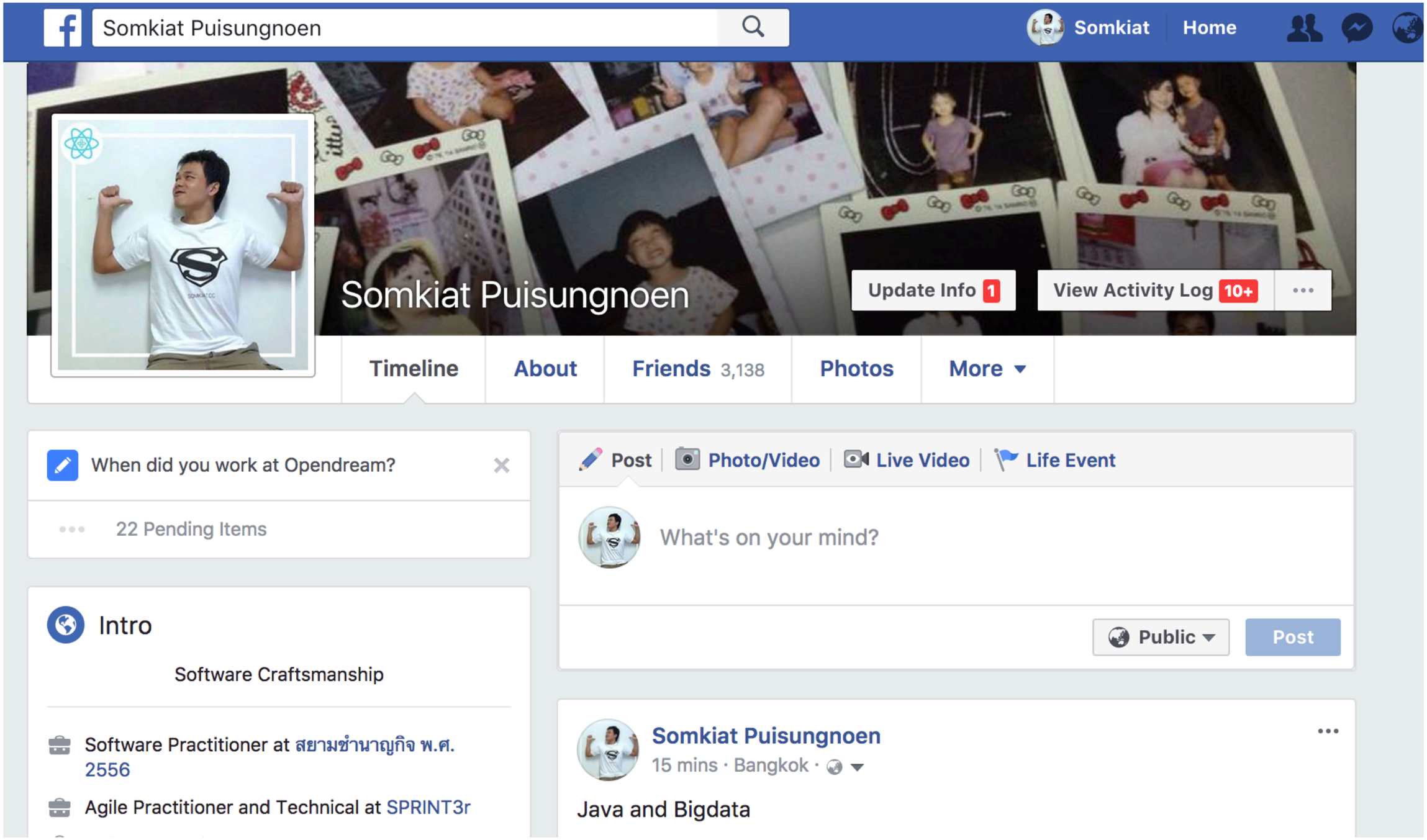
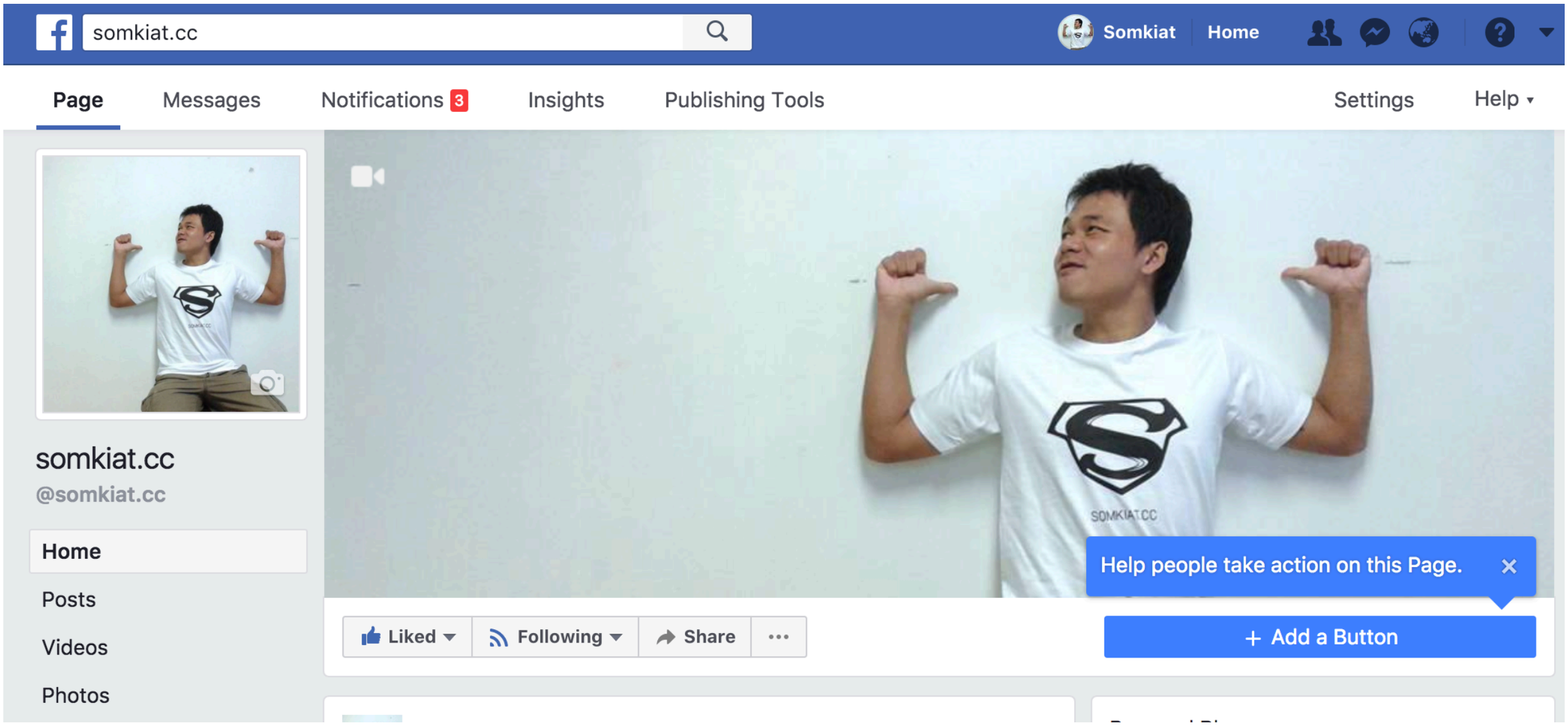
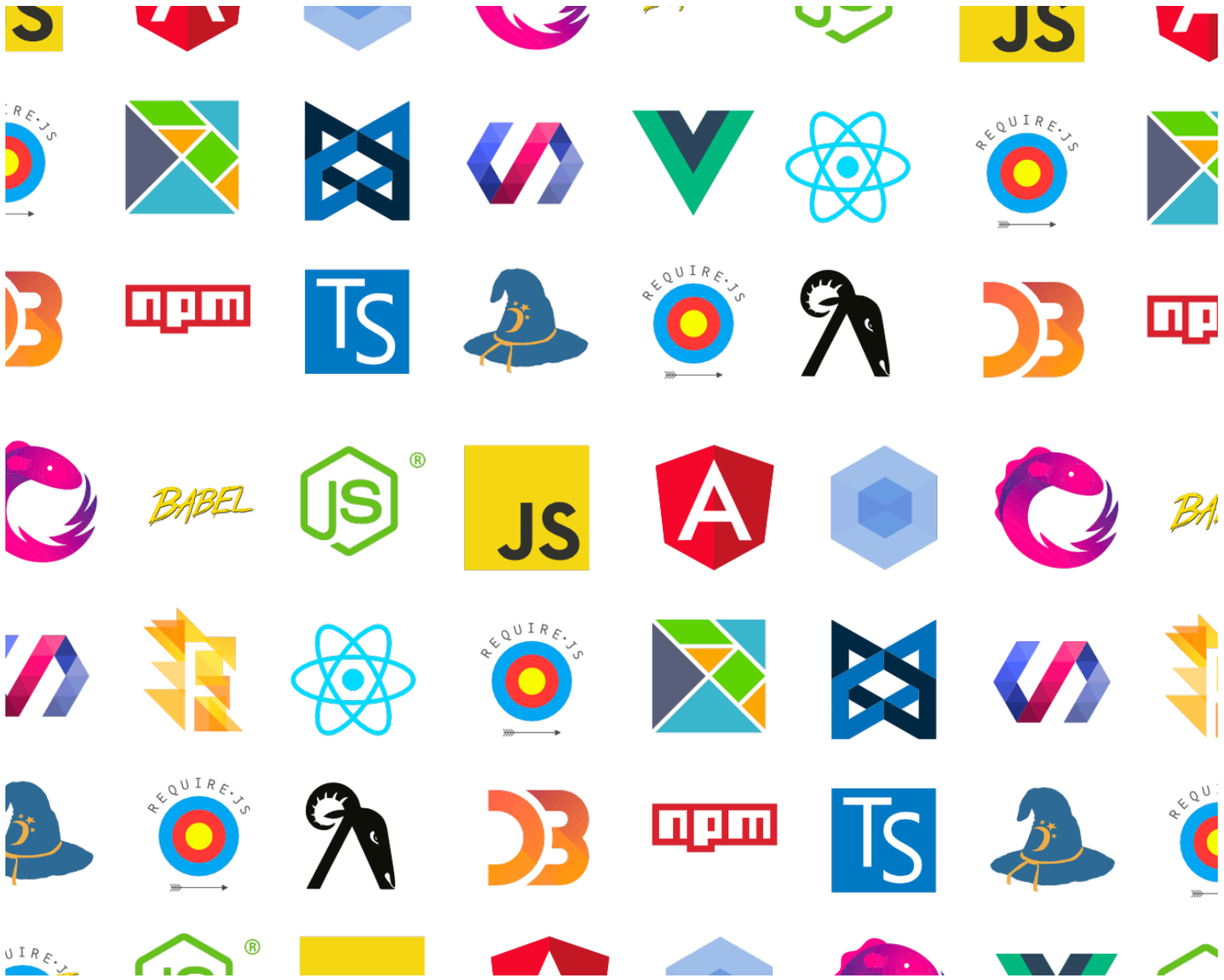
Learning React

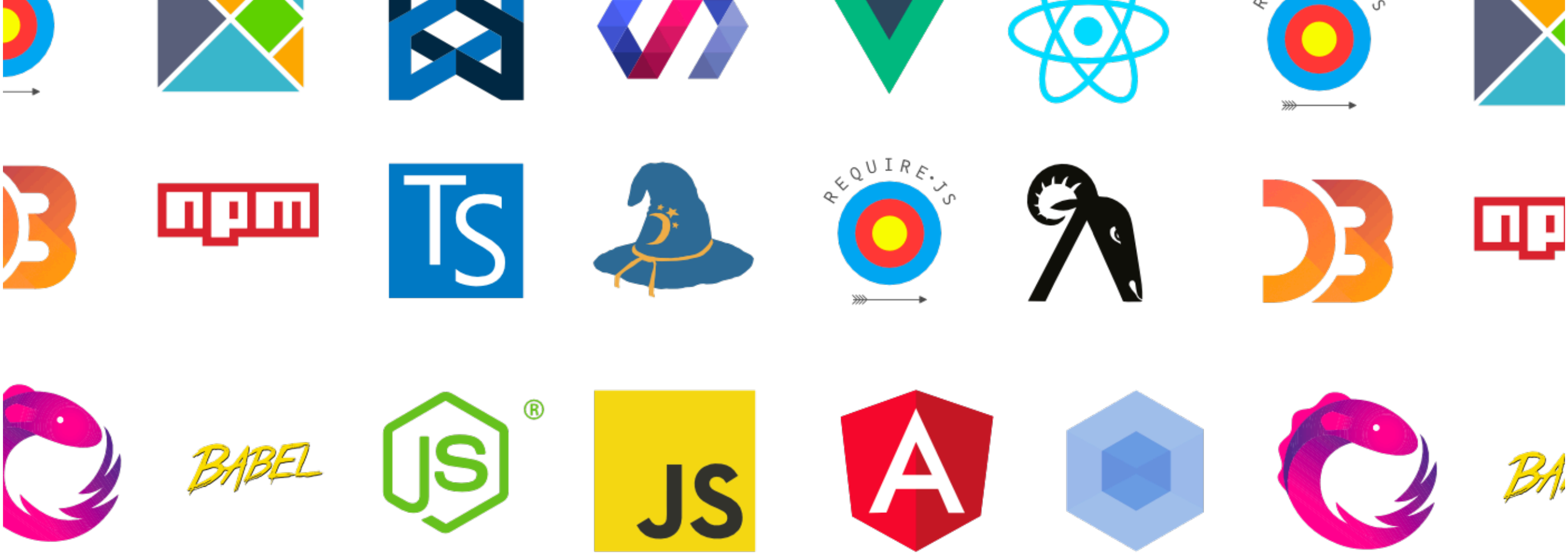
Basic of React



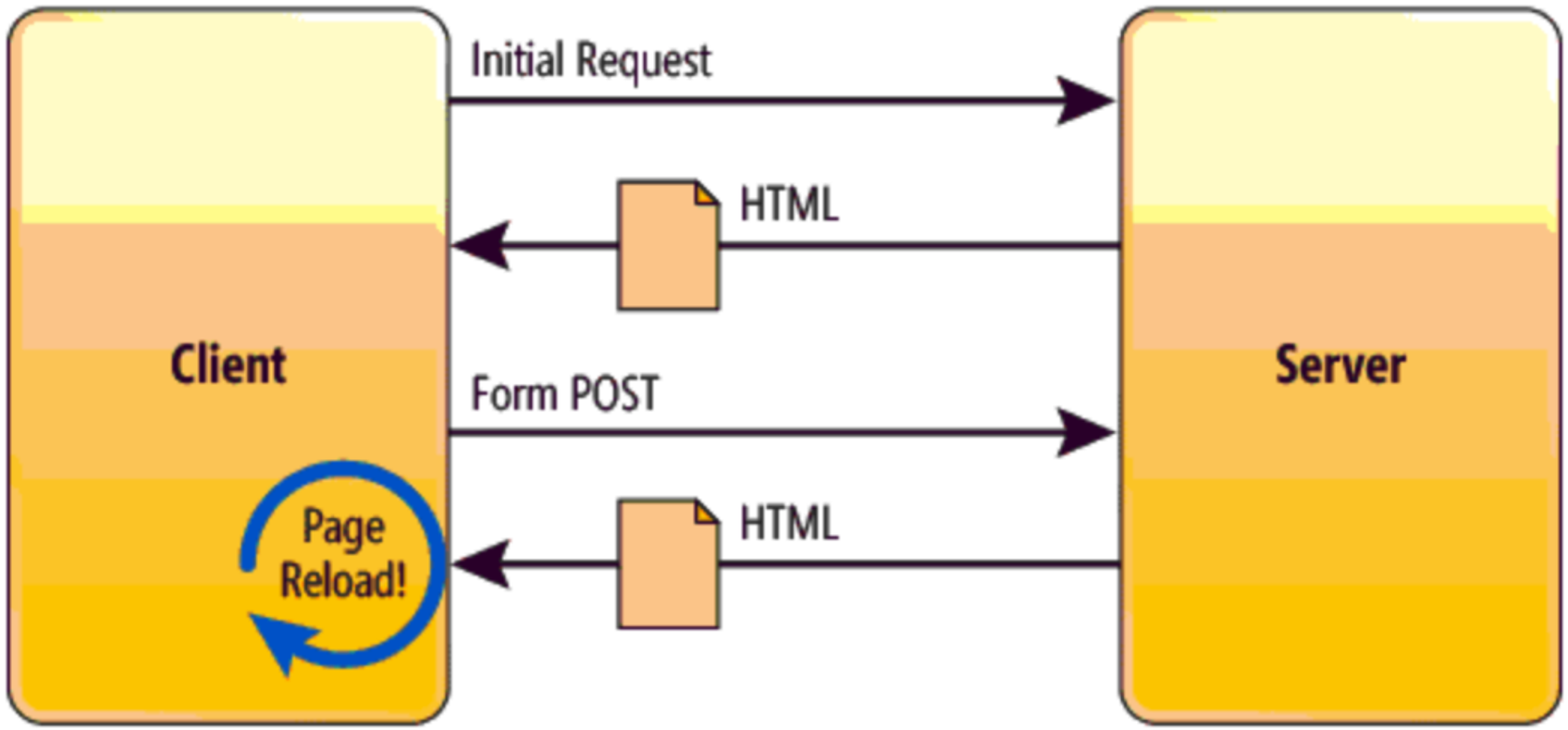


JavaScript

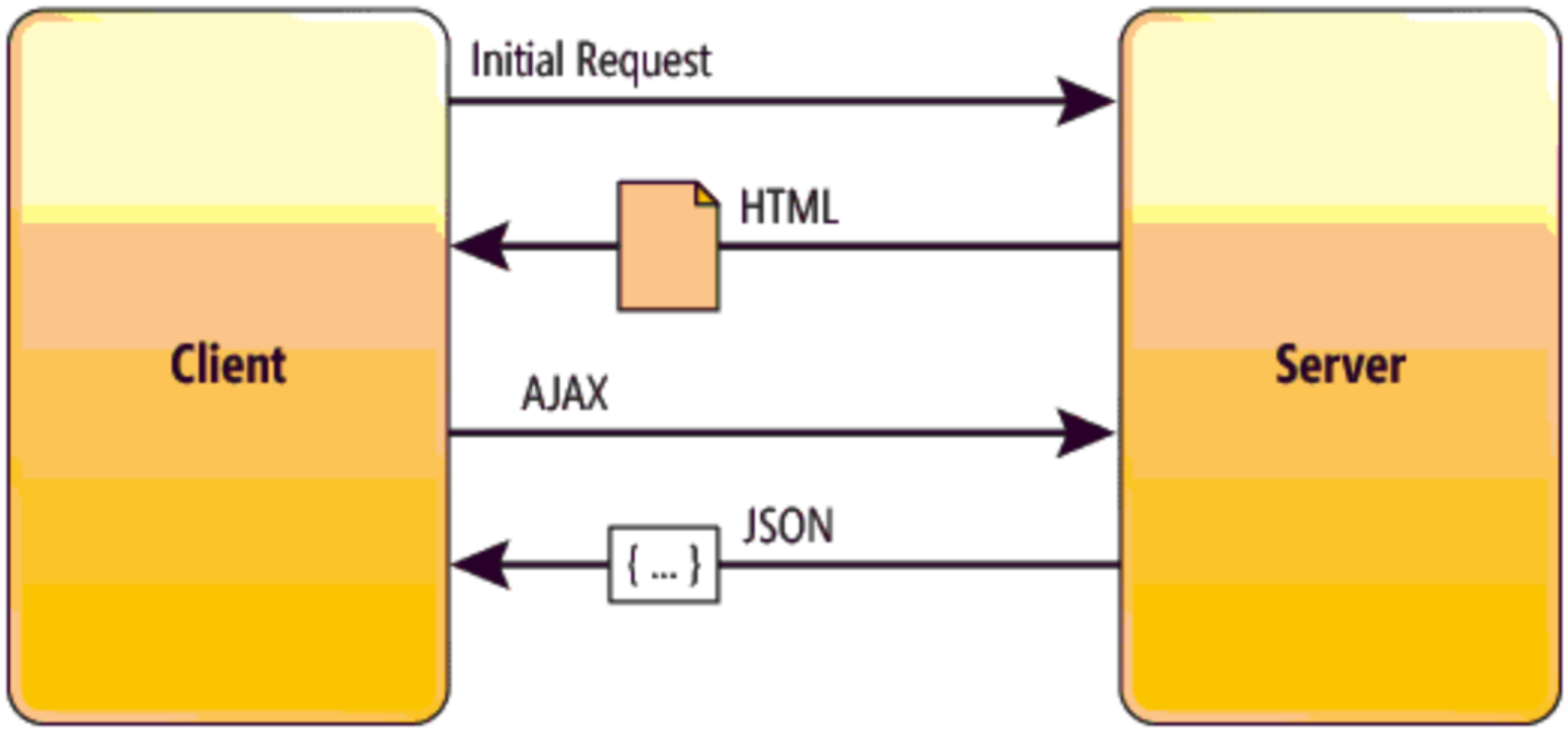




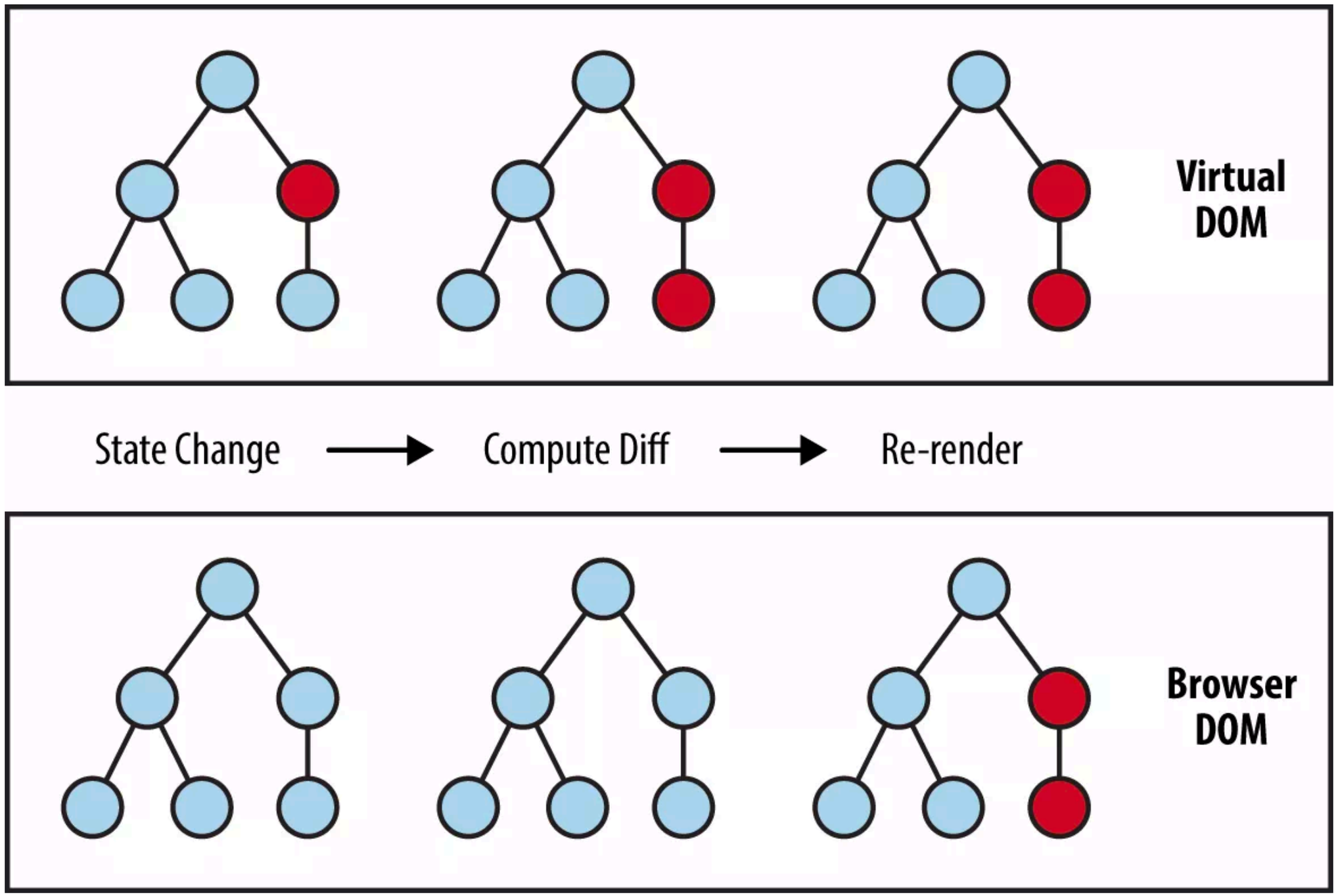
Traditional



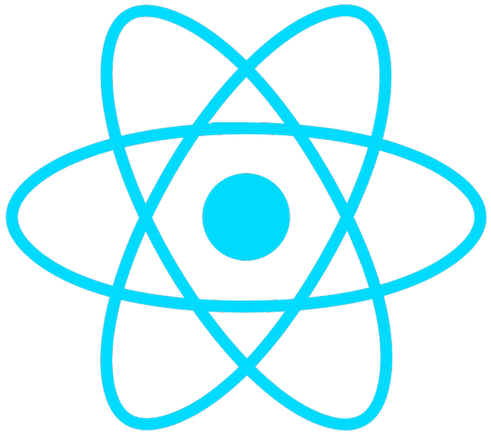
Single Page Application (SPA)



DOM vs Virtual DOM



React and Angular



Learning path

Develop

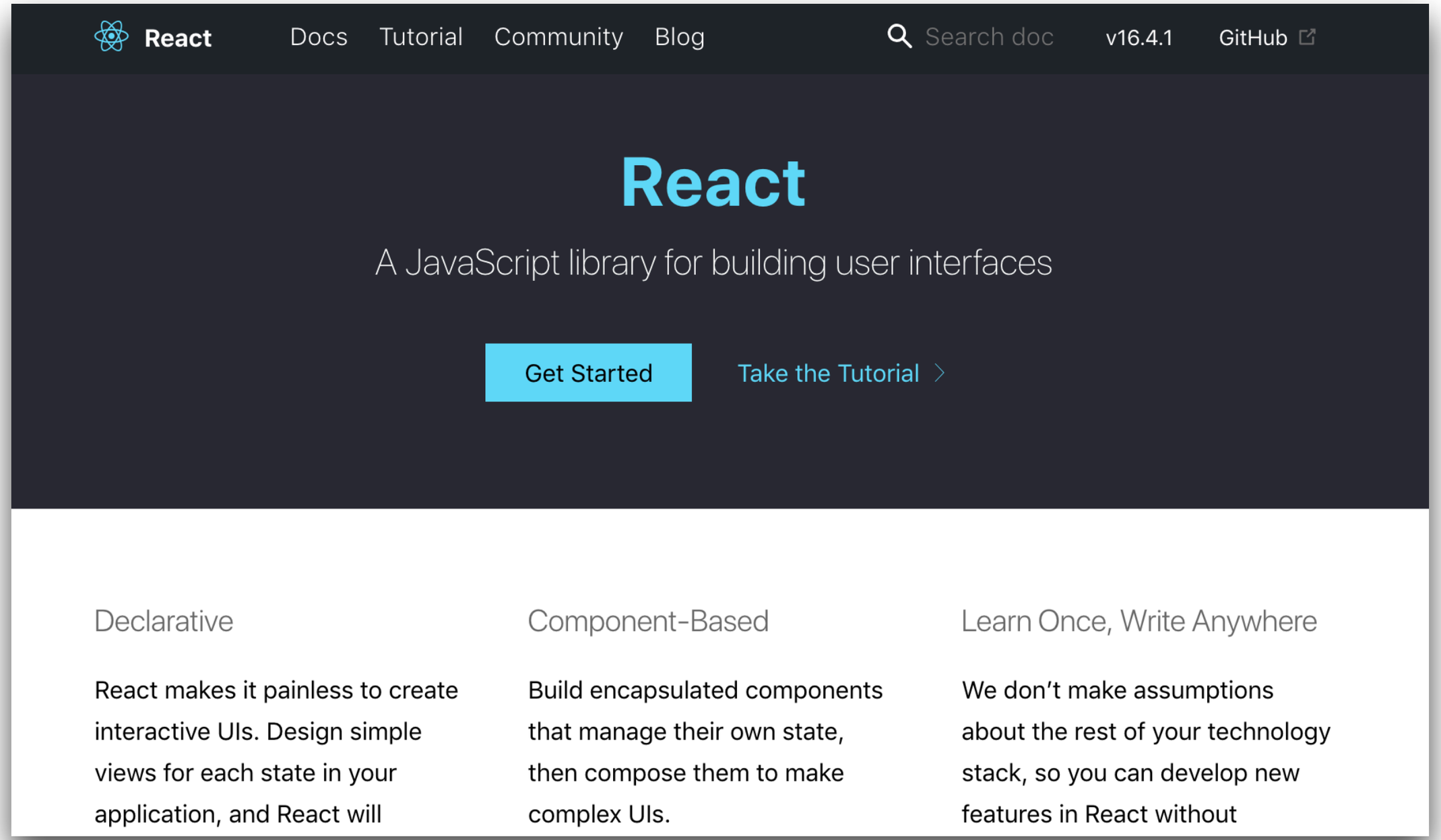
Build

Deploy

Basic of React

1. Introduction to React
2. Installation
3. Create a react application
4. Design and Develop application
5. Deploy application to Github Pages
6. Working with Hooks
7. Workshop

React



https://reactjs.org/

What is React ?

