Web Programming Spring 2021 #22

Chi-Jen Wu

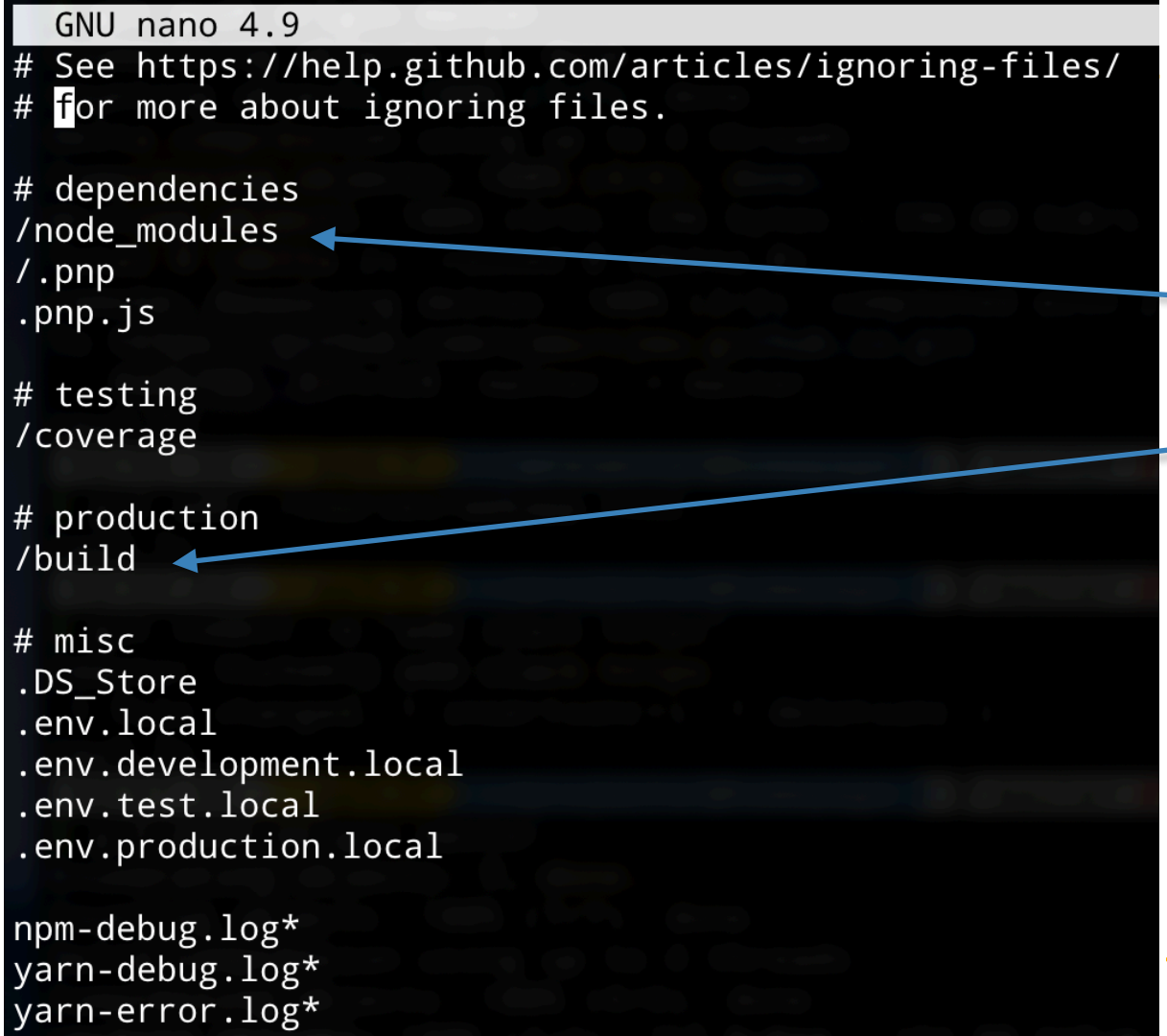
Topics

- The concepts of Web Services
- Web data protocols
  - HTTP, WebSocket, WebRTC
  - HTML, CSS
- Web JavaScript programming
- Cookies and sessions
- **Web Frontend frameworks**
- Web Backend frameworks
- RESTful API design

一些要注意的小細節

About .gitignore

```
$ ls -al
total 3760
drwxr-xr-x    13 cjw  staff      416  4 29 18:11 .
drwxr-xr-x     4 cjw  staff      128  4 29 18:11 ..
-rw-r--r--@    1 cjw  staff     6148  5  3 14:24 .DS_Store
drwxr-xr-x    13 cjw  staff      416  5  3 20:20 .git
-rw-r--r--     1 cjw  staff      310  4 13 22:36 .gitignore
-rw-r--r--     1 cjw  staff     3362  4 13 22:36 README.md
drwxr-xr-x    10 cjw  staff      320  4 29 18:13 build
drwxr-xr-x  1079 cjw  staff    34528  4 27 21:37 node_modules
-rw-r--r--     1 cjw  staff  1408502  4 27 21:37 package-lock.json
-rw-r--r--     1 cjw  staff      981  4 29 18:13 package.json
drwxr-xr-x     8 cjw  staff      256  4 13 22:36 public
drwxr-xr-x    12 cjw  staff      384  4 20 21:04 src
-rw-r--r--     1 cjw  staff   493144  4 27 21:37 yarn.lock
```


GNU nano 4.9
See https://help.github.com/articles/ignoring-files/
for more about ignoring files.
dependencies
/node_modules
/.pnp
.pnp.js
testing
/coverage
production
/build
misc
.DS_Store
.env.local
.env.development.local
.env.test.local
.env.production.local
npm-debug.log*
yarn-debug.log*
yarn-error.log*

