How to link multiple css file :

In main . css file add the below command and this main .css file should be linked to main html page

@import url("./demo1.css");

@import url("./demo2.css");

@import url("./demo3.css");

Selectors:

 Selectors:

    1:Simple Selectors:

    2:combination Selectors

    3:pseudo Selectors

    4:pseudo element Selectors

1:Simple Selectors:

-> ID : symbol is # , id is unique

-> Class : symbol is . dot

->Groupall : no symbol , but separated by comma

->Tagname :

->Universal : : symbol is \* star

Html file:

<h1>hii</h1>

<h2 id="nan">hello</h2>

<h3 class="man">bye </h3>

<h4 class="man">bye bye </h4>

<h5 class="man">preritha</h5>

<h6 id="jkl">reetu</h6>

Css file:

h2{

background-color: brown;}

#nan{

    background-color: blue;}

.man{

    background-color: black;}

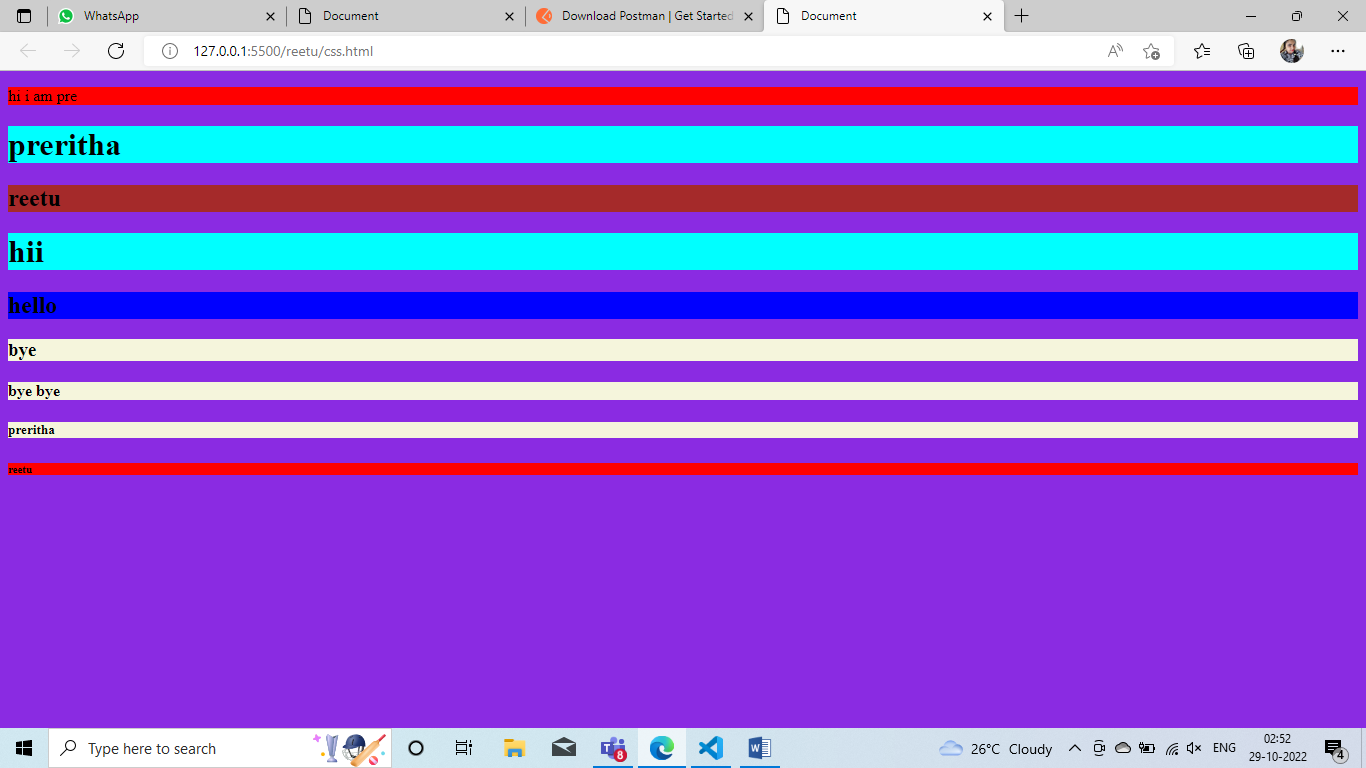
.man,h3{

    background-color: beige;}

#jkl{

    background-color: red;}

\*{background-color: blueviolet;}



=============================================================================

**Combination selectors:**

Decendent Selector: symbol is (-) and selects only direct parent

Child selector: symbol is (>) and selects direct children.

Adjacent sibling: symbol is (+) and select 1 sister or sibling

General sibing: symbol is (~) and select many sisters or siblings

Html file:

<div>

    <p>1son</p>

    <p>2son</p>

    <p>3son</p>

    <section><p>4son</p></section>

</div>

<p>1sister</p>

<p>2sister</p>

<p>3sister</p>

Css file:

div p{

    background-color: aqua;

}

div > p{

    background-color: red;

}

div + p{

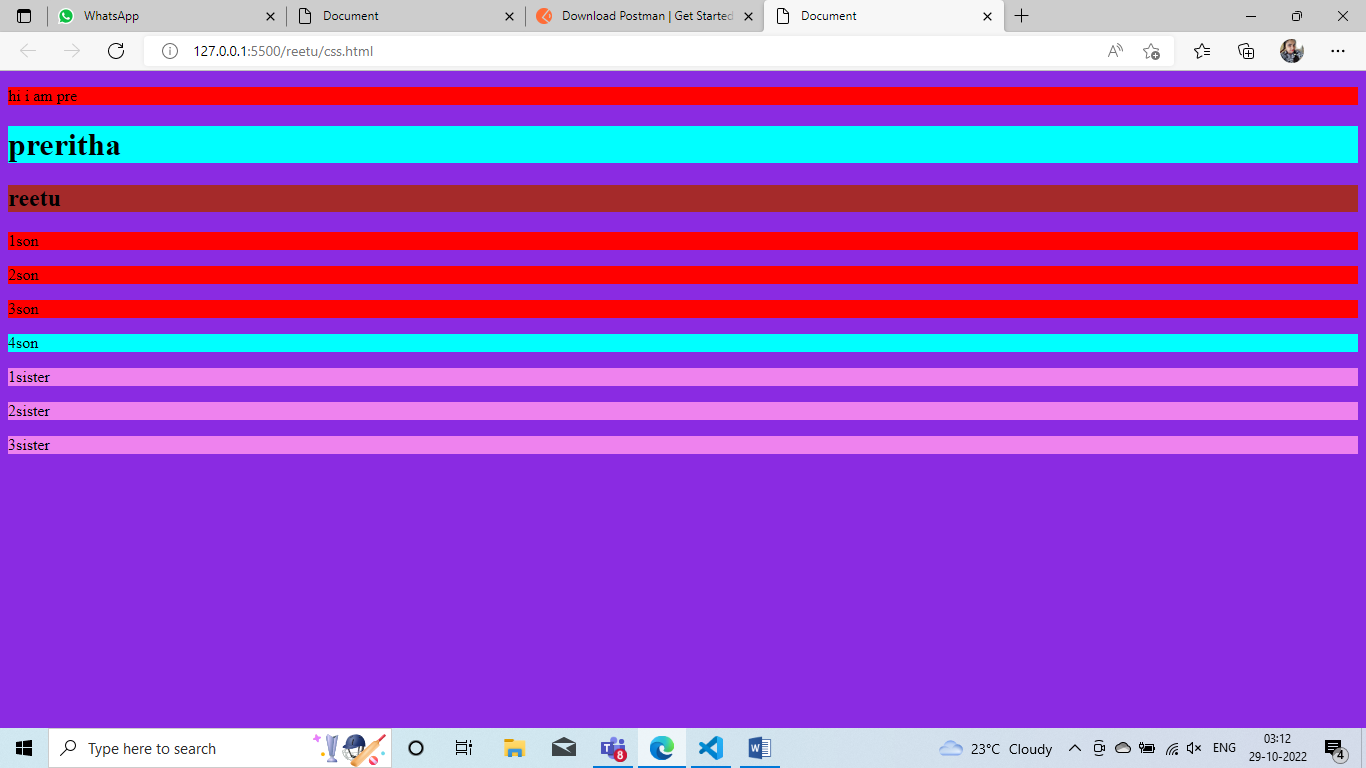
    background-color: blue;

}

div ~ p{

    background-color: violet;

}



**Pseudo Class selector:** symbol is :

**Dynamic Selector:**

: Link -> link color

: visited -> color of link will be changed

: hover -> whn we move to the link , or try to click on link, color gets changed

: active -> whn we click on clink then color changes

In html page

<a href="https://www.bing.com/images/search?q=images&form=HDRSC3&first=1&tsc=ImageHoverTitle">Click</a>

In Assignment. css file

#pre

{

    background-image: url(https://th.bing.com/th/id/OIP.X\_65uIJkSF8bJl\_zyU4twgHaEo?w=272&h=180&c=7&r=0&o=5&pid=1.7);

}

:link

{

    color: brown;

}

:visited{

    color: aqua;

}

:hover{

    color: blue;

}

:active{

    color:black;

