

# Introduction to React 01

# React คือ ?

- React เป็น javascript library สำหรับการสร้าง GUI
- React ทำให้สร้าง GUI ที่ซับซ้อนได้จากส่วนของ code ย่อย ๆ (เรียกว่า component)
- Component เหล่านี้ถูกสร้างขึ้นมาจากภาษา JSX



# เริ่มสร้าง React Project

- ใช้คำสั่งต่อไปนี้
- (คำสั่ง npx เป็นคำสั่งรัน package locally และจะติดตั้ง package กรณี ยังไม่เคยติดตั้งมาก่อน)

```
npx create-react-app finance-frontend  
cd finance-frontend  
npm start
```

# โครงสร้างของ React Projects

- node\_modules สำหรับเก็บ library ที่ติดตั้งเพิ่มเติมผ่านทางคำสั่ง npm install
- public สำหรับเก็บ HTML public ไฟล์ เช่น รูปภาพ และไฟล์ index.html
  - ไฟล์ index.html จะเป็นไฟล์แรกที่ web browser จะโหลดเข้าไปเพื่อเริ่มงาน
- src เก็บ source code ของ project
  - (รายละเอียด Slide ถัดไป)

# Src Folder ประกอบไปด้วย

- index.js และ index.css
  - ไฟล์แรกที่ Node Application จะเริ่มทำงาน
  - ใช้สำหรับการ mount application ลงใน Web Page
  - เพื่อ import css เพิ่มเติมจาก 3rd party library
- App.js และ App.css
  - ไฟล์ initial ของ project โดยจะถูกเรียกใช้โดย index.js
  - ใช้สำหรับ initial components ต่าง ๆ ของโปรแกรม
  - App.js จะเป็น component ที่อยู่ลำดับสูงที่สุด (root node)



# React Hello World!!

- แก้ไขไฟล์ index.js เพื่อให้แสดงผล HelloWorld แทนที่จะแสดง App.js Component

src/index.js

```
root.render(  
  <React.StrictMode>  
    <div>  
      <h1>Hello, world!</h1>  
      <h2>It is {new Date().toLocaleTimeString()}.</h2>  
    </div>  
  </React.StrictMode>,  
  document.getElementById('root')  
) ;
```



# Update Rendered Element

- Element ที่แสดงผลบนหน้าจอของ React
  - เป็น **immutable** ไม่สามารถแก้ไขค่าได้ แต่จะเป็นการสร้าง element ใหม่มาแสดงผลแทน

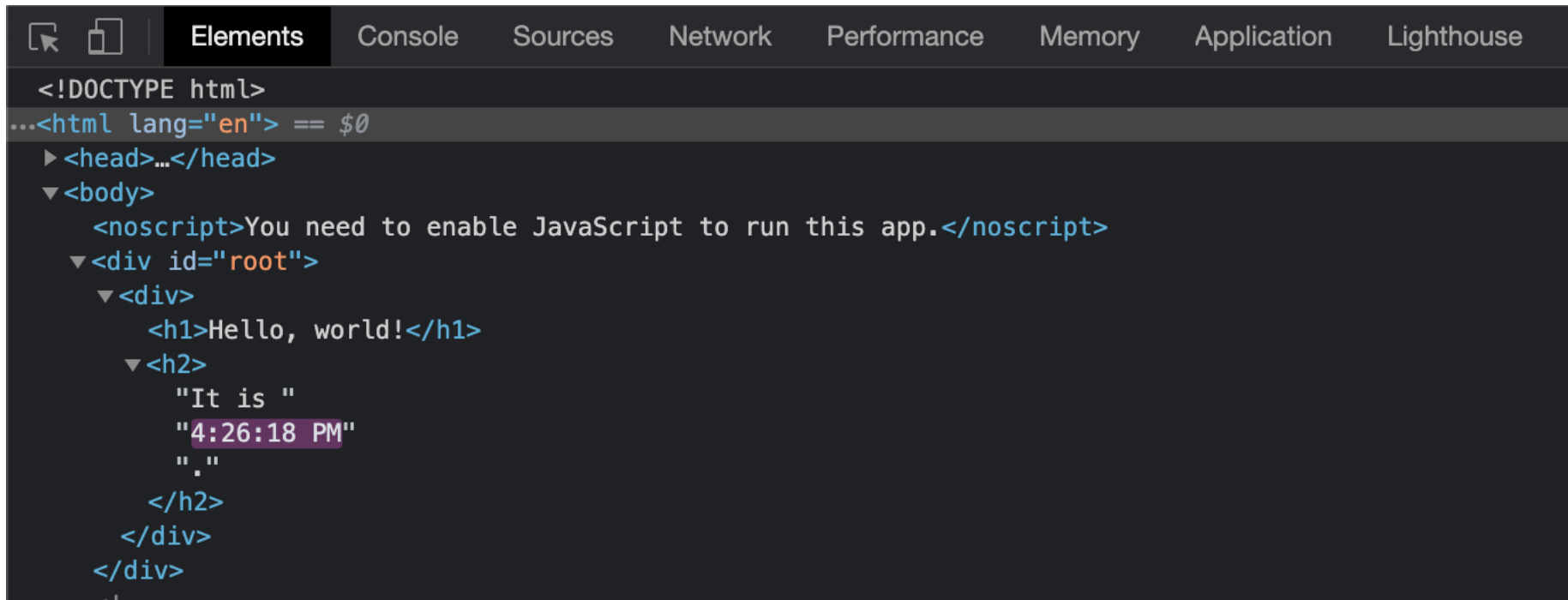
src/index.js

```
const showTime = () => {
  root.render(
    <React.StrictMode>
      <div>
        <h1>Hello, world!</h1>
        <h2>It is {new Date().toLocaleTimeString()}.</h2>
      </div>
    </React.StrictMode>,
    document.getElementById('root')
  );
}

setInterval(showTime, 1000);
```

