

# React

- [【React】建立React 環境](#)
- [【React】資料綁定](#)
- [【React】JSX 如何轉譯](#)
- [【React】JSX 開發常見問題](#)
- [【React】JSX 與 Html 標籤屬性](#)
- [【React Admin】定時refresh](#)

# 【React】建立React 環境

引入js

```
<script crossorigin src="https://unpkg.com/react@18/umd/react.development.js"></script>
<script crossorigin src="https://unpkg.com/react-dom@18/umd/react-dom.development.js"></script>
<script src="https://unpkg.com/babel-standalone@6/babel.min.js"></script>
```

建立根物件

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

渲染Dom物件

# Hello React

```
<script type="text/babel">
function App(){
  return <h1>Hello React</h1>
}

const el = document.getElementById("root");
const root = ReactDOM.createRoot(el);
root.render(<App />);
/* 簡易寫法
ReactDOM
  .createRoot(el)
  .render(<App />);
*/
</script>
```

# 【React】資料綁定

## 單向綁定

使用大括弧綁定變數，標籤內屬性不用加雙引號

```
const data = {
  imgUrl : "https://images.unsplash.com/photo-1650190558669-e8cc1ae3551f?ixlib=rb-4.0.3&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=1770&q=80",
  titl: "AAAA",
  content : "Lorem ipsum dolor sit amet consectetur adipisicing elit. Eius, esse sequi? Dolores, perspiciatis! Minima aliquid tempore placeat ut qui atque asperiores totam iusto, dolorem est blanditiis nesciunt minus. Officiis, voluptate!",
  link: "https://www.google.com.tw"
}

function App(){

  return <div className="card">
    <img src={data.imgUrl} className="card-img-top" alt="..." />
    <div className="card-body">
      <h5 className="card-title">{data.titl}</h5>
      <p className="card-text">{data.content}</p>
      <a href={data.link} className="btn btn-primary">Go somewhere</a>
    </div>
    { /** 可用註解 */ }
    { (function(){return "可function"})() }
    { tt ? tt : "可用三元運算" }
    { /** 不可用物件但可用 JSON.stringify 轉為字串除錯 */ }
    { JSON.stringify(data) }
    { /** true, false, undefined, null 不會顯示 */ }
    { [1,2,3] /** 純值陣列會直接顯示 ex 1 2 3 */ }
  </div>
}
```

## 綁定陣列

```
function App() {
  const data = {
    imageUrl: 'https://images.unsplash.com/photo-1564564321837-a57b7070ac4f?ixlib=rb-1.2.1&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=2352&q=80',
    name: '卡斯伯',
    description: 'Lorem ipsum dolor sit amet consectetur, adipisicing elit. Modi culpa aspernatur tempore cupiditate illo minima voluptatem blanditiis optio accusantium earum voluptate magni repellendus, ipsa dolor adipisci libero qui commodi veritatis.',
    link: 'https://www.casper.tw/'
  }

  // #3 直接插入 HTML 結構的變數
  // https://getbootstrap.com/docs/5.2/components/card/#list-groups
  let list = [
    <li className="list-group-item">An item</li>,
    <li className="list-group-item">A second item</li>,
    <li className="list-group-item">A third item</li>,
  ];
  return <div className="card w-50">
    <img src={data.imageUrl} className="card-img-top" alt="..." />
    <div className="card-body">
      <h5 className="card-title">{ data.name }</h5>
      <p className="card-text">{ data.description }</p>
      <a href={data.link} target="_blank" className="btn btn-primary">Go somewhere</a>
    </div>
    { /* 2. 陣列中的值會直接呈現在畫面上 */ }
    <ul className="list-group list-group-flush">
      {list}
    </ul>
  </div>
```

```
</div>
}
```

# 綁定Style

## 兩個大括號包住 style = {{ }}

```
<!-- 原html-->
<div class="card" style="width: 18rem">
  <div
    class="card-img-top"
    style="
      height: 200px;
      background-image: url('https://images.unsplash.com/photo-1564564321837-a57b7070ac4f?ixlib=rb-
1.2.1&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=2352&q=80');

      background-size: cover;
      background-position: center center;
    "
  ></div>
  <div class="card-body">
    <h5 class="card-title">卡斯伯</h5>
    <p class="card-text">
      Lorem ipsum dolor sit amet consectetur, adipisicing elit. Modi culpa
      aspernatur tempore cupiditate illo minima voluptatem blanditiis optio
      accusantium earum voluptate magni repellendus, ipsa dolor adipisci libero
      qui commodi veritatis.
    </p>
    <a href="https://www.casper.tw/" target="_blank" class="btn btn-primary"
      >Go somewhere</a>
  >
</div>
</div>
```

```
/**
有 - 要改為駝峰式命名
background-image => backgroundImage
*/
return (
  <div className='card' style={{ width: "18rem" }}>
    <div className='card-img-top' style={{
      height: '200px',
      backgroundImage: `url('https://images.unsplash.com/photo-1564564321837-a57b7070ac4f?ixlib=rb-
1.2.1&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w=2352&q=80')`

      backgroundColor: 'cover',
      backgroundPosition: "center center"
    }}>
    </div>
    <div className='card-body'>
      <h5 className='card-title'>{data.name}</h5>
      <p className='card-text'>{data.description}</p>
      <a href={data.link} target='_blank' className='btn btn-primary'>
        Go somewhere
      </a>
    </div>
  </div>
);
```

