**scope.js**

//scope => lifetime of a variable

//let, const => block scope / local scope

//var => global scope / functional scope

//block =>{}

{

var y1 = 10;

let y2 = 20

}

console.log(y1)

// not defined(error) vs undefined(value)

// can a block hold var or will it leak outside => yes

var price = 1000

function getPrice() {

console.log("The old price is", price)

// var price = 500

console.log("The new price is", price)

}

getPrice()

**//hoisting + lexical scope**

//grandFather => 💰 => 🏠(Own scope)

//Father => 💰 + 💰 => 🚗(Own scope) + 🏠(lexical scope => grandFather)

//Child => 💰 + 💰 + 💰 => 💻(Own scope) + 🚗(lexical scope => Father) + 🏠(lexical scope => grandFather)

//closure => own scope + lexical scope

function outer() {

var a = 20

function inner() {

console.log(a)

}

return inner

}

// // console.log(outer())

// var close = outer()

// close()

outer()()

**//HOISTING**

**//JIT => Just In time compilation**

**// 1. compilation phase => console => skip**

**// 2. execution phase => JS & context**

//Example - 1 => var

**// 1. compilation phase**

console.log(x) // skip

//1000 lines

var x = 50 //JS => Do you know x? No || what is the value? => context => x => undefined

console.log(x) //skip

**// 2. execution phase**

console.log(x) // JS => Do you know x? yes || what is the value? => context => x => undefined

var x = 50 //JS => Do you know x? Yes || what is the value? => context => x => 50

console.log(x) //JS => Do you know x? Yes || what is the value? => context => x => 50

//Example - 2 => let

**// 1. compilation phase**

// console.log(z) // skip

// let z = 50 //JS => Do you know z? No || what is the value? => context => z => don't initialize

// console.log(z) //skip

**// 2. execution phase**

console.log(z) // JS => Do you know z? yes || what is the value? => context => z => Uncaught ReferenceError: Cannot access 'z' before initialization

let z = 50

console.log(z)

**class.js**

// class => blueprint of an object

//class creation

class Cars {

constructor() {

this.name = "BMW",

this.model = "X7",

this.color = "Black"

}

}

**//create object**

const bmw = new Cars()

console.log(bmw) //Cars {name: 'BMW', model: 'X7', color: 'Black'}

const bmw1 = new Cars()

console.log(bmw1) //Cars {name: 'BMW', model: 'X7', color: 'Black'}

class Cars1 {

constructor(name1, model1, color1) {

this.name1 = name1

this.model1 = model1

this.color1 = color1

}

}

**//create object**

const rr = new Cars1("RR", "RR1", "white")

console.log(rr)

const vw = new Cars1("VW", "vento", "red")

console.log(vw)

//this => access property of an object

let firstName = "Yukesh"

let lastName = "Chandran"

const user = {

firstName: "Soorya",

lastName: "Deviga",

greet: function () {

return `Say hi to ${this.firstName} ${this.lastName}`

}

}

console.log(user.greet())

**spread.js**

const arr = ["We", "are", "learning", "spread", "operator"]

console.log(arr)//['We', 'are', 'learning', 'spread', 'operator']

console.log(...arr)//We are learning spread operator

console.log("We", "are", "learning", "spread", "operator")//We are learning spread operator

**//copy array**

const arr1 = ["one", "two"]

// const arr2 = [...arr1, "three", "four"]

// console.log(arr1)//['one', 'two']

// console.log(arr2)//['one', 'two', 'three', 'four']

const arr2 = ["three", "four", ...arr1]

console.log(arr1)//['one', 'two']

console.log(arr2)// ['three', 'four', 'one', 'two']

const sameAddress = arr1 === arr2//different address

console.log(sameAddress)//false

**//clone array**

let a = [1, 2, 3]

let b = a //copy by reference

console.log(a)//[1, 2, 3]

console.log(b)//[1, 2, 3]

const c = a === b//same address

console.log(c)//true

let data = ["apple", "orange", "grapes"]

let copiedData = [...data]

console.log(data)

console.log(copiedData)

let address = data === copiedData

console.log(address)

data.push("mango")

console.log(data)

console.log(copiedData)

address = data === copiedData

console.log(address)

**//rest paramaters => using spread operator in function argument**

let result = function (...arg) {

console.log(arg)

}

result(2)//[2]

result(2, 3, 4) // [2, 3, 4]

result(5, 6, 7, 8, 9) //[5, 6, 7, 8, 9]

function multiply(x, y, z, a) {

console.log(x \* y \* z \* a)

}

const multi = [1, 2, 3, 4, 5]

multiply(...multi) //24

function displayInfo(name, age, ...hobbies) {

console.log(`Name: ${name}, Age: ${age}`) //Name: jack, Age: 20

console.log(`Hobbies: "${hobbies.join('","')}"`)//Hobbies: reading,traveling,painting,gardening,cooking

}

displayInfo("jack", 20, "reading", "traveling", "painting", "gardening", "cooking")