}

**Structural Selector:**

<p>pre</p>

     <p>gayi</p>

     <p>reetu</p>

     <div>

        <p>eno</p>

        <p>ondhu</p>

        <p>visya</p>

     </div>

     <section>

        <p>jsp</p>

        <p>jspiders</p>

     </section>

**Html file:**

**: first child** -> first child will be taken

 p:first-child{

    color: red;

}

Pre gayi reetu

Eno ondhu visya

Jsp jspiders

**: last child ->** last child will be selected

p:last-child{

    color: red;

}

Pre gayi reetu

Eno ondhu visya

Jsp jspiders

**: Nth child->** we have to give the number

p:nth-child(2){

    color: red;

}

Pre gayi reetu

Eno ondhu visya

Jsp jspiders

: First of child->

: Last of child->

**Psuedo element selector: symbol is : :**

: : First line-> first line colour gets changed

: : First letter-> first letter colour gets changed

: : Before -> adding picture before the line

: : After-> adding picture after the line

: : Marker->markers of unpaired dots r changing

: : Selector-> if we select the line , tat line colour and background colour changes

**In html file:**

<p>Lorem ipsum dolor sit amet consectetur, adipisicing elit. Voluptatibus, illum dolores excepturi aliquid et fuga

        minima nostrum consequuntur suscipit repellendus, nesciunt commodi facere, aperiam laudantium dolorum soluta

        alias praesentium eveniet.</p>

    <ul>

        <li>pre</li>

        <li>gayi</li>

        <li>reetu</li>

        <li>myself</li>

     </ul>

In css file

::first-line

{

    color: blue;

}

::first-letter{

    color: brown;

}

/\* ::before{

    content: url("./image.jpg.jfif");

} \*/

::marker{

    color: aqua;

}

::selection{

    color: blueviolet;

    background-color: brown;

}

**Display properties:**

In html file:

 <h1>India is my country</h1>

    <nav id="pre">nav</nav>

        <div >hi </div>

        <div>bye</div>

        <div>good bye </div>

    </nav>

In css file:

#pre{

    border: 20px solid red;

    height: 10px;

    display: flex;

    justify-content: space-evenly;

    align-items: center;

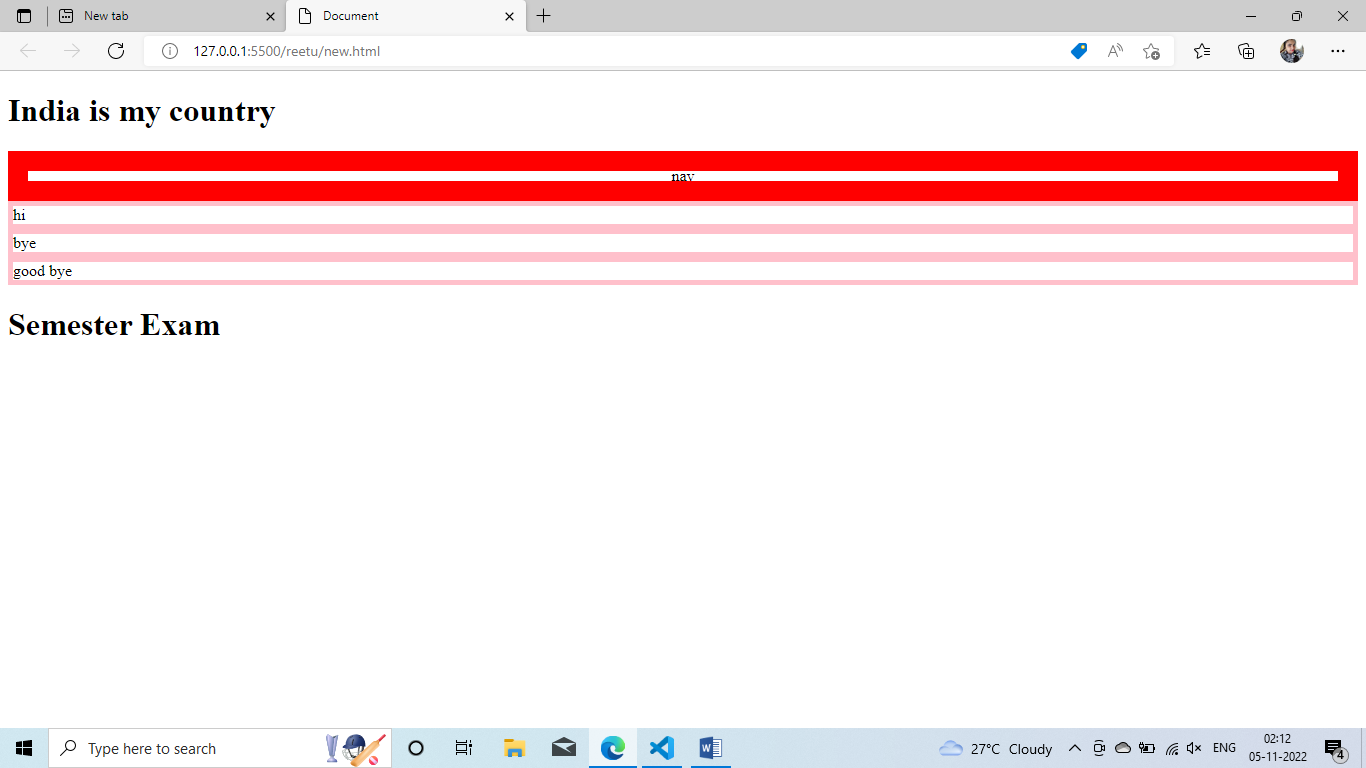
}

div{

    border: 5px solid pink;

    align-items: center;

}



Colors:

R->red

G->Green-

B->Blue

A->alpha-Opecity- > 0-1

**19-11-22**

**Java Script:**

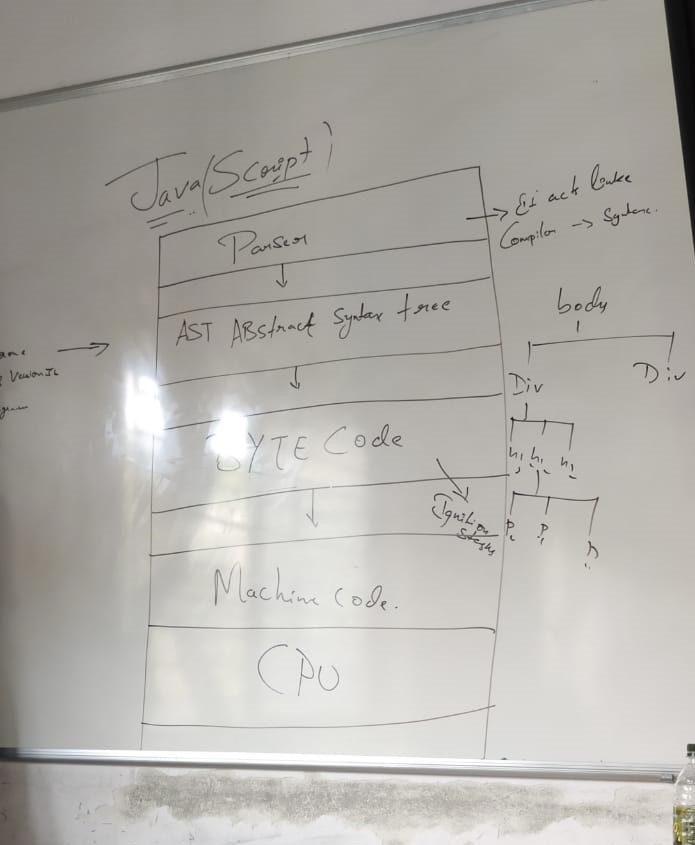
* It is browser and software.
* Dynamically typed language
* Brendian Eich in 1995
* First name is Mocha 1995 september
* Live script 1995 nov
* ECMA took JavaScript
* Current version is ECMA 6
* ES6 version
* ECMA script

**Java :**

* Programming language
* Executed by jvm
* Strictly typed language
* Multi threaded
* Independent language

