

# Introduction to React

BY PANOT WONGKHOT

FRONT-END PROGRAMMER BOOTCAMP

<https://github.com/panotza/frontend-bootcamp>



# React คือ

Javascript library สำหรับสร้าง User Interface สร้างขึ้นโดย Facebook ในปี 2013 รุ่น Stable ล่าสุดอยู่ที่เวอร์ชัน 16.2.0

 facebook / react

 Watch ▼

5,626

★ Star

91,086

 Fork

17,205



# เราใช้ React ตอนไหน ?

---

- ข้อมูล UI ที่เราจะต้องจัดการมันเริ่มยุ่งยากหากใช้ Javascript ธรรมดา
- เราจะสร้าง Single Page Application (SPA)

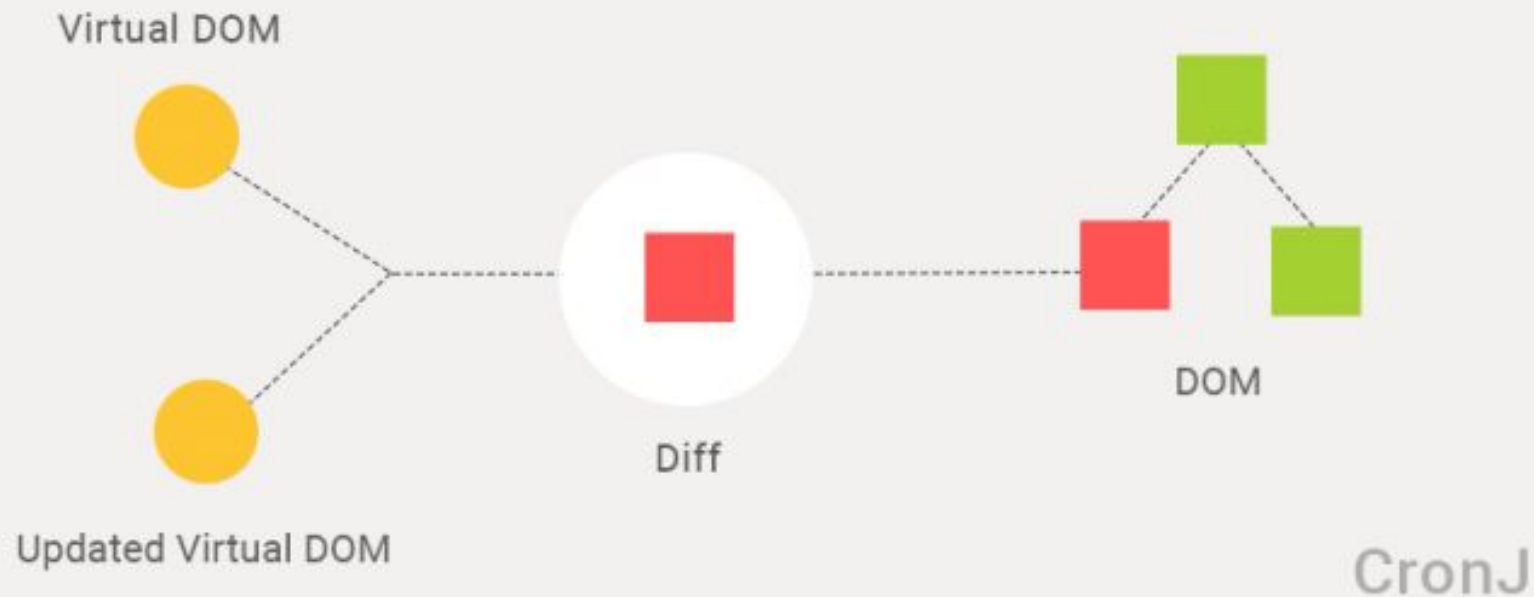


# ข้อดีของ React



# Virtual Dom

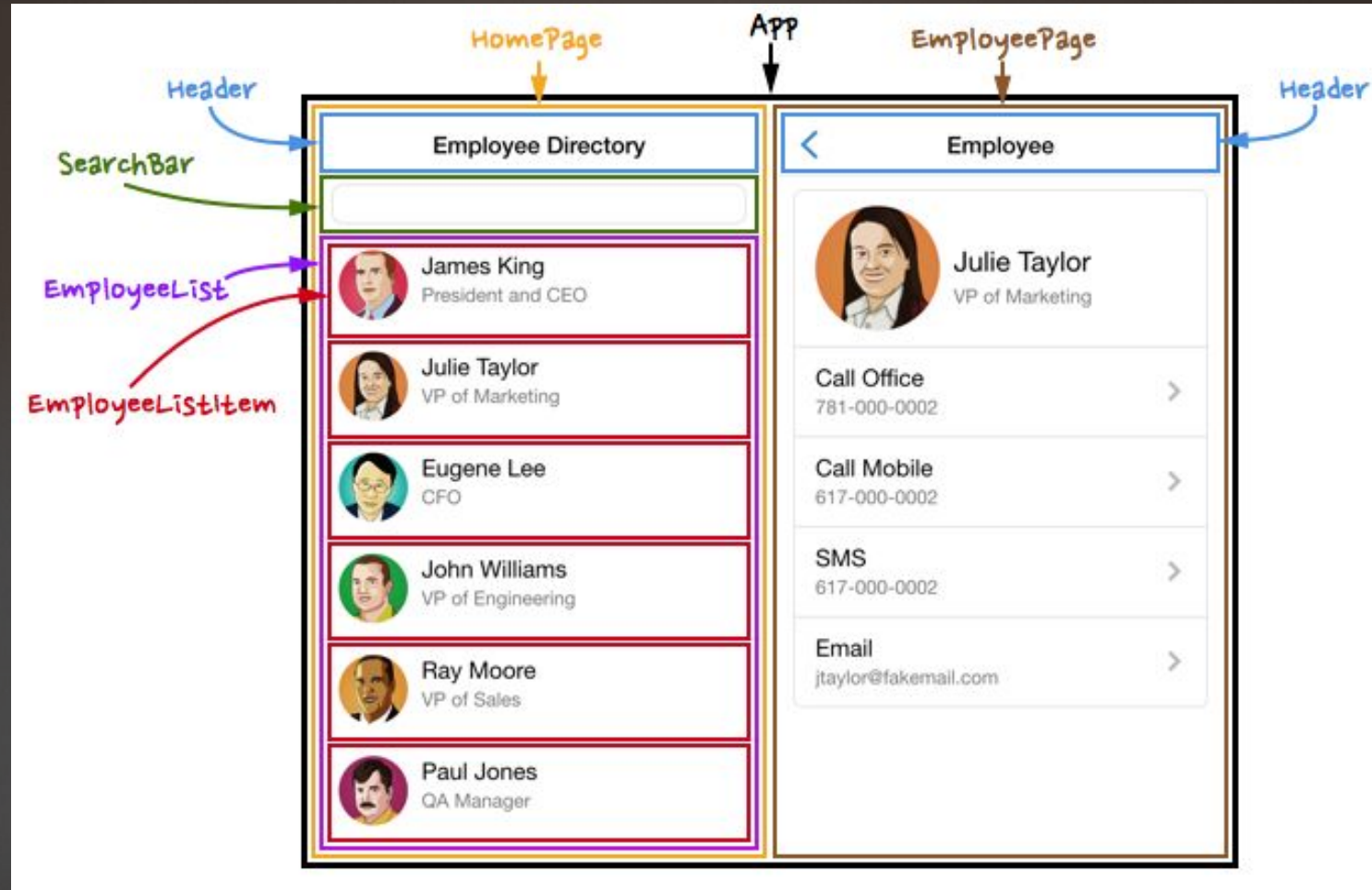
## Document Object Model



<https://www.cronj.com/blog/virtual-dom-react-js/>