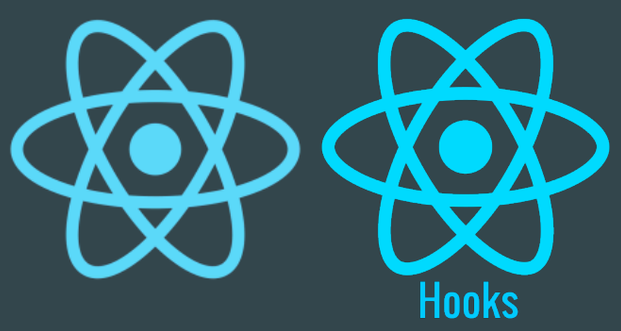
Open source javascript **library** to develop UI
Introduce by **Facebook** on May 2013
Open source on May 2015

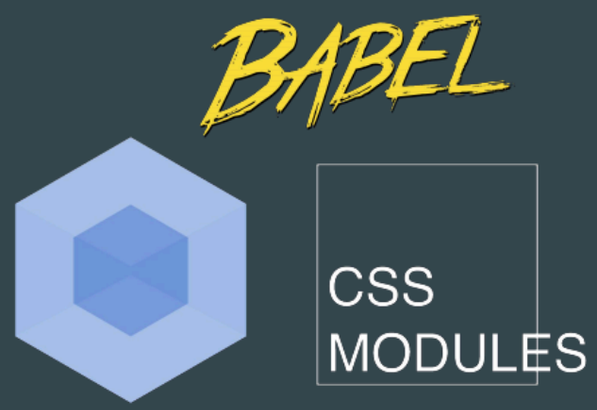
What is React ?

React is concerned with the **components**
Javascript + CSS + HTML

https://reactjs.org/docs/getting-started.html

React ecosystem

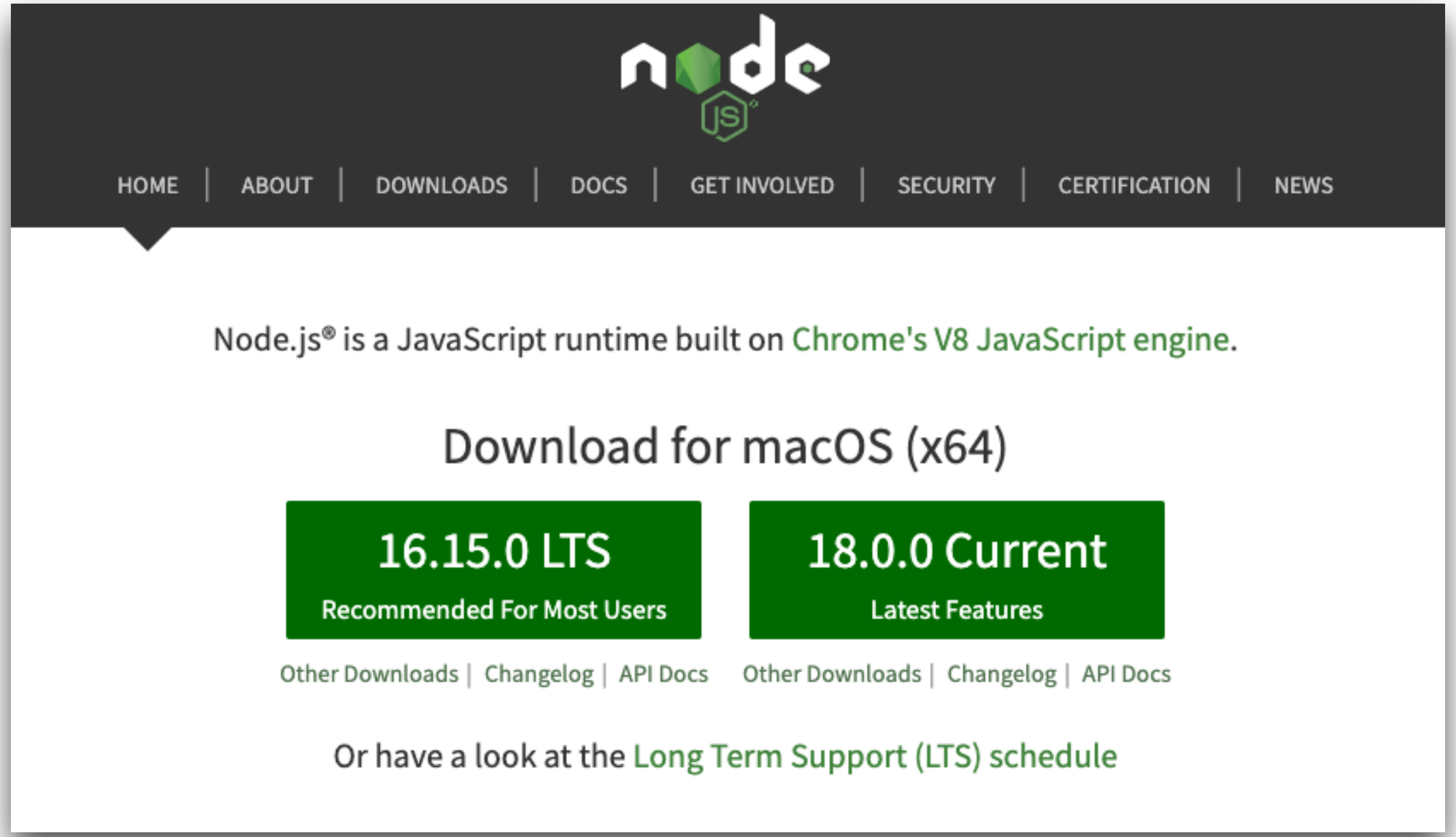




React	State management	Tools
JSX, State/Props, HoC, Routing	Redux, Mobx	Babel, Webpack, CSS Modules, Testing

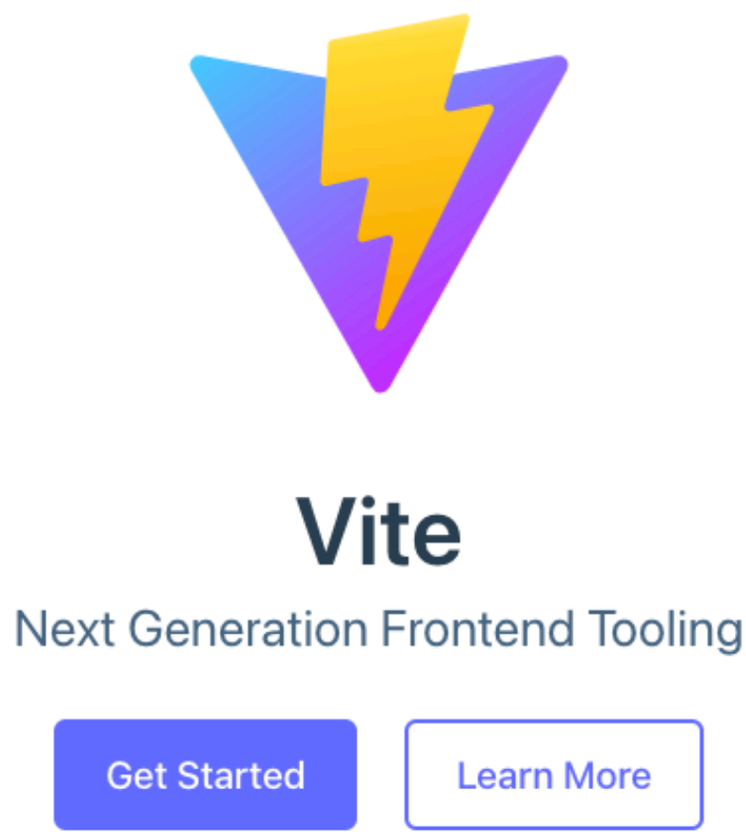
Installation

Installation Node.js



https://nodejs.org/en/

Installation with Vite



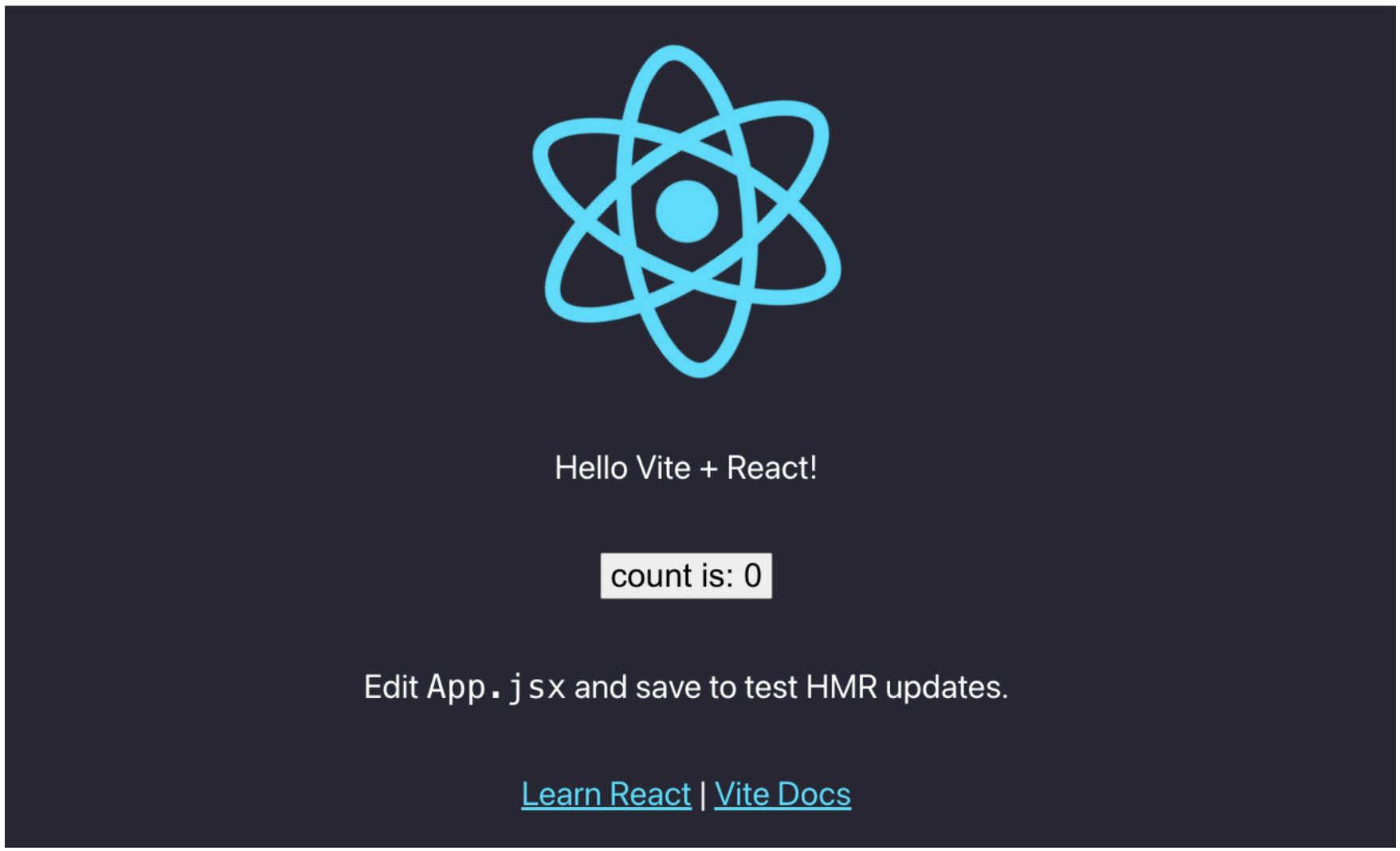
https://vitejs.dev/

Create React project with Vite

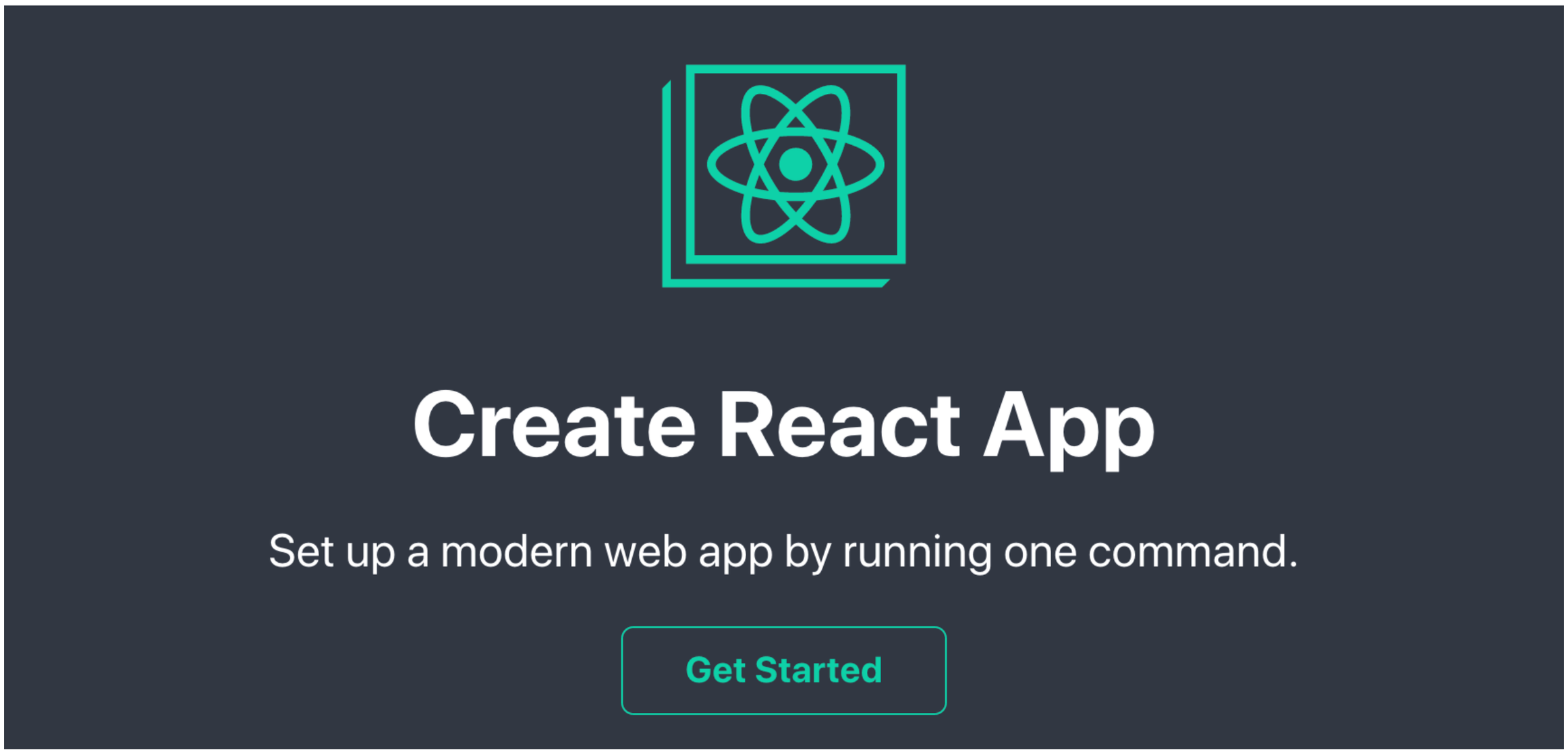
```
$npm create vite@latest demo01
$cd demo01
$npm install
$npm run dev
```

Hello React App

http://localhost:3000/

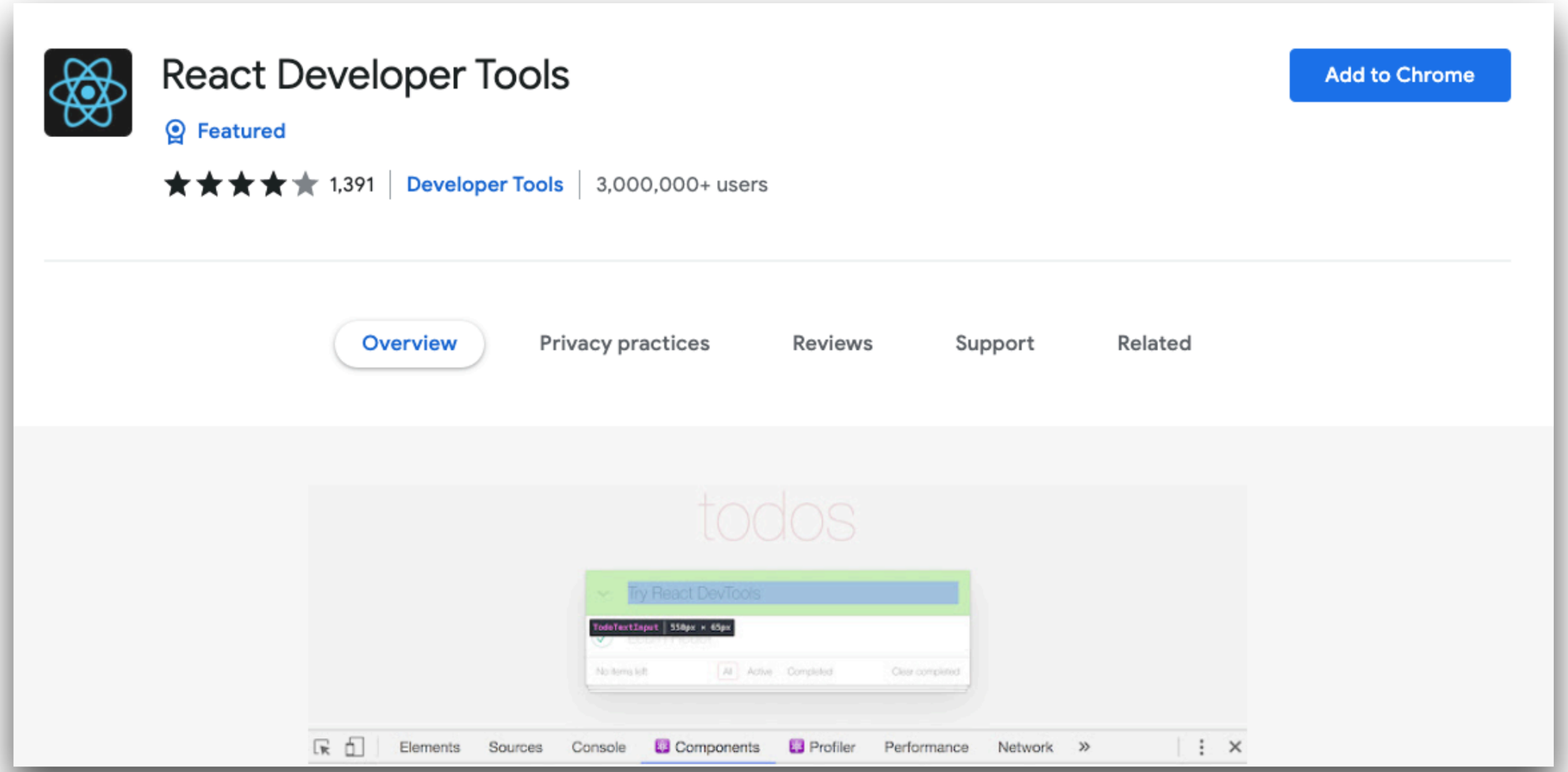


Create React App



https://create-react-app.dev/

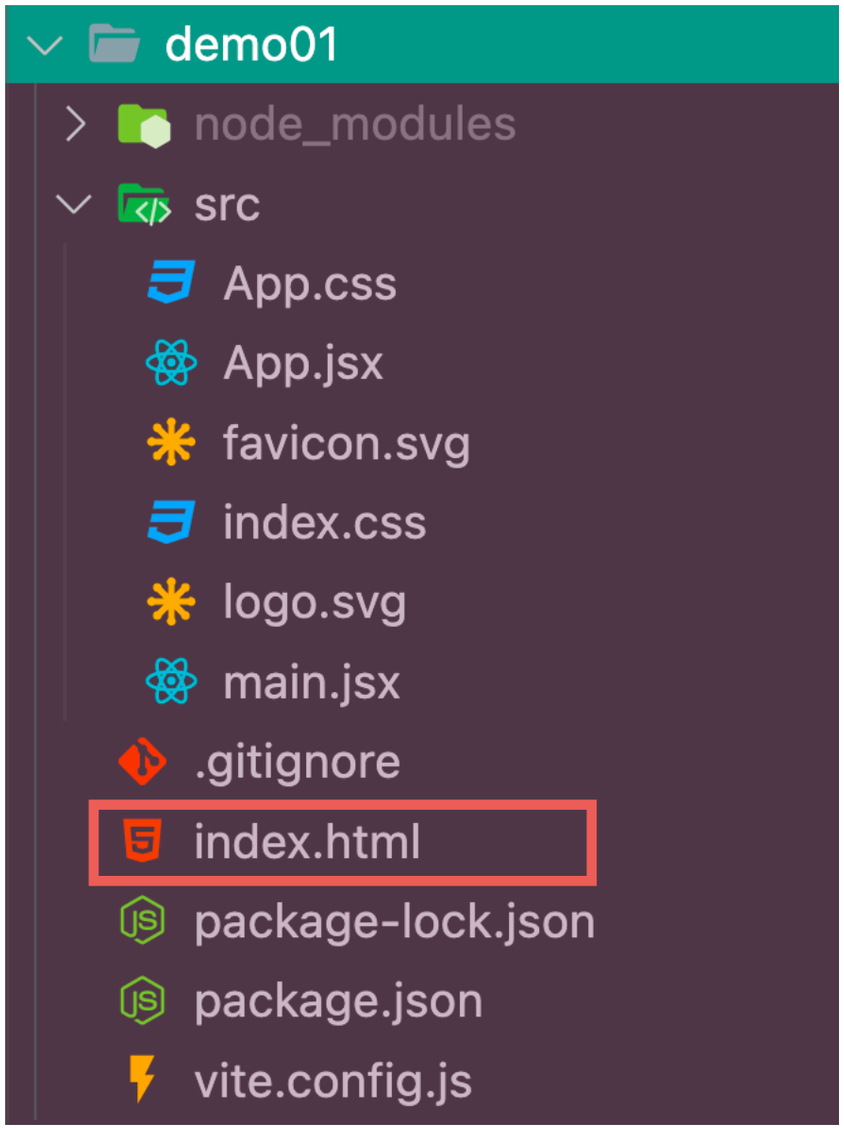
React Developer Tools



https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadopljbjfkapdkoienihi?hl=en

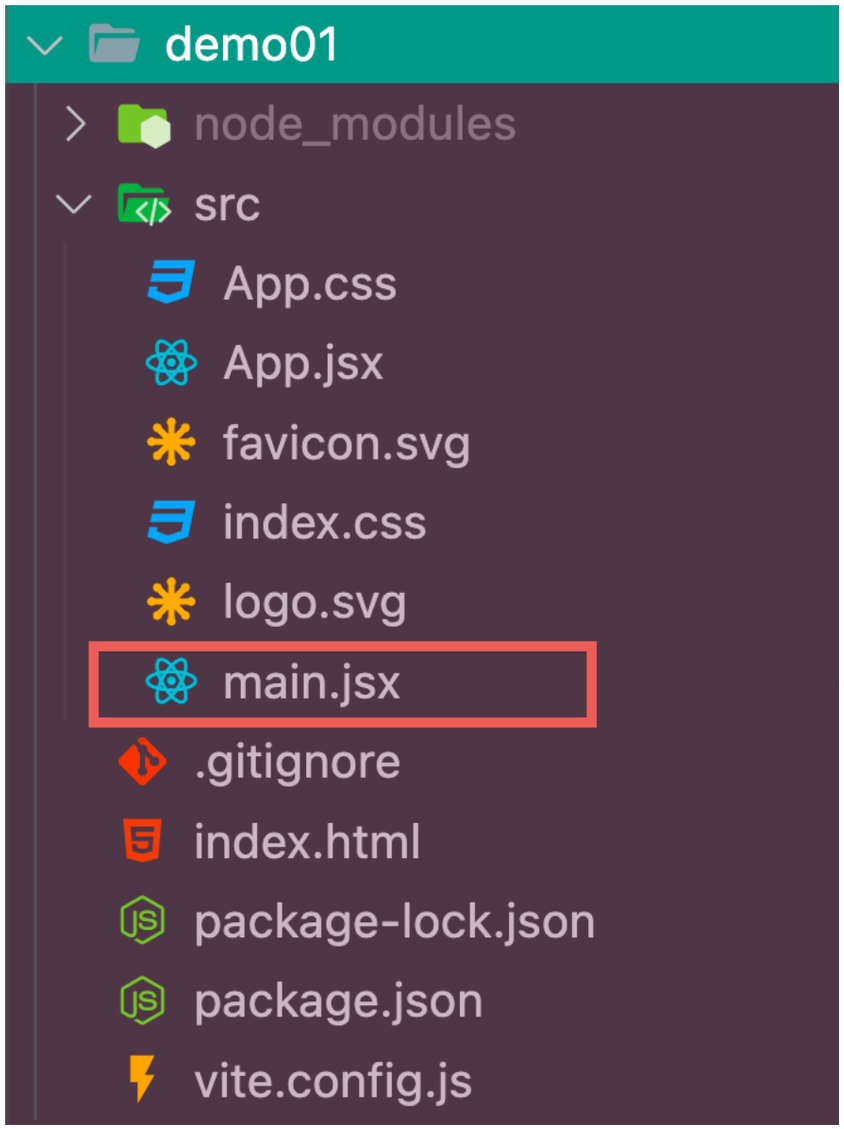
Structure of project

Structure of application

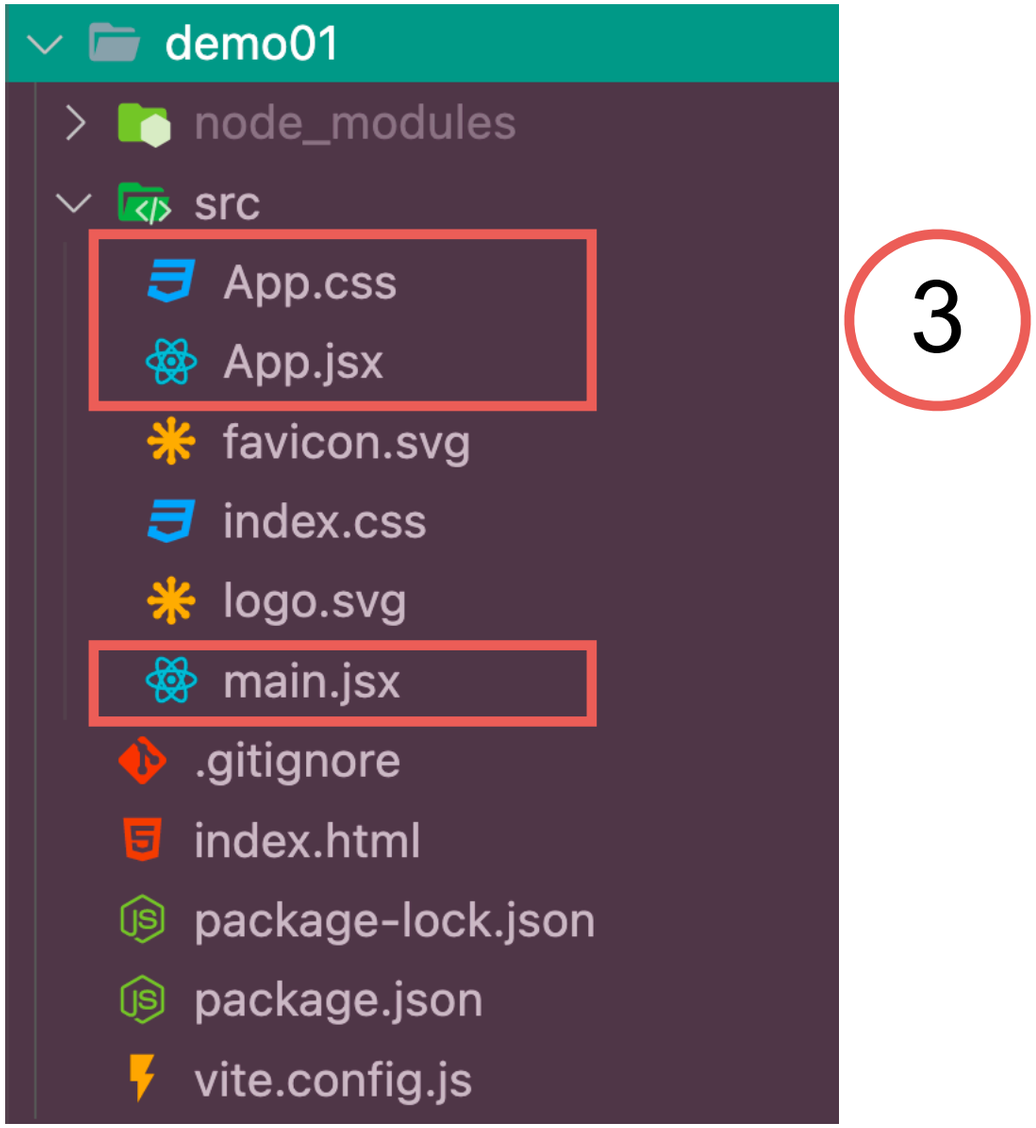


1

Structure of application



Structure of application



package.json

List of dependencies for React project

```
"scripts": {
  "dev": "vite",
  "build": "vite build",
  "preview": "vite preview"
},
"dependencies": {
  "react": "^18.0.0",
  "react-dom": "^18.0.0"
},
"devDependencies": {
  "@types/react": "^18.0.0",
  "@types/react-dom": "^18.0.0",
  "@vitejs/plugin-react": "^1.3.0",
  "vite": "^2.9.5"
}
```

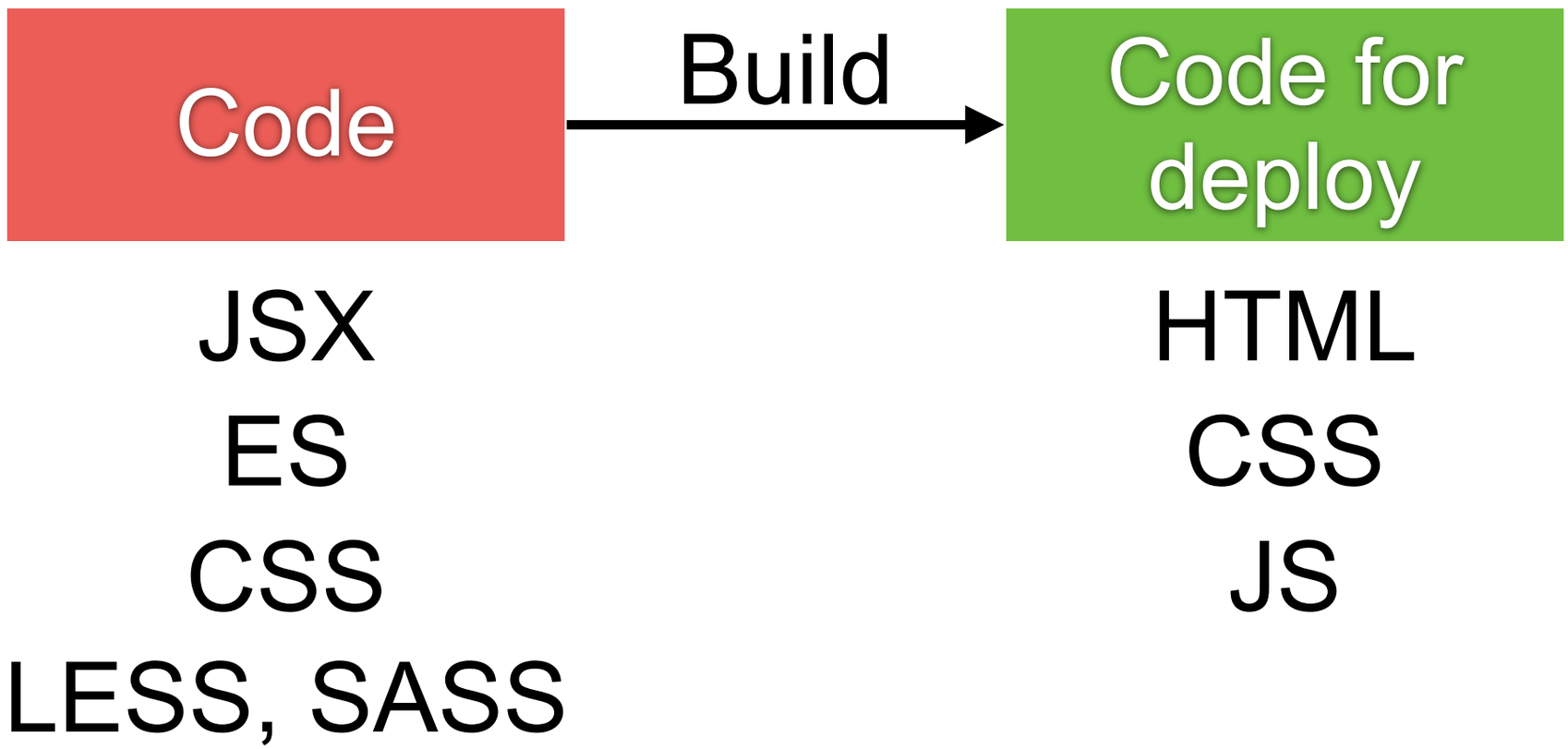
package.json

List of scripts/command to dev/build project

```
"scripts": {
  "dev": "vite",
  "build": "vite build",
  "preview": "vite preview"
},
"dependencies": {
  "react": "^18.0.0",
  "react-dom": "^18.0.0"
},
"devDependencies": {
  "@types/react": "^18.0.0",
  "@types/react-dom": "^18.0.0",
  "@vitejs/plugin-react": "^1.3.0",
  "vite": "^2.9.5"
}
```

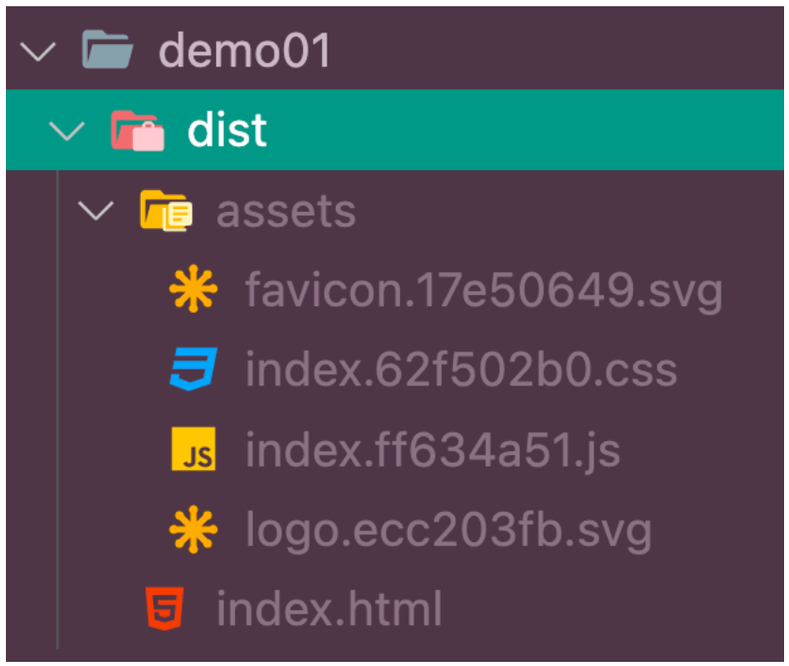
Build your app

$npm run build



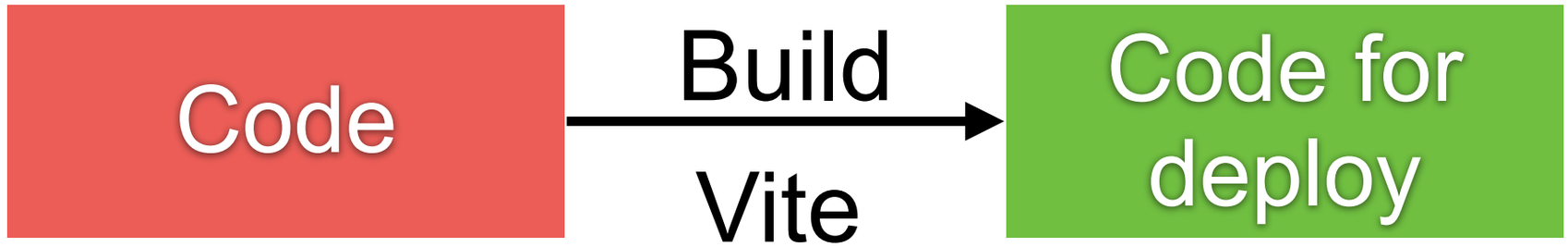
Build your app

$npm run build



Build your app

$npm run build

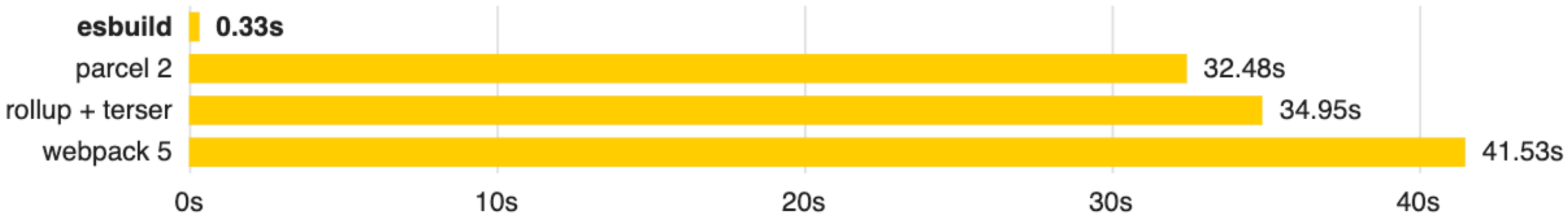


esbuild
An extremely fast JavaScript bundler

rollup.js

https://vitejs.dev/guide/why.html#the-problems

Performance !!



Above: the time to do a production bundle of 10 copies of the three.js library from scratch using default settings, including minification and source maps. More info here.

https://esbuild.github.io/

Deploy React app

https://vitejs.dev/guide/static-deploy.html

Deploy react app to server

Deploy react app to server

Locally
GitHub Pages
GitLab Pages
Netlify
Google Firebase
Vercel

Deploy to GitHub Pages

https://vitejs.dev/guide/static-deploy.html#github-pages

Config in vite.config.js

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

// https://vitejs.dev/config/
export default defineConfig({
  base: '/your-repo-name/',
  plugins: [react()]
})
```

https://vitejs.dev/guide/static-deploy.html#github-pages

Build and preview

$npm run preview

```
> demo01@0.0.0 preview
> vite preview

  > Local: http://localhost:4173/my-react-deploy/
  > Network: use `--host` to expose
```

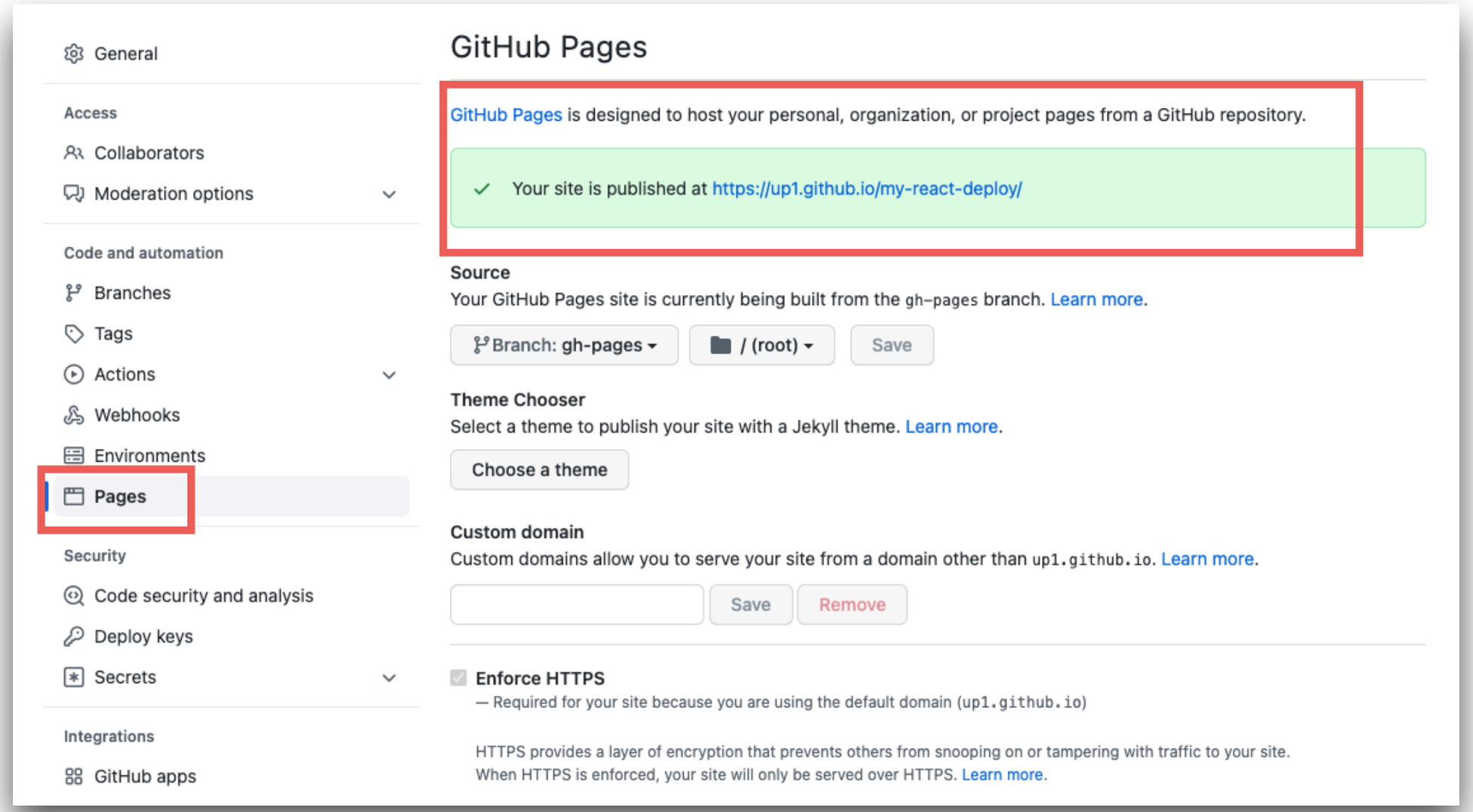
https://vitejs.dev/guide/static-deploy.html#testing-the-app-locally

Build and deploy

$npm run build
$git add dist -f
$git commit -m “Deploy”
$git subtree push --prefix dist origin gh-pages

Your GitHub Pages

Settings -> Pages



Back to React

Workflow

index.html

main.jsx

<App/>

Component

App.jsx

jsx

css

React Component

Class

Function

Function Component

```jsx
import { useState } from 'react'
import logo from './logo.svg'
import './App.css'

function App() {
  const [count, setCount] = useState(0)

  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>Hello Vite + React!</p>
        <p>
          <button type="button" onClick={() => setCount((count) => count + 1)}>
            count is: {count}
          </button>
        </p>
      </header>
    </div>
  )
}

export default App
```

Function Component

```jsx
import { useState } from 'react'
import logo from './logo.svg'
import './App.css'

function App() {
  const [count, setCount] = useState(0)

  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>Hello Vite + React!</p>
        <p>
          <button type="button" onClick={() => setCount((count) => count + 1)}>
            count is: {count}
          </button>
        </p>
      </header>
    </div>
  )
}

export default App
```

Use react hooks

Class Component

```
class AppC extends Component {

  constructor(props) {
    super(props);
    this.state = {
      count: 0,
    };
  }

  setCount(newValue) {
    this.setState({count: newValue})
  }

  render() {
    return (
       <div className="App">
         <header className="App-header">
           <img src={logo} className="App-logo" alt="logo" />
           <p>Hello React Class Component!</p>
           <p>
             <button type="button" onClick={() => this.setCount(this.state.count+1)}>
               count is: {this.state.count}
             </button>
           </p>
         </header>
       </div>
     );
  }
}
```

React Component

Component

props (properties)
States
return with render()

React Fundamentals

JSX (JavaScript XML-like)
Components
Props
States
Lifecycle
Event
Keys
Router

JSX (JavaScript XML)

HTML + JavaScript
Make HTML easy to understand
Boost up performance of JavaScript

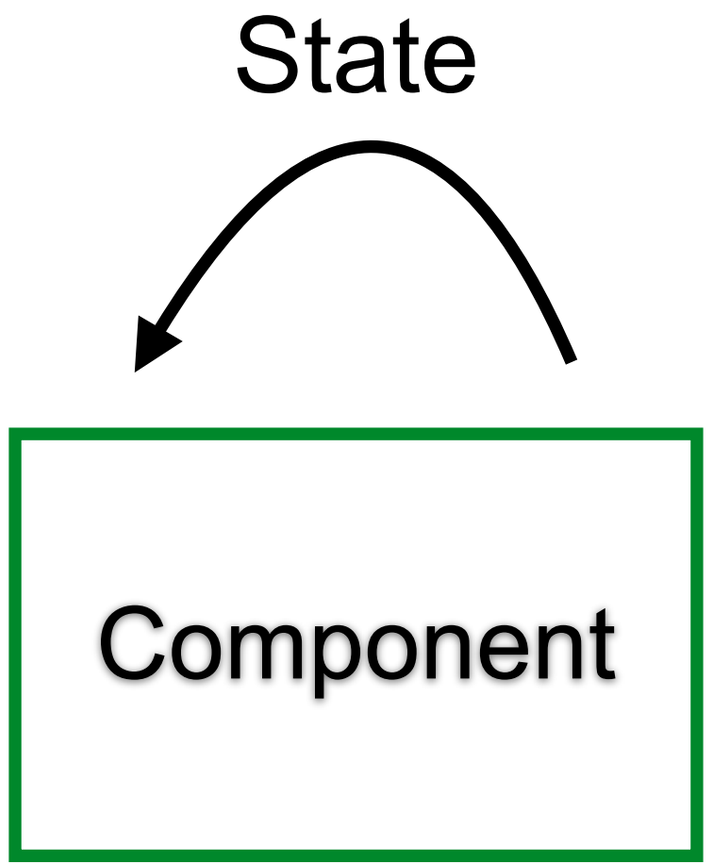
```
// Using JSX to express UI components
var dropdown =
  <Dropdown>
    A dropdown list
    <Menu>
      <MenuItem>Do Something</MenuItem>
      <MenuItem>Do Something Fun!</MenuItem>
      <MenuItem>Do Something Else</MenuItem>
    </Menu>
  </Dropdown>;

render(dropdown);
```

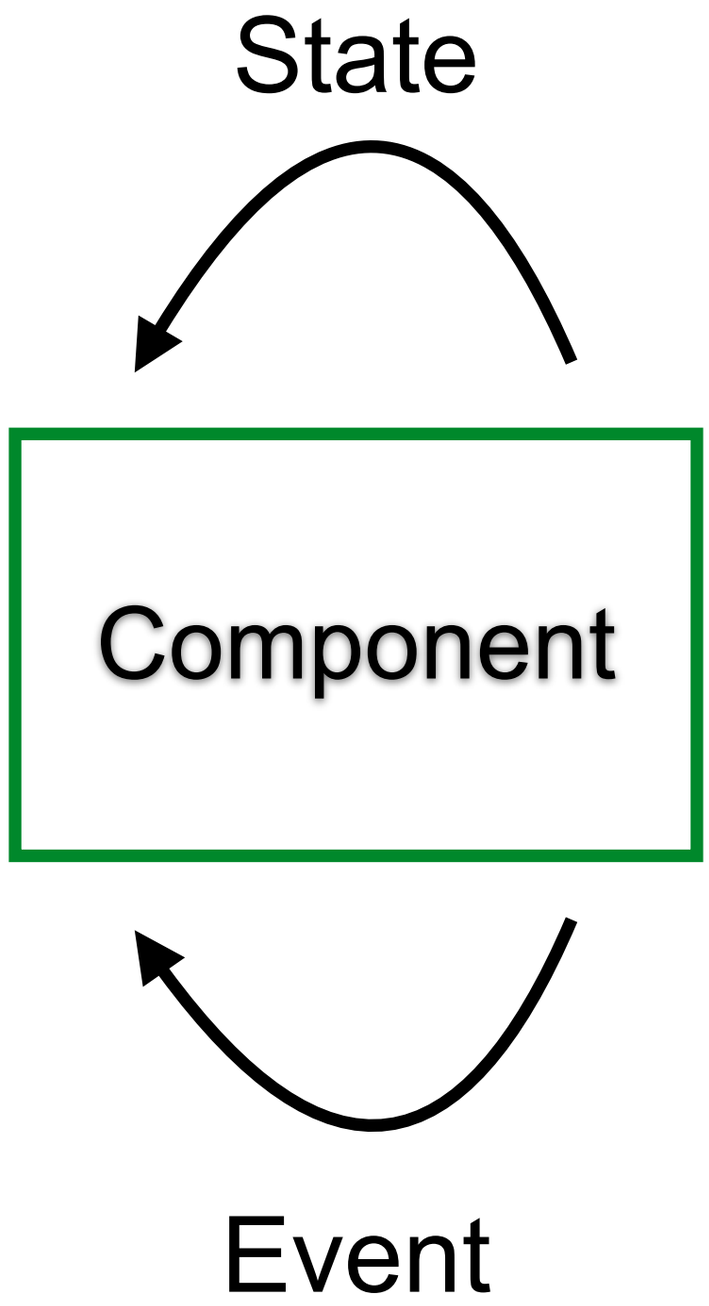
https://reactjs.org/docs/introducing-jsx.html
https://facebook.github.io/jsx/

Components

Everything in React is a component
Each component return a DOM object
Split the UI into independent reusable pieces
Each independent pieces is processed separately

https://reactjs.org/docs/components-and-props.html





Example with states

Use useState() from Hooks

useState(initialValue)

[value, updaterFn] = **useState**(initialValue)

```
/**
     * Returns a stateful value, and a function to update it.
     *
     * @version 16.8.0
     * @see https://reactjs.org/docs/hooks-reference.html#usestate
     */
    function useState<S>(initialState: S | (() => S)): [S, Dispatch<SetStateAction<S>>];
```

https://reactjs.org/docs/hooks-intro.html

useState()