Here

About .gitignore

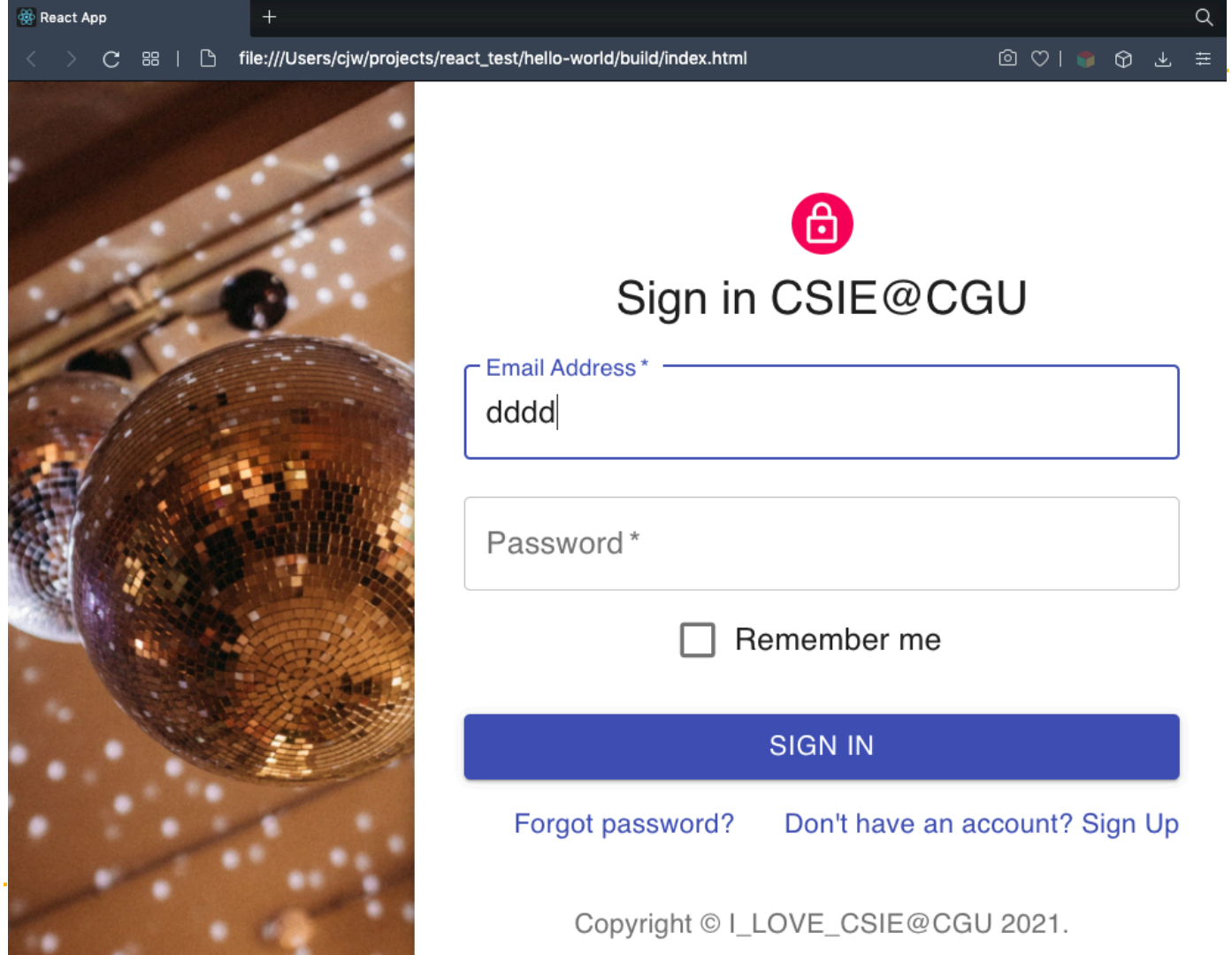
```
$ ls -al
total 3760
drwxr-xr-x    13 cjw  staff      416  4 29 18:11 .
drwxr-xr-x     4 cjw  staff      128  4 29 18:11 ..
-rw-r--r--@    1 cjw  staff     6148  5  3 14:24 .DS_Store
drwxr-xr-x    13 cjw  staff      416  5  3 20:20 .git
-rw-r--r--     1 cjw  staff      310  4 13 22:36 .gitignore
-rw-r--r--     1 cjw  staff     3362  4 13 22:36 README.md
drwxr-xr-x    10 cjw  staff      320  4 29 18:13 build
drwxr-xr-x  1079 cjw  staff    34528  4 27 21:37 node_modules
-rw-r--r--     1 cjw  staff  1408502  4 27 21:37 package-lock.json
-rw-r--r--     1 cjw  staff      981  4 29 18:13 package.json
drwxr-xr-x     8 cjw  staff      256  4 13 22:36 public
drwxr-xr-x    12 cjw  staff      384  4 20 21:04 src
-rw-r--r--     1 cjw  staff   493144  4 27 21:37 yarn.lock
```

Here

About node.js build

```
{
  "name": "hello-world",
  "version": "0.1.0",
  "private": true,
  "homepage": ".",
  "dependencies": {
    "@material-ui/core": "^4.11.3",
    "@material-ui/data-grid": "*",
    "@material-ui/icons": "^4.11.2",
    "@material-ui/pickers": "^3.3.10",
    "@testing-library/jest-dom": "^5.11.4",
    "@testing-library/react": "^11.1.0",
    "@testing-library/user-event": "^12.1.10",
    "react": "^17.0.2",
    "react-dom": "^17.0.2",
    "react-scripts": "4.0.3",
    "web-vitals": "^1.0.1"
  },
```

Add it


React App
file:///Users/cjw/projects/react_test/hello-world/build/index.html
Sign in CSIE@CGU
Email Address *
dddd
Password *
Remember me
SIGN IN
Forgot password?
Don't have an account? Sign Up
Copyright © I_LOVE_CSIE@CGU 2021.

