Lab 4 - Intro to React JS

Introduction to React

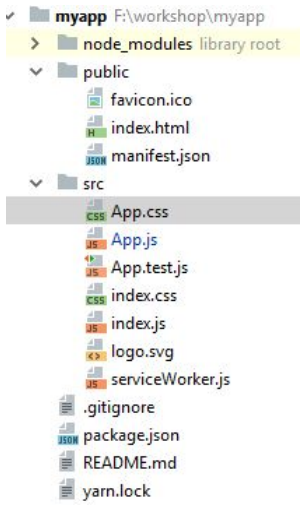
ตัวอย่างโปรแกรมการเขียน React เริ่มต้น

```
<html>
   <head>
       <script src="https://unpkg.com/react@16/umd/react.development.js"
crossorigin></script>
       <script src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
crossorigin></script>
   </head>
   <body>
       <div id="root"></div>
       <script>
           ReactDOM.render('Hello world!', document.getElementById('root'));
       </script>
   </body>
</html>
```

สังเกตได้ว่า ในส่วนของเอกสาร มี div ที่ชื่อ root อยู่ จากนั้น ใช้คำสั่ง render เพื่อแทนข้อความ ใส่ใน div ที่ตำแหน่งนั้น อย่างไรก็ดี การใช้งาน React นิยมใช้คู่กับ JSX ซึ่งต้องอาศัย Babel และเทคโนโลยีอื่น ๆ เช่นการ compress, minification, server-production environment การ setup React project จึงมีขั้นตอนหลายอย่าง ด้วยเหตุนี้ จึงมีการใช้เครื่องมือ เช่น create-react-app ผ่าน npx ในการสร้างโครงสร้างของโปรแกรมขึ้นมาก่อน

Create React App

สร้าง React project โดยใช้คำสั่ง create-react-app my-app หรือถ้าในเครื่องติดตั้ง npm ตั้งแต่ version 5.2 ขึ้นไป จะมี npx มาให้ด้วย




```
npx create-react-app myapp
```

โครงสร้างของ React App ในภาพรวม จะประกอบไปด้วย

- node_modules - เก็บโมดูล ไลบรารี่ต่าง ๆ ของโปรแกรม
- public - เก็บ file index.html เป็น จุดเริ่มต้นของโปรแกรม
- src - เก็บไฟล์ javascript และ css ทั้งหมดที่เราจะทำงานในนี้ โดยเริ่มที่ index.js จะไปเรียกใช้งาน โดยดึงข้อมูลจาก App.js
- .gitignore - ไฟล์แสดงรายชื่อไฟล์ที่ไม่ต้องการ upload ขึ้น git
- package.json - เก็บ dependenc (module) ทั้งหมดที่ใช้ใน project นี้ รวมถึงกำหนด start, stop, test script ในการสั่งให้โปรแกรมทำงาน
- yarn.lock - เก็บรายชื่อ package ที่ติดตั้งไปแล้ว (หากมี dependency เพิ่มจะตรวจสอบจาก yarn.lock กรณีใช้ yarn ในการติดตั้ง หากใช้ npm ก็จะมี package.json.lock)

State และ Event

ตัวอย่างการใช้ state ด้วยโปรแกรม count ทดลองแก้ไฟล์ App.js ดังนี้

\myapp\App.js

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

class App extends Component {
 state = { count:0 }
 constructor(props) {
   super(props)
   this.add = this.add.bind(this)
 }

 add = function() {
   this.setState({count:this.state.count+1})
 }

 render() {
   return (
     <div className="App">
       <h1>Counter</h1>
         {this.state.count} <br/>
        <button onClick={this.add}> Add </button>
        <button> Minus </button>
     </div>
   );
 }
}
export default App
```

4.1 ลอง comment คำสั่ง // `this.add = this.add.bind(this)` จากนั้นลองเปิดหน้า web และกดปุ่มเพื่อเพิ่มการนับอีกครั้ง มีการเปลี่ยนแปลงอย่างไรบ้าง เพราะเหตุใด?

ตอบ

4.2 ลองแก้ ฟังก์ชัน add เป็นแบบ arrow function โดยที่ยัง comment ในข้อ 4.1 เหมือนเดิม มีการเปลี่ยนแปลงอย่างไรบ้าง และ เพราะเหตุใด?

ตอบ

4.3 ลองเขียนฟังก์ชัน delete เพื่อ ลดค่า count และจัดหน้าการแสดงผลให้สวยงาม ในไฟล์ App.css

ตอบ

Components และ Props

ทดลองสร้างโปรแกรม Todo เพื่อส่ง list รายการ จาก Main component ไปยัง child component โดยการแบ่ง component ออกเป็น 3 ส่วนด้วยกัน คือ App.js, TaskList.js และ InputTask.js ดังนี้

- App.js เป็น component หลัก เก็บข้อมูล Task ในรูปของ JSON ประกอบไปด้วย id และ task (ใน state) รวมถึงเชื่อมโยงไปยัง TaskList.js และ InputTask.js โดยส่งผ่าน props ไป
- TaskList.js เป็น component ที่ใช้แสดงผล tasks ที่ถูกส่งมาจาก App.js
- InputTask.js เป็น component ที่ใช้ render ตัว input form เพื่อรับข้อมูล task ใหม่ ส่งไปยัง App.js และ render ใหม่

src/App.js

ให้สังเกต object tasks (this.state.tasks) ที่ถูกส่งมาจาก App.js ไปยัง TaskList.js ในรูปแบบของ props

```
import React, {Component} from 'react';
import './App.css';
import TaskList from './todo/TaskList'
import InputTask from "./todo/InputTask";

class App extends Component {

   state = {
        tasks: [{id: 1, task: 'Do homework'},
                {id: 2, task: 'Read book'}],
        id:3
   }

   addTask = (task) => {
        this.setState({
                  tasks: [...this.state.tasks, {id: this.state.id,task } ],
                  id: this.state.id+1   })
   }

   render() {


        return (
            <div className="App">
                <h1>Todo</h1>
                <TaskList tasks={this.state.tasks}/>
                <InputTask addTask={this.addTask} id={this.state.id}/>
                <br/>
            </div>
        );
   }
}

export default App;
```

src/todo/TaskList.js

ใน TaskList รับ property tasks (this.props.tasks) จาก App.js และ นำข้อมูลไปแสดงผลแบบ bullet

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

class TaskList extends Component {
   render() {
       if ( this.props.tasks )
           return (<ul > {
                   this.props.tasks.map((item) => (
                       <li key={item.id}> {item.task} </li>
                   ))
               }
           </ul>)
   }
}

export default TaskList
```

src/todo/InputTask.js

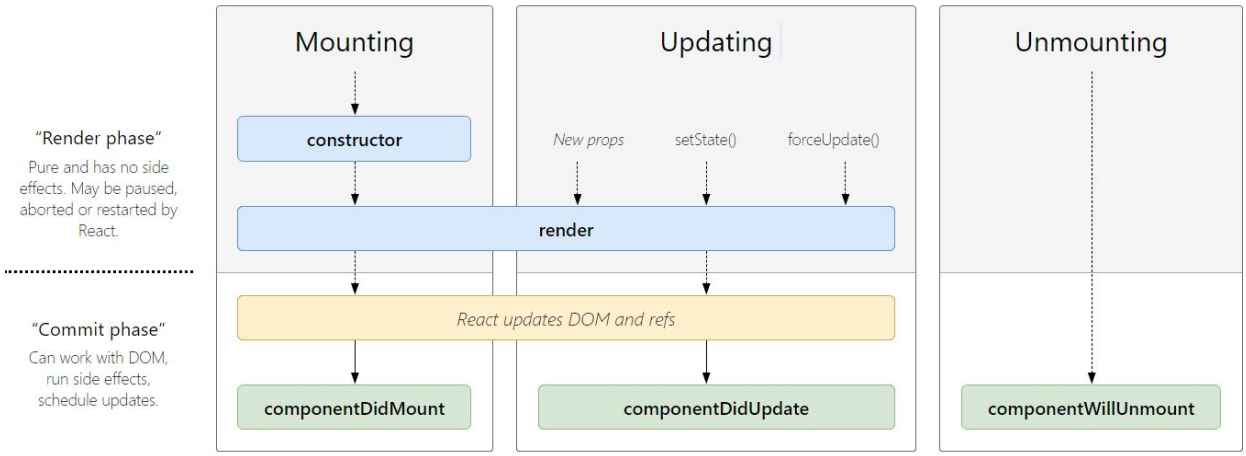
ใน InputTask จะกำหนดค่า task (เป็น state) ไว้ใช้ เพื่อเก็บข้อมูลที่ป้อนใน input box จากนั้น เวลา update ค่า tasks (ใน App.js) จะ ถูกเรียกใช้ โดยส่ง task เป็น พารามิเตอร์ ผ่าน method this.props.addTask(...) เพื่อให้ component update tasks และ re-render ใหม่ โดยอัตโนมัติ

