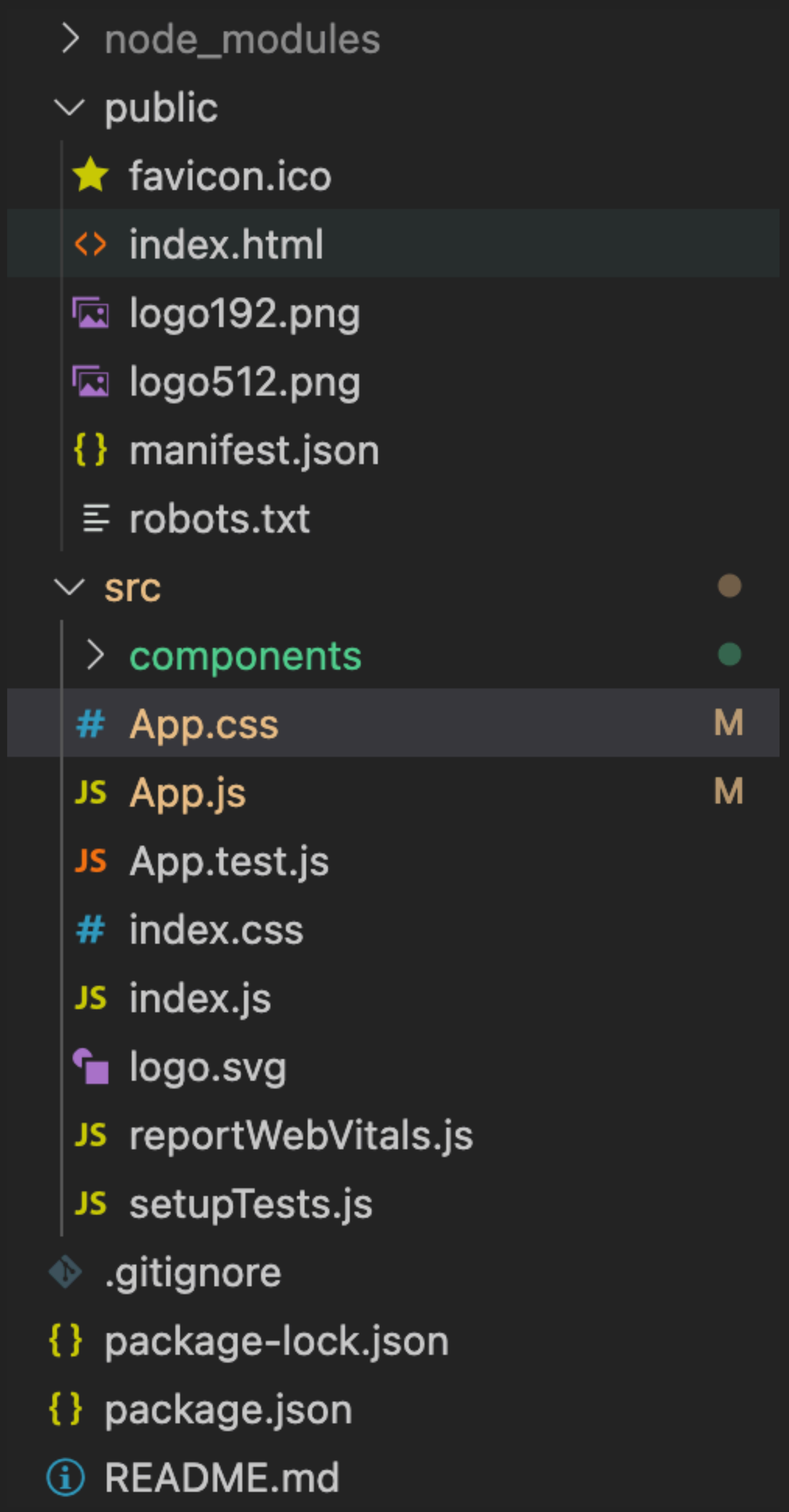
INTRODUCTION TO

REACTJS

SETTING UP REACTJS

- Assuming you have install node v14
- $ npx create-react-app my-app
- $ cd my-app
- $ npm start



```
> node_modules
v public
    favicon.ico
    index.html
    logo192.png
    logo512.png
    manifest.json
    robots.txt
v src
  > components
    App.css          M
    App.js           M
    App.test.js
    index.css
    index.js
    logo.svg
    reportWebVitals.js
    setupTests.js
  .gitignore
  package-lock.json
  package.json
  README.md
```