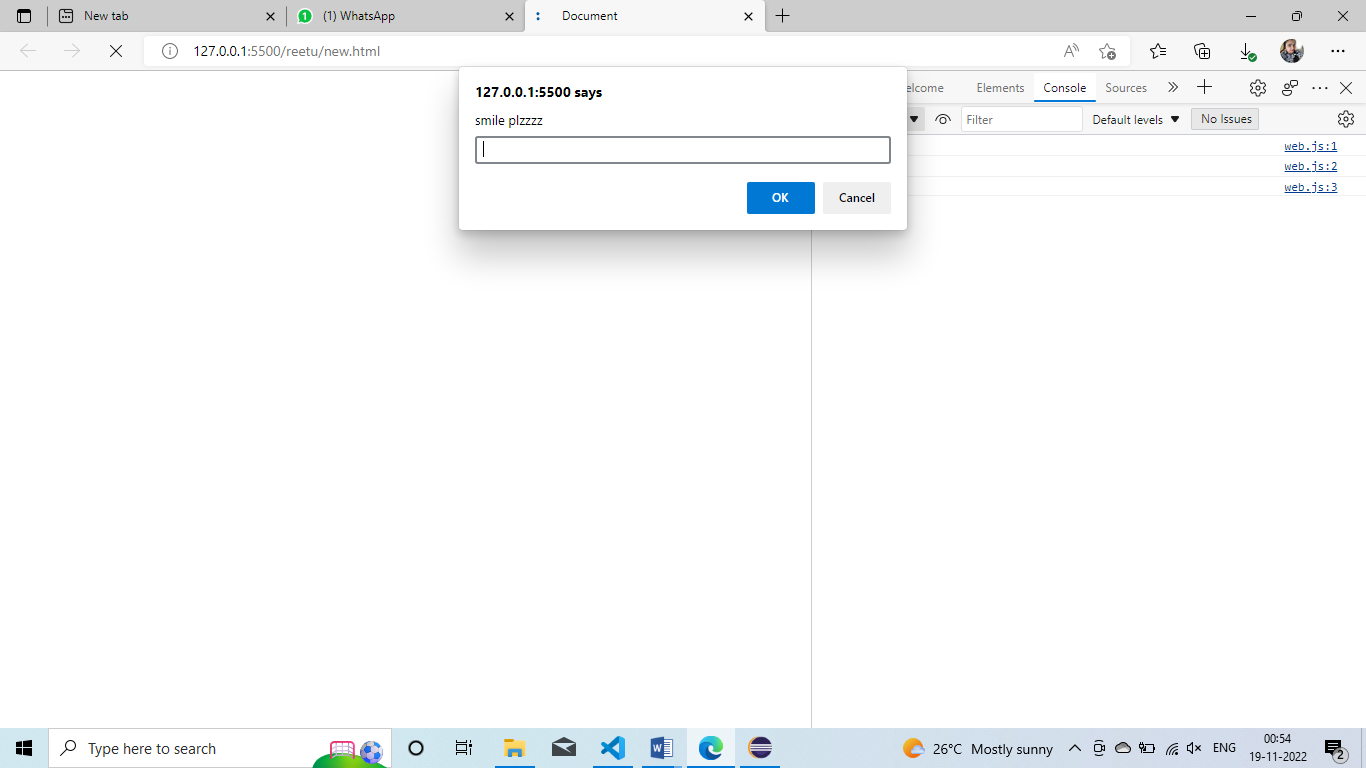
**JavaScript:**

* Scripting language
* Executed in all browsers
* Weekly typed language
* Single threaded
* Dependent language (linked with html, CSS)
* Chrome –version 8
* Parser acts like complier and checks for syntax
* AST-> Abstract Syntax Tree (consists of code which s converted by Parser)
* Byte code (ACT code will be converted to byte code) and ignition starts and its converted t machine code .
* If we are running JS outside the browser then its Node.js
* Node.js was invented by Rayan dal



**Output methods:**

* Console.log() // for printing
* Console is Object, log is a function which performs some function.
* BOM -> Browser Object Module
* Inside BOM , we have prompt() , confirm() , alert(),DOM()
* **alert()-> pop up with only ok**
* **confirm() -> pop up with only ok and confirm**
* **prompt() -> pop up with only ok along with text field**
* console.log(alert("smile plzzzz"));
* console.log(confirm("yes i confirm"));
* console.log(prompt("smile plzzzz"));



* DOM -> Document Object Module
* Inside DOM we have document.writeln() and Document.write()

**Data Types:**

* Primitive Data type
* String(“double invited coma” , ’ single invited coma‘ ,`this is called back tick `)
* Number
* Undefined
* Null
* Boolean
* Bigint(1n)
* console.log(`this is my output : ${10+20}`);

String is a function.

console.log(`this is my output : ${10+20}`);

console.log(typeof console);              // object

console.log(typeof log);                  // undefined

console.log(typeof String)              //function

console.log(typeof Number)              //function

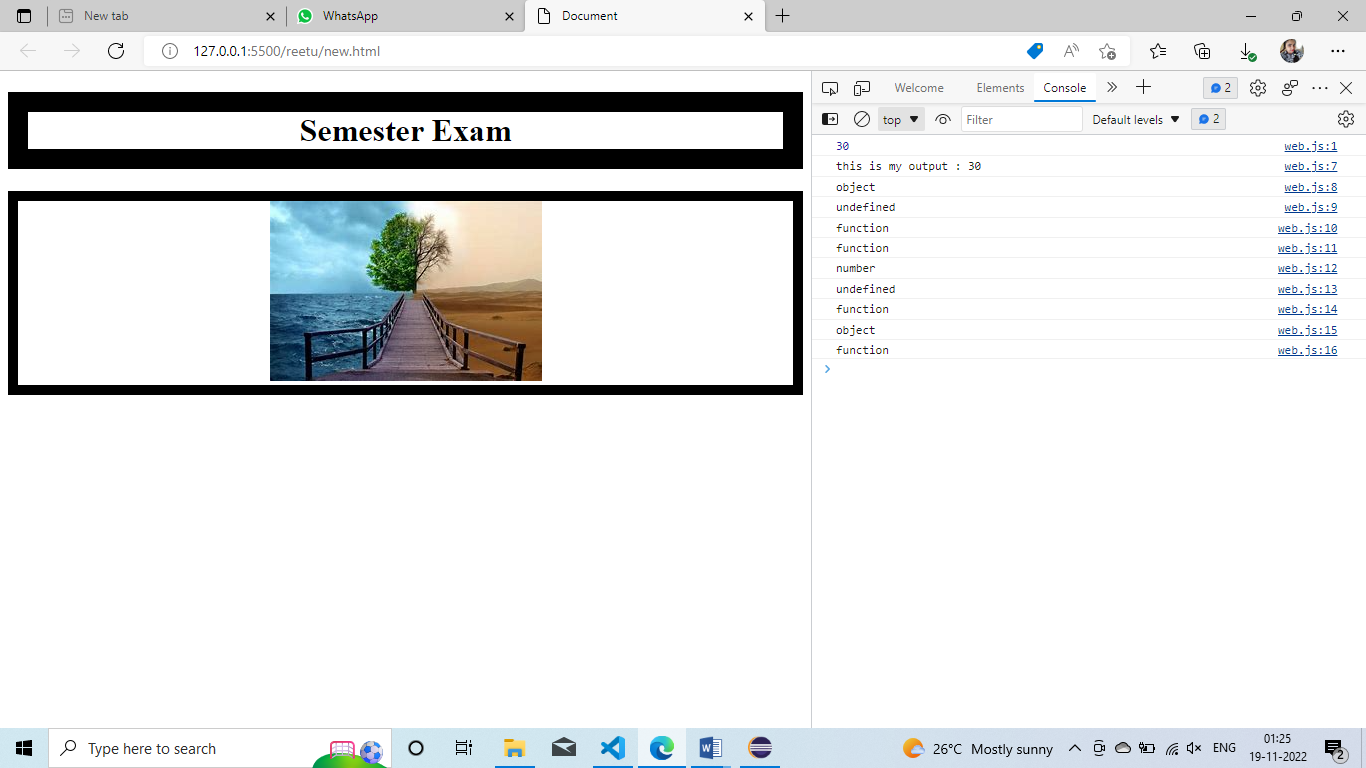
console.log(typeof 123)                  // number

console.log(typeof undefined)            // undefined

console.log(typeof Boolean)              //function

console.log(typeof null)                 // object

console.log(typeof BigInt)                //function



* Non-Primitive Data type
  + - Functions
    - Arrays
    - Objects
* Math Object
* Date Object
* Time Object

**20-11-22**

**Variable:**

**Var:** (Global variable)

* Declaration is possible
* Iniatialization and declaration in same line is possible
* Reiniatialization and declaration is possible
* var b=10;
* console.log(b);
* var b=1000;
* console.log(b);

output:

10

1000

**Let:** (local variable)

* Declaration is possible
* Iniatialization and declaration is possible
* Iniatialization and declaration is not possible
* let c;
* console.log(c);

output: undefined

let c=200;

console.log(c);

output: 200

let c=200;

console.log(c);

let c=500;

console.log(c);

output: Iniatialization and declaration is not possible

**Const:** (local variable)

* Declaration is not possible
* Iniatialization and declaration in same line is possible
* Reiniatialization and declaration not possible
* const a=20;
* console.log(a);

if we type only below , then it is var only

d=900;

console.log(d);

**Difference btw Var and let:**

**Var:**

* Been available from beginning of JS
* It has function scope
* Varibales will be hoisted

Let:

* Introduced as a part of ES6
* Hoisting is not possible
* It has a block scope

**Hoisting:**

* Hoisting is JS mechanism where variables and function declaration are moved to the top of the scope before code execution.
* JS only hoist declarations not initialation

console.log(t);

var t=800;

output: undefined

console.log(t);

let t=800;

output: error

**TDZ= temporal dead zone**

It is a behaviour in JS that occurs when declaring a variable with let and const keywords but not with var.

In ECMA script 6 , accessing let or const variable before its declaration causes a reference errors .

