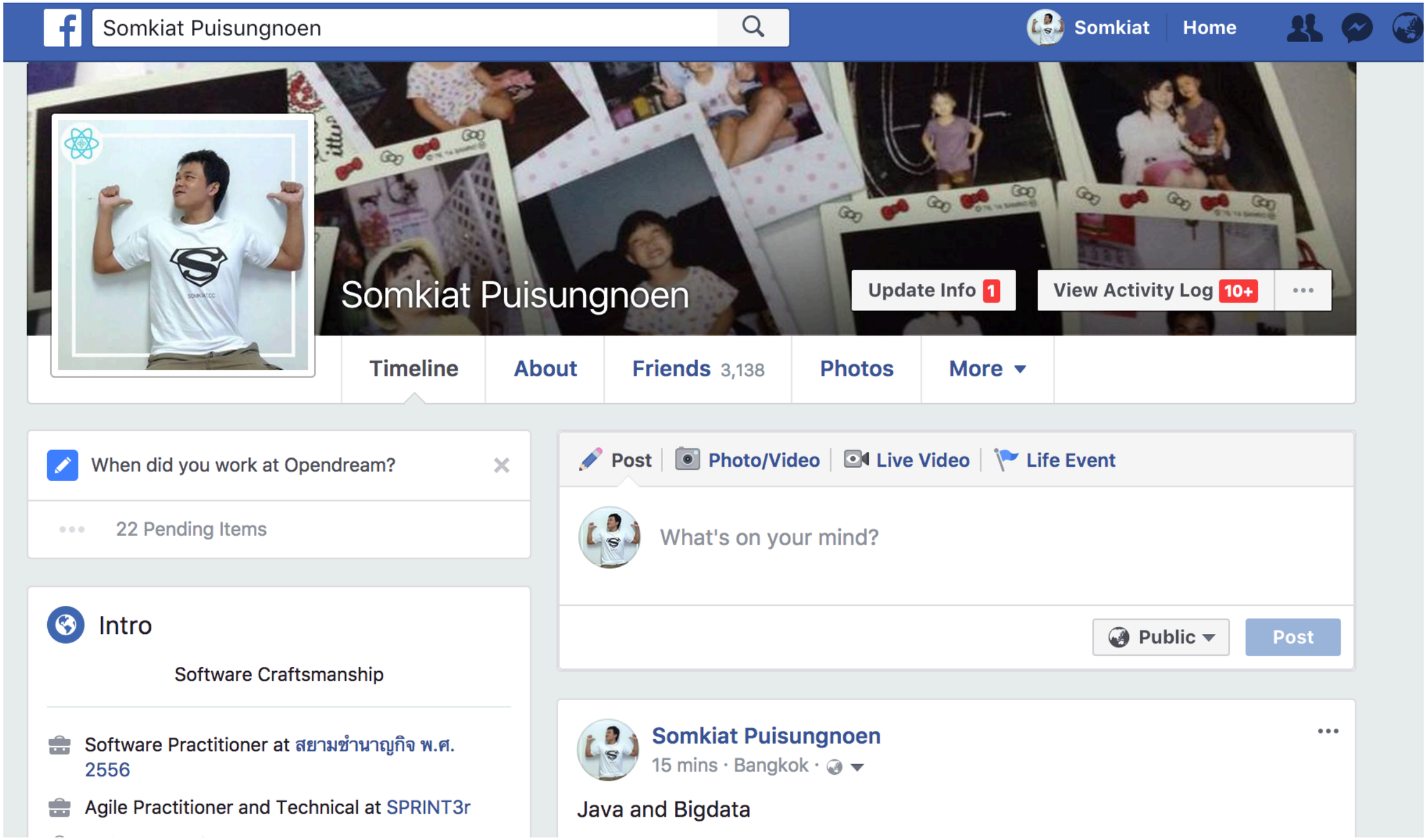
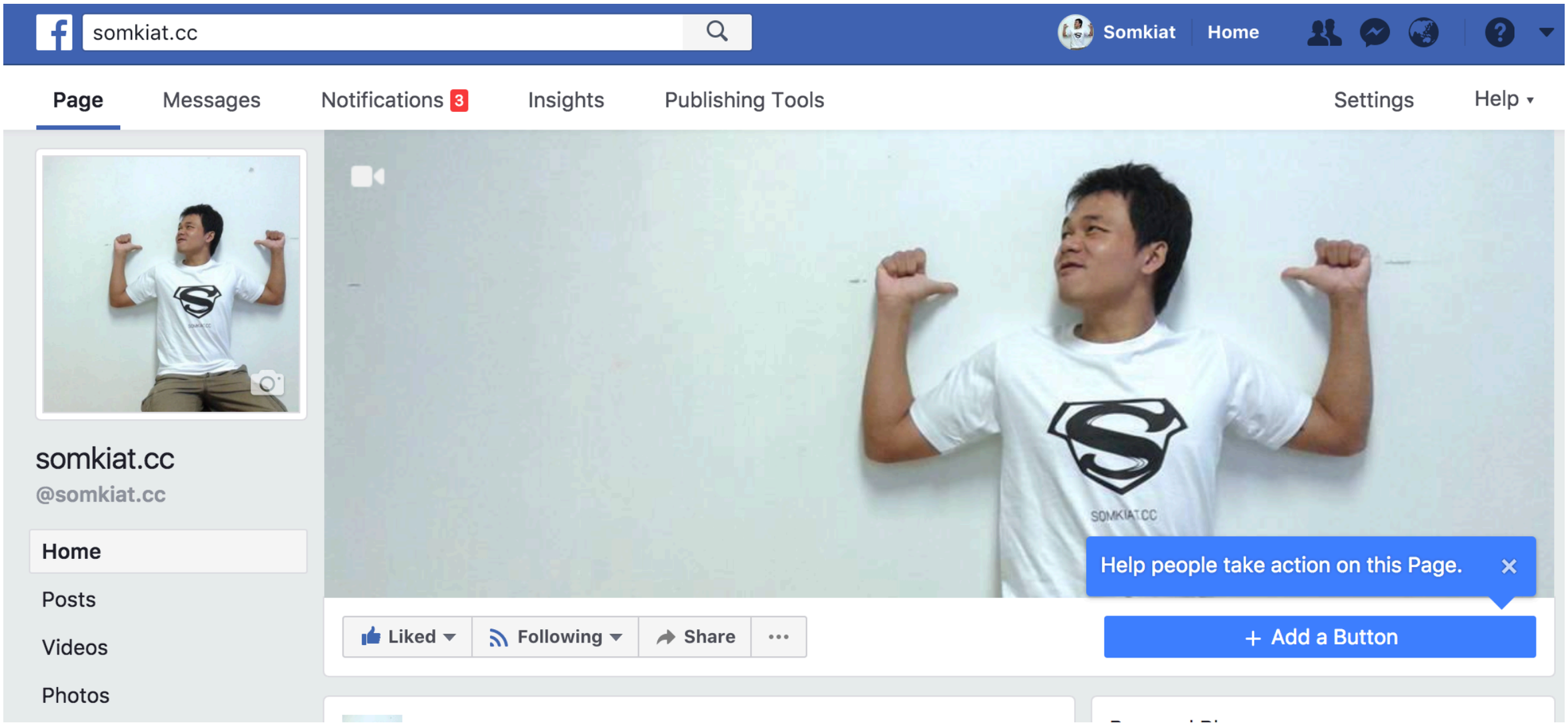
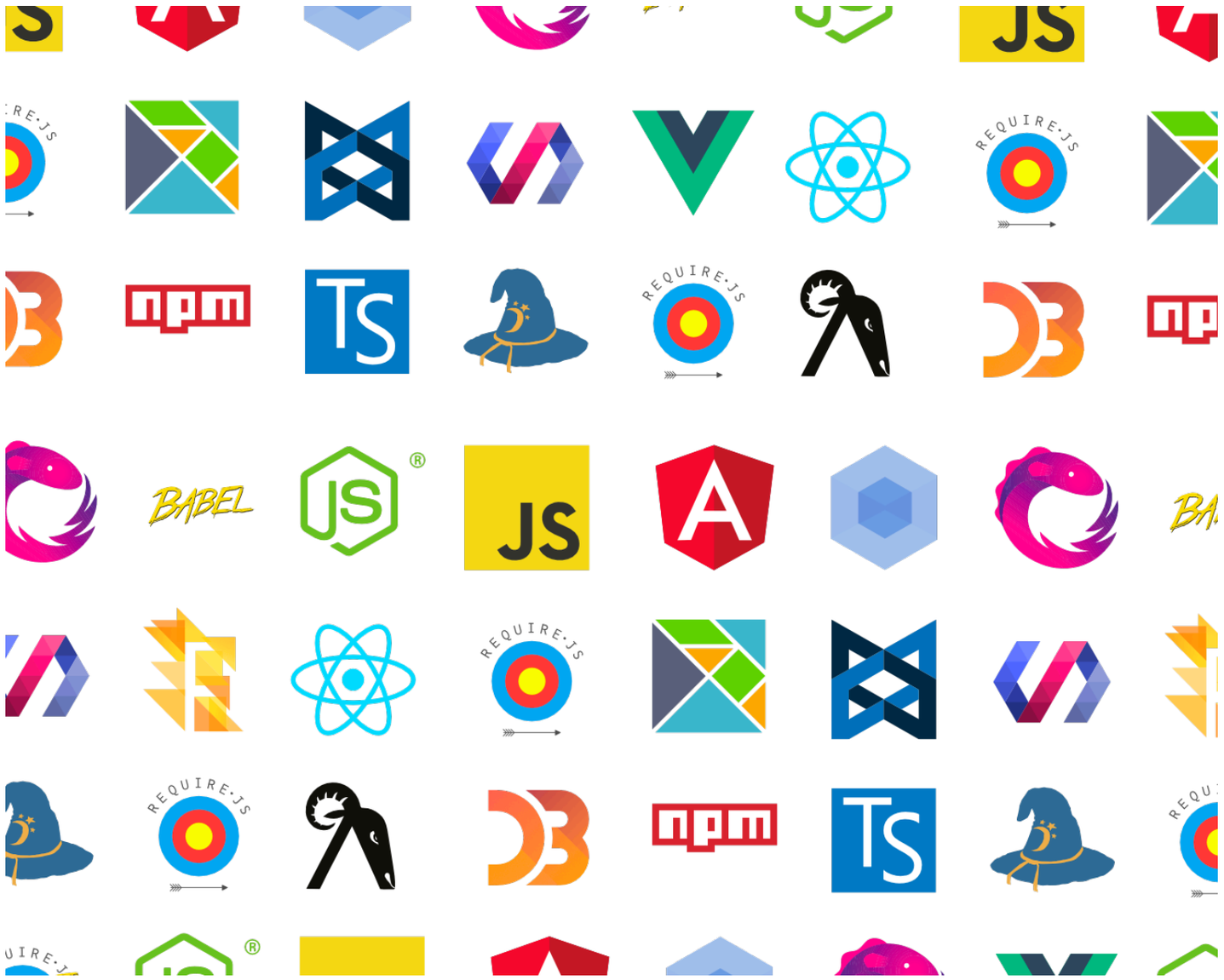
Learning React

