Assignment

① Semantic tag → This are the tags which describe the meaning to the browser & the developer
Eg:- form, table, figure.

② Selectors → Selectors select the Element which you want to style
Eg:- Class, id, univerasal [• , #, *] respectively
[order → universal > id > class > elements]

③ CSS Combinators → The use of combinators is to combin the selector & give relation.

(i) Descendant → give normal space to combin two selectors.

(2) child → (>) style to child Element of specified Element

(3) Adjacent → (+) used for immediate sibling.

(4) General sibling → (~) to all siblings style is added.

adding Nested web page in HTML → webpage Embedded into another webpage

→ can add by using (i) frame & Embed

CSS box model → model defines the design & layout of Elements of CSS

Like → margin, padding, border, content.
(around), (inside), Content) (input).

Doctype → Doctype is declaration which instruct the web browser about the HTML page.

There are 3 kinds of Doctypes

(i) strict, Transitional, frameset.

Strict → does'nt allow much attributes & frames.

tranisitional → allows user to use certine Elements & attributes.

frameset → can use frames replaced framest by body Element.

CSS & uses → Cascading style sheets (External style sheet)
→ uses → used as External style sheet to provide a disciplined code
→ gives front end more user experience.

CSS works → works as cascading to a HTML Page where the style sheet is link Externally through <Link tag>. add provids certain style to Elements through selectors.

New features in HTML 5 → intro of audio/video tags
— Embeds video & audio
→ Header & footer → New layouts
→ attributes like → placeholder, Email,
→ progress tag.

Likes & dislikes of CSS → ~~adv~~ merits ① Bandwidth, site wide consistency
② content sep from presentation.

Demerits ① Limited security
② Extra work for developers.

Position property → static, relative, absolute, fixed, sticky

Static → default position

relative → same as static but can change the position (L, R, T, B).

absolute → same as relative but takes parent element position.

fixed → content get fixed at a defined posi^t

Sticky → Similar to fixed but when page is scrolled it get sticked defined position

Assignment JS

<u>Hoisting.</u>

→ default behavior of moving <u>declarations</u> to the top.
→ defining ~~the~~ / declaring all variables at the beginning of Every scope.

<u>Call & Apply method</u>

Call → used to invoke function by an object as argument.

apply → it is similar to call but the arrguments are taken in an array.

<u>Object methods</u> → methods are sored as a obj properties

Accessing obj methods

Syntax → objname. method ()

Let student = { Name : 'Harsha' ; Age : 24 ;

{ display is a method <- - - display : funct () { this is a property

return this. name + " : " + this age. of student . obj}

consol. log (student . display ()) →

<u>Set Timeout ()</u> → acts as a buffer zone for a function to display / act as a latency

<u>Call back</u> funct → function which used as an argument to another funct.

flex box property ⊛ → Display (utility), responsiveness
(CSS3)
⊛ → Direction → • flex row, flex row - reverse
(inside)

→ used for responsiveness.
⊛ → justify content → start, end, centre
(horizontal)

→
⊛ → align item → start, end, centre.
(vertical)

⊛ → flex wrap → takes space knowingly.

⊛ → flex → length of a flex item.

Display property → mainly → inline, block, flex, grid
inline - block.

(1) inline → default & no change in h & w

(2) block → takes whole width of content.

(3) inline - block → used to give w & H
to inline elements.

Destructuring → unpacking of Arrays & objects & assign them to variable.

Type of → Type of gives use out what kind of Data types does a ~~con~~ variable has.

Spread V/s rest → Spread → it Expands an iterable such as an array (eg [... variable])

→ rest op → rest op compresses the iterable & can be put.

Null v/s undefined v/s Empty → Null → absence of value it is primitive value

undefined → value doesn't Exist

Empty → when an value is Extracted from array & it shows Empty.

Declaration v/s defination → Declaration → comes after a key word Let, const, var
→ it is actually variable

→ Defination → it is value of an variable if there is no defination for vari it shows undefined.

Arrow funct v/s Normal funct → $\boxed{\text{Arrow}}$
↳ ES6 version
↳ Short syntx for function
↳ no need of funct Name.

