LIST OF EXPERIMENTS PERFORMED

NO.	DATE	EXPERIMENT	NO.	DATE	EXPERIMENT
1	02/5	V.S.install and shortcut setting	31		
2	2/5	Basics, Startup, Text, web img & links	32		
3		Lists and Tables, Inline & block element	33		
4		Input tags, attributes, ids and classes	34		
5		HTML Entities | CSS startup	35		
6		Fonts, Colors in CSS, Border	36		
CSS Reset		Background, Box Model, Margin, Padding	37		
8		Float & clear, Button design, selector. pseudo	38		
9		Navigation Bar	39		
10		CSS display, Position, visibility, z-index.	40		
11		Flex box	41		
12		Responsive Design, Selectors.	42		
13		Shadows, Css Variables.	43		
14		ANIMATION/keyframe	44		
15		Transition, Css Transform, Grid	44		
16		Grid span, MinMax, Autofil.	46		
17			47		
18			48		
19		Javascript History	49		
20		Variables, Datatype, operators.	50		
21		operators, STRING IN HTML	51		
22		Strings in js, Substring from string	52		
23			53		
24			54		
25			55		
26			56		
27			57		
28			58		
29			59		
30			60		

Earning, Freelancing

Coding

COMPUTER COURSES

* GBOB (Chrome)
* Affiliate Marketing (Amazon)(Daraz) etc.
* Web development:
  - → Coding
  - → Wordpress
* Android app development (Android Studio)
* Mobile app development
* Graphic Designing (Ai, Ps, Id, Corel draw)
* Video editing (Pr, Ae)
* Audio editing (Au)
* Composing (Data entry)

"Visual Code Installing & Live Server+Shortcut Setting"

* Visual Studio (VS)
→ It is a free coding software.
→ It is recommended for Web development and app development by different languages such as python.
→ Its benefit is that it gives suggestion, make the tag closed automatically.
"Which makes our work easier and faster."

* First of all "shortcuts" will be set according to our regular shortcuts and usage to make in your work. and server installation:-
→ Ctrl + D (to copy the line in next row)
→ Ctrl + / (besides shift key) (to add comment which will not be executed by the server.
→ Ctrl + S (to save the file) or (setting can be settle as autosave)
→ Alt + z toggle word wap.
→ ~~Ctrl~~ Shift + Alt + F format page.

* "Live server" is installed to view your result in browser quickly on saving the coding file.

→ To come out a folder (..) double dot is used.

→ Front End: (ONLY Front End : Static Web)
i) HTML, CSS, Javascript

ii) Back End: (F.E + B.E : Dynamic Web)
Php → My SQL, Nodejs → Mongodb
→ flask/django :→ SQLite

* HTML, CSS, JS, php, nodejs,
* Python => django/flask,

* Code pen

Backup:
Email: webcode backup@gmail.com
Password: Develop and backup.

HTML Text Decoration Tag

* Italic ⇒ em
* Bold ⇒ strong
* Underline ⇒ u
* to do cutting ⇒ s

" To create dead links '#' is use in "href" attribute."

" To form image a link use it in <a> tag."

HTML

Basics and Startups

* ! + Enter (Emmet Abbrivation)
* <Head> (tag includes) 3 things:-
  → Meta information
  → <Title> Abu Hurairah </Title>
  → Url links of ".css" and "javascript" ".js" etc
  (to combine file and complete website)
  (Head tag involves meta tags, titles, url links of "js." and "css"
  </Head>
* <Body>
  ~~</Head>~~

→ Meta tags:
Meta name=Description content ——
Meta name=keywords " —— (SEO)
Meta name=Robots " (Index, Follow)

→ To include ⇒ external css → link: css
⇒ extenal javascript → script src

Text → heading tags h1 → h2, h3, h4, h5, h6 → Smaller Sizes.

→ * ⇒ bold ⇒ Italic

Line break →
 line break

→ <hr> to produce a line. "——"

Web Links → Ctrl + enter to jump in next line

→ <a> - to make links in your web.
* href="link", target="_blank" → to open the link in new tab.
* can also be used to jump into diff. sections of pages. with # id.

Img Add → <Img: src="" alt="text" width="" height "" >

List

Lists and Tables ③

 type → circle, disc, square.
→ Unordered lists.
 ----

type → i, I, a, A

 → ordered lists

Tables

<thead> → table head.
<tr> → table row.
<th> --- </th> → table heading
</tr>
</thead>
<tbody> → table body.
<tr>
<td> --- </td> → table data
</tr>
</tbody>

Inline & Block Element.