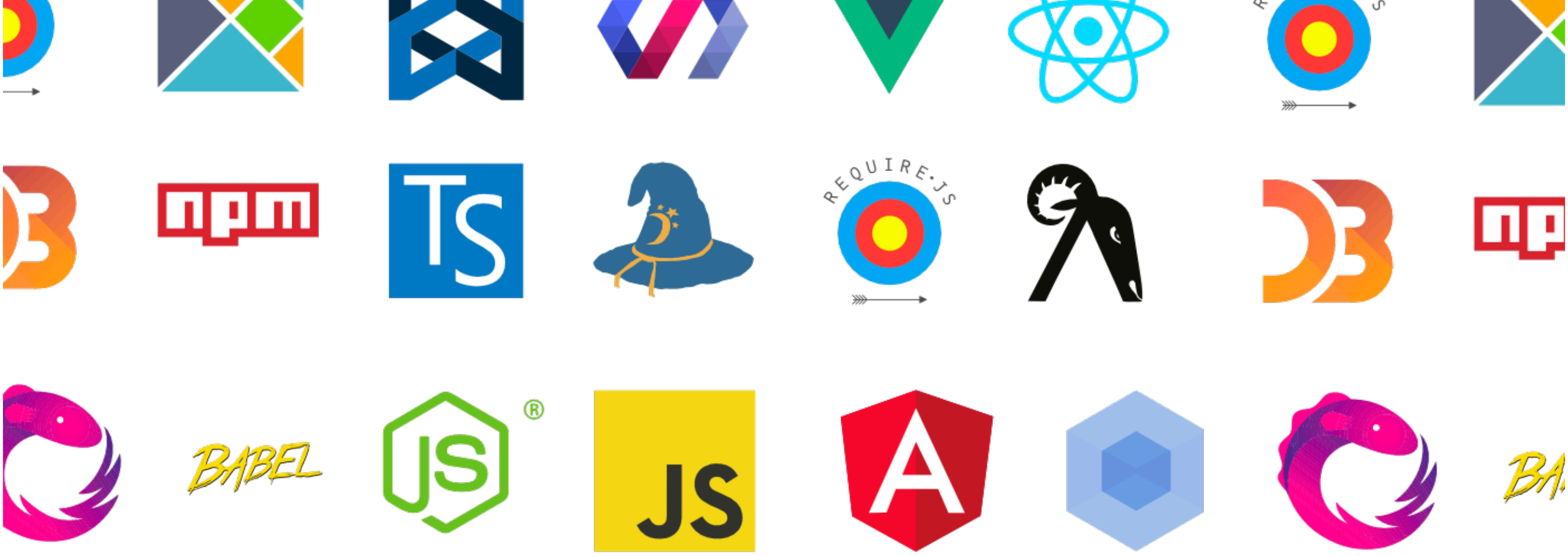
Basic of React



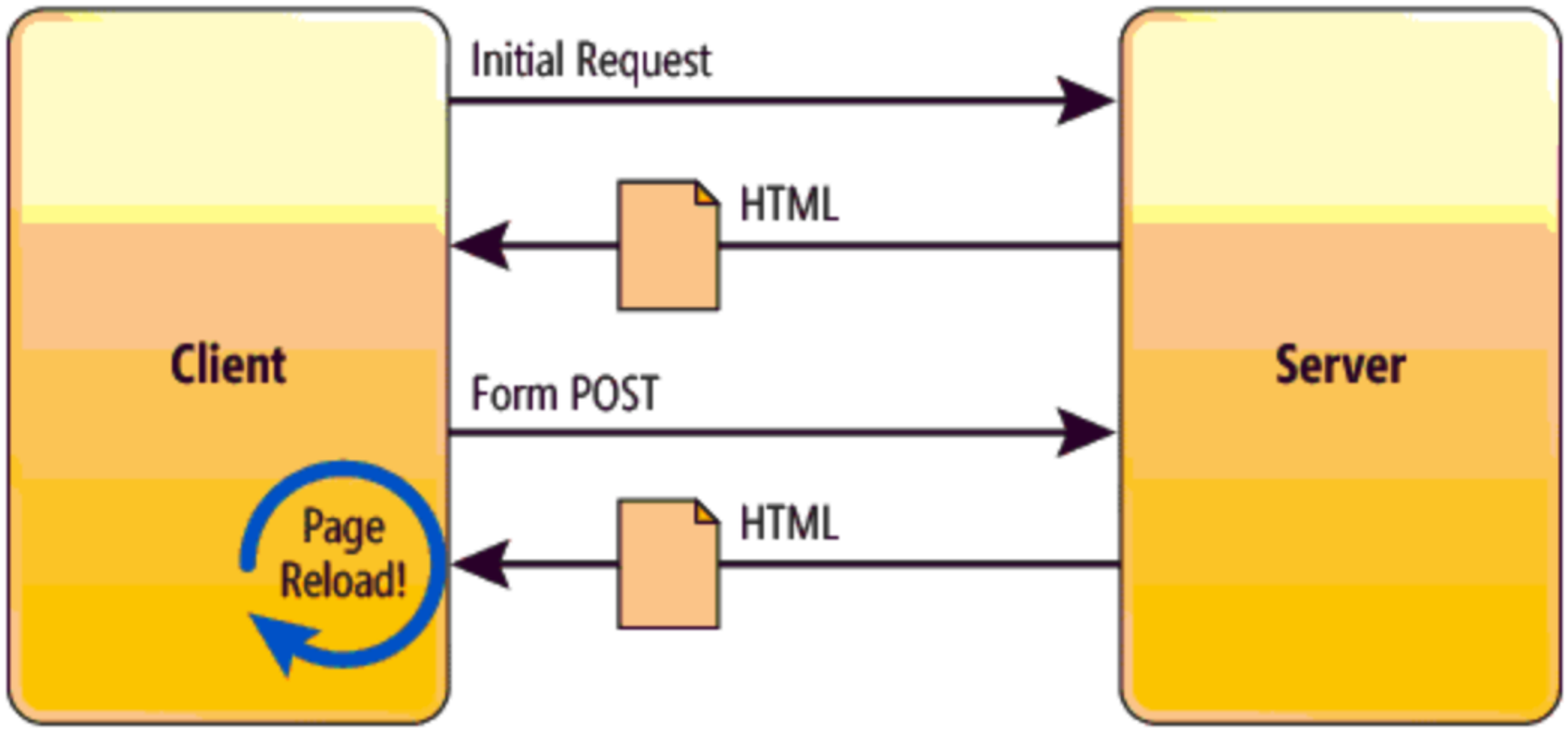


JavaScript

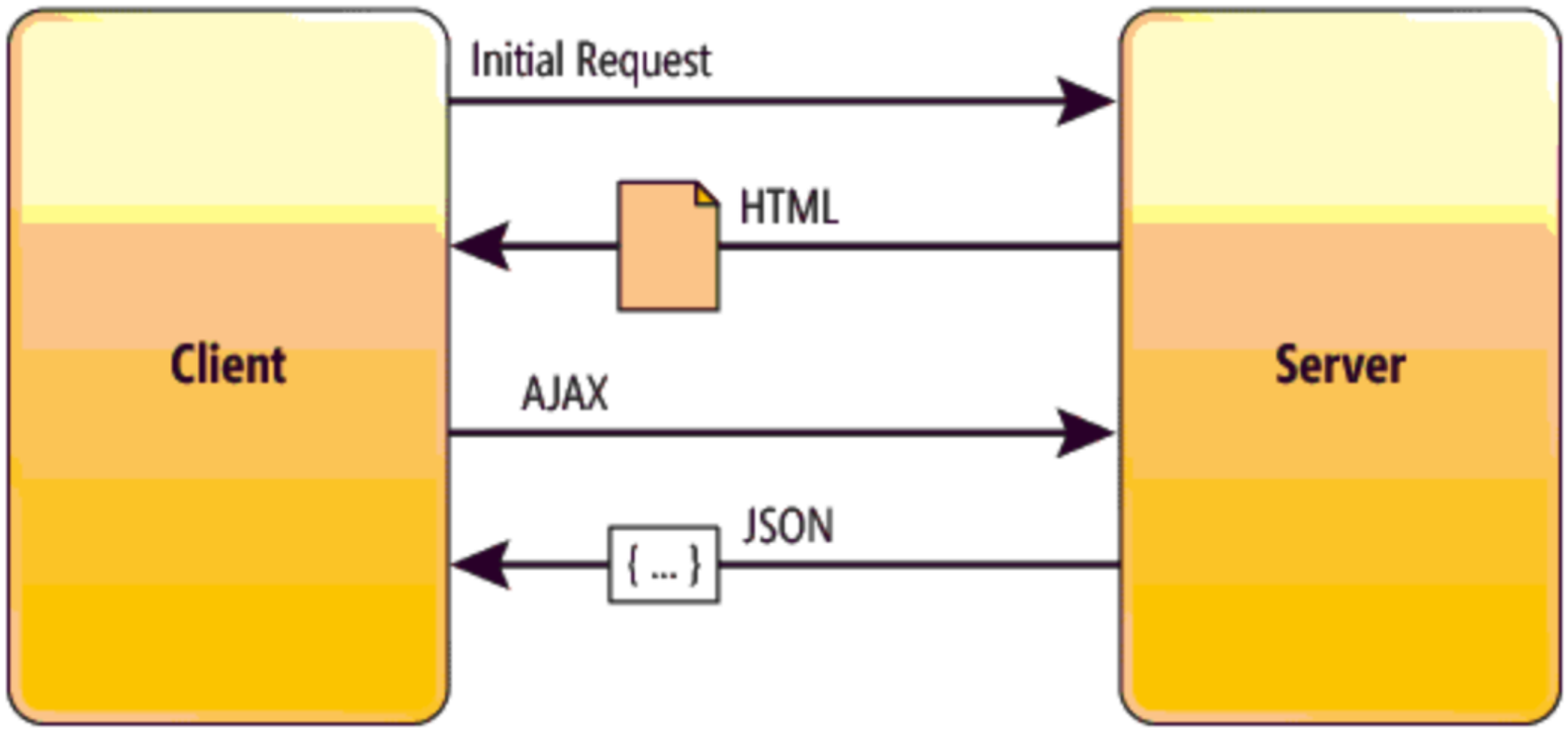




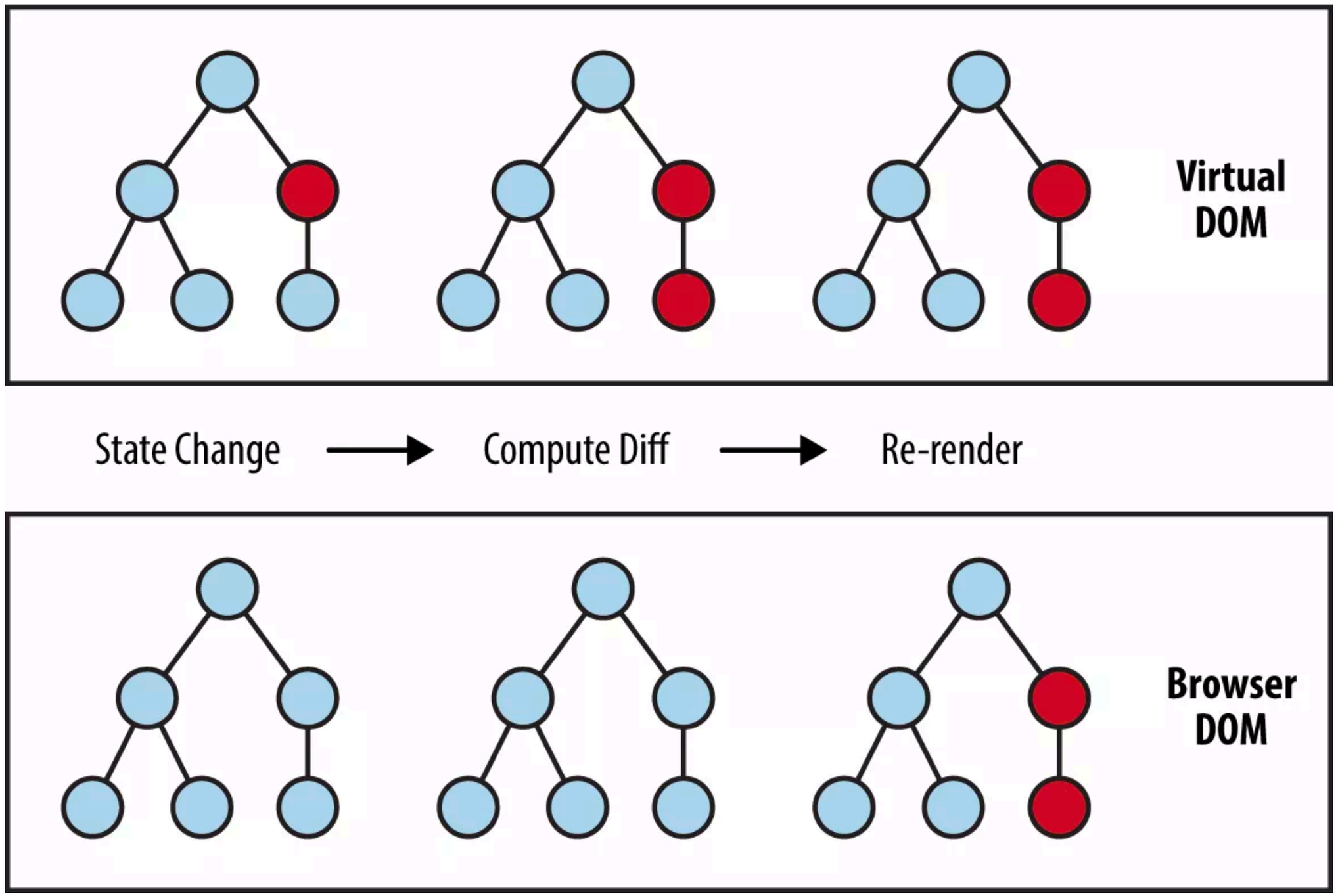
Traditional



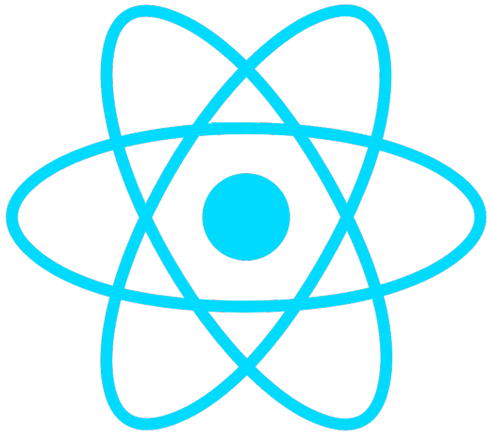
Single Page Application (SPA)



DOM vs Virtual DOM



React and Angular



Learning path

Develop

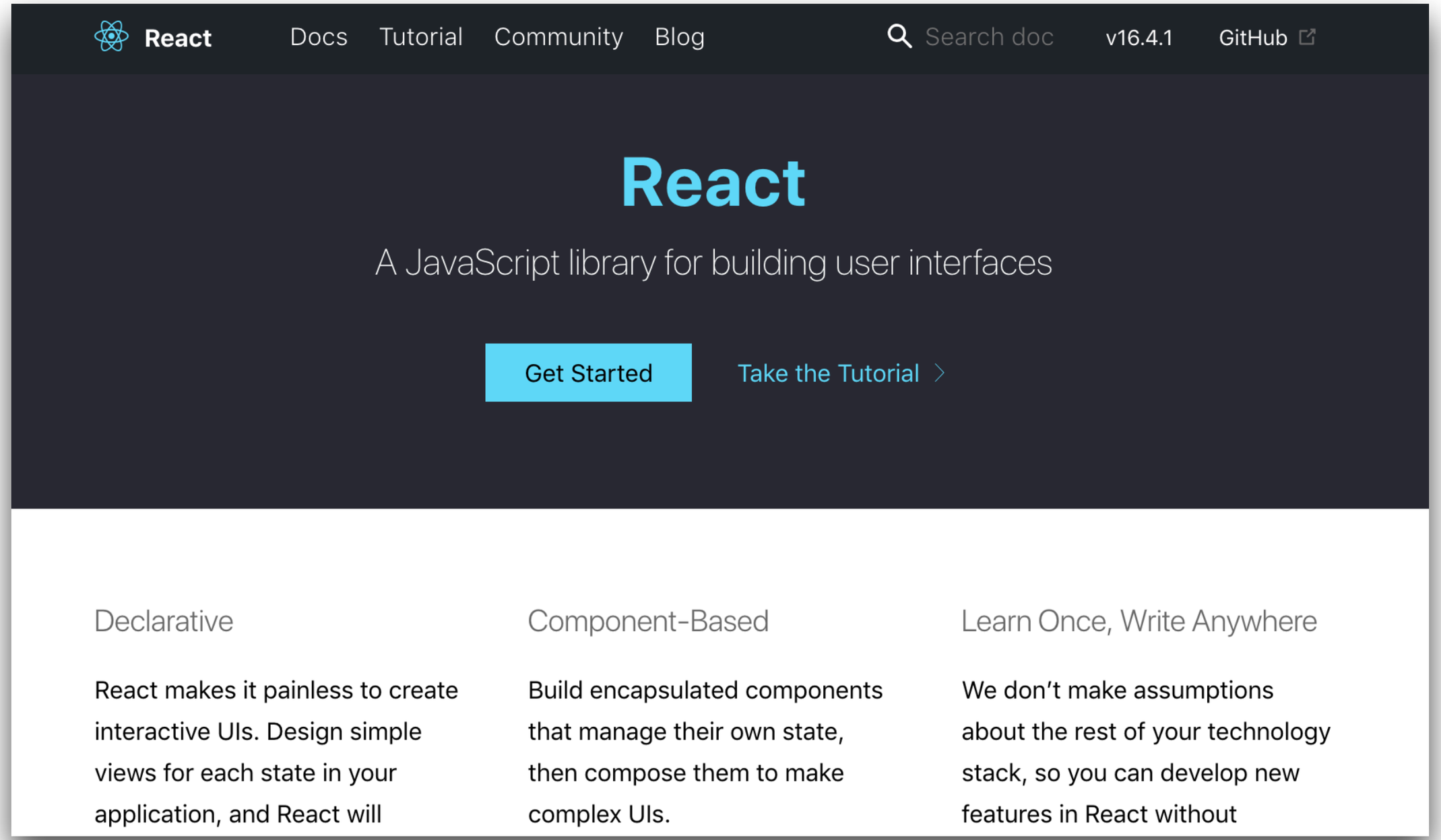
Build

Deploy

Basic of React

1. Introduction to React
2. Install Yarn
3. Create a react application
4. Design and Develop application
5. Deploy application to Github Pages
6. Hooks

