Components

Functional component

```
const FunctionalComponent = props => (
  <button className="content">Hello world</button>
);
```

Functional component

```jsx
const FunctionalComponent = props => (
  <button className="content">{props.text}</button>
);

const OtherComponent = () => <FunctionalComponent text="Hello world" />;

const str = "Lalala";
const OtherComponent2 = () => <FunctionalComponent text={str} />
```

Functional component

```
const FunctionalComponent = props => {
  const greeting = "Hello" + props.name;

  return <button className="content">{greeting}</button>;
};
```

Object destructuring

```
const object = {
  key1: 'val1',
  key2: 1234,
  key3: {
    innerKey1: 4321
  },
};

const { key1, key2 } = object; // key1 === 'val1', key2 === 1234
const { key3: { innerKey1 }} = object; // innerKey1 === 4321

const { key1: renamedKey1 } = object; // renamedKey1 === 'val1'
```

Functional component

```jsx
const FunctionalComponent = ({ name }) => {
  const greeting = "Hello" + name;

  return <button className="content">{greeting}</button>;
};
```

Class component

```
class ClassComponent extends React.Component {
  render() {
    const extendedName = this.props.name + " the developer";

    return (
      <div className="app">
        <FunctionalComponent name={extendedName} />
      </div>
    );
  }
}
```

Class component

```
class ClassComponent extends React.Component {
  state = {
    message: ""
  };

  handleGreetingClick = () => {
    this.setState({ message: "Greeting was clicked!" });
  };

  render() {
    const extendedName = this.props.name + " the developer";

    return (
      <div className="app">
        <FunctionalComponent
          name={extendedName}
          onClick={this.handleGreetingClick}
        />
        <div>{this.state.message}</div>
      </div>
    );
  }
}
```

Class component

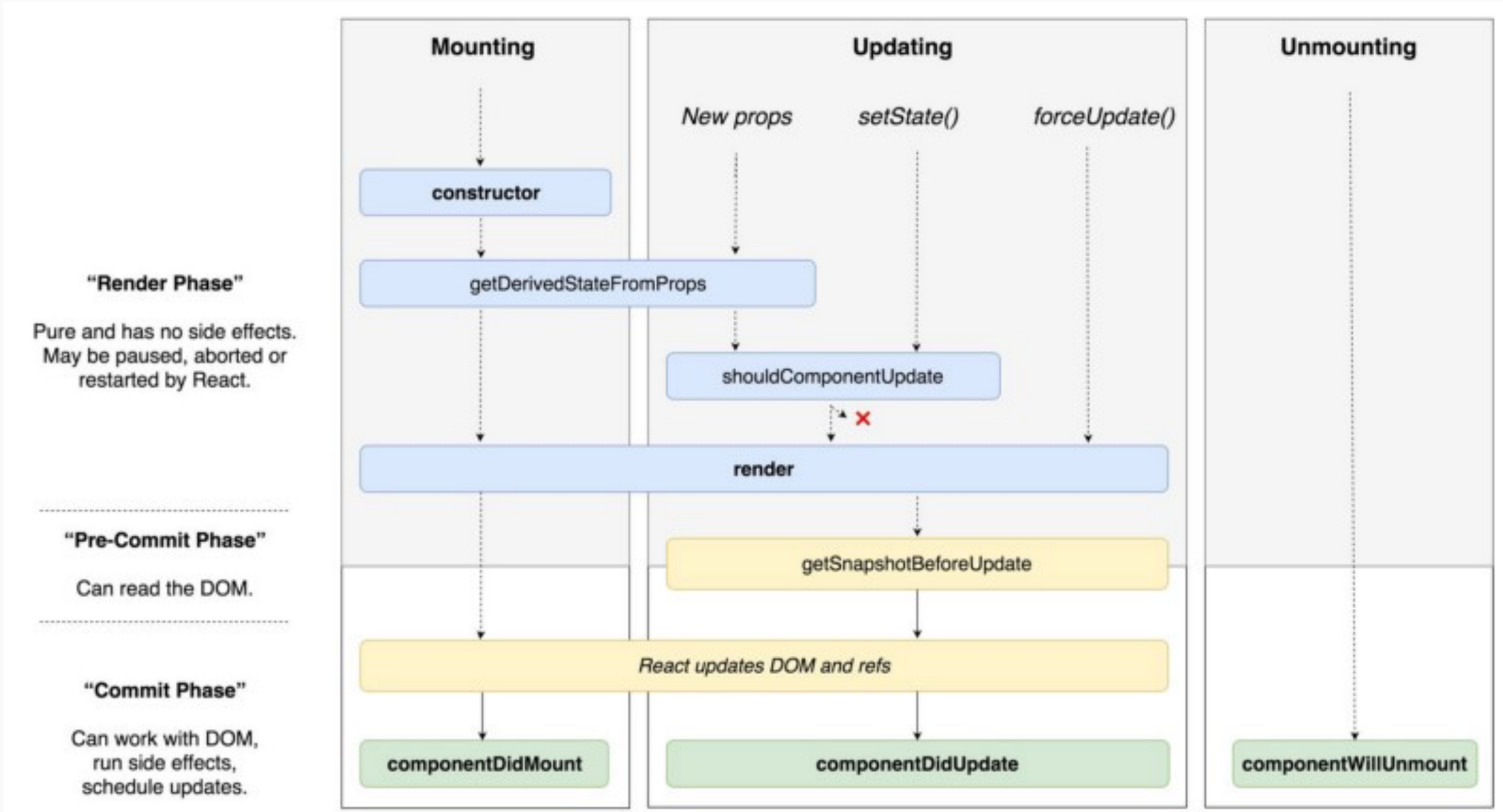
```
this.setState(object);
```

```
this.setState(state => {
  // do smth with access to state

  return newStateObeject;
});
```