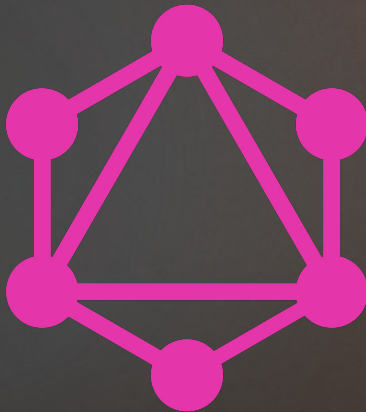
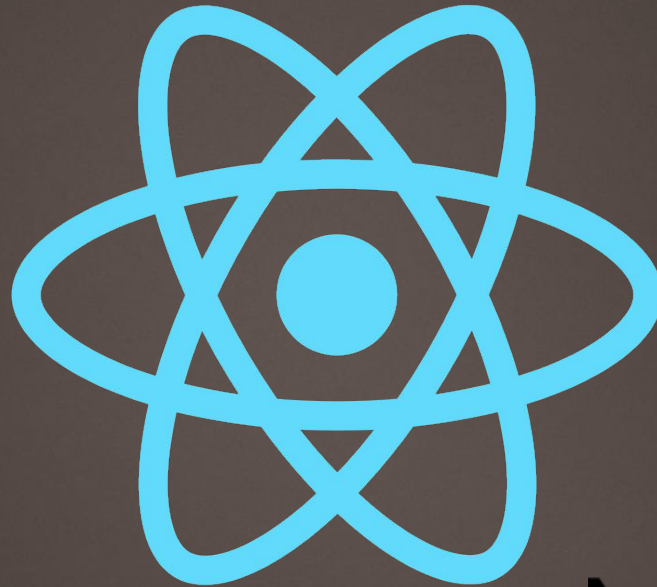
# Component-Based



<http://coenraets.org/blog/wp-content/uploads/2014/12/uimockscript.png>

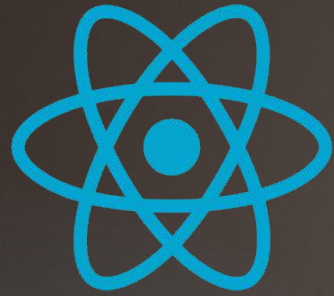
# Ecosystem

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

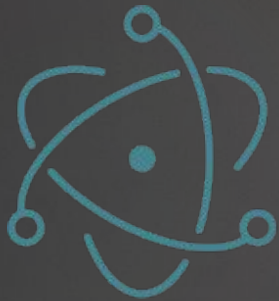


# Learn Once, Write Anywhere

---



# React Native



# ELECTRON





# Pre-basic React



# คำสั่งที่ใช้บ่อย

---

- Async Await Promises
- Const and Let
- Class
- Arrow Functions
- Spread Operator
- Map and Filter
- Import, Export

ใช้ ES5 เขียนได้ แต่เรียน ES $\geq$ 6 กันเถอะครับ



UI ที่เราเห็นกันบนหน้าเว็บที่สร้างด้วย  
React ล้วนเกิดจาก...

