

Introduction to React

<https://github.com/WasinTh/tutorial-react>

React คือ ?

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



เริ่มสร้าง React Project - T001

- 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
 - index.js คือ
 - App.js คือ source code หน้าแรกของโปรแกรม
 - App.css คือ css default สำหรับแสดงหน้าตัวอย่างของ Sample Application

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!! – T002

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

src/index.js

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



Update Rendered Element – T003

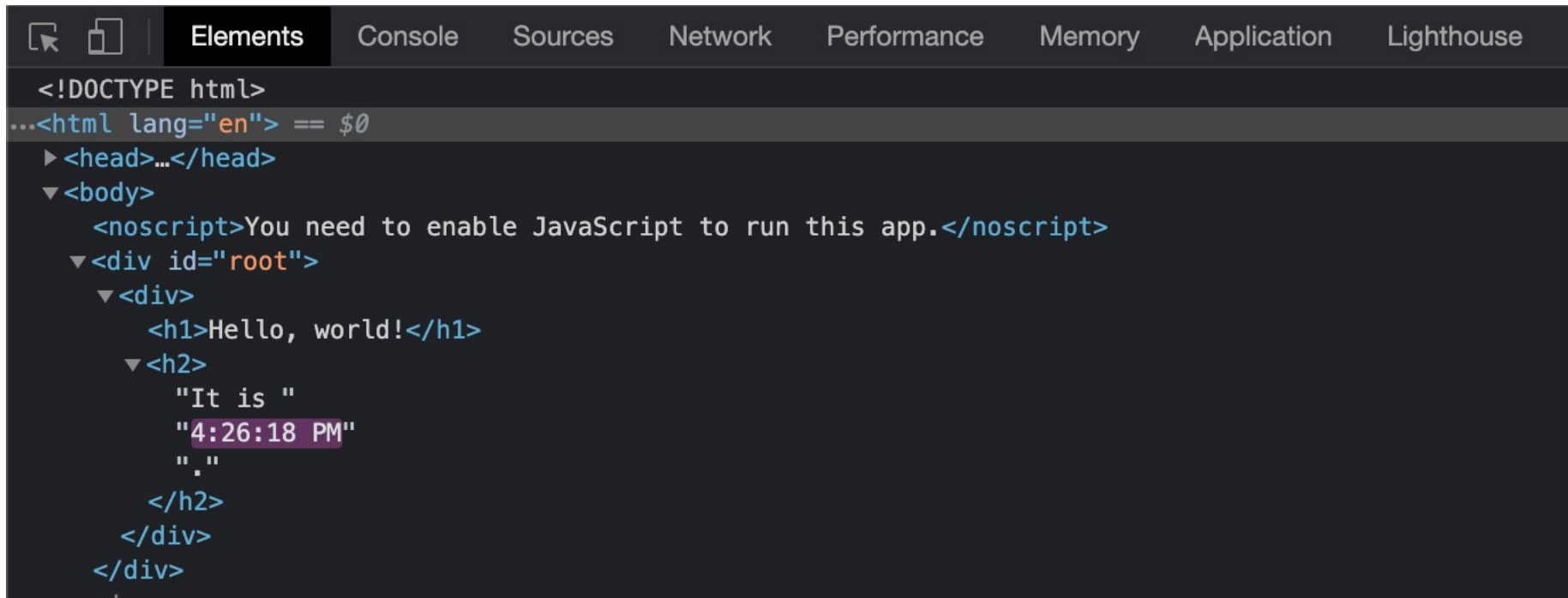
- Element ที่แสดงผลบนหน้าจอของ React
 - เป็น immutable ไม่สามารถแก้ไขค่าได้ แต่จะเป็นการสร้าง element ใหม่มาแสดงผลแทน

src/index.js

```
const showTime = () => {  
  ReactDOM.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

```
ReactDOM.render(  
  <React.StrictMode>  
    <App />  
  </React.StrictMode>,  
  document.getElementById( 'root' )  
);
```

ต่อหน้าถัดไป



ทดลองใช้ App.js Component (2) – T004

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>  
);  
}
```



