**HTML:**

<!DOCTYPE html>

<html lang="en">

<head><!--contains website information-->

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body><!--contains content of the website-->

    <h1>Hello!</h1>

    <h2>Hello!</h2>

    <h3>Hello!</h3>

    <h4>Hello!</h4>

    <h5>Hello!</h5>

    <h6>Hello!</h6>

    <p>Hello! My name is Deven Malla.</p>

    <p>Lorem ipsum dolor sit amet consectetur adipisicing elit. <br>Nulla nam aperiam quam magni, debitis incidunt perspiciatis. <br>Cum a maxime iste.</p>

    <hr>

    <ol>

        <li>Red</li>

        <li>Blue</li>

        <li>Green</li>

    </ol>

    <ul>

        <li>Red</li>

        <li>Blue</li>

        <li>Green</li>

    </ul>

    <img src="nagaland-university-logo.png" alt="Nagaland University logo">

    <br>

    <a href="https://www.youtube.com/watch?v=kkOuRJ69BRY&list=PLSUlCIUmG02WfnUvc4hWPrJPvtH2mn37o">Learn HTML, CSS and JavaScript in Single Video</a>

    <br><br>

    <form><!--container-->

        Enter name:<input type="text" placeholder="enter your name"><br>

        Enter password:<input type="password" placeholder="enter your password"><br>

        Enter date:<input type="date"><br>

        Select gender:<input type="checkbox">Male<input type="checkbox">Female<br>

        Select gender:<input type="radio">Male<input type="radio">Female<br>

        Select colour:<input type="color">

    </form>

    <video autoplay loop muted controls src=""></video>

    <div><!--used as a container to create divisions-->

        <h1>Hello!</h1>

    </div>

    <!--semantics in html:-->

    <!--<main>

    </main>

    <header>

    </header>

    <nav>

    </nav>

    <article>

    </article>

    <aside>

    </aside>-->

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <link rel="stylesheet" href="../CSS/style\_01.css">

</head>

<body>

    <!--<h1>Hello!</h1>-->

    <!--<div>

        <h1>Hello!</h1>

        <h2>Hello!</h2>

        <h3>Hello!</h3>

        <h4>Hello!</h4>

        <h5>Hello!</h5>

        <h6>Hello!</h6>

    </div>-->

    <!--<div>

    </div>-->

    <!--<h1 id="real">real h1</h1>

    <h1>fake h1</h1>

    <h1>rf h1</h1>-->

    <!--<h1>Hello!</h1>

    <h2 class="a">Hey!</h2>

    <h6>Howdy!</h6>

    <h1 class="a">Yo!</h1>

    <p class="a">Lorem ipsum dolor sit amet consectetur adipisicing elit. Voluptatibus, delectus!</p>-->

    <!--<div id="box">

        <h1>Hello!</h1>

    </div>-->

    <!--<div id="parent">

        <div id="child">

        </div>

    </div>-->

    <!--<button>Download</button>-->

    <!--<div id="box1">

    </div>

    <div id="box2">

    </div>-->

    <!--<h1>Hello! I am <br><span>Deven</span></h1>-->

    <div id="parent">

        <div id="child1" class="child"></div>

        <div id="child2" class="child"></div>

        <div id="child3" class="child"></div>

    </div>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <link rel="stylesheet" href="../CSS/style\_02.css">

    <link

    href="https://cdn.jsdelivr.net/npm/remixicon@4.5.0/fonts/remixicon.css"

    rel="stylesheet"/>

</head>

<body>

    <main>

        <nav>

            <h2>The Green One</h2>

            <div class="part2">

                <a href="#">About Us</a>

                <a href="#">Services</a>

                <a href="#">Projects</a>

                <a href="#">Let's talk</a>

            </div>

        </nav>

        <section>

            <div class="hero-text1">

                <h1>Digitize</h1>

                <div class="video">

                    <div class="play">

                        <img height="32" width="32" src="play-circle-fill.svg" />

                    </div>

                </div>

            </div>

            <div class="hero-text2">

                <h1>Ideas</h1>

                <p>The art of visual communication, creatively<br>

                impacting the world around us-one good<br>

                design at a time, design like you mean it!<br>

            </p>

            </div>

        </section>

        <div class="image">

            <div class="arrow">

                <i class="ri-arrow-right-up-line"></i>

            </div>

        </div>

    </main>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <link rel="stylesheet" href="../CSS/style\_03.css">