REACT COMPONENT

```
import React, {Component} from 'react';

class HelloWorld extends Component {
    render() {
        return (
            <div className="title">
                Hello World!
            </div>
        );
    }
}

export default HelloWorld;
```

REACT COMPONENT USING CREATE ELEMENT

```
import React, {Component} from 'react';

class HelloWorldReactElement extends Component {
    // This is just an example, use JSX syntax for better readability
    render() {
        return React.createElement('div', null, 'Hello World!');
    }
}

export default HelloWorldReactElement;
```

REACT FUNCTION COMPONENT

```
import React from 'react';

const HelloWorldFunction = props => {
    // Use this for simple stateless components
    return (
        <div>
            Hello World!
        </div>
    );
};

export default HelloWorldFunction;
```

REACT STATE

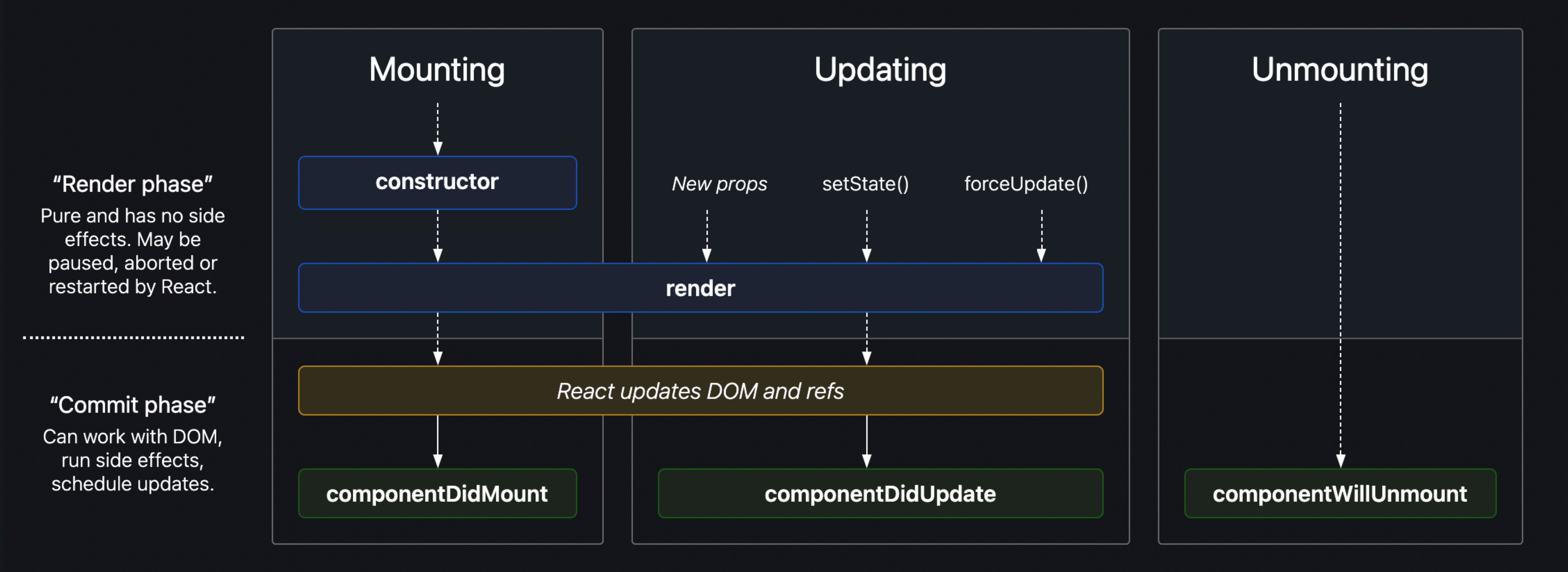
```
import React, {Component} from 'react';

class HelloWorldWithState extends Component {
    state = {
        name: "React"
    }

    handleOnNameChange = (e) => {
        this.setState({name: e.target.value})
    }

    render() {
        return (
            <>
                <div>
                    Hello World from <b>{this.state.name}</b>
                </div>
                <div>
                    <input type="text" onChange={this.handleOnNameChange} placeholder=
"From name" value={this.state.name}/>
                </div>
            </>
        );
    }
}

export default HelloWorldWithState;
```