ประกาศ Component ด้วยตนเอง – T005

- 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) – T006

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>{transaction.note}</td>
        <td>
          <button onClick={() => alert(JSON.stringify(props))}>Debug</button>
        </td>
      </tr>
    ))
  }
  else {
    return null;
  }
}

return(
  <table>
    <tr>
      // ...
      <th>debug</th>
    </tr>
  </table>
)
```




Conditional Rendering – T007

- 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) – T008

- การใช้งานที่พบบ่อยอีกแบบคือ Inline If แบบ Conditional Operator

src/App.js

```
export default function Customer(props) {  
  
  // ...  
  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) – T009

- ตัวอย่างการใช้งาน Inline If คู่กับ && Operator

src/App.js

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



TransactionList Background Color – T010

- 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 – T011

- 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)}  
        {counter < 3 && poorGreeting(counter)}  
        <button onClick={counterClicked}>Add Counter</button>  
        <TransactionList data={transactionData}/>  
      </header>  
    </div>  
  );  
}
```

ค่าของตัวแปรไม่เปลี่ยน
เนื่องจาก React ไม่ทราบ
ว่ามีการเปลี่ยนแปลงค่า
และต้อง reload DOM



State Hook (2) – T012

```
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 >= 3 && richGreeting(counter)}
        {counter < 3 && 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 – T013

- 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 (
    <h3>
      It is {time.toLocaleTimeString()}.
    </h3>
  )
}
```

src/App.js

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

function App() {
  // ...

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

กฎหลัก 2 ข้อของ Hooks

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



Add Transaction Button – T014

- 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: '',  
    }  
  }  
}
```

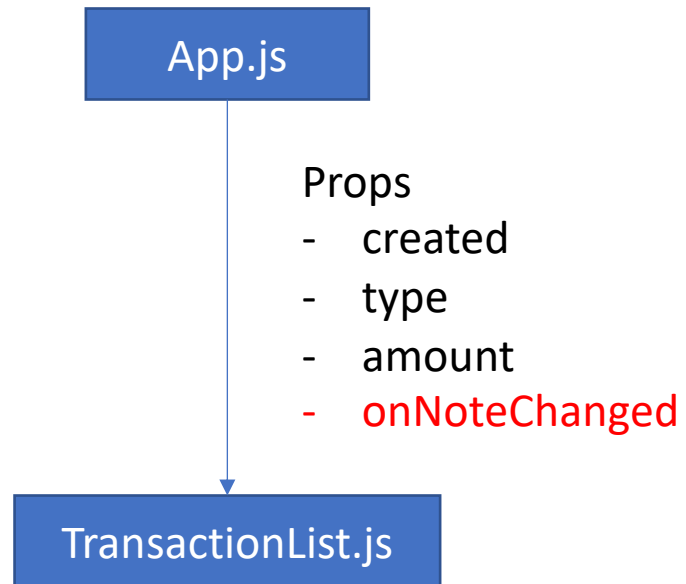
Current Amount 19779

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	

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) – T015

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>
  );
}
```



TransactionCreate Component – T016

- สร้าง Component สำหรับเพิ่ม Transaction ใหม่
 - ชื่อ TransactionCreate.js
- Component นี้เป็นแบบฟอร์มให้กรอกรายละเอียดเกี่ยวกับ Transaction ได้แก่
 - ชนิดว่าเป็น INCOME (รายรับ) หรือ EXPENSE (รายจ่าย) (type)
 - จำนวนเงิน (amount)
 - Note (note)
- หลังจากกรอกแล้วให้ส่งข้อมูลมายัง App.js และ update ข้อมูลใน TransactionList

มีต่อหน้าถัดไป



TransactionCreate Component (TIPS)

- สามารถใช้ `useState` เพื่อเก็บ state ของ form
- ใช้ HTML `<form>` tag
 - ใช้ function `onSubmit` ในการบันทึกข้อมูล และใช้ `evt.preventDefault()` เพื่อป้องกันการเปลี่ยนหน้าของ form จริงๆ
 - ใช้ `input type=submit` ในการสร้างปุ่มบันทึก