</head>

<body>

    <h1>Study JS friends!!!</h1>

    <script src="../JavaScript/script\_01.js"></script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

    <link rel="stylesheet" href="../CSS/style\_04.css">

</head>

<body>

    <!--<h1>this is dom</h1>-->

    <div id="bulb"></div>

    <button>on</button>

    <!--<div id="box1">

        <p>Lorem ipsum dolor sit amet.</p>

    </div>

    <div class="box2">

        <p>Lorem ipsum dolor sit amet.</p>

    </div>

    <h1>hello1</h1>

    <h1>hello2</h1>

    <h1>hello2</h1>-->

    <!--<h1>hello</h1>-->

    <!--<div id="box">

        <h1>hello</h1>

    </div>-->

    <script src="../JavaScript/script\_01.js"></script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>Document</title>

</head>

<body>

    <script src="../JavaScript/script\_02.js"></script>

</body>

</html>

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <script src="../JavaScript/script\_03.js"></script>

    <title>Document</title>

</head>

<body>

</body>

</html>

**CSS:**

/\*cascading style sheets\*/

\*{

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

html, body{

    height: 100%;

    width: 100%;

}

/\*body{

    background-color: black;

}\*/

/\*h1{

    color: royalblue;

    background-color: black;

    font-size: 50px;

    font-family: sans-serif;

    font-weight: 100;

    font-style: italic;

    text-align: center;

}\*/

/\*div{

    color: rebeccapurple;

}\*/

/\*div{

    height: 300px;

    width: 300px;

    background-color: crimson;

}\*/

/\*h1{

    color: red;

}\*/

/\*#real{

    background-color: blue;

}\*/

/\*.a{

    background-color: royalblue;

}\*/

/\*#box{

    height: 200px;

    width: 200px;

    background-color: crimson;

    margin-left: 100px;

    margin-top: 50px;

    padding-left: 20px;

    padding-top: 20px;

}\*/

/\*#parent{

    height: 200px;

    width: 200px;

    background-color: crimson;

}

#child{

    height: 50%;

    width: 50%;

    background-color: royalblue;

}\*/

/\*button{

    margin: 40px;

    font-size: 20px;

    padding: 8px;

    background-color: lightgreen;

    color: white;

    font-weight: 300px;

    /\*border: 5px solid red;\*/

    /\*border: 5px dashed red;\*/

    /\*border: 5px dotted red;

    border: none;

    border-radius: 10px;

}

button:hover{

    background-color: grey;

    color: black;

}\*/

/\*#box1{

    height: 200px;

    width: 200px;

    background-color: crimson;

    border: 2px solid white;

    margin-top: 20px;

    position: absolute;

    /\*left: 50%;

    top: 30%;\*/

    /\*right: 0;\*/

    /\*left: 0;

    bottom: 0;\*/

    /\*right: 0;

    top: 0;

    left: 50%;

    top: 50%;

    transform: translate(-50%, -50%); /\*to center the box

    z-index: 9;/\*box with higher z index is displayed first

}

#box2{

    height: 200px;

    width: 200px;

    background-color: royalblue;

    border: 2px solid white;

    top: 50%;

    left: 50%;

    position: absolute;

    z-index: 10;

}\*/

/\*h1{

    position: absolute;

    top: 50%;

    left: 50%;

    transform: translate(-50%, -50%);

    text-align: center;

    font-size: 60px;

    font-weight: 200;

}

h1 span{

    background-color: red;

    font-size: 55px;

    font-weight: 200;

    background-color: lightseagreen;

    padding: 10px 30px;

    color: white;

    border-radius: 10px;

