React

介紹課程

米克 Merik, 2023/11/05

關於

Merik, 米克

https://hackmd.io/@merik-chen/resume

2013 Sep. to Present Backend / FullStack (10y)
2018 Sep. to Present Locale team lead / Backend Lead (4y)

–

9 years for PHP
4 years for Node.js / TypeScript
3 years for Python
1.5 years for GoLang

–

PHP / Node.js / Golang / Python
MySQL / Memcached / Redis / MongoDB
Typescript / FastAPI / Vue / React / Fastify
Docker / Google Cloud / AWS

OAuth / Third-party payment gateway / Third-party shipment / API integrations

–

Mandarin (Native) / Min Nan Chinese (Native) / English (TOEIC LR 795)

Introduce React

The library for web and native user interfaces

ReactJS 是 Facebook 推出的 JavaScript 函式庫，若以 MVC 框架來看，React 定位是在 View 的範疇。

在 ReactJS 0.14 版之後，ReactJS 更把原先處理 DOM 的部分獨立出去（react-dom），讓 ReactJS 核心更單純，也更符合 React 所倡導的 Learn once, write everywhere 的理念。事實上，ReactJS 本身的 API 相對單純，但由於整個生態系非常龐大，因此學習 React 卻是一條漫長的道路。

此外，當你想把 React 應用在你的應用程式時，你通常必須學習整個 React Stack 才能充分發揮 React 的最大優勢。

Why React?

特性及應用場景簡介

隨著網頁規模愈來越大，那種將後端語法與 HTML CSS 混合再一起的開發方式感覺到越來越痛苦。我們也稱這種為 Spaghetti code

jQuery，它的出現讓開發者得以用一套的語法，跨瀏覽器地取得和操作 DOM（Document Object Model）等功能，使得開發網頁更加輕鬆。同時也迎來了 ajax 的來臨，一時之間前端的功能與架構變得越來越豐富與厚重。大量操作 DOM 的需求也讓這個時代下的開發者越顯痛苦，效能也越來越差...

2013 年，Facebook 開源了他們專注於 UI 的函式庫 React

Why React? - 續

特性

- 不直接操作網頁元件 Virtual DOM
- 能重複利用的元件 Reusable components
- 單一資料流向 Unidirectional data flow
- 只著重在 UI 輕量且效率高

React v.s. Vue v.s. Angular

簡單比較

- Angular：是一個由Google開發並維護的全面型JavaScript框架，可以滿足「企業級」的架構，適合有足夠的人力且延展性很高的產品。

  學習難度：高

- React：由 Facebook 開發並維護的 UI 函式庫，輕巧且靈活能夠快速的讓開發者產出。但需要搭配其他工具來滿足開發功能需求。

  學習難度：中等

- Vue：由社區撐起並維護的框架，保留了 HTML 與 CSS 在組件中，開發者能以較熟悉的思路來學習。

  學習難度：低

Parameter	Angular	React	Vue
Initial Release	2016	2011	2014
Support	Google	Facebook	Community
Type	Framework	Library	Framework
Size	Medium	Small	Very small
Language	TypeScript	JavaScript	JavaScript
Performance	Good	Good	Good
Data Binding	Both	Unidirectional	Bidirectional
Learning Curve	Steep	Easy	Easy
Popular Websites	Paypal, Samsung, Upwork	Netflix, Twitter, Amazon	Alibaba, Grammarly, GitLab

