

## เรียนเขียน Web Application ด้วย ReactJS, NodeJS Express และ MongoDB



สอนโดยผู้พัฒนาระบบ posPOS  
[www.pospos.co](http://www.pospos.co)

สนใจติดต่อ บริษัท โค้ดมوبайлส์ จำกัด

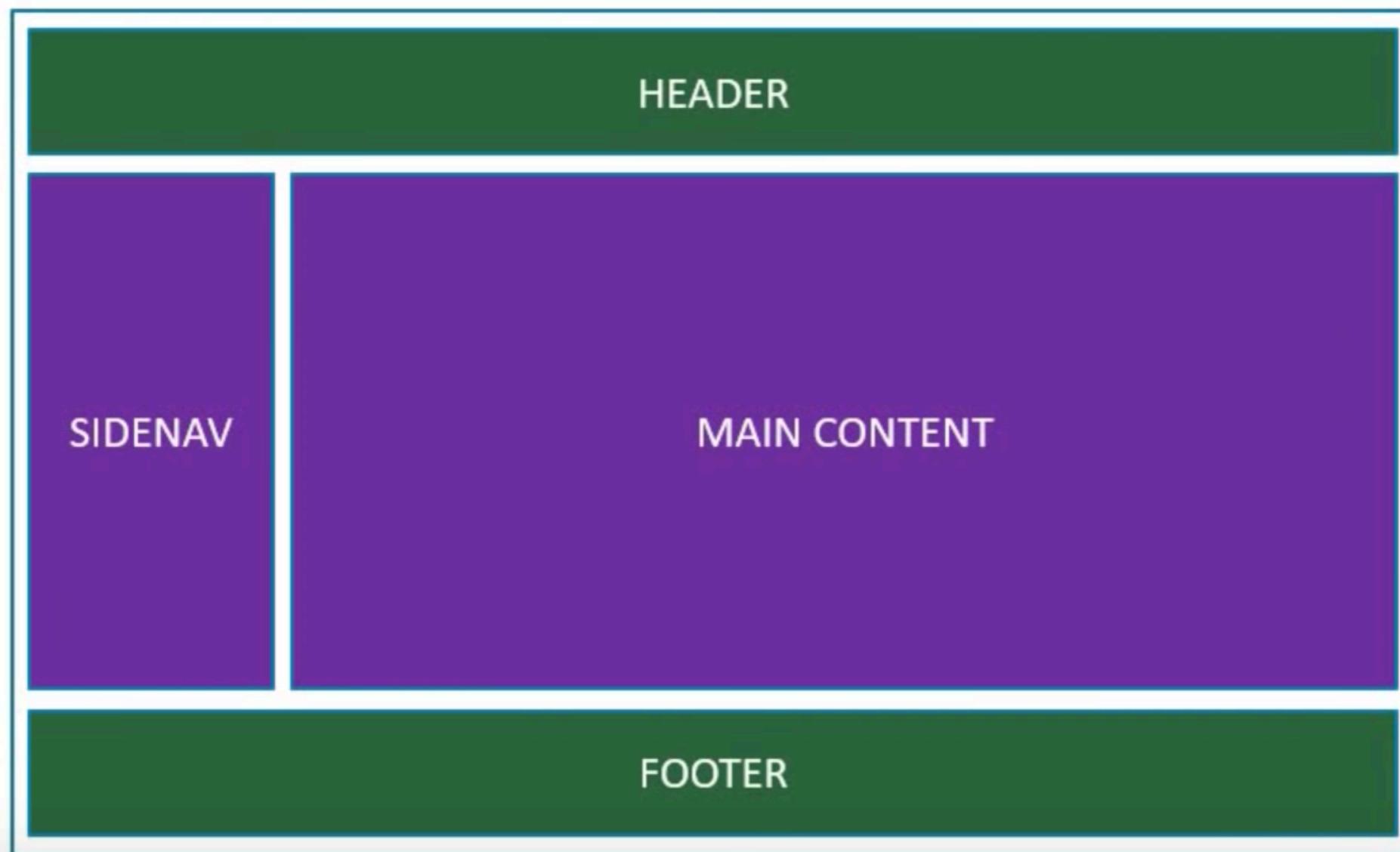
- 📞 081-359-9468
- ✉️ support@codemobiles.com
- 🏠 [www.codemobiles.com](http://www.codemobiles.com)



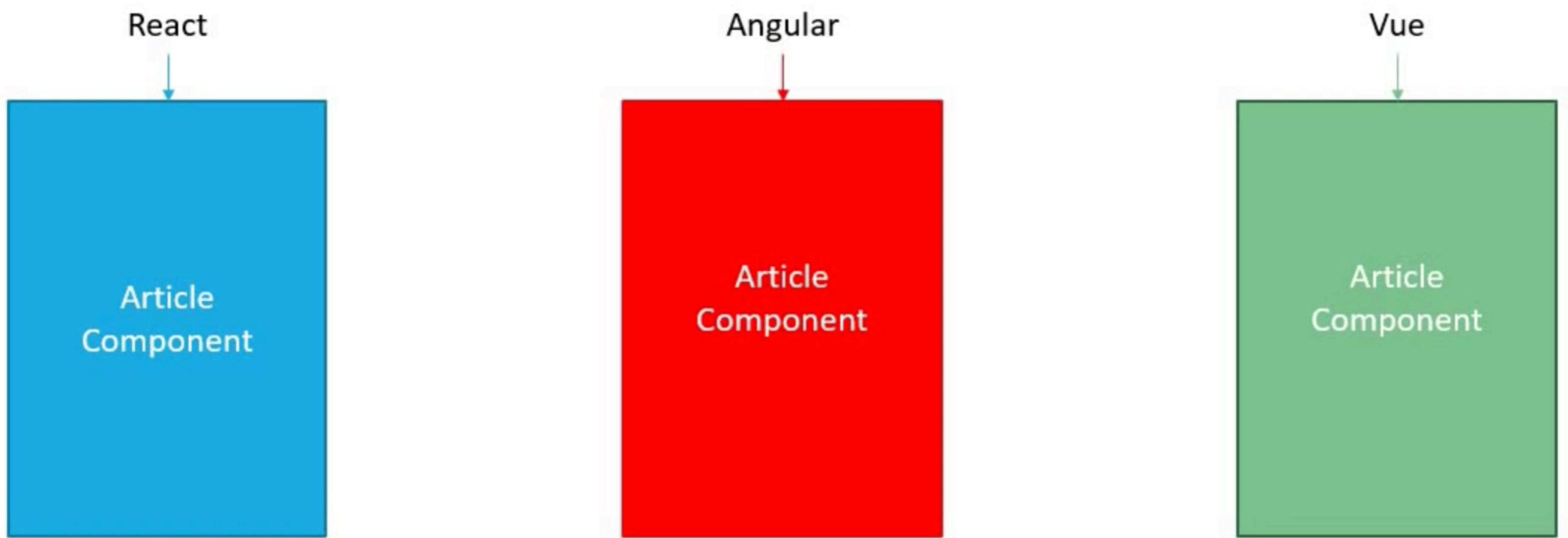
# What is React?

- React is a declarative, efficient Javascript Library - for building user interface.
- React, there are two versions
  - for Web (ReactJS)
  - for Mobile (ReactNative)
- React is like Angular **BUT** it is just an UI library not framework like angular.
- React is used to implement single-page web application "SPA"
- React is developed by Facebook
- React-JS concepts and syntax can be adopt with React-Native - to develop mobile app over 70%.

# Component Based Architecture



# Reusable code



# Prerequisites

HTML, CSS and JavaScript fundamentals

ES6

JavaScript – ‘this’ keyword, filter, map and reduce

ES6 – let & const, arrow functions, template literals, default parameters, object literals, rest and spread operators and destructuring assignment.

React from scratch

# Core 3rd Party Libraries

Fundamentals

HTTP

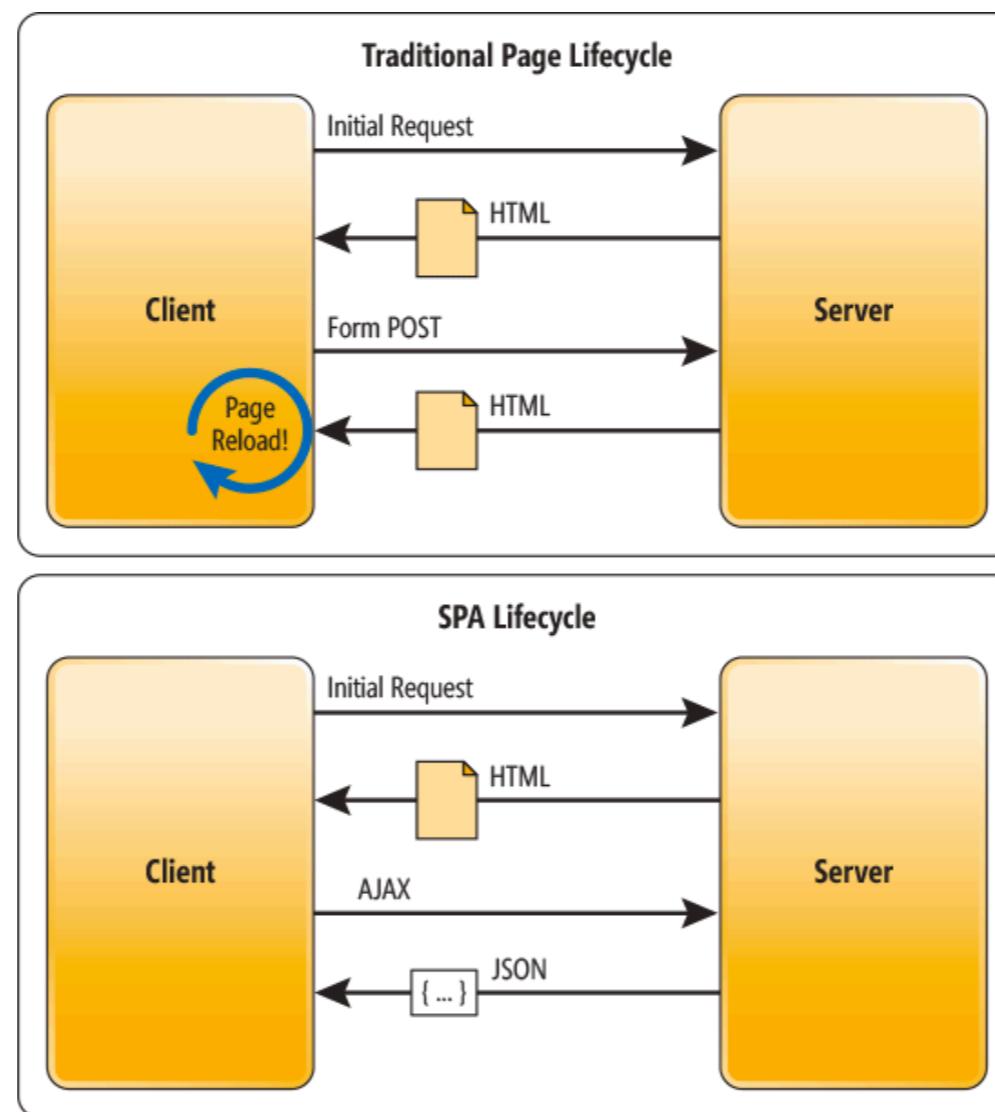
Routing

Redux

Utilities

# Glossary

# Single-Page Application



# ES6, ES2015, ES2016



# Compiler

A JavaScript compiler takes JavaScript code, transforms it and returns JavaScript code in a different format. The most common use case is to take ES6 syntax and transform it into syntax that older browsers are capable of interpreting. Babel is the compiler most commonly used with React.

BABEL

# Bundlers

Bundlers take JavaScript and CSS code written as separate modules (often hundreds of them), and combine them together into a few files better optimized for the browsers. Some bundlers commonly used in React applications include [Webpack](#) and [Browserify](#).



# Package Managers

Package managers are tools that allow you to manage dependencies in your project. npm and Yarn are two package managers commonly used in React applications. Both of them are clients for the same npm package registry.



# JSX

```
const name = 'Clementine';
ReactDOM.render(
  <h1 className="hello">My name is {name}!</h1>,
  document.getElementById('root')
);
```

# Element

```
const element = <h1>Hello, world</h1>;
```

# Component

React components are small, reusable pieces of code that return a React element to be rendered to the page. The simplest version of React component is a plain JavaScript function that returns a React element:

```
function Welcome(props) {  
  return <h1>Hello, {props.name}</h1>;  
}
```

Components can also be ES6 classes:

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

# Installation

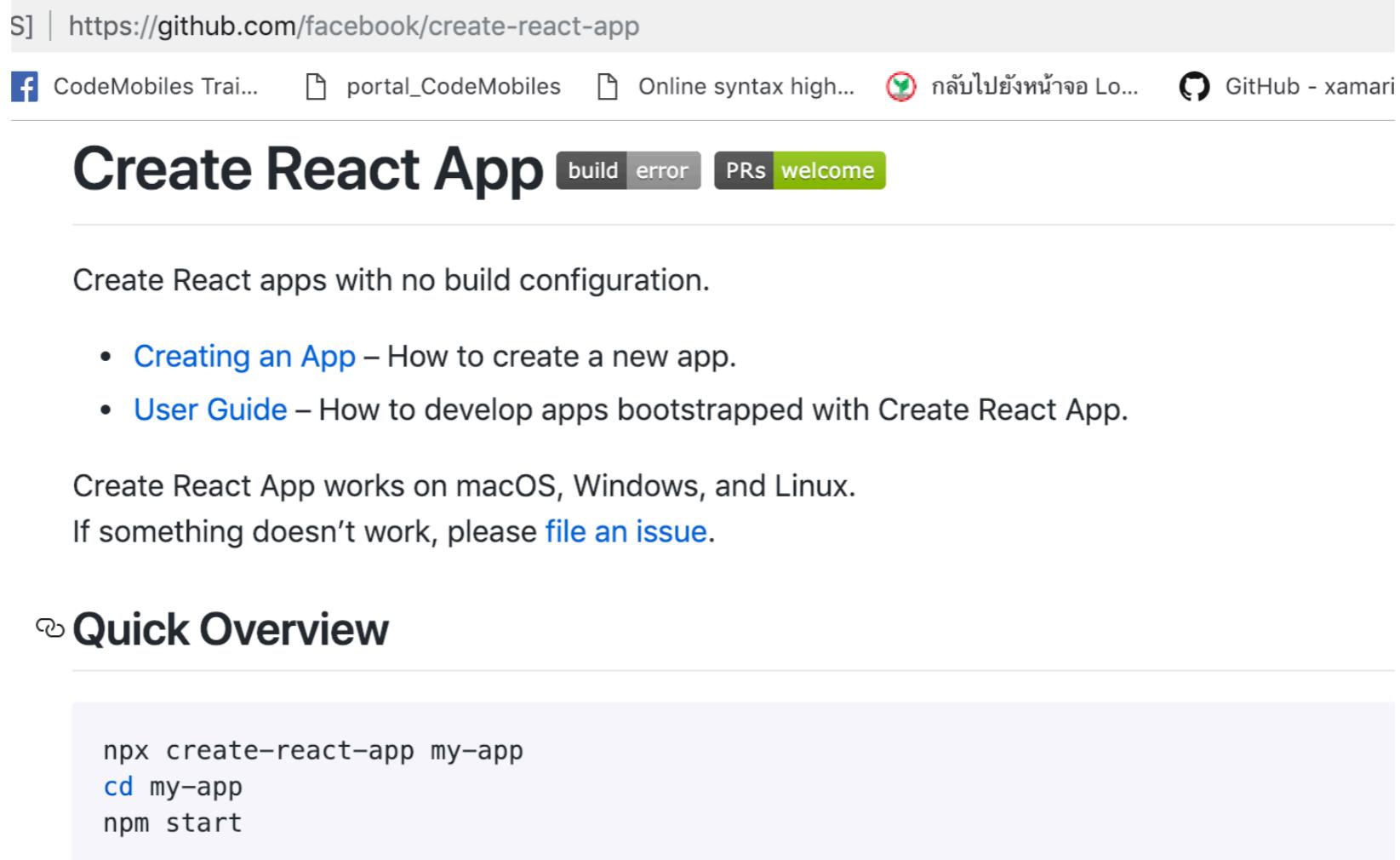
# Installation

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

- **NodeJS :** <https://nodejs.org/en/>
- **React-CLI :** npm i -g create-react-app
- **Run:** create-react-app hello-world

[https://www.youtube.com/watch?v=o5yo\\_0QeSyk](https://www.youtube.com/watch?v=o5yo_0QeSyk)

# More info about React-CLI "create-react-app"



S] | https://github.com/facebook/create-react-app

[!\[\]\(3b10bc7b7e66ba7dde2e8b14bda8bd0f\_img.jpg\) CodeMobiles Trai...](#) [!\[\]\(1b91da81ac67a289bcdc3eda2a94acf8\_img.jpg\) portal\\_CodeMobiles](#) [!\[\]\(60438908634babc0c17407a69ed5b02d\_img.jpg\) Online syntax high...](#) [!\[\]\(675fc01803c541189921c761309eac20\_img.jpg\) កត្តុប្រឈមនៃការ Lo...](#) [!\[\]\(31304ebc6578c169d1a008b8e1c68ec6\_img.jpg\) GitHub - xamari](#)

## Create React App

build error PRs welcome

Create React apps with no build configuration.

- [Creating an App](#) – How to create a new app.
- [User Guide](#) – How to develop apps bootstrapped with Create React App.

Create React App works on macOS, Windows, and Linux.  
If something doesn't work, please [file an issue](#).

### Quick Overview

```
npx create-react-app my-app
cd my-app
npm start
```

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

# Installation

The screenshot shows a video management interface with two entries:

- CMDev: React JS Installation on macOS (ติดตั้ง)**  
Share | Embed | Email  
https://youtu.be/acVnzWPl9ZE
- CMDev: React JS Installation on Windows (ติดตั้ง)**  
Share | Embed | Email  
https://youtu.be/G7PgIQWiU3M

At the bottom right of the interface are buttons for "Video Manager" and "+ Add more videos".

<https://www.youtube.com/watch?v=G7PgIQWiU3M&list=PLjPfp4Ph3gBo5SmWJXwv4oKDfeTXA7xgw>

# Updating to New Version

# Update to New Version

## Updating to New Releases

EDIT

Create React App is divided into two packages:

- `create-react-app` is a global command-line utility that you use to create new projects.
- `react-scripts` is a development dependency in the generated projects (including this one).

You almost never need to update `create-react-app` itself: it delegates all the setup to `react-scripts`.

es

When you run `create-react-app`, it always creates the project with the latest version of `react-scripts` so you'll get all the new features and improvements in newly created apps automatically.

<https://facebook.github.io/create-react-app/docs/updating-to-new-releases>

# Recommended Libraries

# Recommended VSCode Extensions

ES7 React/Redux/GraphQL/React-Native snippets

Javascript ES6 Snippet

Auto-Close Tag

Html to JSX

Prettier - Code Formatter

Color Picker

Color Highlight

Auto Import

VSCode-Icon



JavaScript (ES6) code snippets

charalampos karypidis | 3,412,458 | ★★★★☆

Code snippets for JavaScript in ES6 syntax

Install



ES7 React/Redux/GraphQL

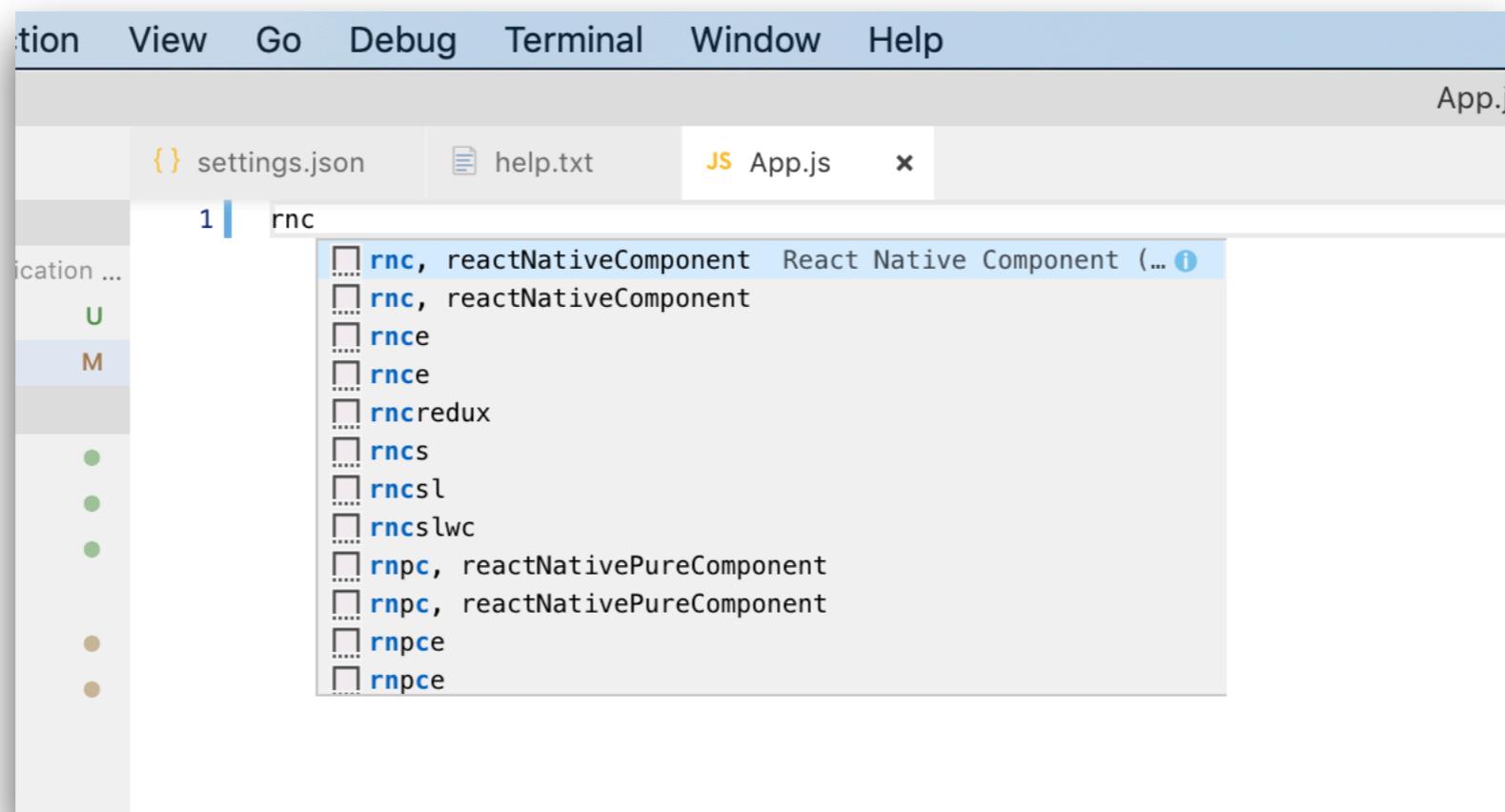
dsznajder | 1,334,100 | ★★★★★

Simple extensions for React, Redux, GraphQL

Disable ▾ Uninstall

# React Snippet

## VSCode Extension



# React Snippet

## Emmet Script

```
"emmet.syntaxProfiles": {  
    "javascript": "jsx"  
},  
"emmet.includeLanguages": {  
    "javascript": "javascriptreact"  
},  
"emmet.triggerExpansionOnTab": true
```

A screenshot of the VS Code settings.json editor. The code shows configurations for Emmet syntax profiles, including 'javascript' set to 'jsx' and 'javascript' set to 'javascriptreact'. Other configurations include 'triggerExpansionOnTab' set to true.

A screenshot of VS Code showing an Emmet abbreviation dropdown menu. The menu lists various abbreviations such as 'div.content-wrap', 'wrap', 'WebGLShaderPrecisionFormat', 'msWriteProfilerMark', 'webkitConvertPointFromPageToNode', 'WSAEPROTOTYPE', 'WSAVERNOTSUPPORTED', 'WSAENOPROTOOPT', 'WSAEPROTONOSUPPORT', and 'webkitConvertPointFromNodeToPage'. The 'div.content-wrap' abbreviation is currently selected.

# Component Generator

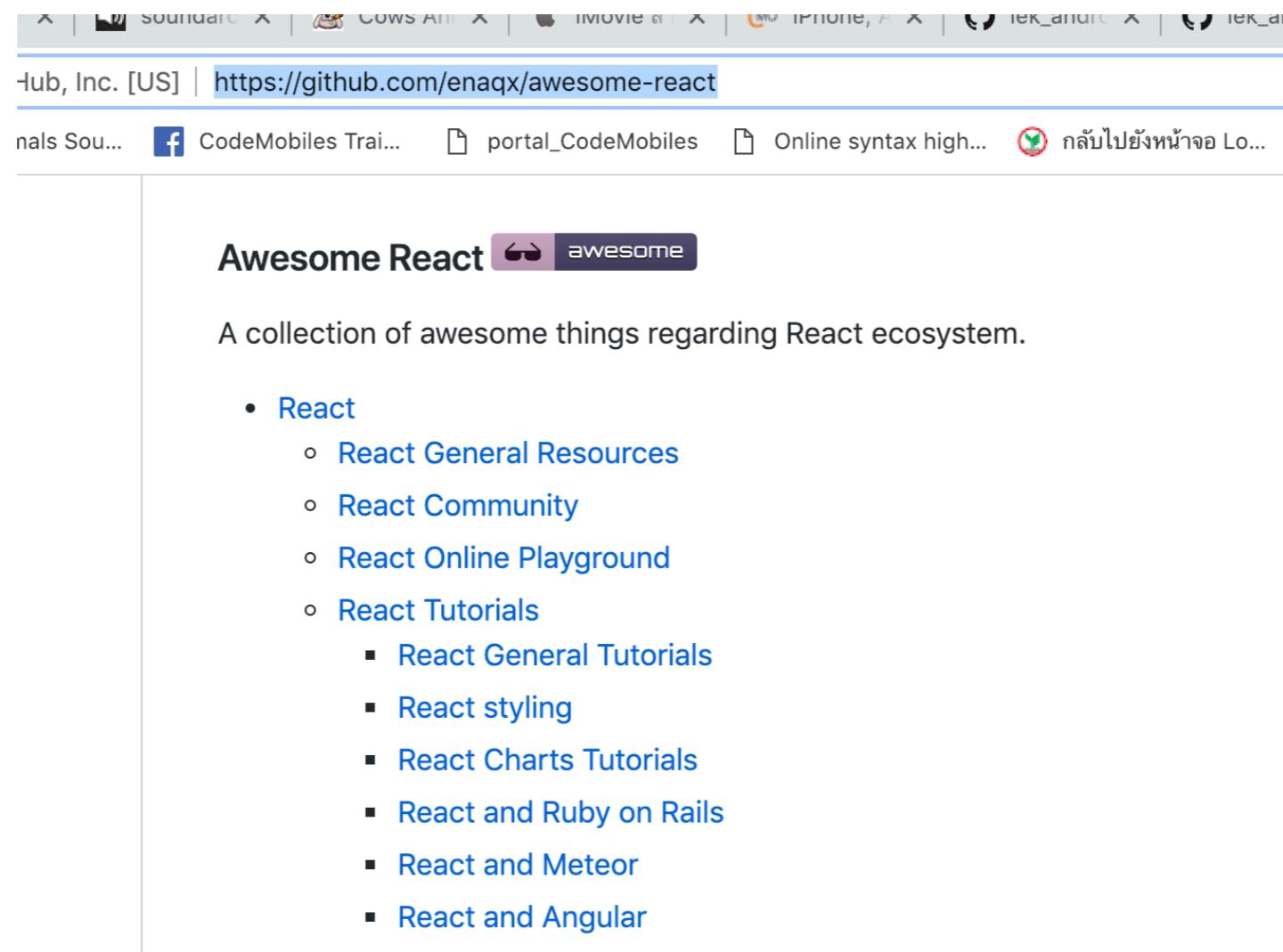
```
npm install --save-dev create-react-component-folder
```

A terminal window with a green header bar containing three colored dots (red, yellow, green). The main area shows the command \$ npx crcf components/myComponent followed by a backslash and the text 'Creating components files...'. Below this, it says 'Created new React components at:' followed by the path '/MyProject/myComponent' and a file tree: index.js, myComponent.js, myComponent.test.js, and myComponent.css. At the bottom, there is a star icon followed by 'Finished in: 118.836ms', and the word 'DONE Success!'.

```
$ npx crcf components/myComponent
\ Creating components files...
Created new React components at:
/MyProject/myComponent
  └── index.js
  └── myComponent.js
  └── myComponent.test.js
  └── myComponent.css

★ Finished in: 118.836ms
DONE Success!
```

# Awesome React Library



The screenshot shows a browser window with multiple tabs open at the top. The active tab is titled "Hub, Inc. [US] | https://github.com/enaqx/awesome-react". Below the tabs, there are several other open tabs with titles like "CodeMobiles Trai...", "portal\_CodeMobiles", "Online syntax high...", and "กลับไปยังหน้าจอ Lo...". The main content area displays the "Awesome React" page. The title "Awesome React" is at the top, followed by a subtitle "A collection of awesome things regarding React ecosystem." A bulleted list under "React" includes: "React General Resources", "React Community", "React Online Playground", "React Tutorials" (with sub-points: "React General Tutorials", "React styling", "React Charts Tutorials", "React and Ruby on Rails", "React and Meteor", "React and Angular").

<https://github.com/enaqx/awesome-react>

# Programming Languages

# ES6 vs JSX

**ES6** : Modern Javascript

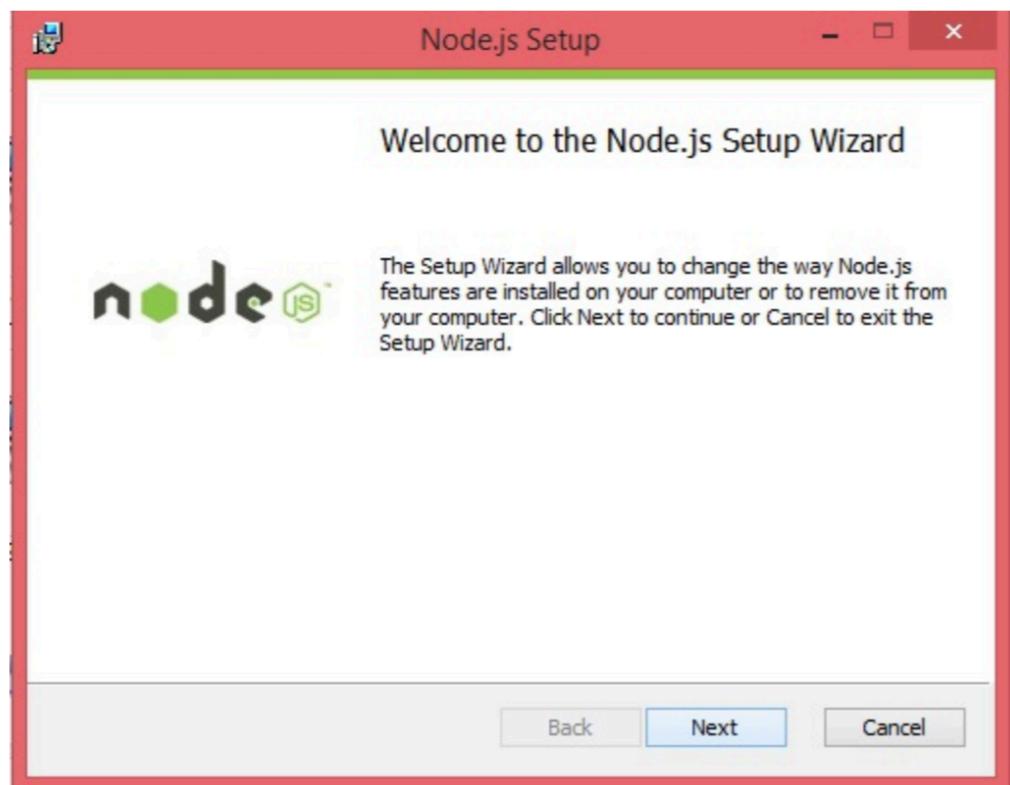
**JSX** : Template Markup XML that can convert to ES6

# Javascript ES6

## Language for React

# ES6

# Development Environment



# ES6 Examples

```
class Test {  
    constructor() {  
        this.name = 'Test constructor';  
    }  
  
    method() {  
        const func1 = () => {  
            console.log(this); //points to instance of Test class  
        }  
        func1();  
  
        function func2() {  
            console.log(this); //undefined  
        }  
        func2();  
    }  
}  
  
const test = new Test();  
test.method()
```

# ES6

## Use Strict

**"use strict";** Defines that JavaScript code should be executed in "strict mode".  
Ex: Un-declaring var, let or const will cause an error

### Example

```
"use strict";
x = 3.14;          // This will cause an error because x is not declared
```

# ES6

## Var, Let and Const

- **Var** is old version of variable declaration with (scope problem)
- **Let** is a new version of var allowing scope the variable
- **const** is just an immutable

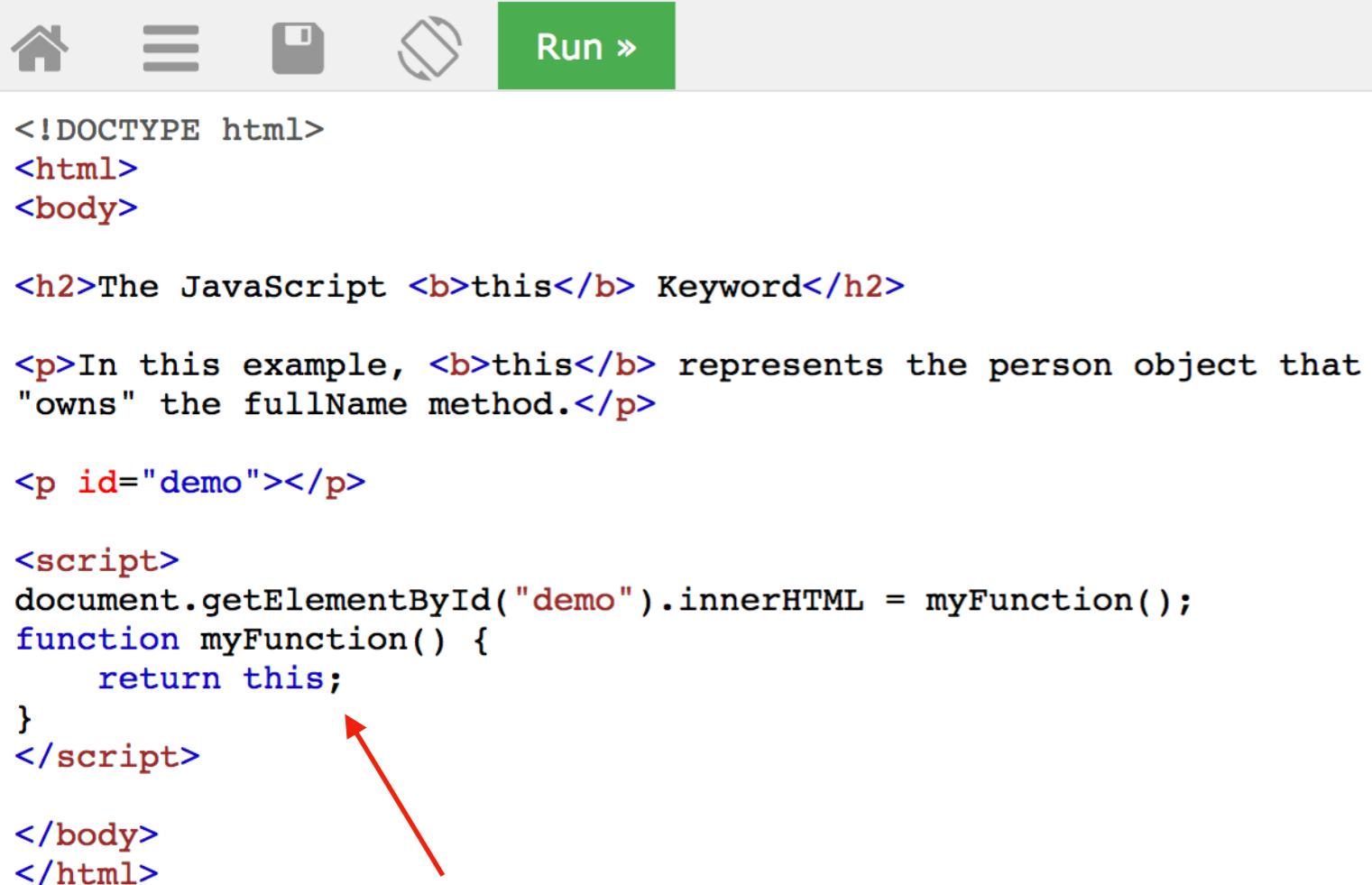
```
'use strict';
let userid = 10;
{
  let userid = 200;
}
console.log(userid);
// 10
```

```
'use strict';
var userid = 10;
{
  var userid = 200;
}
console.log(userid);
// 200
```

```
'use strict';
function updateUserId() {
  userid = 1234;
}
let userid = null;
updateUserId();
console.log(userid);
// 1234
```

# ES6

## Use **this** keyword without “use strict”



The screenshot shows a browser's developer tools interface with a code editor. The toolbar at the top includes icons for home, menu, save, and run, followed by a green "Run >" button. The code editor contains the following HTML and JavaScript:

```
<!DOCTYPE html>
<html>
<body>

<h2>The JavaScript <b>this</b> Keyword</h2>

<p>In this example, <b>this</b> represents the person object that "owns" the fullName method.</p>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = myFunction();
function myFunction() {
    return this;
}
</script>

</body>
</html>
```

A red arrow points from the word "this" in the `return this;` line to the `this` keyword in the `<b>this</b>` line above it.

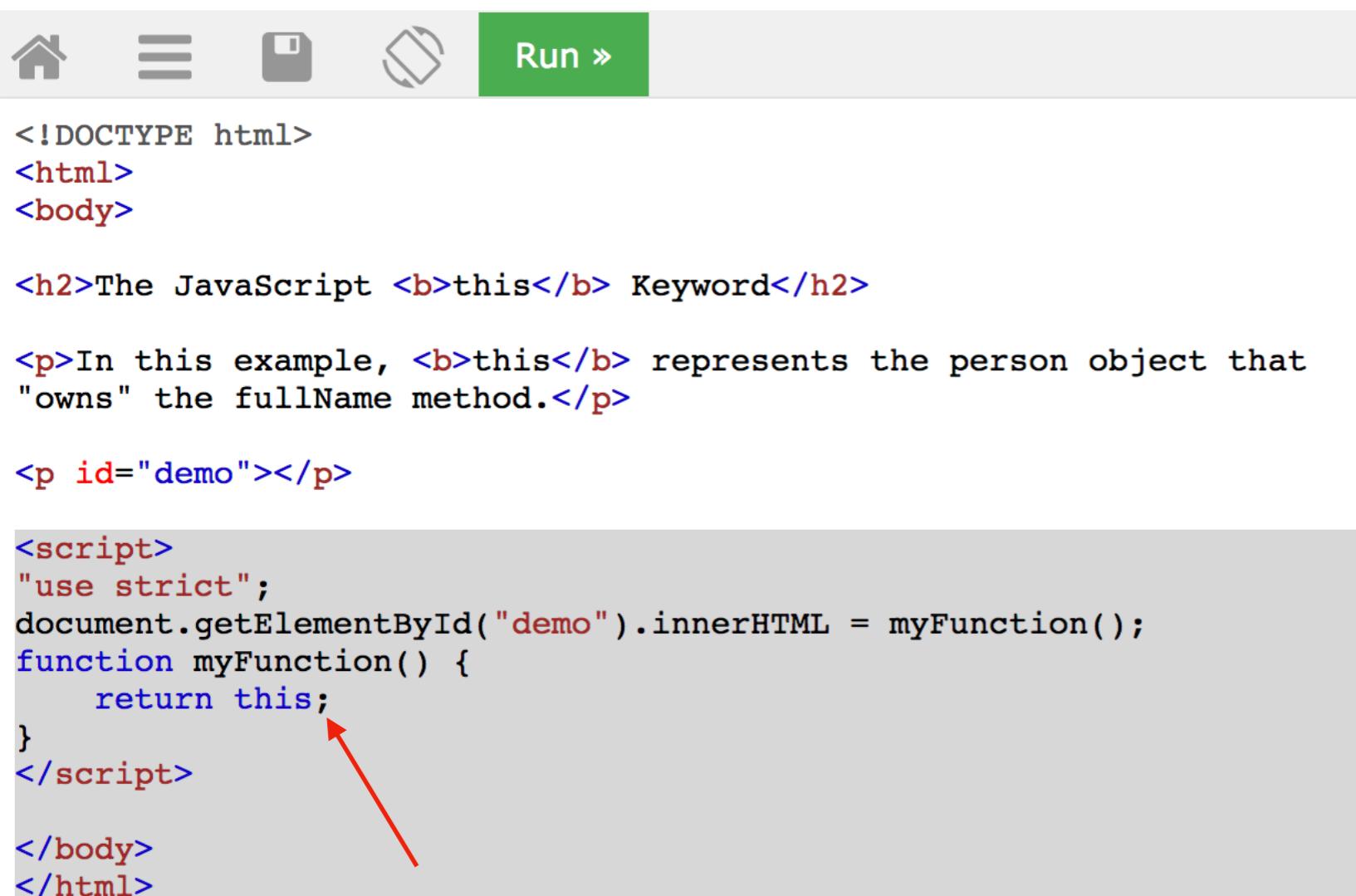
### The JavaScript **this** Keyword

In this example, **this** represents the person object that "owns" the [object Window]



# ES6

## Use **this** keyword with “use strict”



The screenshot shows a browser's developer tools code editor. At the top, there are icons for home, menu, save, and run, followed by a green "Run »" button. The code editor contains the following HTML and JavaScript:

```
<!DOCTYPE html>
<html>
<body>

<h2>The JavaScript <b>this</b> Keyword</h2>

<p>In this example, <b>this</b> represents the person object that "owns" the fullName method.</p>

<p id="demo"></p>

<script>
"use strict";
document.getElementById("demo").innerHTML = myFunction();
function myFunction() {
    return this;
}
</script>

</body>
</html>
```

A red arrow points from the word "this" in the "return this;" line of the script block to the word "this" in the explanatory paragraph above it.

### The JavaScript **this** Keyword

In this example, **this** represents the person object that "owns"

undefined



# ES6

# Export and Import

## // – Single.js

```
export const sqrt = Math.sqrt;  
  
export function square(x) { return x*y; }  
  
export function dialog(x,y) { return ...; }
```

## // – Multiple.js

```
export const sqrt = Math.sqrt;  
  
function square(x) { return x*y; }  
  
function dialog(x,y) { return ...; }  
  
export {square, dialog}
```

## // – main.js

```
import sqrt, square, dialog from 'single.js'  
  
import sqrt, {square, dialog } 'multiple.js'
```

# ES6

## Multiple named export

### // – lib.js

```
export const sqrt = Math.sqrt;  
export function square(x) { return x*y; }  
export function dialog(x,y) { return ...; }
```

### // – main.js

```
import {square, dialog} from 'lib';  
console.log(square(11));  
console.log(dialog(4,3));
```

### // – main.js

```
import * as lib from 'lib'  
console.log(lib.square(11));  
console.log(lib.dialog(4,3));
```

# ES6

# Single Default Export

// – myFunc.js

```
export default function() { ... }
```

// – main.js

```
import myFunc from 'myFunc.js'  
myFunc();
```

---

// – myClass.js

```
export default class { ... }
```

// – main.js

```
import MyClass from 'myClass.js';  
const inst = new MyClass();
```

# ES6 Export

## // – utility.js

```
function generateRandom() { return .... }

function sum (a, b) { return .... }

export { generateRandom, sum }
```

## // – app.js

```
import { generateRandom, sum } from 'utility.js';

console.log(generateRandom());

console.log(sum(1,2));
```

# ES6

## Spread syntax [...x]

```
function run(x,y,z) {}
```

```
var args = [1,2,3]
```

```
run(...args);
```

```
var color1 = ['red', 'yellow'];
```

```
var allColor = ['green', ...color1, 'orange']
```

```
var dateFields = [1970, 0, 1];
```

```
var d = new Date(...dateFields);
```

# ES6

## For-of Iterator

```
var list = [3, 5, 7];
list.foo = 'bar';

for (var key in list) {
  console.log(key); // 0, 1, 2, foo
}

for (var value of list) {
  console.log(value); // 3, 5, 7
}

var string = 'codemobiles';
for (var chr of string) {
  console.log(chr); // c, o, d, e, ...
}
```

# ES6

## use .map with Array

```
var numbers = [1, 2, 3, 4];
var doubled = numbers.map(number => number * 2);
```

# ES6 Clone

```
"use strict"  
var det = { name:"Tom", ID:"E1001" };  
  
var copy = Object.assign({}, det);  
  
console.log(copy);  
for (let val in copy) {  
    console.log(copy[val])  
}
```

# ES6

# Promise

```
function getSum(n1, n2) {
    var isAnyNegative = function() {
        return n1 < 0 || n2 < 0;
    }
    var promise = new Promise(function(resolve, reject) {
        if (isAnyNegative()) {
            reject(Error("Negatives not supported"));
        }
        resolve(n1 + n2)
    });
    return promise;
}
```

The second step details the implementation of the caller (STEP 2).

The caller should use the 'then' method, which takes two callback methods - first for success and second for failure. Each method takes one parameter, as shown in the following code.

```
getSum(5, 6)
  .then(function (result) {
    console.log(result);
  },
  function (error) {
    console.log(error);
});
```

The following output is displayed on successful execution of the above code.

```
11
```

# ES6

# async / await

```
(async function() {  
    Defines an asynchronous  
    function  
  
    let a = await fetch('/')  
        ① Fetches Medium's home  
        ② Waits until home is fetched  
  
    console.log(await a.text())  
        ③ Reads stream  
        ④ Waits until stream is read  
})()
```

# JSX

# What is JSX?

React uses JSX for **templating** instead of regular JavaScript. It is not necessary to use it, however, following are some pros that come with it.

```
<MyButton color="blue" shadowSize={2}>  
  Click Me  
</MyButton>
```

compiles into:

```
React.createElement(  
  MyButton,  
  {color: 'blue', shadowSize: 2},  
  'Click Me'  
)
```

# JSX to Javascript

```
<div className="red">Children Text</div>;
<MyCounter count={3 + 5} />

// Here, we set the "scores" attribute below to a JavaScript object.
var gameScores = {
  player1: 2,
  player2: 5
};

<DashboardUnit data-index="2">
  <h1>Scores</h1>
  <Scoreboard className="results" scores={gameScores} />
</DashboardUnit>;
```

JSX will be converted into React code below.

```
React.createElement("div", { className: "red" }, "Children Text");
React.createElement(MyCounter, { count: 3 + 5 });

React.createElement(
  DashboardUnit,
  { "data-index": "2" },
  React.createElement("h1", null, "Scores"),
  React.createElement(Scoreboard, { className: "results", scores: gameScores })
);
```

# JSX Example

## Hello World

### App.jsx

```
import React from 'react';

class App extends React.Component {
  render() {
    return (
      <div>
        Hello World!!!
      </div>
    );
  }
}
export default App;
```

# JSX Example

## Nested Elements

### App.jsx

```
import React from 'react';

class App extends React.Component {
  render() {
    return (
      <div>
        <h1>Header</h1>
        <h2>Content</h2>
        <p>This is the content!!!</p>
      </div>
    );
  }
}
export default App;
```

# JSX Example

## Javascript Expressions in JSX

### App.jsx

```
import React from 'react';

class App extends React.Component {
  render() {
    return (
      <div>
        <h1>{1+1}</h1>
      </div>
    );
  }
}
export default App;
```

# JSX Example

## Conditional Expression

**If-Else is NOT Allowed in JSX**

# JSX Example

## Conditional Expression

### App.jsx

```
import React from 'react';

class App extends React.Component {
  render() {
    var i = 1;
    return (
      <div>
        <h1>{i == 1 ? 'True!' : 'False'}</h1>
      </div>
    );
  }
}
export default App;
```



# JSX Example

## Conditional Expression

### Inline If with Logical && Operator

You may embed any expressions in JSX by wrapping them in curly braces. This includes the JavaScript logical `&&` operator. It can be handy for conditionally including an element:

```
function Mailbox(props) {
  const unreadMessages = props.unreadMessages;
  return (
    <div>
      <h1>Hello!</h1>
      {unreadMessages.length > 0 &&
        <h2>
          You have {unreadMessages.length} unread messages.
        </h2>
      }
    </div>
  );
}

const messages = ['React', 'Re: React', 'Re:Re: React'];
ReactDOM.render(
  <Mailbox unreadMessages={messages} />,
  document.getElementById('root')
);
```

# JSX Example

## Conditional Expression

### Inline If-Else with Conditional Operator

Another method for conditionally rendering elements inline is to use the JavaScript conditional operator `condition ? true : false`.

In the example below, we use it to conditionally render a small block of text.

```
render() {
  const isLoggedIn = this.state.isLoggedIn;
  return (
    <div>
      The user is <b>{isLoggedIn ? 'currently' : 'not'}</b> logged in.
    </div>
  );
}
```

# JSX Example

## Styling

### App.jsx

```
import React from 'react';

class App extends React.Component {
  render() {
    var myStyle = {
      fontSize: 100,
      color: '#FF0000'
    }
    return (
      <div>
        <h1 style = {myStyle}>Header</h1>
      </div>
    );
  }
}
export default App;

/*
  React recommends using inline styles. React will also automatically append px after the number value
  of specific elements.
*/
```



# JSX Example

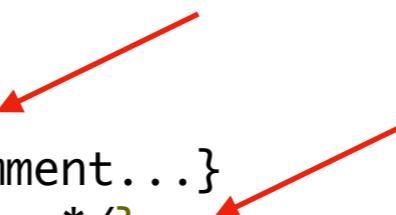
## Comments

### App.jsx

```
import React from 'react';

class App extends React.Component {
  render() {
    return (
      <div>
        <h1>Header</h1>
        { //End of the line Comment...
          /*Multi line comment...*/
        }
      </div>
    );
  }
}

export default App;
```



# JSX Tutorial

<http://buildwithreact.com/tutorial/jsx>

## Exercise: JSX

Try to match the markup of the box contents. I recommend referring to the HTML tab and/or inspecting the DOM.

The screenshot shows a JS Bin editor interface. The top navigation bar includes tabs for JS Bin, Save, HTML, CSS, JSX (React), Console, Output, and Help. The CSS tab is active, displaying the following CSS code:

```
.box {  
  border: 1px solid #ccc;  
  padding: 10px;  
}  
  
.button {  
  background-color: #174E7D;  
  border-radius: 5px;  
  color: #fff;  
  display: inline-block;  
  margin-bottom: 5px;  
  padding: 5px 15px;  
  text-decoration: none;  
}
```

The JSX (React) tab is also visible. Below the CSS code, there is a large text area containing the heading "Match This" and a button labeled "Button". The button has a blue background, white text, and a rounded rectangular shape. To the right of the button is a placeholder text: "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer nec odio. Praesent libero." At the bottom of the page, there is another text area labeled "Your Implementation" with the placeholder text "// Your work here". At the very bottom right, there are two buttons: "Auto-run JS" with a checked checkbox and "Run with JS".

# Nested Elements

```
import React from 'react';

class App extends React.Component {
  render() {
    return (
      <div>
        <h1>Header</h1>
        <h2>Content</h2>
        <p>This is the content!!!</p>
      </div>
    );
  }
}

export default App;
```

# Ternary Condition

? :

```
import React from 'react';

class App extends React.Component {
  render() {
    var i = 1;
    return (
      <div>
        <h1>{i == 1 ? 'True!' : 'False'}</h1>
      </div>
    );
  }
  export default App;
}
```

# Styling

```
import React from 'react';

class App extends React.Component {
  render() {
    var myStyle = {
      fontSize: 100,
      color: '#FF0000'
    }
    return (
      <div>
        <h1 style = {myStyle}>Header</h1>
      </div>
    );
  }
  export default App;
}
```

# Comment

```
import React from 'react';

class App extends React.Component {
  render() {
    return (
      <div>
        <h1>Header</h1>
        { //End of the line Comment... }
        { /*Multi line comment... */ }
      </div>
    );
  }
}

export default App;
```

# JSX Tools

# Covert HTML to JSX

 **html to JSX** riazxrazor.html-to-jsx

Riaz Laskar | ⚡ 4,358 | ★★★★★ | Repository | License

Converts html code to React JSX

[Disable ▾](#) [Uninstall](#)

[Details](#) [Contributions](#) [Changelog](#)

---

## html-to-jsx

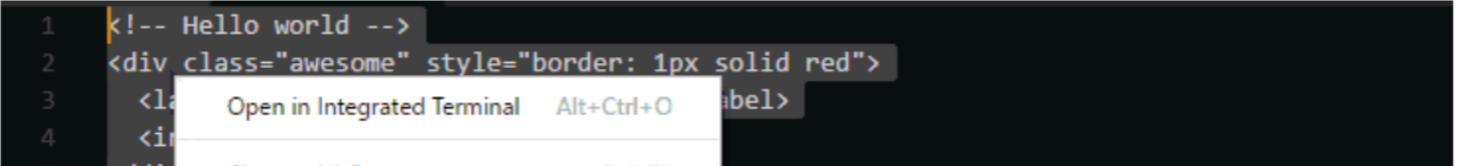
A basic Visual Studio Code plugin that converts html text to JSX string

You can select your html text and use the shortcut **ctrl+alt+x**

**There are many things to do**

- Add functionality to validate html code
- Add support to format the resulting string
- Add support to convert javascript strings to html code

[Change log](#)



# JSX Formatter



**Prettier - Code formatter** esbenp.prettier-vscode

Esben Petersen | ⚡ 7,296,143 | ★★★★☆ | [Repository](#) | [License](#)

VS Code plugin for prettier/prettier

[Disable ▾](#) [Uninstall](#)

[Details](#) [Contributions](#) [Changelog](#)

---

## Prettier formatter for Visual Studio Code

VS Code package to format your JavaScript / TypeScript / CSS using [Prettier](#).

### Installation

Install through VS Code extensions. Search for **Prettier – Code formatter**

[Visual Studio Code Market Place: Prettier - Code formatter](#)

Can also be installed in VS Code: Launch VS Code Quick Open (Ctrl+P), paste the following command, and press enter.

```
ext install esbenp.prettier-vscode
```

# React JS Basic

# React Keywords

- **Component:** React is all about components. Everything is a component such as Button, Div, Image and etc. - And you can create your own components.
- **JSX:** is JavaScript syntax extension. It isn't necessary to use JSX in React development, but it is recommended.
- **Unidirectional data flow and Flux** – React implements one-way data flow which makes it easy to reason about your app. Flux is a pattern that helps keeping your data unidirectional.

# Core React Libraries

- **React :**
- **React Dom**

```
1  {
2    "name": "my-app",
3    "version": "0.1.0",
4    "private": true,
5    "dependencies": {
6      "react": "^16.7.0",
7      "react-dom": "^16.7.0",
8      "react-scripts": "2.1.2"
```

# React vs ReactDOM

**React and ReactDOM** were only recently split into two different libraries. Prior to v0.14, all ReactDOM functionality was part of React. This may be a source of confusion, since any slightly dated documentation won't mention the React / ReactDOM distinction.

As the name implies, ReactDOM is the glue between React and the DOM.

# Component stylesheets

```
const styles = StyleSheet.create({
  container: {
    flex: 1,
    flexDirection: 'column',
    backgroundColor: "#FFFFFF",
    alignItems: 'flex-start',
    justifyContent: 'flex-start'
  },
  myText: {
    width: 300,
    height: 300,
    backgroundColor: "#D0D0D0"
  },
  item: {
    fontSize: 30,
    fontWeight: 'bold',
    color: "#00FF00"
  }
})
```

```
render() {
  return (
    <View style={styles.container}>
      <View style={styles.item} />
      <View style={styles.item} />
      <View style={styles.item} />
      <Text style={styles.myText} >Hello</Text>
    </View>
  );
}
```

# CSS tips

- **Inline style**

```
<view style={{backgroundColor:'#ff0000'}}>  
    ggggg  
</view>
```

- **Inline style + Normal style**

```
<view  
    style={[styles.container, {color: 'white', backgroundColor: '#333'}]} > ...  
</view>
```

- **Two Styles**

```
<view style={[ styles.container , styles.style1 ]}> ...  
</view>
```

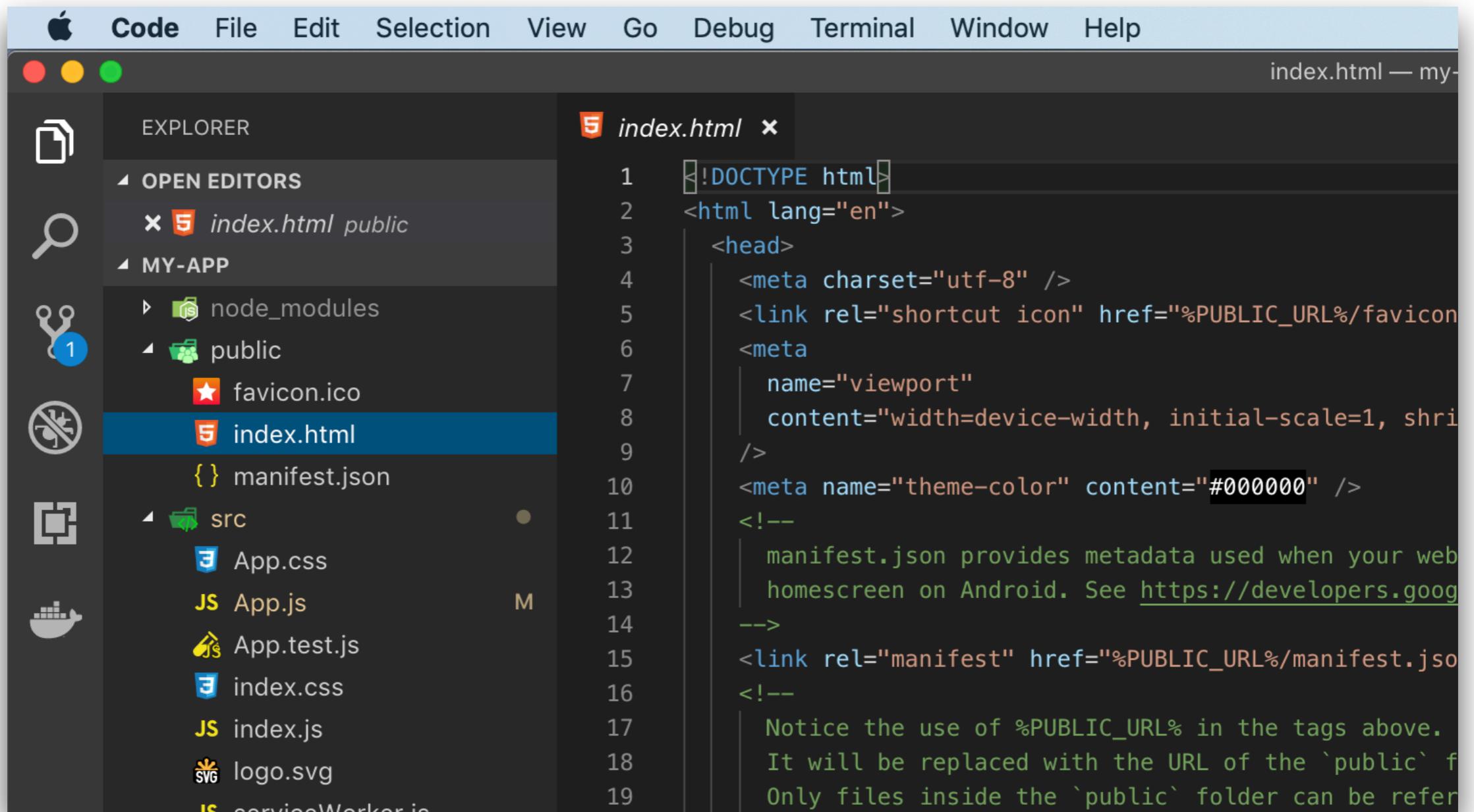
- **Style depend on platform**

```
{marginTop: Platform.OS == 'android' ? 22 : 0}
```

# React Project Structure

# Index.html

## Entry Point

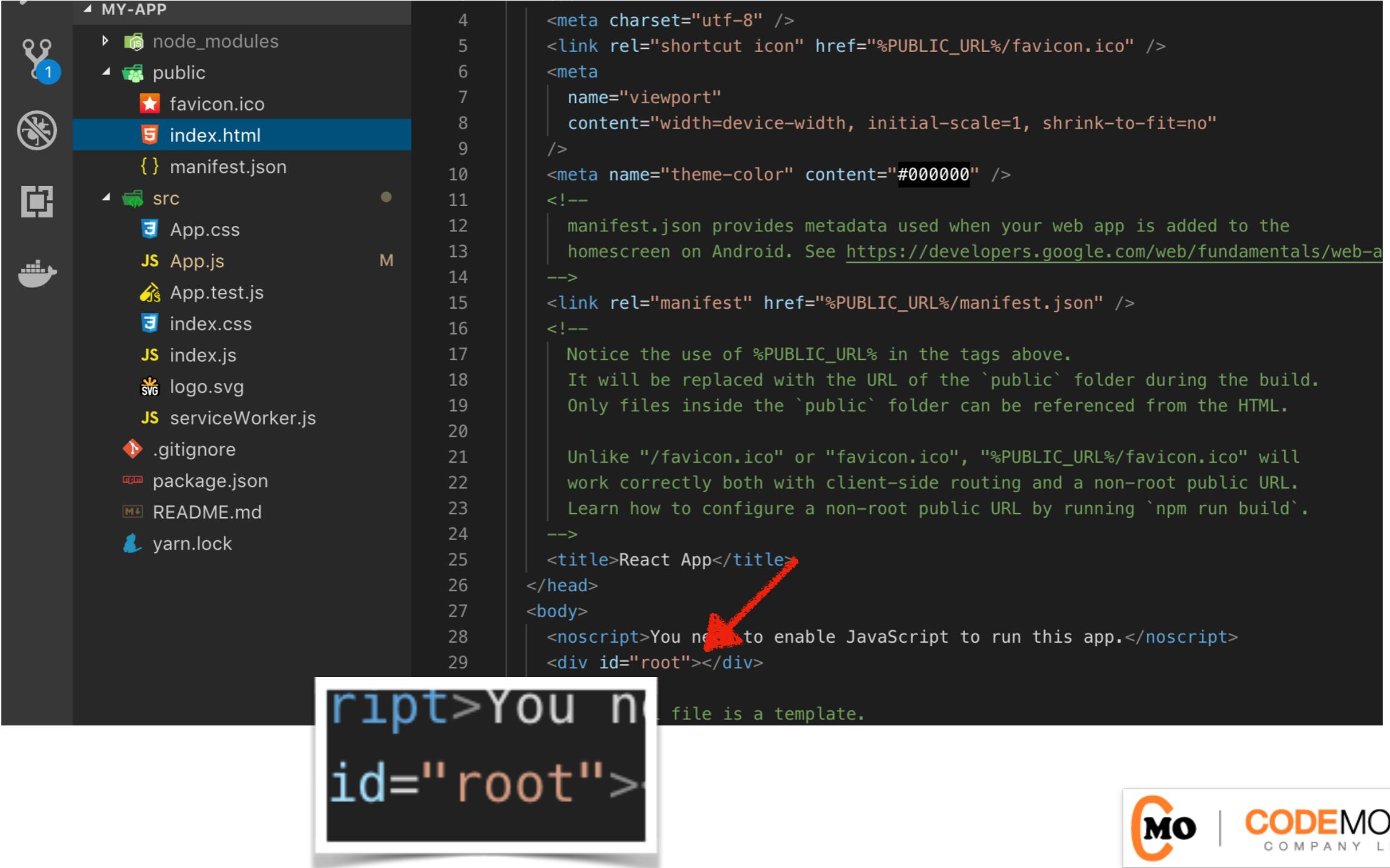


The screenshot shows the Visual Studio Code interface with the following details:

- File Menu:** Code, File, Edit, Selection, View, Go, Debug, Terminal, Window, Help.
- Title Bar:** index.html — my-
- Explorer:** Shows the project structure:
  - OPEN EDITORS: index.html (public)
  - MY-APP
    - node\_modules
    - public
      - favicon.ico
      - index.html
    - manifest.json
    - src
      - App.css
      - App.js
      - App.test.js
      - index.css
      - index.js
      - logo.svg
      - serviceWorker.js
- Editor:** The index.html file is open, showing its code. The code includes DOCTYPE html, html lang="en" tags, head, meta charset="utf-8", link rel="shortcut icon", and meta viewport and theme-color tags. It also includes comments about manifest.json and a link to it. The code uses %PUBLIC\_URL% placeholders.

# Index.html

**Root ID <--> Index.JS (React Part)**



The screenshot shows a code editor with the following file structure:

- MY-APP
  - node\_modules
  - public
    - favicon.ico
    - index.html
  - src
    - App.css
    - App.js
    - App.test.js
    - index.css
    - index.js
    - logo.svg
    - serviceWorker.js
  - .gitignore
  - package.json
  - README.md
  - yarn.lock

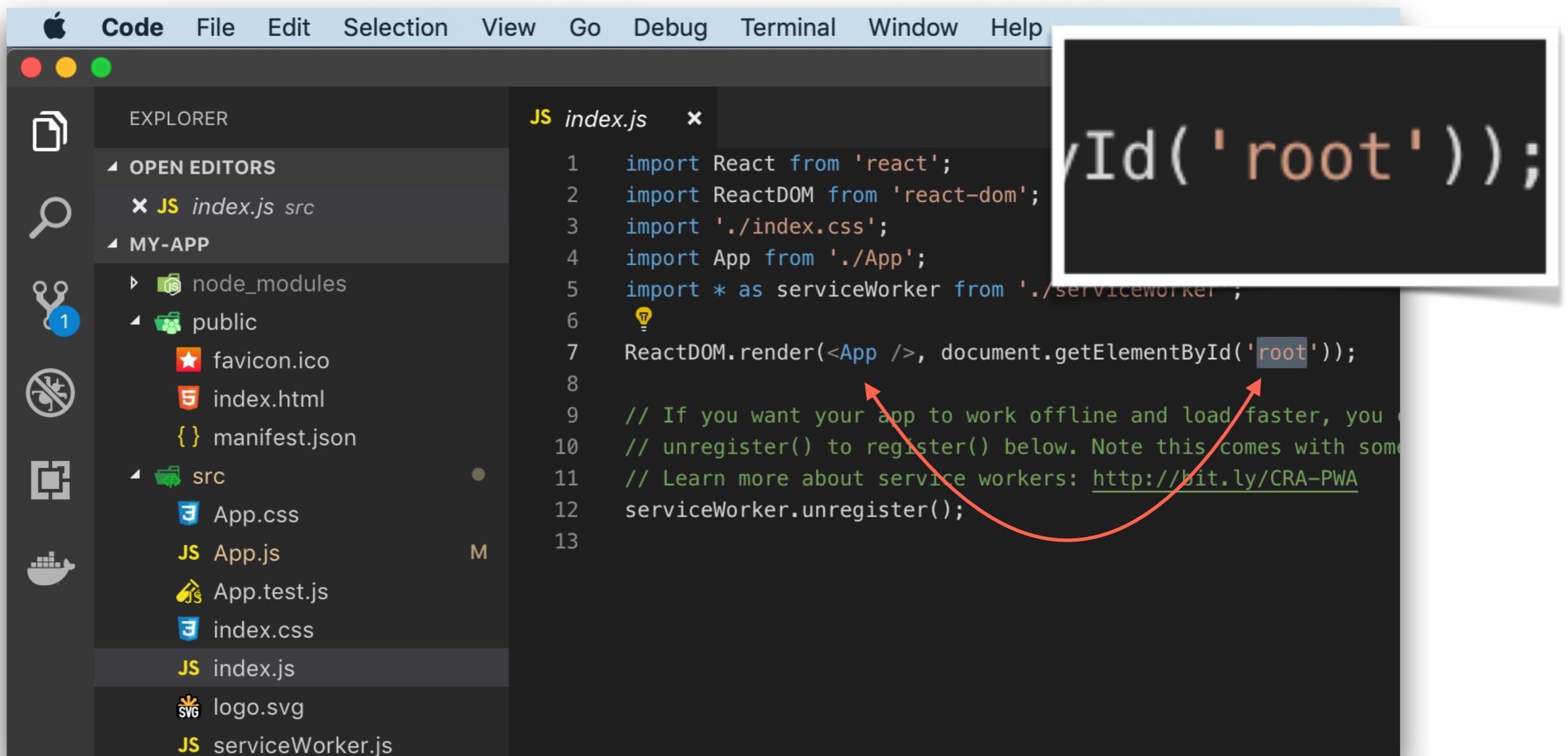
The 'index.html' file is selected in the sidebar. The code content is as follows:

```
4   <meta charset="utf-8" />
5   <link rel="shortcut icon" href="%PUBLIC_URL%/favicon.ico" />
6   <meta
7     name="viewport"
8     content="width=device-width, initial-scale=1, shrink-to-fit=no"
9   />
10  <meta name="theme-color" content="#000000" />
11  <!--
12    manifest.json provides metadata used when your web app is added to the
13    homescreen on Android. See https://developers.google.com/web/fundamentals/web-a
14  -->
15  <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
16  <!--
17    Notice the use of %PUBLIC_URL% in the tags above.
18    It will be replaced with the URL of the `public` folder during the build.
19    Only files inside the `public` folder can be referenced from the HTML.
20
21    Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
22    work correctly both with client-side routing and a non-root public URL.
23    Learn how to configure a non-root public URL by running `npm run build`.
24  -->
25  <title>React App</title>
26  </head>
27  <body>
28    <noscript>You need to enable JavaScript to run this app.</noscript>
29    <div id="root"></div>
```

A red arrow points to the `<noscript>You need to enable JavaScript to run this app.</noscript>` line.

# Index.js

## Inject App Component via root ID



The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows a project structure with files like `index.html`, `manifest.json`, `src/App.css`, `src/App.js`, `src/App.test.js`, `src/index.css`, `src/index.js`, `src/logo.svg`, and `serviceWorker.js`.
- Code Editor:** The file `index.js` is open, displaying the following code:

```
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import './index.css';
4 import App from './App';
5 import * as serviceWorker from './serviceworker';
6 
7 ReactDOM.render(<App />, document.getElementById('root'));
8 
9 // If you want your app to work offline and load faster, you can
10 // unregister() to register() below. Note this comes with some
11 // Learn more about service workers: http://bit.ly/CRA-PWA
12 serviceWorker.unregister();
13 
```

A red callout box highlights the `document.getElementById('root')` line.

# Index.js

## Inject App Component via root ID

JS index.js 

```
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import './index.css';
4 import App from './App';
5 import * as serviceWorker from './serviceWorker';
6 
7 ReactDOM.render(<App />, document.getElementById('root'));
8
9 // If you want your app to work offline and load faster, you c
```

# Index.js

## Inject App Component via root ID

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';

class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <p>
            Revise <code>src/App.js</code> and save to reload
          </p>
          <a
            className="App-link"
            href="https://reactjs.org"
            target="_blank"
            rel="noopener noreferrer"
          >
            Learn React
          </a>
        </header>
      </div>
    );
  }
}

export default App;
```

# App.js

## It is the Root Component

The screenshot shows a code editor interface with the following details:

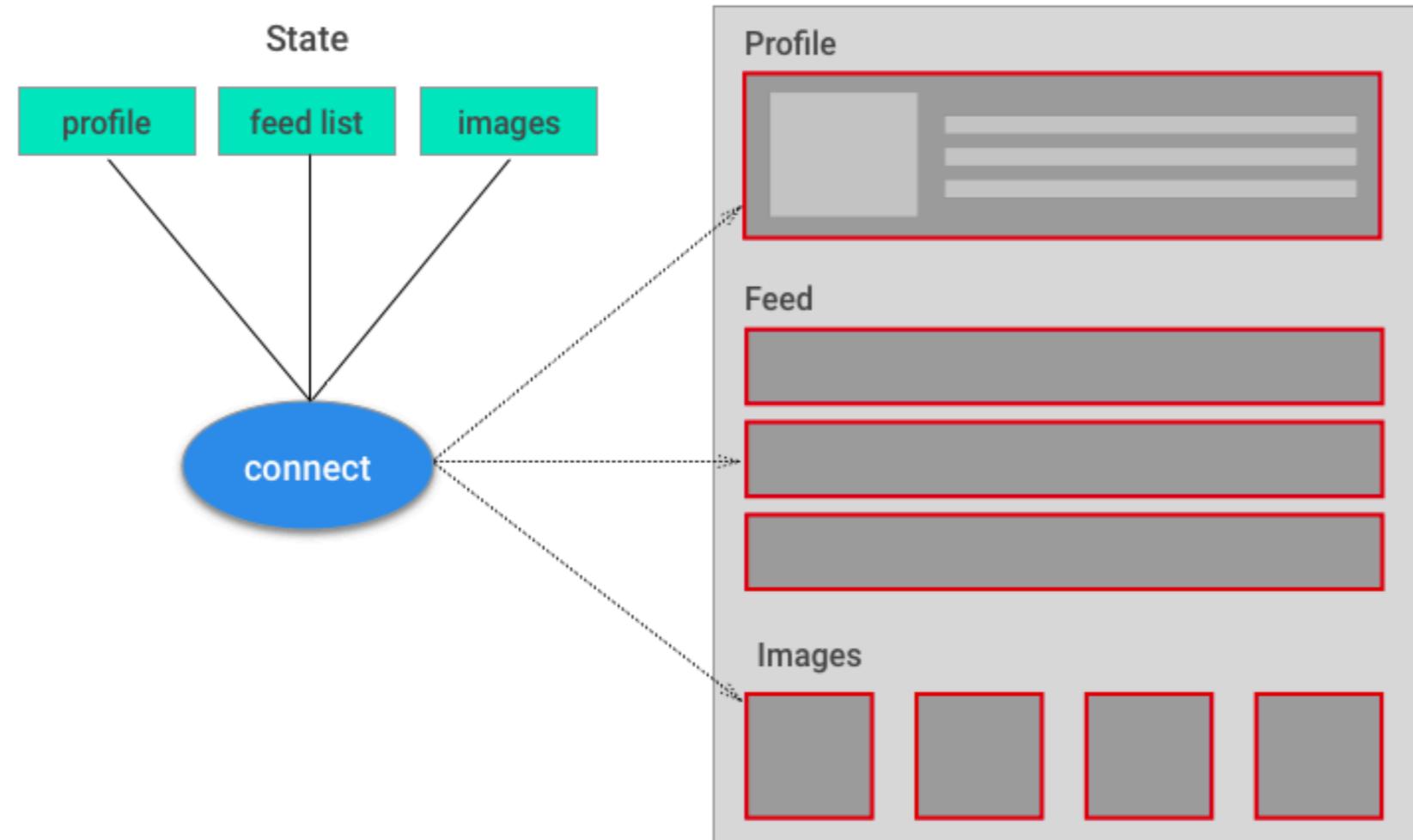
- Editor Area:** Displays the code for `App.js`. The code imports React, logo from `./logo.svg`, and styles from `./App.css`. It defines a class `App` extending `Component`, which renders a header with a logo and a descriptive paragraph.
- Toolbar:** Shows standard menu items: Go, Debug, Terminal, Window, Help.
- File Explorer:** On the left, it shows the project structure under `MY-APP`:
  - `node_modules`
  - `public`: Contains `favicon.ico` and `index.html`.
  - `manifest.json`
  - `src`: Contains `App.css` and the current file `App.js`.
- Status Bar:** Shows file names and modification status (e.g., `M`).

# Component

# What is component?

connect @ root Component and the *profile* state changes.

This triggers a **render** on ALL components.



# Types of Components

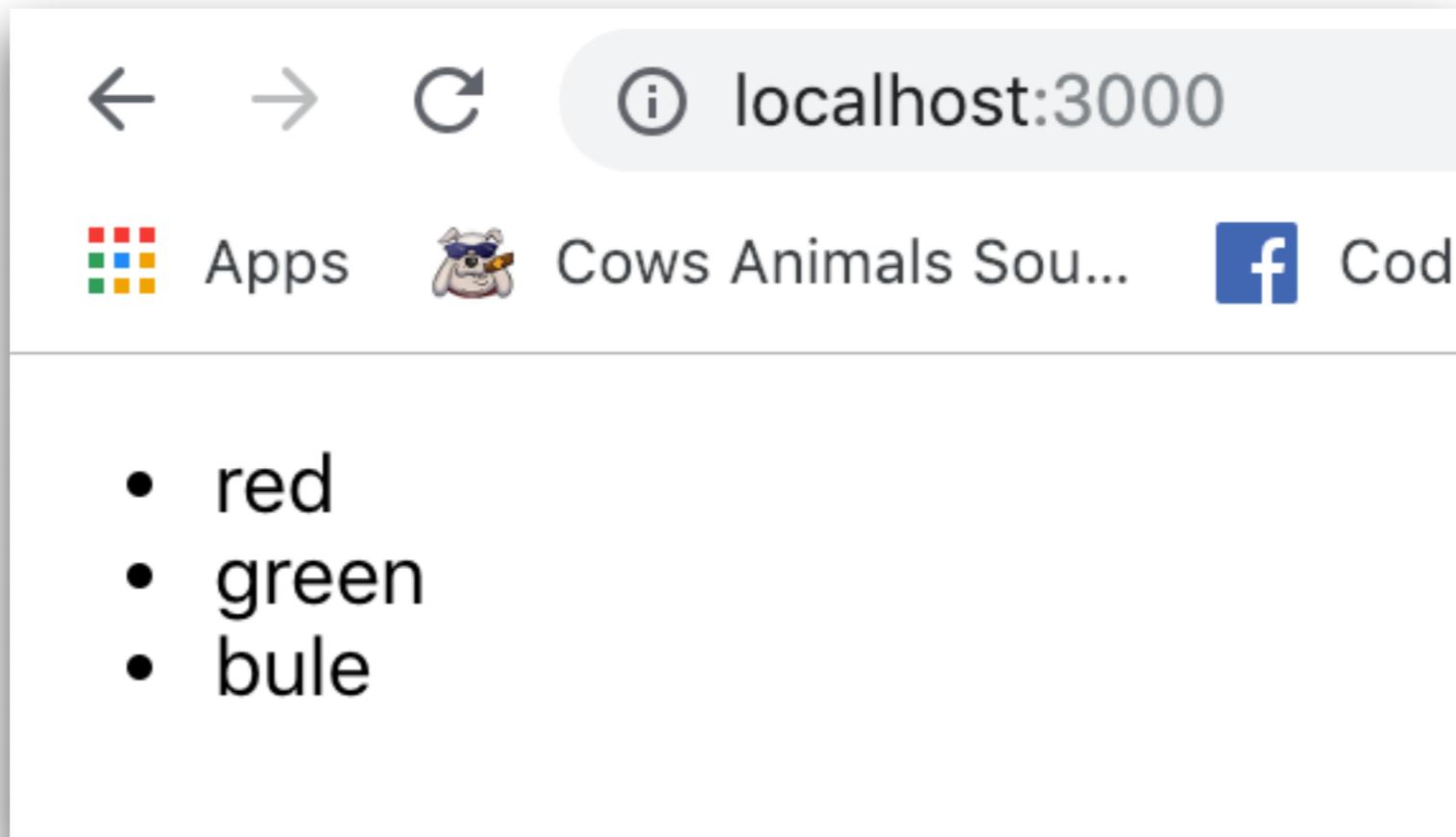
- Class Component (Stateful)
- Functional Component (Stateless)
- Functional Component (Stateful with ReactHook)
- Functional Component (HOC) - Higher Order Component

# Component

## Stateful

```
export default class App extends Component {  
  constructor(props){  
    super(props)  
    this.state = {data: ["red", "green", "blue"] }  
  }  
  
  render() {  
    return (  
      <div>  
        <ul>  
          {this.state.data.map(item=> (<li>{item}</li>))}  
        </ul>  
      </div>  
    ); } }  
}
```

# Component Stateful



# Component Stateless #1

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

export default class App extends Component {
  render() {
    return (
      <div>
        <Header/>
        <Content/>
      </div>
    )
  }
}
```

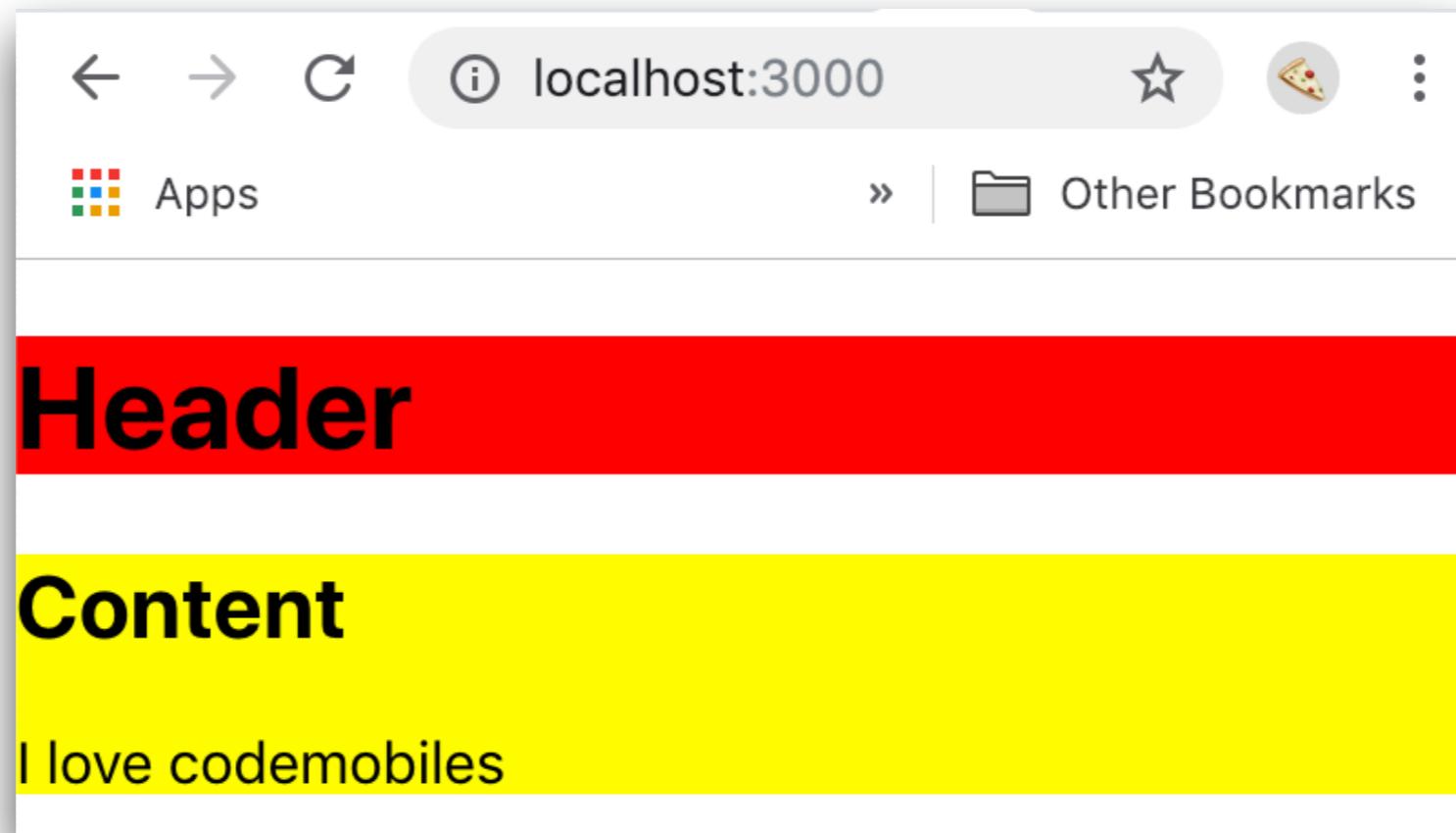
\*Never using state or setState

# Component Stateless #2

```
class Header extends Component {  
  render() {  
    return (  
      <div style={{backgroundColor: '#f00'}}>  
        <h1>Header</h1>  
      </div>  
    ) {}  
  }  
}
```

```
class Content extends Component {  
  render() {  
    return (  
      <div style={{backgroundColor: '#ff0'}}>  
        <h2>Content</h2>  
        <p>I love codemobiles</p>  
      </div>  
    ) {}  
  }  
}
```

# Component Stateless #3





# Install Bootstrap on React (Overview)

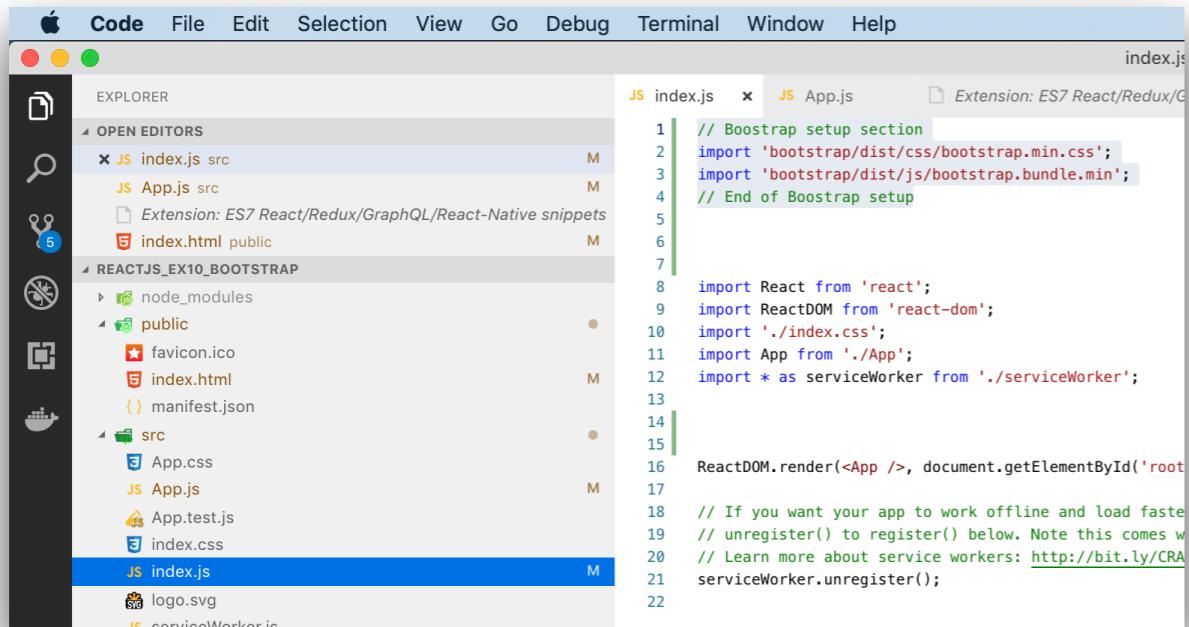
- npm - install libraries
- Revise index.js
- Revise App.js
- npm start

# Install Bootstrap on React #1 (npm)

```
$ npm install bootstrap react-bootstrap jquery popper.js
```

```
chaiyasits-MacBook-Pro-3:reactjs_ex10_bootstrap chaiyasit$ npm install bo
npm WARN ts-pnp@1.0.1 requires a peer of typescript@* but none is installed. You
+ bootstrap@4.3.1
+ popper.js@1.14.7
+ jquery@3.3.1
+ react-bootstrap@1.0.0-beta.5
updated 4 packages and audited 36357 packages in 18.175s
found 63 low severity vulnerabilities
  run `npm audit fix` to fix them, or `npm audit` for details
chaiyasits-MacBook-Pro-3:reactjs_ex10_bootstrap chaiyasit$
```

# Install Bootstrap on React #2 (index.js)



The screenshot shows a code editor interface with the following details:

- File Menu:** Code, File, Edit, Selection, View, Go, Debug, Terminal, Window, Help.
- Explorer:** Shows the project structure:
  - OPEN EDITORS: index.js (selected), App.js
  - REACTJS\_EX10\_BOOTSTRAP:
    - node\_modules
    - public
      - favicon.ico
      - index.html
      - manifest.json
    - src
      - App.css
      - App.js
      - App.test.js
      - index.css
    - index.js
    - logo.svg
    - serviceWorker.js
- Code Editor:** The file index.js contains the following code:

```
// Bootstrap setup section
import 'bootstrap/dist/css/bootstrap.min.css';
import 'bootstrap/dist/js/bootstrap.bundle.min';
// End of Bootstrap setup

import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';
import * as serviceWorker from './serviceWorker';

ReactDOM.render(<App />, document.getElementById('root')
  // If you want your app to work offline and load faster
  // unregister() to register() below. Note this comes with some performance overhead.
  // Learn more about service workers: http://bit.ly/CRA-PWA
  serviceWorker.unregister());
```