}\*/

#parent{

    height: 100%;

    width: 100%;

    background-color: black;

    display: flex;/\*divides in the form of rows and columns\*/

    /\*align-items: flex-start;\*/

    /\*align-items: flex-end;\*/

    align-items: center;

    justify-content: center;

    /\*justify-content: flex-end;

    justify-content: flex-start;\*/

    /\*justify-content: space-between;\*/

    justify-content: space-around;

    /\*justify-content: space-evenly;\*/

}

.child{

    height: 200px;

    width: 200px;

    background-color: crimson;

    border: 3px solid white;

    background-image: url(https://plus.unsplash.com/premium\_photo-1675337267945-3b2fff5344a0?q=80&w=764&auto=format&fit=crop&ixlib=rb-4.1.0&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D);

    background-size: cover;

    background-position: center;

}

#child1{

    background-image: url(https://images.unsplash.com/photo-1743445888873-7b989699663d?w=1000&auto=format&fit=crop&q=60&ixlib=rb-4.1.0&ixid=M3wxMjA3fDB8MHxmZWF0dXJlZC1waG90b3MtZmVlZHwzfHx8ZW58MHx8fHx8);

    background-size: cover;

    background-position: center;

}

#child3{

    background-image: url(https://images.unsplash.com/photo-1745487954749-a33270b757de?w=1000&auto=format&fit=crop&q=60&ixlib=rb-4.1.0&ixid=M3wxMjA3fDB8MHxmZWF0dXJlZC1waG90b3MtZmVlZHw3fHx8ZW58MHx8fHx8);

    background-size: cover;

    background-position: center;

}

\*{

    margin: 0;

    padding: 0;

    box-sizing: border-box;

    font-family: sans-serif;

}

html, body{

    height: 100%;

    width: 100%;

}

main{

    width: 100%;

    background-color: #d5ff40;

}

nav{

    padding: 60px 112px;

    width: 100%;

    display: flex;

    align-items: center;

    justify-content: space-between;

}

nav .part2{

    display: flex;

    align-items: center;

    justify-content: center;

    gap: 100px;

}

nav h2{

    font-size: 20px;

    font-weight: 700;

}

nav a{

    font-size: 15px;

    color: black;

    text-decoration: none;

    font-weight: 500;

}

a:hover{

    background-color: grey;

    padding: 4px;

    border-radius: 10px;

    font-size: 15px;

    color: black;

    text-decoration: none;

    font-weight: 500;

}

section{

    /\*background-color: red;\*/

    padding: 15px 100px;

}

section .hero-text1{

    display: flex;

    align-items: flex-start;

    justify-content: space-between;

}

.hero-text1 h1{

    font-size: 180px;

    text-transform: uppercase;

    font-weight: 900;

    line-height: 200px;

}

.hero-text1 .video{

    height: 80px;

    width: 300px;

    border-radius: 50px;

    margin-top: 50px;

    background-image: url(https://plus.unsplash.com/premium\_photo-1670652222544-18de4c8a4f6f?q=80&w=1032&auto=format&fit=crop&ixlib=rb-4.1.0&ixid=M3wxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8fA%3D%3D);

    background-size: cover;

    background-position: center;

    padding: 5px;

}

.video .play{

    background-color: white;

    height: 100%;

    width: 70px;

    border-radius: 50px;

    border: 7px solid grey;

    display: flex;

    align-items: center;

    justify-content: center;

}

.play img{

    font-size: 50px;

    background-color: white;

}

section .hero-text2{

    display: flex;

    align-items: flex-end;

    justify-content: space-between;

}

.hero-text2 h1{

    font-size: 180px;

    text-transform: uppercase;

    font-weight: 900;

    line-height: 2px;

}

.hero-text2 p{

    font-size: 15px;

    text-transform: uppercase;

    width: 400px;

    font-weight: 500px;