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

JSX, State/Props, HoC, Routing

State management

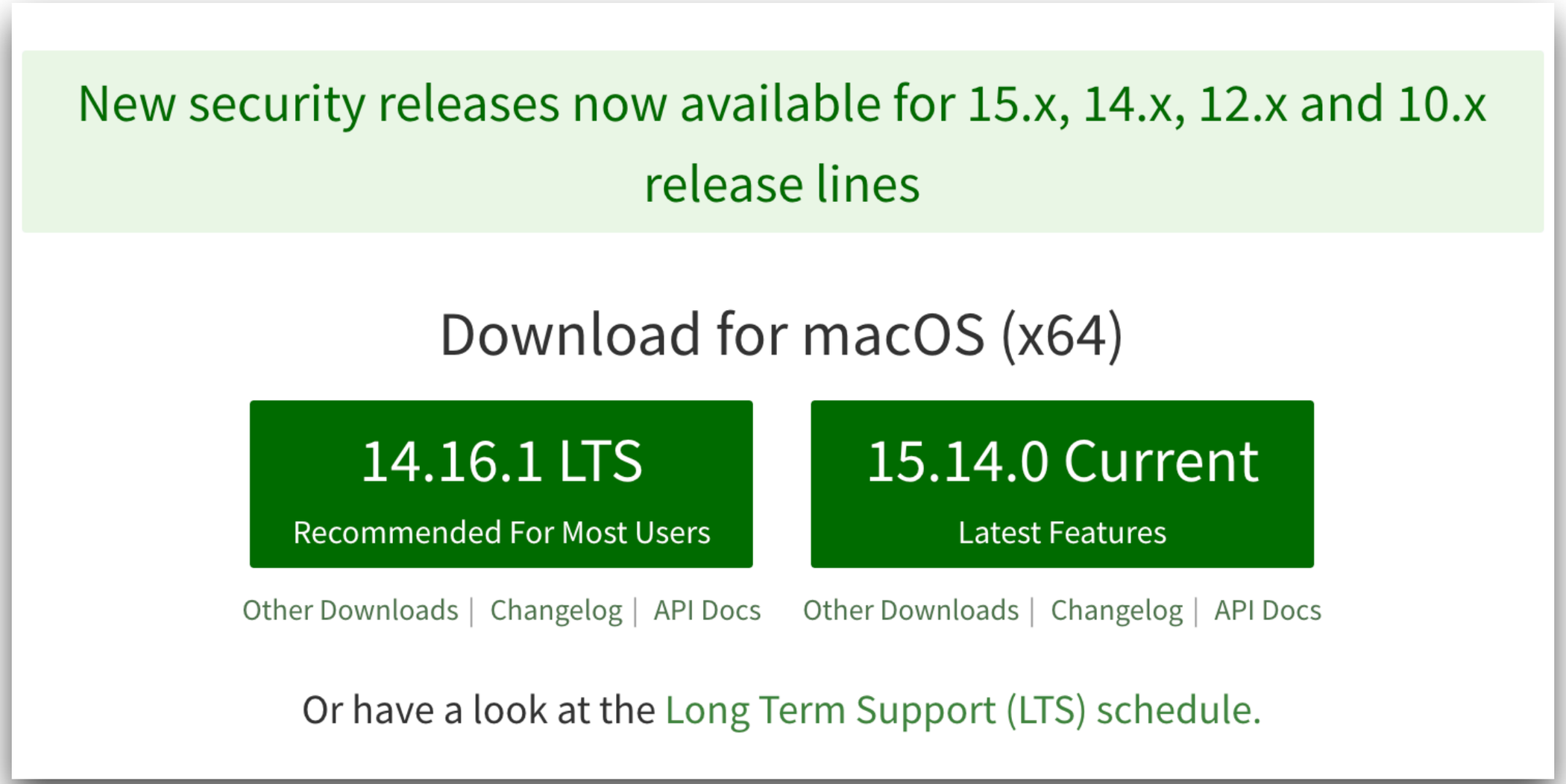
Redux, Mobx

Tools

Babel, Webpack, CSS Modules, Testing

Installation

Installation Node.js



https://nodejs.org/en/

Installation

Yarn or npm
Create-react-app

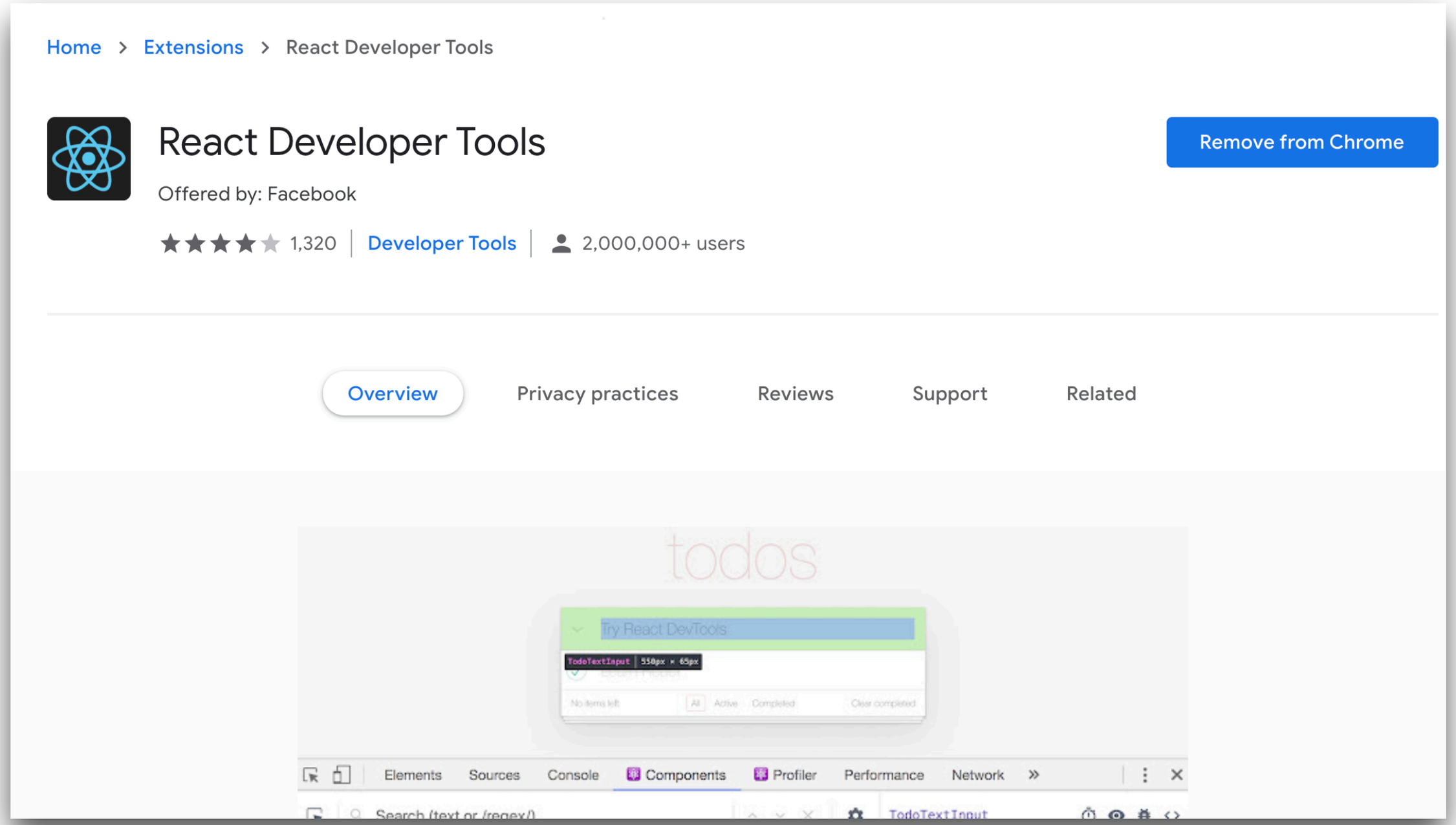
VS

Install Yarn

$brew install yarn
$npm install -g yarn
$yarn --version

https://yarnpkg.com/en/docs/install

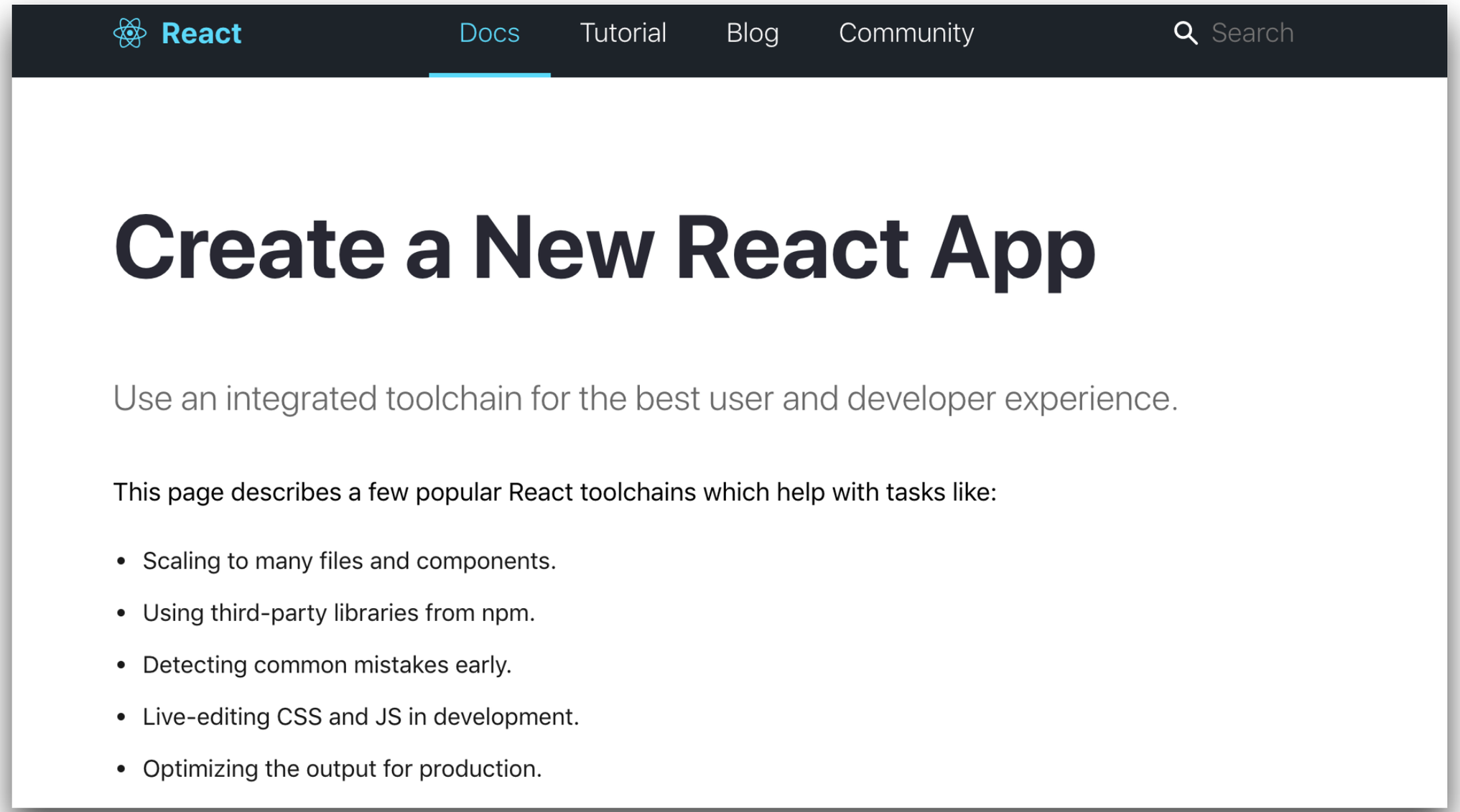
React Developer Tools



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

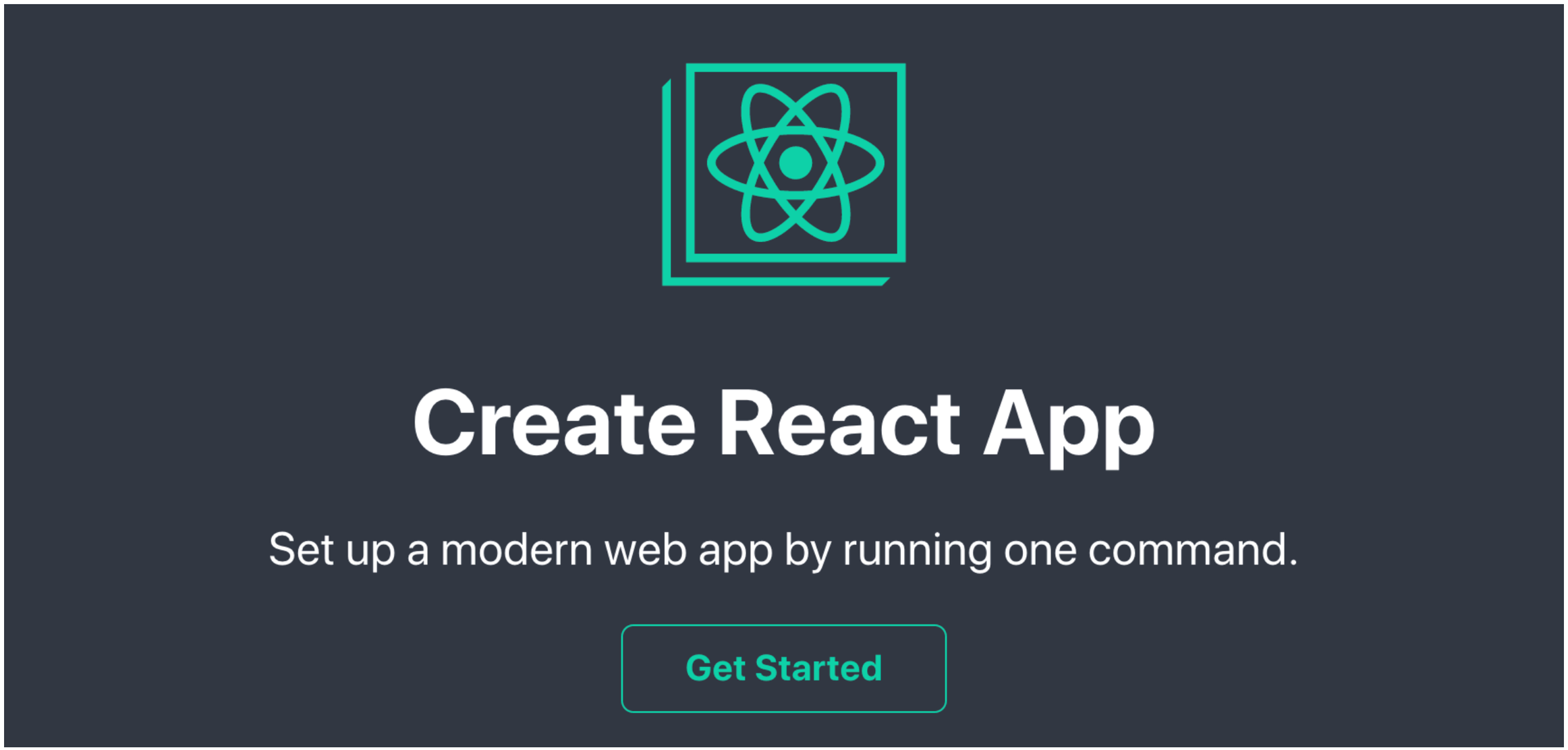
Create React Application

Create a react application



https://reactjs.org/docs/create-a-new-react-app.html

Create React App



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

Create a react application

$**npx** create-react-app hello

```
Inside that directory, you can run several commands:

  yarn start
    Starts the development server.

  yarn build
    Bundles the app into static files for production.

  yarn test
    Starts the test runner.

  yarn eject
    Removes this tool and copies build dependencies, configuration files
    and scripts into the app directory. If you do this, you can’t go back!

We suggest that you begin by typing:

  cd hello
  yarn start
```

Issue !!!

```
Aborting installation.
Unexpected error. Please report it as a bug:
{ Error: Cannot find module '/Users/somkiat/dat
-acdemy/my-app2/node_modules/react-scripts/pack
    at Function.Module._resolveFilename (interr
1:15)
    at Function.Module._load (internal/modules/
    at Module.require (internal/modules/cjs/loa
    at require (internal/modules/cjs/helpers.js
    at checkNodeVersion (/Users/somkiat/node_mo
teReactApp.js:514:23)
```

https://github.com/facebook/create-react-app/issues/4321

With NPM

$**npm** init react-app hello --use-npm

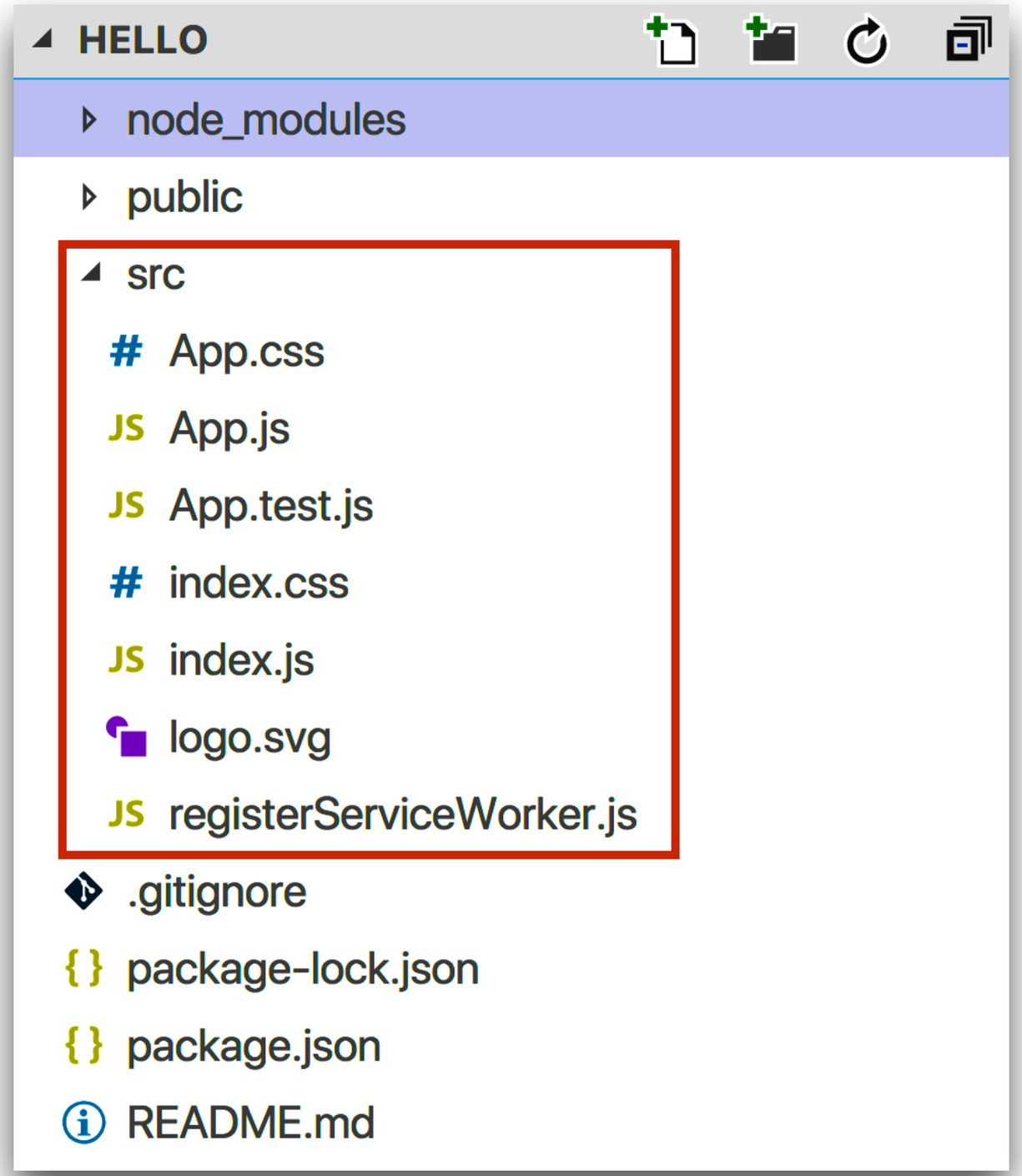
```
npm start
  Starts the development server.

npm run build
  Bundles the app into static files for production.

npm test
  Starts the test runner.

npm run eject
  Removes this tool and copies build dependencies, configuration files
  and scripts into the app directory. If you do this, you can’t go back!

We suggest that you begin by typing:

  cd my-app
  npm start
```

With Yarn

$npm install -g yarn
$**yarn** create react-app hello

```
yarn start
  Starts the development server.

yarn build
  Bundles the app into static files for production.

yarn test
  Starts the test runner.

yarn eject
  Removes this tool and copies build dependencies, configuration files
  and scripts into the app directory. If you do this, you can’t go back!

We suggest that you begin by typing:

  cd hello
  yarn start
```

Structure of application



package.json

```
{
  "name": "hello",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "react": "^16.4.1",
    "react-dom": "^16.4.1",
    "react-scripts": "1.1.4"
  },
  "scripts": {
    "start": "react-scripts start",
    "build": "react-scripts build",
    "test": "react-scripts test --env=jsdom",
    "eject": "react-scripts eject"
  }
}
```

Start your app

$yarn start
$npm start

```
Compiled successfully!

You can now view hello in the browser.

  Local:            http://localhost:3000/
  On Your Network:  http://192.168.20.69:3000/

Note that the development build is not optimized.
To create a production build, use yarn build.
```

Play your app

http://localhost:3000/

View source !!

```
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <!--
      manifest.json provides metadata used when your web app is added to the
      homescreen on Android. See https://developers.google.com/web/fundamentals/engage-and-retain/web-app-manifest/
    -->
    <link rel="manifest" href="/manifest.json">
    <link rel="shortcut icon" href="/favicon.ico">
    <!--
      Notice the use of  in the tags above.
      It will be replaced with the URL of the `public` folder during the build.
      Only files inside the `public` folder can be referenced from the HTML.

      Unlike "/favicon.ico" or "favicon.ico", "/favicon.ico" will
      work correctly both with client-side routing and a non-root public URL.
      Learn how to configure a non-root public URL by running `npm run build`.
    -->
    <title>React App</title>
  </head>
  <body>
    <noscript>
      You need to enable JavaScript to run this app.
    </noscript>
    <div id="root"></div>
    <!--
      This HTML file is a template.
      If you open it directly in the browser, you will see an empty page.
```

Workflow

/public/
index.html

index.js

<App/>

Component

App.js

jsx

css

index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import registerServiceWorker from './registerServiceWorker';

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

*Rendered App component in *

App.js (function)

JSX (JavaScript XML)

```jsx
function App() {
  return (
    <div className="App">
      <header className="App-header">
        <img src={logo} className="App-logo" alt="logo" />
        <p>
          Edit <code>src/App.js</code> and save to reload.
        </p>
      </header>
    </div>
  );
}

export default App;
```

App.js (class)

```
class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <h1 className="App-title">Welcome to React</h1>
        </header>
        <p className="App-intro">
          To get started, edit <code>src/App.js</code> and sa
        </p>
      </div>
    );
  }
}
```

Programming style

Class

Function

React Fundamentals

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

JSX

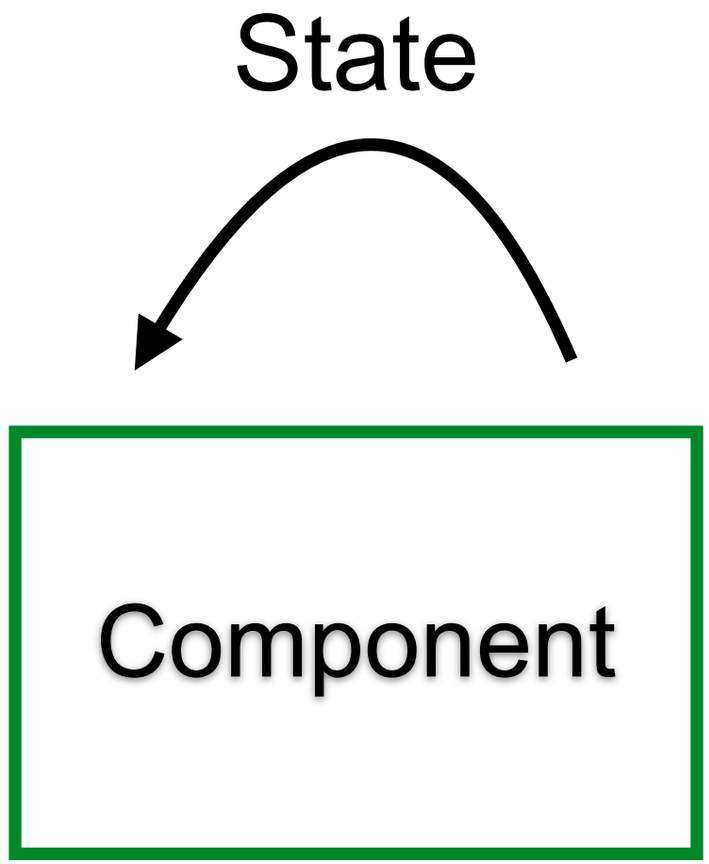
HTML + JavaScript
Make HTML easy to understand
Boost up performance of JavaScript