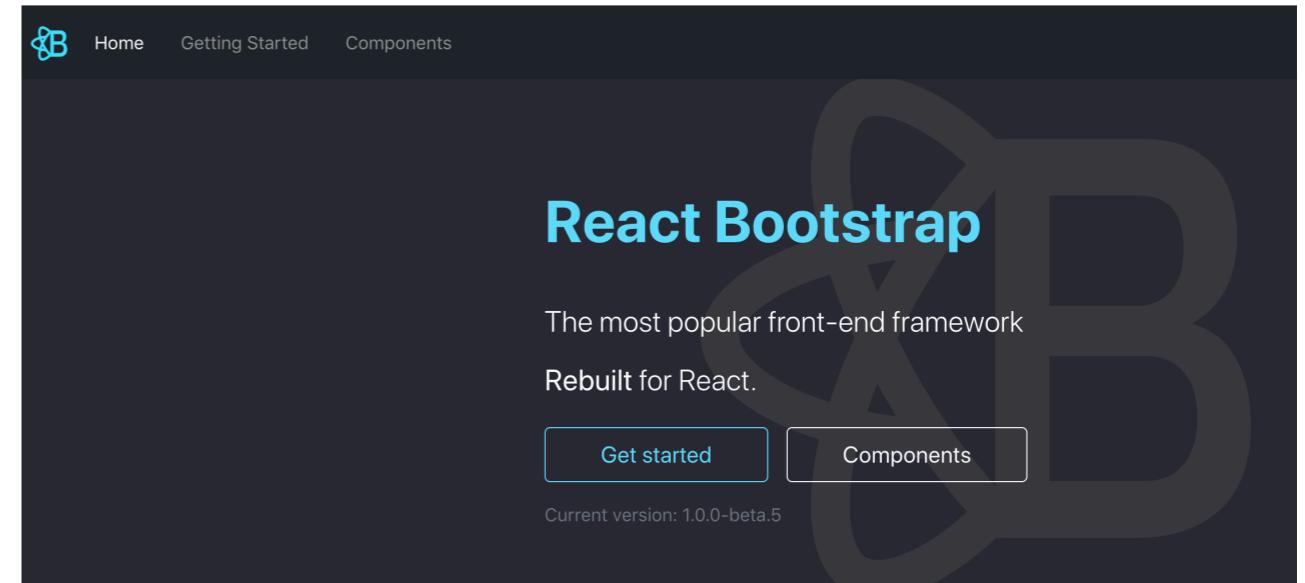
```
// Bootstrap setup section
import 'bootstrap/dist/css/bootstrap.min.css';
import 'bootstrap/dist/js/bootstrap.bundle.min';
// End of Bootstrap setup
```

# Install Bootstrap on React #3 (App.js)

<https://react-bootstrap.github.io/>

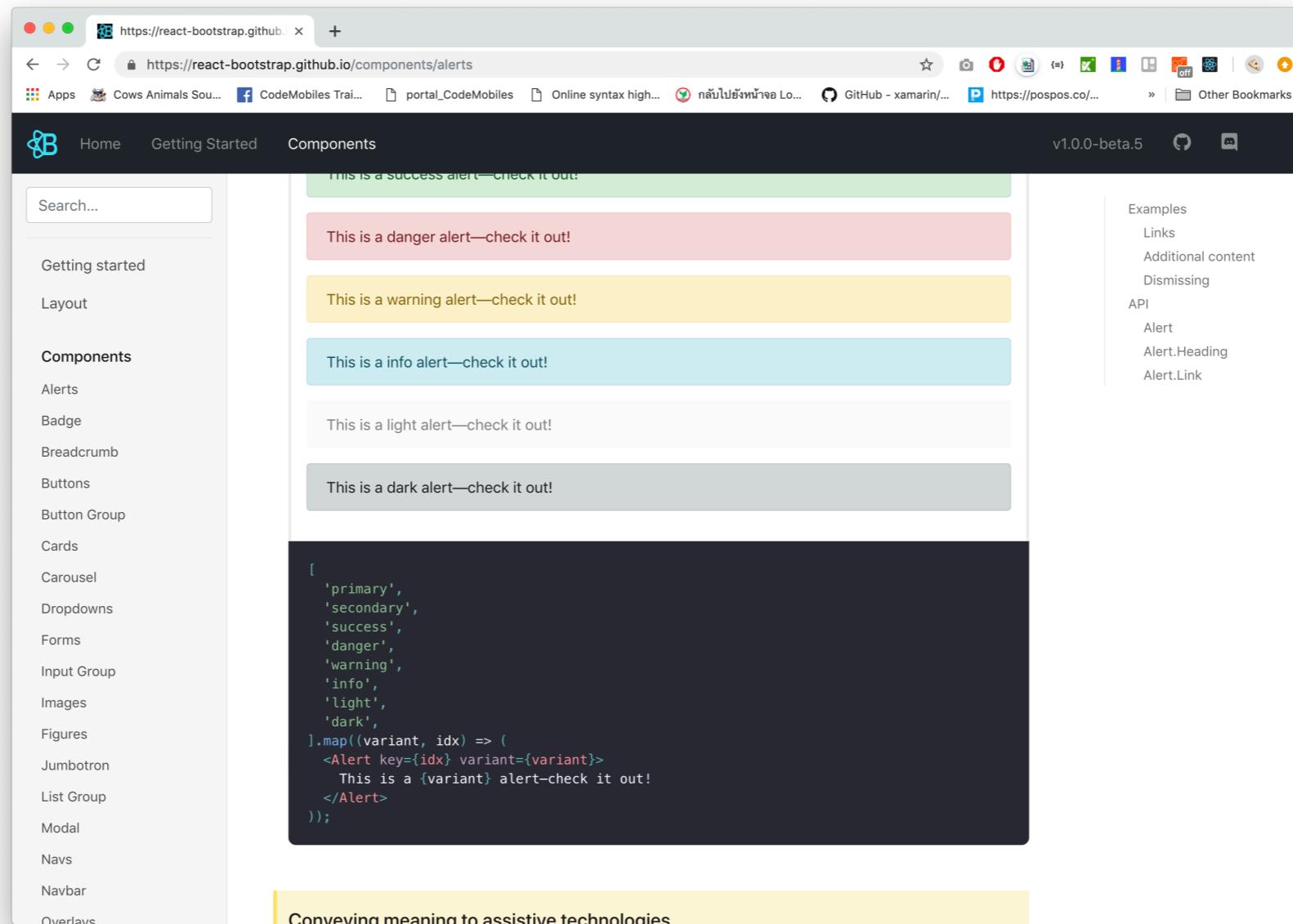
```
import React, { Component } from "react";
import {Button} from 'react-bootstrap'

export default class App extends Component {
  render() {
    return (
      <div>
        <Button>TEST</Button>
      </div>
    );
  }
}
```



# Install Bootstrap on React #3 (App.js)

<https://react-bootstrap.github.io/>

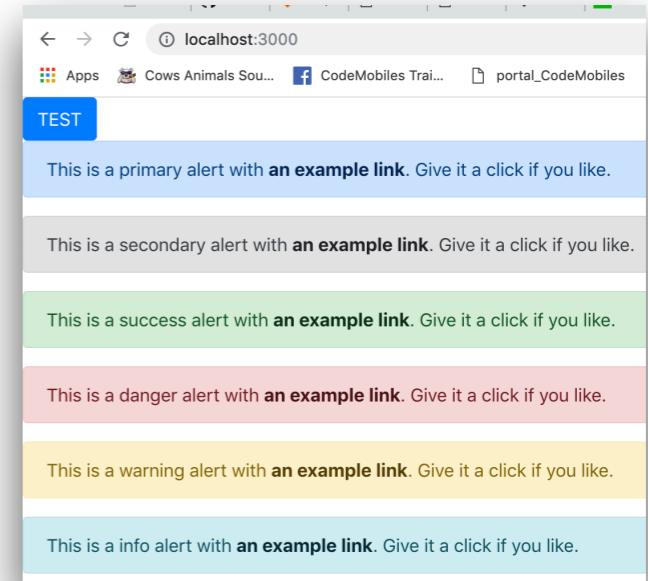


# Install Bootstrap on React #3 (App.js)

<https://react-bootstrap.github.io/>

```
import React, { Component } from "react";
import {Button, Alert} from 'react-bootstrap'

export default class App extends Component {
  render() {
    return (
      <div>
        <Button>TEST</Button>
        {
          [
            'primary',
            'secondary',
            'success',
            'danger',
            'warning'
          ].map((variant, idx) => (
            <Alert key={idx} variant={variant}>
              This is a {variant} alert with{' '}
              <Alert.Link href="#">an example link</Alert.Link>. Give it a click if you
              like.
            </Alert>
          ))
        }
      </div>
    ); {}
  }
}
```



# Flex-box Layout

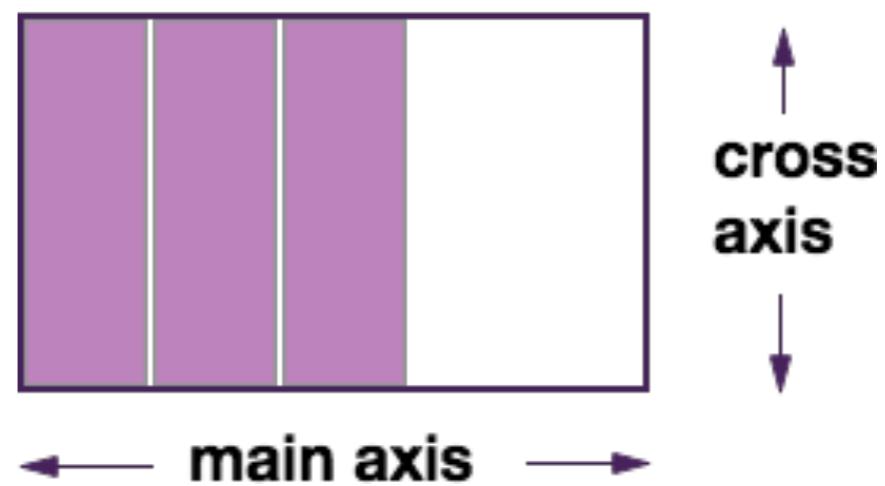
# Flex Axis

## main vs cross

**flex-direction: column**



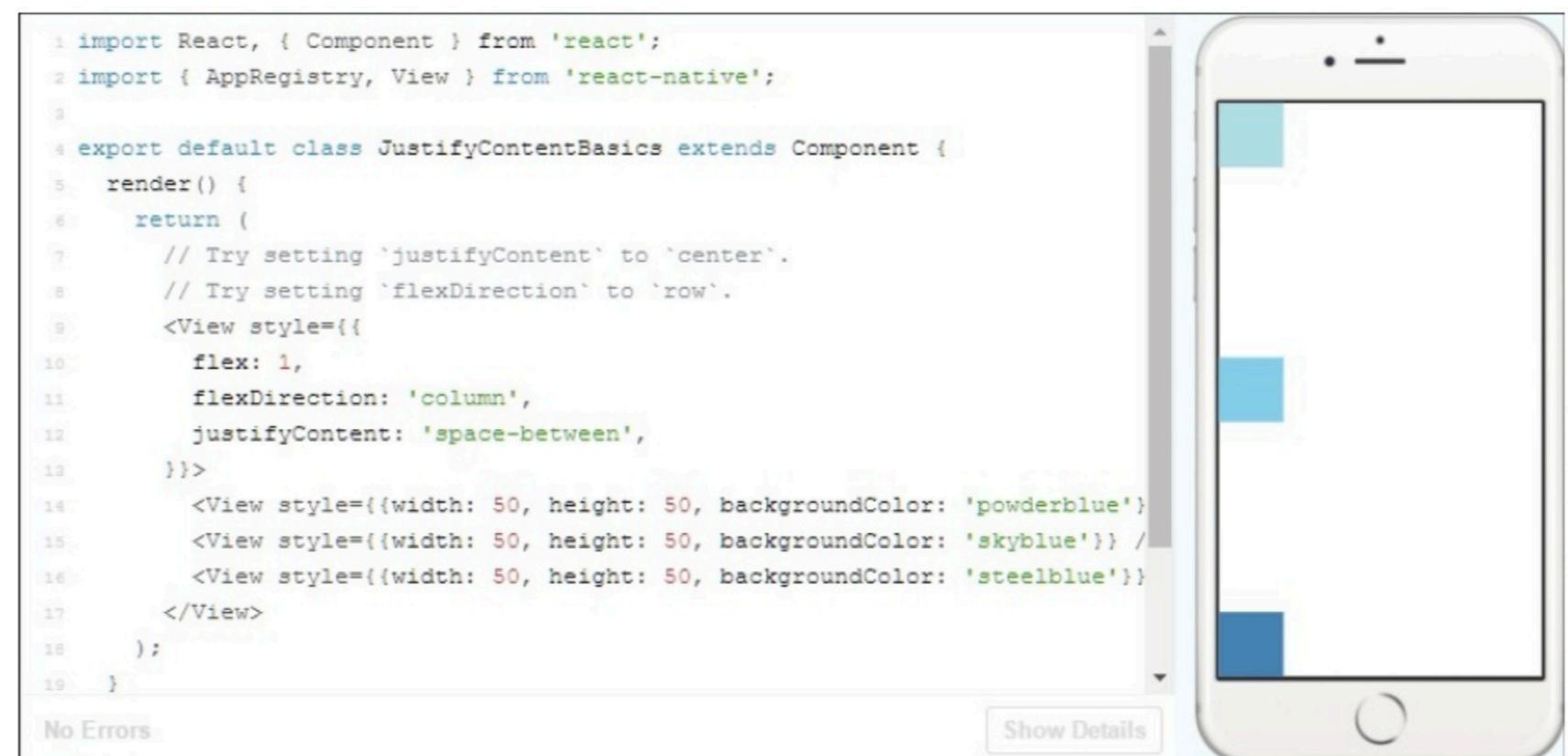
**flex-direction: row**



# JustifyContent

- Determine the distribution of children along the **main axis**.
- Available option are

- flex-start
- center
- flex-end
- space-around
- space-between

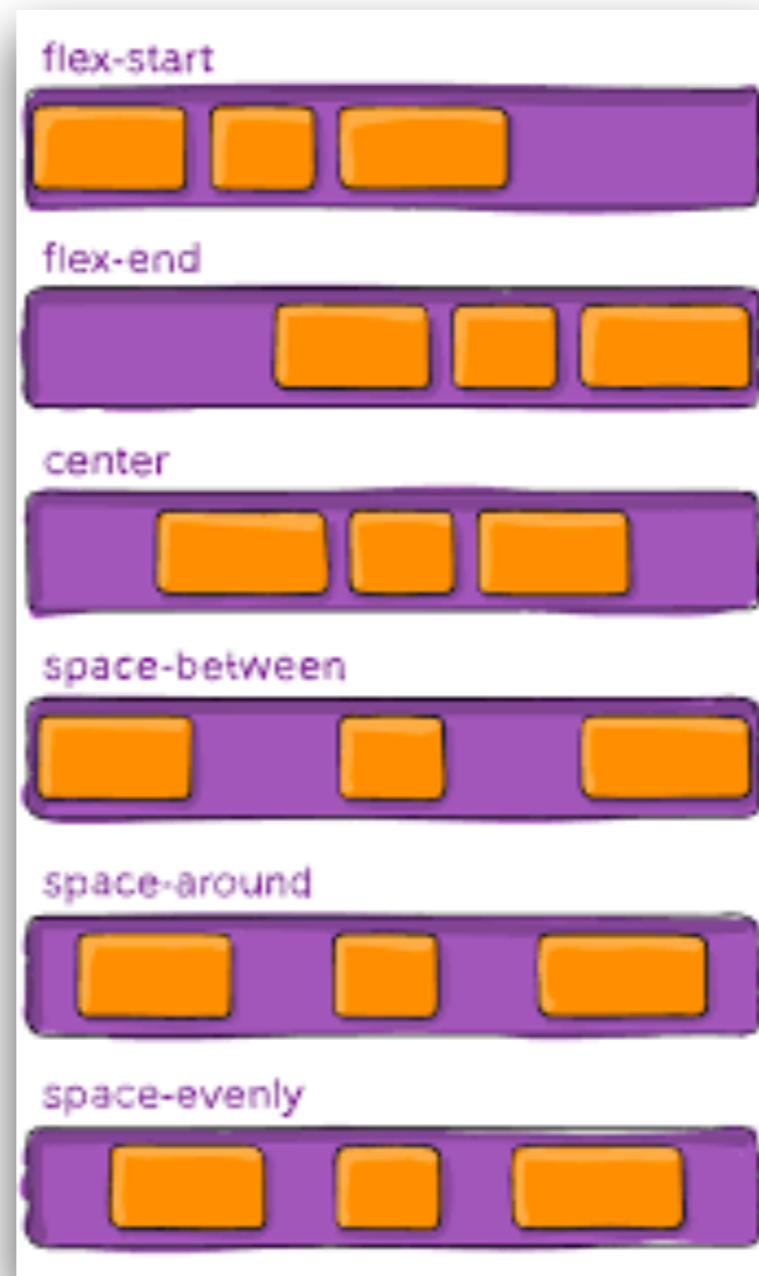


```
1 import React, { Component } from 'react';
2 import { AppRegistry, View } from 'react-native';
3
4 export default class JustifyContentBasics extends Component {
5   render() {
6     return (
7       // Try setting `justifyContent` to 'center'.
8       // Try setting `flexDirection` to 'row'.
9       <View style={{
10         flex: 1,
11         flexDirection: 'column',
12         justifyContent: 'space-between',
13       }}>
14         <View style={{width: 50, height: 50, backgroundColor: 'powderblue'}}
15         <View style={{width: 50, height: 50, backgroundColor: 'skyblue'}}
16         <View style={{width: 50, height: 50, backgroundColor: 'steelblue'}}>
17           </View>
18         );
19     }
20 }
```

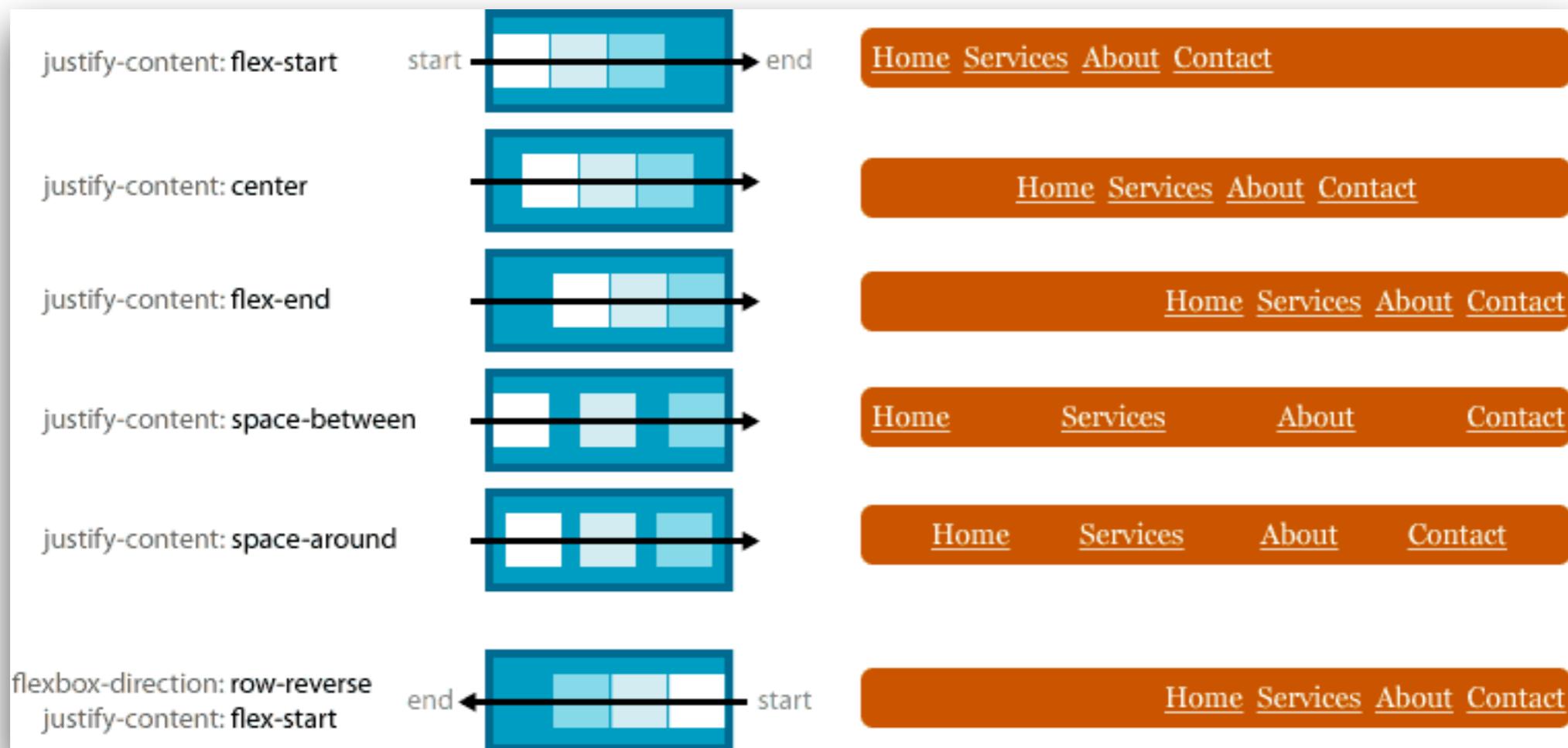
No Errors Show Details

The screenshot shows a React Native application running on an iPhone X simulator. The screen displays three blue squares of increasing shades of blue, arranged vertically with horizontal spacing between them, demonstrating the effect of `justifyContent: 'space-between'`.

# JustifyContent



# JustifyContent



# AlignItems

- Determine the alignment of children along the **secondary axis**
- Available option are

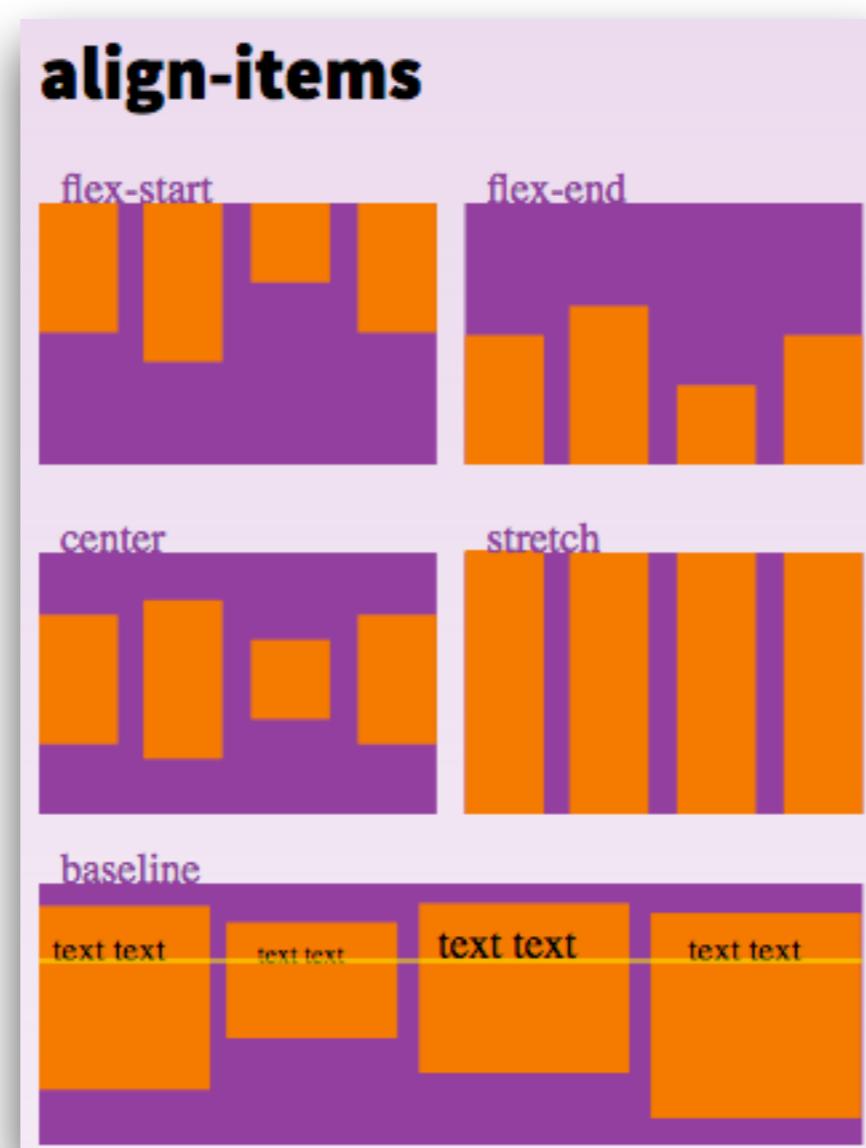
- flex-start
- center
- flex-end
- stretch



```
7 // Try setting 'alignItems' to 'flex-start'
8 // Try setting 'justifyContent' to 'flex-end'.
9 // Try setting 'flexDirection' to 'row'.
10 <View style={{
11   flex: 1,
12   flexDirection: 'column',
13   justifyContent: 'center',
14   alignItems: 'center',
15 }}>
16   <View style={{width: 50, height: 50, backgroundColor: 'powderblue'}}
17   <View style={{width: 50, height: 50, backgroundColor: 'skyblue'}}
18   <View style={{width: 50, height: 50, backgroundColor: 'steelblue'}}>
19   </View>
20 );
21 }
22 };
23
24 // skip this line if using Create React Native App
25 <
```

No Errors Show Details

# AlignItems



## Initialization

setup props and state

## Mounting

componentWillMount

render

componentDidMount

## Updation

props

componentWillReceiveProps

shouldComponentUpdate

true

componentWillUpdate

render

componentDidUpdate

states

shouldComponentUpdate

true

false

componentWillUpdate

render

componentDidUpdate

## Unmounting

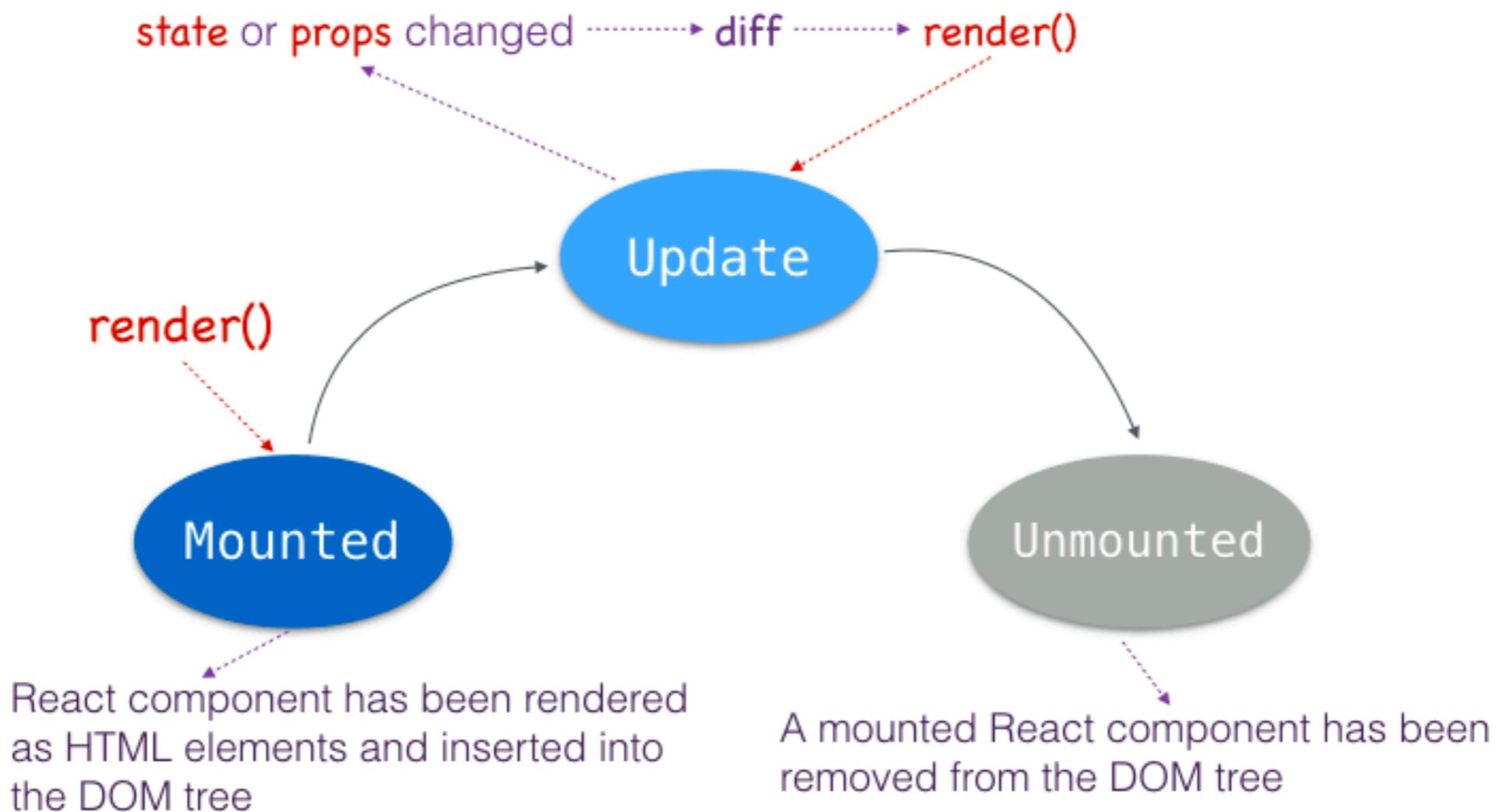
componentWillUnmount

# Component Lifecycle

# Component Lifecycle

- **Mounting** : *these methods are called during a creation of a component - being created and inserted into the DOM.*
  - constructor()
  - componentWillMount()
  - render()
  - componentDidMount()
- **Updating** : *these methods are called in the following order when a component is being re-rendered:*
  - componentWillReceiveProps()
  - shouldComponentUpdate()
  - componentWillUpdated()
  - render()
  - componentDidUpdate()
- **Unmounted** : *this point is called when the component is removed by DOM.*
  - componentWillUnmount

# Component Lifecycle



# Component

## getDerivedStateFromProps()

- Typically, it is used for getting data between component during navigation. This method is called before *componentDidMount*

```
static getDerivedStateFromProps(nextProps, prevState) {
  return {
    username: nextProps.navigation.getParam("username"),
    password: nextProps.navigation.getParam("password")
  };
}
```

# Component APIs

- `setState(..)`
- `forceUpdate(..)`
- `findDOMNode(..)`

# Component APIs

```
/* set state function */
<button onClick={()=>{this.setState({count1: 10})}} type="button" >setState V1 {this.state.count1}</button>
<button onClick={()=>{this.setState(p=>{return {count2: p.count2+1}})}} type="button">setState V2 {this.state.count2}</button>

/* forceUpdate function */
<button onClick={()=>{this.forceUpdate()}} type="button" >ForceUpdate {Math.random()} </button>

/* findDOMNode function */
<button onClick={()=>{
  var myDiv = document.getElementById('comment');
  ReactDOM.findDOMNode(myDiv).style.color = 'red';
}} type="button">FindDomeNode </button>

<div id = "comment">CodeMobiles</div>
```

# React Hook

# React Hook

- It is a feature of React that lets you to use state and other React features (such as lifecycle) without writing a class or extend React Component Class

## STATE HOOKS

```
You, a few seconds ago | 1 author (You)
import React from 'react';
import {Button, Text, View} from 'react-native';

You, a few seconds ago | 1 author (You)
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 0,
    };
  }

  handleButtonClick = () => {
    this.setState({
      count: this.state.count +1
    });
  }

  render() {
    return (
      <View>
        <Text>You clicked {this.state.count} times</Text>
        <Button
          | onPress={this.handleButtonClick}
        />
      </View>
    );
  }
}

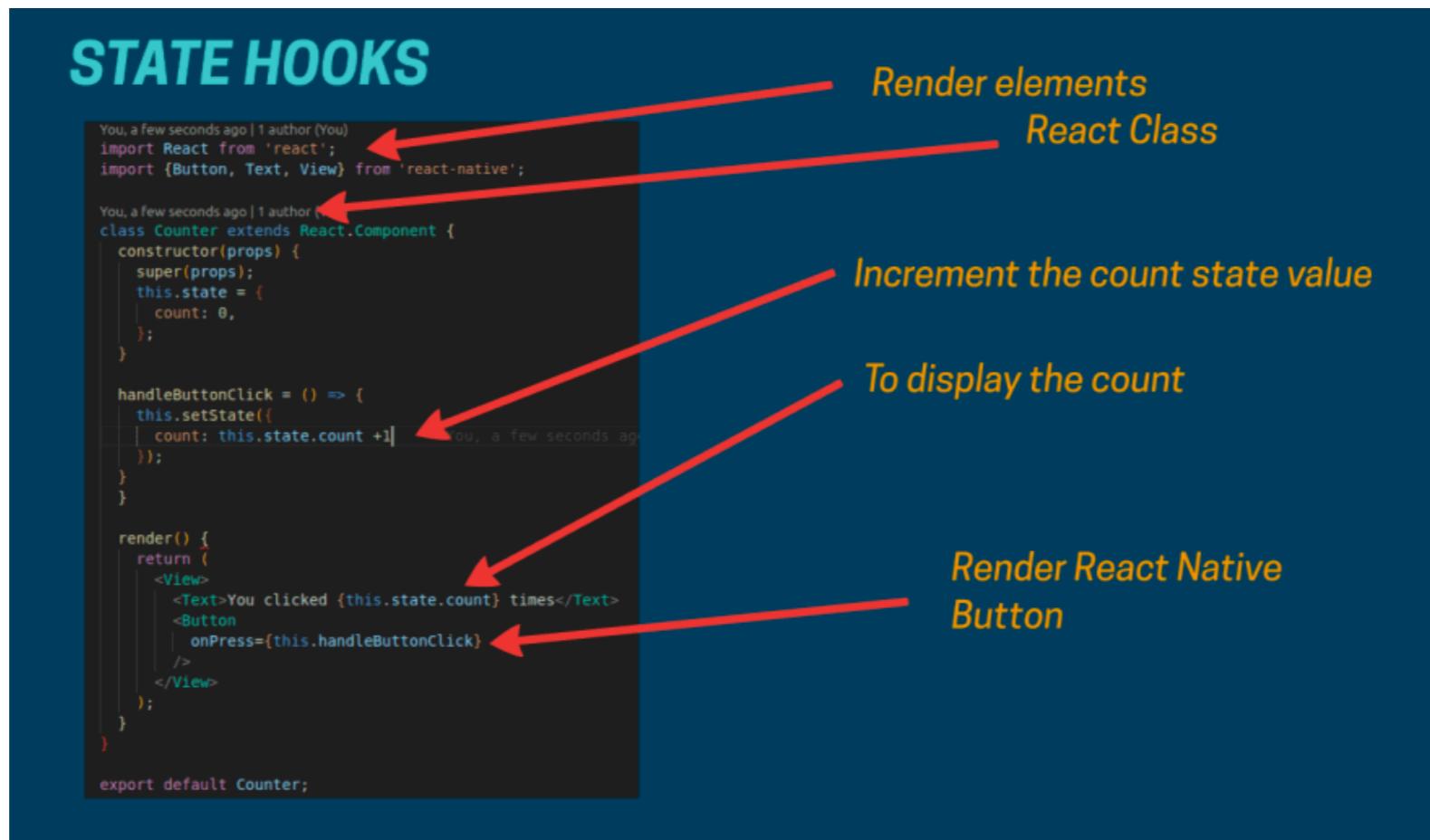
export default Counter;
```

*Render elements*  
*React Class*

*Increment the count state value*

*To display the count*

*Render React Native Button*



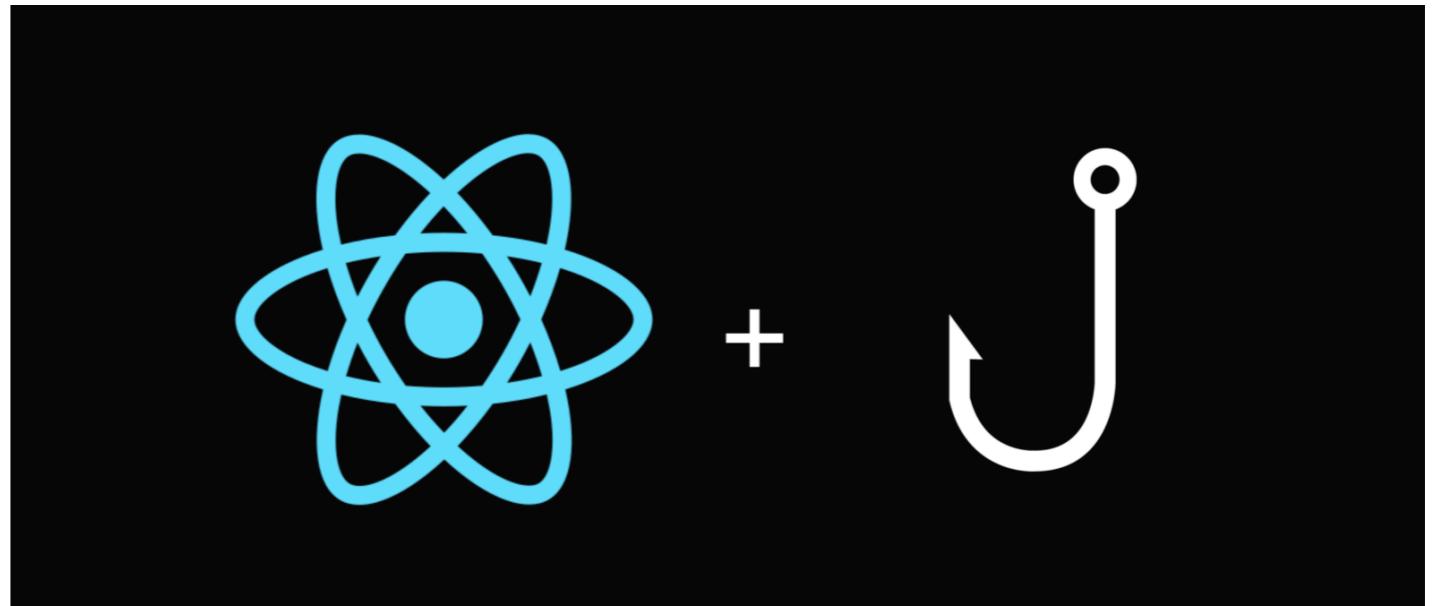
# React Hook

```
function ExampleWithManyStates() {
  // Declare multiple state variables!
  const [age, setAge] = useState(42);
  const [fruit, setFruit] = useState('banana');
  const [todos, setTodos] = useState([{ text: 'Learn Hooks' }]);
  // ...
}
```

Hooks are functions that let you “hook into” React state and lifecycle features from function components. Hooks don’t work inside classes — they let you use React without classes. (We don’t recommend rewriting your existing components overnight but you can start using Hooks in the new ones if you’d like.)

# Common Hooks

- useState
- useEffect
- useContext
- useReducer
- useSelector
- useCallback
- useMemo (not React.memo)
- useRef
- useImperativeHandle
- useLayoutEffect
- useDebugValue



<https://www.elanandkumar.com/blog/react-comp-lifecycle-with-hooks/>

# useState



- Used to create a local state
- count is getter
- setCount is setter
- useState(0) - 0 is just initial value

# useEffect

Syntax:

Callback function that invokes when  
useEffect runs

```
React.useEffect(() => {
```

```
    // ...code goes here
```

```
    // OPTIONAL return statement
```

```
    return () => {}
```

```
}, optionalParam);
```

The second parameter helps to  
control the execution of useEffect

This function executes once effect's execution  
ends. Helps with cleanup task (if any)

- `useEffect(fn, [])`
- `fn` or `()=>{}` is an initialization hook function of component like `componentDidMount`
- `[]` is an array to put watching states. When the watching states is changed, the hook function is called.

# useEffect - Example

## Updating the page title

This effect will run the first time the component is rendered, and then only ever run again if the title has changed.

```
const [title, setTitle] = React.useState("Hooks 101");

React.useEffect(() => {
  document.title = title;
}, [title]);
```

# useEffect - Example

Fetching data from an API into local state.

Since our state changing will not affect the list of products that is returned, we can pass an empty array `[]` as our dependency so that the effect will only run when the component is first mounted.

```
const [products, setProducts] = React.useState([]);

React.useEffect(() => {
  getProducts()
    .then(products => {
      setProducts(products);
    })
}, []);
```

# useEffect - Example

Fetching data from an API into local state, based on a query.

If we have a query or filter to modify the set of API data we want, then we can pass it as a dependency to make sure that React runs this effect every time the component renders using a new query.

```
const [products, setProducts] = React.useState([]);
const [query, setQuery] = React.useState("");

React.useEffect(() => {
  getProducts({name: query})
    .then(products => {
      setProducts(products);
    })
}, [query]);
```

# useEffect - Example

Dispatching a Redux action.

If your GET action already reduces into your Redux state, then you don't need to maintain any of that locally.

By passing `products.length` as a dependency, you only run this

```
const dispatch = Redux.useDispatch();
const products = Redux.useSelector(state => state.products);

React.useEffect(() => {
  dispatch(GetProducts())
}, []);
```

# UseEffect vs ComponentDidMount

## componentDidMount

As explained before about `useEffect`, we can pass the second optionalParameter to achieve the `componentDidMount` behavior. Let's have a look at the following code snippet.

```
React.useEffect(()=>{
  console.log('Robot: componentDidMount');
  // ...code goes here...
  fetchData();
}, []);
```

# UseEffect vs ComponentDidUpdate

## componentDidUpdate

Let's say, we need to invoke api call again if any props changes for the component, we do it in here once the component updates. We can achieve it using `useEffect`. Though, this time, we need to pass an entry to the second argument.

```
React.useEffect(() => {
  console.log('Robot: componentWillMount');
  // code goes here
  fetchData();
}, [props.selectedRobotId]);
```

# UseEffect vs ComponentWillUnmount

## componentWillUnmount

Remember, in the explanation of `useEffect`, I mentioned about an optional return function inside the `useEffect`. This return function serves the purpose of cleanup.

```
React.useEffect(() => {
  console.log('Robot: componentDidUpdate');
  //...code goes here
  fetchData();
  return () => {
    console.log('cleaning up...');
  }
}, [props.selectedRobotId])
```

# React.memo (HOC) vs ShouldComponentUpdate

```
export default React.memo(Robot, (prevProps, nextProps) => {
  console.log('Robot: shouldComponentUpdate');
  return nextProps.selectedRobotId === prevProps.selectedRobotId;
});
```

There is one thing to note here. This works opposite of `shouldComponentUpdate`. You return `true` if you do not want to re-render. You return `false` if you want it to render.

- In example above, Robot is a component
- Instead of exporting Robot component directly, it is wrapped with React.memo (HOC) to define condition of re-rendering condition of the component

# useMemo

Unlike `useEffect`, `React.useMemo` does not trigger every time you change one of its dependencies.

A memoized function will first check to see if the dependencies have changed since the last render. If so, it executes the function and returns the result. If false, it simply returns the cached result from the last execution.

This is good for expensive operations like transforming API data or doing major calculations that you don't want to be re-doing unnecessarily

```
const posts = Redux.useSelector(state => state.posts);

const tags = React.useMemo(() => {
  return getTagsFromPosts(posts)
}, [posts]);
```

## Bonus: React.useCallback

This is a shortcut for a specific `React.useMemo` usage.

`React.useMemo` returns a memoized **value**  
`React.useCallback` returns a memoized **function**

💡 *But a value can totally be a function!*

Correct! That means these two snippets are equivalent

```
const memoizedFunction = React.useMemo(function() {
  return function doTheThing(a, b) {
    // do the thing
  }
}, [a, b])
```

↑ This memoizes the value the first argument (a function) returns, which is a function called `doTheThing`.

```
const memoizedFunction = React.useCallback(function doTheThing(a, b) {
  // do the thing
}, [a, b])
```

# Stylesheet

# Stylesheet

## Stylesheet.create()

```
const styles = StyleSheet.create({
  container: {
    borderRadius: 4,
    borderWidth: 0.5,
    borderColor: '#d6d7da',
  },
  title: {
    fontSize: 19,
    fontWeight: 'bold',
  },
  activeTitle: {
    color: 'red',
  },
});
```

▲ TL;DR Always use `StyleSheet.create()` when you can.

38

The [answer by Nico](#) is correct, but there is more to it.

▼

To summarize:



1. It validates the styles as mentioned by Nico
2. As mentioned in the [documentation](#):

Making a stylesheet from a style object makes it possible to refer to it by ID instead of creating a new style object every time.

```
<View style={styles.container}>
  <Text style={[styles.title, this.props.isActive && styles.activeTitle]} />
</View>
```

# Stylesheet

## containerStyle

```
<Card containerStyle={{ overflow: 'hidden', flexDirection: 'column', marginBottom: 20, borderRadius: 8, padding: 0 }}>  
  <View style={{ flexDirection: 'row', marginBottom: 16, height: 45, alignItems: 'center' }}>  
    <Image source={{ uri: item.avatar_image }} style={{ width: 45, height: '100%', marginRight: 16 }} />  
    <View style={{ flexDirection: 'column' }}>  
      <Text style={{ fontWeight: '700' }}>{item.title}</Text>  
      <Text style={{ fontWeight: '100' }}>{item.subtitle}</Text>  
    </View>  
  </View>  
  <Image source={{ uri: item.youtube_image }} style={{ width: "100%", height: 190 }} />  
</Card>
```

# Stylesheet

```
const styles = StyleSheet.create({
  container: {
    borderRadius: 4,
    borderWidth: 0.5,
    borderColor: '#d6d7da',
  },
  title: {
    fontSize: 19,
    fontWeight: 'bold',
  },
  activeTitle: {
    color: 'red',
  },
});
```

```
<View style={styles.container}>
  <Text style={[styles.title, this.props.isActive && styles.activeTitle]} />
</View>
```

# React

## Props vs State

# Props vs State



Props immutable  
State is mutable

state is used for internal  
communication inside a Component

# state vs props

- **State** can be updated after created while props can't
- **Props** value must set during its definition

# Props

# What is Props?

If components were plain JavaScript **functions**, then props would be the function input. Going by that analogy, a component accepts an input (what we call props), processes it, and then renders some JSX code.

*\*In short, props is a **function variable** that accept input to render component during its creation.*



props are received from a parent component and are **read only**

# Passing Data between components using Props.

```
1 import React, {Component} from 'react';
2 import {AppRegistry, Text, View} from 'react-native';
3
4 import Component1 from './app/components/Component1/Component1';
5
6 export default class myapp extends Component{
7   render(){
8     return(
9       <View>
10         <Component1 message="Hello World" />
11       </View>
12     );
13   }
14 }
15
16 AppRegistry.registerComponent('myapp', () => myapp);
```

Page 1

# Passing Data between components Props.

```
1 import React, {Component} from 'react';
2 import {AppRegistry, Text, View} from 'react-native';
3
4 export default class Component1 extends Component{
5   render(){
6     return(
7       <View>
8         <Text>{this.props.message}</Text>
9       </View>
10    );
11  }
12}
13
14 AppRegistry.registerComponent('Component1', () => Component1);
```

Page 2

# Validating Props

```
import React from 'react';
import PropTypes from 'prop-types'

class App extends React.Component {
  render() {
    return (
      <div>
        <h3>Array: {this.props.propArray}</h3>
        <h3>Bool: {this.props.propBool ? "True..." : "False..."}</h3>
        <h3>Func: {this.props.propFunc(3)}</h3>
        <h3>Number: {this.props.propNumber}</h3>
        <h3>String: {this.props.propString}</h3>
        <h3>Object: {this.props.propObject.objectName1}</h3>
        <h3>Object: {this.props.propObject.objectName2}</h3>
        <h3>Object: {this.props.propObject.objectName3}</h3>
      </div>
    );
  }
}
```

```
App.propTypes = {
  propArray: PropTypes.array.isRequired,
  propBool: PropTypes.bool.isRequired,
  propFunc: PropTypes.func,
  propNumber: PropTypes.number,
  propString: PropTypes.string,
  propObject: PropTypes.object
}

App.defaultProps = {
  propArray: [1,2,3,4,5],
  propBool: true,
  propFunc: function(e){return e},
  propNumber: 1,
  propString: "String value...",

  propObject: {
    objectName1:"objectValue1",
    objectName2: "objectValue2",
    objectName3: "objectValue3"
  }
}

export default App;
```

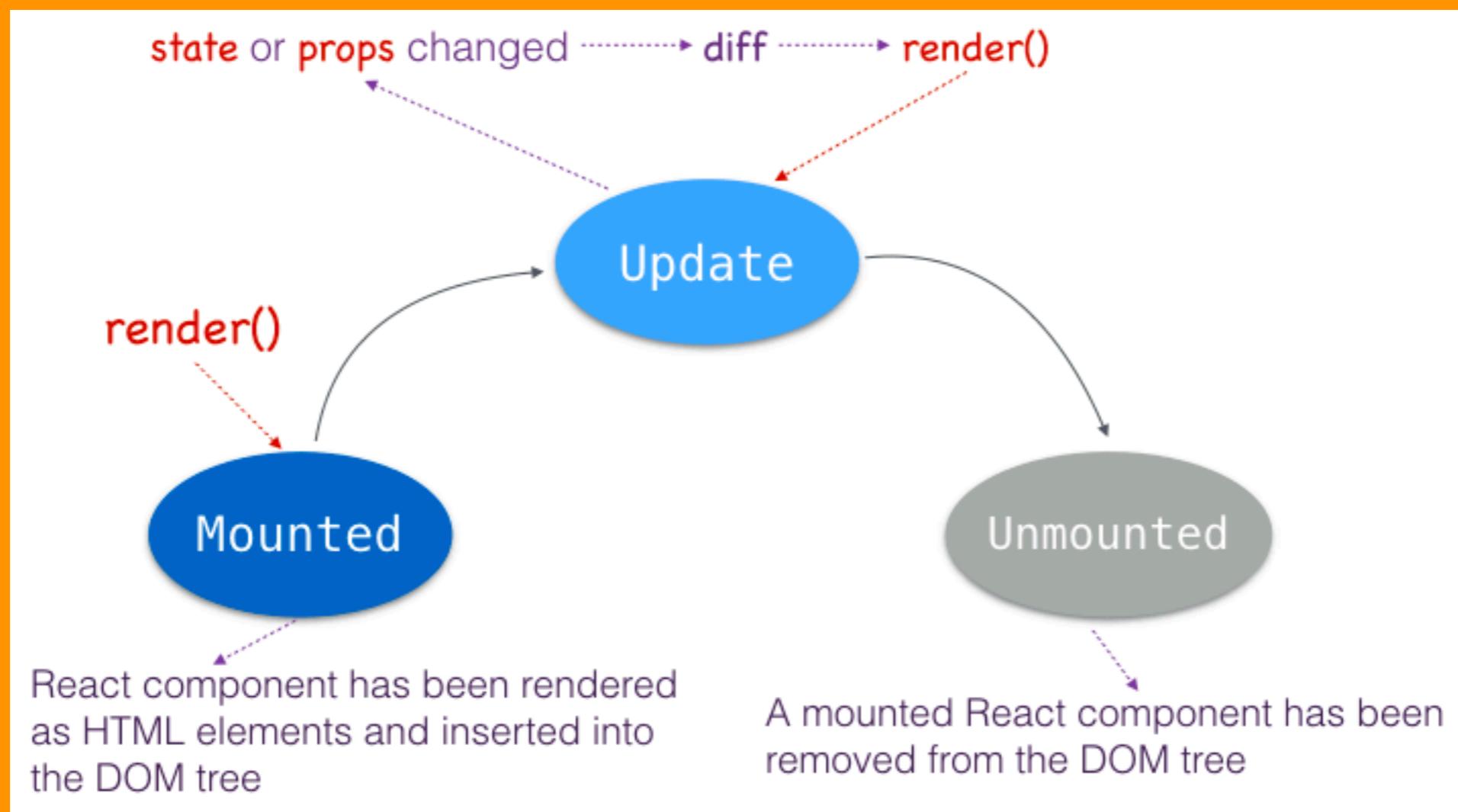
# Validating Props

```
import React from 'react';
import PropTypes from 'prop-types'

class App extends React.Component {
  render() {
    return (
      <div>
        <h3>Array: {this.props.propArray}</h3>
        <h3>Bool: {this.props.propBool ? "True..." : "False..."}</h3>
        <h3>Func: {this.props.propFunc(3)}</h3>
        <h3>Number: {this.props.propNumber}</h3>
        <h3>String: {this.props.propString}</h3>
        <h3>Object: {this.props.propObject.objectName1}</h3>
        <h3>Object: {this.props.propObject.objectName2}</h3>
        <h3>Object: {this.props.propObject.objectName3}</h3>
      </div>
    );
  }
}
```

**Array: 12345**  
**Bool: True...**  
**Func: 3**  
**Number: 1**  
**String: String value...**  
**Object: objectValue1**  
**Object: objectValue2**  
**Object: objectValue3**

# State



# What is State?

State is an object that is owned by the component where it is declared. Its scope is limited to the current component. A component can initialize its state and update it whenever necessary.

*\*In short, state is like an javascript object. the component keeps rendered whenever the states are changed.*

# State

## Initialization at Constructor

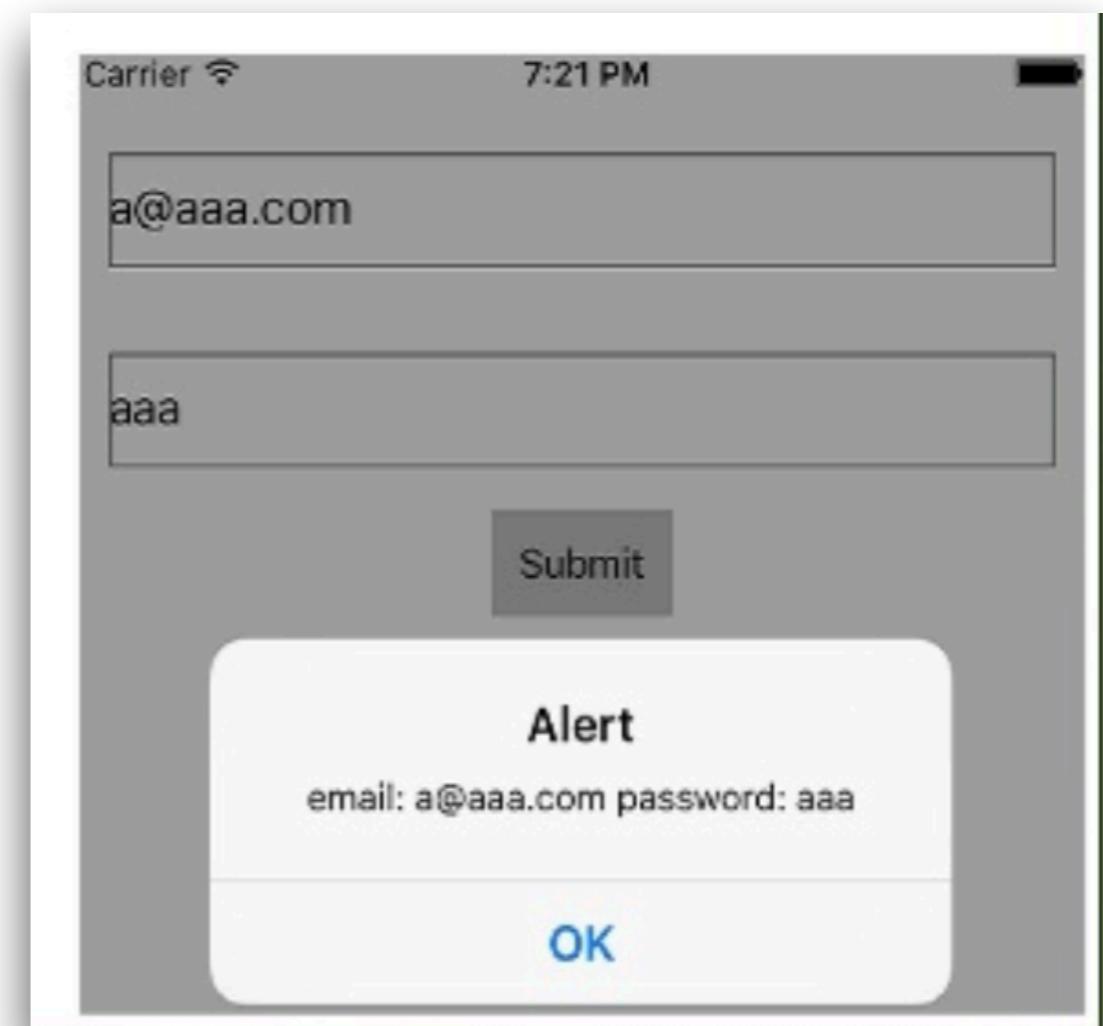
```
export default class Component1 extends Component{
  constructor(){
    super();
    this.state = {
      name:'Brad'
    }
  }

  render(){
    return(
      <View>
        <Text>{this.props.message}</Text>
        <Text>{this.state.name}</Text>
      </View>
    );
  }
}
```

# State

## Example #1

- `setState`
- `TextInput`
- Props to call function



# State

## Example #2

```
import React, { Component } from 'react'
import { View } from 'react-native'
import MyPresentationalComponent from './MyPresentationalComponent'

export default class MyContainerComponent extends Component {
  constructor() {
    super()
    this.state = { email: '', password: '' }
  }
  updateEmail = (text) => {
    this.setState({email: text})
  }
  updatePassword = (text) => {
    this.setState({password: text})
  }
  login = () => {
    alert('email: ' + this.state.email + ' password: ' + this.state.password)
  }

  render(){
    return(
      <View>
        <MyPresentationalComponent
          updateEmail = {this.updateEmail}
          updatePassword = {this.updatePassword}
          login = {this.login}
        />
      </View>
    )
  }
}
```

Page 1

# State

## Example #3

```
import React, { Component } from 'react'
import {View,Text,TouchableHighlight,TextInput, StyleSheet } from 'react-native'
export default MyPresentationalComponent = (props) => {
  return (
    <View style = {styles.container}>
      <TextInput
        style = {styles.input}  placeholder = 'Email'    autoCapitalize = 'none'
        onChangeText = {props.updateEmail}
      />
      <TextInput
        style = {styles.input}  placeholder = 'Password'  autoCapitalize = 'none'
        onChangeText = {props.updatePassword}
      />
      <TouchableHighlight
        style = {styles.submit}
        onPress = { () => props.login(props.email, props.password)}>
        <Text> Submit </Text>
      </TouchableHighlight>
    </View>
  )
}

const styles = StyleSheet.create ({
  container: { flex: 1, alignItems: 'center', justifyContent:'center', paddingTop: 23 },
  input: { margin: 15, height: 40, borderColor: 'grey', borderWidth: 1 },
  submit: {backgroundColor: 'silver', padding: 10 }
})
```

Page 2

# Default Props

- Allow to set default value of a props-key - if that props-key is not yet set value from constructor

```
export default class Component1 extends Component{
  constructor(props){
    super(props);
    this.state = {
      name:'Brad',
      showName: true,
      message: this.props.message
    }
  }

  static defaultProps = {
    message: 'Hi There'
  }

  render(){
    let name = this.state.showName ? this.state.name : 'No name';
    return(
      <View>
        <Text>{this.state.message}</Text>
        <Text>{name}</Text>
      </View>
    );
  }
}
```

# Event Listeners

# Event Listener

```
<!-- HTML --!>
<button onclick="activateLasers()">
```

```
<!-- REACT --!>
<button onClick="activateLasers()">
```

# Event Listener

Event	Value	Description
onchange	script	Script runs when the element changes
onsubmit	script	Script runs when the form is submitted
onreset	script	Script runs when the form is reset
onselect	script	Script runs when the element is selected
onblur	script	Script runs when the element loses focus
onfocus	script	Script runs when the element gets focus
onkeydown	script	Script runs when key is pressed
onkeypress	script	Script runs when key is pressed and released
onkeyup	script	Script runs when key is released
onclick	script	Script runs when a mouse click
ondblclick	script	Script runs when a mouse double-click
onmousedown	script	Script runs when mouse button is pressed
onmousemove	script	Script runs when mouse pointer moves
onmouseout	script	Script runs when mouse pointer moves out of an element
onmouseover	script	Script runs when mouse pointer moves over an element
onmouseup	script	Script runs when mouse button is released

Just Change to  
Uppercase Camel

# Event Listener

For example, the HTML:

```
<button onclick="activateLasers()">  
  Activate Lasers  
</button>
```

is slightly different in React:

```
<button onClick={activateLasers}>  
  Activate Lasers  
</button>
```

# Arrow vs Variable Function

```
(() => {})
```

# Demo

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

export default class App extends Component {
  constructor(props) {
    super(props);
    this.state = {
      message: ""
    };
  }

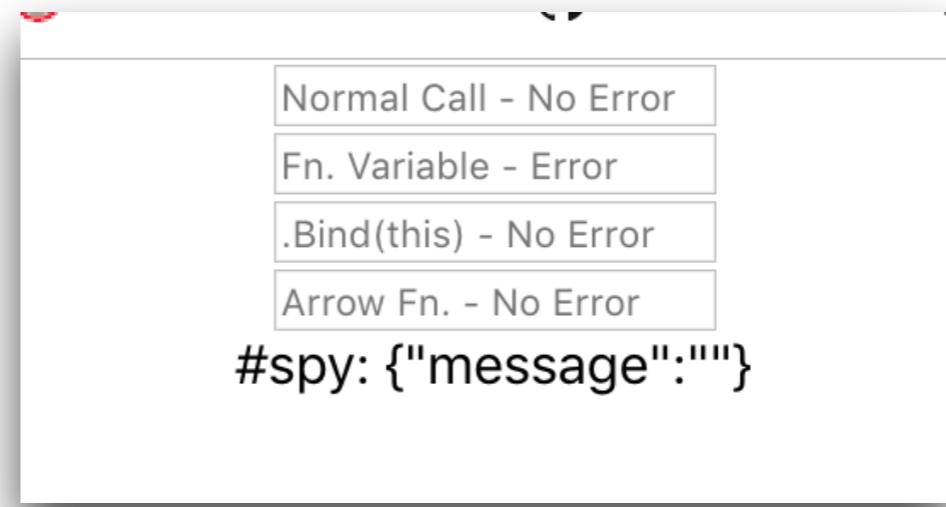
  // No Error
  onChange1(value) {
    this.setState({ message: value });
  }

  // Error Cannot read property 'setState' of undefined
  onChange2 = function(e) {
    this.setState({ message: e.target.value });
  };

  // No Error using bind(this)
  onChange3 = function(e) {
    this.setState({ message: e.target.value });
  };

  // No Error using bind(this)
  onChange4 = e => {
    this.setState({ message: e.target.value });
  };

  render() {
    return (
      <div style={{textAlign:'center'}}>
        <input
          type="text"
          onChange={e => this.onChange1(e.target.value)}
          placeholder="Normal Call - No Error"
        />
        <br />
        <input
          type="text"
          onChange={this.onChange2}
          placeholder="Fn. Variable - Error"
        />
        <br />
        <input
          type="text"
          onChange={this.onChange3.bind(this)}
          placeholder=".Bind(this) - No Error"
        />
        <br />
        <input
          type="text"
          onChange={this.onChange4}
          placeholder="Arrow Fn. - No Error"
        />
        <br />
        <span>#spy: {JSON.stringify(this.state)}</span>
      </div>
    );
  }
}
```



# Calling Function #1

```
<input  
  type="text"  
  onChange={e => this.onChange1(e.target.value)}  
  placeholder="Normal Call – No Error"  
/>
```



© Can Stock Photo - csp14311907

```
// No Error  
onChange1(value) {  
  this.setState({ message: value });  
}
```

# Calling Function Obj #2

```
<input  
    type="text"  
    onChange={this.onChange2}  
    placeholder="Normal Call – No Error"  
/>
```

```
// Error Cannot read property 'setState' of undefined  
onChange2 = function(e) {  
    this.setState({ message: e.target.value });  
};
```



← → 1 of 3 errors on the page

TypeError: Cannot read property 'setState' of undefined

App.\_this.onChange2  
src/App.js:4:21

```
1 | import React, { Component } from "react";  
2 |  
3 | export default class App extends Component {  
> 4 |     constructor(props) {  
5 |         super(props); ^
```

# Calling Function Obj #3

## Using Bind(this)

```
<input  
    type="text"  
    onChange={this.onChange3.bind(this)}  
    placeholder="Normal Call - No Error"  
/>
```

```
onChange3 = function(e) {  
    this.setState({ message: e.target.value });  
};
```



© Can Stock Photo - csp14311907

# Calling Function Obj #4

## Using Arrow Function

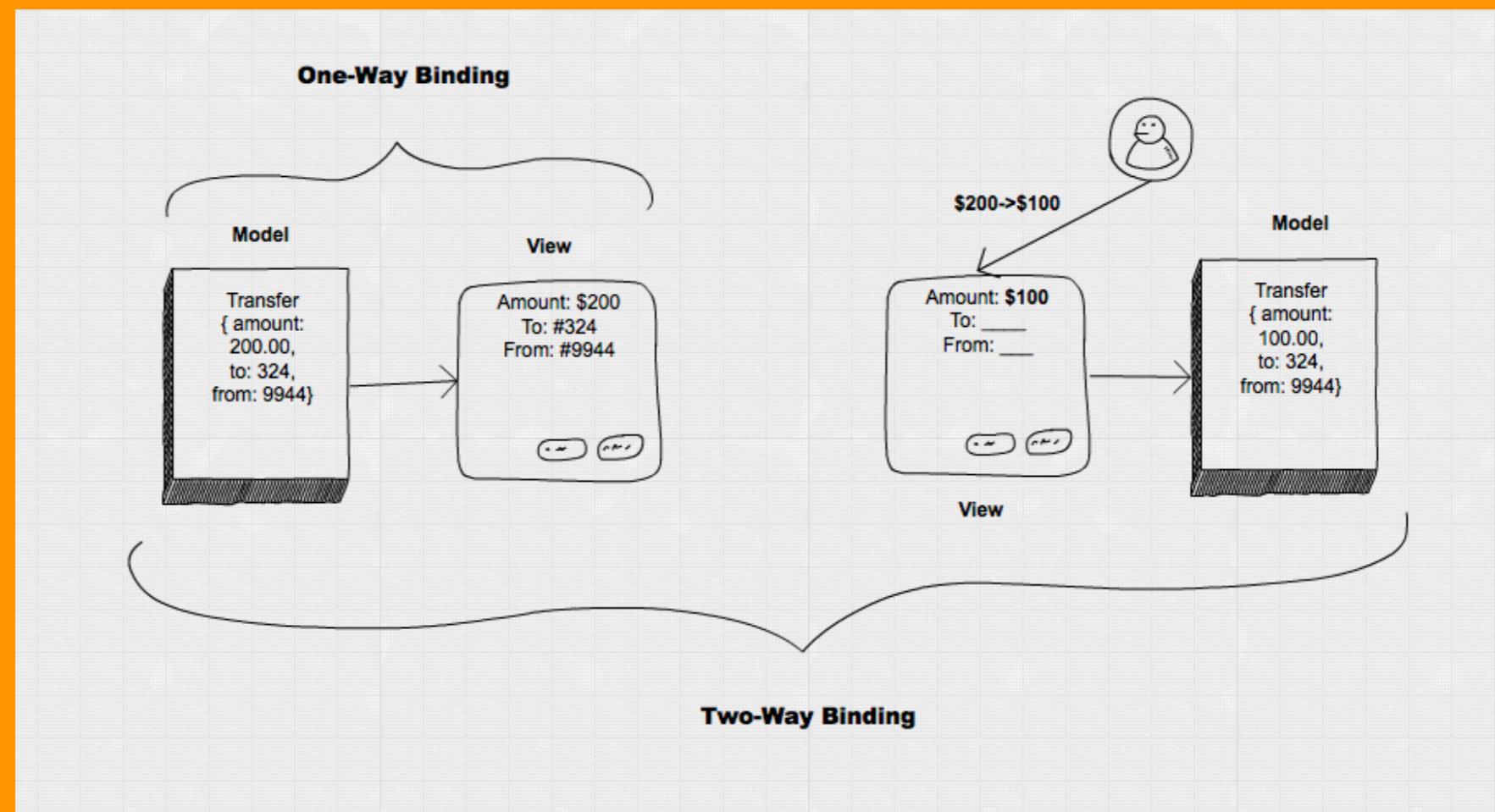
```
<input  
  type="text"  
  onChange={this.onChange4}  
  placeholder="Normal Call – No Error"  
/>
```



```
onChange4 = (e) => {  
  this.setState({ message: e.target.value });  
};
```

( ) => {}

# Form Input



# Redux-Form



# Redux-Form



- Auto inject a form-values into redux-state as a reducer
- Flexible to get and set form's value
- Built-in Form Validation
- Initialize form-values

# Redux-Form

## Simple Form

```
<form onSubmit={handleSubmit(this.onClickSubmit)} >

  <Field
    name="price"
    component="input"
    className="form-control"
    type="number"
    id="price" />

  <button type="submit"> Submit </button>

</form>
```

# Redux-Form

## Example #1

```
16/  
168  
169 const mapStateToProps = state => ({  
170   stockReducer: state.stockReducer,  
171   loginForm: state.form.loginForm  
172});  
173  
174 StockCreate = connect(  
175   mapStateToProps,  
176   actions  
177 )(StockCreate);  
178  
179 export default reduxForm({  
180   fields: {  
181     hidden_field1: "1234",  
182     hidden_field2: "1234"  
183   },  
184   form: "loginForm",  
185   initialValues: { price: 100, stock: 10 }  
186 })(StockCreate);  
187  
188  
189
```

Way to access form values

```
showPreviewImage = () => {  
  if (this.props.loginForm) {  
    let file = this.props.loginForm.value.file_obj;  
    if (file) {  
      return <img src={file} style={{ height: 100 }} />;  
    }  
  }  
};
```

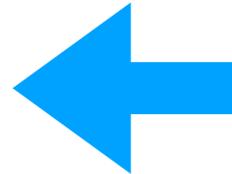
Hidden fields

Initial Values

# Redux-Form

## Example #2

```
~> spans
<input
  onChange={e => {
    e.preventDefault();
    dispatch(change("file", e.target.files[0]));
    dispatch(change("file_obj", URL.createObjectURL(e.target.files[0])))
  }
  component="input"
  type="file"
  name="image"
  click-type="type1"
  className="picupload"
  multiple
  accept="image/*"
  id="files"
  style={{ padding: "20px 0" }}
/>
```



**Way to update  
form values**

# Redux-Form

## Example #3

The screenshot displays a React.js application interface. On the left, a sidebar titled "MAIN NAVIGATION" lists "Stock" (new), "Shop" (16), "Report" (3), and "Transactions" (12). The main content area shows a form for editing a stock item. The form fields are: "Name" (admin), "Stock" (10 PCS.), "Price" (100 \$), and a "Add Picture" button with a camera icon. At the bottom, a developer tools console (Elements tab selected) shows the following log entries:

```
▶ 99 messages
▶ 99 user me...
  ✘ No errors
  ⚠ No warnings
▶ 99 info
  ✍ No verbose
prev state ▶ {appReducer: {...}, loginReducer: {...}, stockReducer: {...}, stockEditReducer: {...}, transactionReducer: {...}, ...}
action ▶ {type: "@@redux-form/CHANGE", meta: {...}, payload: "adm"}
next state ▶ {appReducer: {...}, loginReducer: {...}, stockReducer: {...}, stockEditReducer: {...}, transactionReducer: {...}, ...}
▼ action @@redux-form/CHANGE @ 06:00:07.590
  prev state ▶ {appReducer: {...}, loginReducer: {...}, stockReducer: {...}, stockEditReducer: {...}, transactionReducer: {...}, ...}
  action ▶ {type: "@@redux-form/CHANGE", meta: {...}, payload: "admi"}
  next state ▶ {appReducer: {...}, loginReducer: {...}, stockReducer: {...}, stockEditReducer: {...}, transactionReducer: {...}, ...}
▼ action @@redux-form/CHANGE @ 06:00:07.691
  prev state ▶ {appReducer: {...}, loginReducer: {...}, stockReducer: {...}, stockEditReducer: {...}, transactionReducer: {...}, ...}
  action ▶ {type: "@@redux-form/CHANGE", meta: {...}, payload: "admin"}
  next state ▶ {appReducer: {...}, loginReducer: {...}, stockReducer: {...}, stockEditReducer: {...}, transactionReducer: {...}, ...}
```

# Redux-Form

## Usage

1. `npm i redux-form redux react-redux redux-thunk`
2. Configure redux-store
3. Wrap app with provider and configure store
4. App.js,
  - `import { Field, reduxForm } from "redux-form";`
  - `export default reduxForm({ ... })(App)`

# Redux-Form #1

## NPM

- npm i redux-form redux react-redux redux-thunk

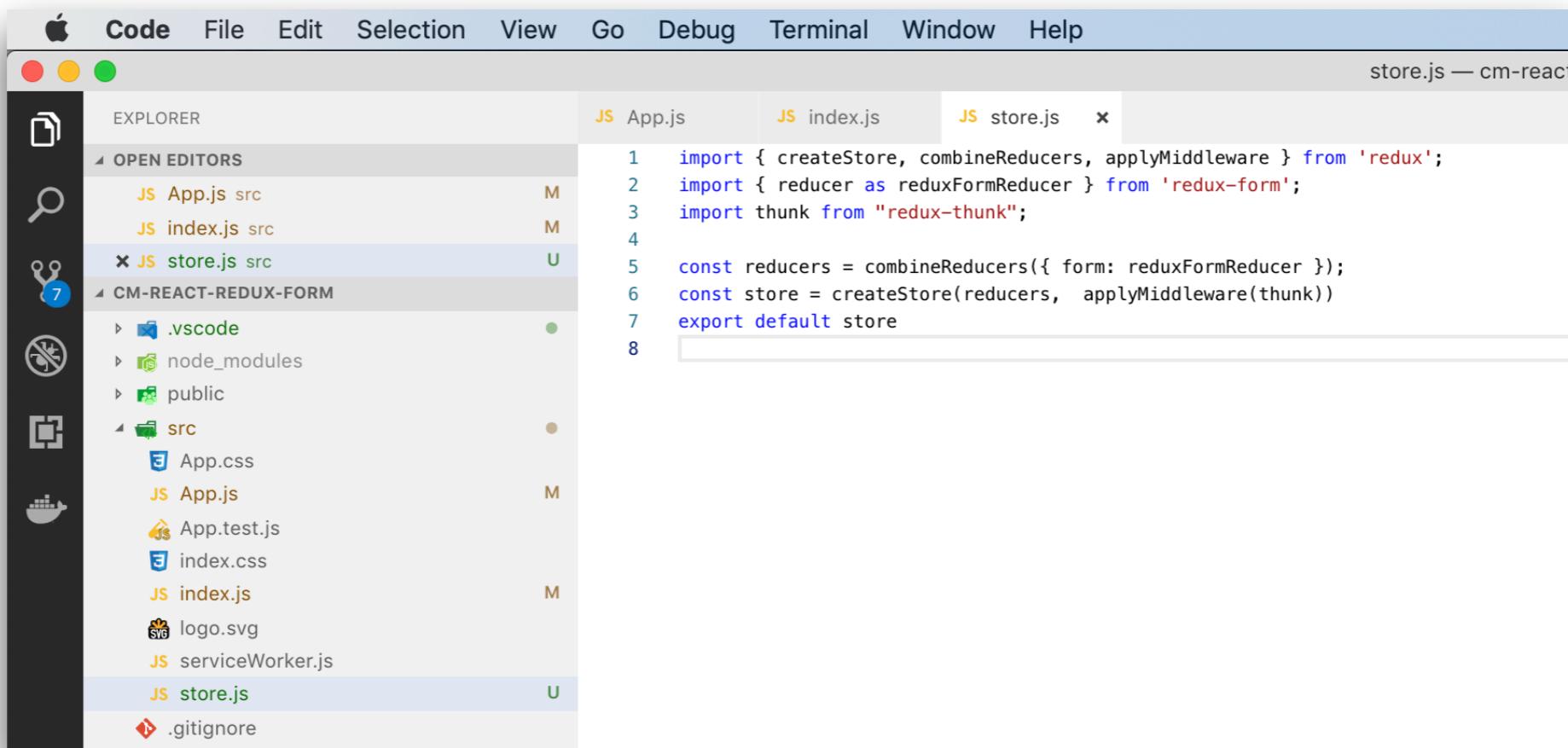
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

```
error: no devices/emulators found
chaiyasits-MacBook-Pro-3:cm-react-redux-form chaiyasit$ npm i redux-form redux react-redux redux-thunk
((:)) :: rollbackFailedOptional: verb npm-session 4337aee72a0e0f1c
```

# Redux-Form #2

## Store Configuration

- Configure redux-store



The screenshot shows the VS Code interface with the following details:

- File Explorer (Left):** Shows the project structure with files like App.js, index.js, store.js, App.css, App.test.js, index.css, index.js, logo.svg, serviceWorker.js, and .gitignore.
- Editor Tab Bar (Top):** Contains tabs for App.js, index.js, and store.js. The store.js tab is active.
- Code Editor (Right):** Displays the content of store.js:

```
1 import { createStore, combineReducers, applyMiddleware } from 'redux';
2 import { reducer as reduxFormReducer } from 'redux-form';
3 import thunk from "redux-thunk";
4
5 const reducers = combineReducers({ form: reduxFormReducer });
6 const store = createStore(reducers, applyMiddleware(thunk))
7 export default store
8
```

# Redux-Form #3

## Wrap app with Provider

- Wrap app with Provider

```
import React from "react";
import ReactDOM from "react-dom";
import "./index.css";
import App from "./App";
import { Provider } from "react-redux";
import store from "./store";
```

```
ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById("root")
);
```

# Redux-Form #4

## App.js

```
import React, { Component } from 'react'
import { Field, reduxForm } from "redux-form";

class App extends Component {
  render() {
    return (
      <div>

        </div>
      )
    }
}

export default reduxForm({
  form: "loginForm" // a unique identifier for this form
})(App);
```

# Redux-Form #5

## App.js

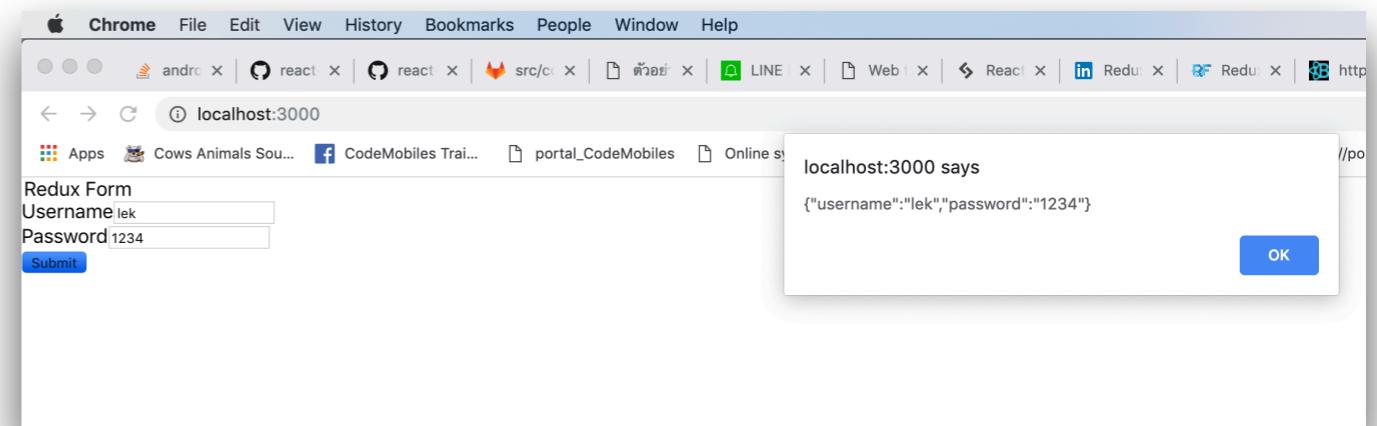
```
class App extends Component {

  onSubmit = formValues => {
    alert(JSON.stringify(formValues));
  }

  render() {
    const { handleSubmit } = this.props;

    return (
      <div>
        <form onSubmit={handleSubmit(this.onSubmit)}>
          <legend>Redux Form</legend>
          <div className="form-group">
            <label htmlFor="name">Username</label>
            <Field
              id="name"
              name="username"
              component="input"
              type="text"
              className="form-control"
              placeholder="Username"
            />
          </div>
        <div>
          <button type="submit" className="btn btn-primary">
            Submit
          </button>
        </div>
      </form>
    );
  }

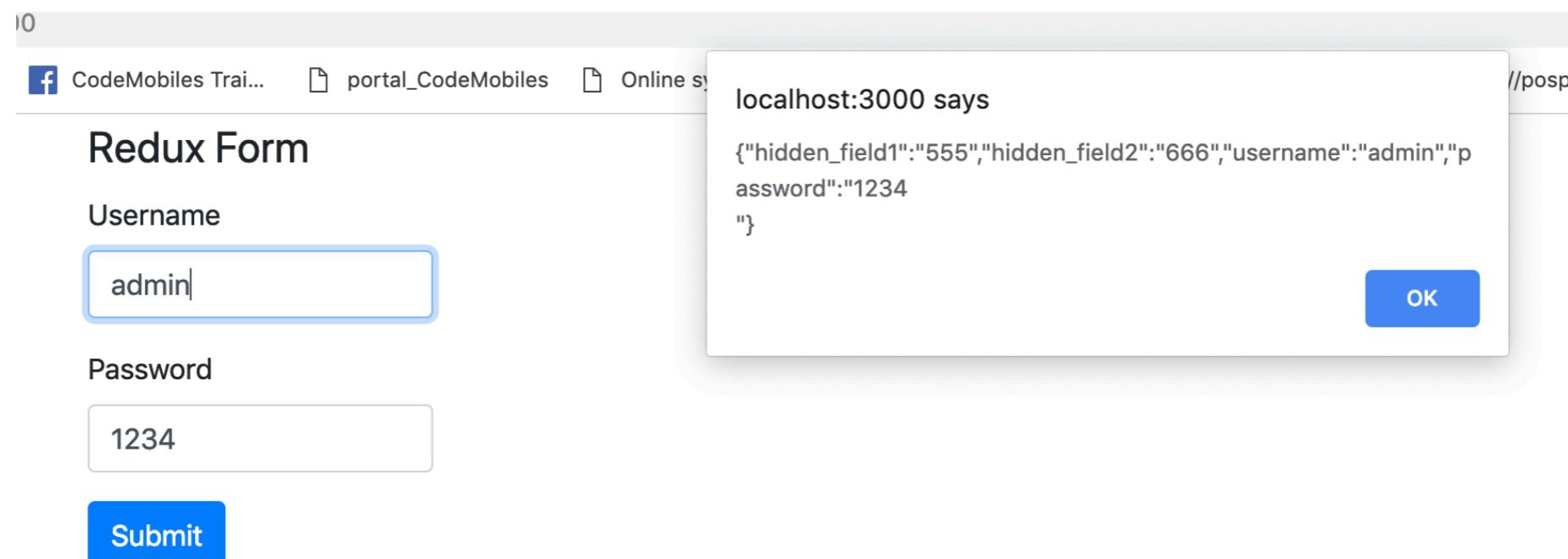
  export default reduxForm({
    form: "loginForm" // a unique identifier for this form
  })(App);
}
```



# Redux-Form Hidden Fields

```
componentDidMount(){
  const { change, dispatch } = this.props;
  dispatch(change("hidden_field1","555"))
  dispatch(change("hidden_field2","666"))
}
```

```
componentDidMount(){
  const { change} = this.props;
  change("hidden_field1","555")
  change("hidden_field2","666")
}
```



# Redux-Form Default Value

```
export default reduxForm({  
  fields: {  
    hidden_field1: "1234",  
    hidden_field2: "1234"  
  },  
  form: "loginForm",  
  initialValues: {  
    username: "admin",  
    password: "1234",  
  },  
})(App);
```

## Redux Form

Username

Password

Submit

# Redux-Form

Access Specific Field's Value  
Using **formValueSelector #1**

```
App = reduxForm({
  fields: {
    hidden_field1: "1234",
    hidden_field2: "1234"
  },
  form: "loginForm",
  initialValues: {
    username: "admin",
    password: "1234",
  },
})(App);
```

```
const selector = formValueSelector('loginForm')
App = connect(
  state => {
    // can select values individually
    return {
      username : selector(state, 'username'),
      password : selector(state, 'password')
    }
  }
)(App)

export default App
```

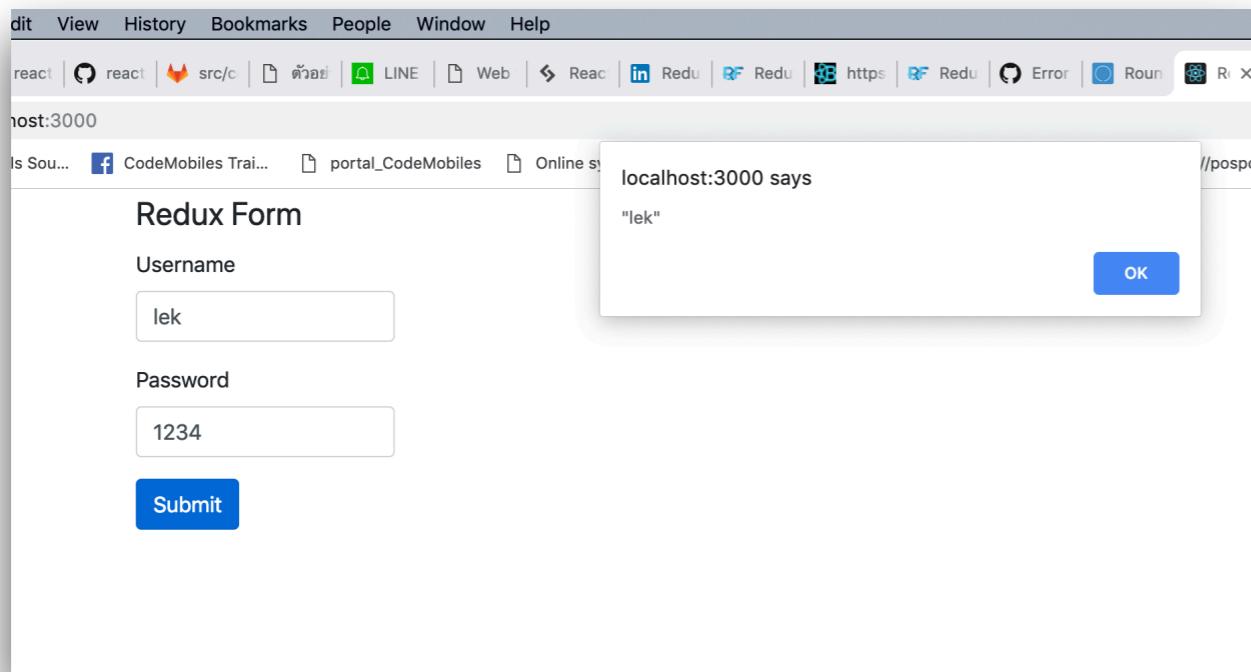
```
<div className="form-group">
  <label htmlFor="name">Username</label>
  <Field
    id="name"
    name="username"
    component="input"
    type="text"
    className="form-control"
    placeholder="Username"
  />
</div>

<div className="form-group">
  <label htmlFor="password">Password</label>
  <Field
    id="password"
    name="password"
    component="input"
    type="text"
    className="form-control"
    placeholder="Password"
  />
</div>
```

# Redux-Form

Access Specific Field's Value  
Using **formValueSelector #2**

```
alert(JSON.stringify(this.props.username)) // using selector to get individual field's value
alert(JSON.stringify(this.props.password))
```



# Redux-Form

## Custom Field (Component)

```
<Field
  id="name"
  label="Username"
  name="username"
  component={renderField}
  type="text"
  className="form-control"
  placeholder="Username"
/>

<Field
  id="password"
  name="password"
  component={renderField}
  label="Password"
  type="text"
  className="form-control"
  placeholder="Password"
/>
```

```
const renderField = ({ input, label, type, meta: { touched, error, warning } }) => (
  <div className="form-group">
    <label style={{color: 'red'}} htmlFor="name">{label}</label>
    <div>
      <input {...input} placeholder={label} type={type}/>
      {touched && ((error && <span>{error}</span>) || (warning && <span>{warning}</span>))}
    </div>
  </div>
)
```

# Redux-Form

## Custom Field (Component)

```
<Field
  id="name"
  label="Username"
  name="username"
  component={renderField}
  type="text"
  className="form-control"
  placeholder="Username"
/>
```

```
<Field
  id="password"
  name="password"
  component={renderField}
  label="Password"
  type="text"
  className="form-control"
  placeholder="Password"
/>
```

```
const renderField = ({ input, label, type, meta: { touched,
error, warning } }) => (
  <div className="form-group">
    <label style={{color: 'red'}} htmlFor="name">{label}</
label>
    <div>
      <input {...input} placeholder={label} type={type}/>
      {touched && ((error && <span>{error}</span>) ||
      (warning && <span>{warning}</span>))
    </div>
  </div>
)
```

### Redux Form

Username

lek

Password

1234

Submit

# Redux-Form

## Field Validation

```
import React from 'react'
import { Field, reduxForm } from 'redux-form'

const required = value => value ? undefined : 'Required'
const maxLength = max => value =>
  value && value.length > max ? `Must be ${max} characters or less` : undefined
const maxLength15 = maxLength(15)
const number = value => value && isNaN(Number(value)) ? 'Must be a number' : undefined
const minValue = min => value =>
  value && value < min ? `Must be at least ${min}` : undefined
const minValue18 = minValue(18)
const email = value =>
  value && !/^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}$/.test(value) ?
    'Invalid email address' : undefined
const tooOld = value =>
  value && value > 65 ? 'You might be too old for this' : undefined
const aol = value =>
  value && /.+@aol\.com/.test(value) ?
    'Really? You still use AOL for your email?' : undefined

const renderField = ({ input, label, type, meta: { touched, error, warning } }) => (
  <div>
    <label>{label}</label>
    <div>
      <input {...input} placeholder={label} type={type}/>
      {touched && ((error && <span>{error}</span>) || (warning && <span>{warning}</span>))}
    </div>
  </div>
)
```

```
<form onSubmit={handleSubmit}>
  <Field name="username" type="text"
    component={renderField} label="Username"
    validate={[ required, maxLength15 ]}
  />
  <Field name="email" type="email"
    component={renderField} label="Email"
    validate={email}
    warn={aol}
  />
  <Field name="age" type="number"
    component={renderField} label="Age"
    validate={[ required, number, minValue18 ]}
    warn={tooOld}
  />
```

# Formik

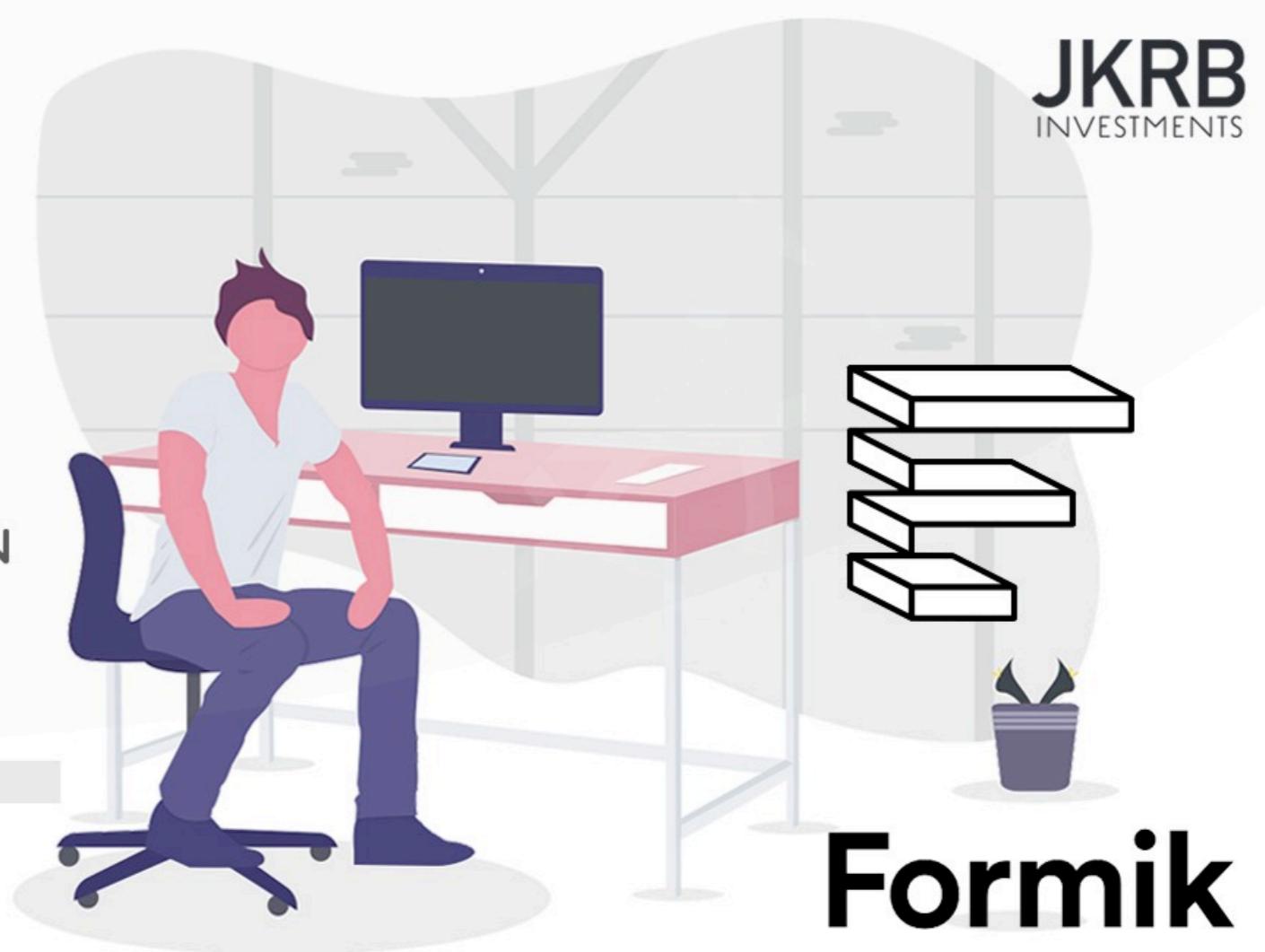
<FORMIK />

<FORM />

<FIELD />

VALIDATION

ONSUBMIT



JKRB  
INVESTMENTS

# Formik

# Why? Formik

- **Managing form state** — done automatically and locally. Packages like Redux Forms tie your form state to your state tree. This means that your top-level reducer is called on every keystroke. This is unnecessary overhead and bad design. Form state should be kept local.
- **Validating a form** — using Formik's validation handlers and (*optionally*) Yup. We are free to handle validation as we please with Formik, however, instead of reinventing the wheel, Formik also supports [Yup](#): the most widely adopted object validation solution for React, directly into its handlers. ([Read more about Yup here](#)).
- **Handling form submission** — easy value parsing and error formatting, via handler functions passed into Formik.

# Install Formik

```
npm i formik
```

For this article we will be importing the following Formik components into our project:

```
import { Formik, FormikProps, Form, Field } from 'formik';
```

# Install Formik

- `<Formik />` is our top most component. Through its props we will configure some of Formik's built in form handlers, which essentially allow our form to function. We will also utilise the `render` prop. **Our entire form is constructed via a function passed into this prop.**
- `FormikProps` is passed through the above `render` prop function, which **gives us access to the state of our form** — values, errors, whether an input has been touched (selected for the first time), whether the form is submitting, and so forth.
- `<Form />` is the simplest of our imports here, which we use to simply wrap our fields with. Hence, all `<Field />` components must be children of `<Form />`.
- `<Field />` will be used to make our elements compatible with Formik. In other words, without the Field component, our input elements will know nothing about Formik. `<Field />` actually provides an input text box as its default input method, as well as a select dropdown and textarea options.

# Structuring a Formik Form

```
import { Formik, FormikProps, Form, Field } from 'formik';

export class MyForm extends React.Component {

  handleSubmit = (values, {
    props = this.props,
    setSubmitting
  }) => {
    //process form submission here

    //done submitting, set submitting to false
    setSubmitting(false);

    return;
  }

  render() {

```

```
    return(
      <Formik
        initialValues={{
          first_name: '',
          email: '',
          gender: ''
        }}
        validate={(values) => {
          let errors = {};

          //check if my values have errors
          return errors;
        }}
        onSubmit={handleSubmit}
        render={formProps: FormikProps => {

```

|

```
          return(
            <Form>
              <Field ... />
              <Field ... />
              <Field ... />

              <button
                type="submit"
                disabled={formProps.isSubmitting}>
                Submit Form
              </button>
            </Form>
          );
        }
      />);
    }
  }
}
```

# Validating a Formik Form

```
<div className="box-body" style={{ marginTop: 30 }}>
  <Formik
    validate={values => {
      let errors = {};
      if (!values.name) errors.name = "Enter name";
      if (!values.stock) errors.stock = "Enter stock";
      if (!values.price) errors.price = "Enter price";
      return errors;
    }}
    initialValues={{ name: "", stock: 10, price: 90 }}
    onSubmit={(values, { setSubmitting }) => {
      let formData = new FormData();
      formData.append("name", values.name);
      formData.append("price", values.price);
      formData.append("stock", values.stock);
      formData.append("image", values.file);
      dispatch(stockActions.addProduct(formData, props.history));
      setSubmitting(false);
    }}
  </Formik>
</div>
```

# Validating a Formik Form

## Create Stock

Name

Enter name

Price

Enter price

Stock

Enter stock

Add Picture  No file chosen

CREATE CANCL

# React Router V4

*Page Navigation*



# Page Navigation

```
npm i react-router-dom
```

<https://github.com/ReactTraining/react-router/tree/master/packages/react-router-dom>

# Keywords

- **Router** : root component of all routes
- **Route** : link match path and component
- **Link** : like <a>
- **History** : help to jump to previous routes
- **Location** : return current path
- **Match** : contain passed-in parameters
- **withRouter**: HOC to provide H / L / M



# BrowserRouter

```
...  
import { BrowserRouter as Router } from 'react-router-dom'  
  
ReactDOM.render(  
  
  <Router>  
    <div>  
      <!-- -->  
    </div>  
  </Router>,  
  
  document.getElementById('app')  
)
```

\* A BrowserRouter component can only have one child element, so we wrap all we're going to add in a `div` element.

# Link

~ <a>

```
ReactDOM.render(  
  <Router>  
    <div>  
      <aside>  
        <Link to={`/dashboard`}>Dashboard</Link>  
        <Link to={`/about`}>About</Link>  
      </aside>  
      <!-- -->  
    </div>  
  </Router>,  
  document.getElementById('app')  
)
```

\* The **Link** component is used to trigger new routes. You import it from `react-router-dom`, and you can add the Link components to point at different routes, with the `to` attribute:

# Route

## ~ href

```
ReactDOM.render(  
  <Router>  
    <div>  
      <aside>  
        <Link to={`/`}>Dashboard</Link>  
        <Link to={`/about`}>About</Link>  
      </aside>  
  
      <main>  
        <Route exact path="/" component={Dashboard} />  
        <Route path="/about" component={About} />  
      </main>  
    </div>  
  </Router>,  
  document.getElementById('app')  
)
```

\* **Route** is like outlet component that will show or hide depending on current path

# Exact

```
<main>
  <Route exact path="/" component={Dashboard} />
  <Route path="/about" component={About} />
</main>
```

**Exact** attribute. Without this, path="/" would also match /about, since / is contained in the route.

# Multiple Paths

```
<Route path="/(about|who)/" component={Dashboard} />
```

\* You can have a route respond to multiple paths simply using a regex, because path can be a regular expressions string:

# Passing Parameters

```
<Route exact path="/post/:id" component={Post} />
```

```
// In Post component
render (
  <div>
    <h2>Post #{this.props.match.params.id}</h2>
    ...
  </div>
)
//...
```

# History

Return to Previous Path

```
export const addProduct = (history, data) => {
  return dispatch => {
    httpClient.post(server.PRODUCT_URL, data).then(result => {
      history.goBack();
    });
  };
};
```

# withRouter HOC #1

Not in Route ?

