



Learning React

# Basic of React





Somkiat
Home

Somkiat Puisungnoen

Update Info 1

View Activity Log 10+

...

Timeline

About

Friends 3,138

Photos

More ▾

When did you work at Opendream?

×

...

22 Pending Items

Intro

Software Craftsmanship

Software Practitioner at สยามชำนานุกิจ พ.ศ. 2556

Agile Practitioner and Technical at SPRINT3r

Post

Photo/Video

Live Video

Life Event

What's on your mind?

Public ▾

Post

Somkiat Puisungnoen

15 mins · Bangkok · ▾

Java and Bigdata

...



Facebook interface for the page **somkiat.cc**. The top navigation bar includes the Facebook logo, the page name **somkiat.cc**, a search icon, and user profile icons for **Somkiat** and **Home**. The main navigation bar shows **Page** (selected), **Messages**, **Notifications** (3), **Insights**, **Publishing Tools**, **Settings**, and **Help**.

The page content area features a large video player showing a man in a white Superman t-shirt with "SOMKIAT.CC" on it, posing against a white wall. A blue overlay on the video says "Help people take action on this Page." with a close button. Below the video are buttons for **Liked**, **Following**, **Share**, and a menu icon. A blue button labeled **+ Add a Button** is also present.

The left sidebar shows the page name **somkiat.cc**, the handle **@somkiat.cc**, and a menu with **Home** (selected), **Posts**, **Videos**, and **Photos**. A small thumbnail of the video is visible above the menu.

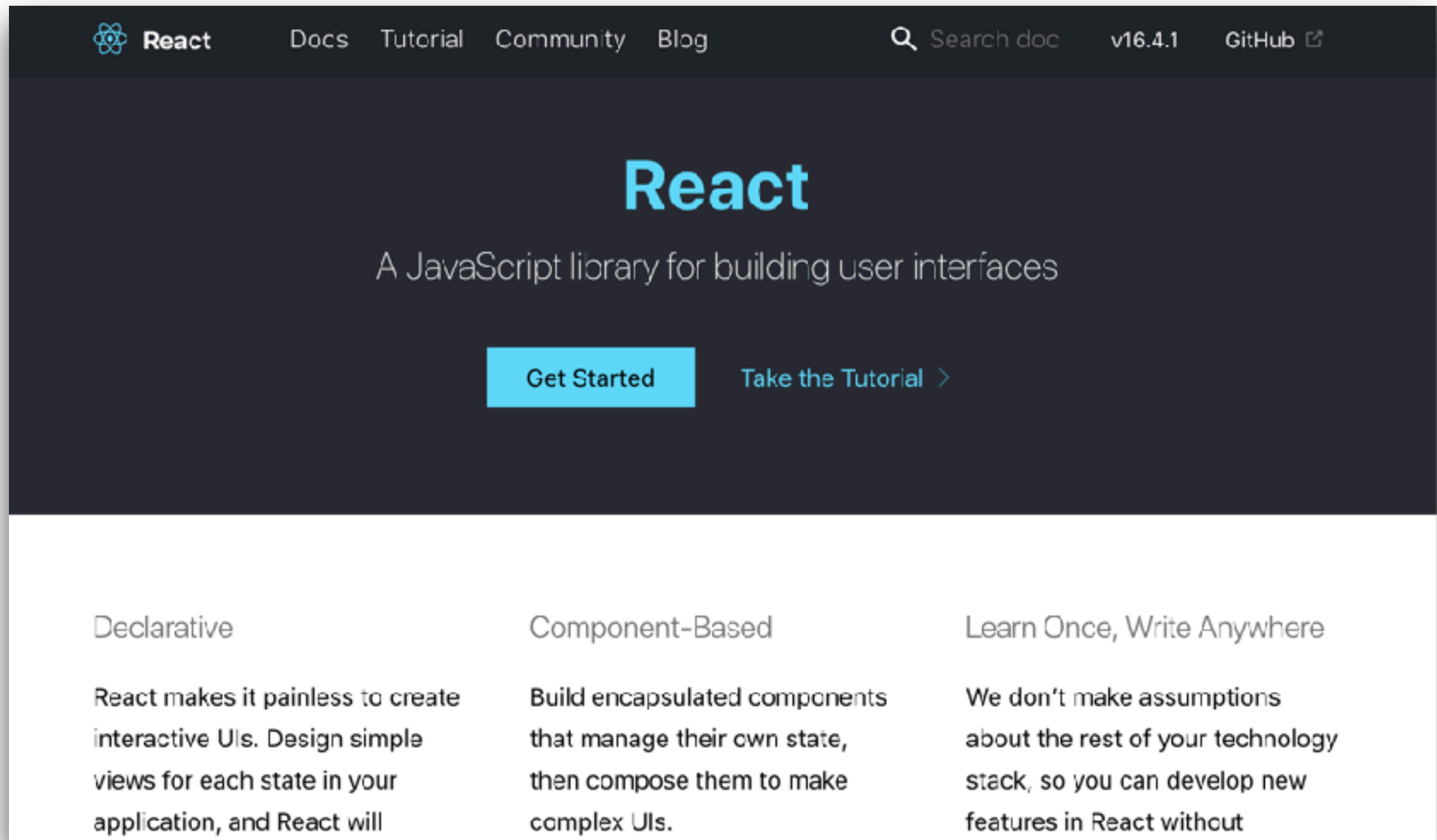


# Basic of React

1. Install Yarn
2. Create a react application
3. Design and Develop application
4. Deploy application to Github Pages



# React



The image is a screenshot of the React.js homepage. At the top, there is a dark navigation bar with the React logo and 'React' text on the left, and links for 'Docs', 'Tutorial', 'Community', and 'Blog' in the center. On the right side of the navigation bar are a search icon with 'Search doc', the version 'v16.4.1', and a 'GitHub' link with an external icon. Below the navigation bar, the word 'React' is displayed in a large, light blue font. Underneath it, the tagline 'A JavaScript library for building user interfaces' is written in a smaller, light gray font. Two buttons are positioned below the tagline: a solid blue 'Get Started' button and a light blue 'Take the Tutorial >' button. The bottom section of the page features three columns of text on a white background. The first column is titled 'Declarative' and describes how React makes it easy to create interactive UIs. The second column is titled 'Component-Based' and explains how components manage their own state and are composed into complex UIs. The third column is titled 'Learn Once, Write Anywhere' and states that React doesn't make assumptions about the rest of the technology stack.

React

A JavaScript library for building user interfaces

[Get Started](#) [Take the Tutorial >](#)

Declarative	Component-Based	Learn Once, Write Anywhere
React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will	Build encapsulated components that manage their own state, then compose them to make complex UIs.	We don't make assumptions about the rest of your technology stack, so you can develop new features in React without

<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

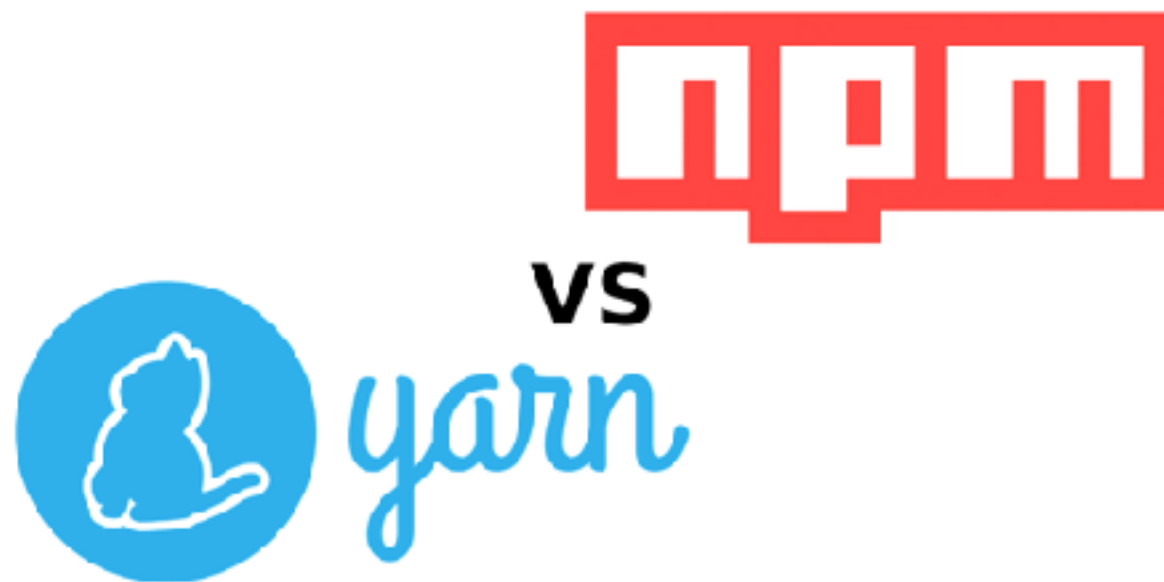
<https://reactjs.org/docs/getting-started.html>