# Update Rendered Element (2)

- สังเกต Tab Elements ของ Chrome จะเห็นว่า
  - ใน Code จะมีการสร้าง Element ใหม่บนหน้าจอทั้ง Hello World และนาฬิกา
  - แต่ใน Tab Element จะเห็นว่า React Render เฉพาะส่วนที่มีการเปลี่ยนแปลงค่าเท่านั้น



```
<!DOCTYPE html>
...<html lang="en"> == $0
  <head>...</head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root">
      <div>
        <h1>Hello, world!</h1>
        <h2>
          "It is "
          "4:26:18 PM"
          "."
        </h2>
      </div>
    </div>
  </body>
</html>
```



# Components

- แนวคิดหลักของ React คือการแบ่งส่วนการทำงานในหน้าจ่อออกเป็นส่วน ๆ
  - Independent แต่ละ component ทำงานเป็นอิสระจากกัน
  - Re-useable เมื่อ component ถูกสร้างมาแล้วสามารถถูกใช้ใหม่ได้โดยง่าย

## Functional Component

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

## Class Component

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

ปัจจุบัน Functional Component เป็นที่นิยมมากกว่า Class Component



# ทดลองใช้ App.js Component

src/index.js

```
root.render(  
  <React.StrictMode>  
    <App/>  
  </React.StrictMode>,  
);
```

ต่อหน้าถัดไป



# ทดลองใช้ App.js Component (2)

src/App.js

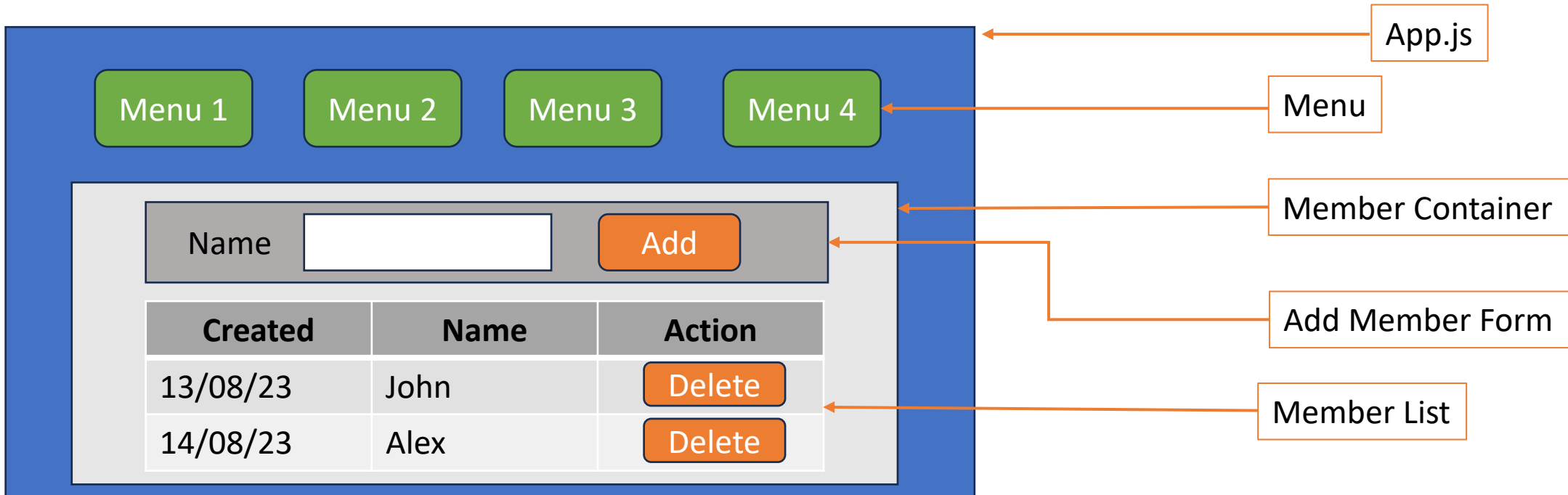
```
function App() {  
  return (  
    <div className="App">  
      <header className="App-header">  
        <table border="1">  
          <tr>  
            <th>Date-Time</th>  
            <th>Type</th>  
            <th>amount</th>  
            <th>note</th>  
          </tr>  
          <tr>  
            <td>01/02/2021 - 08:30</td>  
            <td>รายรับ</td>  
            <td>20,000</td>  
            <td>allowance</td>  
          </tr>  
  
          // ...  
        </table>  
      </header>  
    </div>  
  );  
}
```