```
<Router>
<div>
<Header/>
<Route path="/login" component={Login} />
```

Problem? Header component cannot access history, match, location because it is not inside of Route

withRouter can help

# withRouter HOC #2

Not in Route ?

```
<Router>
  <div>
    <Header/>
    <Route path="/login" component={Login} />
```

```
export default connect(
  mapStateToProps,
  actions
)(withRouter(Header));
```

You can get access to the `history` object's properties and the closest `<Route>`'s `match` via the `withRouter` higher-order component. `withRouter` will pass updated `match`, `location`, and `history` props to the wrapped component whenever it renders.

# Nested Route

```
const Topics = ({ match })=> (
  <div>
    <ul>
      <li><Link to={`${match.url}/angular`}>Angular</Link></li>
      <li><Link to={`${match.url}/react`}>React</Link></li>
      <li><Link to={`${match.url}/android`}>Android</Link></li>
    </ul>

    <hr/>

    <Route path={`${match.url}/:topicId`} render={({match}) =>
      (<h1>{match.params.topicId}</h1>) } />
  </div>
)
```

# For More Info

## React Router

The screenshot shows a browser window displaying the React Router documentation for the `<Link>` component. The URL in the address bar is `https://reacttraining.com/react-router/web/api/Link`. The page has a sidebar on the left with sections for REACT TRAINING / REACT ROUTER, EXAMPLES, GUIDES, and API. The main content area is titled `<Link>` and describes it as providing declarative, accessible navigation around your application. It details the `to` prop (a string or object), the `replace` prop (a boolean), and the `innerRef` prop (a function). The `to` prop is described with examples of pathname, search, hash, and state.

<https://reacttraining.com/react-router/web/api/Link>

# Protected Routes

# Router Guard



# Protected Route

```
<PrivateRoute path="/stock" component={Stock} />
```

```
// Protected Route
const PrivateRoute = ({ component: Component, ...rest }) => (
  <Route {...rest} render={(props) => (
    isLoggedIn() === true
      ? <Component {...props} />
      : <Redirect to='/login' />
  )} />
)
```

*PrivateRoute* component which would render the component only if the user was authenticated. Something like this

From props, get the Component prop and then all other props given to you, and rename props to rest so you can avoid naming issues with the props passed to the Route renderfunction

# Protected Route #1

```
// Protected Route
const PrivateRoute = ({ component: Component, ...rest }) => (
  <Route {...rest} render={(props) => (
    isLoggedIn() === true
      ? <Component {...props} />
      : <Redirect to='/login' />
  )} />
)
```

```
// Requirement 1.
// It has the same API as <Route />
```

```
const PrivateRoute = ({ component: Component, ...rest }) => (
)
```

# Protected Route #2

```
// Protected Route
const PrivateRoute = ({ component: Component, ...rest }) => (
  <Route {...rest} render={(props) => (
    isLoggedIn() === true
      ? <Component {...props} />
      : <Redirect to='/login' />
  )} />
)
```

// Requirement 2.  
// It renders a <Route /> and passes all the props through to it.

```
const PrivateRoute = ({ component: Component, ...rest }) => (
  <Route {...rest} render={() =>
})
```

# Protected Route #3

```
// Protected Route
const PrivateRoute = ({ component: Component, ...rest }) => (
  <Route {...rest} render={(props) => (
    isLoggedIn() === true
      ? <Component {...props} />
      : <Redirect to='/login' />
  )} />
)
```

```
// Requirement 3.
// It checks if the user is authenticated, if they are,
// it renders the "component" prop. If not, it redirects
// the user to /login.
```

# Change Route in ES16

<https://reacttraining.com/react-router/web/guides/quick-start>

# Persistent Storage

## *LocalStorage*

# LocalStorage

## Store

```
LocalStorage.setItem('username', username)  
LocalStorage.setItem('password', password)
```

## Restore

```
let regUsername = LocalStorage.getItem('username')  
let regPassword = LocalStorage.getItem('password')
```

# Axios

**axios**

Promise-based HTTP Client

browser & node.js

replaces vue-resource

in essence an AJAX program

save us time and trouble

**POST & GET**

# HTTP-Client API

npm i -save axios

**Axios** (Recommended)

```
// Make a request for a user with a given ID
axios.get('/user?ID=12345')
  .then(function (response) {
    console.log(response);
  })
  .catch(function (error) {
    console.log(error);
  });

```

# axios Features

## axios

---

npm v0.18.0 build passing coverage 94% install size 387 kB downloads 10M/m chat on gitter code helpers 30

Promise based HTTP client for the browser and node.js

## Features

---

- Make [XMLHttpRequests](#) from the browser
- Make [http](#) requests from node.js
- Supports the [Promise API](#)
- Intercept request and response
- Transform request and response data
- Cancel requests
- Automatic transforms for JSON data
- Client side support for protecting against [XSRF](#)

# axios Example

[http://codemobiles.com/adhoc/youtubes/index\\_new.php?username=admin&password=password&type=foods](http://codemobiles.com/adhoc/youtubes/index_new.php?username=admin&password=password&type=foods)

```
componentDidMount() {
  this.feedYoutbes();
}

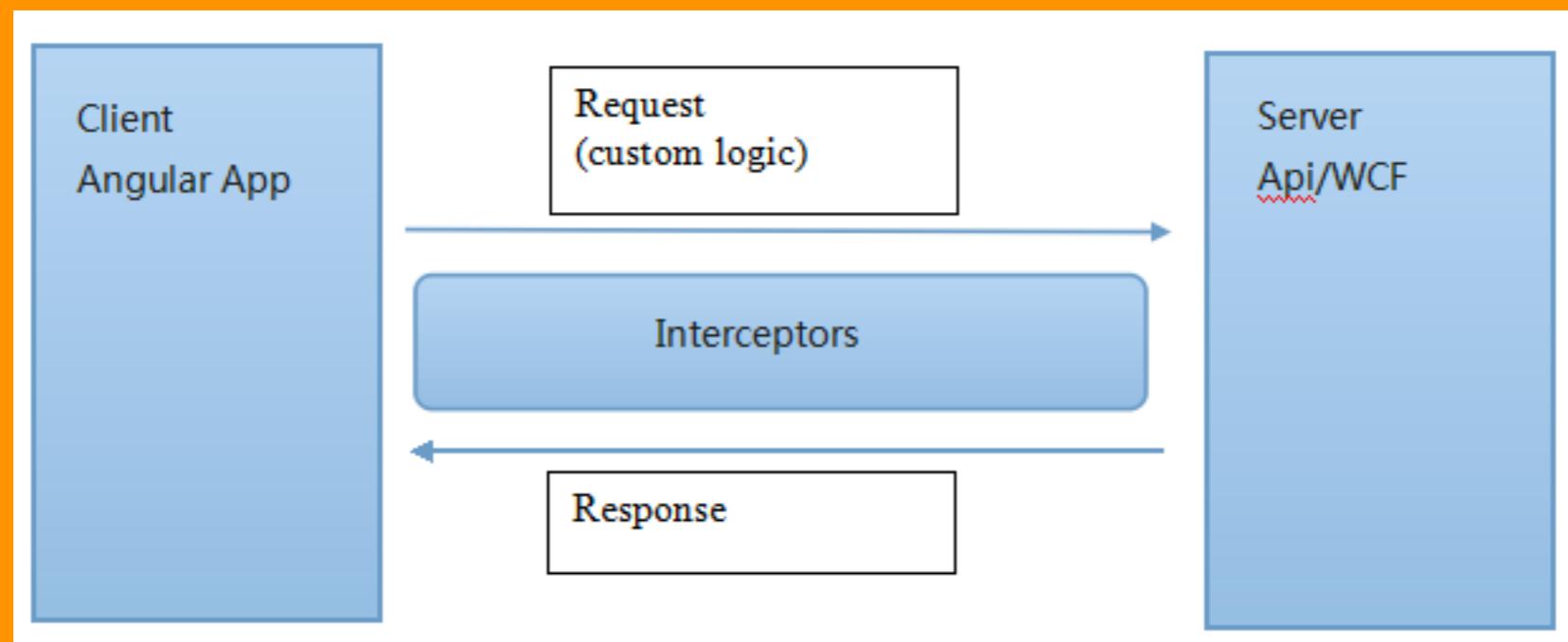
feedYoutbes() {
  const data = { ...this.mAccount, type: "foods" };
  console.log(`Data: ` + JSON.stringify(data));

  const url = "http://codemobiles.com/adhoc/youtubes/index_new.php";
  axios
    .get(url, { params: data })
    .then(response => {
      this.setState({ youtubes: response.data.youtubes });
      console.log(JSON.stringify(this.state));
    })
    .catch(error => {
      console.log(error);
    });
}

{
  "youtubes": [
    {
      "id": "E3u2YoGWZ9k",
      "title": "Laura in the Kitchen",
      "subtitle": "Homemade Sicilian Pizza Reci",
      "avatar_image": "https://yt3.ggpht.com/-k...",
      "youtube_image": "http://img.youtube.com/..."
    },
    {
      "id": "F5MqYWrxYs",
      "title": "Scoff",
      "subtitle": "Bubble & Squeak | Good Food",
      "avatar_image": "https://yt3.ggpht.com/-k...",
      "youtube_image": "http://img.youtube.com/..."
    },
    {
      "id": "gZuDMKXWU_E",
      "title": "Munchies",
      "subtitle": null,
      "avatar_image": null,
      "youtube_image": null
    }
  ]
}
```

# HTTP Axios

## Interceptor Feature



# Axios

## With Interceptor

### HttpClient.js

```
import React, { Component } from 'react';
import { AsyncStorage } from 'react-native';

import axios from 'axios'
import join from 'url-join'

var isAbsoluteURLRegex = /^(?:\w+:)\/\/$/;

axios.interceptors.request.use(async (config)=> {
  if (!isAbsoluteURLRegex.test(config.url)) {
    const jwtToken = LocalStorage.getItem("token")
    if (jwtToken != null) {
      config.headers = { 'x-access-token': jwtToken }
    }
    config.url = join('http://10.0.0.33:8082/api/v1', config.url);
  }
  return config;
});

export const httpClient = axios
```

### FeedScreen.js

```
import {httpClient} from './HttpClient'
...
httpClient
  .get('/feed')
  .then(result=>{
    Alert.alert(JSON.stringify(result.data))
  })
```

# Axios

## Without Interceptor

**Manually customize all requests 1,2,3,4 -** not good for maintenance

```
async feed1() {
  this.state = { feedData: "loading.." }
  const token = LocalStorage.getItem("token")

  axios.get('http://10.0.0.33:8082/api/v1/feed',
    { headers: { 'x-access-token': token } })
    .then(response => {
      const result = response.data.result
      this.setState({ feedData: result })
    })
    .catch(error => {
      Alert.alert(JSON.stringify(error))
      console.log(error);
    });
}
```

# Http Interception

```
axios.interceptors.request.use(async (config) => {
  if (!isAbsoluteURLRegex.test(config.url)) {
    const userToken = localStorage.getItem(server.TOKEN_KEY)
    if (userToken) {
      config.headers = { 'x-access-token': userToken }
    }
    config.url = join(apiUrl, config.url)
  }
  config.timeout = 10000 // 10 Second
  return config
})

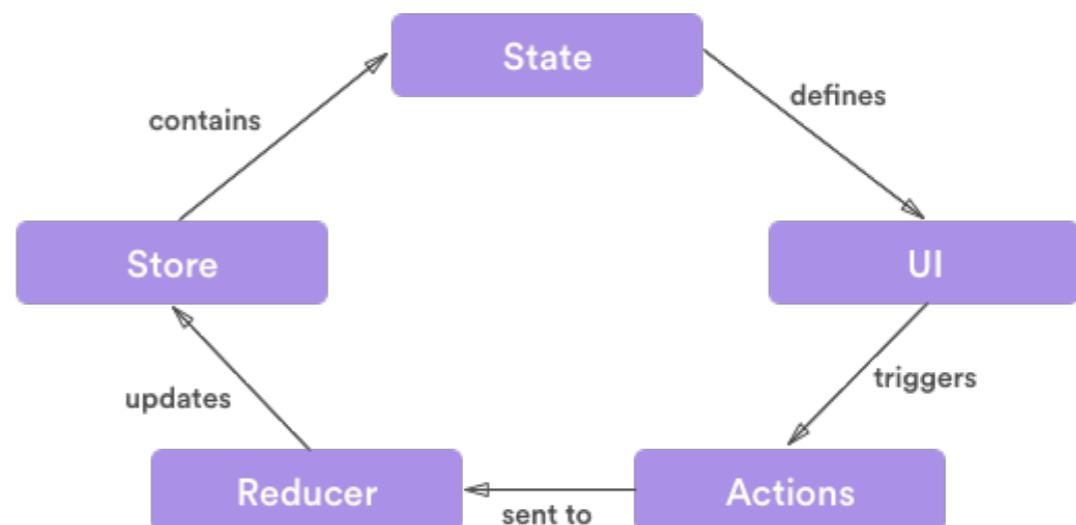
axios.interceptors.response.use((response) => {
  return response
}, error => {
  console.log(JSON.stringify(error, undefined, 2))
  if (axios.isCancel(error)) {
    return Promise.reject(error)
  } else {
    return Promise.reject(error)
}
})
```

# Redux

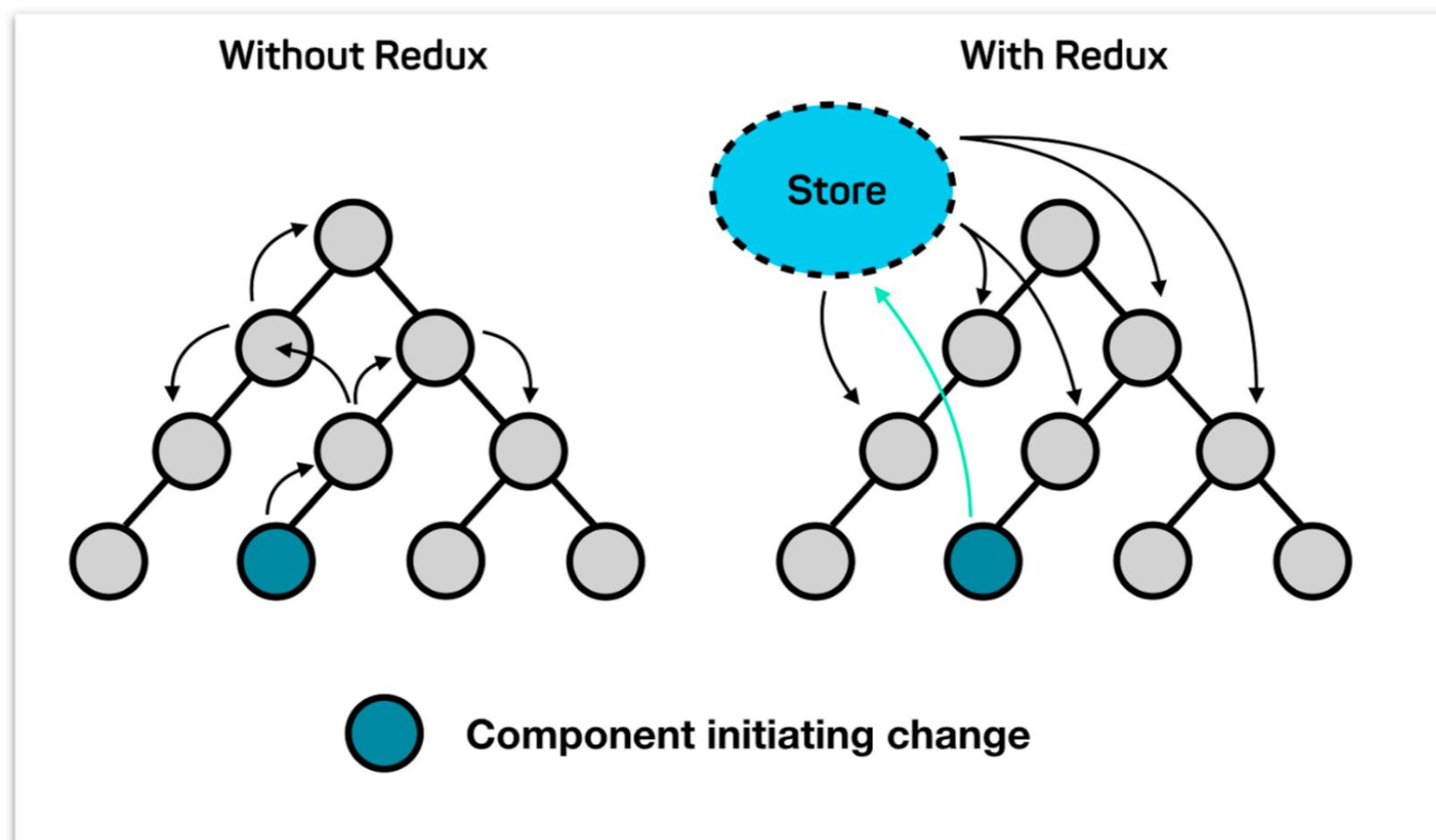
# What is Redux?

## In short

- **Redux** helps use to separate (logic) actions, state management, and presentation in good manner.
  - **Without** redux, sometimes, some programmers tends mix all (complex logics and presentation) together. That is hard to maintenance.



# Redux-Store



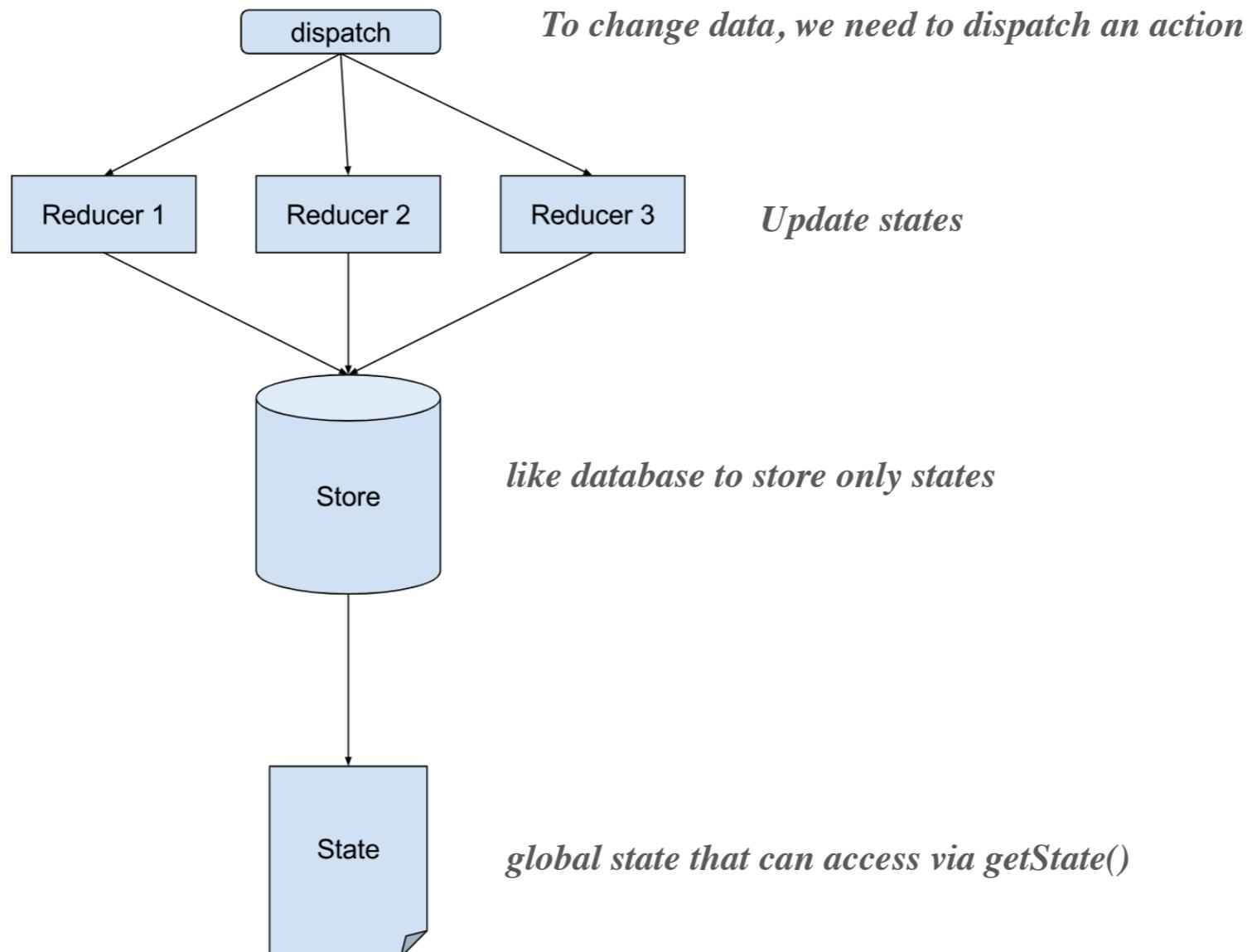
# What is Redux?

- **Redux** is a library that help management of updating state in React.
- **npm install --save react-redux**
- **store.js** : provide a storage to record global states that can share between components
- **reducer.js** : provide functions that can be called by each component to perform some actions. In these functions will update state by cloning a new state that allows any component subscribe the state change can listen

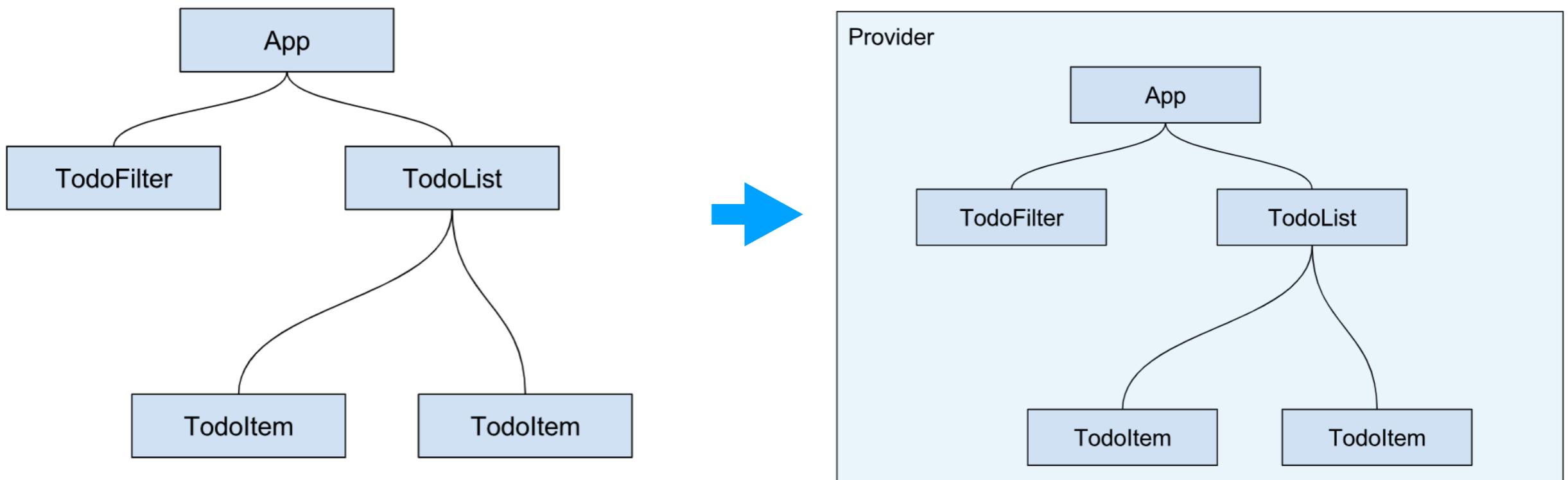


# Redux

# What is Redux?

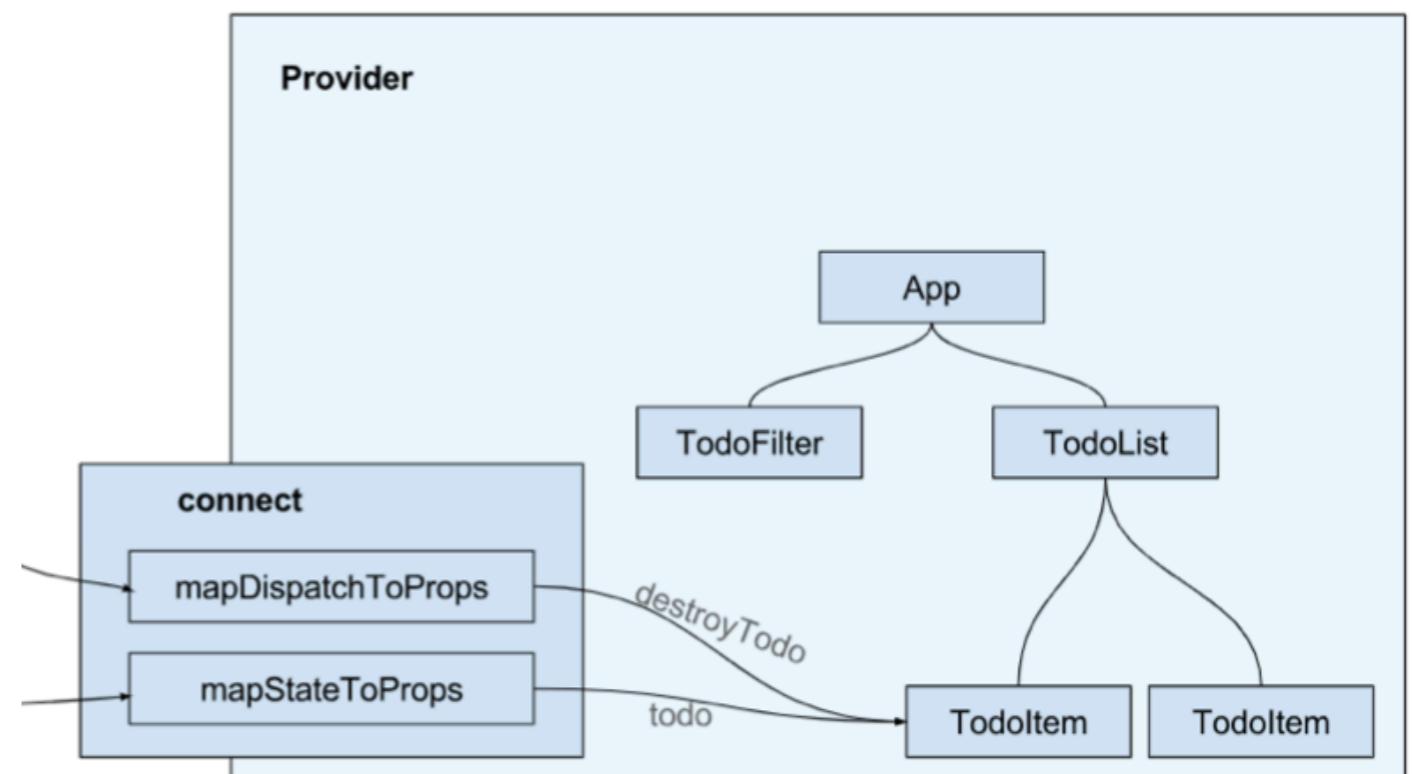


# Provider Redux

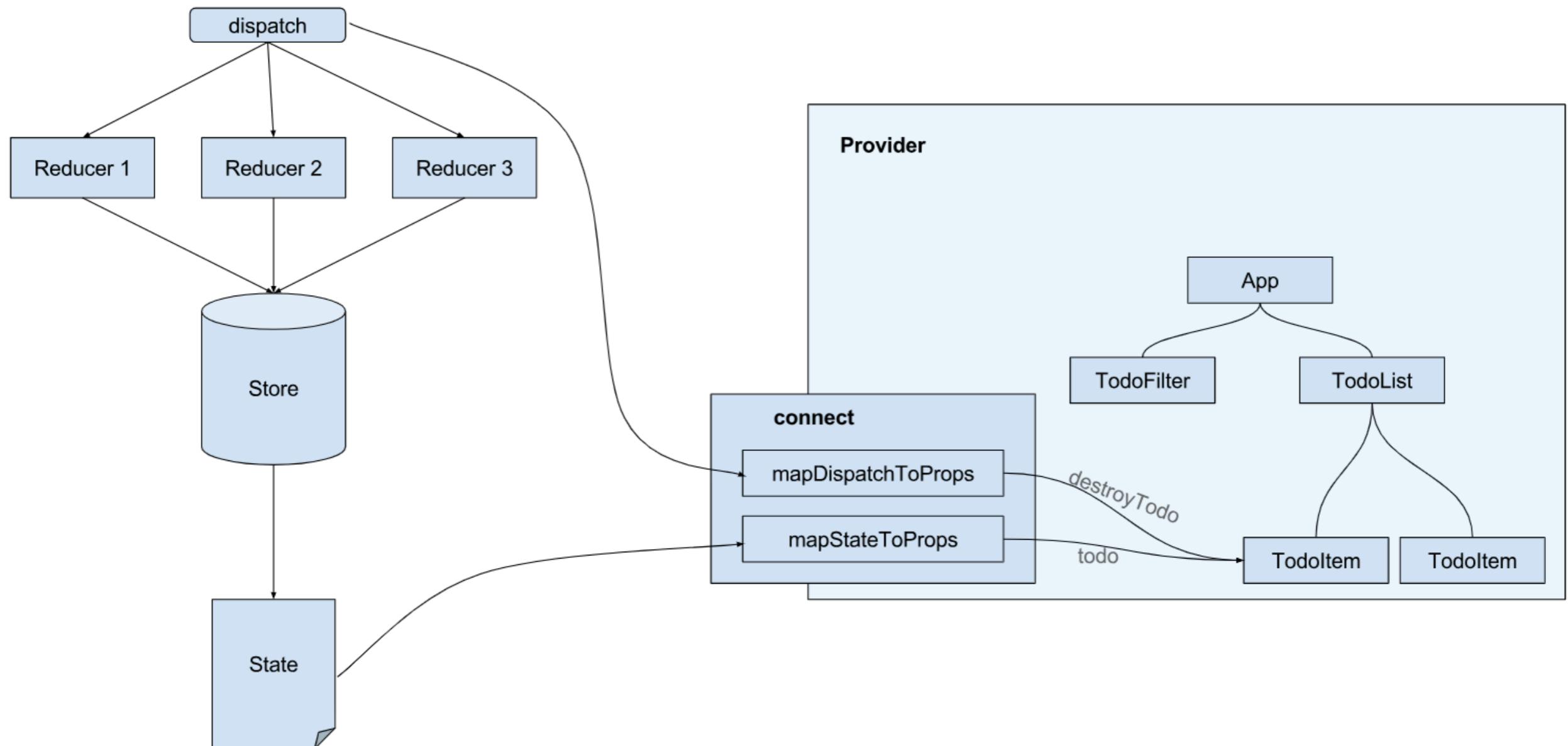


# Connect Redux

```
....  
  
const mapStateToProps = state => {  
  return {  
    todo: state.todos[0]  
  }  
}  
  
const mapDispatchToProps = dispatch => {  
  return {  
    destroyTodo: () =>  
      dispatch({  
        type: 'DESTROY_TODO'  
      })  
  }  
}  
  
export default connect(  
  mapStateToProps,  
  mapDispatchToProps  
)  
(TodoItem)
```

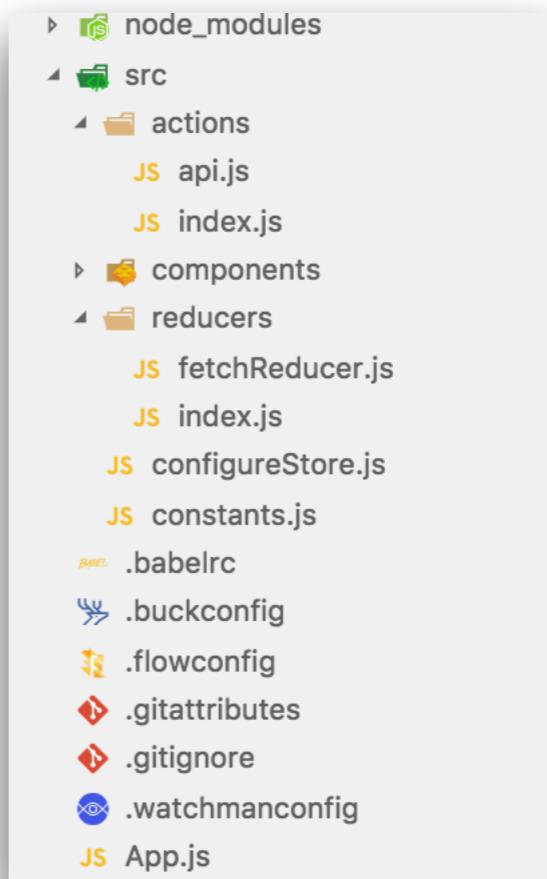


# Connect Redux



# Redux Setup

```
$ npm i -save redux react-redux redux-thunk  
$ mkdir src  
$ mkdir src/reducers, src/actions, src/components  
$ touch src/app.js, src/reducers/index.js, src/actions/index.js  
$ touch src/constant.js, src/configureStore.js
```



# What can redux do?

- Whenever you want to replace the state in the store, you dispatch a **action**
- **action** is caught by one or more reducers.
- **reducer** listen to the action's type change and return a new state that combines the old state and the action's data, to allow re-render the presentational components.
- **store** subscribers are notified that there is a new state.

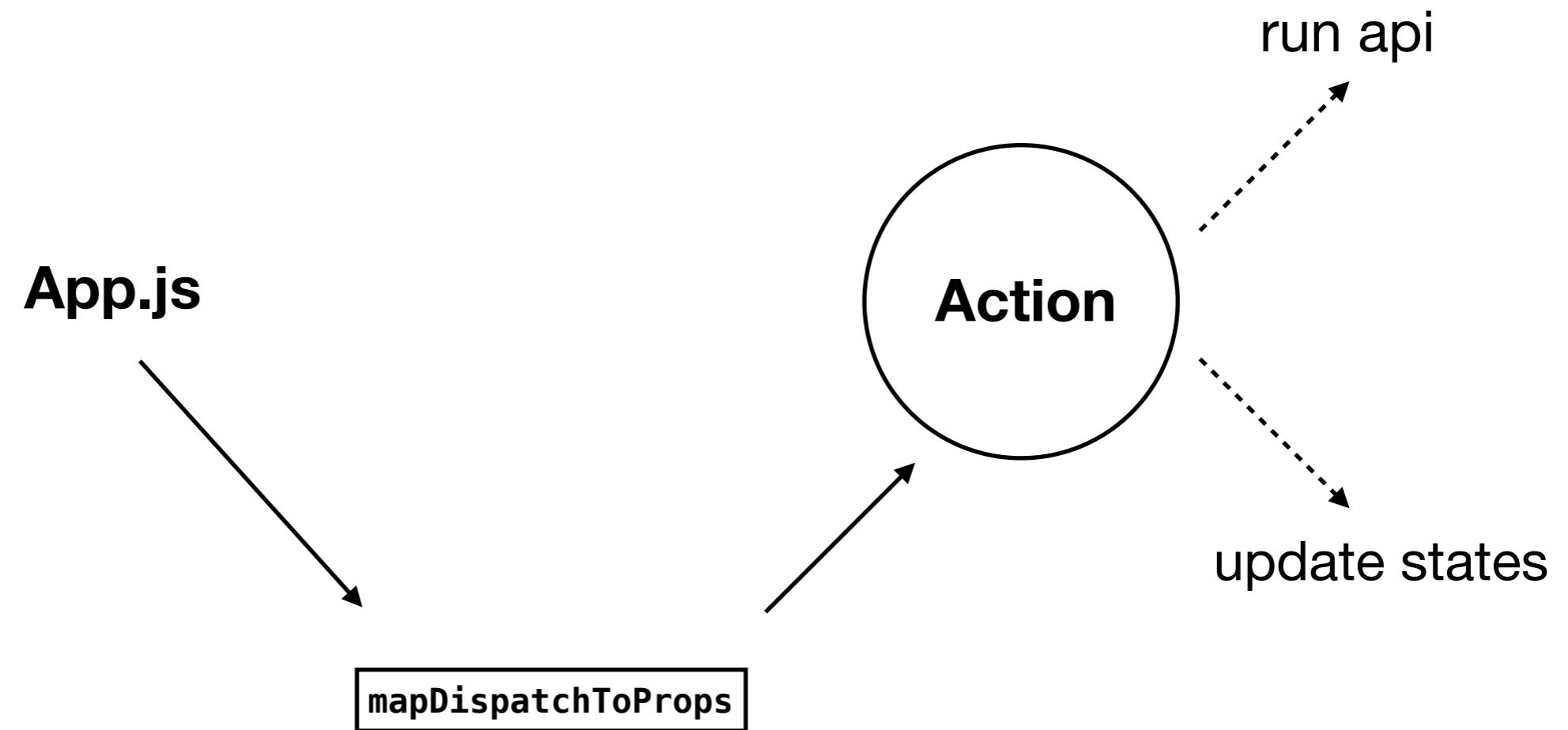
# Roles of components /container / store

- **Store** - holds the state, when a new action arrives run the dispatch -> middleware -> reducer pipeline, and notified subscribers when the state is replaced by a new one.
- **Components** - dump view parts which are not aware of the state directly, Also known as presentational components
- **Container** - pieces of the view that are aware of the state using react-redux. Also known as smart components.

# Redux vs React-Redux

- **redux** - like flow with a single store, that can be used in whatever environment you like including vanilla js, react, angular 1/2, etc....
- **react-redux** - binding between redux and react, that create containers (smart components) that listen to the store's state changes, prepare the props for and re-render the presentational components.
- **redux-thunk** - middleware that allows you to write action creators that return a function instead of an action. The thunk can be used to delay the dispatch of an action, generally, it is used in asynchronous api call such as loading data from network.

# Redux Workshop



```
const mapDispatchToProps = {
  fetchData
};
export default
connect(mapStateToProps,
mapDispatchToProps)(App)
```

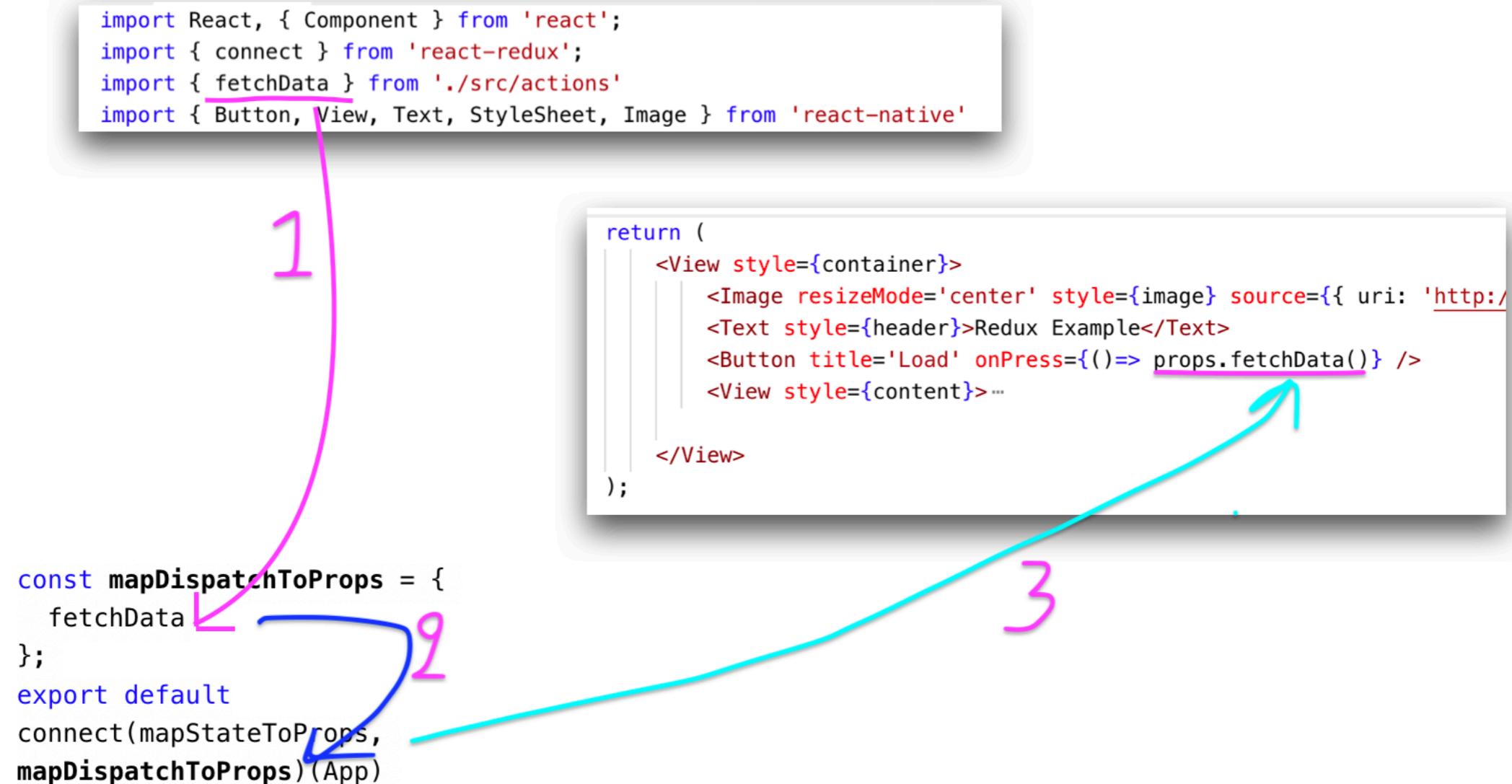
# Redux Shortcut Keyboards

## Redux

Prefix	Method
rxaction→	redux action template
rxconst→	export const \$1 = '\$1'
rxreducer→	redux reducer template
rxselect→	redux selector template

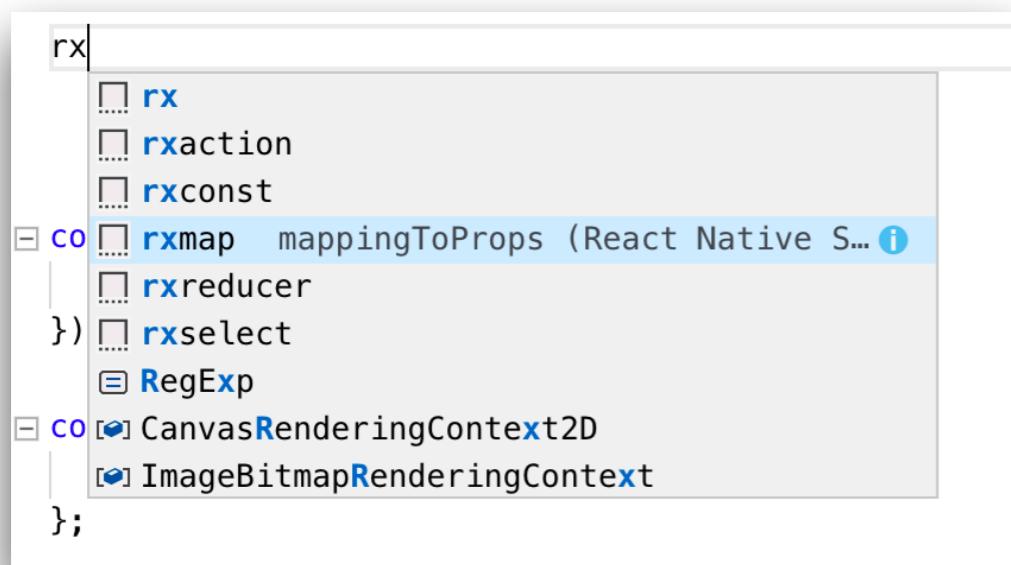
# Redux Workshop

Component <-> mapDispatchToProps <-> Action

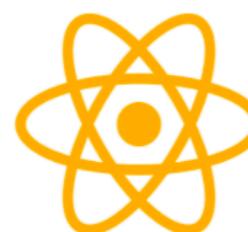


# Redux

**rx-xxx** to automate a snippet code



```
const mapStateToProps = (state) => ({
  );
  const mapDispatchToProps = (dispatch) => ({
  );
}
```



## React Native Snippet jundat95.react-native-snippet

Jundat95 | ↗ 140,186 | ★★★★★ | Repository | License

React Native, Typescript React Native, StyleSheet, Redux Snippet

[Disable ▾](#) [Uninstall](#)

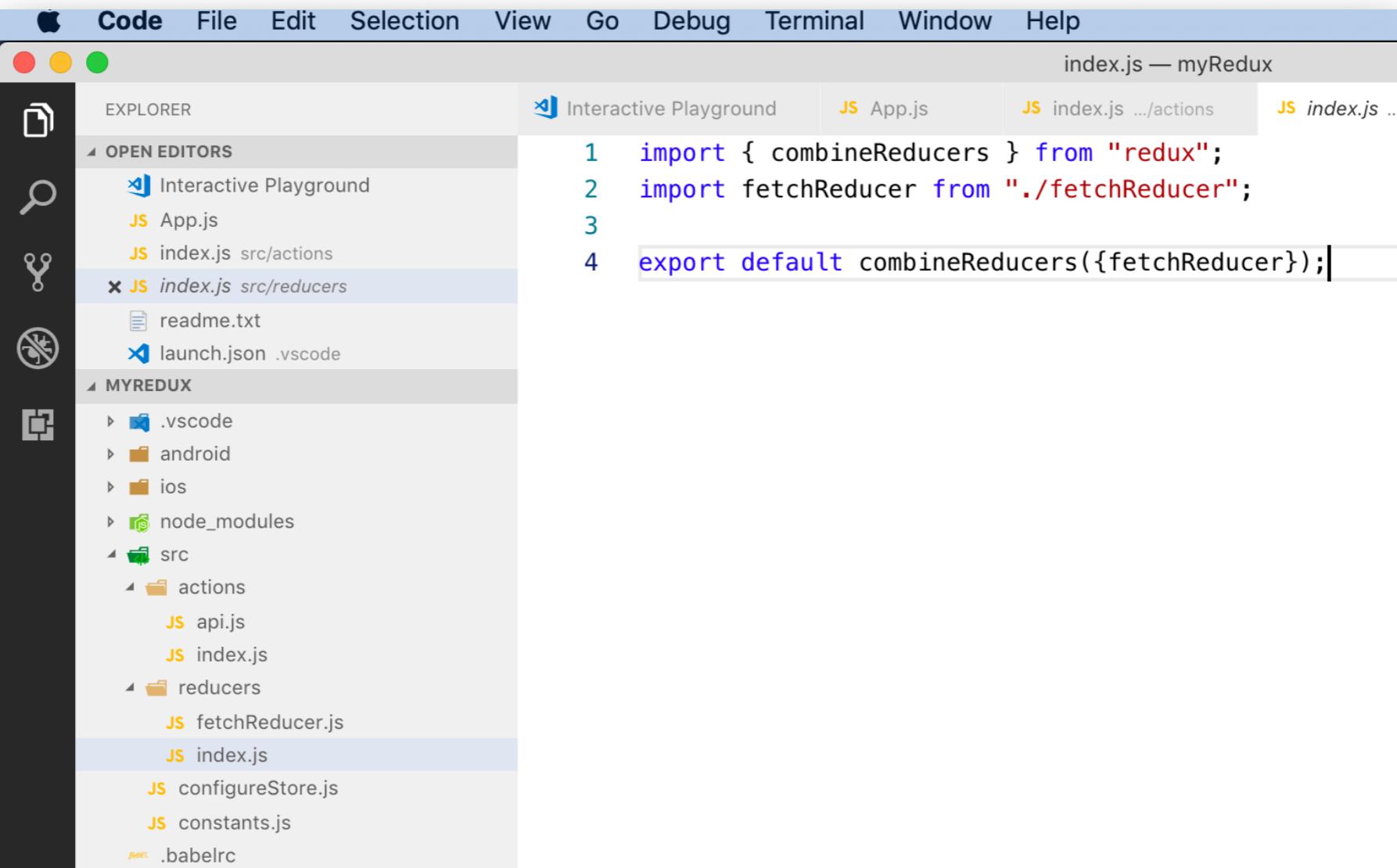
# Redux Workshop

## Define all actions linking to reducer

```
1 import { FETCHING_DATA, FETCHING_DATA_SUCCESS, F
2 import {feedPeople1, feedPeople2} from './api'
3
4
5 export const setStateToSuccess = (data) => ({
6   type: FETCHING_DATA_SUCCESS,
7   payload: data
8 })
9
10 +export const setStateToFetching = () => ({ ...
12   })
13
14 +export const setStateToFailure = () => ({ ...
16   })
17
18 +export const fetchData = ()=>{ ...
28   }
29
```

# Redux Workshop

## Reducer & combineReducers



A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows "Code" as the active tab. The left sidebar has sections for "EXPLORER" and "MYREDUX". In the "EXPLORER" section, files like "Interactive Playground", "App.js", "index.js src/actions", "index.js src/reducers" (which is selected), "readme.txt", and "launch.json .vscode" are listed. In the "MYREDUX" section, there are folders for ".vscode", "android", "ios", "node\_modules", and "src". Under "src", there are "actions" (with "api.js" and "index.js"), "reducers" (with "fetchReducer.js" and "index.js" selected), "configureStore.js", "constants.js", and ".babelrc". The main editor area shows the following code:

```
1 import { combineReducers } from "redux";
2 import fetchReducer from "./fetchReducer";
3
4 export default combineReducers({fetchReducer});
```

# Redux Workshop

## Reducer

reducer.js

index.js

combineReducer

## index.js

```
import { Provider } from 'react-redux';
import configureStore from './src/configureStore'
const store = configureStore();
const ReduxApp = ()=>(
  <Provider store={store}>
    <App/>
  </Provider>
)
AppRegistry.registerComponent(appName, () => ReduxApp);
```

## configureStore.js

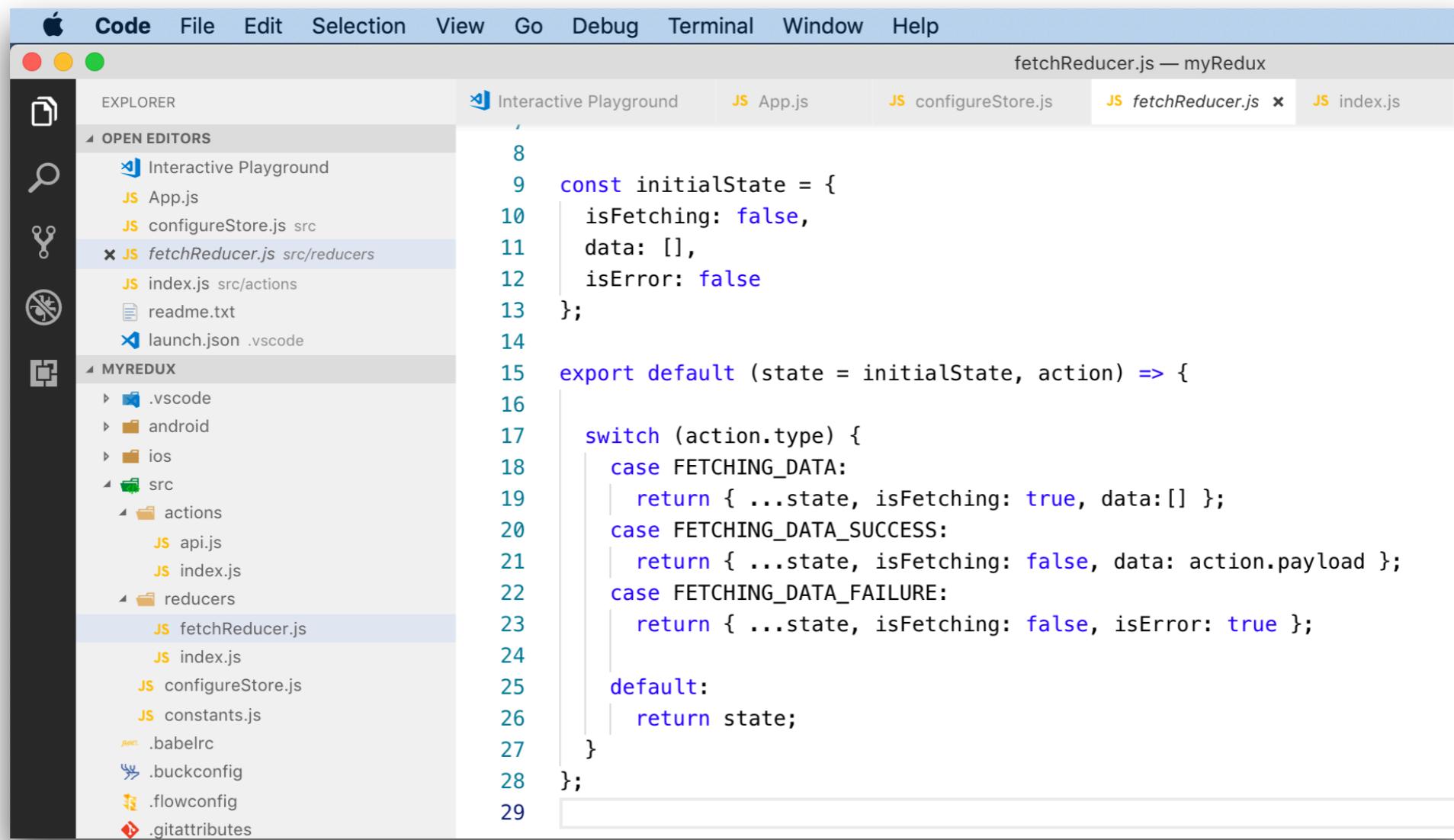
```
import { createStore, applyMiddleware } from "redux";
import thunk from "redux-thunk";
import reducer from './reducers';

export default ()=>{
  const store = createStore(reducer, applyMiddleware(thunk ))
  return store; }
```

# Redux Workshop

## Be careful

Same type name could cause an error because all combined reducers will concat their switch-case



```
8
9 const initialState = {
10   isFetching: false,
11   data: [],
12   isError: false
13 };
14
15 export default (state = initialState, action) => {
16
17   switch (action.type) {
18     case FECTHING_DATA:
19       return { ...state, isFetching: true, data:[] };
20     case FECTHING_DATA_SUCCESS:
21       return { ...state, isFetching: false, data: action.payload };
22     case FECTHING_DATA_FAILURE:
23       return { ...state, isFetching: false, isError: true };
24     default:
25       return state;
26   }
27 };
28
29
```

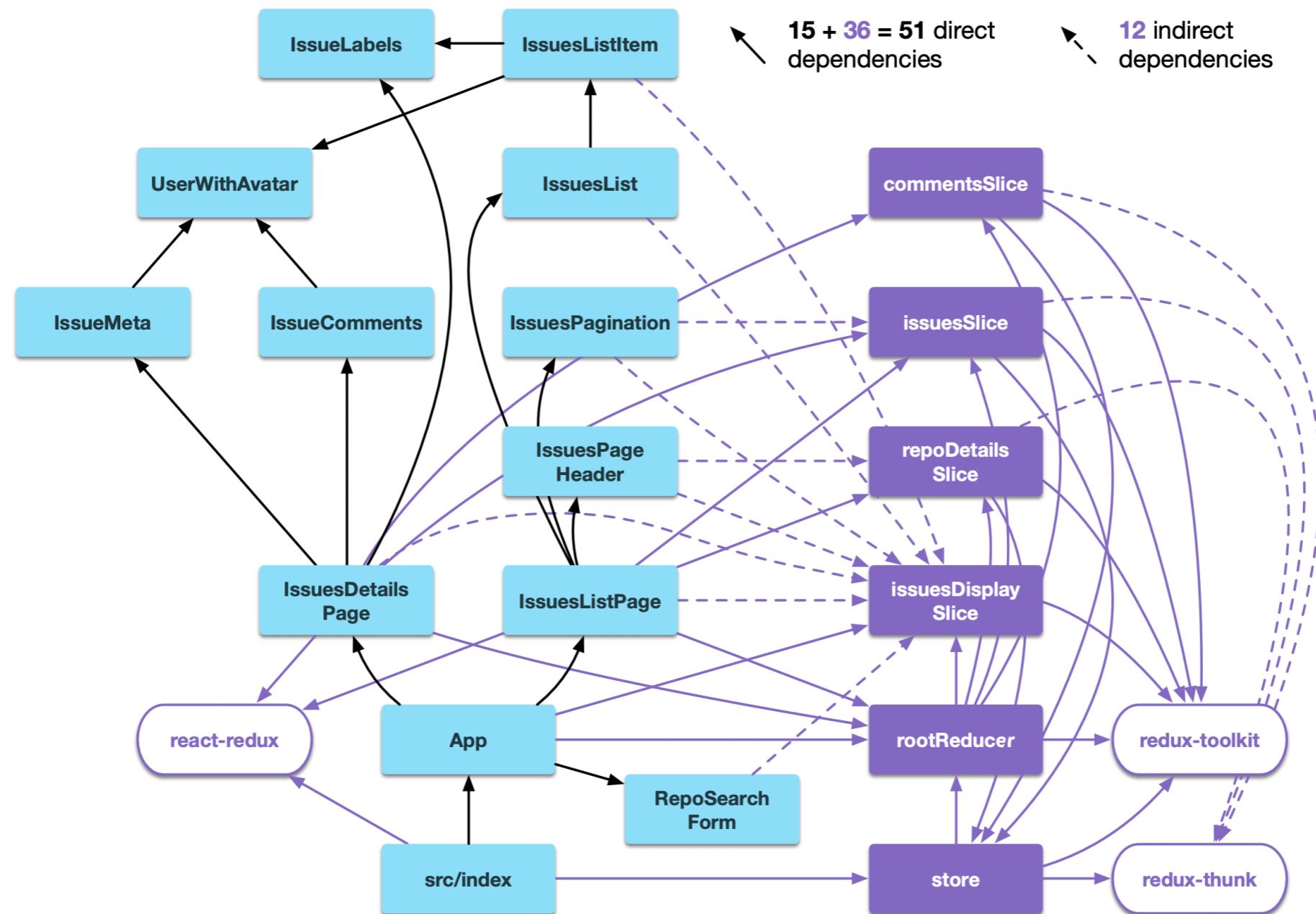


# Redux Toolkit

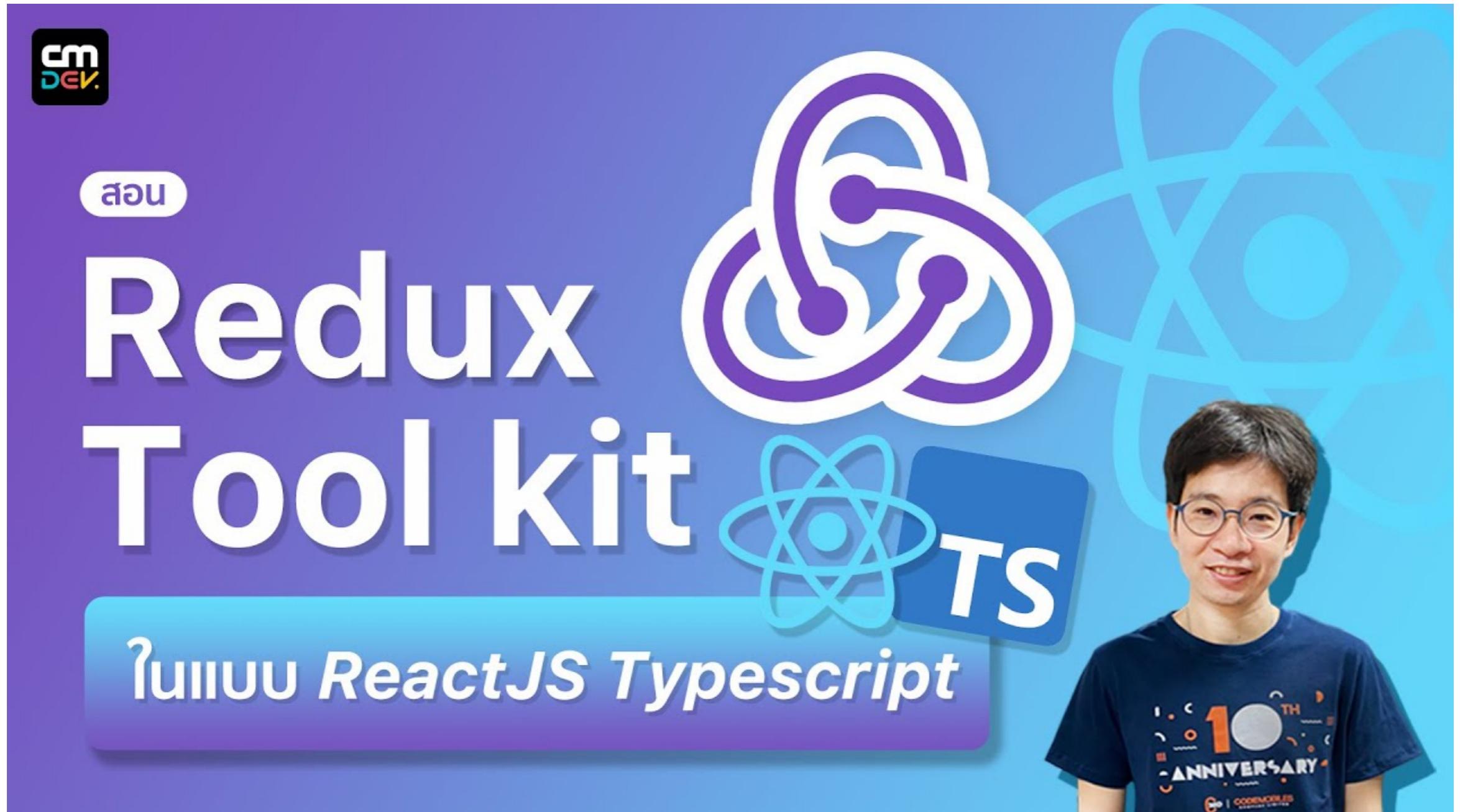
*The best way to use Redux.*

# Redux Toolkit

# Redux Toolkit - Github Issue Browser

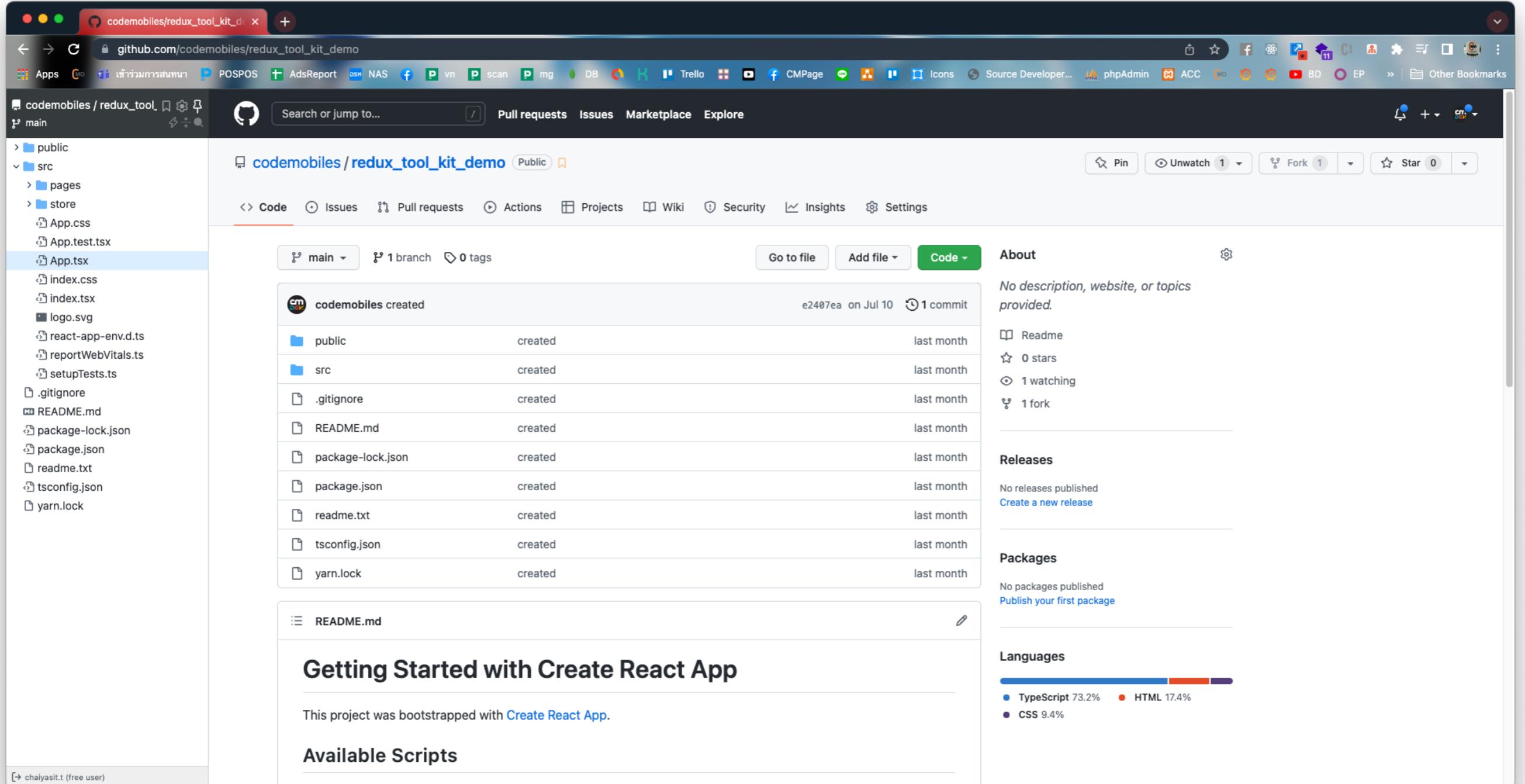


# Redux Toolkit



<https://www.youtube.com/watch?v=zBXf8GojEks>

# Redux Toolkit



The screenshot shows a GitHub repository page for 'codemobiles/redux\_tool\_kit\_demo'. The repository is public and was created by codemobiles on July 10, 2023, with 1 commit. It contains 1 branch and 0 tags. The code tab is selected, showing a list of files and their creation dates:

File	Created	Last Month
public	created	last month
src	created	last month
.gitignore	created	last month
README.md	created	last month
package-lock.json	created	last month
package.json	created	last month
readme.txt	created	last month
tsconfig.json	created	last month
yarn.lock	created	last month

The README.md file contains the following content:

## Getting Started with Create React App

This project was bootstrapped with [Create React App](#).

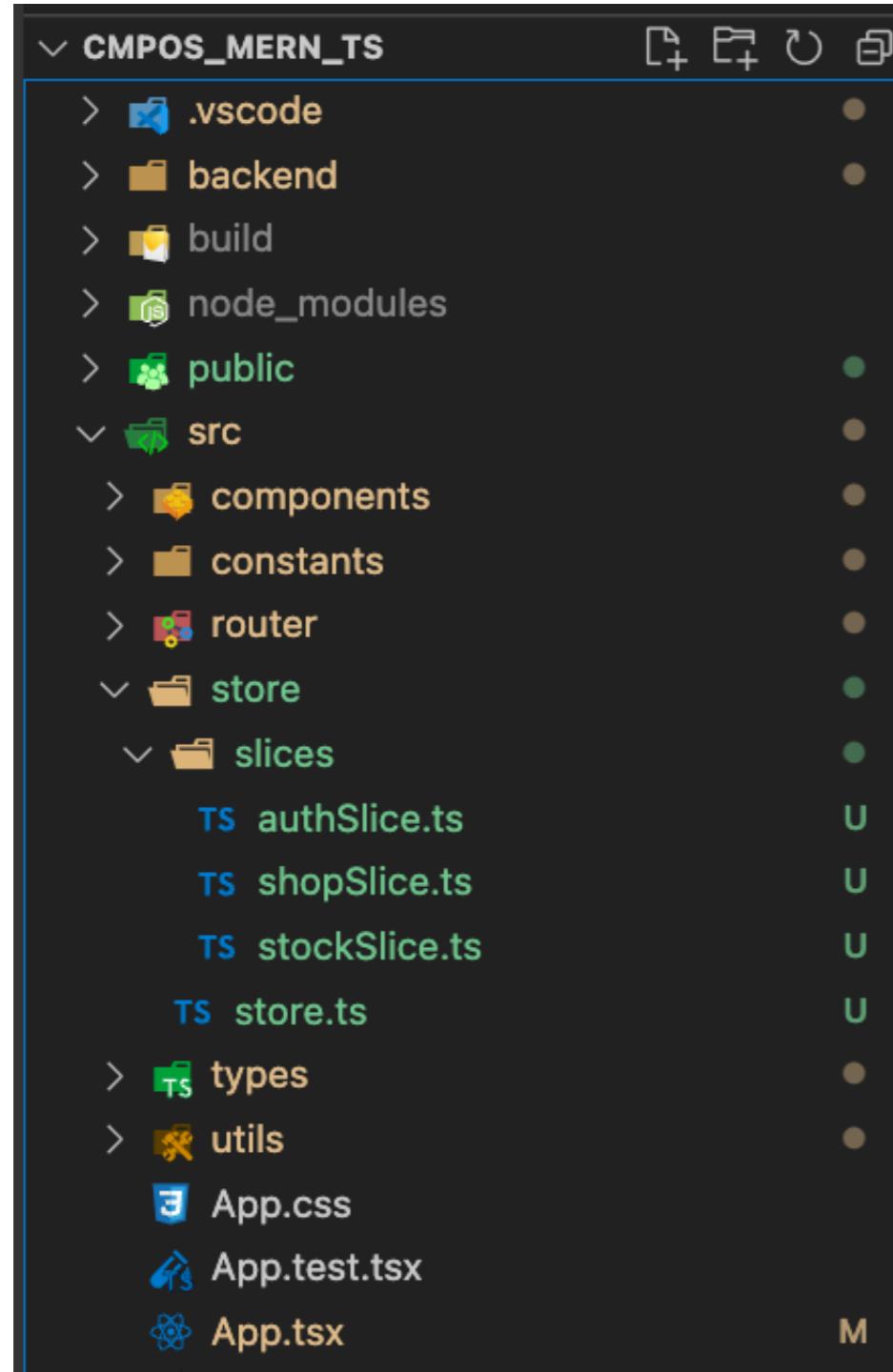
### Available Scripts

languages

TypeScript 73.2% | HTML 17.4% | CSS 9.4%

[https://github.com/codemobiles/redux\\_tool\\_kit\\_demo](https://github.com/codemobiles/redux_tool_kit_demo)

# Redux Toolkit



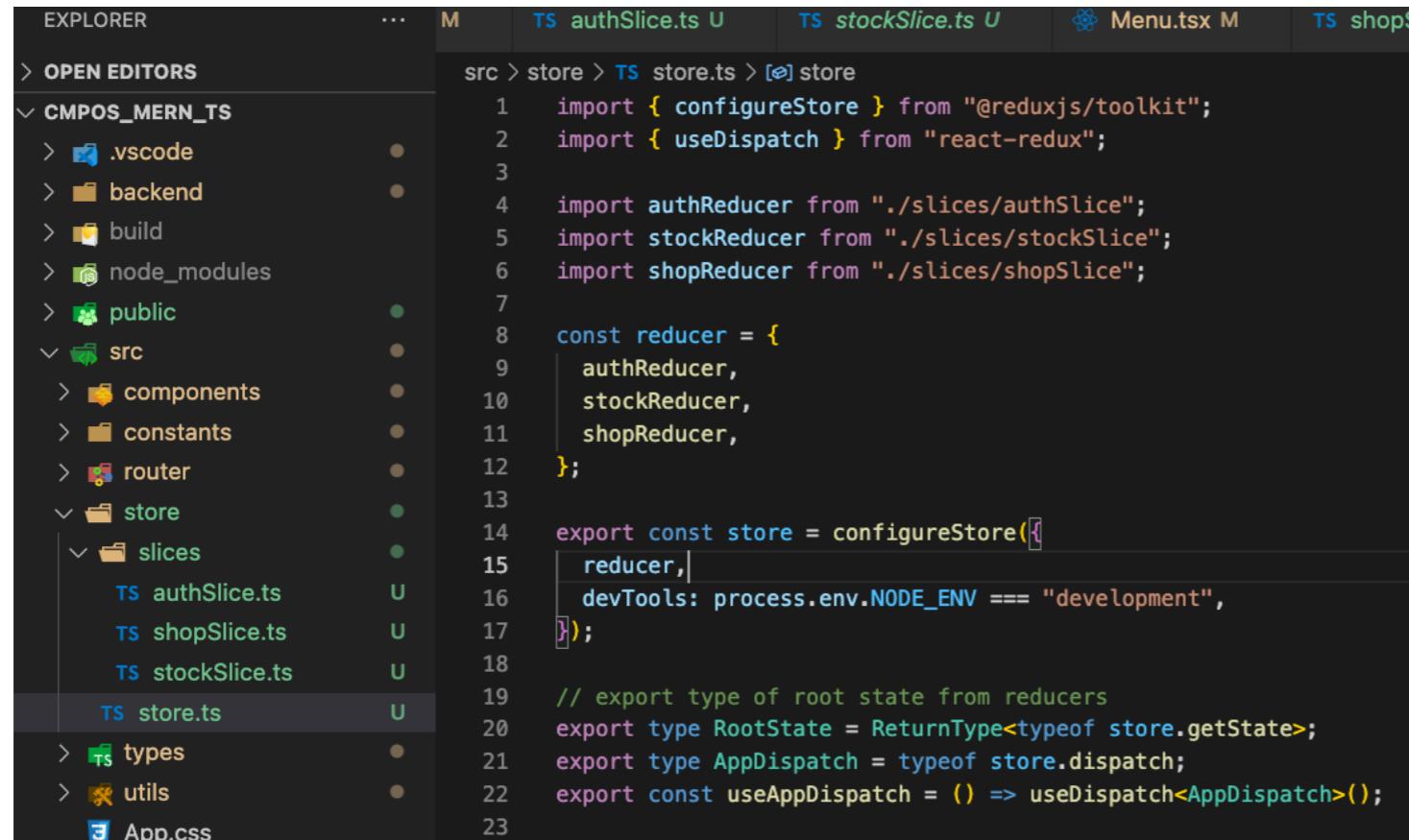
## File Structure for Redux Toolkit

# Redux Toolkit

```
ReactDOM.render([ You, 9 months ago
  <BrowserRouter>
    <Provider store={store}>
      <App />
    </Provider>
  </BrowserRouter>,
  document.getElementById("root")
];
```

Wrap App with Redux Provider to allow all sub-components and App itself - can access redux store (actions and states)

# Redux Toolkit



```

EXPLORER          ... M TS authSlice.ts U TS stockSlice.ts U TS Menu.tsx M TS shopS
> OPEN EDITORS
src > store > TS store.ts > [o] store
1   import { configureStore } from "@reduxjs/toolkit";
2   import { useDispatch } from "react-redux";
3
4   import authReducer from "./slices/authSlice";
5   import stockReducer from "./slices/stockSlice";
6   import shopReducer from "./slices/shopSlice";
7
8   const reducer = {
9     authReducer,
10    stockReducer,
11    shopReducer,
12  };
13
14  export const store = configureStore([
15    reducer,
16    devTools: process.env.NODE_ENV === "development",
17  ]);
18
19 // export type of root state from reducers
20 export type RootState = ReturnType<typeof store.getState>;
21 export type AppDispatch = typeof store.dispatch;
22 export const useAppDispatch = () => useDispatch<AppDispatch>();
23

```

Use configureStore to create a store, which contains all states and the configuration for development. RootState is the root type that contains all sub-state types of each slice. useAppDispatch is used to create a dispatch method that will be in the components

# Redux Toolkit

```
import { createSlice } from "@reduxjs/toolkit";
import { RootState } from "../store";

export interface ExampleState {}

const initialState: ExampleState = {};

const exampleSlice = createSlice({
  name: "demo",
  initialState,
  reducers: {},
  extraReducers: (builder) => {},
});

export const exampleSelector = (store: RootState): ExampleState => store.stockReducer;
export default exampleSlice.reducer;
```

createSlice defines a slice containing name, initial-state, reducer (synchronous-actions), and the extraReducers (asynchronous-actions)

# Redux Toolkit

```
const authSlice = createSlice({
  name: "auth",
  initialState: initialState,
  reducers: {
    logout: (state: AuthState, action: PayloadAction<void>) => {
      localStorage.removeItem(server.TOKEN_KEY);
      localStorage.removeItem(server.REFRESH_TOKEN_KEY);
      state.isAuthenticated = false;
      history.push("/login");
    },
    relogin: (state: AuthState, action: PayloadAction<void>) => {
      const _token = localStorage.getItem(server.TOKEN_KEY);
      const _refreshToken = localStorage.getItem(server.REFRESH_TOKEN_KEY);
      if (_token && _refreshToken) {
        if (!state.loginResult) {
          state.loginResult = {
            refreshToken: _refreshToken,
            token: _token,
            result: "ok",
          };
          state.isAuthenticated = true;
        }
      }
      state.isAuthenticating = false;
    },
  },
  extraReducers: (builder) => {
```

Slice in action

# Redux Toolkit

Access action via dispatch while state via selector

```
const Login = (props: LoginProps) => {
  const dispatch = useAppDispatch(); // used to call actions
  const authReducer = useSelector(authSelector); // used to access state
  const navigate = useNavigate();