React / ReactJS

- React Ecosystem
- 開發環境
- **核心概念 — 元件(Components)**
- 單向資料流 (Unidirectional data flow)
- Virtual DOM
- Declarative UI / JSX
- React Router

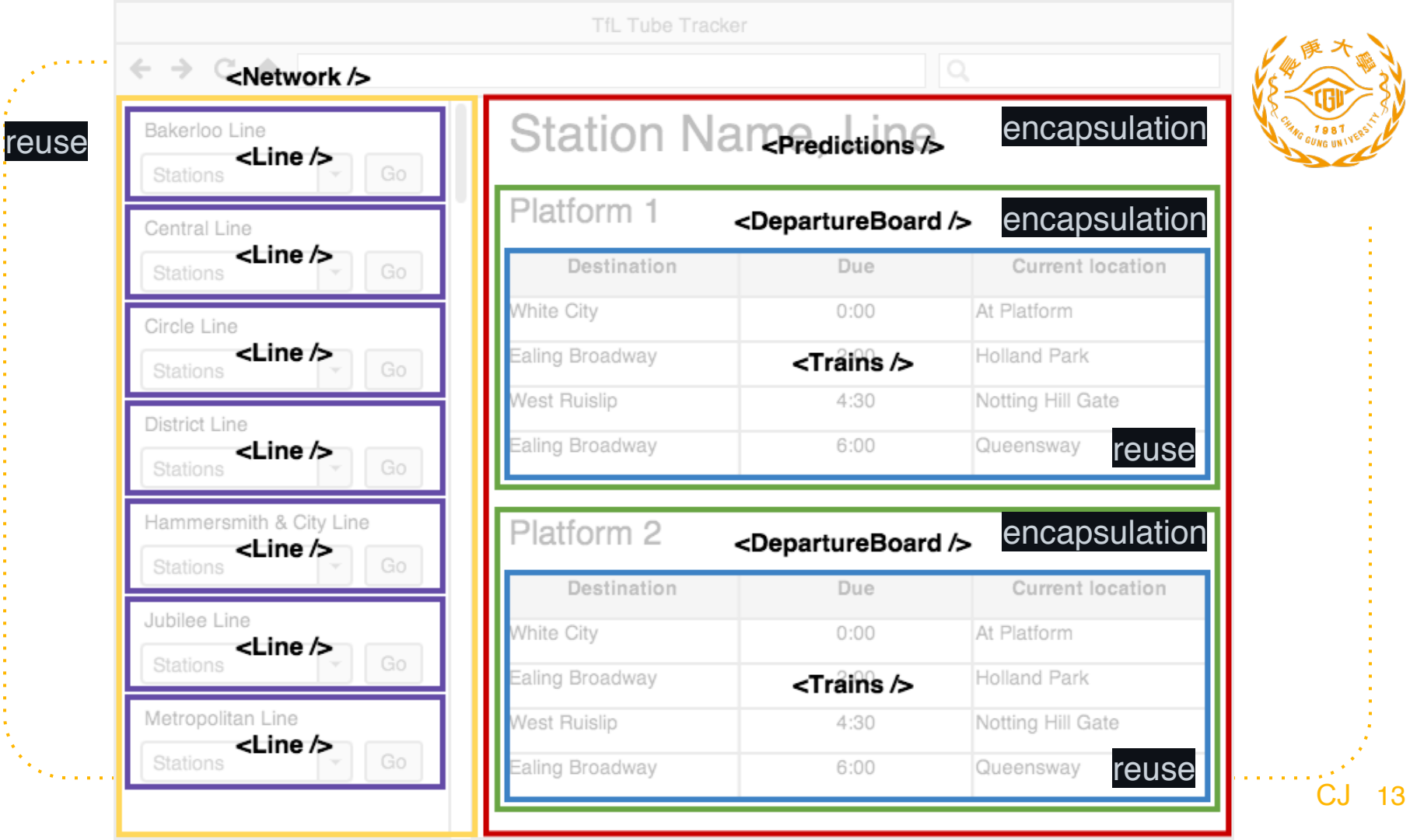
真 程式碼模組化

- 例如我們要讓每一個按鈕計算自己被按了幾次
  - 1. 元件自己計算 ！！
- 模組化後 每個元件應該要可以保持自己的**狀態**
  - **自己管自己的狀態！！**

我們來真的模組化吧！！

React Components

- **核心概念 — 元件(Components)**
- Virtual DOM
- Component 就像個狀態機
- 有生命週期（Life Cycle）
- 元件最重要就是重用(reuse) 和封裝組合 (encapsulation and compose)


TfL Tube Tracker
<Network />
Bakerloo Line
<Line />
Stations
Go
Central Line
<Line />
Stations
Go
Circle Line
<Line />
Stations
Go
District Line
<Line />
Stations
Go
Hammersmith & City Line
<Line />
Stations
Go
Jubilee Line
<Line />
Stations
Go
Metropolitan Line
<Line />
Stations
Go
Station Name, Line
<Predictions />
encapsulation
Platform 1
<DepartureBoard />
encapsulation
Destination
Due
Current location
White City
0:00
At Platform
Ealing Broadway
<Trains />
Holland Park
West Ruislip
4:30
Notting Hill Gate
Ealing Broadway
6:00
Queensway
reuse
Platform 2
<DepartureBoard />
encapsulation
Destination
Due
Current location
White City
0:00
At Platform
Ealing Broadway
<Trains />
Holland Park
West Ruislip
4:30
Notting Hill Gate
Ealing Broadway
6:00
Queensway
reuse
reuse