# Installation

Yarn or npm  
Create-react-app



# Install Yarn

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

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



# Create a react application

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

**\$yarn** create react-app hello

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

**\$npm** install -g create-react-app

**\$create-react-app** hello

<https://github.com/facebook/create-react-app>



# Issue !!!

```
Aborting installation.  
Unexpected error. Please report it as a bug:  
{ Error: Cannot find module '/Users/somkiat/data/learn-react-academy/my-app2/node_modules/react-scripts/package.json'  
    at Function.Module._resolveFilename (internal/modules/cjs/loader.js:815:15)  
    at Function.Module._load (internal/modules/cjs/loader.js:642:12)  
    at Module.require (internal/modules/cjs/loader.js:859:19)  
    at require (internal/modules/cjs/helpers.js:71:18)  
    at checkNodeVersion (/Users/somkiat/node_modules/react-scripts/scripts/build.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!

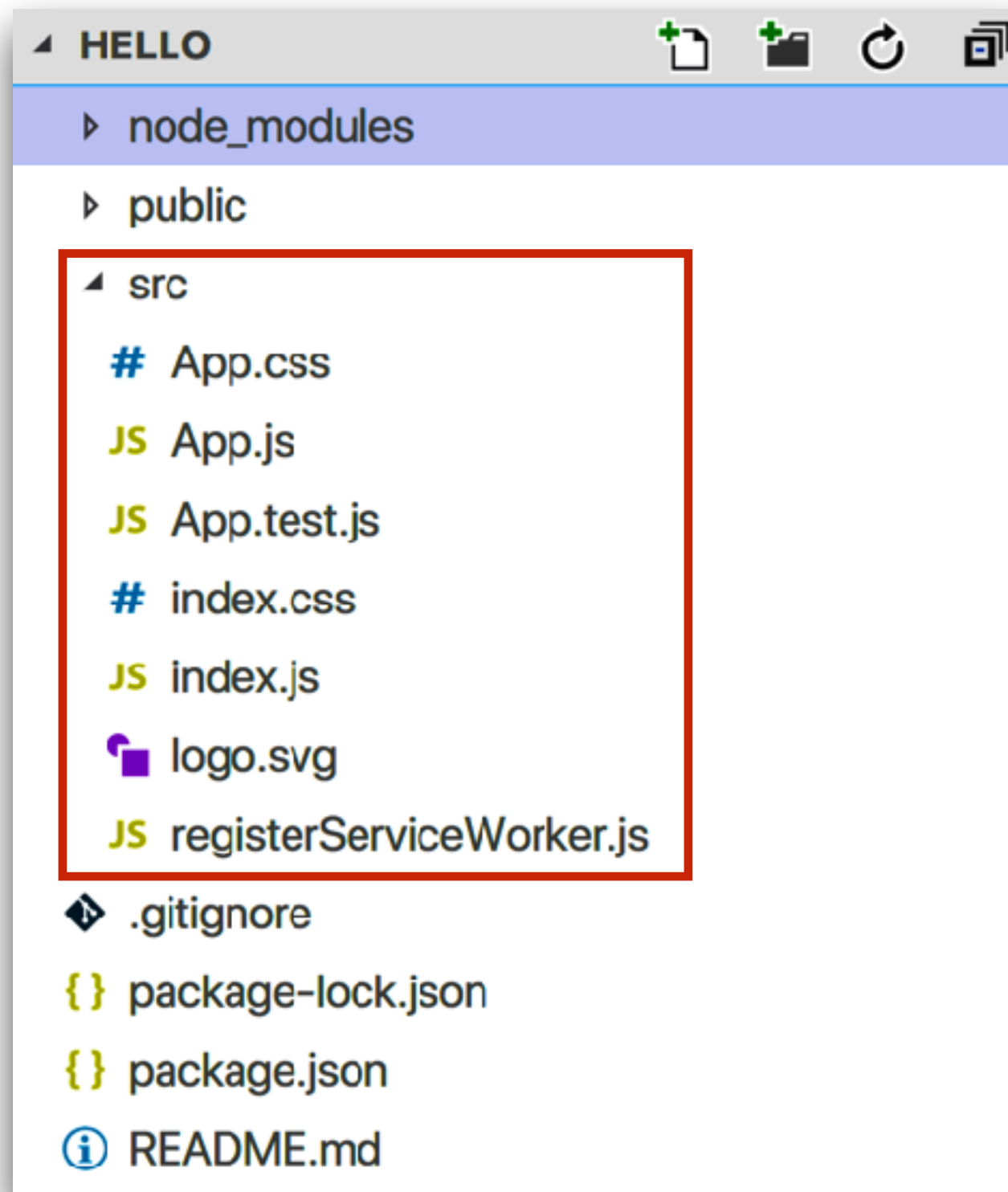
We suggest that you begin by typing:

`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.

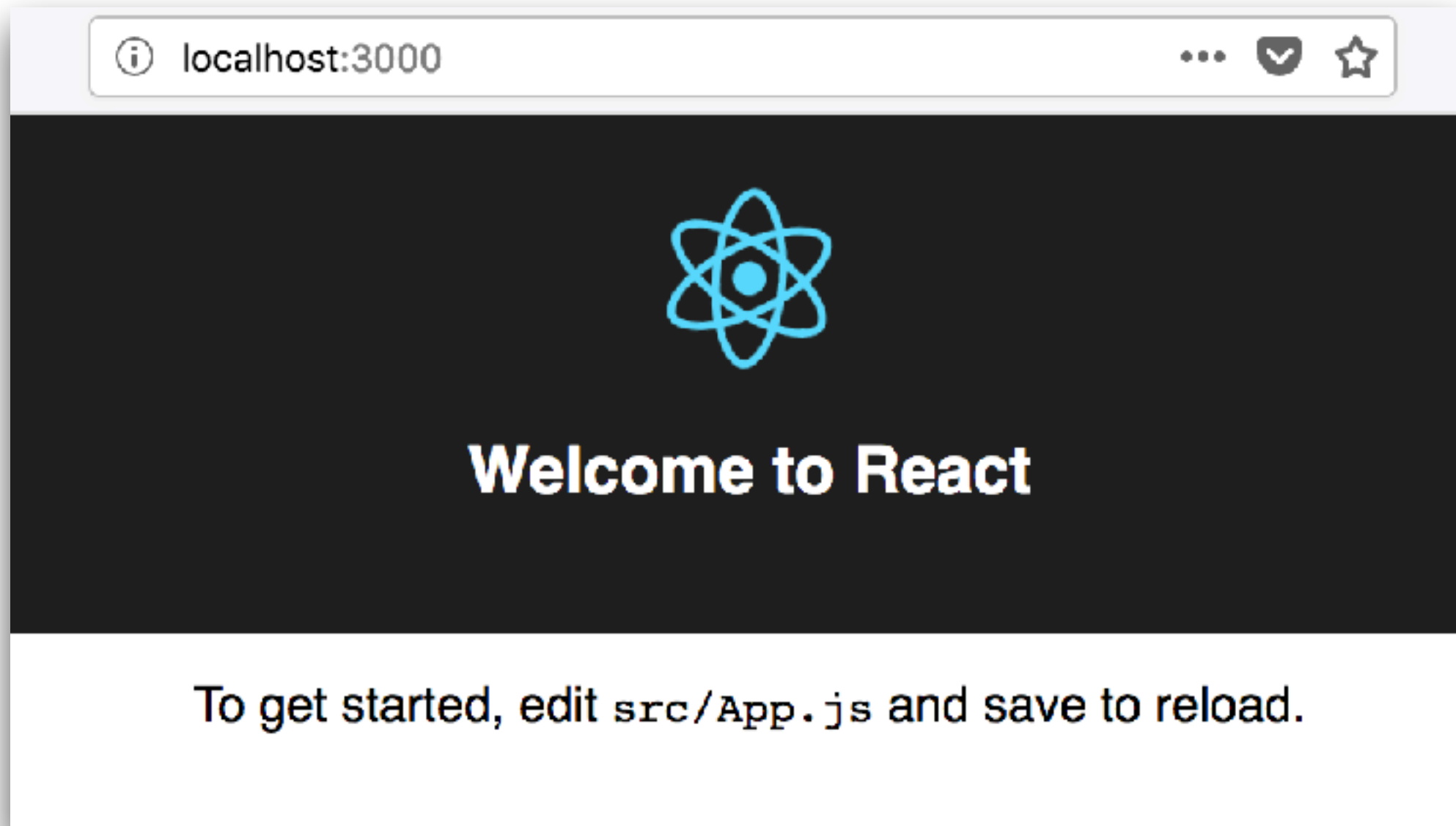
**Local:** `http://localhost:3000/`

**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



# View source !!

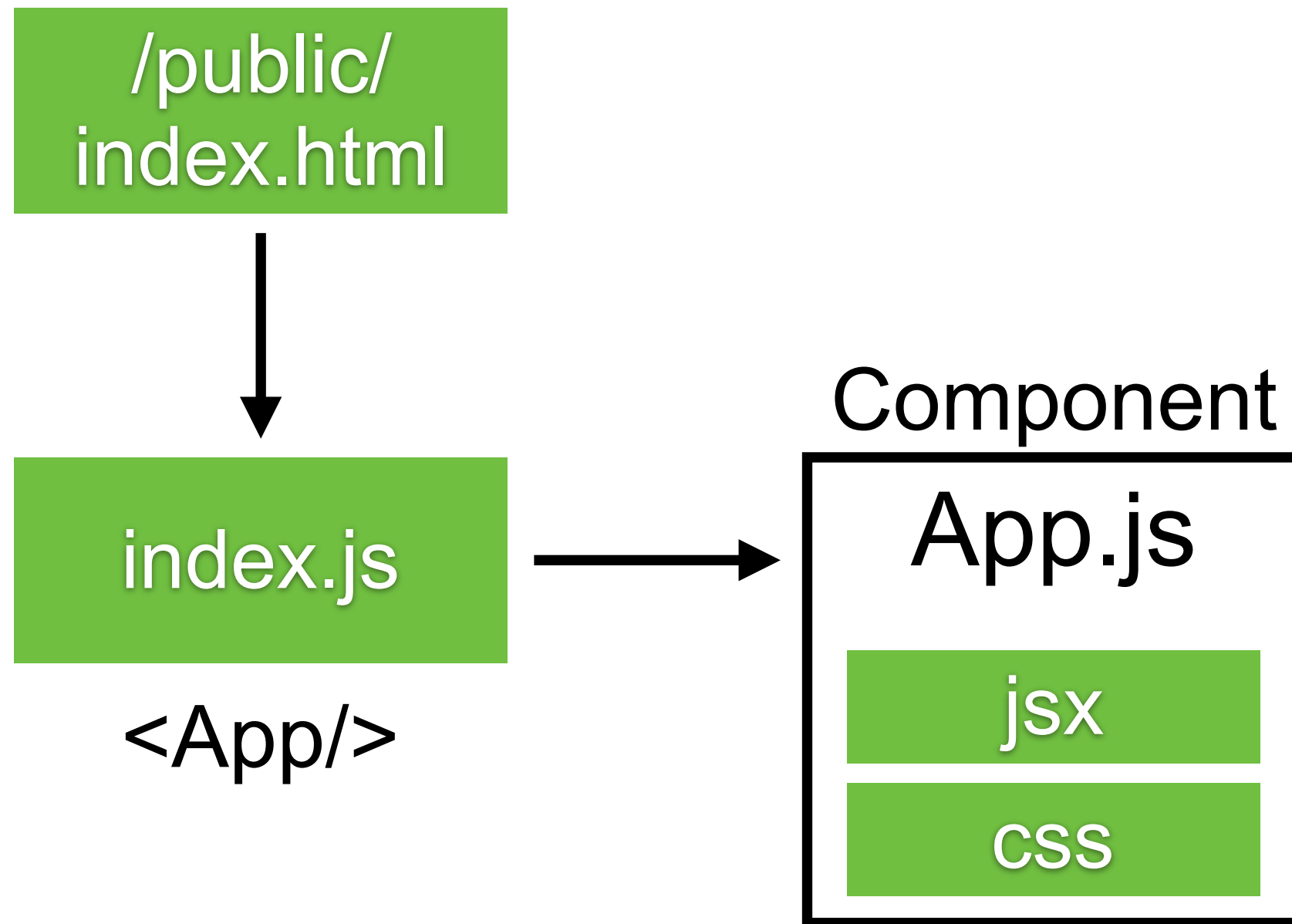
```
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="theme-color" content="#000000">
    <!--
      manifest.json provides metadata used when your web app is added to the
      homescreen on 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.
    -->
  </body>
</html>
```

<div id="root"></div>



# Workflow



# 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 <div id="root"/>*



# App.js

