Batch 3 - React Core Notes (by Re-Re Sudhan)

STATE (Definition & Usage):

State is a JavaScript object owned by a component which stores dynamic data. When state changes via setState(), React re-renders the component.

Example:

```
class Counter extends React.Component {
  constructor(props){
    super(props);
    this.state = { count: 0 };
  }
  render(){ return <h1>{this.state.count}</h1>; }
}
```

SETSTATE (Definition & Patterns):

setState() updates component state and triggers re-render. Never mutate this.state directly.

Object form:

```
this.setState({count: 5});
```

Updater function (safe):

```
this.setState(prev => ({count: prev.count + 1}));
```

STATE vs VARIABLE:

State (this.state): affects UI, persistent, updated via setState.

Variable (let/const): does not affect UI, updated directly, no re-render.

Example:

```
class Example extends React.Component {
  state = { count: 0 };
  clicks = 0;

  handle = () => {
    this.setState({ count: this.state.count + 1 });
    this.clicks++;
  }

  render(){ return <div>State: {this.state.count}, Var: {this.clicks}</div>; }
}
```

CLASS COMPONENT STRUCTURE:

```
class MyComponent extends React.Component {
  constructor(props){
    super(props);
    this.state = { name: 'Re-Re Sudhan' };
  }

  componentDidMount(){ /* API calls */ }
  componentDidUpdate(){ /* respond to updates */ }
  componentWillUnmount(){ /* cleanup */ }

  render(){ return <div>{this.state.name}</div>; }
}
```

Key Parts: constructor, super(props), state, render, lifecycle methods.

CONSTRUCTOR & SUPER:

Constructor runs first when component created, used to init state and bind methods.

super(props) calls parent constructor to make this.props usable inside.

PROPS (Definition & Example):

Props are read-only inputs from parent to child.

Example:

```
class Student extends React.Component {
  render(){ return <h1>{this.props.name}</h1>; }
}
// Parent: <Student name='Re-Re Sudhan' />
```

DEFAULT PROPS:

Default values used if parent does not pass props.

```
class Welcome extends React.Component {
  render(){ return <h1>Welcome {this.props.user}</h1>; }
}
Welcome.defaultProps = { user: 'Guest' };

<Welcome /> => Welcome Guest
<Welcome user='Sudhan'/> => Welcome Sudhan
```

STATE vs PROPS:

State: internal, mutable via setState, owned by component.

Props: external, read-only, passed from parent.

Interview Quick Lines:

- State is component-owned dynamic data.
- setState schedules updates and re-renders.
- Constructor initializes state; super(props) makes props available.
- Props are read-only inputs; defaultProps provide fallback.