```
import { useState } from "react";

const DemoState = (props) => {
    const [count] = useState(10);
    return <div>Counter = {count}</div>
}

export default DemoState;
```

useState() with update function

```
import { useState } from "react";

const DemoStateFn = (props) => {
    const [count, updateCount] = useState(10);
    const onClicked = () => updateCount(count+1)

    return(
        <div>
            Counter = {count}
            <div>
                <button onClick={onClicked}>Increase</button>
            </div>
        </div>
    );
}

export default DemoStateFn;
```

Problem !!

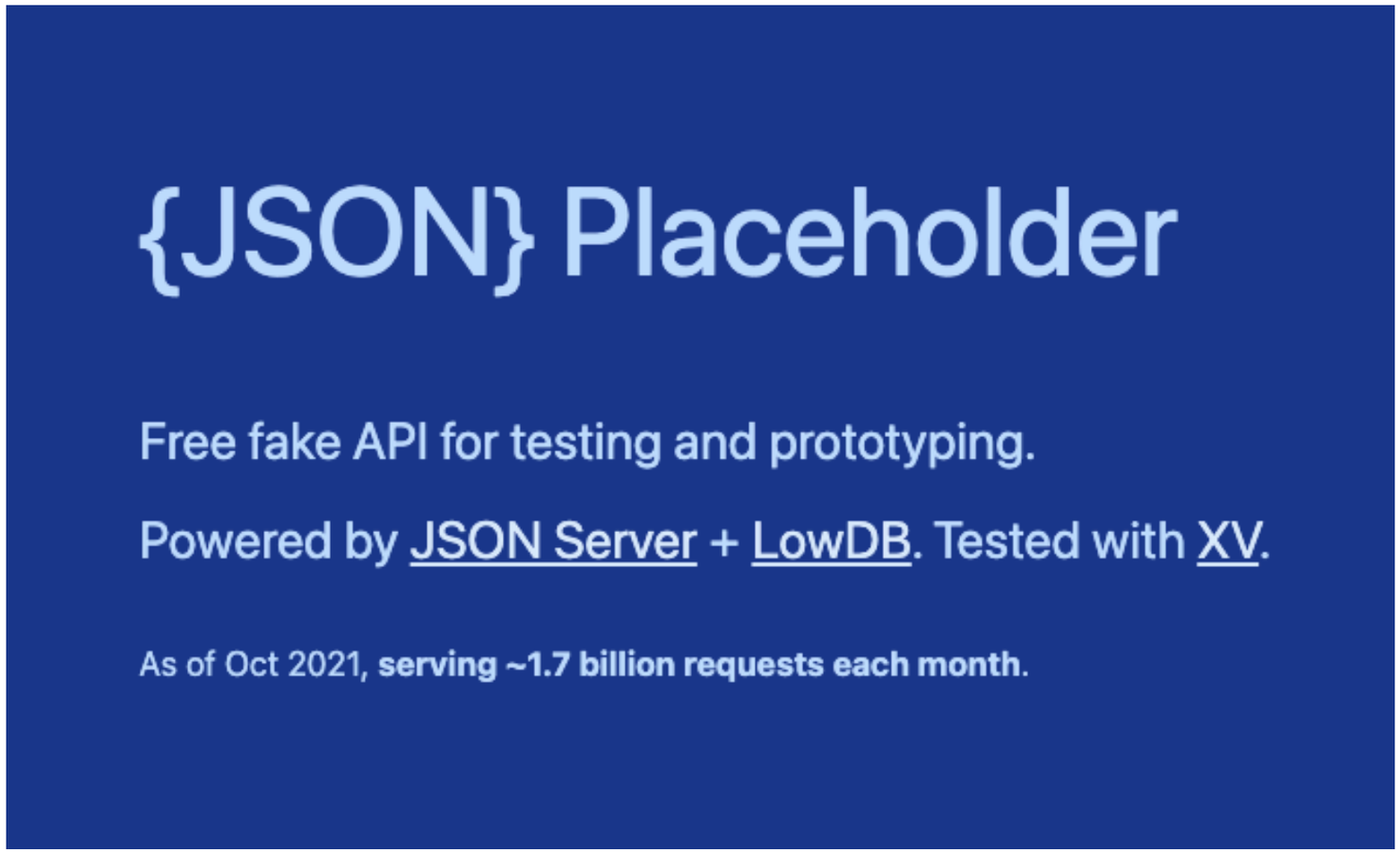
```
const DemoStateFn = (props) => {
    const [count, updateCount] = useState(10);
    const onClicked = (value) => updateCount(count+value)

    return(
        <div>
            Counter = {count}
            <div>
                <button onClick={onClicked(1)}>Increase</button>
            </div>
        );
}
```

```
▶ Uncaught Error: Too many re-renders. React    react-dom.development.js:16187
limits the number of renders to prevent an infinite loop.
    at renderWithHooks (react-dom.development.js:16187:15)
    at updateFunctionComponent (react-dom.development.js:20387:20)
    at beginWork (react-dom.development.js:22430:16)
    at HTMLUnknownElement.callCallback2 (react-dom.development.js:4161:14)
    at Object.invokeGuardedCallbackDev (react-dom.development.js:4210:16)
    at invokeGuardedCallback (react-dom.development.js:4274:31)
    at beginWork$1 (react-dom.development.js:27405:7)
    at performUnitOfWork (react-dom.development.js:26513:12)
    at workLoopSync (react-dom.development.js:26422:5)
    at renderRootSync (react-dom.development.js:26390:7)
```

Solved

```
const DemoStateFn = (props) => {
    const [count, updateCount] = useState(10);
    const onClicked = (value) => updateCount(count+value)

    return(
        <div>
            Counter = {count}
            <div>
                <button onClick={() => onClicked(1)}>Increase</button>
                <button
                   onClick={() => updateCount(current => current+1)}>
               Increase 2</button>
               </div>
           </div>
    );
}
```

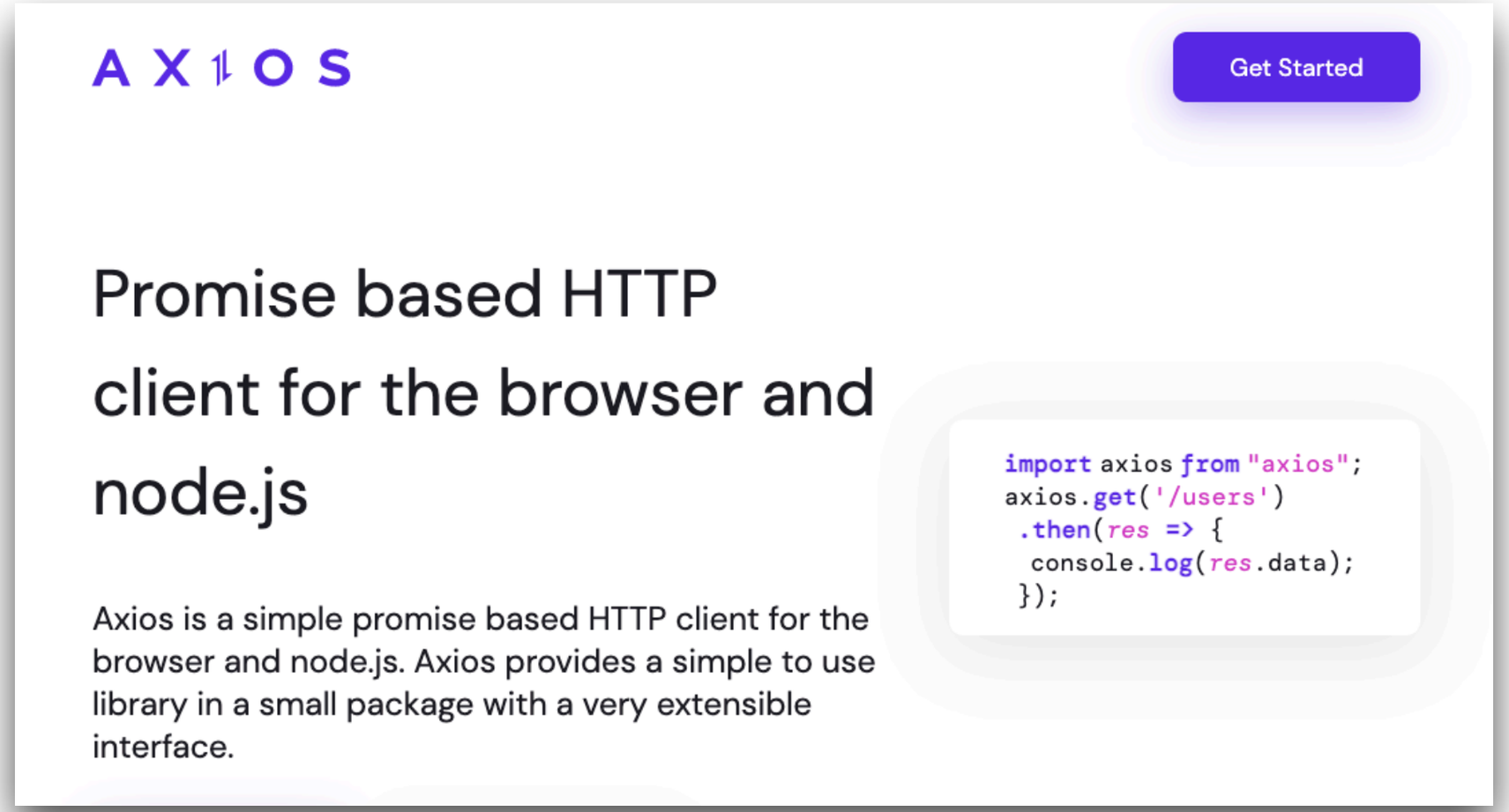
Call external api



https://jsonplaceholder.typicode.com/

Use Axios

$npm install -S axios



https://axios-http.com/

Call api !!

```
const DemoStateApi = (props) => {
    const [users, setUsers] = useState([]);

    axios.get('https://jsonplaceholder.typicode.com/users/')
        .then(function (response) {
            setUsers(response.data)
        });

    console.log(users);

    return(
        <div>
            All users
            <ul>
                {users.map((v) => <li key={v.id}>{v.name}</li>)}
            </ul>
        </div>
    );
}
```

Infinity loop !!

```
▶ (10) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]      DemoStateApi.jsx:12
▶ (10) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]      DemoStateApi.jsx:12
▶ (10) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]      DemoStateApi.jsx:12
▶ (10) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]      DemoStateApi.jsx:12
▶ (10) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]      DemoStateApi.jsx:12
▶ (10) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]      DemoStateApi.jsx:12
▶ (10) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]      DemoStateApi.jsx:12
▶ (10) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]      DemoStateApi.jsx:12
▶ (10) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]      DemoStateApi.jsx:12
▶ (10) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]      DemoStateApi.jsx:12
▶ (10) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]      DemoStateApi.jsx:12
▶ (10) [{…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}, {…}]      DemoStateApi.jsx:12
```

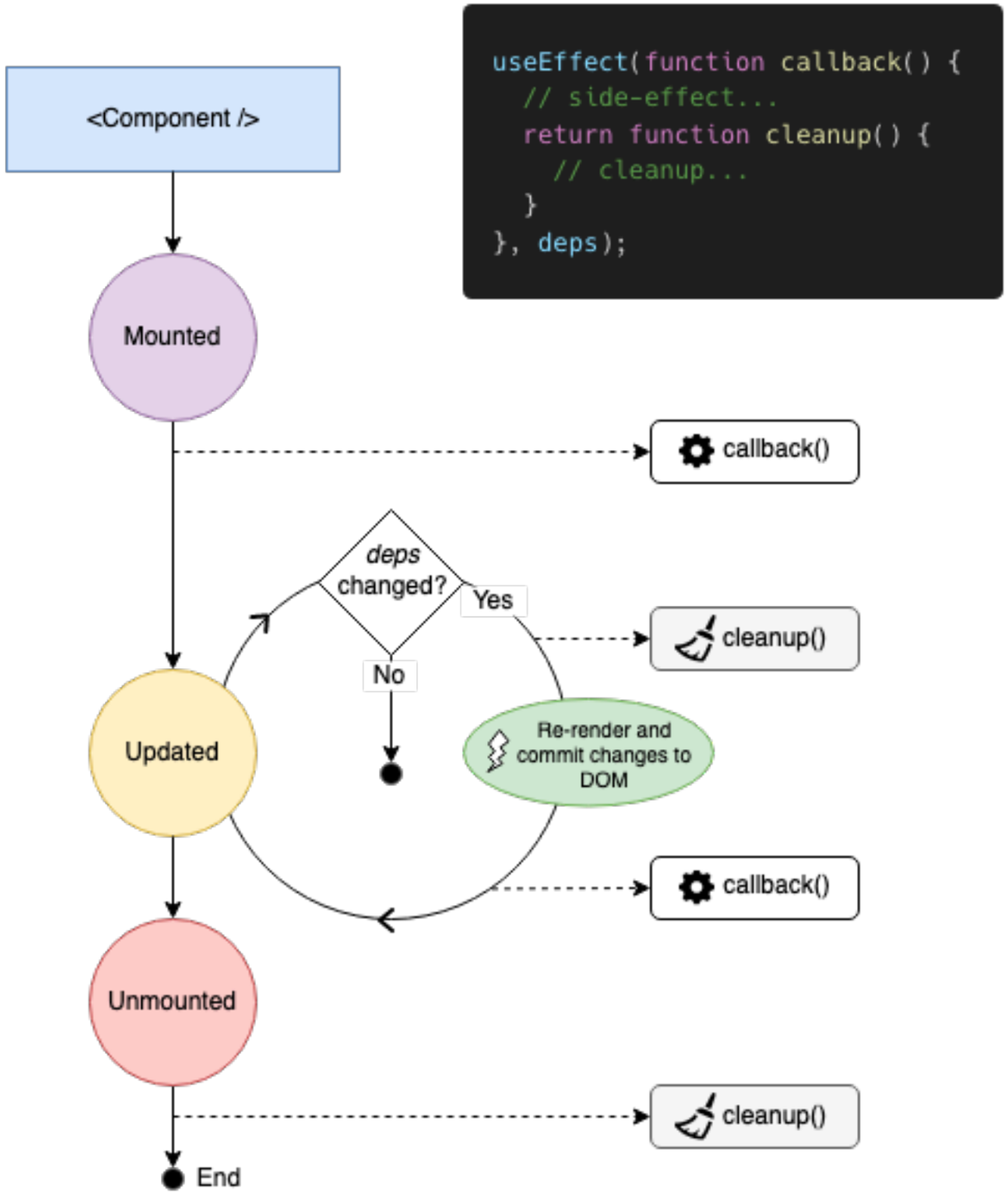
useEffect()

```
const DemoStateApi = (props) => {
    const [users, setUsers] = useState([]);

    useEffect( () => {
        console.log("Call effect");
        axios.get('https://jsonplaceholder.typicode.com/users/')
        .then(response => {
            setUsers(response.data)
        });
    }, []);

    ......

}
```

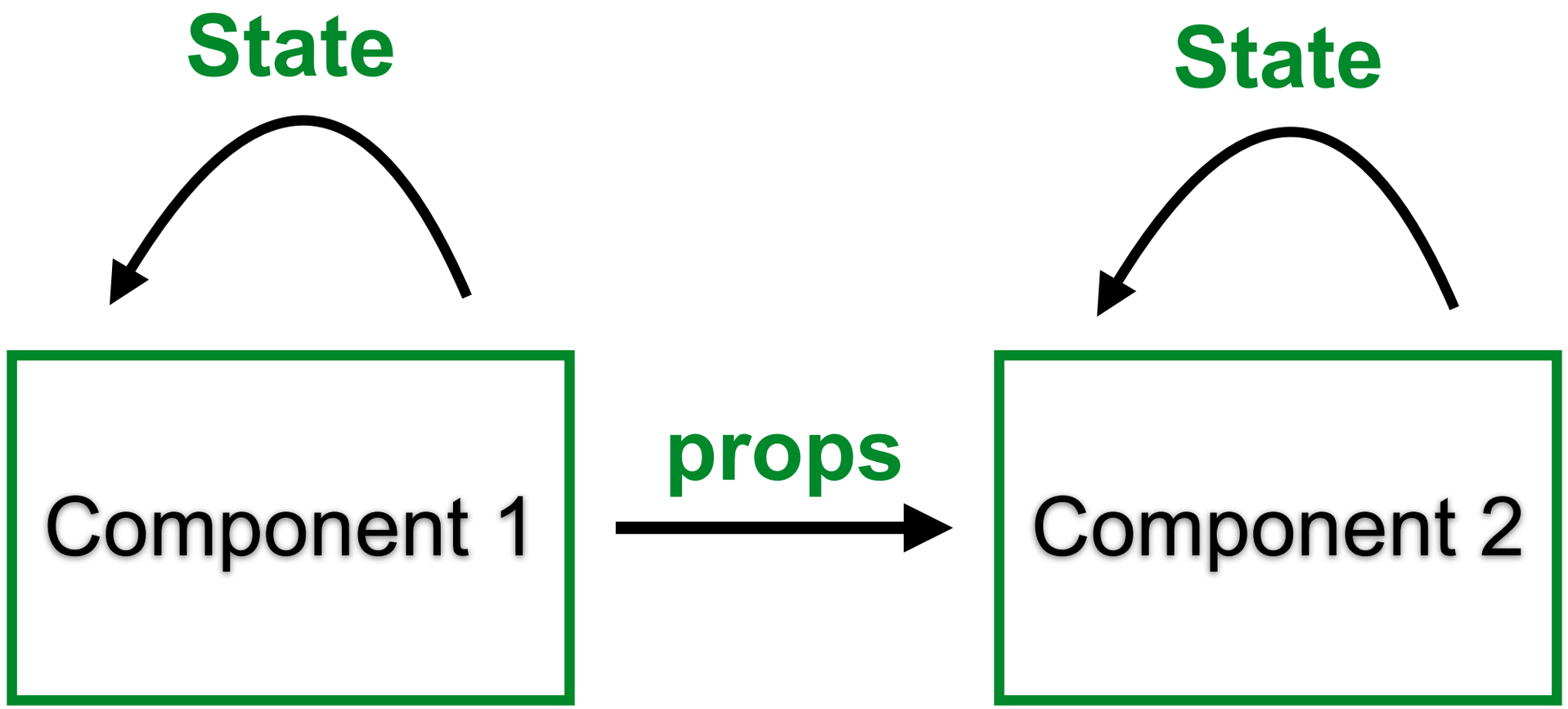
useEffect()

```
useEffect(() => {
    effect

    return () => {
        cleanup
    }

}, [input])
```

useEffect() Hook



Send data between components



States

Heart of react components
Use to determine component rendering and behavior
Create dynamic and interactive components

https://reactjs.org/docs/state-and-lifecycle.html

Props

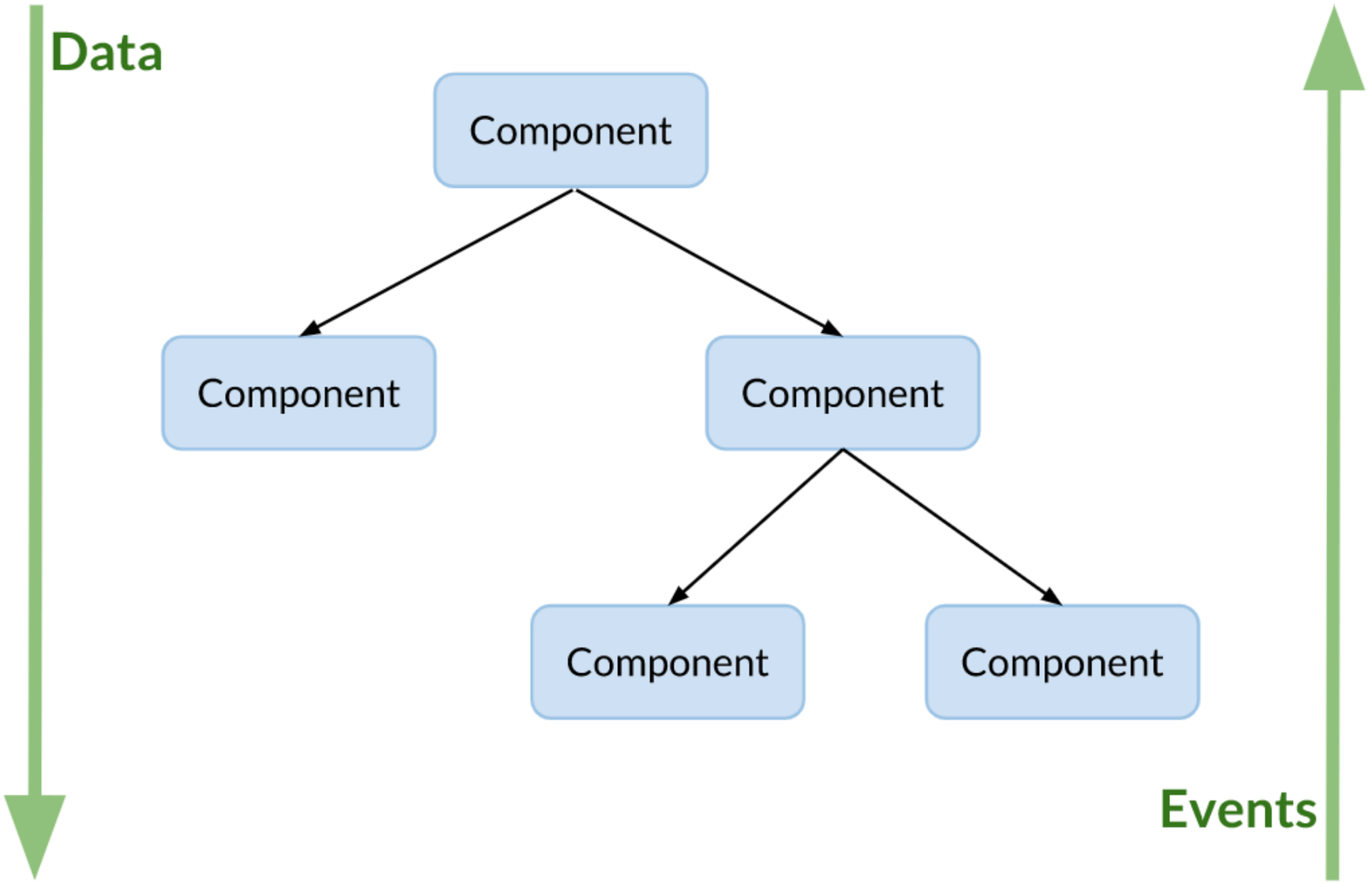
Read-only components

called “pure function”

Use to send data/state and action/event between component

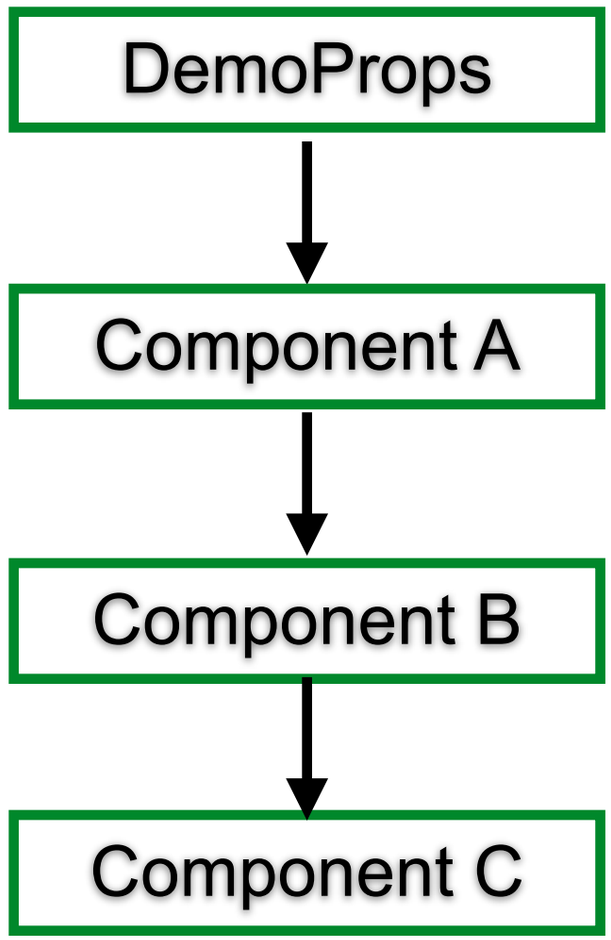
https://reactjs.org/docs/components-and-props.html

Unidirectional data flow

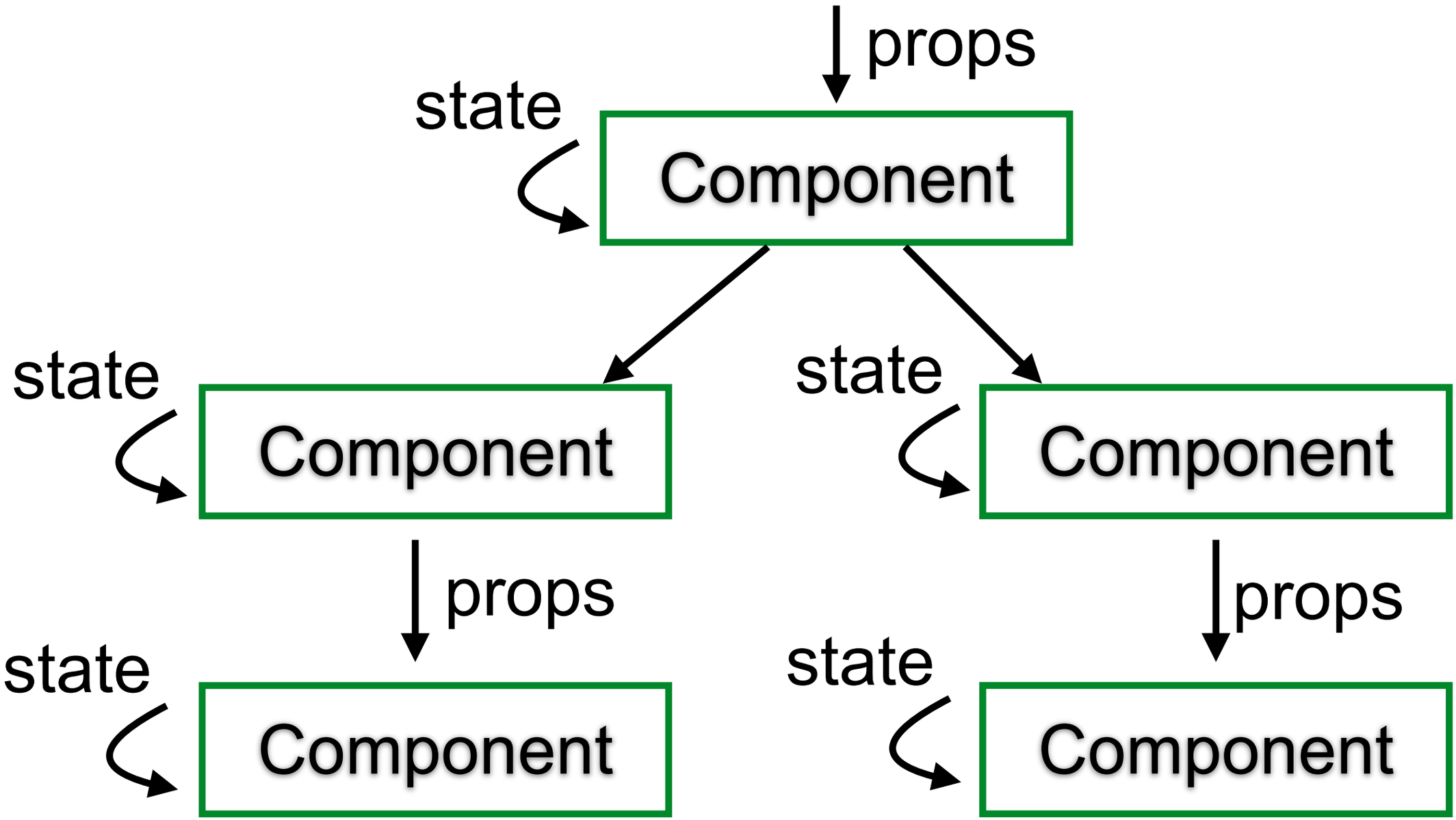


Working with props

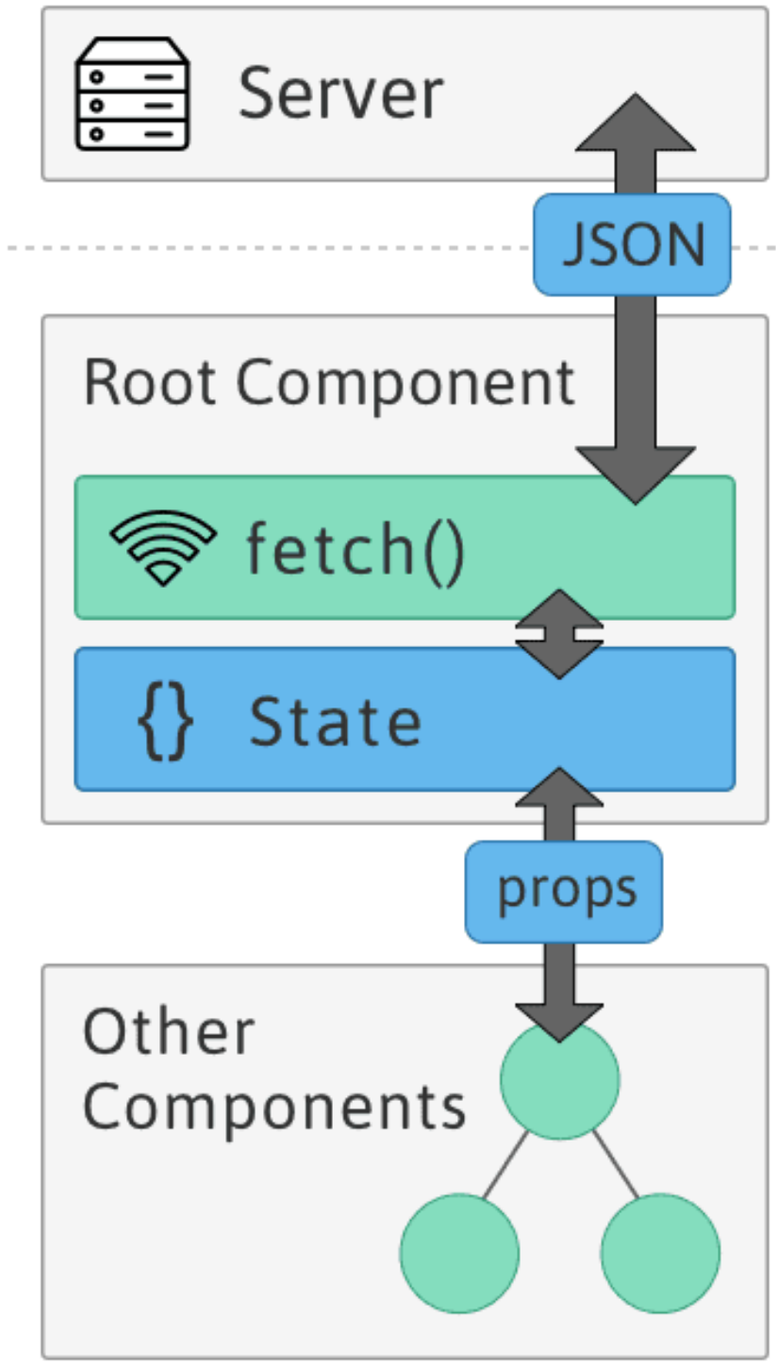
```
const DemoProps = () => {
    const myValue = "Hello from main";
    return <ComponentA value={myValue}></ComponentA>;
}

