```
<!DOCTYPE html>
<html>
  <head>
    <script>
      function validateForm()
      {
        let t = document.forms["myform"]["tf"].value;
        if(t=="")
        { window.alert("please enter some text");
          return false;
        }
      }
    </script>
  </head>
  <body>
    <form name="myform" onsubmit="return validateForm()">
    <input type="text" name="tf" required>
    <input type="submit">
    </form>
  </body>
</html>
```

Note:
[t==""](or)[t.length==0]
both are one & the same.

Validating password field:

→ minimum 1 uppercase letter
→ minimum 1 lowercase letter
→ minimum 1 digit
→ minimum 1 special character
→ minimum length 8 characters
→ we can achieve them using Regular expressions.

* - 0 or more occurences
+ - 1 or more occurences
. - any single character other than \n
? - 0 or 1
^ - starting of the string | Inverse
$ - Ending of the string
[ ] - Range
( ) - group
\d - digits
\S - whitespace
{x,y} - define no. of occurences

How to construct & check expression?
→ It is represented in between slashes i.e "/ — /"

1) uppercase
(?=.*[A-Z])
?= positive lookahead

2) lowercase
(?=.*[a-z])

3) digit
(?=.*[0-9]) (or) (?=.*\d)

?= positive lookahead
?! - Negative lookahead
?<= positive look back
?<! negative look back.

4) Special characters (8)

! . @ # % . $ & * ^ → spl characters

→ no range specifying all.

(?=.*[!@#%.$&*^])

5) length

→ 8 to max

(?=.{8,}

↓ can be any character

Array literals

random() - generates value b/w 0-1 i.e including '0' excluding '1'

→ final pattern is

/^(?=.*[A-Z])(?=.*[a-z])(?=.*[0-9])(?=.*[!@#$%.*^&])

spaces are accepted → (?=.{8,})$/

↓ it can also written as ".{8,}$/"

→ !(/s) ⇒ we can restrict spaces i.e (?!.*\s)

↓ negative lookahead ↓ space

→ '$' works with multiline (m) flag

CAPTCHA:

→ completely Automated public Turing test to tell computers and Humans Apart (captcha).

→ CAPTCHA it is used to provide security.

→ It checks whether the person using is a human or not

→ reduces spam mails

reCAPTCHA is product of google.

RECAPTCHA

How to generate CAPTCHA

→ ideal length of CAPTCHA is 6
it has many ideal ways here we use 2 methods i.e

1st way

```
let arr = newArray('A','B','c',---,'Z','a','b',---'z','0','1',---'9');
```

(or)

```
let arr['A','B',-----,'9'] ; //Array literal
arr[Math.floor(Math.random() * arr.length)]
```

- Math.floor → give to round off
- Math.random() → randomly picks from 0-1 including 0 & excluding 1
- arr.length → multiplies to 62 & then apply

```
for (let i=0; i<6; i++)
```

'g' flag - it searches global occurences

2nd way

```
let s = "ABCD---9";
s.charAt(index)
```

Takes char at a index.

regexr.com

eg: (?=.*[a-z])(?=

```
String.Split ("delimiter, limit)   // returns array of substrings
```

- delimiter → space, comma
- limit → how many splits

```
array.join(seperator)   // returns string
```

seperator → -(hyphen)

RECAPTCHA: It is a product of google

→ It has 3 vertices V_1, V_2, V_3.