Chapter 03

Laying the foundation

Page No. _____
Date

Pollyfill

→ To make older browser understand our new code, the new code is converted into a older code which browser can understand called pollyfill.

→ babel do this conversion automatically

Eg:- ES6 is the newer version of JS

If I'm working on 1999 browser my browser will not understand what is this

const, newPromise etc.

So, there is a replacement code for these functionalities which is compactible with older version of browser.

→ So, this is what happen when we write "browserlist" → Our code is converted to older one.

Babel

Is a JS package/library used to convert code written in newer version of JS (ECMAScript 2015, 2016, 2017 etc) into code that can be run in older JS Engines.

To run our app, command is :-

```
npx parcel index.html
```

We always don't have to write this command. Generally, we build a script inside package.json which runs this command in an easy way

package.json

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

So to run the project I've to use :-

```
npm run start
```

shortcut :-
npm start

"start" script execute this command.

Build command.

```
npx parcel build index.html
```

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

Page No. _____
Date

Note :-

npx = npm run

So npm run build will builds a project

console log are not removed automatically by parcel. You have to configure your projects to remove it.

~~These~~ There is a package which helps to remove console logs :-

babel-plugin-transform-remove-console

→ Before installing this package } for create a folder called babelrc } configuration

→ And include :-

```
{
  "plugins": [["transform-remove-console",
    { "exclude": ["error", "warn"] }]]
}
```

→ Then, build : npm run build to see that all console.logs are removed.

Page No.
Date

Solution Introduction of key

React support - 'key' attribute

When children have keys, React uses the key to match # children in the orginal tree with children in subsequent tree. Thus making tree-conversion efficient.

```
<ul>
    <li key="2014">Connection</li>
    <li key="2015">DURE</li>
    <li key="2016">Villanova</li>
</ul>
```

Thus react has to do very less work so always use keys whenever you have multiple children.

createElement

React.createElement() is creating an object.

This object is converted into HTML code & puts it upon DOM.

If you want to build a big HTML structure then using 'createElement()' is not a good solution.
So there comes introduction of JSX

Page No. _____
Date

- On the next state or props update, render() fn will return a different tree of react element.

Whenever react is updating the DOM for eg:-

```
<ul>
    <li> Duke </li>
    <li> Villanova </li>
</ul>
```

Now, I introduced on child over the top then react will have to do lot of efforts react will have to re-render everything That means, [react will have to change that whole DOM tree]

```
<ul>
    <li> Connection </li>
    <li> Duke </li>
    <li> Villanova </li>
</ul>
```

As react has to re-render everything it will not give you good performance

In large scale application, it is far too expensive.

Page No.
Date

Solution Introduction of key

React support - 'key' attribute

When children have keys, React uses the key to match # children in the orignal tree with children in subsequent tree. Thus making tree - conversion efficient.

```
<ul>
  <li key="2014">Connection</li>
  <li key="2015">DUKE</li>
  <li key="2016">Villanova</li>
</ul>
```

Thus react has to do very less work so always use keys whenever you have multiple children.

createElement

→ React.createElement() is creating an object.

→ This object is converted into HTML code & puts it upon DOM.

If you want to build a big HTML structure then using 'createElement()' is not a good solution.
So these comes introduction of JSX

JSX

When facebook created React, the major concept behind bringing react was - that we want to write a lot of HTML using JS because JS is very performant.

```
import { createElement as CE } from 'react';

const heading = React.ce(
                    "h1",
                    { id : "title",
                      key : "h1"
                    }
                    "Hello World"
                );
```

Instead of writting all these :-

```
const heading = <h1> Hello World </h1>
```

→ This is JSX

→ JSX is not 'HTML inside JS'

→ JSX has 'HTML-like' syntax

This is a valid JS code :-

```
const heading = (
    <h1 id = "title" key "h1">
        Helloworld
    <h1>
);
```

JSX expression

React keeps track of 'key'

HW

1. What is different b/w HTML & JSX ?

Our bowser cannot understand JSX

"Babel" understand this code.

2. What are different usage of JSX ?

3. How to create image tags inside JSX.

→ JSX uses React.createElement behind the scenes.

→ JSX ⇒ React.createElement ⇒ Object ⇒ HTML (DOM)

→ Babel convert JSX to React.createElement()

(Read Babel's Documentation)

Page No.
Date

JSX is created to empower React

Advantage of JSX

- Developer experience
- Syntactical sugar
- Readability
- Less Code
- Maintainability
- Repeation

Babel comes along with parcel

Component

Everything is a component in React.

React components

2 types

(a) Functional component - New way of writing code

(b) Class Based component - OLD way

Functional Component

- is nothing but a JS function
- is a normal JS function which returns some piece of react element (JSX here)

Page No.
Date

Eg:-

```
const headerComponent = () => {
    return <h1> Hello world </h1>;
};
```

- for any component,
  Name starts with capital letter.
  (It is not mandatory but it's a convention)

To render function component write:-
`<HeadesComponent />`

React element	Functional Component
`const heading = (`	`const heading = () => {`
`<h1 id="title"`	`return (`
`key="h1">`	`<h1 id="title"`
Hello world	`key="h1">`
`</h1>`	Hello world
`);`	`</h1>);`
	`};`
React element is finally an object.	Function component is finally a function.

Page No.
Date

The Next Amazing thing

```
const title = () => {
    <h1> Hello world </h1>
}
```

is a functional component.

*
```
const headerComponent = () => {
    return (
        <div>
            <Title />
            <h2> Namaste React </h2>
            <h2> Hai all </h2>
        </div>
    );
};
```

This is a normal js function.

(2) `const title = ( <h1> Hello world </h1> )`

→ is a normal variable

*
```
const HeaderComponent = () => {
    return (
        <div>
            {title}
```

<h2> Namaste React </h2>
<h2> Hai all </h2>

);
};

NB:

* Whenever you write JSX, you can write any piece of JS code between parenthesis {} It will work.

* JSX is very secure
JSX makes sure your app is safe
It does sanitization

```
const data = api.getData();
const HeadComponent = () => {

return (
    <div>
        {data}
    </div>
    <h2> Hello world </h2>
);
};
```

write anything inside {} JSX will sanitize the code.

Page No.
Date

Component Composition

If I have to use a component inside a component then it is called component composition / composing component.