Реализация counter’a

Методы жизненного цикла


"Render Phase"
Pure and has no side effects. May be paused, aborted or restarted by React.
"Pre-Commit Phase"
Can read the DOM.
"Commit Phase"
Can work with DOM, run side effects, schedule updates.
Mounting
Updating
Unmounting
New props
setState()
forceUpdate()
constructor
getDerivedStateFromProps
shouldComponentUpdate
render
getSnapshotBeforeUpdate
React updates DOM and refs
componentDidMount
componentDidUpdate
componentWillUnmount

Class component

```
class ClassComponent extends React.Component {
  state = {
    message: ''
  };

  componentDidMount() {
    setTimeout(() => {
      this.setState({ message: 'Message appeared!' });
    }, 2000);
  }

  handleGreetingClick = () => {
    this.setState({ message: "Greeting was clicked!" });
  };

  render() {
    // render
  }
}
```

Реализация электронных часов

```
const date = new Date();

setInterval(foo, timeInMilliseconds);

setInterval(() => { /* do smth */ }, 1000);

// life cycle method
componentDidMount() { /* do smth */ }
```

Inputs

```
class ClassComponent extends React.Component {
  render() {
    const extendedName = this.props.name + " the developer";

    return (
      <div className="app">
        <FunctionalComponent name={extendedName} />
      </div>
    );
  }
}
```

```jsx
const MyInput = ({ value, onChange }) => (
  <input className="styled-input" value={value} onChange={onChange} />
);

class Container extends React.Component {
  state = {
    inputValue: ""
  };

  handleChange = event => {
    const newValue = event.target.value;
    // or const { value } = event.target;

    this.setState({ inputValue: newValue });
  };

  submit = () => {
    sendDataToServer(this.state.inputValue);
  }

  render() {
    return (
      <div>
        <MyInput
          value={this.state.inputValue}
          onChange={this.handleChange}
         />
        <button onClick={this.submit}>SUBMIT</button>
      </div>
    );
  }
}
```