-- → block
-- → inline

→ Block ⇒ ul, li, div
→ inline ⇒ em, img.

* Input tag has NO closing tag.

* required is written in the "input" tag than it means it is important.

* For RADIO BTN

<Input Type="radio" name = "S">

<label for = "I">I</label

<form action=" " → Backend >

Input taken by user (4)

<Input type = " "> → text → Date → radio → textarea → clms → row
→ nmbr → checkbox → email.

— <Input type = "radio" name = "same"> —
<input type = "radio" name = "same">

  to make other input on next line.

<label for = "s" >
<input type = " " id = "s" > {To make selection by clicking on the text.

<Label for = "s" >
<Select name = "my car" id = "s" >
<option value = " " > --- </option>
<option value = " " > --- </option>.
↳ selected to select the value of one

<Button> contact us </Button>

"Ids AND Classes"

"." is use for class.
"#" is use for Id.

HTML Entities:

→   to make more spaces than one.
→ <p> to print <p> in your web.
→

Semantic Tags

Selector { Property : Value }

P { colour : BLUE; }

header tags. Paragraph with class intro.

header, p.intro { Background-color: red;
Border-radius: 3px;
}

Group of selectors

Declaration block

⇒ Selector types
* CSS element selector → Select tags
* CSS id selector → select an id
* CSS class selector → select a class
* CSS grouping selector.

FONT Design

→ Font family ; font design.
→ font size ; 0px ∵ 1px = (1/96 th) inch
→ Line-height ; 0 em
(Line-spacing)
→ Font-weight ; "bold"
→ font-style ; "italic"

COLOURS IN CSS

(WWW.HTMLcolorcodes.com)

Color : Name
Color : RGB value. (0, 0, 0).
Color : #ffffff value.

BORDER

Border ; size solid color.
→ Border can be indivually set, by
✓ Border-top etc.
→ To make border corners ~~radius~~ round.
✓ Border - radius = 0px,
→ for in divisual.
✓ Border-top-left-radius etc.
→ Border ~~size~~ color : "Name" Ya "RGB"
→ Border Style : solid;
→ Border width : 0px.
→ Border : none;