The time span when that happens between the creation of variable binding and its declaration is called as TDZ

Type Casting : 2 types

1. Implicit Type Casting
2. Explicit Type Casting

var a=Number(prompt("enter the first num"));

var b=Number(prompt("enter the second num"));

console.log(a+b);

output : 300

In the above program if we don’t put Number then output will be 100200

var c=Number(prompt("enter the first num"));

var d=Number(prompt("enter the second num"));

var e=c+d;

var f=c\*d;

console.log("this is my addition output : " + e);

console.log("this is my addition output : " + f);

console.log(`this is my addition output : ${c-d}`);

this is my addition output : 30

web.js:52 this is my addition output : 200

web.js:54 this is my addition output : -10

Operators:

Comparison Operator:

1. == -> compares 2 values
2. === ->
3. !== ->
4. !=== ->

var x=5;

var y=10;

console.log(x==y);        //false

 console.log(x!==y);        //true

console.log(x===y);      //false

console.log(x=="5");        //true

console.log(x==="5");         // false

console.log(x!==y);

26/11/22

String Methods:

let x="preritha"

console.log(x);                       //preritha

console.log(x.length);                  // 8

console.log(x.charAt(3));               // r

console.log(x.indexOf("a"));           // 7

console.log(x.lastIndexOf("e"));        // 2

console.log(x.toLocaleLowerCase());     // preritha

console.log(x.toLocaleUpperCase());     // PRERITHA

console.log(x.split(""));               // (8) ['p', 'r', 'e', 'r', 'i', 't', 'h', 'a']

console.log(x.split("").reverse());             // (8) ['a', 'h', 't', 'i', 'r', 'e', 'r', 'p']

console.log(x.split("").reverse().join());      //a,h,t,i,r,e,r,p

console.log(x.length);

gives the length of the string

**Functions:**

1: Anonymous Function

function ()

{

console.log(20+30);

}

x()

// error

2: Named Function

let x=function pre()

{

console.log(20+30);

}

x()

3: Function with expression

let x=function ()

{

console.log(20+30);

}

x()

4: Arrow Function

For arrow function , there is no need of { } braces

let x=()=>

console.log(20+30);

x()

5: Immediate invoking Function:

Learning in DOM

6: First class Function : same as function with expression

7: Higher order Function:

function higherOrderFunction(a,b,task)

{

    let res=task(a,b);

    console.log(res);

}

let add =higherOrderFunction(20,30,function(a,b)

{

    return (a+b);

})

let mul =higherOrderFunction(20,30,function(a,b)

{

    return (a\*b);

})

8: Addevent listener:

Learning in DOM

9: Nested Function: function inside function called as nested function

function first(){

    var b=200;

    let c=300;

    console.log(b);

    console.log(c);

    function second(){

    var g=200;

    let h=300;

    console.log(g);

    console.log(h);

       function third(){

        let d=1000;

        const e=20000;

        console.log(d);

       console.log(e);

       }

        third()    // return third

    }

    second()       // return second

}

first()          // first()()()

first 200

first 300

seocnd 200

second 300

third 1000 third 20000

=================================================

Note: If we give console statement outside of function then, we need to add return the inside the function.

Simple example

var c=Number(prompt("enter the first num"));

var d=Number(prompt("enter the second num"));

let x=()=>

console.log("addition output is " + (c+d));

console.log("multiplication output is " + c\*d);

x()

function higherOrderFunction(a,b,c,task)

{

    let res=task(a,b,c);

    console.log(res);

}

let add =higherOrderFunction(20,30,40,function(a,b,c)

{

    return (a+b+c);

})

let mul =higherOrderFunction(20,30,40,function(a,b,c)

{

    return (a-b-c);

})

let sub=higherOrderFunction(20,30,40,function(a,b,c)

{

    return (a\*b\*c);

})

let div =higherOrderFunction(20,30,40,function(a,b,c)

{

    return (a/b/c);

})

let x=20;

var b=100;

function dingi()

{

    var a=1000;

    let v=2000;

    const q=30000;

    console.log(a);

    console.log(v);

    console.log(q);

}

dingi()

console.log(x);

console.log(b);

1000

2000

30000

20

100

**GEC : Global Execution Context**

* GEC is a default or first execution context that is created by JS engine before code is executed .
* All the global code that is not inside a function or object will be executed inside the global execution context .
* Since JS engine is single threaded , there will be only one global environment and only one global execution context .
* Has 2 context

1: Variable State

2: Execution or Function State

Var- global -> it will be inside global scope

Let and const -> local ->it will be inside script scope

console.log("start");

//console.log(a);                     // undefined // hositing

var a=20;

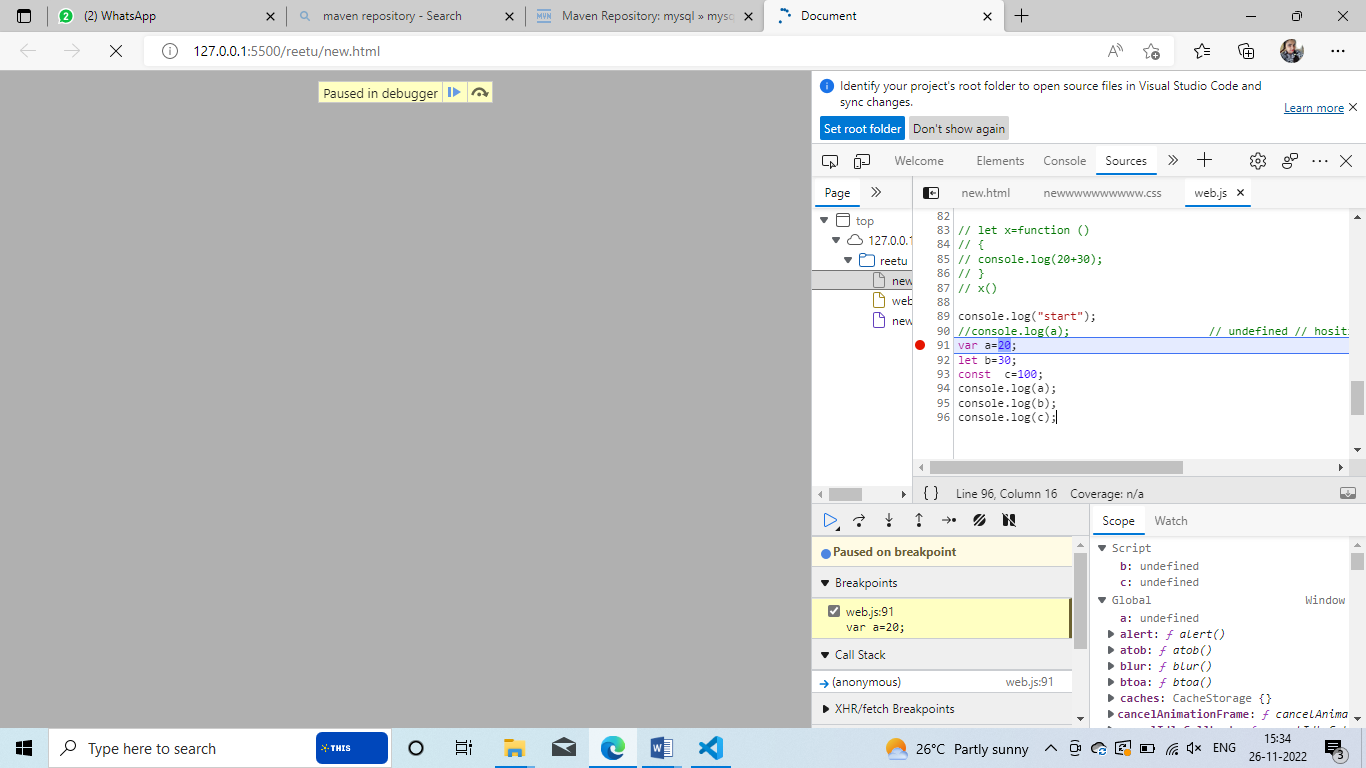
let b=30;

const  c=100;

console.log(a);

console.log(b);

console.log(c);



**What is function execution context?**

Whenever a function is invoked , JS engine creats a different type of execution context known as function execution context within the global execution context to evaluate and execute the code within that function.

**What is Closure:**

Closure is a combination of function and lexical environment within which the function was declared ,It is an inner function that has access to the outer or inclosing function variables known as Closure.

**Arrays :**

It is mixture of Homogenous and heterogeneous data.