```
class App extends Component {
```

```
  render() {
```

```
    return (
```

*JSX (JavaScript XML)*

```
      <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>
```

```
    );
```

```
  }
```

```
}
```





# 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

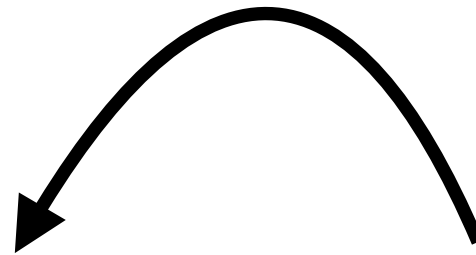
Split the UI into independent reusable pieces

Each independent pieces is processed separately

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

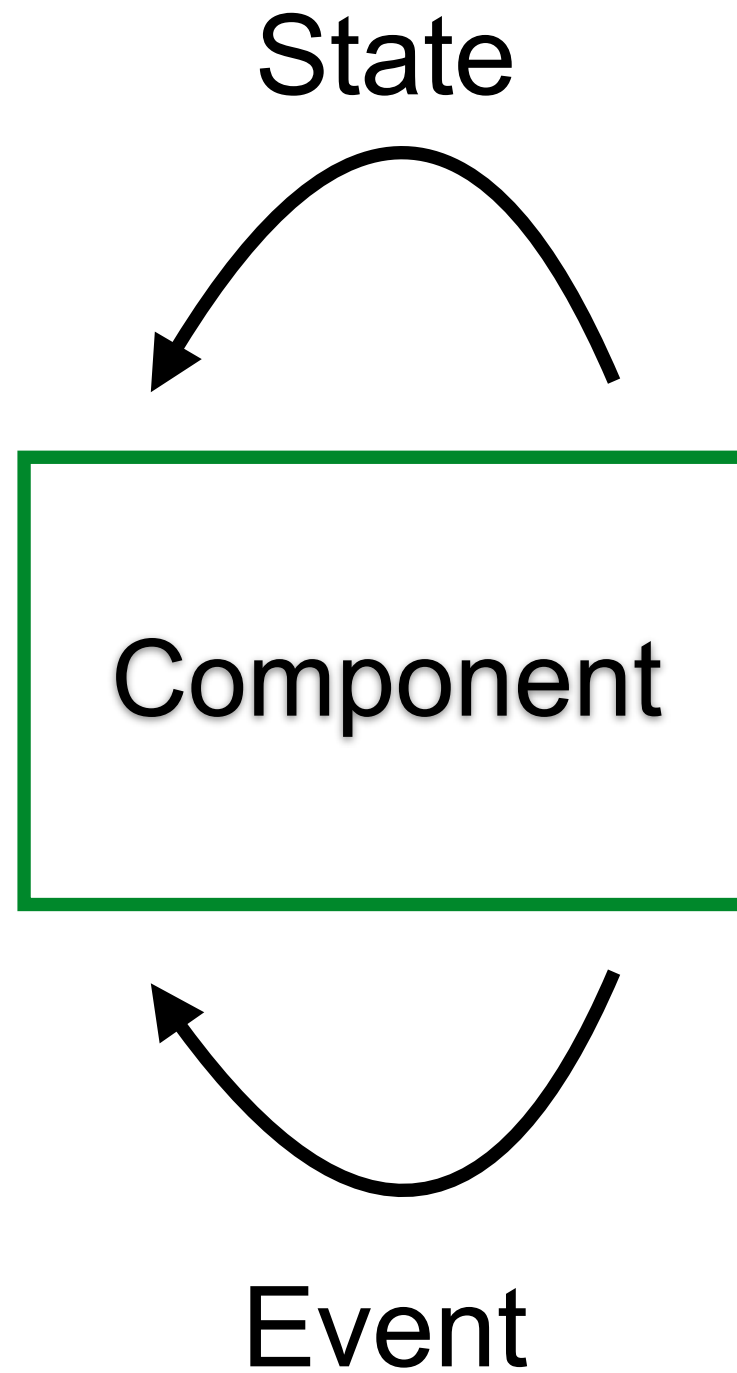


