



Web Programming Spring 2021

#23

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





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

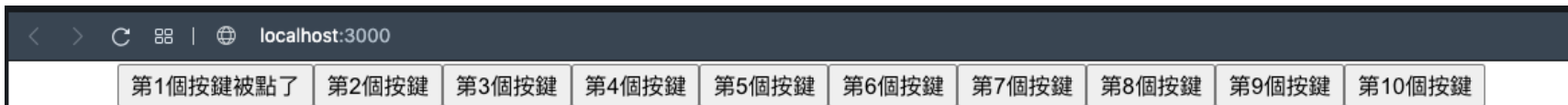
今天在公園椅子上無聊開筆電找
程式bug

看到一位賣口香糖的阿婆，想說
買一個一邊吃一邊找
後來阿婆看到我苦惱的樣子走了
過來

看了螢幕
指著螢幕：「這裡少了分號」
我：.....

真 程式碼模組化

- 例如我們要让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 ！！**

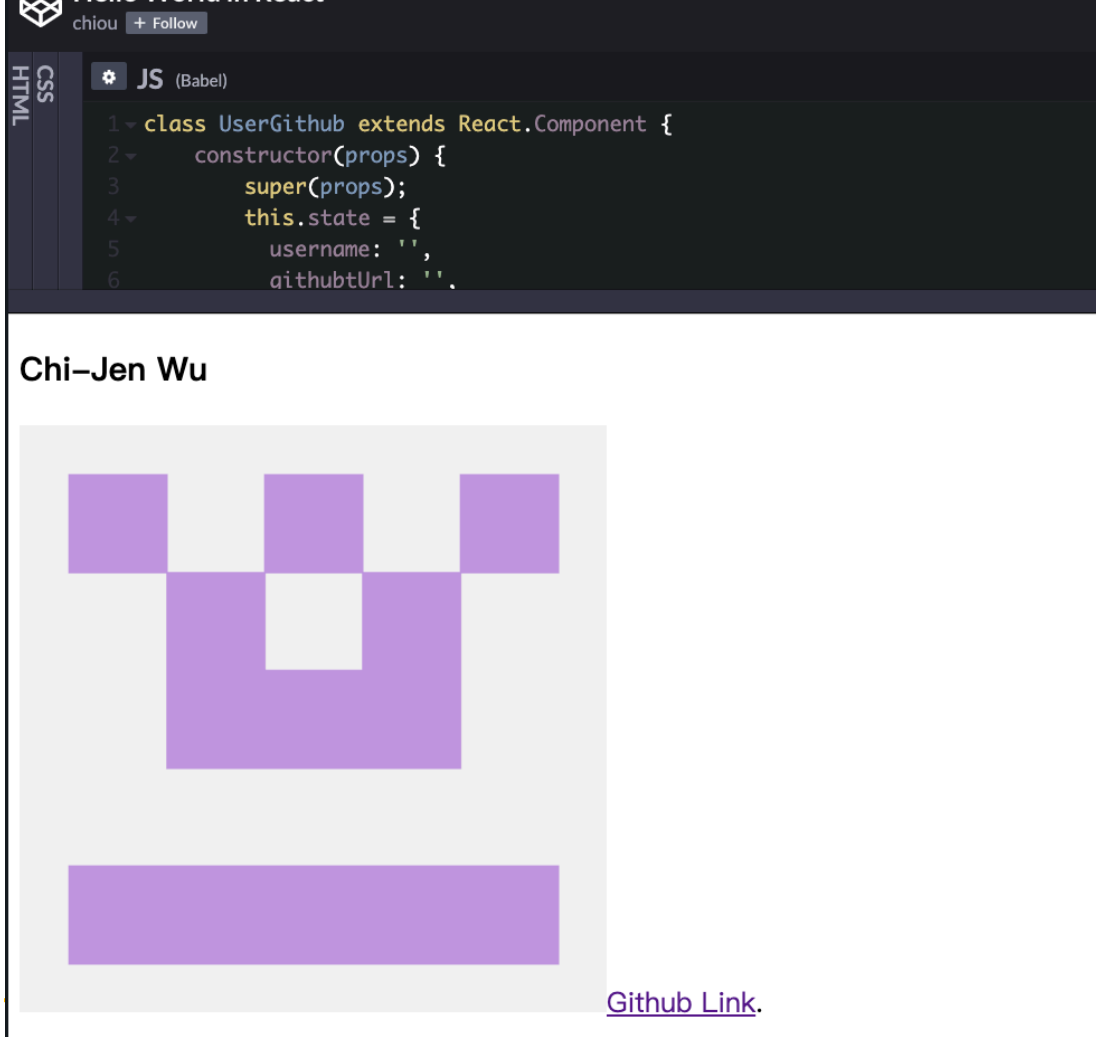




React Components + jQuery

- **State** 有些是從網路上拿到的資料
 - 應該大部分都會是從**API**拿到資料
- 所以我們要了解
 - **Component** 怎麼拿到**API**的資料
 - 先從 **jQuery + Ajax**來

React Components + jQuery + Ajax





```
<html>
<script src="https://fb.me/react-15.1.0.js"></script>
<script src="https://fb.me/react-dom-15.1.0.js"></script>
<script src="https://code.jquery.com/jquery-3.1.0.js"></script>
<body>
  <div id="root"></div>
</body>
</html>
```




```
<script>
class UserGithub extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      username: '',
      githubtUrl: '',
      avatarUrl: ''
    }
  }
  componentDidMount() {
    $.get(this.props.source, (result) => {
      console.log(result);
      const data = result;
      if (data) {
        this.setState({
          username: data.name,
          githubtUrl: data.html_url,
          avatarUrl: data.avatar_url
        });
      }
    });
  }
  render() {
    return (
      <div>
        <h3>{this.state.username}</h3>
        <img src={this.state.avatarUrl} />
        <a href={this.state.githubtUrl}>Github Link</a>
      </div>
    );
  }
}

ReactDOM.render(
  <UserGithub source="https://api.github.com/users/cjwu" />,
  document.getElementById('root')
);
</script>
```



建構者

```
<script>
class UserGithub extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      username: '',
      githubUrl: '',
      avatarUrl: '',
    }
  }

  componentDidMount() {
    $.get(this.props.source, (result) => {
      console.log(result);
      const data = result;
      if (data) {