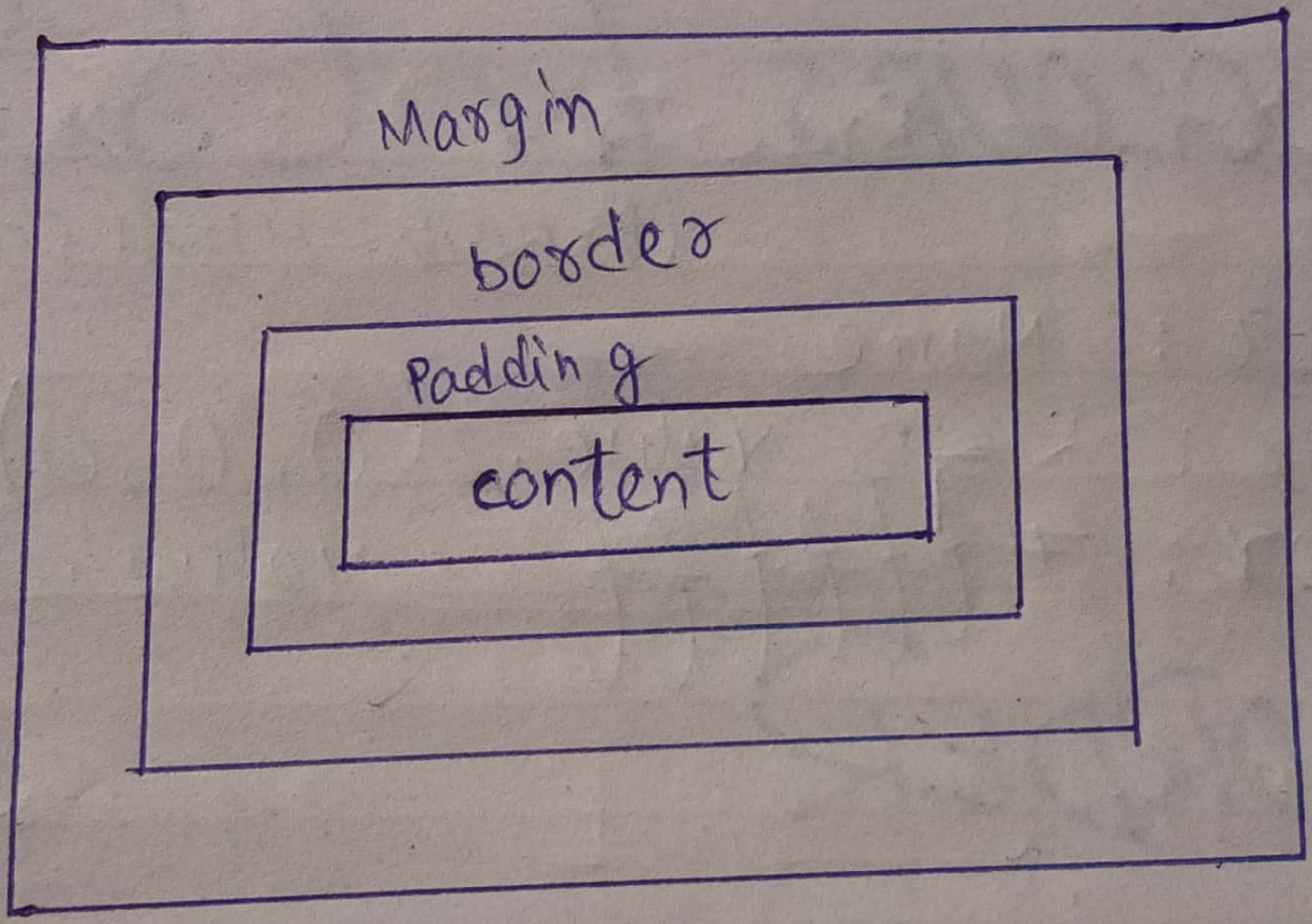


Fig: Box MODEL

BACKGROUND:

- Background-color:;
- Background-height;
- Background-width;
- Background-image; url(' ');
- Background-repeat; no repeat, repeat x repeat y.
- Background-position: centre-centre.
  (x) (axis) (y) (axis)

BOX MODEL:

Margin, Padding:-

- Margin: 0px.
- Padding: 0px.

(They can be set left, right, top and bottom seperatly)

- Padding y(top/bottom) x(left/right)

- Box-sizing; border-box.
- * { } to set by defaul values. for all elements.

CSS Reset

FLOAT Property:

→ Float : left; "left, right"
→ Clear : left/right; "to remove the overright of float
→ Width : % [%, px.]

Designing a button:

→ cursor : pointer;
→ <Button> contact us </Button>

→ text decoration : none; "to remove underline or other properties."

Pseudo Selector:

→ a:hover { } "This property is used to change something when you move cursor to content."
→ a:visited { } "To show visited site sepezatly"
→ a:active { } "When clicked on it changing occurs."

Also on pg #12

NAVIGATION BAR

→ list styling : none; "to remove bullets".

```
<header>
    <nav class=" ">
    <ul>
        <li>  <li>
    </ul>
    </nav>
</header>
```

* "Float property will be use to float the elements in the list"

→ Then Bg-color will be removed this is because when float property is used then the element overflow from parent element. To accomodate this problem we have to target <ul> in the markup

→
{overflow : auto;}

CSS DISPLAY PROPERTY:

→ Header is a block element.
→ And it does not works with "margin: auto;" (to centre the element).

* Display can be block, inline or inline-block.
→ Opacity: 0.

Position Property:

(it will leave a gap if relative is used).

→ Relative: relatived to the elements position.
→ Absolute: relative to the parent element
→ Fixed: relative to browser's window.
→ Sticky: To stick an element.

VISIBILITY:

→ visibilty: hidden "to reserve element space on hidding it"
→ Display: none "to not reserve space for element on hidding it."

Z-INDEX: "it will work only when two objects are overlapping and "position property" is used."

→ Small value will move the element to back.
→ Big value will move the element to front.

FLEX BOX:

Flex container

Flex item.

Display: Flex;
x-axis ⇒ main-axis
y-axis ⇒ cross-axis

Just make sure that the "display: flex" is formed/made of "CONTAINER".

CONTAINER PROPERTIES.