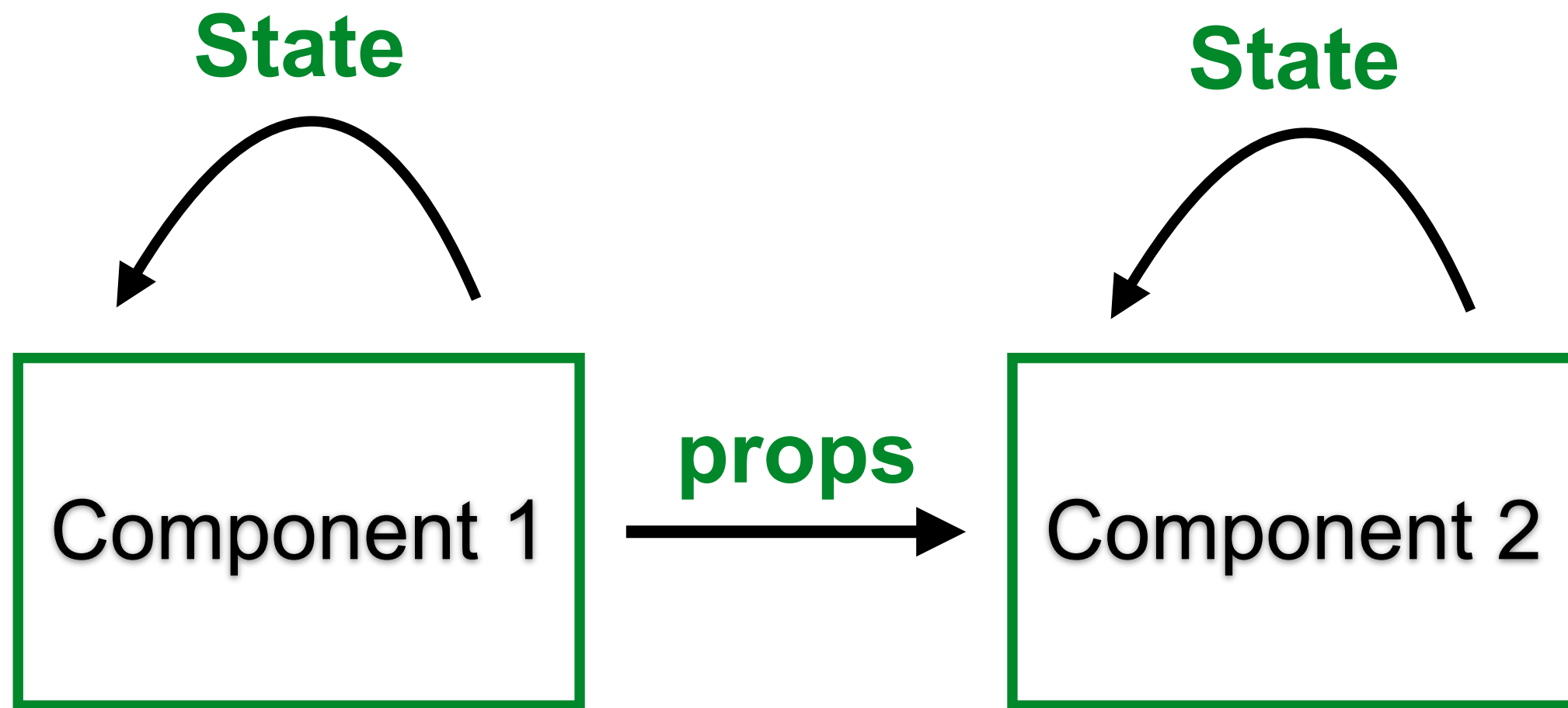
State



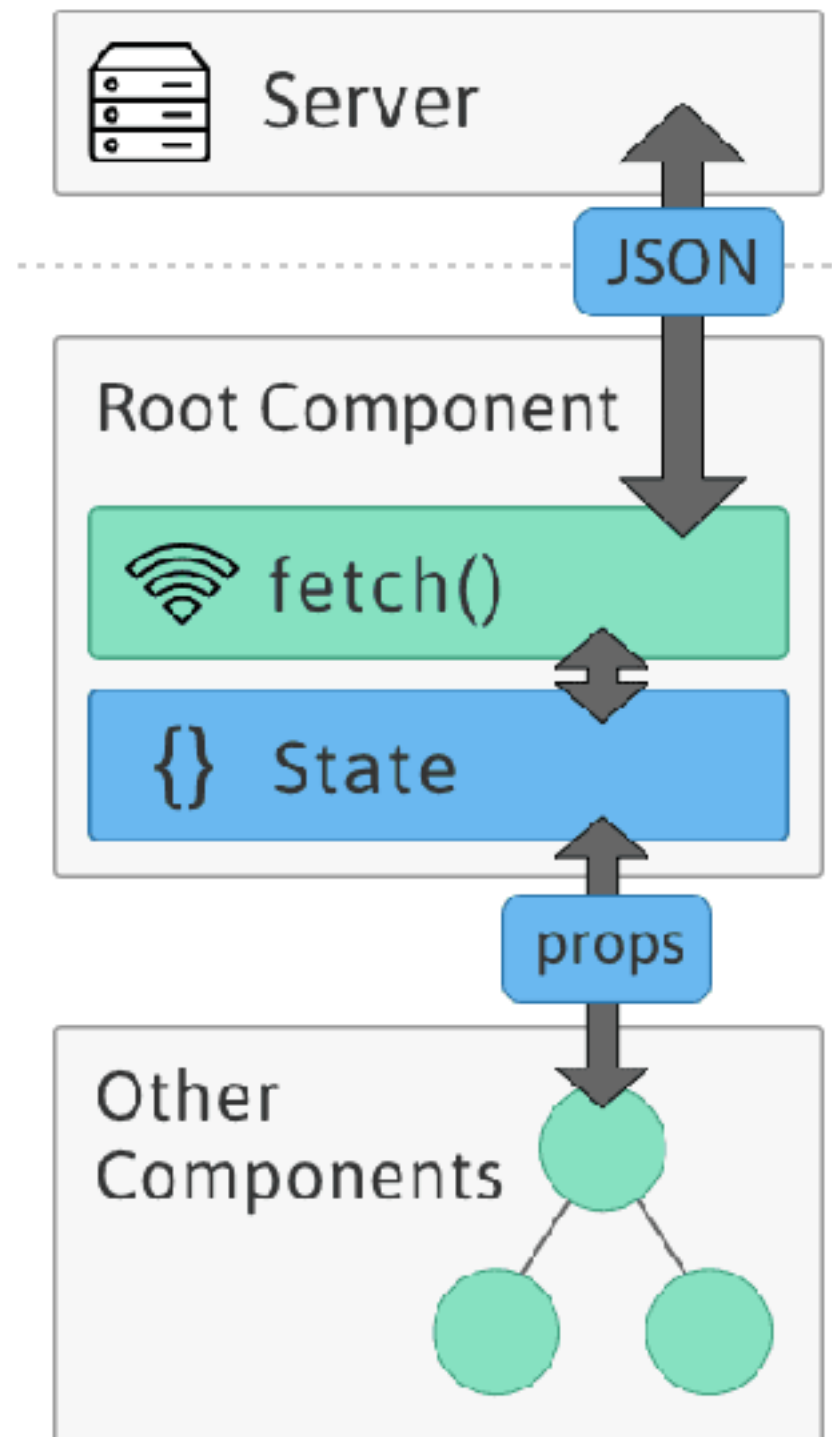
Component





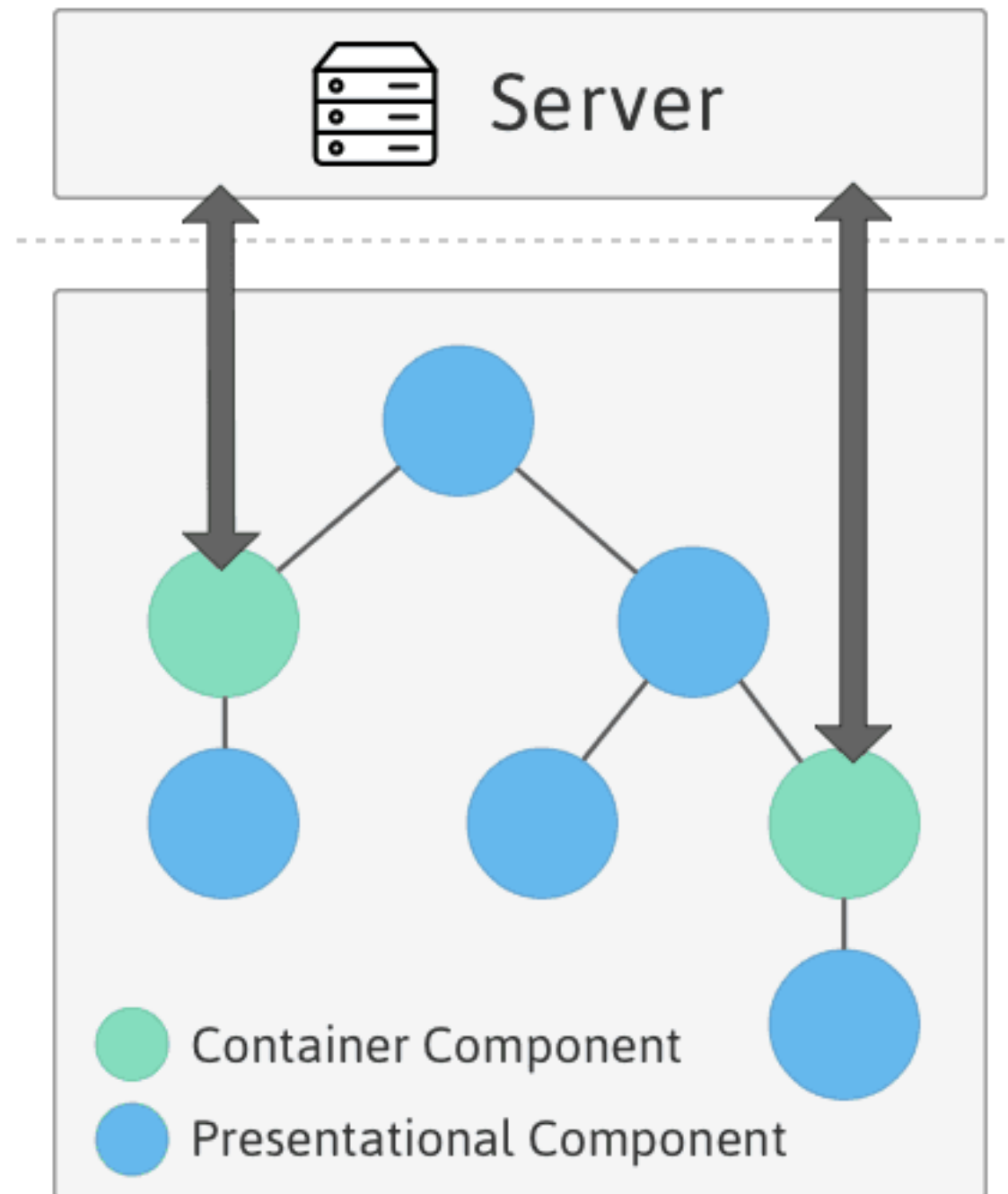


# State and Props





# Container component



# Smart and Dumb 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

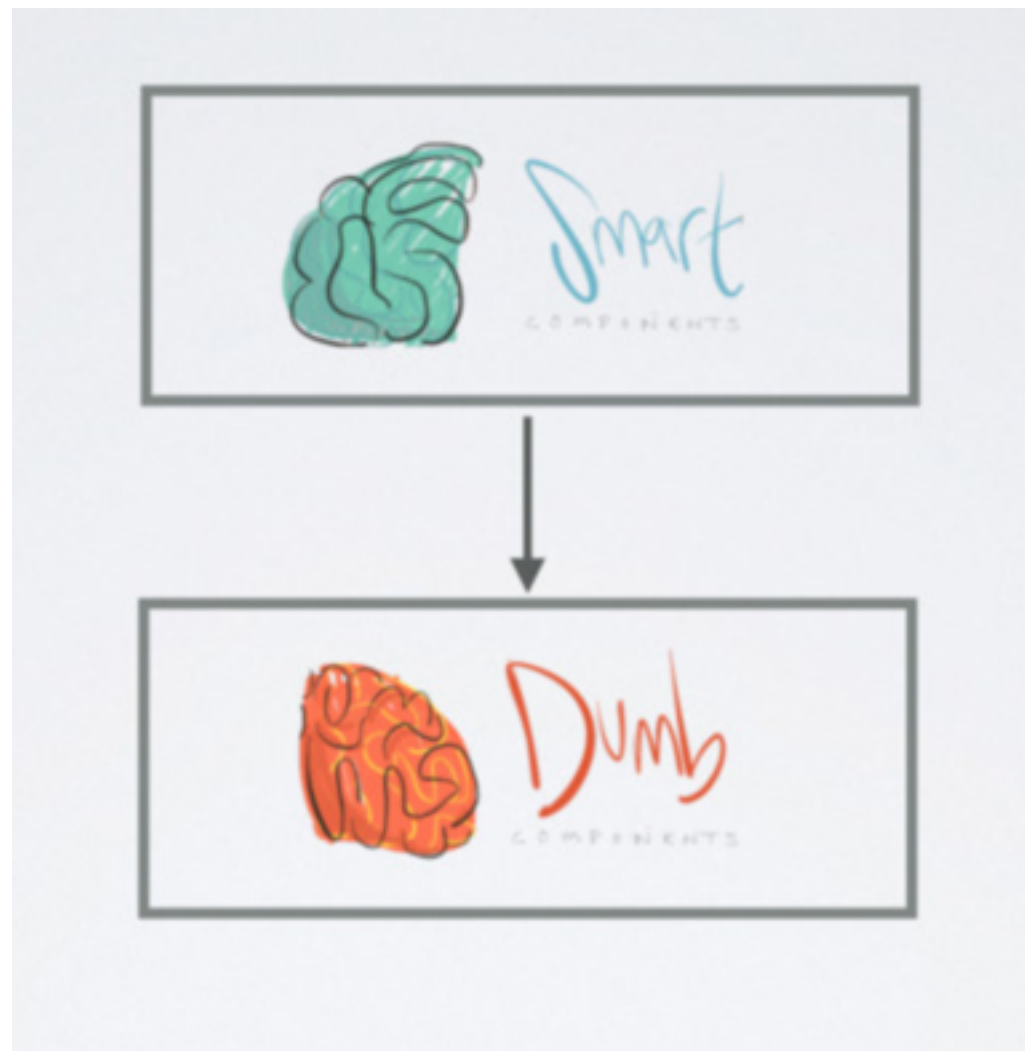
No app dependencies

Just props, for data and callbacks

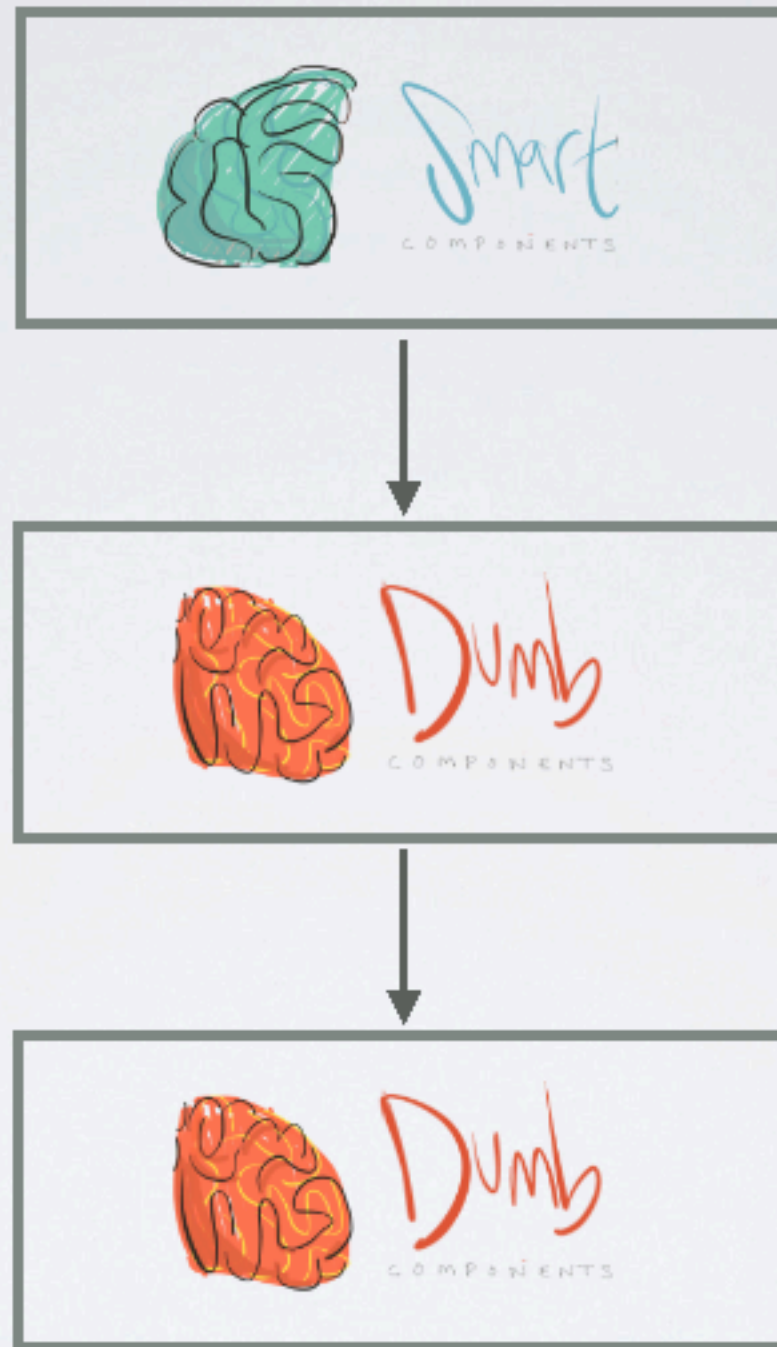
Rerely have own state, only UI state



# Smart and Dumb component

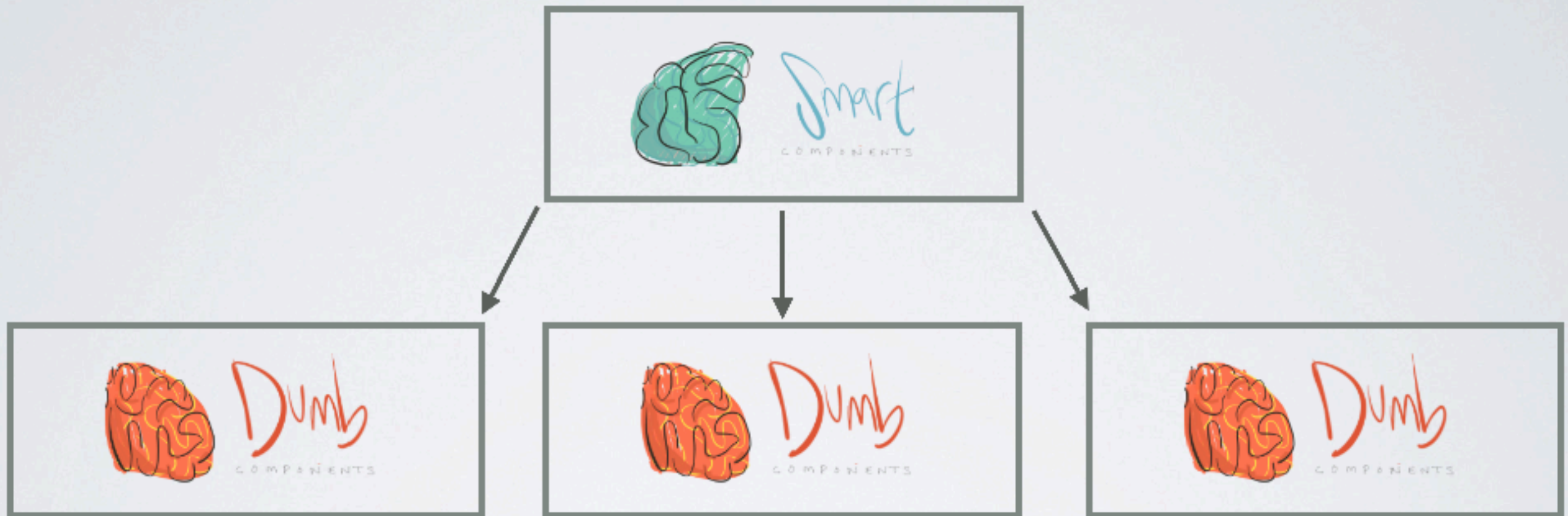