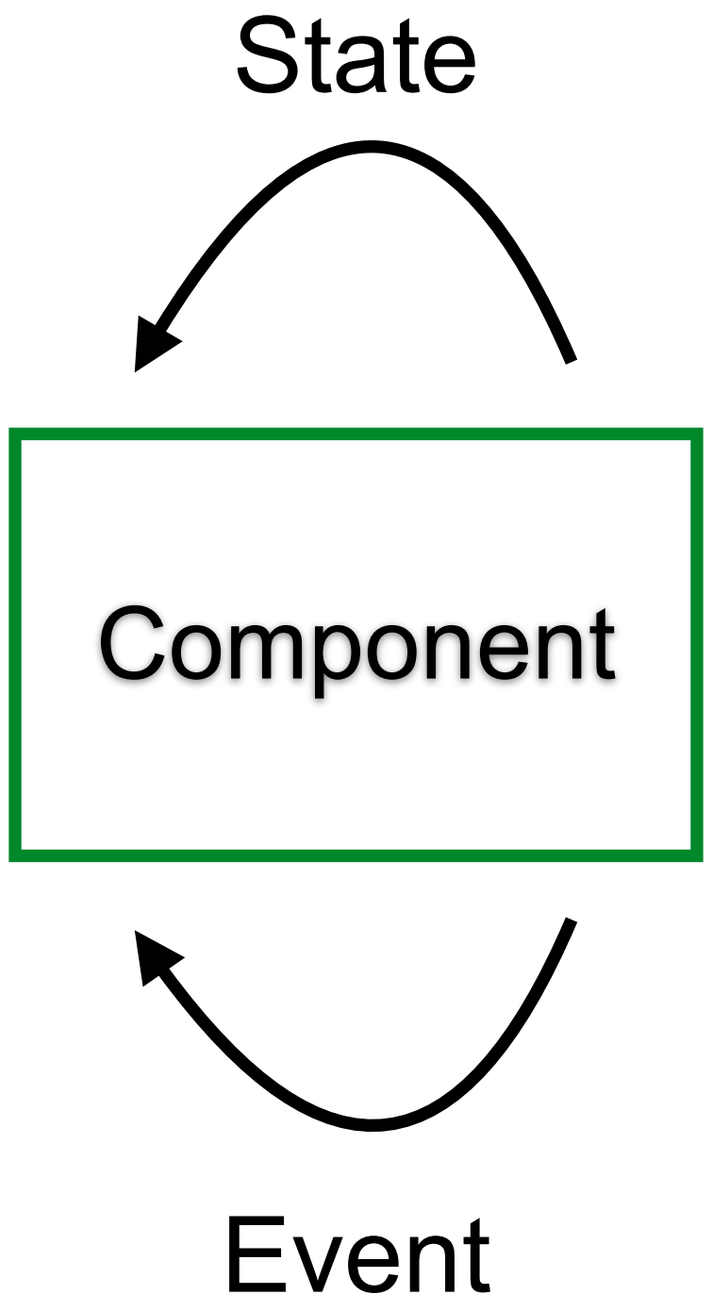
https://facebook.github.io/jsx/

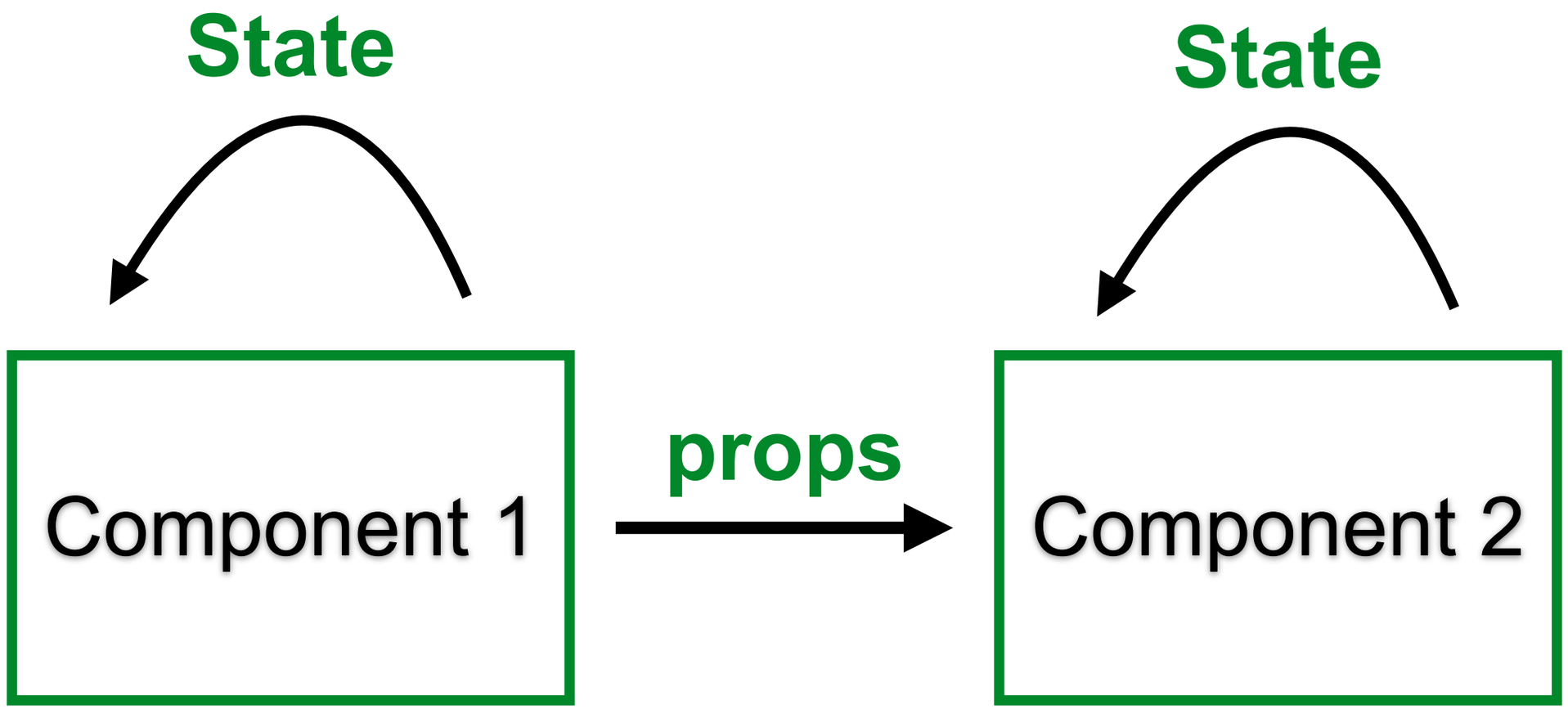
Components

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

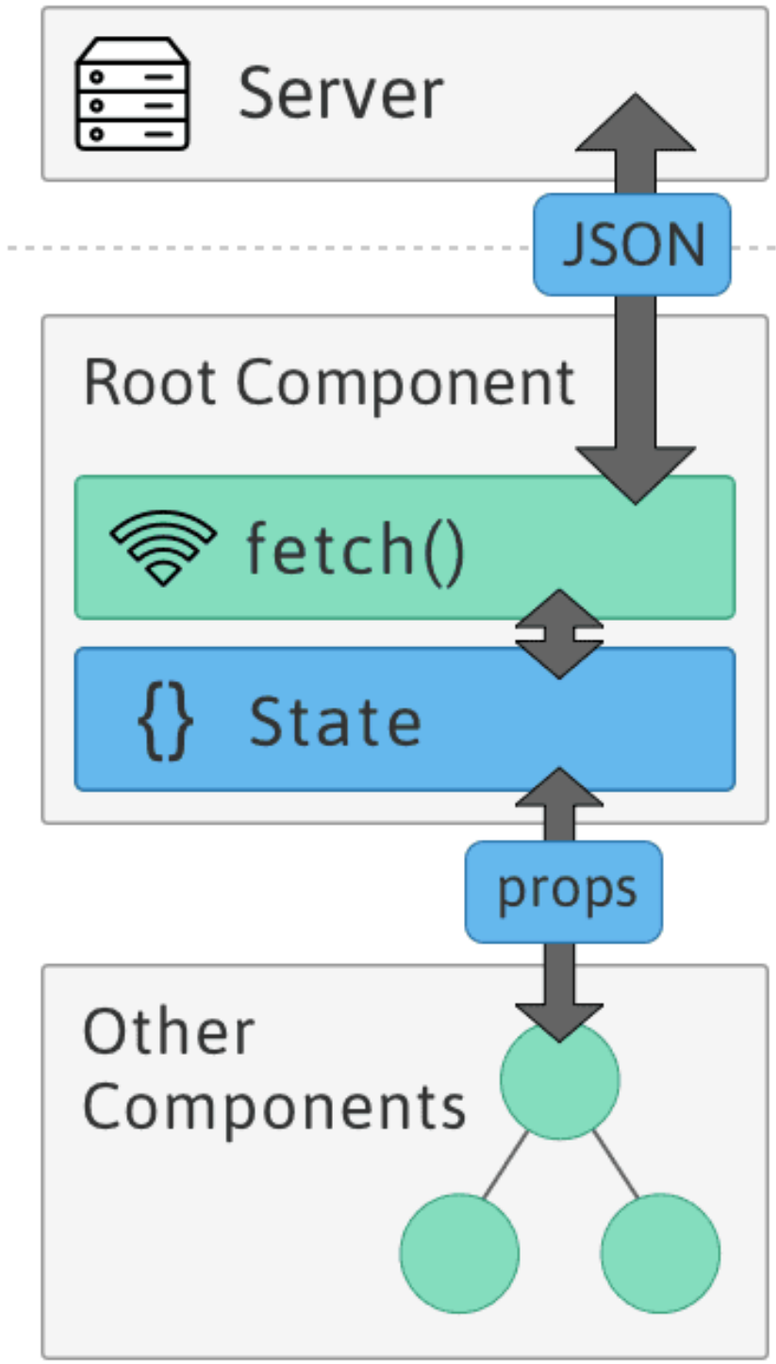
https://reactjs.org/docs/components-and-props.html



