Fundamental concepts of React

React is a JavaScript library for building user interfaces, particularly single-page applications where efficient updating and rendering of components is crucial. Below are the core concepts:

1. JSX (JavaScript XML)

concept:

JSX is a syntax extension for JavaScript that allows you to write HTML-like code within JavaScript. It makes it easier to create and visualize the structure of the UI.

Example:

```
const element = <h1>Hello, world!</h1>
```

Explanation:

JSX allows you to write elements in a syntax that looks similar to HTML.

The above JSX code will be transpiled to a React.createElement function call, which returns a JavaScript object representing the element.

This object can then be rendered to the DOM by React.

const element = React.createElement('h1', null, 'Hello, world!');

2. Components

Concept:

Components are the building blocks of a React application. A component is a self-contained module that renders some output. You can think of components as JavaScript functions that return HTML elements.

Types:

Functional Components: These are simple JavaScript functions.

Class Components: These are ES6 classes that extend from React.Component.

Example of a functional component:

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

Example of a class component:

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

Explanation:

components can accept inputs called PROPS and return React elements that describe what should appear on the screen.
components can be nested, managed, and handled independently.
3. State
concept:
State is a built-in React Object that is used to contain data or information about the component. A components state can change over time whenever it changes, the component re-renders.

Example:

```
class Clock extends React.Component
constructor(props)
super(props)
this.state = {date: new Date()}


componentDidMount()
this.timerID = setInterval() => this.tick(),
1000)
```

```
componentWillUnmount()
clearInterval(this.timerID)

tick()
this.setState({
date: new Date()
});


render()
return (
<div>
<h1>Hello, world!</h1>
<h2>It is
{this.state.date.toLocaleTimeString()}.</h2>
</div>
);
```

Explanation:

The clock component maintains its state using the this.state object.
The state is initialized in the constructor.
The setState method is used to update the state.
The component re-renders every second due to

the state update.

4. PROPS

concept:

PROPS (short for properties) are a way of passing data from parent to child components. PROPS are read-only and cannot be modified by the child component.

Example:

```
function welcome(PROPS)
return <h1> Hello, PROPS.name</h1>


function APP()
return (
<div>
<welcome name=sara />
<welcome name=canal />
<welcome name=Edite />
</div>
)
```

Explanation:

The welcome component is used multiple times in the APP component, each time with different PROPS.

PROPS are passed to the component as

attributes.
The welcome component reads the name PROP and displays it.
HOW These Concepts contribute to BUILDING a React APPLICATION
JSX:

simplifies the creation of react elements and components.
makes the code more readable and easier to write.
components:

Encourage modularity and reusability.
Enable complex UIs to be broken down into smaller, manageable pieces.
State:

manages dynamic data within a component.
Facilitates interactive and real-time updating of the UI.
PROPS:

Allows data to be passed between components.
Enables components to be dynamic and reusable with different data.
Putting It All Together
Below is a complete example that uses all these concepts to build a small React

application:

jax code:

```
import React from react
import ReactDOM from react-dom

function welcome(PROPS)
return < h1> Hello, PROPS.name< h1>


class clock extends React.component
constructor(PROPS)
super(PROPS)
this.state =  date:  new Date()


componentDidmount()
this.timerID = setInterval() => this.tick(),
1000


componentwillunmount()
clearInterval(this.timerID)


tick()
this.setstate(
date:  new Date()
)
```

```
render()
return (
<div>
<h1> Hello, world!</h1>
<h2> It is
this.state.date.toLocaleTimeString().</h2>
</div>
);
}

function App() {
return (
<div>
<Welcome name="Sara" />
<Welcome name="Cahal" />
<Welcome name="Edite" />
<Clock />
</div>
);
}

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

Explanation:

The welcome component is a functional

component that displays a greeting message.
The Clock component is a class component that displays the current time and updates it every second.
The APP component combines the Welcome and Clock components, demonstrating how PROPS and state work together in a React application.
The ReactDOM.render function renders the APP component into the DOM.
This comprehensive example illustrates how JSX, components, state, and PROPS are used to build a React application, showcasing the modularity and reusability of React components.