Methods in an Array:

Push

Pop

Shift

Unshift

Splice

slice

let x=[100,200,3003,400,500]

console.log(x);   // [100, 200, 3003, 400, 500]

x.push("600"); // add to last index

console.log(x);   //[100, 200, 3003, 400, 500, '600']

x.pop();           // removes last value

console.log(x);   //[100, 200, 3003, 400, 500]

x.unshift(800);   //  adds value to fisrt of the array

console.log(x);    //[800, 200, 3003, 400, 500]

x.shift();        // removes first element

console.log(x);   //[200, 3003, 400, 500]

x.splice(3);       // only gives 3 values

console.log(x);    //[800, 200, 3003]

x.splice(3,0,10000);       // adding value to third index

console.log(x);             //[800, 200, 3003, 10000]

console.log(x.slice(2,5));     // only getting the values btw index of 2 -5 tat is 3003 400 500  // output [3003, 10000]

let x="preritha";

console.log(x);               //preritha

console.log(x.split(""));    // ['p', 'r', 'e', 'r', 'i', 't', 'h', 'a']

console.log(x.split("").reverse());    // ['a', 'h', 't', 'i', 'r', 'e', 'r', 'p']

console.log(x.split("").reverse().join());    //a,h,t,i,r,e,r,p

Split() converts string to array :

Reverse() is going the reverse the array

 let arr=[1,2,3,4,5,6,6,7,8]

let x=arr.filter((z)=>{

    return z<5                 // returns values less than 5 or filter the values

})

console.log(x);              // [1, 2, 3, 4]

let y=arr.map((val)=>{

    return val+50                 // maps to every values

})

console.log(y);                 //[51, 52, 53, 54, 55, 56, 56, 57, 58]

let z=arr.reduce((acc,val)=>{

return acc\*val                          // values get multiplied and gives the values

},5)

console.log(z)                         // 1209600

**For loops :**

let arr=[500,1000,2000,3000,4000,6000,10000]

  console.log(arr.indexOf(2000));                     //2

for(arr1 in arr){

    console.log(arr1);                               // gives index values 0 1 2 3 4 5 6

}

for(arr1 of arr){

    console.log(arr1);                               // gives values 500,1000,2000,3000,4000,6000,10000

}

arr.forEach((values,index)=>{

    console.log(`this is my values ${values+100} and index value ${index}`);

})

//ouput for above ex

// this is my values 600 and index value 0

// this is my values 1100 and index value 1

// this is my values 2100 and index value 2

//  this is my values 3100 and index value 3

//  this is my values 4100 and index value 4

//  this is my values 6100 and index value 5

//  this is my values 10100 and index value 6

==========================================================================

**Objects :**

Object is real world entity which has some state and behaviour.

let obj=

{

id:1234,

name:"darshan",

comp:"torry",

}

console.log(obj.id);                     // 1234

console.log(obj.name);                    // darshan

console.log(obj.designation="developer");  // adding the key and value

console.log(obj);        // {id: 1234, name: 'darshan', comp: 'torry', designation: 'developer'}

console.log(Object.keys(obj));        //['id', 'name', 'comp', 'designation']

console.log(Object.values(obj));       // [1234, 'darshan', 'torry', 'developer']

console.log(Object.entries(obj));   //  ['id', 1234] ['name', 'darshan'] ['comp', 'torry'] ['designation', 'developer']

// In seal method , we can only update , we cannot do delete and other operations

let x=Object.seal(obj)

console.log(x);                         //{id: 1234, name: 'darshan', comp: 'torry'}

console.log(x.location="bangalore");    // bangalore

console.log(x);                   //{id: 1234, name: 'darshan', comp: 'torry'}

console.log(x.name="pre");         // {id: 1234, name: 'pre', comp: 'torry'}     , here only name can be chnaged

console.log(delete(x));                  //false

//In freeze method we cannot do any operations

let x=Object.freeze(obj);

console.log(x);                         //{id: 1234, name: 'darshan', comp: 'torry'}

console.log(x.id="bangalore");    // bangalore

console.log(x);                    //id: 1234, name: 'darshan', comp: 'torry'}

* **Date object:**
* let x=new Date()
* console.log(x);                  //Sun Dec 04 2022 14:51:22 GMT+0530 (India Standard Time)
* console.log(x.getDate());        // 4 (today is 4th of december )
* console.log(x.getDay());         //0
* console.log(x.getFullYear());    //2022
* console.log(x.getHours());        // 14 ( approximate time that is now its 2)
* console.log(x.getMilliseconds());   //572
* console.log(x.getMonth());          //11
* console.log(x.getSeconds());        //30
* console.log(x.getTime());
* **Math object:**

console.log(Math.floor(23.9));        //23

console.log(Math.ceil(20.9));          //21

console.log(Math.round(23.9));         //24

console.log(Math.floor(23.9));   //23 (it will round off to nearest value 23)

console.log(Math.ceil(23.1));     //21    (it will be 24)

console.log(Math.sqrt(100));           //10

console.log(Math.cbrt(100));           //4.641588833612779

console.log(Math.random());             //0.8066223419103675

* **Time object:**
* To make synchronous to asynchoronous
* SetTimeout: it is a method , which is used to call a function or evaluate an expression after a specified number of milliseconds .
* SetInternal: : it is a method , which is used to call a function or evaluate an expression after a specified Intervals.

var a="i am first";

console.log(a);

let x=setTimeout(()=>{

    console.log("i am late")

},10000)

var b="i am second";

console.log(b);               //output will be like i am first i am second  , i am late comes after 10 seconds

let x=setInterval(()=>{

    console.log("repeating")

},10000)                              // repeating , after 10sec we get repeating .

**This Keyword:**

let x=

{

id:1234,

name:"darshan",

comp:function(){

let z="universe"

console.log(z)

}

}

console.log(x.comp())              // universe

let y=

{

id:1234,

name:"darshan",

comp:function(){

return this.id

}

}

console.log(y.comp())           // if we use normal function , we will get 1234

id=9353

let z=

{

id:1234,

name:"darshan",

comp:()=>{

return this.id

}

}

console.log(z.comp())

10-12-22 absent

11-12-22

Query selectors:

In body in html

    <h1 id="nan">bangalore</h1>

    <h2 class="nan">managlore</h2>

    <h3>preritha</h3>

    <script src="./web.js"></script>

In js file

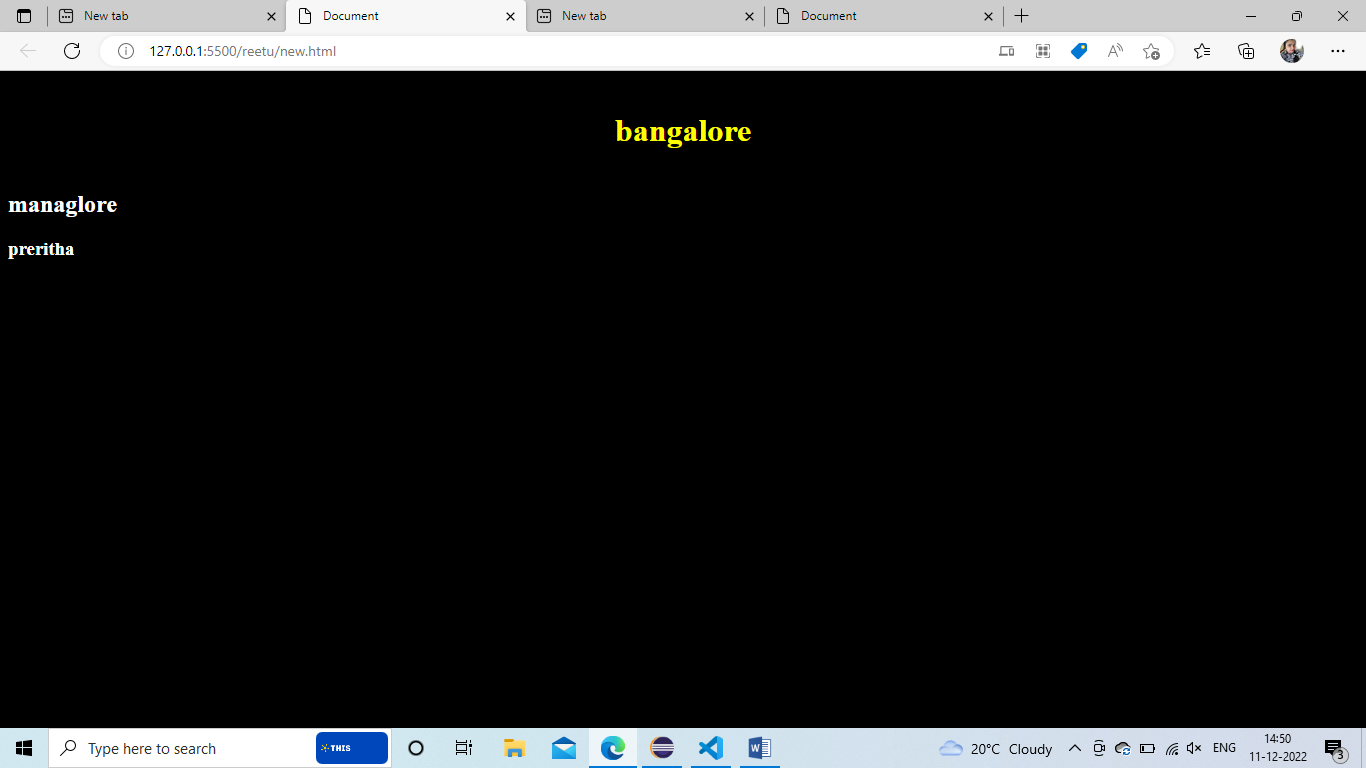
let x=document.querySelectorAll(".man,#nan,h3" );

console.log(x);

document.body.style.backgroundColor="black";

document.body.style.color="white";

x[0].style.color="yellow";



**Window ad Document:**

**Window:**

* It is a route level element in any web pages
* By default window object I available in the implicitly in the page
* It has methods like alert, confirm, prompt .
* It has properties like DOM, location

**Document:**

* It is the direct child of window object
* Also known as DOM
* We can access it via window.document or document.
* It has methods like getElementID() , getElementByClassame() , queryselector(),