const ComponentA = (props) => {
    return <ComponentB value={props.value}></ComponentB>;
}

const ComponentB = (props) => {
    return <ComponentC value={props.value}></ComponentC>;
}

const ComponentC = (props) => {
    return <h3>{props.value}</h3>;
}
```



Props and States



Props and States



State management

Redux

useContext() in Hooks

Recoil

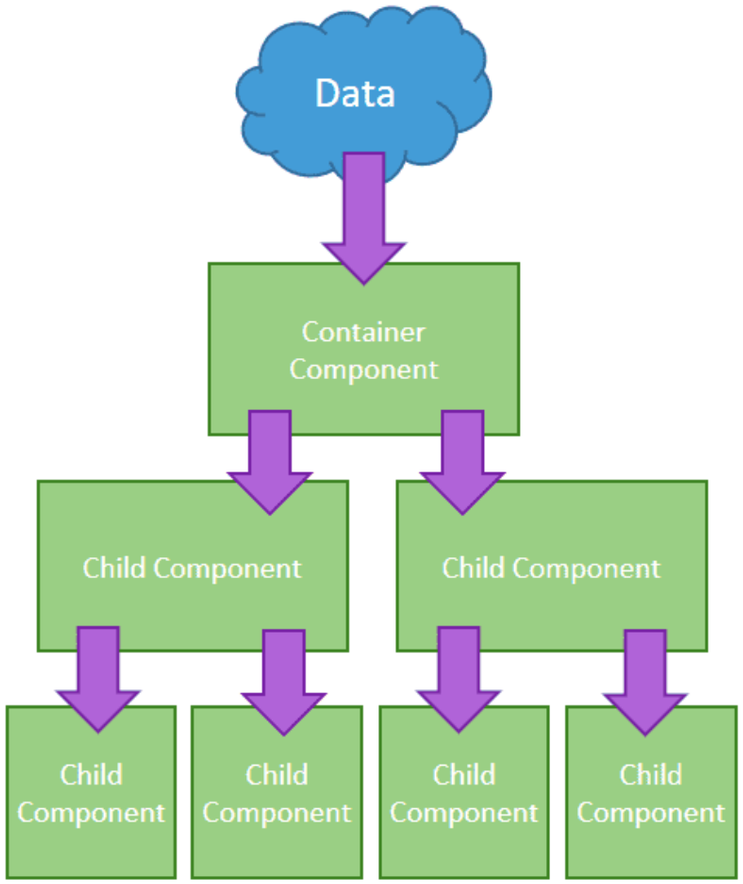
Jotai

Rematch

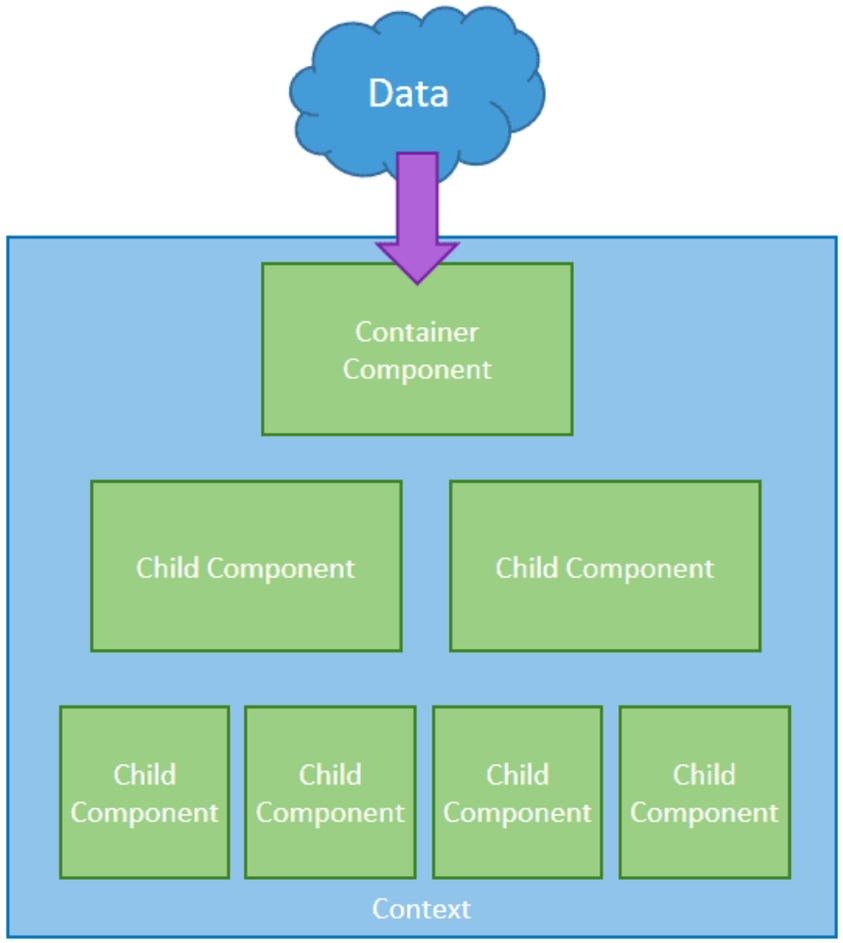
Zustand

useContext()

Shared states between component



prop drilling



context API

https://reactjs.org/docs/context.html

Working with context

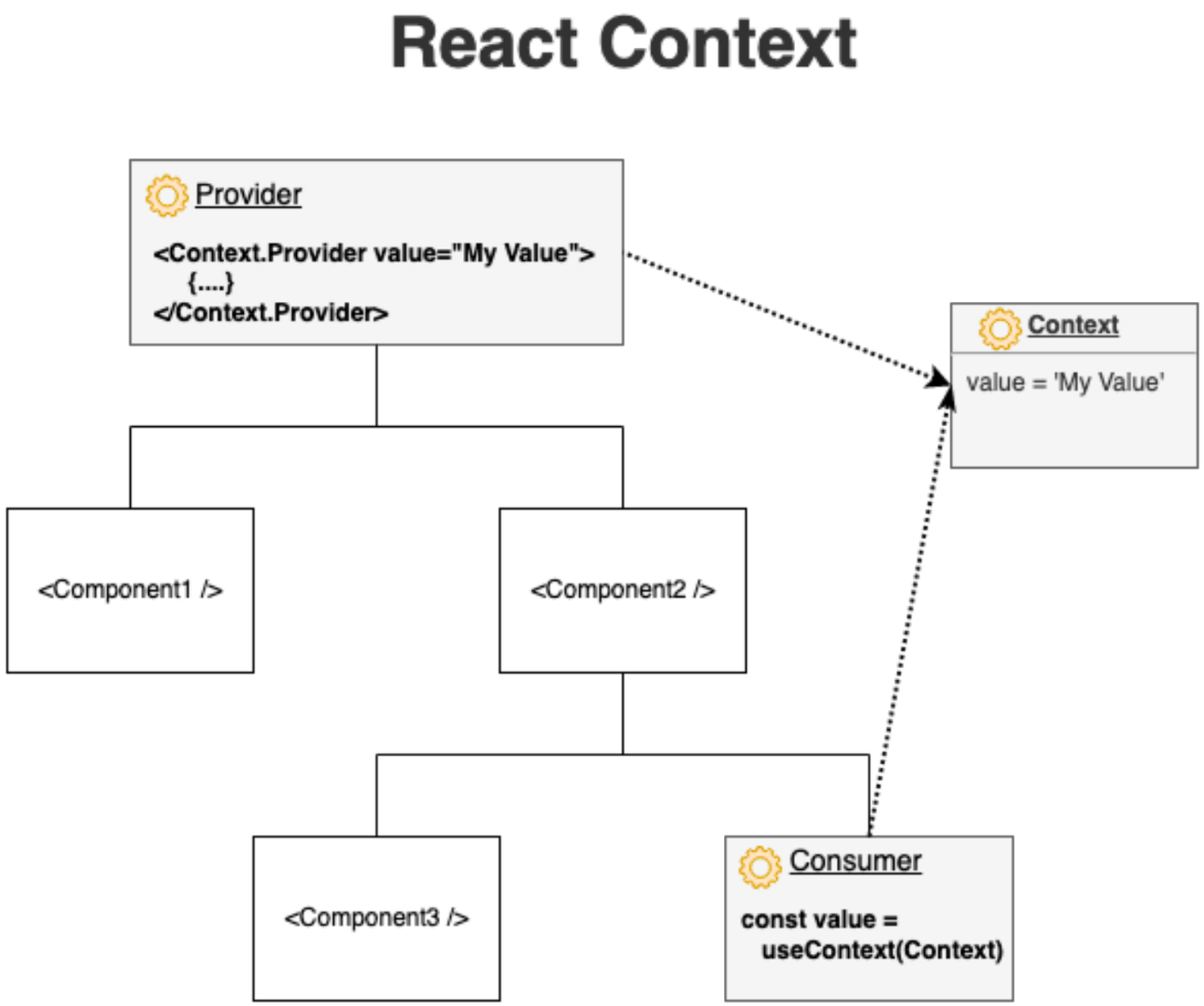
Create context

Provider

Consumer

https://reactjs.org/docs/hooks-reference.html#usecontext

Working with context



Working with context

```
import { createContext, useContext } from "react";

// 1. Create context
const MyContext = createContext("demo context");

const DemoContext = (props) => {
    const myValue = "Hello from context”;

    // 2. Provider
    return(
        <MyContext.Provider value={myValue}>
            <ChildComponent/>
        </MyContext.Provider>
    );
}

const ChildComponent = () => {
    // 3. Consumer
    const value = useContext(MyContext);
    return(
        <div>{value}</div>
    );
}
```

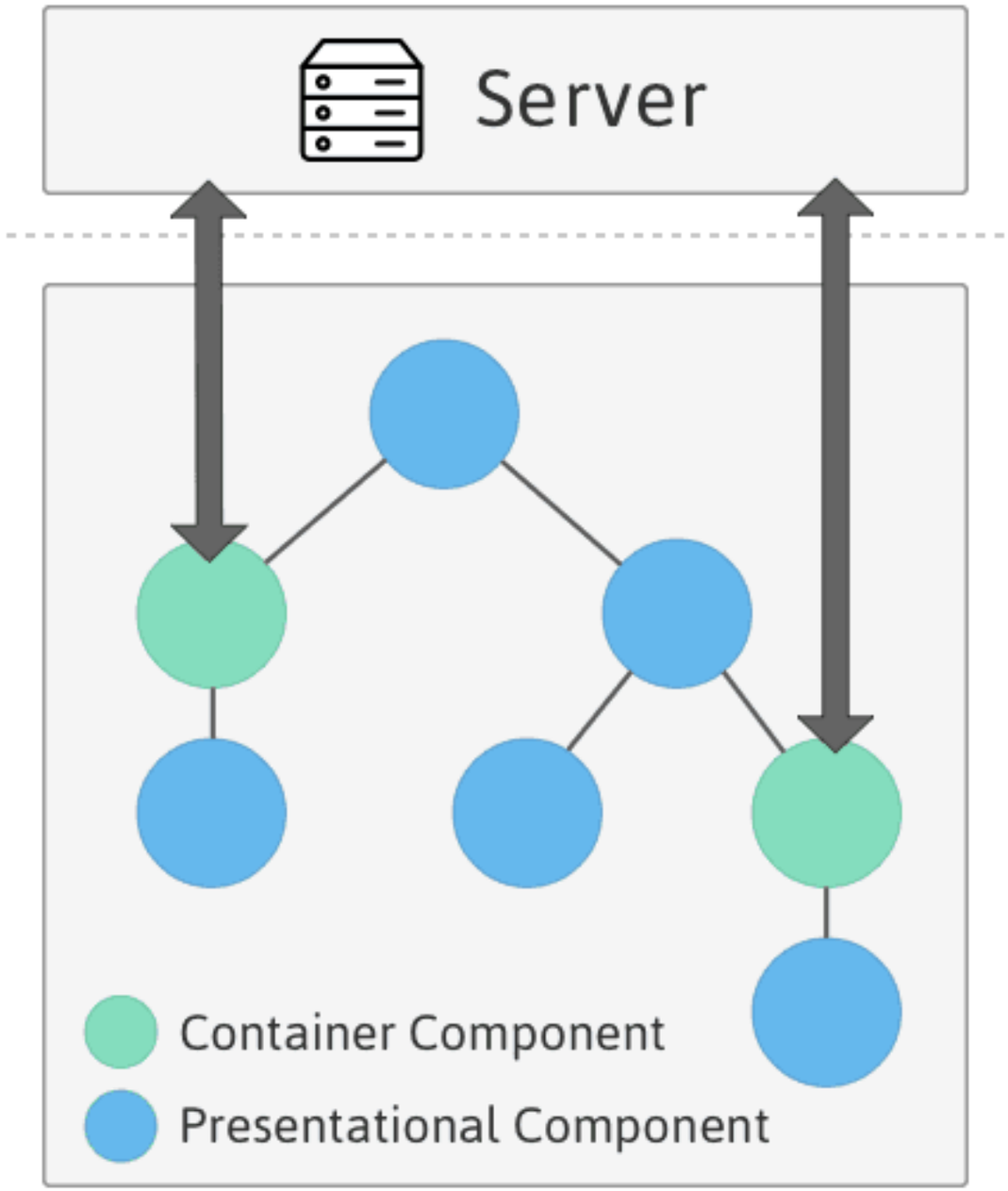
When to use Context ?

Global state
Theme
Application configuration
Authentication
Group of services

Complexity and hard to testing

Smart and Dumb component

Container component



Smart and Dumb component

Stateful and Stateless component

Smart = C in MVC

Dumb = V in MVC

Smart component

How things work

No DOM markup, no style

Provide data

Call action

Dumb component

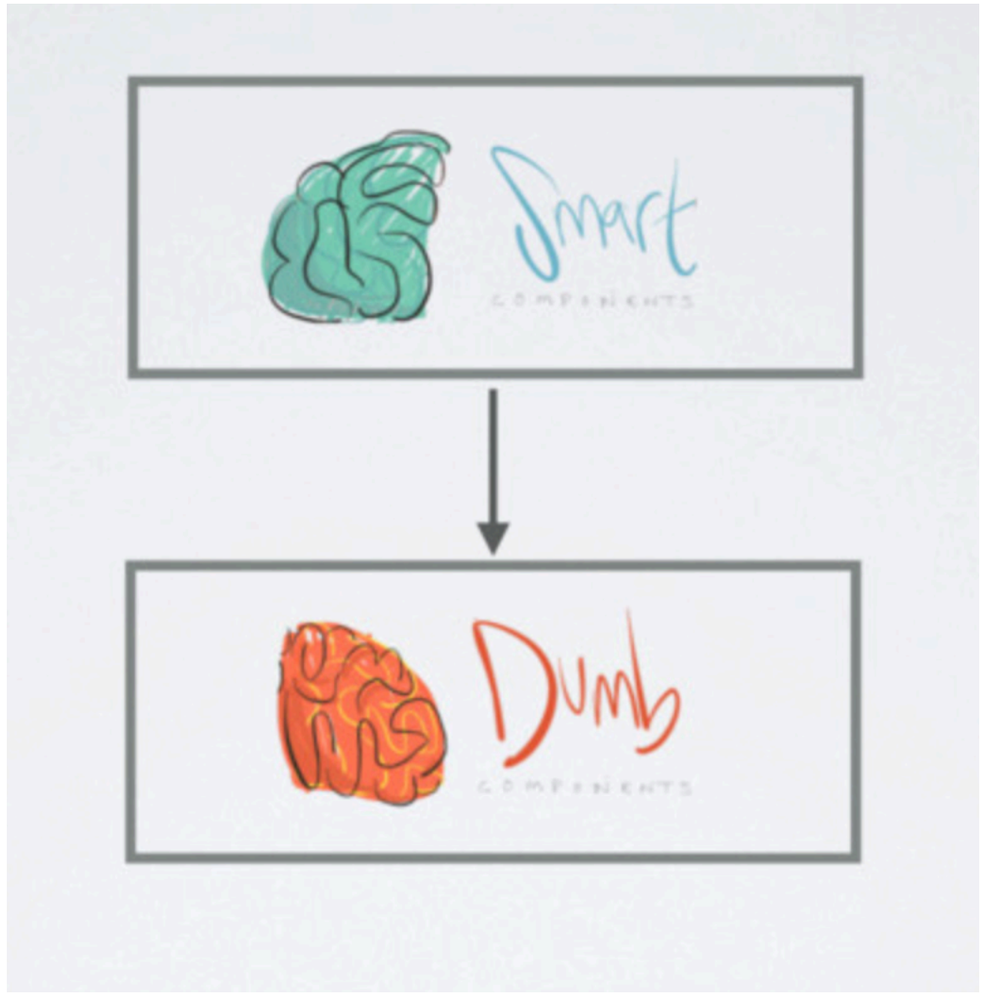
How things look

No app dependencies

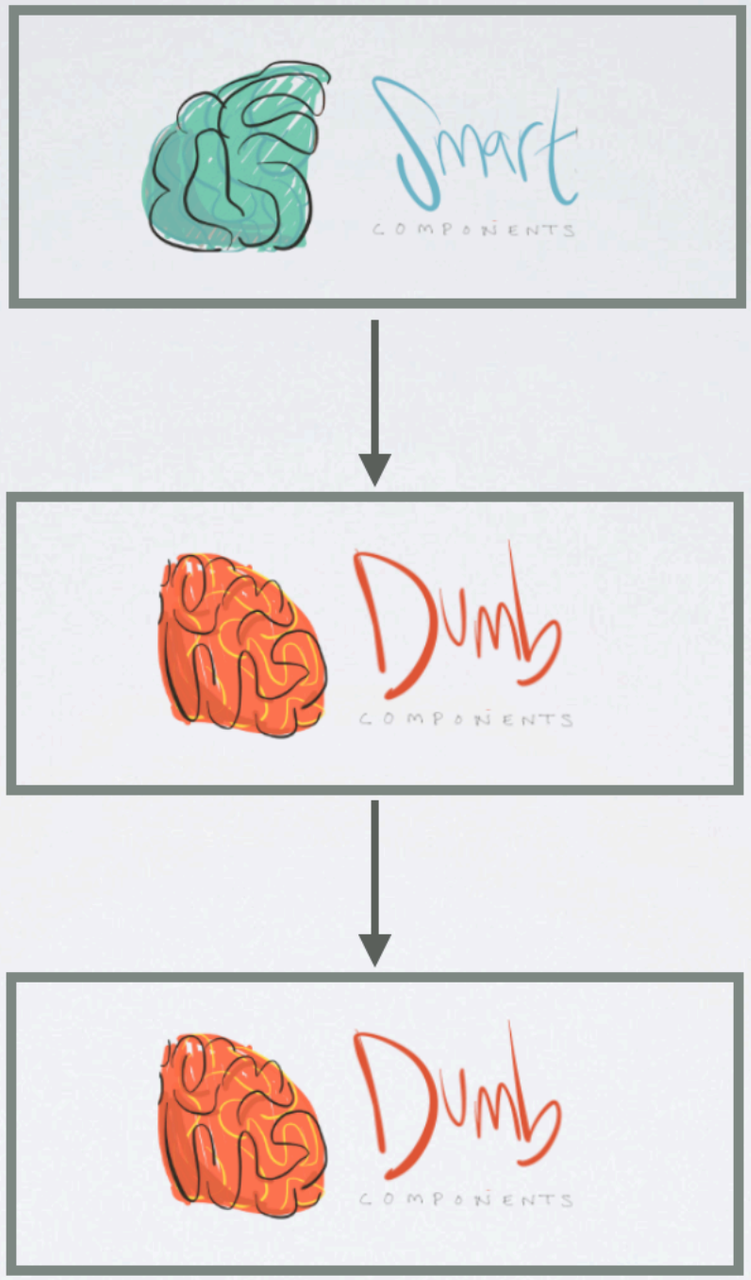
Just props, for data and callbacks

Rerely have own state, only UI state

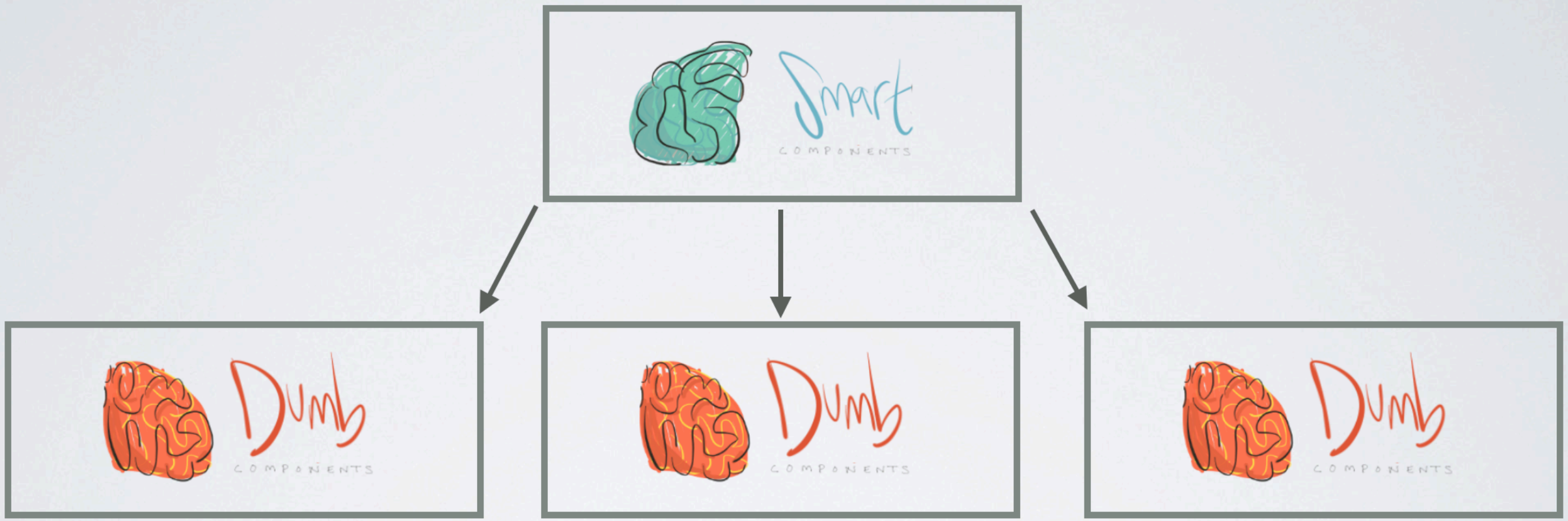
Smart and Dumb component



Smart and Dumb component



Smart and Dumb component



Create Welcome component

Create folder **components** in **src**

Create folder **welcome** in components

Create file **index.js** in welcome

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


class Welcome extends Component {
    render() {
        return <h1>Welcome to React</h1>
    }
}

export default Welcome
```

Use Welcome component

Edit file **App.js** in src

```
import Welcome from './components/welcome'

class App extends Component {
  render() {
    return (
      <Welcome/>
    )
  }
}
```

Send name to Welcome component

Edit file **App.js** in src

```
class App extends Component {
  render() {
    return (
      <Welcome name="somkiat"/>
    )
  }
}
```

Edit Welcome component

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

Initial state(s) of component

```
constructor(props) {
    super(props)
    this.state = {
        name: 'N/A'
    }
    this.change1 = this.change.bind(this)
}
```

Update state(s) of component

Use setState() method

```
change(e) {
  console.log('this is:', e.target.value);
  this.setState({
      name: e.target.value
  })
}
```

Events

Events are the triggered reactions to specific actions link mouse click, key press etc.

Events pass as props of element

https://reactjs.org/docs/handling-events.html

Events

onClick()
onSubmit()
onChange()
onKeyUp()

Example of onChange()

```
render() {
  return (
    <div>
      <input type="text"
             value={this.state.name}
             onChange={this.change2}
      /><br/>
      {this.state.name}
    </div>
  )
}
```

How to handling event ?

Solution 1 :: Use Arrow function

```
change2 = (e) => {
   console.log('this is:', e.target.value);
   this.setState({
       name: e.target.value
   })
}
```

How to handling event ?

Solution 2 :: Use binding to component

```
constructor(props) {
    super(props)
    this.state = {
        name: 'N/A'
    }
    this.change1 = this.change.bind(this)
}

change(e) {
  console.log('this is:', e.target.value);
  this.setState({
      name: e.target.value
  })
}
```

Develop your application

Start with React !!

Design your app ?