React Components 寫法 #1

```
<script>
const MyComponent = () => (
   <div>Hello, World!</div>
);

ReactDOM.render(
        <MyComponent/>,
        document.getElementById('root'));
</script>
```

我們目前一直用的寫法
沒有狀態 只處理UI和event

React Components 寫法 #2

```
<script>
class MyComponent extends React.Component {
    render() {
        return (
            <div>Hello, World!</div>
        );
    }
}

ReactDOM.render(<MyComponent/>, document.getElementById('root'));
</script>
```

extends 寫法

完整的 component 狀態 UI和event

HTML

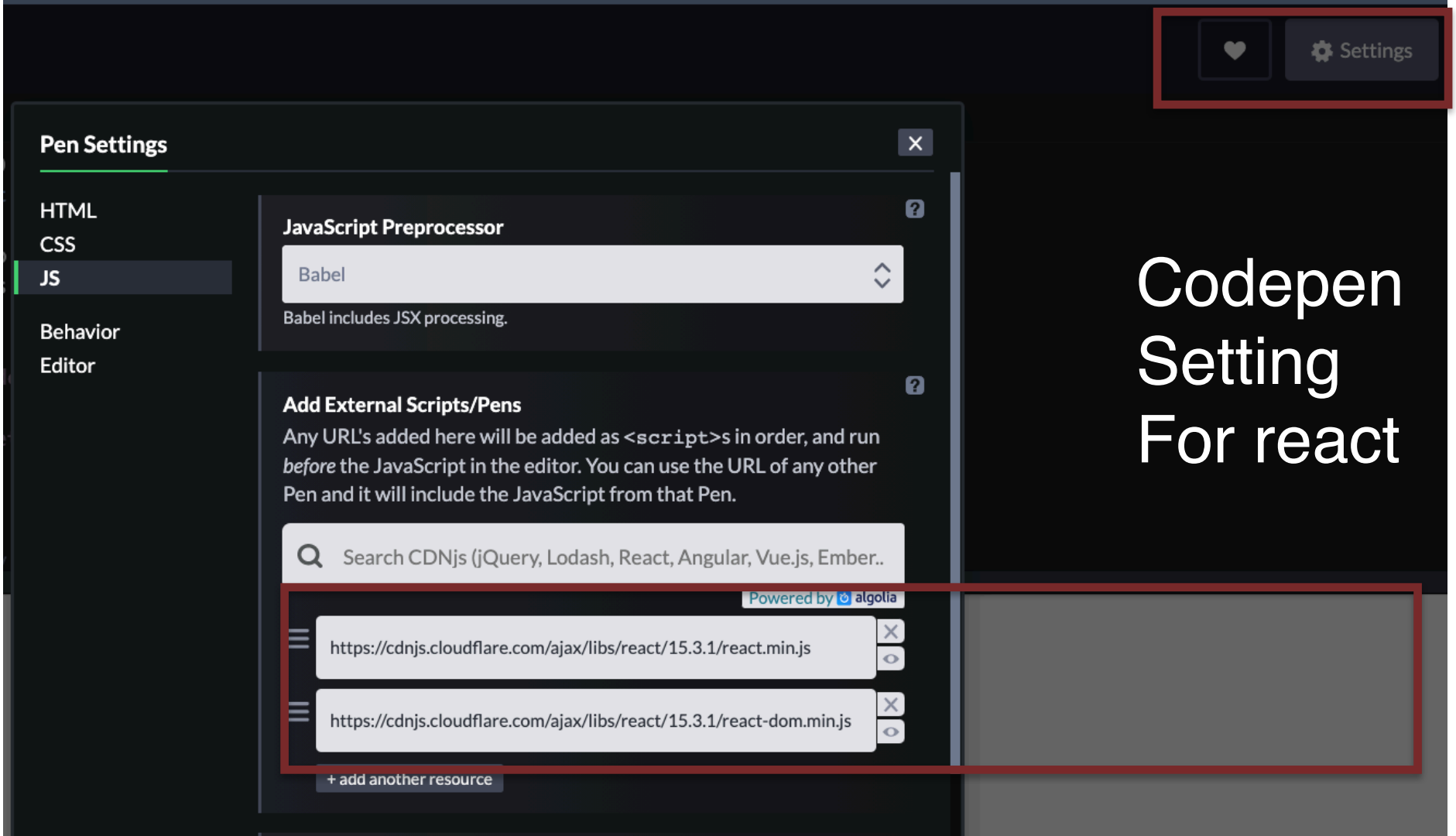
```
<div id="root"></div>
```

CSS

JS (Babel)

```
class MyComponent extends React.Component {
  render() {
    return (
      <div>Hello, World!</div>
    );
  }
}

ReactDOM.render(<MyComponent/>, document.getElementById('root'));
```

Hello, World!