  ...
  <Formik
    initialValues={initialValue}
    onSubmit={(values, { setSubmitting }) =>
      dispatch(login(values));
      setSubmitting(false);
    }
  >
  ...
  {authReducer.isError && <Alert severity="error">Login failed</Alert>}

```



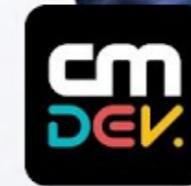
# React MaterialUI

A screenshot of the mui.com homepage displayed in a web browser. The page features a large hero section with the text "Move faster with intuitive React UI tools". Below this, a paragraph explains that MUI offers a comprehensive suite of UI tools to help you ship new features faster. A "Get started" button and an npm install command are shown. To the right, there are several UI component examples: a card with a todo item for Michael Scott, a media player showing a video titled "Ultraviolet" by Basement, and a date picker showing August 2022. The browser's toolbar, address bar, and menu bar are visible at the top, and the Mac OS Dock is visible at the bottom.



# React MaterialUI (Components)

<https://www.youtube.com/watch?v=VQ85QLEa270&list=PLjPfp4Ph3gBomMX1g9FPJFmTMw0gkQM3J&index=1>





# React MaterialUI (Component)

<https://www.youtube.com/watch?v=21TOW92-BUU&list=PLjPfp4Ph3gBomMX1g9FPJFmTMw0gkQM3J>

The screenshot shows a browser window with a YouTube video player on the right and a code editor on the left. The code editor displays a file named App.tsx with the following content:

```
import Button from '@mui/material/Button';
import { Box } from '@mui/material';

export default function App() {
  return [
    <Box>
      /* SX */
      <Stack direction="row">
        <Button variant="text">Text</Button>
        <Button sx={{ width: 300, margin: 10 }} variant="contained">
          Contained
          You, 8 minutes ago * Uncommitted changes
        </Button>
        <Button style={{ width: 300, margin: 10 }} variant="outlined">
          Outlined
        </Button>
      </Stack>
    </Box>
  ];
}

PROBLEMS OUTPUT TERMINAL GITLENS JUPYTER
```

The terminal output shows the development build process:

```
Local: http://localhost:3000
On Your Network: http://192.168.1.215:3000

Note that the development build is not optimized.
To create a production build, use yarn build.
webpack compiled successfully
Files successfully emitted, waiting for typecheck results...
Issues checking in progress...
No issues found.
```

The YouTube video player is displaying a series of six video thumbnails for a playlist titled "สอน Material (MUI V5) UI Style". The thumbnails are labeled Ep#1 - sx through Ep#6 - theme components, each with a duration indicator (e.g., 11:27, 3:18, 6:30, 6:00, 11:58, 5:31). The video player interface includes a search bar, a subscribe button, and navigation controls.

Below the video player, the video details are shown:

**Material (MUI V5) UI Style Ep#2 - theme**  
146 views • Jul 22, 2022

Interaction buttons: Like (2), Dislike, Share, Download, Clip, Save, More.

Category: Computer programming

Related videos:

**Material (MUI V5) UI Style Ep#3 - nested element**



# React MaterialUI (Style)

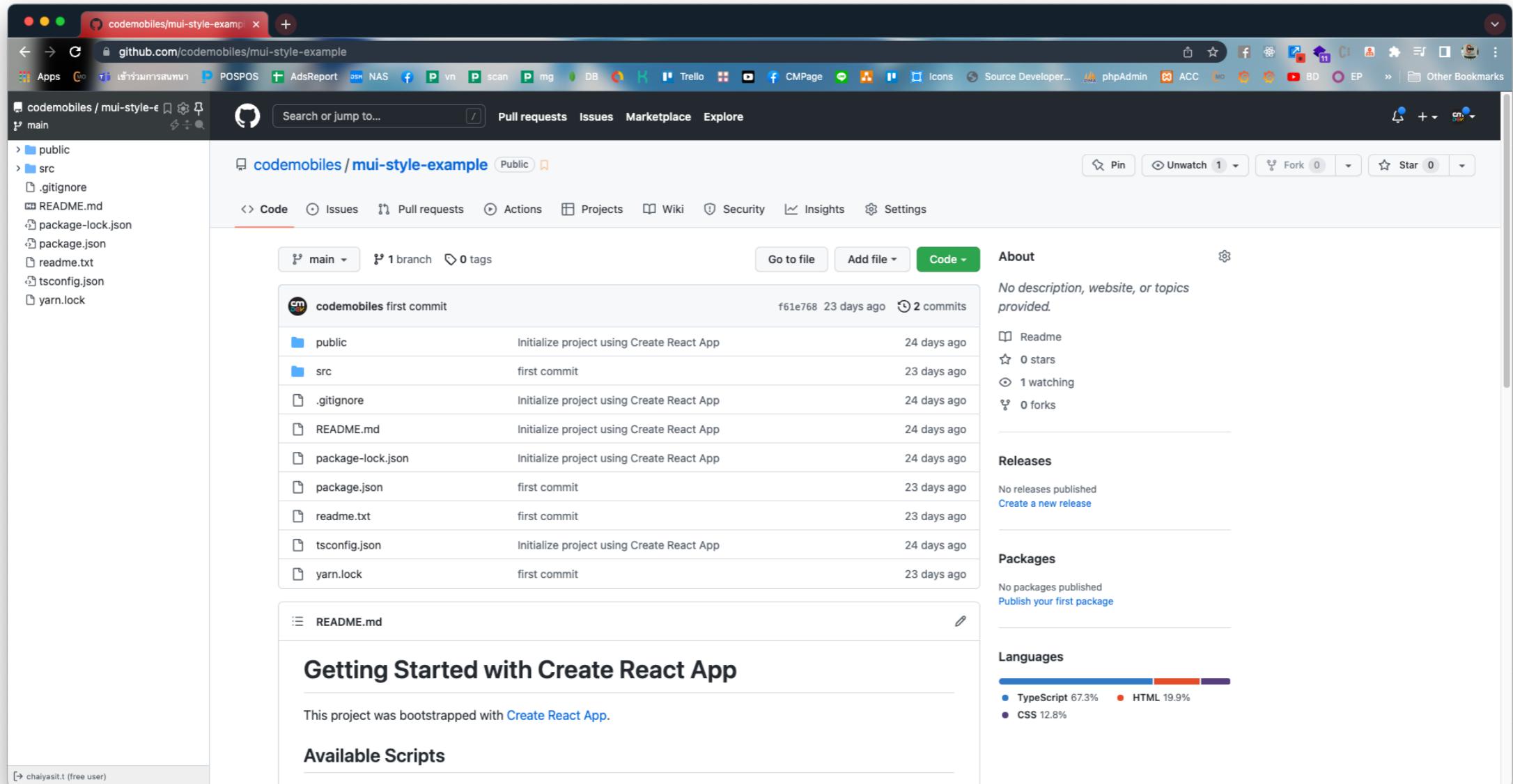
<https://www.youtube.com/watch?v=F6J0-n6fRYc&list=PLjPfp4Ph3gBpbVPivsolaOhfeNwmBskl8>



<https://github.com/codemobiles/mui-style-example/blob/main/src/App.tsx>

# React MaterialUI (Style)

<https://github.com/codemobiles/mui-style-example>



The screenshot shows a GitHub repository page for 'codemobiles / mui-style-example'. The repository is public and contains 1 branch and 0 tags. The commit history shows the initial setup of the project using Create React App. The 'About' section indicates no description, website, or topics provided. The 'Releases' section shows no releases published, with a link to 'Create a new release'. The 'Packages' section shows no packages published, with a link to 'Publish your first package'. The 'Languages' section shows TypeScript at 67.3%, HTML at 19.9%, and CSS at 12.8%.

<https://github.com/codemobiles/mui-style-example/blob/main/src/App.tsx>

# React MaterialUI (Import)

```
import {
  Box,
  createTheme,
  CssBaseline,
  Divider,
  FormControlLabel,
  GlobalStyles,
  Slider,
  styled,
  Switch,
  ThemeProvider,
  Typography,
} from "@mui/material";
```



# React MaterialUI (sx)

```
/* SX */
<Stack direction="row">
  <Button variant="text">Text</Button>
  <Button sx={{ width: 300, margin: 1 }} variant="contained">
    Contained
  </Button>
  <Button style={{ width: 300, margin: 10 }} variant="outlined">
    Outlined
  </Button>
</Stack>
```

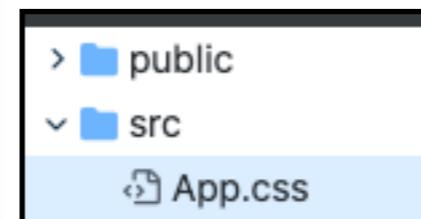
# React MaterialUI (nested class)

```
{/* & .Classss */}
<Stack spacing={2} direction="row">
  <Slider
    sx={{ "& .MuiSlider-thumb": { borderRadius: 0 } }}
    defaultValue={50}
    aria-label="Default"
    valueLabelDisplay="auto"
  />
</Stack>
```

# React MaterialUI (Class)

```
{/* App.css */}  
<Stack spacing={2} direction="row">  
  <Button className="MyButton" variant="text">  
    Text  
  </Button>  
  
  <Button className="MyButton" disabled variant="text">  
    Text  
  </Button>  
</Stack>
```

```
.MyButton{  
  background-color: #61dafb !important;  
}  
  
.MyButton:hover{  
  background-color: yellow !important;  
}  
  
.MyButton.Mui-disabled{  
  background-color: red !important;  
}
```



# React MaterialUI (Reusable Component)

```
/* Reusable Component (Styled Component) */
<Stack spacing={2} direction="row">
  <CMButton>1234</CMButton>
  <CMButton>1234</CMButton>
  <CMButton>1234</CMButton>
</Stack>
```

main ▾ mui-style-example / src / AppStyle.ts / Jump to ▾

codemobiles first commit

1 contributor

20 lines (18 sloc) | 398 Bytes

```
1 import {
2   Button,
3   ButtonProps,
4   Divider,
5   DividerProps,
6   styled,
7 } from "@mui/material";
8
9 export const CMButton = styled(Button)<ButtonProps>(({ theme }) => ({
10   width: 100,
11   backgroundColor: "#FF0",
12   color: "#F00",
13 }));
14
15 //
16 export const CMDivider = styled(Divider)<DividerProps>(({ theme }) => ({
17   marginTop: theme.spacing(2),
18   marginBottom: theme.spacing(2),
19   background: "#0F0",
20 }));
```

# React MaterialUI (Custom Theme)

```
const theme = createTheme({
  components: {
    MuiCssBaseline: {
      styleOverrides: `
        h3 {
          color: green;
        },
        button {
          background: green;
          color: white;
        }
      `,
    },
    MuiButton: {
      styleOverrides: {
        contained: {
          borderRadius: 10,
        },
      },
    },
    MuiTypography: {
      styleOverrides: {
        h1: {
          color: "#FF0",
        },
        h2: {
          color: "#0F0",
        },
      },
    },
    spacing: 8,
  });
});
```

```
{/* Global Theme component */}
<Stack spacing={2} direction="row">
  <Typography variant="h1">1234</Typography>
  <Typography variant="h2">1234</Typography>
</Stack>
```

# React MaterialUI (Dynamic Css)

```
/* Dynamic CSS */
<Stack spacing={2} direction="row">
  <FormControlLabel
    control={
      <Switch
        checked={success}
        onChange={handleChange}
        color="primary"
        value="dynamic-class-name"
      />
    }
    label="Success"
  />
  <DynamicButton online={success}>1234</DynamicButton>
</Stack>
```

```
export default function App() {
  const [success, setSuccess] = React.useState(false);

  const handleChange = (event: React.ChangeEvent<HTMLInputElement>) => {
    setSuccess(event.target.checked);
  };

  interface StyledDynamicButtonProps extends ButtonProps {
    online?: boolean;
  }

  const DynamicButton = styled(Button)<StyledDynamicButtonProps>(
    ({ online, theme }) => ({
      width: 100,
      backgroundColor: online ? "#FF0" : "#0F0",
      color: "#F00",
    })
  );
}
```

<https://github.com/codemobiles/mui-style-example/blob/main/src/App.tsx>



# React MaterialUI (Global CSS override)

```
/* Global CSS override */
<Stack spacing={2} direction="row">
  <h1>Grey h1 element</h1>
  <GlobalStyles
    styles={{ h1: { color: "grey" }, span: { color: "green" } }}>
  </GlobalStyles>
  <h1>Grey h1 element</h1>
  <span>1234</span>
</Stack>
```

<https://github.com/codemobiles/mui-style-example/blob/main/src/App.tsx>



# React MaterialUI (CSSBaseLine)

```
const theme = createTheme({  
  components: {  
    MuiCssBaseline: {  
      styleOverrides: `  
        h3 {  
          color: green;  
        },  
        button {  
          background: green;  
          color: white;  
        }  
      `,  
    },  
  },  
});
```

```
return (  
  <ThemeProvider theme={theme}>  
    <CssBaseline />  
    <Box>  
      {/* SX */}  
      <Stack direction="row">  
        <Button variant="text">Text</Button>  
        <Button sx={{ width: 300, margin: 0 }}>  
          Contained  
        </Button>  
        <Button sx={{ width: 300 }}>Outlined</Button>  
      </Stack>  
    </Box>  
  </ThemeProvider>  
)
```

```
{/* CSSBaseLine */}  
<Stack spacing={2} direction="row">  
  <h3>Green h3 element</h3>  
  <button>1234</button>  
</Stack>
```

# Mocking Data with MSW Library



## msw

Seamless REST/GraphQL API  
mocking library for browser and  
Node.js.

# Mocking APIs During Development

The slide features a large yellow title 'Mocking APIs' at the top left, followed by 'During Development' below it. To the right of the text is a circular portrait of a person wearing glasses and a dark t-shirt with '10TH ANNIVERSARY' and 'CODEMOBILES' printed on it. Above the portrait is a blue React logo icon. In the background, there is a dark-themed GitHub code editor interface showing code snippets related to API mocking.

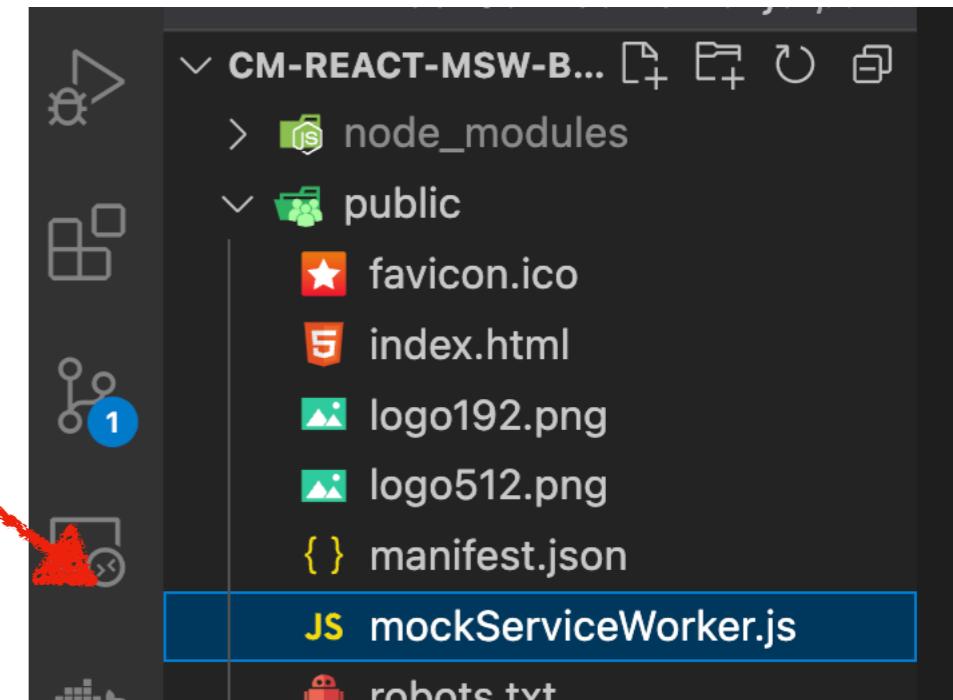
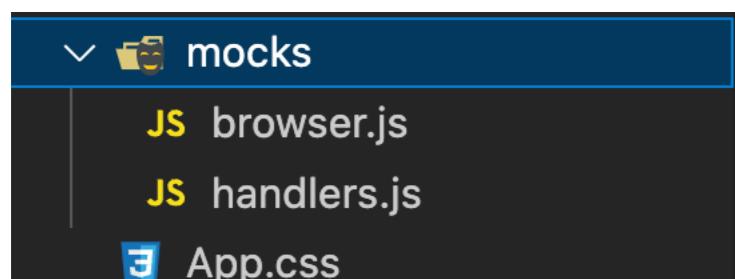
## Mocking APIs

# During Development

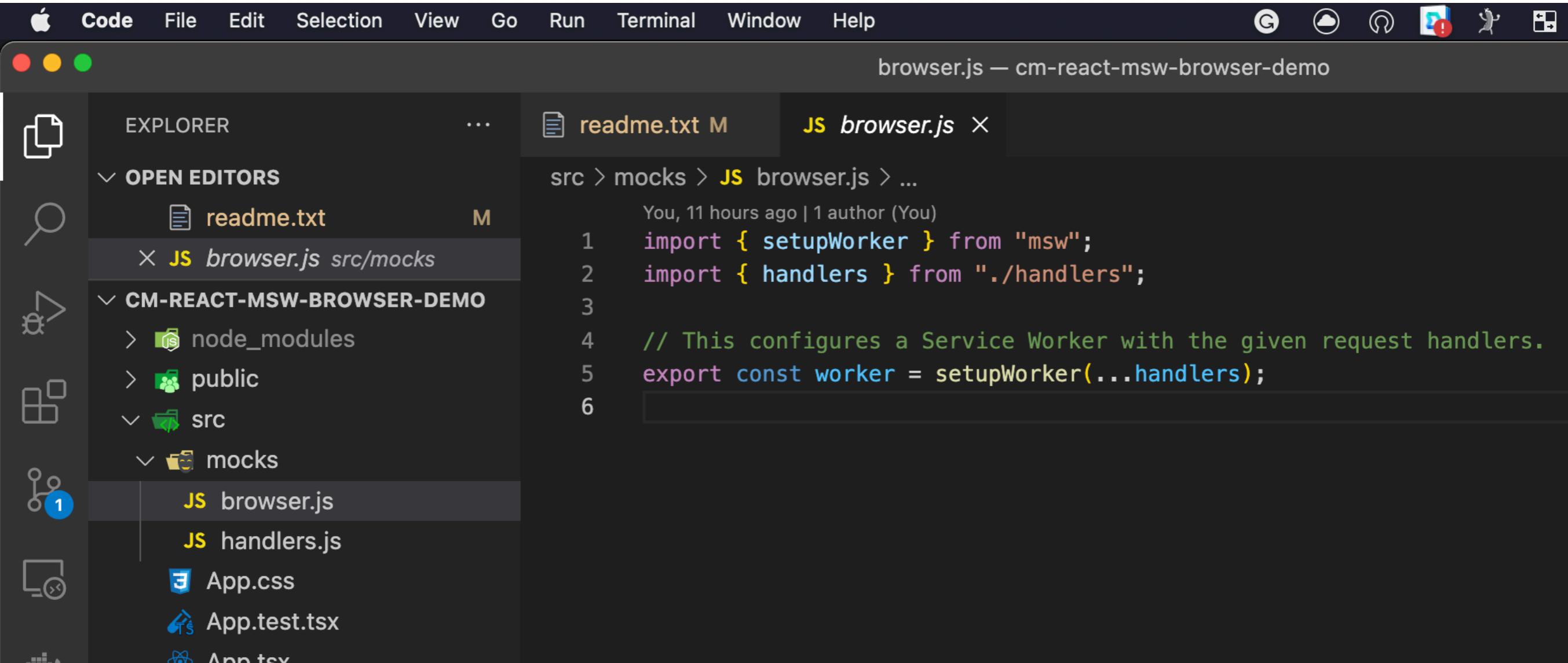
<https://github.com/codemobiles/cm-react-msw-browser-demo>

# Mocking APIs During Development

```
https://mswjs.io/docs/getting-started/install
yarn add axios
yarn add msw --dev
npx msw init public/ --save
```



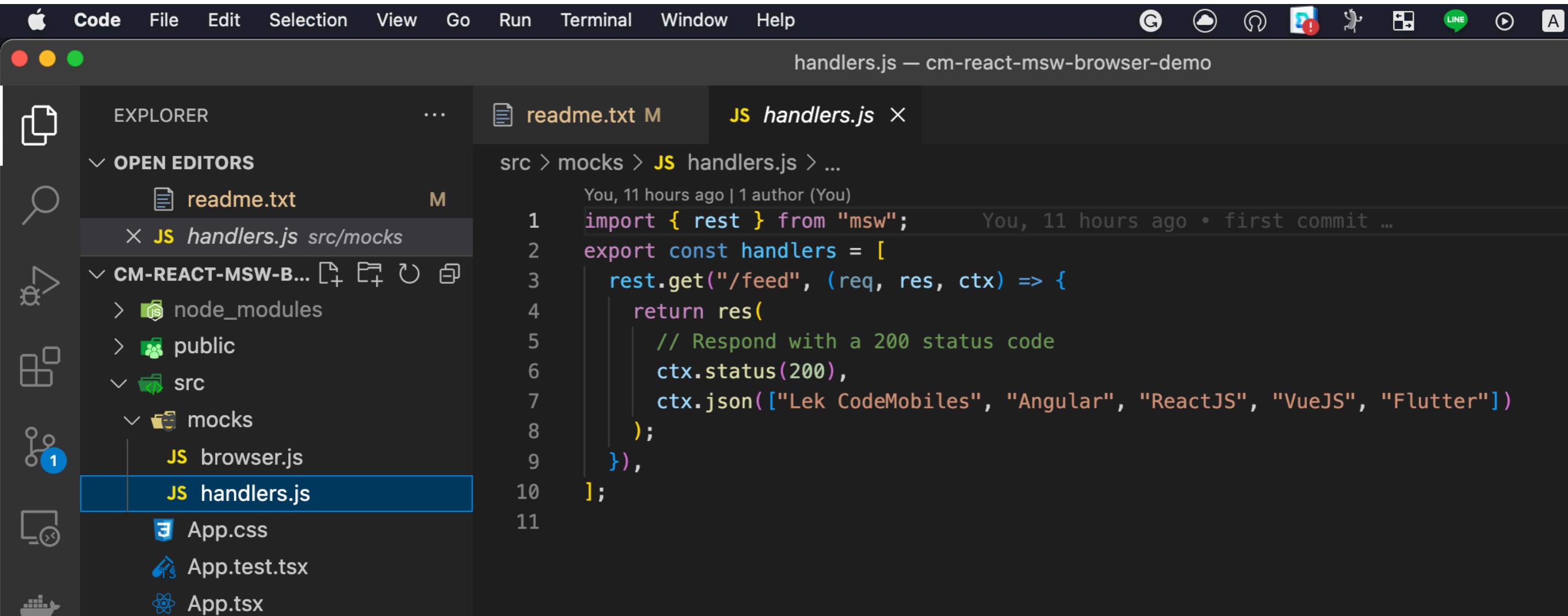
# Mocking APIs During Development



A screenshot of a macOS desktop environment showing a code editor window. The window title is "browser.js — cm-react-msw-browser-demo". The menu bar includes "Code", "File", "Edit", "Selection", "View", "Go", "Run", "Terminal", "Window", and "Help". The toolbar icons include a G, a cloud, a refresh, a file with a red exclamation mark, and a cursor. The left sidebar has icons for Explorer, Open Editors, CM-REACT-MSW-BROWSER-DEMO, and a notifications badge. The "OPEN EDITORS" section lists "readme.txt" and "JS browser.js src/mocks". The main editor area shows the file structure: "src > mocks > JS browser.js > ...". The code content is as follows:

```
You, 11 hours ago | 1 author (You)
1 import { setupWorker } from "msw";
2 import { handlers } from "./handlers";
3
4 // This configures a Service Worker with the given request handlers.
5 export const worker = setupWorker(...handlers);
6
```

# Mocking APIs During Development



Code — cm-react-msw-browser-demo

EXPLORER

OPEN EDITORS

- readme.txt M
- JS handlers.js src/mocks

CM-REACT-MSW-B... [+] ⏎ ⏴ ⏵

- > node\_modules
- > public
- src
  - mocks
    - JS browser.js
    - JS handlers.js
  - JS App.css
  - TSX App.test.tsx
  - TSX App.tsx

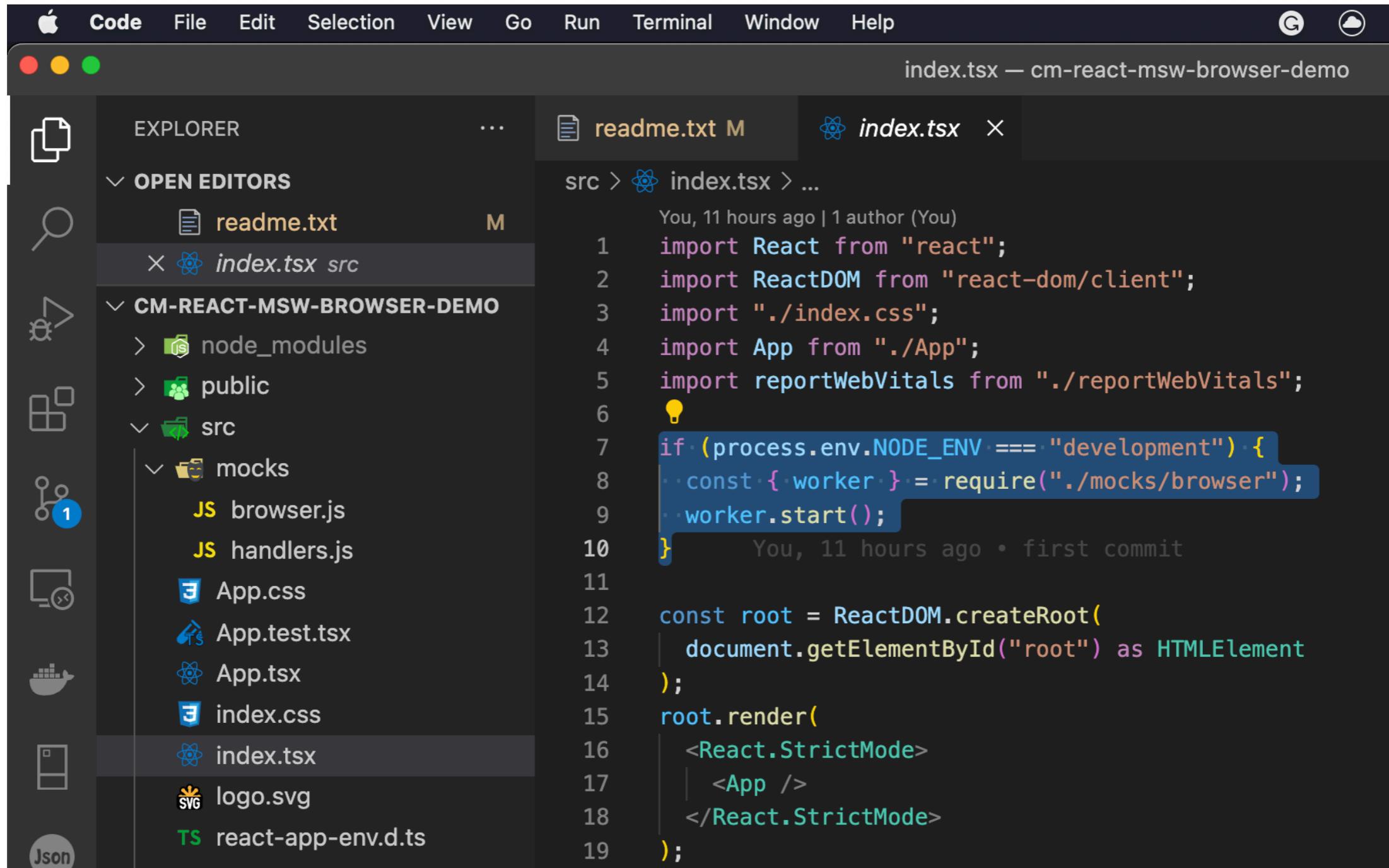
handlers.js — cm-react-msw-browser-demo

src > mocks > JS handlers.js > ...

You, 11 hours ago | 1 author (You)

```
1 import { rest } from "msw";           You, 11 hours ago • first commit ...
2 export const handlers = [
3   rest.get("/feed", (req, res, ctx) => {
4     return res(
5       // Respond with a 200 status code
6       ctx.status(200),
7       ctx.json(["Lek CodeMobiles", "Angular", "ReactJS", "VueJS", "Flutter"])
8     );
9   },
10 ];
11 ];
```

# Mocking APIs During Development

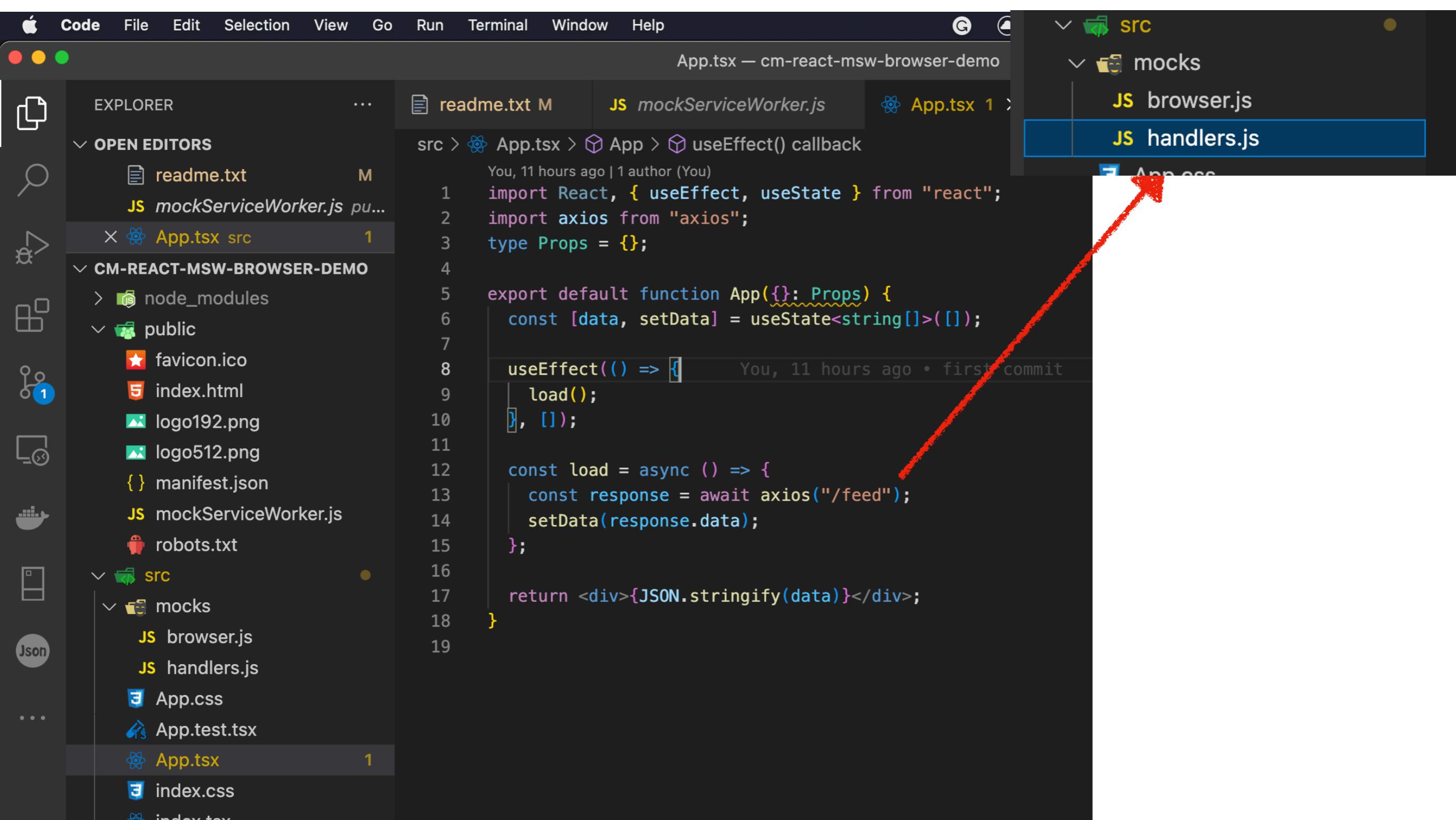


The screenshot shows a dark-themed code editor interface, likely VS Code, with the following details:

- File Menu:** Code, File, Edit, Selection, View, Go, Run, Terminal, Window, Help.
- Toolbar:** Includes standard Mac OS X-style icons for close, minimize, maximize, and quit.
- Editor Area:** Title bar: "index.tsx — cm-react-msw-browser-demo".
- Explorer:** Shows the project structure:
  - CM-REACT-MSW-BROWSER-DEMO folder
    - node\_modules
    - public
    - src
      - mocks
        - browser.js
        - handlers.js
      - App.css
      - App.test.tsx
      - App.tsx
      - index.css
      - index.tsx
      - logo.svg
      - react-app-env.d.ts
- Code Editor:** The "index.tsx" file is open, showing the following code:

```
1 import React from "react";
2 import ReactDOM from "react-dom/client";
3 import "./index.css";
4 import App from "./App";
5 import reportWebVitals from "./reportWebVitals";
6
7 if (process.env.NODE_ENV === "development") {
8   const { worker } = require("./mocks/browser");
9   worker.start();
10 }
11
12 const root = ReactDOM.createRoot(
13   document.getElementById("root") as HTMLElement
14 );
15 root.render(
16   <React.StrictMode>
17   |   <App />
18   </React.StrictMode>
19 );
```

# Mocking APIs During Development



A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows "Code" as the active tab. The left sidebar contains icons for Explorer, Search, Preview, Problems, and others. The main area shows a file tree under "OPEN EDITORS" with "App.tsx" selected. The "src" folder contains "mockServiceWorker.js", "App.tsx", "index.css", and "index.tsx". The "public" folder contains "node\_modules", "favicon.ico", "index.html", "logo192.png", "logo512.png", "manifest.json", "mockServiceWorker.js", and "robots.txt". A red arrow points from the bottom right towards the "App.css" file in the file tree.

App.tsx — cm-react-msw-browser-demo

EXPLORER OPEN EDITORS

src > App.tsx > App > useEffect() callback

You, 11 hours ago | 1 author (You)

```
1 import React, { useEffect, useState } from "react";
2 import axios from "axios";
3 type Props = {};
4
5 export default function App({}: Props) {
6   const [data, setData] = useState<string>([]);
7
8   useEffect(() => {
9     You, 11 hours ago • first commit
10    load();
11  }, []);
12
13  const load = async () => {
14    const response = await axios("/feed");
15    setData(response.data);
16  };
17
18  return <div>{JSON.stringify(data)}</div>;
19}
```

src mocks browser.js handlers.js App.css

# Mocking APIs for Tester



Mock Service Worker  
Seamless API mocking library

# Mocking APIs

## For Tester

The background shows a code editor with several files open, including 'server.js', 'setupTests.ts', 'App.test.tsx', and 'App.tsx'. The 'server.js' file contains code for setting up a mock server using the Mock Service Worker library.

Overlaid on the code editor are two large, semi-transparent icons: an orange diamond-shaped icon with a black dashed arrow pointing left and right, and a light blue React atom logo.

A portrait of a man with short dark hair and glasses, wearing a dark blue t-shirt with a graphic for '10TH ANNIVERSARY' and 'CODEMOBILES COMPANY LIMITED'. He is smiling and looking towards the camera.

<https://www.youtube.com/watch?v=8mm3XrY7q1c>

# Mocking APIs for Tester



**Mock Service Worker**  
Seamless API mocking library

```
yarn add msw --dev
```



**Mock Service Worker**  
Seamless API mocking library

# Mocking APIs for Tester

App.tsx

```
export default function App() {
  const [data, setData] = useState<string[]>();

  useEffect(() => {
    load();
  }, []);

  const load = async () => {
    try {
      const response = await axios("http://localhost:4000/feed");
      setData(response.data);
    } catch (e) {
      setData([]);
    }
  };

  return <div>{JSON.stringify(data)}</div>;
}
```

# Mocking APIs for Tester



Mock Service Worker  
Seamless API mocking library

server.js

The screenshot shows a macOS desktop with a dark-themed Code editor window open. The title bar reads "server.js — cm-react-msw-node-demo". The editor has five tabs: "App.test.tsx M", "readme.txt M", "App.tsx M", "JS server.js X", and "server.js" (which is currently active). The left sidebar shows the "OPEN EDITORS" section with "App.test.tsx", "readme.txt", "App.tsx", and "JS server.js" listed. Below it, the "CM-REACT-MSW-N..." folder contains a "backend" subfolder with "node\_modules" and "package.json". The "JS server.js" file is selected in the sidebar. The main editor area displays the following code:

```
const express = require("express");
const app = express();
const cors = require("cors");
app.use(cors());

app.get("/feed", (req, res) => {
  res.json(["Angular", "ReactJS", "VueJS", "Flutter"]);
});

app.listen(4000, () => console.log("Server is running..."));
```



**Mock Service Worker**  
Seamless API mocking library

```
App.test.tsx M X readme.txt M App.tsx

src > App.test.tsx > ...
You, 1 second ago | 1 author (You)
1 import React from "react";
2 import { act, render, screen } from "@testing-library/react";
3 import App from "./App";
4 import { rest } from "msw";
5 import { setupServer } from "msw/lib/node";
6
7 const server = setupServer(
8   rest.get("http://localhost:4000/feed", (req, res, ctx) => {
9     return res(
10       // Respond with a 200 status code
11       ctx.status(200),
12       ctx.json(["Lek CodeMobiles", "Angular", "ReactJS", "VueJS", "Flutter"])
13     );
14   })
15 );
16
17 // Establish API mocking before all tests.
18 beforeEach(() => server.listen());
19
20 // Reset any request handlers that we may add during the tests,
21 // so they don't affect other tests.
22 afterEach(() => server.resetHandlers());
23
24 // Clean up after the tests are finished.
25 afterAll(() => server.close());
26
27 > test("renders learn react link", async () => {
28   ...
29 });
30
31
32
33
34
35
36
37
38
39
40
```

# Mocking APIs for Tester



**Mock Service Worker**  
Seamless API mocking library

App.test.tsx

```
test("renders learn react link", async () => {
  render(<App />);

  await act(async () => {
    await new Promise((r) => setTimeout(r, 100));
  });

  // screen.debug();
  const linkElement = screen.getByText(
    ["Lek CodeMobiles", "Angular", "ReactJS", "VueJS", "Flutter"]
  );
  expect(linkElement).toBeInTheDocument();
});
```

# Mocking APIs for Tester



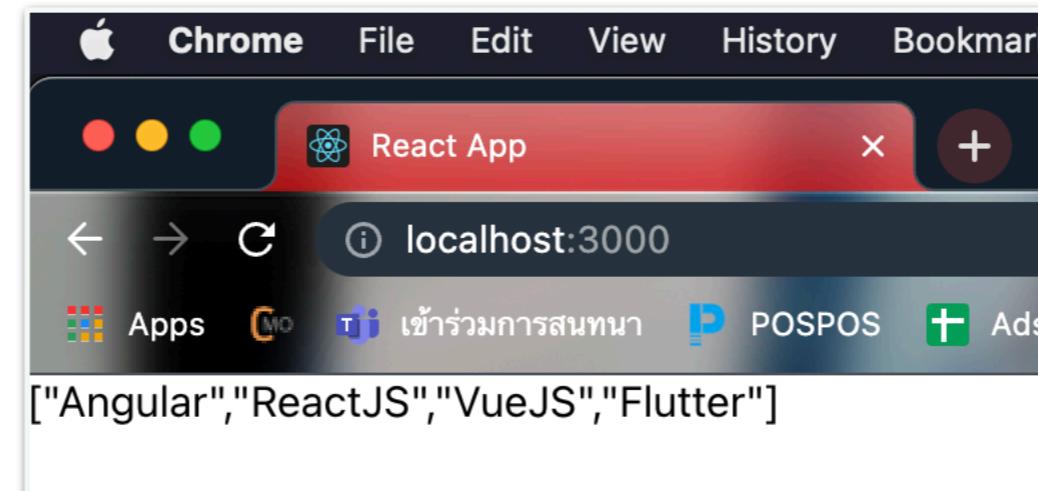
Mock Service Worker  
Seamless API mocking library

PROBLEMS OUTPUT TERMINAL GITLENS JUPYTER

## TERMINAL

```
Note that the development build is not optimized.  
To create a production build, use yarn build.  
  
webpack compiled successfully  
Files successfully emitted, waiting for typecheck results...  
Issues checking in progress...  
No issues found.
```

```
● → cm-react-msw-node-demo git:(main) ✘ cd backend  
○ → backend git:(main) ✘ node server.js  
Server is running...
```



# Mocking APIs for Tester



**Mock Service Worker**  
Seamless API mocking library

```
test("renders learn react link", async () => {
  render(<App />);

  await act(async () => {
    await new Promise((r) => setTimeout(r, 100));
  });

  // screen.debug();
  const linkElement = screen.getByText(
    ['Lek CodeMobiles', "Angular", "ReactJS", "VueJS", "Flutter"]
  );
  expect(linkElement).toBeInTheDocument();
});
```

PROBLEMS    OUTPUT    TERMINAL    GITLENS    JUPYTER

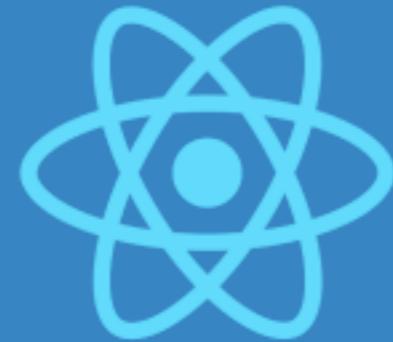
▽ TERMINAL

**PASS** src/App.test.tsx

✓ renders learn react link (152 ms)

Test Suites: 1 passed, 1 total  
Tests: 1 passed, 1 total  
Snapshots: 0 total  
Time: 1.718 s  
Ran all test suites related to changed files.

# React.JS UnitTest



React



React Testing Library



Jest

# React.JS UnitTest



We use Jest and React Testing Library for testing React Component and unit-test. Both Jest and React Testing Library come pre-packaged with Create React App

# React.JS UnitTest



## jest

- test
- expect
- to.....

## testing-library/react

- render
- screen
- get/find/query
- fireEvent

src/App.test.js

```
...
test('renders the landing page', () => {
  render(<App />

    expect(screen.getByRole("heading")).toBeInTheDocument(/Doggy Directory/);
    expect(screen.getByRole("combobox")).toHaveDisplayValue("Select a breed");
    expect(screen.getByRole("button", { name: "Search" })).toBeDisabled();
    expect(screen.getByRole("img")).toBeInTheDocument();
  );
});
```

Copy

# React.JS UnitTest (React)



<https://testing-library.com/docs/>

The screenshot shows a Mac OS X desktop with a dark-themed browser window open to the [Testing Library documentation](https://testing-library.com/docs/). The URL in the address bar is `https://testing-library.com/docs/`. The page title is "Introduction | Testing Library". The main content area has a light gray background. On the left, there's a sidebar with a vertical navigation menu. The menu items include "Testing Library" (with a red octopus icon), "Docs" (which is the active tab), and "Examples". Under "Docs", there are several sections: "Guiding Principles", "FAQ", "Core API" (with a dropdown arrow), "Queries" (with a dropdown arrow), "About Queries", "ByRole", "ByLabelText", "ByPlaceholderText", "ByText", "ByDisplayValue", "ByAltText", "ByTitle", "ByTestId", "User Actions" (with a dropdown arrow), "Firing Events", "Async Methods", and "Appearance and Disappearance". The main content area starts with a breadcrumb navigation: "Home > Getting Started > Introduction". Below this is a large section titled "Introduction". A text block states: "The `@testing-library` family of packages helps you test UI components in a user-centric way." A quote box follows: "The more your tests resemble the way your software is used, the more confidence they can give you." Then, two large sections are shown: "The problem" and "The solution". The "The problem" section contains text about writing maintainable tests that give high confidence in component functionality. The "The solution" section describes the core library, `DOM Testing Library`, as a light-weight solution for querying and interacting with DOM nodes, providing utilities for querying the DOM in a user-like manner.

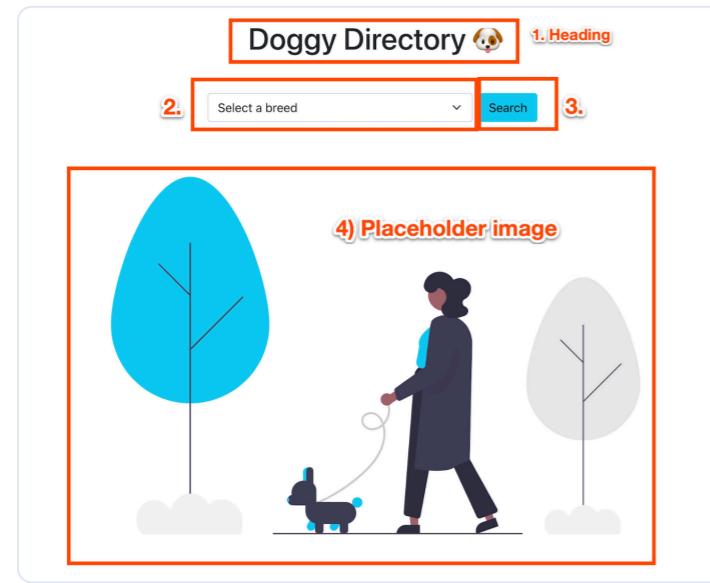
# React.JS UnitTest (Jest-dom)



<https://github.com/testing-library/jest-dom/>

A screenshot of a Mac OS X desktop showing a browser window for the Jest-dom GitHub repository. The page displays the README.md file, which includes a title 'jest-dom', a logo of an owl, and a subtitle 'Custom jest matchers to test the state of the DOM'. It shows build status (passing), coverage (100%), npm version (v5.16.5), and other metrics like 26M/month downloads and MIT license. The repository tree on the left shows files like .github, other, src, tests, and various configuration and ignore files. The right sidebar shows contributor statistics (+ 78 contributors) and a language bar indicating 100% JavaScript. A 'Table of Contents' section at the bottom lists several sections including 'The problem', 'This solution', and 'Usage'.

# React.JS UnitTest



src/App.test.js

```
...
test('renders the landing page', () => {
  render(<App />);

  expect(screen.getByRole("heading")).toHaveTextContent(/Doggy Directory/);
  expect(screen.getByRole("combobox")).toHaveDisplayValue("Select a breed");
  expect(screen.getByRole("button", { name: "Search" })).toBeDisabled();
  expect(screen.getByRole("img")).toBeInTheDocument();
});
```

Copy

The `expect` function is used every time you want to verify a certain outcome, and it accepts a single argument representing the value that your code produces. Most `expect` functions are paired with a *matcher* function to assert something about a particular value. For most of these assertions, you will use additional matchers provided by [jest-dom](#) to make it easier to check for common aspects found in the DOM. For example, `.toHaveTextContent` is the matcher for the `expect` function in the first line, while `getByRole("heading")` is the selector to grab the DOM element.

React Testing Library provides the `screen` object as a convenient way to access the pertinent queries needed to assert against the test DOM environment. By default, React Testing Library provides queries that allow you to locate elements within the DOM. There are three main categories of queries:

- `getBy*` (most commonly used)
- `queryBy*` (used when testing the absence of an element without throwing an error)
- `findBy*` (used when testing asynchronous code)

# React.JS UnitTest



64

React Testing Library and Jest both have different responsibilities "react testing library" is not a test runner meaning when you enter command `npm test` or `npm run test` it is **Jest responsibility** that collects all the files ending with `.test.js` and runs each test case and shows pass and fail results in your console like below

The screenshot shows the VS Code interface with the terminal tab selected. The output in the terminal is as follows:

```
PASS  src/App.test.js
  ✓ renders learn react link (1 ms)

  Test Suites: 1 passed, 1 total
  Tests:       1 passed, 1 total
  Snapshots:   0 total
  Time:        2.314 s
  Ran all test suites related to changed files.

  Watch Usage
```

react testing library provides you functions to catch DOM elements and perform some actions. Below are some of its functions:

```
render, fireEvent, waitFor, screen
```

# React.JS UnitTest



Screenshot of a GitHub repository page for `testing-library/jest-dom`.

The page shows the `README.md` file content:

## jest-dom



Custom jest matchers to test the state of the DOM

Key statistics:

- build: passing
- coverage: 100%
- npm: v5.16.5
- downloads: 26M/month
- license: MIT

Community metrics:

- all contributors: 28
- PRs: welcome
- code of conduct
- chat: 473 online

Watchers: 27 Stars: 3.5k Tweet

### The problem

You want to use `jest` to write tests that assert various things about the state of a DOM. As part of that goal, you want to avoid all the repetitive patterns that arise in doing so. Checking for an element's attributes, its text content, its css classes, you name it.

### This solution

The `@testing-library/jest-dom` library provides a set of custom jest matchers that you can use to extend jest. These will make your tests more declarative, clear to read and to maintain.

Table of Contents

# React.JS UnitTest



testing-library / jest-dom

main

.github

other

src

tests

.all-contributorsrc

.gitattributes

.gitignore

.huskyrc.js

.npmrc

.prettierignore

.prettierrc.js

CHANGELOG.md

CONTRIBUTING.md

LICENSE

README.md

extend-expect.js

jest.config.js

matchers.js

package.json

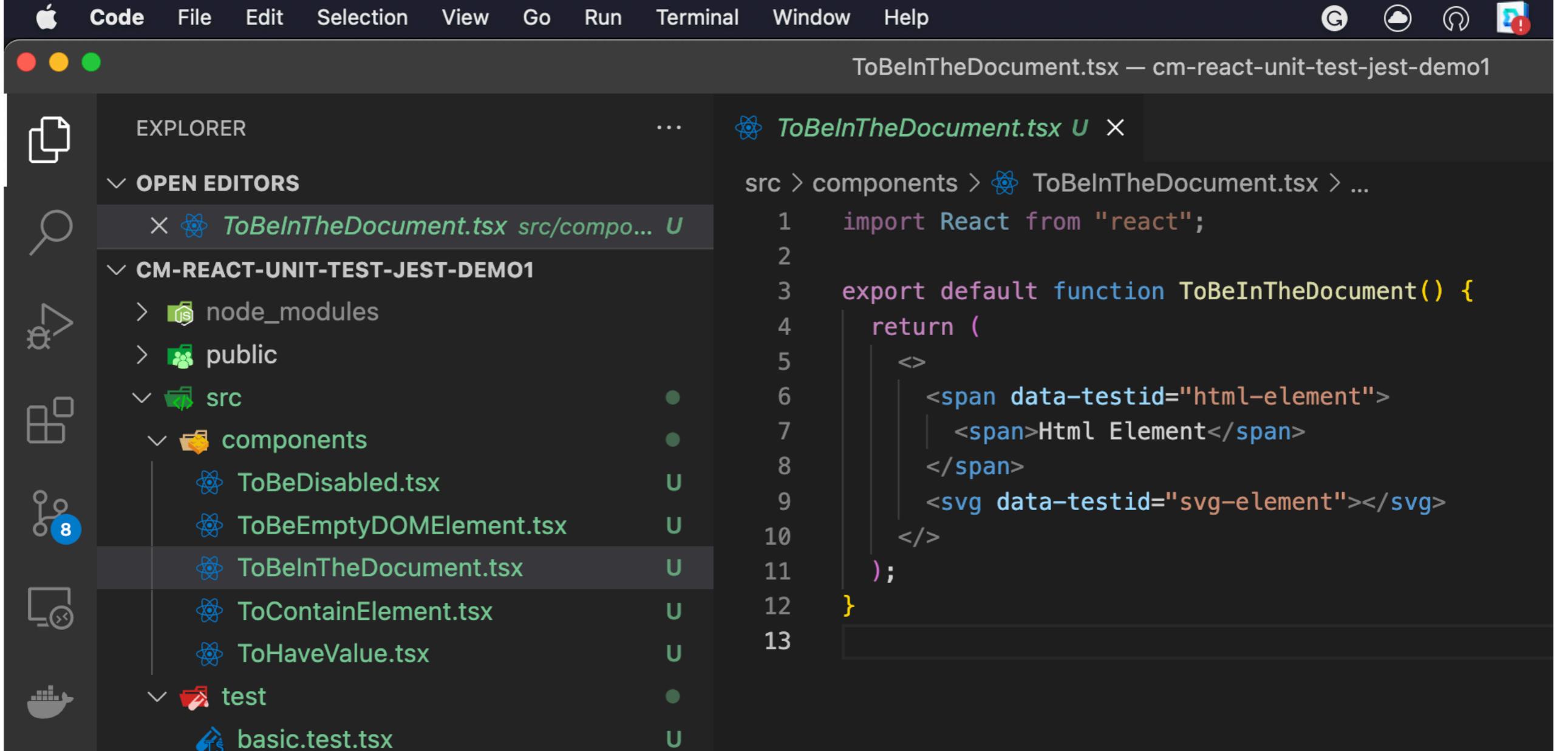
README.md

## Table of Contents

- Installation
- Usage
  - With TypeScript
- Custom matchers
  - [toBeDisabled](#)
  - [toBeEnabled](#)
  - [toBeEmptyDOMElement](#)
  - [toBeInTheDocument](#)
  - [toBeInvalid](#)
  - [toBeRequired](#)
  - [toBeValid](#)
  - [toBeVisible](#)
  - [toContainElement](#)
  - [toContainHTML](#)
  - [toHaveAccessibleDescription](#)
  - [toHaveAccessibleName](#)
  - [toHaveAttribute](#)
  - [toHaveClass](#)
  - [toHaveFocus](#)
  - [toHaveFormValues](#)
  - [toHaveStyle](#)
  - [toHaveTextContent](#)

<https://github.com/testing-library/jest-dom/#tobeinthedocument>

# React.JS UnitTest (Synchronous)



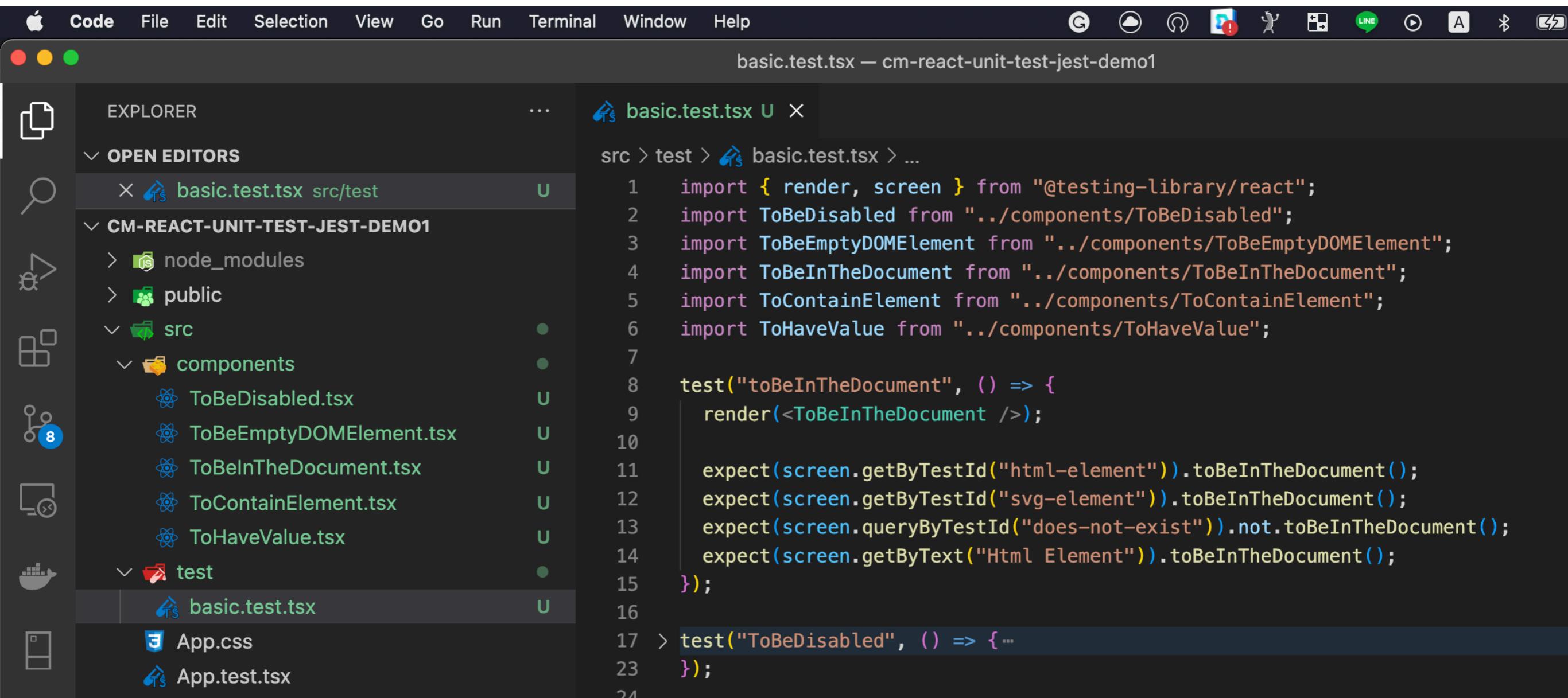
The screenshot shows the Visual Studio Code interface with the following details:

- File Menu:** Code, File, Edit, Selection, View, Go, Run, Terminal, Window, Help.
- Toolbar:** Includes icons for file operations, search, and extensions.
- Title Bar:** Displays "ToBeInTheDocument.tsx — cm-react-unit-test-jest-demo1".
- Left Sidebar (EXPLORER):** Shows the project structure:
  - OPEN EDITORS: ToBeInTheDocument.tsx
  - CM-REACT-UNIT-TEST-JEST-DEMO1
    - node\_modules
    - public
    - src
      - components
        - ToBeDisabled.tsx
        - ToBeEmptyDOMElement.tsx
        - ToBeInTheDocument.tsx
        - ToContainElement.tsx
        - ToHaveValue.tsx
      - test
        - basic.test.tsx
- Right Editor Area:** Displays the code for `ToBeInTheDocument.tsx`.

```
import React from "react";
export default function ToBeInTheDocument() {
  return (
    <>
      <span data-testid="html-element">
        <span>Html Element</span>
      </span>
      <svg data-testid="svg-element"></svg>
    </>
  );
}
```



# React.JS UnitTest (Synchronous)



The screenshot shows a macOS desktop environment with a Code Editor window open. The window title is "basic.test.tsx — cm-react-unit-test-jest-demo1". The editor displays a TypeScript test file with Jest assertions for React components.

**File Explorer:**

- CM-REACT-UNIT-TEST-JEST-DEMO1
  - node\_modules
  - public
  - src
    - components
      - ToBeDisabled.tsx
      - ToBeEmptyDOMElement.tsx
      - ToBeInTheDocument.tsx
      - ToContainElement.tsx
      - ToHaveValue.tsx
    - test
      - basic.test.tsx
  - App.css
  - App.test.tsx

**Code Editor:**

```
basic.test.tsx
src > test > basic.test.tsx > ...
1 import { render, screen } from "@testing-library/react";
2 import ToBeDisabled from "../components/ToBeDisabled";
3 import ToBeEmptyDOMElement from "../components/ToBeEmptyDOMElement";
4 import ToBeInTheDocument from "../components/ToBeInTheDocument";
5 import ToContainElement from "../components/toContainElement";
6 import ToHaveValue from "../components/toHaveValue";
7
8 test("toBeInTheDocument", () => {
9   render(<ToBeInTheDocument />);
10
11   expect(screen.getByTestId("html-element")).toBeInTheDocument();
12   expect(screen.getByTestId("svg-element")).toBeInTheDocument();
13   expect(screen.queryByTestId("does-not-exist")).not.toBeInTheDocument();
14   expect(screen.getText("Html Element")).toBeInTheDocument();
15 });
16
17 > test("ToBeDisabled", () => {
18   ...
19 });
20
21
22
23 });
24
```



# React.JS UnitTest

The screenshot shows a Chrome browser window with the address bar pointing to [testing-library.com/docs/dom-testing-library/cheatsheet/#queries](https://testing-library.com/docs/dom-testing-library/cheatsheet/#queries). The page displays the 'Queries' section of the Testing Library documentation. On the left, a sidebar lists various query types: About Queries, ByRole, ByLabelText, ByPlaceholderText, ByText, ByDisplayValue, ByAltText, ByTitle, ByTestId, User Actions, Firing Events, Async Methods, and Appearance and Disappearance. The main content area is titled 'Queries' and contains a note: 'See [Which query should I use?](#)'. Below this is a table comparing six query methods based on the number of matches they return:

	No Match	1 Match	1+ Match	Await?
<b>getBy</b>	throw	return	throw	No
<b>findBy</b>	throw	return	throw	Yes
<b>queryBy</b>	null	return	throw	No
<b>getAllBy</b>	throw	array	array	No
<b>findAllBy</b>	throw	array	array	Yes
<b>queryAllBy</b>	[]	array	array	No



# React.JS UnitTest

src/App.test.js/describe

origin: [tomwayson/create-a...](#) 

```
1 describe('smoke tests', function() {
2   describe('w/o a previous session', function() {
3     const title = 'Test Title';
4     it('renders app title and sign in button', () => {
5       const { getAllByText } = render(
6         <Router>
7           <App title={title} />
8         </Router>
9       );
10      expect(getAllByText(title)[0]).toBeInTheDocument();
11      expect(getAllByText('Sign In')[0]).toBeInTheDocument();
12    });
13    it('matches snapshot', () => {
14      const { asFragment } = render(
15        <Router>
16          <App title={title} />
17        </Router>
18      );
19      expect(asFragment()).toMatchSnapshot();
20    });
21  });
22});
```



# Tailwind + React.JS



สอนติดตั้ง Tailwind

uu React.JS

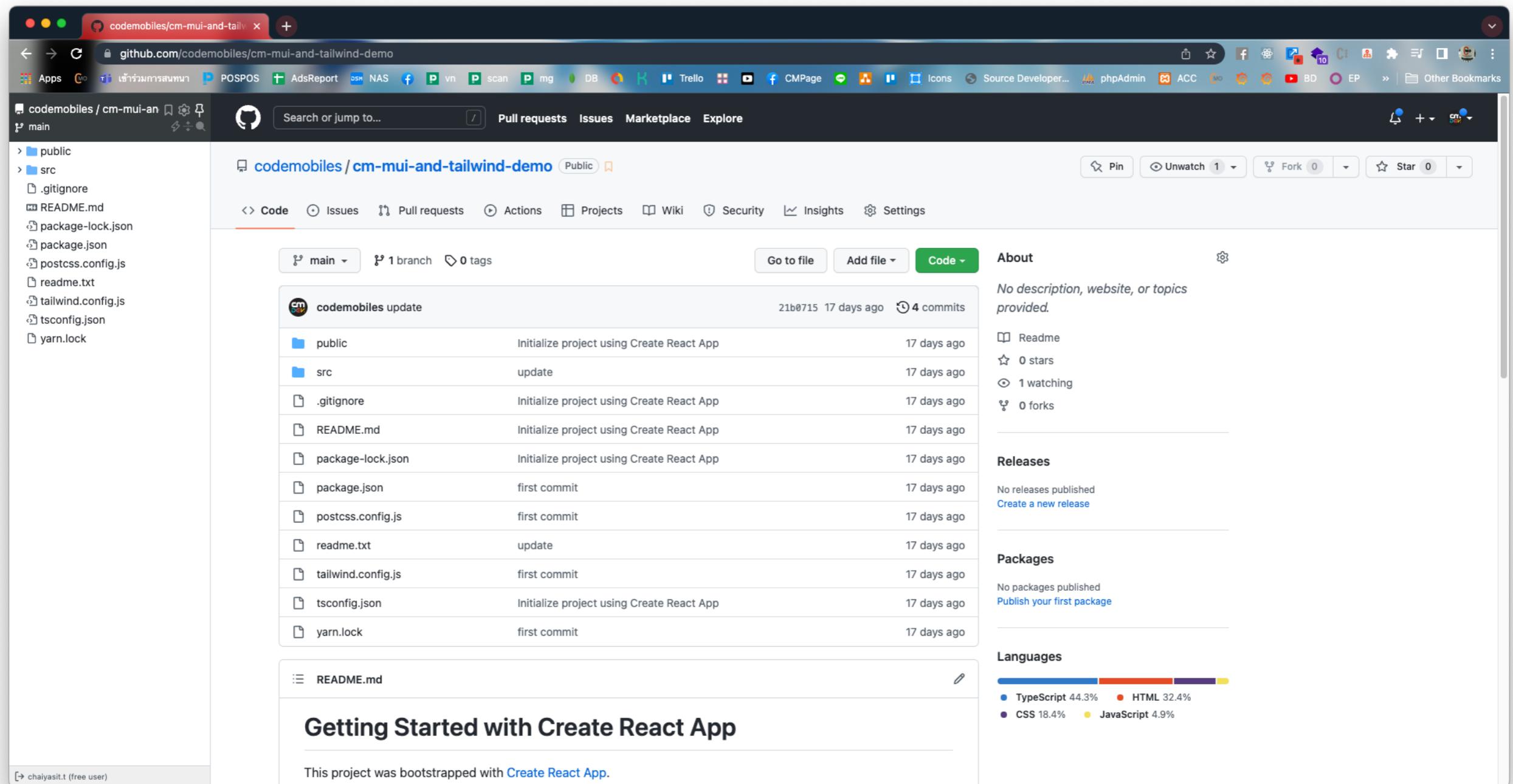
Material UI V.5 (TS)



# Tailwind + React.JS



<https://github.com/codemobiles/cm-mui-and-tailwind-demo>



The screenshot shows a GitHub repository page for 'codemobiles/cm-mui-and-tailwind-demo'. The repository has 1 branch and 0 tags. The commit history shows 4 commits from 'codemobiles' made 17 days ago. The files listed in the commit history include public, src, .gitignore, README.md, package-lock.json, package.json, postcss.config.js, readme.txt, tailwind.config.js, tsconfig.json, and yarn.lock. The repository has 0 stars, 1 watching, and 0 forks. It also has no releases published and no packages published. The Languages section shows TypeScript (44.3%), HTML (32.4%), CSS (18.4%), and JavaScript (4.9%). The README.md file contains the text 'Getting Started with Create React App' and a note that the project was bootstrapped with Create React App.

# Tailwind + React.JS



A screenshot of the VS Code interface showing the README file for a project named "cm-mui-and-tailwind-demo". The README contains instructions for setting up the project, including commands like `npx create-react-app cm-mui-and-tailwind-demo --template typescript` and `yarn add @mui/material @emotion/react @emotion/styled`. It also includes steps for tailwindcss initialization and configuration.

```
You, 7 minutes ago | 1 author (You)
1 # https://github.com/codemobiles/cm-mui-and-tailwind-demo
2 npx create-react-app cm-mui-and-tailwind-demo --template typescript
3 yarn add @mui/material @emotion/react @emotion/styled
4 yarn add @mui/icons-material
5
6
7 - npx tailwindcss init -p
8 - In index.css, add the last line
9
10 @tailwind components;
11 @tailwind utilities;
12
13 - In tailwind.config.js, update content and important
14
15 /** @type {import('tailwindcss').Config} */
16 module.exports = {
17   content: ['./src/**/*.{js,jsx,ts,tsx}'],
18   important: '#root',
19   theme: {} You, 2 weeks ago • first commit
20     extend: {},
21   },
22   plugins: [],
23 }
24
25 - copy and paste InjectTailwind.tsx in src
26 - In index.tsx,
27   import InjectTailwind from './InjectTailwind'
28   and
29   <InjectTailwind>
30     <App />
31   </InjectTailwind>
32
33
34
35
36 # vscode extension for tailwind
37 bradlc.vscode-tailwindcss
38
39 # document You, 2 weeks ago • first commit
40 https://tailwindcss.com/
41
42 # Useful example
43 <div className="hover:text-yellow-400">1234</div>
44
45
46 # youtube
47 https://www.youtube.com/watch?v=hgMP2ulxPlA
```

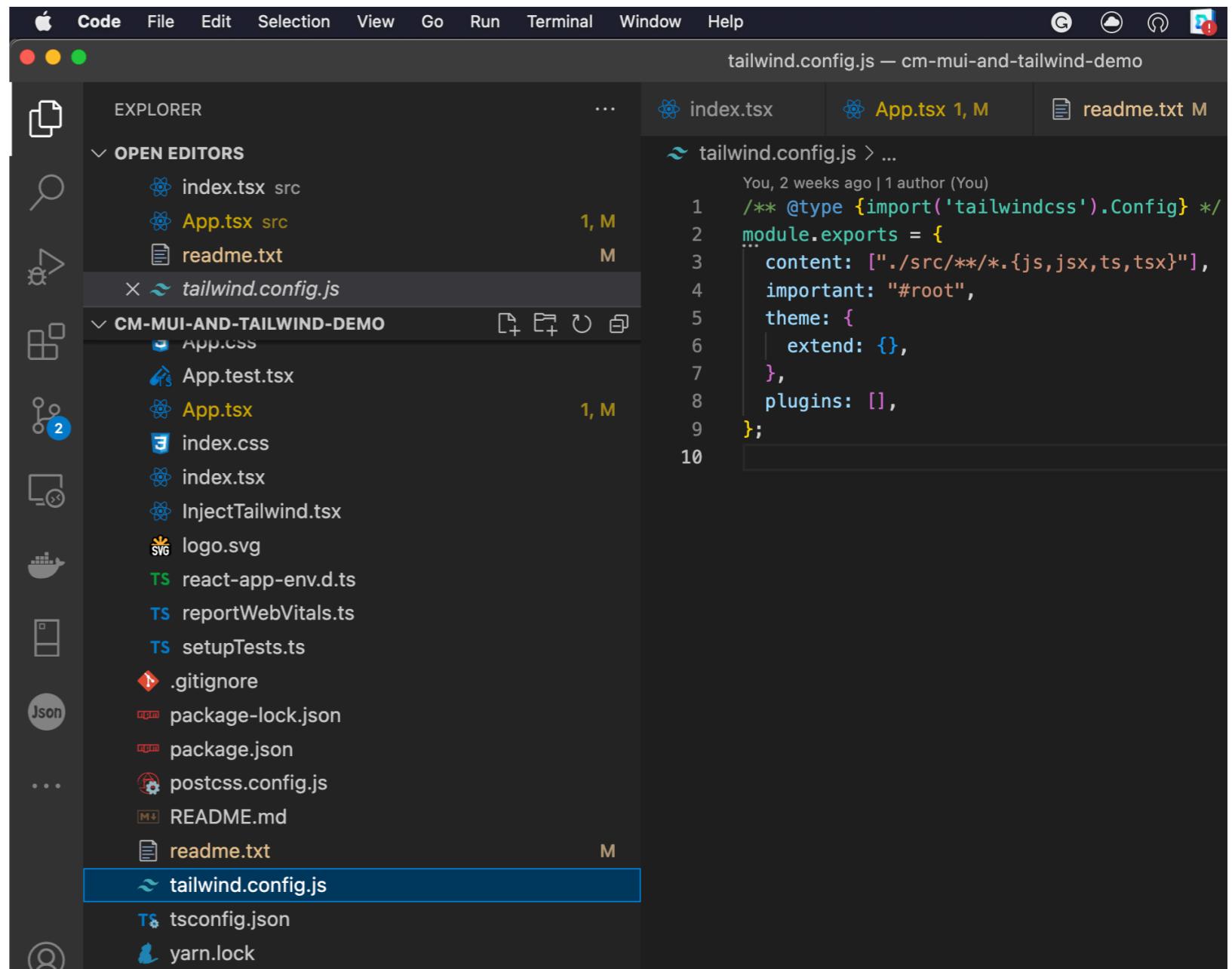
A screenshot of the VS Code interface showing the README file for the same project. This version of the README is more concise, providing links to the Tailwind CSS documentation and a YouTube tutorial.

```
readme.txt — cm-mui-and-tailwind-demo
readme.txt M X
readme.txt
You, 7 minutes ago | 1 author (You)
1 # https://github.com/codemobiles/cm-mui-and-tailwind-demo
2 npx create-react-app cm-mui-and-tailwind-demo --template typescript
3 yarn add @mui/material @emotion/react @emotion/styled
4 yarn add @mui/icons-material
5
6
7 - copy and paste InjectTailwind.tsx in src
8 - In index.tsx,
9   import InjectTailwind from './InjectTailwind'
10  and
11  <InjectTailwind>
12    <App />
13  </InjectTailwind>
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36 # vscode extension for tailwind
37 bradlc.vscode-tailwindcss
38
39 # document You, 2 weeks ago • first commit
40 https://tailwindcss.com/
41
42 # Useful example
43 <div className="hover:text-yellow-400">1234</div>
44
45
46 # youtube
47 https://www.youtube.com/watch?v=hgMP2ulxPlA
```

# Tailwind + React.JS



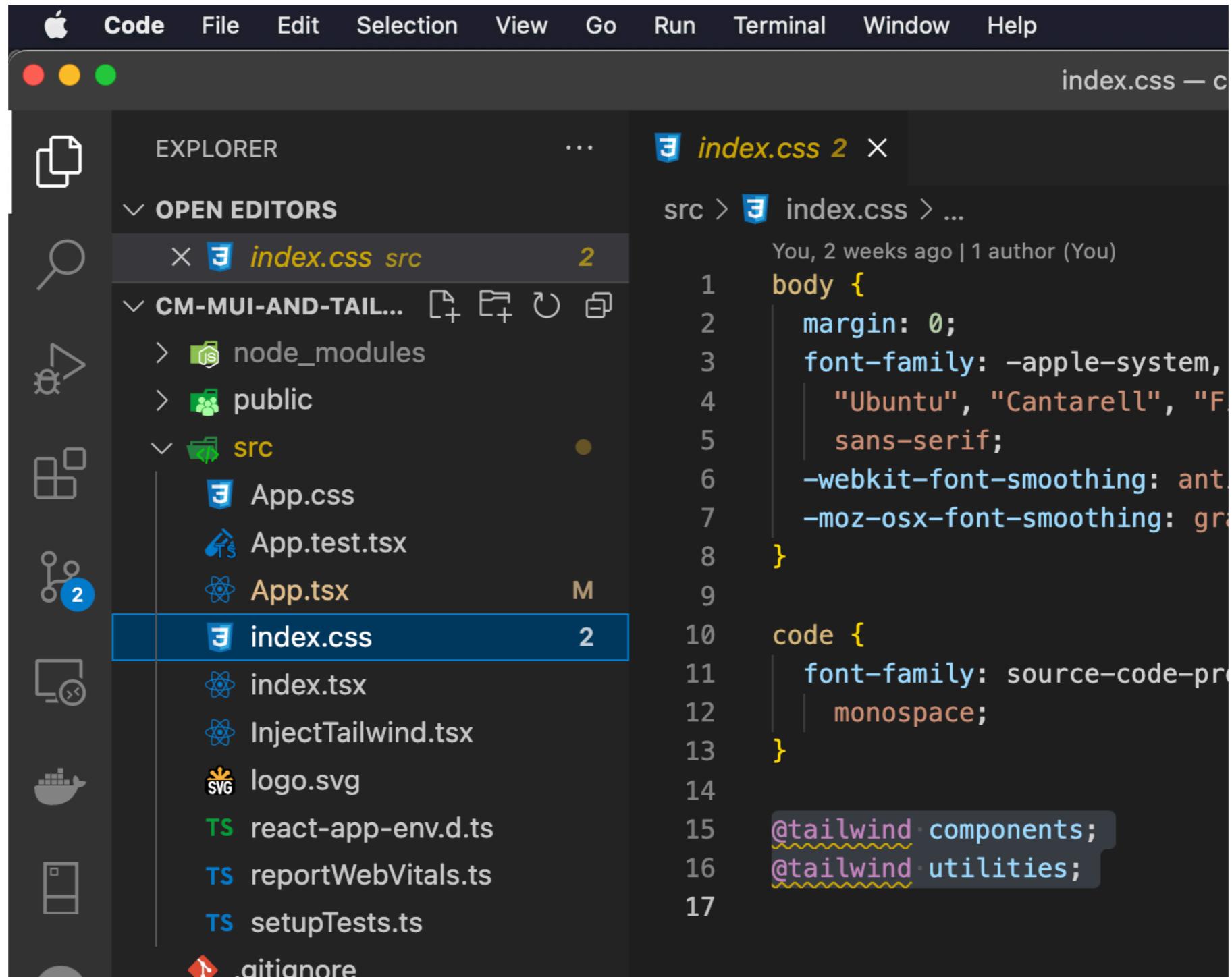
npx tailwindcss init -p



A screenshot of the VS Code interface. The left sidebar shows the file structure of a React application named "CM-MUI-AND-TAILWIND-DEMO". The "tailwind.config.js" file is selected in the Explorer and is open in the main editor. The editor tab bar shows "tailwind.config.js — cm-mui-and-tailwind-demo". The code in the editor is:

```
/* @type {import('tailwindcss').Config} */
module.exports = {
  content: ["./src/**/*.{js,jsx,ts,tsx}"],
  important: "#root",
  theme: {
    extend: {},
  },
  plugins: [],
};
```

# Tailwind + React.JS



A screenshot of the Visual Studio Code (VS Code) interface. The title bar shows "index.css — c". The left sidebar has icons for file operations, search, file tree, and other developer tools. The main area shows the "EXPLORER" view with a list of files and folders. The "OPEN EDITORS" section shows two files: "index.css" (selected) and "src/index.css". The "index.css" editor pane displays the following CSS code:

```
body {  
  margin: 0;  
  font-family: -apple-system,  
  "Ubuntu", "Cantarell", "F  
  sans-serif;  
  -webkit-font-smoothing: ant  
  -moz-osx-font-smoothing: gr  
 }  
  
code {  
  font-family: source-code-pr  
  monospace;  
}  
  
@tailwind components;  
@tailwind utilities;
```

# Tailwind + React.JS

A screenshot of the Visual Studio Code interface. The title bar shows "tailwind.config.js — cm-mui-and-tailwind". The left sidebar is the Explorer, showing a project structure with files like "readme.txt", "tailwind.config.js", "node\_modules", "public", "src", ".gitignore", "package-lock.json", "package.json", "postcss.config.js", "README.md", and "tsconfig.json". The "tailwind.config.js" file is selected in the Explorer and is also the active editor in the main pane. The code in the editor is:

```
/* @type {import('tailwindcss').Config} */
module.exports = {
  content: ['./src/**/*.{js,jsx,ts,tsx}'],
  important: "#root",
  theme: {
    extend: {},
  },
  plugins: [],
};
```

# Tailwind + React.JS



A screenshot of the Visual Studio Code interface. The title bar shows "InjectTailwind.tsx — cm-mui-and-tailwind-demo". The left sidebar is the Explorer view, showing a project structure with files like "readme.txt", "InjectTailwind.tsx" (which is selected), and "tailwind.config.js". The main editor area displays the following code:

```
You, 2 weeks ago | 1 author (You)
1 import React from 'react'
2 import { StyledEngineProvider } from '@mui/material/styles'
3
4 export default function InjectTailwind({ children }: any) {
5   return <StyledEngineProvider injectFirst>{children}</StyledEngineProvider>
6 }
7
```

# Tailwind + React.JS



```
index.tsx — cm-mui-and-tailwind-demo

readme.txt M index.tsx X InjectTailwind.tsx tailwind.config.js

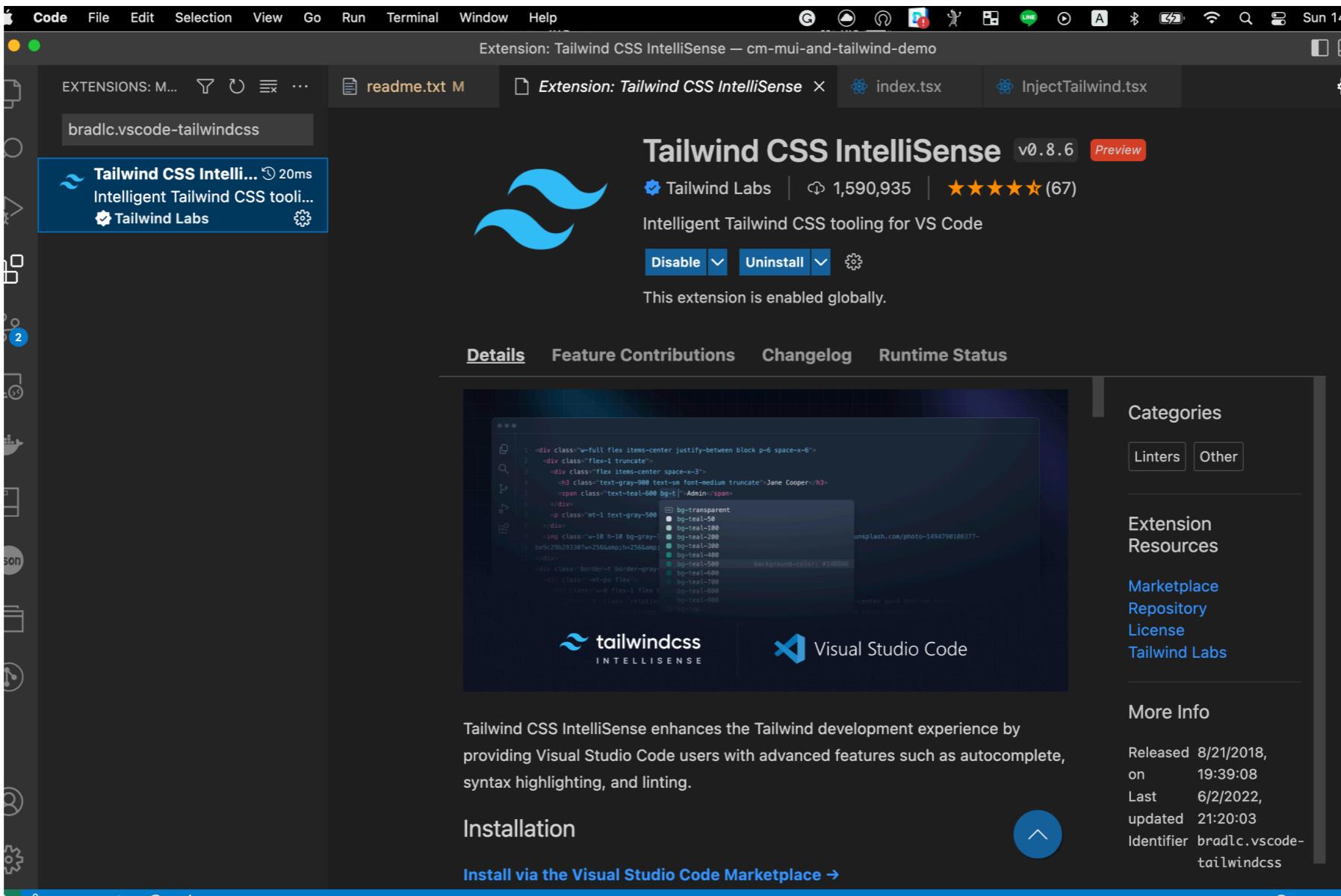
src > index.tsx > ...