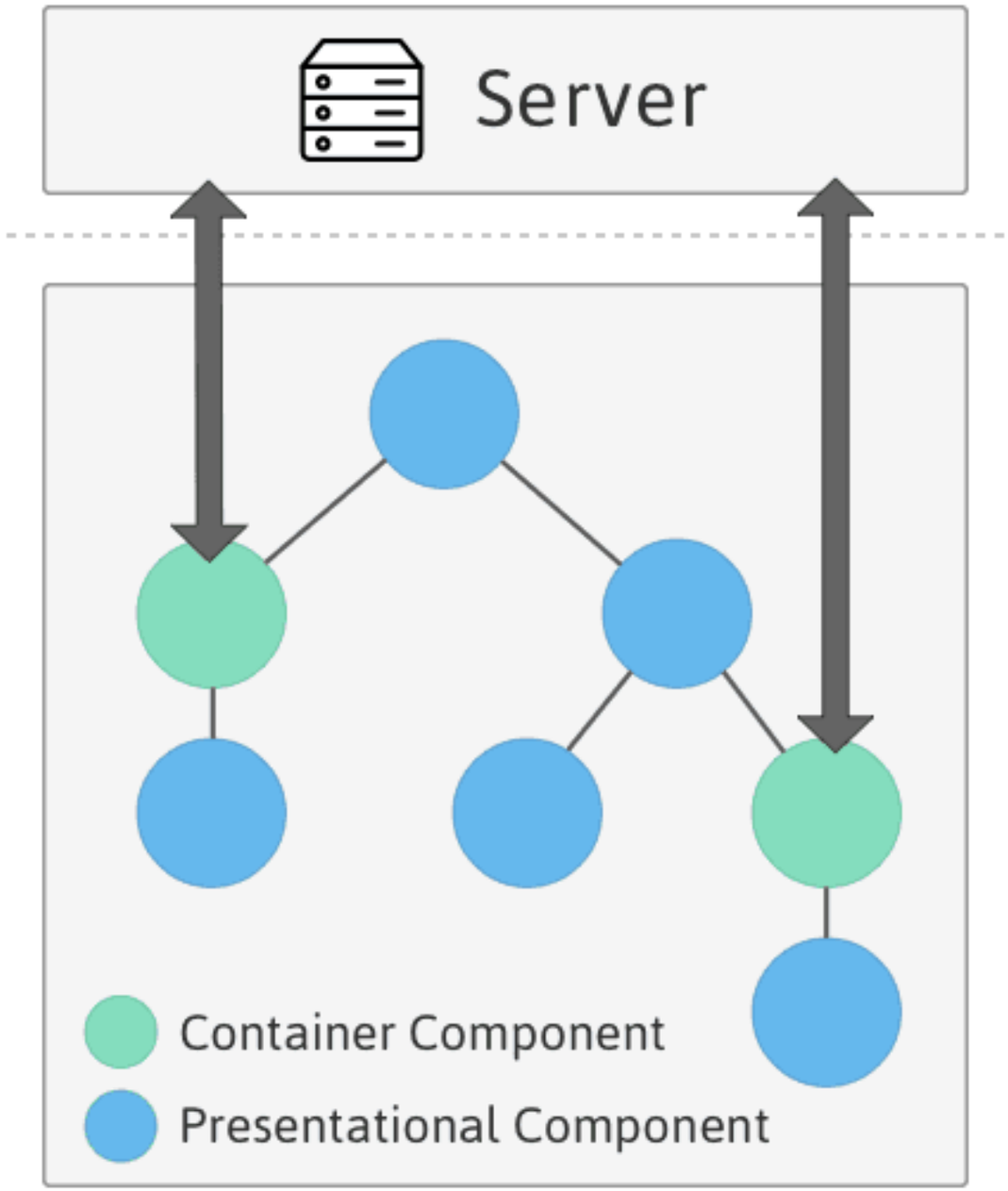




State and Props



Container component



Smart and Dumb 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

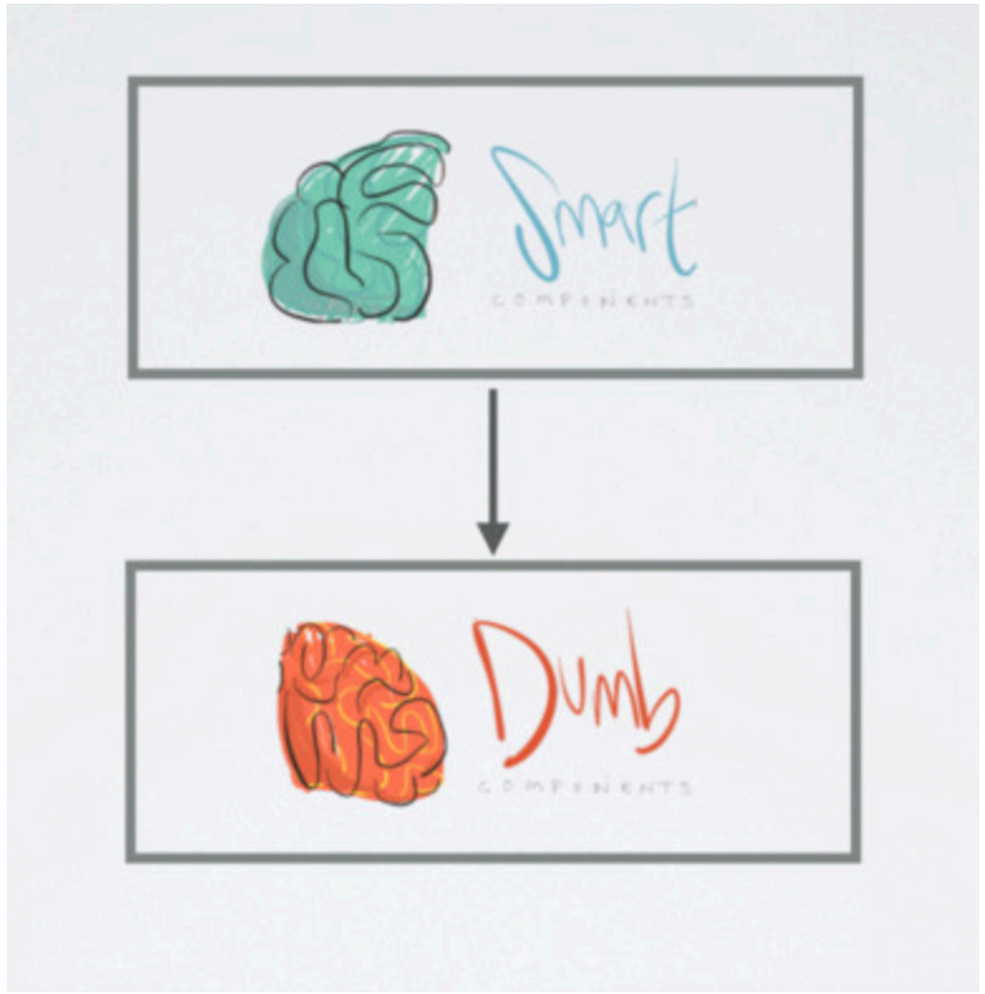
How things look

No app dependencies

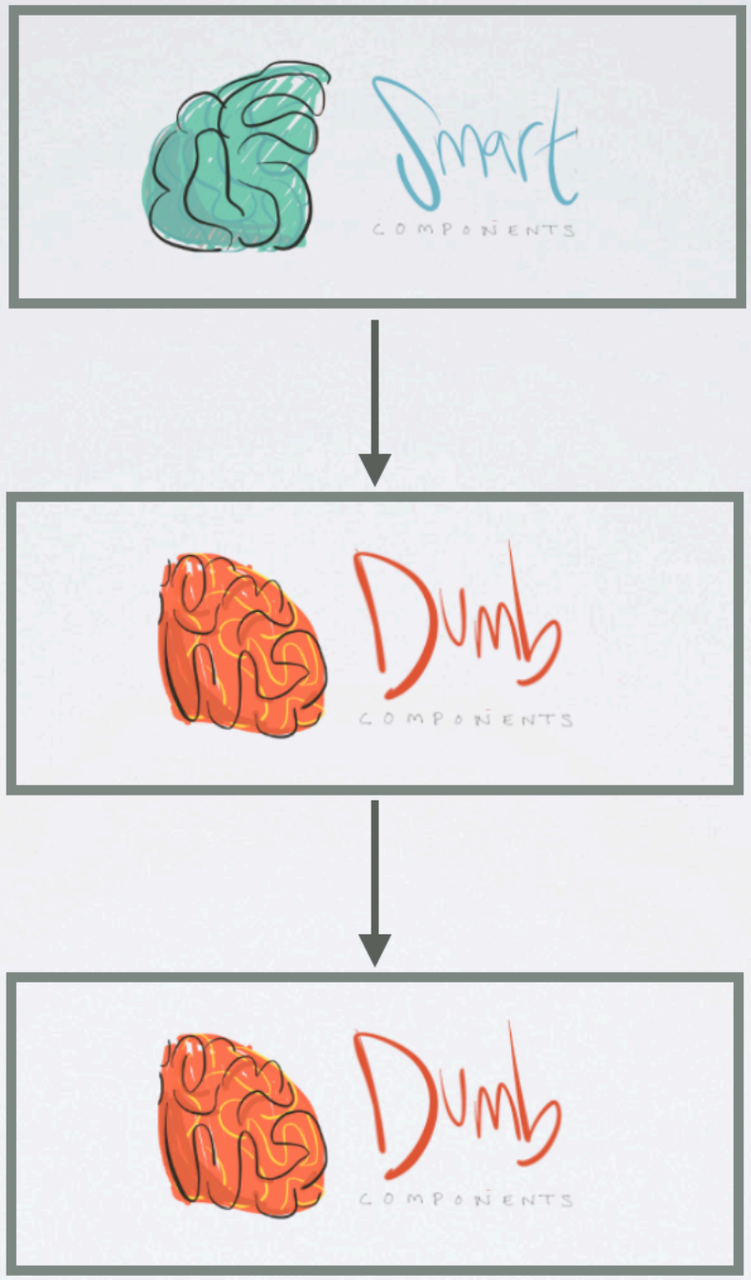
Just props, for data and callbacks

Rerely have own state, only UI state

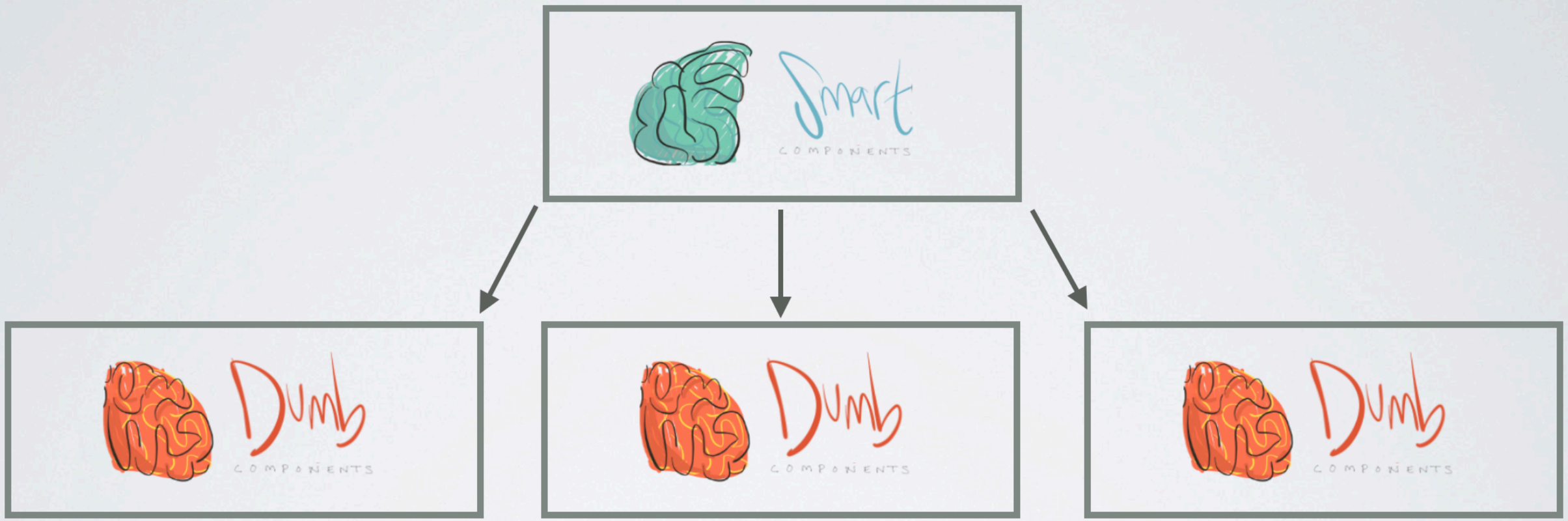
Smart and Dumb component



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/>
    )
  }
}
```

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

Send name to Welcome component

Edit file **App.js** in src

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

Edit Welcome component

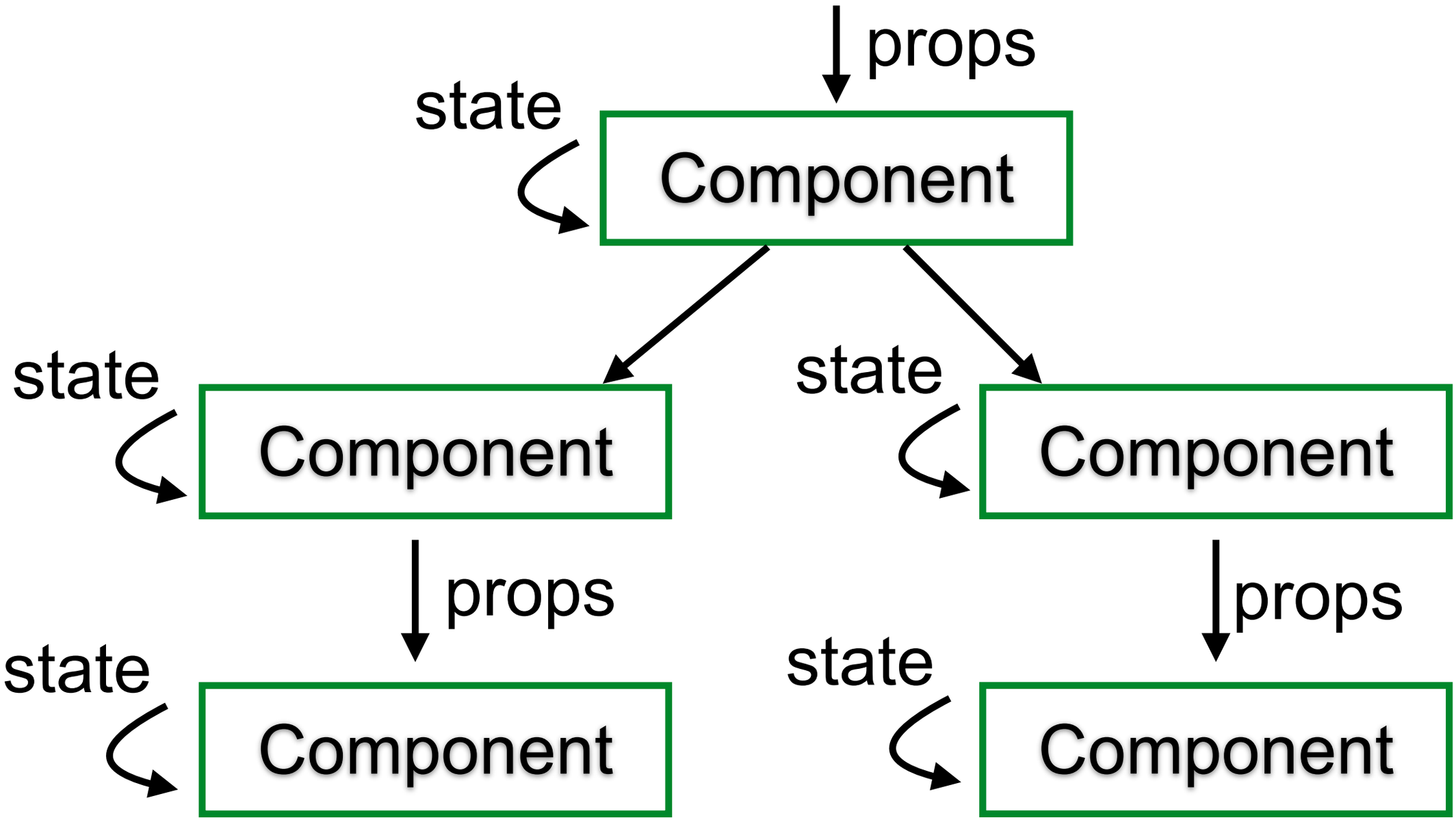
```
class Welcome extends Component {
    render() {
        return(
            <h1>Welcome {this.props.name}</h1>
        )
    }
}
```

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 and States



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 ?

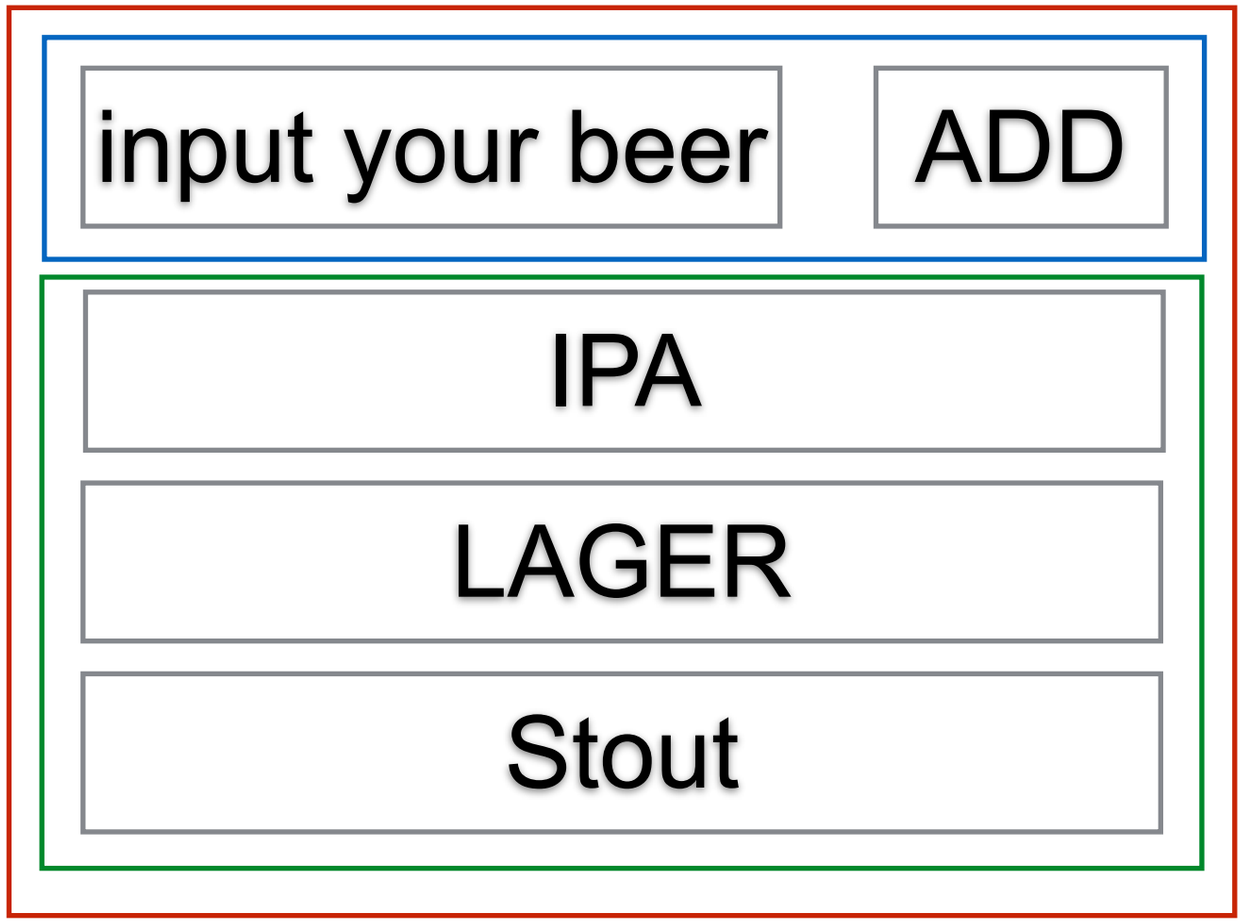
Let’s design your component

Component for react ?

1 component ?

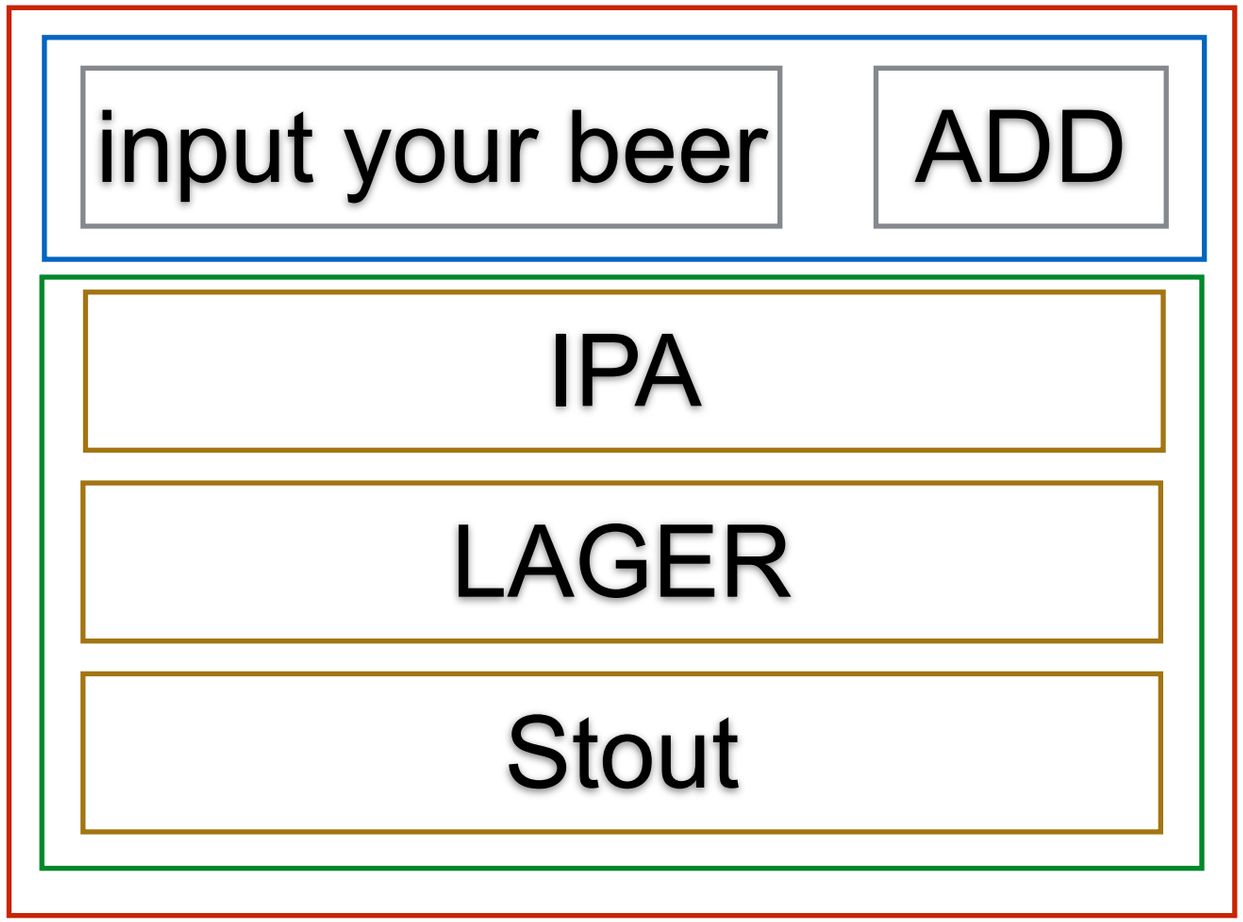
Component for react ?

3 component ?



Component for react ?

4 component ?



Components

BeerListContainer

AddBeerComponent

Input text

Button

BeerListContainer

BeerItemComponent

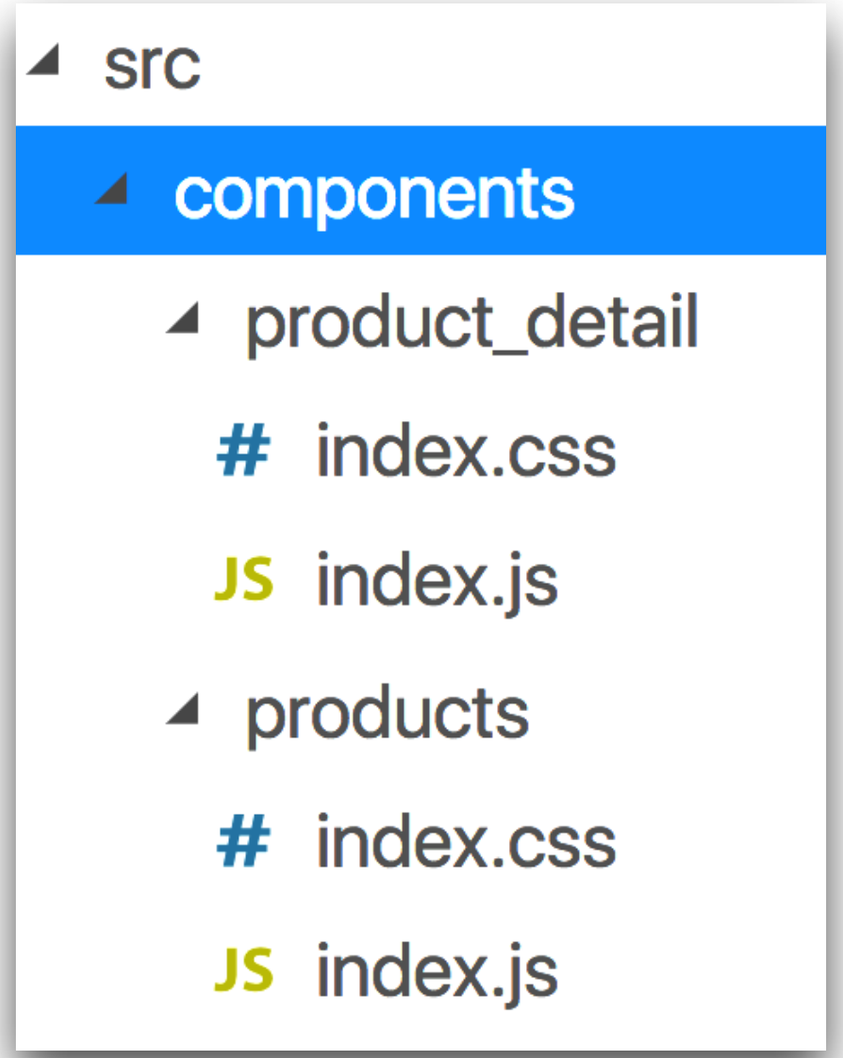
Components

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

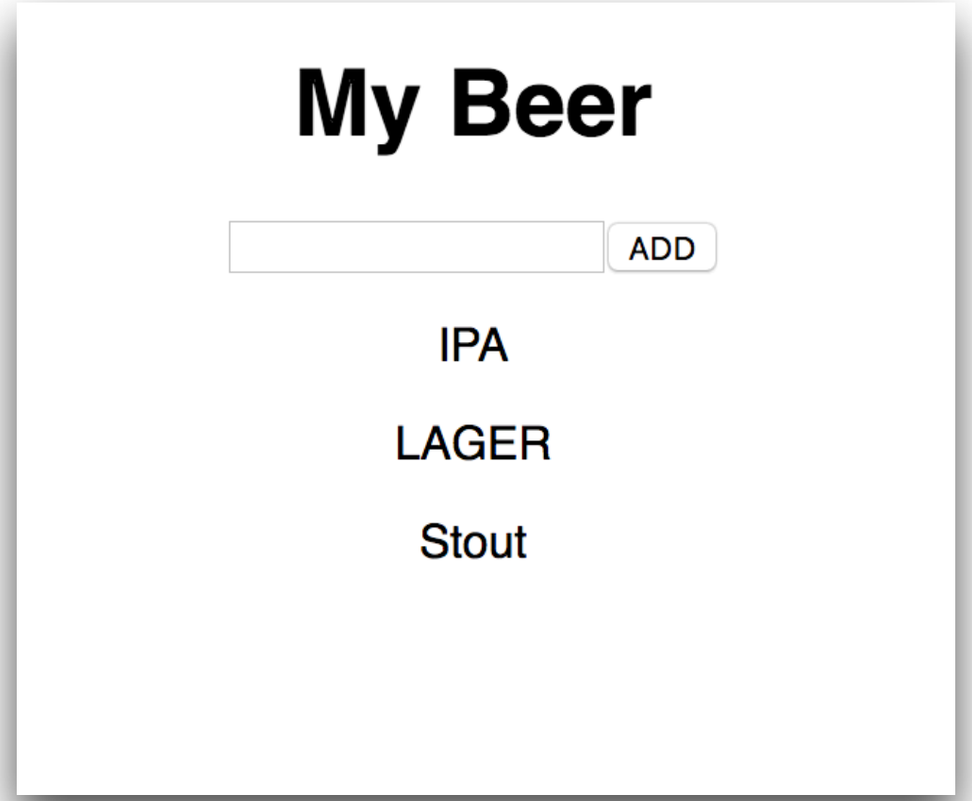
```
import React, { Component } from 'react';