React 架構

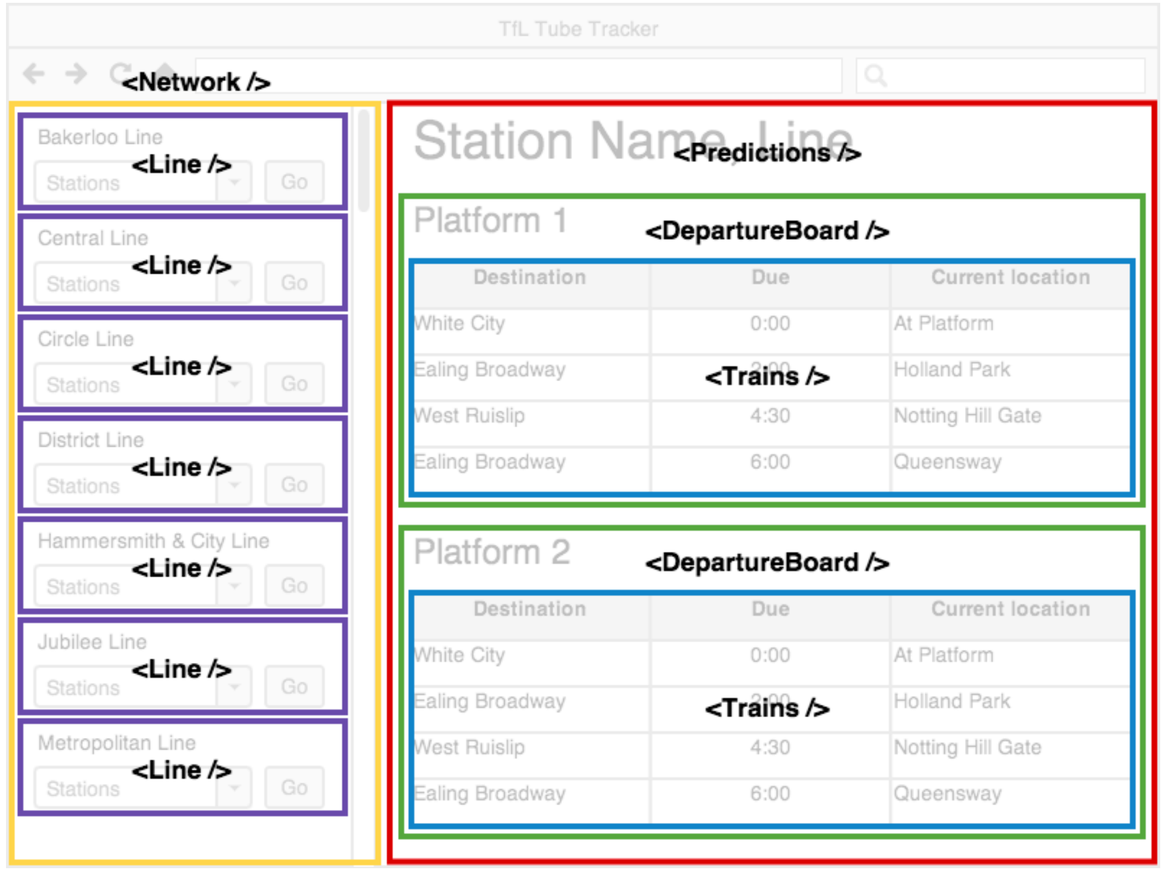
React components

利用元件，讓網頁像樂高一樣拼裝起來

在 React 的世界中最基本的單元為元件（Component），每個元件也可以包含一個以上的子元件，並依照需求組裝成一個組合式的（Composable）

- 封裝（encapsulation）
- 關注點分離 (Separation of Concerns)
- 複用 (Reuse)
- 組合 (Compose)

React components



React props v.s. state

React props v.s. state

在 React 中 state 與 props 雖然都被定義在 component 中，但是它們的用途卻大相逕庭

在一個元件中 props 代表在元件函式中傳入的參數；而 state 則是在元件內自行管理的，可以想像成在函式內定義的變數。

所以 state 如果被當作參數傳給子元件時，它就變成了子元件的 props。

Props

```
/* Parent.js */
function Parent() {
  const [moneyForSister, setMoneyForSister] = useState(40)
  const increaseMoney = () => {
    setMoneyForSister(moneyForSister + 20)
  }
  return (
    <div>
      <Sister money={moneyForSister} increase={increaseMoney}/>
    </div>
  )
}

export default Parent;
```

```
/* Sister.js */
function Sister({ money, argue }) {
  <div>我是女兒，我拿到{money}<button onClick={increase}>要求增加20塊</button></div>
}
export default Sister;
```

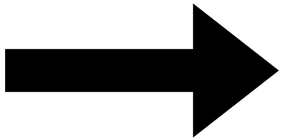
當按下 argue 鍵的時候，moneyForSister 會被改為 60，並觸發 Parent 重新渲染，所以 Sister 也會顯示 60 元。

State

```
const [count, setCount] = useState(0)

function increment() {
  setCount((preState) => {
    preState ++
    console.log(count)
    return preState
  });
}

function handleIncrementThreeTimes() {
  increment()
  increment()
  increment()
}
```