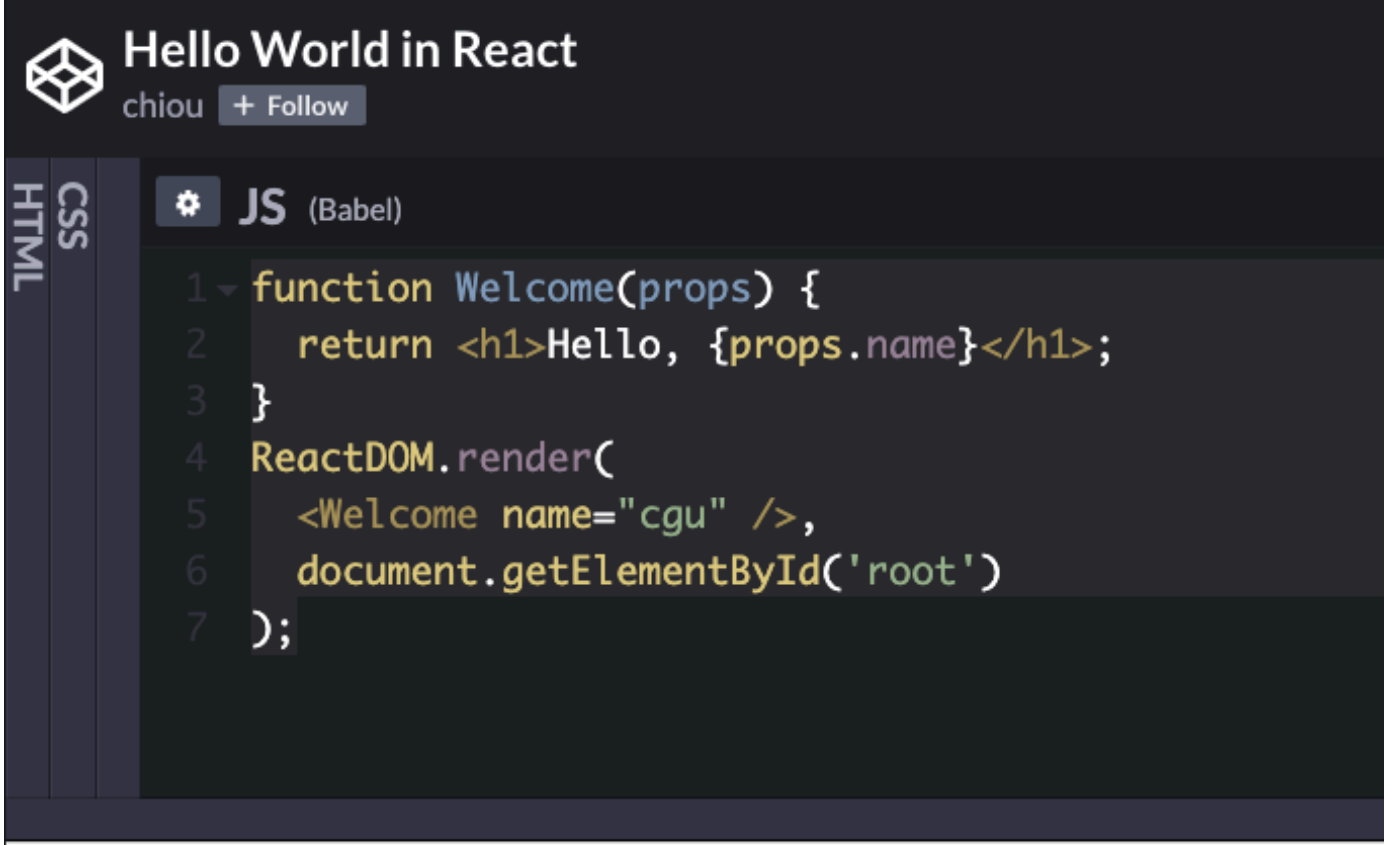

Settings
Pen Settings
HTML
CSS
JS
Behavior
Editor
JavaScript Preprocessor
Babel
Babel includes JSX processing.
Add External Scripts/Pens
Any URL's added here will be added as <script>s in order, and run before the JavaScript in the editor. You can use the URL of any other Pen and it will include the JavaScript from that Pen.
Search CDNjs (jQuery, Lodash, React, Angular, Vue.js, Ember..
Powered by algolia
https://cdnjs.cloudflare.com/ajax/libs/react/15.3.1/react.min.js
https://cdnjs.cloudflare.com/ajax/libs/react/15.3.1/react-dom.min.js
+ add another resource
Codepen
Setting
For react

React Components 狀態

- **props**
  - **父元件傳入**
  - **唯讀**
- **State**
  - **component自己的**
  - **私有的**
  - **要改變state，必須使用setState()**

React Components props狀態

```
<script>
function Welcome(props) {
  return <h1>Hello, {props.name}</h1>;
}
ReactDOM.render(
  <Welcome name="cgu" />,
  document.getElementById('root')
);
</script>
```


Hello World in React
chiou
+ Follow
HTML
CSS
JS (Babel)
function Welcome(props) {
return <h1>Hello, {props.name}</h1>;
}
ReactDOM.render(
<Welcome name="cgu" />,
document.getElementById('root')
);

Hello, cgu

時鐘例子：以前的寫法

```
<script>
function tick() {
  const element = (
    <div>
      <h1>長庚標準時間</h1>
      <h2>目前為：{new Date().toLocaleTimeString()}.</h2>
    </div>
  );
  ReactDOM.render(element, document.getElementById('root')
  );
}

setInterval(tick, 1000);
</script>
```

Hello World in React

chiou + Follow

HTML

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

CSS

JS (Babel)

```
function tick() {
  const element = (
    <div>
      <h1>長庚標準時間＃1</h1>
      <h2>目前為：{new Date().toLocaleTimeString()}.</h2>
    </div>
  );
  ReactDOM.render(element, document.getElementById('root')
  );
}

setInterval(tick, 1000);
```

長庚標準時間＃1

目前為：下午9:25:11.

時鐘例子：props寫法

```
<script>
class Clock extends React.Component {
  render() {
    return (
      <div>
        <h1>長庚標準時間 # props</h1>
        <h2>目前為： {this.props.date.toLocaleTimeString()}.</h2>
      </div>
    );
  };
}

function tick() {
  ReactDOM.render(<Clock date={new Date()} />, document.getElementById('root'));
}
setInterval(tick, 1000);
</script>
```

HTML

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

CSS

JS (Babel)

```
    return (
      <div>
        <h1>長庚標準時間 # props</h1>
        <h2>目前為： {this.props.date.toLocaleTimeString()}.</h2>
      </div>
    );
  };
}

function tick() {
  ReactDOM.render(<Clock date={new Date()} />, document.getElementById('root'));
}
setInterval(tick, 1000);
```

長庚標準時間 # props

目前為： 下午9:38:40.