**BOM**

**Browser object modules**

* It allows JS to talk to the browsers
* It consists of object navigator , history , screen,location and DOM which are childrens of the window
* Bom is not standrardized and can be changed based of different browser

DOM Events:

* Onclick
* OnSubmit

Onclick

 <!-- <button id="nan" onclick="max">click</button> -->

    <!-- <button id="nan" onclick="click()">click</button> -->

    <!-- <button id="man" onclick="pre()">click</button> -->

In js file

let btn=document.getElementById("nan");

console.log(btn);

function max()

{

    document.body.style.backgroundColor="black";

}

let btn=document.getElementById("nan");

btn.addEventListener('click',()=>

{

    document.body.style.backgroundColor="yellow";

})

let btn=document.getElementById("nan");

let man=document.getElementById("man");

btn.addEventListener('click',()=>

{

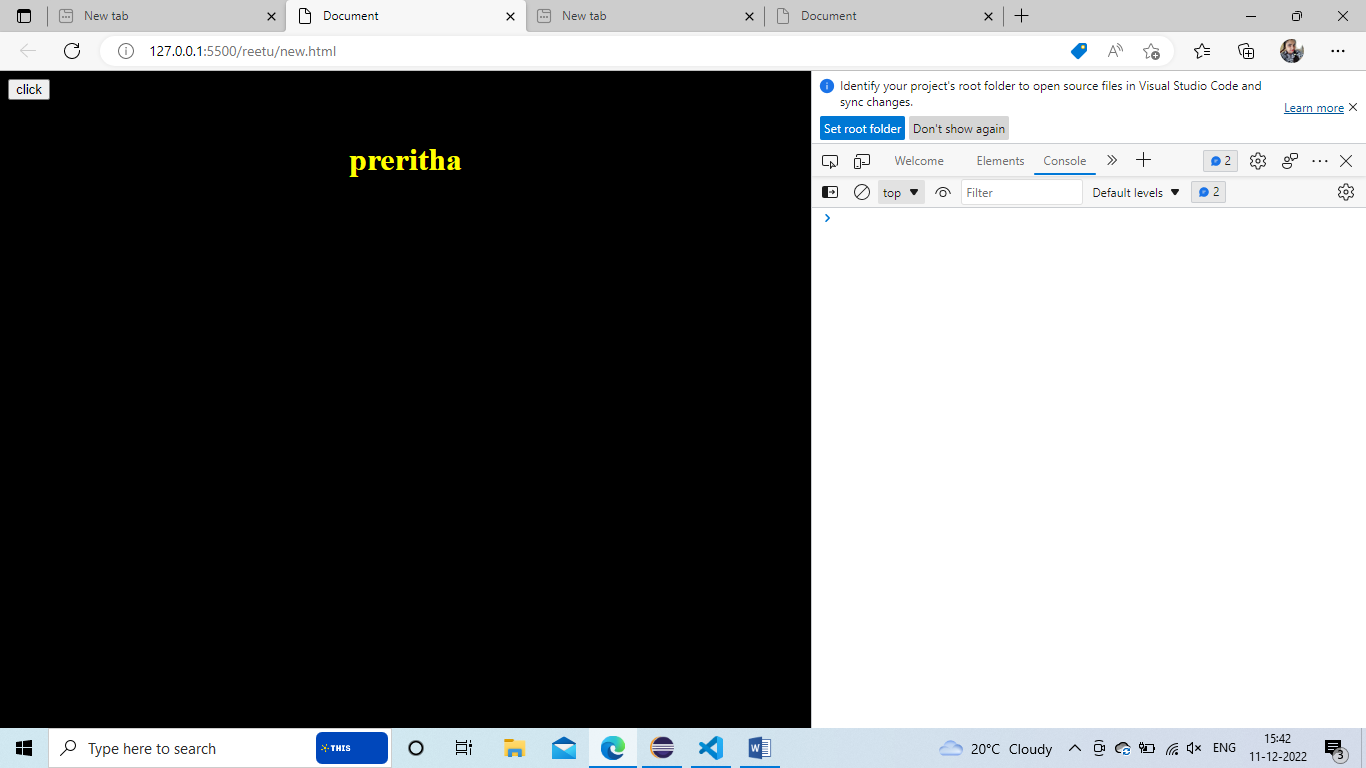
    document.body.style.backgroundColor="black";

    man.innerText="preritha"

    man.style.color="yellow"

    man.style.border="px solid yellow"

})



**Event Propogations:**

**What is an Event Flow:**

**Event Flow** is an order in which event is received on the web page .When we click an element that is nested in various other elements before u click actually reaches its destination.

It must trigger the click event for each of its parent element

Starting at the top with global window object

There are two ways of event flow

* **Bubbling:** bubbling takes place -> bottom to top
* **Capturing:** capturing of function-> -> top to bottom

**What is Event Bubbling?**

It is a type of event propagation where the event first trigger on the innermost target element and then successfully triggers on the ancestors of the target element in the same nesting hierarchy till it reaches the outermost DOM element.

**What is Event capturing?**

It is a type of event propagation where the event first captured by outermost element and then successfully triggers on the descendants of the target element in the same nesting hierarchy till it reaches the innermost most DOM element.

**Bubling:**

In html file

* <div id="grand" style="border: 10px solid red; height: 200px; width: 200px; padding: 200px;" > grand
* <div id="parent" style="border: 8px solid rgb(0, 255, 13); height: 100px;  width: 10px; padding: 100px;">parent
* <div id="child" style="border: 6px solid rgb(212, 196, 52); height: 50px;  width: 50px;padding: 1px;">child
* </div>
* </div>
* </div>

In JS file

let grand=document.getElementById("grand");

let parent=document.getElementById("parent");

let child=document.getElementById("child");

grand.addEventListener('click',()=>{

    console.log("grand clicked");

})

parent.addEventListener('click',()=>{

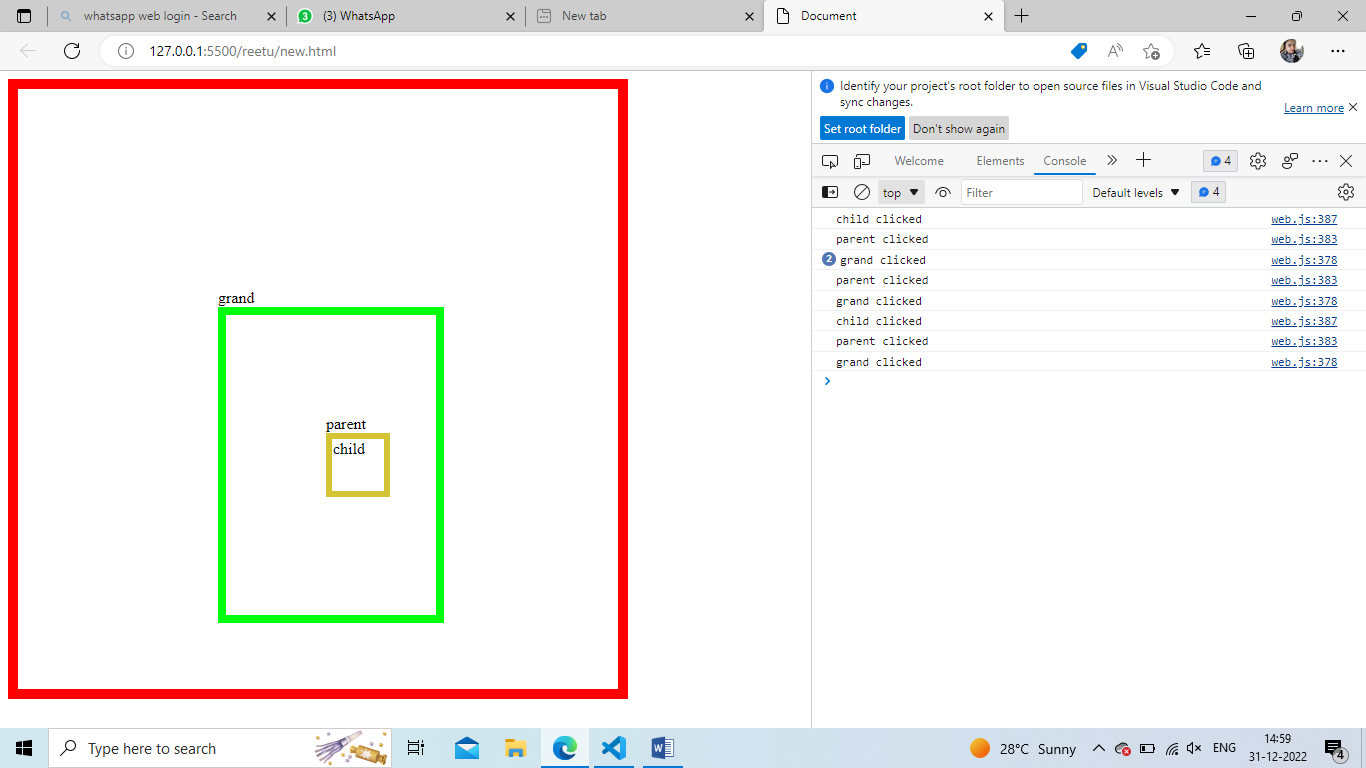
    console.log("parent clicked");