src/App.js

```
        // ...  
  
        <tr>  
          <td>01/02/2021 - 10:30</td>  
          <td>รายจ่าย</td>  
          <td>150</td>  
          <td>อาหารเที่ยง</td>  
        </tr>  
      </table>  
    </header>  
  </div>  
);  
}
```

# React Components

- การพัฒนา Web Application ด้วย React
  - คล้ายกับการเล่น Lego
  - เป็นการสร้างชิ้นส่วนการทำงานขึ้นเล็กๆ (component) แล้วนำมาประกอบกัน





# ประกาศ Component ด้วยตนเอง

- เราสามารถสร้าง Component สำหรับ Element ที่ต้องการแสดงผล
  - เพื่อลดความซับซ้อนของ code หลักง
  - สามารถ Re-use component ทั้งในหน้าเดียวกัน และในหน้าอื่น ๆ ได้

src/App.js

```
import './App.css'
import TransactionList from './components/TransactionList';

function App() {
  const transactionData = [
    {id: 1, created: "01/02/2021 - 08:30", type: "รายรับ", amount: 20000, note: "allowance"},
    {id: 2, created: "01/02/2021 - 10:30", type: "รายจ่าย", amount: 150, note: "อาหารเที่ยง"},
  ]
  return (
    <div className="App">
      <header className="App-header">
        <TransactionList data={transactionData}/>
      </header>
    </div>
  );
}
```



## ประกาศ Component ด้วยตนเอง (2)

src/components/TransactionList.js

```
export default function TransactionList(props) {

  const generateRows = () => {
    if(props.data !== null) {
      return props.data.map( transaction => (
        <tr key={transaction.id}>
          <td>{transaction.created}</td>
          <td>{transaction.type}</td>
          <td>{transaction.amount}</td>
          <td>{transaction.note}</td>
        </tr>
      ))
    }
    else {
      return null;
    }
  }

  return(
    <table border="1">
      <thead>
        <tr>
          <th>Date-Time</th>
          <th>Type</th>
          <th>amount</th>
          <th>note</th>
        </tr>
      </thead>
      <tbody>{generateRows()}</tbody>
    </table>
  )
}
```

- Key เป็น property พิเศษ เพื่อให้ React รู้ว่า element ใดมีการเปลี่ยนแปลง, เพิ่ม, หรือลบ
- Key จะถูกกำหนดค่าให้แก่ element ภายใน list
- Key ต้องเป็นค่า unique เช่น id ของ object ข้างใน array (กรณีมี 2 List ไม่จำเป็นต้องกำหนด key ให้เป็นค่า unique ทั้งคู่)

# Props

- สังเกตว่า Component ที่อยู่ในลำดับสูงกว่า สามารถส่งตัวแปร props หรือ property ของ component เพื่อนำไปประมวลผลหรือแสดงผลได้
- ค่าของ props จะเป็นตัวแปร read only ไม่สามารถแก้ไขค่าได้

# Event Handling

- การจัดการ event เช่น mouse click
  - เขียนคล้ายกับ HTML ธรรมดา
  - JSX จะรับ function เป็น parameter สำหรับการจัดการ event

```
<button onclick="onMouseClicked()"> ClickMe! </button>
```

```
<button onClick={onMouseClicked}> ClickMe! </button>
```





## Event Handling (2)

src/components/TransactionList.js

```
const generateRows = () => {
  if(props.data !== null) {
    return props.data.map( transaction => (
      <tr>
        <td>{transaction.created}</td>
        <td>{transaction.type}</td>
        <td>{transaction.amount}</td>
        <td>
          <button onClick={() => alert(JSON.stringify(transaction))}>Debug</button>
        </td>
      </tr>
    ))
  }
  else {
    return null;
  }
}
```



# Conditional Rendering

- React ใช้ประโยชน์จาก JSX ที่จะ return สิ่งแสดงผลบนหน้าจอ ขึ้นอยู่กับ logic การทำงานของโปรแกรมได้

src/App.js

```
function App() {  
  // ...  
  const summary = () => {  
    const currentAmount = transactionData.reduce( (sum, transaction) => sum += transaction.amount, 0);  
    if (currentAmount > 10000) {  
      return <p style={{ 'color': 'green' }}>Wow you're so rich - {currentAmount}</p>;  
    }  
    else {  
      return <p style={{ 'color': 'red' }}>So Poor... - {currentAmount}</p>  
    }  
  }  
  
  return (  
    <div className="App">  
      <header className="App-header">  
        {summary()}  
        <TransactionList data={transactionData}/>  
      </header>  
    </div>  
  );  
}
```