時鐘例子：state寫法

```
<script>
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
    this.tick = this.tick.bind(this);
  }
  componentDidMount() {
    this.timerID = setInterval(this.tick, 1000);
  }
  componentWillUnmount() {
    clearInterval(this.timerID);
  }
  render() {
    return (
      <div>
        <h1>長庚標準時間＃state</h1>
        <h2>目前為： {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
  tick() {
    this.setState({
      date: new Date()
    });
  }
}
ReactDOM.render( <Clock />, document.getElementById('root'));
</script>
```

有點複雜

一個一個看

HTML

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

CSS

JS (Babel)

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
    this.tick = this.tick.bind(this);
  }
  componentDidMount() {
    this.timerID = setInterval(this.tick, 1000);
  }
  componentWillUnmount() {
    clearInterval(this.timerID);
  }
  render() {
    return (
      <div>
        <h1>長庚標準時間 # state</h1>
        <h2>目前為： {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
  tick() {
    this.setState({
      date: new Date()
    });
  }
}
ReactDOM.render( <Clock />, document.getElementById('root'));
```

長庚標準時間 # state

目前為： 下午9:57:12.

時鐘例子：state寫法

```
<script>
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
    this.tick = this.tick.bind(this);
  }
  componentDidMount() {
    this.timerID = setInterval(this.tick, 1000);
  }
  componentWillUnmount() {
    clearInterval(this.timerID);
  }
  render() {
    return (
      <div>
        <h1>長庚標準時間＃state</h1>
        <h2>目前為： {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
  tick() {
    this.setState({
      date: new Date()
    });
  }
}
ReactDOM.render( <Clock />, document.getElementById('root'));
</script>
```

這裡一樣

```
<script>
class Clock extends React.Component {
  constructor(props) { // 初始 這個 component 時要跑的
  }
  componentDidMount() { // 掛載 這個 component 時要跑的
  }
  componentWillUnmount() { // 移除 這個 component 時要跑的
  }
  render() { // render component (一定要有)
  }
  tick() { // component private function (計算一秒一秒)
  }
}
</script>
```

React Components 生命週期方法

- componentDidMount()
  - 在 component 被 render 到 DOM 之後才會執行
  - 只會被呼叫一次
- componentWillUnmount()
  - 從實體DOM階段移除「之前」的時候觸發
  - 只會被呼叫一次

```
<script>
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
    this.tick = this.tick.bind(this);
  }
  componentDidMount() {
    this.timerID = setInterval(this.tick, 1000);
  }
  componentWillUnmount() {
    clearInterval(this.timerID);
  }
  render() {
    return (
      <div>
        <h1>長庚標準時間 # state</h1>
        <h2>目前為： {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
  tick() {
    this.setState({
      date: new Date()
    });
  }
}
ReactDOM.render( <Clock />, document.getElementById('root'));
</script>
```

建構者

state: 和render有關

```
<script>
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
    this.tick = this.tick.bind(this);
  }
  componentDidMount() {
    this.timerID = setInterval(this.tick, 1000);
  }
  componentWillUnmount() {
    clearInterval(this.timerID);
  }
  render() {
    return (
      <div>
        <h1>長庚標準時間 # state</h1>
        <h2>目前為： {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
  tick() {
    this.setState({
      date: new Date()
    });
  }
}
ReactDOM.render( <Clock />, document.getElementById('root'));
</script>
```

Private func
要bind給這個元件

```
<script>
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
    this.tick = this.tick.bind(this);
  }
  componentDidMount() {
    this.timerID = setInterval(this.tick, 1000);
  }
  componentWillUnmount() {
    clearInterval(this.timerID);
  }
  render() {
    return (
      <div>
        <h1>長庚標準時間#state</h1>
        <h2>目前為： {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
  tick() {
    this.setState({
      date: new Date()
    });
  }
}
ReactDOM.render( <Clock />, document.getElementById('root'));
</script>
```

元件初始/結束

```
<script>
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
    this.tick = this.tick.bind(this);
  }
  componentDidMount() {
    this.timerID = setInterval(this.tick, 1000);
  }
  componentWillUnmount() {
    clearInterval(this.timerID);
  }
  render() {
    return (
      <div>
        <h1>長庚標準時間＃state</h1>
        <h2>目前為： {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
  tick() {
    this.setState({
      date: new Date()
    });
  }
}
ReactDOM.render( <Clock />, document.getElementById('root'));
</script>
```

元件真正呈現

```
<script>
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
    this.tick = this.tick.bind(this);
  }
  componentDidMount() {
    this.timerID = setInterval(this.tick, 1000);
  }
  componentWillUnmount() {
    clearInterval(this.timerID);
  }
  render() {
    return (
      <div>
        <h1>長庚標準時間 # state</h1>
        <h2>目前為： {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
  tick() {
    this.setState({
      date: new Date()
    });
  }
}
ReactDOM.render( <Clock />, document.getElementById('root'));
</script>
```

Tick func
設定新的時間

HTML

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

JS (Babel) 13 unsaved changes

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
    this.tick = this.tick.bind(this);
  }
  componentDidMount() {
    this.timerID = setInterval(this.tick, 1000);
  }
  componentWillUnmount() {
    clearInterval(this.timerID);
  }
  render() {
    return (
      <div>
        <h1>長庚標準時間 # state</h1>
        <h2>目前為： {this.state.date.toLocaleTimeString()}.</h2>
      </div>
    );
  }
  tick() {
    this.setState({
      date: new Date()
    });
  }
}
ReactDOM.render( <Clock />, document.getElementById('root'));
```

長庚標準時間 # state

目前為： 上午10:49:38.