1. **React.createElement**

2. **ReactDOM.render**



# 1. React.createElement



## React Element

# React Element

---

คือ Javascript **Object { }** ธรรมดาๆ ที่ใช้กำหนดว่า html tag (Component Instance) จะมี ประเภท รูปร่างเป็นอย่างไร







```
const helloReact = React.createElement(  
  'div',  
  null,  
  'Hello React!'  
);
```

# React Element

```
▼ Object ⓘ  
  $$typeof: Symbol(react.element)  
  key: null  
  ▶ props: {children: "Hello React!"}  
  ref: null  
  type: "div"  
  _owner: null  
  ▶ __proto__: Object
```



**1. ~~React.createElement~~**



**2. ReactDOM.render**



```
<div id='root'></div>
<script type='text/javascript'>
  const helloReact = React.createElement(
    'div',
    null,
    'Hello React!'
  );
ReactDOM.render(helloReact, document.getElementById('root'));
</script>
```

# Hello React!





# Lab 1: Create Element & Render

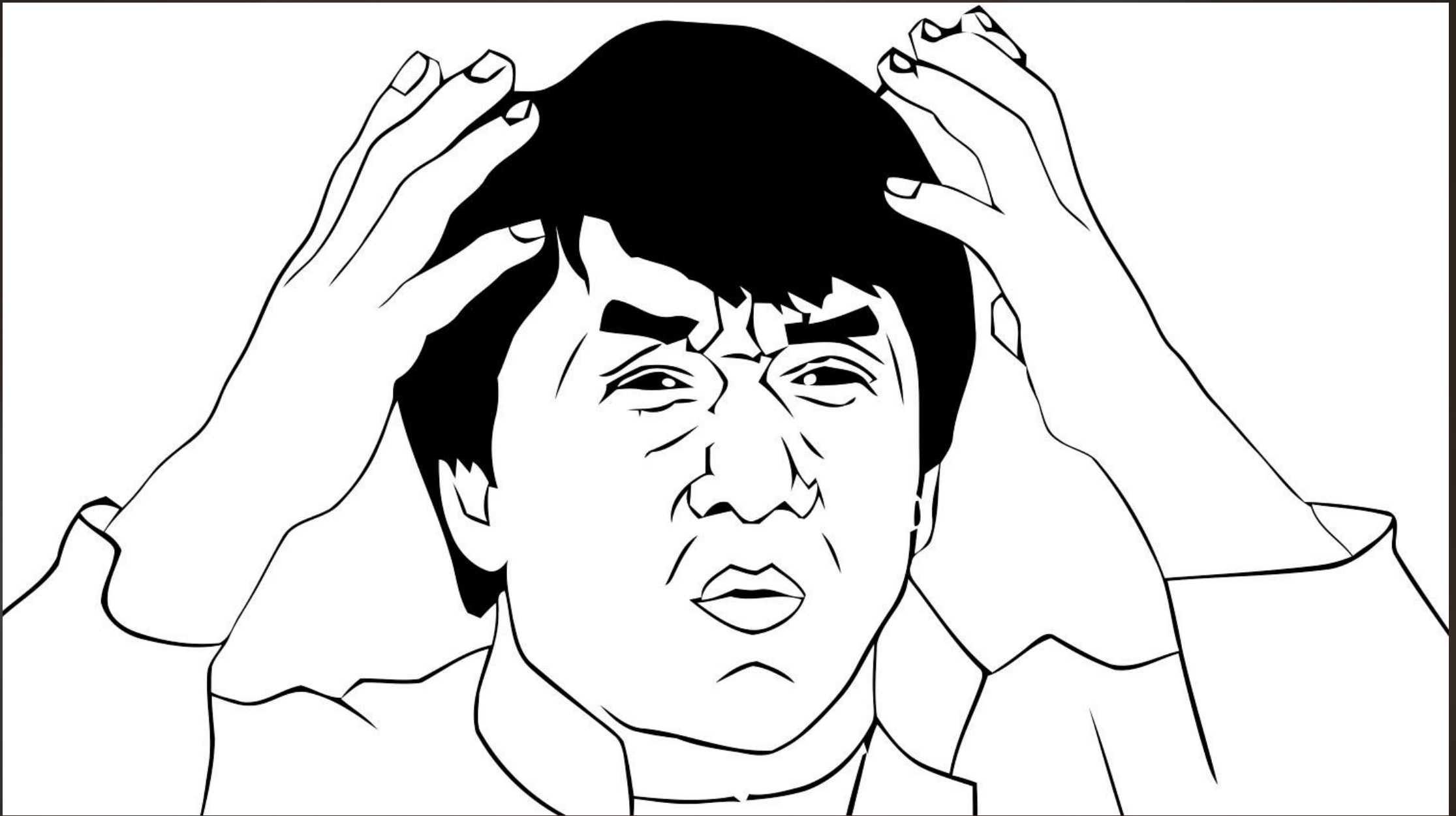
---

เปิดไฟล์ lab1.html ขึ้นมาแล้วลอง แก้ไข สร้าง element  
อะไรก็ได้ แล้วก็ลองใช้ ReactDOM.render สร้าง UI ออกมา  
ดู





```
ReactDOM.render(  
  React.createElement('div', null, React.createElement('h1', null, 'Hello  
React'),  
    React.createElement('p', null, 'this is paragraph',  
      React.createElement('span', null, 'this is span'),  
      React.createElement('b', null, React.createElement('span', null, 'this  
is icon'))  
    ), React.createElement('button', null, 'Click me!')  
  ),  
  document.getElementById('root')  
);
```