})

child.addEventListener('click',()=>{

    console.log("child clicked");

})



Capturing :

In html file

* <div id="grand" style="border: 10px solid red; height: 200px; width: 200px; padding: 200px;" > grand
* <div id="parent" style="border: 8px solid rgb(0, 255, 13); height: 100px;  width: 10px; padding: 100px;">parent
* <div id="child" style="border: 6px solid rgb(212, 196, 52); height: 50px;  width: 50px;padding: 1px;">child
* </div>
* </div>
* </div>

In JS file

let grand=document.getElementById("grand");

let parent=document.getElementById("parent");

let child=document.getElementById("child");

grand.addEventListener('click',()=>{

    console.log("grand clicked");

},true)

parent.addEventListener('click',()=>{

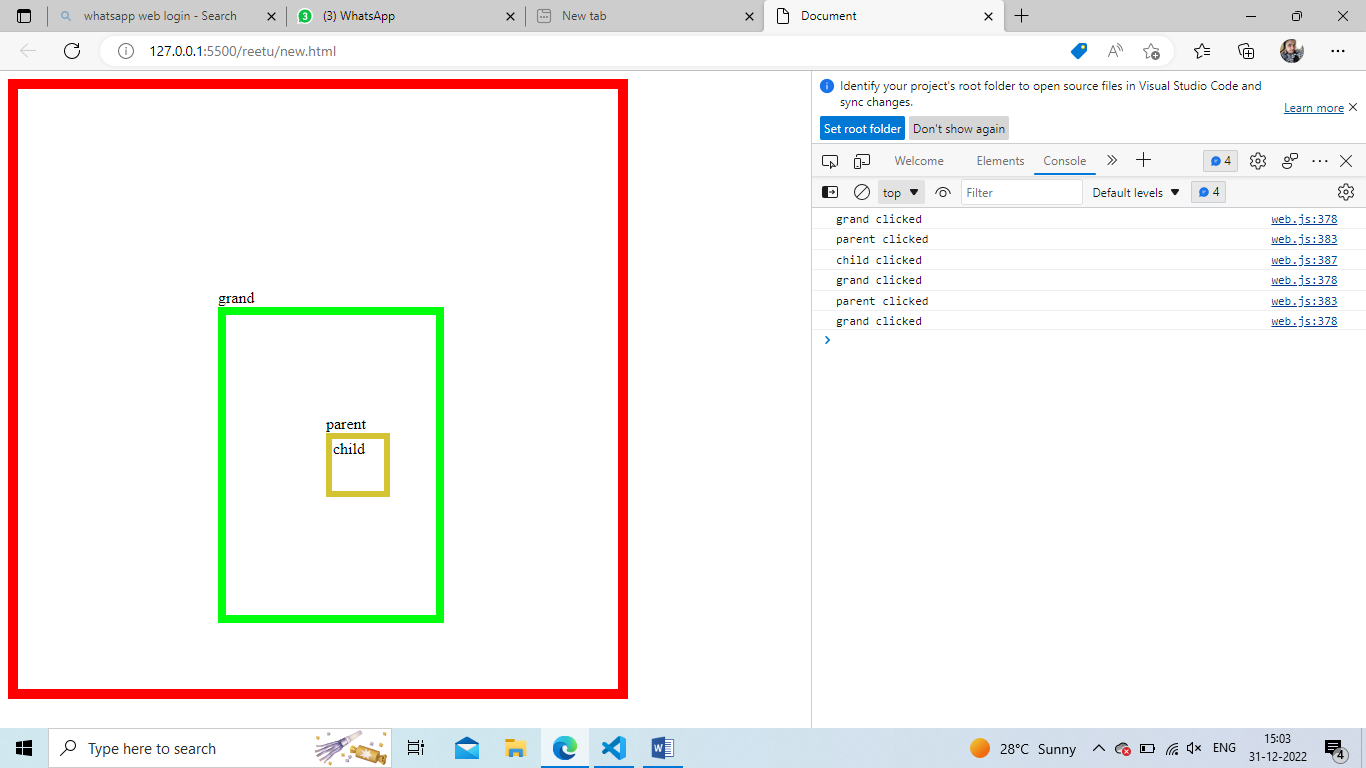
    console.log("parent clicked");

},true)

child.addEventListener('click',()=>{

    console.log("child clicked");

},true)



==========================

WE need like below output

In html file

* <div id="grand" style="border: 10px solid red; height: 200px; width: 200px; padding: 200px;" > grand
* <div id="parent" style="border: 8px solid rgb(0, 255, 13); height: 100px;  width: 10px; padding: 100px;">parent
* <div id="child" style="border: 6px solid rgb(212, 196, 52); height: 50px;  width: 50px;padding: 1px;">child
* </div>
* </div>
* </div>

In JS file

let grand=document.getElementById("grand");

let parent=document.getElementById("parent");

let child=document.getElementById("child");

grand.addEventListener('click',(e)=>{

    e.stopPropagation()

    console.log("grand clicked");

})

parent.addEventListener('click',(e)=>{

    e.stopPropagation()

    console.log("parent clicked");

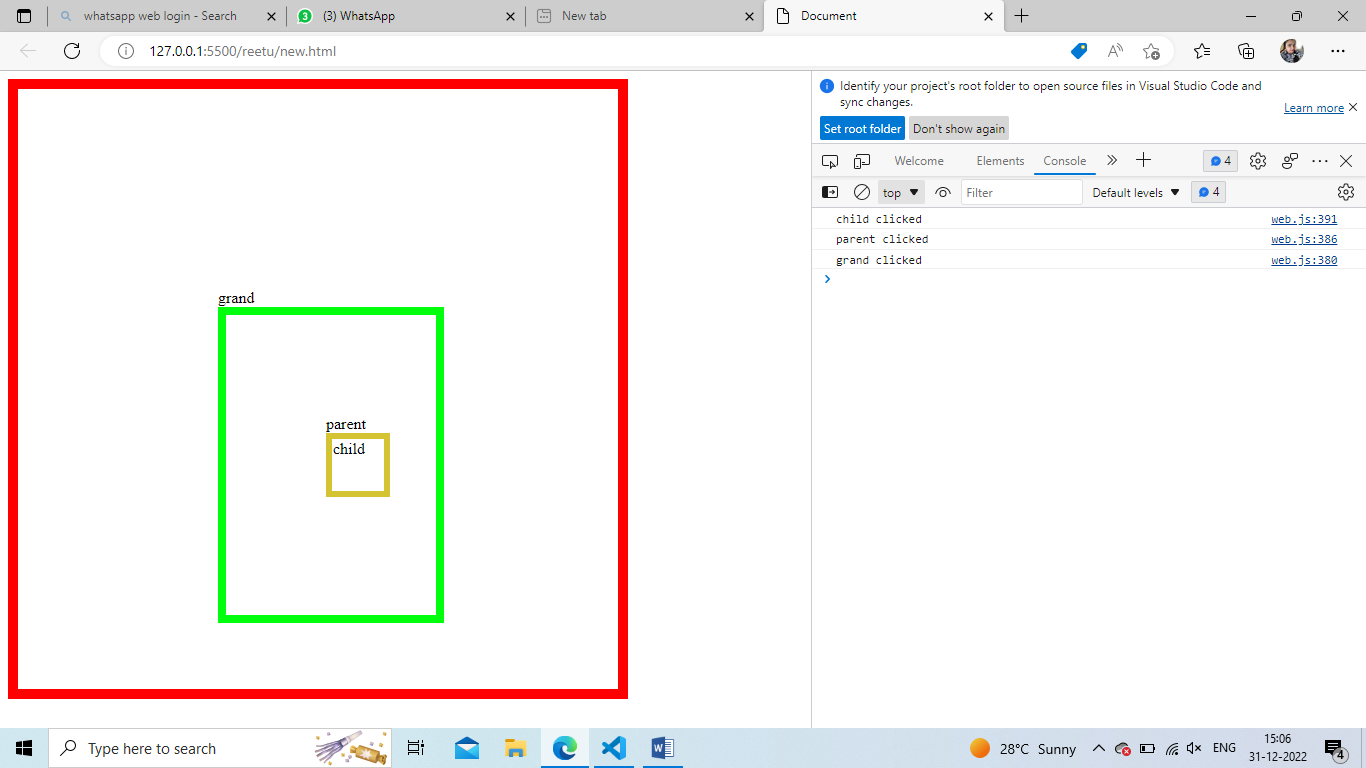
})

child.addEventListener('click',(e)=>{

    e.stopPropagation()

    console.log("child clicked");

