**Css tag : List-style:none**

**Types of css:**

1.Inline css

2.internal css

3.external css

Color: red;

Background-color:blue;

Text-decoration:color underline/overline wavy;

Text align:center;

Font-weight:600;

Letter-spacing:15px;

Line-height:2;

Rgb(255,0,0) (red)

Font-size:80px;

Font-family: font1, font2;(order preference)

Text-transform: uppercase;

**Css selectors**

1. \*{

select everything in the webpage

}

1. Element selector:

Img{

Select and style only the specific element

}

1. Selector list:

H1,p ,Img

{

Select and style all the specified element

}

1. #(id name){

Use id name to select and style

}

1. **.**(class name){

Use class name to select and style

}

1. desendant selector:

li a{

select element under any tag and style it

eg. Styling anchor element inside the Li

}

1. direct desendant selector:

li >a{

select the anchor tag which is the child of li tag

}

1. attribute selector:

input[type:”password”] {

}

**pseudo**

1. button:hover

{

Background-Color:blue;

}

1. **button::selection{**

**color:red;**

**}**

**Properties:**

**Width:100px;**

**Height:120px;**

**Border properties:**

**Border-width:5px;**

**Border-color:black;**

**Border-style:solid**

**Box-sizing:border-box;**

**(we can have 4values top right bottom left )**

**(we can also use border-left-width for border property)**

**Border: width|style|color (we can also use all the border properties under one by using the above syntax)**

**Border-radius:50% or 10px (for rounded edges)**

**Padding:1px 1px 2px 3px (top right bottom left)**

**Padding: 1px 4px(vertical horizontal)**

**Margin: 1px 1px 2px 3px (top right bottom left)**

**Display property:**

**Display:Inline;(it takes actual space of the element leaving the rest of the space)**

**Display:block;(it takes the whole space of the space for the element)**

**Opacity and alpha:**

**Rgba(123,345,545,(a=1))**

**(displays content but background alone gets transperency) range(0 to 1)**

**Opacity:0.3;**

**(partially display content with this attribute)**

**range(0 to 1)**

**position:**

**position: static;(it is default one)**

**position: relative;(we can use -ve values and we can change position by top, left,bottom,right)**

**position: absolute;**

**position: fixed;**

**transition:**

**transition: property name,duration,timing function,delay;**

**transform:**

**transform:rotate(degrees);**

**transform:scale();**

**box-shadow: (offset, offset ,blur radius,spread radius);**

**background-image: url();**

**background-position:top;(focuses on top of the image)**

**background-size:cover;**

**FLEXBOX:**

**Display:flex;**

Flex-direction:row;

Flex-direction:row-reverse;

Flex-direction:column;

Flex-direction:column-reverse;

Justify-content:flex-start;

Justify-content:flex-end;

Justify-content:center;

Justify-content: space-around;

Justify-content: space-between;

Justify-content: space-evenly;

Flex-wrap:wrap;

Flex-wrap:wrap-reverse;

Align-items:flex-start;

Align-items:flex-end;

align-content: space-around;

align-content: space-between;

<use n-thof child> align-self:flex-end;

Align-items:flex-start;

Flex-basis:200px;

Flex- grow:1;

Flex-shrink:1;

Flex: Flex- grow:1; | Flex-shrink:1;| flex-basis

Media queries:

@media (min-width or max-width=700px)

{

H1{

Color:purple;

}

}

Javascript:

Primitive types:

1.number(1 type +ve -ve decimal )

2. string

3.boolean

4.null

5.undefined

// comments

Nan – not a number

Variable declaration :

Let a=10;

Const (same as let but it is constant value it cannot be changed)

Nan (not a number)(null,placeholder)

Let firstName=”rishi” (string declaration using let)

firstName[3]

'h'

firstName.length

5

Ab= “lol”+”lol”

Ab

“Lollol”

Let firstName=”rishi”

FirstName.UpperCase()

RISHI

(accessing firstName displays lowercase LETTERs)

