DAY 2

Tuesday, 6 May 2025 8:01 AM

Javascript → playground
React → pages.

- parameters are always constant.

```
function function2 ( parameter) {
// parameter is a function
reference.
```

const parameter = function 1

JS → Functional programming language.

C, C++ → function pointers
JS → function reference.

function function2 (parameter)

parameter ()

function 2 (function 1)

* Square each no. & collect in an array.

Instead of doing traditional method. We will use map. more simple way. Everything already done by map.

Map → looping over a function.

Anonymous

```
// using map function.

const squares = numbers.map((number) =>
{
    return number ** 2

})
```

In JS, It is called Arrow Function.
In Java, Lambda. expression.

```
const squares = numbers.map((number)
=> number ** 2)
```

This will be used to render array.

map → selecting columns from DB

Map() → Used to iterate over a collection to transform the values to new ones.

→ accepts a function as a parameter which gets called every time for every value.

→

→

→

—

3 files to be imported.
2 React & 1 Babel.

div root by id → mandatory.

Rendering scalar values

ReactDOM.createRoot (Document.getElementById ('root'

* Rendering object (as car object)

{car['model']} → just this difference here. syntax.

* Rendering Array of scalar values. Using Index

```
const contries = ['india', 'USA', ---]
// array of strings.
```

- Every country in separate Div.

```
const div = {
    <div>
        <div> { countries [0] } </div>
```

Drawback → ① We are Hardcoding index values here. If we add new country.
② If Japan deleted, still one div is empty remaining.

Solution → Map

* Rendering All values using Map.

countries → JS variable
array

```
{ countries.map((country) => {
    return <div> {country} <div>
}
```

→ this does not leave empty div.

* Rendering with css class.

Just to look good.

return <div className {country} <div>
Here we call by className.

* Rendering an array with objects.

```
const product = [
    { id : 1, title : 'product 1', price : 100,
      desc : 'good' },
    - - - - - } ]
```

Jim Rohn → Burning Desire. lacking or not interested.

Sir → Body has Body clock.

Max - 6hrs. sleep.

After 10:30 → your brain captures less.

11 - 5 → sleep. wake up early in the morning.

Exercise → give energy through out day.

as it gives only scalar values → cannot render object directly.

```
return (
    <div>
    <div> Id : { product ['id'] } </div>
)
```

→ By doing this prblm solved.

- Now this written in a table to look good.
- returning row with td elements. change in return. in above function.

- Now will add Bootstrap.

className = 'table'.

className = 'container'
page header — text align. keeping at centre.

table className = --- stripped --- → looks more good.

Rendering scalar.
" object.
" array of scalar
" array of object

4 Types.

Note → yarn → Better package manager than npm.

Use yarn. (node package manager)

Bash commands.

```
# install yarn.
npm install -g yarn.  → windows
sudo                  → Linux.
```

Next.js → Framework react good to create App.

vite → Tool.

* create react app using vite.

```
npm create vite @ latest < app. name>
```

Vanilla → pure JS

```
yarn create vite@latest < app.name>
```

* Refer more to sir's notes.

package.json → all dependencies inside.
react
react-dom.

- npm install / yarn.

Q. Diff. dependencies & Dev Dependencies?

dependencies → modules packages added when you are deploying project. To run the project. Eg. React React-DOM.

Dev Dependencies → Used while you are developing but not needed to run. Eg. Babel → To develop app. not to run it.

Q. What is package.json?

node modules → contains all modules dependencies to develop/run app.

- will be downloaded every time when npm install

- never commit it this directory in git.

* gitignore → file which contains the rules of the Files which needed to

. → hidden.

☐ Write notes after Lecture in colours.

SPA so only one html file.

* yarn.lock → contain latest version of dependencies in node modules directory.

* src → which contains all components of application.
- contains assets (image, video, ---)

JSX → JS + XML

main.jsx → contains code to start react subsystem.
this does import & create Root. & here we render App instead of DIV.

☐ Write env. setup notes in Notebook or copy paste here in these notes.

* yarn dev → vite → It will start a lite web server on port 5173.

div id = root → To load JSX
To load.

Main.jsx → creates root & renders app component
App component
starts loading UI.

Q. What is a component? Advantages / Disadvantages.

Reusable entity which contains logic in JS & UI in JSX syntax.

- Everything in React is a component.

Eg. gaana, air bnb — Same structure multiple times.

- component Used multiple times.

- component can be small as part of app. or as big as entire page.

* Just develop components once → reuse multiple Times.

Just data changes.

In MPA → we need to create multiple pages.

* Types of component

class component
- component created using a class. — used for creating statefull component.
- Earlier, class component

class component has overhead members which make slower.

- Later, Hooks introduced so don't use class components.

One who return JSX UI components → Functional components.

Earlier → stateless.

Now, → Hooks → can store state.

* Functional components are preferred over class components since they don't have overhead members & it is Fast.

App → component. not a Tag.

< App >

component returns JSX. mgdh toh component mahar apan.

node on server side.

module.export.
No need to do it here. import & export is simple.