}

.image{

    margin-top: 10%;

    height: 500px;

    width: 90%;

    margin-left: 5%;

    background-color: red;

    border-top-right-radius: 100px;

    border-top-left-radius: 100px;

    background-image: url(https://images.pexels.com/photos/5011647/pexels-photo-5011647.jpeg);

    background-position: cover;

    background-position: center;

    position: relative;

}

.image .arrow{

    height: 250px;

    width: 250px;

    border-radius: 50%;

    background-color: black;

    position: absolute;

    right: 5%;

    top: -10%;

    color: white;

    display: flex;

    align-items: center;

    border: 10px solid grey;

}

.arrow i{

    font-size: 220px;

}

\*{

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

html, body{

    height: 100%;

    width: 100%;

}

body{

    background-color: black;

    color: white;

    font-family: sans-serif;

}

\*{

    margin: 0;

    padding: 0;

    box-sizing: border-box;

}

html, body{

    height: 100%;

    width: 100%;

}

/\*body{

    background-color: black;

    color: aliceblue;

}\*/

#bulb{

    height: 300px;

    width: 300px;

    border-radius: 50%;

    border: 4px solid black;

}

button{

    padding: 10px 20px;

    font-size: 80px;

    margin: 50px;

}

/\*#box{

    height: 300px;

    width: 300px;

    background-color: salmon;

}\*/

**JavaScript:**

//types of console and print statements

//console.log("hello")

//console.warn("this is warning")

//console.error("this is error")

//alert("this is alert") //shows a popup in the website

//confirm("are you an adult?") //asks the users yes or no

//prompt("enter your name") //to ask for user input

//variables

// to create a variable in JS we use "var"

//var a //declaration

//a = 10 //assigning a value to the variable

//var a = 10

//a = 20

//var a = "s"

//var a = true

//var a = confirm("okay?")

//console.log(a) //true

//var ans = prompt("enter your name") //to ask for user input

//console.log(ans)

/\*alert("message")

confirm("confirmed or not?")

prompt("enter you name")\*/

//var username = prompt("enter your name")

//console.log("username is", username)

//arithmetic operations in JS

/\*var a = 10

var b = 20

console.log(a+b)

console.log(a-b)

console.log(a\*b)

console.log(a/b)

console.log(a%b) //remainder/modulus\*/

//comparison operator

/\*var a = 10

var b = 10 //correct

if(a==b)

{

    console.log("correct")

}

else

{

    console.log("incorrect")

}\*/

/\*var a = 10

var b = '10' //correct

if(a==b) //"==" compares only values

{

    console.log("correct")

}

else

{

    console.log("incorrect")

}\*/

/\*var a = 10

var b = '10'

if(a===b) //"===" compares values and types

{

    console.log("correct")

}

else

{

    console.log("incorrect")

}\*/

//dataypes in JS

//-primitive:

  //-number (1, 2, 3, 20.5, 6.66666)

  //-string ('a', 'john')

  //-boolean (true, false)

  //-undefined (no value)

  //-null (nothing)

  //-symbol

  //-NaN (not a number)

//-reference:

  //-Array

  //-Object

  //-Function

  /\*var a = 10

  console.log(a)

  var a = "string"

  console.log(a)

  var a = true

  console.log(a)

  var a

  console.log(a)

  var a = null

  var a = 10

  var b = "string"

  console.log(a\*b) //NaN

  console.log(a+b) //10string (concatenation)

  //"+" or "addition" is not only used for addition, but for concatenation\*/

//conditionals in JS

//if else

/\*if(10>5){

    console.log("correct")

}else{

    console.log("not correct")

}\*/

/\*var a = 10

if(a>5)

{

    console.log("yes")

}

else

{

    console.log("no")

}\*/

/\*var age = prompt("enter your age")

if(age>18)

{

    console.log("you can vote")

}

else

{

    console.log("you cannot vote")

}\*/

//while loop

/\*var a = 0

while(a<5)

{

    console.log("run")

    a++

}\*/

/\*var a = 10

while(a > 0)