Flex-direction: row/colomn; (row-reverse/colomn-reverse)
Flex-wrap: ; (no wrap; wrap), wrap-reverse)
flex-flow: (flex-direction flexwrap) shortcut to add flex-diection, flex-wrap along same).

Justify content: ; (centre, space-between, space-evenly, space around
→ Horizontally centre.

Align-items: centre; (flex-start, flex-end, stretch)
↳ Vertically centre (vertically centre)
↳ vertical even space between elements.

Item-Properties:

Order: ; higher the order later the element will show.
Flex-grow: ; to increase the width of item
Flex-shrink: ; to decrease the width of item
Flex-Basis: ; (when flex-direction is set to row it will controll width)
or vice versa.
align-self (flex-start, flex-end, centre)

Responsive Design

UNITS:

rem: ; it resize with html font size.
em: ; it resize with parent element. but element's padding or margin will be added with respect to the font-size set.

Media Queries

selector

@media (min-width: ...px) and (max-width: ...px){ }

If min-width is not mentioned Then min-width = 0px.
If max-width is not mentioned then max-width = ∞ px.
→ Media-queries are set at the last or after the classes on which we have to set it.

Selectors:

div > p {if "p" is right inside "div"}
div + p { if "p" is right after "div" }
Input [Type = " "] { }
a [target] { } , a [target = " "] { }
This will apply css to the number
Li:nth-child ( )
→ any number.
→ odd, even
→ (xn+y) where 'x' and 'y' is sent by develpor.

Before and After:

header :: before {}
header :: after {}

Text and Box Shadows

→ Box-shadow: x px y px; | if x and y are +ve then shadow will be right and down.

→ Box-shadow: x px y px blur-radius colour of shadow;

→ Box-shadow: x px y px blur-radius spread-radius colour of shadow.

* colour can be written as RGBA {a, b, c, d}
* in-set property will be used to form the shadow inside box.

→ text-shadow: x px y px colour;

* Also, it is same with the Box shadows.

CSS VARIABLES:

To set a variable,

→ --variable-name : variable-value;

To use variable,

→ ~~Set~~ margin : var(variable-name).

To set use variable any where we have set it in:

:root { }

ANIMATIONS / KEYFRAME.

→ Animation-Name: ;
→ Animation-Duration: ;
→ Animation-Itration-count ; ;
@ keyframes Animation-name {
from { }
to { } }

→ Animation-fill-mode: ; → forward (to stop at the last), alternate (to stop at the start).
→ Animation-timing-function: ; ease-in, ease-out, ease-in-out
→ Animation-delay: ; 0s (to delay the animation.)
→ Animation-direction: ; → alternate, reverse.

@ key-frame Animation-name {
0% { }
25% { }
75% { }
100% { }

→ animation: animation-name animation-duration animation-timing-function animation-delay animation-iteration-count animation-fill-mode;

grid-gap-row: ;

grid-gap-column: ;

grid-gap: 0px; to make every side same.

Short hand.

grid-gap: grid-gap-row grid-gap-column;

grid-gap: 0px 0px;

TRANSITION:

→ transition-property: ;
box:hover { }

→ transition-duration:;
→ transition-Timing-function:;
→ transition-delay:;

Transition:
transition-property
transition-duration
transition-delay
transition-function.

→ transition: property duration timing-function delay;

CSS TRANSFORM.

To move.

→ Transform: ; Rotate, skew, scale( ), translateX, translateY, translate(x, Y)

CSS GRID:

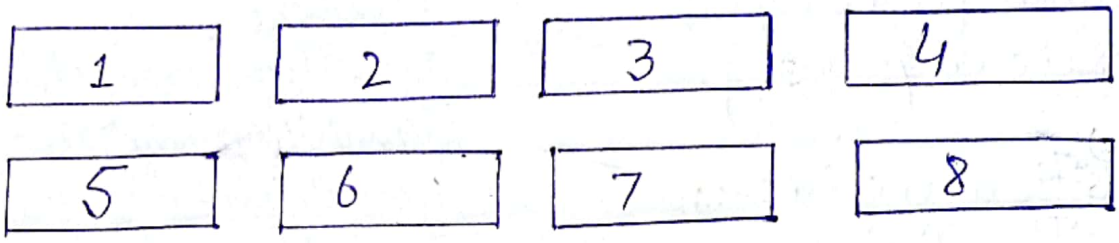
Display: grid;
grid-template-columns: 300px 100px 100x;
(1st column width, 2nd column width, 3rd column width)

It can be used with unit "fr" to form ratios.