Brain Tracy — No Excuses.

Understand the project structure.

- By creating div. & by using . or [] we can render the object.

- cannot directly render the object.

Another. function → component

components dir > person.jsx

write one component & export it.

<hr/> → ?

component which returns JSX.

< Person />
This will call the function & jsx will be written.
In Main file.

convention — we keep File Name same.

How to make component reusable?

What is props in React?

* props → object which is a collection of all the properties passed to component.

How to pass data with props

img src = " " alt " "
props.

component — person — 3 properties
& that 3 properties is passed as parameter.
function App (props)

- is the only way for a parent component to send the data/properties to child component.

App → parent component
Person → child component

- props is only read only object. child component must not change the values sent by parent component.

- props of name
props dot name.

- with props we send data/props. to child from parent.

cars.jsx

Component

```
function Car ( props ) {
    return {
        <div>       <div>.

}
export default Car
```

```
import Car - - - - -

< Car
    model = {           }
    company = {         }
/>
```

prblm → we need to write props many times.

solⁿ → destructuring

```
// object Destructuring
const { model, company } = props.
```

Use props & then destructure. Instead write directly.

```
function Computer( { brand, model, cpu, memory } )
```

```
< Computer
    cpu = {           }
```

→ ctrl + space now for intellisense.

* Destructuring at time of declaring component.

See the difference in sir's code file.

cars. person computer.

* Always start the component name with Upper case.

* Name of the file should be same as the component name.

Disadvantage of React

React does not have any architecture where as Angular has predefined architecture.

components same in Angular.

Learning Angular will be easy after react.

Angular → Architecture
React → performance.

- If component is representing page or screen, keep it in directory named pages or screen.

- * These are common conventions not compulsions.

All components in one File → Not recommended.

export → Used to export any entity from a file for others to import.
- A File can export multiple entities for others
- { }

export default. → By default, only one entity can be exported from a file with default keyword. class Fun. variable const.
- no { }

export → multiple
export default → single.

App — default
{ create Root } — without default.

* importing with an alias is possible without { }.

* Both App & Cars returning num so for no confusion, we use alias.
Namespace → multiple class with same name

import App, {myVar, num}
default chya jagi syntax ahe so.
Non-default Kuthe pan sake.
To resolve conflict.

export default → not comp.
→ recommended.

Q. What is state in React?

Hot loading / lite server / Development server.

Live server.

interpolation → variable to jsx.

<button onClick = { onIncrement }
this by react. onclick
Fn. reference.

```
// click event Handler.

const onIncrement = () => {
    alert = - - - - - - - -
    counter += 1
}
```

* Know when to use ${ } & { }

Q. What is state? Diff b/w prop & state.

prblm → counter : 0. → not changing UI but in console changing.
never rerender on its own.

if refresh on own → MPA.

" A state is the only way to trigger component refresh action. "

open Diagram → draw.io

JS similar to Python.

Every component has state.

To rerender, you need to modify the state.

By keeping counter inside state, it will reload automatically.

* state — object (collection of property-value pairs) maintained by component to trigger component re-render action.

- If there is a change in the component state, the component will re-render itself.

- unlike props, state object is readwritable.
- Every component will maintain its own state.

- To add a state member inside a Functional Component Use a react hook name useState().

* React Hooks :-

- special function which start with "use"

Types

Built in Hooks
use State
" Effect
" Reducer
" Memo
" Id
" Ref
" callback
" Navigate
" selector
" Dispatch.

custom Hooks
user defined Hooks.

* useState()

- hook used to add a member inside a component's state.

- accepts a parameter which is the initial value of member.

- returns an array with 2 values.

0th position → ref. to mem (read) → getter arr[0]

1st position → ref. to fun (update) → setter arr[1]

* VS code extensions:-

① Auto Import
② Auto rename Tag
③ ES7 code snippets.

const arr = useState(10)

Array destructuring
converting into variables.

const [counter, setCounter] = useState(10)
getter, setter

Here position matters

in object destructure → Name same as key.

set counter [counter +1] → as it is const. → that's why ~~counter~~ don't work.

* State maintained for rerender the component.

Accordion → JQuery Animation.

Accordion component → Acc .css
Acc .jsx

Flow → Function, return, export, import, div, classname.

Now doing .css for style.

index.css → for all css components. so write there.

Toggle — need to create state. & re-rendering For rendering

const [isVisible, setIsVisible] = useState (false)

* Conditional rendering

```
{ isVisible && (
        <div className = ' body's>
        Lorem - - - -
    ) }
```

if this True this then execute this

isVisible false to pude check ch nhi krt.

```
const onToggle = () => {
    setVisible ( ! isVisible)
}
```

<button onClick - - - >

state	props
① read writable	① read only
② re-renders itself	② Nothing changes.
③ maintained by individual component	③ will be sent by parent comp. to child comp.
④ useState() hook is required to create state inside Fun. component	④ No Hook

counter, Accordion → stateful component.

① Component life cycle
② Hook next
③ Router.