```
const onSubmitForm = evt => {  
  evt.preventDefault();  
  //...  
}  
  
return(  
  <form onSubmit={onSubmitForm}>  
    <input name='myInput' />  
    <input type='submit' value='Submit'></input>  
  </form>  
)
```

Current Amount 20350				
รายรับ	▼	0	◆	Submit
Date-Time	Type	amount	note	debug
01/02/2021 - 08:30	รายรับ	20000	allowance	Debug
01/02/2021 - 10:30	รายจ่าย	150	อาหารเที่ยง	Debug
4/5/2021, 2:05:32 PM	รายจ่าย	500	ซื้อเสื้อ	Debug
4/5/2021, 2:05:53 PM	รายรับ	1000	งานพิเศษ	Debug

React Back-End Integration

HTTP Request

- การเชื่อมต่อระหว่าง React Application และ Web Application จะทำผ่านทาง HTTP Request
 - ข้อมูลที่รับและส่ง จะอยู่ในรูปแบบของ JSON (REST API)
- การทำ HTTP Request จาก React Application สามารถทำได้หลายช่องทางได้แก่
 - Fetch API เป็น Library มาตรฐานสำหรับส่ง HTTP Request
 - Axios เป็น 3rd party library สำหรับส่ง HTTP Request
- ในเอกสารนี้จะนำเสนอการใช้ Axios เนื่องจากมีความง่ายในการใช้งานมากกว่า

```
npm install axios --save
```



JWT Authentication – T017

src/component/Login.js

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

export default function Login(props) {
  const [state, setState] = useState({username: '', password: ''});

  const onInputChanged = evt => {
    setState({
      ...state,
      [evt.target.name]: evt.target.value
    })
  }

  const onLogin = evt => {
    evt.preventDefault();
    axios.post('http://localhost:8000/api-token-auth/', {username: state.username, password: state.password})
      .then(response => {
        props.onLoginSuccess(response.data.token);
      })
      .catch(err => alert(err))
  }

  return (
    <form onSubmit={onLogin}>
      <p>Username <input name='username' value={state.username} onChange={onInputChanged}/></p>
      <p>Password <input type='password' name='password' value={state.password} onChange={onInputChanged} /></p>
      <input type='submit' value='Login'></input>
    </form>
  )
}
```