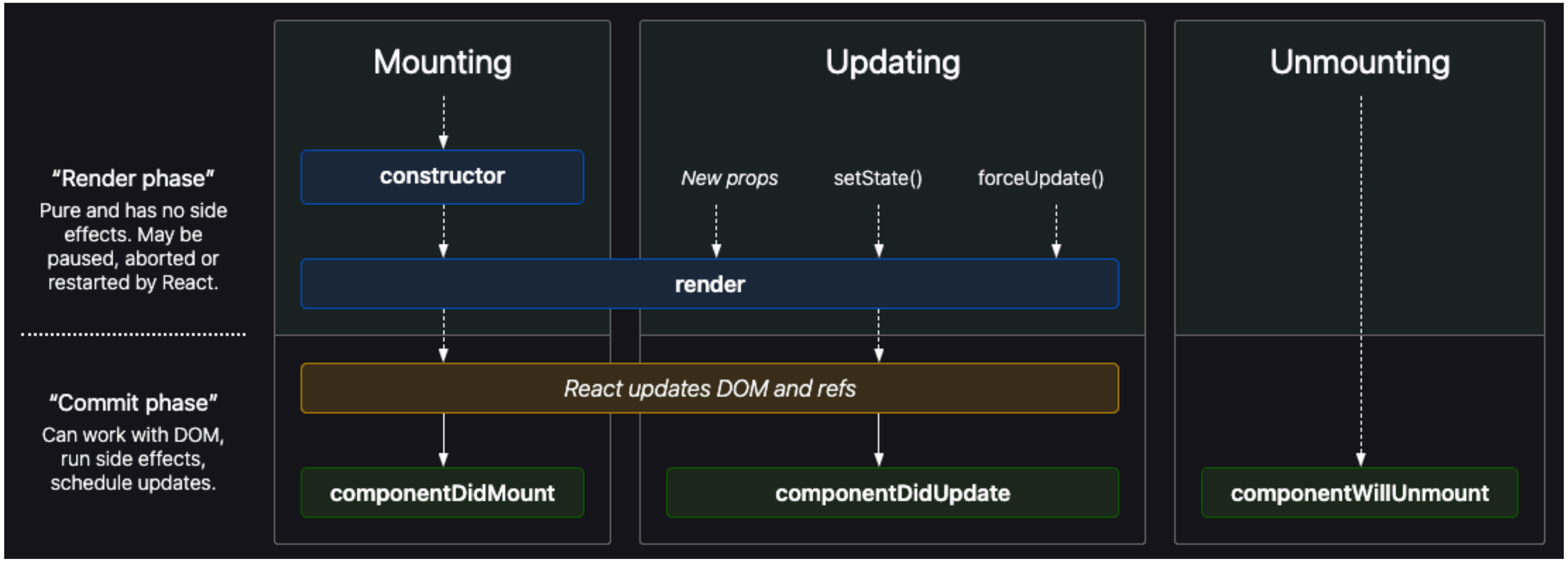


```
count: 0
count: 0
count: 0
```

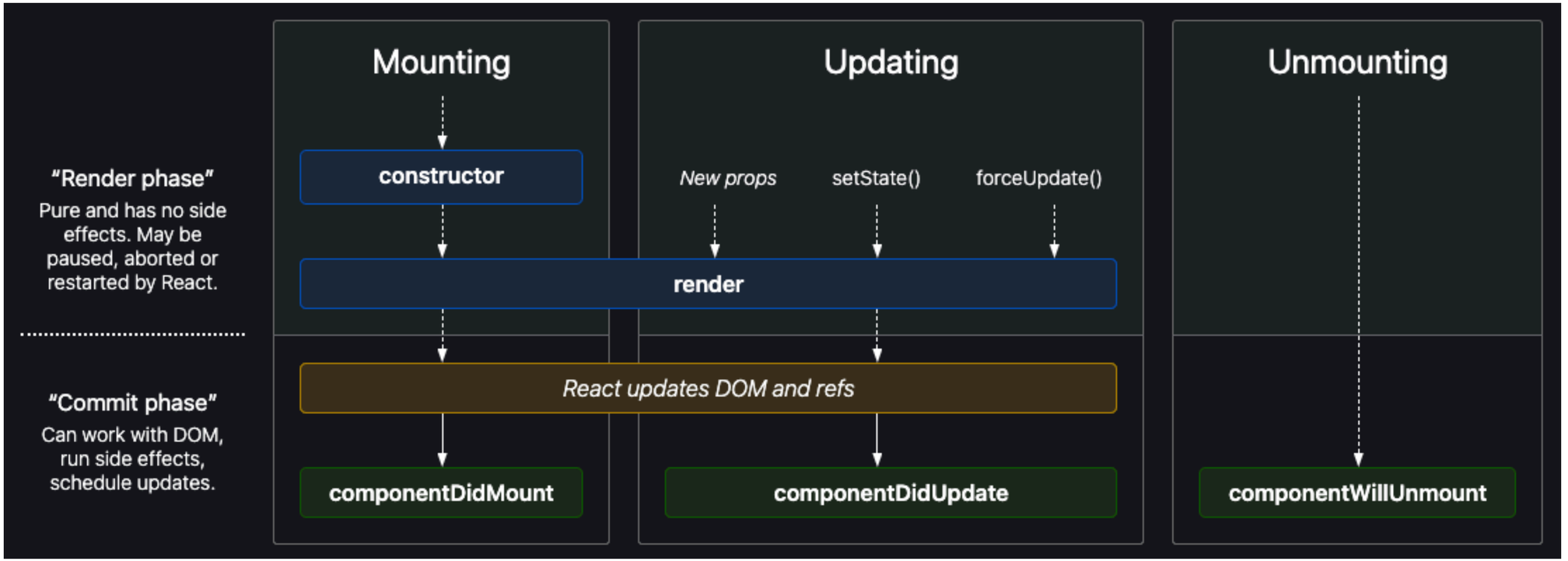
State 為非同步更新，只會在完成渲染後更新數值。
此舉是為了效能考量，當多個 setState(setCount) 呼叫時，
React會自動將他們合成成一個批次一起執行再更新畫面。

```
count: 3
```

React Lifecycle



React Lifecycle - 續



- constructor :用來初始化的地方，還沒掛載到DOM的時候，假設沒有寫super() ，就調用this的話會報錯
- render：必須實作的，回傳JSX
- componentDidMount: DOM已經掛載完成 ，在這個階段可以呼叫api來更新DOM ，適合做一些初始化的工作
- componentDidUpdate: 當props or state更新 ，就會觸發組件更新DOM，所以千萬不要在這個階段setState，會造成無限循環
- componentWillUnmount: component從DOM被移除 ，在這階段可以用來清除一些計時器

JSX 基礎語法

```
const element = <h1>你好，世界！</h1>;
```

在 JSX 中嵌入 Expression

```
const name = 'Josh Perez';
const element = <h1>Hello, {name}</h1>;
```

React DOM 預設會在 render 之前 escape 所有嵌入在 JSX 中的變數。這保證你永遠不會不小心注入任何不是直接寫在你的應用程式中的東西。所有變數都會在 render 之前轉為字串，這可以避免 XSS（跨網站指令碼）攻擊。

在 JSX 中嵌入 Expression 並呼叫 Function

```
function formatName(user) {
  return user.firstName+ ' ' + user.lastName;
}

const user = {
  firstName: 'Harper',
  lastName: 'Perez'
};

const element = (
  <h1>
    Hello, {formatName(user)}!
  </h1>
);
```

在 JSX 中指定屬性

由於 JSX 比較接近 JavaScript 而不是 HTML，React DOM 使用 camelCase 來命名屬性而不是使用慣有的 HTML 屬性名稱。

```
const user = {
  href: ‘https://www.reactjs.org',
};

const element = <img src={user.href} className=“link”></img>;
```

在 JSX 中指定 Children 且使用 inline css

```
const element = (
  <div>
    <h1>Hello!</h1>
    <h2 style={{
      backgroundColor: 'black',
      color: ‘pink'
    }}>Good to see you here.</h2>
  </div>
);
```

一個完整的 JSX 片段

```
const element = (props) => {
  const {status} = props;
  const [updating, setUpdating] = useState(false);
  useEffect(() => {
    console.log(status);
  }, [status])

  return <Fragment>
    <h2>The status: {status}</h2>
    <button onClick={() => {setUpdating(true)}}>Update</button>
    {updating ? <p>Updating…</p> : null}
  </Fragment>
};

export default element;
```