Inline Style



# Conditional Rendering (2)

- การใช้งานที่พบบ่อยอีกแบบคือ Inline If แบบ Conditional Operator

src/App.js

```
function App() {  
  
  // ...  
  const currentAmount = transactionData.reduce((sum, transaction) => sum += transaction.amount, 0);  
  const richGreeting = amount => <p style={{ 'color': 'green' }}>Wow you're so rich - {amount}</p>  
  const poorGreeting = amount => <p style={{ 'color': 'red' }}>So Poor... - {amount}</p>  
  
  return (  
    <div className="App">  
      <header className="App-header">  
        {currentAmount > 10000 ? richGreeting(currentAmount) : poorGreeting(currentAmount)}  
        <TransactionList data={transactionData}/>  
      </header>  
    </div>  
  );  
}
```



# Conditional Rendering (3)

- ตัวอย่างการใช้งาน Inline If คู่กับ && Operator

src/App.js

```
function App() {  
  // ...  
  
  return (  
    <div className="App">  
      <header className="App-header">  
        {currentAmount >= 10000 && richGreeting(currentAmount)}  
        {currentAmount < 10000 && poorGreeting(currentAmount)}  
        <TransactionList data={transactionData}/>  
      </header>  
    </div>  
  );  
}
```



# TransactionList Background Color

- แก้ไข TransactionList.js ให้แสดงผลพื้นหลังเป็น

- สีเขียวถ้ามีชนิดเป็น “รายรับ”
- สีแดงถ้ามีชนิดเป็น “รายจ่าย”

```
<tr bgColor="green"> </tr>  
<tr bgColor="red"> </tr>
```

Date-Time	Type	amount	note
01/02/2021 - 08:30	รายรับ	20000	allowance
01/02/2021 - 10:30	รายจ่าย	150	อาหารเที่ยง

# Hooks

- Hooks เป็น Feature ที่ถูกพัฒนาเข้ามาใหม่สำหรับ React version 16.8+
- Hooks คือการเขียน Logic ของโปรแกรม และสามารถ re-use ใช้ได้ในหลายๆที่
  - เช่น การเขียน hook สำหรับ fetch ค่าจาก server
- Hooks ที่ถูกใช้เยอะที่สุดคือ
  - State Hook ใช้เพื่อให้ functional component มีความสามารถของ state
  - Effect Hook สามารถทำให้ component มีความสามารถ side effect
    - Side Effect คือความสามารถของ Function ที่สามารถ modify ค่าต่าง ๆ นอก function ได้
    - เช่น การเปลี่ยนค่า DOM ที่แสดงผล, เปลี่ยนค่าตัวแปร Global Variable



# State Hook

- สำหรับเก็บค่าของตัวแปรภายใน Component
  - จากตัวอย่างก่อนหน้านี้จะเห็นว่า Component ไม่มีการประกาศตัวแปรใด ๆ ขึ้นมาเลย
  - สังเกต Code ต่อไปนี้

src/App.js

```
function App() {  
  // ...  
  let counter = 0;  
  const counterClicked = () => {  
    console.log(`Clicked ${counter}`);  
    counter++;  
  }  
  
  return (  
    <div className="App">  
      <header className="App-header">  
        {counter > 3 ? richGreeting(counter) : poorGreeting(counter)}  
        <button onClick={counterClicked}>Add Counter</button>  
        <TransactionList data={transactionData}/>  
      </header>  
    </div>  
  );  
}
```

ค่าของตัวแปรไม่เปลี่ยน  
เนื่องจาก React ไม่ทราบ  
ว่ามีการเปลี่ยนแปลงค่า  
และต้อง reload DOM



# State Hook (2)

```
const [variable, setVariableFunction] = useState(<defaultValue>);
```

src/App.js

```
import { useState } from 'react';

function App() {

  // ...

  const [counter, setCounter] = useState(0);

  const counterClicked = () => {
    console.log("Clicked");
    setCounter(counter+1);
  }

  return (
    <div className="App">
      <header className="App-header">
        {counter >= 10000 && richGreeting(counter)}
        {counter < 10000 && poorGreeting(counter)}
        <button onClick={counterClicked}>Add Counter</button>
        <TransactionList data={transactionData}/>
      </header>
    </div>
  );
}
```

Array  
Destructuring



# Effect Hook

- ใช้ผ่าน `useEffect` โดยจะทำงาน function ที่อยู่ใน `useEffect` หลังจากที่มีการเปลี่ยนแปลง DOM เสร็จสิ้น
- React จะเรียก effect function ทุกๆครั้งที่มีการเปลี่ยนแปลง DOM
  - รวมทั้งการ render ครั้งแรกด้วย

```
useEffect(<Effect Function>, [<var1>, <var2>]);
```