        this.setState({
          username: data.name,
          githubUrl: data.html_url,
          avatarUrl: data.avatar_url
        });
      }
    });
  }

  render() {
    return (
      <div>
        <h3>{this.state.username}</h3>
        <img src={this.state.avatarUrl} />
        <a href={this.state.githubUrl}>Github Link</a>.
      </div>
    );
  }
}
</script>
```



```
<script>
```

```
class UserGithub extends React.Component {  
  constructor(props) {  
    super(props);  
    this.state = {  
      username: '',  
      githubUrl: '',  
      avatarUrl: '',  
    }  
  }  
}
```

```
  componentDidMount() {  
    $.get(this.props.source, (result) => {  
      console.log(result);  
      const data = result;  
      if (data) {  
        this.setState({  
          username: data.name,  
          githubUrl: data.html_url,  
          avatarUrl: data.avatar_url  
        });  
      }  
    });  
  }  
}
```

```
  render() {  
    return (  
      <div>  
        <h3>{this.state.username}</h3>  
        <img src={this.state.avatarUrl} />  
        <a href={this.state.githubUrl}>Github Link</a>.  
      </div>  
    );  
  }  
}
```

```
</script>
```

元件初始 Ajax

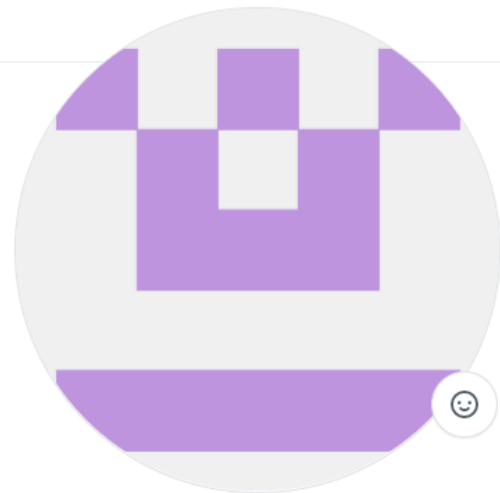


```
<script>
class UserGithub extends React.Component {
  constructor(props) {
    super(props);
    this.state = {
      username: '',
      githubtUrl: '',
      avatarUrl: '',
    }
  }
  componentDidMount() {
    $.get(this.props.source, (result) => {
      console.log(result);
      const data = result;
      if (data) {
        this.setState({
          username: data.name,
          githubtUrl: data.html_url,
          avatarUrl: data.avatar_url
        });
      }
    });
  }
  render() {
    return (
      <div>
        <h3>{this.state.username}</h3>
        <img src={this.state.avatarUrl} />
        <a href={this.state.githubtUrl}>Github Link</a>.
      </div>
    );
  }
}
</script>
```