8  const root = ReactDOM.createRoot(
9    | document.getElementById("root") as HTMLElement
10   );
11  root.render(
12    <React.StrictMode>
13      <InjectTailwind>
14        <App />
15      </InjectTailwind>
16    </React.StrictMode>
17  );
18
19 // If you want to start measuring performance in your app, pass a function
20 // to log results (for example: reportWebVitals(console.log))
21 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
22 reportWebVitals();
23
```

# Tailwind + React.JS



## bradlc.vscode-tailwindcss



The screenshot shows the Visual Studio Code extension marketplace. The search bar at the top contains "bradlc.vscode-tailwindcss". Below it, the results list the "Tailwind CSS IntelliSense" extension by Tailwind Labs. The extension card includes the following details:

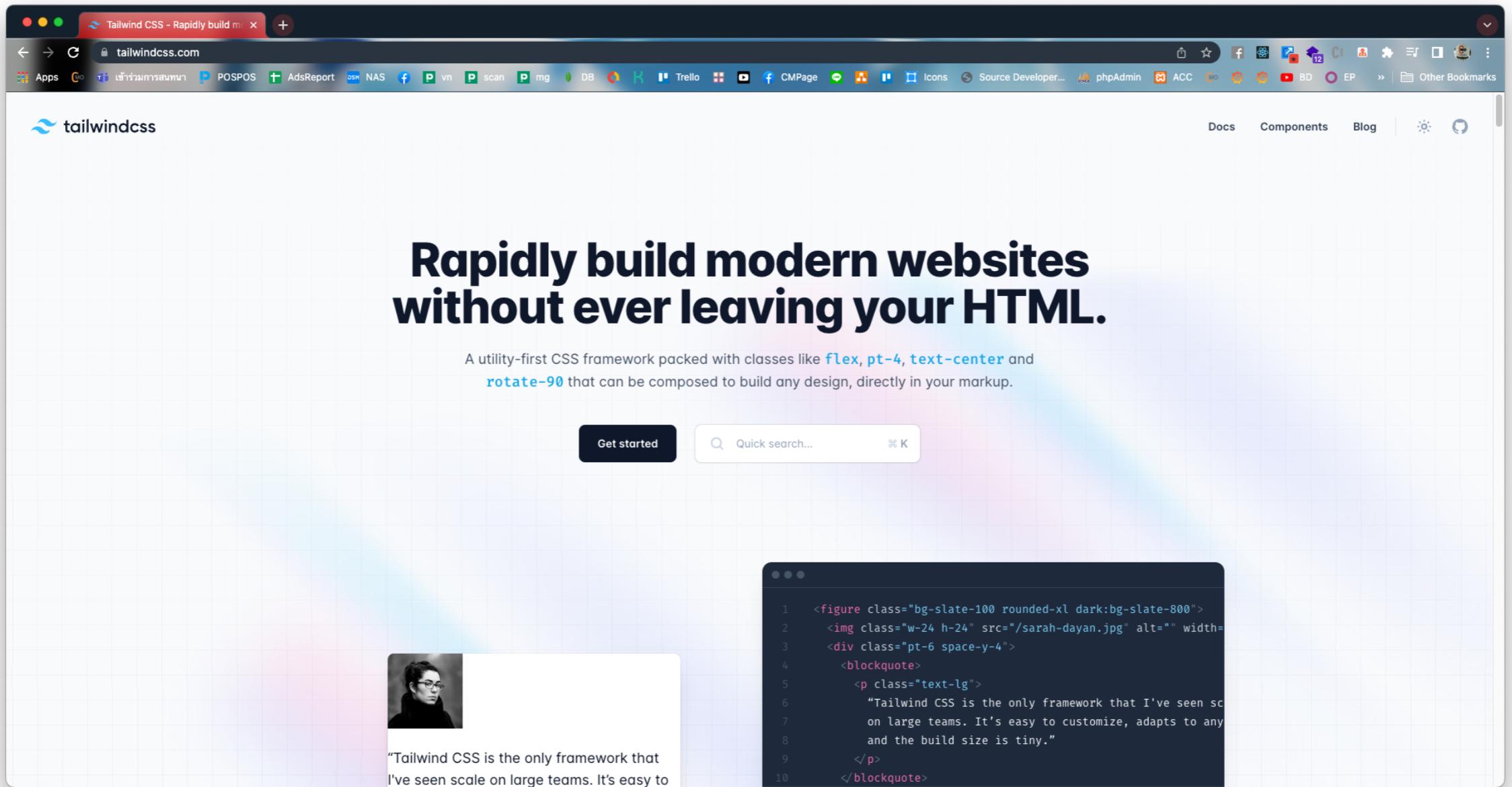
- Extension: Tailwind CSS IntelliSense — cm-mui-and-tailwind-demo**
- v0.8.6** (Preview)
- Tailwind Labs** | 1,590,935 installs | ★★★★★ (67) rating
- Intelligent Tailwind CSS tooling for VS Code**
- Disable** | **Uninstall** | **⚙️**
- A note: "This extension is enabled globally."

The main content area displays a screenshot of the extension in action within VS Code, showing a code editor with Tailwind CSS classes like `.bg-teal-500` and `.text-gray-900` being autocompleted. Below the screenshot, the text reads: "Tailwind CSS IntelliSense enhances the Tailwind development experience by providing Visual Studio Code users with advanced features such as autocomplete, syntax highlighting, and linting." A "More Info" section on the right provides release details: Released 8/21/2018, on 19:39:08, Last 6/2/2022, updated 21:20:03, Identifier bradlc.vscode-tailwindcss.

# Tailwind + React.JS



<https://tailwindcss.com/>



The screenshot shows the homepage of tailwindcss.com. The header features the Tailwind logo and navigation links for Docs, Components, and Blog. The main headline reads: "Rapidly build modern websites without ever leaving your HTML." Below it, a subtext explains: "A utility-first CSS framework packed with classes like `flex`, `pt-4`, `text-center` and `rotate-90` that can be composed to build any design, directly in your markup." A "Get started" button and a search bar are visible. On the left, there's a portrait of a person and a testimonial quote from Sarah Dayan.

**Rapidly build modern websites without ever leaving your HTML.**

A utility-first CSS framework packed with classes like `flex`, `pt-4`, `text-center` and `rotate-90` that can be composed to build any design, directly in your markup.

Get started

Quick search... ⌘ K

  
"Tailwind CSS is the only framework that I've seen scale on large teams. It's easy to

```
1 <figure class="bg-slate-100 rounded-xl dark:bg-slate-800">
2   
4     <blockquote>
5       <p class="text-lg">
6         "Tailwind CSS is the only framework that I've seen sc
7         on large teams. It's easy to customize, adapts to any
8         and the build size is tiny."
9       </p>
10      </blockquote>
```

# Tailwind + React.JS



A screenshot of the Tailwind CSS documentation website. The search bar at the top contains the query "Border". The results are categorized under "Borders" and "Tables".

- Borders**
  - # [Border Width](#)
  - # [Border Style](#)
  - # [Border Radius](#)
  - # [Border Color](#)
  - # [Divide Width](#)
    - Add [borders](#) between horizontal children
- Tables**
  - ## [Border Spacing](#)
  - # [Border Spacing](#)
    - Setting the [border](#) spacing

Search by algolia

# Tailwind + React.JS



Border Width - Tailwind CSS

tailwindcss.com/docs/border-width

tailwindcss v3.1.8 Tailwind CSS v3.1 · Arbitrary variants, TypeScript types, and more >

Docs Components Blog

Utilities for controlling the width of an element's borders.

Quick search... ⌘K

**Borders**

- Background Color
- Background Origin
- Background Position
- Background Repeat
- Background Size
- Background Image
- Gradient Color Stops
- Border Width**
- Border Color
- Border Style
- Divide Width
- Divide Color
- Divide Style
- Outline Width
- Outline Color
- Outline Style
- Outline Offset
- Ring Width
- Ring Color
- Ring Offset Width
- Ring Offset Color

Class	Properties
<code>border-8</code>	<code>border-width: 8px;</code>
<code>border</code>	<code>border-width: 1px;</code>
<code>border-x-0</code>	<code>border-left-width: 0px; border-right-width: 0px;</code>
<code>border-x-2</code>	<code>border-left-width: 2px; border-right-width: 2px;</code>
<code>border-x-4</code>	<code>border-left-width: 4px; border-right-width: 4px;</code>
<code>border-x-8</code>	<code>border-left-width: 8px; border-right-width: 8px;</code>

**Basic usage**

**All sides**

Use the `border`, `border-0`, `border-2`, `border-4`, or `border-8` utilities to set the border width for all sides of an element.



**On this page**

- Quick reference
- Basic usage
  - > All sides
  - > Individual sides
  - > Horizontal and vertical sides
  - > Between elements
- Applying conditionally
  - > Hover, focus, and other states
  - > Breakpoints and media queries
- Using custom values
  - > Customizing your theme
  - > Arbitrary values

# Tailwind + React.JS

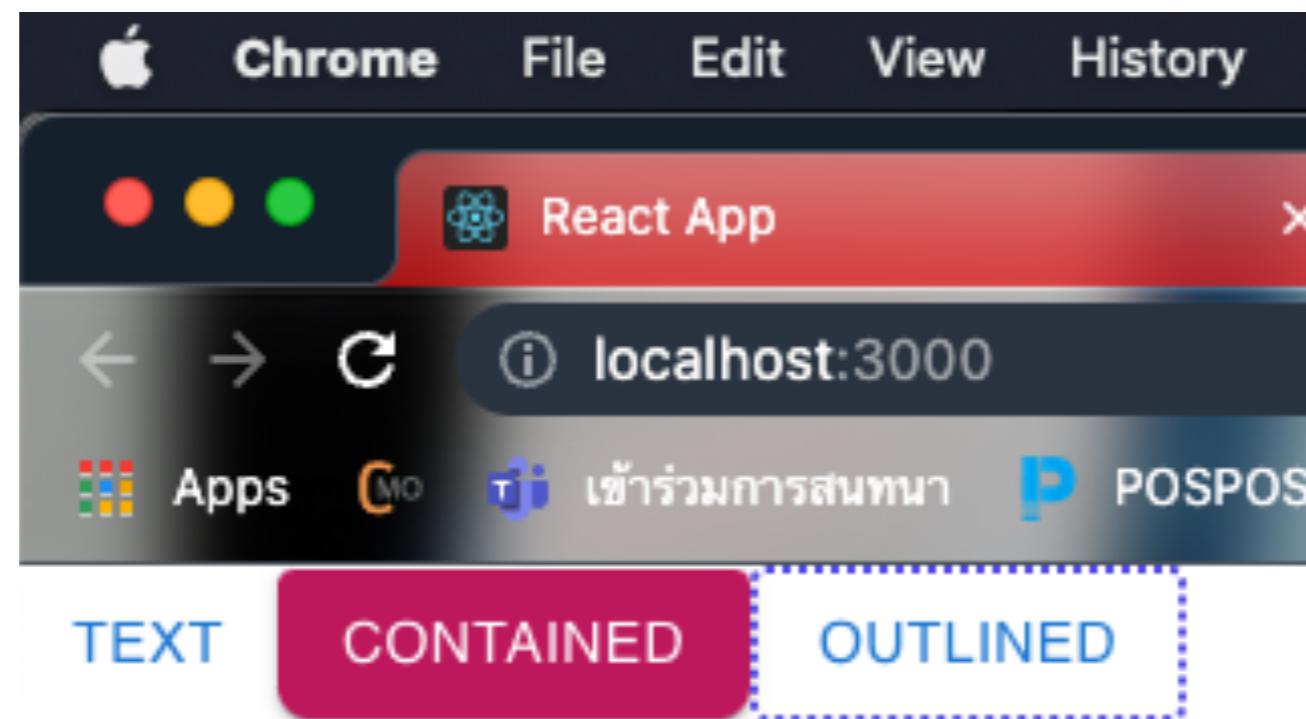


A screenshot of a code editor showing the file `App.tsx`. The code uses Tailwind CSS utility classes like `bg-pink-700`, `text-white`, `hover:bg-pink-300`, `rounded-b-lg`, `variant="text"`, `variant="contained"`, `variant="outlined"`, `border-dotted border-2 border-indigo-600`, and `border-2 border-indigo-600`. The code defines a function `App()` that returns a JSX structure with two buttons: one with the `variant="text"` prop and one with the `variant="contained"` prop. Both buttons have Tailwind CSS classes applied to their `Button` components.

```
App.tsx — cm-mui-and-tailwind-demo
readme.txt M App.tsx M Extension: Tailwind CSS IntelliSense index.tsx ⌂ ⌂ ⌂

src > App.tsx > ...
You, 14 seconds ago | 1 author (You)
1 import "./App.css";
2 import { Button } from "@mui/material";
3
4 function App() {
5   return (
6     <>
7       <Button variant="text">Text</Button>
8       <Button
9         className="bg-pink-700 text-white hover:bg-pink-300 rounded-b-lg "
10        variant="contained"
11       >
12         Contained
13       </Button>
14       <Button
15         className="border-dotted border-2 border-indigo-600"
16        variant="outlined"
17       >
18         Outlined
19       </Button>
20     </>
21   );
22 }
23
24 export default App;
25
```

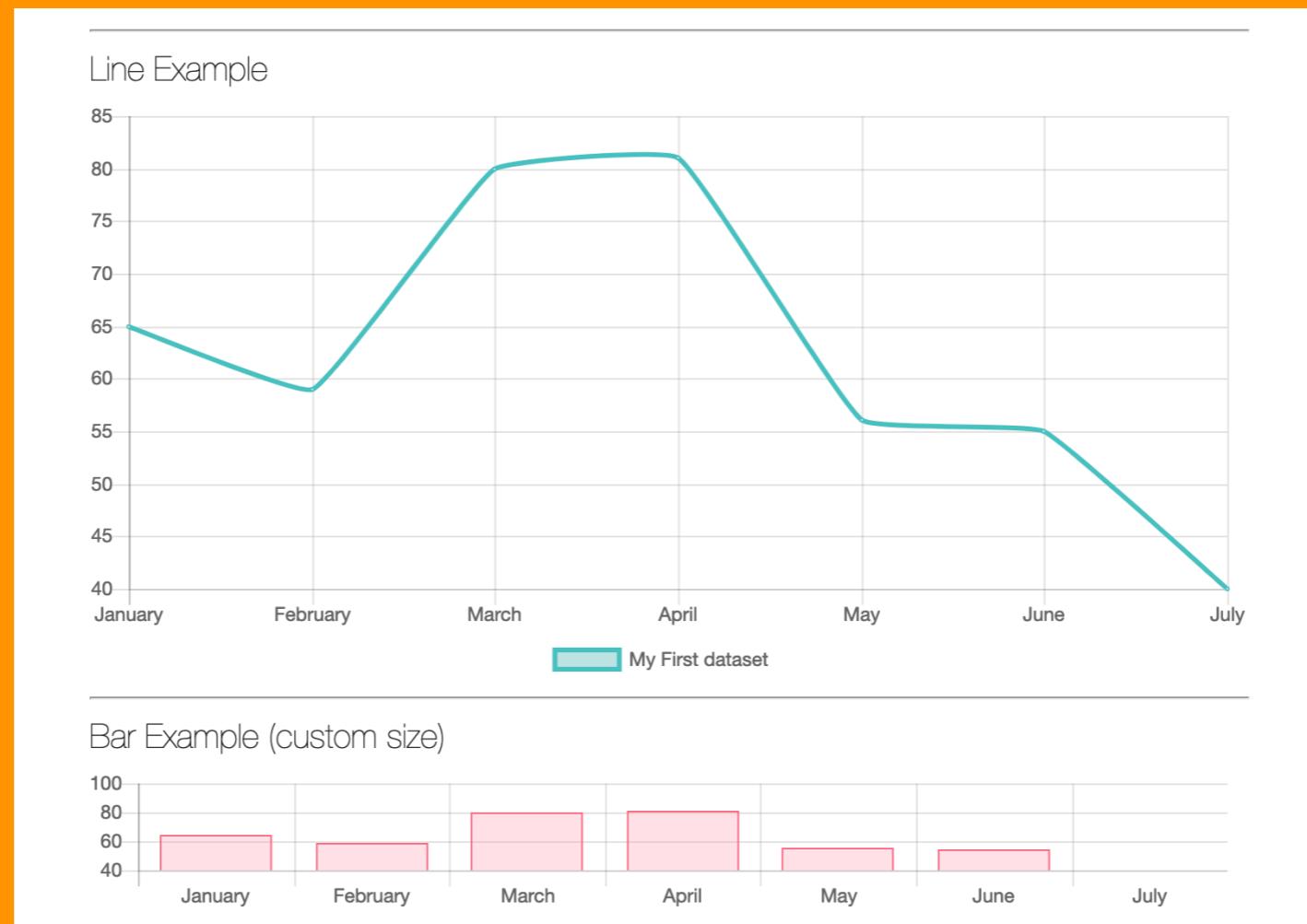
# Tailwind + React.JS



```
className="bg-pink-700 text-white hover:bg-pink-300 rounded-b-lg "
```

```
className="border-dotted border-2 border-indigo-600"
```

# Chart



# ChartJS

```
npm i react-chartjs-2 chart.js
```

Live demo: [jerairrest.github.io/react-chartjs-2](http://jerairrest.github.io/react-chartjs-2)

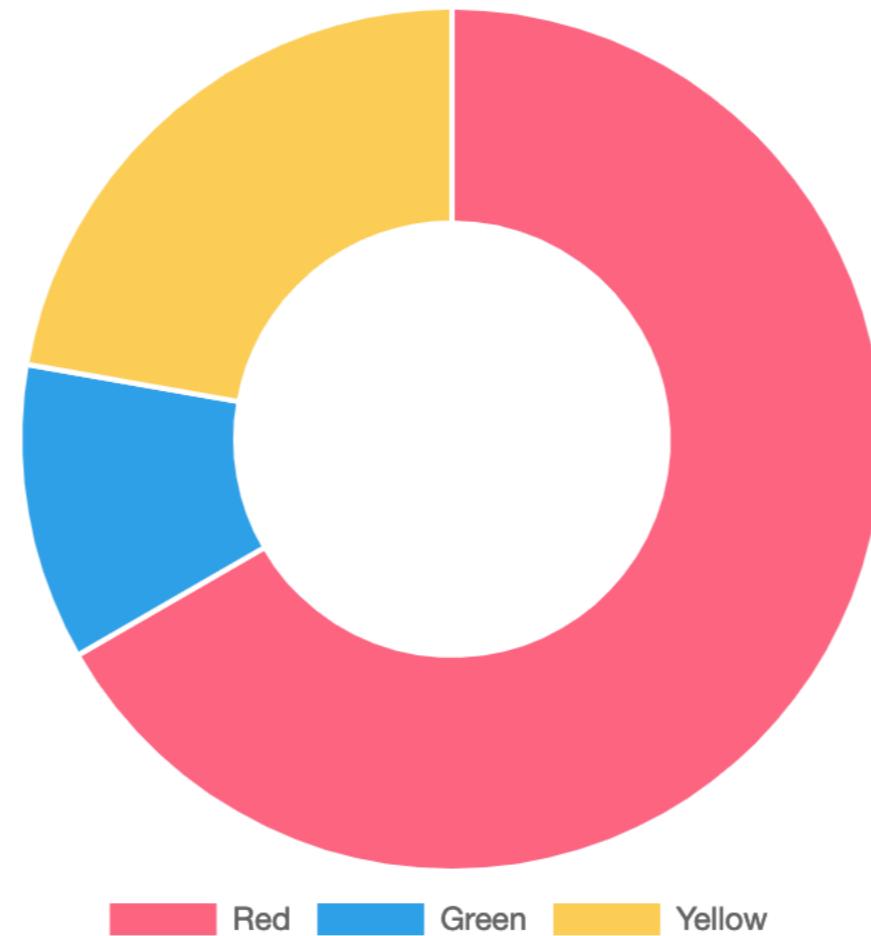
# ChartJS

react-chartjs-2

[View project on GitHub](#)

---

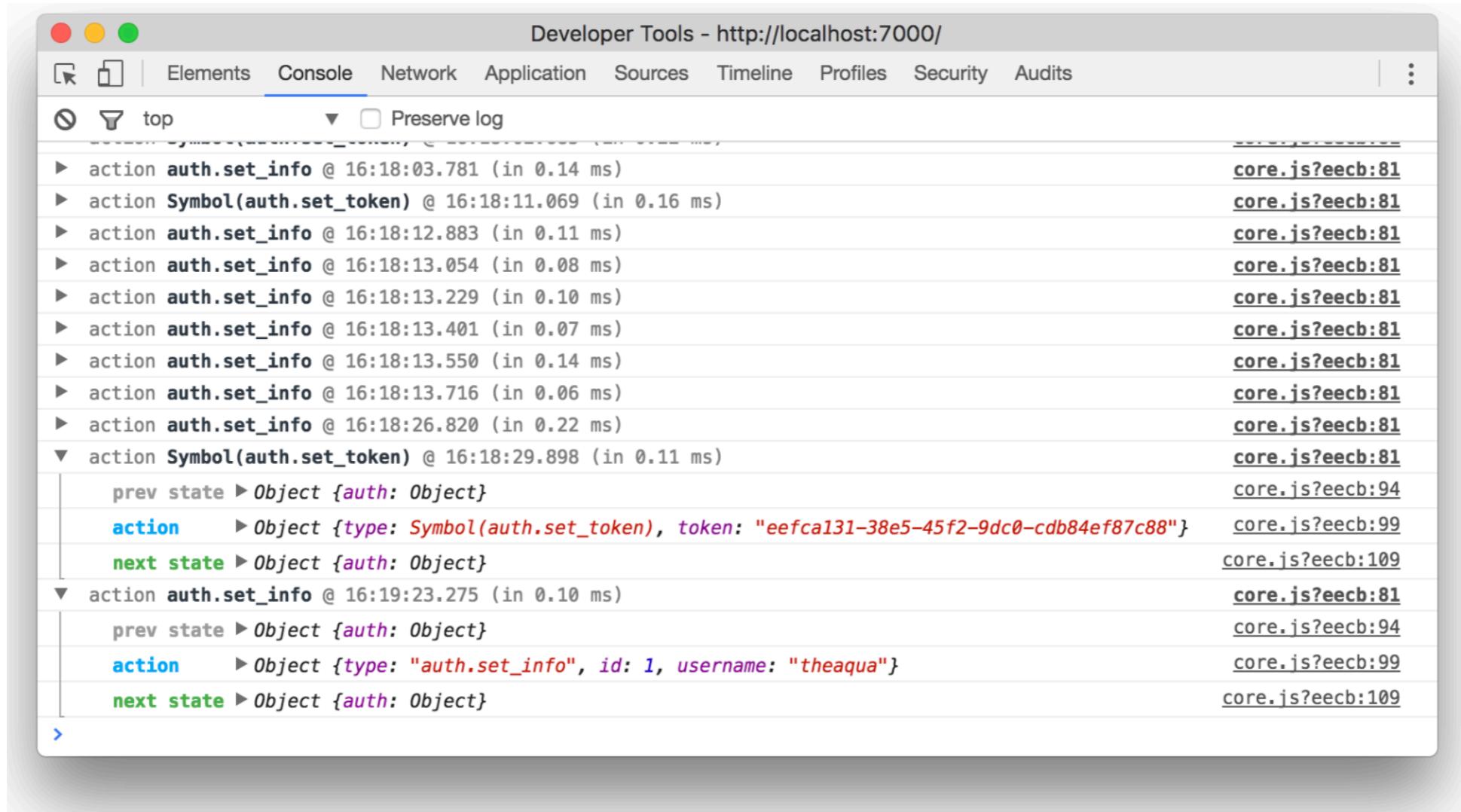
Doughnut Example



# Log and Debug

# Redux-Logger

<https://github.com/LogRocket/redux-logger>

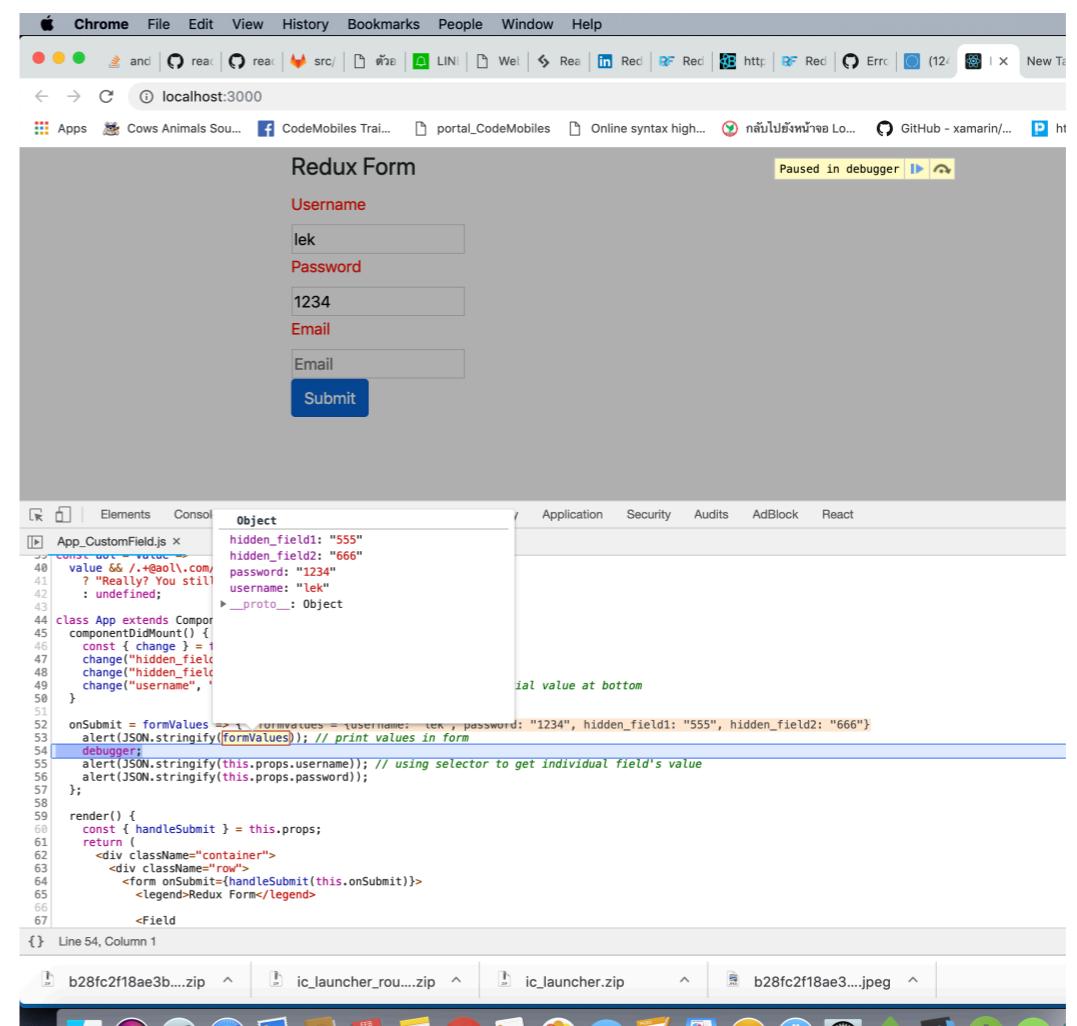


The screenshot shows the Chrome Developer Tools interface with the 'Console' tab selected. The title bar reads 'Developer Tools - http://localhost:7000/'. The console log displays a series of Redux actions and their corresponding state changes, color-coded by type (e.g., auth.set\_info in purple, auth.set\_token in red). The log entries are as follows:

- ▶ action `auth.set_info` @ 16:18:03.781 (in 0.14 ms) core.js?eeccb:81
- ▶ action `Symbol(auth.set_token)` @ 16:18:11.069 (in 0.16 ms) core.js?eeccb:81
- ▶ action `auth.set_info` @ 16:18:12.883 (in 0.11 ms) core.js?eeccb:81
- ▶ action `auth.set_info` @ 16:18:13.054 (in 0.08 ms) core.js?eeccb:81
- ▶ action `auth.set_info` @ 16:18:13.229 (in 0.10 ms) core.js?eeccb:81
- ▶ action `auth.set_info` @ 16:18:13.401 (in 0.07 ms) core.js?eeccb:81
- ▶ action `auth.set_info` @ 16:18:13.550 (in 0.14 ms) core.js?eeccb:81
- ▶ action `auth.set_info` @ 16:18:13.716 (in 0.06 ms) core.js?eeccb:81
- ▶ action `auth.set_info` @ 16:18:26.820 (in 0.22 ms) core.js?eeccb:81
- ▼ action `Symbol(auth.set_token)` @ 16:18:29.898 (in 0.11 ms) core.js?eeccb:81
  - prev state ▶ Object {auth: Object} core.js?eeccb:94
  - action ▶ Object {type: `Symbol(auth.set_token)`, token: "eefca131-38e5-45f2-9dc0-cdb84ef87c88"} core.js?eeccb:99
  - next state ▶ Object {auth: Object} core.js?eeccb:109
- ▼ action `auth.set_info` @ 16:19:23.275 (in 0.10 ms) core.js?eeccb:81
  - prev state ▶ Object {auth: Object} core.js?eeccb:94
  - action ▶ Object {type: "auth.set\_info", id: 1, username: "thequa"} core.js?eeccb:99
  - next state ▶ Object {auth: Object} core.js?eeccb:109

# Debug

```
onSubmit = formValues => {
  alert(JSON.stringify(formValues));
  debugger;
  alert(JSON.stringify(this.props.username));
  alert(JSON.stringify(this.props.password));
};
```



# Deployment

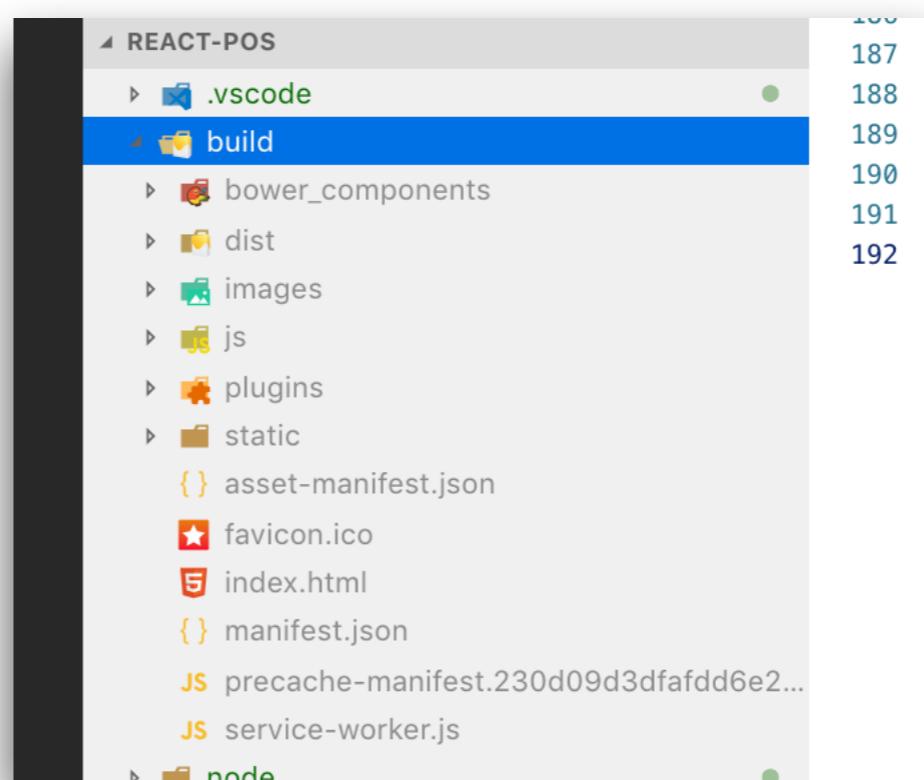
# Deployment

- npm run build
- Test with [npm i -g serve] - run serve
- Deploy in nginx
- Fix
  - Fallback Issue
  - Subfolder
- Run
- Optional
  - Separating Development and Production Configuration
  - Disable Redux-Logger

# Deployment

## npm run build

- npm run build
- npm i -g serve
- serve -s build



# Deployment

npm i -g serve

- npm i -g serve
- serve -s build

```
chaiyasits-MacBook-Pro-3:react-pos chaiyasit$ sudo npm install -g serve
Password: Cmd + click to follow link
Sorry, try again.
Password:
/usr/local/bin/serve -> /usr/local/lib/node_modules/serve/bin/serve.js
+ serve@10.1.2
added 78 packages from 39 contributors in 1.855s
chaiyasits-MacBook-Pro-3:react-pos chaiyasit$ cd build/
chaiyasits-MacBook-Pro-3:build chaiyasit$ serve
```

Serving!

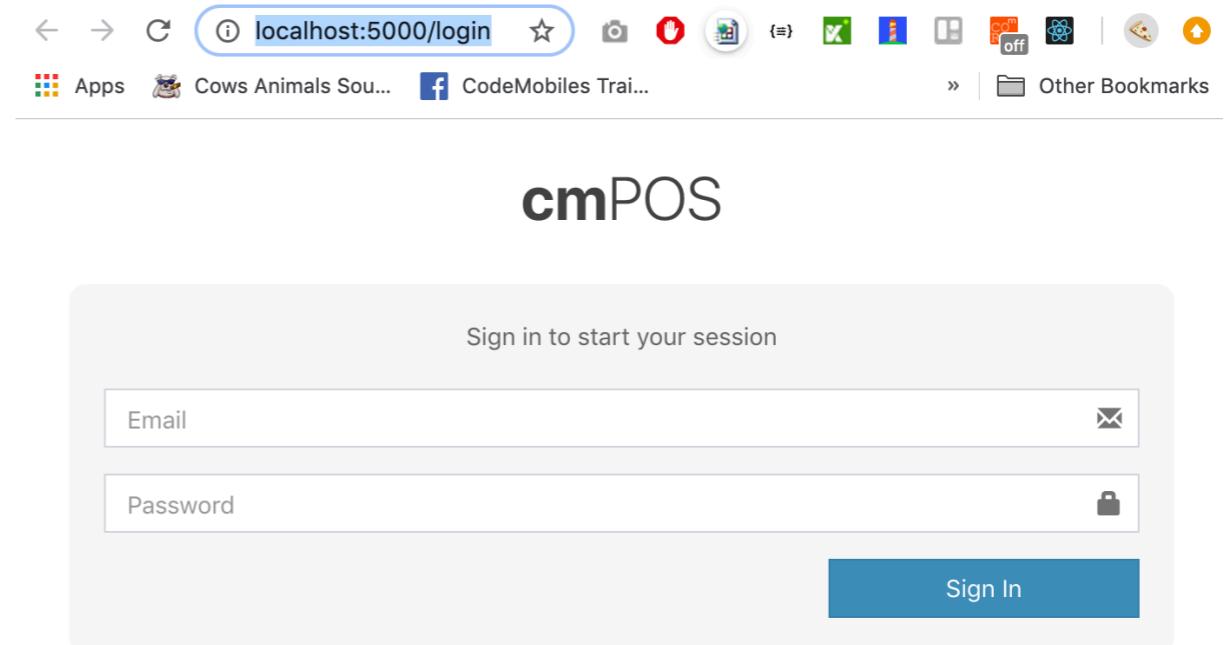
- Local: http://localhost:5000  
- On Your Network: http://192.168.0.107:5000

Copied local address to clipboard!

# Deployment

npm i -g serve

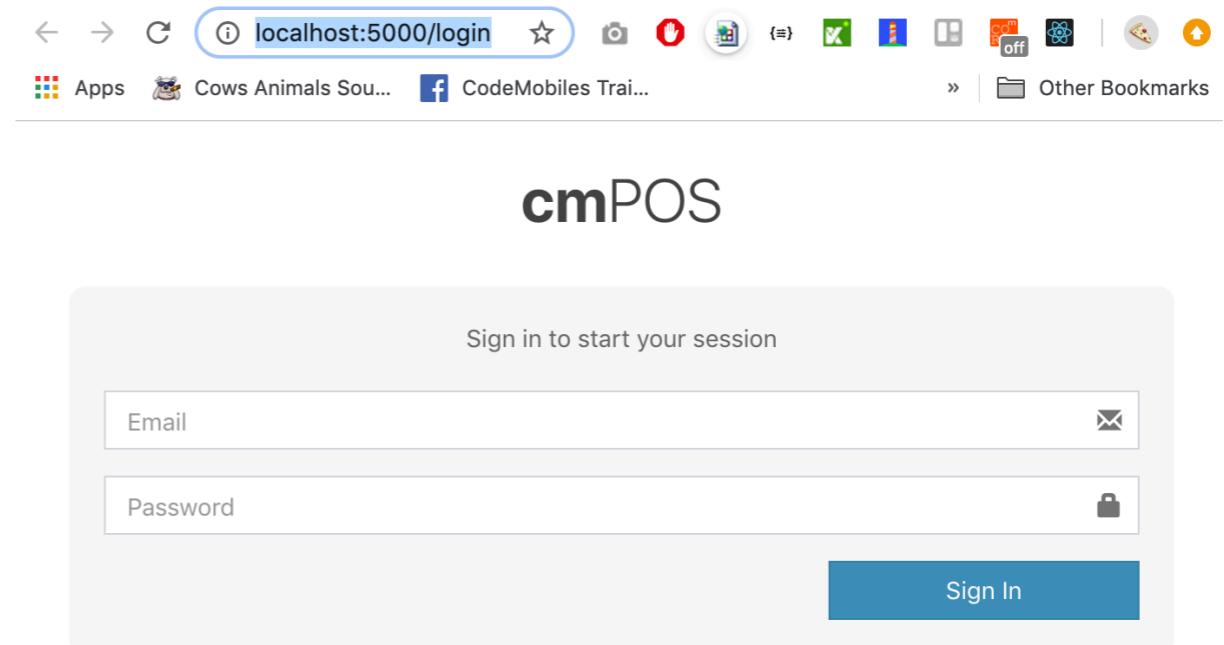
- serve
- Open localhost:5000



# Deployment

npm i -g serve

- serve
- Open localhost:5000

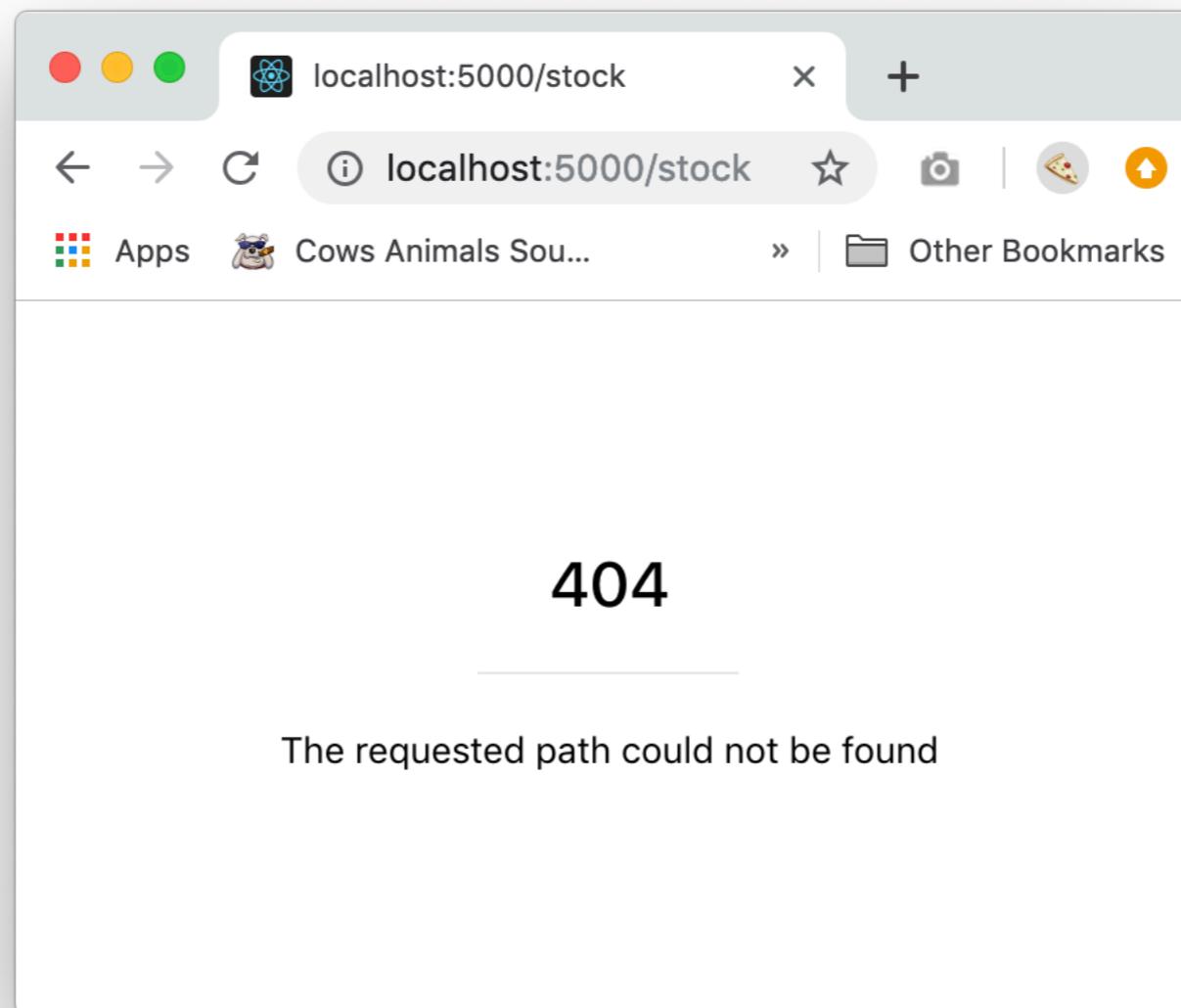


# Fallback 404 Error

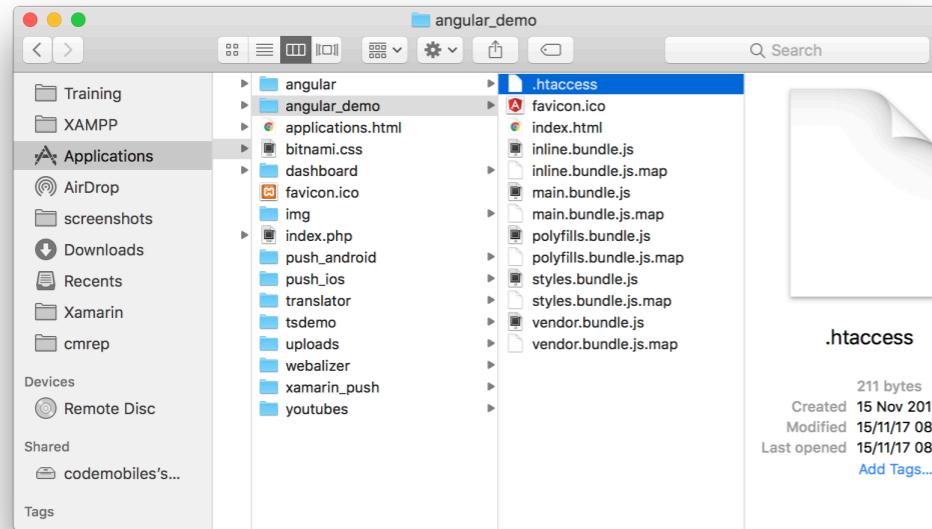
Cross-Origin Resource Sharing

# Using Fallback

404 error when enter url from browser's address bar directly



# Using Fallback (APACHE)

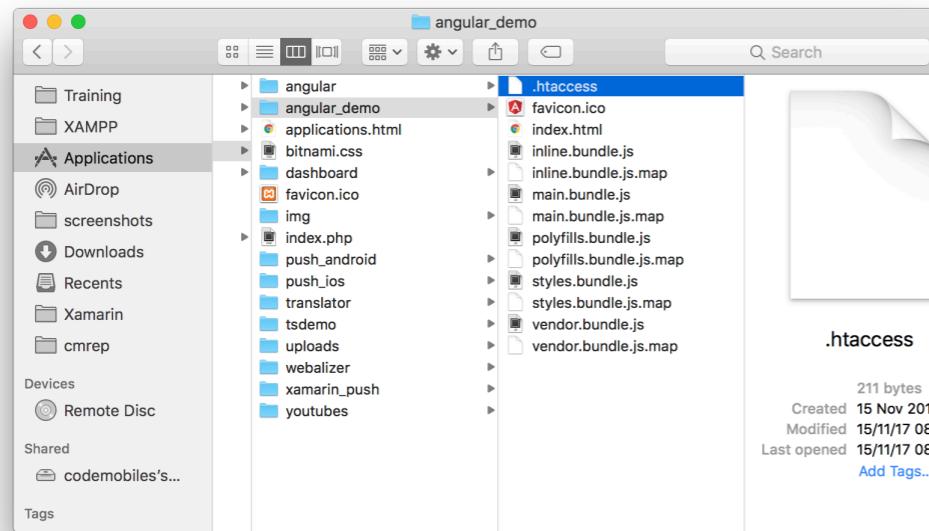


## Case of Running on Apache

```
<ifModule mod_rewrite.c>
    RewriteEngine On
    RewriteCond %{REQUEST_FILENAME} !-f
    RewriteCond %{REQUEST_FILENAME} !-d
    RewriteCond %{REQUEST_URI} !index
    RewriteRule (.*) index.html [L]
</ifModule>
```

1. Create file .htaccess in the root deployed angular folder
2. Write fallback-configuration to make all request to index.html. This makes angular to translate url request as routing command correctly

# Using Fallback (NGINX)



## Case of Running on **NGINX**

### # in case of no sub-folder

```
location / {  
    try_files $uri $uri/ /index.html;  
}
```

### # in case of under demo folder

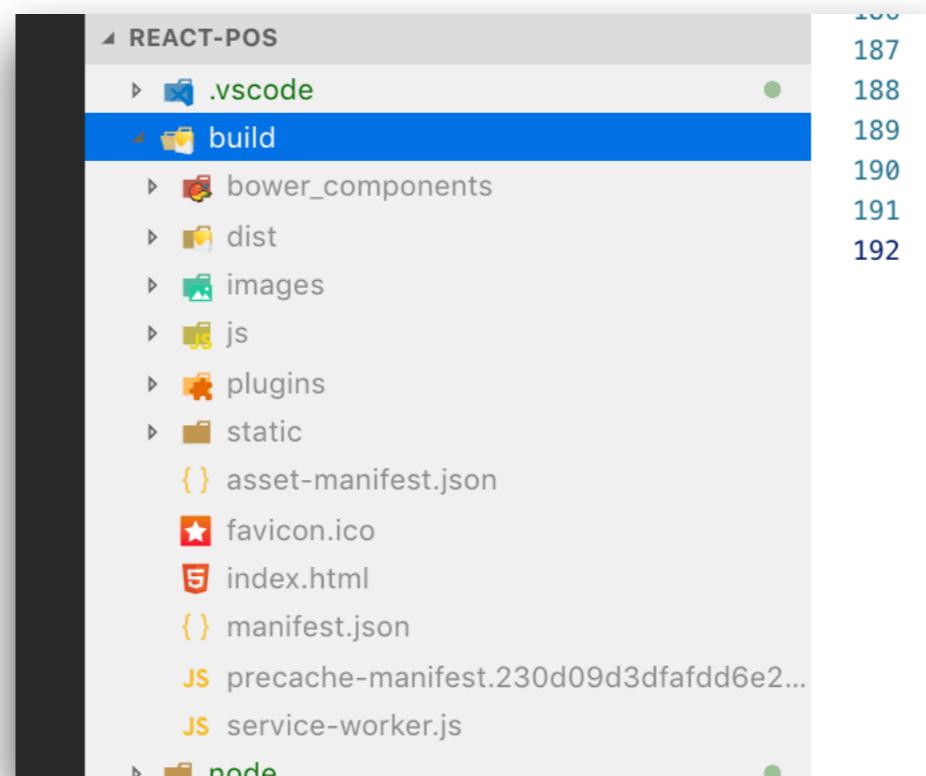
```
location /demo {  
    try_files $uri $uri/ /demo/index.html;  
}
```

1. Modify /usr/local/etc/nginx/nginx.conf
2. Write fallback-configuration to make all request to index.html. This makes angular to translate url request as routing command correctly

# Deployment

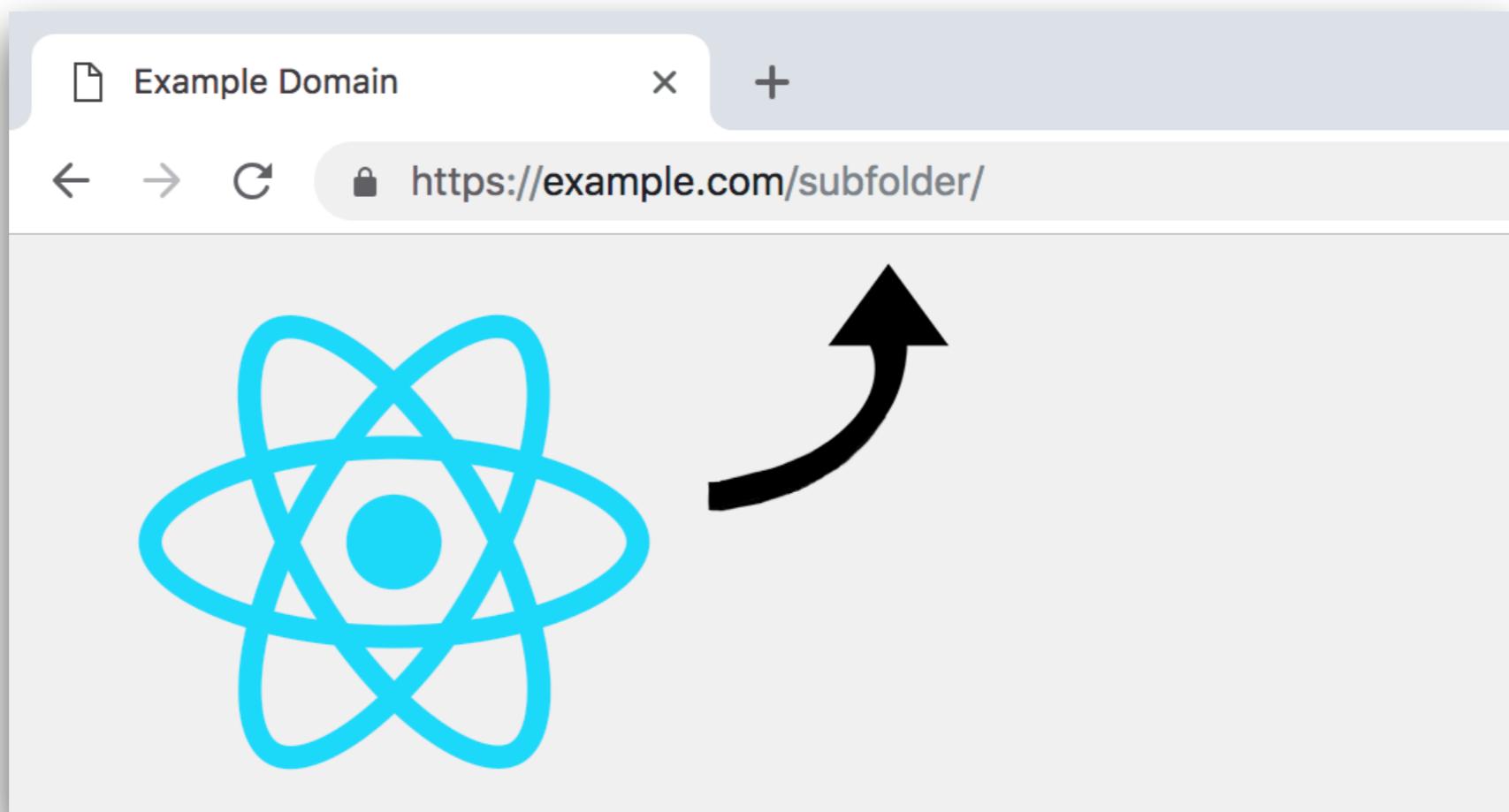
## npm run build

- npm run build



# Deploy into a Sub Folder

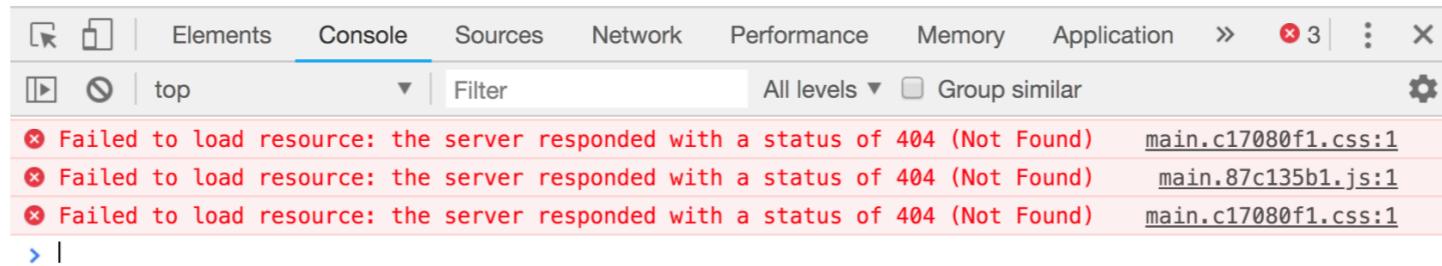
# Deployment into a sub-folder



CR. <https://skryvets.com/blog/2018/09/20/an-elegant-solution-of-deploying-react-app-into-a-subdirectory/>

# Deployment into a sub-folder

Let's start with an obvious statement: *React application deployed to a subfolder will not work out of the box:*



The reason for that is the path: React still tries to load its resources and assets from the root folder without accounting for a subdirectory.

In fact, here we're dealing with two types of issues:

- 1. Resources/assets loading.** Javascript, CSS, and assets such as images, fonts, etc. are broken.
- 2. Router navigation.** The default router configuration doesn't intend to work within the subdirectory.

# Deployment

## into a sub-folder

### 1. set baseURL

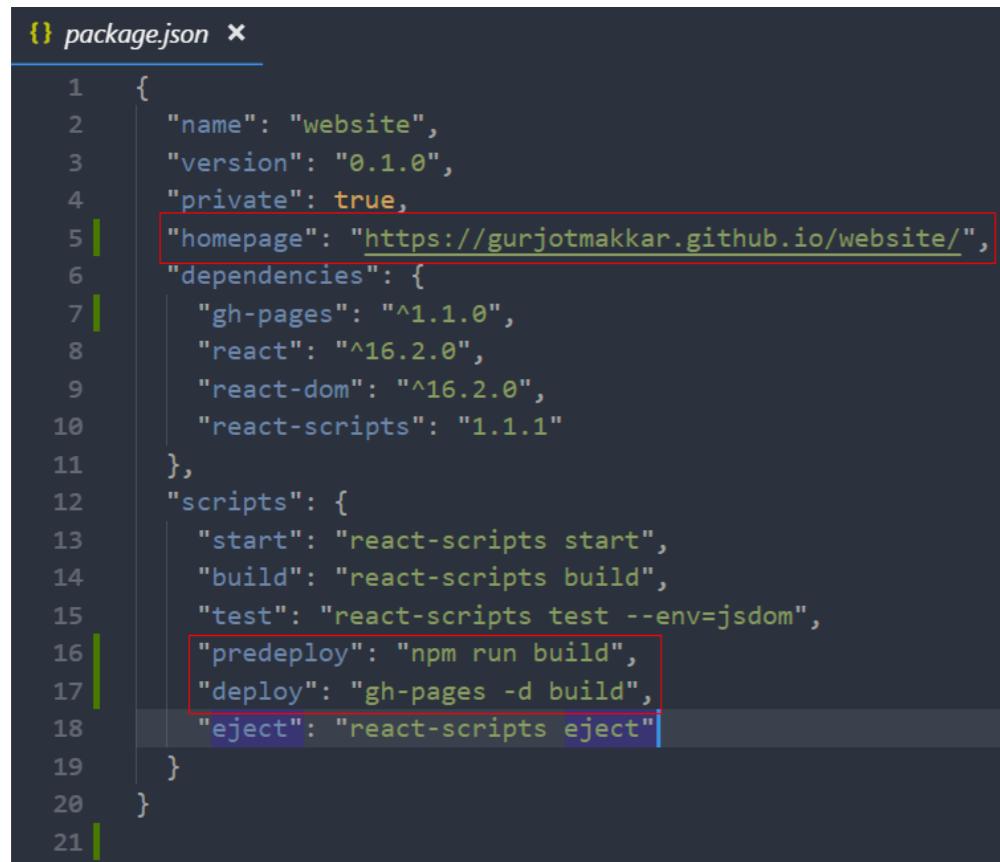
```
<Router basename={'/directory-name'}>
  <Route path='/' component={Home} />
  {/* ... */}
</Router>
```

Remark : Base on react-routerv4+

# Deployment into a sub-folder

## 2. Set app homepage

"homepage": "https://myapp.com/directory-name",



```
{ } package.json ×
1  {
2    "name": "website",
3    "version": "0.1.0",
4    "private": true,
5    "homepage": "https://gurjotmakkar.github.io/website/",
6    "dependencies": {
7      "gh-pages": "^1.1.0",
8      "react": "^16.2.0",
9      "react-dom": "^16.2.0",
10     "react-scripts": "1.1.1"
11   },
12   "scripts": {
13     "start": "react-scripts start",
14     "build": "react-scripts build",
15     "test": "react-scripts test --env=jsdom",
16     "predeploy": "npm run build",
17     "deploy": "gh-pages -d build",
18     "eject": "react-scripts eject"
19   }
20 }
21 |
```

# Deployment into a sub-folder

## 3. Update the routes

```
18
19
20    render() {
21      const LoginRedirect = () => <Redirect to='/login' />
22      return (
23        <Router basename={'/demo'}>
24          <div>
25            {isLoggedIn() ? <Header/> : null }
26            {isLoggedIn() ? <Menu/> : null }
27
28            <Route path="/login" component={Login} />
29            <PrivateRoute path="/stock" component={Stock} />
30            <PrivateRoute path="/stock-create" component={StockCreate} />
31            <PrivateRoute path="/stock-edit/:id" component={stockEdit} />
32            <PrivateRoute path="/report" component={Report} />
33            <Route path="/transaction" component={Transaction} />
34            <Route path="/shop" component={Shop} />
35            <Route exact path="/" component={LoginRedirect} />
36            {isLoggedIn() ? <Footer/> : null }
37          </div>
38        </Router>
39      );
40    }
41
42  
```

# Deployment into a sub-folder

## 4. Link to Image and other resources

```
>
  <img
    src={`${process.env.PUBLIC_URL}/images/react_js_logo.jpg`}
    style={{ height: 230 }}
    className="img-responsive"
    alt="profile"
  />
</li>
```

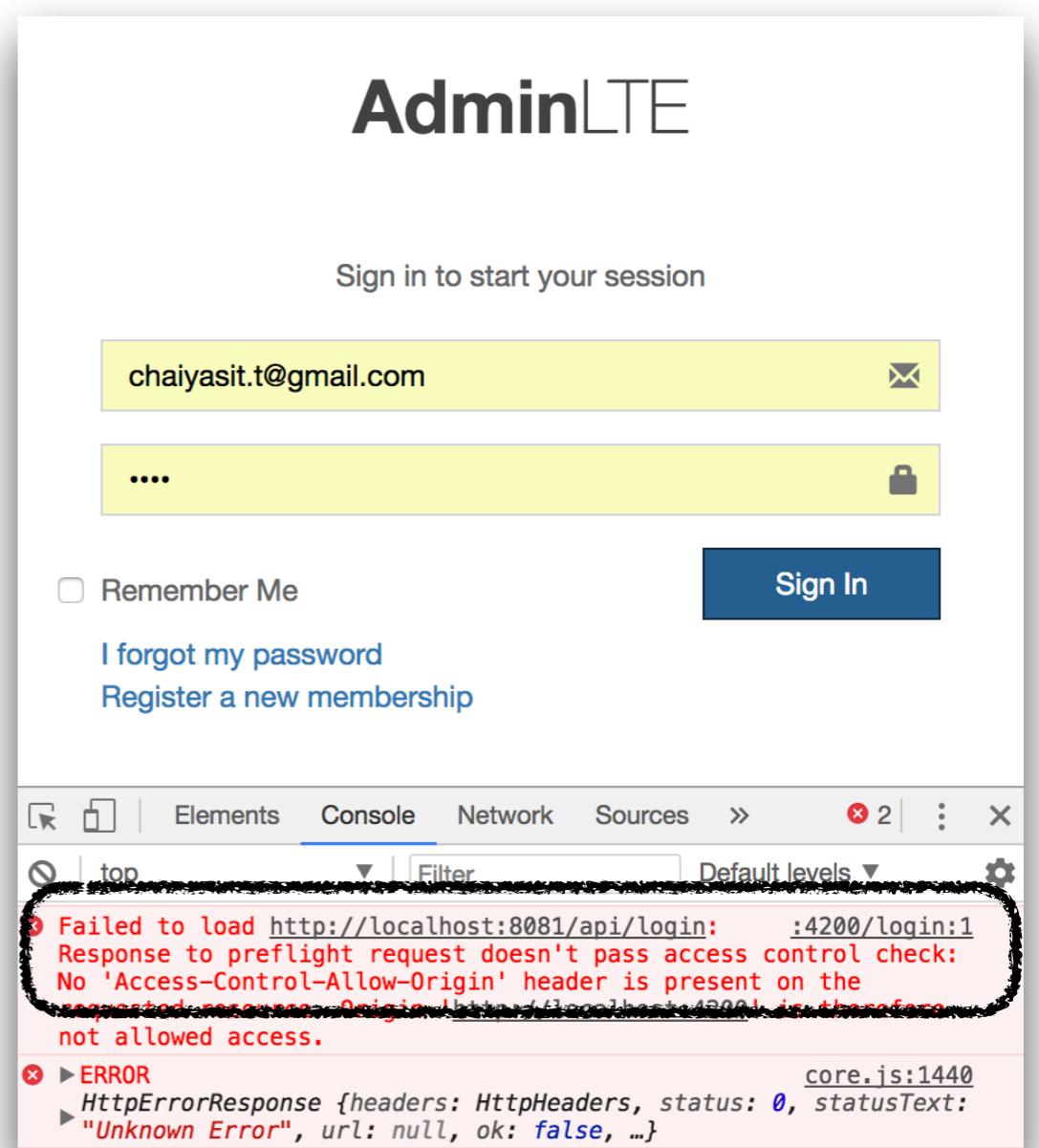
`src={`${process.env.PUBLIC_URL}/images/....`}`

# CORS

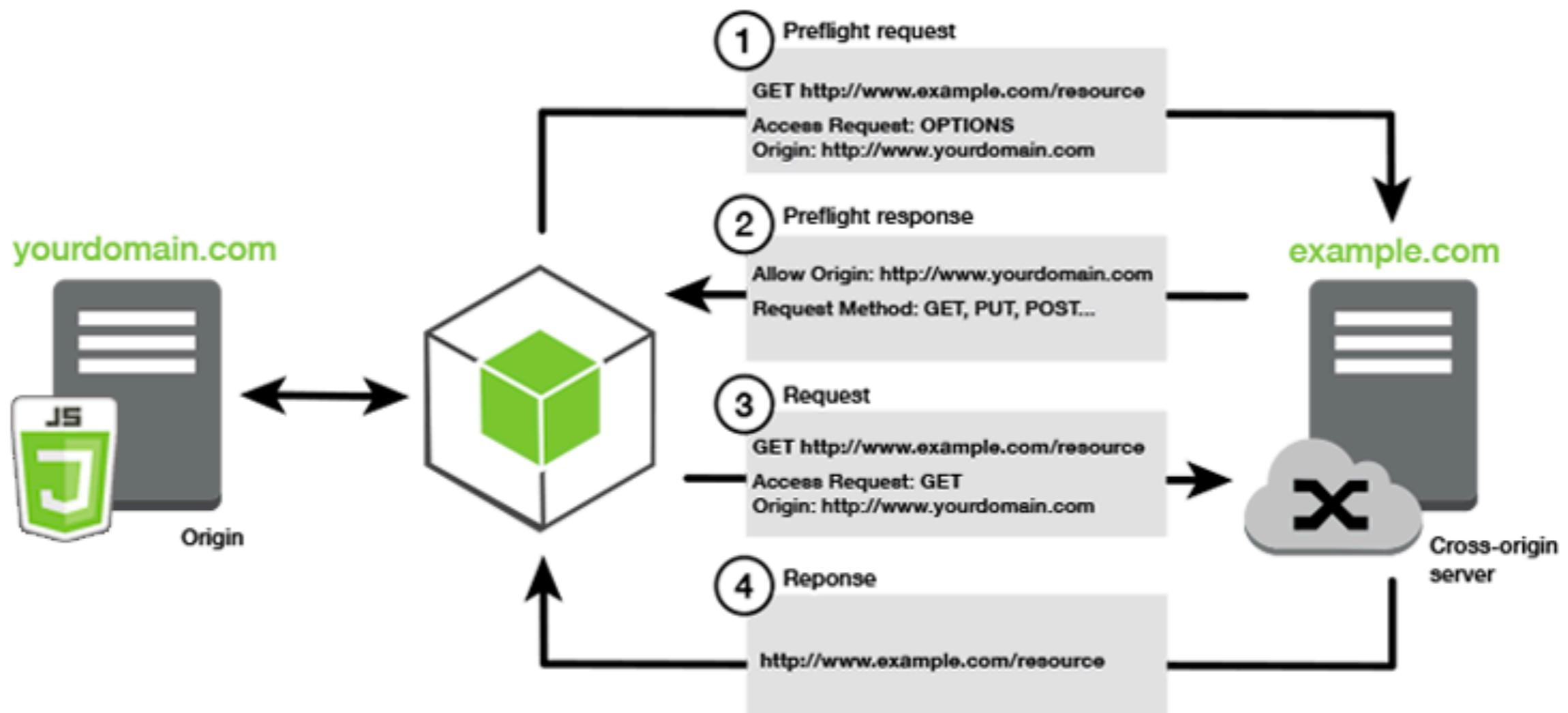
Cross-Origin Resource Sharing

# What is CORS?

- Solution to enable sharing resource from different origin domain.
- **Cross-origin resource sharing, or CORS** - a security feature of modern web browsers. It enables browser to negotiate which domain can make request of external website or service.



# What is CORS?



# Use npm cors

npm install cors

## Usage

### Simple Usage (Enable All CORS Requests)

```
var express = require('express')
var cors = require('cors')
var app = express()

app.use(cors())

app.get('/products/:id', function (req, res, next) {
  res.json({msg: 'This is CORS-enabled for all origins!'})
})

app.listen(80, function () {
  console.log('CORS-enabled web server listening on port 80')
})
```

# How to enable CORS?



**CORS on the server**

or



**CORS on the client**

# Enable CORS at Browser

## Using Plugin or Extension

### Browser support

#### Cross-Origin Resource Sharing

Method of performing XMLHttpRequests across domains

	Firefox		Chrome		Opera		Safari		iOS		Safari	
IE	54	60	55	61	45	46	9.1	10	47	9.3	10.0-10.3	10.3
8	13	56	57	62	46	47	10.1	11	48	10.3	11.0-11.1	11.1
9	14	58	58	63	47	48	10.1	11	49	10.3	11.0-11.1	11.1
10	15	59	59	64	48	49	10.1	TP	50	11.0-11.1	11.1	11.1
11	16	60	60	66	49	50	11.0-11.1	TP	51	11.1	11.1	11.1

Not running... **TURN ON**

SHOW DETAILED OPTIONS

Read the [Authoritative guide to CORS \(Cross-Origin Resource Sharing\)](#) for [REST APIs](#) first to understand how it works and common pitfalls.

For security, it is recommended to disable plugin when done with debug.

This tool is provided without warranty of any kind.

**moesif**  
Moesif is the leading API analytics platform for developers and more.

LEARN MORE

88.59% + 0.82% = 89.41%

Firefox	IE	UC	Samsung	Internet	QQ	Baidu
Mobile	Mobile	Browser	Mobile	Mobile	Browser	Browser
Android	10	for	Android	4	5	7.12
7	11	Android	11.4	6.2	1.2	

CODEMOBILES COMPANY LIMITED

# Enable CORS at Server Side

```
app.use(function (req, res, next) {
  //res.setHeader('Access-Control-Allow-Origin', '*');
  res.setHeader('Access-Control-Allow-Origin', 'http://localhost:4200');
  res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE');
  res.setHeader('Access-Control-Allow-Headers', 'Content-Type');
  res.setHeader('Access-Control-Allow-Credentials', true);
  next();
});
```

<https://github.com/expressjs/cors>

# Simple CORS Example

Here is a simple CORS example of when a browser requests a resource from another domain.

Let's say DomainX makes a request to DomainY for a particular resource. CORS uses HTTP headers to determine whether or not DomainX should have access to that resource. The browser automatically sends a request header to DomainY with

```
Origin: http://domainx.com
```

DomainY receives that request and will respond back with either:

- 1 **Access-Control-Allow-Origin: http://domainx.com**
- 2 **Access-Control-Allow-Origin: \*** (meaning all domains are allowed)
- 3 An error if the cross-origin requests are not allowed

# Simple CORS Example



**Only firefox requires to set Header, otherwise cors will not work.**

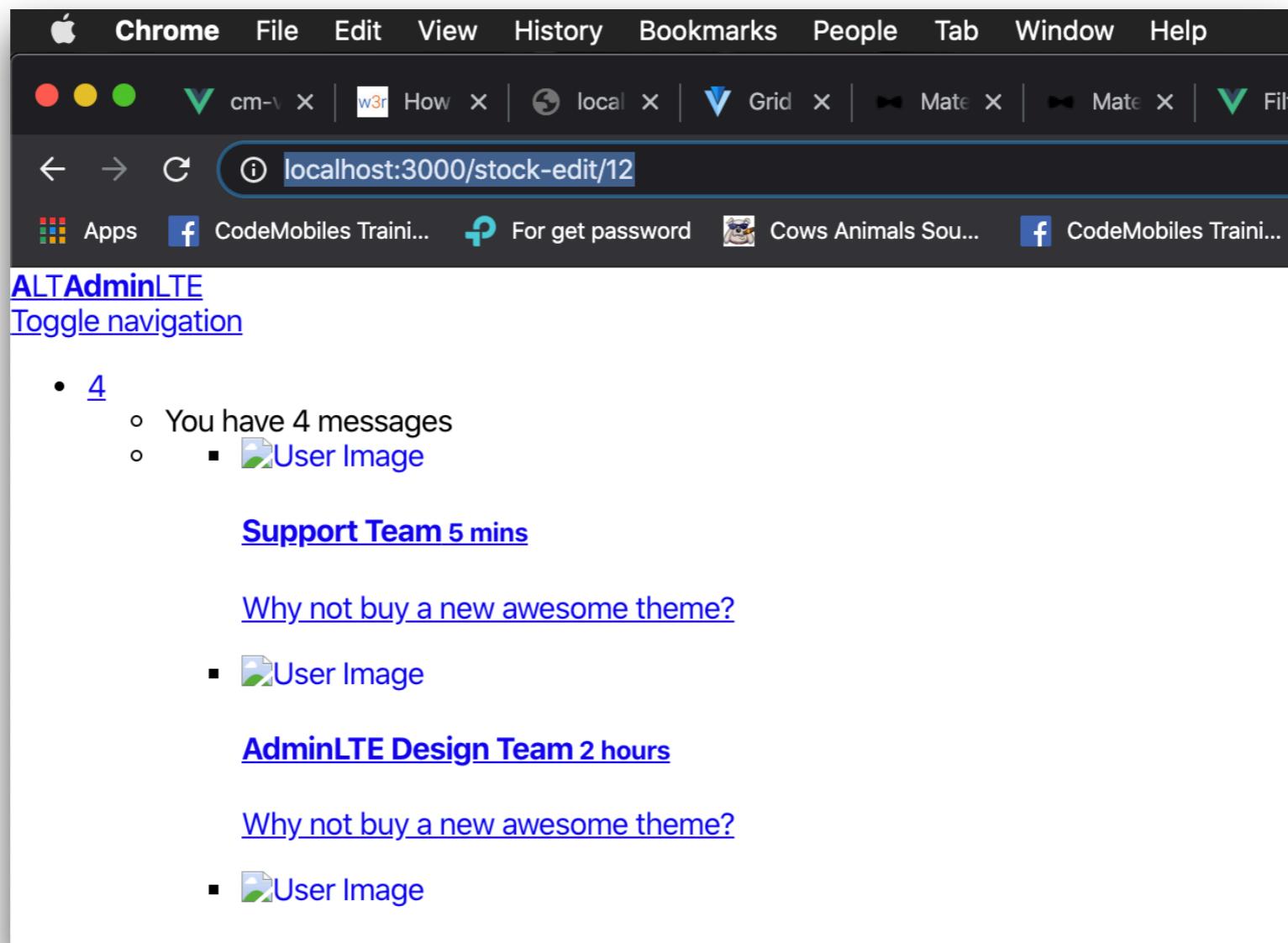
```
app.use(function (req, res, next) {
  res.setHeader('Access-Control-Allow-Origin', '*');
  //res.setHeader('Access-Control-Allow-Origin', 'http://localhost:4200');
  //res.setHeader('Access-Control-Allow-Origin', 'http://localhost:8080');
  res.setHeader('Access-Control-Allow-Methods', 'GET, POST, PUT, DELETE');
  res.setHeader('Access-Control-Allow-Headers', 'content-type, x-access-token');
  res.setHeader('Access-Control-Allow-Credentials', true);
  next();
});
```

```
<base href="/" />
```

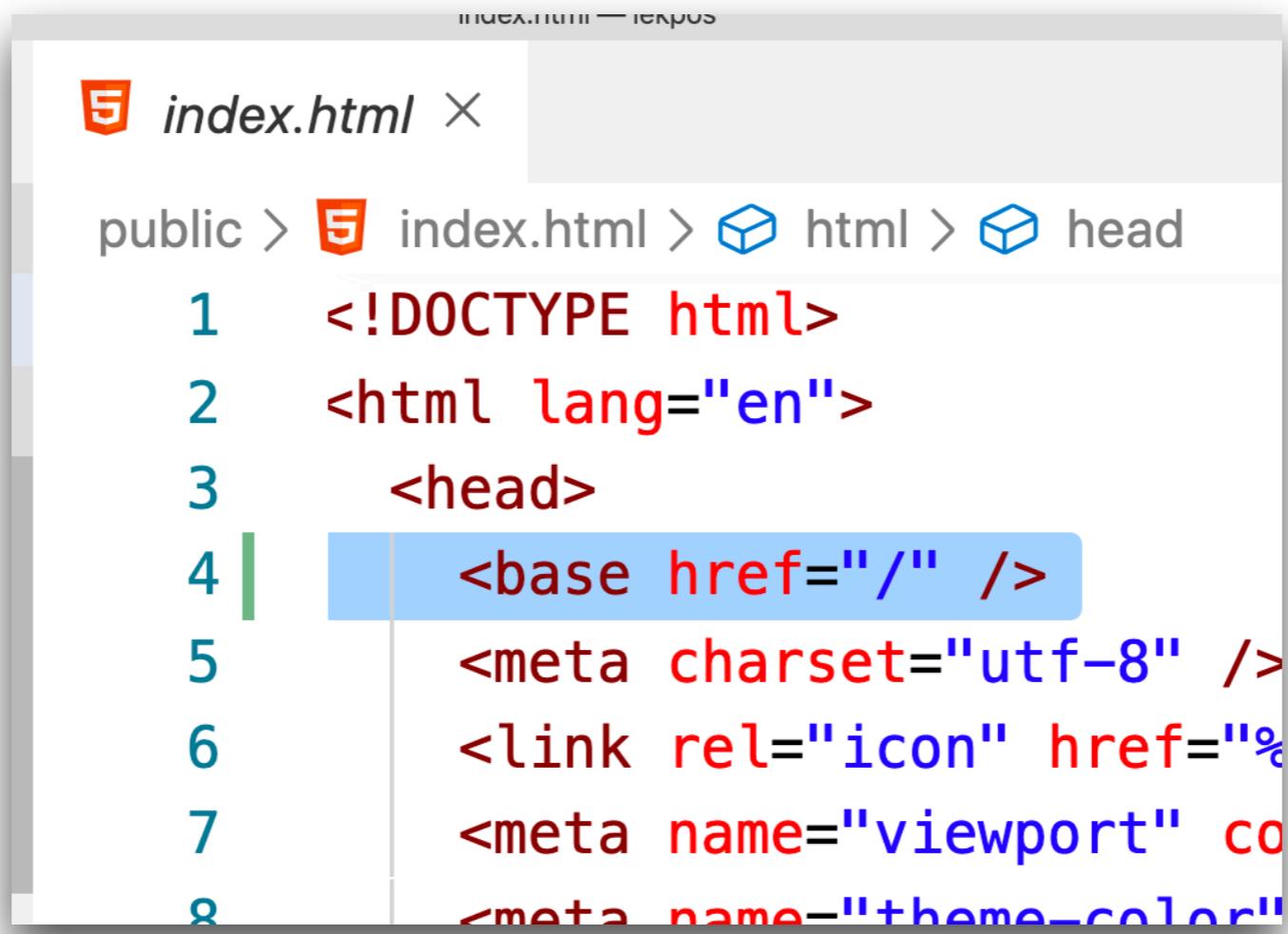
<http://nginx.org/>

# Router Error

# when page is refreshed



# Fixing by add <base href="/">



The screenshot shows a code editor window titled "index.html — terpus". The file structure is shown as "public > index.html > html > head". The code is numbered from 1 to 8. Line 4 contains the highlighted line: <base href="/" />. The code is as follows:

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <base href="/" />
5      <meta charset="utf-8" />
6      <link rel="icon" href="%PUBLIC_URL%/
7      <meta name="viewport" co
8      <meta name="theme-color" />
```

\* This will fix page routing error when refresh by telling base url to browser

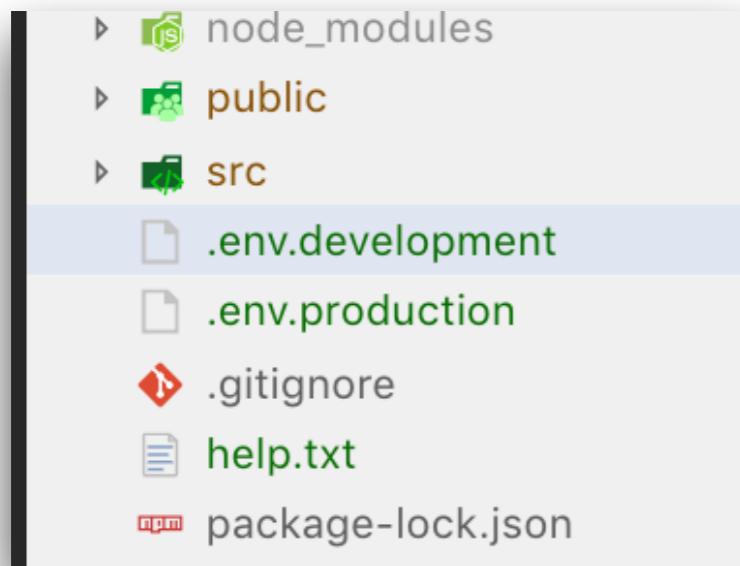
# process.env.

Environment Variable

<http://nginx.org/>

# .env files

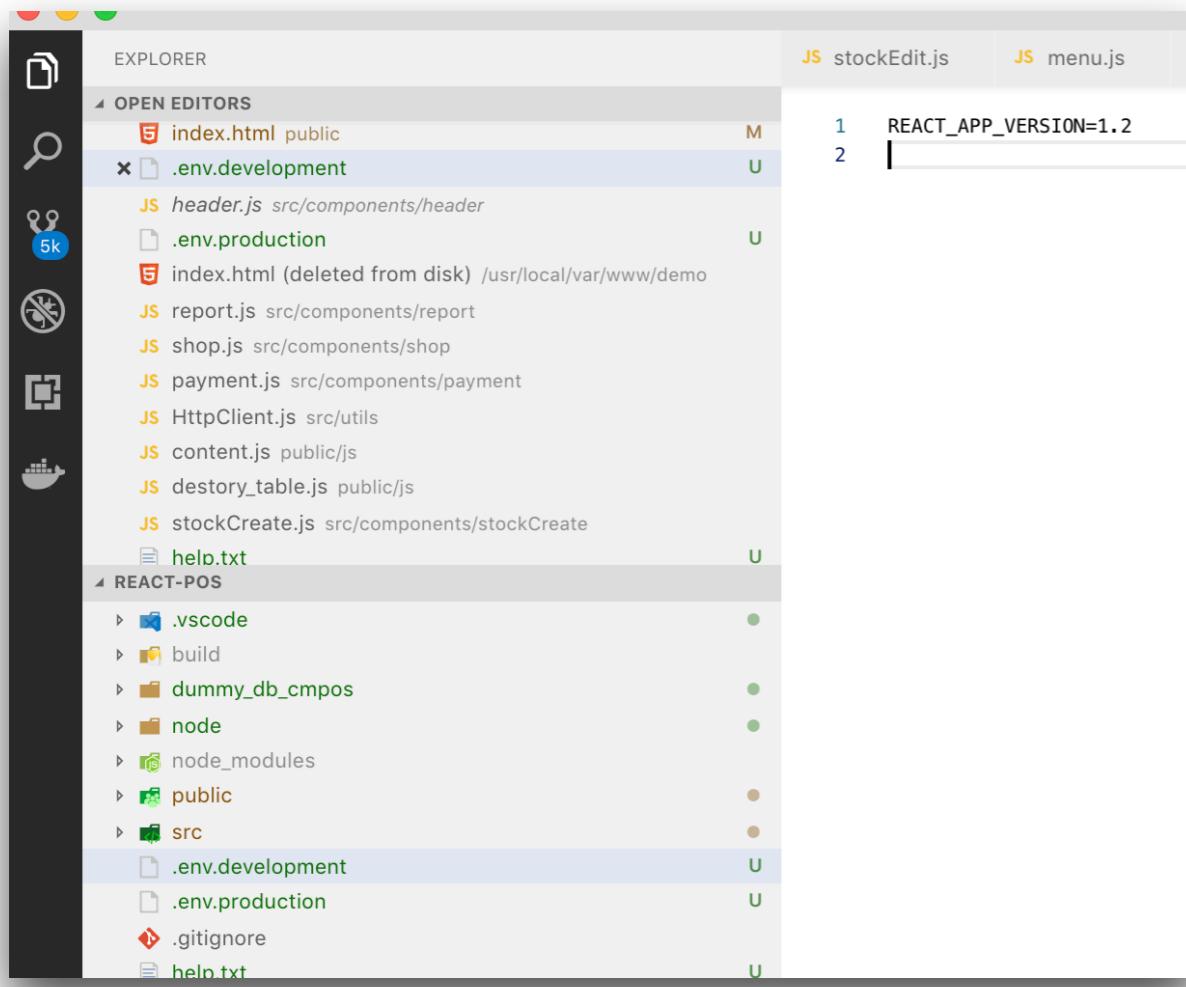
## Create



- **.env.development** : variable env. in development
- **.env.production** : variable env. in production

# .env files

## Read in JS



The screenshot shows the VS Code interface. On the left is the Explorer sidebar with a tree view of files and folders. The 'OPEN EDITORS' section shows 'index.html public' and '.env.development'. The 'REACT-POS' section shows '.vscode', 'build', 'dummy\_db\_cmpos', 'node', 'node\_modules', 'public', 'src', '.env.development', '.env.production', '.gitignore', and 'help.txt'. In the center, there are two code editors: 'stockEdit.js' and 'menu.js'. 'stockEdit.js' contains the line 'REACT\_APP\_VERSION=1.2'. 'menu.js' contains a snippet of JSX code that includes a call to `process.env.REACT_APP_VERSION`.

```
/* mini logo for sidebar mini 50x50 pixels */

  <b>C</b>POS

/* logo for regular state and mobile devices */

  <b>reactPOS {process.env.REACT_APP_VERSION}</b>

</span>
</a>
/* Header Navbar: style can be found in header.less */
<nav className="navbar navbar-static-top">
  /* Sidebar toggle button*/
```

# .env files

## Adding Temporary Environment Variables In Your Shell

Defining environment variables can vary between OSes. It's also important to know that this manner is temporary for the life of the shell session.

### Windows (cmd.exe)

```
set "REACT_APP_NOT_SECRET_CODE=abcdef" && npm start
```

(Note: Quotes around the variable assignment are required to avoid a trailing whitespace.)

### Windows (Powershell)

```
($env:REACT_APP_NOT_SECRET_CODE = "abcdef") -and (npm start)
```

### Linux, macOS (Bash)

```
REACT_APP_NOT_SECRET_CODE=abcdef npm start
```

## Adding Temporary Environment Variables In Your Shell

Defining environment variables can vary between OSes. It's also important to know that this manner is temporary for the life of the shell session.

### Windows (cmd.exe)

```
set "REACT_APP_NOT_SECRET_CODE=abcdef" && npm start
```

(Note: Quotes around the variable assignment are required to avoid a trailing whitespace.)

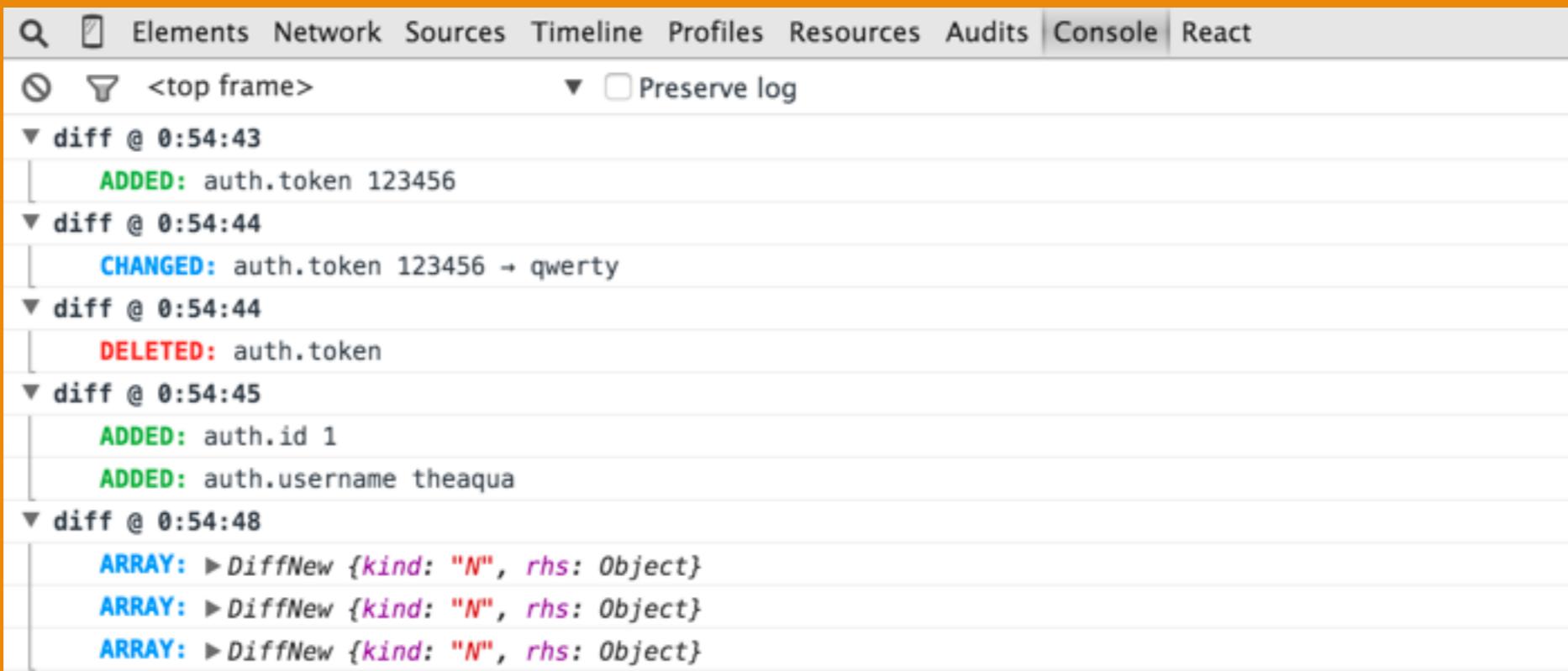
### Windows (Powershell)

```
($env:REACT_APP_NOT_SECRET_CODE = "abcdef") -and (npm start)
```

### Linux, macOS (Bash)

```
REACT_APP_NOT_SECRET_CODE=abcdef npm start
```

# Disable Redux-Logger in Production



The screenshot shows the Chrome DevTools interface with the "Console" tab selected. The log output displays a series of state changes over time (0:54:43 to 0:54:48), categorized by type: ADDED, CHANGED, and DELETED. The ADDED entries show the addition of 'auth.token' with values '123456' and '1'. The CHANGED entry shows 'auth.token' changing from '123456' to 'qwerty'. The DELETED entry shows 'auth.token' being removed. The final three entries are ARRAY: >DiffNew objects.

```
Elements Network Sources Timeline Profiles Resources Audits Console React
<top frame> ▾  Preserve log
▼ diff @ 0:54:43
  | ADDED: auth.token 123456
▼ diff @ 0:54:44
  | CHANGED: auth.token 123456 => qwerty
▼ diff @ 0:54:44
  | DELETED: auth.token
▼ diff @ 0:54:45
  | ADDED: auth.id 1
  | ADDED: auth.username theaqua
▼ diff @ 0:54:48
  | ARRAY: >DiffNew {kind: "N", rhs: Object}
  | ARRAY: >DiffNew {kind: "N", rhs: Object}
  | ARRAY: >DiffNew {kind: "N", rhs: Object}
```

<http://nginx.org/>

## ② Log only in development

```
const middlewares = [];

if (process.env.NODE_ENV === `development`) {
  const { logger } = require(`redux-logger`);

  middlewares.push(logger);
}

const store = compose(applyMiddleware(...middlewares))(createStore)(reducer);
```

```
import logger from 'redux-logger'

const dev_middlewares = [];
if (process.env.NODE_ENV === `development`) {
  dev_middlewares.push(logger);
}

//const store = createStore(reducers, applyMiddleware(thunk,logger)) // Simple way
const store = createStore(reducers, applyMiddleware(thunk,...dev_middlewares)) // Advance way
```

```
ReactDOM.render(
  <Provider store={store}>
    <App />
  </Provider>,
  document.getElementById("root")
);
```

# Webserver

**NGINX**

<http://nginx.org/>

# Install Nginx on macOS

1. Install Homebrew : `/usr/bin/ruby -e "$(curl -fSSL https://raw.githubusercontent.com/Homebrew/install/master/install")"`
2. `brew install nginx`
3. `sudo nginx`
4. <http://localhost:8080>
5. `/usr/local/etc/nginx/nginx.conf`
6. `sudo nginx -s stop`
7. `vim /usr/local/etc/nginx/nginx.conf`
8. <https://www.youtube.com/watch?v=2QppqlkIBLE>

# Config Nginx on macOS

- vim /usr/local/etc/nginx/nginx.conf
- sudo nginx

```
server {  
    listen      8080;  
    server_name localhost;  
  
    #access_log  logs/host.access.log  main;  
  
    location / {  
        root   html;  
        index  index.html index.htm;  
    }  
}
```

to:

```
server {  
    listen      80;  
    server_name localhost;  
  
    #access_log  logs/host.access.log  main;  
  
    location / {  
        root   html;  
        index  index.html index.htm;  
    }  
}
```

# Deployment Folder on macOS

- cd /usr/local/Cellar/nginx/1.13.9/html/
- note: change **\*\*1.13.9\*\*** to your nginx version

```
server {
    server_name  localhost;

    #access_log  logs/host.access.log  main;

    location / {
        root   html;
        index  index.html index.htm;
    }
}
```

To let say Users/to/www:

```
server {
    listen      80;
    server_name localhost;

    #access_log  logs/host.access.log  main;

    location / {
        root   /Users/to/www;
        index  index.html index.htm;
    }
}
```

# Install Nginx on Windows

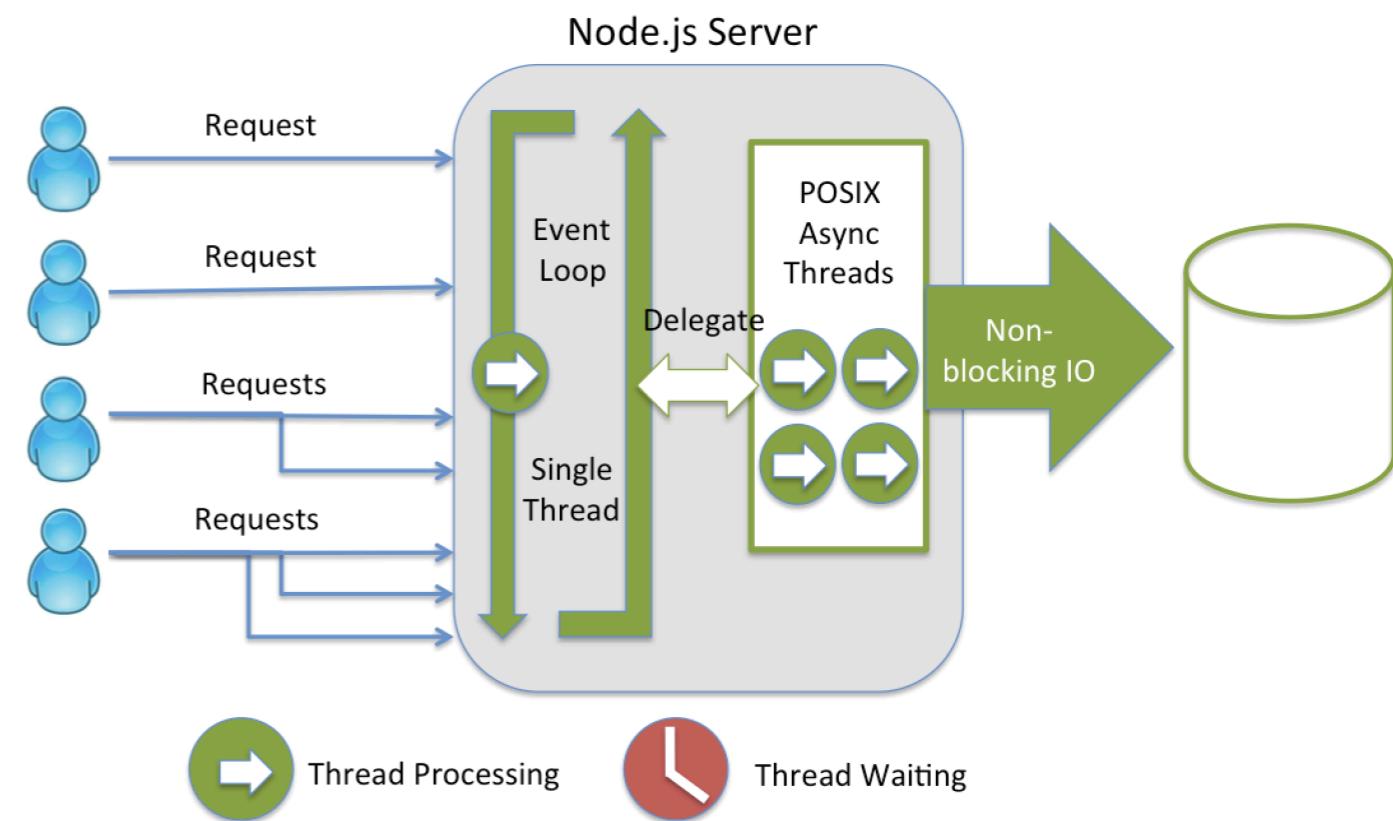
- <https://www.youtube.com/watch?v=Kv67mJzEsS8>



CodeMobiles

# What and Why Node.js?

- **Node.js** is the Javascript runtime for high-performance, low-latency application.
- It uses an event-driven, non-blocking I/O model that makes it lightweight, efficient and highly-performant event under extreme load.