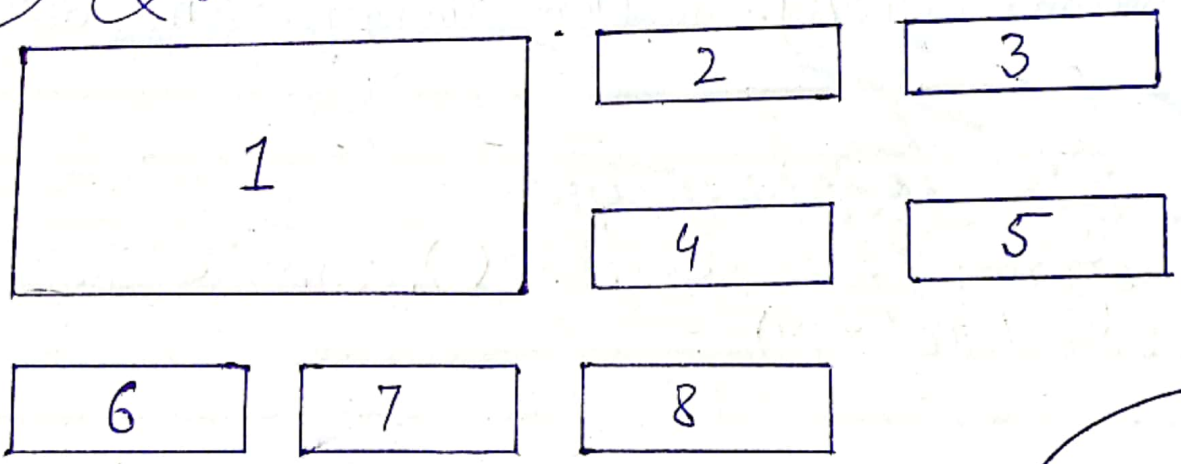
grid-template-columns: repeat(times, value)
grid-gap: 0rem; spacing between elements.

grid-template-rows: ; → selected rows
grid-auto-rows: 2fr; → all the other rows.

From:



To:



→ align-items: start / center / End ; (vertically for all)

→ align-self: start / center / End ; (vertically for one)

→ justify-items: start / center / end ; (horizontally all)

→ align-self: start / center / end ; (horizontally one)

---

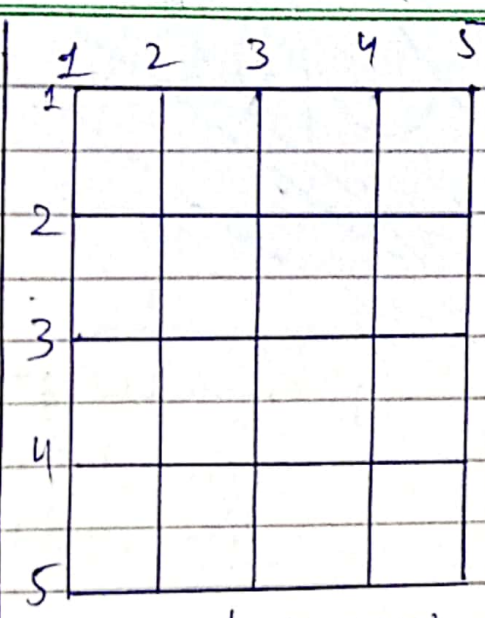
grid-template-area: val ; (one element)

GRID SPANNING.

→ Grid-row-gap: ;
→ Grid-column-gap: ;

→ class/Id: first child {
grid-column-start: 1;
grid-column-End: 3;
grid-row-start: 1;
grid-row-End: 3; }

Short hand
→ class/Id : first child {
grid-column: 1 /span 2
grid-row: 1 /span 2



same for row.
Now if we want an element to be cover two elements thus we have to do is
grid-column: starting line / ending line

IMP

GRID : Auto Fit & MinMax

grid-template-areas : ; "form a matrix for its values"

Media Queries: in CSS grid.

Grid

Grid-area: Grid horizontal Line start / Grid vertica
Line start / Grid horizontal Line end / Grid vertic
Line end;

What can't in-browser javascript do?

→ Can not read/write to & from computer hard disk.
→ Same origin, ajax request | cannot access restricted data.
→ can access permitted resources only.