**objectDestruct.js**

const avenger = {

name: "Tony Stark",

house: "🏠",

networth: "💰💰💰💰💰💰",

power: "🤖",

phrase: "Live happily"

}

console.log(avenger)

const {

name,

networth,

power = "💿",

skill = ["genius", "billionarie", "philanthropist"]

} = {

name: "Tony Stark",

house: "🏠",

networth: "💰💰💰💰💰💰",

power: "🤖",

phrase: "Live happily"

}

console.log(name, power, skill)

const person = {

firstName: "Jack",

lastName: "Doe",

age: 20

}

console.log(person.firstName, person.lastName)

const { firstName, lastName } = person

console.log(firstName, lastName)

**//rename var during desctructring**

const { firstName: userName, age: personAge } = person

console.log(userName, personAge)

//nested destruct

const person1 = {

name1: 'John',

age1: 25,

address: {

city: "india",

zipcode: "100001"

},

hobbies: ["reading", "writing", "travelling", "coding"]

}

console.log(person1.hobbies[0])

const { name1, age1, address: { city, zipcode }, hobbies: [reading, writing, travelling, coding] } = person1

console.log(reading)

console.log(city)

console.log(coding)

**//object shorthand => key and value name should be same**

const studentName = "renuka"

const batch = "FSD59WD2"

const mark = 90

const student1 = {

studentName,

batch,

score: mark,

}

console.log(student1)

**index.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 type="module" src="fun.js"></script>

<script type="module" src="cool.js"></script>

</body>

</html>

**funcDestruct.js**

const studentName = "Renuka"

const batch = "FSD59WD2"

const mark = 90

const student1 = {

studentName,

batch,

score: mark,

}

console.log(student1)

//funct destrcut

//Renuka has got 90 and she belongs to FSD59WD2 batch

function printDetails(student1) {

return student1.studentName + " has got " + student1.score + " and she belongs to " + student1.batch + " batch "

}

console.log(printDetails(student1))

//destrcuturing

function printDetails1(student1) {

const { studentName, score, batch } = student1

return studentName + " has got " + score + " and she belongs to " + batch + " batch "

}

console.log(printDetails1(student1))

//destrcut in argument itself

function printDetails2({ studentName, score, batch }) {

return studentName + " has got " + score + " and she belongs to " + batch + " batch "

}

console.log(printDetails2(student1))

//Template literals => `` => backtick + interpolation ${} => substitute value

function printDetails3({ studentName, score, batch }) {

return `${studentName} has got ${score} and she belongs to ${batch} batch`

}

console.log(printDetails3(student1))

//arrow function + Template literals + destructuring => es6

const printDetails3 = ({ studentName, score, batch }) => `${studentName} has got ${score} and she belongs to ${batch} batch`

console.log(printDetails3(student1))

**fun.js**

const printDetails = ({ studentName, batch }) =>

`${studentName} belongs to ${batch} batch`

export const msg = "Happy day😀"

// export { printDetails, msg }

export default printDetails

**//export & import**

// 1. named export & import - preferred

// 2. default export & import // js file typeof => commonJS, Module

**cool.js**

// import { printDetails, msg } from "./fun.js"

import printDetails from "./fun.js"

import { msg } from "./fun.js"

const student = {

studentName: "Renuka",

batch: "FSD59WD2",

}

console.log(printDetails(student))

console.log(msg)

**CRUD.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>

* **C - CREATE => INSERT/ADD => POST**
* **R - READ => GET**
* **U - UPDATE => PUT**
* **D -DELETE => DELETE**

<script>

let url = "https://659e6ba547ae28b0bd35caec.mockapi.io/products";

// fetch(url, { method: "GET" })

// .then((res) => res.json())

// .then((data) => console.log(data))

async function getProductData() {

let data = await fetch(url, { method: "GET" })

let res = await data.json()

console.log(res)

}

getProductData()

const id = 1

async function getProductById() {

let data = await fetch(`${url}/${id}`, { method: "GET" })

let res = await data.json()

console.log(res)

}

getProductById()

**//create products**

async function createProduct() {

let newProduct = {

name: "Macbook m3",

poster: "https://m.media-amazon.com/images/I/71d7rfSl0wL.\_AC\_UY218\_.jpg",

price: "₹2,00,000",

rating: 5,

summary: "Macbook m3"

}

//POST

// 1. method - POST

// 2. headers - data is in JSON

// 3. body - pass data in body and it should be in JSON format

let data = await fetch(url,

{

method: "POST",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(newProduct)

}

)

let res = await data.json()

console.table(res)

}

**// createProduct()**

**//delete products**

async function deleteProductById(id) {

let data = await fetch(`${url}/${id}`, { method: "DELETE" })

let res = await data.json()

console.table(res)

}

deleteProductById(12)

**//update product**

async function updateProductById(id) {

let updatedProduct = {

"name": "Macbook m3",

"poster": "https://m.media-amazon.com/images/I/71d7rfSl0wL.\_AC\_UY218\_.jpg",

"price": "₹3,00,000",

"rating": 4,

"summary": "Macbook m3 chip",

}

//PUT

// 1. method - PUT

// 2. headers - data is in JSON

// 3. body - pass data in body and it should be in JSON format

let data = await fetch(`${url}/${id}`,

{

method: "PUT",

headers: { "Content-Type": "application/json" },

body: JSON.stringify(updatedProduct)