- Effect Function จะรันทุกครั้งที่เกิดการ Render
- Effect Function สามารถ return ค่าเป็น function ได้ ซึ่งจะถูกระบุตอน element ถูกนำออกจากหน้าจอ (clean up function)

- List ของตัวแปรที่จะส่งผลให้เกิดการเรียก Effect Function เมื่อเกิดการเปลี่ยนแปลงค่าของตัวแปร
- ใช้สำหรับป้องกันการ Render Effect ใหม่ทุกครั้ง
- สามารถส่ง [ ] (list ว่าง) กรณีต้องการรัน Effect Function แค่ครั้งเดียวตอนสร้าง Element



# Effect Hook

- ทดลองสร้าง Component ใหม่ชื่อ Clock ที่แสดงเวลาปัจจุบัน

src/components/Clock.js

```
import { useState, useEffect } from 'react';

export default function Clock(props) {
  const [time, setTime] = useState(new Date());

  useEffect(() => {
    setInterval(() => {
      setTime(new Date());
    }, 1000);
  }, []);

  return (
    <h2>
      It is {time.toLocaleTimeString()}.
    </h2>
  )
}
```

src/App.js

```
// ...
import Clock from '../components/Clock';

function App() {
  // ...

  return (
    <div>
      <Clock/>
    </div>
  );
}
```

# กฎหลัก 2 ข้อของ Hooks

- เรียกใช้ Hooks ที่ top level ของ component เท่านั้น
  - ห้ามเรียก Hooks ภายใน Loop, Condition, หรือ Nested Function
  - ให้เรียก Hooks ที่จุดเริ่มของ Function เท่านั้น
  - เพื่อป้องกันลำดับการสร้าง Hooks ที่สับสน
- เรียกใช้ Hooks จาก React Function (Components) เท่านั้น
  - ห้ามเรียก Hooks จาก JavaScript Functions ธรรมดา



# Add Transaction Button

- แก้ไข App.js ให้มีปุ่ม Add Transaction โดยเมื่อกดแล้วให้เพิ่ม Transaction
- เพิ่ม App.js ให้มีการคำนวณ Current Amount หรือจำนวนเงินทั้งหมด

src/App.js

```
function App() {  
  
  const [amount, setAmount] = useState(0);  
  const [transactionData, setTransactionData] = useState([  
    { id: 1, created: "01/02/2021 - 08:30", type: "รายรับ", amount: 20000, note: "allowance" },  
    { id: 2, created: "01/02/2021 - 10:30", type: "รายจ่าย", amount: 150, note: "อาหารเที่ยง" }  
  ]);  
  
  const generateTransaction = () => {  
    return {  
      id: transactionData.length + 1,  
      created: new Date().toLocaleString(),  
      type: ['รายรับ', 'รายจ่าย'][Math.floor(Math.random() * 2)],  
      amount: Math.floor(Math.floor(Math.random() * 1000) + 1),  
      note: '',  
    }  
  }  
}
```

Current Amount 19779

Add Transaction

Date-Time	Type	amount	note
01/02/2021 - 08:30	รายรับ	20000	allowance
01/02/2021 - 10:30	รายจ่าย	150	อาหารเที่ยง
3/23/2021, 12:04:08 PM	รายรับ	419	
3/23/2021, 12:04:09 PM	รายรับ	423	



# Text Input

- ถ้าต้องการนำค่าจาก input ไปใช้งานจะทำอย่างไร?

src/App.js

```
export default function App() {  
  return(  
    <div>  
      <input />  
    </div>  
  )  
}
```

src/App.js

```
export default function App() {  
  const [inputValue, setInputValue] = useState("")  
  
  return(  
    <div>  
      <input value={inputValue} />  
      </div>value={inputValue}  
    )      onChange={ (evt) => {  
  }          setInputValue(evt.target.value)  
    }      }>  
    </div>  
  )  
}
```



## Add Transaction Button (2)

- ปรับให้การเพิ่ม Transaction มีการเลือก Type (รายรับ, รายจ่าย), กรอกจำนวน, และ note เพื่อ add ลงไป

รายรับ ▾

จำนวนเงิน

Note

Add Transaction

Date-Time	Type	Amount	Note	
01/02/2021 - 08:30	รายรับ	20000	allowance	Debug
01/02/2021 - 10:30	รายจ่าย	-150	อาหารเที่ยง	Debug

```
const [type, setType] = useState("รายรับ")
<select value={type} onChange={evt => setType(evt.target.value)}>
  <option value="รายรับ">รายรับ</option>
  <option value="รายจ่าย">รายจ่าย</option>
</select>
```