正確的使用 State

- 不要直接修改 State

```
// 錯誤
this.setState({
  counter: this.state.counter + this.props.increment,
});
```

```
// 正確
this.setState({comment: 'Hello'});
```

- State 的更新可能是非同步的
  - this.props 和 this.state 是非同步的被更新

```
// 錯誤
this.setState({
  counter: this.state.counter + this.props.increment,
});
```

```
// 正確
this.setState(function(state, props) {
  return {
    counter: state.counter + props.increment
  };
});
```

線上練習 VS code

https://codepen.io/pen/

動態生成兩個時區的時鐘

Hello World in React

Chi-Jen Wu

HTML

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

CSS

JS (Babel)

```
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
    this.tick = this.tick.bind(this);
  }
```

長庚標準時間 # state

Asia/Taipei 目前為： 上午11:45:34.

Europe/London 目前為： 上午4:45:34.

America/New_York 目前為： 下午11:45:33.

喜歡不喜歡IU example

- 要改變state，
  - 使用setState()
- 這裡是指內部資料
- 要render出來就要setState

Hello World in React

chiou + Follow

HTML CSS JS (Babel)

```
class LikeButton extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
      liked: false
    }
    this.handleClick = this.handleClick.bind(this)
  }
  handleClick(event) {
    this.setState({ liked: !this.state.liked });
    console.log(this.state);
  }
  render() {
    var text = this.state.liked ? '喜歡' : '不喜歡';
    return (
      <p onClick={this.handleClick}>
        你 {text} IU.
      </p>
    );
  }
}
ReactDOM.render(<LikeButton />, document.getElementById('root'));
```

你 喜歡 IU.

喜歡不喜歡IU example


```
<script>
class LikeButton extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
      liked: false
    }
    this.handleClick = this.handleClick.bind(this)
  }
  handleClick(event) {
    this.setState({ liked: !this.state.liked });
    console.log(this.state);
  }
  render() {
    var text = this.state.liked ? '喜歡' : '不喜歡';
    return (
      <p onClick={this.handleClick}>
        你 {text} IU.
      </p>
    );
  }
}
ReactDOM.render(<LikeButton />, document.getElementById('root'));
</script>
```

開關 example

- 很常見的UI
- 打開和關掉

Toggle
Dan Abramov PRO + Follow

HTML CSS JS (Babel)

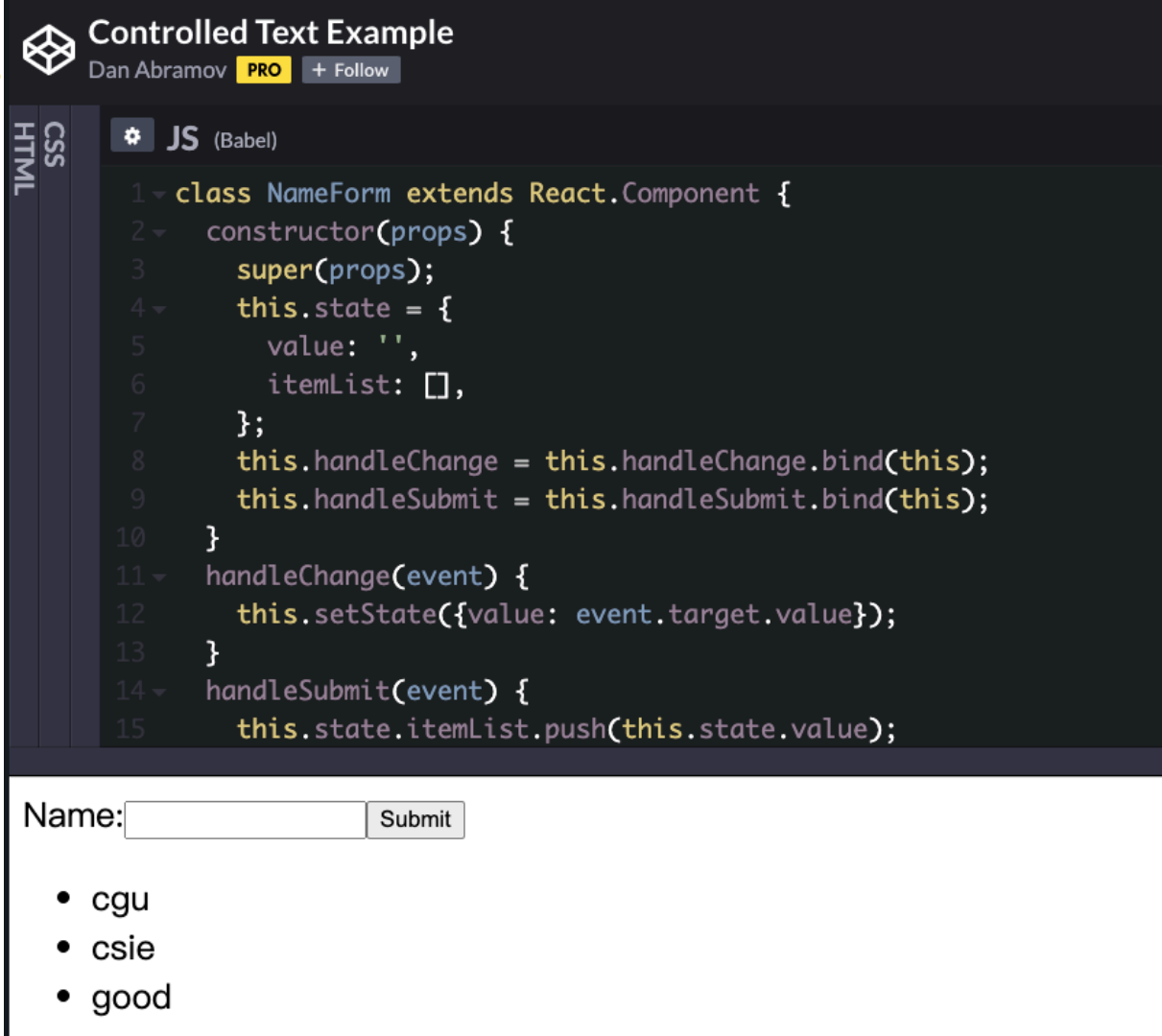
```
class Toggle extends React.Component {
  constructor(props) {
    super(props);
    this.state = {isToggleOn: true};
    this.handleClick = this.handleClick.bind(this);
  }

  handleClick() {
    this.setState(prevState => ({
      isToggleOn: !prevState.isToggleOn
    }));
  }

  render() {
    return (
      <button onClick={this.handleClick}>
        {this.state.isToggleOn ? '我被打開了' : '我被關掉了'}
      </button>
    );
  }
}
```