Let name= FirstName.UpperCase()

(accessing name displays UPPERCASE LETTERs)

Let firstName=” rishi ”

firstName.trim()

“rishi”

**Let firstName=” rishi ”**

**firstName.trim().UpperCase()**

“RISHI”

Let firstName =”helloman”

firstName.index(hello)

0

firstName.index(man)

5

(if we use index for element which is not inside the string then it returns -1 )

Let firstName =”helloman”

firstName.slice(2,6)

Lom

Let firstName =”hello man”

firstName.replace(man , guys)

**` (backtick)**

**Template literal**

` strings u want to display ${ 1+2+3or variable name } `

**===**

**!==**

**Console .log**

**Console.error**

**Console .warn**

**Alert(“hello”)**

**(pops a window )**

Prompt(“what is your age ?”)

(input)

parseInt(“101”)

* 101

To connect js file to html

<script src=”filename.js”></script>

Tip: insert js script at end of the body

**Logical operators**

**And (&&)**

**Or (||)**

**Not(!)**

Switch case:

switch(expression) {  
  case x:  
    *// code block*    break;  
  case y:  
    *// code block*    break;  
  default:  
    // code block  
}

**js datastructures:**

**array:**

**\*ordered collection of values**

**\*the values inside array can be changed after instantiating the array by using**

**Let colors=[“blue”,”rad”,”green”];**

**colors[1]=”red”**

**colors**

**o/p:**

**[“blue”,”red”,”green”];**

**syntax:**

**let colors=[]; // empty array**

**accessing arrays**

**let greet=[“hello”, “hi”,”bonjour”];**

**greet[0]**

**“hello”**

**greet[0][0]**

**“h”**

**greet[1]**

**“hi”**

**Array methods:**

**Push**

**let greet=[“hello”, “hi”,”bonjour”];**

**greet.push(“vanakkam”,”dhanyavadh”)**

**greet**

**[“hello”,“hi”,”bonjour”,“vanakkam”,”dhanyavadh”];**

**o/p:**

**5**

**(it just displays the length of array when we use push method we have to access the array to see the array elements)**

let funny=["smile","laught"]

funny[1]="laugh"

'laugh'

funny.push("feel","hehhehe")

o/p:

4

funny.pop()

'hehhehe'

**We can even add variables to access the popped element**

**Eg:**

**let heee=[1,2,3,4,5];**

**undefined**

**heee.shift()**

**o/p:**

**1**

**heee**

**o/p:**

**(4) [2, 3, 4, 5]**

**heee.unshift(1)**

**5**

Heee

(5) [1, 2, 3, 4, 5]

Array methods:

Push(values) //enter elements at last

Pop() //deletes elements at last

Shift() //deletes elements at first

unshift( values) //inserts elements at first

concate():

const helll=[1,2,3]

undefined

const heaven=[4,5,6,7]

undefined

const helhev=helll.concat(heaven)

undefined

helhev

(7) [1, 2, 3, 4, 5, 6, 7]

**Includes(values)**

const helll=[1,2,3]

undefined

const heaven=[4,5,6,7]

undefined

const helhev=helll.concat(heaven)

undefined

helhev

(7) [1, 2, 3, 4, 5, 6, 7]

helhev.includes(7)

true

reverse():

const helll=[1,2,3]

undefined

const heaven=[4,5,6,7]

undefined

const helhev=helll.concat(heaven)

undefined

helhev

(7) [1, 2, 3, 4, 5, 6, 7]

helhev.reverse()

(7) [7, 6, 5, 4, 3, 2, 1]

INDEXOF:

let ssa=["hello","hii","heeeee"];

undefined

ssa.indexOf("hii")

o/p:

1

Slice():

let hell=[10,20,30,40,50,60,70,80,90];

undefined

hell.slice(3,6)

o/p:

(3) [40, 50, 60]

splice(start,delete single or multiple elements from start point, insert new elements):

let hell=[10,20,30,40,50,60,70,80,90];

undefined

hell.splice(0,0,0)