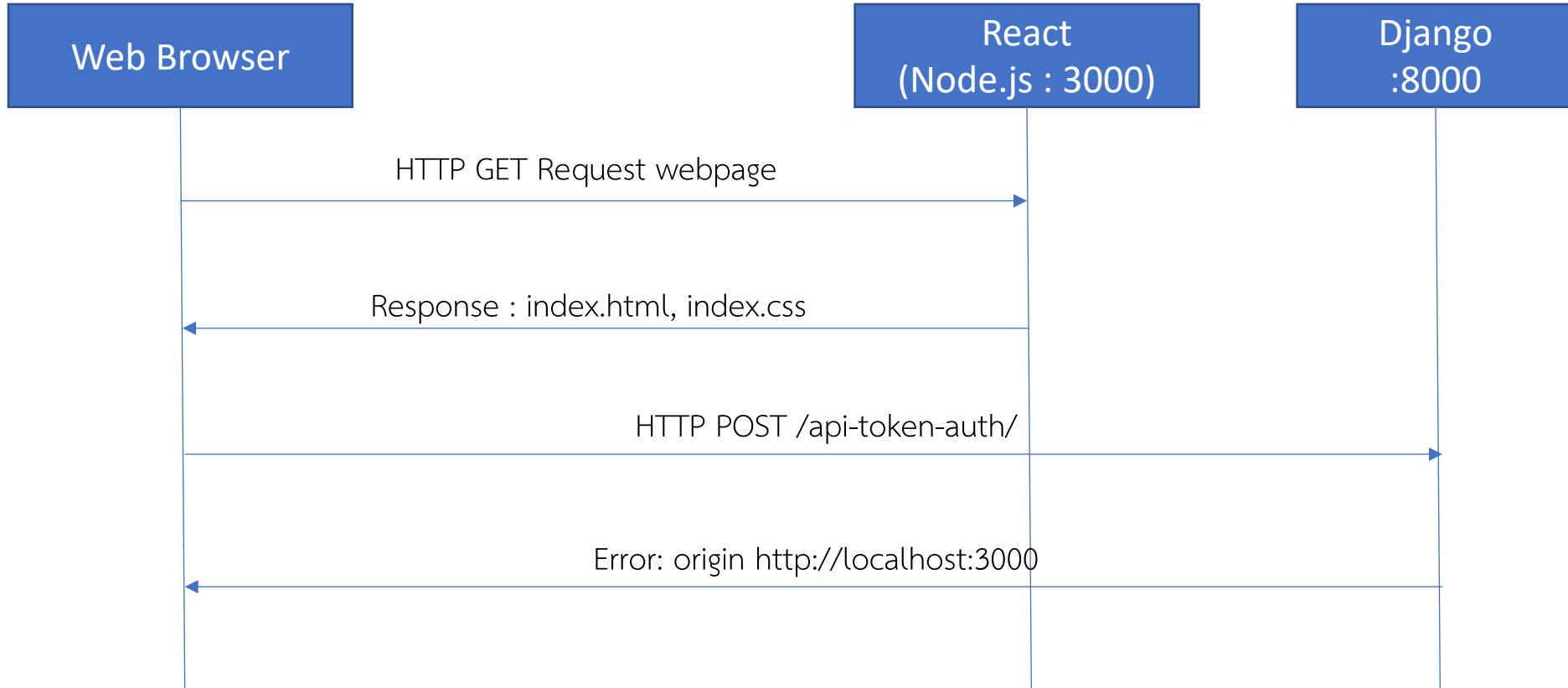
JWT Authentication (2)

src/App.js

```
function App() {  
  // ...  
  const [token, setToken] = useState(null);  
  return (  
    <div className="App">  
      <header className="App-header">  
        {!token && <Login onLoginSuccess={token => setToken(token)} />}  
        {token &&  
          <div>  
            <p>Current Amount {amount} </p>  
            <TransactionCreate onCreate={data => addTransaction(data)} />  
            <TransactionList data={transactionData} />  
          </div>  
        }  
      </header>  
    </div>  
  );  
}
```

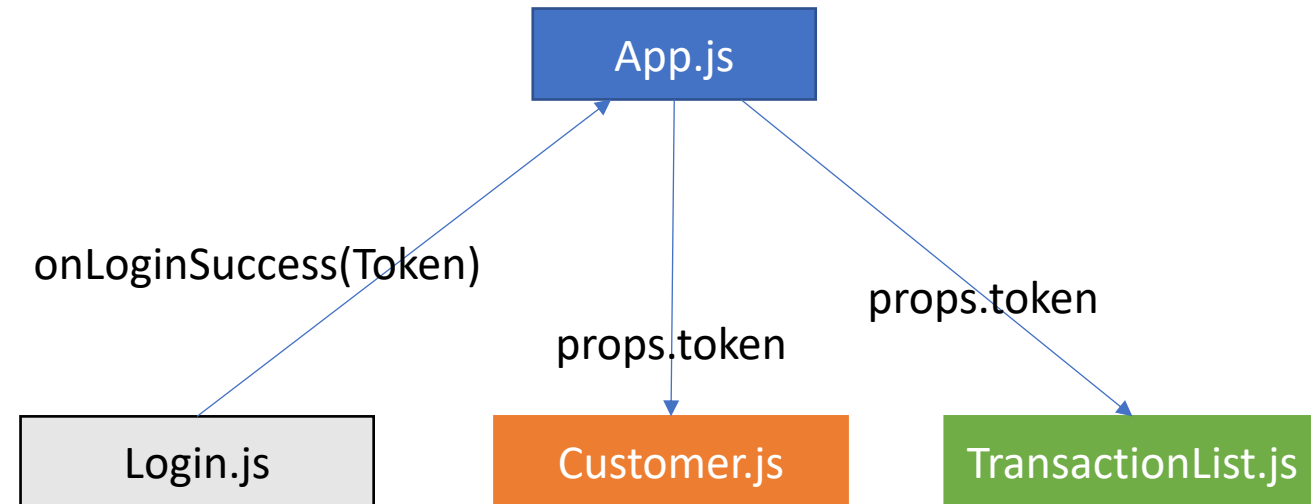
Cross-Origin Resource Sharing (CORS)

```
✖ Access to XMLHttpRequest at 'http://localhost:8000/api-token-auth' from origin 'http://localhost:3000' has been blocked by CORS policy: Response to preflight request doesn't pass access control check: Redirect is not allowed for a preflight request. localhost/:1
✖ ▶ POST http://localhost:8000/api-token-auth net::ERR_FAILED xhr.js:177
✖ ▶ Uncaught (in promise) Error: Network Error createError.js:16
    at createError (createError.js:16)
    at XMLHttpRequest.handleError (xhr.js:84)
```