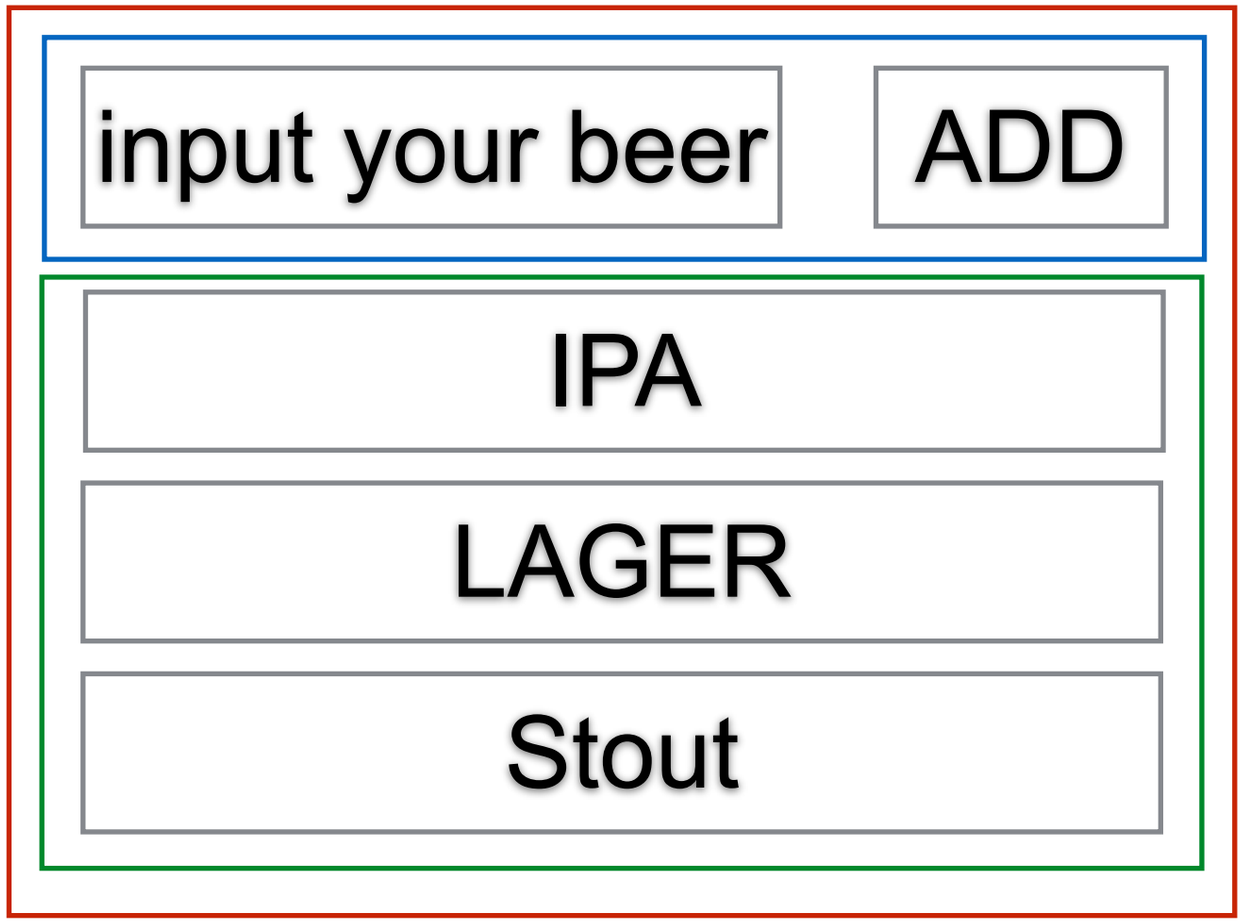
Let’s design your component

Component for react ?

1 component ?

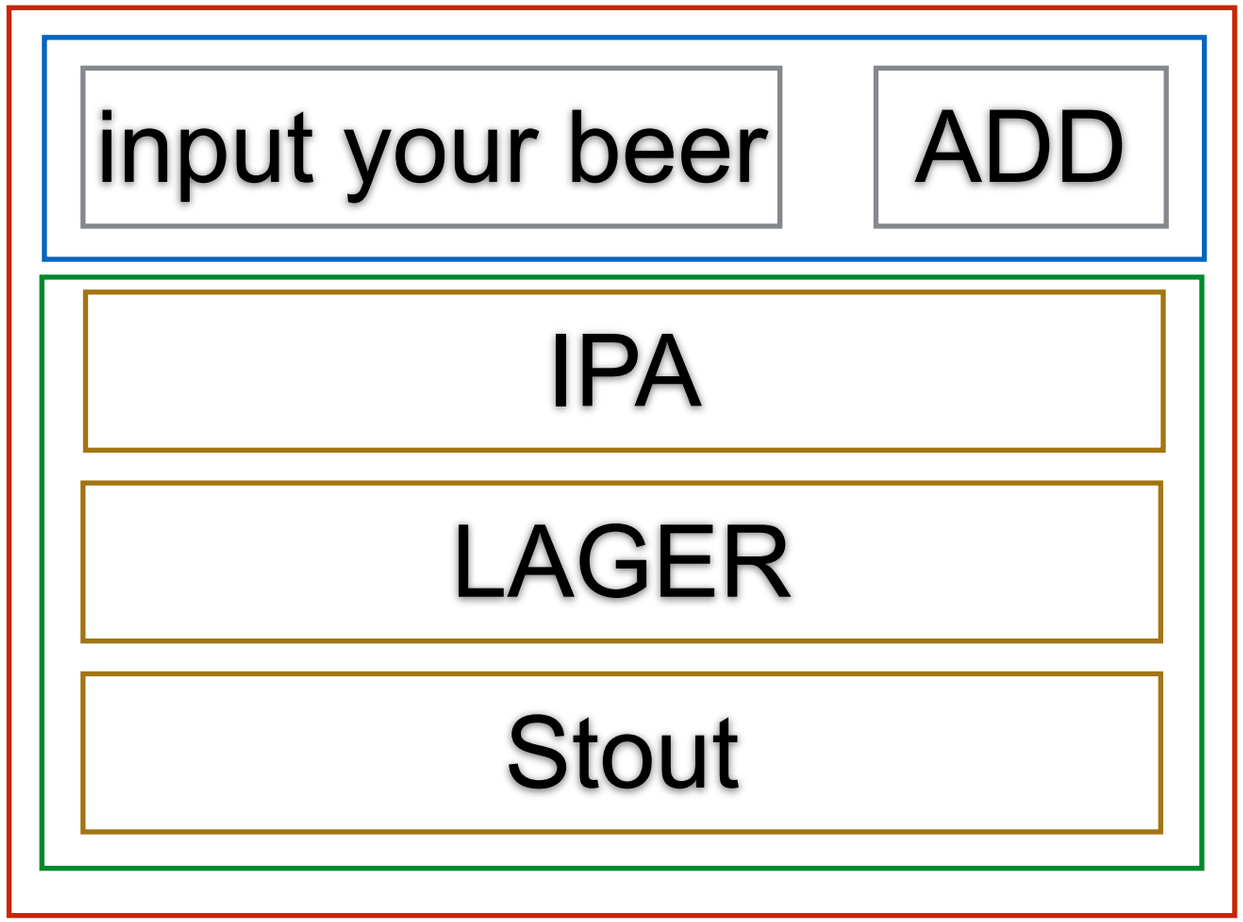
Component for react ?

3 component ?



Component for react ?

4 component ?



Components

BeerListContainer

- AddBeerComponent
  - Input text
  - Button

BeerListContainer

- BeerItemComponent

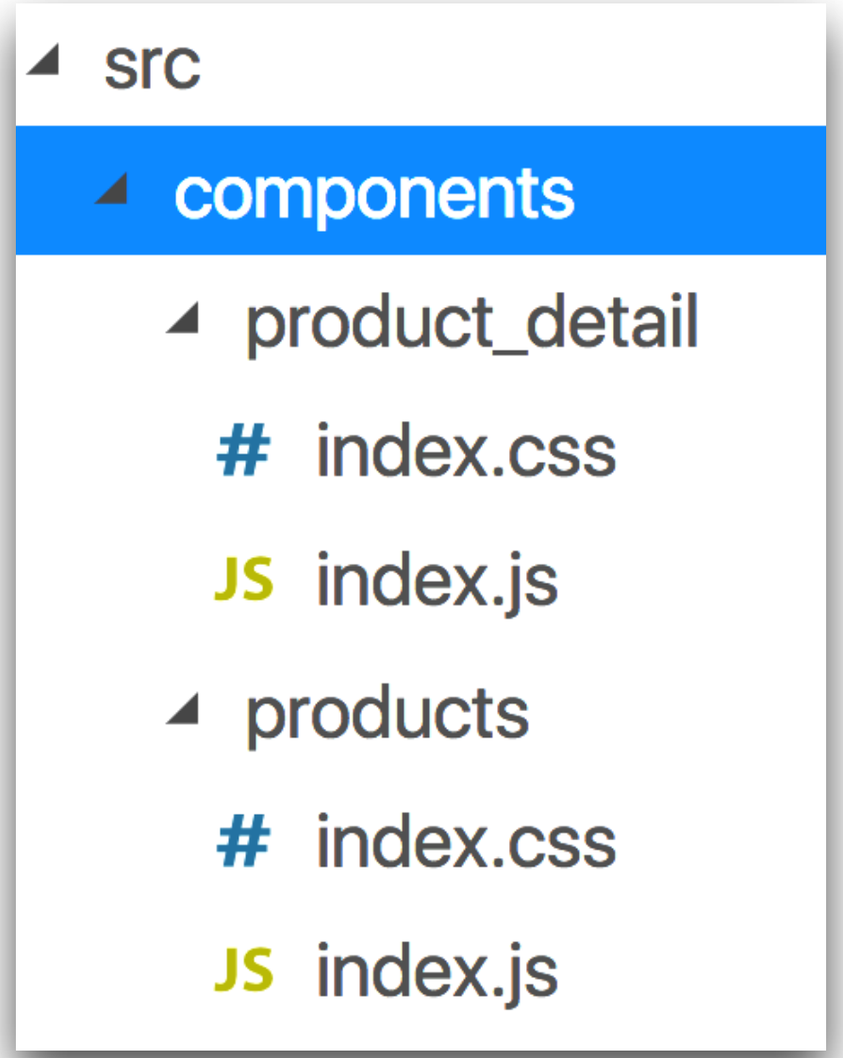
Components

BeerContainer/App

- **AddBeer**
  - Input text
  - Button
- **BeerList**
  - BeerItem

Step 1
Create react component

Component structure



AddBeer component

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

class AddBear extends Component {
  render() {
    return (
      <div>
        <input type="text" id="name"/>
        <button>ADD</button>
      </div>
    );
  }
}

export default AddBear;
```

BeerList component

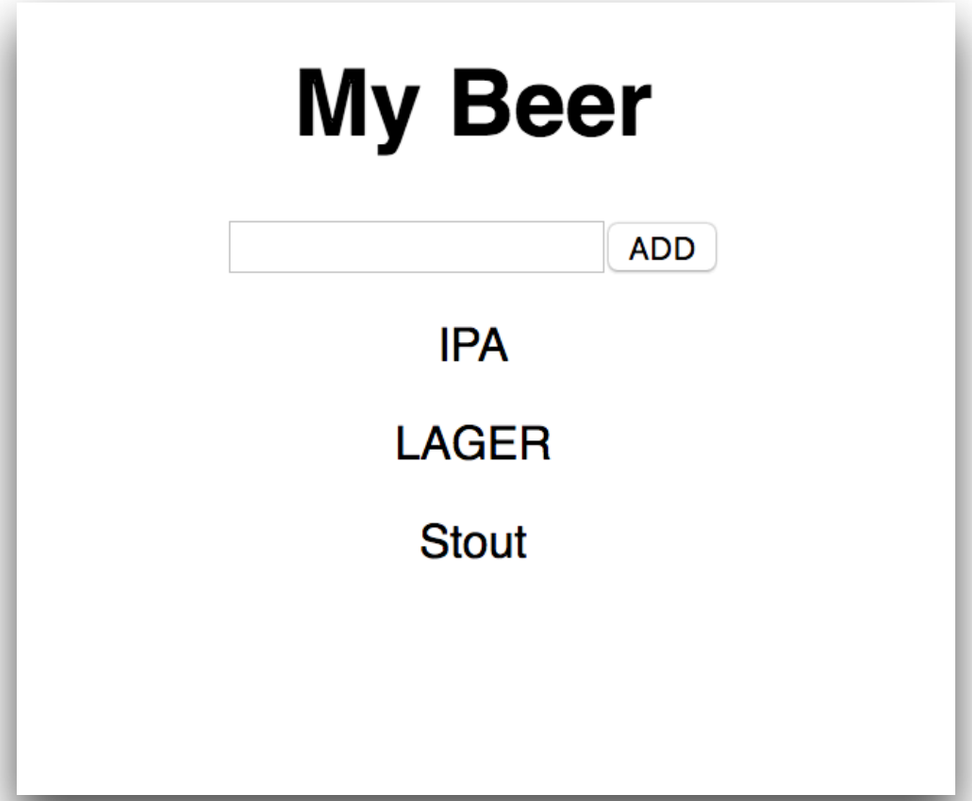
```
import React, { Component } from 'react';

class BeerList extends Component {
  render() {
    return (
      <div>
        <p>IPA</p>
        <p>LAGER</p>
        <p>Stout</p>
      </div>
    );
  }
}

export default BeerList;
```

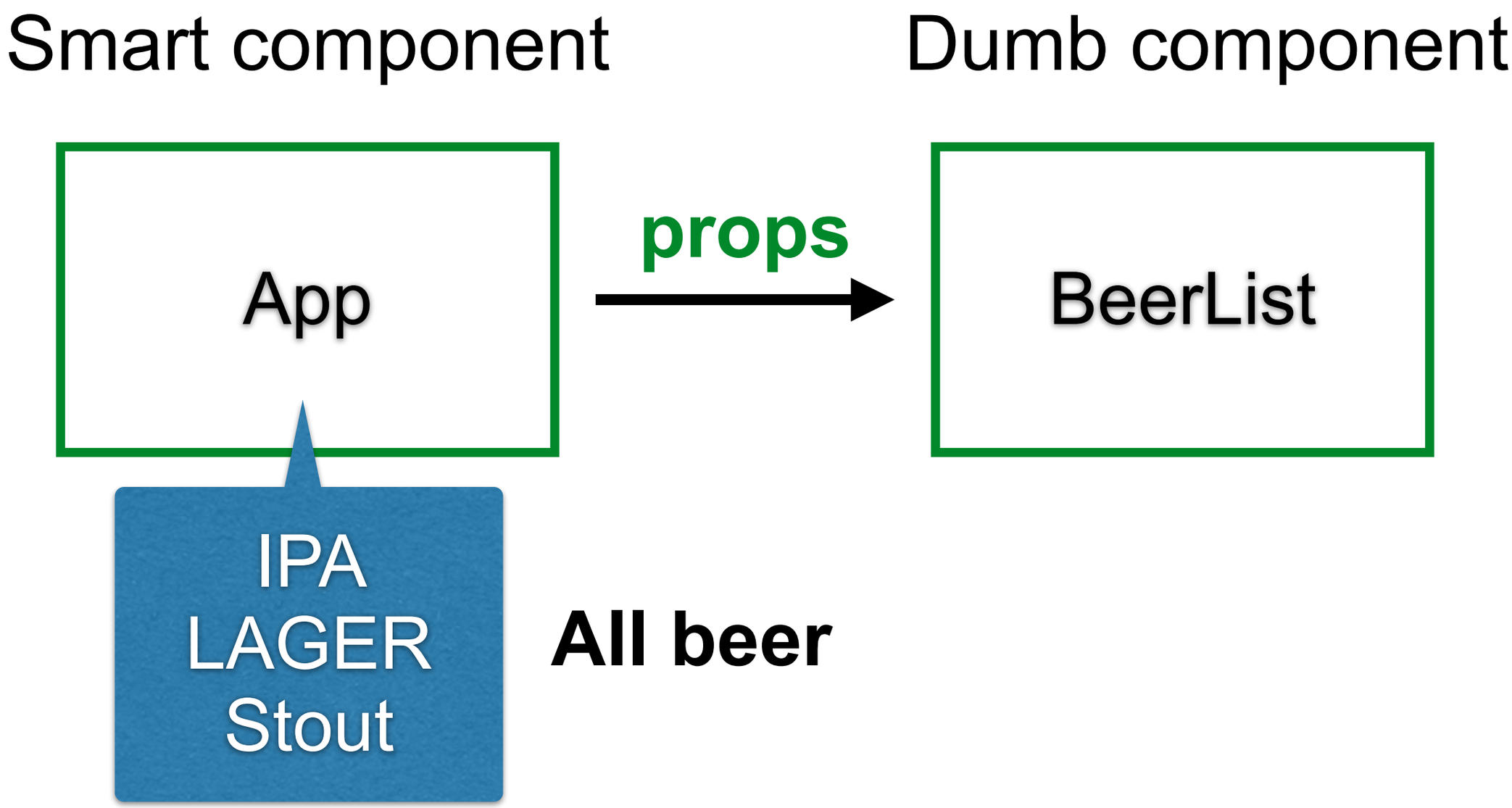
App component

```
import React, { Component } from 'react';
import BeerList from './components/BeerList';
import AddBeer from './components/AddBeer';

class App extends Component {
  render() {
    return (
      <div align="center">
        <h1>My Beer</h1>
        <AddBeer/>
        <BeerList/>
      </div>
    );
  }
}

export default App;
```

Composition
over
Inheritance

See result





All beer

App component

Provide data

```
class App extends Component {

  constructor(props) {
    super(props)
    this.state = {
      beers: [
        {id: 1, name: 'IPA'},
        {id: 2, name: 'LAGER'},
        {id: 3, name: 'Stout'},
      ]
    }
  }

}
```

App component

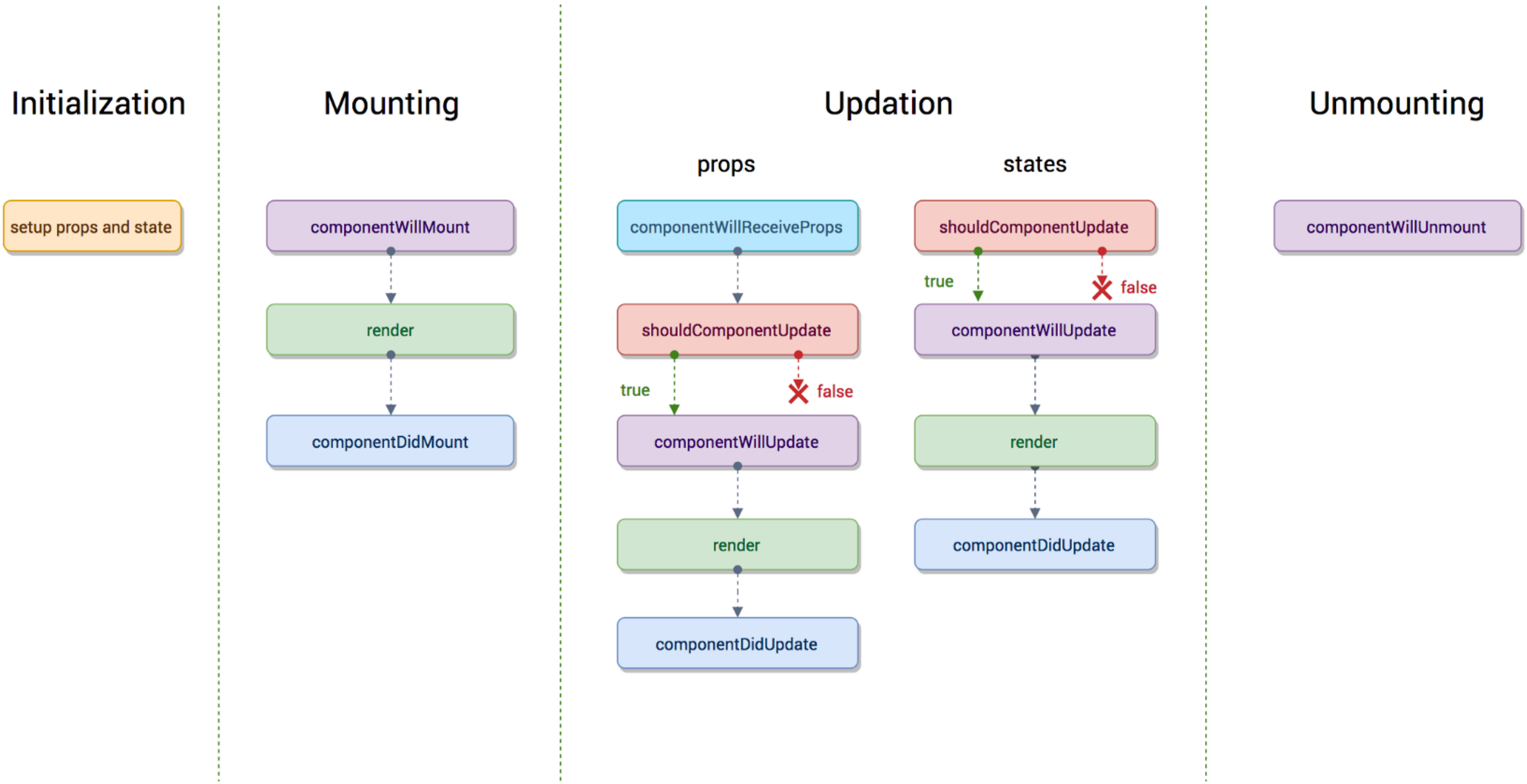
Send data via props to BeerList component

```
class App extends Component {

  render() {
    return (
      <div align="center">
        <h1>My Beer</h1>
        <AddBeer/>
        <BeerList beers={this.state.beers} />
      </div>
    );
  }
}
```

Lifecycle

React provide various methods which notifies when certain stage of the lifecycle occurs called **“Lifecycle methods”**

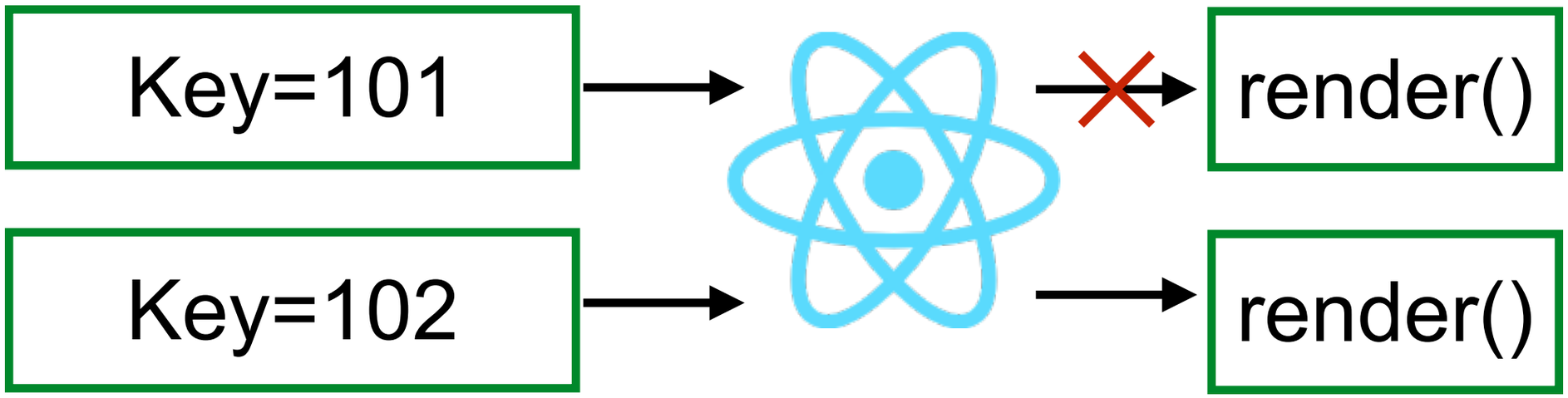
Lifecycle



https://hackernoon.com/reactjs-component-lifecycle-methods-a-deep-dive-38275d9d13c0

Keys

Keys are the elements which helps React to identify components uniquely



https://reactjs.org/docs/lists-and-keys.html

Router

Use React Router

Help to add new screen and flow to application

It’s keeps the URL in sync with data that display on page

https://github.com/ReactTraining/react-router

Advantage of Router

Easy to understand the application flows/views
It can restored any state and view with simple URL
It handled nested views
It maintains a standard structure and behaviour

https://github.com/ReactTraining/react-router

React Router

$npm install --save react-router
$npm install --save react-router-dom

Add router to component

```
import { BrowserRouter, Switch, Route, Link } from 'react-router-dom';

<BrowserRouter>
<div>
  <h2>Main</h2>
  <ul>
    <li>
      <Link to="/beers">List of beers</Link>
    </li>
    <li>
      <Link to="/item/1">Item 1</Link>
    </li>
  </ul>
  <Switch>
    <Route path="/beers" component={Products} />
    <Route exact path="/item/:id" component={ProductDetail} />
  </Switch>
</div>
</BrowserRouter>
```

Add router to component

```
import { BrowserRouter, Switch, Route, Link } from 'react-router-dom';

<BrowserRouter>
<div>
  <h2>Main</h2>
  <ul>
    <li>
      <Link to="/beers">List of beers</Link>
    </li>
    <li>
      <Link to="/item/1">Item 1</Link>
    </li>
  </ul>
  <Switch>
    <Route path="/beers" component={Products} />
    <Route exact path="/item/:id" component={ProductDetail} />
  </Switch>
</div>
</BrowserRouter>
```

Link with parameter

```
<div className="item" key={i}>
  <Link to={`/item/${product.id}`}>
    <div className="product-img">
      <img alt={product.name} src={product.image
    </div>
```

```
this.props.match.params.id
```

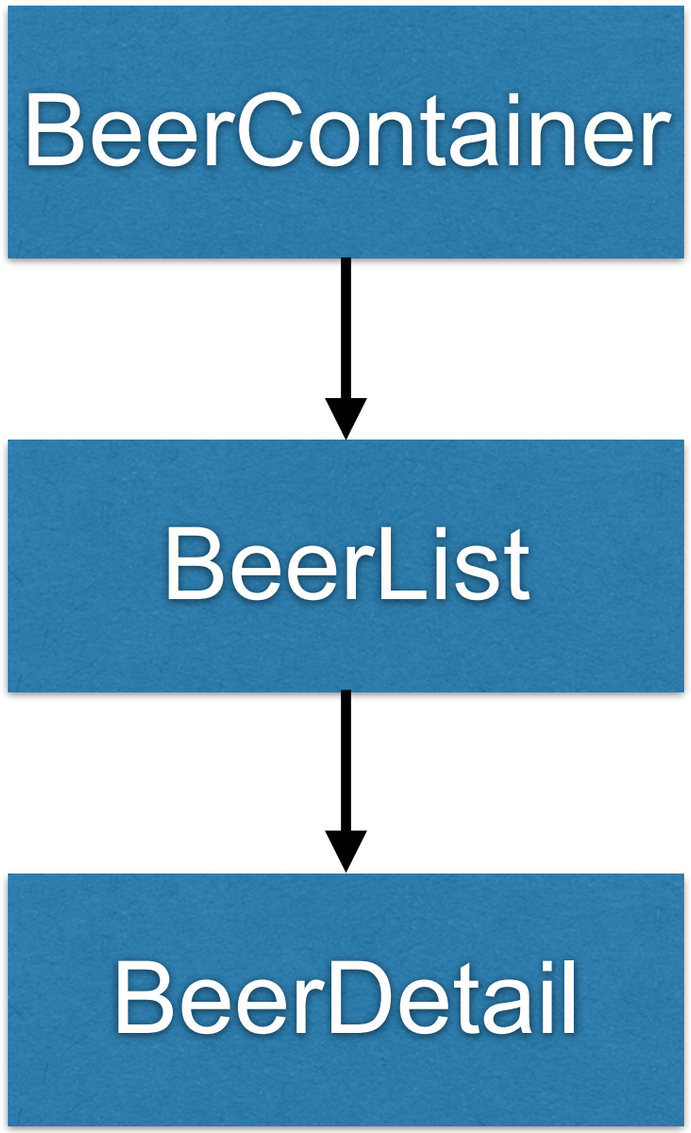
Working with API

PUNK API V2

V2 Documentation

https://api.punkapi.com/v2/beers?per_page=10

https://punkapi.com/documentation/v2



Workshop

https://github.com/up1/workshop-react-note

1. Create project

$npx create-react-app note

2. Create Note component

Delete /src/App*
Create /src/Note.js

2. Create Note component

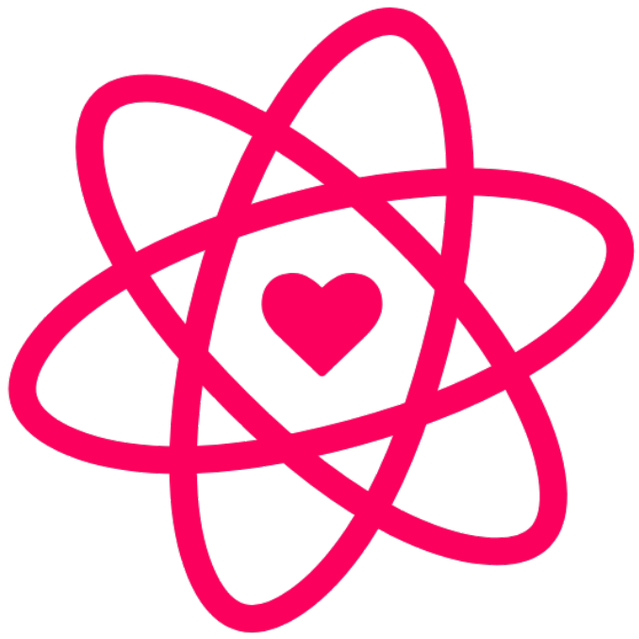
```
import React, { Component } from "react";

class Note extends Component {

    render() {
        return(
            <div className="note">
                <p>Hello React</p>
                <span>
                    <button>Edit</button>
                    <button>Delete</button>
                </span>
            </div>
        )
    }