# Try Moment.js

- ติดตั้ง Moment.js - <https://momentjs.com/>

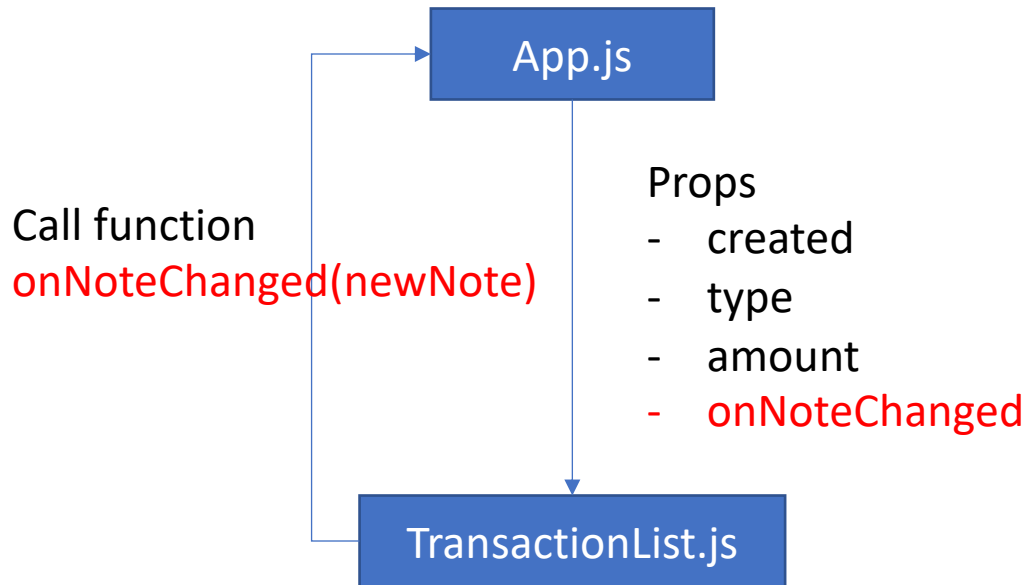
```
cd finance-frontend  
npm install moment --save
```

- ทดลองใช้ Moment.js

```
import moment from 'moment';  
  
function App() {  
  const handleAddItem = () => {  
    setTransactionData([...transactionData, {  
      id: transactionData.length + 1,  
      created: moment().format('DD/MM/YYYY - HH:mm'),  
      // .....  
    }])  
  }  
}
```

# Lifting State Up

- State ที่เก็บอยู่ใน component สามารถส่งขึ้นไปยัง ancestor node ได้ โดยการส่ง callback function ลงมาพร้อมกับ props



Date-Time	Type	amount	note
01/02/2021 - 08:30	รายรับ	20000	<input type="text" value="allowancesssss"/>
01/02/2021 - 10:30	รายจ่าย	150	<input type="text" value="อาหารเที่ยง"/>





# Lifting State Up (1)

src/component/TransactionList.js

```
export default function TransactionList(props) {  
  const generateRows = () => {  
    // ...  
    <td>  
      <input value={transaction.note}  
        onChange={(evt) => {  
          props.onNoteChanged(transaction.id, evt.target.value)  
        }}  
      />  
    </td>  
  }  
  
  // ...  
}
```



# Lifting State Up (2)

src/App.js

```
function App() {

  const handleNoteChanged = (id, note) => {
    setTransactionData(
      transactionData.map( transaction => {
        transaction.note = transaction.id === id ? note : transaction.note;
        return transaction
      })
    )
  }

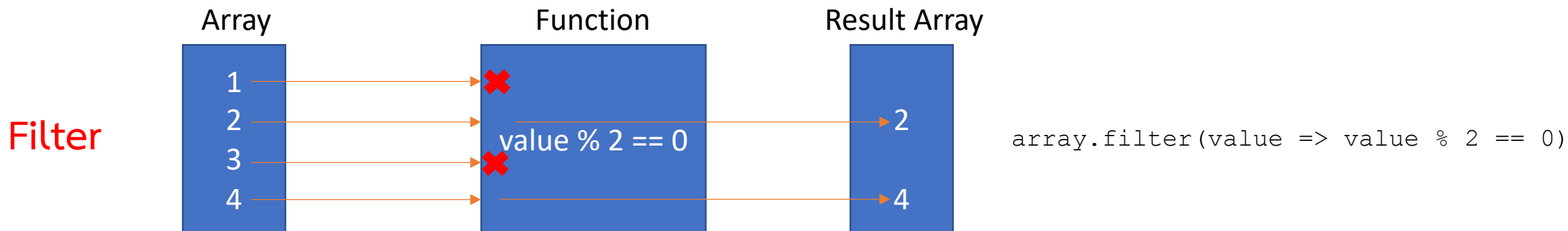
  return (
    <div className="App">
      <header className="App-header">
        <p>Current Amount {amount}</p>
        <button onClick={addTransaction}>Add Transaction</button>
        <TransactionList data={transactionData} onNoteChanged={handleNoteChanged}/>
      </header>
    </div>
  );
}
```