# What can we do with Node.JS?

- REST API and Backend Application
- Real-Time services (Chat, Game etc.)
- Blog, CMS, Social Applications
- Anything that is not CPU intensive

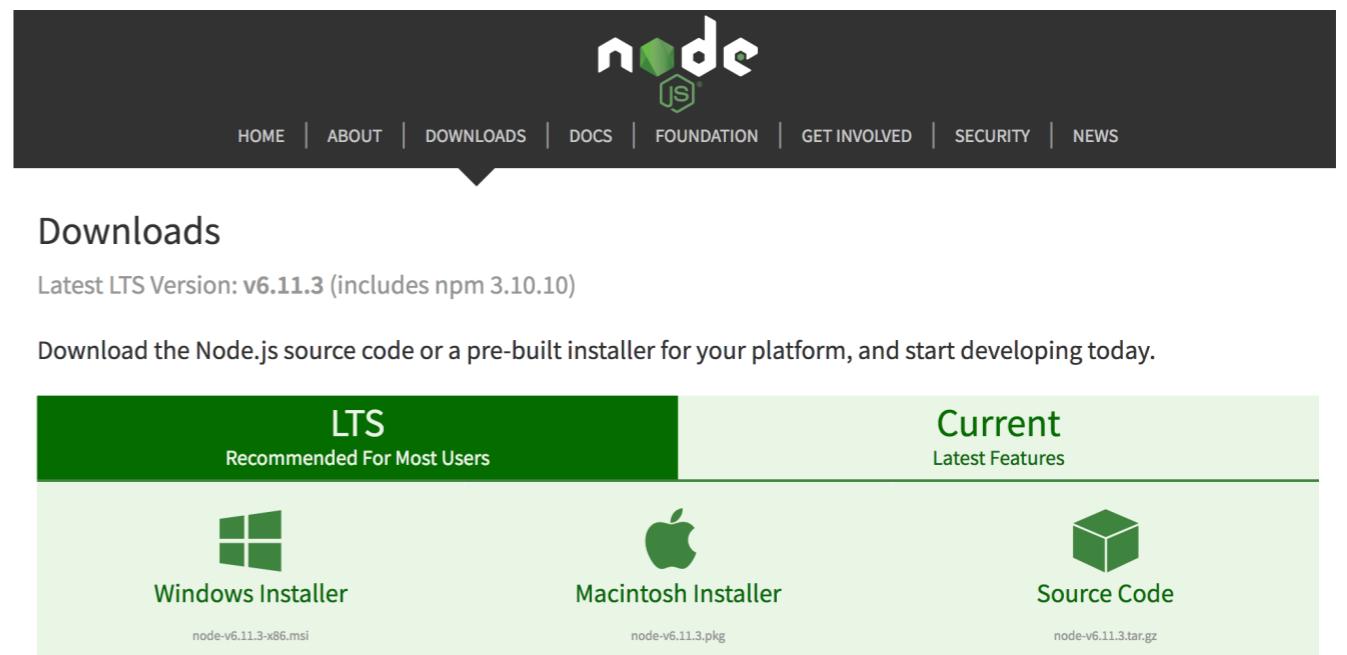


# Node.js **NOT** Good when

- The request need heavy CPU consumption (CPU Intensive Bound) such as solving her algorithm
- Because this will block Event-Loop and Thread Pool

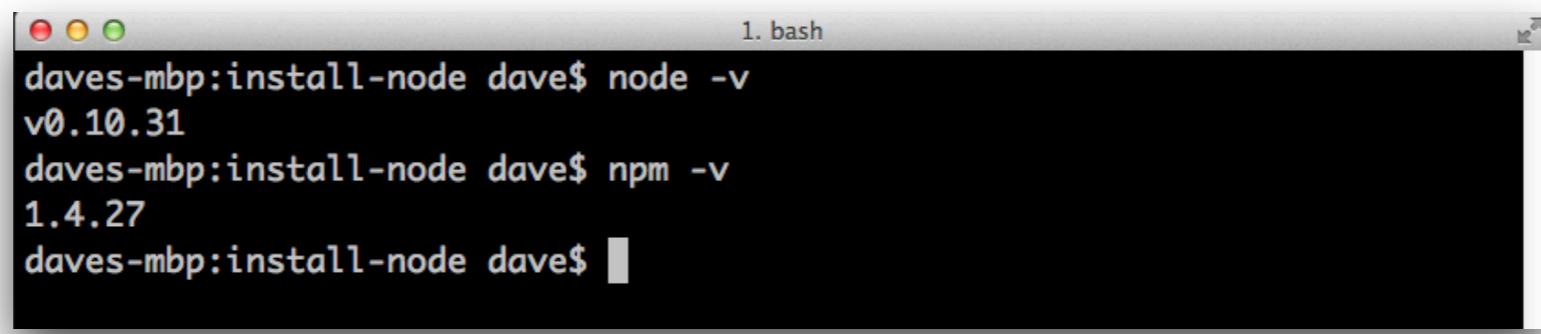
# Install Node.JS

- Go to [www.nodejs.org/en/download/](http://www.nodejs.org/en/download/)
- Download and Install the right platform-version (win, macOS, linux)
- Verify Installation
  - node -v
  - npm -v



# NodeJS

## Check and Update Version



```
daves-mbp:install-node dave$ node -v
v0.10.31
daves-mbp:install-node dave$ npm -v
1.4.27
daves-mbp:install-node dave$
```

▲ **Linux/Mac:**

570 The module `n` makes version-management easy:

▼  
✓

```
sudo npm install n -g
sudo n 0.12.2
```

For the latest stable version:

```
sudo n stable
```

For the latest version:

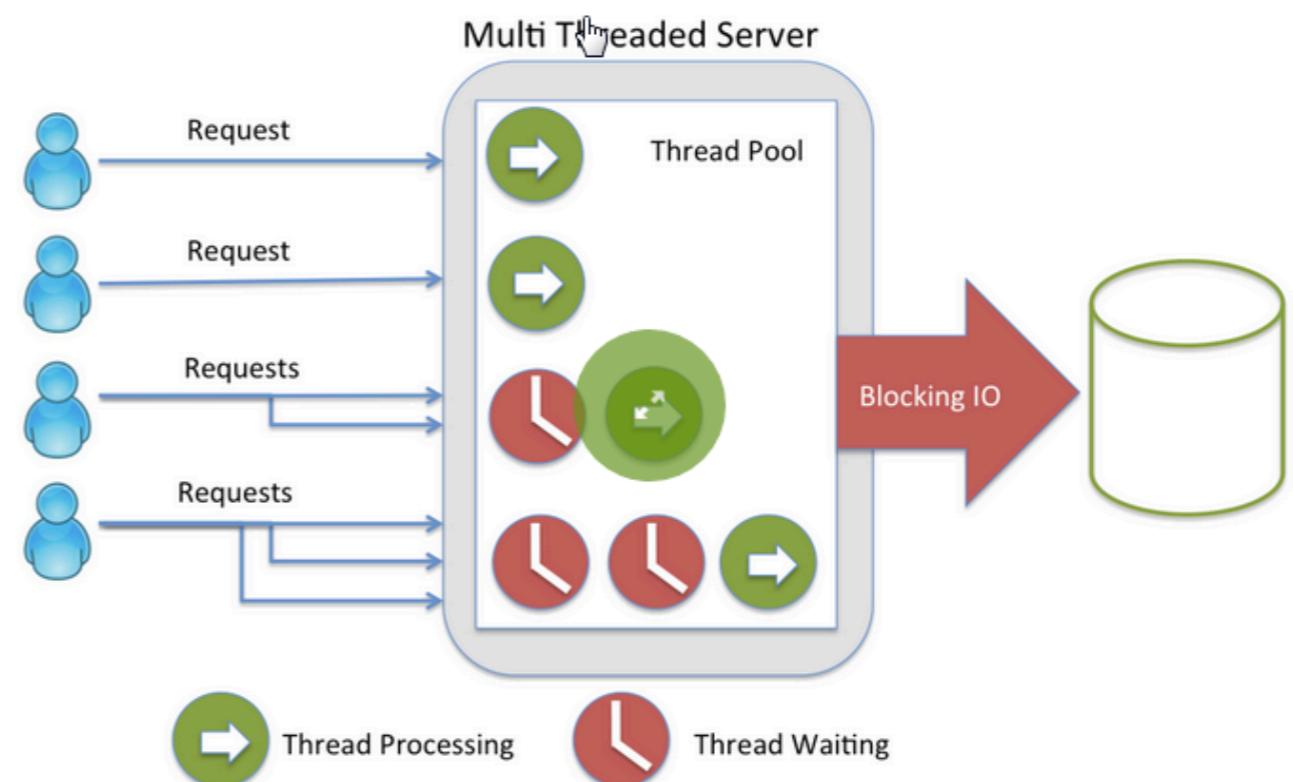
```
sudo n latest
```

**Windows:**

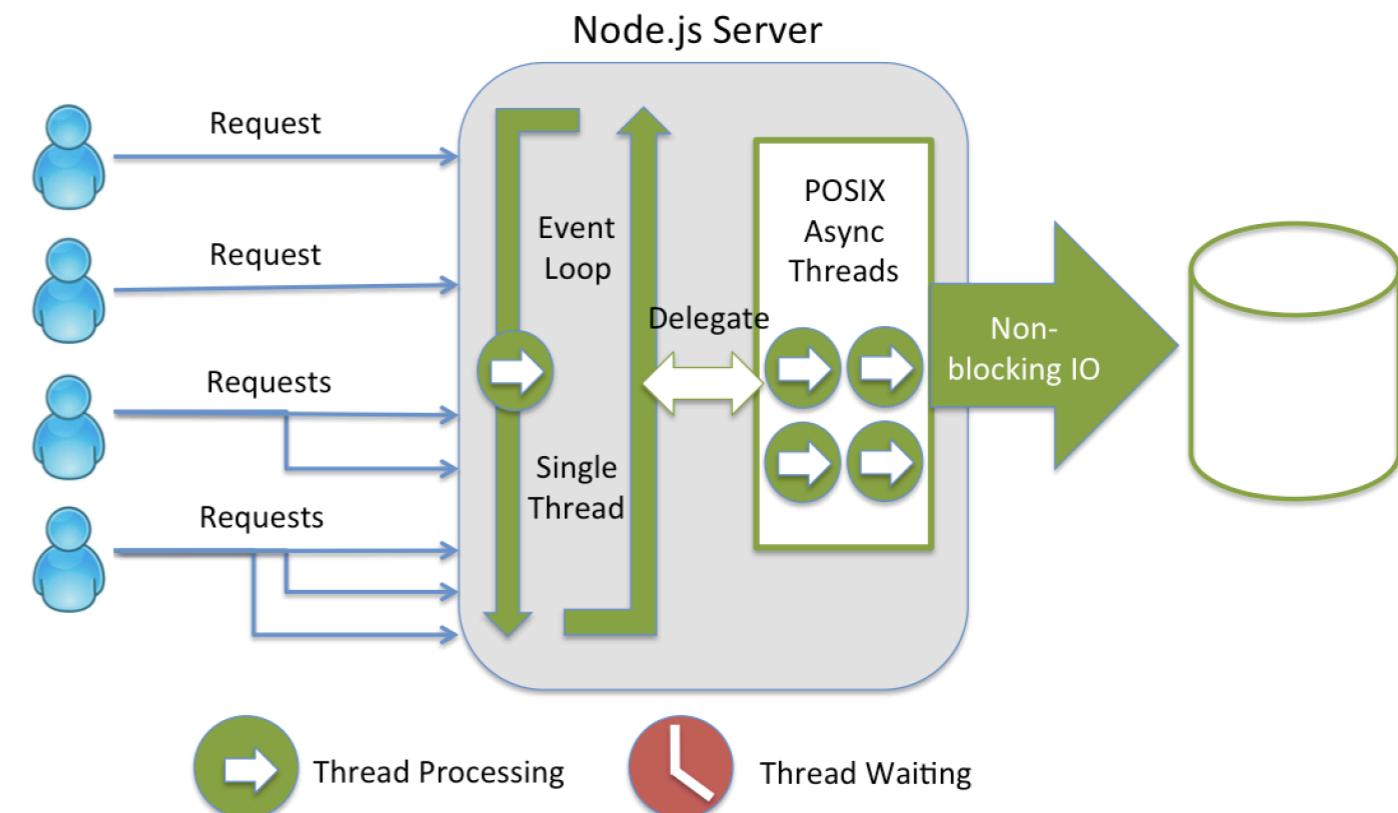
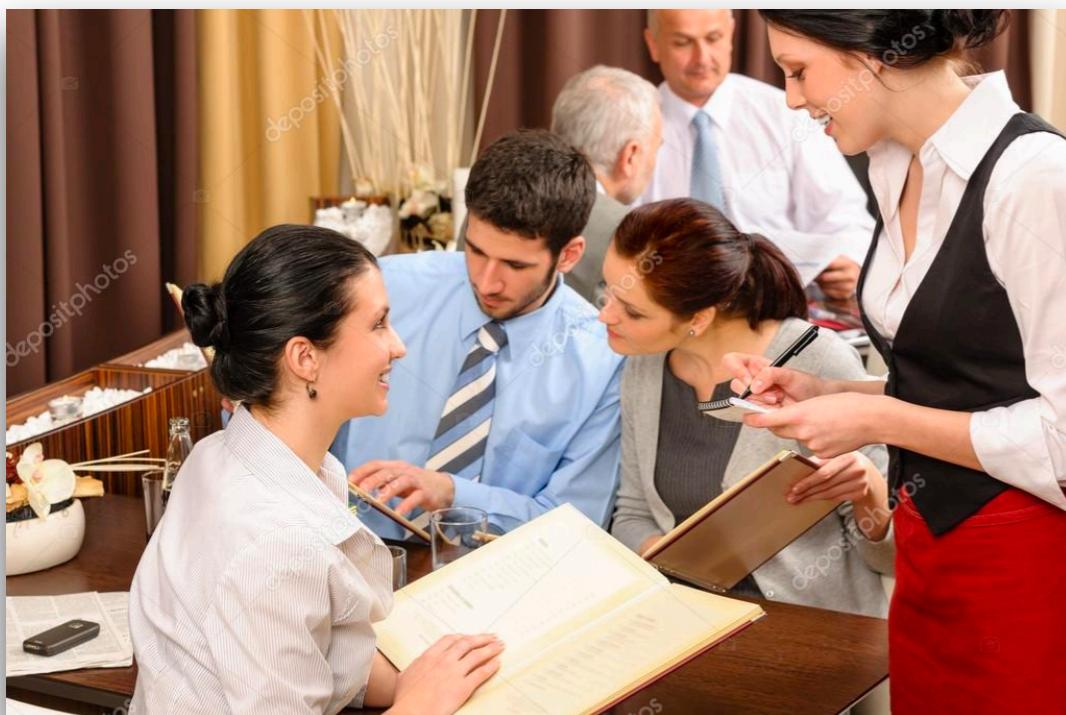
just reinstall node from the .msi in Windows from the node website.

# Blocking I/O (Problem)

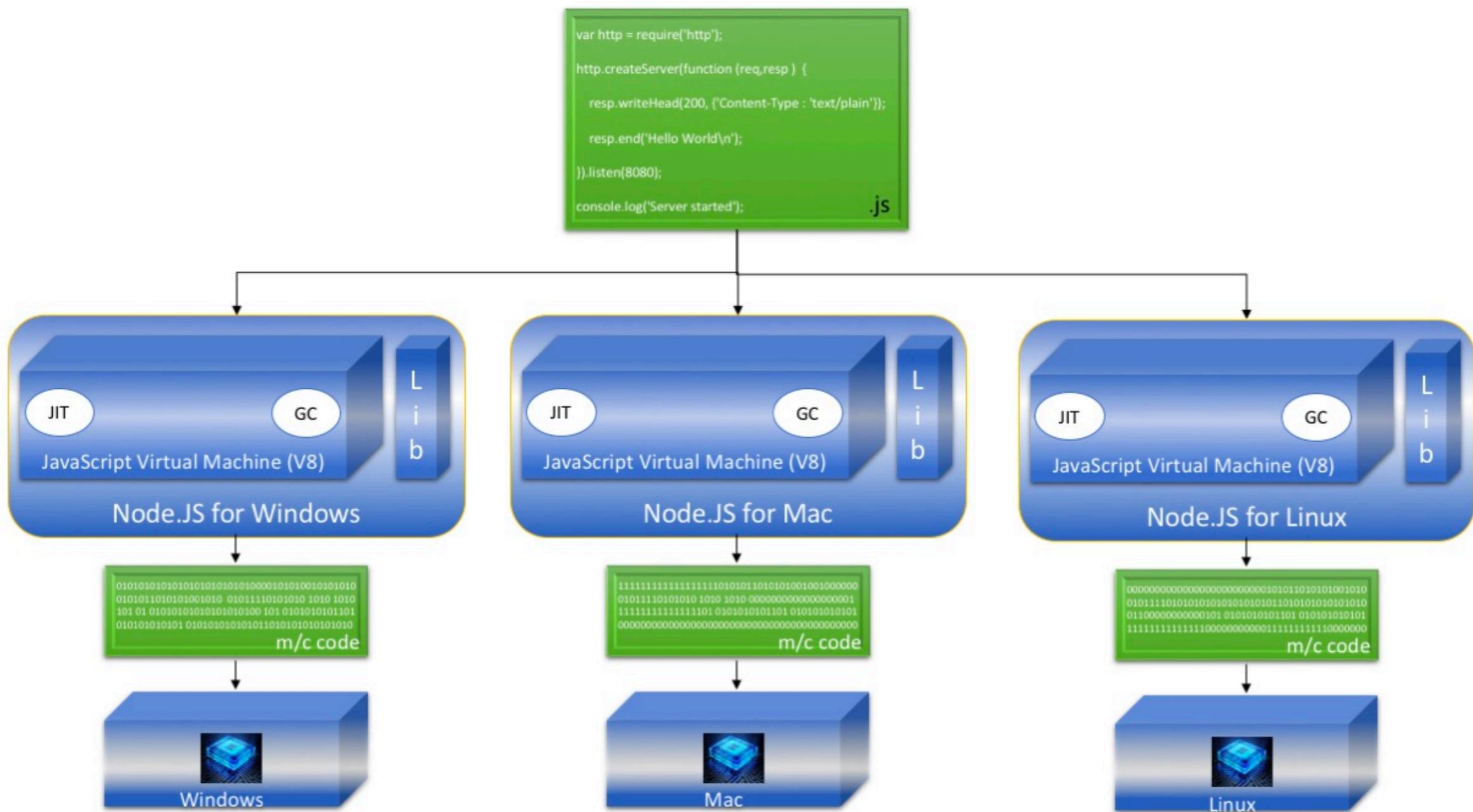
## *Blocking I/O Model*



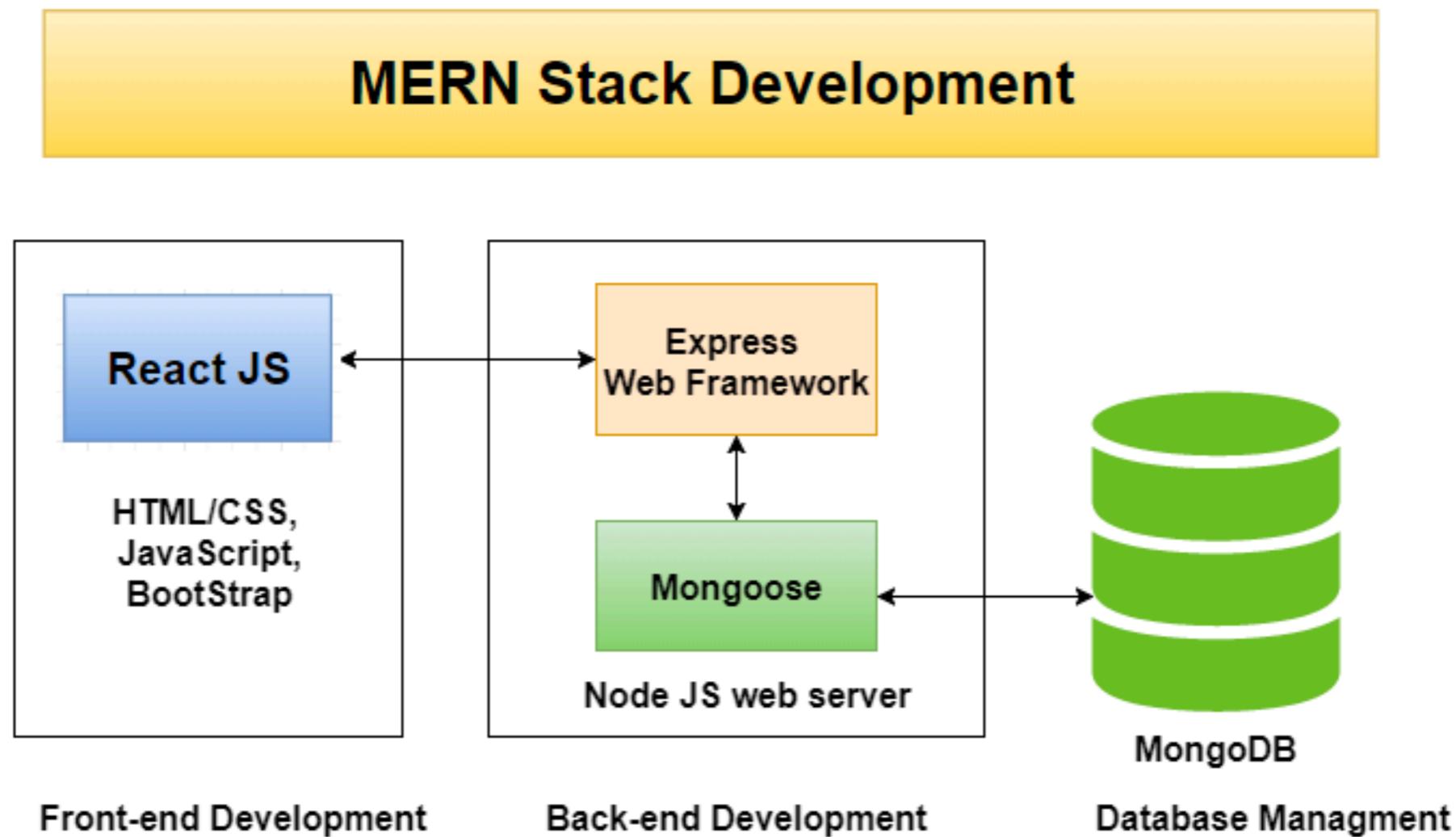
# Non-Blocking I/O (NodeJS)



# Node.js Architecture



# Mern Architecture



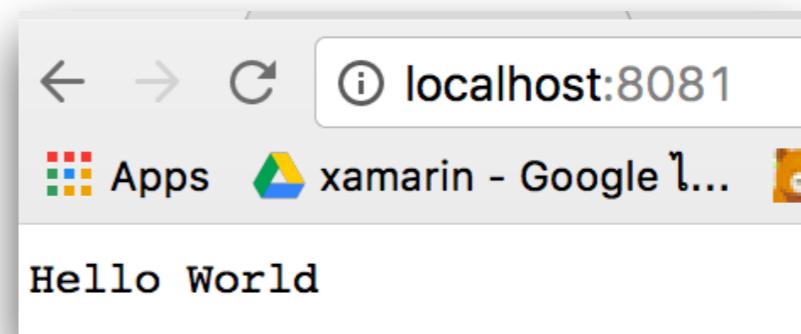
# Node.js Keywords

- Module
- Express
- Router
- Nodemon
- PM2
- body-parser
- Web service



# Hello Node.js

```
var http = require('http');
http.createServer(function (req, res) {
    res.writeHead(200, {'Content-Type': 'text/plain'});
    res.end('Hello World');
}).listen(8081);
console.log("Server running at http://localhost:8081")
```



# npm

# Node Package Manager

- NPM is a package manager for Node.js packages, or modules if you like.
- [www.npmjs.com](http://www.npmjs.com) hosts thousands of free packages to download and use.
- The NPM is installed by default when you install Node.js



Download "upper-case":

```
C:\Users\Your Name>npm install upper-case
```

```
| var uc = require('upper-case');
```

# NPM

# Package.json #1

- **package.json** is present in the root directory of any Node application/module and is used to define the properties of a package.

- Ex:

- npm install express
- npm uninstall express
- npm install express -g

## Attributes of Package.json

- **name** – name of the package
- **version** – version of the package
- **description** – description of the package
- **homepage** – homepage of the package
- **author** – author of the package
- **contributors** – name of the contributors to the package
- **dependencies** – list of dependencies. NPM automatically installs all the dependencies mentioned here in the node\_module folder of the package.
- **repository** – repository type and URL of the package
- **main** – entry point of the package
- **keywords** – keywords

Ref: [https://www.tutorialspoint.com/nodejs/nodejs\\_npm.htm](https://www.tutorialspoint.com/nodejs/nodejs_npm.htm)

# NPM

# Package.json #2

- **npm init** : create package.json
- npm —version
- sudo npm install npm -g
- var express = require('express')
- npm install express
- npm uninstall express
- npm install express -g
- npm search <package>

**Ref:** [https://www.tutorialspoint.com/nodejs/nodejs\\_npm.htm](https://www.tutorialspoint.com/nodejs/nodejs_npm.htm)

# Node.js Module

- **Module in Node.js** is a simple or complex functionality organized in single or multiple Javascript files, which can be reused throughout the Node.js.
- Each **module** in Node.js has its own context, so it cannot interface with other modules or pollute global scope.

**Ref:** [https://www.tutorialspoint.com/nodejs/nodejs\\_npm.htm](https://www.tutorialspoint.com/nodejs/nodejs_npm.htm)

# Module

# Private Code

## my\_module.js

```
module.exports = ()=>{
  var current = null;
  function init(){
    // public
  }
  function change(){
    // public
    verify();
  }
  function verify(){
    // private
  }

  return {
    init: init,
    change: change
  }
}
```

## client.js

```
var module2 = require('./my_module');
module2().init(); // public
module2().change(); // public
//module2().verify(); // private
```

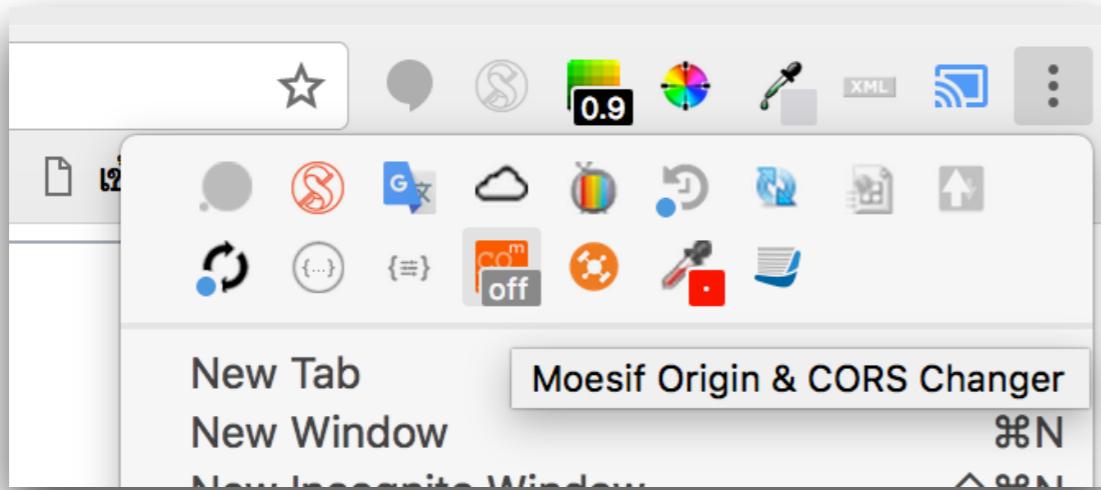
# Allow All IP (Node JS)



pradhyu commented on Mar 24

using following in the package.json makes it accessible to the host machine  
ng serve --host 0.0.0.0

ng serve –host 192.168.200.65



# Global Objects

- Process : provides information about and control over and listen event happening to nodes process such as 'exit'
- Console : Logging to stdout and stderr
- Require : load external library
- \_\_dirname : string of current directory
- Module.exports : defining what a module exports and make available through require()
- setTimeout(callbackFn, mills) : Run callback function after reaching to 'mills'
- clearTimeout : Stop a timer previously created by setTimeout()

# NodeJS

## General Template

```
const express = require('express');
const path = require('path');
const autoIncrement = require("mongodb-autoincrement");
const formidable = require('formidable');
const router = express.Router();
const mongoClient = require("mongodb").MongoClient;
const mongo_string = "mongodb://localhost:27017/cmpos";
const fs = require('fs');
const session = require('express-session');
router.use(session({ secret: 'cmpos', cookie: { maxAge: 60000000 },
resave: true, saveUninitialized: false}));

// GET
router.get('/product', function (req, res) {...});

router.get('/product/:id', function (req, res) {...});

module.exports = router;
```

Used by

```
const express = require('express');
const path = require('path');
const bodyParser = require('body-parser');
const session = require('express-session');
const app = express();

app.use(express.static(path.join(__dirname, 'dist')));
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: false}));
app.use('/api', require('./server/api.js'));

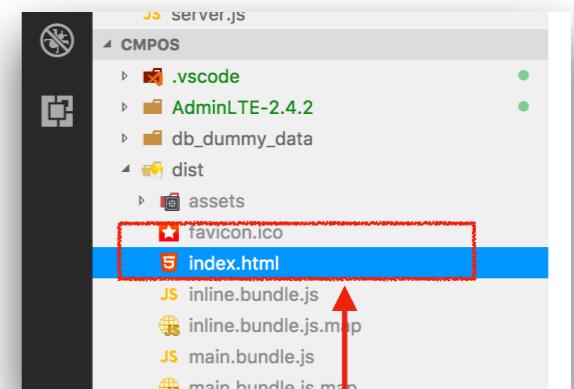
app.get('*', function(req, res){
    res.sendFile(path.join(__dirname, 'dist/index.html'), {
        new:{":n":1}
    });
    console.log();
});

const server = app.listen(8081, function(){
    var host = server.address().address;
    var port = server.address().port;

    console.log('Running ... http://localhost:%s', host, port);
})
```

express-routing

Executable JS



# HTTP Module

```
var http = require('http');

//create a server object:
http.createServer(function (req, res) {
  res.write('Hello World!'); //write a response to the client
  res.end(); //end the response
}).listen(8080); //the server object listens on port 8080
```

- Http is built-in module
- Allow Node.js to transfer data between client and server with HTTP protocol
- Support Managing: Header, Query String, Split the Query String

# FS (File System) Module

```
var http = require('http');
var fs = require('fs');

http.createServer(function (req, res) {
  fs.readFile('demofile1.html', function(err, data) {
    res.writeHead(200, {'Content-Type': 'text/html'});
    res.write(data);
    res.end();
  });
}).listen(8080);
```

- Allow you to work the file system such as loading html file or any raw json file for response
- Support Managing: Read, Write, Append, Delete

# Event Module

## Event Emitter

- Event Module allows you to create, fire and listen to your own event

- Core Functions

- `.On('ev', cb)`
- `.Emit('ev')`

```
var events = require('events');
var eventEmitter = new events.EventEmitter();

//Create an event handler:
var myEventHandler = function () {
  console.log('I hear a scream!');
}

//Assign the event handler to an event:
eventEmitter.on('scream', myEventHandler);

//Fire the 'scream' event:
eventEmitter.emit('scream');
```

**Remark :** `.on()` is exactly the same as `.addListener()` in the `EventEmitter` object.

# Event Emitter

## on()

```
const EventEmitter = require('events');
class MyEmitter extends EventEmitter {}
const myEmitter = new MyEmitter();

myEmitter.on('event', function(a, b) {
  console.log(a, b, this);
});

myEmitter.emit('event', 'Technoetics', 'Club');
```

# Event Emitter

## Asynchronously

```
const EventEmitter = require('events');
class MyEmitter extends EventEmitter {}
const myEmitter = new MyEmitter();
myEmitter.on('event', function(a, b) {
  setImmediate(() => {
    console.log('this happens asynchronously');
  });
  function cb(){
    console.log('processed in next iteration',a,b);
  }
  process.nextTick(cb)
  console.log('processed in first iteration',a,b);
});

myEmitter.emit('event', 'Technoetics', 'Club');
```

# Event Emitter

# Error Handling

```
const myEmitter = new MyEmitter();
myEmitter.on('error', (err) => {
  console.error('whoops! there was an error');
});

myEmitter.emit('error', new Error('whoops!'));
// Prints: whoops! there was an error
```

# Error Handling using Domain

```
const d = require('domain').create();
d.on('error', (er) => {
  console.log(`error, but oh well ${er.message}`);
});

d.run(() => {
  require('http').createServer((req, res) => {
    handleRequest(req, res);
  }).listen(PORT);
});
```

# UploadFile Module

```
var express = require('express')
var multer = require('multer')
var upload = multer({ dest: 'uploads/' })

var app = express()

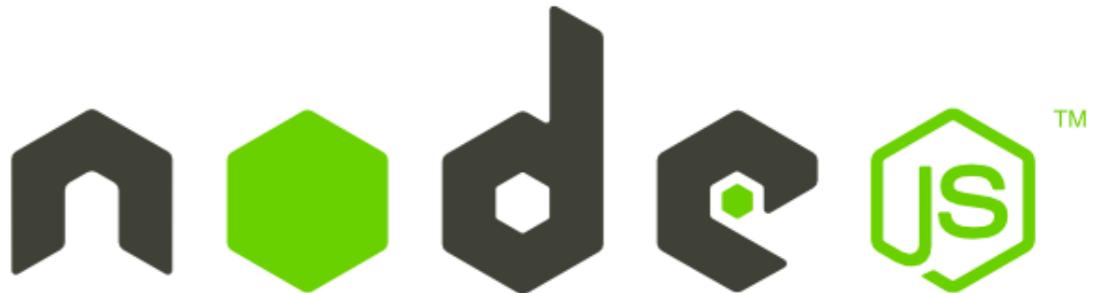
app.post('/profile', upload.single('avatar'), function (req, res, next) {
  // req.file is the `avatar` file
  // req.body will hold the text fields, if there were any
})

app.post('/photos/upload', upload.array('photos', 12), function (req, res, next) {
  // req.files is array of `photos` files
  // req.body will contain the text fields, if there were any
})

var cpUpload = upload.fields([{ name: 'avatar', maxCount: 1 }, { name: 'gallery', maxCount: 12 }])
app.post('/cool-profile', cpUpload, function (req, res, next) {
  // req.files is an object (String -> Array) where fieldname is the key
  //
  // e.g.
  //   req.files['avatar'][0] -> File
  //   req.files['gallery'] -> Array
  //
  // req.body will contain the text fields, if there were any
})
```

# Upload File

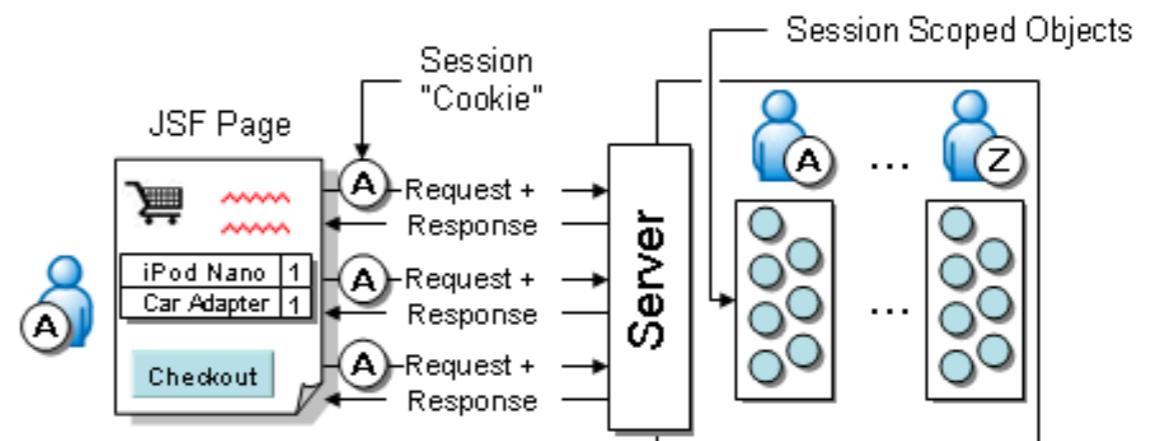
- Install '**formidable**' and 'fs'
- npm install formidable
- var formidable =  
require('formidable')
- [https://github.com/felixge/  
node-formidable](https://github.com/felixge/node-formidable)



*File Upload using Formidable*

# Session

- Using “express-session”
- npm install express-session
- ```
var session =  
require('express-session')
```
- <https://github.com/expressjs/session>



# Session (Example)

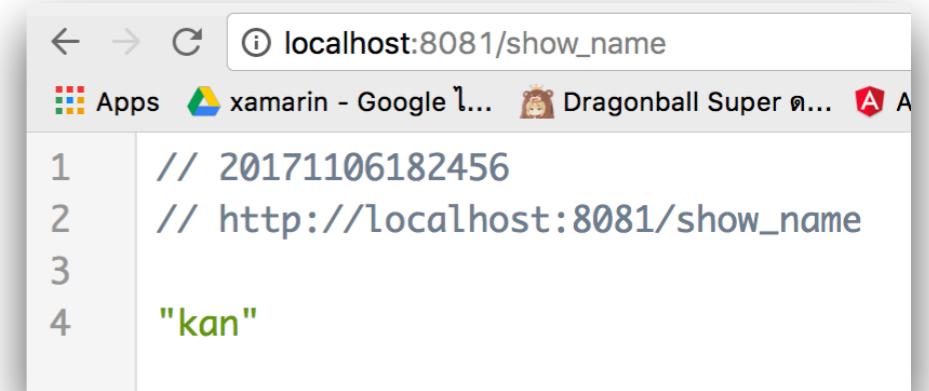
```
const express = require('express');
const path = require('path');
const session = require('express-session');
const app = express();
app.use(session({secret:'codemobiles', cookie:{maxAge:60000}, resave: true, saveUninitialized: false}));

app.get('/show_name', function (req, res) {
    console.log("show_name");
    res.json(req.session.name)
});

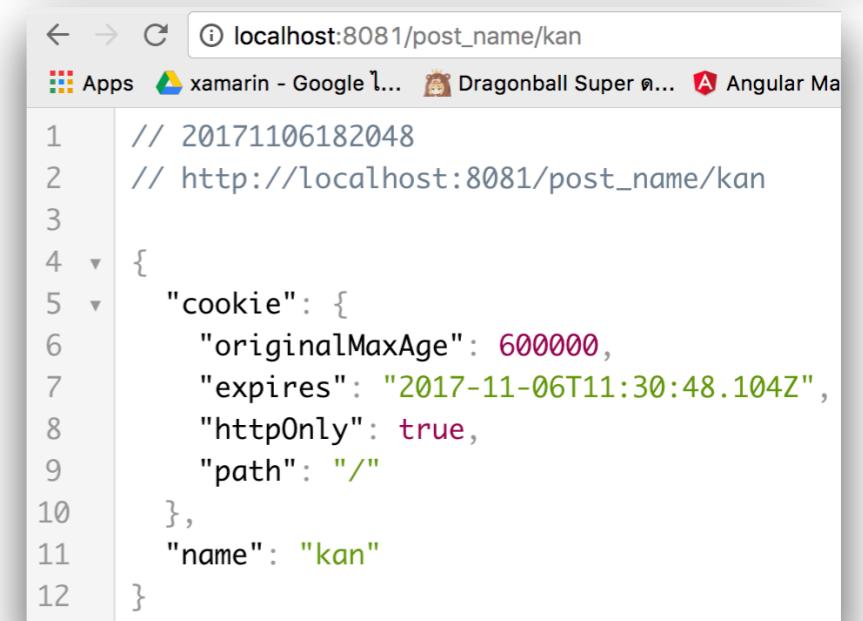
app.get('/post_name/:name', function (req, res) {
    req.session.name = req.params.name;
    res.json(req.session);
})

const server = app.listen(8081, function(){
    var host = server.address().address;
    var port = server.address().port;

    console.log('Running ... http://localhost%', host, port);
})
```

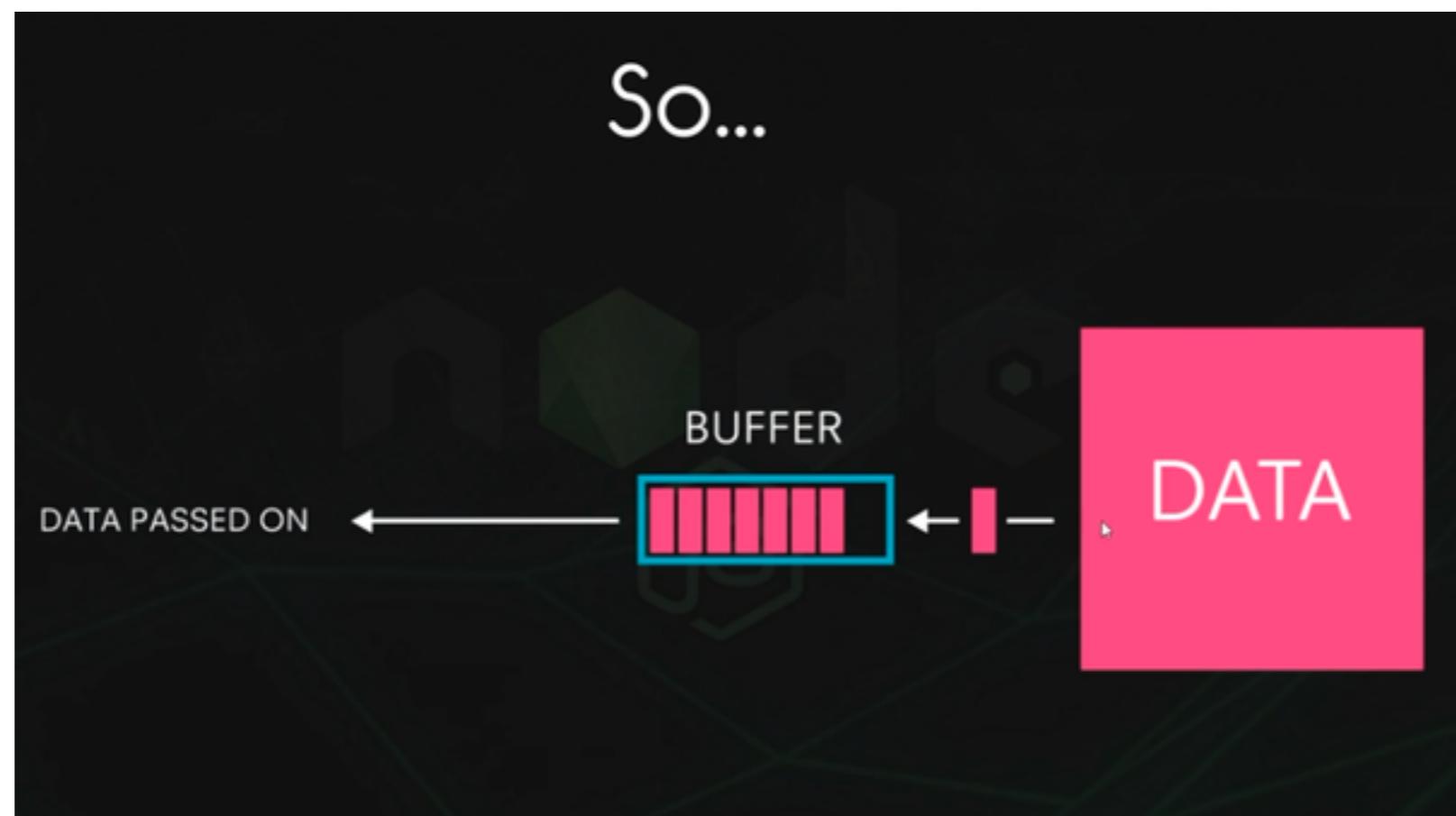


```
// 20171106182456
// http://localhost:8081/show_name
"kan"
```



```
// 20171106182048
// http://localhost:8081/post_name/kan
{
  "cookie": {
    "originalMaxAge": 600000,
    "expires": "2017-11-06T11:30:48.104Z",
    "httpOnly": true,
    "path": "/"
  },
  "name": "kan"
}
```

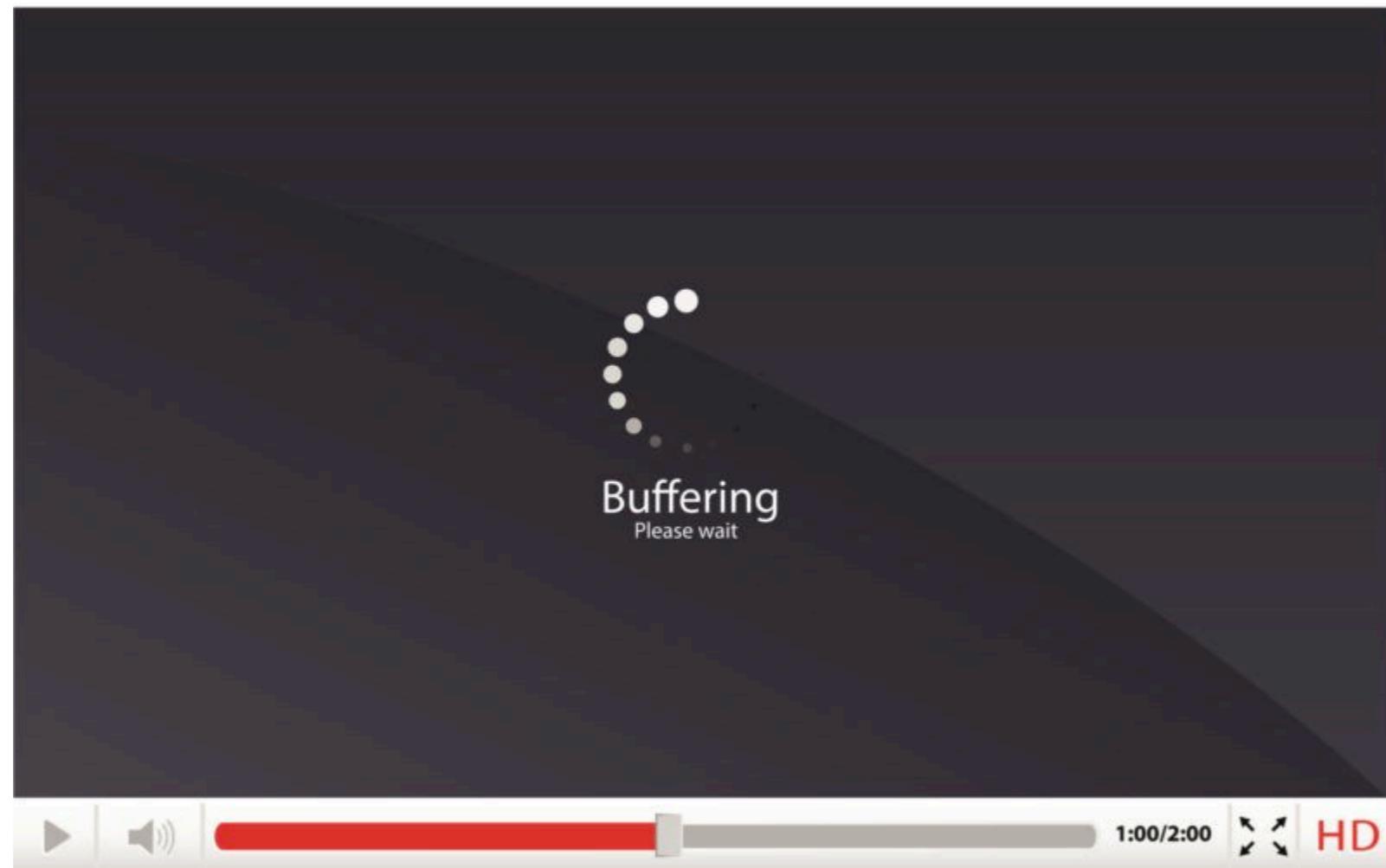
# Node Buffer



**Credit:**

<https://medium.com/front-end-hacking/js-buffers-matter-sometimes-56150a35417f>

# Node Buffer



**Credit:**

<https://medium.com/front-end-hacking/js-buffers-matter-sometimes-56150a35417f>

# Node Buffer

```
var string1 = new Buffer("codemobiles.com");
console.log(string1);
console.log(Buffer.isEncoding('utf8'))
console.log(Buffer.isEncoding('utf-8'))
console.log(Buffer.isEncoding('utf=8'))

var string2 = new Buffer(15);
string2.write("123456789", "utf8");

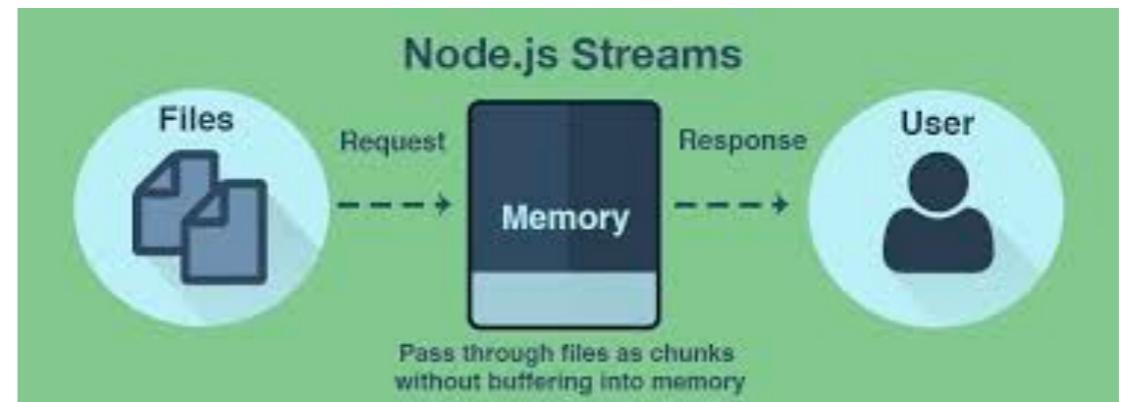
const offset = 9;
string2.write(" abcd", offset, "utf8");
console.log(string2.toString('utf8'))
```

# Node.js Stream

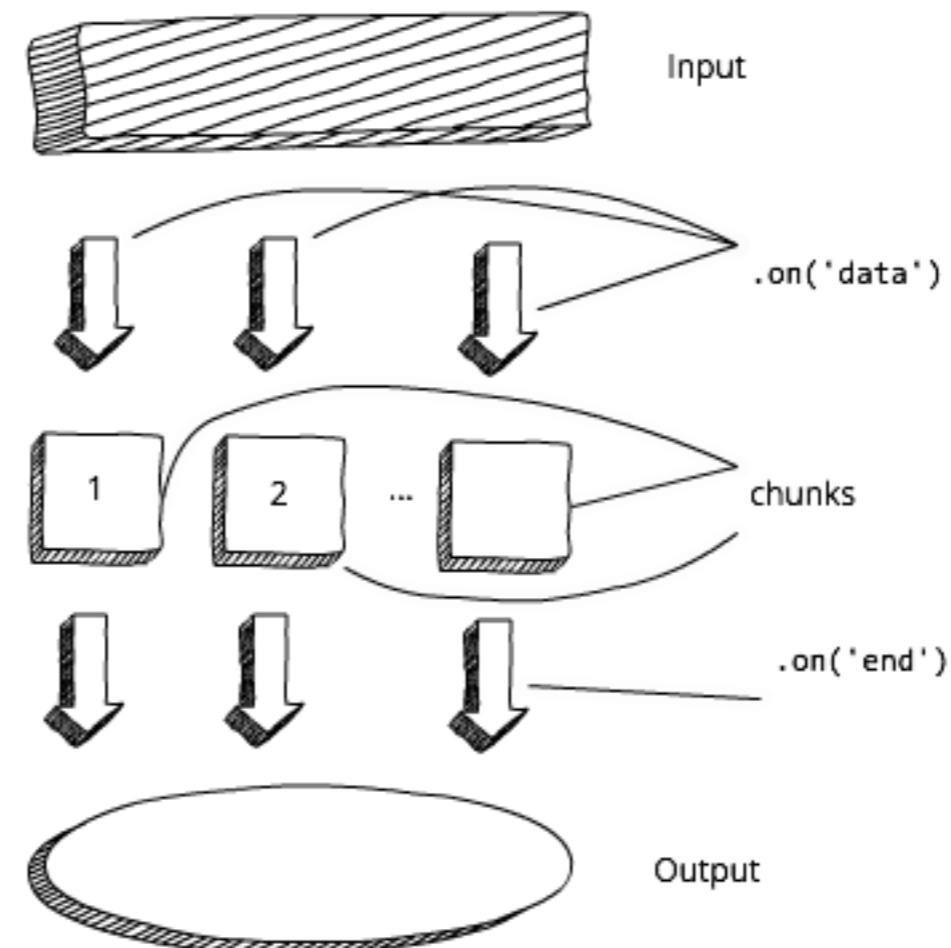
**Making Angular's SEO friendly**

# Node Stream

- Streams are collections of data—just like arrays or strings.
- The streams might not be available all at once, and they don't have to fit in memory. This makes streams really powerful when working with large amounts of data,
- Or data that's coming from an external source one *chunk* at a time.



# Node Stream



# Node Stream

## Streams

### Readable Streams

HTTP responses, on the client

HTTP requests, on the server

fs read streams

zlib streams

crypto streams

TCP sockets

child process stdout and stderr

process.stdin

### Writable Streams

HTTP requests, on the client

HTTP responses, on the server

fs write streams

zlib streams

crypto streams

TCP sockets

child process stdin

process.stdout, process.stderr

# Node Stream

```
//https://textmechanic.com/text-tools/numeration-tools/generate-list-numbers/
var fs = require("fs");
var data = '';

// Create a readable stream
var readerStream = fs.createReadStream('long_input.txt');

// Set the encoding to be utf8.
readerStream.setEncoding('UTF8');

// Handle stream events --> data, end, and error
readerStream.on('data', function(chunk) {
    data += chunk;
    if (data.length > 4000000){
        console.log(data)
        readerStream.destroy()
    }
});

readerStream.on('end',function() {
    console.log(data);
});

readerStream.on('error', function(err) {
    console.log(err.stack);
});

console.log("Program Ended");
```

# Node Stream

The screenshot shows a medium.com post by freeCodeCamp. The title is "Node.js Streams - Everything You Need To Know". The post content includes a code snippet demonstrating how to create a simple HTTP server using streams:

```
const fs = require('fs');
const server = require('http').createServer();

server.on('request', (req, res) => {
  const src = fs.createReadStream('./big.file');
  src.pipe(res);
});

server.listen(8000);
```

Below the code, there is a note: "Now when you connect to this server, a magical thing happens (look at the memory consumption):". Below the note, there is a screenshot of a Mac OS X Activity Monitor showing the memory usage of the "node" process. The process is using approximately 19.9 MB of memory. Below the Activity Monitor, there is a terminal window showing the output of a curl command to the server, which is a large block of Lorem ipsum text.

Credit : <https://medium.freecodecamp.org/node-js-streams-everything-you-need-to-know-c9141306be93>

# Node Non-Blocking I/O Concept

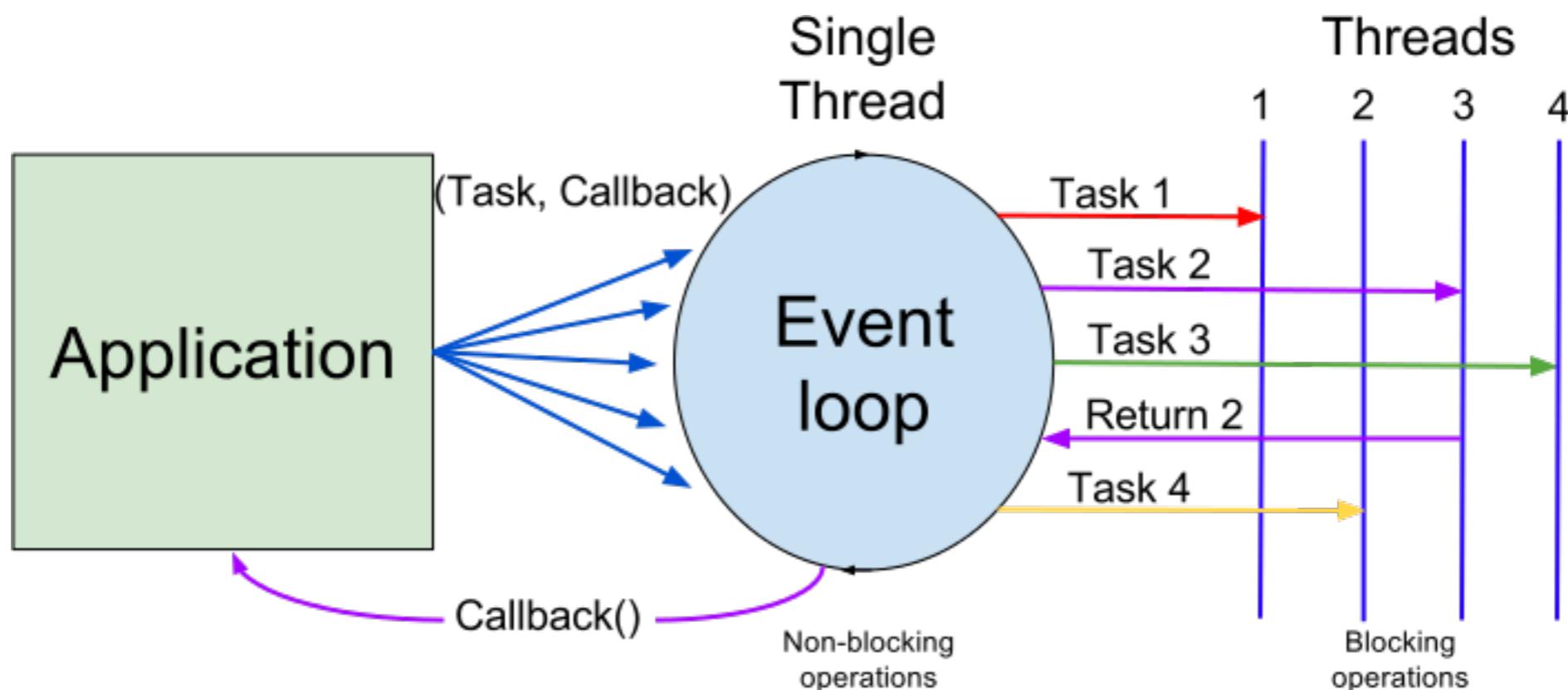
# Outline

- Why node is well for concurrent but intensive CPU consumption?
- Event-Loop or Main Thread?
- Phase of event-loop
- setTimeout, setInterval, setImmediate, process.nextTick.
- Thread or Worker Pool by libuv c++
- Non-Blocking Event-Loop
- Blocking Event-Loop
- Blocking Thread Pool
- Possible Blocking Causes
  - Vulnerable Regular Expression
  - JSON Parsing stringify
- Optimization
  - Partitioning works
  - Offloading to Thread pool

**non-blocking I/O,  
event loops,  
javascript and  
node.js.**

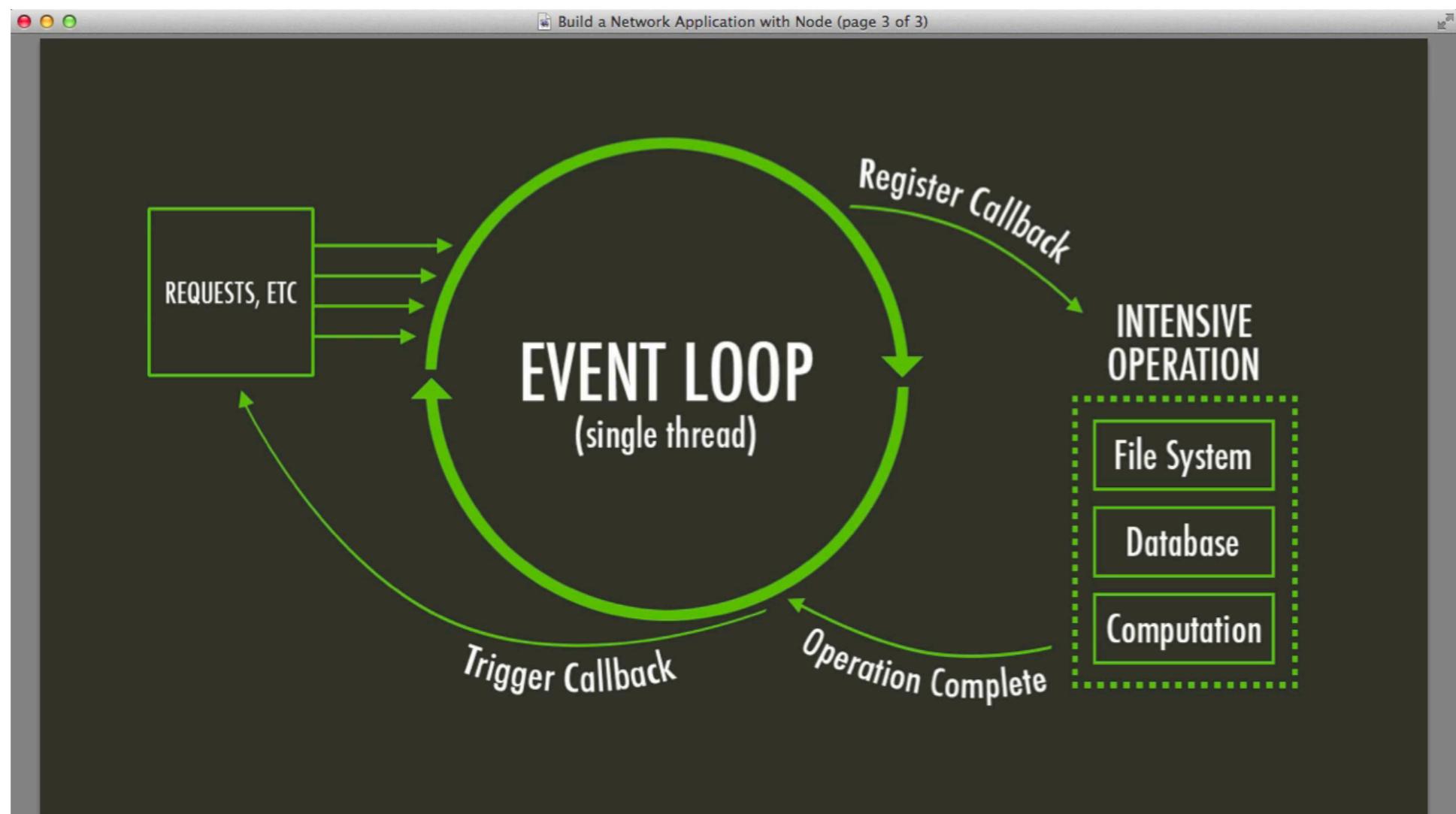
# Node.js

## Event-Loop and Thread-Pool



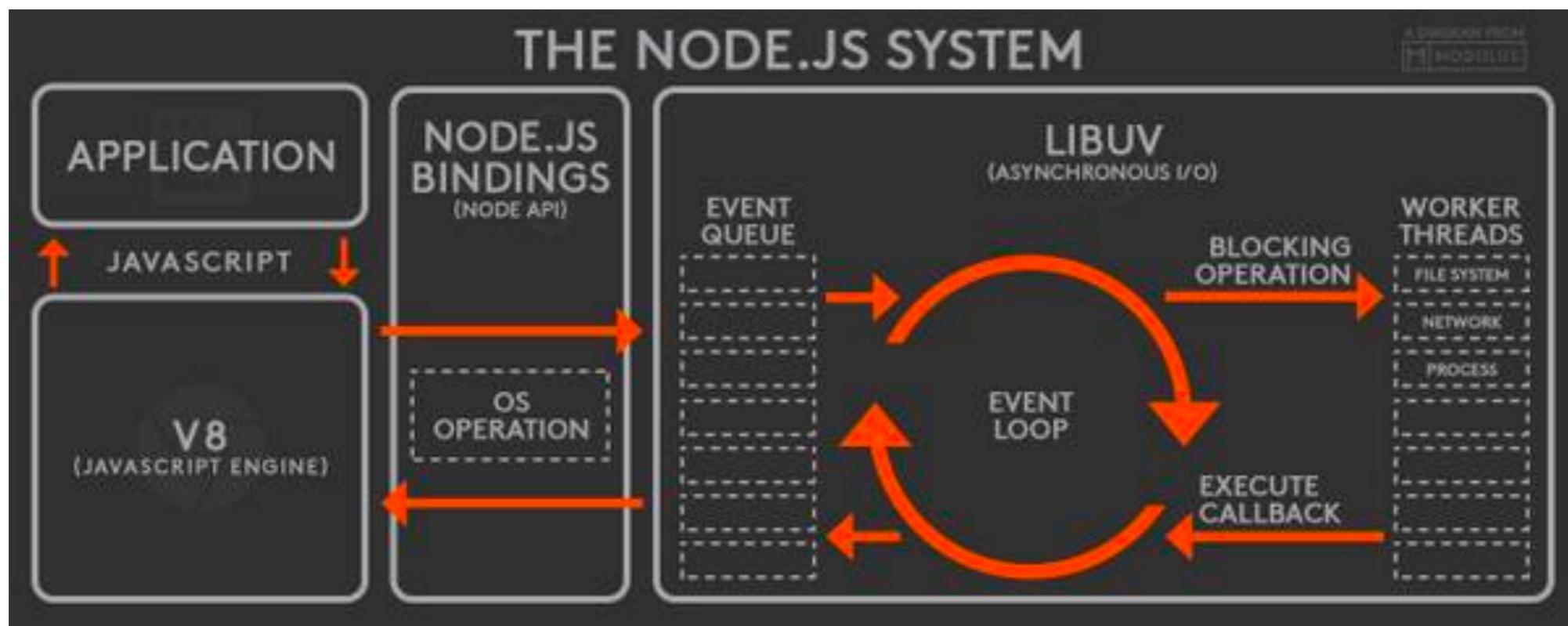
# Node

## Event-Loop and Thread-Pool

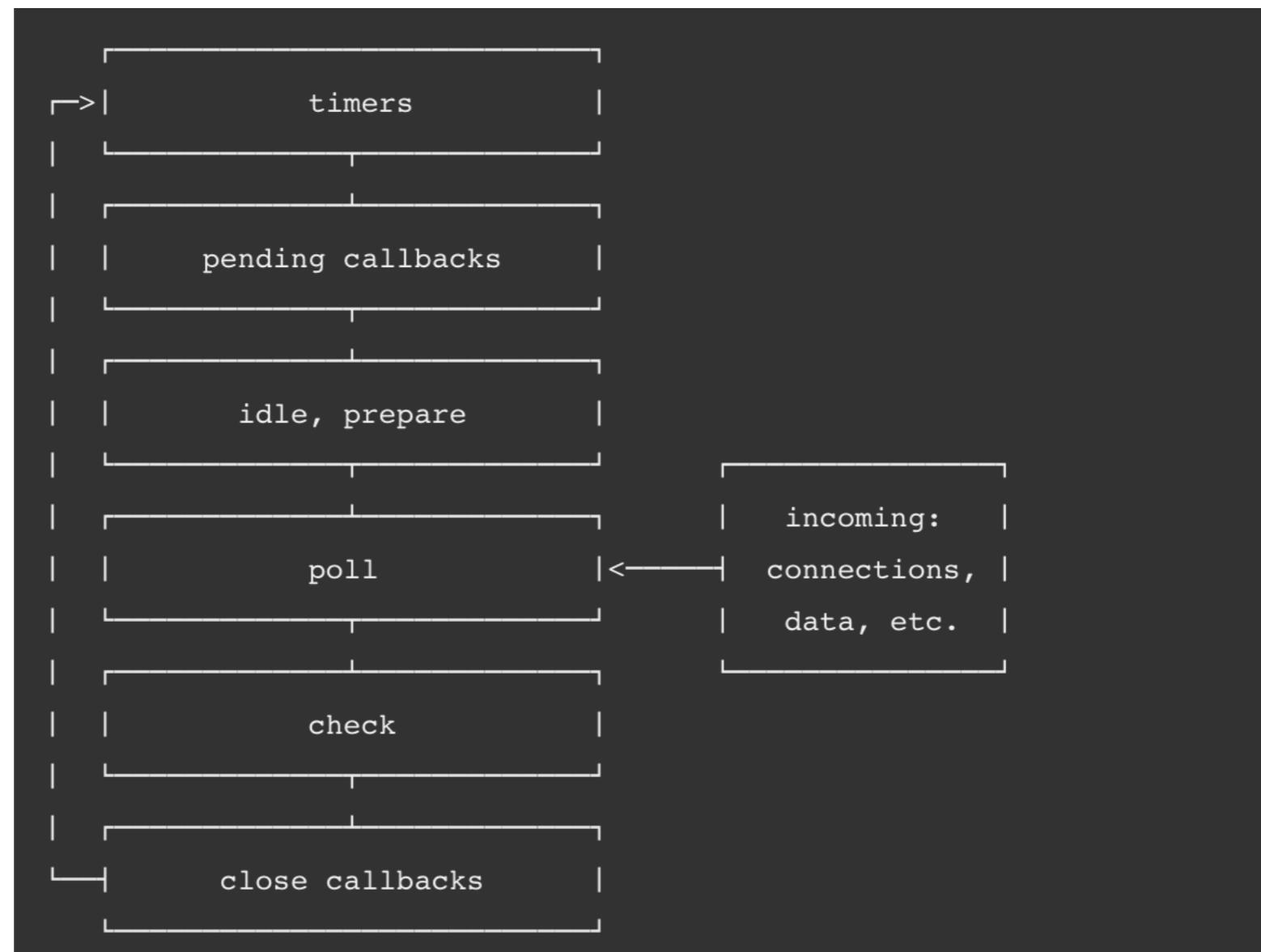


# Node

## Event-Loop and Thread-Pool



# Node Event Loop



# Node Event Loop

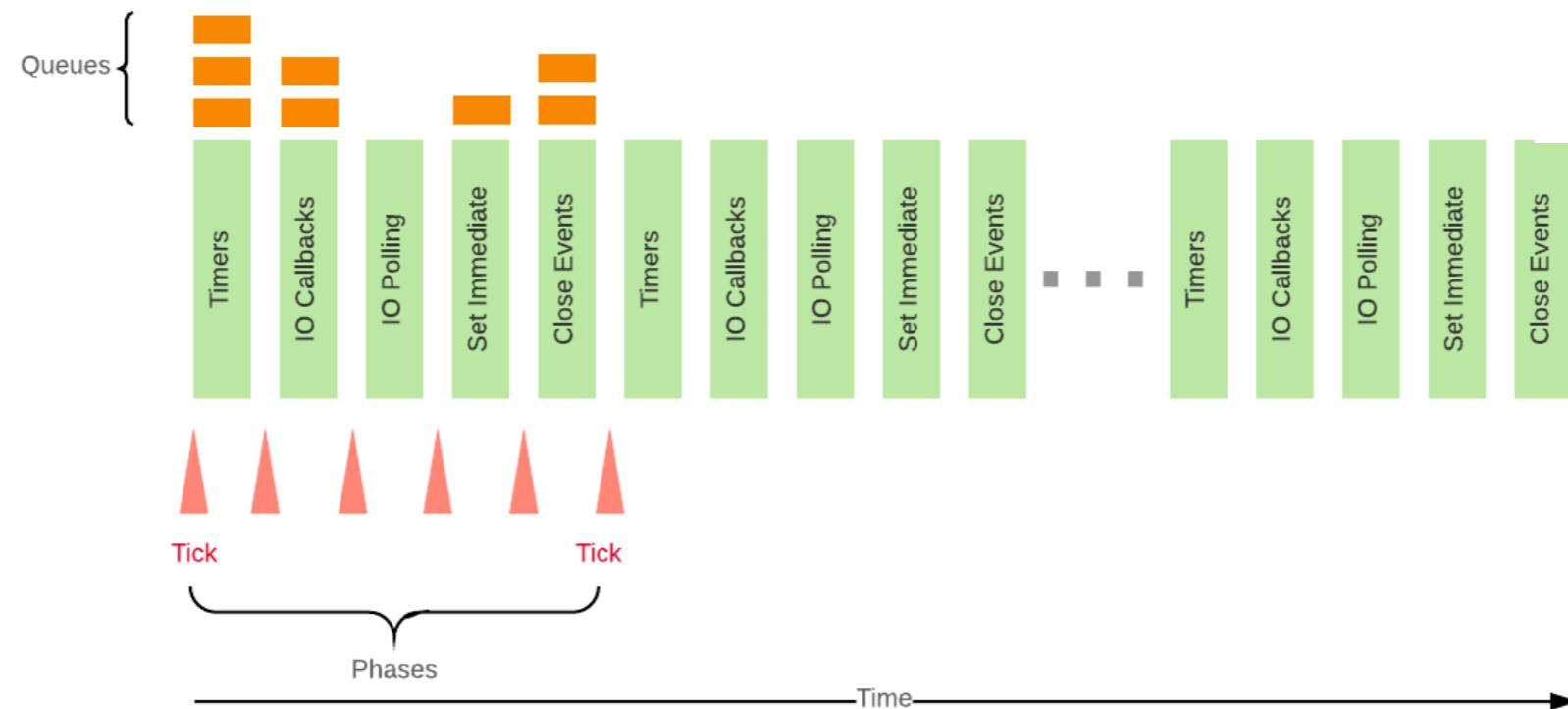
## Phases Overview

- **timers**: this phase executes callbacks scheduled by `setTimeout()` and `setInterval()`.
- **pending callbacks**: executes I/O callbacks deferred to the next loop iteration.
- **idle, prepare**: only used internally.
- **poll**: retrieve new I/O events; execute I/O related callbacks (almost all with the exception of close callbacks, the ones scheduled by timers, and `setImmediate()`); node will block here when appropriate.
- **check**: `setImmediate()` callbacks are invoked here.
- **close callbacks**: some close callbacks, e.g. `socket.on('close', ...)`.

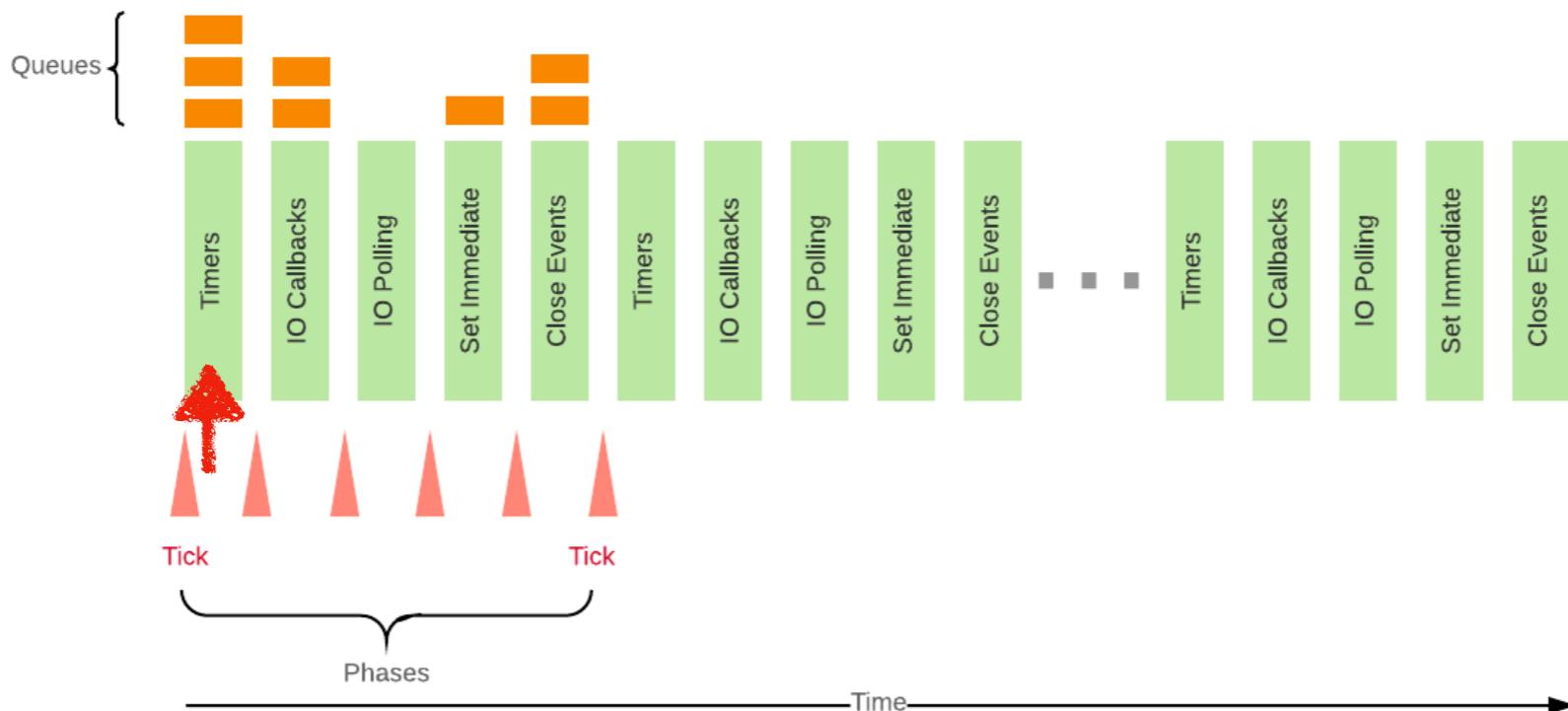
# Node Event Loop



calling these phases by round-robin manner

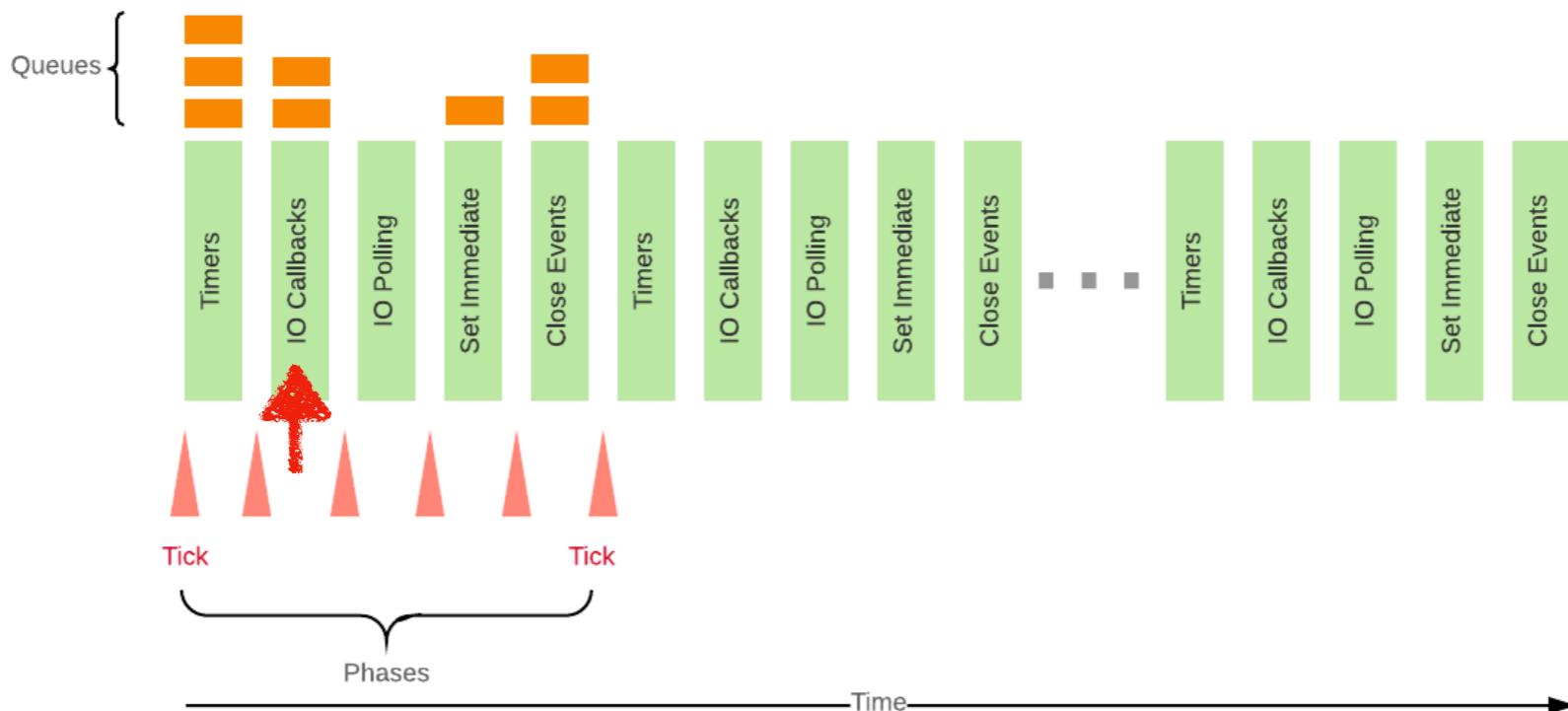


The event loop as a process which is a set of phases with specific tasks (mentioned previously) that are executed in a round-robin manner



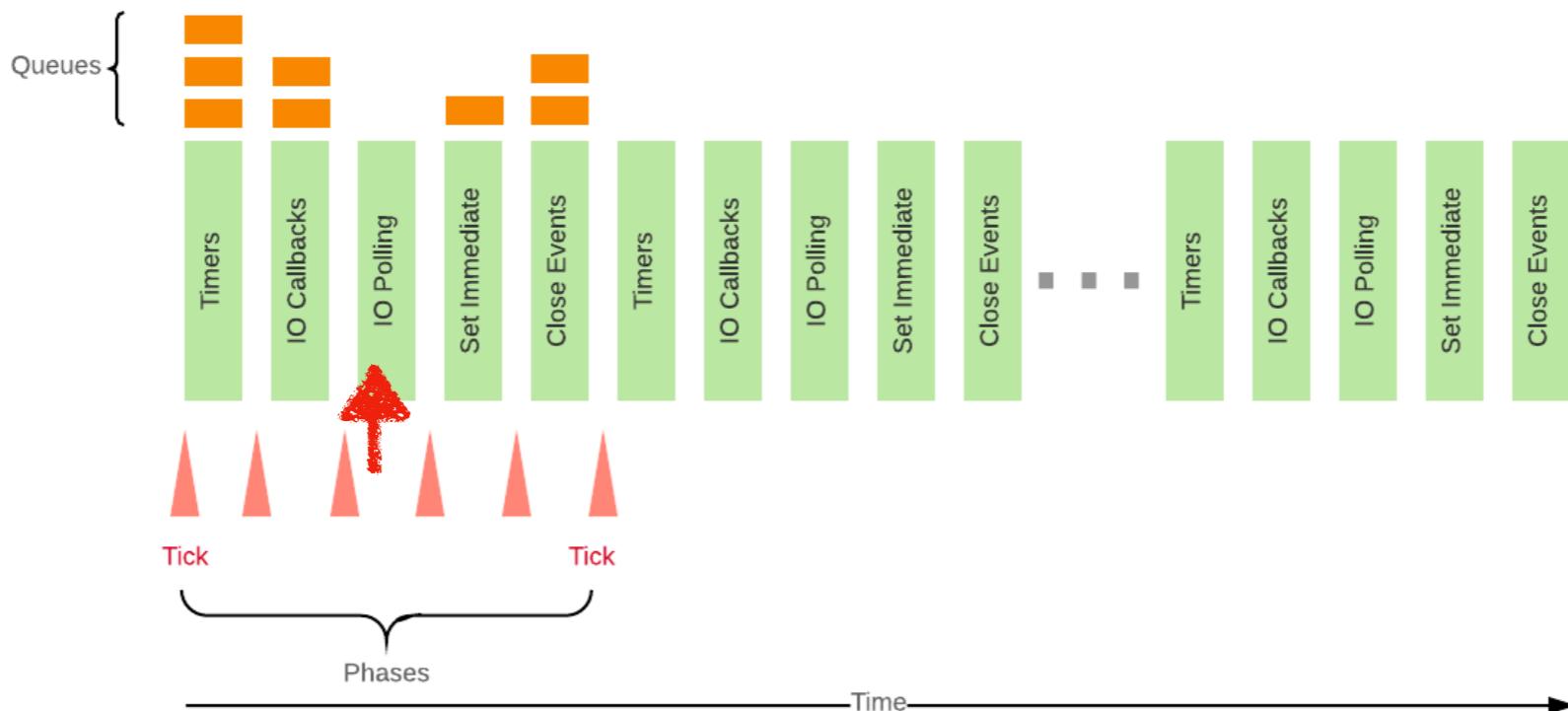
## Timer

Everything that was scheduled via `setTimeout()` or `setInterval()` will be executed here



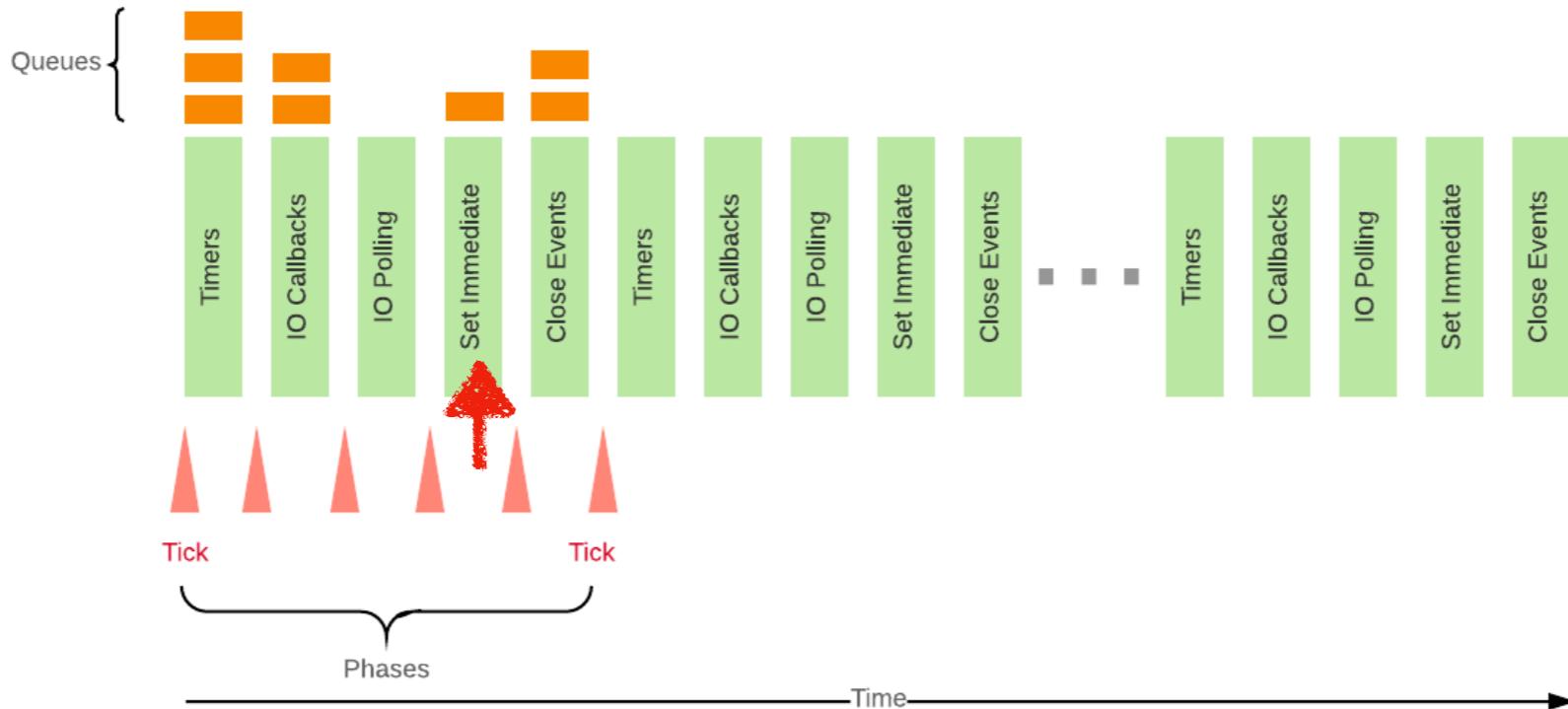
## IO Callbacks (Pending callbacks)

As all userland code in Node.js is basically in callback. This is the userland code to be excuted



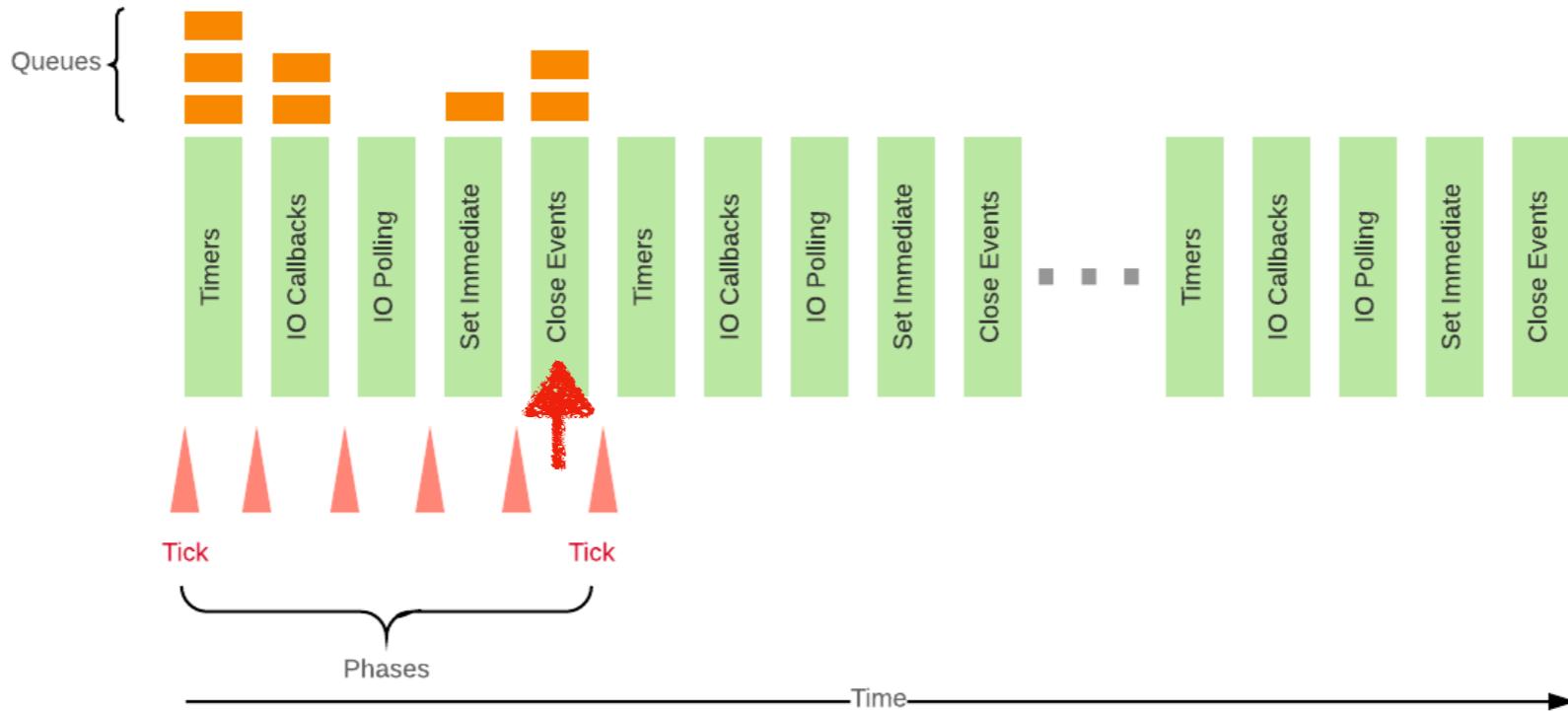
## IO Polling

Polls for new events to be executed on the next run.