我被打開了

開關 example

```
<script>
class Toggle extends React.Component {
  constructor(props) {
    super(props);
    this.state = {isToggleOn: true};
    this.handleClick = this.handleClick.bind(this);
  }
  handleClick() {
    this.setState(prevState => ({
      isToggleOn: !prevState.isToggleOn
    }));
  }
  render() {
    return (
      <button onClick={this.handleClick}>
        {this.state.isToggleOn ? '我被打開了' : '我被關掉了'}
      </button>
    );
  }
}
ReactDOM.render(<Toggle />, document.getElementById('root')
);
</script>
```

新增DOM元件example



```
<script>
class NameForm extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      value: '',
      itemList: [],
    };
    this.handleChange = this.handleChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
  }
  handleChange(event) {
    this.setState({value: event.target.value});
  }
  handleSubmit(event) {
    this.state.itemList.push(this.state.value);
    this.setState({value: '', itemList: this.state.itemList,});
    event.preventDefault();
  }
  render() {
    return (
      <div>
      <form onSubmit={this.handleSubmit}>
        <label>
          Name:
          <input type="text" value={this.state.value} onChange={this.handleChange} />
        </label>
        <input type="submit" value="Submit" />
      </form>
      <ul style={{ textAlign: 'left' }}>
        {this.state.itemList.map((item, index) => <li key={`item_${index}`}>{item}</li>) }
      </ul>
      </div>
    );
  }
}

ReactDOM.render(
  <NameForm />,
  document.getElementById('root')
```

```
<script>
class NameForm extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      value: '',
      itemList: [],
    };
    this.handleChange = this.handleChange.bind(this);
    this.handleSubmit = this.handleSubmit.bind(this);
  }
  handleChange(event) {
    this.setState({value: event.target.value});
  }
  handleSubmit(event) {
    this.state.itemList.push(this.state.value);
    this.setState({value: '', itemList:
this.state.itemList,});
    event.preventDefault();
  }
```

```
  render() {
    return (
      <div>
      <form onSubmit={this.handleSubmit}>
        <label>
          Name:
          <input type="text" value={this.state.value} onChange={this.handleChange} />
        </label>
        <input type="submit" value="Submit" />
      </form>
      <ul style={{ textAlign: 'left' }}>
        {this.state.itemList.map((item, index) => <li key={`item_${index}`}>{item}</li>) }
      </ul>
      </div>
    );
  }
}

ReactDOM.render(
  <NameForm />,
  document.getElementById('root')
);
```

新增DOM元件 example


```
<script>

ReactDOM.render(
  <NameForm />,
  document.getElementById('root')
);
</script>
```

線上練習 quiz#5

https://codepen.io/pen/

把之前的button改成可以計算點自己被點了幾次

1. Commit **quiz5.html, quiz5.js** to quiz and push to quiz repository

```
<script>
const changeText=(event)=>{
  console.log(event.target)
  event.target.innerText = event.target.innerText + "被點了"
}
const multiButton=(num)=>{
  var output=[];
  for(let i=1;i<num+1;++i)
      output.push(<button onClick={changeText}>第{i}個按鍵</button>)
  return output;
}
function App() {
  return (
    <div className="App">
      { multiButton(10) }
    </div>
  );
}
</script>
```

Hello World in React

chiou + Follow

HTML CSS JS (Babel)

```
class CounterButton extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
      counter: 0
    }
    this.handleClick = this.handleClick.bind(this)
  }
  handleClick(event) {
    this.setState({ counter: this.state.counter+1 });
    console.log(this.state);
  }
  render() {
    var text = this.state.counter;
    return (
      <p onClick={this.handleClick}>
        你點了 {text} 次.
      </p>
    );
  }
}
```

你點了 3 次.

```
<script>
class Clock extends React.Component {
  constructor(props) {
    super(props);
    this.state = {date: new Date()};
    this.tick = this.tick.bind(this);
  }
  componentDidMount() {
    this.timerID = setInterval(this.tick, 1000);
  }
  componentWillUnmount() {
    clearInterval(this.timerID);
  }
  render() {
    return (
      <div>
        <h2>{this.props.timeZone} 目前為： {this.state.date.toLocaleTimeString('zh-TW', {timeZone:
this.props.timeZone})}.</h2>
      </div>
    );
  }
  tick() {
    this.setState({
      date: new Date()
    });
  }
}
ReactDOM.render(<div> <h1>長庚標準時間＃state</h1>
    <Clock timeZone='Asia/Taipei'/>
    <Clock timeZone='Europe/London'/>
    <Clock timeZone='America/New_York'/></div>,
    document.getElementById('root'));
</script>
```

Thanks!

Open for any questions

CJ Wu

cjwu@mail.cgu.edu.tw