Web Programming 03/28

* React.js (Part I)

Class Preparations

- (Must) Install “create-react-app“
  - (建議) node.js & npm 要先安裝
  - npm install -g create-react-app
  - 如果真的裝不起來，只好先用 CodePen
- (強烈建議) Install “eslint”
  - (建議) 裝完後跑 “eslint --init”, 可選擇 “Use a popular style guide” —> “Airbnb”
  - (建議) 根據你習慣用的 editor, 安裝 eslint 的 plugin package, 並且到設定處啟用它

Intro to React.js

- React is a JavaScript library.
- JavaScript 裡頭對於 DOM element 的產生與操作是很慢的；React 則使用了 Virtual DOM 的概念，去 monitor 頁面改變的地方，而當改變發生的時候，只重新 render 改變部分，因此，可以大幅提升畫面更新的速度
  - Virtual DOM 是由 React elements 所組成，而每個 element 是用 JSX 語法描述，長得就像是個 DOM node，但事實上是個 plain JS object
    - const element = <h1>Hello, world</h1>;

JSX in React Element

- 一個用 JSX 寫的 React element 基本上就長得像個 HTML 的 node
  - 可以巢狀或是加上 attribute：
    const element = … ;
- 但如果多行，最好用 ( ) 括起來，免得造成 ';' 的誤解

```
const element = (
  <div>
    <h1 id="hello">Hello, world!</h1>
  </div>
);
```

- 如果沒有 child, 可以(建議)用 " />" 來 close.
  - ****
  - **<input … />**
  - **<App />**

JSX in React Element

- 可以在適當的地方用 {...} 插入任何 JavaScript 的 expressions
  - ```
    const element = (
      <h1>
        Hello, {iAmAFunction(pp)}!
      </h1>
    );
    ```
- ```
  const element = <img src={user.avatarUrl}></img>;
  ```
- ```
  const element = <p> 2 + 3 = { 2+3 } </p>
  ```
- ```
  const title = response.potentiallyMaliciousInput;
  // This is safe:
  const element = <h1>{title}</h1>;
  ```

Intro to React.js

- 用 React element 描述好 Virtual DOM 之後，用 ReactDOM 去 render 畫面：
  - ```
    import React from 'react';
    import ReactDOM from 'react-dom';
    var element = Hello World! ;
    ReactDOM.render(element, document.getElementById('root'));
    ```
- Save the above file as “index.js”. Create an HTML “index.html”:
  - ```
    <head></head>
    <body>
      <div id='root'></div>
      <script type='text/javascript' src='index.js'></script>
    </body>
    ```
- 用瀏覽器打開 index.html. Will this work?
  - No. 瀏覽器看不懂 JSX.
  - 需要把 JSX compile 成 JS (e.g. Babel)

Converting JSX to JS

- 先確定 Babel 有安裝好
- npm install --save-dev babel-plugin-transform-react-jsx
- babel --plugins transform-react-jsx
- // 先把 index.js rename 成 index.jsx
  babel --plugins transform-react-jsx index.jsx > index.js
  - ```
    import React from 'react';
    import ReactDOM from 'react-dom';
    var element = React.createElement(
      'p',  null,  'Hello World!'
    );
    ReactDOM.render(element, document.getElementById('root'));
    ```
- Still doesn't work….
  ==> Need to properly import "JavaScript modules" (Cover later)
    ```

Intro to React.js

- 這是為什麼我們先用 “create-react-app” 的原因
  - It takes care of Babel, JSX to JS, React, ReactDOM… etc. under a node.js framework.
- After installation, in command line, do —
  - create-react-app test1
  - cd test1
  - // take a look at “package.json”
  - (Recommded) eslint --init // 選 airbnb, js
  - // check “package.json” again!!
  - npm start
  - // replace “src/index.js” with the index.js in the previous page.
  - npm start // Your first “hello world” react app!

- 到目前為止，使用 React 看起來只是像在 JS 裡面寫 HTML。
- 我們接下來要來了解 —
  - 如何利用 function / class 來定義 element 的原型 (i.e. component)
  - 傳參數給 component 的 properties, 以控制 component 呈現出來的頁面
  - 定義 component 內部的 (private) states，讓下一個瞬間呈現出來的頁面，可以受到前一個時間的 states 所控制 (i.e. stateful)
  - 綁定 event 與 handler，完成互動式網頁
  - Thinking in React.

React Component

- 就像是 JavaScript 利用 function/class 來定義一個 object 的 prototype; React 利用 function/class 來定義 React element 的 prototype

  // 注意：props 是保留字，用來指定 React component 的 properties,

  // 不要隨便換名字

  1. ```
     function Welcome(props) {
       return <h1>Hello, {props.name}</h1>; }
     ```
  2. ```
     import React, { Component } from 'react';
     class Welcome extends React.Component {
       render() { return <h1>Hello, {this.props.name}</h1>; }
     }
     ```

- 依 component 產生 element
  - ```
    const element = <Welcome name="Ric" />;
    ReactDOM.render(element,
    document.getElementById('root'));
    ```
    ```

React Practice #1

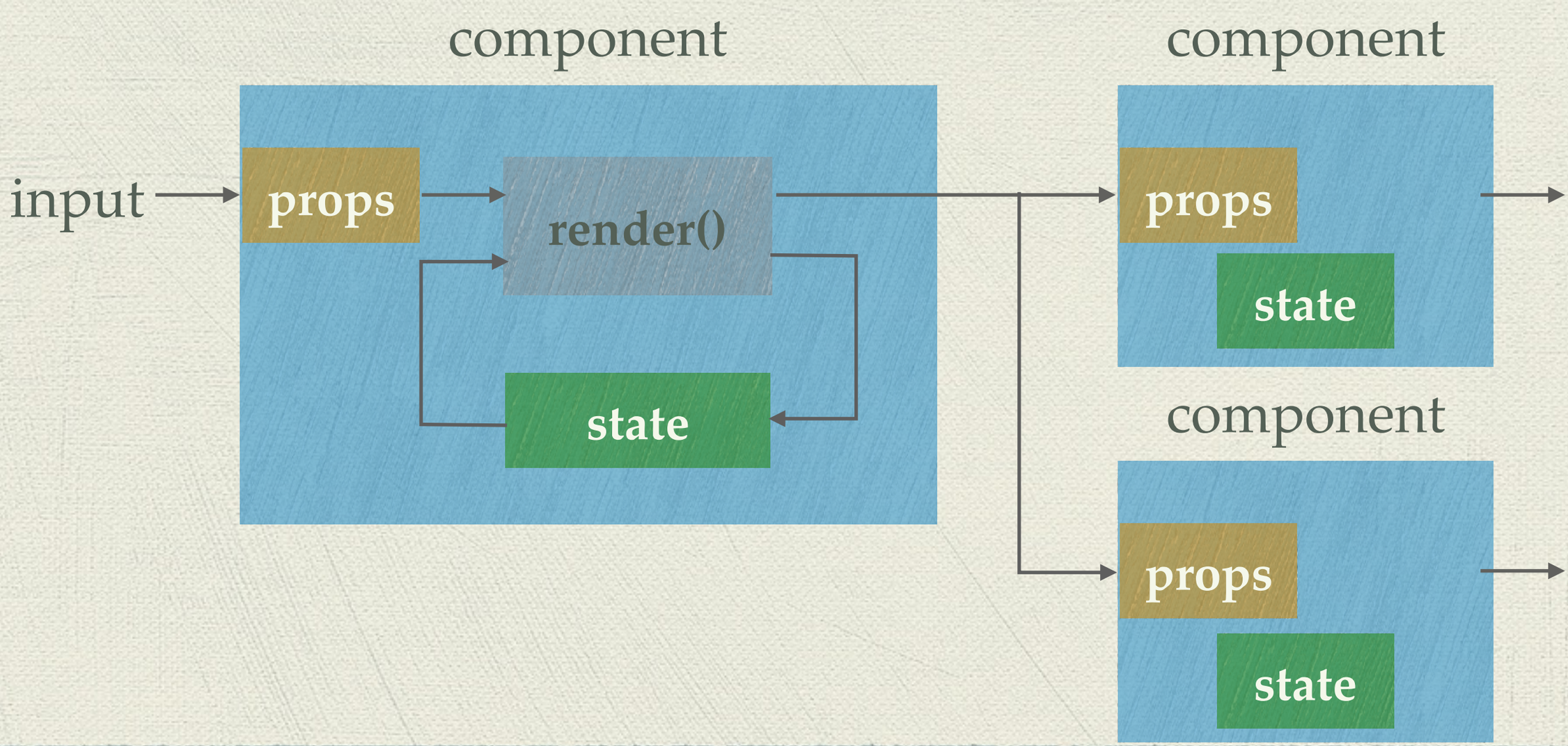
- Write an HTML file for a table like this:

Ric's score

Subject	Score
Math	100
Chinese	87

- Write an object to represent the data for the table. Note the number of subjects may vary.
  - ```
    const schoolRecord =
    { name: 'Ric',
      records:[
        {subject: 'Math', score:100},
        {subject: 'Chinese',score: 87}]
    };
    ```
    ```

React Practice #1

Ric´s score

Subject	Score
Math	100
Chinese	87

- Assume 'record' is dynamically received from server. Let's write a React app that can display the record accordingly:
  - **`const schoolRecord = …; // received from server`**
    `ReactDOM.render(`
      `<ScoreCard record={schoolRecord} />,`
      `document.getElementById('root'));`

- What will be the React components to constitute the table?
  - ScoreCard — top-level component { props.scoreCard }
    - Caption — table caption { props.name }
    - SubjectList — table body { props.subjectList }
      - Subject — table row { props.subject; props.score }

Now, let’s complete the React App!!

All React components must act like pure functions with respect to their props.

- Pure function: 不會去改變它的 input parameter (i.e. read-only)
  - `function sum(a, b) { return a + b; }`
- Impure function: input parameter 會被改變
  - `function inc(counter) { return counter++; }`
- 所以你不可以...
  - ```
    class Subject extends React.Component {
      render() {
        this.props.totalScore += this.props.score;
        return (…);
    }}
    ```
    ```

所以如果想要在 component 裡頭 “記住” 一些資訊(e.g.按讚數量,目前計算結果...)，用來增加網頁的互動，則需要用 “state”



React State

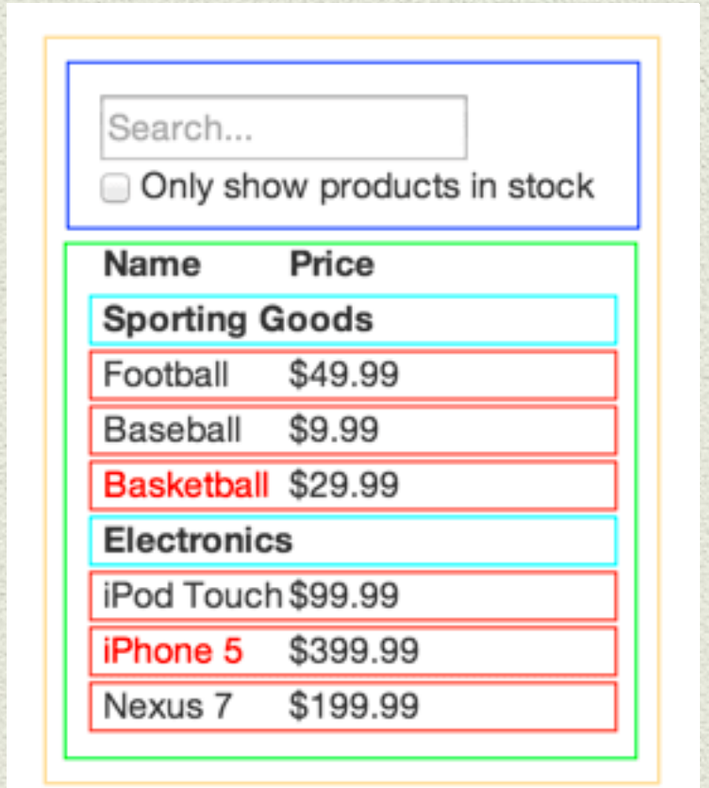
- 使用 state 這個保留字 (note: state is private to this class)
  - ```
    class StatefulButton extends React.Component {
      constructor(props) {
        super(props);
        this.state = { count: 0 };  // JS object
      }
      render() {
        if (…) this.setState({count:this.state.count+1});
        // what if we do “++”?
        else this.state.count–;
        return ( {this.state.count} );
      }
    }
    ```
- 用 constructor 來初始化 state 的值
- Note: Use “setState()” to change the state value. Otherwise, the component will not be re-rendered.
    ```

React Component Lifecycle

- **Mounting** — These methods are called when an instance of a component is being created and inserted into the DOM:
  - constructor()
  - componentWillMount()
  - render()
  - componentDidMount()

- **Updating** — An update can be caused by changes to props or state. These methods are called when a component is being re-rendered:
  - componentWillReceiveProps()
  - shouldComponentUpdate()
  - componentWillUpdate()
  - render()
  - componentDidUpdate()

- **Unmounting** — This method is called when a component is being removed from the DOM:
  - componentWillUnmount()

Ref: https://reactjs.org/docs/state-and-lifecycle.html

React Lifecycle Example (Bad)

- Try this on CodePen —

```
function tick() {
  const element = (
    <div>
      <h1>Hello, world!</h1>
      <h2>It is {new Date().toLocaleTimeString()}.</h2>
    </div>
  );
  ReactDOM.render(
    element,
    document.getElementById('root')
  );
}
setInterval(tick, 1000);
```

- 這是不好的做法，一般來說，一個 app 只會呼叫 ReactDOM.render() 一次，如果要隨時間改變，應該要用 state 以及 component 的 lifecycle 來做到 virtual DOM 的改變

React Lifecycle Example (Good)

- Use React component lifecycle —

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
  }
  componentDidMount() {
    this.timerID = setInterval(() => this.tick(), 1000);
  }
  componentWillUnmount() { clearInterval(this.timerID); }
  tick() { this.setState({date: new Date()});}
  render() {
    return (
      <div>
        <h1>Hello, world!</h1>
        <h2>It is {this.state.date.toLocaleTimeString()}.</h2>
      </div>
);}}
ReactDOM.render(<Clock />, document.getElementById('root'));
```

React Event Handling

- Similar to JavaScript, except that React events are named using camelCase (rather than lower case), and you pass a function object as the event handler (rather than s string):
  - `<button onClick={this.decNum}>{'-'}</button>`
  - `<button mouseOver={doSomething}>touch me</button>`
- Beside, if the handler is a function in the same class, you need to bind "this" to this event handler (下兩列種做法皆可)
  - ```
    class Counter extends React.Component {
      constructor(props) {
        …
        this.incNum = this.incNum.bind(this);
      }
      incNum() { … }
      render() { … }
    }
    ```
  - `<button onClick={(e)=>this.decNum(e)}>{'-'}</button>`
    ```

React Practice #2

- Write an HTML/Javascript file for a counter like this:
  - When the '+' number is pressed, the number should be incremented.
  - When the '-' number is pressed, the number should be decremented.

- Write a React app to perform the same task.
  - Pay attention to where you place your state.
  - Also pay attend to asynchronous update issues on state and props. If that happens, use —
    `this.setState((prevState, props) => ({…}));`
  - What's the benefit to write the code in React?

Thinking in React Step 1: Break The UI Into A Component Hierarchy



ProductItem <tr> … </tr>

ProductRow <tr> … </tr>

ProductTable

```
const rows = […];  // array of elements (ProductRow, ProductItem)
<table>
   <thead> … Name… Price… </thead>
   <tbody>{ rows }</tbody>
</table>
```

SearchBar <form>… </form>

TableTop

```
<div> <SearchBar />
        <ProductTable products={this.props.products} /></div>
```

```
const productList = […];   // product information

ReactDOM.render(
   <TableTop products={productList} />,
   document.getElementById('container')
);
```

Thinking in React

- Step 1: Break The UI Into A Component Hierarchy
- Step 2: Build A Static Version in React
- Step 3: Identify The Minimal (but complete) Representation Of UI State
- Step 4: Identify Where Your State Should Live
- Step 5: Add Inverse Data Flow

After Class・Week #3

- **Homework assignment**
  1. 將 HW#2 的 “TODO List” 網頁設計改寫成 React 版本
  2. (Optional：自我練習) 將 HW#1 自己 and/or 別人的網頁設計重新用 React 實現
- **Deadline**: 5pm, 04/11(Wed).
- 以上網頁，請用 React.js 完成，但可以使用別人的 CSS style file 或是框架

After Class ◦ Week #3

- **Readings /Other actions**
  - **把之前的 React tutorials 確實看完**
  - **行有餘力，可以看一下 node.js, npm, express (ref provided later)**