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

class InputTask extends Component {

   state = {task:''}

   handleChange = (e) => {
       this.setState({ [e.target.name] : e.target.value });
   }

   render() {
       return (
           <div>
               <input type="hidden" name="id" value={this.props.id} /><br/>
               <input type="text" name="task" onChange={this.handleChange} /> <br/>
               <button onClick={() =>
this.props.addTask(this.state.task)}>Add</button>
           </div>
       )
   }
}

export default InputTask
```

4.4 จากโปรแกรม Todo list จงแก้ไขให้โปรแกรมสามารถเพิ่ม ข้อมูลอื่น ๆ นอกจาก task เช่น สถานที่จัดกิจกรรม, วันที่, เวลา หรือ อื่น ๆ ที่เหมาะสม

ตอบ

React Lifecycle



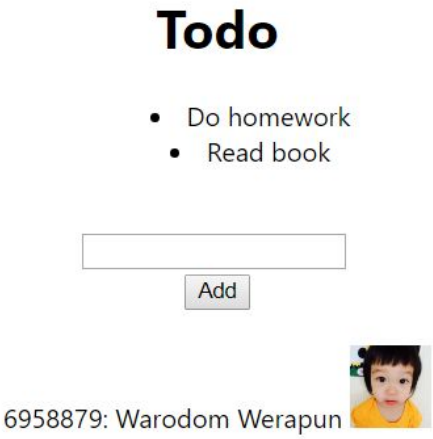
From: http://projects.wojtekmaj.pl/react-lifecycle-methods-diagram/

วงจรชีวิตการทำงานของ React ในปัจจุบันจะแบ่งออกเป็น 3 Phase ใหญ่ ๆ คือ Mouting, Updating และ Unmounting

ส่วนที่ใช้งานหลัก ๆ คือส่วน mouting ไม่ว่าจะเป็นการ Initialize กำหนดค่าเริ่มต้นให้กับตัว object ต่าง ๆ ผ่าน constructor การ render() (เป็น method ที่บังคับให้มีใน class component) และ สั่ง componentDidMount ซึ่งหมายถึงหลังจากที่ render() DOM ต่าง ๆ ในหน้าเพจเรียบร้อยแล้ว ก็สามารถสั่ง request API ต่าง ๆ เพื่อ update DOM และ re-render อีกครั้ง ซึ่งตรงนี้จะเข้าสู่กระบวนการ updating และไปเรียก componentDidUpdate

ในส่วน componentWillUnmount จะถูกเรียกหลังจากที่ component นั้น ไม่ถูกใช้อีกต่อไป โดยส่วนมากจะใช้สำหรับการยกเลิกการติดต่อ จัดการเคลียร์ค่าต่าง ๆ เช่น ยกเลิกการร้องขอการใช้ระบบเครือข่าย หรือการเชื่อมต่อต่าง ๆ ที่กำหนดไว้ใน componentDIdMount ข้อสำคัญคือ ไม่ควรเรียกใช้ setState() ใน componentWillUnmount() เพราะว่า component นั้นจะไม่ถูก render() ใหม่อีกครั้งหลังจากที่ instance ของ component นั้นถูก unmount แล้ว มันจะไม่ถูก mounted ใหม่อีกครั้ง

HTTP Request



React สามารถ ร้องขอ HTTP Request ไปยัง server (backend) เพื่อนำข้อมูลมาแสดงผลได้ โดยใช้ library axios

การติดตั้ง ด้วยคำสั่ง *npm i --save axios*

ในตัวอย่างนี้ จะให้ดึงข้อมูล Github ของผู้ใช้ผ่าน Github API ดังนี้

โดยเมื่อ Render หน้า page มาจะดึงข้อมูลเกี่ยวกับ Github API

ตัวอย่างเช่น https://api.github.com/users/wwarodom