# JSX to the rescue!



# Basic React





# Introducing JSX

**JSX** คือรูปแบบการเขียนที่รวม **Javascript** กับ **XML** เข้าด้วยกัน ทำให้เราสามารถเขียน syntax ที่หน้าตาคล้าย HTML ในภาษา Javascript ได้ โดยปกติแล้วต้องใช้ควบคู่กับ **Transpiler** เช่น **Babel**



# Babel ???



# BABEL

Transpiler คือตัวแปลง source code จาก  
เวอร์ชันหนึ่งไปอีกเวอร์ชันหนึ่ง เช่น **ES6** เป็น **ES5**  
ปัจจุบันมีหน้าที่แปลง **JSX** เป็น **Javascript** ด้วย



# จากเดิม

```
const helloReact = React.createElement(  
  'div',  
  null,  
  'Hello React!'  
);
```



# เขียนด้วย JSX

```
const helloReact = <div>Hello React!</div>;
```





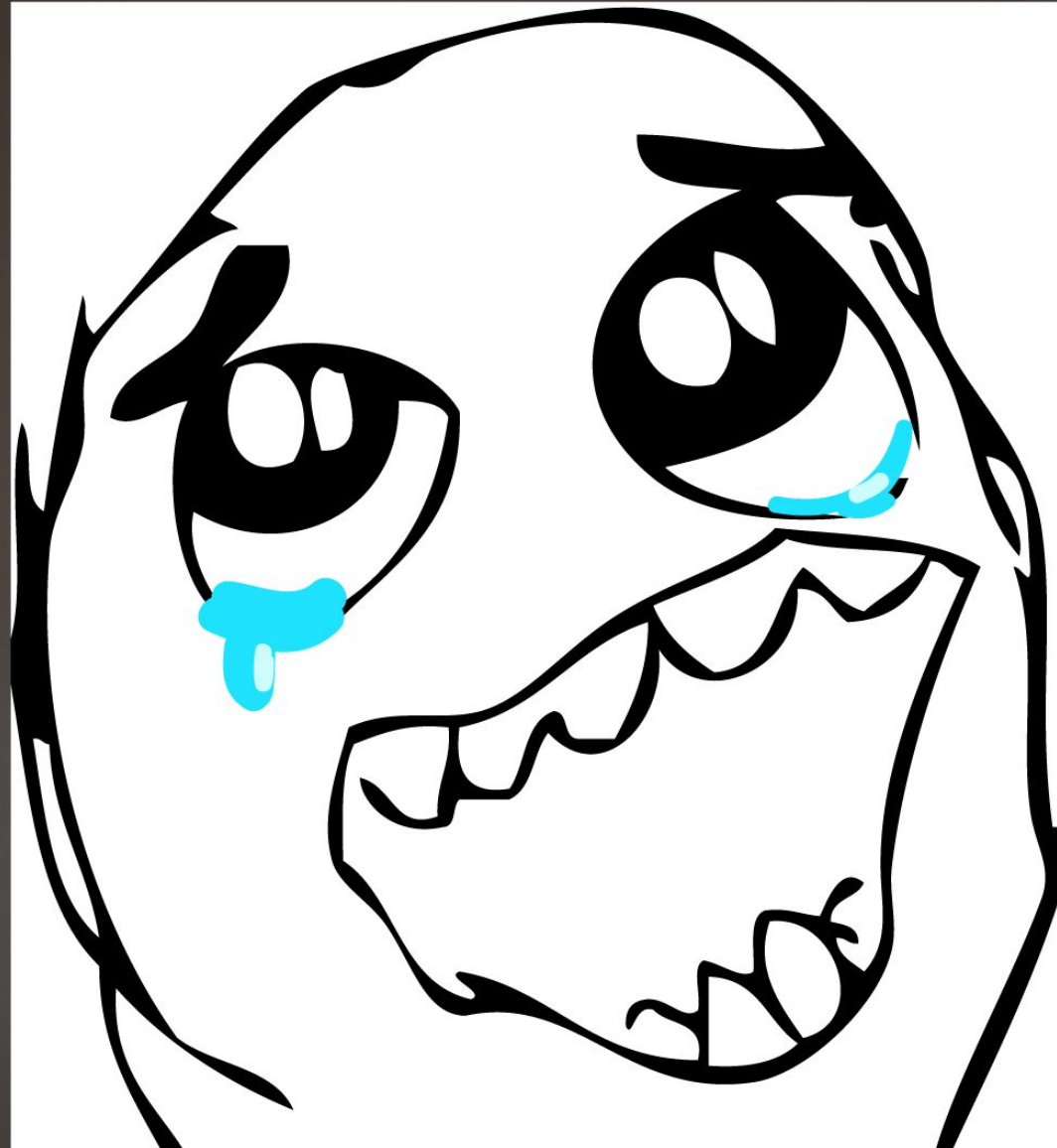
# จากเดิม

```
ReactDOM.render(  
  React.createElement('div', null, React.createElement('h1', null, 'Hello  
React'),  
    React.createElement('p', null, 'this is paragraph',  
      React.createElement('span', null, 'this is span'),  
      React.createElement('b', null, React.createElement('span', null, 'this  
is icon'))  
    ), React.createElement('button', null, 'Click me!')  
  ),  
  document.getElementById('root')  
);
```



# เขียนด้วย JSX

```
ReactDOM.render(  
  <div>  
    <h1>Hello React</h1>  
    <p>this is paragraph  
      <span>this is span</span>  
      <b><span>this is icon</span></b>  
    </p>  
    <button>Click me!</button>  
  </div>,  
  document.getElementById('root')  
) ;
```



# ข้อควรระวังเมื่อใช้ JSX

---

- เราสามารถ render jsx ได้เพียงแค่ 1 top level เท่านั้น
- Tag ที่ไม่มี Children จะต้องเขียนแบบ Self-Closing Tag





# ตัวอย่างที่ผิด

```
ReactDOM.render(  
  <h1>Hello world</h1>  
  <button>Click me!</button>,  
  document.getElementById('root')  
) ;
```



# ตัวอย่างที่ถูกต้อง

```
ReactDOM.render(  
  <div>  
    <h1>Hello world</h1>  
    <button>Click me!</button>  
  </div>,  
  document.getElementById('root')  
) ;
```





## ตัวอย่างที่ผิด

`<br>`

`<hr>`

`<img>`

## ตัวอย่างที่ถูกต้อง

`<br />`

`<hr />`

`<img />`

# Javascript Expression in JSX



# Javascript Expression in JSX

---

เราสามารถเขียน javascript ลงไปใน JSX ได้ โดยใช้เครื่องหมาย `{ }` ล้อมรอบ Javascript syntax

คำสั่งที่ใช้ได้มีดังนี้

- แทนค่าด้วยตัวแปร
- conditional (ternary) operator (If แบบย่อ)
- การคำนวณ
- เรียกใช้ function



# แทนค่าด้วยตัวแปร

ดูได้ใน variable-in-jsx.html



```
const msg = 'hello world!';  
const arr = ['arr1', 'arr2', 'arr3'];  
<div>  
  <div>{msg}</div>  
  <div>{arr}</div>  
</div>
```



false, null, undefined, และ true สามารถใช้ได้ แต่แค่ไม่ถูก Render ออกมาแสดงเป็น UI ทำให้ทุกคำสั่งด้านล่างนี้แสดงผลออกมาเหมือนกัน

```
<div />
```

```
<div></div>
```

```
<div>{false}</div>
```

```
<div>{null}</div>
```

```
<div>{undefined}</div>
```

```
<div>{true}</div>
```



```
const a = true;  
<div>{a ? 'this is true' : 'this is false'}</div>
```

```
const showMsg = true;  
const msg = 'hello world!';  
<div>{showMsg && msg}</div>
```





```
const showMsg = [];  
const msg = 'hello world!';  
<div>{showMsg.length && msg}</div>
```

ถึงแม้ว่า showMsg จะเป็น array ที่มีค่า length เป็น 0 ก็ตาม แต่ผลลัพธ์ที่แสดงออกมาจริงๆคือ 0  
จึงต้องแก้ไขด้วยการเขียน if ด้วยวิธีด้านล่าง

```
<div>{showMsg.length > 0 && msg}</div>
```

หาความรู้เพิ่มเติมได้ที่

<https://stackoverflow.com/questions/4490274/returning-with>

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Logical\\_Operators](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Logical_Operators)



```
<div>result of 2 + 2: {2 + 2}</div>
```

```
<div>result of 2 - 2: {2 - 2}</div>
```

```
<div>result of 2 ÷ 2: {2 / 2}</div>
```

```
<div>result of 2 x 2: {2 * 2}</div>
```

# เรียกใช้ Function

ดูได้ใน call-function-in-jsx.html



```
const title = 'cLiCk me';  
<button>{title.toUpperCase()}</button>
```

# Styling in JSX



# styling in JSX วิธีที่ 1: CSS Stylesheet

---

เราสามารถ **import** หรือใช้ **<link>** เพื่อโหลดไฟล์ css เข้ามาใช้ style element ของเราได้

โดยเราจะใช้คำว่า **className** แทนคำว่า class ในการกำหนดว่า element แต่ element จะเรียกใช้ class ของ css อันไหน



# CSS Stylesheet

ดูได้ใน css-styles-sheet.html



styles.css

```
.title {  
  font-size: 50px;  
  color: teal;  
}
```

```
div {  
  background: black;  
}
```

<div>

<p>Hello this is

<span className="title">

CSS style sheet

</span>

</p>

</div>



# styling in JSX วิธีที่ 2: Inline Styling

---

เราสามารถกำหนด style ใน tag ได้แบบเดียวกับตอนเราเขียน style ใน html แต่ว่าเราต้องใช้ `{ }` ล้อมรอบ object `{ }` ที่กำหนด style และต้องเขียน style ด้วย convention ของ javascript เช่น

font-size -> fontSize, background-color -> backgroundColor



# Inline styling

ดูได้ใน css-inline-style.html



```
<div style={{ backgroundColor: 'black' }}>
  <p>Hello this is
    <span style={{ fontSize: 50, color: 'teal' }}>
      CSS style sheet
    </span>
  </p>
</div>
```