# Delete Button

- เพิ่มปุ่ม Delete เมื่อกดแล้วให้ลบ Row ของรายการบรรทัดที่กดลบข้อมูล

Date-Time	Type	Amount	Note	Action
01/02/2023 - 08:30	รายรับ	20000	allowance	ลบ
01/02/2023 - 10:30	รายจ่าย	500	อาหารเที่ยง	ลบ
02/02/2023 - 12:55	รายจ่าย	1000	อาหารเย็น	ลบ



# React UI Framework

- React เป็นเพียง Library สำหรับช่วยแสดงผลบนหน้าจอเท่านั้น
- ส่วนปรับแต่งการแสดงผลจะเรียกว่า UI Framework ซึ่งมีหลายตัวให้เลือกใช้ เช่น
  - Semantic UI React
  - Material Design
  - Ant Design
  - React Bootstrap
- UI Framework แต่ละตัวมีการเรียกใช้ที่ไม่เหมือนกัน
- ในบทเรียนนี้จะเลือกใช้ Ant Design เนื่องจากความง่ายในการใช้งาน และไม่จำเป็นต้องรู้ CSS มากนัก





# ติดตั้ง Ant Design

- รันคำสั่งต่อไปนี้

```
cd finance-frontend  
npm install antd --save
```

- ทดสอบการติดตั้ง - เปลี่ยนปุ่มไปใช้ Antd

```
import { Button } from 'antd';  
  
function App() {  
  return(  
    <Button  
      type="primary"  
      onClick={handleAddItem}>Add</Button>  
  )  
}
```

A screenshot of a web form on a dark background. At the top is a dropdown menu with the text 'รายรับ' and a downward arrow. Below it is a text input field containing the number '0'. Underneath the input is a label 'Note'. At the bottom right is a blue button with the text 'Add'.

A screenshot of a web form on a dark background. At the top is a dropdown menu with the text 'รายรับ' and a downward arrow. Below it is a text input field containing the text 'จำนวนเงิน'. Underneath the input is a label 'Note'. At the bottom right is a blue button with the text 'Add'.



# สร้าง Component ใหม่ AddItem.js

รายการ ▼

จำนวนเงิน

Note

Add Transaction

Date-Time	Type	Amount	Note	
01/02/2021 - 08:30	รายรับ	20000	allowance	Debug
01/02/2021 - 10:30	รายจ่าย	-150	อาหารเที่ยง	Debug

AddItem.js

Call function  
`onItemAdded(newItem)`

App.js

Props  
- `onItemAdded`

AddItem.js



# สร้าง Component ใหม่ AddItem.js (1)

src/components/AddItem.js

```
import { Button, Form, Select, Input, InputNumber } from 'antd';

export default function AddItem(props) {

  return (
    <Form layout="inline" onFinish={props.onItemAdded}>
      <Form.Item
        name="type"
        label="ชนิด"
        rules={[{ required: true }]}
      >
        <Select
          allowClear
          style={{width:"100px"}}
          options=[
            {
              value: 'income',
              label: 'รายรับ',
            },
            {
              value: 'expense',
              label: 'รายจ่าย',
            },
          ]
        />
      </Form.Item>
```



## สร้าง Component ใหม่ AddItem.js (2)

src/components/AddItem.js

```
<Form.Item
  name="amount"
  label="จำนวนเงิน"
  rules={[{ required: true }]}>
  <InputNumber placeholder="จำนวนเงิน" />
</Form.Item>
<Form.Item
  name="note"
  label="หมายเหตุ"
  rules={[{ required: true }]}>
  <Input placeholder="Note" />
</Form.Item>
<Form.Item>
  <Button type="primary" htmlType="submit">Add</Button>
</Form.Item>
</Form>
)
}
```





# ปรับ App.js และ App.css

src/App.js

```
import AddItem from '../components/AddItem';

function App() {

  const handleAddItem = (itemData) => {
    setTransactionData([...transactionData, {
      id: transactionData.length + 1,
      created: moment().format('DD/MM/YYYY - HH:mm'),
      ...itemData,
    }])
  }

  return (
    <div className="App">
      <header className="App-header">
        <AddItem onItemAdded={handleAddItem}/>

        <TransactionList
          data={transactionData}
          onDelete={handleDeleteItem}
        />
      </header>
    </div>
  );
}
```

src/App.css

```
.App-header {
  background-color: #282e34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}
```



# ปรับ Table ไปใช้ Ant Design

- ศึกษาการใช้งาน Table จาก <https://ant.design/components/table>
- นำ Table มาใช้แทนที่ HTML Table ใน component TransactionList.js
- ศึกษา component อื่นๆ แล้วปรับการแสดงผลให้สวยงาม