## Set Immediate

Run all callbacks registered by calling `setImmediate()`.



## Close

Here all on ('close') event callbacks are processed

# Node

## Timer Phase

```
const fs = require('fs');

function someAsyncOperation(callback) {
    // Assume this takes 95ms to complete
    fs.readFile('/path/to/file', callback);
}

const timeoutScheduled = Date.now();

setTimeout(() => {
    const delay = Date.now() - timeoutScheduled;
    console.log(` ${delay}ms have passed since I was scheduled`);
}, 100);

// do someAsyncOperation which takes 95 ms to complete
someAsyncOperation(() => {
    const startCallback = Date.now();

    // do something that will take 10ms...
    while (Date.now() - startCallback < 10) {
        // do nothing
    }
});
```

# Node

## Scheduling : setTimeout

Allows to run a function once after the interval of time.

```
let timerId = setTimeout(func|code, delay[, arg1, arg2...])
```

Clear Timeout

```
let timerId = setTimeout(() => alert("never happens"), 1000);  
alert(timerId); // timer identifier
```

```
clearTimeout(timerId);  
alert(timerId); // same identifier (doesn't become null after  
canceling)
```

# Node

## Scheduling : setInterval

Allows to run a function regularly with the interval between the runs.

```
let timerId = setInterval(func|code, delay[, arg1, arg2...])
```

### Clear Interval

```
// repeat with the interval of 2 seconds
let timerId = setInterval(() => alert('tick'), 2000);

// after 5 seconds stop
setTimeout(() => { clearInterval(timerId); alert('stop'); }, 5000);
```

# Node

## nextTick vs setImmediate

### process.nextTick() vs setImmediate()

We have two calls that are similar as far as users are concerned, but their names are confusing.

- `process.nextTick()` fires immediately on the same phase
- `setImmediate()` fires on the following iteration or 'tick' of the event loop

```
function callback(msg){  
  console.log("run by ", msg)  
}
```

```
setImmediate(callback.bind(null,"setImmediate()")) // will be run immediately in next loop  
process.nextTick(callback.bind(null,"nextTick()")); // will be run in next phase (tick) – faster
```

# Don't Block Event Loop

- Node uses a small number of threads to handle many clients
- There are TWO types of threads in nodeJS
  - Event Loop (main thread)
  - Worker (thread pool)
- Any work job taking a long time to execute a callback (Event Loop) or a (Worker Pool), we call it 'blocked'.
- Blocking event loop from one client is not good and cause nodeJS cannot handle requests from any other client

# Don't Block Event Loop

```
const express = require('express');
const app = express();

// Used to test if event-loop is block
app.get('/', (req, res) => res.send('Hello, World'));

app.get('/block', (req, res) => {
  const end = Date.now() + 5000;
  while (Date.now() < end) {
    const doSomethingHeavyInJavaScript = 1 + 2 + 3;
  }
  res.send('I am done!');
});

app.get('/non-block', (req, res) => {
  // Imagine that setTimeout is IO operation
  // setTimeout is a native implementation, not the JS
  setTimeout(() => res.send('I am done!'), 5000);
});

app.listen(3000, () => console.log('app listening on port 3000'));
```

# Offload to Thread Pool

## workerpool

---

JavaScript is based upon a single event loop which handles one event at a time. Jeremy Epstein [explains this clearly](#):

In Node.js everything runs in parallel, except your code. What this means is that all I/O code that you write in Node.js is non-blocking, while (conversely) all non-I/O code that you write in Node.js is blocking.

## Features

---

- Easy to use
- Runs in the browser and on node.js
- Dynamically offload functions to a worker
- Access workers via a proxy
- Cancel running tasks
- Set a timeout on tasks
- Handles crashed workers
- Small, less than 4 kB minified and gzipped

```
npm install workerpool
```

# Offload to Thread Pool

**EX1:**

```
var workerpool = require('workerpool');
var pool = workerpool.pool();

function add(a, b) {
    return a + b;
}

pool.exec(add, [3, 4])
  .then(function (result) {
    console.log('result', result); // outputs 7
  })
  .catch(function (err) {
    console.error(err);
  })
  .then(function () {
    pool.terminate(); // terminate all workers when done
});
```

# Offload to Thread Pool

**EX2:**

```
const express = require('express');
const app = express();
var workerpool = require('workerpool');
var pool = workerpool.pool();

function blockWork() {
  const end = Date.now() + 10000;
  while (Date.now() < end) {
    const doSomethingHeavyInJavaScript = 1 + 2 + 3;
  }
}

app.get('/', (req, res) => res.send('Hello, World'));

app.get('/block', (req, res) => {
  pool.exec(blockWork, [])
    .then(function (result) {
      res.send('I am done!');
    })
    .catch(function (err) {
      console.error(err);
    })
    .then(function () {
      pool.terminate(); // terminate all workers when done
    });
});
app.listen(3000, () => console.log('app listening on port 3000'));
```

# Tuning Thread Pool

- By default, libuv will create a thread pool with the size of 4. The default size of the pool can be overridden by setting the environment variable named “UV\_THREADPOOL\_SIZE”
- There are 2 ways to set the size
  - command line : “UV\_THREADPOOL\_SIZE=64 node”
  - in javascript : `process.env.UV_THREADPOOL_SIZE=64`

# Blocking the Event Loop

## Node Core Modules

Blocking the Event Loop: Node core modules

Several Node core modules have synchronous expensive APIs, including:

- [Encryption](#)
- [Compression](#)
- [File system](#)
- [Child process](#)

# Blocking the Event Loop

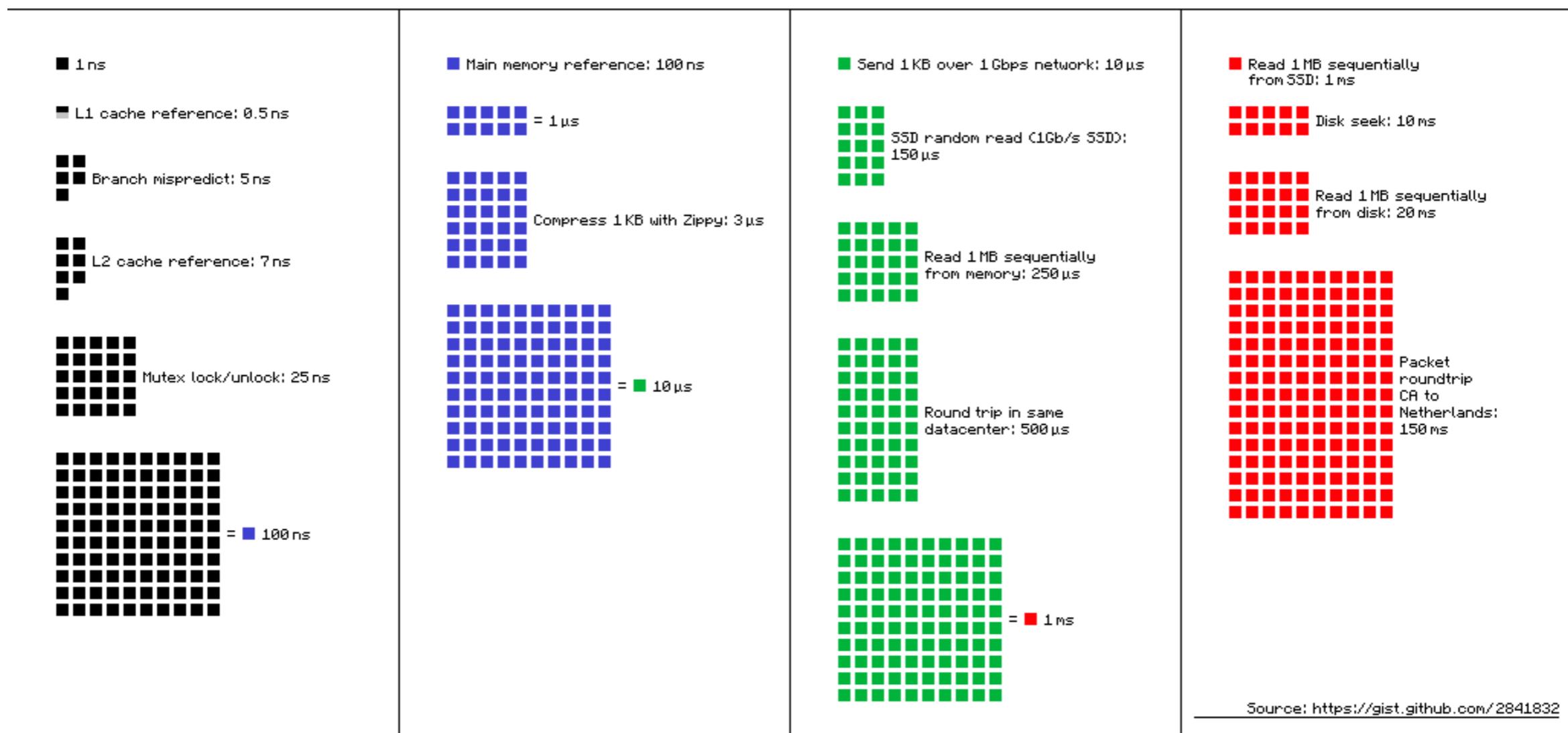
## In nodeJS Modules

**In a server, you should not use the following synchronous APIs from these modules:**

- Encryption:
  - `crypto.randomBytes` (synchronous version)
  - `crypto.randomFillSync`
  - `crypto.pbkdf2Sync`
  - You should also be careful about providing large input to the encryption and decryption routines.
- Compression:
  - `zlib.inflateSync`
  - `zlib.deflateSync`
- File system:
  - Do not use the synchronous file system APIs. For example, if the file you access is in a **distributed file system** like **NFS**, access times can vary widely.
- Child process:
  - `child_process.spawnSync`
  - `child_process.execSync`
  - `child_process.execFileSync`

# IO & Ram Latency

Latency Numbers Every Programmer Should Know



# Latency Monitoring Tool

## Performance Testing Apache Benchmark

### 4 Load testing

```
$ ab -n 10000 -c 10 http://localhost/ # c too low (server is not
$ ab -n 10000 -c 100 http://localhost/ # saturated, response
$ ab -n 10000 -c 250 http://localhost/ # times are stable)
$ ab -n 10000 -c 500 http://localhost/
$ ab -n 10000 -c 750 http://localhost/ # c too high (server is
$ ab -n 10000 -c 1000 http://localhost/ # saturated, response)
   times are increasing)
```



# Latency Monitoring Tool

**Windows: XAMPP**  
**macOS : Installed by default**

It sounds like you don't want to install Apache, but you do want [Apache Bench](#). May I recommend [XAMPP](#)? Download the zip or 7zip versions (much larger download than regular Apache but you won't have to install). Unzip the download.

This is where what I would recommend, and where you currently are converge. Since you already have Apache installed you should already have Apache Bench.

You will find it under

```
.\apache\bin\ab.exe
```

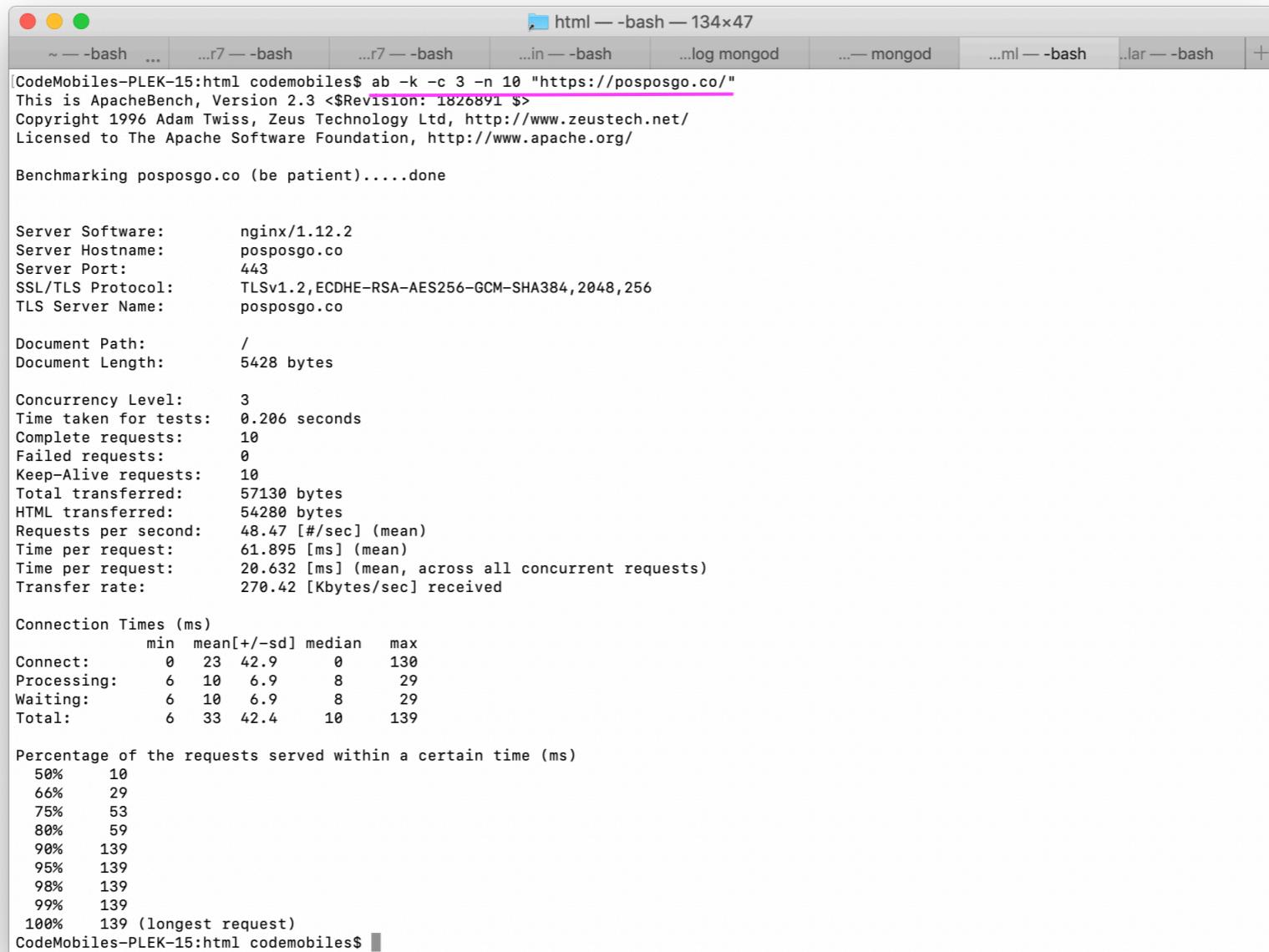
in the zip file, if that is what you downloaded, it will be under

```
.\xampp\apache\bin\ab.exe
```

# Latency Monitoring Tool

```
NODE_ENV=production node --prof app.js
```

```
ab -k -c 3 -n 10 "https://posposgo.co/"
```



The screenshot shows a macOS terminal window titled "html — bash — 134x47". The window contains the output of an ApacheBench command. The command was run with the following parameters: ab -k -c 3 -n 10 "https://posposgo.co/". The output provides detailed performance metrics for the request.

```
[CodeMobiles-PLEK-15:html codemobiles$ ab -k -c 3 -n 10 "https://posposgo.co/"
This is ApacheBench, Version 2.3 <$Revision: 1826891 $>
Copyright 1996 Adam Twiss, Zeus Technology Ltd, http://www.zeustech.net/
Licensed to The Apache Software Foundation, http://www.apache.org/

Benchmarking posposgo.co (be patient).....done

Server Software:      nginx/1.12.2
Server Hostname:     posposgo.co
Server Port:          443
SSL/TLS Protocol:   TLSv1.2,ECDHE-RSA-AES256-GCM-SHA384,2048,256
TLS Server Name:    posposgo.co

Document Path:        /
Document Length:     5428 bytes

Concurrency Level:   3
Time taken for tests: 0.206 seconds
Complete requests:   10
Failed requests:     0
Keep-Alive requests: 10
Total transferred:   57130 bytes
HTML transferred:   54280 bytes
Requests per second: 48.47 [#/sec] (mean)
Time per request:   61.895 [ms] (mean)
Time per request:   20.632 [ms] (mean, across all concurrent requests)
Transfer rate:       270.42 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        0    23   42.9     0    130
Processing:     6    10    6.9     8    29
Waiting:        6    10    6.9     8    29
Total:         6    33   42.4    10    139

Percentage of the requests served within a certain time (ms)
  50%    10
  66%    29
  75%    53
  80%    59
  90%   139
  95%   139
  98%   139
  99%   139
100%   139 (longest request)
CodeMobiles-PLEK-15:html codemobiles$
```

# **Middleware Express**

If you'll write a nodejs app on server, you can't live without express module

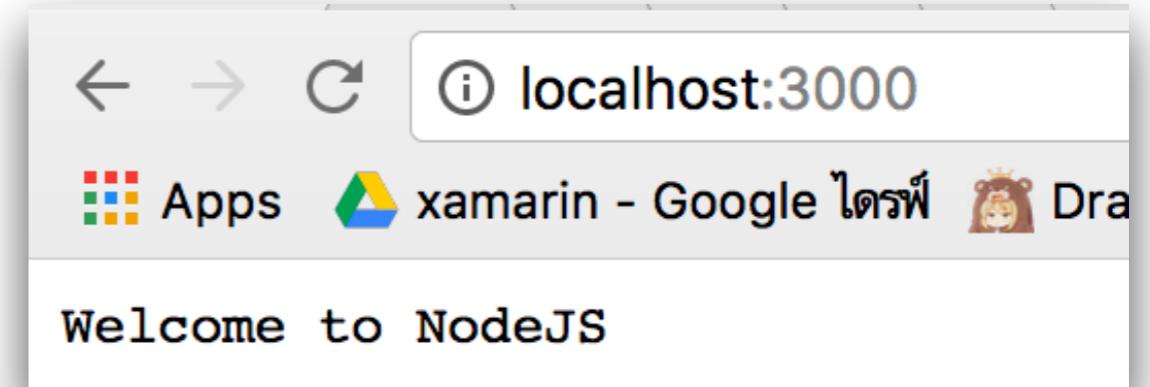
# Express

## Hello World

```
var express = require('express');
var app = express();

app.get('/', function(req, res){
  res.end('Welcome to NodeJS');
});

var server = app.listen(3000, function(){
  var host = server.address().address;
  var port = server.address().port;
  console.log("Listening at http://%s:%s", host, port);
});
```



# Express Routing

```
app.get('/', function(req, res){  
  res.end('Welcome to NodeJS');  
});  
  
app.get('/profile', function(req, res){  
  res.end('Profile');  
});  
  
app.get('/show', function(req, res){  
  res.json({res: 'show show'});  
});  
  
app.get('/authen', function(req, res){  
  res.sendFile(path.join(__dirname, '/public/login.html'));  
});
```

# Express

# Route Methods

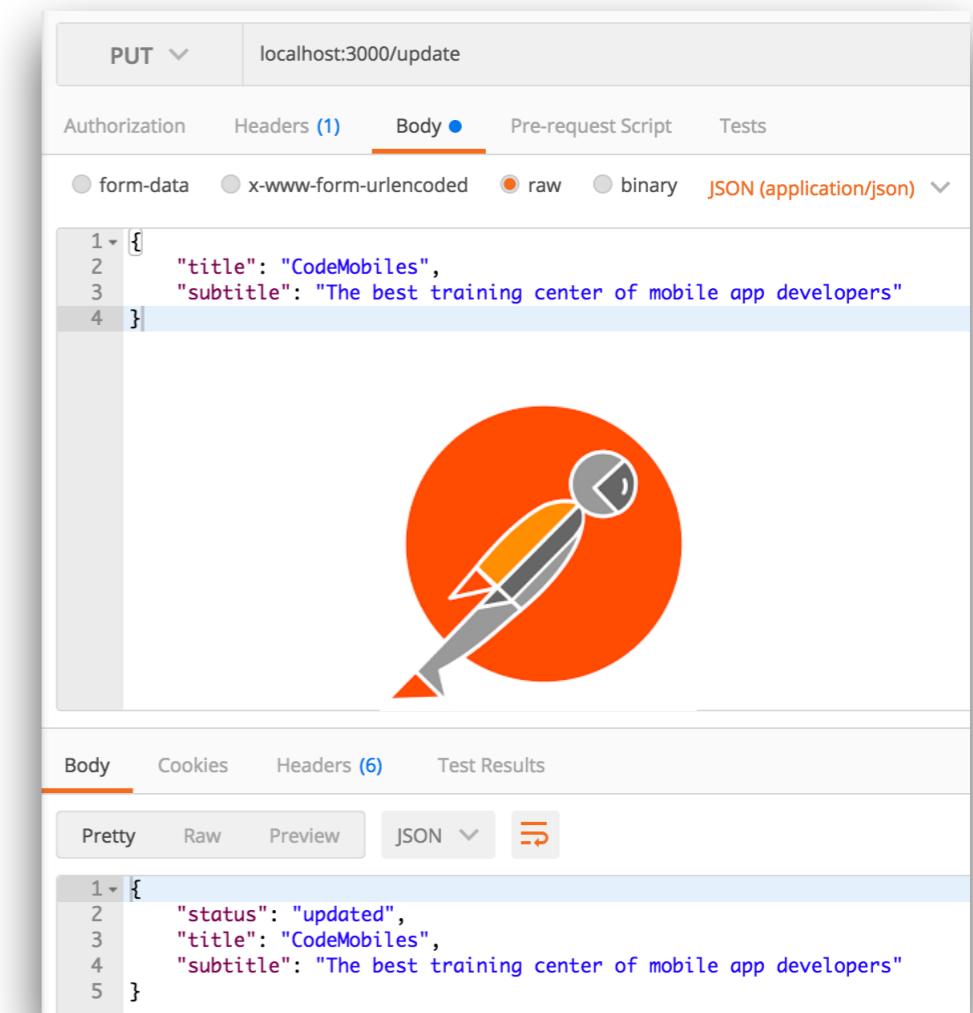
```
var express = require('express');
var bodyParser = require('body-parser')
var app = express();
app.use(bodyParser.json())

app.get('/', function(req, res){
  res.json({res: 'hello world'});
});

app.post('/add', function(req, res){
  var _title = req.body.title;
  var _subtitle = req.body.subtitle;
  res.json({status: 'added', title: _title, subtitle: _subtitle});
});

app.put('/update', function(req, res){
  var _title = req.body.title;
  var _subtitle = req.body.subtitle;
  res.json({status: 'updated', title: _title, subtitle: _subtitle});
});

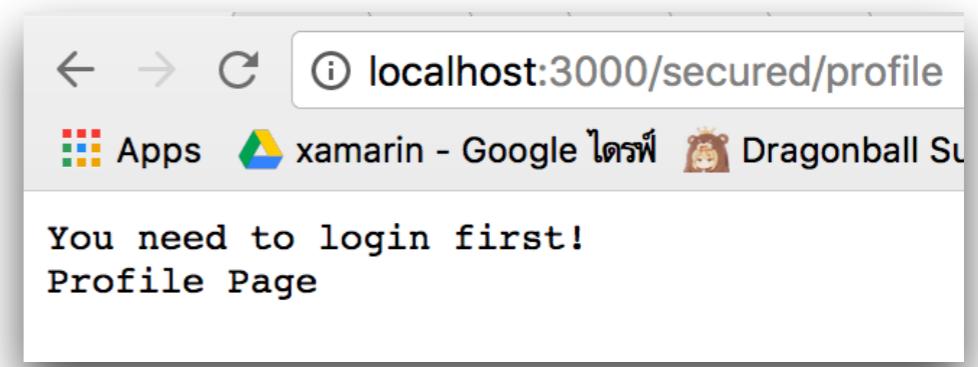
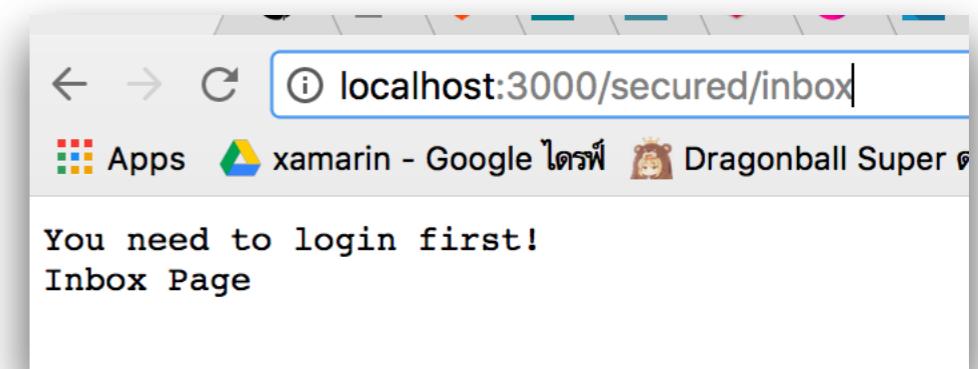
app.delete('/del', function(req, res){
  var _title = req.body.title;
  var _subtitle = req.body.subtitle;
  res.json({status: 'deleted', title: _title, subtitle: _subtitle});
})
```



# Express

## all and next function

```
app.all('/secured/*', function(req, res, next){  
    console.log('Accessing the secret section');  
    res.write('You need to login first!\n');  
    next();  
});  
  
app.get('/', function(req, res){  
    res.end("Home Page");  
});  
  
app.get('/secured/profile', function(req, res){  
    res.end('Profile Page');  
});  
  
app.get('/secured/inbox', function(req, res){  
    res.end('Inbox Page');  
});
```



# Express

# GET / POST / PUT / DELETE

```
app.get('/', function(req, res){  
  res.json({res: 'hello world'});  
});  
  
app.post('/add', function(req, res){  
  var _title = req.body.title;  
  var _subtitle = req.body.subtitle;  
  res.json({status: 'added', title: _title, subtitle: _subtitle});  
});  
  
app.put('/update', function(req, res){  
  var _title = req.body.title;  
  var _subtitle = req.body.subtitle;  
  res.json({status: 'updated', title: _title, subtitle: _subtitle});  
});  
  
app.delete('/del', function(req, res){  
  var _title = req.body.title;  
  var _subtitle = req.body.subtitle;  
  res.json({status: 'deleted', title: _title, subtitle: _subtitle});  
})
```

The screenshot shows the Postman application interface. At the top, there is a toolbar with various icons and a dropdown menu labeled "No Environment". Below the toolbar, a header bar displays "POST" and the URL "localhost:3000/add". On the right side of the header bar are buttons for "Params" and "Send". Underneath the header, there are tabs for "Authorization", "Headers (1)", "Body" (which is selected), and "Pre-request Script" and "Tests". The "Body" tab has sub-options for "form-data", "x-www-form-urlencoded", "raw", and "binary", with "raw" and "JSON (application/json)" selected. The "Body" field contains the following JSON payload:

```
1 {  
2   "title": "CodeMobiles",  
3   "subtitle": "The best training center of mobile app developers"  
4 }
```

At the bottom of the interface, there are tabs for "Body", "Cookies", "Headers (6)", and "Test Results", with "Body" selected. To the right of these tabs, the status is shown as "Status: 200 OK". Below the tabs, there are buttons for "Pretty", "Raw", "Preview", and "JSON" (with a dropdown arrow). The "JSON" section displays the same response JSON as the body field:

```
1 {  
2   "status": "added",  
3   "title": "CodeMobiles",  
4   "subtitle": "The best training center of mobile app developers"  
5 }
```

# Express

## Route Path

```
// root path
app.get('/', function(req, res){
  res.end('default');
});

// fix path
app.get('/api/add', function(req, res){
  res.end('/api/add');
});

// ? = optional
// ex: /abcd or /acd
app.get('/ab?cd', function(req, res){
  res.end('/ab?cd');
});

// + with same...
// ex: /x1cd, x11cd, x11111cd (not call if xcd only)
app.get('/x1+cd', function(req, res){
  res.end('/x1+cd', This route path will match ex: /x1cd, x11cd, x11111cd');
});
```

# Express

## Route Paramaters

```
/*
Route path: /users/:userId/books/:bookId
Request URL: http://localhost:3000/users/34/books/8989
req.params: { "userId": "34", "bookId": "8989" }
*/
app.get('/users/:userId/books/:bookId', function(req, res){
  console.log("users: " + req.params.userId);
  console.log("book: " + req.params.bookId);
  res.json({userId: req.params.userId, bookId: req.params.bookId});
});

/*
localhost:3000/flights/23-42
{
  "from": "23",
  "to": "42"
}
*/
app.get('/flights/:from-:to', function(req, res){
  res.json({from: req.params.from, to: req.params.to});
});
```

# Express

## Nest Handler #1

```
app.get('/example/a', function (req, res) {
  //res.json({result: "/example/a"});
  res.end('/example/a');
});

app.get('/example/b', function (req, res, next) {
  res.write('1:) /example/b');
  next();
}, function (req, res) {
  res.write(' ');
  res.end("2:) /example/b");
});
```

```
var f1 = function (req, res, next) {
  console.log("f1");
  req.pass = "f1";
  next()
};

var f2 = function (req, res, next) {
  console.log("f2");
  req.pass = req.pass + ", f2";
  next();
};

var f3 = function (req, res, next) {
  console.log("f3");
  req.pass = req.pass + ", f3";
  next();
};

app.get('/example/array/', f1, f2, f3, function (req, res) {
  res.json({ result: '/example/array', pass: req.pass })
})
```

# Express

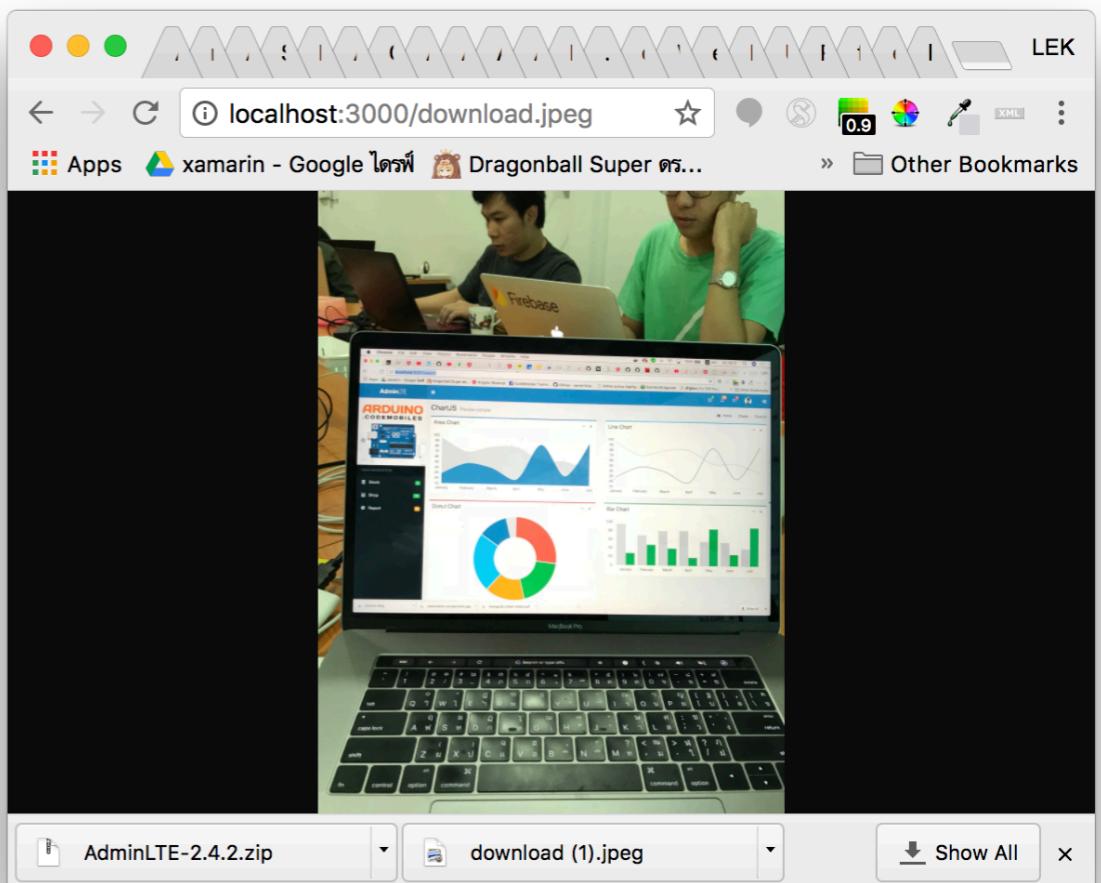
# Base Directory of Static Files

```
var express = require('express');
var path = require('path');

var app = express();
app.use(express.static(path.join(__dirname, '/public')))

// now you download file directly
// localhost:3000/download.jpeg

var server = app.listen(3000, function(){
  var host = server.address().address;
  var port = server.address().port;
  console.log("Listening at http://%s:%s", host, port);
});
```

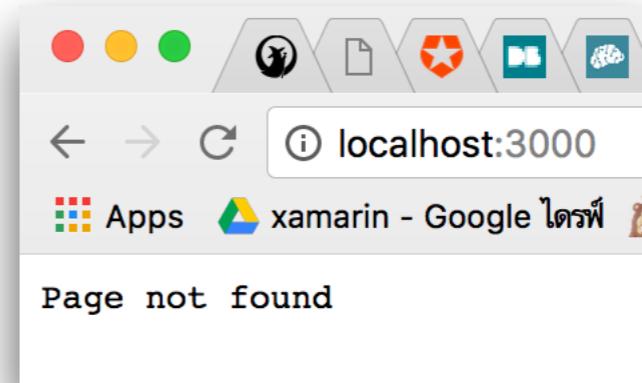
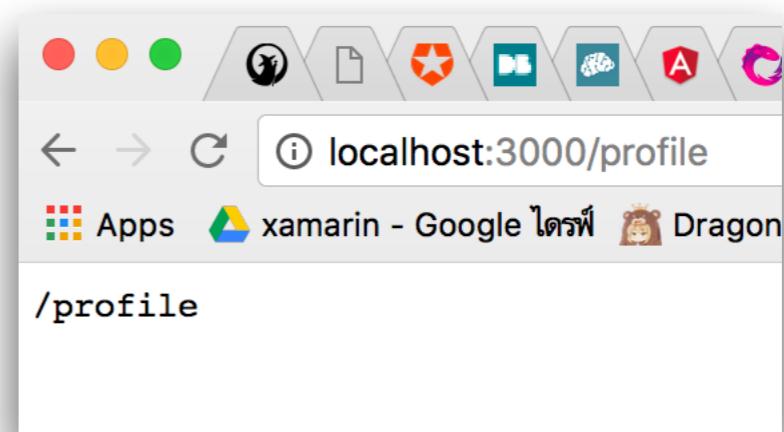


# Express

# Page not found (Error Handler)

```
app.get('/profile', function(req, res){  
  res.end('/profile');  
});
```

```
// any unroutes will be replied with this handler  
app.use(function(req, res, error){  
  res.end('Page not found');  
});
```



# Express

## Smart Route

```
var express = require('express');
var app = express();

app.route('/routes').get(function(req, res){
    res.json({result: "get was called"});
}).post(function(req, res){
    res.json({result: "post was called"});
}).put(function(req, res){
    res.json({result: "put was called"});
}).delete(function(req, res){
    res.json({result: "delete was called"});
});

var server = app.listen(3000, function(){
    var host = server.address().address;
    var port = server.address().port;
    console.log("Listening at http://%s:%s", host, port);
});
```

# Express

## Make Router with Module.exports

```
var express = require('express');
var router = express.Router();

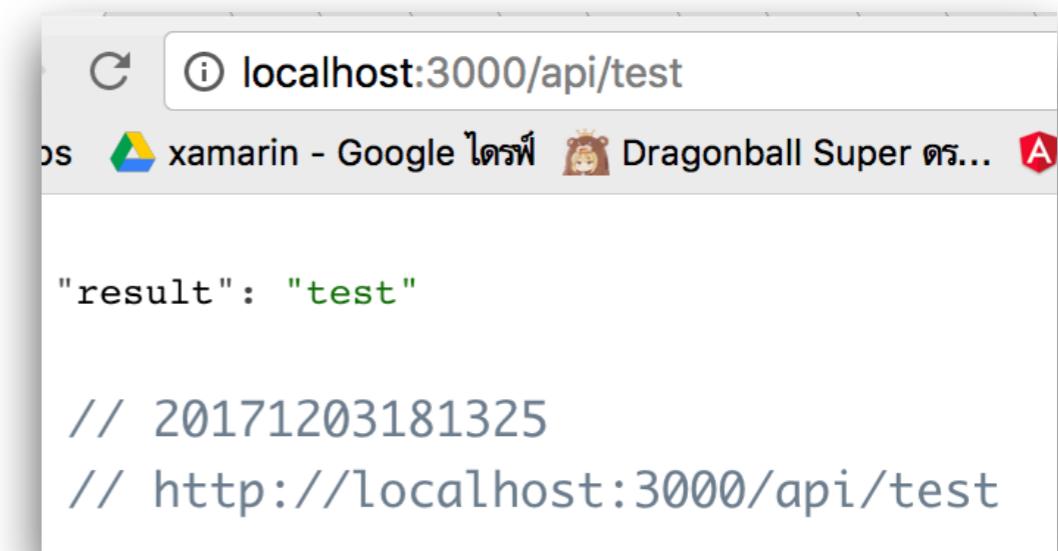
router.route('/routes').get(function(req, res){ ...
  router.get('/test', function(req, res){
    res.json({result: 'test'})
  });
}

module.exports = router;
```

```
var express = require('express');
var router = require('./ex9_export_route_module');

var app = express();
app.use('/api', router);

var server = app.listen(3000, function(){
  var host = server.address().address;
  var port = server.address().port;
  console.log("Listening at http://%s:%s", host, port);
});
```



# Express

# Response Functions

```
app.get('/download', function(req, res){  
  res.download(path.join(__dirname, 'download.jpeg'));  
});  
  
app.get('/send', function(req, res){  
  res.send('send');  
})  
  
app.get('/end', function(req, res){  
  res.setHeader('content-type', 'text/plain')  
  res.writeHead(res.statusCode);  
  res.write('Hi, ')  
  res.end("Lek"); // End the response process.  
})
```

```
app.get('/json', function(req, res){  
  res.json({result: 'ok'}); //Send a JSON response.  
})  
  
app.get('/redirect', function(req, res){  
  res.redirect('/json'); // Redirect a request.  
})  
  
app.get('/render', function(req, res){  
  res.render('/public/index.html'); // Redirect a request.  
})  
  
app.get('/sendFile', function(req, res){  
  res.sendFile(path.join(__dirname, '/index.html'))  
})  
  
app.get('/send', function(req, res){  
  // res.send(new Buffer('whoop'));  
  // res.send({ some: 'json' });  
  // res.send('<p>some html</p>');  
  res.status(404).send('Sorry, we cannot find that!');  
  // res.status(500).send({ error: 'something blew up' });  
});
```

# Express

# Response Functions

```
app.get('/download', function(req, res){  
  res.download(path.join(__dirname, 'download.jpeg'));  
});  
  
app.get('/send', function(req, res){  
  res.send('send');  
})  
  
app.get('/end', function(req, res){  
  res.setHeader('content-type', 'text/plain')  
  res.writeHead(res.statusCode);  
  res.write('Hi, ')  
  res.end("Lek"); // End the response process.  
})
```

```
app.get('/json', function(req, res){  
  res.json({result: 'ok'}); //Send a JSON response.  
})  
  
app.get('/redirect', function(req, res){  
  res.redirect('/json'); // Redirect a request.  
})  
  
app.get('/render', function(req, res){  
  res.render('/public/index.html'); // Redirect a request.  
})  
  
app.get('/sendFile', function(req, res){  
  res.sendFile(path.join(__dirname, '/index.html'))  
})  
  
app.get('/send', function(req, res){  
  // res.send(new Buffer('whoop'));  
  // res.send({ some: 'json' });  
  // res.send('<p>some html</p>');  
  res.status(404).send('Sorry, we cannot find that!');  
  // res.status(500).send({ error: 'something blew up' });  
});
```

# Body-Parser

- **Require('body-parser')**
- Parsing incoming request bodies in a middleware before your handlers, available under the req. body property
- Ex: req.body.<property>
- Work with Express.js

```
app.use(bodyParser.urlencoded({extended: false})  
  
app.use(bodyParser.json())
```

```
var express = require('express')  
var bodyParser = require('body-parser')  
  
var app = express()  
  
// parse application/x-www-form-urlencoded  
app.use(bodyParser.urlencoded({ extended: false }))  
  
// parse application/json  
app.use(bodyParser.json())  
  
app.use(function (req, res) {  
  res.setHeader('Content-Type', 'text/plain')  
  res.write('you posted:\n')  
  res.end(JSON.stringify(req.body, null, 2))  
})
```

# Node.js

## Built-in Module #1

|                      |                                                        |
|----------------------|--------------------------------------------------------|
| <u>assert</u>        | Provides a set of assertion tests                      |
| <u>buffer</u>        | To handle binary data                                  |
| <u>child_process</u> | To run a child process                                 |
| <u>cluster</u>       | To split a single Node process into multiple processes |
| <u>crypto</u>        | To handle OpenSSL cryptographic functions              |
| <u>dgram</u>         | Provides implementation of UDP datagram sockets        |
| <u>dns</u>           | To do DNS lookups and name resolution functions        |
| <u>domain</u>        | Deprecated. To handle unhandled errors                 |

# Node.js

## Built-in Module #2

|                 |                                                 |
|-----------------|-------------------------------------------------|
| <u>events</u>   | To handle events                                |
| <u>fs</u>       | To handle the file system                       |
| <u>http</u>     | To make Node.js act as an HTTP server           |
| <u>https</u>    | To make Node.js act as an HTTPS server.         |
| <u>net</u>      | To create servers and clients                   |
| <u>os</u>       | Provides information about the operation system |
| <u>path</u>     | To handle file paths                            |
| <u>punycode</u> | Deprecated. A character encoding scheme         |

# Node.js

## Built-in Module #3

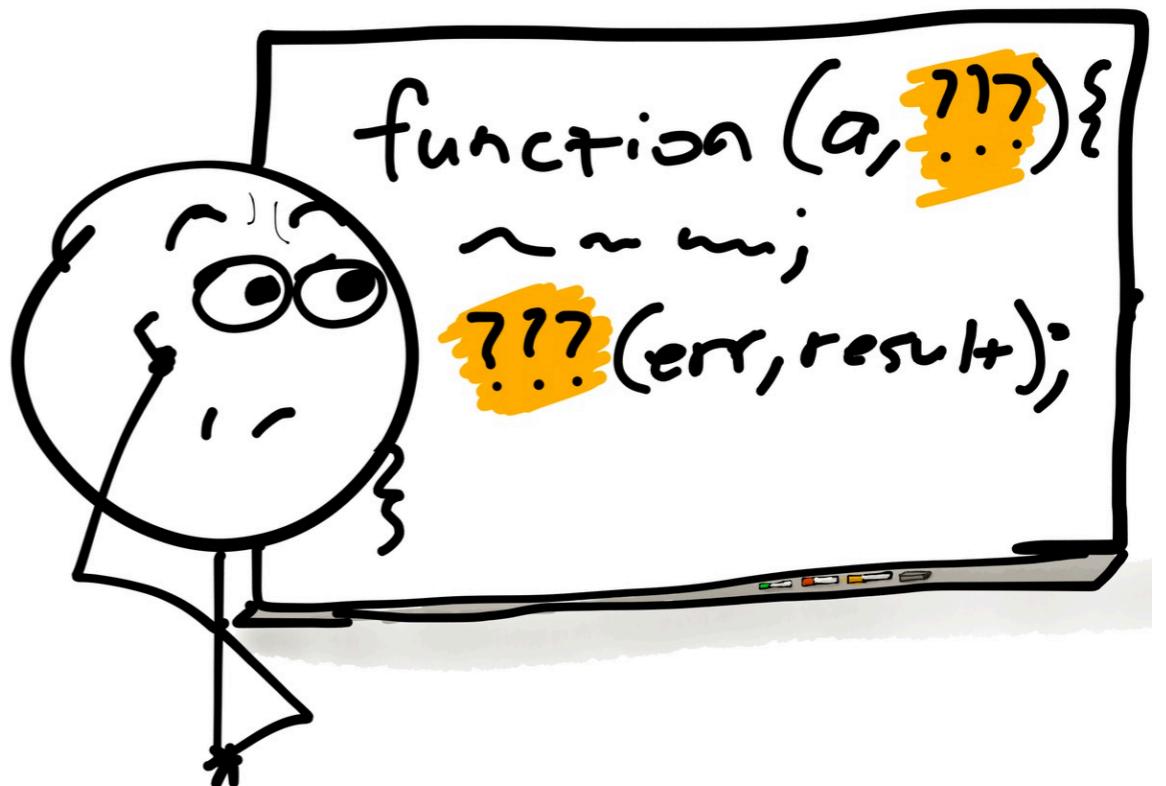
|                       |                                                            |
|-----------------------|------------------------------------------------------------|
| <u>querystring</u>    | To handle URL query strings                                |
| <u>readline</u>       | To handle readable streams one line at the time            |
| <u>stream</u>         | To handle streaming data                                   |
| <u>string_decoder</u> | To decode buffer objects into strings                      |
| <u>timers</u>         | To execute a function after a given number of milliseconds |
| <u>tls</u>            | To implement TLS and SSL protocols                         |
| <u>tty</u>            | Provides classes used by a text terminal                   |
| <u>url</u>            | To parse URL strings                                       |
| <u>util</u>           | To access utility functions                                |
| <u>v8</u>             |                                                            |
| <u>vm</u>             | To compile JavaScript code in a virtual machine            |
| <u>zlib</u>           | To compress or decompress files                            |

# Callback Patterns

**JSON Web Token**

# What is callback?

- A callback is a function that is to be executed after another function has finished, function are objects.
- Any function that is passed as an argument is called a callback function



# callback function in Javascript

```
// Patter 1# - standard
function callback1(msg){
    console.log(msg);
}
setTimeout(callback1.bind(null, "Callback #1"), 2000);

// Patter 2# - arrow function
var callback2 = (msg)=>{
    console.log(msg);
}
setTimeout(callback2.bind(null,"Callback #2"), 2000);

// Patter 3# - lamda exp
setTimeout(()=>{ callback2("Callback #3") }, 2000)
```

# callback function

## INCORRECT Error Handling

```
// Throw error in callback does not work

var callback = (msg) => {
    console.log(msg);
    if (msg === 'please error') {
        throw new Error("I am error exception")
    }
}

try {
    setTimeout(callback.bind(null, "please error"), 2000);
} catch (ex) {
    console.info(ex);
}
```

# callback function

## CORRECT Error Handling

```
var devide = (a, b, result, error) => {
    if (b == 0) {
        error("b cannot be zero")
    }else{ result(a/b) } }

devide(3,2, result=>{
    console.log(result);
}, error=>{
    console.log(error);
});

devide(3,0, result=>{
    console.log(result);
}, error=>{
    console.log(error);
});
```

# callback function

## CORRECT Error Handling

```
var devide = (a, b, result, error) => {
    if (b == 0) {
        error("b cannot be zero")
    }else{ result(a/b) } }

devide(3,2, result=>{
    console.log(result);
}, error=>{
    console.log(error);
});

devide(3,0, result=>{
    console.log(result);
}, error=>{
    console.log(error);
});
```

# Error-First & Callback-Last

## Design Pattern

```
nodeStyle(params, function (err, data) {  
  if (err) {  
    // error  
  } else {  
    // success  
  }  
};  
  
yourStyle(params, function (data) {  
  if (isError(data)) {  
    // error  
  } else {  
    // success  
  }  
};  
  
promiseStyle(params)  
  .then(function (data) {  
    // success  
  })  
  .catch(function (err) {  
    // error  
  });
```

The **error-first & callback-last** Node-style callbacks is what was originally used by Ryan Dahl and it is now pretty universal and expected for any asynchronous functions that take callbacks

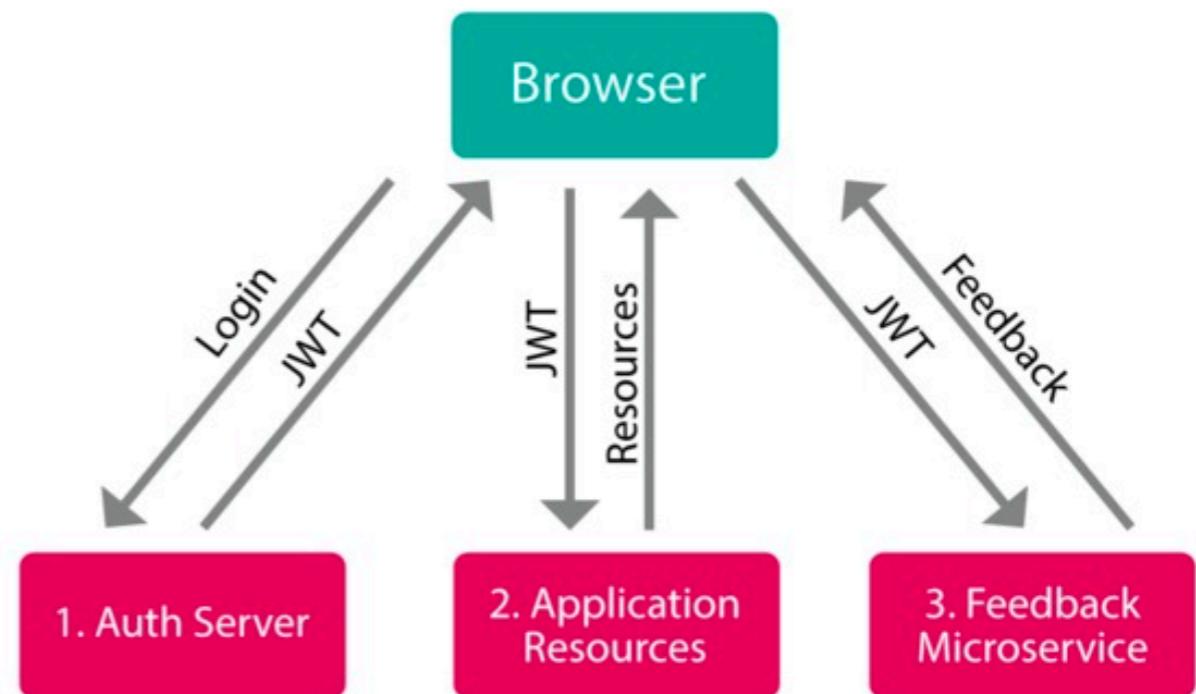
# JWT

## Secure Service Communication

**JSON Web Token**

# What is JWT?

JWT (JSON Web Token) let's us authenticate an API quickly and easily by serialize and deserialize JSON data such as email, account information through the token.



<https://scotch.io/tutorials/the-anatomy-of-a-json-web-token>

# JWT Structure

TOKEN



# JWT Serialization



Serialize to

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzY290Y2guaw8iLCJleHAiOjEzMjA4MTkzODAsIm5hbWUiOiJDaHJpcyBTZXZpbGxlamEiLCJhZG1pbmI6dHJ1ZX0.03f329983b86f7d9a9f5fef85305880101d5e302afafa20154d094b229f75773

# JWT De-Serialization

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzY290Y2guaw8iLCJleHAiOjEzMzQ4MTkzODAsIm5hbWUiOiJDaHJpcyBTZXZpbGxlamEiLCJhZG1pbmci6dHJ1ZX0.03f329983b86f7d9a9f5fef85305880101d5e302afafa20154d094b229f75773

De-Serialize to



# JWT Example

## Register - Generate Token

The screenshot shows a Postman collection named "register". The request is a POST to `localhost:3000/api/register`. The "Body" tab is selected, showing raw JSON input:

```
{"username": "admin", "email": "chaiyasit.t@gmail.com", "password": "1234"}
```

The response status is 200 OK, with a time of 70 ms and a size of 408 B. The response body is a JSON object:

```
{ "auth": true, "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9 .eyJpZCI6IjVhYmNjZWEwNzc0NDZkMGYwZDA4ZjdkMiIsImhdCI6MTUyMjMyMzEwNCwiZXhwIjoxNTIyNDA5NTA0fQ .i3MEWhG0IFbqRCF2XgNk6rF5XV3ppyfD07z4i-13r8Q" }
```

# JWT Example

## Login - Generate Token

The screenshot shows a POST request to `localhost:3000/api/login`. The `Body` tab is selected, showing a JSON payload:

```
1 {"email": "chaiyasit.t@gmail.com", "password": "1234"}
```

The response status is `200 OK`, time: `95 ms`, size: `408 B`. The response body contains a JSON object:

```
1 {
2   "auth": true,
3   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
4     .eyJpZCI6IjVhYmNjZWEwNzc0NDZkMGYwZDA4ZjdkMiIsImhdCI6MTUyMjMyMzM0OCwiZXhwIjoxNTIyNDA5NzQ4fQ.GQMPUcEZ3kMIZ0FZ
5     -7iW5RzLzoXuZ_0G2Nd_NEYDLjo"
6 }
```

# JWT Example

## Profile - Verify Token

The screenshot shows a Postman collection named "profile". A GET request is made to "localhost:3000/api/profile". The "Headers" tab is selected, showing an "x-access-token" header with the value: eyJhbGciOiJIUzI1NilsInR5cCI6IkpXVCJ9.eyJpZCI6IjVhOWZjOTc2MTA4OGFjZjY3ZTZjNDVmYSIsImIhdCI6MTUyMjMxOTE1MywiZXhwIjoxNTIyNDA1NTUzfQ.6IXzL88NulqouxeVb9xUVR9eS1qxpRlmosMlaAPb1GY. The response status is 200 OK, time is 15 ms, and size is 286 B. The response body is a JSON object:

```
1 {  
2   "_id": "5a9fc9761088acf67e6c45fa",  
3   "email": "chaiyasit.t@gmail.com",  
4   "__v": 0  
5 }
```

# JWT Ex: (Overview)

- Configure Core Lib.
- Configure Database
- Configure JWT
- Create Verify Token Function
- Define authen. Function
  1. Register
  2. Login
  3. Logout
  4. Profile

```
// Core Library
var express = require('express');
var bodyParser = require('body-parser');
var app = express();
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());

// Database
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/jwtdemo');

// JWT Configure
var jwt = require('jsonwebtoken');
var bcrypt = require('bcryptjs');
var User = require('./user');

var secret_key = 'codemobiles';
+ var VerifyToken = function (req, res, next) {
  }

// Register
+ app.post('/api/register', function (req, res) {
  });

// Login
+ app.post('/api/login', function(req, res) {
  });

// Logout
+ app.get('/api/logout', function(req, res) {
  });

// Profile (Verify Token)
+ app.get('/api/profile', VerifyToken, function(req, res, next) {
  });

module.exports = app;
```

# JWT Ex: 1# (Import)

```
// Core Library
var express = require('express');
var bodyParser = require('body-parser');
var app = express();
app.use(bodyParser.urlencoded({ extended: false }));
app.use(bodyParser.json());

// Database
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/jwtdemo');

// JWT Configure
var jwt = require('jsonwebtoken');
var bcrypt = require('bcryptjs');
var User = require('./user');
```

# JWT Ex: 2# (Verify Token)

```
var secret_key = 'codemobiles';

var VerifyToken = function (req, res, next) {
  var token = req.headers['x-access-token'];
  if (!token)
    return res.status(403).send({ auth: false, message: 'No token provided.' });
  jwt.verify(token, secret_key, function(err, decoded) {
    if (err)
      return res.status(500).send({ auth: false, message: 'Failed to authenticate token.' });
    req.userId = decoded.id;
    next();
  });
}
```

# JWT Example 3# (Register)

## Serialize Token

```
// Register
app.post('/api/register', function (req, res) {

  User.create({
    name: req.body.name,
    email: req.body.email,
    password: bcrypt.hashSync(req.body.password, 8)
  },
  function (err, user) {
    if (err) return res.status(500).send("There was a problem registering the user.")

    // create a token
    var token = jwt.sign({ id: user._id }, secret_key, {
      expiresIn: 86400 // expires in 24 hours
    });

    res.status(200).send({ auth: true, token: token });
  });
});
```

# JWT Example 4# (Login)

## Serialize Token

```
// Login
app.post('/api/login', function(req, res) {
  User.findOne({ email: req.body.email }, function (err, user) {
    if (err) return res.status(500).send('Error on the server.' + JSON.stringify(err));
    if (!user) return res.status(404).send('No user found.');

    var passwordIsValid = bcrypt.compareSync(req.body.password, user.password);
    if (!passwordIsValid) return res.status(401).send({ auth: false, token: null });

    var token = jwt.sign({ id: user._id }, secret_key, {
      expiresIn: 86400 // expires in 24 hours
    });

    res.status(200).send({ auth: true, token: token });
  });
});
```

# JWT Ex: 5# (Profile)

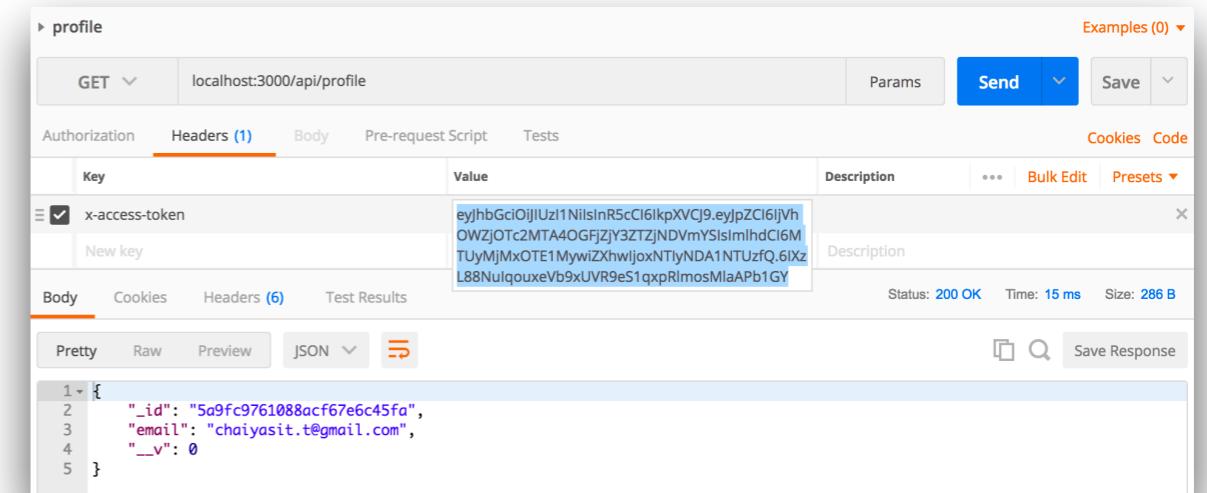
## De-Serialize Token

```
// Profile (Verify Token)
app.get('/api/profile', VerifyToken, function(req, res, next) {
  User.findById(req.userId, { password: 0 }, function (err, user) {
    if (err) return res.status(500).send("There was a problem finding the user.");
    if (!user) return res.status(404).send("No user found.");

    res.status(200).send(user);
  });
});

var secret_key = 'codemobiles';

var VerifyToken = function (req, res, next) {
  var token = req.headers['x-access-token'];
  if (!token)
    return res.status(403).send({ auth: false, message: 'No token provided.' });
  jwt.verify(token, secret_key, function(err, decoded) {
    if (err)
      return res.status(500).send({ auth: false, message: 'Failed to authenticate token.' });
    req.userId = decoded.id;
    next();
  });
}
```



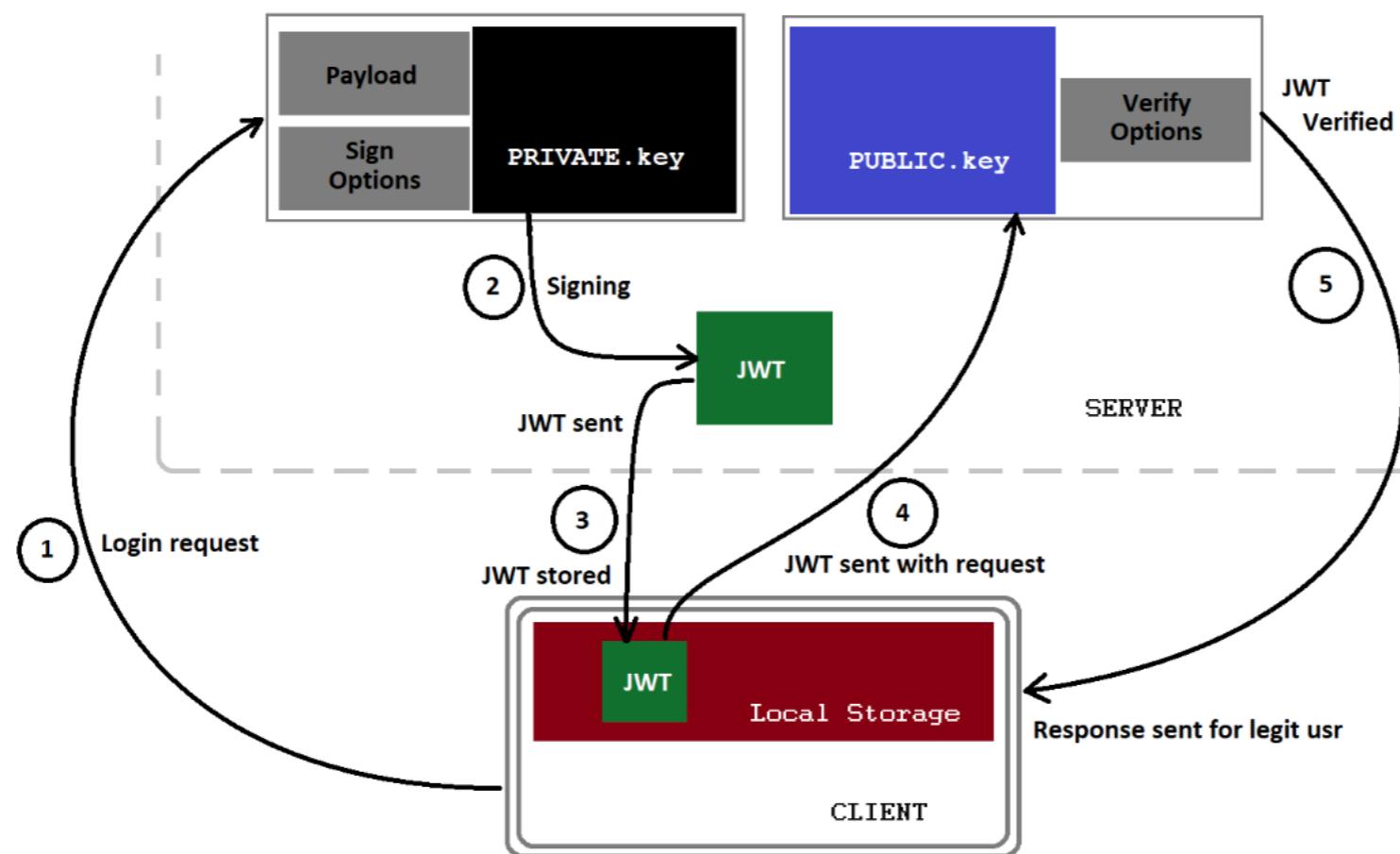
# JWT

# Private & Public Keys

# JWT

## How to secure secret keys?

**JSON Web Token (JWT)—The right way of implementing, with Node.js**



# JWT

# Generate Keys

<http://travistidwell.com/jsencrypt/demo/>

### Online RSA Key Generator

Key Size    1024 bit ▾

Generated in 136 ms

Async

**Private Key**

```
-----BEGIN RSA PRIVATE KEY-----
MIICXAIBAAKBgQCCe+hniLCzpiqmoQhW0UNs78mBDGpJf3CYtgS3N142DTSKY
PgD
vRxqy97MT5yCFFOAmNdZvSEaBBznitlI0B5RlaS5C8vy7FGE/Fy2jLdd40cPZciJ
+weU2p65jAY4ICtGPfqqSWwmj/WZAisVuYtOQmgpsHMCeNcqoJ+1cYZA6QIDA
QAB
AoGAUGgMiahYwwuVYsL+wYklJuav3QoFwx6FRctDT4IRySSGO1N0lp5pXH5F7
2
oc3muur0YW9Dc6WvLhuEbhq2YwTGp7Dv42Qpglfh270q8pRzoW/ehqpAdwtkoP
gx
9DFPrfUjmXZ0nYFYrlk5eCozCsqYVRnE7vbGOyPFrUaKFECQQDBv5j7Xp2J1A66
aRDDD9Sly34T2wAJ06zX1ONQkHgJx05ZWVBDCv8nAEZgWdTuhiyYJDQ+iLeQy
YiI
OxwhHS61AkEArGieRQfZwVCO+hWNMuiyY7eeGkP2Ee2SHlb421/Dp/62t5VtNED
c
```

**Public Key**

```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCCe+hniLCzpiqmoQhW0UN
Ns78mB
DGpJf3CYtgS3N142DTSkYPqDvRxqy97MT5yCFFOAmNdZvSEaBBznitlI0B5RlaS
5
C8vy7FGE/Fy2jLdd40cPZciJ+weU2p65jAY4ICtGPfqqSWwmj/WZAisVuYtOQmgp
sHMCeNcqoJ+1cYZA6QIDAQAB
-----END PUBLIC KEY-----
```

# JWT

# Generate Keys

<http://travistidwell.com/jsencrypt/demo/>

The screenshot shows a Mac OS X desktop environment. In the foreground, a terminal window displays a long string of RSA private key characters, starting with "-----BEGIN RSA PRIVATE KEY-----" and ending with "-----END RSA PRIVATE KEY-----". Below the terminal, a Visual Studio Code (VS Code) instance is open. The Explorer sidebar on the left shows several files and folders, including 'private.key' and 'public.key' under a 'node' folder, and various files like 'environment.ts', 'menu.component.html', and 'styles.css' in the main workspace. The status bar at the bottom of the VS Code window indicates the date and time: 'Date: 2018-07-31T04:37:43.169Z'. The terminal also shows webpack compilation logs, including successful chunking and compilation.

# JWT

# Generate Keys

<http://travistidwell.com/jsencrypt/demo/>

The screenshot shows a Mac OS X desktop environment. In the foreground, a terminal window displays a long string of RSA private key characters, starting with "-----BEGIN RSA PRIVATE KEY-----" and ending with "-----END RSA PRIVATE KEY-----". Below the terminal, a VS Code editor window is open, showing the same file 'private.key'. The file content is identical to what is shown in the terminal. The VS Code interface includes a sidebar with file navigation, a top bar with tabs for 'environment.ts', 'menu.component.html', 'readme.txt', 'menu.component.css', and 'styles.css', and a bottom status bar showing build logs for Webpack.

```
-----BEGIN RSA PRIVATE KEY-----  
MIICXAIBAAKBgQCJHDZtSAXqdn3yhkVSEMjabnwpZyjwzwKqZI1Dhi0nrWbFv305  
mBj6bH52qLsGalaxX+ImT3FRjuS5YGYIxzbQAB/L0kDiYJs4yDnbrIPHlmKAKmb  
w06z7Vkv1Muy2ocnZiuyL9DUmJF6ECceoKqH5sEzFmGnQhVrvvJv2vCAPwIDAQAB  
AoGAVW1HmAU08mXLiU0RadVdX4NRRAocol4brPhtD2if3mxu+nSb80NZJIupT3x  
8UhWJYPJQvdicl9msP2cAg2JamyvqRu4bBBuNrXKIT08SmZ8AoBnmdHF9njJBIh3  
YPTKg+mf8d0gHxuwhsMCvQZr9t+V0s5VN0Hs1aUMthAYXSECQD2bxmGCnC3lKLL  
SanpRa7kC2QsawhNA8XAugl+yticT5h16fQ73xNDGSZ6P/vBxDI5U+80epWxfoEDv  
ryq0gSMvAkEAjm65dhbVtxR4tHLpiwT7PbT94kZZQ4gSg3gaCENLCERmc1QimdgQ  
nLxXPLalnI4Pqm81u7/LxBwD5qYr3Nv8QJBALu22ylYUI32ZpKb1RqE4viAmNig  
2xzXD86Dp3H2jqhwsN213y3qt0El+0tU1dA3MUAY4iFC/u9EgdJRiuUeLS8CQBYt  
valawuqqQG9w2Vbx bqHcNhBu/eVbIe7g28Tfnazq6D0bI8wq7eNKHJdy pXUsVdon  
0QnEqtmzP//eBdqEMjECQDEXjibfFApXVRPyXN8qHFz4IjbKsr7kY6NnakvuJVww  
1m61IsGnqWiHE7cSFnrrMvViF0cMlSglwDZyH8h0qiM=  
-----END RSA PRIVATE KEY-----
```

# JWT

# JWT.js

```

const fs = require('fs');
const path = require('path');
const jwt = require('jsonwebtoken');
var publicKey = fs.readFileSync(path.join(__dirname + '/public.key'), 'utf8');
var privateKey = fs.readFileSync(path.join(__dirname + '/private.key'), 'utf8');

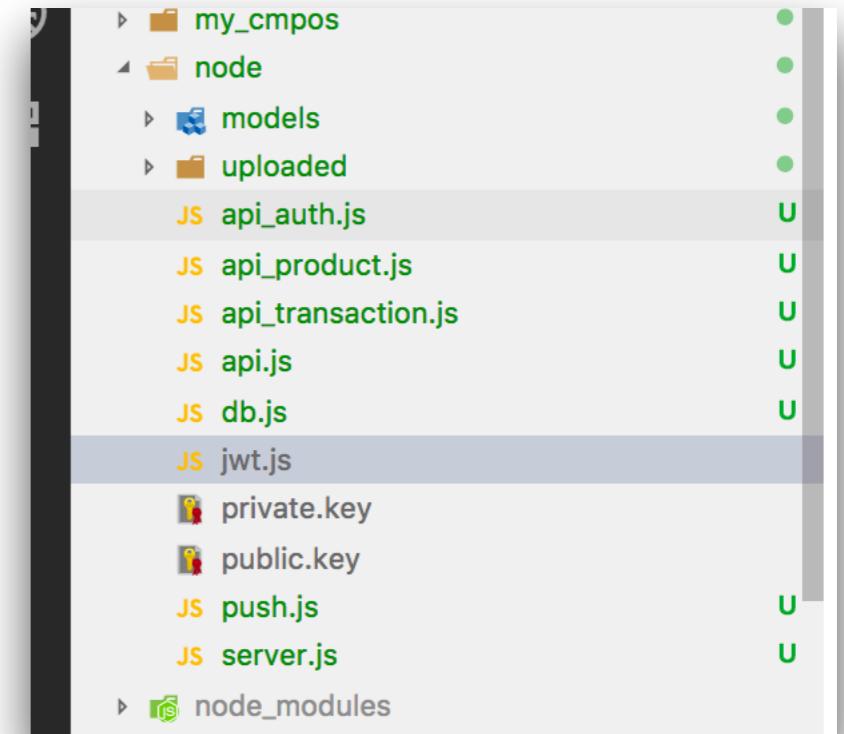
var i = 'Mysoft corp'; // Issuer (Software organization who issues the token)
var s = 'some@user.com'; // Subject (intended user of the token)
var a = 'http://mysoftcorp.in'; // Audience (Domain within which this token will live and function)

module.exports = {
  sign: (payload) => {
    // Token signing options
    var signOptions = {
      issuer: i,
      subject: s,
      audience: a,
      expiresIn: "30d", // 30 days validity
      algorithm: "RS256"
    };
    return jwt.sign(payload, privateKey, signOptions);
  },
  verify: (req, res, next) => {
    var token = req.headers['x-access-token'];
    if (!token)
      return res.status(403).send({ auth: false, message: 'No token provided.' });

    var verifyOptions = {
      issuer: i,
      subject: s,
      audience: a,
      expiresIn: "12h",
      algorithm: ["RS256"]
    };

    jwt.verify(token, publicKey, verifyOptions, function(err, decoded) {
      if (err)
        return res.status(500).send({ auth: false, message: 'Failed to authenticate token.' });
      // if everything good, save to request for use in other routes
      req.userId = decoded.id;
      next();
    });
  },
  decode: (token) => {
    return jwt.decode(token, { complete: true });
    //returns null if token is invalid
  }
}

```



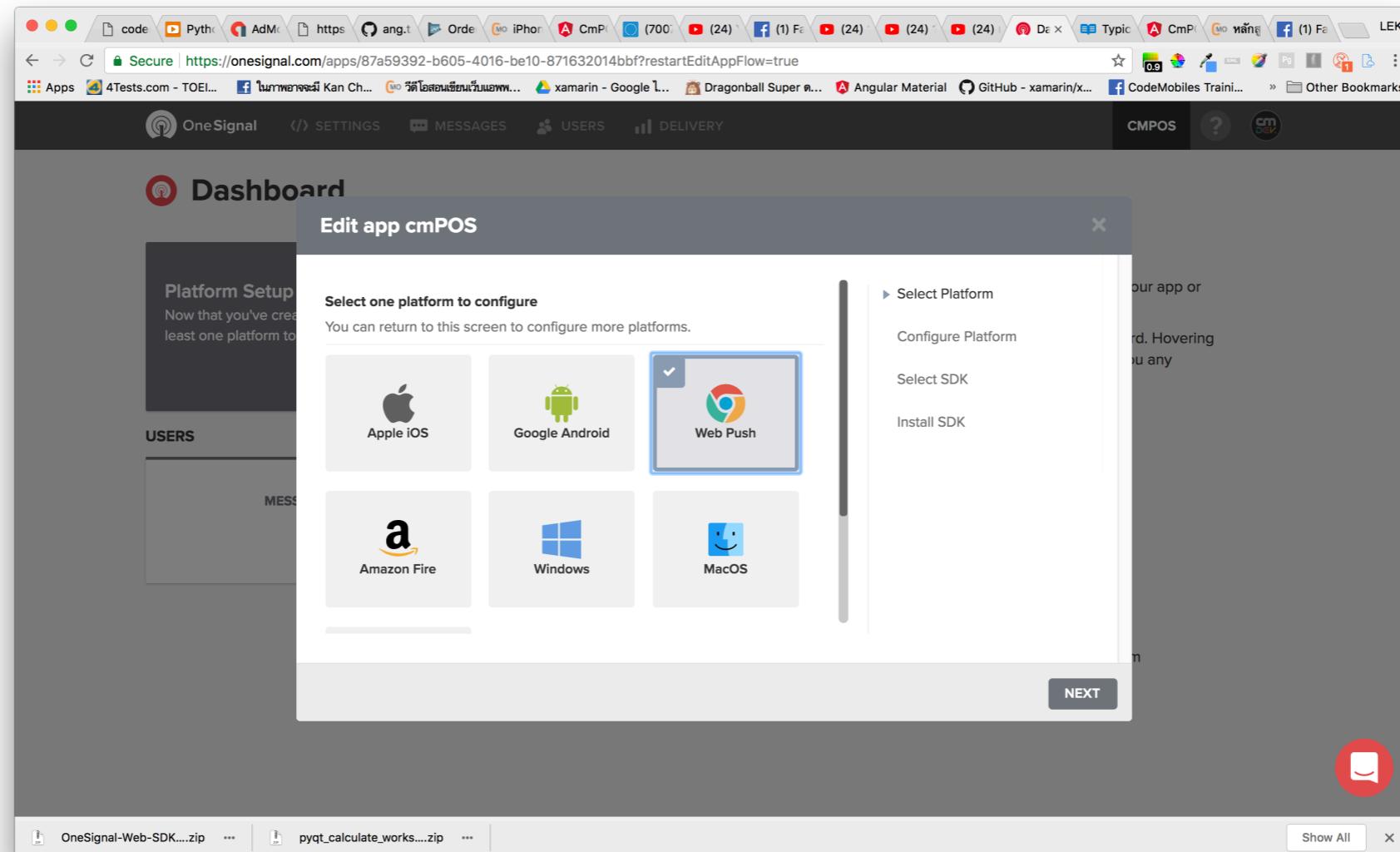
# Push Notification

# Setup Push Notification

- Register One Signal Account
- Add Project
- Configure Project
- Install SDK (Just copy snippet code)
- Test



# Setup Push Notification



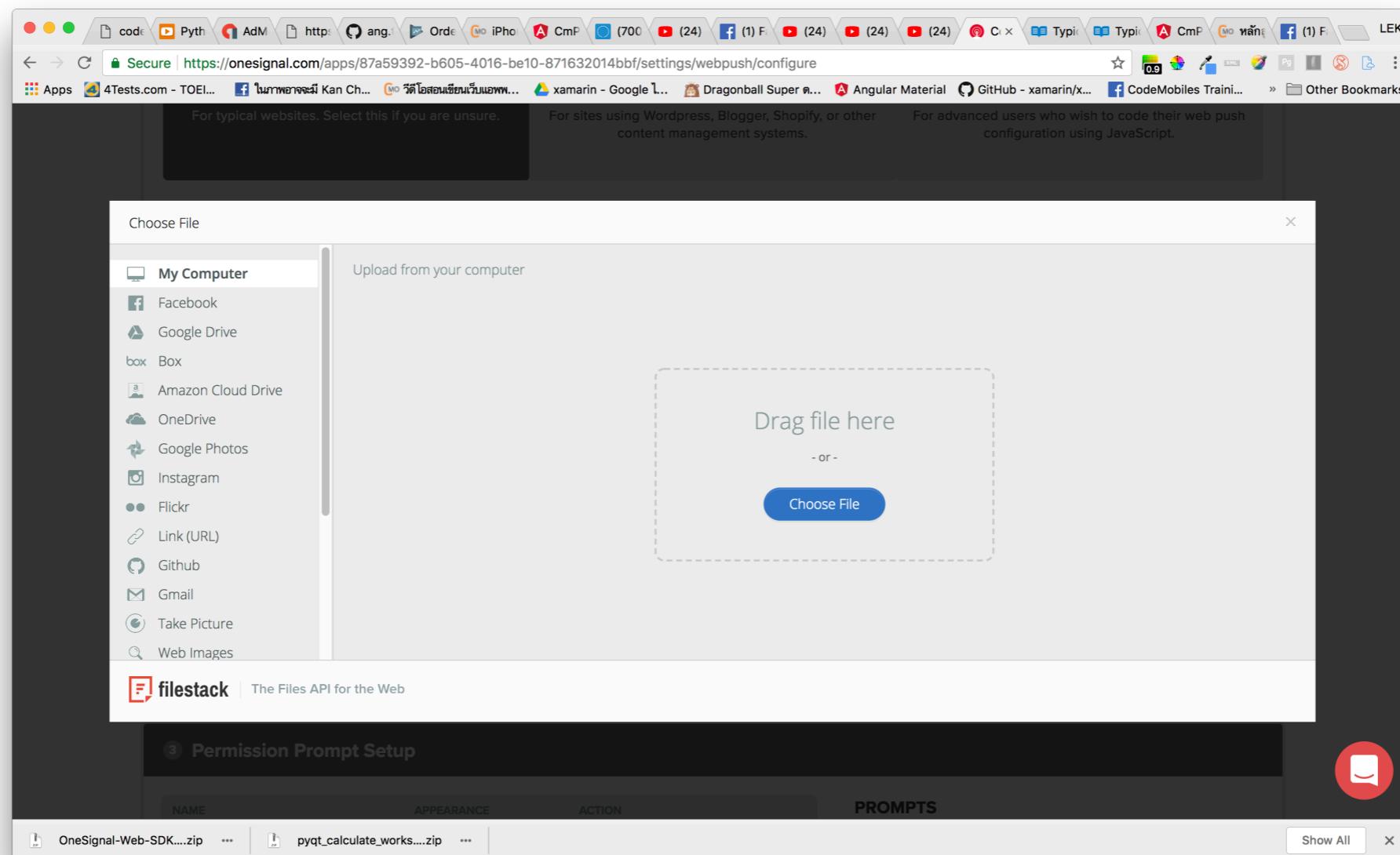
# Setup Push Notification

The screenshot shows the 'Configure Web Push' page in the OneSignal dashboard. At the top, there's a navigation bar with tabs for OneSignal, SETTINGS, MESSAGES, USERS, and DELIVERY. Below the navigation is a sub-navigation for 'SETTINGS' with 'MESSAGES' selected. The main content area has two sections: '1 Choose Integration' and '2 Site Setup'.  
**1 Choose Integration:** This section contains three options:

- Typical Site:** Represented by a computer monitor icon. Description: 'For typical websites. Select this if you are unsure.'
- Wordpress Plugin or Website Builder:** Represented by a computer monitor icon with WordPress, Shopify, and Squarespace logos. Description: 'For sites using Wordpress, Blogger, Shopify, or other content management systems.'
- Custom Code:** Represented by a code editor icon. Description: 'For advanced users who wish to code their web push configuration using JavaScript.'

  
**2 Site Setup:** This section contains fields for 'SITE NAME\*' (My Website) and 'SITE URL\*' (https://site.com or https://www.site.com (check www prefix)). To the right, there's a 'SITE SETUP' description: 'Enter your site name, URL, and icon URL to set up Web Push on your site.' Below this is a 'READ OUR DOCUMENTATION' link and a red circular button with a white icon. At the bottom of the page are download links for 'OneSignal-Web-SDK....zip' and 'pyqt\_calculate\_works....zip'.