# Smart and Dumb component





# 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

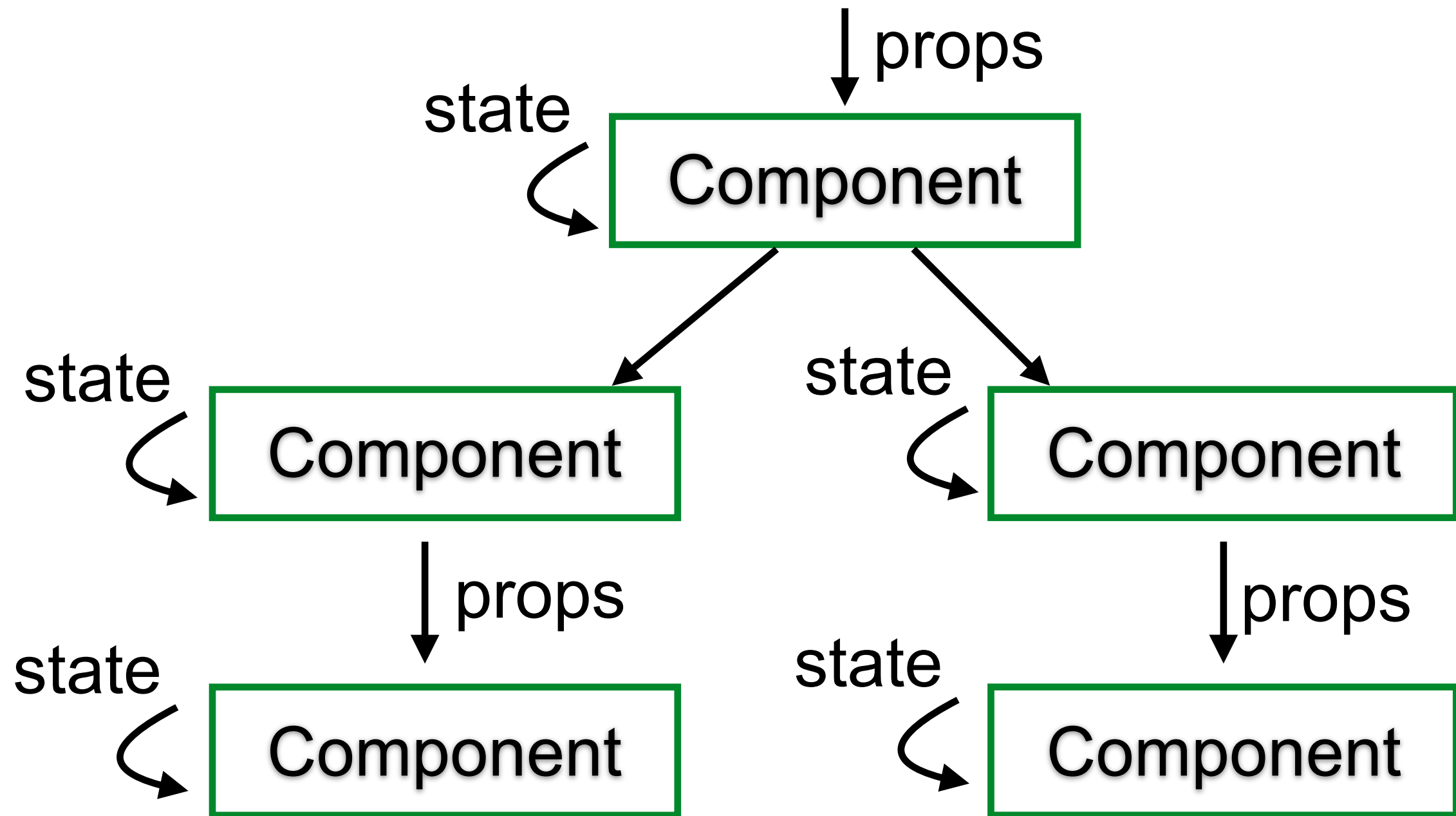
Use to determine component rendering and behavior

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 like 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

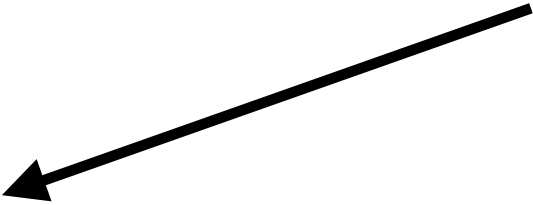
```
change2 = (e) => {  
  console.log('this is:', e.target.value);  
  this.setState({  
    name: e.target.value  
  })  
}
```



# 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  
  })  
}
```