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

export default BeerList;
```

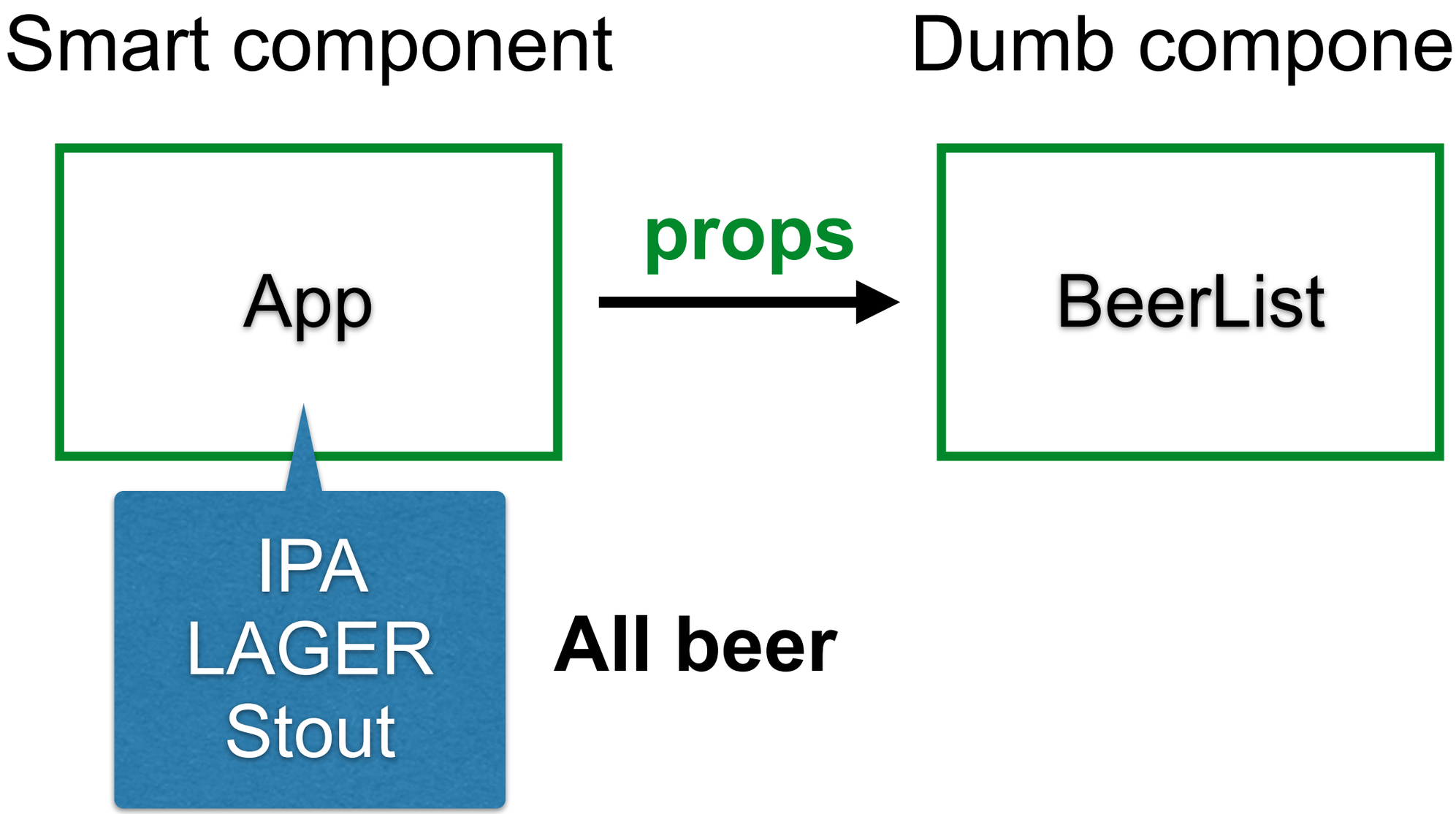
App component

```
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

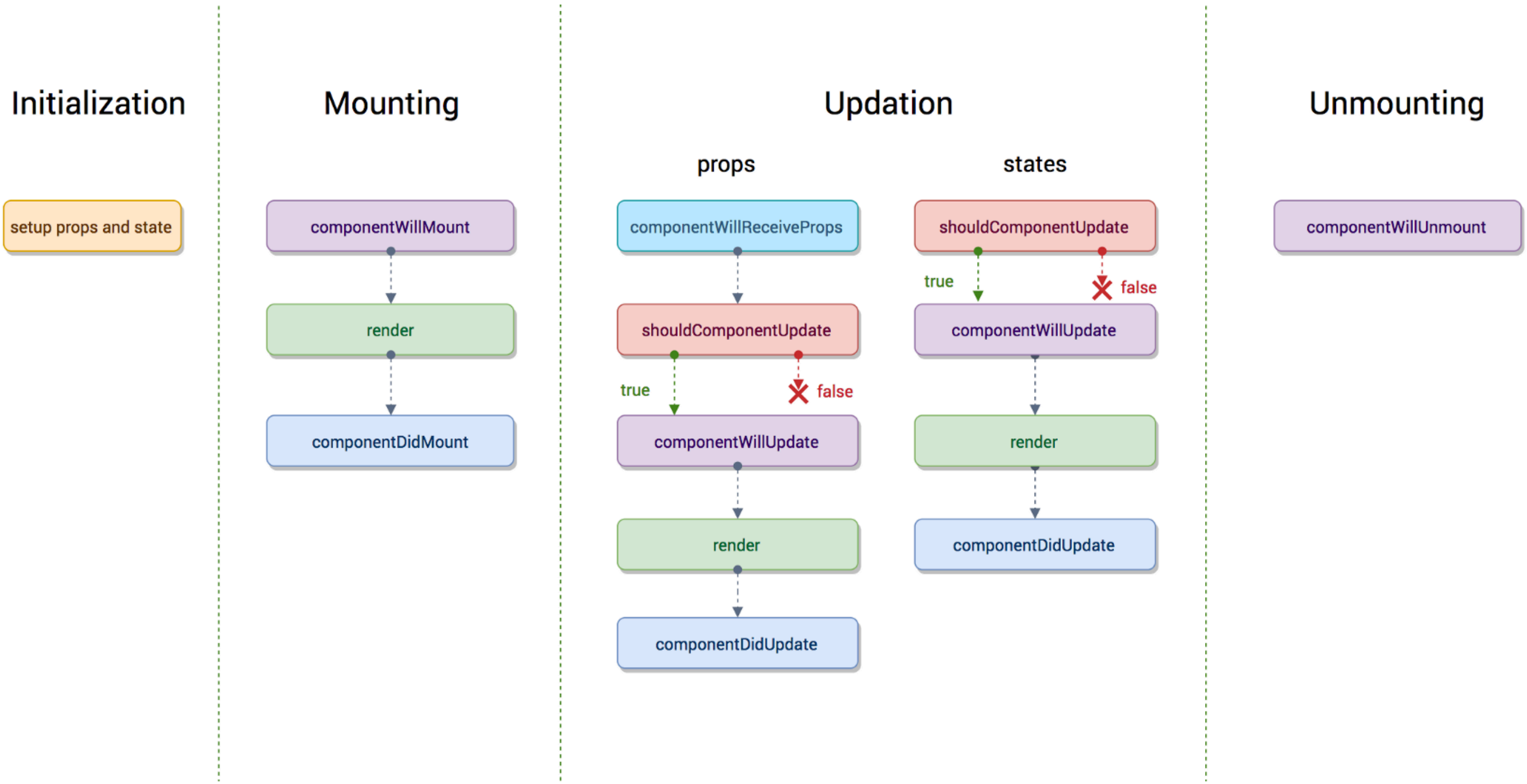
Send data via props to BeerList component

```
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”**

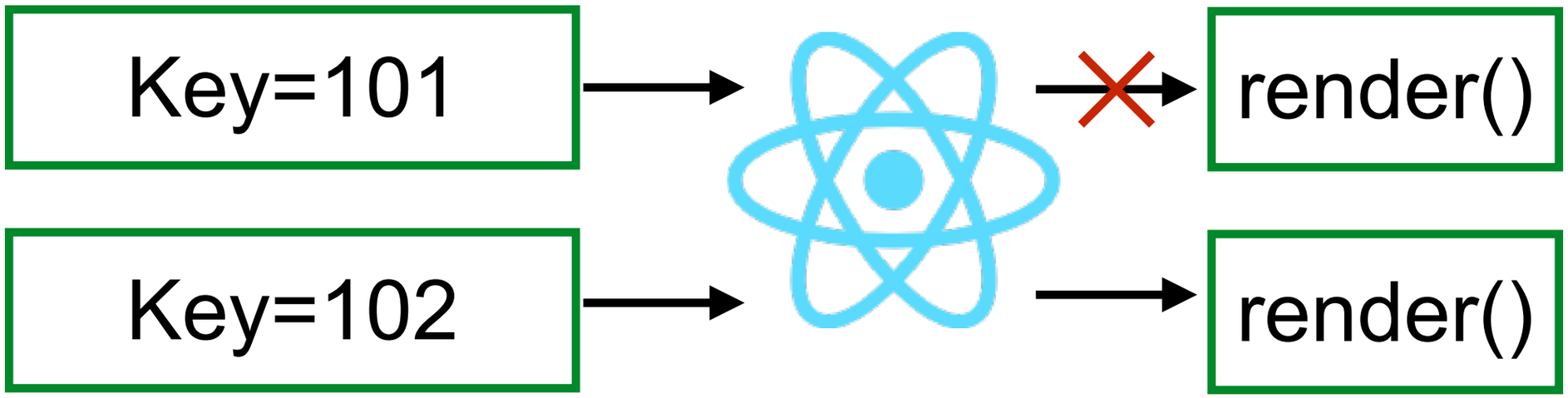
Lifecycle



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
```

Deploy React app to Github Pages

1. Install library gh-pages

$npm install --save gh-pages

2. Create new repos in Github

$git init

$git remote add origin https://github.com/<user>/demo-react.git

3. Edit file package.json (1)

Add homepage to your Github Pages

```
"name": "hello",
"version": "0.1.0",
"private": true,
"homepage": "https://up1.github.io/demo-react/",
"dependencies": {
```

3. Edit file package.json (2)

Add new script to deploy

```
"scripts": {
  "start": "react-scripts start",
  "build": "react-scripts build",
  "test": "react-scripts test --env=jsdom",
  "eject": "react-scripts eject",
  "predeploy": "npm run build",
  "deploy": "gh-pages -d build"
}
```

4. Deploy

$npm run deploy

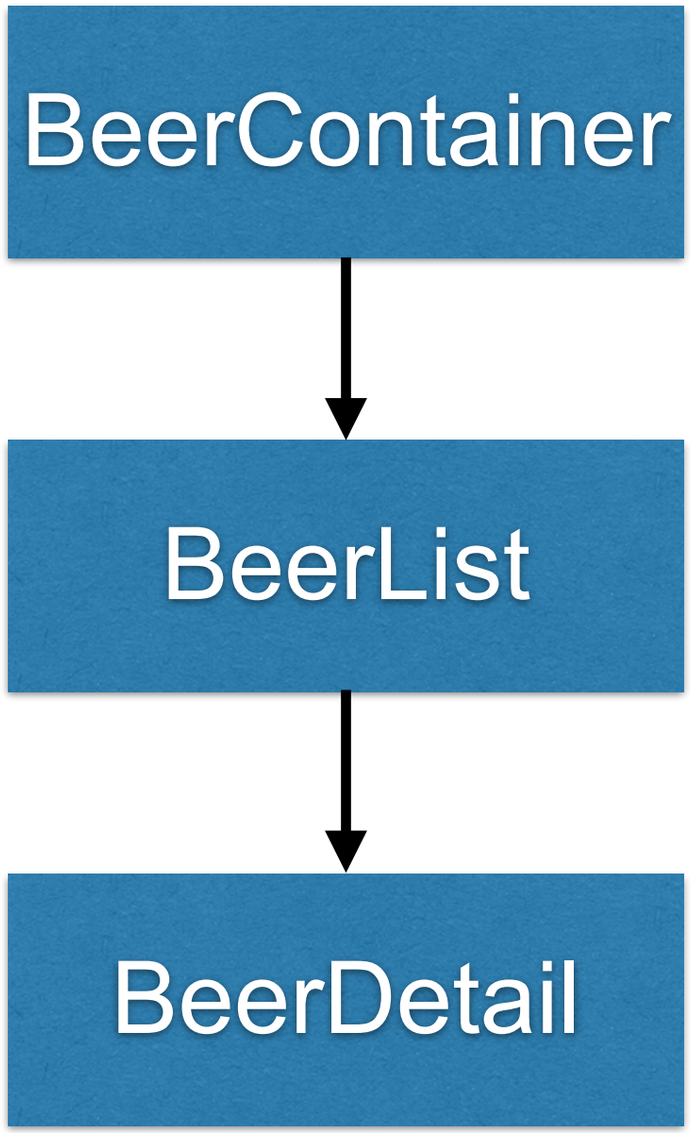
Working with API

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

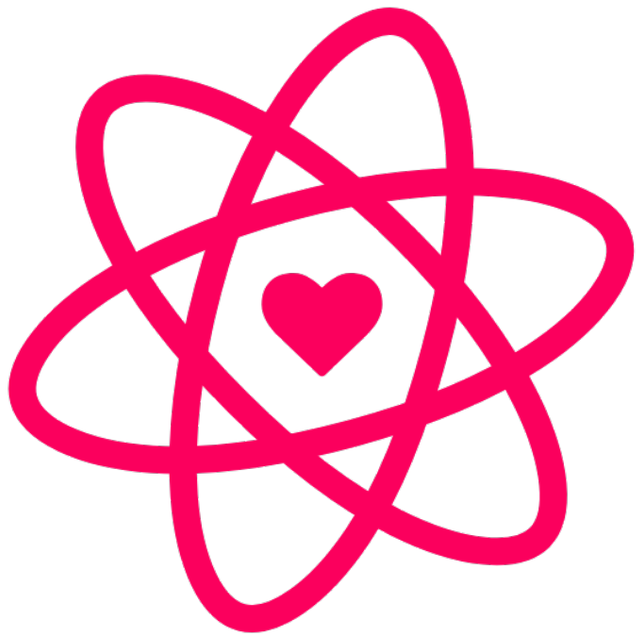
```
import React, { Component } from "react";

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

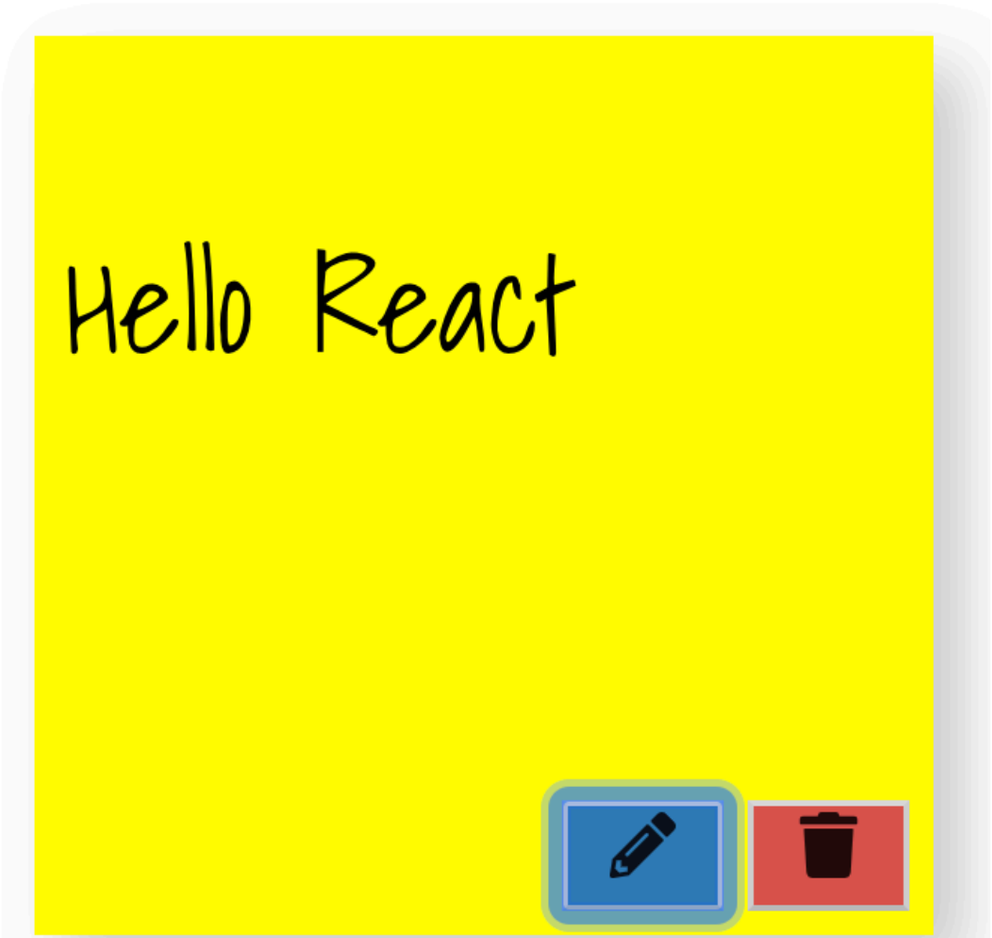
4. Change Icon in app

$npm install react-icons --save

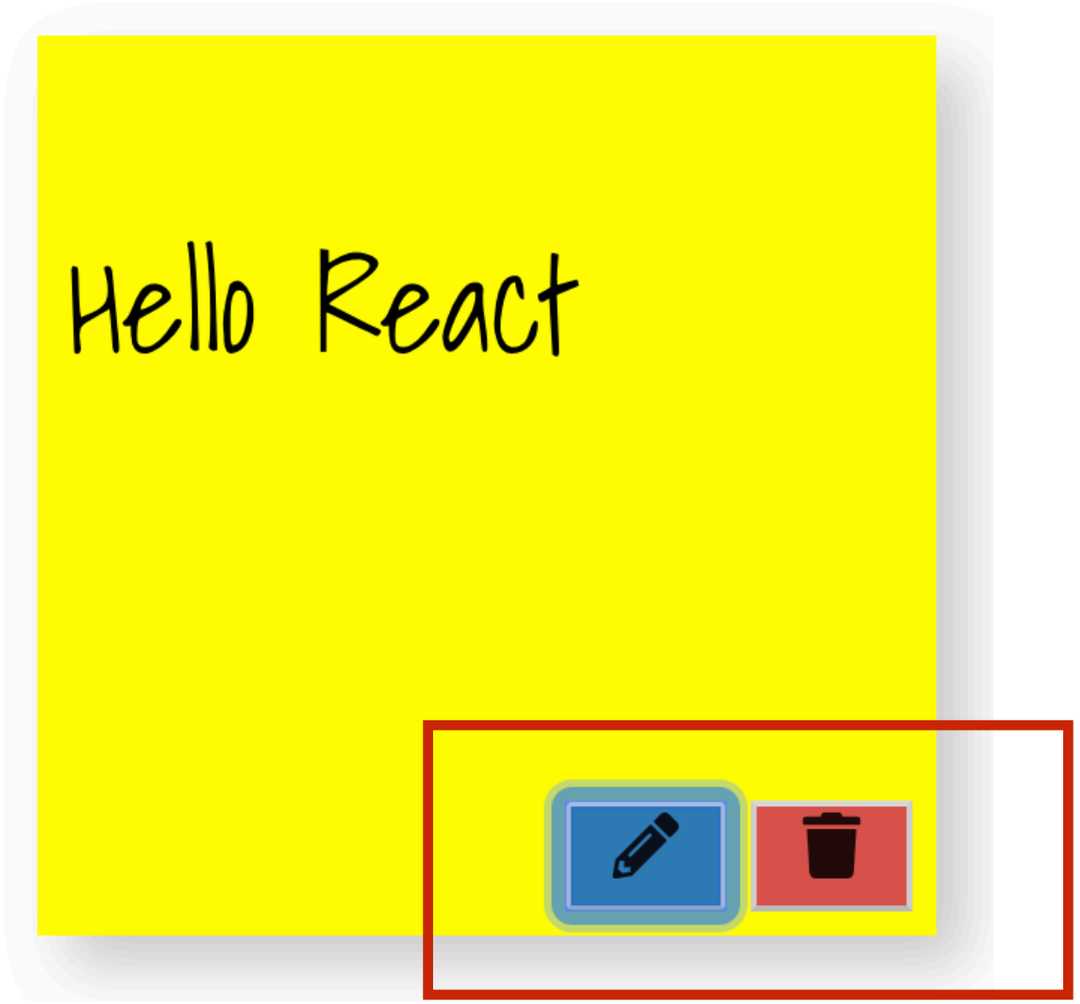
```
import React, { Component } from "react"
import {FaPencilAlt, FaTrash} from "react-icons/fa/index";