}

export default Note
```

3. Run project

$yarn start

4. Change Icon in app

Using React-Icon library



https://github.com/react-icons/react-icons

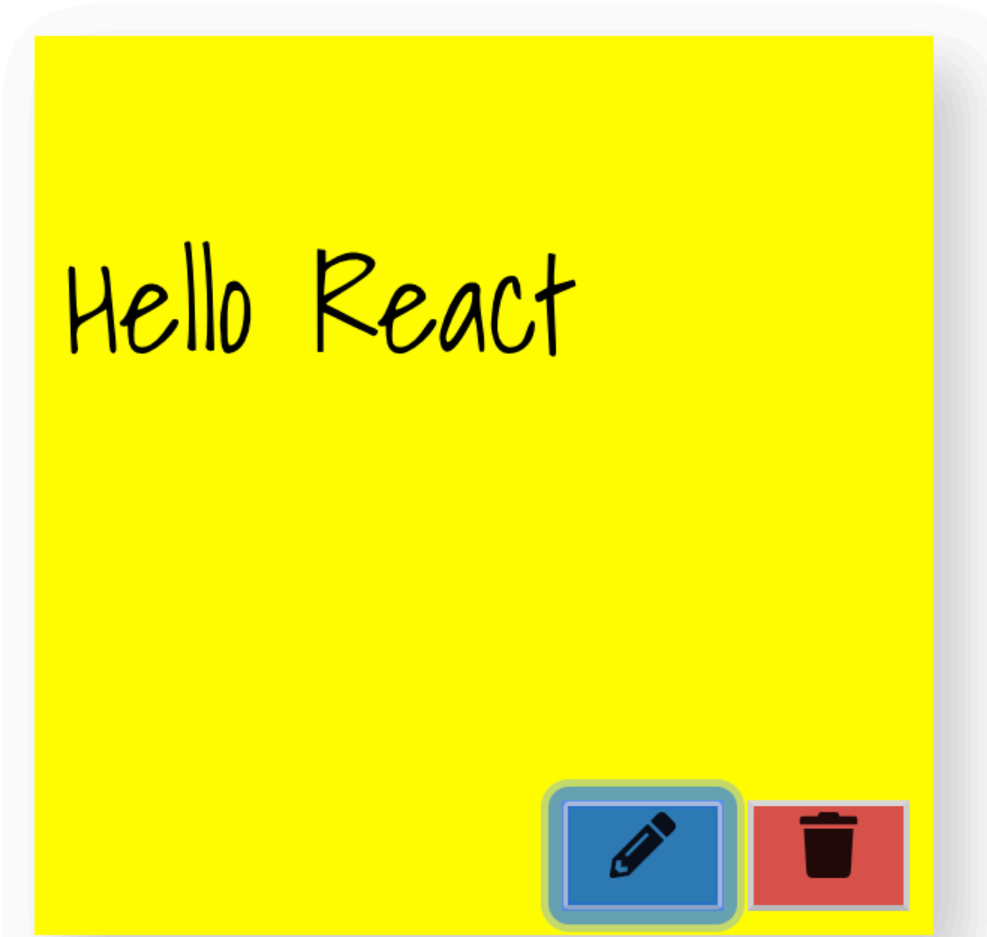
4. Change Icon in app

$npm install react-icons --save

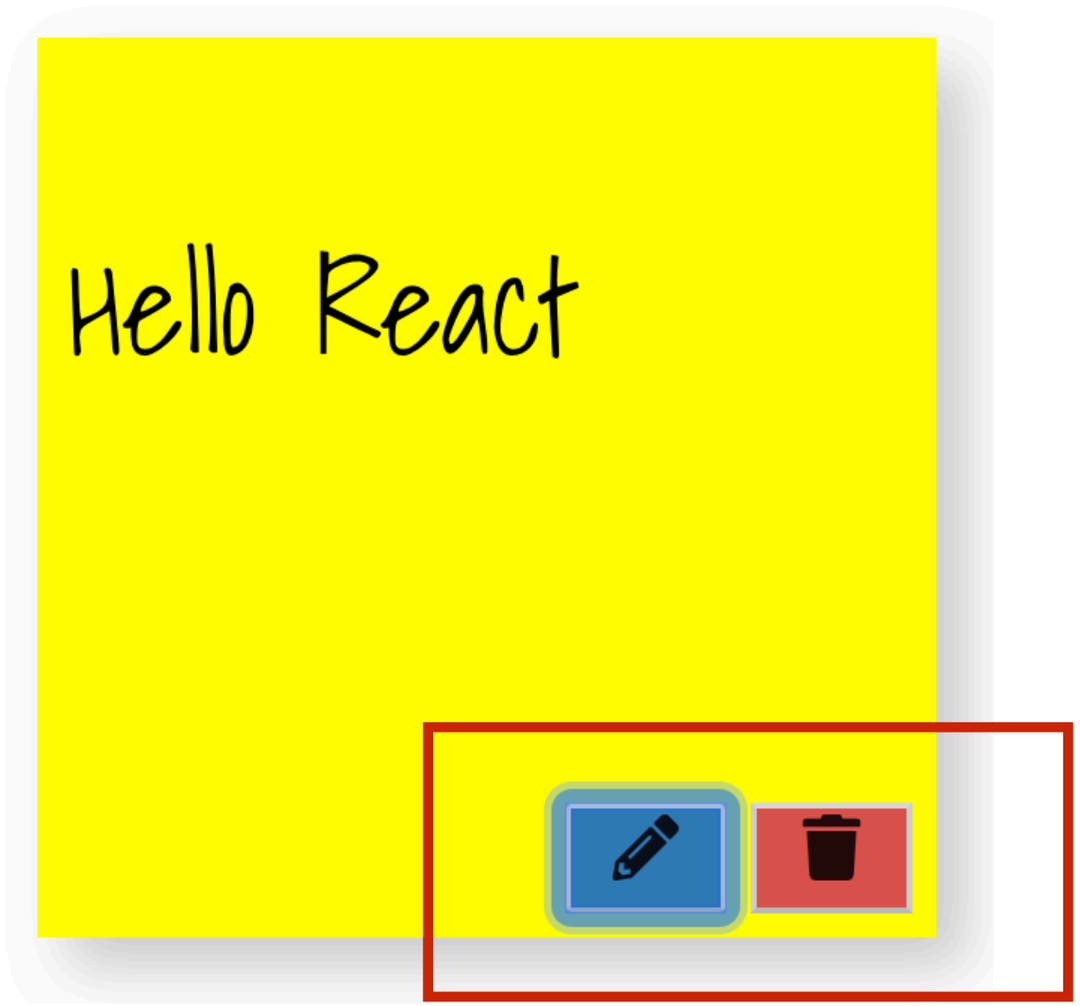
```
import React, { Component } from "react"
import {FaPencilAlt, FaTrash} from "react-icons/fa/index";

class Note extends Component {

    render() {
        return(
            <div className="note">
                <p>Hello React</p>
                <span>
                    <button id="edit"><FaPencilAlt/></button>
                    <button id="remove"><FaTrash/></button>
                </span>
            </div>
        )
    }

}
```

4. Change Icon in app

See result



5. Handling events



5. Handling events

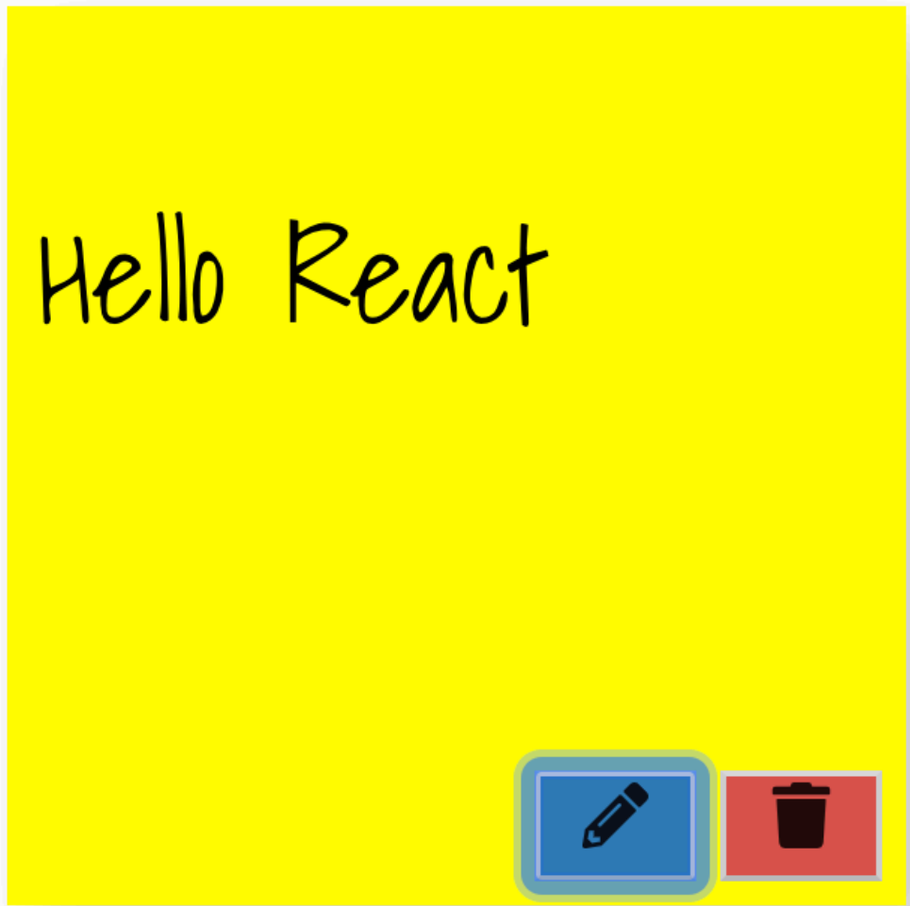
Click on edit and delete button

```
render() {
    return(
        <div className="note">
            <p>Hello React</p>
            <span>
                <button id="edit" onClick={this.edit}>
                    <FaPencilAlt/>
                </button>
                <button id="remove" onClick={this.remove}>
                    <FaTrash/>
                </button>
            </span>
        </div>
    )
}
```

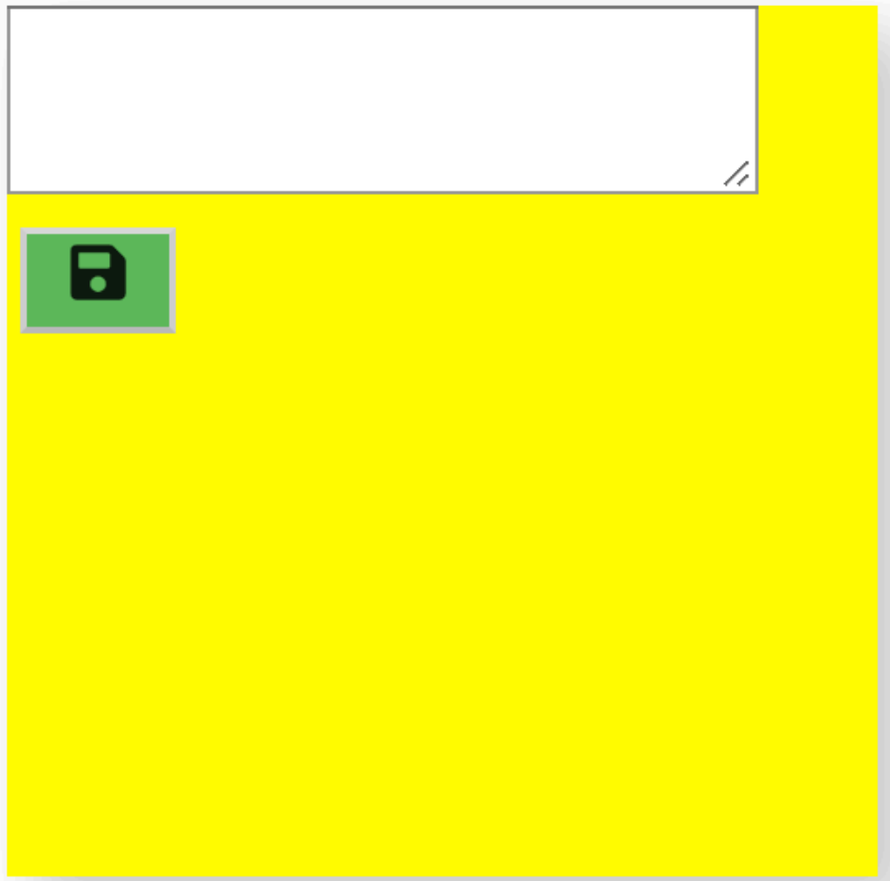
6. Add state to component

isEdit = true | false

isEdit = false



isEdit = true



6. Add state to component

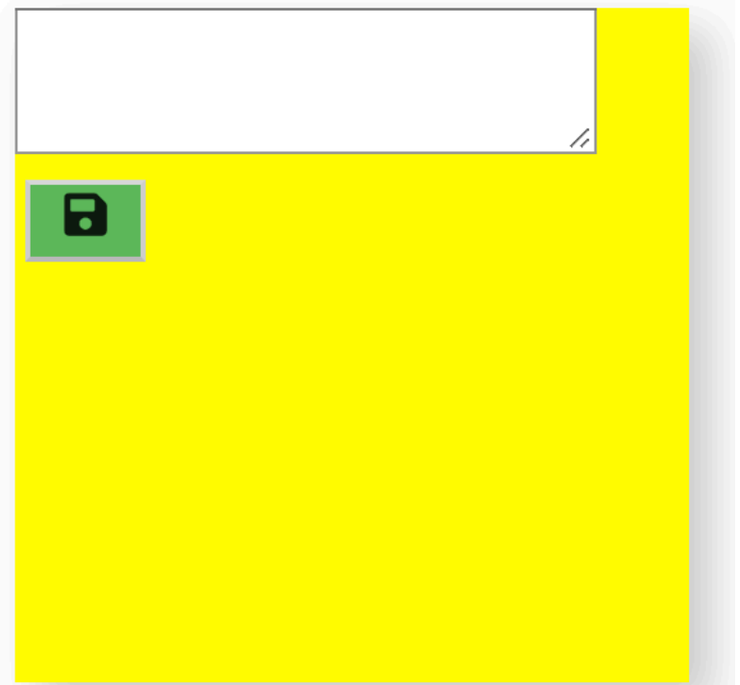
Create state in constructor()

```
class Note extends Component {

    constructor(props) {
        super(props);
        this.state = {
            isEdit: false
        };
        this.edit = this.edit.bind(this)
    }
```

6. Add state to component

Change state when clicked edit button

```
edit() {
    this.setState({
        isEdit: true
    })
}
```

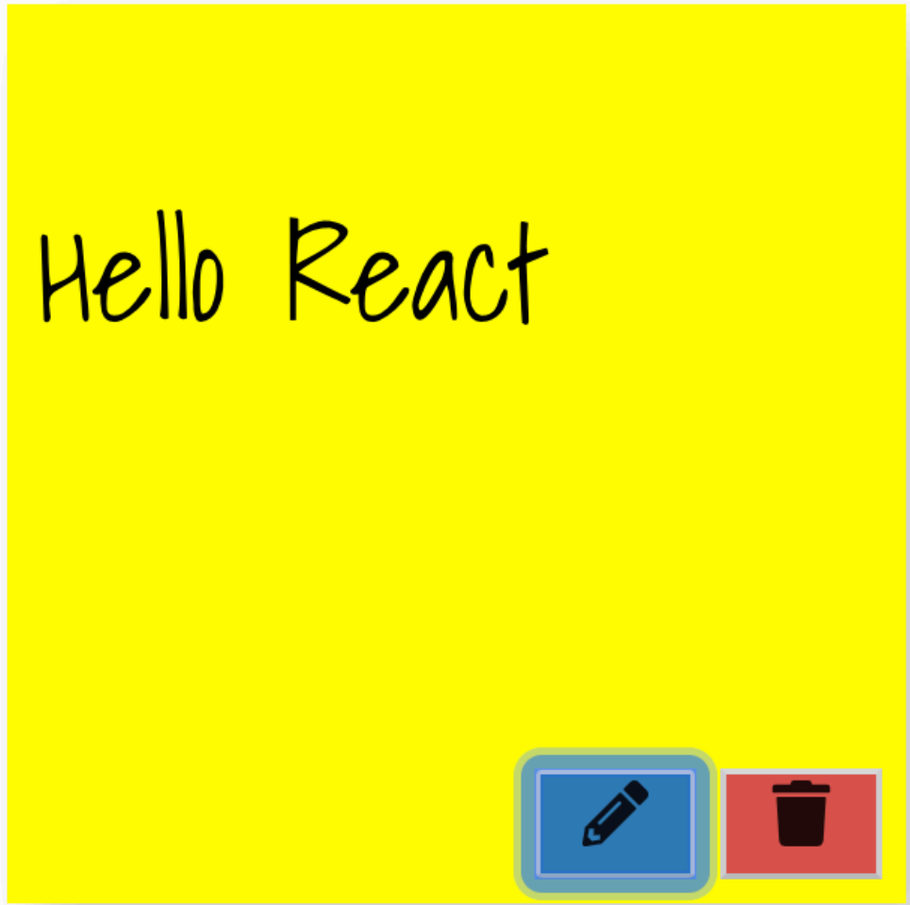


```
renderFormForEdit() {
    return(
        <div className="note">
            <form>
                <textarea />
                <button id="save"><FaSave/></button>
            </form>
        </div>
    )
}
```

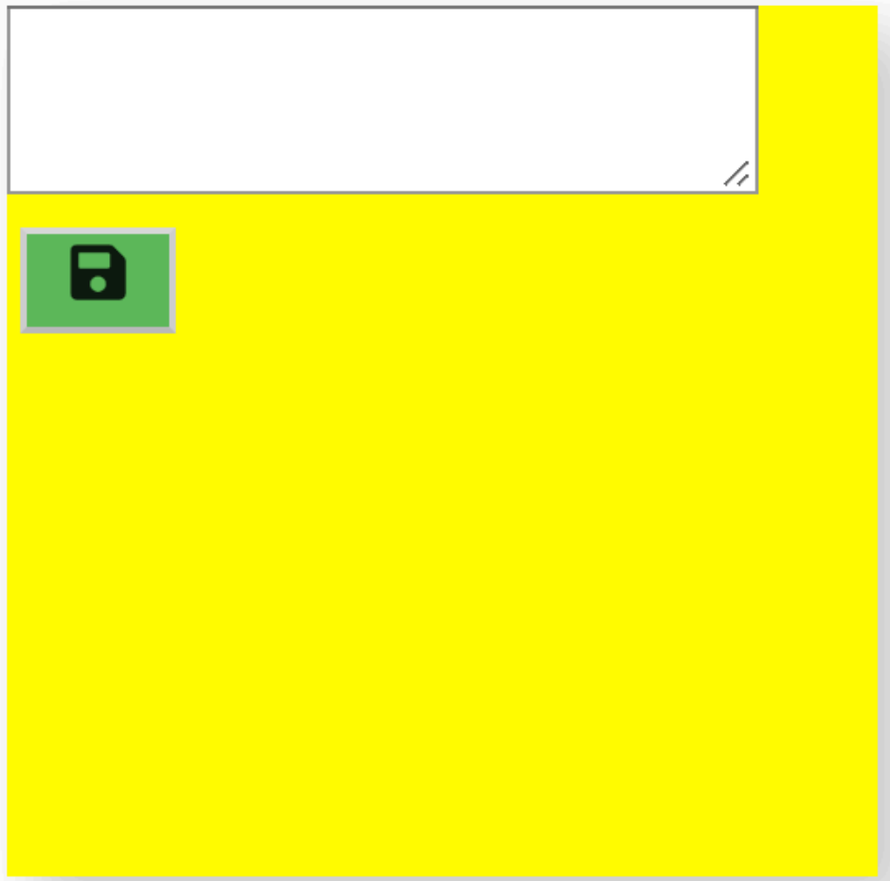
6. Add state to component

UI for display in each state

renderForShow()



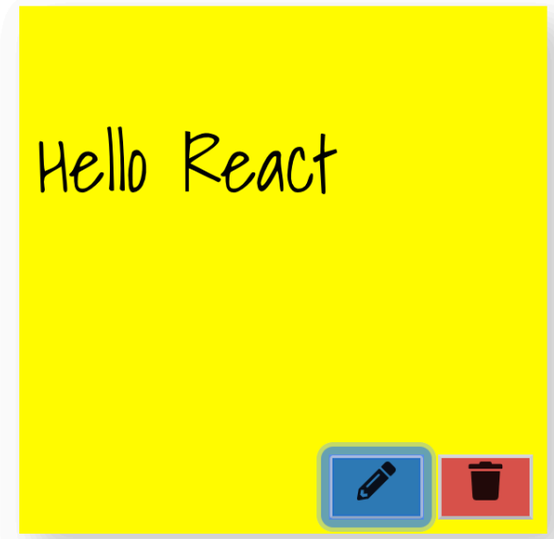
renderFormForEdit()



6. Add state to component

Change state when clicked edit button

```
renderForShow() {
    return(
        <div className="note">
            <p>Hello React</p>
            <span>
                <button id="edit" onClick={this.edit}>
                    <FaPencilAlt/>
                </button>
                <button id="remove" onClick={this.remove}>
                    <FaTrash/>
                </button>
            </span>
        </div>
    )
}
```



6. Add state to component

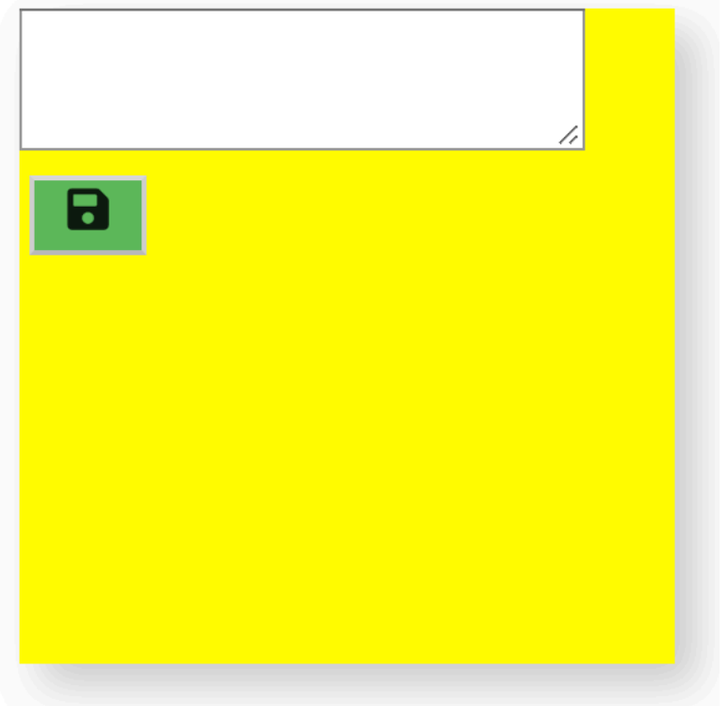
Condition to display UI in each state

```
render() {
    if(this.state.isEdit) {
        return this.renderFormForEdit()
    }
    return this.renderForShow()
}
```

7. Save data from textarea

Using refs

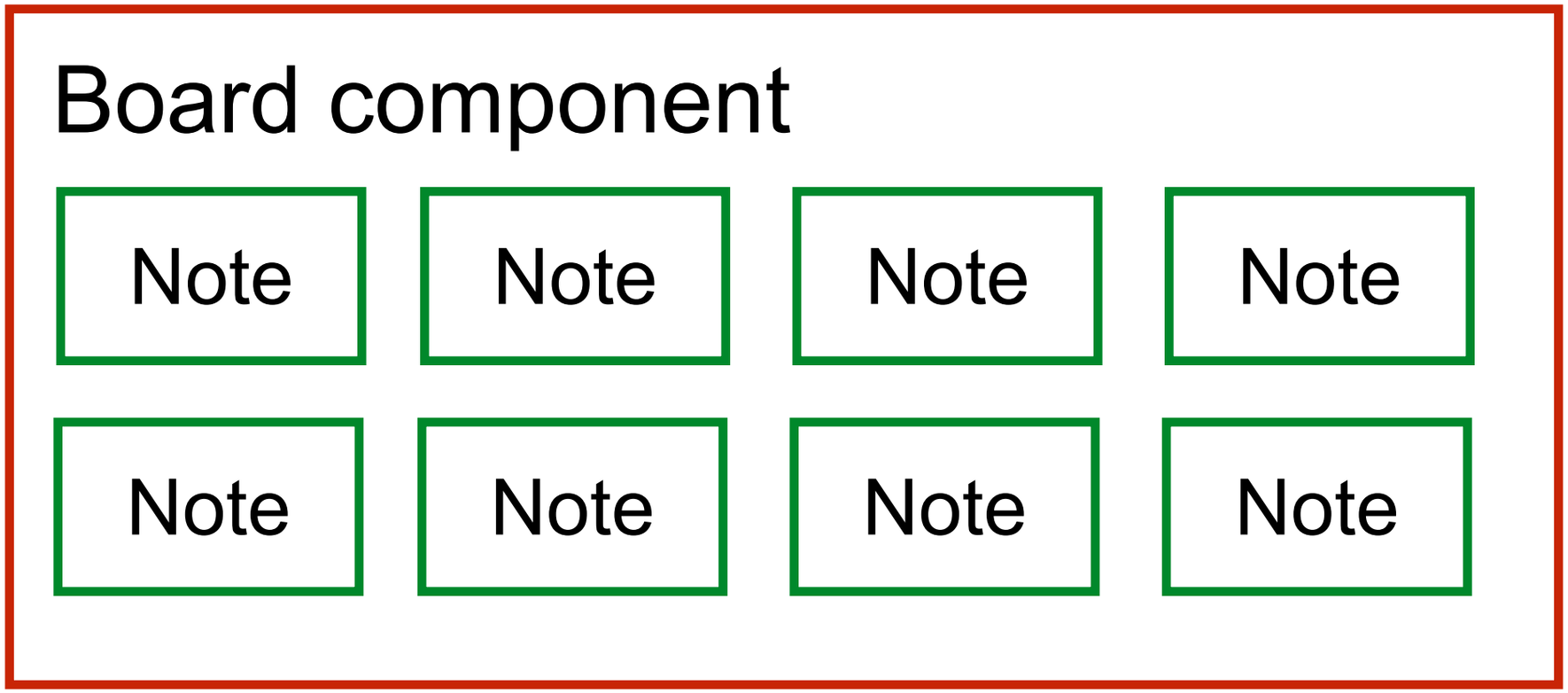
```
save() {
    alert(this._newText.value)
}
```