REACT PROPS

```
import './App.css';
import HelloWorldWithProps from
"./components/HelloWorldWithProps";

function App() {
    return (
        <HelloWorldWithProps name="React"/>
    );
}

export default App;
```

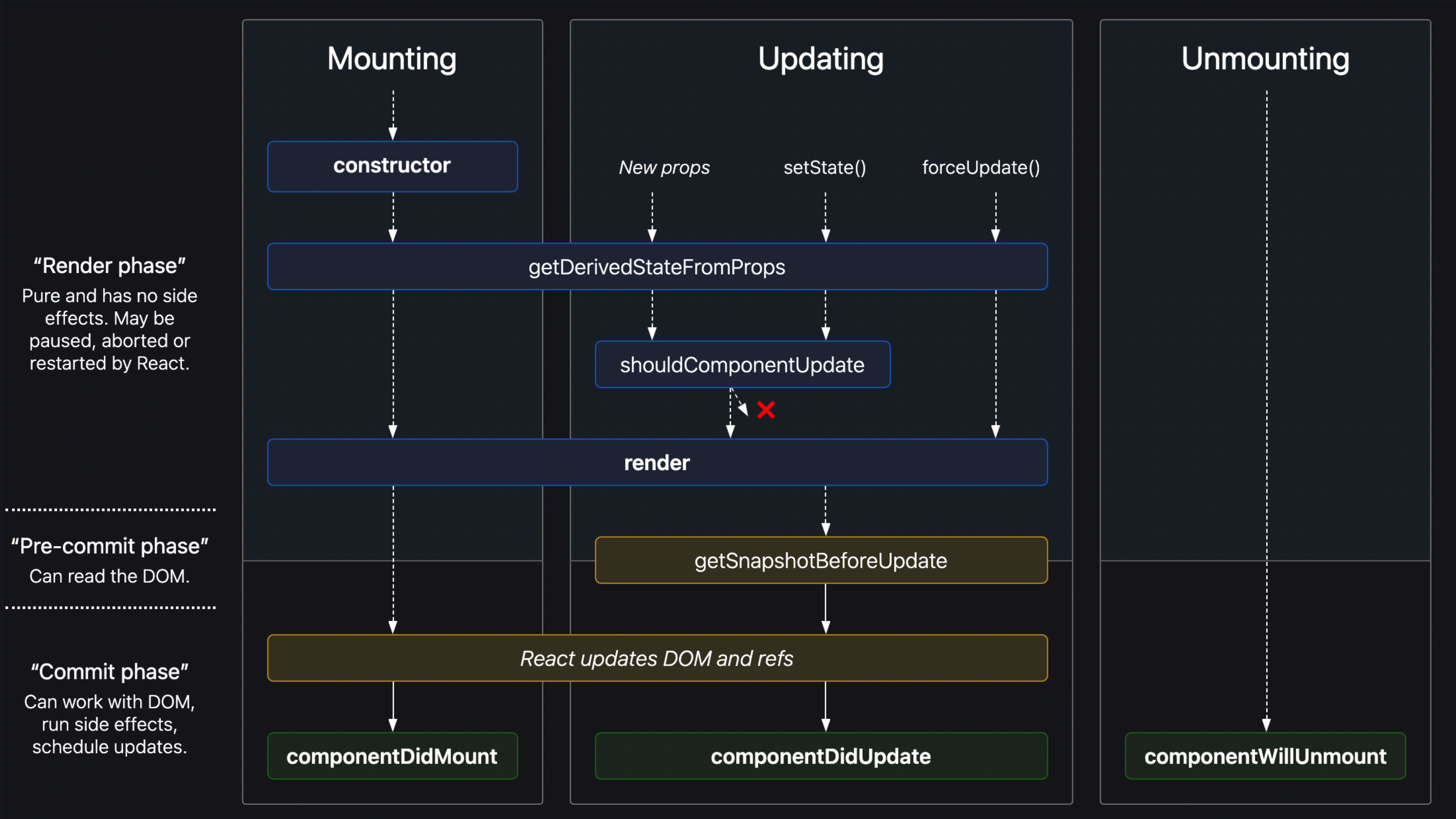
```
import React, {Component} from 'react';

class HelloWorldWithProps extends Component {

    render() {
        return (
            <div>
                Hello World from <b>{this.props.name}</b>
            </div>
        );
    }
}

export default HelloWorldWithProps;
```