React 基礎知識

Events

事件的名稱在 React 中都是 camelCase，而在 HTML DOM 中則是小寫

```
<button onclick="activateLasers()">
  Activate Lasers
</button>
```

HTML DOM

```
<button onClick={activateLasers}>
  Activate Lasers
</button>
```

React

另外一個差異是，在 React 中，你不能夠使用 `return false` 來避免瀏覽器預設行為。你必須明確地呼叫 `preventDefault`。

```
function activateLasers(e) {
    e.preventDefault();
    console.log('You clicked submit.');
  }
```

Conditional Rendering

在 React 中，你可以建立不同的 component 來封裝你需要的行為。接著，你可以根據你的應用程式的 state，來 render 其中的一部份。

```jsx
function Item({ name, isPacked }) {
  return (
    <li className="item">
      {isPacked ? <del>{name + ' ✔'}</del> : name}
    </li>
  );
}

export default function PackingList() {
  return (
    <section>
      <h1>Sally Ride's Packing List</h1>
      <ul>
        <Item
          isPacked={true}
          name="Space suit"
        />
        <Item
          isPacked={false}
          name="Photo of Tam"
        />
      </ul>
    </section>
  );
}

```

Rendering Lists

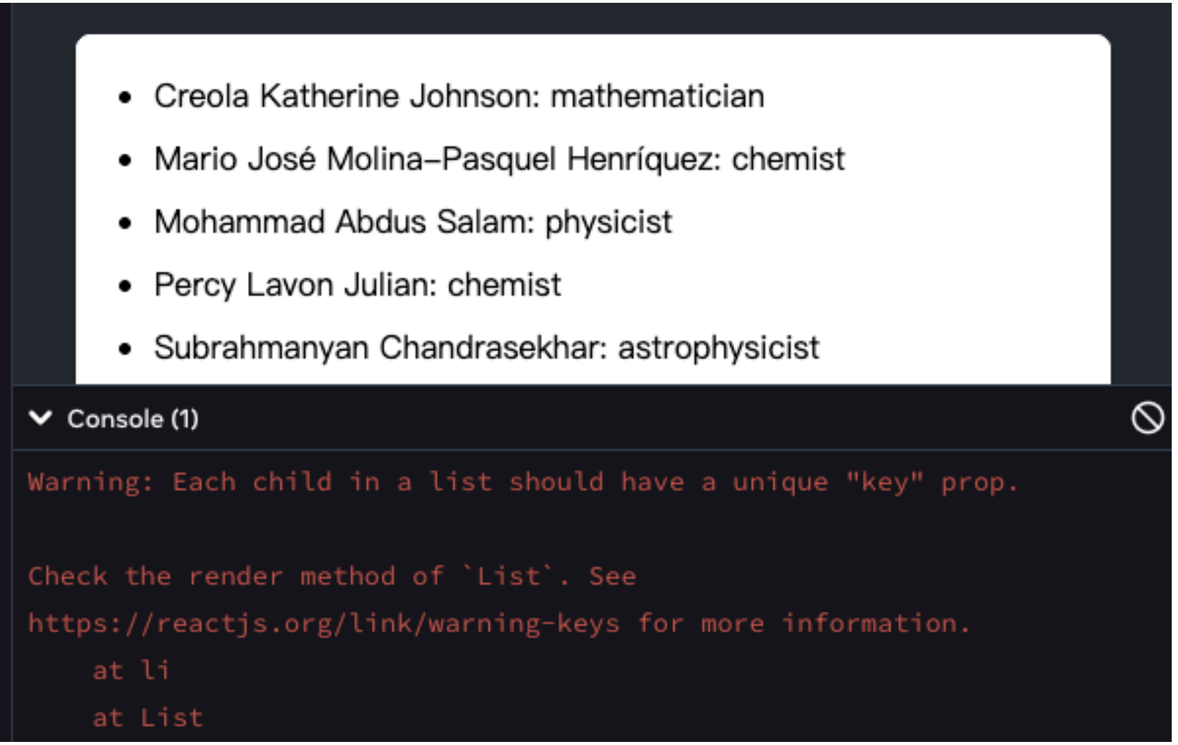
Array.prototype.map()

map() 方法會建立一個新的陣列，其內容為原陣列的每一個元素經由回呼函式運算後所回傳的結果之集合。

[1, 4, 9, 16].map(number => number * 2) => [2, 8, 18, 32]

Rendering Lists

```
1  const people = [
2    'Creola Katherine Johnson: mathematician',
3    'Mario José Molina-Pasquel Henríquez: chemist',
4    'Mohammad Abdus Salam: physicist',
5    'Percy Lavon Julian: chemist',
6    'Subrahmanyan Chandrasekhar: astrophysicist'
7  ];
8
9  export default function List() {
10   const listItems = people.map(person =>
11     <li>{person}</li>
12   );
13   return <ul>{listItems}</ul>;
14 }
15
```



Warning: Each child in a list should have a unique “key” prop.