A black arrow points from the `this.change1 = this.change.bind(this)` line in the constructor to the `change(e) {` line in the `change` method, illustrating how the method is bound to the component instance.

# Develop your application

Start with React !!





# Design your app ?

IPA

LAGER

Stout



# Let's design your component



# Component for react ?

1 component ?

IPA

LAGER

Stout



# Component for react ?

3 component ?

input your beer

ADD

IPA

LAGER

Stout



# Component for react ?

4 component ?

input your beer

ADD

IPA

LAGER

Stout



# Components

## BeerListContainer

AddBeerComponent

Input text

Button

BeerListContainer

BeerItemComponent



# Components

## BeerContainer/App

### AddBeer

Input text

Button

### BeerList

BeerItem



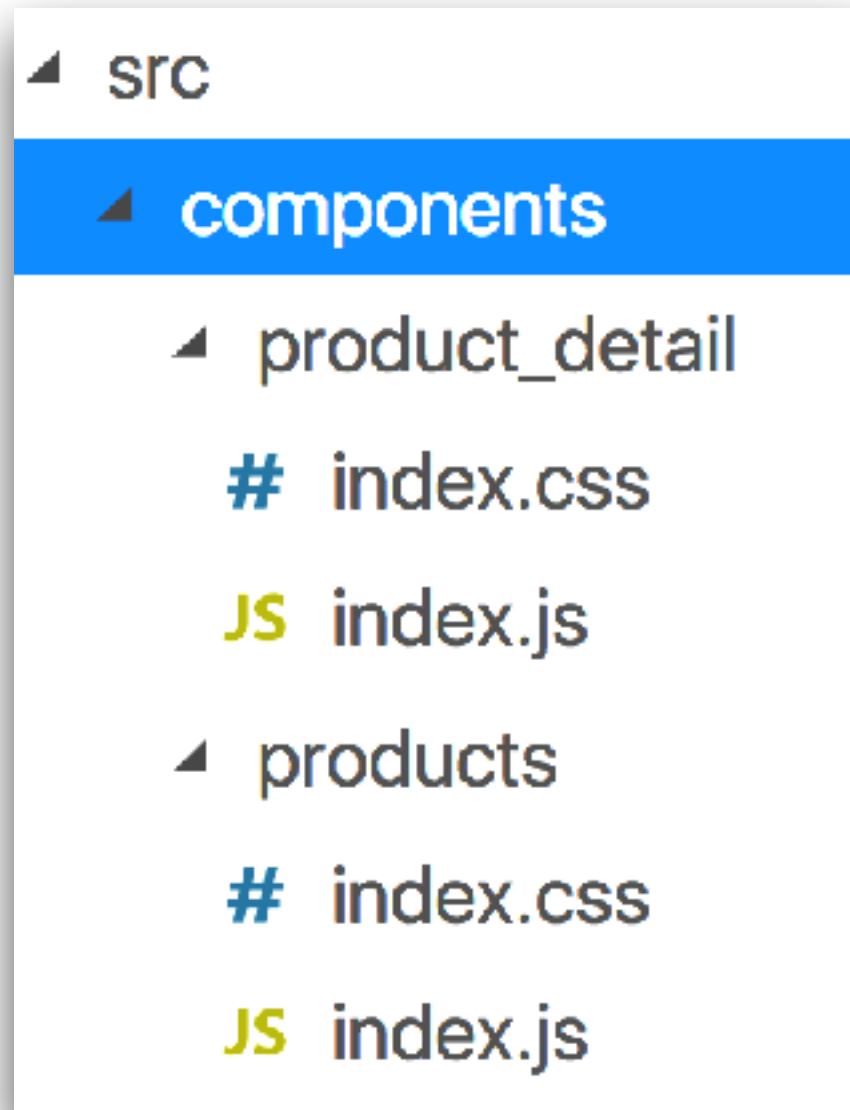
# Step 1

## Create react component





# Component structure



# AddBeer component

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

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

```
export default AddBeer;
```



# 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>
    );
  }
}

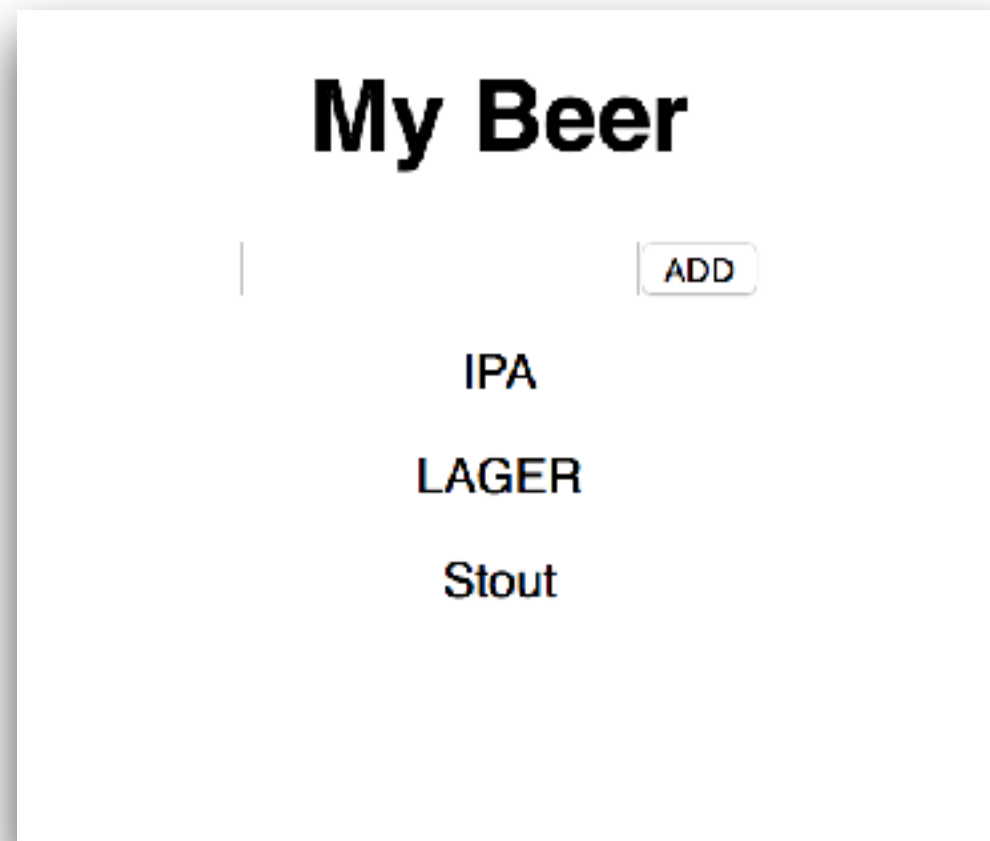
export default App;
```



# Composition over Inheritance

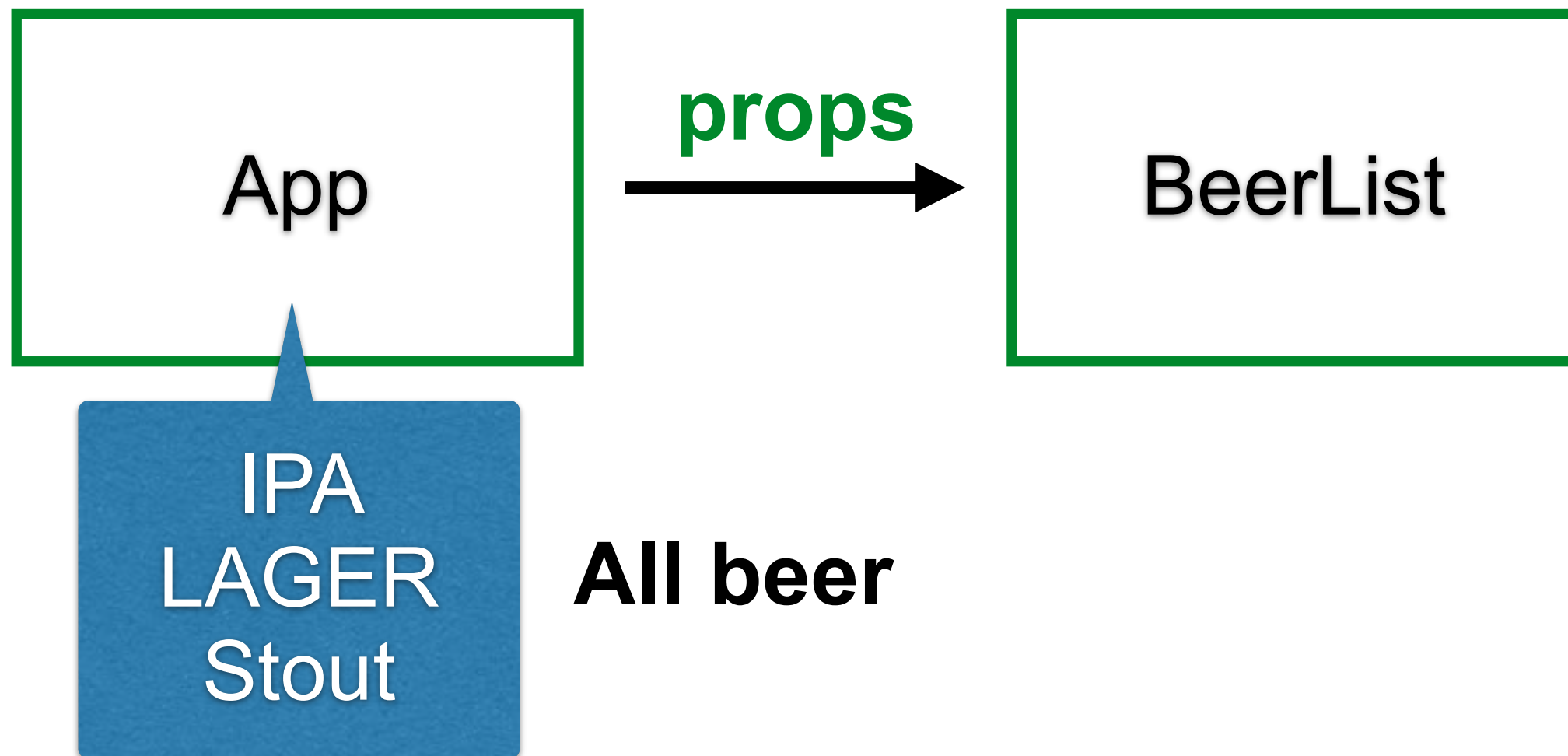


# See result



Smart component

Dumb component



# 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>  
    );  
  }  
}
```