}

)

let res = await data.json()

console.table(res)

}

updateProductById(11)

</script>

</body>

</html>

**products.json**

[

{

"id": "1",

"name": "iPhone 15 (128 GB)",

"poster": "https://m.media-amazon.com/images/I/71d7rfSl0wL.\_AC\_UY218\_.jpg",

"price": "₹77,900 ",

"rating": 4.5,

"summary": "DYNAMIC ISLAND COMES TO IPHONE 15 — Dynamic Island bubbles up alerts and Live Activities — so you don’t miss them while you’re doing something else. You can see who’s calling, track your next ride, check your flight status, and so much more."

},

{

"id": "2",

"name": "iPhone 15 Pro (128 GB)",

"poster": "https://m.media-amazon.com/images/I/81SigpJN1KL.\_AC\_UY218\_.jpg",

"price": "₹1,34,900 ",

"rating": 5,

"summary": " iPhone 15 Pro has a strong and light aerospace-grade titanium design with a textured matte-glass back. It also features a Ceramic Shield front that’s tougher than any smartphone glass. And it’s splash, water, and dust resistant."

},

{

"id": "3",

"name": "Samsung Galaxy S23 5G (256GB Storage)",

"poster": "https://m.media-amazon.com/images/I/51L8W6d-DNL.\_AC\_UY218\_.jpg",

"price": "₹65,320",

"rating": 4.7,

"summary": "FASTEST MOBILE PROCESSOR AVAILABLE: Whether you’re working hard, playing hard or doing both at the same time, smoothly switch between apps with our fastest processor ever."

},

{

"id": "4",

"name": "Samsung Galaxy S23 Ultra 5G (256GB Storage)",

"poster": "https://m.media-amazon.com/images/I/51hqXIAVXAL.\_AC\_UY218\_.jpg",

"price": "₹1,00,000",

"rating": 5,

"summary": "Create crystal-clear content worth sharing with Galaxy S23 Ultra’s 200MP camera — the highest camera resolution on a phone; Whether you’re posting or printing, Galaxy S23 Ultra always does the moment justice."

},

{

"id": "5",

"name": "Apple AirPods Pro (2nd Generation)​​​​​​",

"poster": "https://m.media-amazon.com/images/I/61SUj2aKoEL.\_AC\_UY218\_.jpg",

"price": "₹20,999 ",

"rating": 4.8,

"summary": "Active Noise Cancellation reduces unwanted background noise.Adaptive Transparency lets outside sounds in while reducing loud environmental noise.Personalised Spatial Audio with dynamic head tracking places sound all around you."

},]

import "./styles.css";

import { useState } from "react";

export default function App() {

//JS starts

// const name = "Ajay";

// const name1 = "Aravind";

const person = ["Aravind", "Ajay", "Dhana"]; //Array of Strings

//Array of object

const users = [

{

name: "Aravind",

pic: "",

},

{

name: "Dhana",

pic: "",

},

];

//JS ends

//JSX starts

return (

<div className="App">

{/\* <Counter /> \*/}

{/\* <Welcome name={person[0]} />

<Welcome name={person[1]} /> \*/}

{/\* {person.map((personName) => (

<Welcome name={personName} />

))} \*/}

{/\* <Welcome name="Aravind" age="25" gender="male" />

<Welcome name="Ajay" age="20" />

<Welcome name="Dhana" age="20" /> \*/}

{users.map(

(usr) => {

return (

<div>

<img src={usr.pic} />

<h3>{usr.name}</h3>

<Counter />

</div>

);

}

// <ProfilePic name={usr.name} pic={usr.pic} />

)}

{/\* <ProfilePic

name="Aravind"

pic=""

/>

<ProfilePic

name="Dhana"

pic=""

/> \*/}

</div>

);

//JSX ends

}

//JSX => extended JS XML

//class => reserved keyword

//{} => template syntax

//custom component

// 1.First letter must be capital

// 2.It return a JSX element

function Welcome(props) {

console.log(props);

return (

<div>

<h1>Hello {props.name}</h1>

</div>

);

}

function ProfilePic({ name, pic }) {

return (

<div>

<img src={pic} />

<h3>{name}</h3>

<Counter />

</div>

);

}

function Counter() {

// let like = 10;

const [like, setLike] = useState(0);

const [dislike, setDislike] = useState(0);

console.log("updated Like", like);

return (

<div>

<button

onClick={() => {

setLike(like + 1);

console.log(like);

}}

>

👍 {like}

</button>

<button

onClick={() => {

setDislike(dislike + 1);

console.log(dislike);

}}

>

👎 {dislike}

</button>

{/\* <Sample lk={like} /> \*/}

</div>

);

}

function Sample({ lk }) {

return <div>Like value is {lk}</div>;

}

//hooks => function

// useState => const [state, setState]= useState(initialValue)

// state => current state