```
{
            login: "wwarodom",
            id: 6958879,
            node_id: "MDQ6VXNlcjY5NTg4Nzk=",
            avatar_url: "https://avatars2.githubusercontent.com/u/6958879?v=4",
            gravatar_id: "",
            url: "https://api.github.com/users/wwarodom",
            html_url: "https://github.com/wwarodom",
            followers_url: "https://api.github.com/users/wwarodom/followers",
            following_url:
            "https://api.github.com/users/wwarodom/following{/other_user}",
            gists_url: "https://api.github.com/users/wwarodom/gists{/gist_id}",
            starred_url:
            "https://api.github.com/users/wwarodom/starred{/owner}{/repo}",
            subscriptions_url: "https://api.github.com/users/wwarodom/subscriptions",
            organizations_url: "https://api.github.com/users/wwarodom/orgs",
            repos_url: "https://api.github.com/users/wwarodom/repos",
            events_url: "https://api.github.com/users/wwarodom/events{/privacy}",
            received_events_url:
            "https://api.github.com/users/wwarodom/received_events",
            type: "User",
            site_admin: false,
            name: "Warodom Werapun",
            company: "CoE, PSU",
            blog: "http://plex.coe.psu.ac.th/",
            location: "Phuket, Thailand",
            email: null,
            hireable: null,
            bio: "CoE Lecturer",
            public_repos: 40,
            public_gists: 14,
            followers: 76,
            following: 18,
            created_at: "2014-03-15T10:42:07Z",
            updated_at: "2019-01-13T17:32:55Z"
}
```

เพิ่มการเรียกใช้ Github component ดังนี้ (ไฟล์ src/github/index.js)

```
import Github from "./github";
...

render() {
   return (
       <div className="App">
           <h1>Todo</h1>
           <TaskList tasks={this.state.tasks}/>
           <InputTask addTask={this.addTask} id={this.state.id}/>
           <br/>
           <Github/>
       </div>
   );
}
...
```

สร้าง /src/github/index.js เพื่อดึงข้อมูลจาก github

```
class Github extends Component {

   state = { user: 'wwarodom', data:''}

   componentDidMount = () => this.fetchUser(this.state.user)

   fetchUser = (USER) => {
       axios.get(`https://api.github.com/users/${USER}`)
           .then(response => {
               this.setState({data: response.data})
               console.log(response.data)
           })
   }

   render() {
       const {data} = this.state
       if (data)
           return ( <div>{data.id}: {data.name} <img src={data.avatar_url}
alt="avatar" width="50px"/> </div> )
       return (<div>.</div>);
   }
}

export default Github
```

4.5 จงแก้ไขโปรแกรมให้รับค่า Github user จากผู้ใช้ แล้วสั่งให้ axios ไป query ข้อมูลของผู้ใช้คนนั้น มาแสดงผล

ตอบ

4.6 จงเขียน component เพิ่มมาอีก 1 component ที่สามารถเรียกใช้ API จากที่นักศึกษาหามาเอง (เช่น https://openweathermap.org/api) พร้อมทั้งแสดงผล และ จัดการแสดงผลให้สวยงาม โดยใช้ CSS หรือ Bootstrap จาก ครั้งที่แล้ว

ตอบ