การดึงข้อมูลจาก Server

- หลังจากได้รับ Token มาแล้ว ขั้นตอนต่อไปจะเป็นการดึงข้อมูลจาก server โดยใช้ Token ในการยืนยันตัวตน



Mr.A - 1000 ▼

Date-Time	Type	Amount	Note



ดึงข้อมูล Customer จาก Server (2) – T018

src/component/CustomerSelect.js

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

export default function CustomerSelect(props) {
  const [customers, setCustomers] = useState([]);

  useEffect(() => {
    axios.get('http://localhost:8000/account/customer-views/', {
      headers: {
        'Authorization': `Bearer ${props.token}`
      }
    })
    .then(response => {
      setCustomers(response.data.map( customer =>
        <option key={customer.id} value={customer.id}>
          {customer.name} - {customer.current_amount}
        </option>
      ));
    })
    .catch(err => alert(err));
  }, [props]);

  return(
    <select name="customers" onChange={evt => props.onCustomerSelected(evt.target.value)}>
      <option disabled selected value> Select Customer </option>
      {customers}
    </select>
  )
}
```



ดึงข้อมูล Customer จาก Server (2)

src/App.js

```
const handleCustomerChanged = customerId => {
  console.log(customerId)
}

return (
  <div className="App">
    <header className="App-header">
      { !token && <Login onLoginSuccess={ token => setToken(token)} /> }
      { token &&
        <div>
          <CustomerSelect token={token} onCustomerSelected={handleCustomerChanged} />
          <p>Current Amount {amount} </p>
          <TransactionCreate onCreated={data => addTransaction(data)} />
          <TransactionList data={transactionData} />
        </div>
      }
    </header>
  </div>
);
```




ดึงข้อมูล Transaction จาก Server – T019

- แก้ไข App.js และ TransactionList.js ให้ดึงข้อมูลจาก Server และแสดงผลให้ถูกต้อง ตาม customer ที่ได้เลือกมาจาก combobox
 - รับ Customer ID มาจาก CustomerSelect.js
 - ยิง GET ไปที่ URL `http://localhost:8000/account/transaction-viewssets/?customer__id=???`
 - นำ Response ที่ได้จาก transaction-viewssets มาแสดงผล

admin - 1980 ▼			
Date-Time	Type	amount	note
2021-03-20T16:01:00Z	รายรับ	1000	ทดสอบ
2021-03-23T10:09:49Z	รายรับ	1000	เก็บเงินได้จากถนน
2021-03-23T10:10:10Z	รายจ่าย	20	ซื้อลูกชิ้น

PropTypes

- ใช้สำหรับประกาศว่าของ Component ที่ประกาศขึ้นมามี props อะไรบ้าง และมีชนิดอะไร
- <https://www.npmjs.com/package/prop-types>

```
npm install prop-types --save
```

```
import PropTypes from 'prop-types';  
export default function CustomerSelect(props) {  
  // ...  
}  
  
CustomerSelect.propTypes = {  
  token: PropTypes.string,  
  onCustomerSelected: PropTypes.func  
}
```