REACT COMPONENT LIFECYCLE



Reference: https://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

LESS COMMONLY USED LIFECYCLE



API CALLS USING COMPONENTDIDMOUNT

```
import React, {Component} from 'react';

class HelloWorldApi extends Component {

    state = {
        jsFrameworks: []
    }

    constructor(props) {
        super(props);
    }

    async componentDidMount() {
        const r = await fetch("./jsFrameworks.json");
        const jsFrameworks = await r.json();
        this.setState({jsFrameworks});
    }

    render() {
        return (
            <>
                <div>
                    {this.state.jsFrameworks.length === 0
                        && 'No frameworks to show'}
                </div>
                <div>
                    {this.state.jsFrameworks.map(f => {
                        return <div>
                            <a href={f.url}>{f.name}</a>
                        </div>
                    })}
                </div>
            </>
        );
    }
}
```

TICKER USING COMPONENTWILLUNMOUNT

```
import React, {Component} from 'react';

class Ticker extends Component {

    state = {
        cancelInterval: null,
        tick: 0
    }

    componentDidMount() {
        const cancelInterval = setInterval(() => {
            this.setState(({tick}) => ({tick: tick + 1}));
        }, 1000);
        this.setState({cancelInterval});
    }

    componentWillUnmount() {
        window.alert('Clearing interval');
        clearInterval(this.state.cancelInterval);
    }

    render() {
        return (
            <h1>
                {this.state.tick}
            </h1>
        );
    }
}
```

HELLOWORLD USING COMPONENTDIDUPDATE

```
import React, {Component} from 'react';

class HelloWorldWithDidUpdate extends Component {

    state = {
        name: null
    }

    componentDidUpdate(prevProps) {
        // Typical usage (don't forget to compare props):
        if (this.props.name !== prevProps.name) {
            this.setState({name: prevProps.name});
        }
    }

    render() {
        return (
            <div>Hello World from {this.state.name}</div>
        );
    }
}

export default HelloWorldWithDidUpdate;
```

CLEAN CODE

1. Don’t repeat your self DRY
2. Descriptive function/variable names
3. Short functions 5 lines, 3 arguments
4. Avoid global variables as much as possible
5. Use syntax checks like eslint with airbnb
6. File should not exceed 200 lines
7. If you are not able to write test with ease you are writing bad code
8. Use functional paradigm and immutability
9. Keep it simple and stupid

REACT TESTING LIBRARY

```
import {render, screen} from '@testing-library/react';
import HelloWorld from './HelloWorld';

test('renders hello world', () => {
    render(<HelloWorld/>);
    const helloWorldText = screen.getByText(/Hello World!/i);
    expect(helloWorldText).toBeInTheDocument();
});
```

```
import {render, screen, fireEvent} from '@testing-library/react';
import HelloWorldWithState from './HelloWorldWithState';

test('greets by name in the input box', () => {
    render(<HelloWorldWithState/>);
    const inputBox = screen.getByTestId('name-input-box');
    fireEvent.change(inputBox, {target: {value: '273'}});
    expect(inputBox.value).toBe('273');
});
```

```
import React from 'react'
import {rest} from 'msw'
import {setupServer} from 'msw/node'
import {render, waitFor, screen} from '@testing-library/react'
import '@testing-library/jest-dom/extend-expect'
import HelloWorldApi from './HelloWorldApi'

const server = setupServer(
    rest.get('/jsFrameworks.json', (req, res, ctx) => {
        return res(ctx.json([{name: 'ReactJS', url: 'http://react.com'}]))
    })
)

beforeAll(() => server.listen())
afterEach(() => server.resetHandlers())
afterAll(() => server.close())

test('loads and displays list of JavaScript frameworks', async () => {
    render(<HelloWorldApi/>);
    const nothingToShow = screen.getByText(/No frameworks to show/i);
    expect(nothingToShow).toBeInTheDocument();
    await waitFor(() => screen.getByText(/ReactJS/i))
    expect(screen.getByText('ReactJS').href).toBe("http://react.com/")
})
```