**REACT**

**WEEK-7 HANDSON-10 EXPLANATION**

**Define JSX**

JSX - which is short for JavaScript XML - is a language extension. JSX is that which gives the syntax of an extended language for JavaScript.

This syntax extension can be used for writing HTML-like code directly inside JavaScript using React.

JSX is optional, but encourages better reading and writing efficient React code.

JSX is compiled to React.createElement() calls by the tools like Babel.

Example:

const element = <h1>Hello, JSX!</h1>;

**Define ECMA Script**

It is JavaScript, not as defined on an official basis through ECMAScript (ES). So the different versions according to the way in which the features totally differ Sep, es5, es6 and others.

ES6 and other features import their own emojis like let, const, arrow functions, classes, promises, and modules.

Most often, modern React apps make use of ES6+ features.

Say what React.createElement() is.

The function JSX is compiled into is React.createElement().

It creates and returns a React element which defines what should be displayed on screen.

Its syntax is:

React.createElement(type, props, children)

Example:

const element = React.createElement("h1", { className: "title" }, "Hello React");

This is equal to:

<h1 className="title">Hello React</h1>

**Explain how to use JSX to create React nodes**

To create React nodes, you write JSX tags, which are components, elements, or fragments.

You can then assign the JSX to a variable or return it directly in the component.

For example:

const element = <div><h2>Welcome</h2><p>This is JSX</p></div>;

You can use custom components as JSX:

<MyComponent />

**Define how to render JSX to DOM**

JSX elements need to be rendered into a root DOM element wherein React does use the functions ReactDOM.render() or createRoot().render().

New Syntax (React 18+):

import ReactDOM from 'react-dom/client';

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<App />);

Old Syntax:

ReactDOM.render(<App />, document.getElementById('root'));

**Explain how to use JavaScript expressions in JSX**

You can use any JavaScript expression inside JSX by wrapping it with curly braces {}.

Examples:

const name = "Nikitha";

const element = <h1>Hello, {name}</h1>;

const total = 200 + 300;

const showTotal = <p>Total: {total}</p>;

Statements like if, for cannot be used directly in JSX.

You need to use ternary operators or array methods like map().

**Explain how to apply styles inline in JSX**

Inline styling is passed to the style prop as JavaScript objects.

CSS property names are written in camelCase and not kebab-case.

Example:

let styleObj = {

color: 'blue',

fontSize: '20px',

backgroundColor: 'lightgray'

};

Now, write code to create an element -<h2 style={styleObj}>Styled Text</h2>

You can also apply inline style as follows:

<h2 style={{ color: 'green', padding: '10px' }}>Inline Styling</h2>