Rendering Lists

```
const people = [
  'Creola Katherine Johnson: mathematician',
  'Mario José Molina-Pasquel Henríquez: chemist',
  'Mohammad Abdus Salam: physicist',
  'Percy Lavon Julian: chemist',
  'Subrahmanyan Chandrasekhar: astrophysicist'
];

export default function List() {
  const listItems = people.map((person, index) =>
    <li key={index}>{person}</li>
  );
  return <ul>{listItems}</ul>;
}

```

- Creola Katherine Johnson: mathematician
- Mario José Molina–Pasquel Henríquez: chemist
- Mohammad Abdus Salam: physicist
- Percy Lavon Julian: chemist
- Subrahmanyan Chandrasekhar: astrophysicist

React, a little more…advanced

use...

- useRef：與 useState 類似，但是他是返回一個可變動的 object 裡面有擁有 current 物件來存放資料。但是不會觸發渲染，所以無法用來關聯畫面中的變數。
- useState：在 function component 裡面提供相當於 class 的 this.state / this.setState 並且為非同步，批次堆疊 / 觸發渲染的資料儲存物件。因此可用來關聯畫面中的變數
- useEffect：提供 React class 的生命週期方法。你可以把 useEffect 視為 componentDidMount，
  componentDidUpdate 和 componentWillUnmount 的組合。

useEffect

Side effects

- 無需清除的 Effect：有時候我們需要呼叫一些 API 或是 logging 紀錄，他們這種都是無需清除的 effect 執行完我們即可釋放 / 忘記他們
- 需清除的 Effect：最直觀的範例是 websocket / database connection。我們需要連線，但在結束之後也必須做清理關閉以防止程式效能下降甚至是 memory leak 的情況
- 每次重新渲染畫面，useEffect都會重新執行一次。
- 通過選定觀察目標來優化效能

無需清除的 Effect

```
function Example() {
  const [count, setCount] = useState(0);

  useEffect(() => {
    document.title = `You clicked ${count} times`;
  });
}
```

無需清除的 Effect

```
function FriendStatus(props) {
  const [isOnline, setIsOnline] = useState(null);

  useEffect(() => {
    function handleStatusChange(status) {
      setIsOnline(status.isOnline);
    }

    ChatAPI.subscribeToFriendStatus(props.friend.id, handleStatusChange);
    // 指定如何在這個 effect 之後執行清除：
    return () => {
      ChatAPI.unsubscribeFromFriendStatus(props.friend.id, handleStatusChange);
    };
  });
  if (isOnline === null) {
    return 'Loading...';
  }
  return isOnline ? 'Online' : 'Offline';
}
```

忽略 Effect 來提升效能

```
function Example() {
  const [count, setCount] = useState(0);
  const [updating, setUpdating] = useState(false);

  useEffect(() => {
    document.title = `You clicked ${count} times`;
  });

  useEffect(() => {
    console.log(updating);
  });

}
```

忽略 Effect 來提升效能

```
function Example() {
  const [count, setCount] = useState(0);
  const [updating, setUpdating] = useState(false);

  useEffect(() => {
    document.title = `You clicked ${count} times`;
  }, [count]);

  useEffect(() => {
    console.log(updating);
  }, [updating]);

}
```