Render

Homework #5

- 整合一個 user profile 頁面
 - material-ui
 - 或是自己組也是可以
- 要有
 - id, name, 頭像 六個以上資訊
 - <https://api.github.com/users/cjwu>



Chi-Jen Wu

cjwu

Edit profile

🔍 3 followers · 0 following · ☆ 0

📍 Taiwan

✉ cjwu@arbor.ee.ntu.edu.tw

🔗 <https://cjwu.github.io>



```
{
  "login": "cjwu",
  "id": 1336309,
  "node_id": "MDQ6VXNlcjEzMzYzMDk=",
  "avatar_url": "https://avatars.githubusercontent.com/u/1336309?v=4",
  "gravatar_id": "",
  "url": "https://api.github.com/users/cjwu",
  "html_url": "https://github.com/cjwu",
  "followers_url": "https://api.github.com/users/cjwu/followers",
  "following_url": "https://api.github.com/users/cjwu/following{/other_user}",
  "gists_url": "https://api.github.com/users/cjwu/gists{/gist_id}",
  "starred_url": "https://api.github.com/users/cjwu/starred{/owner}/{/repo}",
  "subscriptions_url": "https://api.github.com/users/cjwu/subscriptions",
  "organizations_url": "https://api.github.com/users/cjwu/orgs",
  "repos_url": "https://api.github.com/users/cjwu/repos",
  "events_url": "https://api.github.com/users/cjwu/events{/privacy}",
  "received_events_url": "https://api.github.com/users/cjwu/received_events",
  "type": "User",
  "site_admin": false,
  "name": "Chi-Jen Wu",
  "company": null,
  "blog": "https://cjwu.github.io",
  "location": "Taiwan",
  "email": null,
  "hireable": null,
  "bio": null,
  "twitter_username": null,
  "public_repos": 62,
  "public_gists": 0,
  "followers": 3,
  "following": 0,
  "created_at": "2012-01-17T10:30:23Z",
  "updated_at": "2021-05-04T14:31:16Z"
}
```

Github API 提供32個資訊



React Components 之間溝通

- children Component 沒有辦法改變 parent Component 的 state
- 把 parent 改變 state 的 function 傳下去給 children
 - 當成其中一個 props 的屬性



子 Components

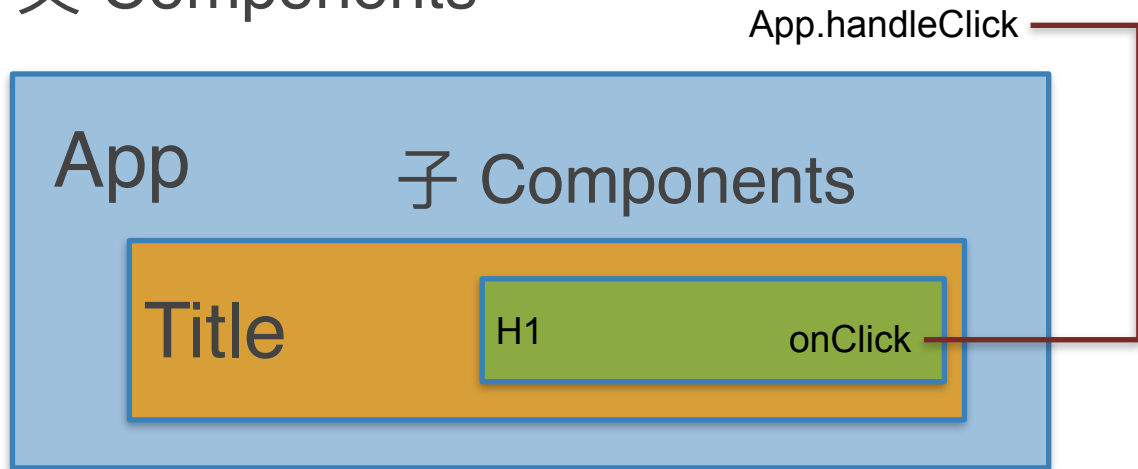
```
<script>
const Title = (props) => {
  return (
    <h1 onClick={props.handleClick}>{props.text}</h1>
  )
}
</script>
```