{

    console.log(a)

    a--

}\*/

//for loop

/\*for(var a = 0; a<5; a++)

{

    console.log("hello")

}\*/

/\*for(var a = 0; a<5; a++)

{

    console.log(a)

}\*/

//functions in JS

/\*function hey()

{

    //function body

    console.log("good morning")

}

//function call

hey()\*/

/\*function walk()

{

    console.log("keep walking")

}

function sing()

{

    console.log("keep singing")

}

function dance()

{

    console.log("keep dancing")

}

sing()

walk()

dance()

function greet(a)

{

    console.log("good morning", a)

}

greet("deven")

greet(10)

greet(6.57)

function abc()

{

    console.log("hello")

    return 20

}

var a = abc() //20 goes inside function abc()

console.log(a)

var abc = function abc() //this is known as first class function

{

    console.log("hello")

}

abc()

var abc = ()=> //fat arrow function

{

    console.log("hello") //hello

}

abc()

//var is a global varibale

var a = 10

if(10>5)

{

    var b = 20

    a = 30

}

console.log(a)

console.log(b)

//var:

  //-let (similar to var)

  //-const (we use const when we want to initialise only once)

  var groom = "john"

  var bride = "tia"

  groom = "shane"

  console.log(groom, "weds", bride) //shane weds tia

  const groom = "john"

  const bride = "tia"

  groom = "shane"

  console.log(groom, "weds", bride) //Uncaught SyntaxError: Identifier 'groom' has already been declared (at script.js:254:9)\*/

//arrays in JS

/\*var a = 10

console.log(a) //in primitive datatype we cannot store more than one values\*/

/\*var a = [10, 20, 30, 40]

console.log(a)\*/

/\*var arr = [10, 'john', 30.66, true, 80]

console.log(arr) //[10, 'john', 30.66, true, 80]\*/

/\*var arr = [10, 'john', 30.66, true, 80]

console.log(arr[2]) //30.66\*/

/\*var arr = [10, 20, 30, 40]

console.log(arr) //[10, 20, 30, 40]

arr.push(99)

console.log(arr) //[10, 20, 30, 40, 99]

arr.pop()

console.log(arr) //[10, 20, 30, 40]

console.log(arr.length)\*/

/\*function abc(a)

{

    console.log("hello", a)

}

abc(10)\*/

/\*var arr = [10, 20, 30, 40, 50, 60]

arr.forEach(function(elem) //automatically calls the function based on the size of the array

{

    console.log('hey', elem) //hey 10, hey 20, ..., hey 60

})\*/

//objects in JS

//"()"-functions

//"{}"-objects

//"[]"-array

/\*var obj = {

    user: "john",

    age: 20

}

console.log(obj)\*/

/\*var obj = {

    user: "john",

    age: 20

}

console.log(obj.user) //john\*/

/\*var obj1 = {

    model: "iphone",

    price: 70000,

    color: 'white'

}

var obj2 = {

    model: 'oneplus',

    price: 30,

    color: "blue"

}

var obj3 = {

    model: '50000',

    price: 50000,

    color: "black"

}

console.log(obj1)

console.log(obj2)

console.log(obj3)

console.log(obj2.color)\*/

//function inside an object is a method

/\*var user = {

    userName: 'john',

    age: 20,

    greet: function() //this is a method

    {

        console.log("good morning")

        return 10

    }

}

console.log(user.userName)

console.log(user.age)

console.log(user.greet()) //good morning, undefined //good morning, 10\*/

//array objects

/\*var arr = [

    {name: "john", age: 22},

    {name: "tia", age: 24},

    {name: "shane", age: 21},

    40

]

console.log(arr[0]) //{name: 'john', age: 22}

console.log(arr[1]) //{name: 'tia', age: 24}

console.log(arr[2].name) //shane\*/

/\*//dom (document object model)

//frontend's JS is dom

//dom has 4 pillars:

  //-selection of an element

  //-changing HTML

  //-changing CSS

  //-event listener

  //selection of an element

    var a = document.querySelector("h1")

    console.log(a)

  //changing HTML

    var a = document.querySelector("h1")

    a.innerHTML = "hi"

  //changing CSS

    var a = document.querySelector("h1")

    a.style.color = "brown"

    a.style.backgroundColor = "royalblue"

  //event listener

    //something which is happen

    //listener is the one who understands and listens to the events

    var a = document.querySelector("h1")

    a.addEventListener("click", function() //click is an event

{

    a.innerHTML = "changed"

    a.style.color = "black"

    console.log("hello")

})

    //mouseenter

    //dblclick\*/

//dom

  //in dom we write in the form of camel case

  //var a = document .querySelector("h1") //selecting h1 from HTML

  //console.log(a)

  //(".box") for selecting class

  //("#box") for selecting id

  //var a = document .querySelector("h1").innerHTML = "okay"

  //a.innerHTML = "okay"

  //var a = document .querySelector("h1")

  //a.innerHTML = "okay"

  //a.style.color = "red"

  //a.style.backgroundColor = "grey"

  /\*var a = document .querySelector("h1")

  a.style.color = "red"

  a.style.backgroundColor = "grey"

  a.addEventListener("click", function()

  {

    a.innerHTML = "okay"

    a.style.color = "yellow"

    a.style.backgroundColor = "royalblue"

  })\*/

  var bulb = document.querySelector("#bulb")

  var btn = document.querySelector("button")

  var flag = 0

  btn.addEventListener("click", function()

  {

    if(flag == 0)

        {

            bulb.style.backgroundColor = ("yellow")

            console.log("clicked")

            flag = 1

            btn.innerHTML = "on"

        }

        else

        {

            bulb.style.backgroundColor = ("transparent")

            console.log("clicked again")

            flag = 0

            btn.innerHTML = "off"

        }

  })

  //selecting multiple events at a same time

  /\*var h = document.querySelectorAll("h1")

  console.log(h) //NodeList(3) [h1, h1, h1]

  console.log(h[2]) //<h1>hello2</h1>

  h.forEach(function(e)

    {

        console.log(e)

        /\*

        h1

        h1

        h1

        h1

        }

  )\*/

  /\*var bx = document.getElementById("box1")

  console.log(bx)

  var cl = document.getElementsByClassName("box2")

  console.log(cl)\*/

  /\*var h = document.querySelector("h1")

  h.textContent = "bye"\*/

  /\*var box = document.querySelector("#box")

  box.textContent = "<h1>hello</h1>" //<h1>hello</h1>\*/

//JS is not asynchronous

//synchronous JS

  //until and unless one task is finished, the next task is not performed, line by line, one by one

  //synchronous JS:

    /\*console.log("string1")

    console.log("string2")

    console.log("string3")

    console.log("string4")\*/

//asynchronous JS

  //when multiple tasks are performed at the same time, and the task that is executed first and replies first, altogether

  //asynchronous JS:

    /\*setTimeout

    setInterval

    promises

    fetch

    axios

    XMLHttpRequest\*/

    //send request:

      /\*setTimeout

      setInterval

      promises

      fetch

      axios

      XMLHttpRequest\*/

    //after request is completed, we run using:

      //then catch

      //callbacks

      //async await

    //async code is used whhen we don't know how much time it will take to get the answer from the code

    //callback function

      //function is called after a specfic task is completed

      //callback function runs after async code completion

      //in setTimeout there are only two things that are used, they are: callback and timeinms

      //setTimeout(callback, timeinms)

      //the function is called back after the given time

      /\*console.log("hey1") //this is displayed first

      setTimeout(function(){

          console.log("hey2") //whatever is inside the function() will be displayed after two seconds

      }, 2000) //2000 ms = 2 seconds\*/

      //setTimeout(function(){}, 12000)

    //promises

      //two situations in promises:

        //-resolve

        //-reject

        /\*var ans = new Promise((res, rej)=>{

            if(true){

                return res()