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

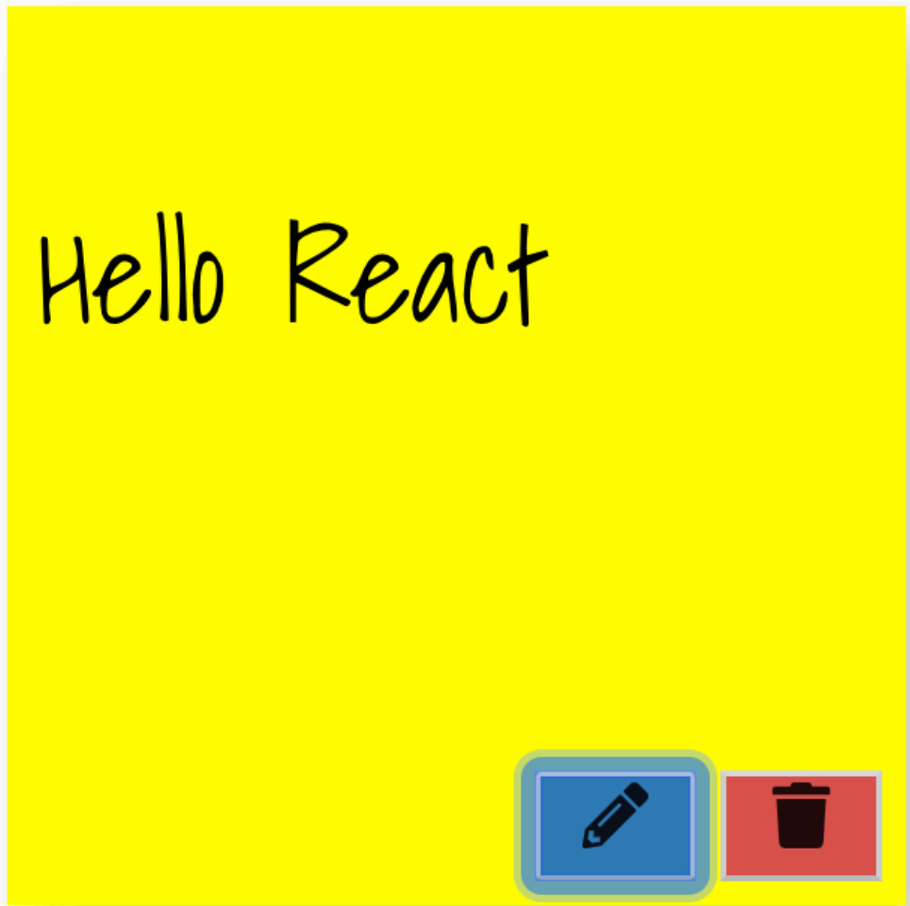
Click on edit and delete button

```
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>
    )
}
```

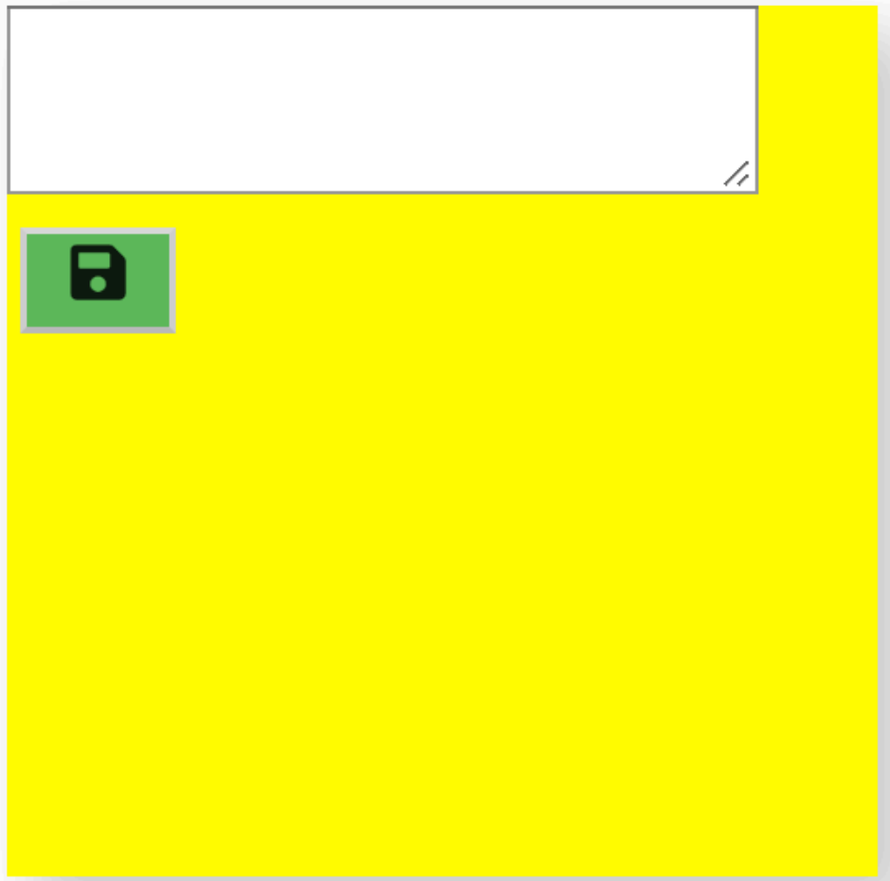
6. Add state to component

isEdit = true | false

isEdit = false



isEdit = true



6. Add state to component

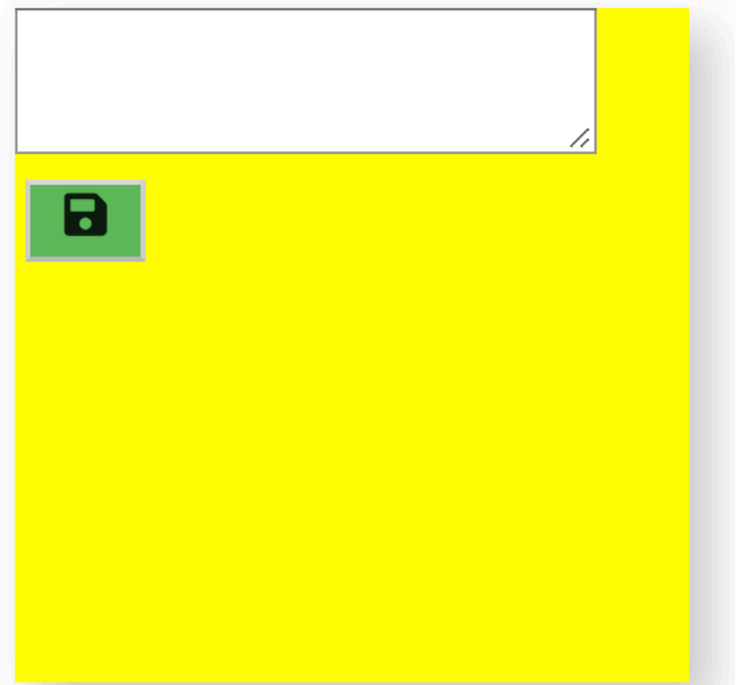
Create state in constructor()

```
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
    })
}
```

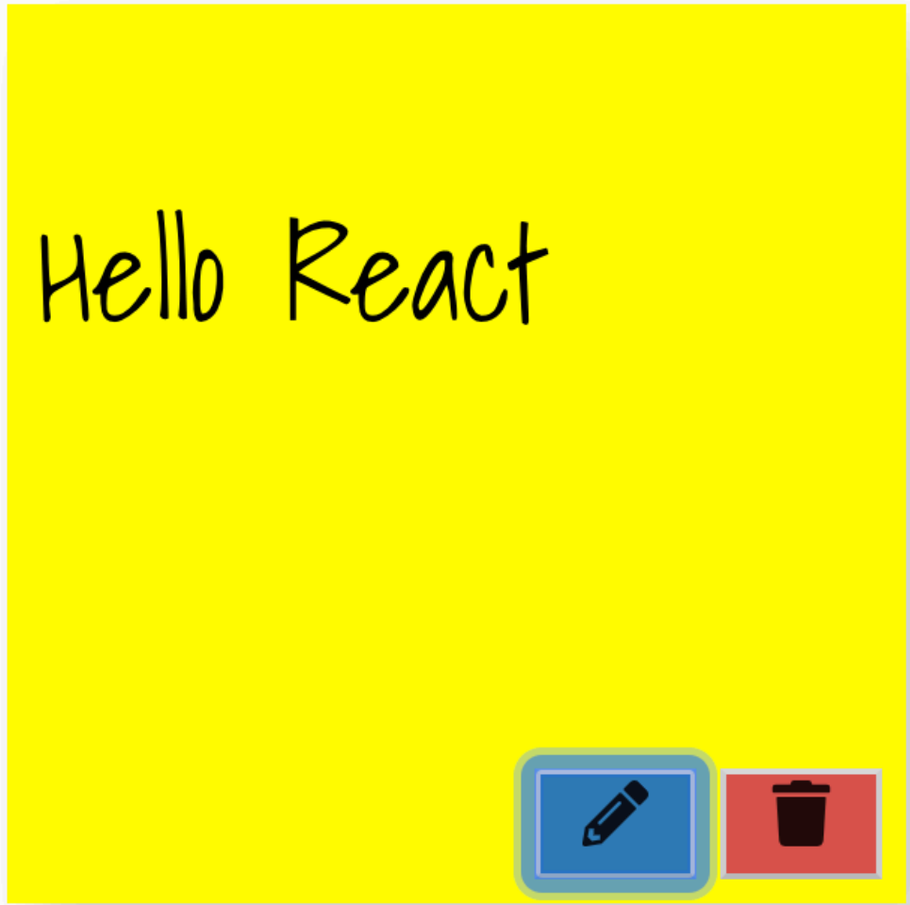


```
renderFormForEdit() {
    return(
        <div className="note">
            <form>
                <textarea />
                <button id="save"><FaSave/></button>
            </form>
        </div>
    )
}
```

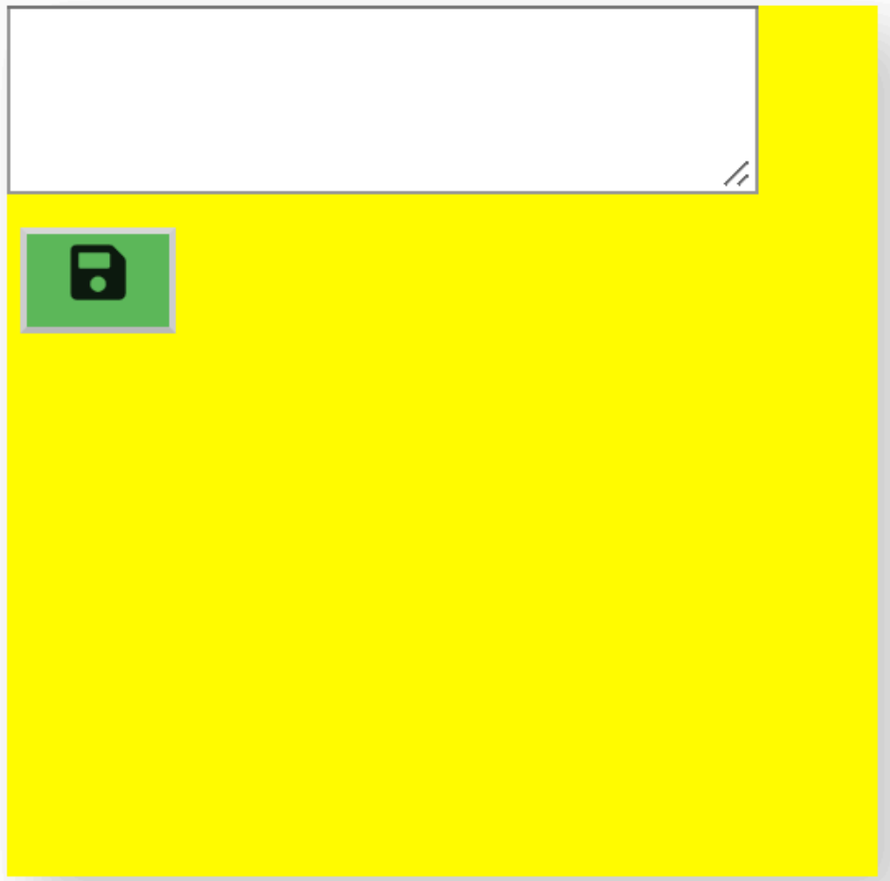
6. Add state to component

UI for display in each state

renderForShow()



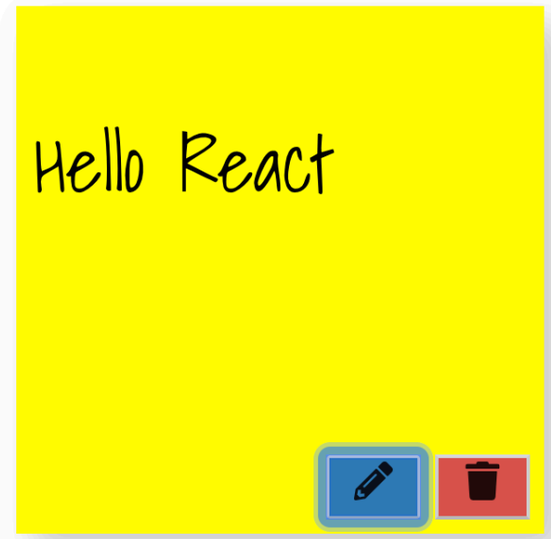
renderFormForEdit()



