React Fundamentals.

Introduction.

React is a popular JavaScript library for building user interfaces, particularly single - page applications. Developed by Facebook, React allows · developers to create large web applications that can change data without reloading the page. its main goal is to be fast, scalable, and simple.

Key Concept in React:

1. JSX (Java Script XML) 2. Components. 3. State. 4. Props (Properties).

JSX (Java Script XML)

JSX is a Syntax extension for JavaScript that looks similar to XML or HTML. it is used with React to describe what The UI should look like. JSX makes it easier to write and add HTML in React.

Example of JSX !-

```javascript
const element = <h1> Hell.world </h1>
```

Explanation :-

JSX produces React "element". Instead of separating technologies by putting markup and logic in separate files, React separates concerns with loosely coupled units called "components" that contain both. React. doesn't require using JSX, but most developers find it helpfull as a visual aid when working with UI · inside the JavaScript Code.

Components

Components are the building blocks of a React application. They can be thought of as custom HTML element, and they can be either class - based or function - based.

Functional Components

A functional Component is a JavaScript function that returns a React element.

Explanation

Components let you split the UI into independent, reusable pieces, and think about each piece in isolation. They can accept inputs called "props" and maintain internal State.

State

State is a built-in React object that is used to contain data or information about the component. A component's State can change over time; whenever it changes, the component re-renders.

Example of State

javascript

```
Class Clock extends React. Component}
 Constructor(Props) {
 Super(props);
 this. State = {date : new Date()};
 }
ComponentDid Mount() {
this.timerID = set Interval(
 () => this.tick(),
   1000
);

}
Component Will Unmount() {
ClearInterval(this.timerID);
}
```

```
tick {
 this. setState({
 date : new Date()
 });
 }
render() {
 return (
 <div>
 <h1>Hello, world!</h1>
 <h2>It is
{this.State.date.toLocal Time
String()}.</h2>
 </div>
 );
}     }
```

Explanation

In this example, the Clock component maintain a Private State object (date). The tick method updates the state every second, and React automatically updates the DOM to match the new state.

Props (Properties)

Props are arguments passed into React components. Props are passed to components via HTML attributes.

Example of Props

```
javascript
Function welcome (props) {
 return <h1> Hello, {props.name} </h1>;     }
```

Using Props in a Component

```
javascript
Const element = <welcome name="Sara" />;
```

Explanation

In this example, the welcome component receives a name props and uses it to render a greeting message. Props are read-only, which means that a component cannot change its own props.

JavaScript

```
function welcom (props) {
    retur <h1> Hello , { props.name } </h1>;
  }
```

class Components

A class Component requires you to. extend from React . Component and create a render function that returns a React element .

JavaScript

```
class welcom extends React . Component {
   render () {
     return <h1> Hello, { this . props . name } </h1>
    }
      }
```