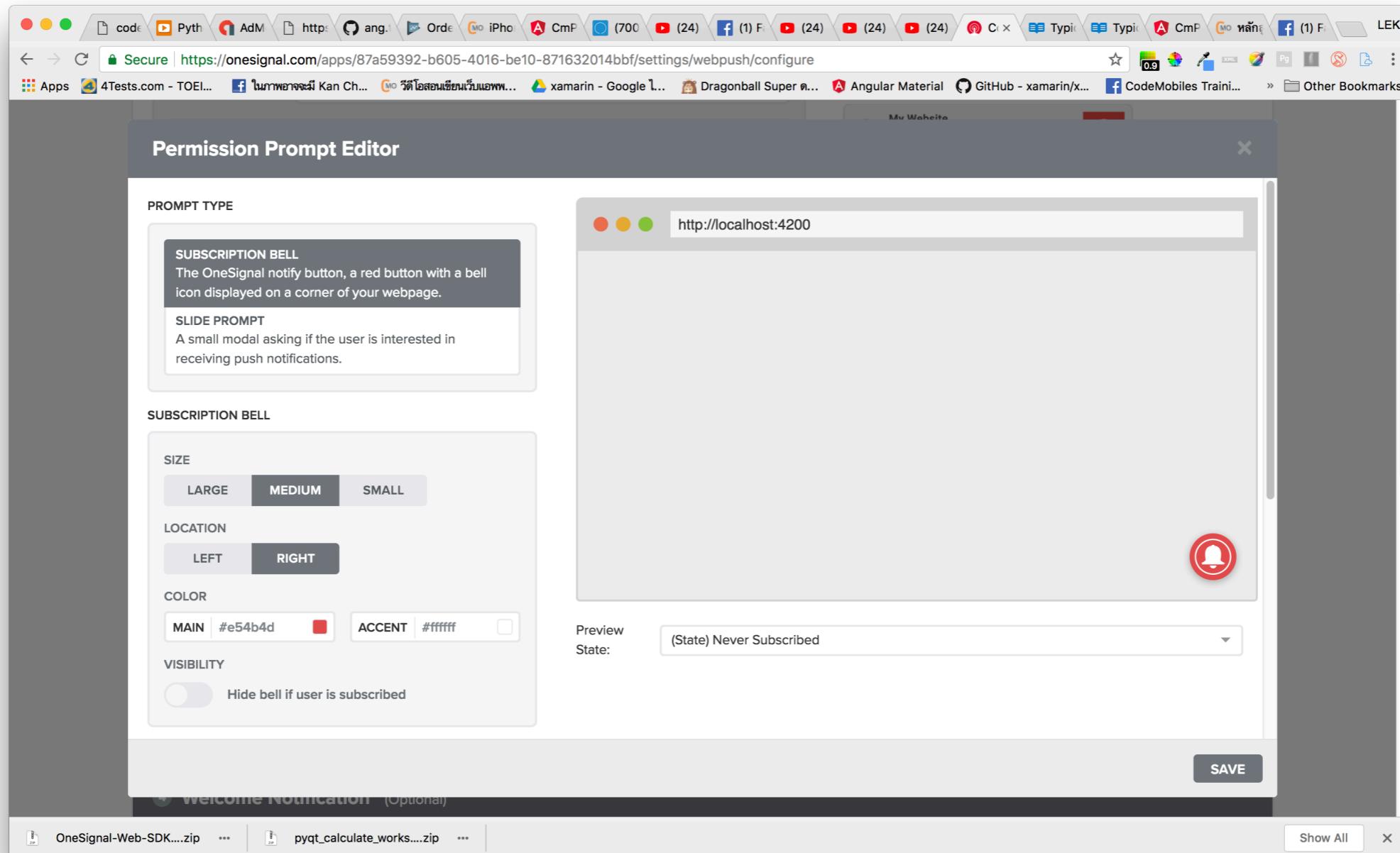
# Setup Push Notification



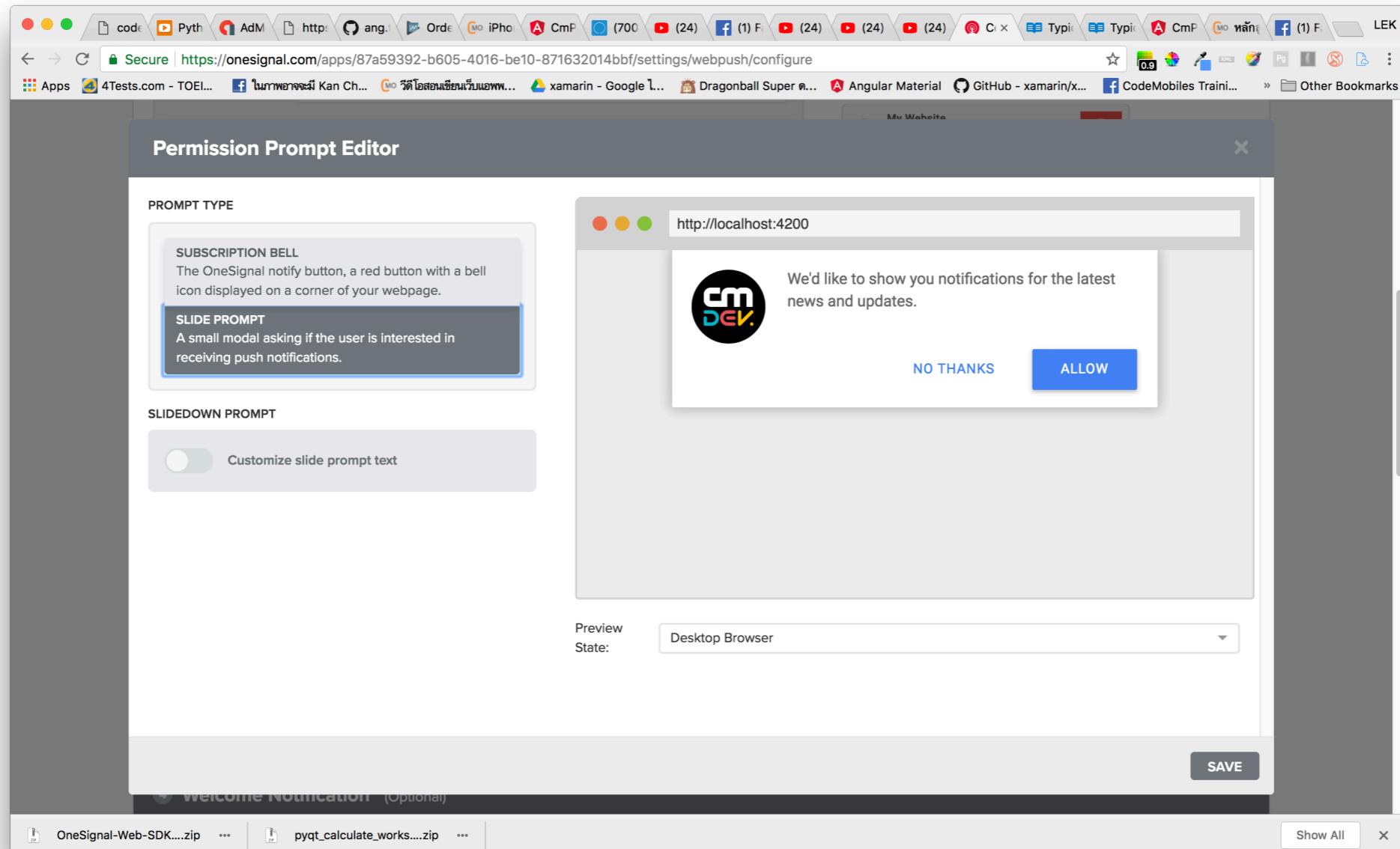
# Setup Push Notification

The screenshot shows the OneSignal web push configuration interface. At the top, there are three options for selecting a configuration type: "For typical websites. Select this if you are unsure.", "For sites using Wordpress, Blogger, Shopify, or other content management systems.", and "For advanced users who wish to code their web push configuration using JavaScript." The main area is titled "2 Site Setup". It contains fields for "SITE NAME\*" (cmpos), "SITE URL\*" (http://localhost:4200), "DEFAULT ICON URL" (with a file input field and "+ UPLOAD" button), and a toggle switch for "My site is not fully HTTPS". Below these is a "CHOOSE A LABEL\*" field with the value "yourlabel .OS.TC". To the right of the "Site Setup" section is a "DEFAULT NOTIFICATION ICON URL" panel with instructions about icon dimensions and hosting. At the bottom is a "3 Permission Prompt Setup" section with a table showing two rows: "OneSignal-Web-SDK....zip" and "pyqt\_calculate\_works....zip". The table has columns for "NAME", "APPEARANCE", "ACTION", and "PROMPTS". A red circular icon with a white letter "S" is located to the right of the "PROMPTS" column.

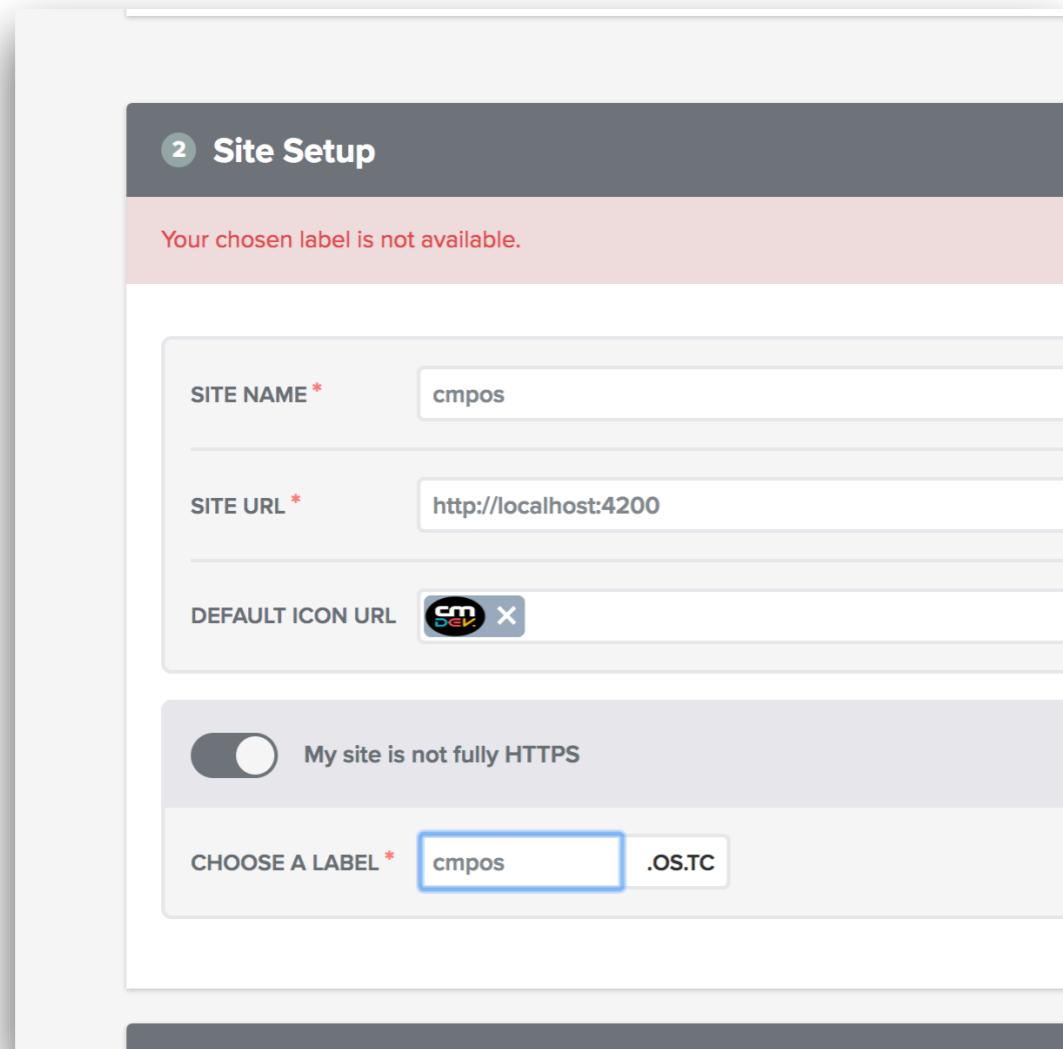
# Setup Push Notification



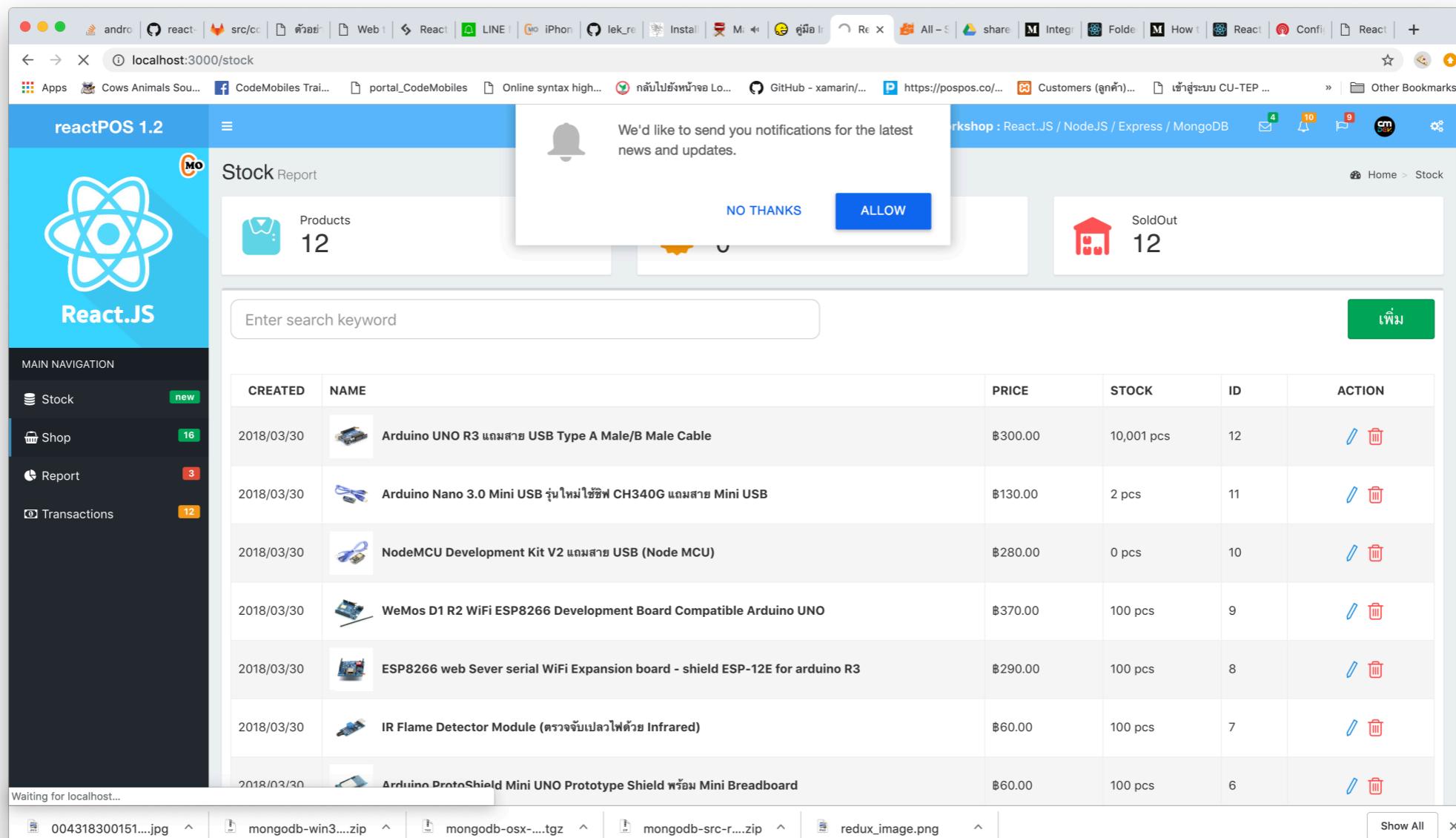
# Setup Push Notification



# Setup Push Notification



# Setup Push Notification



# Setup Push Notification Server in NodeJS

```
var OneSignal = require('onesignal-node');

// first we need to create a client
var myClient = new OneSignal.Client({
  userAuthKey: 'MzE0YTMzOTAtNGFiZC00OGU5LTlkNmItMDI1NjNiMGM0OWY2',
  app: { appAuthKey: 'ZmMzNTQzMIDtM2JZS00NWl3LWE1OGMtNWVhYmNiMzFjMjE1', appId: '51a68888-4ee1-4fe1-aadb-d6b2af6ec131' }
});

// we need to create a notification to send
var firstNotification = new OneSignal.Notification({
  contents: {
    en: "CodeMobiles.com",
    th: "ໂຄ້ດ ໂມບາຍສ໌ ຈຳກັດ"
  }
});

// set target users
firstNotification.setIncludedSegments(['All']);
firstNotification.setExcludedSegments(['Inactive Users']);

// set notification parameters
firstNotification.setParameter('data', {"abc": "123", "foo": "bar"});
//firstNotification.setParameter('send_after', 'Thu Sep 24 2015 14:00:00 GMT-0700 (PDT)');

// send this notification to All Users except Inactive ones
myClient.sendNotification(firstNotification, function (err, httpResponse,data) {
  if (err) {
    console.log('Something went wrong... ');
  } else {
    console.log(data, httpResponse.statusCode);
  }
});
```

# PM2



P(rocess) M(anager) 2

 bitHound 85    downloads 679k/month    build passing

PM2 is a production process manager for Node.js applications with a built-in load balancer. It allows you to keep applications alive forever, to reload them without downtime and to facilitate common system admin tasks.

Starting an application in production mode is as easy as:

```
$ pm2 start app.js
```

# PM2

## (<https://github.com/Unitech/pm2>)

- Installation

- sudo su
- npm install pm2 -g
- pm2 start server.js
- pm2 stop (app | name | id)
- pm2 update

```
2 daemon with pm2_home=/var/root/.pm2
fully daemonized
training/Angular/all_templates/angularLTE2.4m/server in fork_mod

mode pid status restart uptime cpu mem us
fork 38759 online 0 0s 0% 9.7 MB r

d|name>` to get more details about an app
server
tion restartProcessId on app [server](ids: 0)
,
cessfully started

mode pid status restart uptime cpu mem us
fork 38774 online 1 0s 0% 9.9 MB r

d|name>` to get more details about an app
s

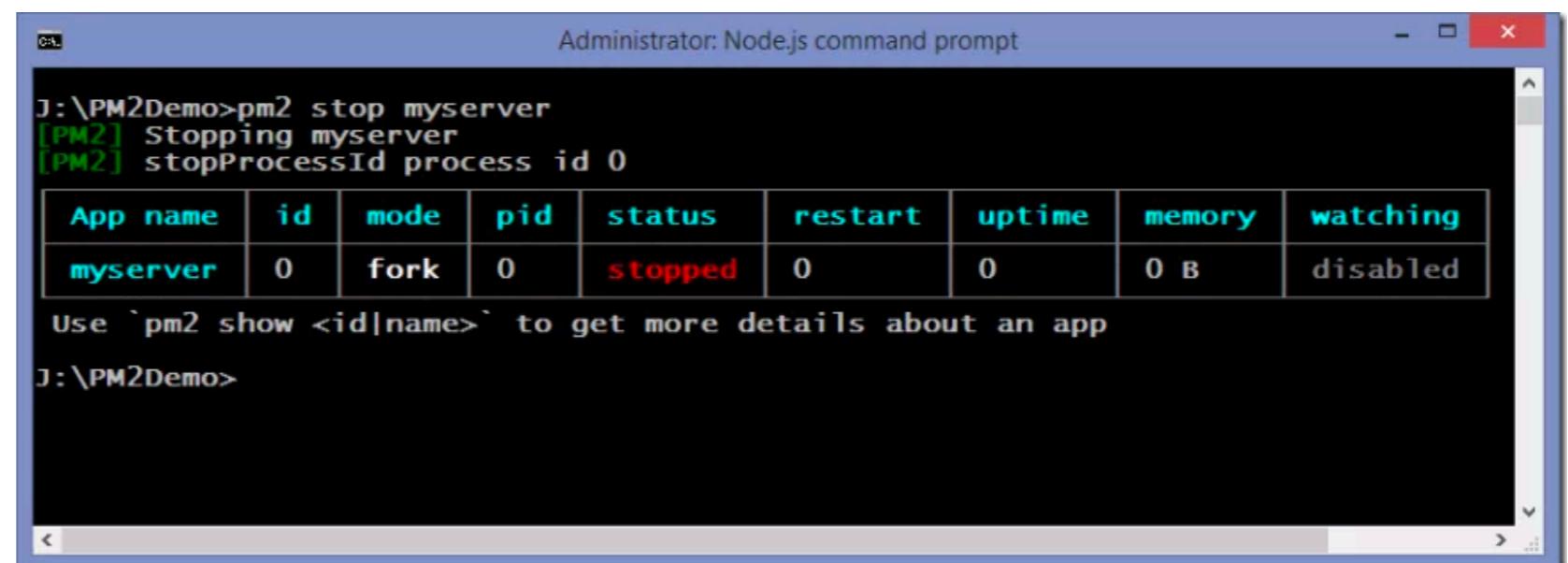
mode pid status restart uptime cpu mem us
fork 38774 online 1 4s 0% 53.9 MB r

d|name>` to get more details about an app
```

# PM2

## Common Commands

- npx pm2 start server.js == npx nodemon server.js
- npx pm2 start server.js
- npx pm2 status
- npx pm2 stop server
- npx pm2 stop 0
- npx pm2 start server
- npx pm2 (start/stop/restart) all
- npx pm2 log
- npx pm2 log server/0
- npx pm2 start server/0 --watch
- npx pm2 delete server/0



```
J:\PM2Demo>pm2 stop myserver
[PM2] Stopping myserver
[PM2] stopProcessId process id 0
App name    id   mode   pid   status   restart   uptime   memory   watching
myserver     0    fork    0    stopped   0        0        0 B      disabled
Use `pm2 show <id|name>` to get more details about an app
J:\PM2Demo>
```

# PM2

## CheatSheet

- PM2 or Process Manager 2, is an Open Source, production Node.js process manager helping Developers and Devops manage Node.js applications in production environment.
- key features of PM2 are automatic application load balancing, declarative application configuration, deployment system and monitoring
- main cli : <http://pm2.keymetrics.io/docs/usage/quick-start/>

```
# Fork mode
pm2 start app.js --name my-api # Name process
# Cluster mode
pm2 start app.js -i 0           # Will start 1 process
pm2 start app.js -i max         # Same as above
# Listing
pm2 list                         # Display all processes
pm2 jlist                         # Print process json
pm2 prettylist                     # Print process list
pm2 describe 0                   # Display all info about process
pm2 monit                         # Monitor all processes
# Logs
pm2 logs [--raw]                 # Display all process logs
pm2 flush                         # Empty all log files
```

# PM2

## NodeJS

```
CodeMobiles-PLEK-15:nodejs-jwt codemobiles$ pm2 status
[PM2] Spawning PM2 daemon with pm2_home=/Users/codemobiles/.pm2
[PM2] PM2 Successfully daemonized
```

| App name | id | mode | pid | status | restart | uptime | cpu | mem | user | watching |
|----------|----|------|-----|--------|---------|--------|-----|-----|------|----------|
|----------|----|------|-----|--------|---------|--------|-----|-----|------|----------|

```
Use `pm2 show <id|name>` to get more details about an app
CodeMobiles-PLEK-15:nodejs-jwt codemobiles$ █
```

```
CodeMobiles-PLEK-15:nodejs-jwt codemobiles$ pm2 start server.js
[PM2] Starting /Users/codemobiles/Downloads/jwt/nodejs-jwt/server.js in fork_mode (1 instance)
[PM2] Done.
```

| App name | id | mode | pid  | status | restart | uptime | cpu | mem     | user        | watching |
|----------|----|------|------|--------|---------|--------|-----|---------|-------------|----------|
| server   | 0  | fork | 4191 | online | 0       | 0s     | 0%  | 10.0 MB | codemobiles | disabled |

```
CodeMobiles-PLEK-15:nodejs-jwt codemobiles$ pm2 start server
[PM2] Applying action restartProcessId on app [server](ids: 0)
[PM2] [server](0) ✓
[PM2] Process successfully started
```

| App name | id | mode | pid  | status | restart | uptime | cpu | mem    | user        | watching |
|----------|----|------|------|--------|---------|--------|-----|--------|-------------|----------|
| server   | 0  | fork | 4305 | online | 1       | 0s     | 0%  | 9.1 MB | codemobiles | disabled |

```
CodeMobiles-PLEK-15:nodejs-jwt codemobiles$ pm2 delete server
[PM2] Applying action deleteProcessId on app [server](ids: 0)
[PM2] [server](0) ✓
```

| App name | id | mode | pid | status | restart | uptime | cpu | mem | user | watching |
|----------|----|------|-----|--------|---------|--------|-----|-----|------|----------|
|          |    |      |     |        |         |        |     |     |      |          |

- **Ex:**

- **pm2 status**

- **pm2 start server.js**

- **pm2 start server**

- **pm2 delete server**

- **pm2 delete 0 (0=id)**

# PM2

## MongoDB

```
CodeMobiles-PLEK-15:nodejs-jwt codemobiles$ pm2 start mongod
[PM2] Starting /usr/local/bin/mongod in fork_mode (1 instance)
[PM2] Done.
```

| App name      | id  | mode      | pid    | status         | restart | uptime | cpu   | mem        | user                    | watching          |
|---------------|-----|-----------|--------|----------------|---------|--------|-------|------------|-------------------------|-------------------|
| mongod server | 1 0 | fork fork | 4641 0 | online stopped | 0 0     | 0s 0   | 0% 0% | 1.5 MB 0 B | codemobiles codemobiles | disabled disabled |

```
CodeMobiles-PLEK-15:nodejs-jwt codemobiles$ pm2 stop mongod
[PM2] Applying action stopProcessId on app [mongod](ids: 1)
[PM2] [mongod](1) ✓
```

| App name      | id  | mode      | pid    | status         | restart | uptime | cpu   | mem         | user                    | watching          |
|---------------|-----|-----------|--------|----------------|---------|--------|-------|-------------|-------------------------|-------------------|
| mongod server | 1 0 | fork fork | 0 4957 | stopped online | 0 1     | 0 14s  | 0% 1% | 0 B 64.3 MB | codemobiles codemobiles | disabled disabled |

- Ex:

- pm2 start mongod
- pm2 stop mongod

# PM2

## React (npm start)

```
CodeMobiles-PLEK-15:cmPOS codemobiles$ pm2 start ng -- serve
[PM2] Starting /usr/local/bin/ng in fork_mode (1 instance)
[PM2] Done.
```

| App name  | id     | mode         | pid           | status           | restart | uptime    | cpu      | mem               | user                       | watching             |
|-----------|--------|--------------|---------------|------------------|---------|-----------|----------|-------------------|----------------------------|----------------------|
| ng server | 3<br>0 | fork<br>fork | 10141<br>3859 | online<br>online | 0<br>2  | 0s<br>28h | 0%<br>0% | 9.7 MB<br>24.8 MB | codemobiles<br>codemobiles | disabled<br>disabled |

```
Use `pm2 show <id|name>` to get more details about an app
CodeMobiles-PLEK-15:cmPOS codemobiles$ █
```

- Ex:
- pm2 start ng —serve
  - (—) = 2 dash

# PM2

## Watching (Restart)

### Auto restart apps on file change

PM2 can automatically restart your application when a file is modified in the current directory or its subdirectories:

```
pm2 start app.js --watch
```

If `--watch` is enabled, stopping it won't stop watching:

- `pm2 stop 0` will not stop watching
- `pm2 stop --watch 0` will stop watching

Restart toggle the `watch` parameter when triggered.

# NodeMon



**nodemon**

<https://github.com/remy/nodemon>

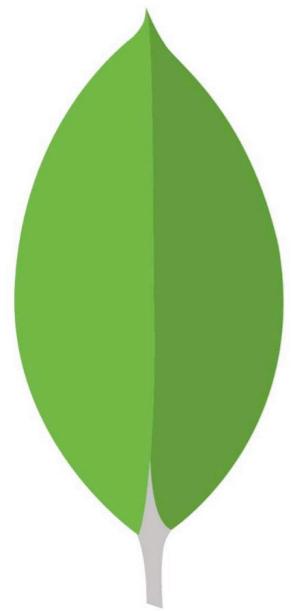
**nodemon** will watch the files in the directory in which nodemon was started, and if any files change, nodemon will automatically restart your node application.

# NodeMon



**nodemon**

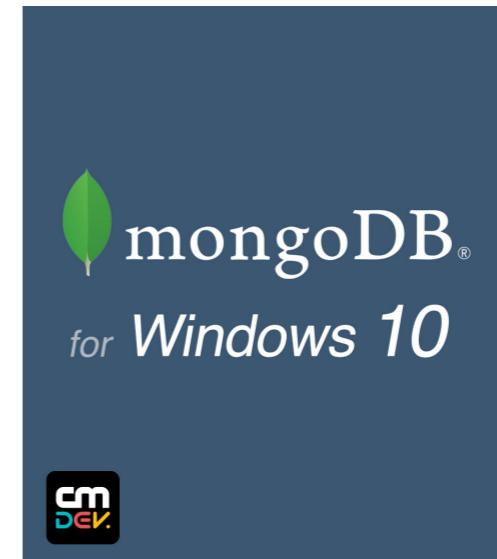
```
nodemon ./server.js localhost 8080
```



mongoDB

# Install MongoDB

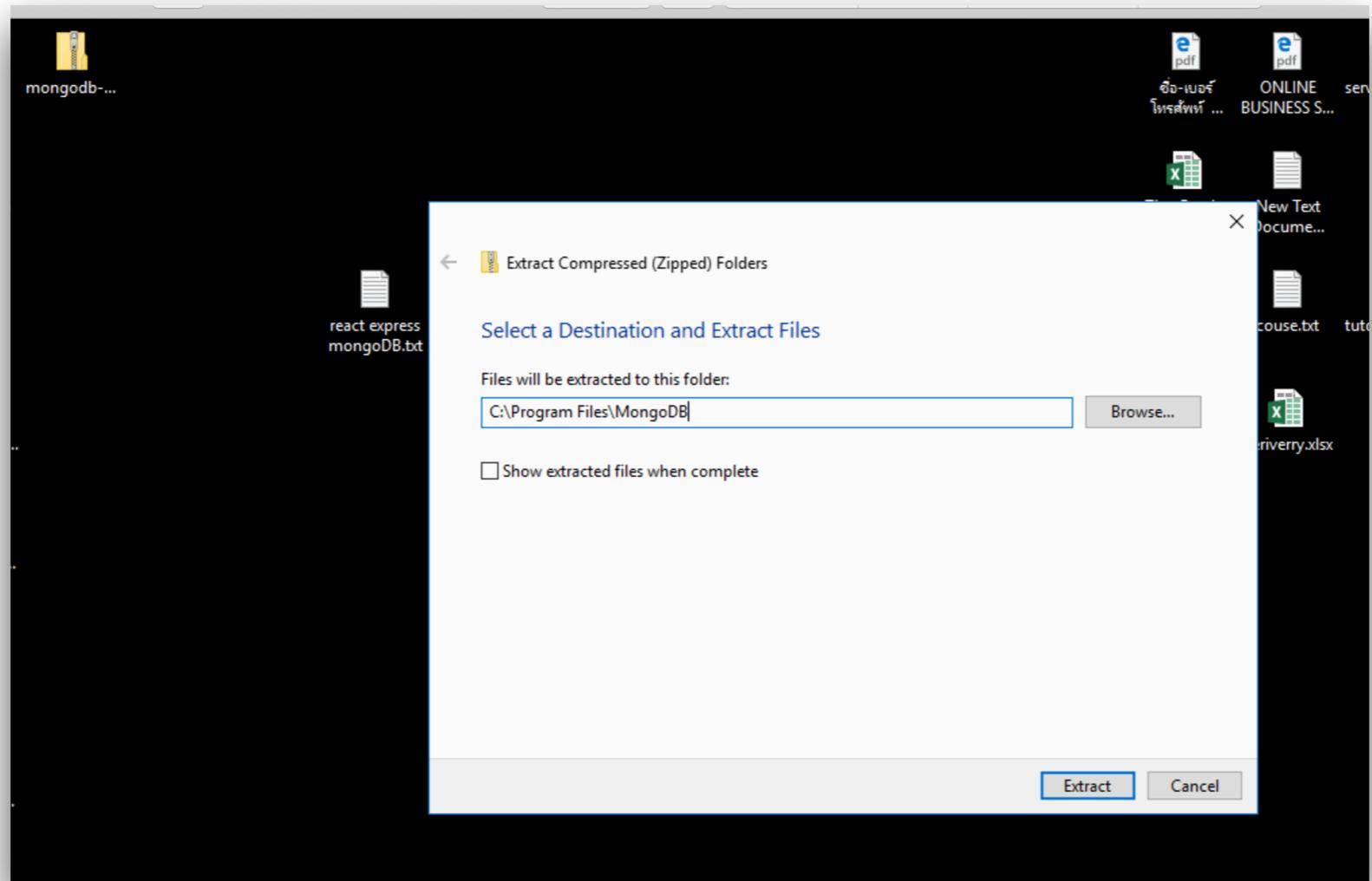
- <https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/>
- <https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/>
- Youtube : [https://www.youtube.com/watch?v=UqMbgmpWkSY&list=PLjPfp4Ph3gBq\\_BpBqdE2VE0Y1dfZByK9U](https://www.youtube.com/watch?v=UqMbgmpWkSY&list=PLjPfp4Ph3gBq_BpBqdE2VE0Y1dfZByK9U)



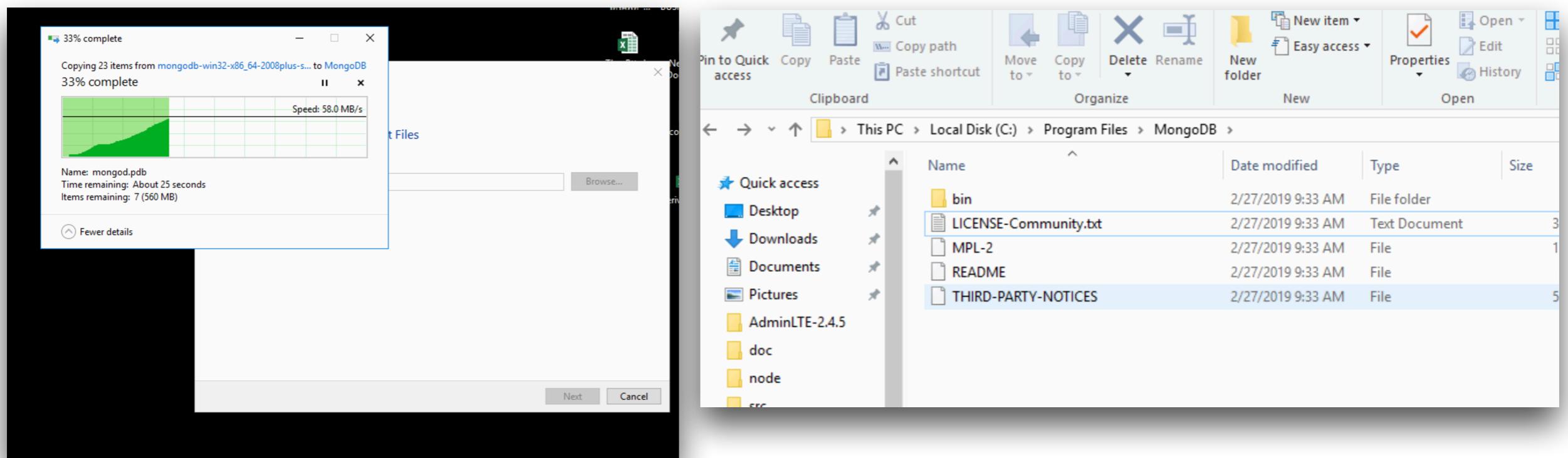
# Install Windows MongoDB

- Installation
  - download and unzip setup file
  - set environment variable
  - create folder c:/data/db
  - new cmd session and type "mongod"

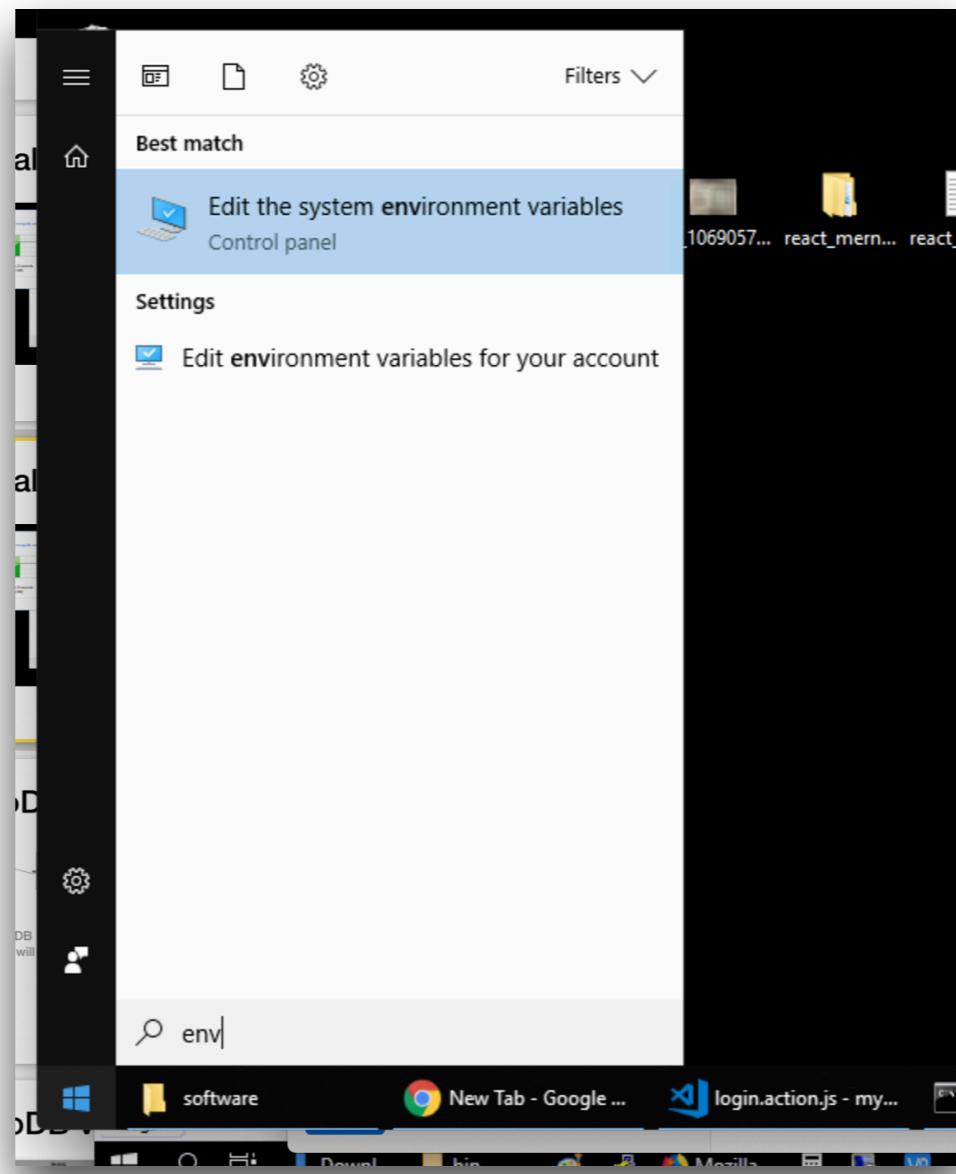
# Install Windows MongoDB



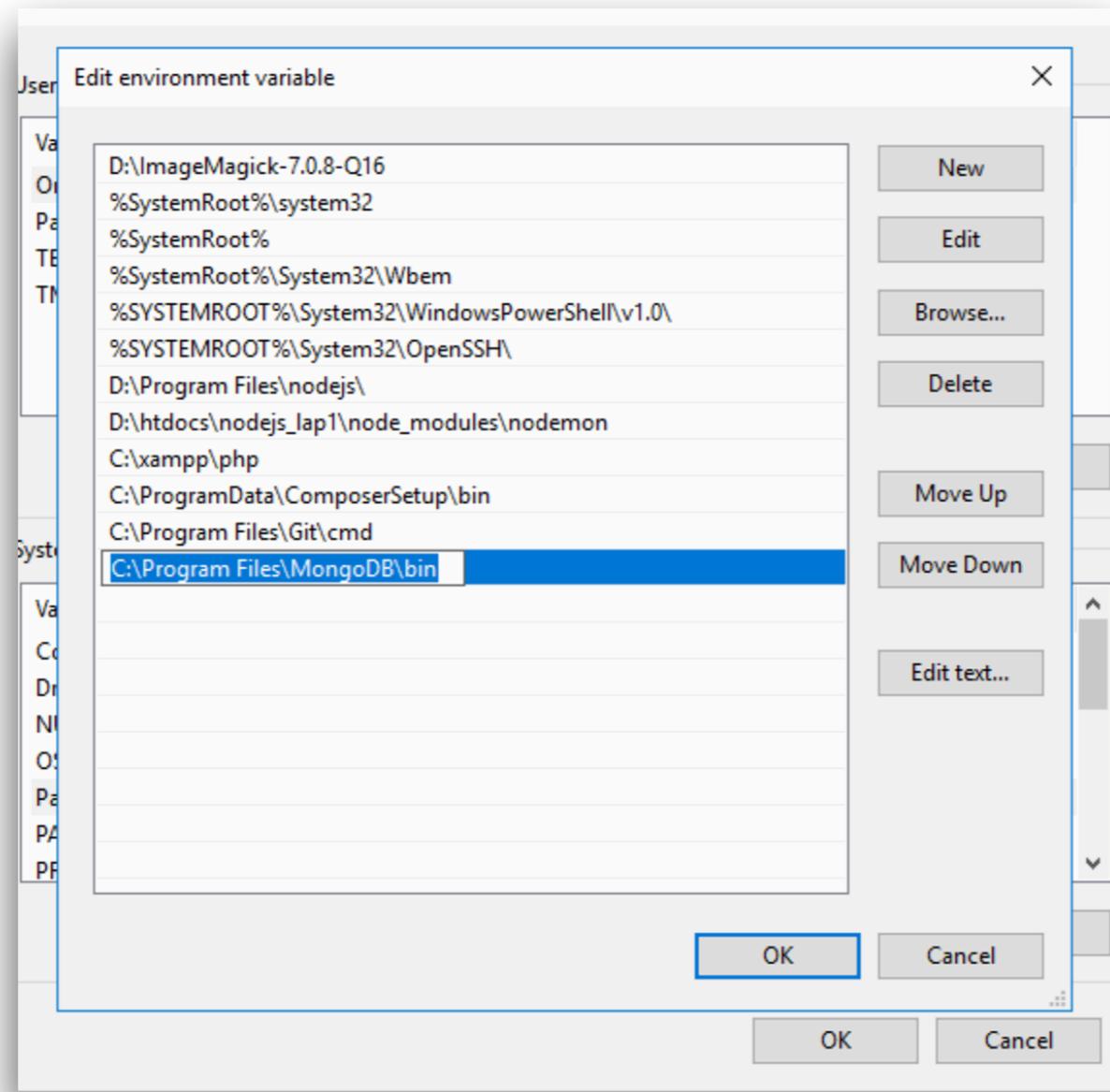
# Install Windows MongoDB



# Install Windows MongoDB

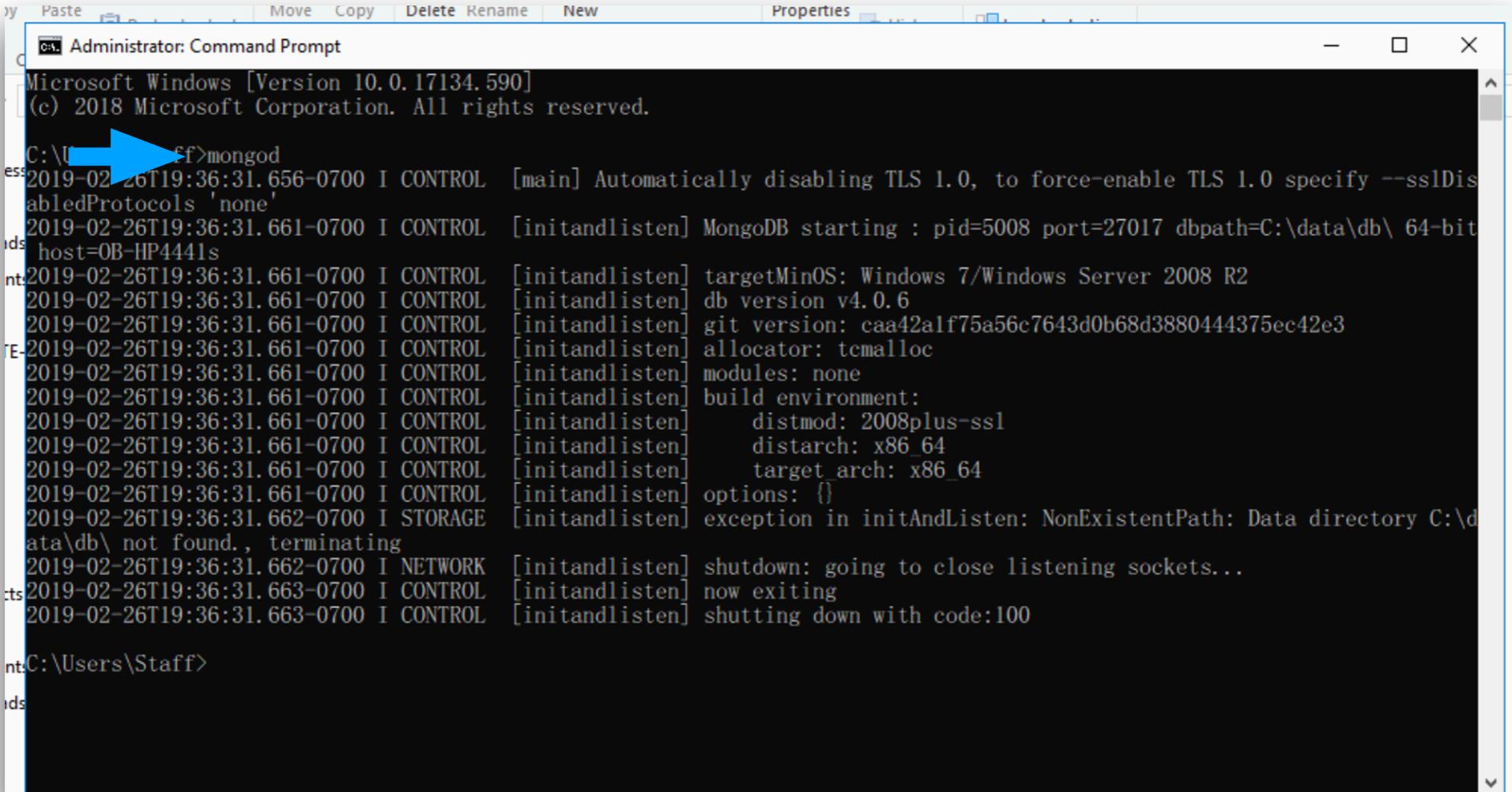


# Install Windows MongoDB



# Install Windows

# mongod



The screenshot shows an Administrator Command Prompt window on a Windows 10 system. The title bar reads "Administrator: Command Prompt". The command "mongod" is being typed at the prompt. A blue arrow points to the command line. The output shows the MongoDB daemon starting up, including log messages about TLS 1.0 disabling, targetMinOS, db version, git version, allocator, modules, build environment, distmod, distarch, target\_arch, options, and finally an exception due to a non-existent data directory. The process then shutdowns, exiting with code 100.

```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Uff>mongod
2019-02-26T19:36:31.656-0700 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2019-02-26T19:36:31.661-0700 I CONTROL [initandlisten] MongoDB starting : pid=5008 port=27017 dbpath=C:\data\db\ 64-bit
host=0B-HP4441s
2019-02-26T19:36:31.661-0700 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2019-02-26T19:36:31.661-0700 I CONTROL [initandlisten] db version v4.0.6
2019-02-26T19:36:31.661-0700 I CONTROL [initandlisten] git version: caa42a1f75a56c7643d0b68d3880444375ec42e3
2019-02-26T19:36:31.661-0700 I CONTROL [initandlisten] allocator: tcmalloc
2019-02-26T19:36:31.661-0700 I CONTROL [initandlisten] modules: none
2019-02-26T19:36:31.661-0700 I CONTROL [initandlisten] build environment:
2019-02-26T19:36:31.661-0700 I CONTROL [initandlisten]     distmod: 2008plus-ssl
2019-02-26T19:36:31.661-0700 I CONTROL [initandlisten]     distarch: x86_64
2019-02-26T19:36:31.661-0700 I CONTROL [initandlisten]     target_arch: x86_64
2019-02-26T19:36:31.661-0700 I CONTROL [initandlisten] options: {}
2019-02-26T19:36:31.662-0700 I STORAGE [initandlisten] exception in initAndListen: NonExistentPath: Data directory C:\data\db\ not found., terminating
2019-02-26T19:36:31.662-0700 I NETWORK [initandlisten] shutdown: going to close listening sockets...
2019-02-26T19:36:31.663-0700 I CONTROL [initandlisten] now exiting
2019-02-26T19:36:31.663-0700 I CONTROL [initandlisten] shutting down with code:100

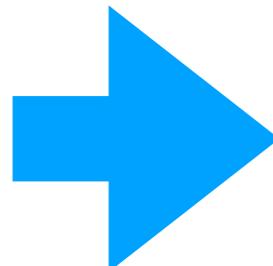
C:\Users\Staff>
```

# Install Windows mongod

```
scary disabling TLS 1.0, to force enable TLS 1.0 specify --sslVersion
| MongoDB starting : pid=5008 port→27017 dbpath=C:\data\db\ 64-bit
```

# Install Windows

# mongod



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.17134.590]
(c) 2018 Microsoft Corporation. All rights reserved.

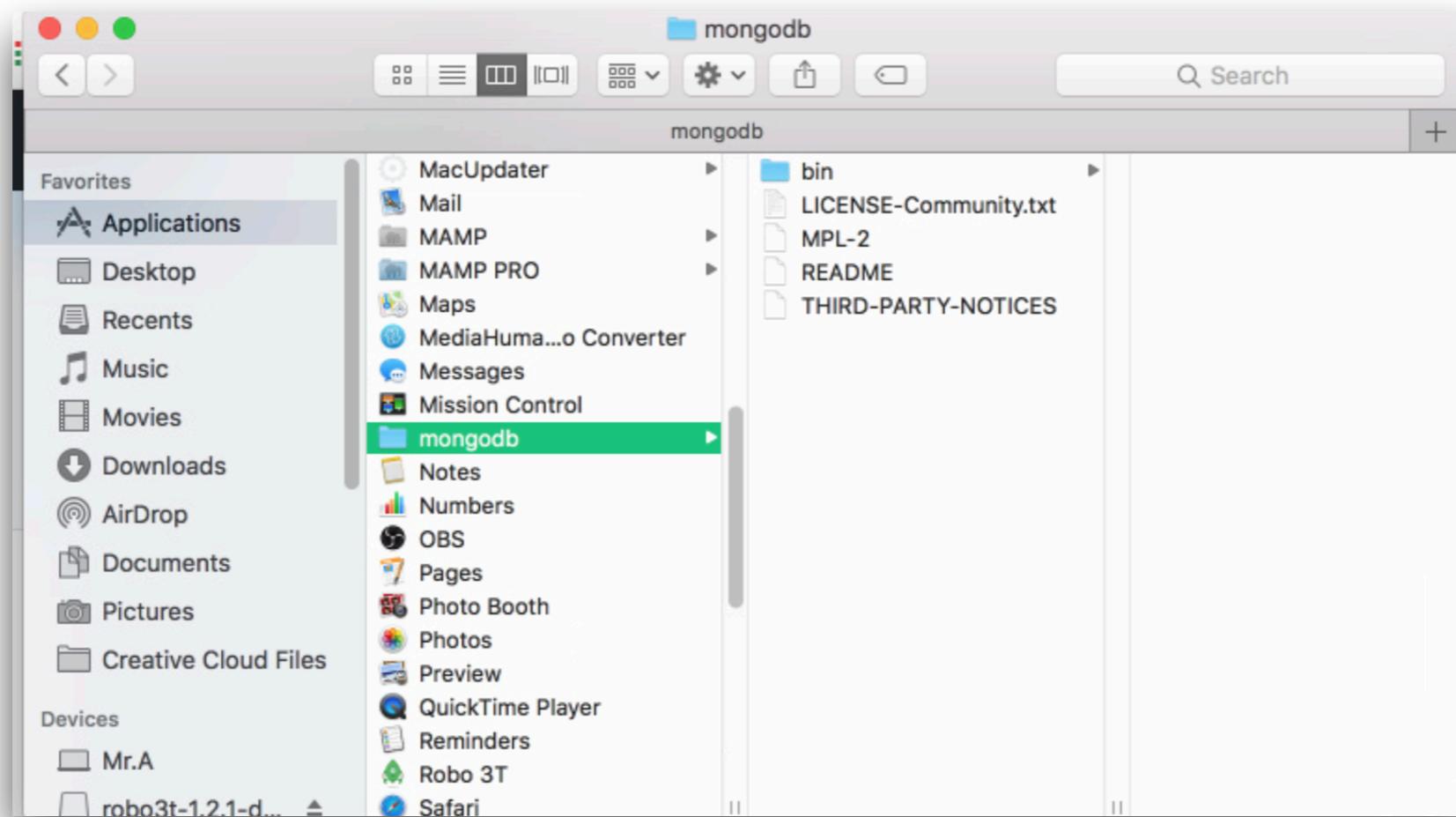
C:\Users\Staff>mongod
2019-02-26T19:36:31.656-0700 I CONTROL  [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2019-02-26T19:36:31.661-0700 I CONTROL  [initandlisten] MongoDB starting : pid=5008 port=27017 dbpath=C:\data\db\ 64-bit
host=OB-HP4441s
2019-02-26T19:36:31.661-0700 I CONTROL  [initandlisten] targetMinOS: Windows 7/Windows Server 2008 R2
2019-02-26T19:36:31.661-0700 I CONTROL  [initandlisten] db version v4.0.6
2019-02-26T19:36:31.661-0700 I CONTROL  [initandlisten] git version: caa42a1f75a56c7643d0b68d3880444375ec42e3
2019-02-26T19:36:31.661-0700 I CONTROL  [initandlisten] allocator: tcmalloc
2019-02-26T19:36:31.661-0700 I CONTROL  [initandlisten] modules: none
2019-02-26T19:36:31.661-0700 I CONTROL  [initandlisten] build environment:
2019-02-26T19:36:31.661-0700 I CONTROL  [initandlisten]   distmod: 2008plus-ssl
2019-02-26T19:36:31.661-0700 I CONTROL  [initandlisten]   distarch: x86_64
2019-02-26T19:36:31.661-0700 I CONTROL  [initandlisten]   target_arch: x86_64
2019-02-26T19:36:31.661-0700 I CONTROL  [initandlisten] options: {}
2019-02-26T19:36:31.662-0700 I STORAGE  [initandlisten] exception in initAndListen: NonExistentPath: Data directory C:\data\db\ not found., terminating
2019-02-26T19:36:31.662-0700 I NETWORK  [initandlisten] shutdown: going to close listening sockets...
2019-02-26T19:36:31.663-0700 I CONTROL  [initandlisten] now exiting
2019-02-26T19:36:31.663-0700 I CONTROL  [initandlisten] shutting down with code:100

C:\Users\Staff>mkdir c:\data
C:\Users\Staff>mkdir c:\data\db
C:\Users\Staff>
```

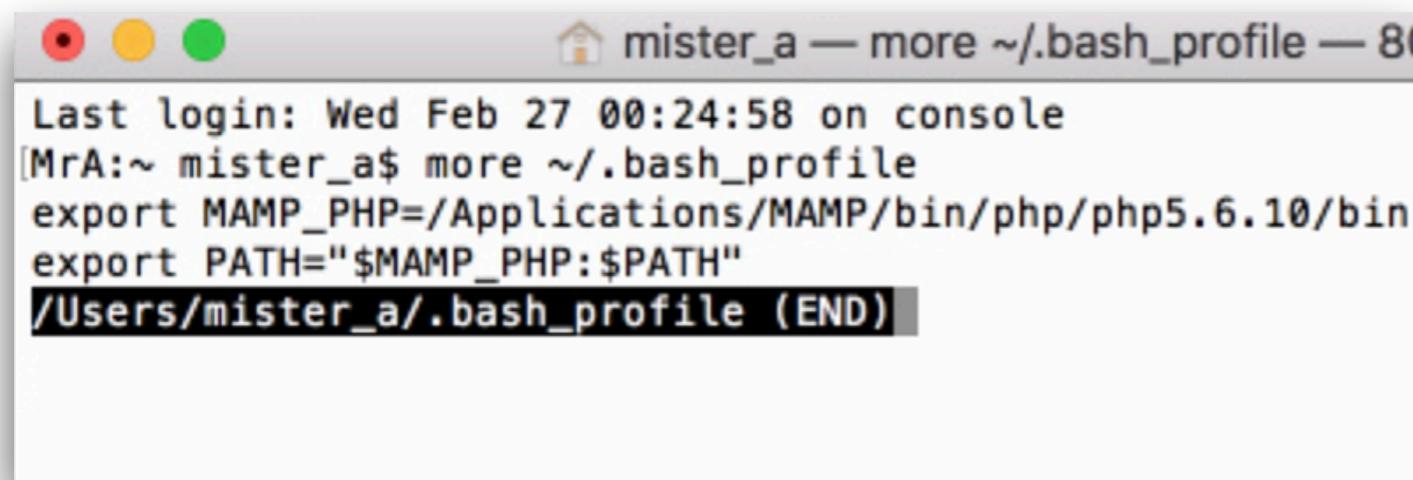
# Install macOS

- Installation
  - download and unzip setup file
  - set environment variable
  - create folder c:/data/db
  - new cmd session and type "mongod"

# Install macOS MongoDB

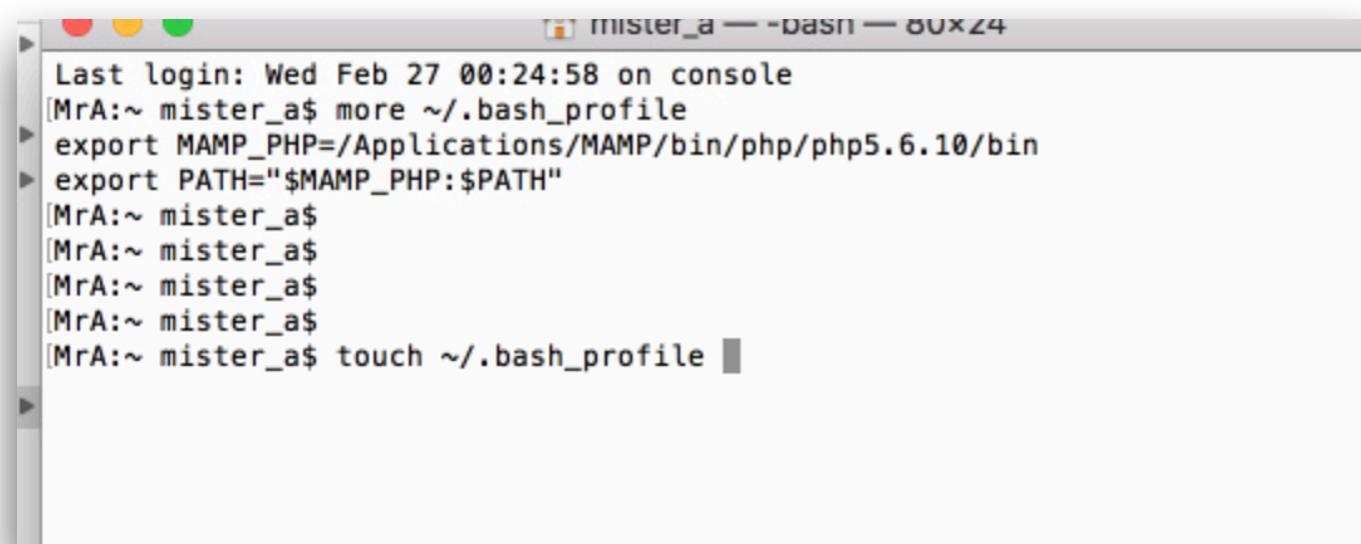


# Install macOS MongoDB



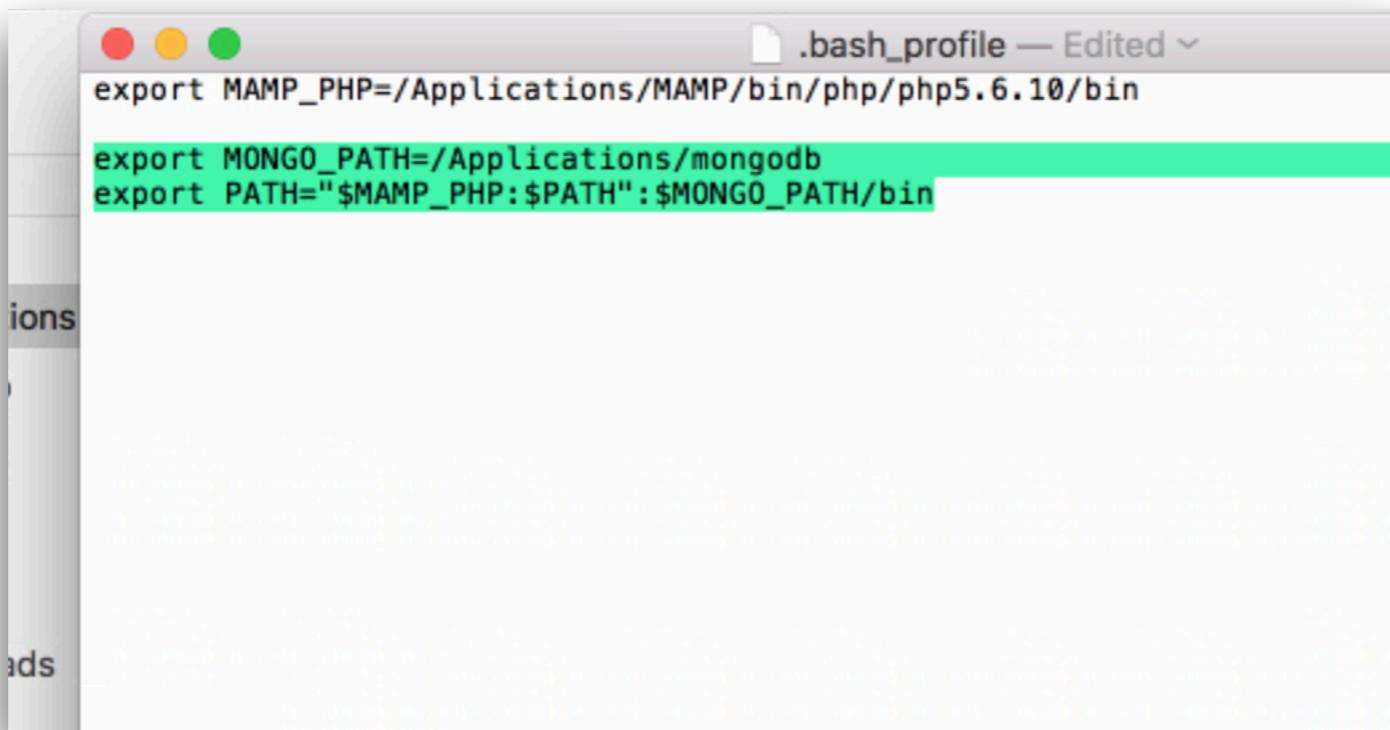
```
mister_a — more ~/.bash_profile — 80
Last login: Wed Feb 27 00:24:58 on console
[MrA:~ mister_a$ more ~/.bash_profile
export MAMP_PHP=/Applications/MAMP/bin/php/php5.6.10/bin
export PATH="$MAMP_PHP:$PATH"
/Users/mister_a/.bash_profile (END)
```

open ~/.bash\_profile  
touch ~/.bash\_profile



```
mister_a — -bash — 0x24
Last login: Wed Feb 27 00:24:58 on console
[MrA:~ mister_a$ more ~/.bash_profile
> export MAMP_PHP=/Applications/MAMP/bin/php/php5.6.10/bin
> export PATH="$MAMP_PHP:$PATH"
[MrA:~ mister_a$
[MrA:~ mister_a$
[MrA:~ mister_a$
[MrA:~ mister_a$
[MrA:~ mister_a$ touch ~/.bash_profile
```

# Install macOS MongoDB

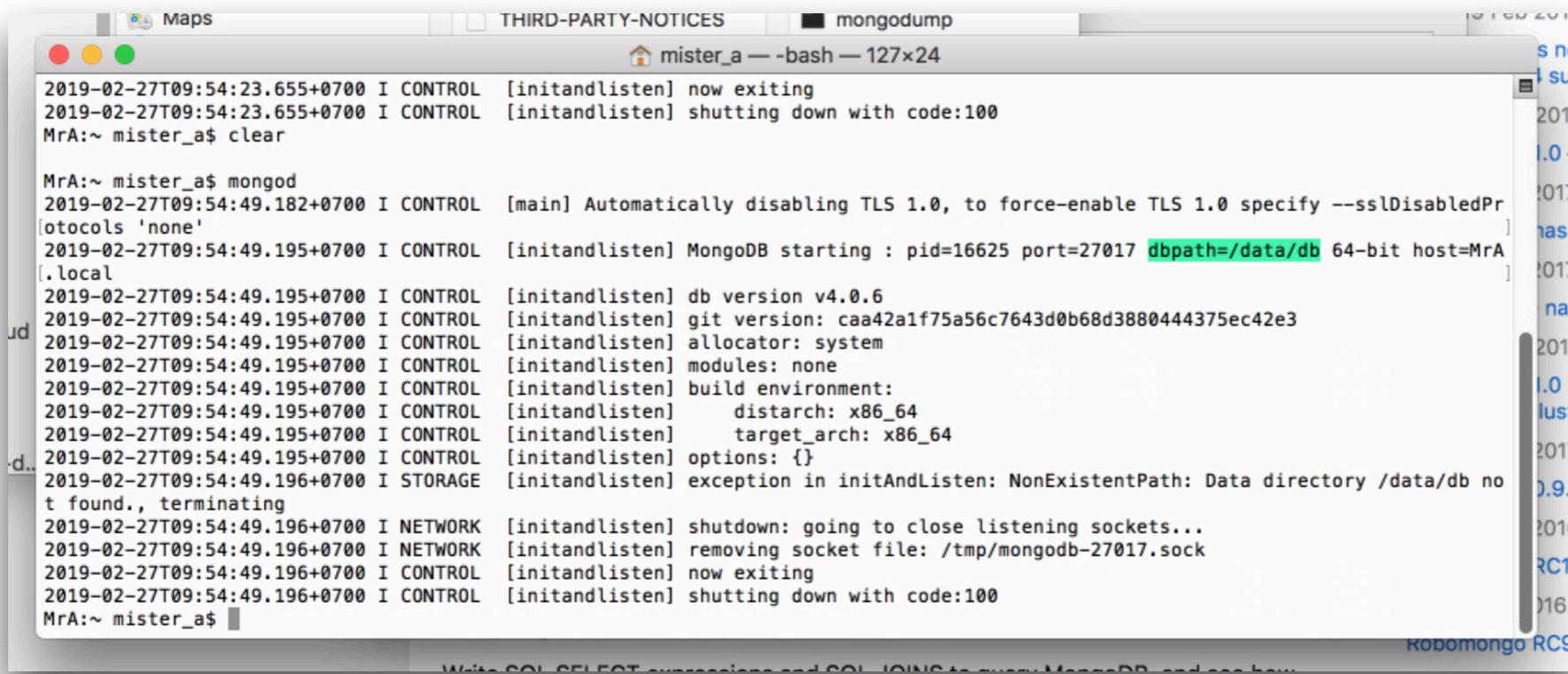


```
.bash_profile — Edited ▾  
export MAMP_PHP=/Applications/MAMP/bin/php/php5.6.10/bin  
export MONGO_PATH=/Applications/mongodb  
export PATH="$MAMP_PHP:$PATH":$MONGO_PATH/bin
```

```
export MONGO_PATH=/Applications/mongodb  
export PATH="$MAMP_PHP:$PATH":$MONGO_PATH/bin
```

# Install macOS

# MongoDB



The screenshot shows a macOS terminal window titled "mister\_a — bash — 127x24". The window contains the following log output from MongoDB:

```
2019-02-27T09:54:23.655+0700 I CONTROL [initandlisten] now exiting
2019-02-27T09:54:23.655+0700 I CONTROL [initandlisten] shutting down with code:100
MrA:~ mister_a$ clear

MrA:~ mister_a$ mongod
2019-02-27T09:54:49.182+0700 I CONTROL [main] Automatically disabling TLS 1.0, to force-enable TLS 1.0 specify --sslDisabledProtocols 'none'
2019-02-27T09:54:49.195+0700 I CONTROL [initandlisten] MongoDB starting : pid=16625 port=27017 dbpath=/data/db 64-bit host=MrA [.local]
2019-02-27T09:54:49.195+0700 I CONTROL [initandlisten] db version v4.0.6
2019-02-27T09:54:49.195+0700 I CONTROL [initandlisten] git version: caa42a1f75a56c7643d0b68d3880444375ec42e3
2019-02-27T09:54:49.195+0700 I CONTROL [initandlisten] allocator: system
2019-02-27T09:54:49.195+0700 I CONTROL [initandlisten] modules: none
2019-02-27T09:54:49.195+0700 I CONTROL [initandlisten] build environment:
2019-02-27T09:54:49.195+0700 I CONTROL [initandlisten]   distarch: x86_64
2019-02-27T09:54:49.195+0700 I CONTROL [initandlisten]   target_arch: x86_64
2019-02-27T09:54:49.196+0700 I STORAGE [initandlisten] exception in initAndListen: NonExistentPath: Data directory /data/db not found., terminating
2019-02-27T09:54:49.196+0700 I NETWORK [initandlisten] shutdown: going to close listening sockets...
2019-02-27T09:54:49.196+0700 I NETWORK [initandlisten] removing socket file: /tmp/mongodb-27017.sock
2019-02-27T09:54:49.196+0700 I CONTROL [initandlisten] now exiting
2019-02-27T09:54:49.196+0700 I CONTROL [initandlisten] shutting down with code:100
MrA:~ mister_a$
```

\*Error due to no the folder data/db

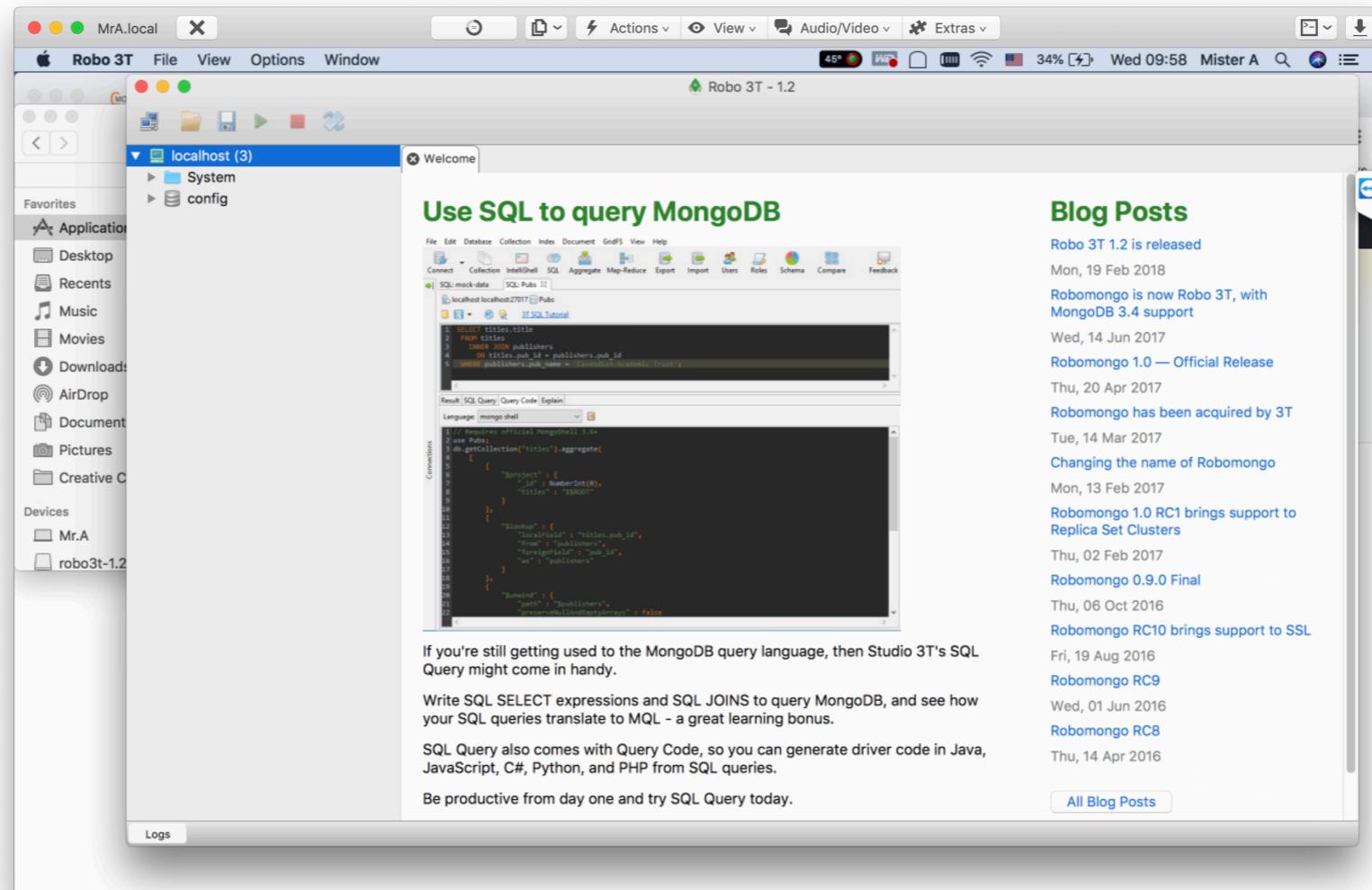
# Install macOS MongoDB

```
2019-02-27 10:54:49.190+07/00 T CONTROL [1
[MrA:~ mister_a$ sudo mkdir /data
[Password:
[MrA:~ mister_a$ sudo mkdir /data/db
[MrA:~ mister_a$ sudo chmod 777 /data/db
MrA:~ mister_a$ ]
```

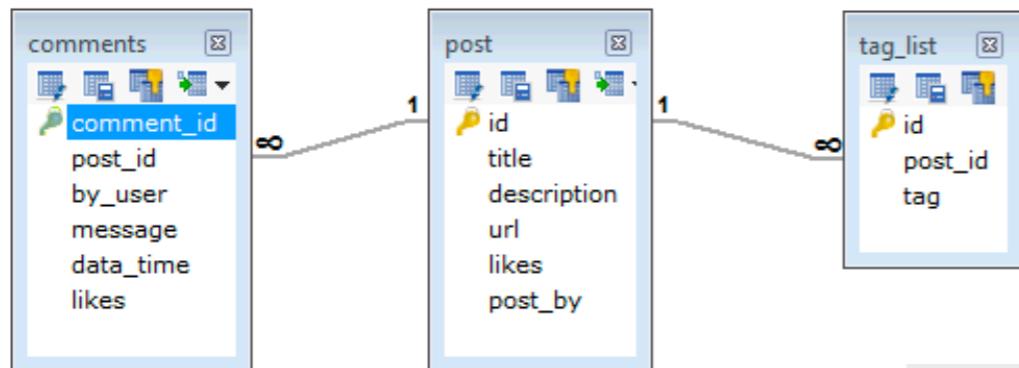
```
sudo mkdir /data
sudo mkdir /data/db
sudo chmod 777 /data/db
```

# Install macOS

# Test on Robot3T



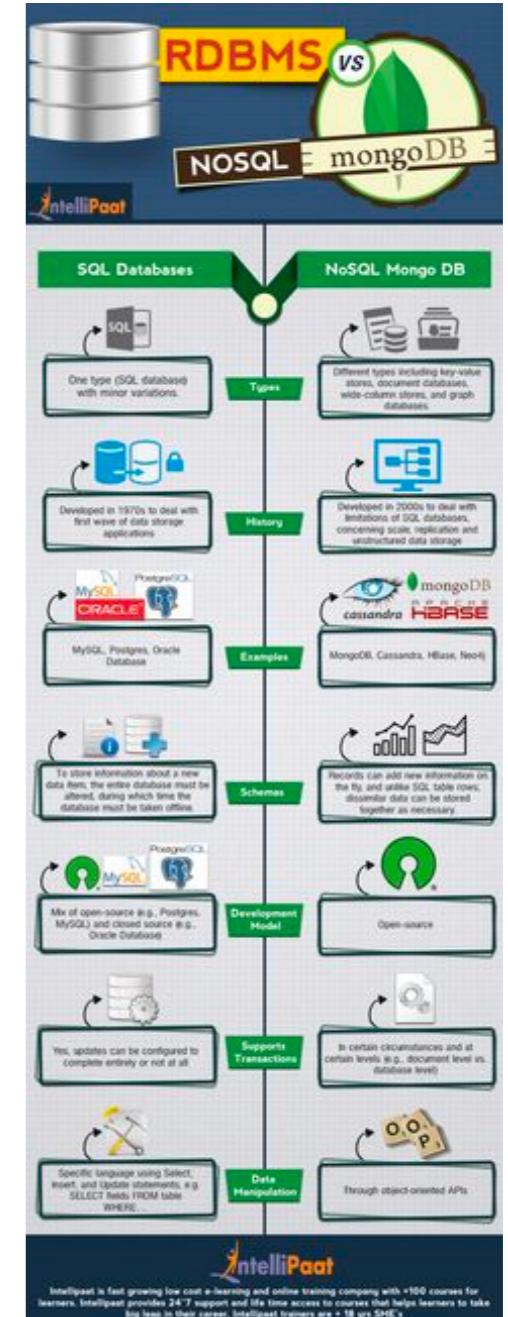
# MongoDB vs RDBMS



While in MongoDB schema, design will

```

{
  _id: ObjectId("7df78ad8902c"),
  title: 'MongoDB Overview',
  description: 'MongoDB is no sql database',
  by: 'tutorials point',
  url: 'http://www.tutorialspoint.com',
  tags: ['mongodb', 'database', 'NoSQL'],
  likes: 100,
  comments: [
    {
      user:'user1',
      message: 'My first comment',
      dateCreated: new Date(2011,1,20,2,15),
      like: 0
    },
    {
      user:'user2',
      message: 'My second comments',
      dateCreated: new Date(2011,1,25,7,45),
      like: 5
    }
  ]
}
  
```



# MongoDB vs MySQL

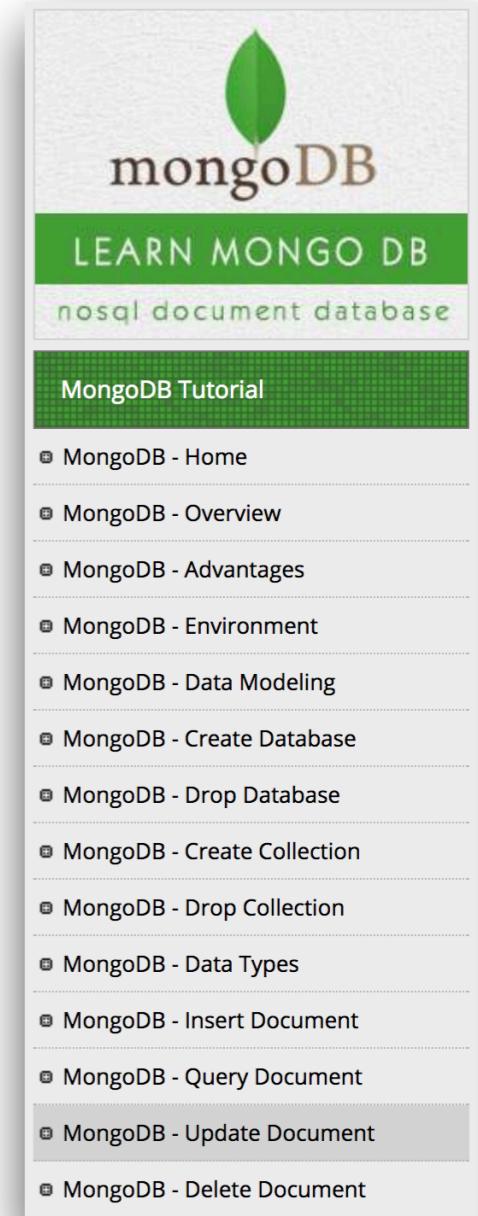
|  |  |
|------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| Table                                                                              | Collection                                                                          |
| Row                                                                                | Document                                                                            |
| Column                                                                             | Field                                                                               |
| Secondary Index                                                                    | Secondary Index                                                                     |
| JOINS                                                                              | Embedded Document, \$lookup & graphLookup                                           |
| GROUP_BY                                                                           | Aggregation Pipeline                                                                |

# MongoDB vs RDBMS

| RDBMS                      | MongoDB                                                               |
|----------------------------|-----------------------------------------------------------------------|
| Database                   | Database                                                              |
| Table                      | Collection                                                            |
| Tuple/Row                  | Document                                                              |
| column                     | Field                                                                 |
| Table Join                 | Embedded Documents                                                    |
| Primary Key                | Primary Key (Default key <code>_id</code> provided by mongodb itself) |
| Database Server and Client |                                                                       |
| Mysqld/Oracle              | <code>mongod</code>                                                   |
| mysql/sqlplus              | <code>mongo</code>                                                    |

# MongoDB Shell #2

- mongo
- show dbs
- use codemobiles
- show collections
- db.courses.insert({name: "new value"})
- db.courses.find()
- db.courses.find().pretty()
- db.courses.find({name: /Java/})
- db.courses.update(name: /Java/, {\$set {name: "New Java"}})



Ref : [https://www.tutorialspoint.com/mongodb/mongodb\\_update\\_document.htm](https://www.tutorialspoint.com/mongodb/mongodb_update_document.htm)

<https://chartio.com/resources/tutorials/how-to-use-a-sql-like-statement-in-mongodb/>

# MongoDB Shell #2

## Query Regular Expression

### Definition

#### \$regex

Provides regular expression capabilities for pattern matching *strings* in queries. MongoDB uses Perl compatible regular expressions (i.e. "PCRE" ) version 8.41 with UTF-8 support.

To use \$regex, use one of the following syntax:

```
{ <field>: { $regex: /pattern/, $options: '<options>' } }
{ <field>: { $regex: 'pattern', $options: '<options>' } }
{ <field>: { $regex: /pattern/<options> } }
```

In MongoDB, you can also use regular expression objects (i.e. /pattern/) to specify regular expressions:

```
{ <field>: /pattern/<options> }
```

For restrictions on particular syntax use, see [\\$regex vs. /pattern/ Syntax](#).

#### \$options

The following <options> are available for use with regular expression.

<https://docs.mongodb.com/manual/reference/operator/query/regex/>

# MongoDB Shell #3

Find all contacts with at least one work phone or  
hired after 2014-02-02

SQL

```
select A.did, A.lname, A.hiredate, B.type,  
B.number from contact A left outer join phones B  
on (B.did = A.did) where b.type = 'work' or  
A.hiredate > '2014-02-02'::date
```

MongoDB CLI

```
db.contacts.find({ "$or": [  
    { "phones.type": "work" },  
    { "hiredate": { "$gt": new ISODate("2014-02-02") } }  
]});
```

# MongoDB Shell #4

## Export and Import

### #Export

```
mongodump -d cmpos -o /Users/codemobiles/Desktop/
```

### #Import

```
mongorestore -d cmpos /Users/codemobiles/Desktop/
```

# MongoDB Tools

# MongoDB Tools

- Mongoose - MongoDB Schema Framework
- Robo 3T - MongoDB Client
- PM2 - start/stop service (mongod)

mongoose



PM2

# Mongoose



**using mongooseJS**

in node and mongoDB applications

**npm install mongoose**

**<http://mongoosejs.com/docs/>**

# Mongoose

## Connecting to MongoDB

```
1 //Import the mongoose module
2 var mongoose = require('mongoose');
3
4 //Set up default mongoose connection
5 var mongoDB = 'mongodb://127.0.0.1/my_database';
6 mongoose.connect(mongoDB);
7 // Get Mongoose to use the global promise library
8 mongoose.Promise = global.Promise;
9 //Get the default connection
10 var db = mongoose.connection;
11
12 //Bind connection to error event (to get notification of connection errors)
13 db.on('error', console.error.bind(console, 'MongoDB connection error:'));
```

# Mongoose

## Defining Schemas

```
1 //Require Mongoose
2 var mongoose = require('mongoose');
3
4 //Define a schema
5 var Schema = mongoose.Schema;
6
7 var SomeModelSchema = new Schema({
8     a_string: String,
9     a_date: Date
10});
```

# Mongoose

## Creating Model

```
1 // Define schema
2 var Schema = mongoose.Schema;
3
4 var SomeModelSchema = new Schema({
5     a_string: String,
6     a_date: Date
7 });
8
9 // Compile model from schema
10 var SomeModel = mongoose.model('SomeModel', SomeModelSchema );
```

# Mongoose

## Creating Model

```
1 // Define schema
2 var Schema = mongoose.Schema;
3
4 var SomeModelSchema = new Schema({
5     a_string: String,
6     a_date: Date
7 });
8
9 // Compile model from schema
10 var SomeModel = mongoose.model('SomeModel', SomeModelSchema );
```

# Mongoose

## Creating and Modify Documents

```
// Create an instance of model SomeModel
var awesome_instance = new SomeModel({ name: 'awesome' });

// Save the new model instance, passing a callback
awesome_instance.save(function (err) {
  if (err) return handleError(err);
  // saved!
});
```

```
SomeModel.create({ name: 'also_awesome' }, function (err, awesome_instance) {
  if (err) return handleError(err);
  // saved!
});
```

# Mongoose

## Searching for records

```
var Athlete = mongoose.model('Athlete', yourSchema);

// find all athletes who play tennis, selecting the 'name' and 'age' fields
Athlete.find({ 'sport': 'Tennis' }, 'name age', function (err, athletes) {
  if (err) return handleError(err);
  // 'athletes' contains the list of athletes that match the criteria.
})
```

# Mongoose

## Searching for records with option

```
// find all athletes that play tennis
var query = Athlete.find({ 'sport': 'Tennis' });

// selecting the 'name' and 'age' fields
query.select('name age');

// limit our results to 5 items
query.limit(5);

// sort by age
query.sort({ age: -1 });

// execute the query at a later time
query.exec(function (err, athletes) {
  if (err) return handleError(err);
  // athletes contains an ordered list of 5 athletes who play Tennis
})
```

# Mongoose

## Join / Aggregate

| Key                                        | Key                                        | Value                                                      | Type     |
|--------------------------------------------|--------------------------------------------|------------------------------------------------------------|----------|
| ▼ (1) ObjectId("5b8f8756d898fb0405af7cb3") | ▼ (1) ObjectId("5aa7c5137a3066a2b958947d") | { 6 fields }                                               | Object   |
| ❑ _id                                      | ❑ _id                                      | ObjectId("5aa7c5137a3066a2b958947d")                       | ObjectId |
| # subtotal                                 | # username                                 | chaiya...@gmail.com                                        | String   |
| # discount                                 | # password                                 | \$2a\$08\$mq4EZf1kuYoLQ2pgwFERTu0ULXnfSWrlim...            | String   |
| # shipping_cost                            | # level                                    | normal                                                     | String   |
| # tax_percent                              | # created                                  | 2018-03-13 12:33:23.161Z                                   | Date     |
| # total                                    | # __v                                      | 0                                                          | Int32    |
| # paid                                     | ► (2) ObjectId("5b8f8756d898fb0405af7cb3") | { 6 fields }                                               | Object   |
| # change                                   |                                            |                                                            |          |
| # order_list                               |                                            | [{"_id": "5abd69902ded8b1b1a6f7d73", "product_id": "..."}] | String   |
| # payment_type                             |                                            | cash                                                       | String   |
| # payment_detail                           |                                            | full                                                       | String   |
| ❑ staff_id                                 |                                            | ObjectId("5aa7c5137a3066a2b958947d")                       | ObjectId |
| ❑ seller_id                                |                                            | sr0001                                                     | String   |
| ❑ buyer_id                                 |                                            | by0000                                                     | String   |
| # comment                                  |                                            | x                                                          | String   |
| # timestamp                                |                                            | 2018-09-07 12:00:54.938Z                                   | Date     |
| # transaction_id                           |                                            | 119                                                        | Int32    |
| # __v                                      |                                            | 0                                                          | Int32    |
| (2) ObjectId("5b8f8756d898fb0405af7cb3")   |                                            | { 12 fields }                                              | Object   |

# Mongoose

## Join / Aggregate

```
{  
  "_id": "5b9268765b41135e92d50bd4",  
  "subtotal": 0,  
  "discount": 0,  
  "shipping_cost": 0,  
  "tax_percent": 0,  
  "total": 630,  
  "paid": 630,  
  "change": 0,  
  "order_list": "[{\\"_id\\":\\"5abd69902ded8b1b1a6f7d73\\",\\"product_id\\":11,\\"image\\":\\"product_23.jpg\\",\\"name\\":\\"Arduino  
Nano 3.0 Mini USB รุนใหม่ใช้ชิพ CH340G แอมساຍ Mini USB\\",\\"price\\":130,\\"stock\\":2,\\"created\\":\\"2018-03-30T00:11  
:45.109Z\\",\\"__v\\":0,\\"qty\\":1},{\"_id\":\\\"5abd69902ded8b1b1a6f7d35\\\",\\\"product_id\\\":12,\\\"name\\\":\\\"Arduino UNO R3  
แอมساຍ USB Type A Male/B Male Cable\\\",\\\"image\\\":\\\"product_04.jpg\\\",\\\"stock\\\":1000,\\\"price\\\":300,\\\"created\\\":\\\"2018  
-03-30T00:11:45.109Z\\\",\\\"__v\\\":0,\\"qty\\":1},{\"_id\\":\\"5abd69902ded8b1b1a6f7d63\\",\\"product_id\\":13,\\"image\\"  
:\\"product_05.jpg\\",\\"name\\":\\"Arduino MEGA 2560 R3 ใช้ชิพ USB CH340 รุนใหม่ แอมساຍ USB\\",\\"stock\\":100,\\"price\\"  
:200,\\"created\\":\\"2018-03-30T00:11:45.109Z\\",\\"__v\\":0,\\"qty\\":1}]}",  
  "payment_type": "cash",  
  "payment_detail": "full",  
  "staff_id": "chaiyasit.t@gmail.com",  
  "seller_id": "sr0001",  
  "buyer_id": "by0000",  
  "comment": "x",  
  "timestamp": "2018-09-07T12:00:54.938Z",  
  "transaction_id": 119,  
  "__v": 0}
```

# Mongoose

## Join Collection : Aggregate

```
router.get("/transaction/id/:id", function(req, res) {
  Trans.aggregate([
    { $match: { transaction_id: Number(req.params.id) } }, → Search Condition
    { $lookup: {
        from: "users",
        localField: "staff_id",
        foreignField: "_id",
        as: "staff_id"
      }
    },
    { $project: { "staff_id.password": 0 } },
    { $unwind: "$staff_id" }
  ]).then(transaction => {
    const resObj = transaction.map(value => {
      value.staff_id = value.staff_id.username;
      return value;
    });
    res.json(resObj[0]);
  })
});
```

→ join condition

→ include/exclude

→ array to dictionary

→ map to transforms  
to simple data  
format

# Mongoose

## Join Collection : Aggregate

```
router.get("/transaction/id/:id", function(req, res) {
  Trans.aggregate([
    { $match: { transaction_id: Number(req.params.id) } },
    { $lookup: {
        from: "users",
        localField: "staff_id",
        foreignField: "_id",
        as: "staff_id"
      }
    },
    { $project: { "staff_id.password": 0 } },
    { $unwind: "$staff_id" }
  ]).then(transaction => {
    const resObj = transaction.map(value => {
      value.staff_id = value.staff_id.username;
      return value;
    });
    res.json(resObj[0]);
  })
});
```

# Mongoose

## Common Functions

- Mongoose - MongoDB Schema Framework
- Robo 3T - MongoDB Client
- PM2 - start/stop service (mongod)