```
renderFormForEdit() {
    return(
        <div className="note">
            <form>
                <textarea ref={ input => this._newText = input} />
                <button id="save" onClick={this.save}><FaSave/></button>
            </form>
        </div>
    )
}
```

8. Parent and child component

Parent = Board
Child = Note



8. Parent and child component

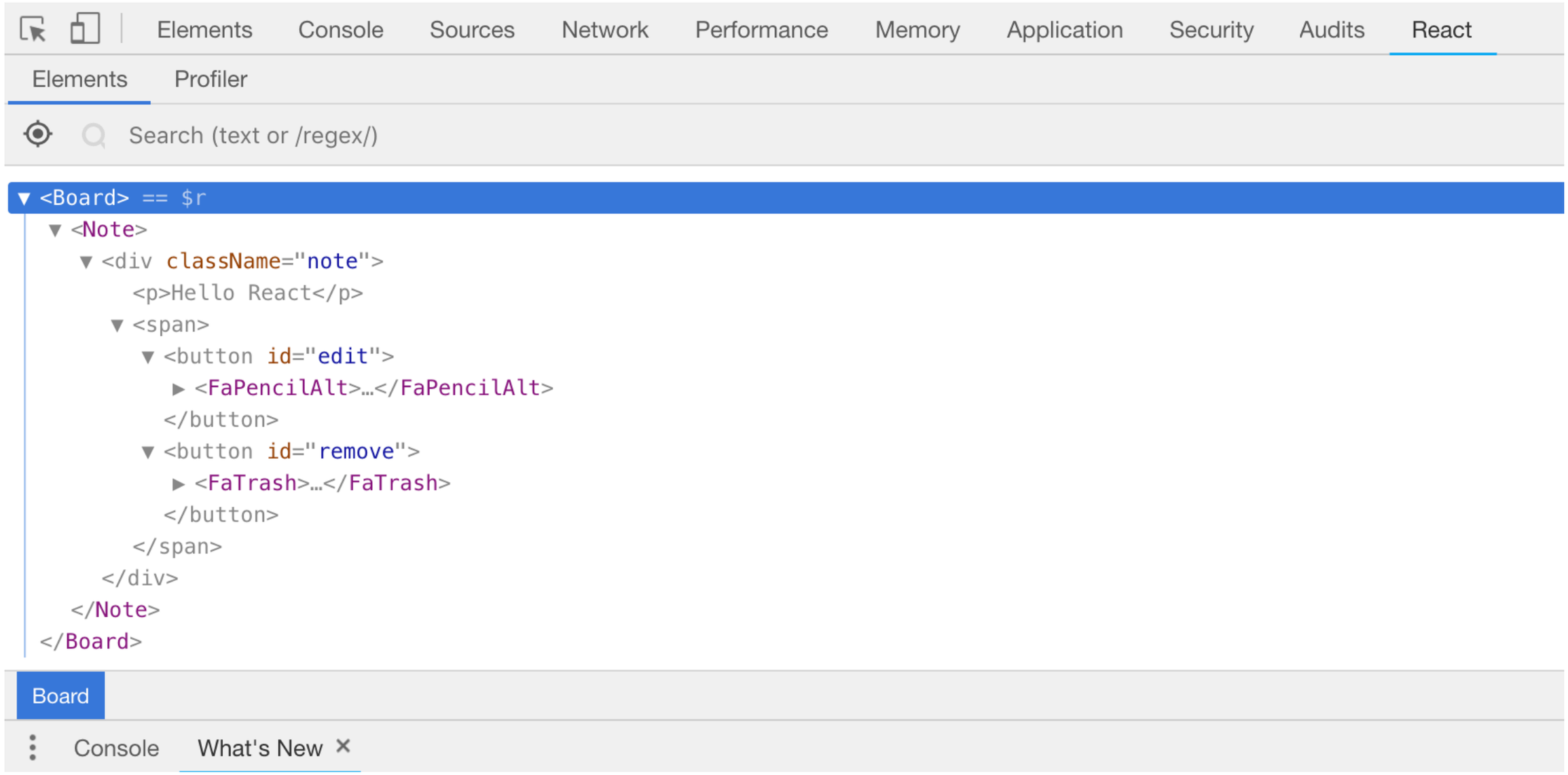
Create Board component and add a Note

/src/Board.js

```
import React, {Component} from "react"
import Note from "./Note"

class Board extends Component {
    render() {
        return(
            <Note/>
        )
    }
}

export default Board
```

8. Parent and child component

Run and see result in React Developer Tools



8. Parent and child component

Add more Notes in a board

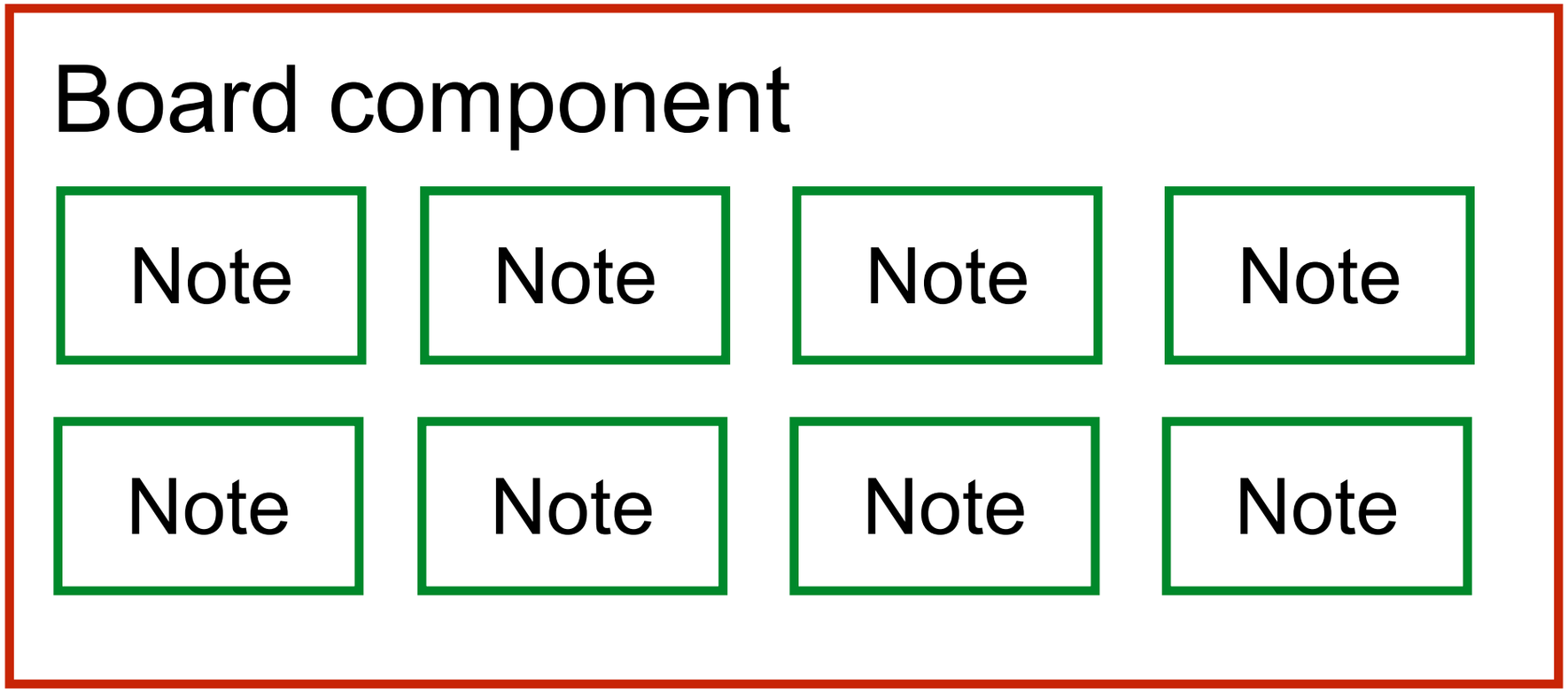
```
class Board extends Component {
    render() {
        return(
            <div>
                <Note/>
                <Note/>
                <Note/>
            </div>
        )
    }
}
```

9. Send data from parent to child

Board Component

Stateful component

Hold data of all notes



Board component

Create list of all notes in state

```
class Board extends Component {

    constructor(props) {
        super(props);
        this.state = {
            notes: [
                {
                    id: 1,
                    title: "Note 1"
                },
                {
                    id: 2,
                    title: "Note 2"
                }
            ]
        }

    }
```

Board component

Send data to Note with props

```
eachNote(note, index) {
    return(
        <Note key={index} index={index}>{note.title}</Note>
    )
}
```

```
render() {
    return(
        <div>
            {this.state.notes.map(this.eachNote)}
        </div>
    )
}
```

https://reactjs.org/docs/lists-and-keys.html#keys

Note component

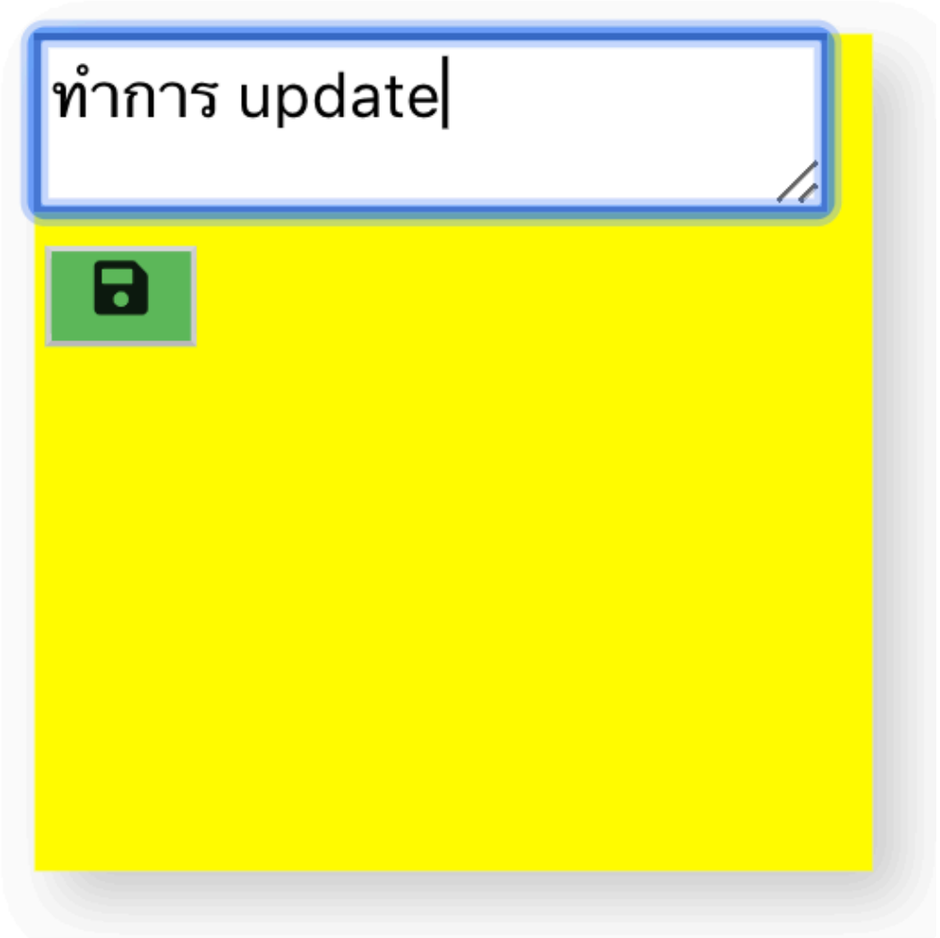
Read data from props

```
renderForShow() {
    return(
        <div className="note">
            <p>{this.props.children}</p>
            <span>
                <button id="edit" onClick={this.edit}>
                    <FaPencilAlt/>
                </button>
                <button id="remove" onClick={this.remove}>
                    <FaTrash/>
                </button>
            </span>
        </div>
    )
}
```

See result

Edit/Update Note

Edit Note



Edit Note

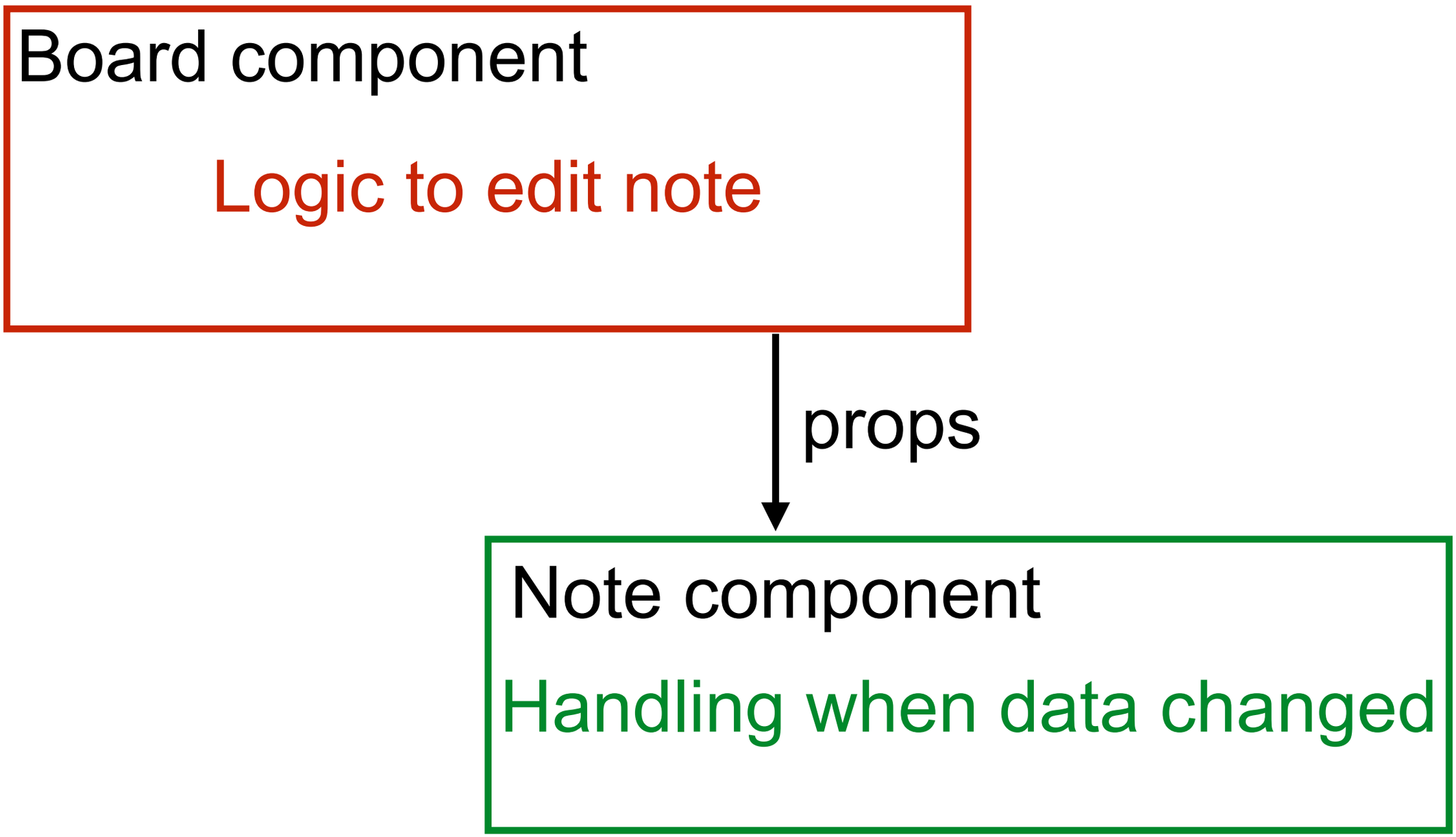
Board component

Logic to edit note

Note component

Handling when data changed

Edit Note



Board component

Create function edit() and send to child via props

```
edit(newTitle, index) {
    console.log('Edit ', newTitle, index);
    this.setState(prevState => ({
        notes: prevState.notes.map(
            note => (note.id === index) ?
                        {...note, title: newTitle} : note
        )
    }))
}

eachNote(note, index) {
    return(
        <Note key={note.id} index={note.id} onChange={this.edit}>
            {note.title}
        </Note>
    )
}
```

Note component

Handling onSubmit() and call onChange() from parent

```
save(e) {
    e.preventDefault();
    this.props.onChange(this._newText.value, this.props.index);
    this.setState({
        isEdit: false
    })
}

renderFormForEdit() {
    return(
        <div className="note">
            <form onSubmit={this.save}>
                <textarea ref={ input => this._newText = input} />
                <button id="save"><FaSave/></button>
            </form>
        </div>
    )
}
```

Delete Note

Delete Note

Board component

Logic to delete note

Note component

Handling when deleted

Board component

Try to delete !!

```
delete(index) {
    console.log('Delete ', index);
    this.setState(prevState => ({
        notes: prevState.notes.filter(note => note.id !== index)
    }))
}
```

Add new Note

Board component

Try to add new note !!

```
addNew(newTitle) {
    this.setState(prevState => ({
        notes: [
            ...prevState.notes,
            {
                id: 3,
                title: newTitle
            }
        ]
    }))
}
```

Add new data to list

```
render() {
    return(
        <div className="board">
            <button id="add" onClick={this.addNew.bind(null, 'New Title')}>
                <FaPlus/>
            </button>
            {this.state.notes.map(this.eachNote)}
        </div>
    )
}
```

Board component

Try to add new note !!

```
addNew(newTitle) {
    this.setState(prevState => ({
        notes: [
            ...prevState.notes,
            {
                id: 3,
                title: newTitle
            }
        ]
    }))
}

render() {
    return(
        <div className="board">
            <button id="add" onClick={this.addNew.bind(null, 'New Title')}>
                <FaPlus/>
            </button>
            {this.state.notes.map(this.eachNote)}
        </div>
    )
}
```

Binding with function

See result

Fix error !!

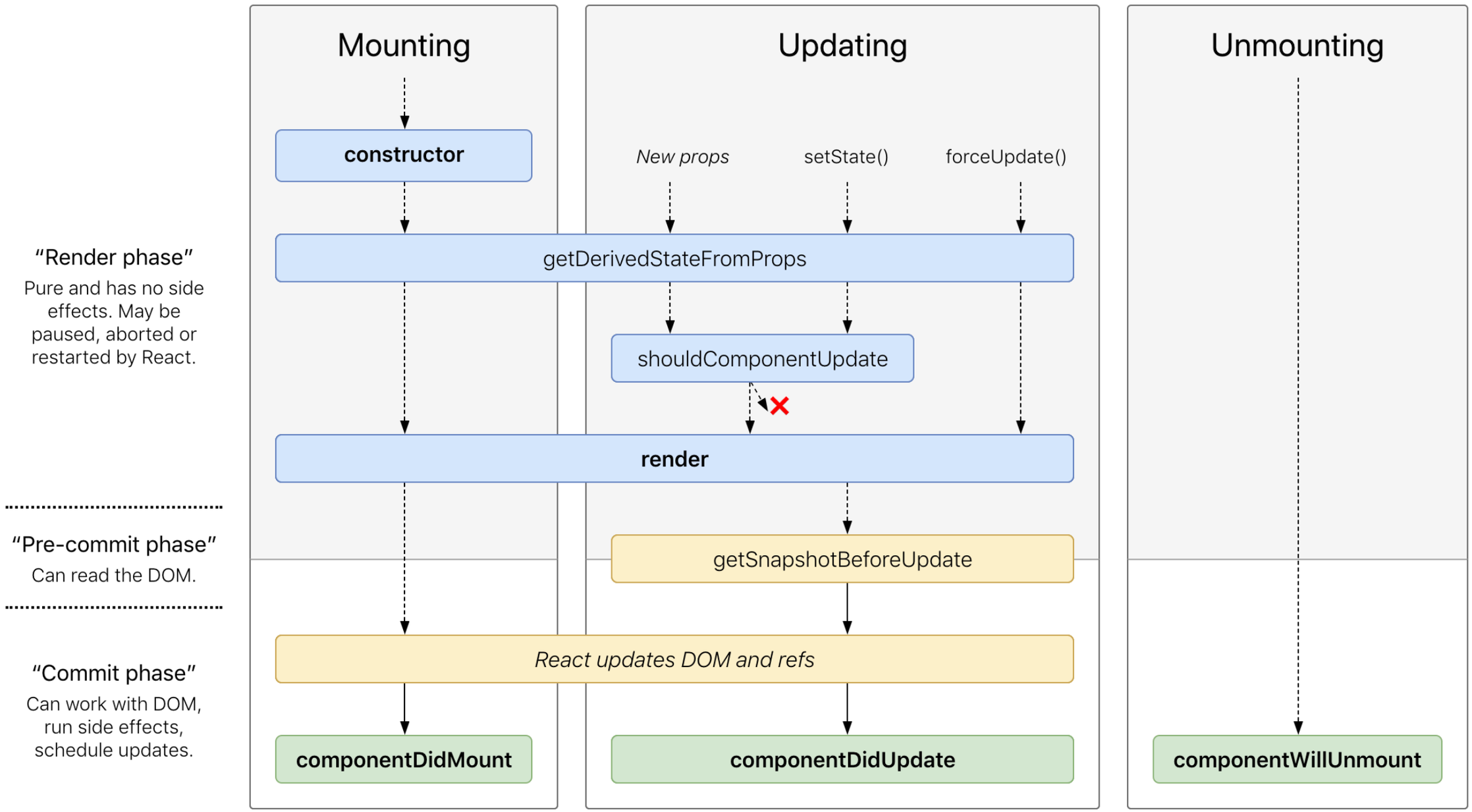
```
Render in note  3
Component did update
▶Warning: Encountered two children with the same key, `3`. Keys should b
to be duplicated and/or omitted — the behavior is unsupported and could
    in div (at Board.js:81)
    in Board (at src/index.js:7)
Render in note  1
Render in note  2
```

Component Life Cycle

Component Life Cycle

Mounting
Update
Unmount

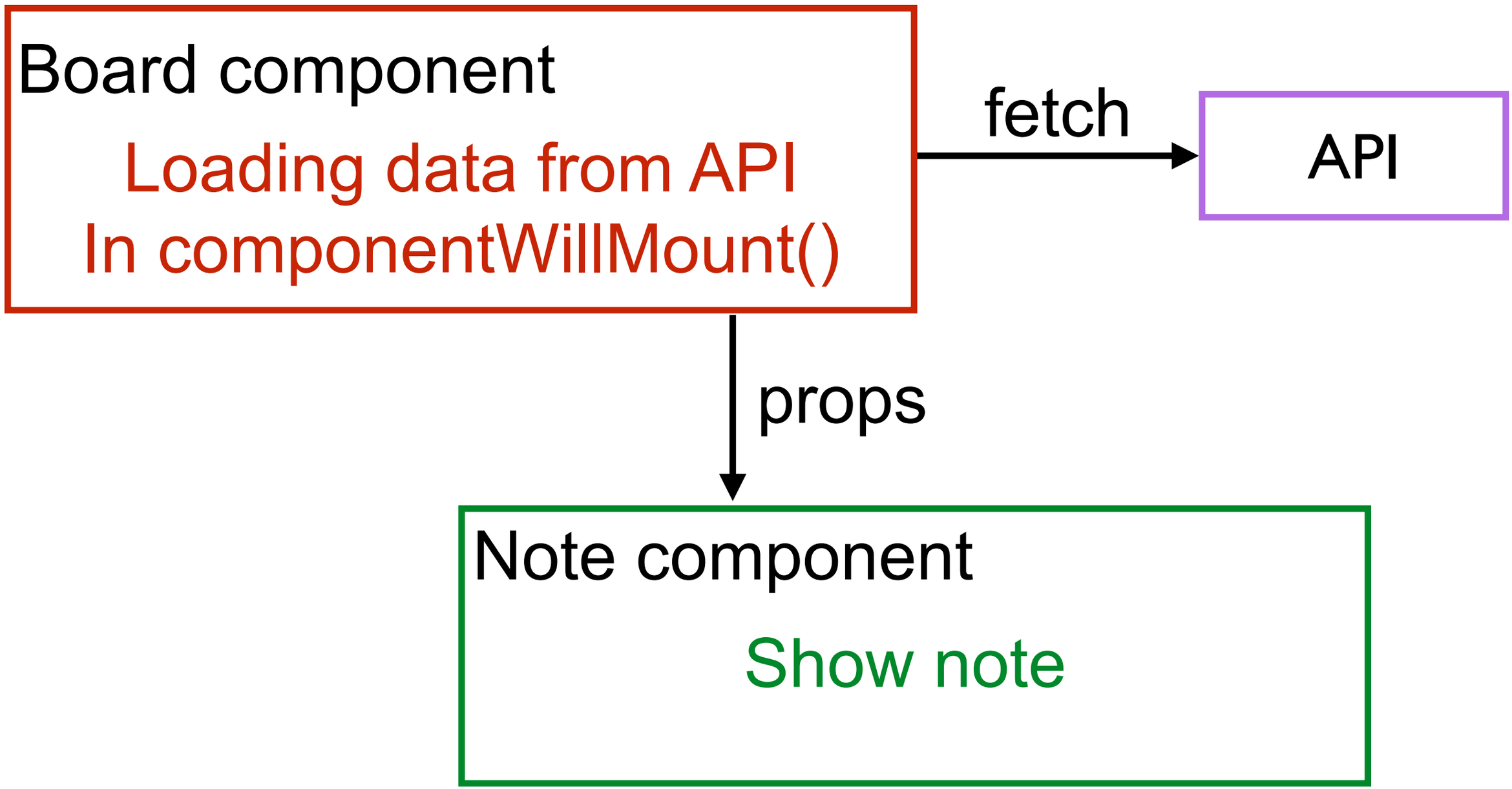
Component Life Cycle



https://reactjs.org/docs/react-component.html

Loading data from API

Loading data from API



https://baconipsum.com/json-api/

Board component

```
componentWillMount() {
    console.log("Component will mount");
    fetch(`https://baconipsum.com/api/?type=meat-and-filler&sentences=3`)
        .then(response => response.json())
        .then(json => json[0].split('. ')
            .forEach(input => this.addNew(input.substring(0, 10))))
}
```

https://baconipsum.com/json-api/

Deploy