父 Components



```
<script>
class App extends React.Component {
  constructor() {
    super();
    this.state = {
      counter : 1
    }
    this.handleClick = this.handleClick.bind(this);
  }
  handleClick() {
    this.setState({
      counter: this.state.counter + 1
    })
  }
  render() {
    return (
      <div>
        <Title handleClick={this.handleClick} text={this.state.counter}/>
      </div>
    )
  }
}
</script>
```

父 Components



 JS (Babel)

```
1 const Title = (props) => {  
2   return (  
3     <h1 onClick={props.handleClick}>{props.text}</h1>  
4   )  
5 }
```

4

父 Components



```
<script>
class App extends React.Component {
  constructor() {
    super();
    this.state = {
      counter : 1
    }
    this.handleClick = this.handleClick.bind(this);
  }
  handleClick() {
    this.setState({
      counter: this.state.counter + 1
    })
  }
  render() {
    return (
      <div>
        <Title handleClick={this.handleClick} text={this.state.counter}/>
      </div>
    )
  }
}
</script>
```



線上練習 VS code

<https://codepen.io/pen/>

以父之名

修改父元素的h1

```
1 const Title = (props) => {  
2   return (  
3     <h2 onClick={props.handleClick}>點一下</h2>  
4   )  
5 }
```

爸爸的：11

點一下

以父之名



```
<script>
const Title = (props) => {
  return (
    <h2 onClick={props.handleClick}>點一下</h2>
  )
}

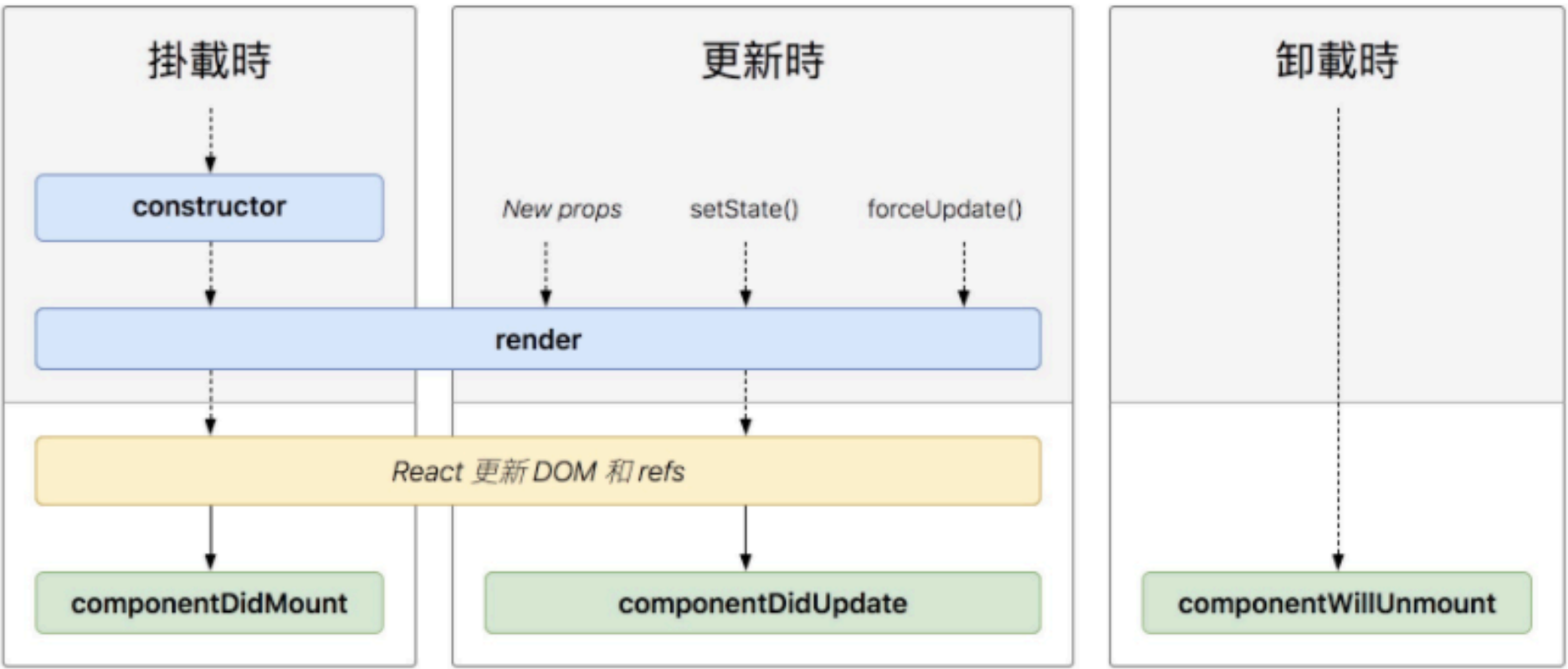
class App extends React.Component {
  constructor() {
    super();
    this.state = {
      counter : 1
    }
    this.handleClick = this.handleClick.bind(this);
  }
  handleClick() {
    this.setState({
      counter: this.state.counter + 1
    })
  }
  render() {
    return (
      <div>
        <h1>爸爸的 : {this.state.counter}</h1>
        <Title handleClick={this.handleClick} text={this.state.counter}/>
      </div>
    )
  }
}

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



Component Lifecycle - 元件的生命周期

- **Mounting:** 準備要掛載實際 dom 節點時
 - **componentDidMount()**
- **Updating:** state 異動要重新 render 時
 - **componentWillUpdate()**
- **Unmounting:** 即將卸載時
 - **componentWillUnmount()**





React 裡面最重要的觀念就是

Component 的 state 會對應到一個 UI，
當 state 一有變動就會自動 call render()

所以元件可以自己管理自己的狀態和顯示
相對的也要管理自身被掛載和移除時應該要
做的事 自己管裡自己的生命週期



Components setState 同步問題

