React (Day-3)

While making development & prod builds, instead of using 'npx parcel...'

define scripts in package.json.

"scripts": { "start": "parcel index.html"
"build": "parcel build index.html" }

* Defining scripts is an industry standard! When you have no clue about a new project that you should work on, just check the scripts in package.json.

How to run scripts? npm run <script_name>

Exception: npm run start == npm start (shortcut)
(only for the "start" script)

→ On day-1, we say how complex it is to create react elements.

So Facebook developers invented JSX.

JSX is javascript syntax which makes creating react elements easier.

→ JSX - makes it possible to write HTML & JS in one single file instead of traditionally creating separate HTML, CSS & JS files.

→ JSX is not HTML inside JS. JSX is HTML-like syntax. "JSX is JSX" That's it!

React.createElement("h1" {}, "This is h1");

JSX Equivalent:-

const heading = <h1>This is h1</h1>
// This ain't html bro! This is JSX.

→ JSX creates react elements, but JSX is not a part of React. It is a separate entity.

→ Btw, how is the browser able to process JSX?
Browser has JS engine which understands only pure JS.
And we know react and react-dom doesn't have anything to do with JSX.
Then who's converting this JSX such that react and browser can process JSX?
It is PARCEL! Bhe not just parcel, but when we install parcel, babel is also installed. So BABEL does the job!

The whole process of transpilation of JSX before JSX reaches browser is done by both Parcel and Babel. (it is a JS compiler)

JSX —(Babel)→ React.createElement —→ React Element Object —(Render)→ HTML Element

→ all the attributes in JSX are camelCase. And class in HTML is className in JSX.

→ If we write one line of JSX, then it's okay to write without (parenthesis) also but for multiple lines of JSX, we should wrap our jsx in parenthesis ().

```
const heading = <h1> Heading </h1>; ✓

const heading = (<h1> Heading </h1>); ✓

const heading = (
    <h1 className="head">
        Heading
    </h1>
); ✓

const heading = <h1 className="head">
    Heading ing
    </h1>; ✗
```

Components : Everything in react is made of components.

2 types of components

- Functional (NEW)
- class (OLD)

Functional component → Just another javascript function which returns JSX. → In other words, can also say, functional component returns a react element.

Ex:
```
const HeadingComponent = () => {
    return <h1> Hello </h1>
};
```

Note:
```
const HeadComp = () => <h1> Hello </h1>
```
This HeadComp and HeadingComponent are both same and return same JSX. Both are correct.

If you wanna write like this without return keyword, in multiple lines, then enclose JSX in parenthesis.

Note: How do we render components btw?

```
render(HeadComp); ✗

render(<HeadComp />); ✓
```

=> How to render one component inside another?

Same!

```
const HeadingComponent = () => (
    <div id="container">
        <Title />
        <h1> Hello World. </h1>
    </div>
);
```

another component →

And this is what is known as component composition

Note: To access js code inside jsx, just put the js code inside curly braces.

i) can put jsx element in functional component jsx.

ii) can put a functional component in jsx element.

Both are possible.

iii) can put component inside component is no need of curly braces

iv) can put jsx element inside another jsx element. (using {} of c)

Components can be used inside jsx as:-

```
<Title />
<Title></Title>
{Title()}
```

→ lmao. can call the function too at the end of day, component is a function that returns jsx.

When ppl ask why your web app is fast why/how is it so much readable?

fast? → Parcel / babel

readable? → jsx / babel

ofc react makes it fast, but don't forget about what actually makes react apps fast.