# Lab 2: React with JSX

---

เปิดไฟล์ lab2.html ขึ้นมาแล้วลองทำสิ่งต่อไปนี้ด้วย JSX

- สร้าง / แก้ไข element
- สร้าง / แก้ไข style
- ทดลองใช้ Javascript Expression



# React Component



# React Component

---

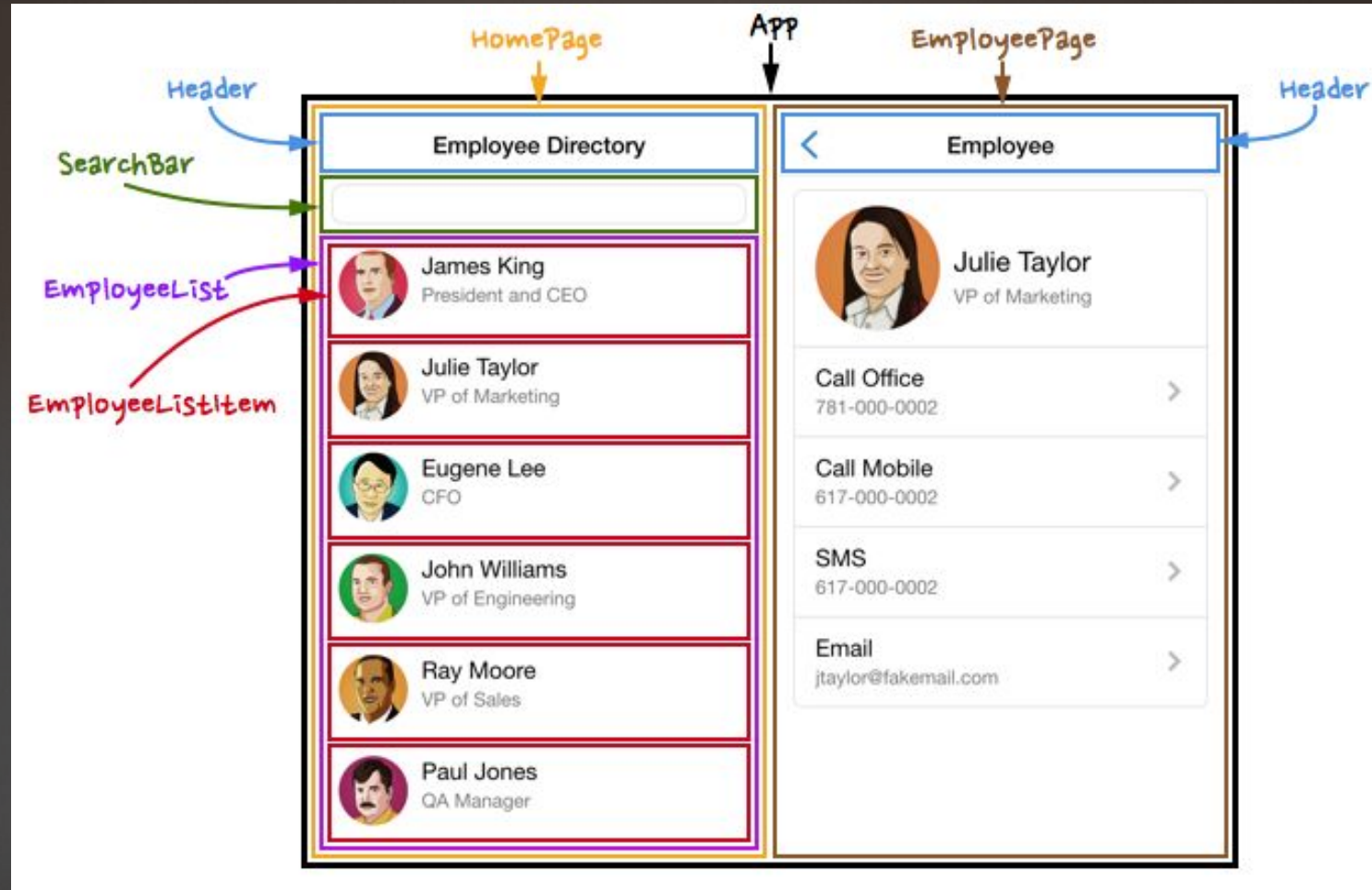
Component ใน React คือแนวคิดการแบ่ง UI ของเว็บออกมาเป็นส่วนๆ ภายใน Component มี logic ของตัวเอง ทำให้สามารถนำมาใช้ใหม่ (Reuse) ได้ทั้งเว็บ

ในทาง Javascript Component ก็เป็นแค่ Function หรือ Class ที่ return React Element ออกมา





# React Component



<http://coenraets.org/blog/wp-content/uploads/2014/12/uimockscript.png>



# การสร้าง Component แบบ Function

ดูได้ใน function-component.html



```
const Hello = () => {  
  return (<div>Hello React!</div>);  
}
```

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

# การสร้าง Component แบบ Class

ดูได้ใน class-component.html

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

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

# Nested Component

ดูได้ใน nested-component.html




```
const MyTitle = () => {  
  return (  
    <h1>  
      Hello React!  
    </h1>  
  );  
}
```

```
class Hello extends React.Component {  
  render() {  
    return (  
      <div>  
        <MyTitle />  
        <p>this is paragraph</p>  
      </div>  
    );  
  }  
}  
  
ReactDOM.render(<Hello />,  
  document.getElementById('root'));
```

