Episode 8 : Laying the foundations

Date: ____

↳ tips to start a project

- To Build development Build or to start our app in dev mode we use
  npx parcel index.html

- Same way for production build we use,
  npx parcel build index.html

- To make our life easier, we can create scripts for this commands to build our project instead of writing this commands again and again

- **Creating scripts or shortcut for devBuild & prodBuild.**

- It is a npm script need to create in package.json.

1. Script for starting our project in dev mode

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

Write the command needs to be execute using npm in this scripts.

2. script for production

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

- How we no longer need to write previous commands to run or build our project in dev or prod mode

- How to run this npm scripts?
  1. npm run start or npm start
  2. npm run build

Note: npm build - It won't work

↳ fundamentals (Laying the foundation)

- HTML elements (P, h1, div) are the DOM elements.

- React element, equivalent to this DOM elements

- React element is object, react.createElement create an object

- when we render this element onto the DOM becomes an HTML element

- React.createElement() ⇒ Object ⇒ HTMLElement (render)

- Render the react elements to DOM
  - using root

↳ JSX

- Creating react element using core react fundamentals like React.createElement() is not a good way. It is very clumsy.

- React.createElement() is a core of react, when react was build developers used to create react element.

- It is not a developer friendly.

- To help developers, to code more efficiently or create react elements fast & easy way, facebook developers create something known as JSX.

JSX is a javascript syntax to create react react element.

Note: JSX is not a part of react. Both are different.

- Make developers life easy.

- JSX is a convention where we can merge this HTML & CSS together.

- JSX is where we can merge DOM elements (HTML elements) and Javascript (logic for elements) in a single file

- Create tags using JSX.

```
const jsxHeading = <h1> Namaste React
    using JSX </h1>
```

Note - JSX is not HTML inside Javascript
JSX is different than HTML.

- JSX is a HTML like syntax or XML like syntax

- JSX is different syntax.

```
const jsxHeading = <h1> Aditi Deokar </h1>
```

jsxHeading → React Element (object)

<h1> Aditi Deokar </h1> → JSX (HTML like syntax)

- creating h1 tag in react

```
const heading = React.createElement ("h1",
    {}, "This heading from react");
```

- creating h1 tag in JSX

```
const jsxHeading = <h1> This heading from
    JSX </h1> ;
```

- **Is JSX a valid Javascript ?**

- It is not pure javascript

- Javascript does not comes with JSX build inside it

- JS engine (i.e Browsers engine) does not understand JSX

- JS engine understands only Ecmascript 6 and all versions of the ES.

- JS engine throw an error on jsx Heading.

- **How JSX working with Javascript ?**

- parcel doing the job behind the scenes

- JSX written by us goes to the js engine (babel) and it is transpiled before it goes to the browser engine (ES).

- JSX is transpiled before it reaches the Js engine

- This transpilation is done by parcel.

- parcel is not alone doing transpilation process.

- parcel is like a manager for us.

- parcel gives the responsibility of transpilation to the package i.e babel.

babel:

- babel transpiling code into react will understand.

- babel is JS compiler / transpiler

• How code is running ?

1] const heading = React.createElement(
"h1",
{},
"heading from React");

React.createElement => ReactElement (JS object)
=> HTMLElement (render)

2] const jsxHeading = <h1 id="heading">
Namaste React using JSX </h1>;

JSX => React.createElement => React Element
(object) => HTML Element (render)

- babel converting JSX into react.createElement

- How to give attribute to tag in JSX

```
<h1 id = "heading" className = "heading">
    Namaste React </h1>
```

- Use camelCase (Give attributes to tag)

- explore : JSX tags, attributes

- JSX in multiple lines

- use () parenthesis

```
const jsxHeading = (<h1 id = "heading"
    Namaste React using JSX
    </h1>);
```

↳ React Components

- There are two types of components

1] Class based components. - OLD way
2] functional components - NEW way

- Functional Components -

- It is just a normal Javascript function.

- Always name a component with first capital letter (It is a way to understand react that it is a component)

- React component returns some piece of JSX.

A function that returns some piece of JSX code is known as functional component

e.g const HeadingComponent = () => {
return(
<h1> React functional component </h1>);
};

- A function that returns react element is call react functional component

- another way of writing functional component

const HeadingComponent1 = () => {
<h1> JSX Heading </h1>);

• Nested JSX

const HeadingComponent = () => (

<h1> Nested JSX Element </h1>

);

- Rendering component into root

```
root.render ( <HeadingComponent /> );
```

It is syntax which babel understands.

↳ Component compositions

- Rendering component into another component

- It is like nested component structure

```
- const Heading = () => (
    <div>
      <h1> Heading </h1>
      <Title />
    </div>
  );
```

- creating component using function keyword.

```
const Title = function () {
  return (
      <h1> Namaste React </h1>
  );
};
```

- In JSX, anywhere you write { }, in that we can write any piece of Javascript expression.

Writing Javascript inside JSX using {}.

↳ put react element inside component

- using {} in JSX.

e.g const title = (
<h1> Namaste React </h1>

const Heading = () => (

{title}
);

- we can put react element inside react element using same way.

- we can't use component before initialising it

- we can put component inside react element also.

↳ crossite scripting

cross-scripting is an attack by attracker able to run javascript code in our browser to get our personal data.

It can steal cookies, local storage reads, reads session storage, Info about laptop.

JSX prevent all cross-scripting attacks.

When we are using any api in project, if api passes some malicious code, JSX will escape it

- All React functional component in Javascript function.
  We can also call it in JSX.

e.g
```
{Title()}
<Title/>
<Title></Title>
```
] same things

"Everything we code should have a purpose"