            }

            else{

                return rej()

            }

        })

        ans

        .then(function()

            {

                console.log("resolved")

            }

        )

        .catch(function(){

            console.log("rejected")

        })\*/

        //problem: user will for ask for a number between 0-9 and if the number iss below 5, resolve, if not reject

        /\*var ans = new Promise((res, rej)=>{

            var n = Math.floor(Math.random()\*10)

            if(n<5){

                return res();

            }

            else{

                return rej();

            }

        })

        ans.then(function()

            {

                console.log("below")

            }

        )

        .catch(function(){

            console.log("above")

        })\*/

    //main stack

      //whatever is in main stack, it outputs, sync code always moves to main stack, when main stack is empty after running all the processes, the side stack is checked whether the processing is done, once the processing is done, the side stack code is brought into the main stack

    //side stack

      //in side stack processing is done, after processing is completed, it is brought in main stack and is executed

    //event loop

      //the communication and tranferring of processes between main stack and side stack

      /\*console.log("hey1") //this is sync

      console.log("hey2") //this is sync

      setTimeout(function(){

        console.log("hey3") //this is async

      }, 0)

      console.log("hey4") //this is sync

      //hey1, hey2, hey4, hey3\*/

    //single threading

      //one task at one time

    //multi threading

      //multiple tasks, at one time, aka concurrency

    //async await

      //the function where you write async code, to make that code shorter, since its an async code, we can use promises, when we get the answer we have to use .then, if we don't want to use .then we can use async await

      async function abcd()

      {

        let raw = await fetch('https://randomuser.me/api') //this is async code

        let ans = await raw.json() //this is sync code, this will run first, this code is in the main stack

        console.log(ans)

      }

      abcd()

      /\*function abcd()

      {

        fetch('https://randomuser.me/api')

        .then(function(raw){

            return raw.json()

        })

        .then(function(data){

            console.log(data)

        })

      }

      abcd()\*/

    //fetch

      //to give a call to the backend, to bring some data, after data comes, use .then, if you don't want to use .then, use await

    //concurrency

      //when two codes run together in main stack and side stack

    //parallelism

      //deals with our processor's core, when through our code we can make the different cores worrk on different tasks

    //throttling

      //we can decrease the number of calls

//for each

  //for each always takes a function

  //for each is a method which is available in array

  /\*const names = ["alice", "john", "charlie", "david", "emma"]

  names.forEach(function(value){

    //console.log(value)

    if(value === "charlie")

    {

    }

    else

    {

        console.log(value)

    }

  })\*/

  //in for each break and continue is not allowed

//map

  //we make a blank array of the elements in a given array

  //when we want to make an array using a given array

  /\*const names = ["alice", "john", "charlie", "david", "emma"]

  let newarr = names.map(function(value){

    return 1 //type newarr and you get the output //(5) [1, 1, 1, 1, 1]

  })\*/

  /\*const names = ["alice", "john", "charlie", "david", "emma"]

  let newarr = names.map(function(value){

    return value + " ji" //(5) ['alice ji', 'john ji', 'charlie ji', 'david ji', 'emma ji']

  })\*/

  //newarr has become the duplicate of names array

//filter

  //we make a blank array, but we are unsure of the exact number of array

  //we have to return true or false in filter unlike map

  //if true the array goes to the newarr, but if false it does not

  /\*const names = ["alice", "john", "charlie", "david", "emma"]

  let newarr = names.filter(function(value){

    return flase //Uncaught ReferenceError: flase is not defined

  })\*/

  /\*const names = ["alice", "john", "charlie", "david", "emma"]

  let newarr = names.filter(function(value){

    return true //(5) ['alice', 'john', 'charlie', 'david', 'emma']

  })\*/

  /\*const names = ["alice", "john", "charlie", "david", "emma"]

  let newarr = names.filter(function(value){

    if(value === "alice")

    return true //['alice']