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

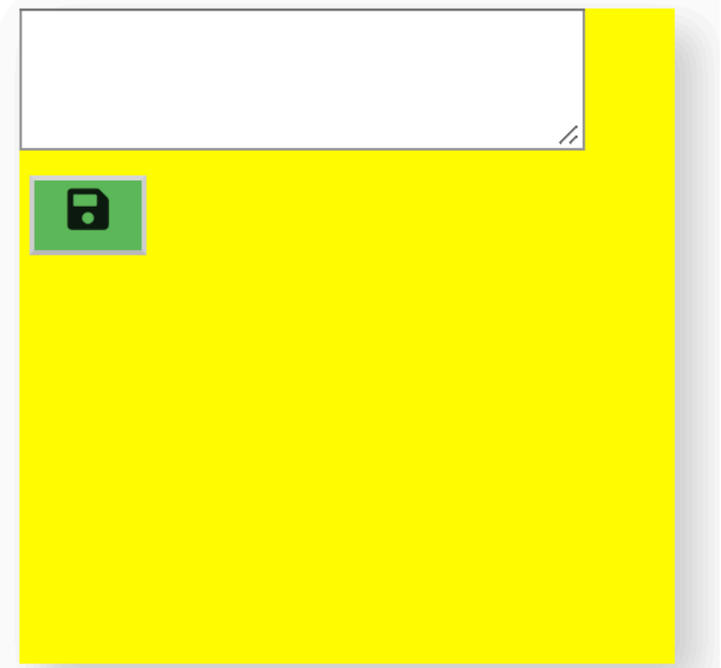
Condition to display UI in each state

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

7. Save data from textarea

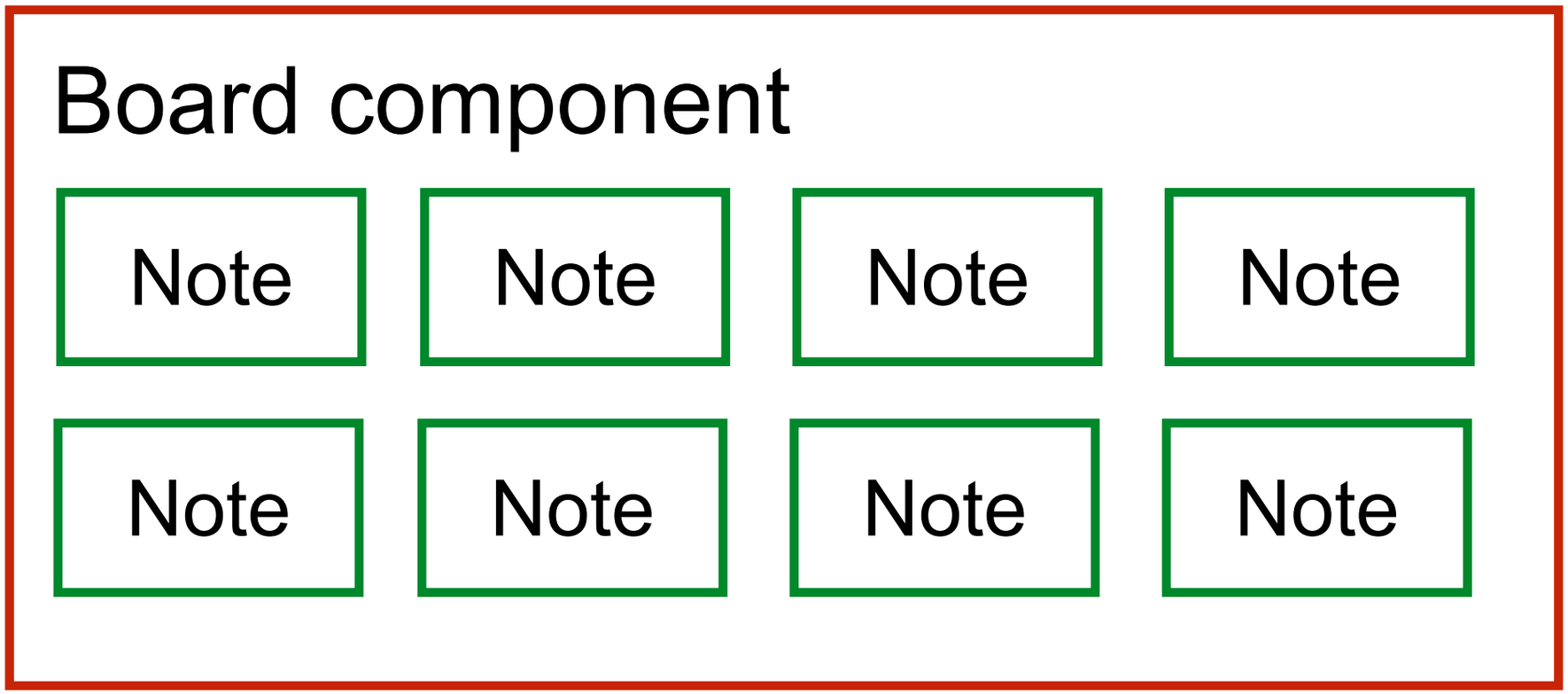
Using refs

```
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



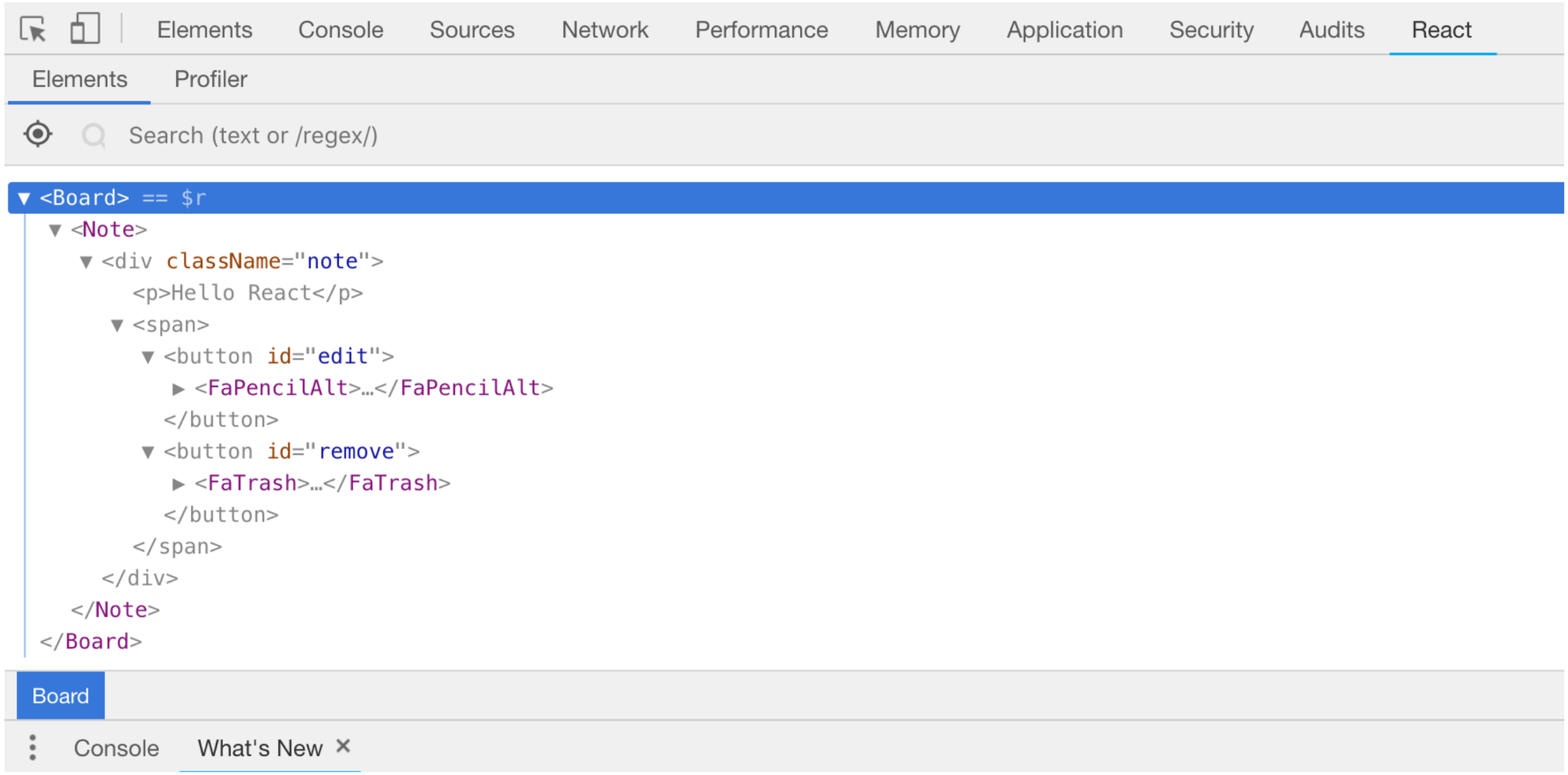
8. Parent and child component

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

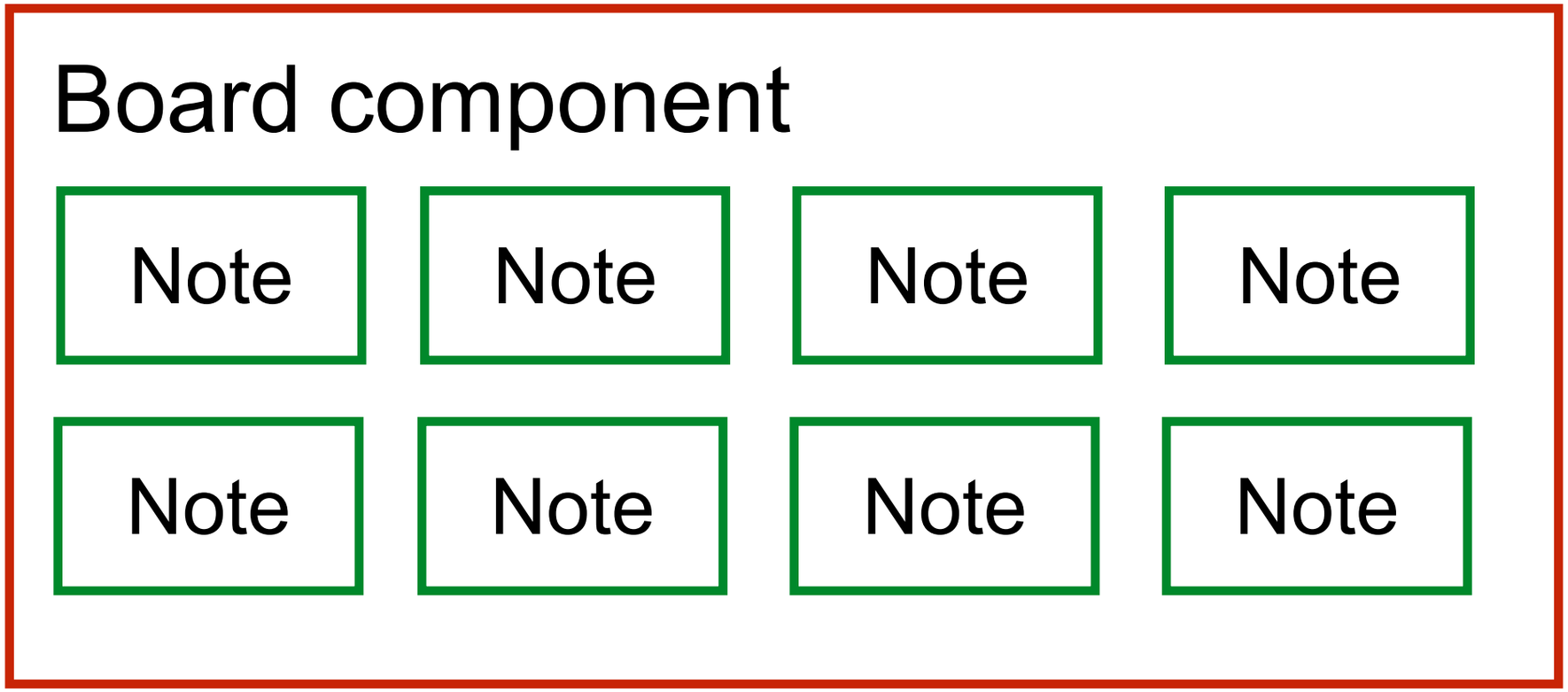
```
class Board extends Component {
    render() {
        return(
            <div>
                <Note/>
                <Note/>
                <Note/>
            </div>
        )
    }
}
```

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

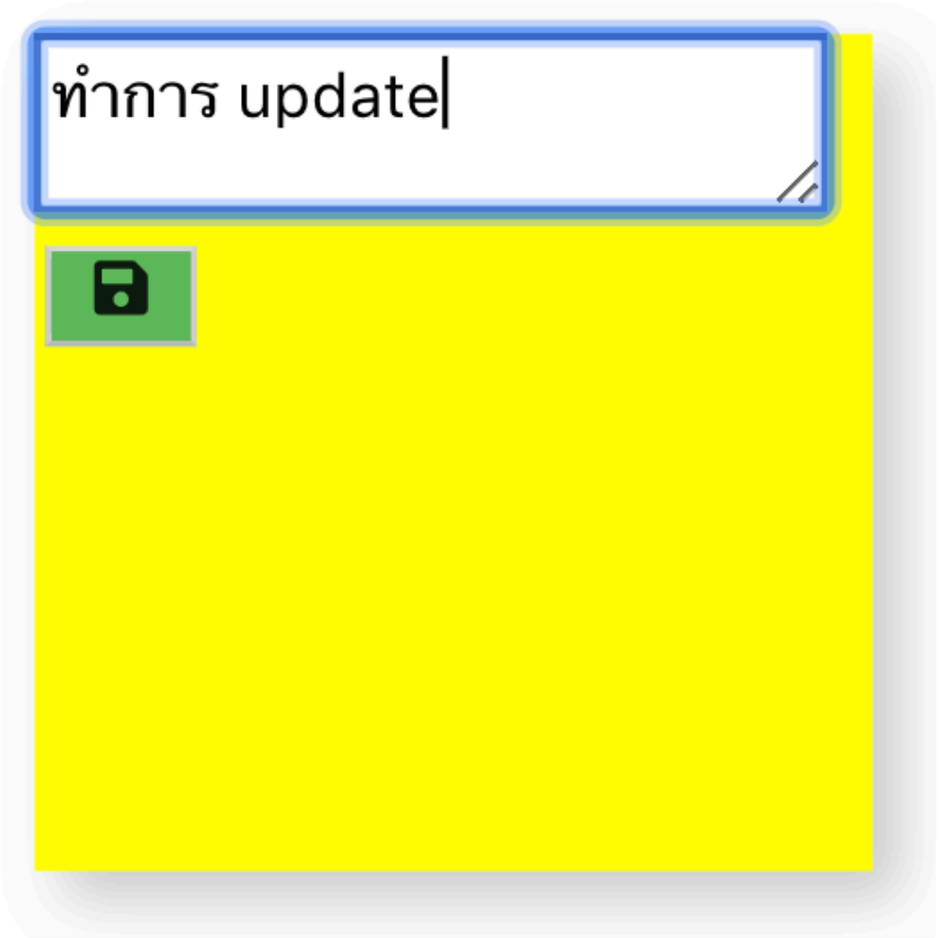
Read data from props

```
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

Logic to edit note

Note component

Handling when data changed

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 !!

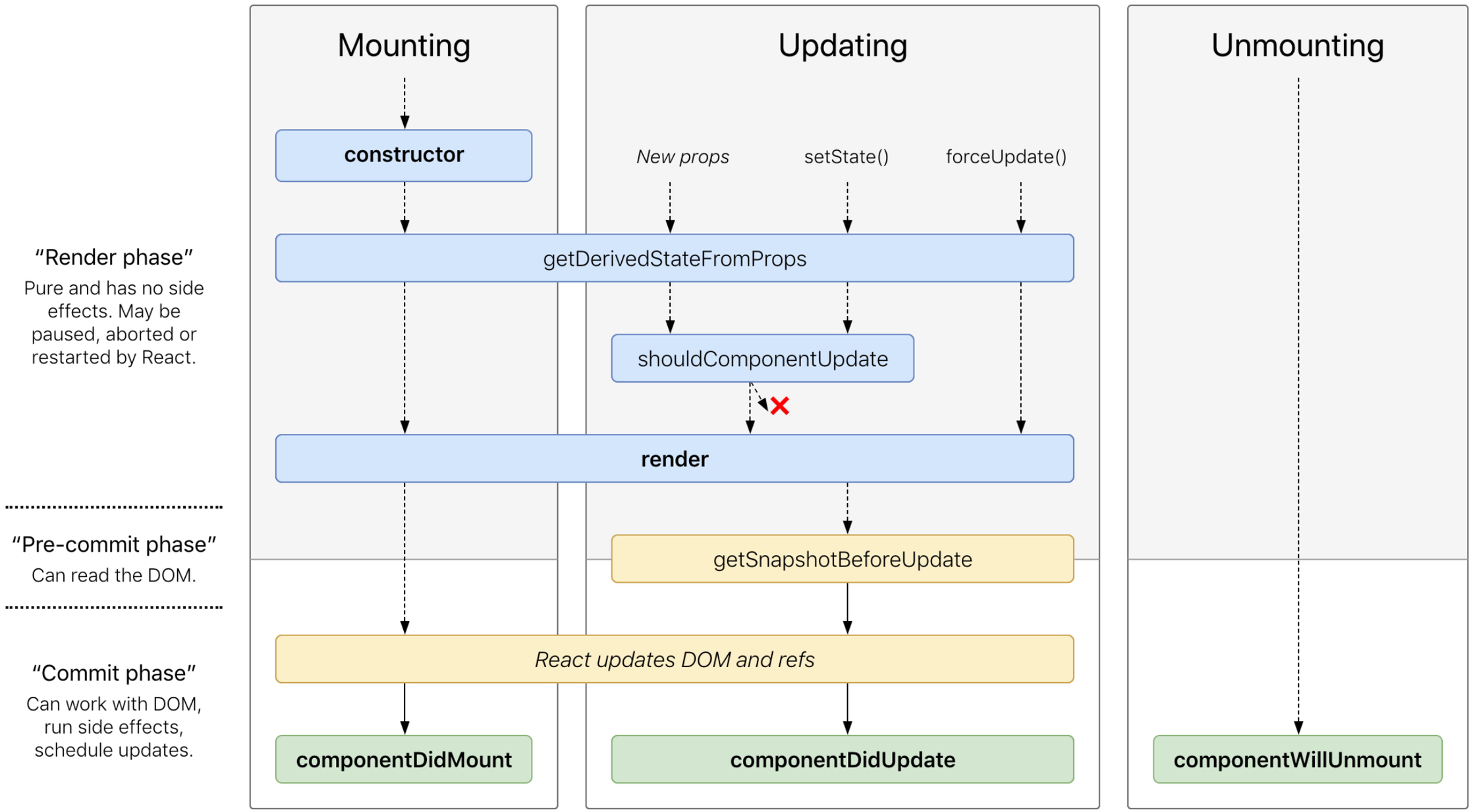
```
Render in note  3
Component did update
▶Warning: Encountered two children with the same key, `3`. Keys should b
to be duplicated and/or omitted — the behavior is unsupported and could
    in div (at Board.js:81)
    in Board (at src/index.js:7)
Render in note  1
Render in note  2
```

Component Life Cycle

Component Life Cycle

Mounting
Update
Unmount

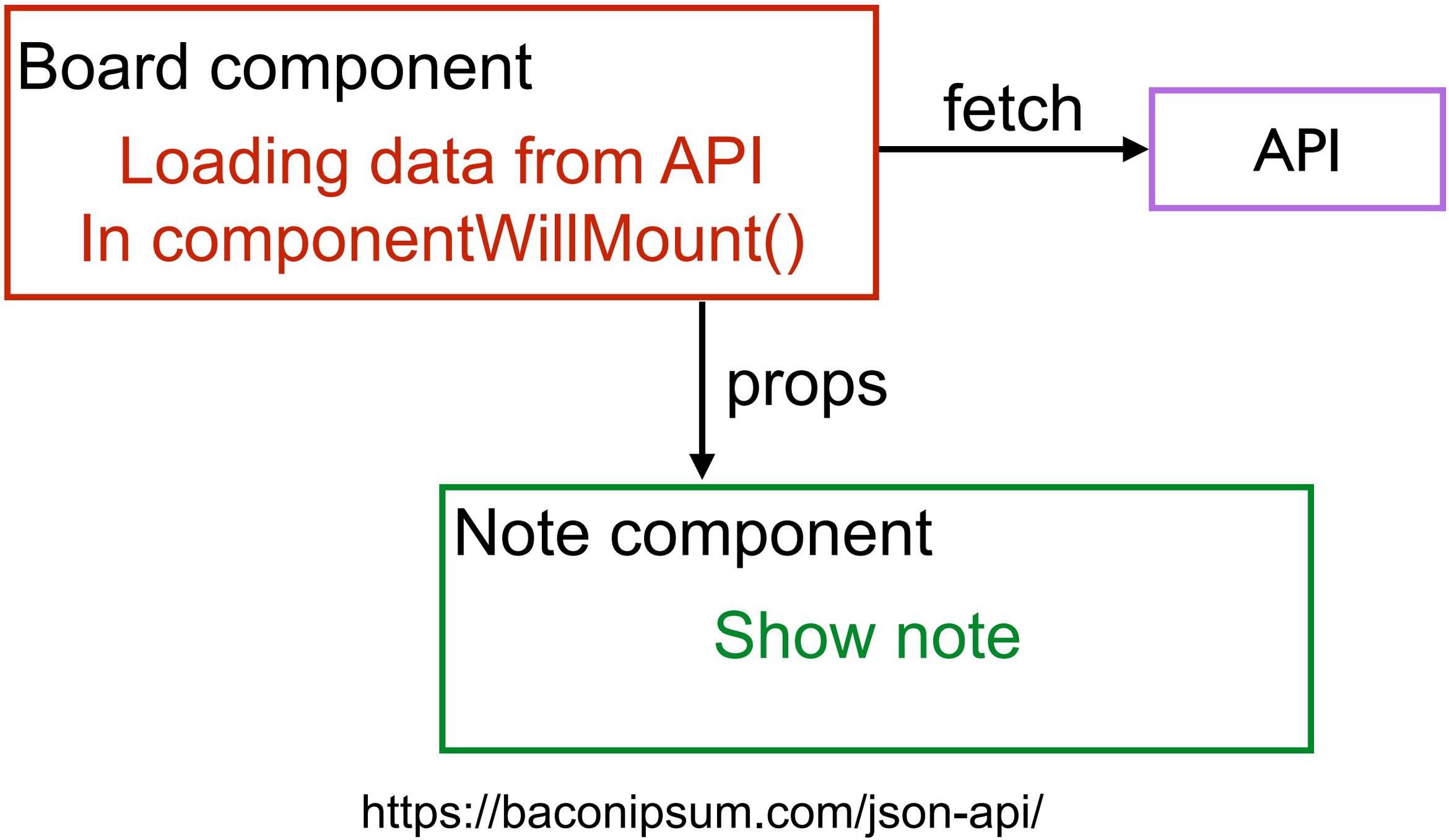
Component Life Cycle



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

React Hooks

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