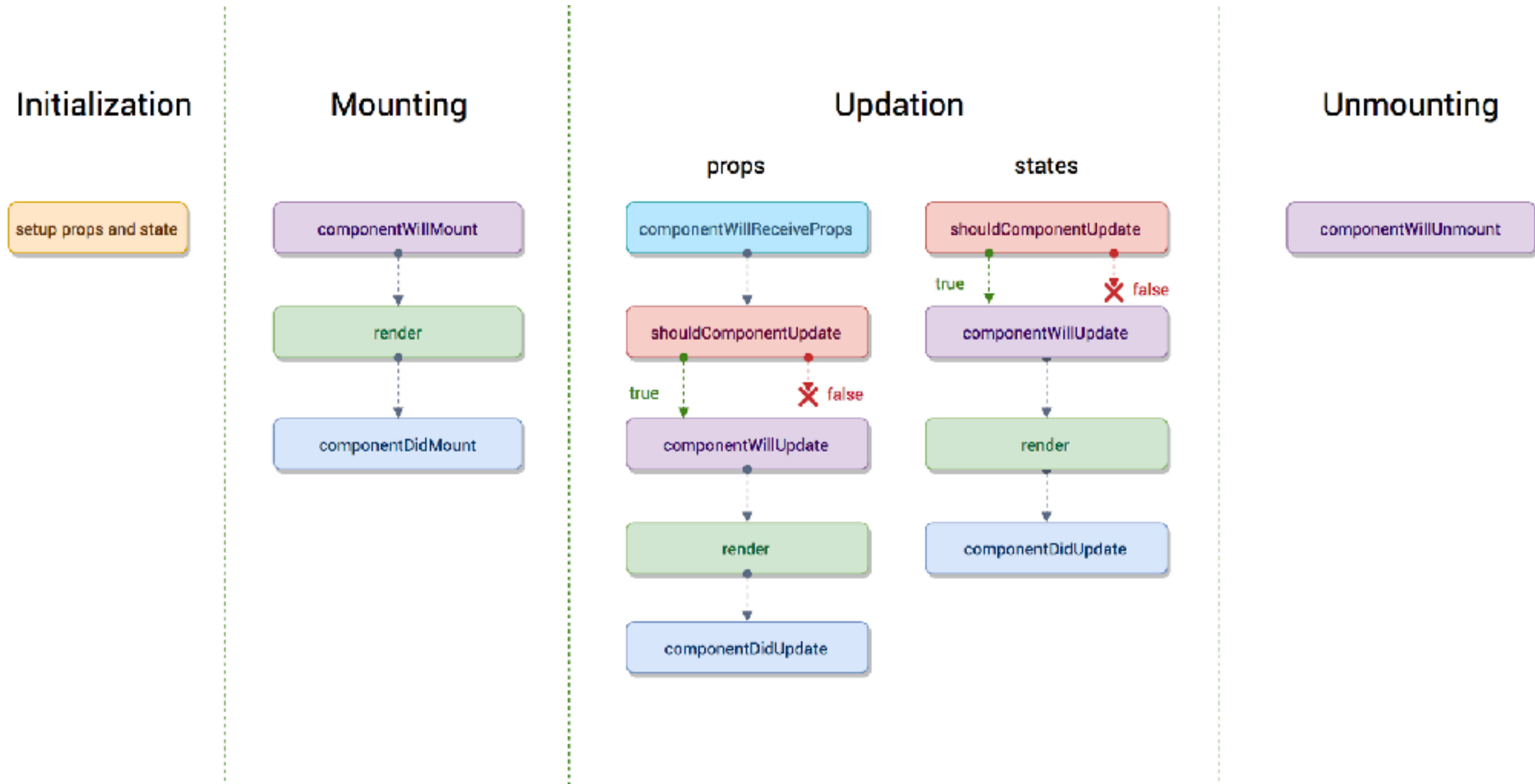


# Lifecycle

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



# Lifecycle

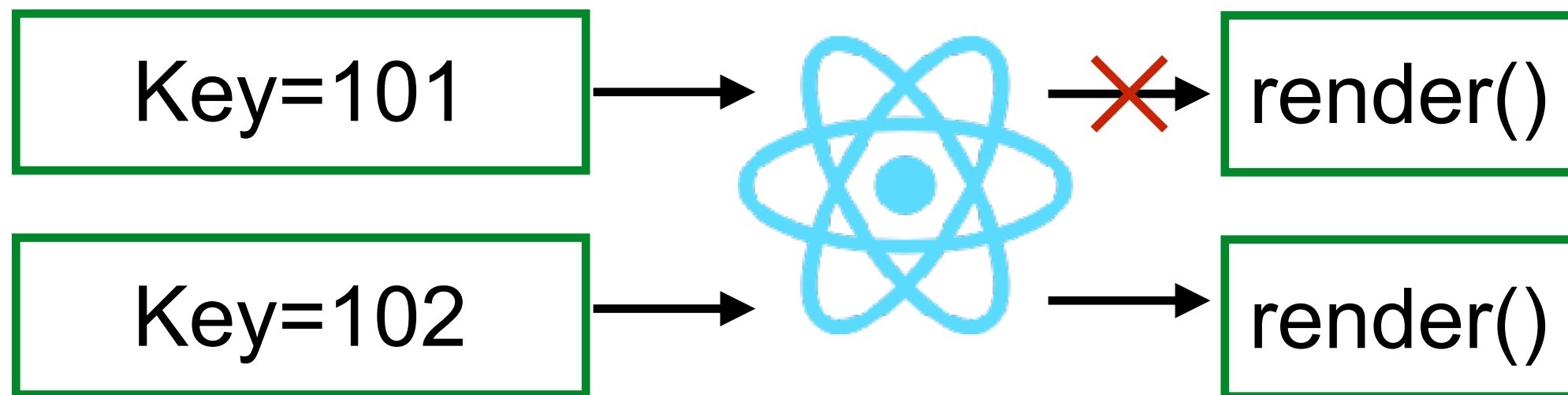


<https://hackernoon.com/reactjs-component-lifecycle-methods-a-deep-dive-38275d9d13c0>



# 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>  
  </Link>  
</div>
```



`this.props.match.params.id`



# Working with API



## PUNK API<sub>v2</sub>

[V2 Documentation](#)

[https://api.punkapi.com/v2/beers?per\\_page=10](https://api.punkapi.com/v2/beers?per_page=10)

<https://punkapi.com/documentation/v2>



# 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