$\boxed{\text{Normal funct}}$
↳ funct should have name
↳ must give funct keyword.
↳ & have a proper funct syntx.

function to reverse a string (with & without using built in method)

Built in

```
function revString (str) {

        let split string = str.split("");

        let rever string = ~~reverse~~ split string. reverse (");

        let join string = rever string .join(")

            return join string;

        }

        rever~~se~~ string (" Hello"),
```

No built in

```
  function rev (str) {

        let r = " ";

        for (let i = str. length -1 , i>0 , i--) {

            r+ = str[i]

            }

            return r;

          }
        console.log (reverse("Javas"))
```

frequence / occurrences in a array

```
function freq (arr, x). {

    let count = 0;
    let n = arr.length;

   for (let i = 0 ; i<n ; i++)
     if (arr[i] == = x)
        count ++;
          return count;
   }

 Let arr = [~~1, 2, 2, 2, 2, 3, 4, 7~~] [1, 1, 2, 3, 4, 4]
  Let x = ~~7~~;
      x = 4;
  consol.log (x + ~~count~~ freq (arr, x))
```

Palindrom

```
pal = (inp) => inp === inp. split(''). join('').
    oup put = pal ("CABBAGE").
```

function to sort no. in ascending order

```
↳ Let numbers = [ 50, 90, 0, 2, 20, 10]

    ↳ number. Sort (function (a, b) {a - b} );

      ↳ Console.log(number);
```

function to duplicate in an array (set)

```
↳ Const

      fone = [4, 1, 2, 1, 3, 1];

      Let result = (fone) => ~~new Set(fone)~~
          {
               return [...new Set(fone)];

          }

      Console.log (result (fone)));
```

Closures → combination of func^t bundled together refrence to it surrounding state
Closures gives you access to an outer func^t Scope from an inner func't.

Classes → classes are not an object but it is template for JS objects. & use constructor method.

Inheritance → Enables you to define a class that takes all funct. from a parent class & allows you to add more.

Uses of spread operators in object
→ clone an object
→ merging object
→

Super → is a keyword used to call the constructor of its parent class to access the parents properties & methods

static → Static method for a class only not Object
→ Such to create or clone object.

~~Let~~ closure

```
Let ans = "global";
    Const outer = () => {
          Let outerVar = "outer";
           console.log(ans, outerVar);

    outer();
```

Armstrong Number

$153 = n^n = 1\times1\times1 + 5\times5\times5 + 3\times3\times3$

```
 Let sum = 0;
 const number = prompt()

   Let temp = number;
    while (temp>0) {
           Let remainder = temp % 10;
           Sum += reminder * reminder x reminder

   temp = parseInt(temp/10);
if (sum = number) {
        console.log(`${number} is a A`);
              }

   else {
   console.log(`${number} is not a A.`);
   }
```

Escape characters → helps in interpreted in some alternate way then what we intendend too

Eg :- \b → backspace \n → new line

Type of

Null	→ object
undefined	→ undefined
array	→ object
object	→ object

Output of (comparimon)

null = = undefined → false.
null = = = undefind → false

Ternary op
→ Evalutes a condition & Executes a block of code based on condition.

[condition ? Exp1 : Exp2]

How do you loop through on objects
→ for. in loop
→ object. key method
→ object. Values method
→ object. Entries method.

function to reverse a string (with & without using built in method)

Built in

```
function revString (str) {

        Let split string = str.split("");

        Let rever String = ~~reverse~~ split string. reverse ('');

        Let join string = rever string .join('')

            return join string;

        }

        reve~~r~~ string (" Hello "),
```

No built in

```
    function rev (str) {

            Let r = " ";

            for (Let i = str.length -1 , i>0 , i--) {

                ~~{~~ r+ = str[1]

            }

            return r;

        }
        Console . log (reverse(" Javas"))
```

Palindrom

```
function pal (word) {

        const palsplit = word.split("")   ;

        cont revword = palsplit.reverse('');

        cont joinword = rev word . join ("");

          if (string word = = joinword) {

                     consol.log ('It is a palindrome');

             }

          Else { console . log ( Its is not a palindrom);

          const string = prompt ('Enter a string:'")

                ~~Check palindrom~~ pal (word);
```

Closure

```
function word() {

          Let name = 'John';

          function display () {

                   return 'Hi' + ' ' + name;

           }

          return display;

Const g1 = word();
   C.log (g1);    , c.log (g1());
```

function to sort no. in ascending order

```
↳ Let numbers = [ 50, 90, 0, 2, 20, 10 ]

    ↳ number. Sort (function (a, b) {a - b} );

        ↳ Console. log (number);
```

function to duplicate in an array (set)

```
↳ Const
      fone = [ 4, 1, 2, 1, 3, 1 ];

      Let result = (fone) => ~~[illegible] ... new Set (fone)~~
         {
             return [... new Set(fone) ];

         }
         Console. log (result (fone) ));
```

→ Write a funct to merge 2 arrays & removing duplicates with out using inbuilt function.

```
Var array1. = [1,2,3]
Var array2 = [2,3,0,1]

    funct link (array1, array2) {
        Var result = array = [ ];
            Var arro = arr. length.
        Var assoc = { };
            While (length) { var item = arr[Leng];
        if (!assoc [item])
        { result . array . unshif (item);
                assoc [item] = true;
        }

    };
```