# 【React】JSX 如何轉譯

```
// JSX 寫法
function App() {
  return <h1>React 我來了 <small className="text-danger">{new Date().toLocaleDateString()}</small></h1>
}

// JS 寫法 React.createElement
function App(){
  // React.createElement({tag}, {屬性}, {值})
  return React.createElement('h1',null, "React 我來了",
    React.createElement(
      'small',
      {
        className : "text-danger"
      },new Date().toLocaleDateString())
    );
}
```

# 【React】JSX 開發常見問題

## 1. 定義元件一定是手字大寫

```
/** wrong*/  
function app(){  
}  
  
/** correct*/  
function App(){  
}
```

## 2. 多個元素外層需要使用標籤包覆，或使用 React.Fragment

```
// wrong  
function App(){  
  // return 後沒有東西預設會補; => 顯示錯誤  
  return  
    <div>  
      .....  
    </div>  
}  
  
// correct  
function App(){  
  return <div>  
    .....  
  </div>  
}
```

```
// 不想包div 用 React.Fragment  
function App(){  
  return <React.Fragment>  
    .....  
  </React.Fragment>  
}  
  
// 同上  
function App(){  
  return <>  
    .....  
  </>  
}
```

## 3. 沒有正確結尾，結尾不正確

```
<!-- wrong -->  
<input class="form-control" list="datalistOptions"  
  id="exampleDataList" placeholder="Type to search...">  
  
<!-- correct -->  
<input className="form-control" list="datalistOptions"  
  id="exampleDataList" placeholder="Type to search..." />
```

## 4. 屬性轉換名稱運用要正確

```
<!-- wrong-->  
<label htmlFor="exampleDataList" className="form-label" style="color: var(--bs-blue)">  
  Datalist example  
</label>  
  
<!-- correct-->  
<label htmlFor="exampleDataList" className="form-label" style={{color: 'var(--bs-blue)'}}>  
  Datalist example  
</label>
```

## 5. return 建議加入括號

```
// wrong  
function App(){  
  // return 後沒有東西預設會補; => 顯示錯誤
```

```
return
  <div>
    ....
  </div>
}
// correct
function App(){
  return (<div>
    .....
  </div>)
}
```

# 【React】JSX 與 Html 標籤屬性

```
<script type="text/babel">
function App() {
  const htmlTemplate = { __html : '<div>這裡有一段文字</div>' }

  return (
    <div>
      { /* 使用 dangerouslySetInnerHTML 顯示html */ }
      <div dangerouslySetInnerHTML={htmlTemplate}></div>

      { /* class => className */ }
      <div><button type="button" className="btn btn-primary">Primary</button></div>

      { /* checked => defaultChecked 與 結尾標籤 */ }
      <div><input type="checkbox" defaultChecked /></div>

      { /* value => defaultValue */ }
      <div><input type="text" defaultValue="卡斯伯" /></div>

      { /* for => htmlFor */ }
      <div>
        <label htmlFor="email">請輸入 Email</label>
        <input type="email" id="email" />
      </div>

      { /* selected => defaultValue */ }
      <div>
        <select name="" id="" defaultValue="2">
          <option value="1">1</option>
          <option value="2">2</option>
          <option value="3">3</option>
        </select>
      </div>

      { /* textarea 文字 要使用 defaultValue */ }
      <div>
        <textarea name="" id="" cols="30" rows="5" defaultValue="這裡可以放多行文字"></textarea>
      </div>

      { /* onChange */ }
      <div><input type="text" onChange={
        function(event){
          console.log(event.target.value);
        }
      } /></div>
    </div>
  );
}

const el = document.getElementById('root');
const root = ReactDOM.createRoot(el);
root.render(<App />);
```



# 【React Admin】定時refresh

React Admin 是一個基於 React 框架的開源後台管理界面框架。若要定時刷新 React Admin 的數據，可以使用 React 的 `useEffect` 鉤子來定時刷新。

以下是一個示例代碼：

```
import React, { useState, useEffect } from 'react';
import { Admin, Resource, ListGuesser } from 'react-admin';
import jsonServerProvider from 'ra-data-json-server';

const dataProvider = jsonServerProvider('https://jsonplaceholder.typicode.com');

const App = () => {
  const [refreshInterval, setRefreshInterval] = useState(0);

  useEffect(() => {
    const interval = setInterval(() => {
      setRefreshInterval(prevRefreshInterval => prevRefreshInterval + 1);
    }, 10000); // 10 秒刷新一次

    return () => {
      clearInterval(interval);
    };
  }, []);

  return (
    <Admin dataProvider={dataProvider}>
      <Resource name="posts" list={ListGuesser} refresh={refreshInterval} />
    </Admin>
  );
};

export default App;
```

在這個示例中，我們使用 `useState` 鉤子來保存刷新間隔時間，然後使用 `useEffect` 鉤子來啟動定時器，每10秒刷新一次。我們還在 `Resource` 中傳遞了 `refreshInterval`，它將在 `useState` 中更新，這樣它會觸發 `Resource` 的重新加載。

需要注意的是，經常性的刷新會增加服務器負載，並且可能會降低應用程序的性能。因此，應該謹慎使用定時刷新，並根據實際需要進行調整。