# การ Loop สร้าง Component / Element

ดูได้ใน `loop-component.html`



```
const MyTitle = () => (<h1>Hello React!</h1>);  
class Hello extends React.Component {  
  render() {  
    const msgs = new Array(4).fill().map((_, i) => (  
      <MyTitle key={i} />  
    ))  
    ;  
    return (<div>{msgs}</div>);  
  }  
}  
  
ReactDOM.render(Hello, document.getElementById('root'));
```

# Key ใน Component / Element

---

เวลาเรา Loop สร้าง Component / Element นั้น React ใช้ attribute key ในการจำ children เก่ากับใหม่ ทำให้ React รู้ว่ามี children ไหนเกิดการเปลี่ยนแปลงบ้าง ดังนั้นเราจึงควรใส่ attribute key ทุกครั้งเวลา Loop สร้าง Component / Element



# ข้อควรระวังตอนสร้าง Component

---

- Component ที่เราสร้างเองต้องตั้งชื่อขึ้นต้นด้วยตัวอักษรตัวใหญ่เท่านั้น



# Lab 3: React Component

---

เปิดไฟล์ lab3.html ขึ้นมาแล้วลอง Refactor ด้วยการสร้าง Component ที่เพิ่งได้เรียนไปดู





# Tools



# Yarn

---

คือ package manager เหมือนกับ npm แต่มีจุดเด่นตรงที่ yarn จะ cache package ที่เราเคยดาวน์โหลดเอาไว้ ทำให้ไม่ต้องดาวน์โหลดใหม่ จึงมีความเร็วกว่า npm

Download link : <https://yarnpkg.com/>



# คำสั่ง Yarn vs Npm



```
npm install  
yarn
```

```
npm install --save [package]  
yarn add [package]
```

```
npm install -g [package]  
yarn global add [package]
```

```
npm uninstall [package]  
yarn remove [package]
```

```
npm start  
yarn start
```

```
npm run dev  
yarn dev
```

# Create-react-app

---

คือ Boilerplate (Template) สำหรับสร้าง React App  
ทำให้เราไม่ต้องมา config webpack, babel หรือสร้าง folder structure เอง หากนักภาพไม่ออกให้นึกถึงบะหมี่กึ่งสำเร็จรูปแบบกระป๋อง



# วิธีลง Create-react-app

 เปิด terminal แล้วพิมพ์

```
npm install -g create-react-app
```

หรือ

```
yarn global add create-react-app
```

# การสร้าง React App ด้วย Create-react-app

● ● ● เปิด terminal ในโฟลเดอร์ที่จะสร้างโปรเจกต์แล้วพิมพ์

`create-react-app` [ชื่อโปรเจกต์ด้วยตัวพิมพ์เล็กทั้งหมด]

หลังจากเสร็จแล้วให้เข้าไปในโฟลเดอร์โปรเจกต์  
พิมพ์

`npm start`

หรือ

`yarn start`

เพื่อรัน React App

# React App Folder structure



```
my-app
├── README.md
├── node_modules
├── package.json
├── .gitignore
├── public
│   ├── favicon.ico
│   ├── index.html
│   └── manifest.json
└── src
    ├── App.css
    ├── App.js
    ├── App.test.js
    ├── index.css
    ├── index.js
    ├── logo.svg
    └── registerServiceWorker.js
```



# React Developer Tools

---

ใน Chrome หรือ Firefox จะมี Extension ที่ชื่อว่า React Developer Tools เอาไว้สำหรับ Debug React App ที่เราสร้างขึ้นมา ตามโค้ดได้ที

Chrome:

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

FireFox:

<https://addons.mozilla.org/en-US/firefox/addon/react-devtools/>



# Lab 4: React App

---

สร้างโปรเจกต์ด้วย create-react-app แล้วลองศึกษา  
โครงสร้างของ app ว่าอะไรอยู่ตรงไหนบ้าง  
ลองสร้าง component ขึ้นมาแล้วเรียกใช้ดู



# การบ้าน #1 - สร้าง Component

---

- ใช้ Create React App ในการสร้าง Project ขึ้นมาก่อน
- หัดสร้าง Component (ทั้งแบบ Class, Function) จาก 0 หลากๆรอบจนจำ pattern ได้
- หลังจากนั้นให้เรียก TA มา แล้วสร้าง Component ใหม่ (ทั้งแบบ Class, Function) จาก 0 โดยไม่เปิดดูวิธีการสร้างจากที่อื่นเลย หากรันแล้วมี Component นี้แสดงออกมา จึงจะให้ผ่าน

\*ห้าม Copy & Paste, ห้ามใช้ Extension ช่วย Gen ออกมา ให้เขียนสดทุกตัวนะครับทุกคน

\*ไม่จำกัดความคิดสร้างสรรค์ อยากสร้าง Component รูปร่างหน้าตาแบบไหน ใช้ทำอะไร ตามใจเลยครับ



# การบ้าน #2 - สร้าง Random Box

- ใช้ Create React App ในการสร้าง Project ขึ้นมาแล้วลบ template เดิมทิ้ง
- สร้าง Component ชื่อ RandomBox ให้ได้หน้าตาแบบนี้
- ให้ **สีกล่อง** และ **ขนาดของ font** ภายใน RandomBox จะต้องถูกสุ่มขึ้นมาจากข้อกำหนดดังนี้
- สี: ['red', 'blue', 'green', 'purple', 'pink']
- ขนาด font : 20-40px
- ทุกครั้งที่ refresh page ใหม่ สีของกล่อง และขนาด font จะต้องเปลี่ยนไปเรื่อยๆ

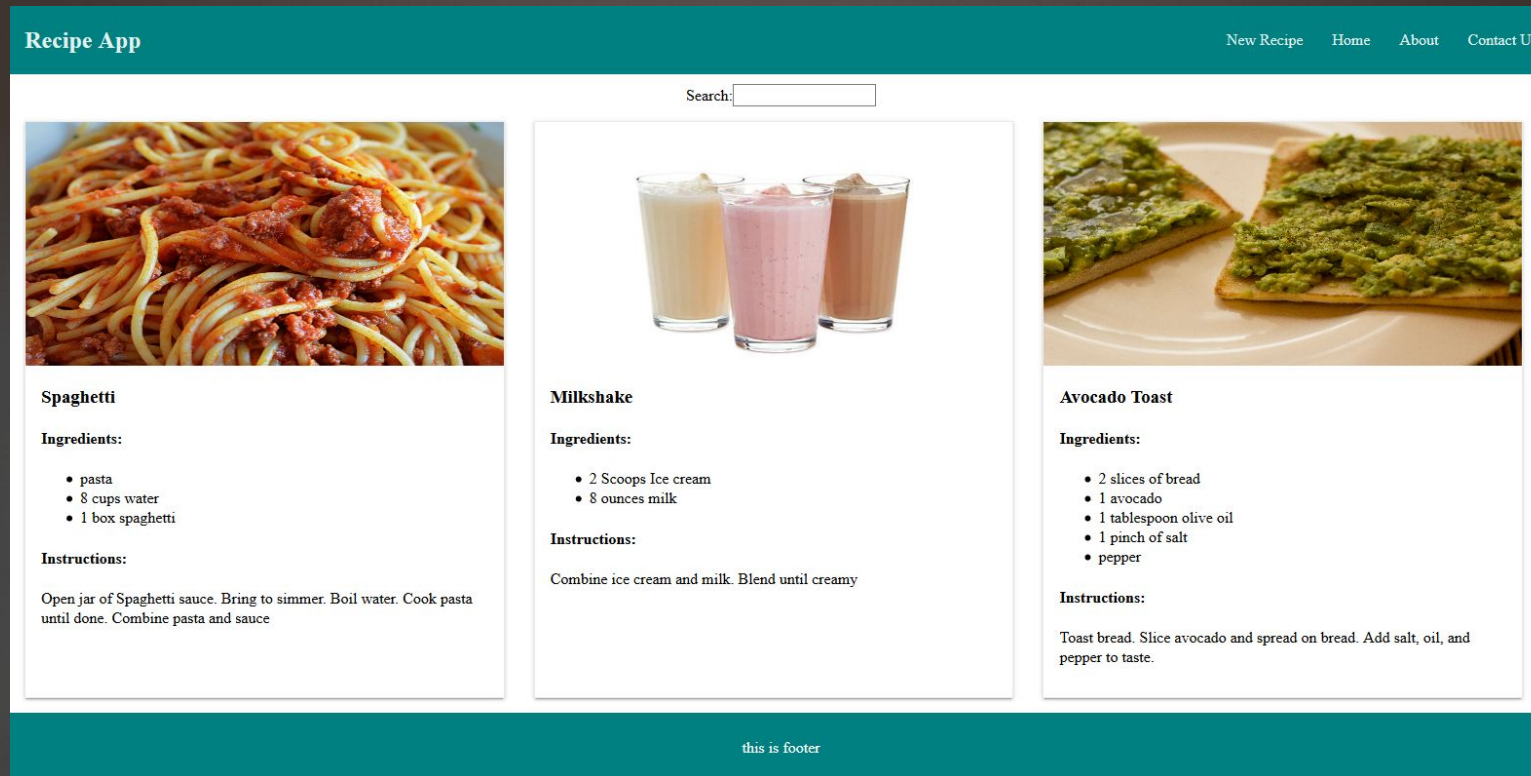


Random function: [https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Math/random](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random)



# การบ้าน #3 - สร้าง Recipe App

- สร้าง Recipe App ให้มีองค์ประกอบเหมือนตามภาพต่อไปนี้ (สวยงามนี้ก็ได้นะ)



ดูภาพขนาดใหญ่ได้ที่ <https://ibb.co/cUfA5H>



# การบ้าน #4 - Clone your favorite web app

---

- การบ้านนี้เป็น Optional ทำหรือไม่ทำก็ได้
- เลือก Clone หน้าเว็บที่คุณชอบมาสักหนึ่งเว็บ ด้วยความรู้วันนี้ที่ได้เรียนไป



# สำหรับหาความรู้เพิ่มเติม

React Element, React Component, Instance of React Component

[https://medium.com/@fay\\_jai/react-elements-vs-react-components-vs-component-backing-instances-14d42729f62](https://medium.com/@fay_jai/react-elements-vs-react-components-vs-component-backing-instances-14d42729f62)

Understanding unique keys for array children in React.js

<https://stackoverflow.com/questions/28329382/understanding-unique-keys-for-array-children-in-react-js>