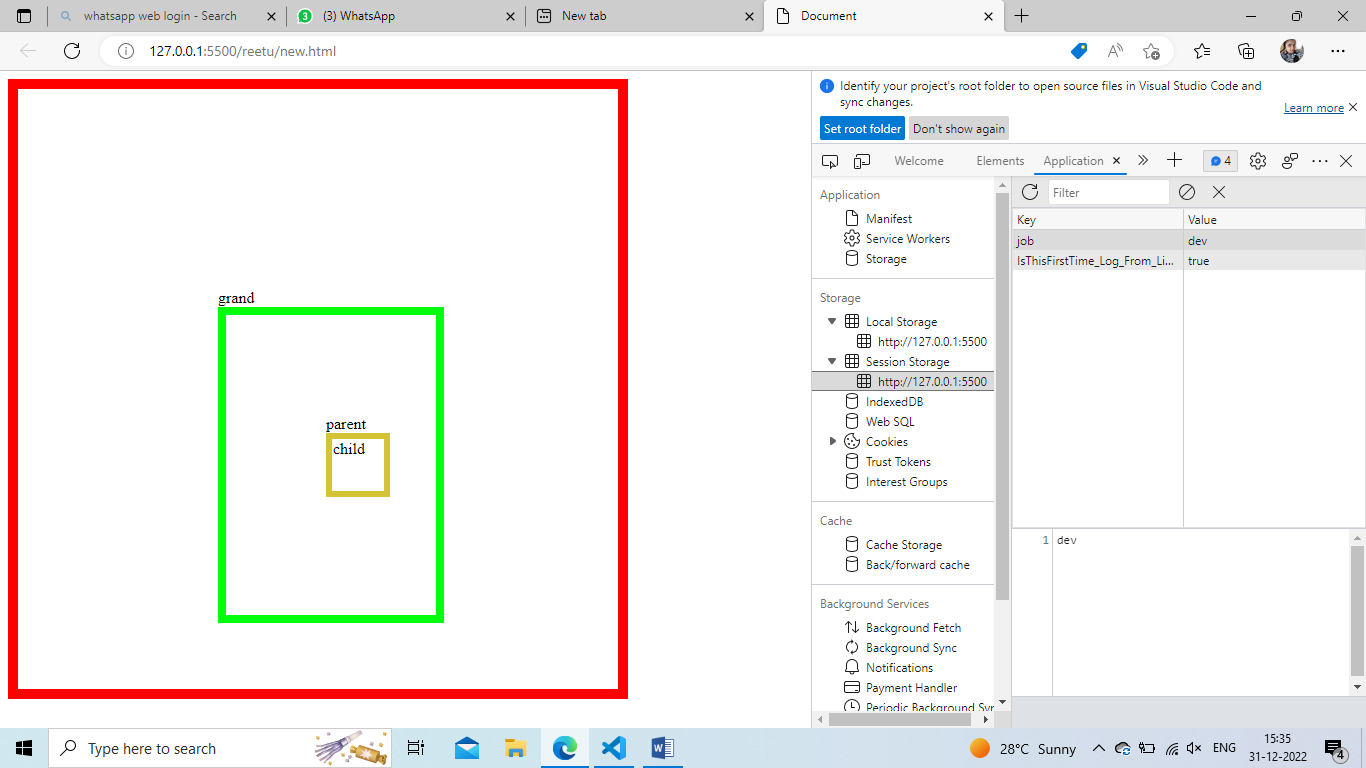
})



**Session Storage:** storage is temporary , storage is 5MB. , In this data is erased when close or refresh the tab or page

let userStoragee=window.sessionStorage.setItem("job","dev")

console.log(userStoragee);



**Local Storage:** storage is permanent, storage is 5MB , In this data is stored even if we close or refresh the tab or page

let userStorage=window.localStorage.setItem("name","pre")

console.log(userStorage);

let userStorage1=window.localStorage.setItem("class","10")

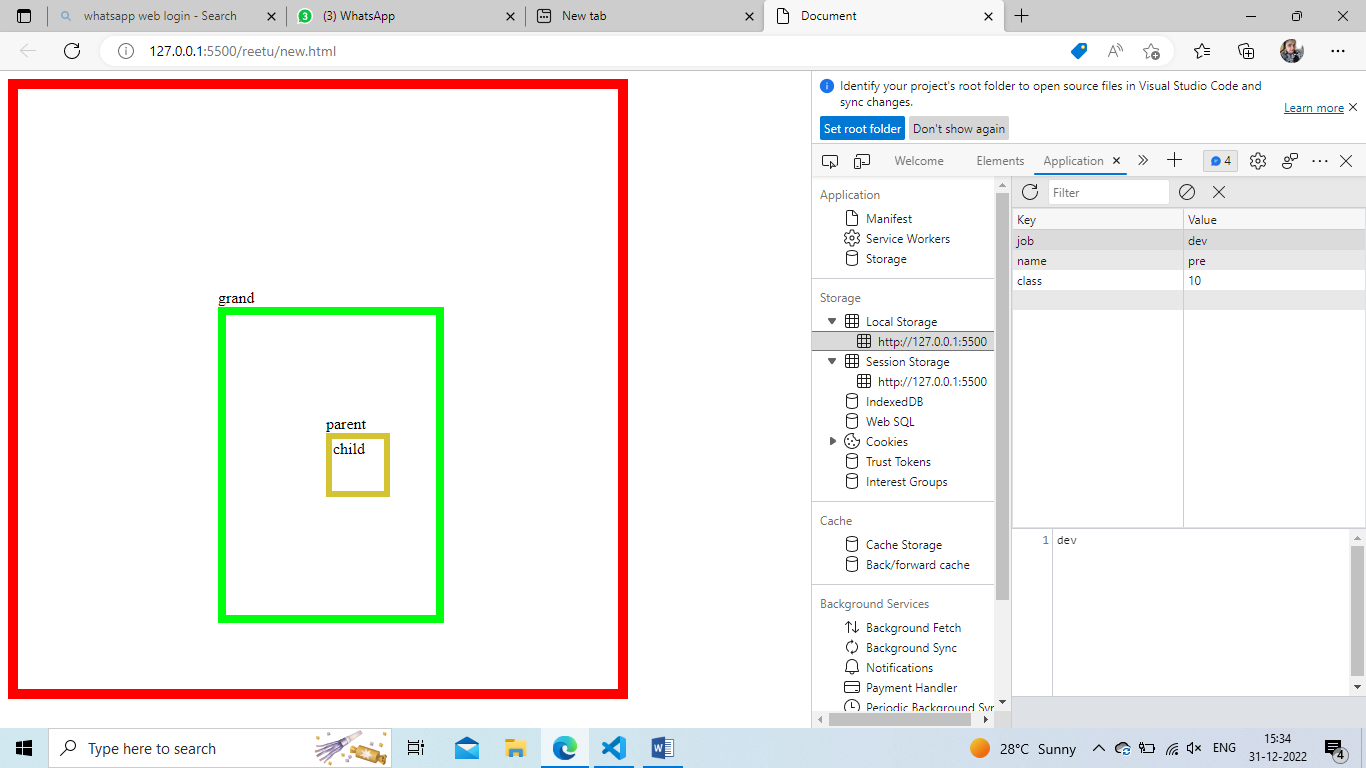
console.log(userStorage1);

let userStorage2=window.localStorage.setItem("job","dev")

console.log(userStorage1);

let userStorage3=window.localStorage.getItem("job","dev")

console.log(userStorage3);



In order to clear the key and value pairs :

window.localStorage.clear()

window.sessionStorage.clear()

**What is diff btw local and session storage :**

Local storage: is same as session storage but it persist the data even when the browser is closed and reopened where as in Session storage data gets cleared when the page session ends .

**7/01/23 Saturday**

**Promises:** to make asynchronous to synchronous :

I will give only 1 output

* Then
* Catch
* Finally

let x=new Promise((resolve,reject)=>{

    let roomcleaned=true

    if(roomcleaned){

        resolve("yes room is cleaned")}

    else{

        reject("no not cleaned")

    }

})

console.log(x);

x.then((data)=>{

    console.log(data);

})

.catch((err)=>{

    console.log(err);

})

.finally(()=>{

    console.log("yes task done");

})

Output:

yes room is cleaned

yes task done

let z=new Promise((x,y)=>{

    if(Math.random()>0.2){

        x("mail sent")}

    else{

        y("mail not sent")

    }

})

console.log(z);

z.then((data)=>{

    console.log(data);

})

.catch((err)=>{

    console.log(err);

})

.finally(()=>{

    console.log("yes task done");

})

*Promise {<fulfilled>: 'mail sent'}*

* 1. [[Prototype]]: Promise
  2. [[PromiseState]]: "fulfilled"
  3. [[PromiseResult]]: "mail sent"

mail sent

yes task done

let z=new Promise((x,y)=>{

   setTimeout(()=>{

    x("i am first");

   },2000)

   setTimeout(()=>{

    y("i am last");

   },1000)

})

console.log(z);

z.then((data)=>{

    console.log(data);

})

.catch((err)=>{

    console.log(err);

})

.finally(()=>{

    console.log("yes task done");

})

Output:

i am last yes task done

function q(){

setTimeout(()=>{

    console.log(x);

    console.log(y);

},3000)

let x=2;

let y=3;

}

q();

**after 3sec we get**

**2**

**3**

let count=10;

(function innerfunction(){

    if(count===10){

        let count=11;

        console.log(count);

    }

    console.log(count);

    }

)

();

**11**

**10**

function add(item ,items=[]){

    items.push(item);

    return items;

}

console.log(add("orange, apple"));

console.log(add("apple"));

**['orange, apple']**

**['apple']**

function add(items=[]){

    return items;

}

console.log(add("orange, apple"));

console.log(add("apple"));

**orange, apple**

**apple**

console.log(typeof typeof typeof true)

**string**

**end of web**