[]

hell

(10) [0, 10, 20, 30, 40, 50, 60, 70, 80, 90]

hell.splice(9,1,100,110)

[90] // deleted item

hell

(11) [0, 10, 20, 30, 40, 50, 60, 70, 80, 100, 110]

Sort

Object literals :

(key value pair)

(turn everything into strings)

const hell={ firstName:"rishi",secondName:"mahadev",thirdName:"v"};

undefined

hell.firstName //accessing object values

'rishi'

hell.secondName //accessing object values

'mahadev'

hell["firstName"] //accessing object values

'rishi'

hell["thirdName"] //accessing object values

'v'

The values& keys outside the objects can use variables

To access

const eee={fstyear:2001,sndyear:2002};

let frnd="sndyear";

eee[frnd]

2002

Update values inside the objects

const hell={ firstName:"rishi",secondName:"mahadev",thirdName:"v"};

undefined

hell.firstName //accessing object values

'rishi'

hell.secondName //accessing object values

'mahadev'

hell["firstName"] //accessing object values

'rishi'

hell["thirdName"] //accessing object values

'v'

hell.thirdName="victory";

'victory'

hell

o/p:

{firstName: 'rishi', secondName: 'mahadev', thirdName: 'victory'}

Add values into objects:

const hell={ firstName:"rishi",secondName:"mahadev",thirdName:"v"};

undefined

hell.firstName //accessing object values

'rishi'

hell.secondName //accessing object values

'mahadev'

hell["firstName"] //accessing object values

'rishi'

hell["thirdName"] //accessing object values

'v'

hell.thirdName="victory";

'victory'

hell.fourthName="ajith"

'ajith'

Hell[“fifthName”]="vj"

'vj'

Hell

o/p:

o/p:

{firstName: 'rishi', secondName: 'mahadev', thirdName: 'victory', thirdName:"victory",fourthName:"ajith", fifthName:"vj"}

FOR LOOP :

for(let i=1;i<==10;i++) {

console.log(i);

}

o/p

prints no 1,2,3,4,5,6,7,8,9,10a

while loop:

const code="rishi";

let guess=prompt("enter the passcode:");

while(guess!==code){

guess=prompt("enter the passcode:");

}

console.log("congrats you have signed in....")

BREAK (escaping a loop):

let inp=prompt("say something man!!!!")

while(true){

inp=prompt(inp);

if(inp.toLowerCase()==="stop copying") break;

}

console.log("jeichitom maara")

o/p: jeichitom maara

**for of loop:**

**let subTopic=["hello","dghx","bgvj","fdxghtxuj","hjcjck","fjlkg"**

**,"dhfpld","dxgsion"];**

**for(topic of subTopic){**

**console.log("the topics are",topic)**

**}**

**o/p:**

**the topics are hello**

**the topics are dghx**

**the topics are bgvj**

**the topics are fdxghtxuj**

**the topics are hjcjck**

**the topics are fjlkg**

**the topics are dhfpld**

**the topics are dxgsion**

**example of (for of) loop:**

**const numbers = [1,2,3,4,5,6,7,8,9]; //DON'T CHANGE THIS LINE PLEASE!**

**// WRITE YOUR LOOP BELOW THIS LINE:**

**for(let num of numbers){**

**let squareNum = num\*num**

**console.log(squareNum)**

**}**

**For in loop:#refer net**

FUNCTIONS:

SYNTAX:

FUNCTION FUNCTIONNAME{

//RESUABLE CODE

}

function printHeart(){

console.log("<3")

}

o/p:<3

functions with arguments:

function rant(message){

console.log(`${message.toUpperCase()}`)

console.log(`${message.toUpperCase()}`)

console.log(`${message.toUpperCase()}`)

}

o/p:

rant("fuckkin")

VM627:2 FUCKKIN

VM627:3 FUCKKIN

VM627:4 FUCKKIN

functions with multiple arguments:

// define isSnakeEyes below:

function isSnakeEyes(firstNum,secondNum){

if(firstNum==1 && secondNum==1){

console.log(`${firstNum},${secondNum} is snake Eyes`);

}

else

{console.log(`${firstNum},${secondNum} is not snake Eyes`);

}

}

o/p isSnakeEyes(1,5)

1,5 is not snake eyes

RETURN KEYWORD:

(Stops the execution of function)

Function greet(){

Let wish=”hello ”;

Return wish;

}

**Function scope :**

let greet = "hel"

function hello(){

greet = "hello rishi"

}

console.log(greet, "(this is global variable)")

hello();

console.log(greet, "(this is local variable)")

case 1:

let greet = "hel"

function hello(){

//let greet = "hello rishi"

console.log(greet)

}

hello();

case 2:

let greet = "hel"

function hello(){

let greet = "hello rishi"

console.log(greet)

}

hello();

in case1 the local variable is not declared(commented) and hence it takes the value of global variable if both variables are of same names

in case 2 the global and local variables are declared but inside functions greet variable is declared as hello rishi and it get printed when calling the function

block scope:

wrong approach!!!!

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

var msg="hello";

console.log(msg);

}

console.log(i)

if we use var we can also access it outside the block and if in case we use let we cannot access outside the block(if we use let it is block scoped)

for(let i=0;i<=10;i++){

let msg="hello";

console.log(msg); // only hello is printed 11 times

}

console.log(i) // I is not defined

console.log(msg)// msg is not defined

becoz (loops or functions) are inside a block means enclosed by {}(curly braces) this is known as blocked scope

**LEXICAL SCOPE:**

In simple language, lexical scope is a variable defined outside your scope or upper scope is automatically available inside your scope which means you don't need to pass it there.the data can be accessed only in the decendent functions but not in ancestoral functions

Let a=10;

function greet(){

const wishes=["hello","vanakam","andiriki namaskaram","namasthe"];

function greetEveryone(){

for(i=0;i<wishes.length;i++){

console.log(`${i}.${wishes[i]}`);

}

}

greetEveryone();

}

o/p:

greet()

0.hello10

1.vanakam10

2.andiriki namaskaram10

3.namasthe10

FUNCTION EXPRESSION:

STORING functions as a variable(eg storing function in square variable.

it accepts other functions as arguments and

return a function)

const square= function functionexpression(num){

sqnum= Math.pow(num,2);

return sqnum;

}

square(7)

O/P: 49

DEFINING METHODS:

const math={

add: function addition(num1,num2){

return num1+num2;

},

sub: function subtraction(num1,num2){

return num1-num2;

},

mul: function multiplication(num1,num2){

num1\*num2;

},

div: function division(num1,num2){

return num1/num2;

},

exp2: function division(num){

return num\*num;

},

exp3: function division(num){

return num\*num\*num;

},

}

o/p: math.add(1,3) //calling

4

(or)

Short hand defining methods:

const math={

add (num1,num2){

return num1+num2;

},

sub(num1,num2){

return num1-num2;

},

mul (num1,num2){

num1\*num2;

},

div (num1,num2){

return num1/num2;

},

exp2 (num){

return num\*num;

},

exp3 (num){

return num\*num\*num;

},

}

o/p: math.add(1,3) //calling

4

THIS KEYWORD:

const hen ={

name: "Helen",

eggCount: 0,

layAnEgg() { e="EGG";

increaseEggCount= this.eggCount++;

return increaseEggCount,e ;

}

}

We can access the other properties inside objects by using “this” keyword.

TRY & CATCH:

function greetThreeTimes(msg){

try{threeTimes=msg+msg+msg;

console.log(threeTimes.toUpperCase());

}

catch(e){

console.log(`${e},${msg}is not a string please enter a string`)

}

}

greetThreeTimes("hell")

o/p HELLHELLHELL

DOM: