

# React

---

Nuttachai Kulthammanit

# คอร์สนี้ เหมาะกับใคร

---

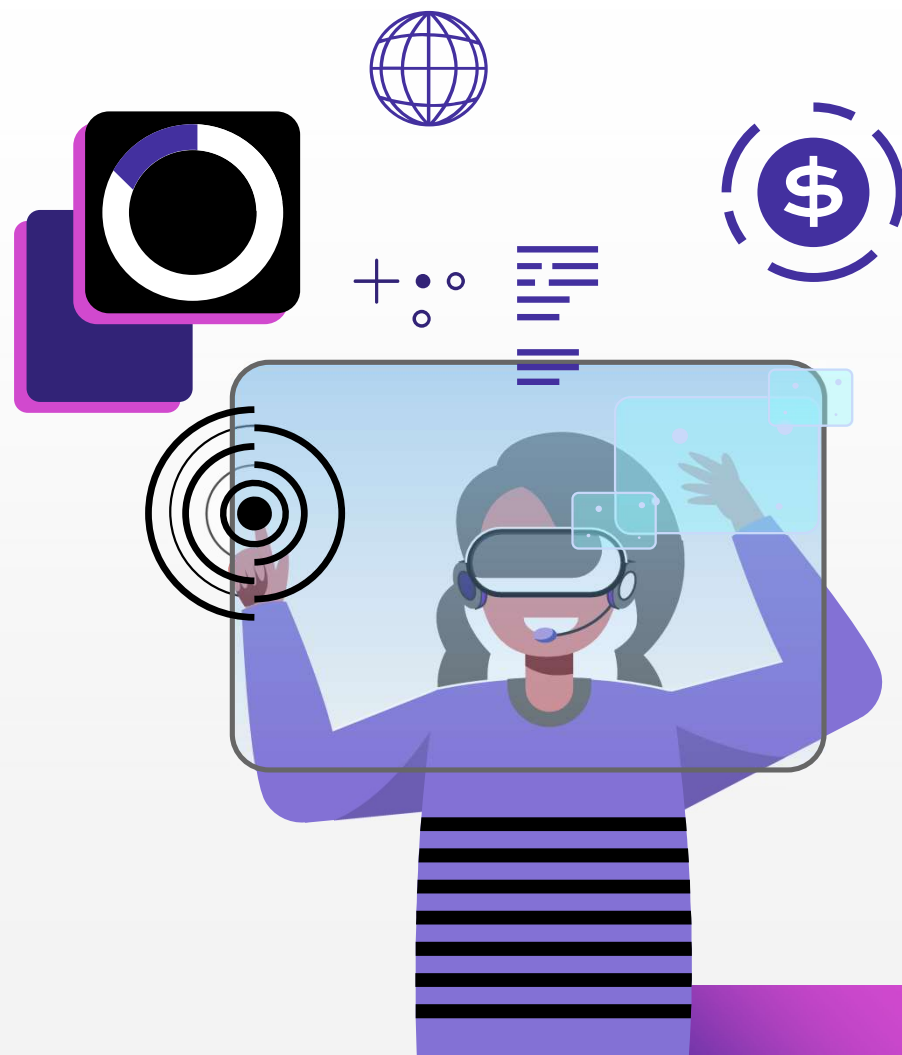


## คอร์สนี้เหมาะกับใคร

1. คนที่เป็น Developer ที่เขียนเฟรมเวิร์กอื่นมาก่อน เช่น Vue.js หรือ Angular
2. คนที่หัดเขียนโปรแกรม แต่มีความรู้ JavaScript, HTML และ CSS

หัวข้อที่ต้องรู้  
ก่อนเรียนคอร์สนี้

---



# หัวข้อที่ต้องรู้ก่อนเรียนคอร์สนี้

## HTML/CSS

### DOM คืออะไร

- DOM คืออะไร
- รู้วิธีการจัดการ DOM เบื้องต้น
- Event คืออะไร

### CSS Properties

- เคยใช้ CSS ในการแต่งหน้าเว็บ HTML มาบ้าง

# หัวข้อที่ต้องรู้ก่อนเรียนคอร์สนี้

## JavaScript

### Declaring variables (การประกาศตัวแปร)

- Let กับ Const

### Function creations (การสร้างฟังก์ชัน)

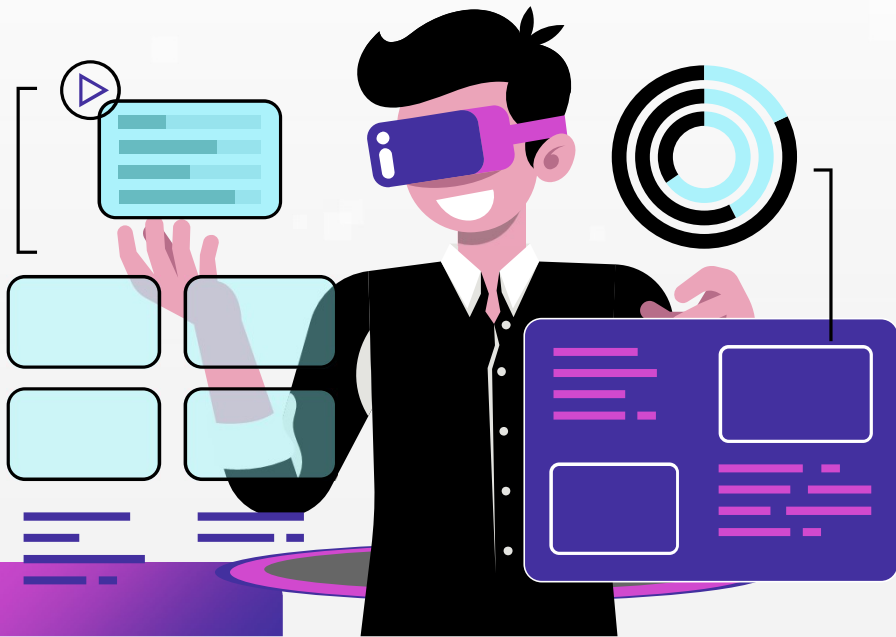
- Function declaration
- การ Return ของ Function
- Arrow Functions

### Object และ Array

- Destructuring Object
- Spread Operator

# React คืออะไร ?

---



# React คืออะไร?

Library สำหรับสร้าง User Interface  
ของภาษา JavaScript

**React**

A JavaScript library for building user interfaces

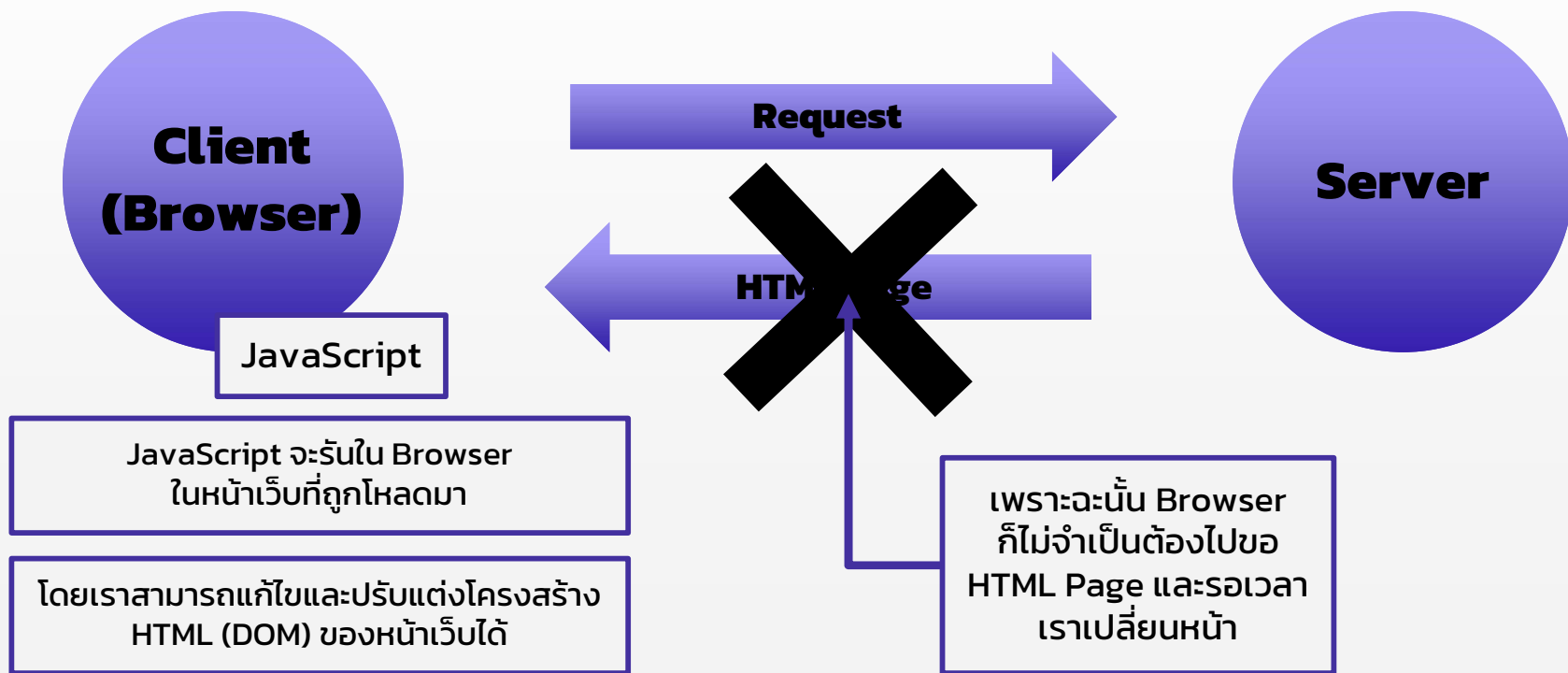


# React คืออะไร?

## ย้อนกลับไปสมัยก่อน

- สมัยก่อนเว็บไซต์เวลาเรากดไปหน้าใหม่ เช่น จากหน้า Home ไปหน้า Shopping เว็บไซต์ก็จะไปโหลดหน้า Shopping ใหม่มาจาก Server
- แต่ JavaScript จะมาช่วยเรา ลักษณะคล้ายกับแอปมือถือ คือเวลาเรากดไปหน้าใหม่มันไม่จำเป็นต้องไปโหลดใหม่มาจาก Server เพราะว่ามันโหลดทุกอย่างมาก่อนแล้วจึงทำให้ผู้ใช้รู้สึกว่ามันเร็ว

# React คืออะไร?



# React คืออะไร?

- เป็น Client-Side JavaScript Library
- การเป็น Library สำหรับการเขียนเว็บไซต์สมัยใหม่ ที่ช่วยให้ UI ที่สามารถโต้ตอบกับผู้ใช้ได้ (Reactive)
- ในเว็บปกติทั่ว ๆ ไปที่ไม่ได้ซับซ้อนอะไรมากมาย การใช้ JavaScript อย่างเดียวก็น่าจะเป็นสิ่งที่โอเคและทำได้ง่าย แต่ในกรณีเว็บที่ซับซ้อนมากขึ้นการใช้ JavaScript ในการแก้ไข (Manipulate) DOM จะทำให้โค้ดซับซ้อนและเขียนยากขึ้นไปด้วย
- เว็บที่สร้างด้วย React จะเป็น Single-Page-Application (SPAs)

# React คืออะไร?

## Single Page Application (SPAs)

- React จะสามารถใช้ควบคุมส่วนต่าง ๆ ของ HTML ในหน้าเว็บได้ทั้งหมด
- ในเว็บที่เป็น multi-page-application จะมีบางหน้าที่ต้อง Render และโหลดมาจาก Backend
- ส่วนใน Single Page Application ตัว Backend (server) จะส่งหน้าเว็บ HTML มาแค่ครั้งแรกรั้งเดียว หลังจากนั้นไม่ว่าเราจะกดไปหน้าไหน React ก็จะจัดการทั้งหมดหลังจากนั้นโดยไม่ต้องไปโหลดมาจาก Backend อีกรอบ

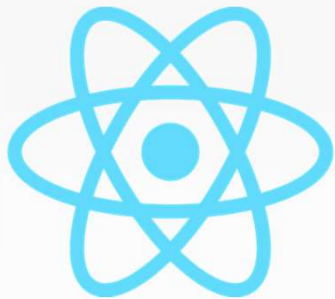
# ตัวเลือกอื่นนอกจาก React



## Angular

เป็นเฟรมเวิร์กที่เป็น Component-base UI เต็มตัวมี Features หลาย ๆ ตัวมากับเฟรมเวิร์ก และใช้ TypeScript ในการเขียน ไม่ค่อยยุ่งในการทำโปรเจกเล็ก ๆ

# ตัวเลือกอื่นนอกจาก React



## **React.js**

เป็นไลบรารีที่เป็น Component-base UI ที่เน้นความเบา  
เพราะฉะนั้นบาง Feature ต้องลงเพิ่มเช่น Routing

# ตัวเลือกอื่นนอกจาก React

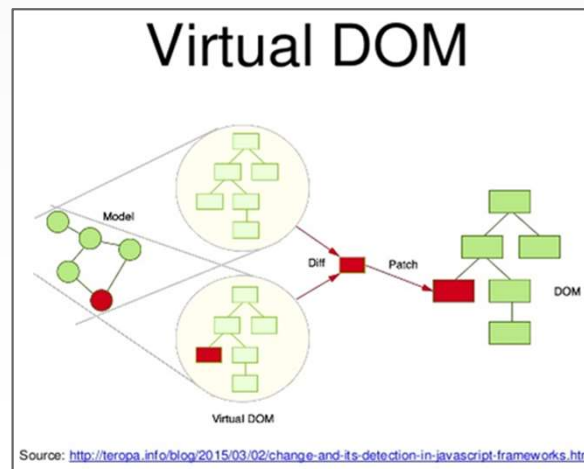


## Vue.js

เป็นเฟรมเวิร์กที่เป็น Component-base UI เต็มตัวมี Features หลาย ๆ ตัวมากับเฟรมเวิร์กเหมือนกันแต่มีความนิยมน้อยกว่า React กับ Angular

# ข้อดีของ React (1) – มี Virtual DOM

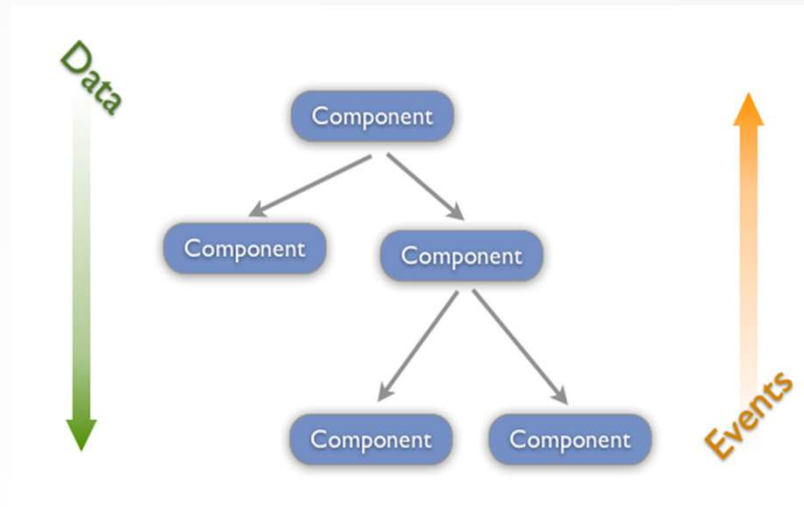
ใช้ Virtual DOM เป็นตัวเทียบว่าต้องไปแก้ไข DOM ตัวไหนใหม่บ้าง React จะไม่เข้าไปแก้ไข DOM ถ้าไม่จำเป็น แต่จะหาความแตกต่างระหว่าง Dom ของจริงและ Virtual Dom จากนั้นค่อยไปแก้ไข DOM เฉพาะตัวที่มีการเปลี่ยนแปลง





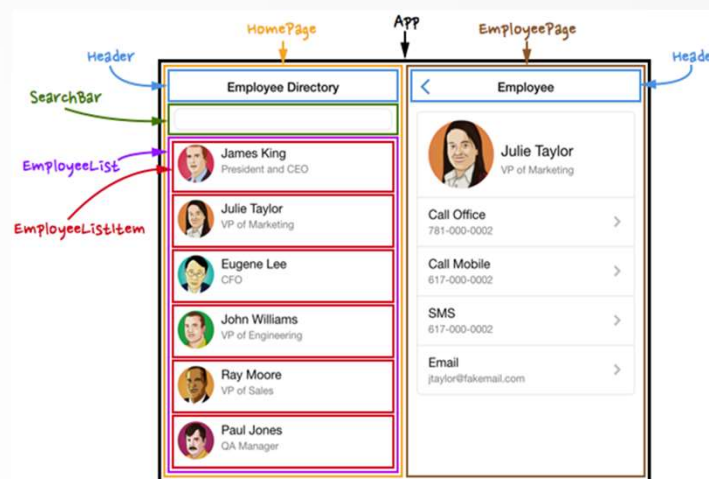
## ข้อดีของ React (2) – ข้อมูลถูกส่งลงล่าง

- one-way binding หรือ one-way data flow
- ข้อมูลจะถูกส่งจากตัวแม่ไปตัวลูกเท่านั้น (ลงด้านล่างเท่านั้น)



## ข้อดีของ React (3) – Components base

การทำเป็น Component ทำให้เราสามารถนำมาใช้ซ้ำ ยกตัวอย่าง เช่น สมมติเราสร้าง Header ขึ้นมา และนำไปใช้ในหน้า Home เราก็สามารถนำ Header ไปใช้ในหน้าอื่นได้ เช่น หน้า About



## ข้อดีของ React (4) – Learn once, Write Everywhere

- ตัว React สามารถนำไปประยุกต์เขียนได้หลาย Platform เช่น การที่เราเขียนเว็บจาก React ได้
- เราก็สามารถเขียน React Native ที่เป็นการสร้าง Mobile Application ทั้ง Android และ IOS
- และเราก็สามารถเขียน Electron ที่เป็น Desktop App ที่รันได้บนทุก OS





# การ Setup สำหรับ การเขียน React

---

# การ Setup สำหรับการเขียน React

วิธีสร้าง React App มีหลายวิธี เช่น create-react-app แต่ในคอร์สนี้เราจะใช้ vite ช่วยเราในการสร้าง React App

- npx create-react-app
- **npm create vite**

# การ Setup สำหรับการเขียน React

**Mini Workshop:** การ Setup สำหรับการเขียน React

Note: Lab-r1-l1



# การ Setup สำหรับการเขียน React

**Mini Workshop:** การ Setup โปรแกรม React

Note: Lab-r1-l2





# การ Setup สำหรับการเขียน React

**Mini Workshop:** อธิบายโครงสร้างของ React

Note: Lab-r1-l3





# อีลิเมนต์ (Element)

---



# อีลิเมนต์ (Element) คืออะไร?

พวกปุ่มหรือสิ่งต่าง ๆ ที่เราเห็นกันบนหน้าเว็บที่สร้างด้วย React เกิดจาก

**React createElement**

**ReactDOM.render()**

ซึ่ง Function นี้จะทำให้เกิด React Element เกิดขึ้น

# อีลีเมนต์ (Element)

- `React.createElement()` จะทำให้เกิด `ReactElement` ขึ้น
- Element หน้าตาเหมือนกับ tag ใน HTML
- Element เป็นข้อมูลประเภท Obj รรรมดา
- `ReactDOM.render()` จะนำ Element ที่เป็น Obj มาแสดงผลที่ DOM
- `ReactDOM.render()` จะทำการจับคู่ Element กับ tag ใน DOM ที่จะทำการแสดงผล

## 2.1 อิลิเมนต์ (Element)

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

หน้าตาของ element

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

หน้าตาของ root DOM ใน HTML ทุกอย่างใน tag นี้จะถูกจัดการด้วย React DOM

```
ReactDOM.render(element, document.getElementById('root'));
```

ReactDOM.render จะจับคู่ Element และ root DOM ใน HTML

**React** createElement



React Element

# อีลีเมนต์ (Element) คืออะไร?

ซึ่งอีลีเมนต์ก็คือสิ่งที่ใช้ในการกำหนดว่าหน้าเว็บว่าจะมีหน้าตาเป็นอย่างไรคล้าย ๆ กับ HTML Tag

```
React.createElement(type, props, children)
```

`type` คือ ชื่อของ tag

`props` คือ เหมือน attribute ของ tag และ เป็นข้อมูลประเภท Obj

`children` คือ ข้อมูลภายใน tag

# อีลีเมนต์ (Element) คืออะไร?

```
React.createElement(type, props, children)
```

ข้อกำหนดของ **type**

ถ้าชื่อ **type** เหมือน tag HTML ให้ใช้ตัวอักษรอังกฤษตัวพิมพ์เล็ก

ถ้าชื่อ **type** ไม่มีใน tag HTML ให้ขึ้นต้นด้วยตัวอักษรอังกฤษตัวพิมพ์ใหญ่

# อีลีเมนต์ (Element) คืออะไร?

```
React.createElement(type, props, children)
```

## ข้อกำหนดของ **props**

- **Props** เป็นข้อมูลประเภท Object
- key ของ Object เปรียบเสมือน attribute ของ tag



# อีลีเมนต์ (Element) คืออะไร?

```
React.createElement(type, props, children)
```

ข้อกำหนดของ `children`

- สามารถเป็น number, string , และ React.createElement() ได้
- ข้อมูลที่อยู่ใน `children` จะถูกแสดงผลออกมาบนหน้าเว็บ

# อีลีเมนต์ (Element)

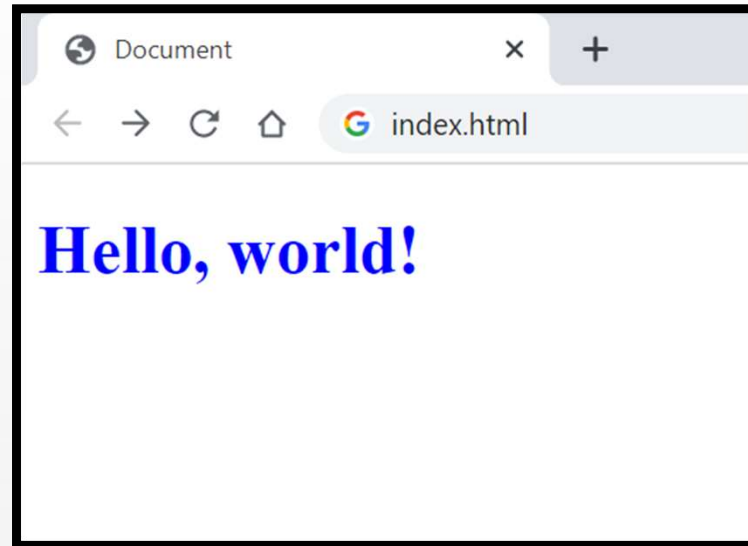
```
React.createElement(  
  'h1',  
  {style:{color: 'blue'}},  
  "Hello, world!"  
)
```



```
<h1 style="color:blue">Hello, world!</h1>
```

React.createElement(type, props, children)

# อีลีเมนต์ (Element)



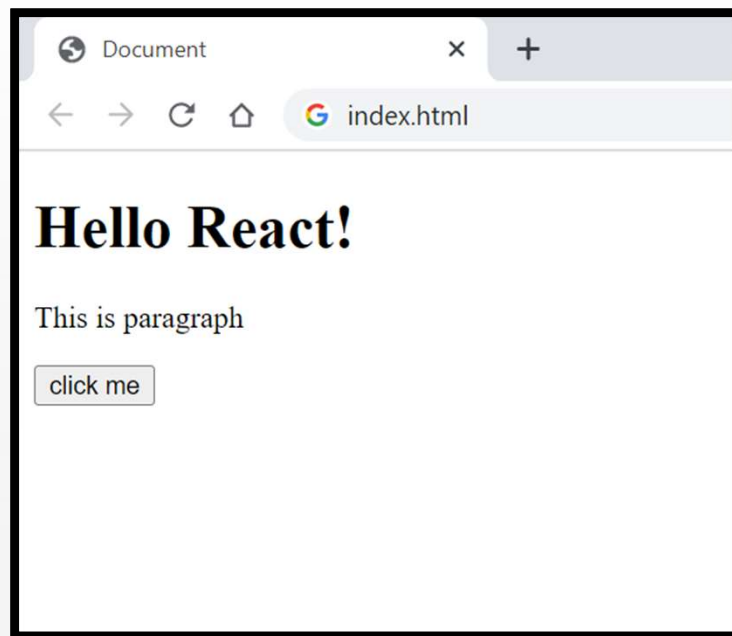
# อีลีเมนต์ (Element)

```
React.createElement("div", null,  
  React.createElement("h1", null, "Hello React!"),  
  React.createElement("p", null, "This is paragraph"),  
  React.createElement("button", null, "click me")  
);
```



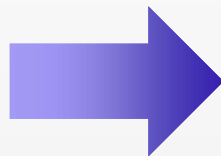
```
<div>  
  <h1>Hello React!</h1>  
  <p>This is paragraph</p>  
  <button>click me</button>  
</div>
```

# อีลีเมนต์ (Element)



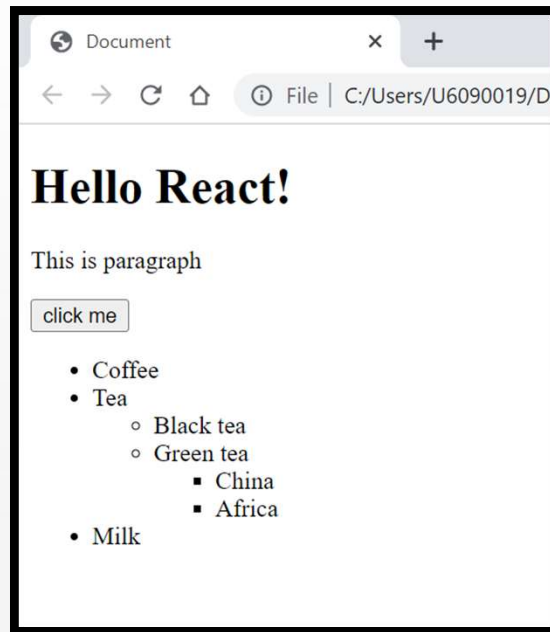
# อีลีเมนต์ (Element)

```
<div>
  <h1>Hello React!</h1>
  <p>This is paragraph</p>
  <button>click me</button>
  <ul>
    <li>Coffee</li>
    <li>Tea
      <ul>
        <li>Black tea</li>
        <li>Green tea
          <ul>
            <li>China</li>
            <li>Africa</li>
          </ul>
        </li>
      </ul>
    </li>
    <li>Milk</li>
  </ul>
</div>
```



```
React.createElement("div", null,
  React.createElement("h1", null, "Hello React!"),
  React.createElement("p", null, "This is paragraph"),
  React.createElement("button", null, "click me"),
  React.createElement("ul", null,
    React.createElement("li", null, "Coffee"),
    React.createElement("li", null, "Tea",
      React.createElement("ul", null,
        React.createElement("li", null, "Black tea"),
        React.createElement("li", null, "Green tea",
          React.createElement("ul", null,
            React.createElement("li", null, "China"),
            React.createElement("li", null, "Africa")
          )
        )
      )
    ),
    React.createElement("li", null, "Milk")
  )
);
```

# อีลีเมนต์ (Element)



# อีลีเมนต์ (Element)

## Mini Workshop: อีลีเมนต์ (Element)

Note: <https://github.com/soncomqiq/react-with-create-elements>



# อีลิเมนต์ (Element)

## Mini-Lab

1. สร้าง element h1 ขึ้นมาโดยมีคำว่า Resume อยู่ข้างหน้า
2. ปรับแต่งให้คำว่า Resume เป็นสีเขียว
3. สร้าง element h2 ขึ้นมาโดยมีชื่อตัวเองอยู่ในนั้น
4. สร้าง element p ขึ้นมาโดยใส่
  - 4.1 สีที่ชอบ
  - 4.2 ความสูง
  - 4.3 น้ำหนัก
  - 4.4 คำคมที่ชอบ

# JSX คืออะไร

---



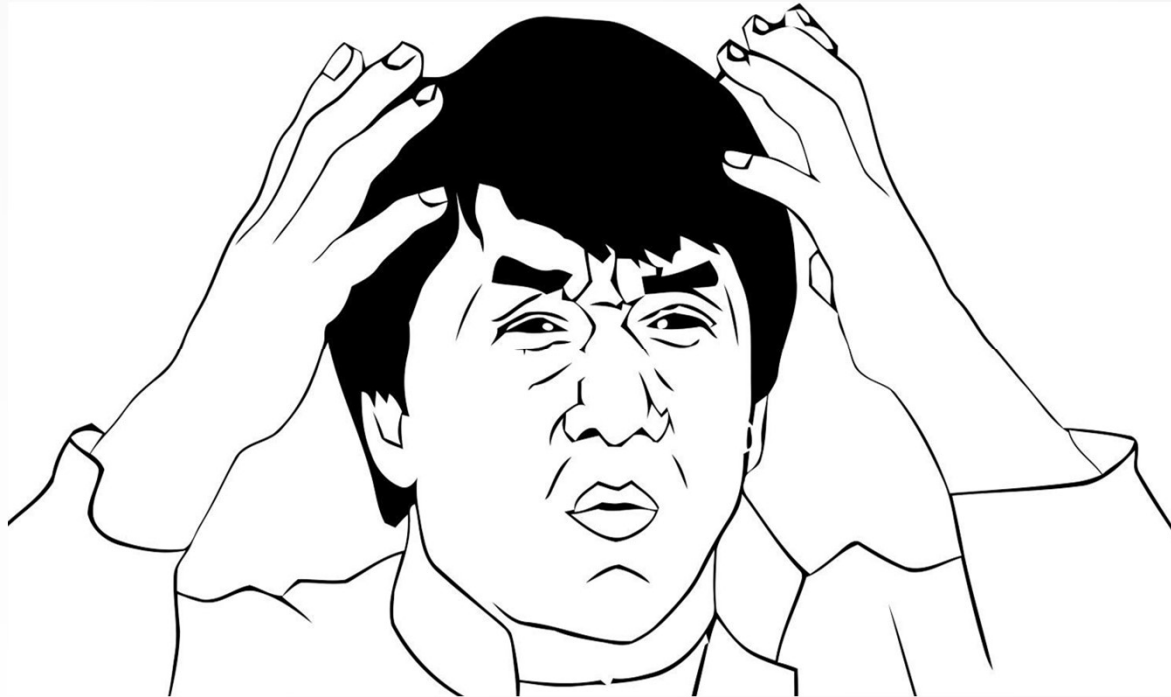
# JSX คืออะไร

## ปัญหา

```
React.createElement("div", null,
  React.createElement("h1", null, "Hello React!"),
  React.createElement("p", null, "This is paragraph"),
  React.createElement("button", null, "click me"),
  React.createElement("ul", null,
    React.createElement("li", null, "Coffee"),
    React.createElement("li", null, "Tea",
      React.createElement("ul", null,
        React.createElement("li", null, "Black tea"),
        React.createElement("li", null, "Green tea",
          React.createElement("ul", null,
            React.createElement("li", null, "China"),
            React.createElement("li", null, "Africa")
          )
        )
      )
    )
  ),
  React.createElement("li", null, "Milk")
);
```

# JSX คืออะไร

ปัญหา



# JSX คืออะไร

- JSX ย่อมาจาก JavaScript Syntax Extension
- จะใช้สำหรับสร้าง UI ของ React แทน React.createElement
- JSX คือ syntax ที่รวม **JavaScript** กับ **XML** เข้าด้วยกัน ที่หน้าตาคล้าย HTML โดยปกติแล้วต้องใช้ควบคู่กับ **Transpiler** เช่น **Babel** ซึ่งจะทำให้เราเขียนได้ง่ายขึ้น
- JSX สามารถเขียน JavaScript expression ภายในส่วนต่าง ๆ ของ element ได้

# JSX คืออะไร

ตัวอย่าง

```
React.createElement("div", { className: "App" },  
  React.createElement("div", null, "element ภายใน"),  
  React.createElement("div", null, "element ภายใน")  
);
```



```
<div className="App">  
  <div>  
    element ภายใน  
  </div>  
  <div>  
    element ภายใน  
  </div>  
</div>
```

# JSX คืออะไร

## ข้อกำหนด

- JSX ต้องเป็นแท็กปิดเสมอ เช่น ใน HTML จะเขียนแบบนี้ได้ `<br>` แต่ใน JSX จะต้องเป็น `<br/>`
- ถ้าเป็นแท็กเปิดต้องมีแท็กปิดเสมอ เช่น `<Content>JSX</Content>`
- การ return JSX จะ return แค่ Element เดียวเท่านั้น
- สามารถใช้ tag แบบไม่มีชื่อ `<> </>` ได้ด้วย แต่เมื่อ render จะไม่เห็น tag นี้
- การใช้ JSX ต้องอยู่ใน scope ของ react (ต้องมี import "React" from "react" สำหรับเวอร์ชันเก่า)

# JSX คืออะไร

## ข้อกำหนด

- จะใช้ className แทน class ในการกำหนดคลาสของ CSS

### HTML

```
<div class="App">  
    เนื้อหาที่แสดงใน App Component  
</div>
```

### JSX

```
<div className="App">  
    เนื้อหาที่แสดงใน App Component  
</div>
```



# JSX คืออะไร

## ข้อกำหนด

- แท็ก Label จะใช้ htmlFor แทน for

### HTML

```
<label for="name">  
  Name  
</label>
```

### JSX

```
<label htmlFor="name">  
  Name  
</label>
```

# JSX คืออะไร

## ข้อกำหนด

- Attribute ของ event จะใช้ camelCase จากที่ใช้ตัวเล็กทั้งหมด

### HTML

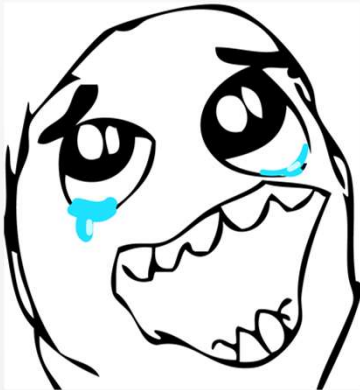
```
<label onclick="name">  
  Name  
</label>
```

### JSX

```
<label onClick="name">  
  Name  
</label>
```

# JSX คืออะไร

```
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!')  
)
```



```
<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>
```



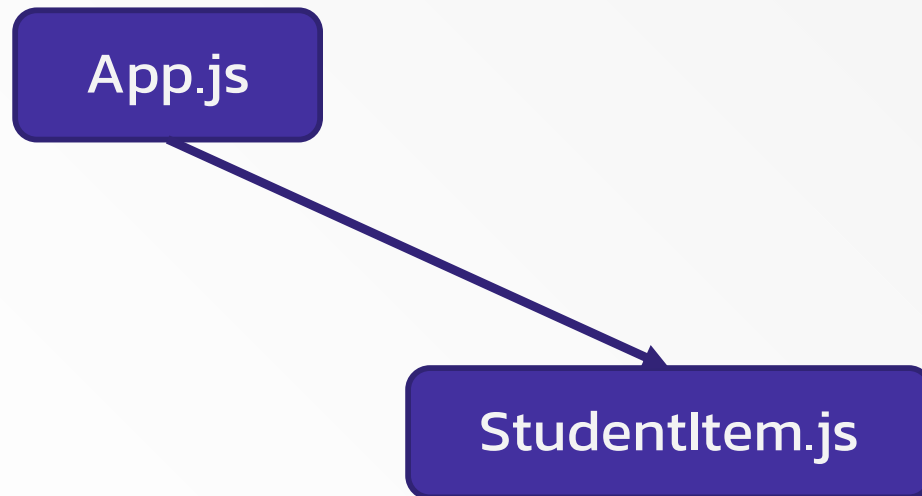
# JSX คืออะไร

**Mini Workshop:** JSX คืออะไร

Note: Lab-r1-l4



# JSX คืออะไร



# JSX คืออะไร

**Mini Workshop:** JSX คืออะไร - 2

Note: Lab-r1-l5

# JSX คืออะไร

## การเพิ่ม CSS ใน JSX (Styling ใน React)

- Inline
- Plain CSS
- CSS Module
- etc.

Ref - <https://www.freecodecamp.org/news/how-to-style-react-apps-with-css>

# JSX คืออะไร

## การเพิ่ม CSS ใน JSX (Styling ใน React)

- Inline
- Plain CSS
- CSS Module
- etc.

\*ที่จริงมีแบบอื่นอีกแต่เราจะเรียนกันแค่ 2 แบบก่อนเท่านั้นลองดูเพิ่มเติมใน <https://www.freecodecamp.org/news/how-to-style-react-apps-with-css>



# JSX คืออะไร

## การใส่ CSS แบบ Inline

1. เขียน CSS properties ใน object และใส่ไปให้กับ props ที่ชื่อว่า "style"
2. CSS properties ที่เขียนใน JSX จะต้องเป็น camelCase เช่น จาก background-color จะกลายเป็น backgroundColor

# JSX คืออะไร

## ตัวอย่างที่ 1 - Inline

ตัวอย่างนี้เปลี่ยนสีของพื้นหลังให้เป็นสีแดง

```
class MyHeader extends React.Component {  
  render() {  
    const mystyle = {  
      backgroundColor: "red"  
    };  
    return (  
      <div>  
        <h1 style={mystyle}>Hello Style!</h1>  
        <p>Add a little style!</p>  
      </div>  
    );  
  }  
}
```

# JSX คืออะไร

## ตัวอย่างที่ 2 – **Inline, Ref:** React CSS (w3schools.com)

หรือเราจะใส่ตัว object ของ css properties เข้าไปใน JSX เลยก็ได้

```
class MyHeader extends React.Component {  
  render() {  
    const mystyle = {  
      backgroundColor: "red"  
    };  
    return (  
      <div>  
        <h1 style={mystyle}>Hello Style!</h1>  
        <p>Add a little style!</p>  
      </div>  
    );  
  }  
}
```


# JSX คืออะไร

## ตัวอย่างที่ 2 – Inline, Ref: [React CSS \(w3schools.com\)](https://www.w3schools.com/react/react_css.asp)

หรือเราจะใส่ตัว object ของ css properties เข้าไปใน JSX เลยก็ได้

```
class MyHeader extends React.Component {  
  render() {  
    const mystyle = {  
      backgroundColor: "red"  
    };  
    return (  
      <div>  
        <h1 style={mystyle}>  
          Hello Style!  
        </h1>  
        <p>Add a little style!</p>  
      </div>  
    );  
  }  
}
```

```
class MyHeader extends React.Component {  
  render() {  
    return (  
      <div>  
        <h1 style={{ backgroundColor: "red" }}>  
          Hello Style!  
        </h1>  
        <p>Add a little style!</p>  
      </div>  
    );  
  }  
}
```



# JSX คืออะไร

## การใส่ CSS แบบ Plain CSS

1. เขียนเป็นไฟล์ CSS แยกเป็นอีกไฟล์หนึ่ง
2. Import เข้ามาใช้ใน JSX

# JSX คืออะไร

## ตัวอย่างที่ 1 – Plain CSS

```
/* styles.css */
```

```
.box-wrapper {  
  text-align: center;  
  max-width: 950px;  
  margin: 0 auto;  
  border: 1px solid #e6e6e6;  
  padding: 40px 25px;  
  margin-top: 50px;  
}
```

```
import './styles.css';
```

```
class MyHeader extends React.Component {  
  render() {  
    return (  
      <div className="box-wrapper">  
        <h1>Hello Style!</h1>  
        <p>Add a little style!</p>  
      </div>  
    );  
  }  
}
```

# JSX คืออะไร

## ตัวอย่างที่ 1 – Plain CSS

```
/* styles.css */  
  
.box-wrapper {  
  text-align: center;  
  max-width: 950px;  
  margin: 0 auto;  
  border: 1px solid #e6e6e6;  
  padding: 40px 25px;  
  margin-top: 50px;  
}
```

```
import './styles.css';  
  
class MyHeader extends React.Component {  
  render() {  
    return (  
      <div className="box-wrapper">  
        <h1>Hello Style!</h1>  
        <p>Add a little style!</p>  
      </div>  
    );  
  }  
}
```

# JSX คืออะไร

## ตัวอย่างที่ 1 – Plain CSS

และต้อง **import** ไฟล์ styles.css ในไฟล์ Component

```
/* styles.css */
```

```
.box-wrapper {  
  text-align: center;  
  max-width: 950px;  
  margin: 0 auto;  
  border: 1px solid #e6e6e6;  
  padding: 40px 25px;  
  margin-top: 50px;  
}
```

```
import './styles.css';
```

```
class MyHeader extends React.Component {  
  render() {  
    return (  
      <div className="box-wrapper">  
        <h1>Hello Style!</h1>  
        <p>Add a little style!</p>  
      </div>  
    );  
  }  
}
```



# JSX คืออะไร

## ตัวอย่างที่ 1 – Plain CSS

และต้อง `import` ไฟล์ `styles.css` ในไฟล์ Component

```
/* styles.css */
```

```
.box-wrapper {  
  text-align: center;  
  max-width: 950px;  
  margin: 0 auto;  
  border: 1px solid #e6e6e6;  
  padding: 40px 25px;  
  margin-top: 50px;  
}
```

```
import './styles.css';
```

```
class MyHeader extends React.Component {  
  render() {  
    return (  
      <div className="box-wrapper">  
        <h1>Hello Style!</h1>  
        <p>Add a little style!</p>  
      </div>  
    );  
  }  
}
```

# JSX คืออะไร

## ตัวอย่างที่ 1 – Plain CSS

หลังจากนั้นก็ import CSS class ชื่อ **box-wrapper** ในไฟล์ Component

```
/* styles.css */  
  
.box-wrapper {  
  text-align: center;  
  max-width: 950px;  
  margin: 0 auto;  
  border: 1px solid #e6e6e6;  
  padding: 40px 25px;  
  margin-top: 50px;  
}
```

```
import './styles.css';  
  
class MyHeader extends React.Component {  
  render() {  
    return (  
      <div className="box-wrapper">  
        <h1>Hello Style!</h1>  
        <p>Add a little style!</p>  
      </div>  
    );  
  }  
}
```

# JSX คืออะไร

**Mini Workshop:** เพิ่ม CSS ให้ Component

Note: Lab-r1-l6

CSS link:

<https://raw.githubusercontent.com/soncomqiq/react-app-1/main/src/components/StudentItem.css>

# JSX คืออะไร

## Expression

- การแสดงผล JavaScript expression ใน element จะใช้ { } ครอบ
- คำสั่งที่ใช้ได้มีดังนี้
  - print variable (แสดงค่าตัวแปร)
  - conditional (ternary) operator (If แบบย่อ)
  - การคำนวณ
  - เรียกใช้ function
  - อื่น ๆ

# JSX คืออะไร

## Expression

```
<div className="App" value={...}>  
  {...}  
</div>
```

# JSX คืออะไร

## JavaScript expression มีอะไรบ้าง

- ข้อมูลประเภทต่าง ๆ เช่น string, number, boolean, null, undefined
- โครงสร้างข้อมูล (Data structure) เช่น array , object
- ตัวดำเนินการ (Operator) เช่น && , || , !, + , - , \* , /
- ตัวดำเนินการทางตรรกะ (Logic Operator) เช่น if, else if , else และ ternary
- ฟังก์ชัน

# JSX คืออะไร

## JavaScript expression มีอะไรบ้าง

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

# JSX คืออะไร

## Ternary

```
const a = true
```

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



# JSX คืออะไร

การนำอาเรย์มาผ่าน map ฟังก์ชันทำให้เป็น Element

```
let boo = false
let arr = ["a","b","c"]

<div className="App" value={boo ? "have" : "no have"}>
  {arr.map(item => <p> {item}</p>)}
  {arr}

  {2 + 2}
</div>
```

# JSX คืออะไร

## การใช้ && Operation

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

# JSX คืออะไร

## การคำนวณทางคณิตศาสตร์

`<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>`

# JSX คืออะไร

## การใช้ฟังก์ชัน

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

# JSX คืออะไร

**แต่ JavaScript Expression บางตัวก็ไม่สามารถแสดงผลหน้าเว็บได้**

- ข้อมูลประเภท string, number, boolean , null, undefined แสดงผลบนหน้าเว็บไม่ได้ หน้าเว็บได้แค่ string และ number
- โครงสร้างข้อมูล (เช่น array, object) แสดงผลบนหน้าเว็บได้เฉพาะ array

# JSX คืออะไร

แต่ JavaScript Expression บางตัวก็ไม่สามารถแสดงผลหน้าเว็บได้

```
<div/>  
<div></div>  
<div>{false}</div>  
<div>{null}</div>  
<div>{undefined}</div>  
<div>{true}</div>
```

ค่า false, null, undefined, และ true จะไม่ถูกแสดงออกมาให้เห็นในเว็บ

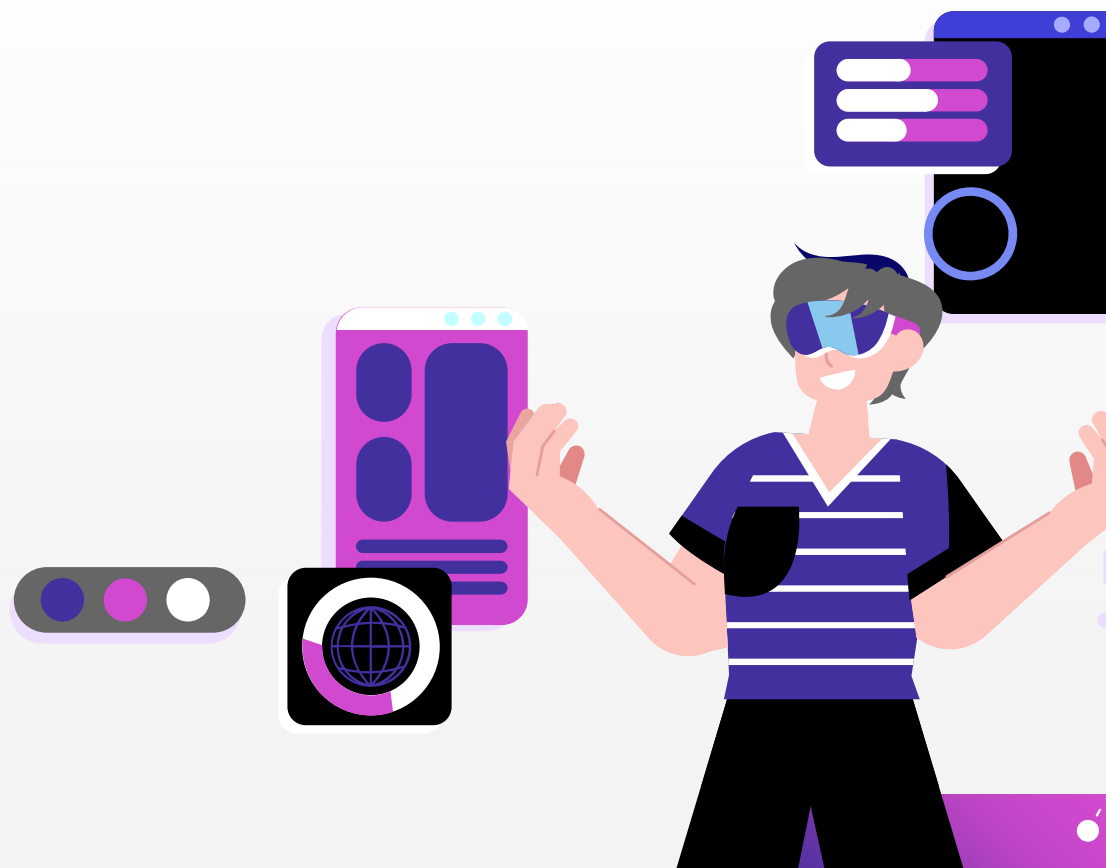
# JSX คืออะไร

**Mini Workshop:** Expression ใน JSX

Note: Lab-r1-l7

# คอมพิวเตอร์

---





# คอมโพเนนต์

## คอมโพเนนต์ (Components) คืออะไร

- ใน React ส่วนประกอบต่าง ๆ จะถูกแบ่งออกเป็น Component ย่อย ๆ
- แนวคิดการแบ่ง UI ของเว็บออกมาเป็นส่วนๆ ภายใน Component มี logic ของตัวเอง ทำให้สามารถนำมาใช้ใหม่ (Reuse) ได้ทั้งเว็บ
- และการแบ่ง Component จะช่วยแบ่ง Logic ต่าง ๆ ออกเป็นย่อย ๆ เพื่อช่วยไม่ให้เกิดการรวมกันของ Logic ขนาดใหญ่ไว้ในที่เดียว (ไว้ใน function เดียว)
- โดย Components ใน React จะมี 2 ประเภท
  1. Function Component
  2. Class Component

# คอมโพเนนต์

## เขียนแบบ Function declaration

```
import React from "react";

function Person(props) {
  return (
    <div>
      <h1>Hello, {props.name}</h1>
    </div>
  )
}

export default Person;
```

## เขียนแบบ Arrow function

```
import React from "react";

const Person = props => {
  return (
    <div>
      <h1>Hello, {props.name}</h1>
    </div>
  )
}

export default Person;
```

ตัวอย่าง Function Component ก็คือที่เราเขียนกันใน Lab-r1-l7

# คอมโพเนนต์

```
import React from "react";

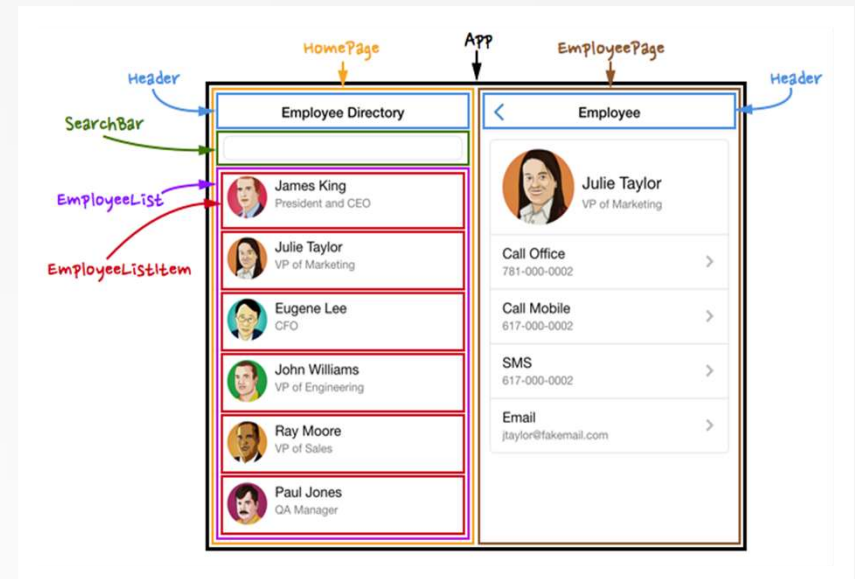
class Person extends React.Component {
  render() {
    return (
      <div>
        <h1>Hello, {this.props.name}</h1>
      </div>
    )
  }
}

export default Person;
```

ตัวอย่าง Class Component ในคอร์สนี้ เราจะยังไม่เน้น Class Component แต่อยากให้เราทราบว่าสามารถเขียนแบบนี้ได้ด้วย

# คอมโพเนนต์

1. การนำกลับมาใช้ซ้ำ (Reuse)
2. Component แต่ละอันที่สร้างมา ตอนหน้า Home page เราสามารถนำไปใช้ซ้ำได้ตอนที่สร้างหน้า หน้าใหม่ เช่น EmployeePage
3. การไม่รวมโค้ดไว้ทีเดียว
4. การที่เราแยกเป็น Component ย่อย ๆ แต่ละคอมโพเนนต์ ก็จะทำงานแค่อย่างเดียวนะ เช่น EmployeeListItem ทำหน้าที่แสดง ชื่อ ตำแหน่งและภาพ ส่วน EmployeeList ก็ทำหน้าที่แสดง EmployeeListItem ส่วน SearchBar ก็ทำหน้าที่ Search แต่ถ้าเราไม่แบ่งเป็น Component ย่อย ๆ โค้ดทั้งหมดที่พูดมาอยู่ใน Component เดียว ก็จะทำให้โค้ดอ่านยากและซับซ้อน



# Component VS Element

รายละเอียดความแตกต่างของ Component และ Element มีรายละเอียดลึกกว่า  
นี้ สามารถอ่านต่อได้ที่ <https://www.geeksforgeeks.org/what-is-the-difference-between-element-and-component>

แต่ในคอร์สนี้เราจะขอพูดคร่าว ๆ ว่า Component กับ Element ต่างกันอย่างไร

# Component VS Element

ก่อนอื่นเราจะมาดูจากตัวอย่างนี้

```
import React from "react";

function Person(props) {
  return (
    <div>
      <h1>Hello, {props.name}</h1>
    </div>
  )
}

export default Person;
```

# Component VS Element

ทั้งก่อนนี้เราจะเรียกว่า **Component** ซึ่งสามารถเป็นแบบ Class หรือ Function ก็ได้

```
import React from "react";

function Person(props) {
  return (
    <div>
      <h1>Hello, {props.name}</h1>
    </div>
  )
}

export default Person;
```

# Component VS Element

แต่ Element ก็คือพวกที่เป็น Tag JSX พวกนี้ อันแรกคือ element ที่เป็น div

```
import React from "react";

function Person(props) {
  return (
    <div>
      <h1>Hello, {props.name}</h1>
    </div>
  )
}

export default Person;
```



# Component VS Element

ส่วนอันที่สองคือ element ที่เป็น h1

```
import React from "react";

function Person(props) {
  return (
    <div>
      <h1>Hello, {props.name}</h1>
    </div>
  )
}

export default Person;
```

# Component VS Element

เพราะว่าในที่สุดแล้ว JSX ก็กลายเป็น `React.createElement()`

```
import React from "react";

function Person(props) {
  return (
    <div>
      <h1>Hello, {props.name}</h1>
    </div>
  )
}

export default Person;
```

```
import React from "react";

class Person extends React.Component {
  render() {
    return React.createElement("div", null,
      React.createElement("h1", null, "Hello, ", props.name)
    );
  }
}

export default Person;
```

# Component VS Element

เพราะว่าในที่สุดแล้ว JSX ก็กลายเป็น `React.createElement()`

```
import React from "react";

function Person(props) {
  return (
    <div>
      <h1>Hello, {props.name}</h1>
    </div>
  )
}

export default Person;
```

```
import React from "react";

class Person extends React.Component {
  render() {
    return React.createElement("div", null,
      React.createElement("h1", null, "Hello, ", props.name)
    );
  }
}

export default Person;
```

# Component VS Element

เพราะว่าในที่สุดแล้ว JSX ก็กลายเป็น React.createElement()

```
import React from "react";

function Person(props) {
  return (
    <div>
      <h1>Hello, {props.name}</h1>
    </div>
  )
}

export default Person;
```

```
import React from "react";

class Person extends React.Component {
  render() {
    return React.createElement("div", null,
      React.createElement("h1", null, "Hello, ", props.name)
    );
  }
}

export default Person;
```

# ส่งข้อมูลผ่าน Props

**Mini Workshop:** ส่งข้อมูลผ่าน Props

Note: ดูไฟล์ <Lab-r1-l8.txt>



# ส่งข้อมูล ผ่าน Props

---



# ส่งข้อมูลผ่าน Props

## Props คืออะไร

**<App />**

```
const name = "Potion";
```

สมมุติใน App.js ตอนนี้เรามีตัวแปร  
ชื่อ name ที่เก็บค่า "Potion" ไว้อยู่

# ส่งข้อมูลผ่าน Props

## Props คืออะไร

**<App />**

```
const name = "Potion";
```

**<Item />**

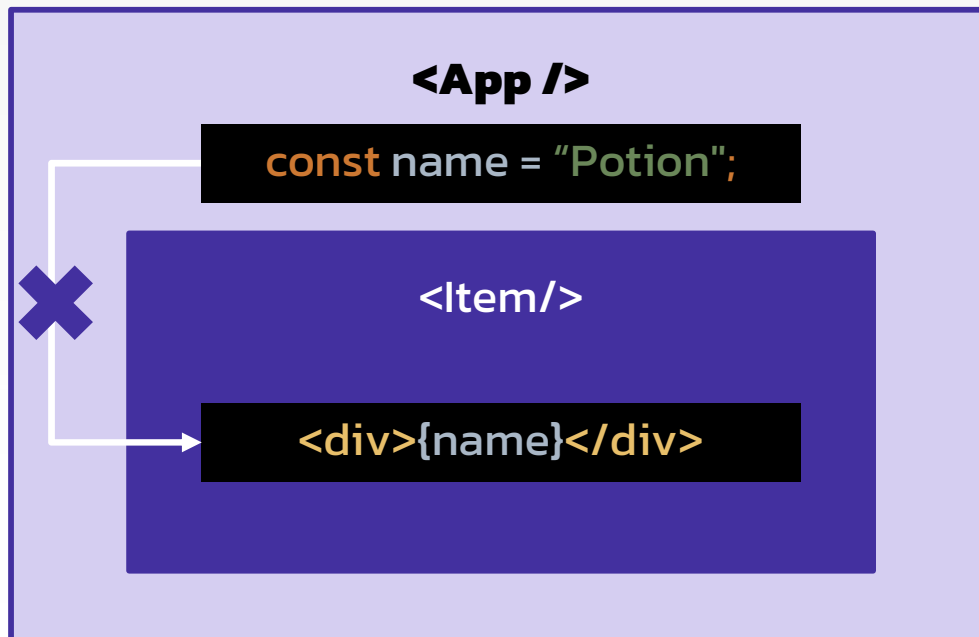
```
<div>{name}</div>
```

และสมมติเรามี Custom Component ที่เราสร้างขึ้นเองชื่อ Item โดยมี div ข้างในที่เราอยากให้ name มาแสดงใน div ตรงนี้ ซึ่งแน่นอนว่าปัญหาคือตัวแปร name อยู่ใน App Component ไม่ได้อยู่ใน Item Component



# ส่งข้อมูลผ่าน Props

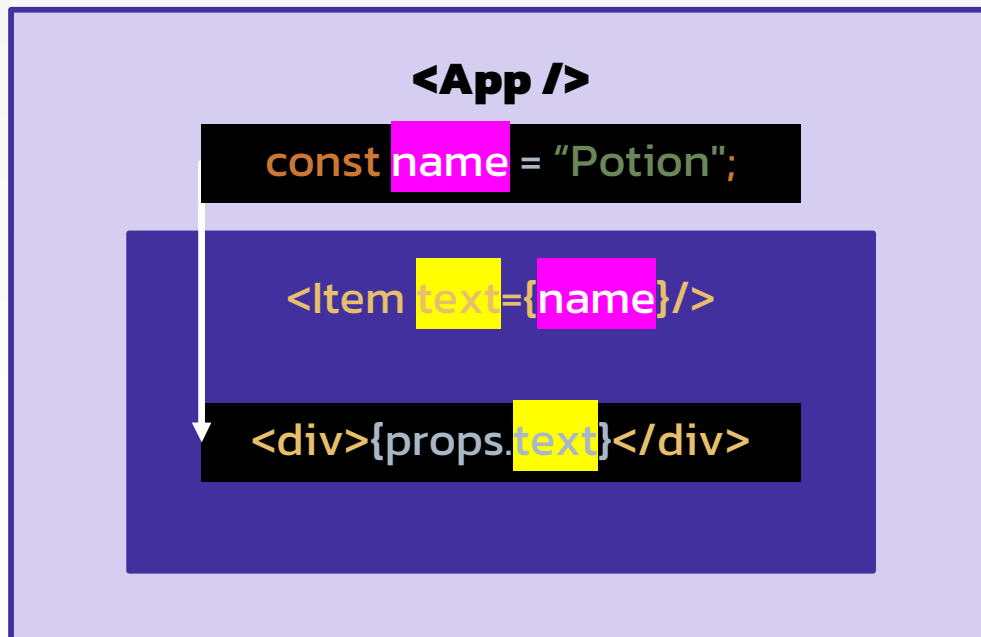
## Props คืออะไร



ซึ่งตัว Item Component ก็เข้าถึงไม่ได้  
ตัวแปร name นี้ไม่ได้

# ส่งข้อมูลผ่าน Props

## Props คืออะไร



แต่เราสามารถใช้ Concept Props ได้ ซึ่งทำให้เราสามารถส่งข้อมูล (ในที่นี้ก็คือ ตัวแปร `name`) จาก App Component ลงไปให้ Item Component ใช้ได้ โดยการเพิ่ม Attribute ที่ชื่อ `Text` เข้าไป และใน Item Component เราก็สามารถเข้าถึงข้อมูลนั้นได้ผ่าน Attributes เหล่านี้

# ส่งข้อมูลผ่าน Props

## Props คืออะไร



เราอาจจะจำได้ว่าพวก HTML Element จะมีพวก Attributes ต่าง ๆ เช่น style  
`<div style="background:black">test</div>`  
ซึ่งใน React ที่เราเขียน Custom Component ขึ้นมา ก็สามารถกำหนด Attributes ได้เหมือนกัน ซึ่งคอนเซปต์การส่งข้อมูลผ่านลงไปที่ตัวลูกนี้ เรียกว่า Props (ไม่ได้เรียกว่า Attributes)

# ส่งข้อมูลผ่าน Props

## Props คืออะไร

**<App />**

```
const name = "Potion";
```

```
<Item text={name}/>
```

```
<div>{props.text}</div>
```

Props ก็ย่อมาจาก Properties  
พูดสรุปก็คือเราสามารถ Set  
ตัว Properties ของ Custom  
Component ของเราเองได้

# ส่งข้อมูลผ่าน Props

เวลาเราส่ง props name surname และ age ไป มันก็จะกลายเป็น key และ value ใน object props เพราะฉะนั้นเวลาเรียนใช้ เราจึงต้อง props.name props.surname

```
<StudentItem name="Tom" surname="Cruise" age={25} />
```

```
function StudentItem(props) {
```

```
  return (
```

```
    <div className="StudentItem">
```

```
      <div>{props.name}</div>
```

```
      <div>{props.surname}</div>
```

```
      <div>{props.age}</div>
```

```
    </div>
```

```
  )
```

```
}
```

```
{  
  name: "Tom",  
  surname: "Cruise",  
  age: 25  
}
```

# ส่งข้อมูลผ่าน Props

เวลาเราส่ง props name surname และ age ไป มันก็จะกลายเป็น key และ value ใน object props เพราะฉะนั้นเวลาเรียนใช้ เราจึงต้อง props.name props.surname

```
<StudentItem name="Tom" surname="Cruise" age={25} />
```

```
function StudentItem(props) {
```

```
  return (
```

```
    <div className="StudentItem">
```

```
      <div>{props.name}</div>
```

```
      <div>{props.surname}</div>
```

```
      <div>{props.age}</div>
```

```
    </div>
```

```
  )
```

```
}
```

```
{  
  name: "Tom",  
  surname: "Cruise",  
  age: 25  
}
```

# ส่งข้อมูลผ่าน Props

เวลาเราส่ง props name surname และ age ไป มันก็จะกลายเป็น key และ value ใน object props เพราะฉะนั้นเวลาเรียนใช้ เราจึงต้อง props.name props.surname

```
<StudentItem name="Tom" surname="Cruise" age={25} />
```

```
function StudentItem(props) {
```

```
  return (  
    <div className="StudentItem">  
      <div>{props.name}</div>  
      <div>{props.surname}</div>  
      <div>{props.age}</div>  
    </div>  
  )  
}
```

```
{  
  name: "Tom",  
  surname: "Cruise",  
  age: 25  
}
```

# ส่งข้อมูลผ่าน Props

เวลาเราส่ง props name surname และ age ไป มันก็จะกลายเป็น key และ value ใน object props เพราะฉะนั้นเวลาเรียนใช้ เราจึงต้อง props.name props.surname

```
<StudentItem name="Tom" surname="Cruise" age={25} />
```

```
function StudentItem(props) {  
  
  return (  
    <div className="StudentItem">  
      <div>{props.name}</div>  
      <div>{props.surname}</div>  
      <div>{props.age}</div>  
    </div>  
  )  
}
```

```
{  
  name: "Tom",  
  surname: "Cruise",  
  age: 25  
}
```



# ส่งข้อมูลผ่าน Props

เวลาเราส่ง props name surname และ age ไป มันก็จะกลายเป็น key และ value ใน object props เพราะฉะนั้นเวลาเรียนใช้ เราจึงต้อง props.name props.surname

```
<StudentItem name="Tom" surname="Cruise" age={25} />
```

```
function StudentItem(props) {
```

```
  return (  
    <div className="StudentItem">  
      <div>{props.name}</div>  
      <div>{props.surname}</div>  
      <div>{props.age}</div>  
    </div>  
  )  
}
```

```
{  
  name: "Tom",  
  surname: "Cruise",  
  age: 25  
}
```

# ส่งข้อมูลผ่าน Props

เวลาเราส่ง props name surname และ age ไป มันก็จะกลายเป็น key และ value ใน object props เพราะฉะนั้นเวลาเรียนใช้ เราจึงต้อง props.name props.surname

```
<StudentItem name="Tom" surname="Cruise" age={25} />
```

```
function StudentItem(props) {
```

```
  return (  
    <div className="StudentItem">  
      <div>{props.name}</div>  
      <div>{props.surname}</div>  
      <div>{props.age}</div>  
    </div>  
  )  
}
```

```
{  
  name: "Tom",  
  surname: "Cruise",  
  age: 25  
}
```

# ส่งข้อมูลผ่าน Props

เวลาเราส่ง props name surname และ age ไป มันก็จะกลายเป็น key และ value ใน object props เพราะฉะนั้นเวลาเรียนรู้ เราจึงต้อง props.name props.surname

```
<StudentItem name="Tom" surname="Cruise" age={25} />
```

```
function StudentItem(props) {
```

```
  return (  
    <div className="StudentItem">  
      <div>{props.name}</div>  
      <div>{props.surname}</div>  
      <div>{props.age}</div>  
    </div>  
  )  
}
```

```
{  
  name: "Tom",  
  surname: "Cruise",  
  age: 25  
}
```

# ส่งข้อมูลผ่าน Props

**Mini Workshop:** เพิ่ม Logic ใน Component

Note: ดูไฟล์ <Lab-r1-l9.txt>

Link สำหรับ CSS ใน Component Tag:

<https://raw.githubusercontent.com/soncomqiq/react-app-1/main/src/components/StudentTags.css>



# ส่งข้อมูลผ่าน Props

**Mini Workshop:** เพิ่ม Logic ใน Component 2

Note: ดูไฟล์ <Lab-r1-l10.txt>

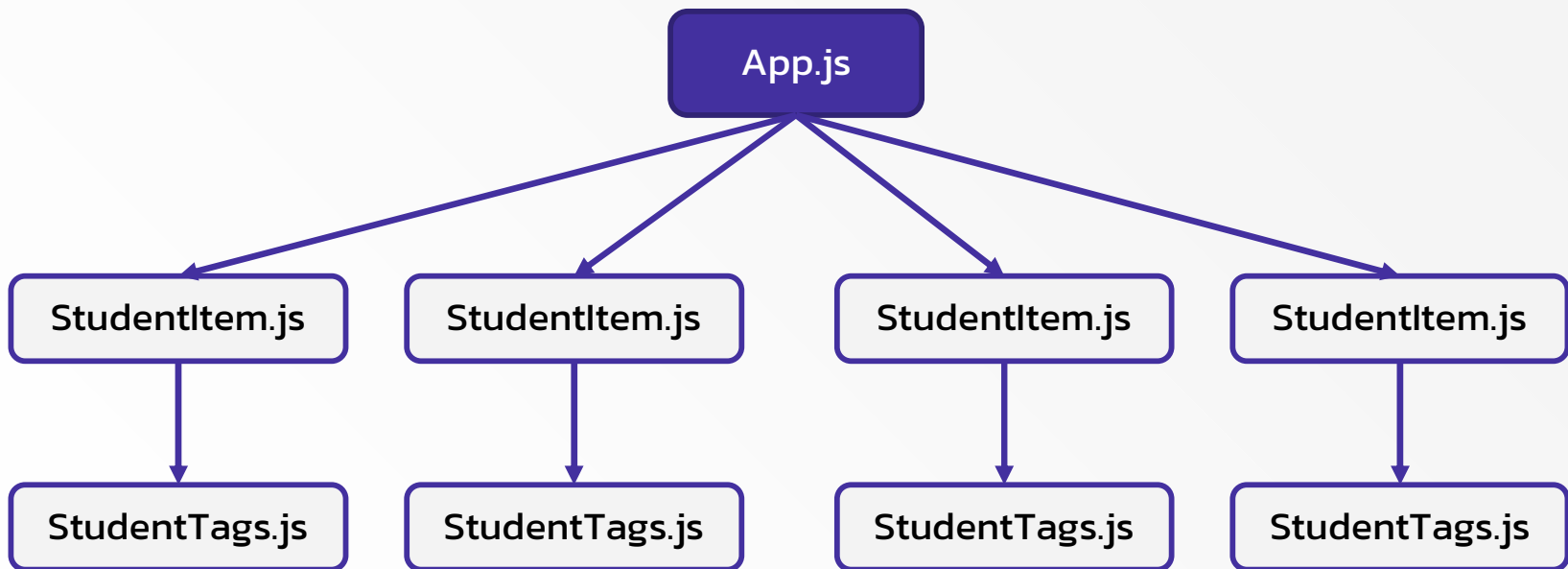


# ส่งข้อมูลผ่าน Props

**Mini Workshop:** การแยกคอมโพเนนต์

Note: ดูไฟล์ <Lab-r1-l11.txt>

# แผนผังคอมโพเนนต์ปัจจุบัน



## โจทย ให้ลองฝึก

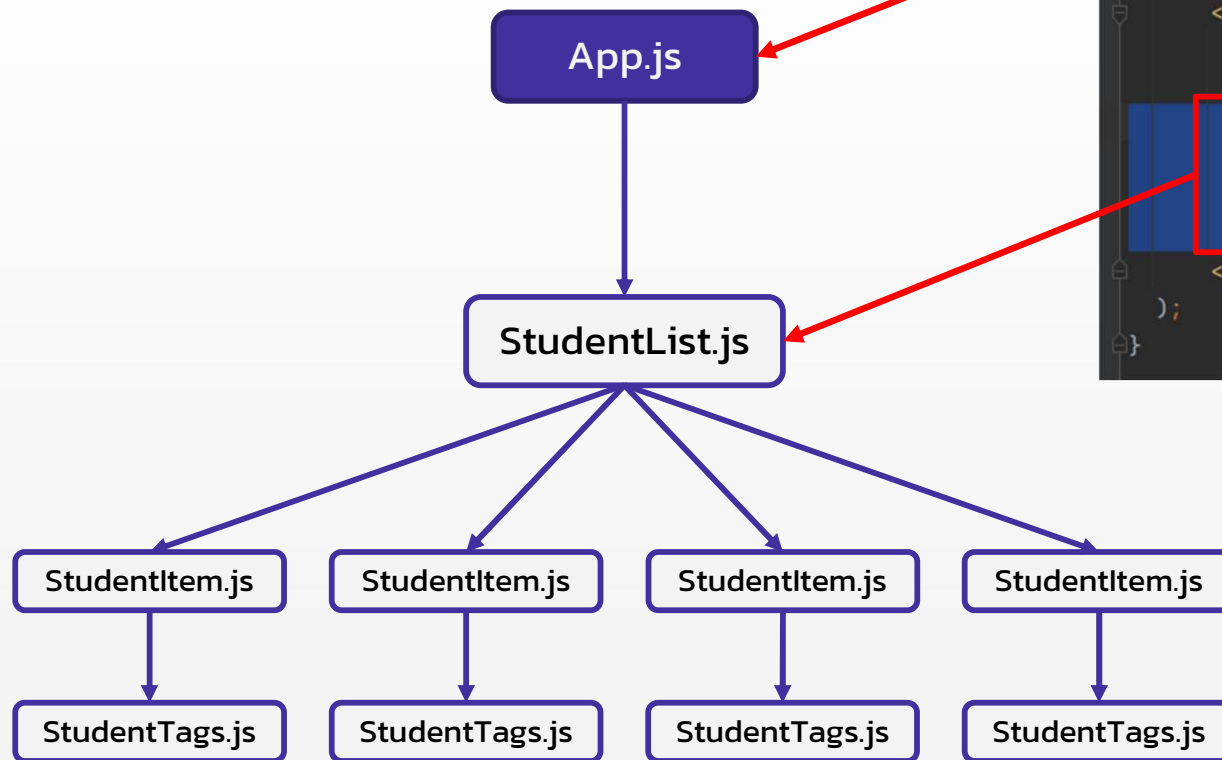
ตอนนี้ใน App Component มีโค้ดส่วนนี้ (ภาพด้านล่าง) ที่สามารถแยกมาเป็น Components ย่อยได้ โดยโจทยจะให้ผู้เรียนทำส่วนนี้เป็น Component ย่อยที่ชื่อ StudentList โดยที่เก็บ JSX ที่ไฮไลท์ไว้ แต่ ตัวแปร studentList ยังต้องอยู่ใน App component

```
    age: 39
  }
]

return (
  <div className="App">
    <h2>This is React Application</h2>
    <p>Let's get started!</p>
    <StudentItem name={studentList[0].name} surname={studentList[0].surname} age={studentList[0].age}></StudentItem>
    <StudentItem name={studentList[1].name} surname={studentList[1].surname} age={studentList[1].age}></StudentItem>
    <StudentItem name={studentList[2].name} surname={studentList[2].surname} age={studentList[2].age}></StudentItem>
    <StudentItem name={studentList[3].name} surname={studentList[3].surname} age={studentList[3].age}></StudentItem>
  </div>
);
}
```



# แผนผังคอมโพเนนต์ปัจจุบัน



```
function App() {  
  const studentList = [...]  
  
  return (  
    <div className="App">  
      <h2>This is React Application</h2>  
      <p>Let's get started!</p>  
      <StudentItem name={studentList[0].name}>  
      <StudentItem name={studentList[1].name}>  
      <StudentItem name={studentList[2].name}>  
      <StudentItem name={studentList[3].name}>  
    </div>  
  );  
}
```

ตัวแปรนี้ยังอยู่ใน App Component

อันนี้ย้ายไปเป็น Component ใหม่

# Full Workshop

Link: <https://github.com/soncomqiq/workshop-todolist-2023>