**Babel** is a compiler, which converts jsx code into ES5 JavaScript that all browser understands.

**Webpack** is a tool which that is watching files to change and when they do it feeds those files into Babel, which turns jsx into Js.

**Return without brackets will give error if we use wrapper html element in next line of return**

function HelloWorld() {

return

<div>

<Hello/> <World/>!

</div>;

}

This will fail with an error.

**Correct way**

function HelloWorld() {

return <div>

<Hello/> <World/>!

</div>;

}

**Best way Wrap With a Tag**

**A component function must return a single element.**

function HelloWorld() {

return (

<div>

<Hello/> <World/>!

</div>

);

}

A lot of the time, this is perfectly fine. But sometimes, you won’t want to have a wrapper element, like

if you have a component that returns two table cells:

function NameCells() {

return (

<td>First Name</td>

<td>Last Name</td>

);

}

You can’t wrap these elements in a <div>, because the <td> table cells need to be direct descendants

of a <tr> table row. How can you combine them?

**Fragments**

React’s answer is the *fragment*. This component was added in React 16.2, and can be used like this:

function NameCells() {

return (

<React.Fragment>

<td>First Name</td>

<td>Last Name</td>

</React.Fragment>

);

}

After rendering, the React.Fragment component will “disappear”, leaving only the children inside it,

so that the DOM structure will have no wrapper components.

31

*Contents*

Fragments make it easier to produce valid HTML (such as keeping <td> elements directly inside

<tr>s), and they keep the DOM structure flatter which makes it easier to write semantic HTML

(which is also usually more accessible HTML).

**Fragment Syntax use empty tag**

function NameCells() {

return (

<>

<td>First Name</td>

<td>Last Name</td>

</>

);

}

**“If” in JSX**

The next question you might wonder is, “How do I write a conditional if I can’t use ‘if’?” There are a

couple of options.

The first is the ternary operator (the question mark, ?). Use it like this:

33

function ValidIndicator() {

const isValid = true;

return (

<span>{isValid ? 'valid' : 'not valid'}</span>

);

}

**You can also use Boolean operators such as && like this:**

function ValidIndicator() {

const isValid = true;

return (

<span>

{isValid && 'valid'}

{!isValid && 'not valid'}

</span>

);

}

**Capitalize Component Names**

The components you write must begin with an uppercase letter. This means using names like

**UserList** and **Menu** and **SubmitButton**, and not names like **userList**, menu, and **submit\_button**.

**In JSX, a component that starts with a lowercase letter is assumed to be a built-in HTML or SVG**

element (div, ul, rect, etc.).

**Concatenation of string and variables by using back quote `**

function Greetings(){

    let username= "root";

    //let username= undefined;

    return(

        <div>

            {username ? `Hello ${username}`:'Not logged in'}

        </div>

    );

**Learn React Testing Tool:**

**1-jest.**

**2-Enzyme.**