What makes JS Unique?

→ Html/Css full support.
→ Simple thing → Simple API (application programming Interface).
→ Major modern browser. Enabled by default.

Javascript Version:

→ Maintained by community. (1995)
→ ECMA (1997) (Standard for making .js modern.)

→ 1st version 1997
→ 2nd version 1998
→ 3rd version 1999
⋮ ⋮

→ ECMA script 2015
" 2016
" 2017
" 2018
" 2019
⋮

→ Javascript can be written in head tag. also ~~after~~ Just before "body" closing tag. But, it is recommended to write just before body closing tag.
→ It can be written in Markup by using <script> tag

VARIABLES, DATA TYPE & OPERATORS

→ var a= val
→ console.log(a); (to print "val").

OPERATORS:

→ Unary operators (It has a single operand (x=-x))
→ Binary operators (It has two operands (x=x+6))

{In 3+4, "+" is operator, 3 and 4 are two operands}

→ console.log(x);

→ var num1 = 56
→ var num2 = 67

Arithmatic operations in js.

String

→ console.log("The value of ~~num~~ num1 + num2 is" + (num1 + num2))

→ console.log("The value of num1-num2 is" + (num1 - num2));
console.log("The value of num1 ** num2 is" + (num1 ** num2));
(> exponentiation operator

→ Increment operator: (Increment by "1").
console.log("The value of ++num1 is " + (++num1));

→ (If "++" is written after "variable" first value will be printed and then the increment will take place. But variable value remain same).
→ (But in case of "++" is written earliar. then first increment take place and then value will be printed and the printed value will have a "1" added)

STRINGS IN HTML:

→ var string = "this" {if we want to add double
→ var string = 'this' {goutes in side string it
→ console.log(string) {should be surrounded by single goute or vice versa

★ "+" operator will merge or attach strigs. also concat function be used. (Pg # 23).

→ Vara temp = `${string} is a book`

To print, This is a book

{ Back tips are below Esc.

var len = string.length;

→ "To find length of variable"

→ \n → New line, \\ to print \.

* var str = "This is a string";
  console.log(str);

"For first occurance of substring"

→ Position = str.indexOf('is');
  console.log(position).

→ Position = str.lastindexOf('is');
  console.log(position).

Substring from a string:

→ var substr = str.slice(1,7); (can take -ve values)
  console.log(substr)

→ var substr = str.substring(1,7); (can not take -ve values).

→ var substr = str.substr(where to start, How much characters are selected)

var

→ str = "This is a string"

var replaced = str.replace('String', 'Harry')

→ console.log(replaced) * to print "This is a Harry".

→ console.log(str.toUpperCase()); * to print all in uppercase.

and lowercase with lowercase

→ var newstring = str.('New String')
console.log(newstring). * to print "This a is a stringNew String."

→ (.trim()) function is used to remove extra spaces from start and end.

→ var char2 = str.charAt(2) → index NO.
console.log(char2) * To print "3rd" character.

→ var char2 = str.charCodeAt(2)
console.log(char2) * to find 3rd character's code.

→ console.log(str[3]) → INDEX NO. * to get same as charAt.

Scoping:

* to describe a variable:

let, a = 4 → this can be used for all places

```
{
    let a = 6
}
```

This is a block and its a variable cannot be used outside the block.

→ const a = "constant";

CONDITIONS:

~~let, (age > 18) {~~
~~if~~

```
let a = 34;
if (a > 18) { console.log("You can play") }
else { console.log("You can't play") }.
```

```
let a = 2
if (a > 18) { console.log("You can play") }
else if (a == 2) { console.log("a is 2") }
```

console.log.
else {"You can't play"}

Alternative

```
Switch (variable) {
    case value:
        console.log( )
    break;
  default:
    console.log( )
    break;
```

ARRAYS & Objects:

→ Variable Types:

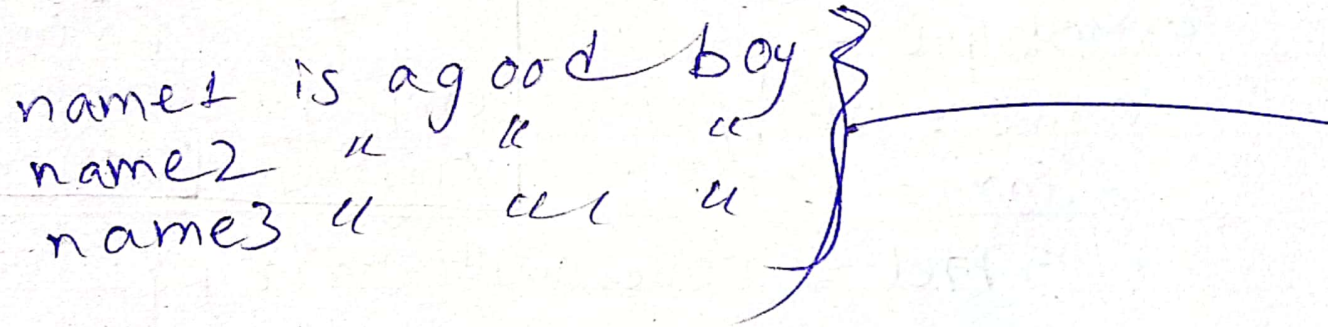
```
let a= 34
let b= "string"
let c= "true/false
let d = "null".
let e = "undefined".
```

→ Objects:

```
let employee = {
    name = "Harry".
    salary = 10
    channel = "code with harry }

console.log (employee).
```

for Printing
employee.name
employee['channel']



→ Arrays:

→ let name = [0,1,2,3, Harry]
→ let name = new Array [0,1,2,3, Harry].
→ console.log (name)
→ console.log (name[ ]) → Index no.
* for printing only one element.
→ console.log(name.length) → To find array's length.
→ name = name.sort. → (to sort the array).
→ name.push(" ") → This is use to push element in the last of array

→ let, names = new array(23)
→ "for getting "23" elements with empty space"

FUNTIONS IN.js

→ For repitation.
→ Function greet (name) {
→ console.log ('name' + " is a good boy.")}
→ greet (name1);
→ greet (name2);
→ greet (name3);

Return: Function:

```
function sum(a,b,c){
    let d= a+b+c;
    return d;
}
```

* After return no value will be executed with in function.

```
let retvalue = sum(1,2,3)
console.log(retvalue).
```

INTERACTIONS:

→ alert("This is a message")

* Shows a message
* alert function takes a string. → string

→ To ask user:

→ let name= prompt("What is your name?", "Guest")
↳ Default value.

→ Console.log(name).

→ For verification:

```
let delete-post = confirm("Do you really want to delete the post?");
console.log("deletepost");
```

LOOPS IN JAVASCRIPT:

→ Let friends = [A, B, C, D, E]

For Loop:

→ for (let index = 0; index < friends.length; index++){

→ console.log("Hello friend, " + friends[index]);}

Foreach Loop:

→ friends.ForEach(function f(element){
console.log("Hello friend," "element" + to modern javascript"); })

ForOf Loop:

→ for (element of ~~loop~~ friends){

→ console.log("Hello friend, "+ element + " to modern .js again);}

ForIn Loops;

→ let,
employee = {
name = Aba Hurairah
Salary = 200,000,
Role = Web Developer. }

→ for(key in employee){
console.log('The ${key} of employee is ${employee[key]}');}

While Loop In .js:

```
→ Let i=0;
→ While (i<4){
→     console.log(`${i} is less than 4`);
→     i++; }
```

Do While Loop In js: (Not Necessary)

Let, j=34

```
do {console.log(`${j} is less than 4 because we are inside do while loop`);
    j++; }
while(j<4);
```

NAVIGATING The DOM:

→ Let main = document.getelement byId('id').

→ console.log(main);

→ nav.innerHTML; (To get HTML in console as written in format {VS})

→ nav.innerHTML = " --- "
(To get only one element on page.)

→ let main = document.get element by class name('class')
console.log('containers')

container [0]
container [1]

→ Let sel = document.query selector ('.selector')
console.log (selector)
→ console.log('selector returns', sel) (valid for only 1st matching selectors.)
sel

→ let sel = document.query selector ALL ('.selector')
(For all matching selectors)

* About all css selectors can be used.

EVENTS IN JS:

Browser Event:
click
context menu
mouseover
mouseout
mouse down
mouse up
mouse move
submit
focus
Dom content Loaded
C&S transition end.

OnClick:

On Mouse over:

```
let para = document.getelement by Id('Para');
para.addEvent Listner('mouseover', function run(){
    alert('mouse Inside'); });
```