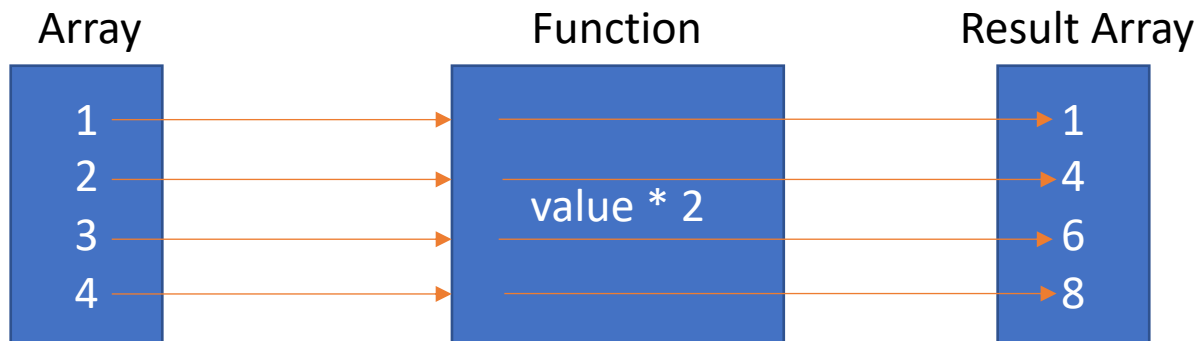
\* ชนิด:  \* จำนวนเงิน:  \* หมายเหตุ:

บันทึก รายรับ-รายจ่าย

Date-Time	Type	Amount	Note	Action
08/12/2023 - 14:38	รายรับ	20000	allowance	<input type="button" value="Delete"/>
08/12/2023 - 14:38	รายจ่าย	500	อาหารเที่ยง	<input type="button" value="Delete"/>
08/12/2023 - 14:38	รายรับ	5000	งานพิเศษ	<input type="button" value="Delete"/>

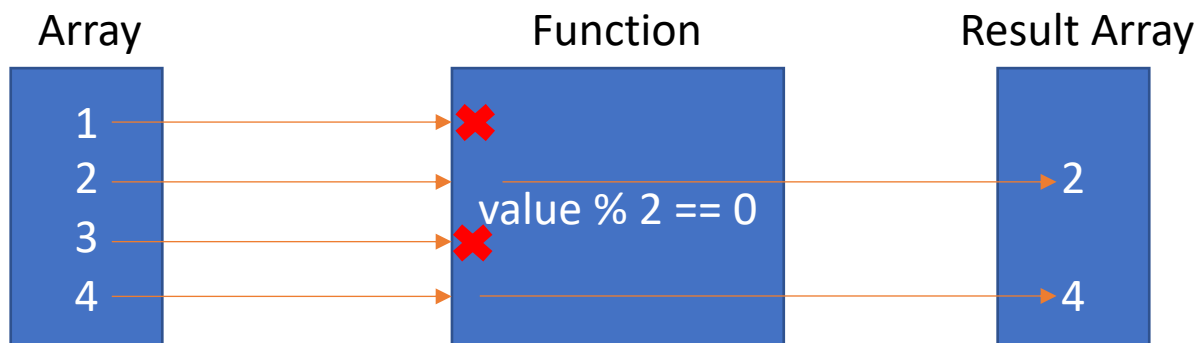
< 1 >

## Map



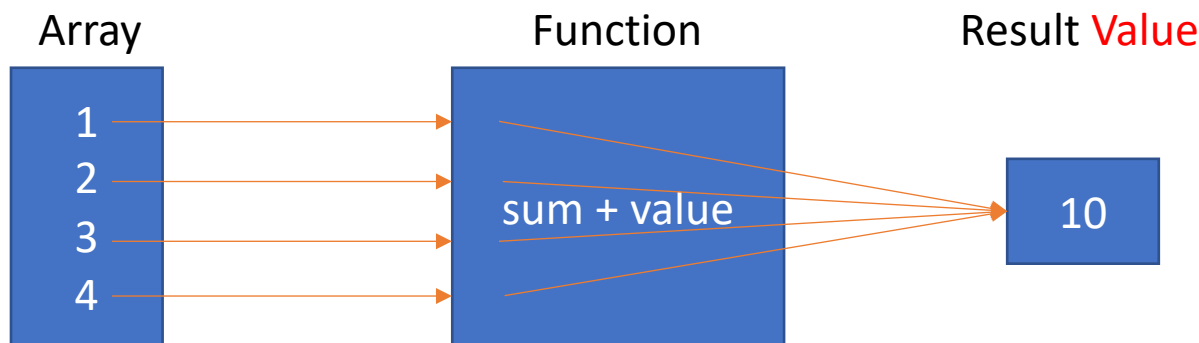
```
array.map(value => value * 2)
```

## Filter



```
array.filter(value => value % 2 == 0)
```

## Reduce



```
array.reduce((sum, value) => sum + value, 0)
```