  })\*/

  /\*const names = ["alice", "john", "charlie", "david", "emma", "axiom", "alex"]

  let newarr = names.filter(function(value){

    if(value.startsWith('a'))

    return true //(3) ['alice', 'axiom', 'alex']

  })\*/

//destructuring

  /\*const user = {

  id: 1,

  firstName: "John",

  lastName: "Doe",

  email: "john.doe@example.com",

  phone: "+1-555-123-4567",

  address: {

    street: "123 Main St",

    city: "Anytown",

    state: "CA",

    zip: "90210"

  },

  dateOfBirth: new Date("1990-05-15"),

  isActive: true,

  roles: ["user", "admin"],

  getFullName: function() {

    return `${this.firstName} ${this.lastName}`;

  },

  getAge: function() {

    const today = new Date();

    let age = today.getFullYear() - this.dateOfBirth.getFullYear();

    const m = today.getMonth() - this.dateOfBirth.getMonth();

    if (m < 0 || (m === 0 && today.getDate() < this.dateOfBirth.getDate())) {

      age--;

    }

    return age;

  },

  activate: function() {

    this.isActive = true;

  },

  deactivate: function() {

    this.isActive = false;

  }

};

let { zip }= user.address

console.log(zip)

let [\_, second] = user.roles

console.log(second) //admin

let [first, second] = user.roles

console.log(first, second) //user admin\*/

//spread and rest

  //spread is used to spread the values

  /\*const names = ["alice", "john", "charlie", "david", "emma"]

  //const copynames = names //using copynames, the array does not get copied, it gets referenced, so if we change copynames, names will also change

  const copynames = [...names] //"..." is the spread operator, now when we remove a value from copynames, the value is removed only from copynames and not names\*/

/\*const smallObject = {

  name: "Alice",

  age: 25,

  city: "Anytown"

  }

  //for smallObject also we get the same output //{name: 'Alice', age: 25, city: 'Anytown'}

  const newobj = {...smallObject} //{name: 'Alice', age: 25, city: 'Anytown'}\*/

  //res

  /\*function abcd(a,b,c,...random){

    console.log(a,b,c,random)

  }

  abcd(1,2,3,4,5,6) //1 2 3 (3) [4, 5, 6]\*/

//question 1: merging two arrays of data fetched from separate apis into a single list

/\*const names = ["alice", "john", "charlie", "david", "emma"]

const lastnames = ["al", "jo", "ch", "da", "em"]

const fullnames = [...names, ...lastnames]\*/

//question 2: filtering an array of objects to display items based on a search query

/\*const products = [

  { name: "Apple", type: "Electronics" },

  { name: "Banana", type: "Electronics" },

  { name: "Cherry", type: "Electronics" },

  { name: "Date", type: "Electronics" },

]

let newarr = products.filter(function(product){

  return product.type === "Electronics"

})

console.log(newarr)\*/

//question 3: mapping over an array of user data to create a list of user cards

/\*const users = [

  { name: "Alice", age: 25 },

  { name: "Bob", age: 30 },

  { name: "Charlie", age: 35 },

]

let newarr = users.map(function(user){

  return '<div><h3>${user.name}</h3><h5>${user.age}</h5></div>'

})\*/

//question 4: grouping an array of objects by a specific property

/\*const users = [

  { name: "Alice", age: 25, role: "admin" },

  { name: "Bob", age: 30, role: "admin"},

  { name: "Charlie", age: 35, role: "user"},

];

let obj = {}

users.forEach(function (user){

  if(obj[user.role]){

    obj[user.role].push(user)

  }

  else

  {

    obj[user.role] = []

    obj[user.role].push(user)

  }

})

console.log(obj)\*/

//question 5: removing or updating a specific object in an array of data based on a unique ID

const users = [

  { id: 1, name: "Alice", age: 25 },

  { id: 2, name: "Bob", age: 30 },

  { id: 3, name: "Charlie", age: 35 },

]

let newarr = users.filter(function(user){

  return user.id !== 2

})

console.log(newarr);