```
<script>
onClick = () => {
  this.setState({counter: this.state.counter + 1 })
  this.setState({counter: this.state.counter + 1 })
}
render() {
  console.log(this.state.counter)
  return(
    <button onClick={this.onClick}>Click Me</button>
  );
}
```

```
</script>
```

console 會印出什麼

Comp

```
<script>
onClick = ()
  this.s
  this.s
}
render() {
  console.
  return(
    <butto
  );
}
</script>
```

陷入沉思

什麼

Components
setState
是非同步的
所以....

```
1 class App extends React.Component {  
2   constructor(props) {  
3     super(props)  
4     this.state = {  
5       counter: 0  
6     }  
7   }  
8   onClick = () => {  
9     this.setState({counter: this.state.counter + 1 })  
10    this.setState({counter: this.state.counter + 1 })  
11  }  
12  render() {  
13    console.log(this.state.counter)  
14    return(  
15      <button onClick={this.onClick}>Click Me</button>  
16    );  
17  }  
18 }  
19  
20 ReactDOM.render(  
21   <App />,  
22   document.getElementById('root')
```

Click Me



Hello World in React

Chi-Jen Wu



HTML

CSS

JS (Babel)

```
1 class App extends React.Component {  
2   constructor(props) {  
3     super(props)  
4     this.state = {  
5       counter: 0  
6     }  
7   }  
8   onClick = () => {  
9     this.setState({counter: this.state.counter + 1 })  
10    this.setState({counter: this.state.counter + 1 })  
11  }  
12  render() {
```

Click Me

Console

1



Components setState

```
<script>
class App extends React.Component {
  constructor(props) {
    super(props)
    this.state = {
      counter: 0
    }
  }
  onClick = () => {
    this.setState((prevState, props) => ({ counter: prevState.counter + 1 }))
    this.setState((prevState, props) => {
      // console.log(prevState)
      return { counter: prevState.counter + 1 }
    })
  }
  render() {
    console.log(this.state.counter)
    return(
      <button onClick={this.onClick}>Click Me</button>
    );
  }
}

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

把之前的狀態傳進去



HTML
CSS

JS (Babel)

```
7 }  
8 onClick = () => {  
9   this.setState((prevState, props) => ({ counter: prevState.counter + 1 }));  
10  this.setState((prevState, props) => {  
11    // console.log(prevState)  
12    return { counter: prevState.counter + 1 };  
13  })  
14 }
```

Click Me

Drag to resize. Double-click to expand.

Console

0

2



Thanks!

Open for any questions

CJ Wu

cjwu@mail.cgu.edu.tw