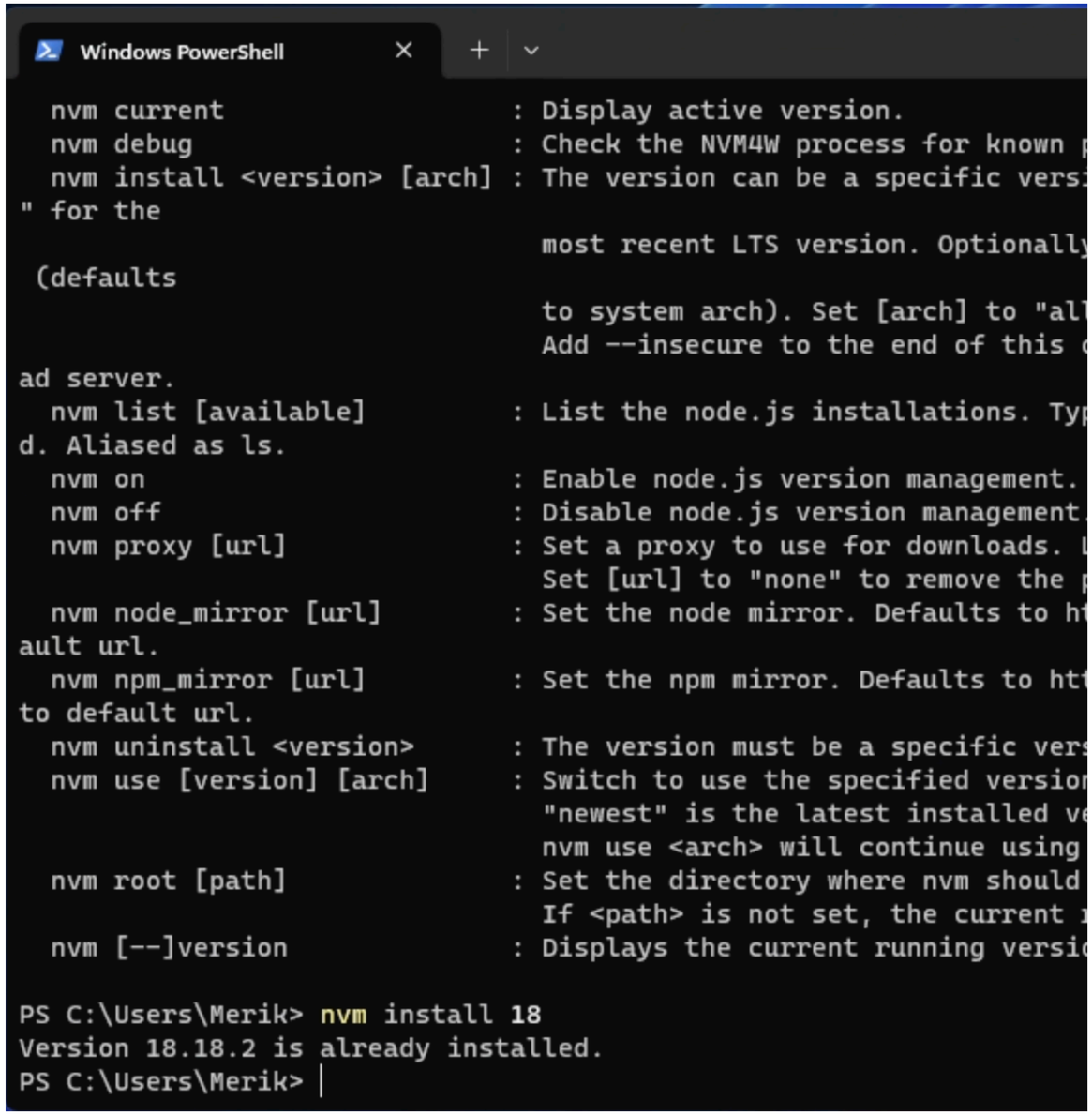
nvm for windows

Installation

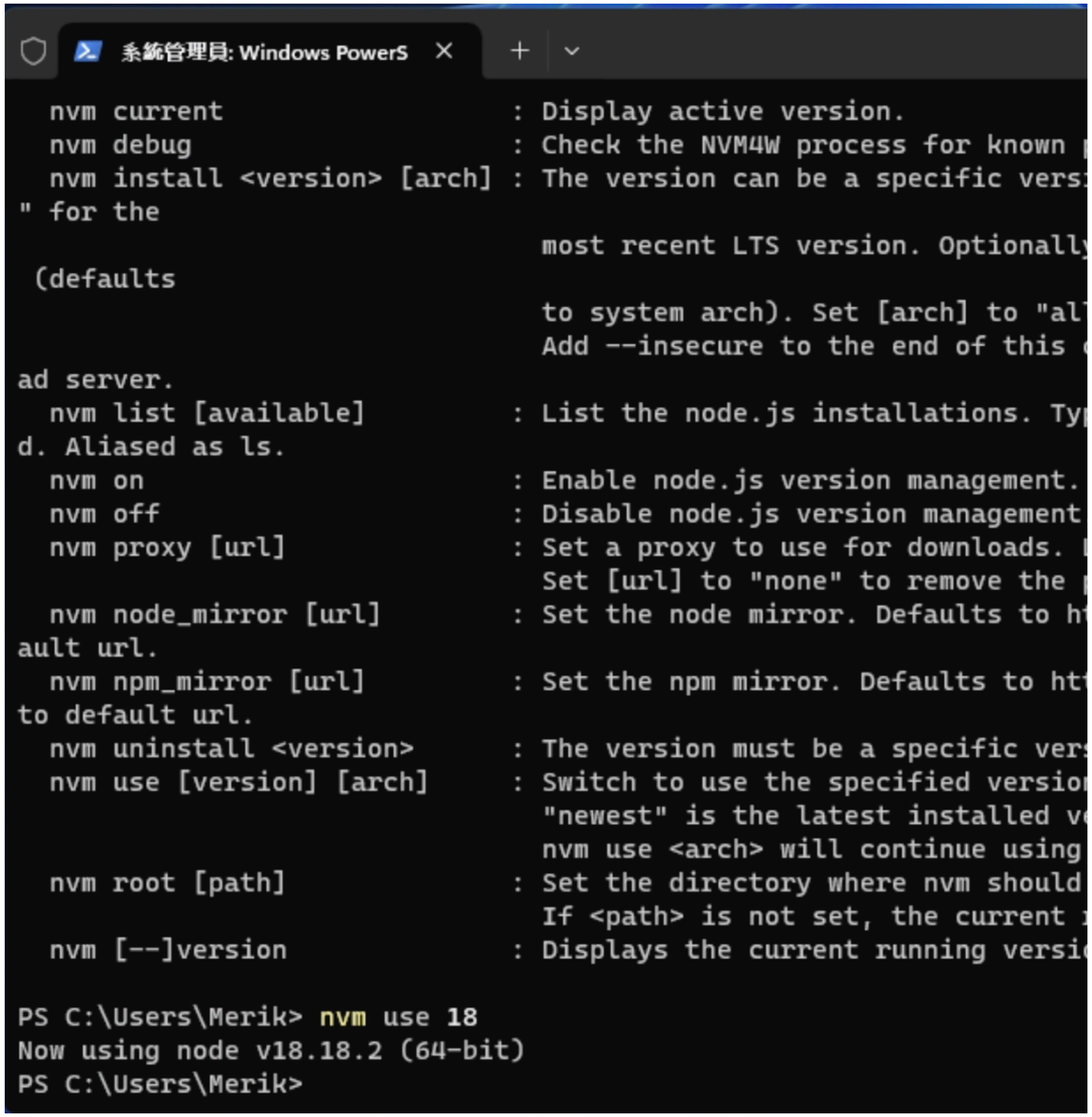
Prepare the environment

- https://github.com/coreybutler/nvm-windows/releases
- > nvm install 18
- > nvm use 18

nvm install 18



nvm use 18


系統管理員: Windows PowerS
nvm current : Display active version.
nvm debug : Check the NVM4W process for known
nvm install <version> [arch] : The version can be a specific vers
" for the
most recent LTS version. Optionall
(defaults
to system arch). Set [arch] to "al
Add --insecure to the end of this
ad server.
nvm list [available] : List the node.js installations. Ty
d. Aliased as ls.
nvm on : Enable node.js version management.
nvm off : Disable node.js version management
nvm proxy [url] : Set a proxy to use for downloads.
Set [url] to "none" to remove the
nvm node_mirror [url] : Set the node mirror. Defaults to h
ault url.
nvm npm_mirror [url] : Set the npm mirror. Defaults to ht
to default url.
nvm uninstall <version> : The version must be a specific ver
nvm use [version] [arch] : Switch to use the specified versio
"newest" is the latest installed v
nvm use <arch> will continue using
nvm root [path] : Set the directory where nvm should
If <path> is not set, the current
nvm [--]version : Displays the current running versi
PS C:\Users\Merik> nvm use 18
Now using node v18.18.2 (64-bit)
PS C:\Users\Merik>

Simple Showcase: ToDo List