

Component Life Cycle Refs, HOC

Ant Design, Recharts

BY PANOT WONGKHOT

FRONT-END PROGRAMMER BOOTCAMP

<https://github.com/panotza/frontend-bootcamp>



Component Life Cycle



Component Life Cycle

Component มีวงจรชีวิตของมัน เราเรียกว่า Life Cycle โดยปกติแล้วเราจะใช้ Life Cycle ในการทำอะไรสักอย่าง ณ ช่วงเวลาใดเวลาหนึ่งของ Component นั้นๆ

Life Cycle มีให้ใช้แค่ Component ที่สร้างจาก Class เท่านั้น



Component Life Cycle

React ได้สร้าง hook ของ Life Cycle มาให้เราแล้ว เราแค่เรียนรู้ว่า hook แต่ละตัวจะถูกเรียกใช้เมื่อไหร่ แล้วก็เติม logic ลงไปใน hook ให้ถูกต้องเท่านั้น



Component Life Cycle

Life Cycle มีทั้งหมด 4 ช่วงได้แก่

- Initialization / Mounting
- Updation
- Unmounting
- Error Catching



Hooks

Initialization / Mounting



constructor

เกิดขึ้นเมื่อ component ถูกสร้าง component จะมีการกำหนด default props เหมาะสำหรับ

- Initialize State
- Bind Method (หากไม่ได้เขียน Method ด้วย Arrow Function)

ไม่ควร

- ทำ action ที่มี side effect เช่น Fetch



componentWillMount

เกิดขึ้นหลังจาก constructor แต่ก่อนจะ mount (ก่อน render)

*จะถูกเอาออกใน React 17

*ใช้ `getDerivedStateFromProps` หรือ `componentDidMount` แทน



getDerivedStateFromProps

เกิดขึ้นหลังจาก constructor แต่ก่อนจะ mount (ก่อน render)

hook นี้จะ return ข้อมูลเพื่อ update state ถ้าไม่มีอะไร update ให้ return null

เหมาะสำหรับ

- setState โดยอิงจาก props



getDerivedStateFromProps



ดูได้ใน [getderivedstatefromprops.html](#)

```
static getDerivedStateFromProps(nextProps, prevState) {  
  if (nextProps.reset === true) {  
    return { score: 0 };  
  }  
  return null;  
}
```

render

เกิดขึ้นหลังจาก `getDerivedStateFromProps` เหมาะสำหรับ

- สร้าง react element

ไม่ควร

- เรียก `setState` เพราะจะทำให้เกิด Infinity Loop

componentDidMount (ใช้บ่อย)

เกิดขึ้นหลังจาก component ถูก render แล้ว
เหมาะสำหรับ

- ทำ action ที่มี side effect เช่น Fetch
- Subscribe Event ต่างๆ
- Initialization ที่ต้องการ DOM
- DOM manipulation



componentDidMount



ดูได้ใน componentdidmount-fetch.html

```
async componentDidMount() {  
  const response = await fetch('https://api.coindesk.com/api..');  
  const data = await response.json();  
  this.setState({ USD: data.bpi.USD });  
}
```


componentDidMount



ดูได้ใน componentdidmount-setinterval.html

```
increment = () => {  
  this.setState((prevState) => (  
    { counter: prevState.counter + 1 }  
  ));  
}  
  
componentDidMount() {  
  this.counter = setInterval(this.increment, 1000);  
}
```

Hooks Updation



componentWillReceiveProps

เกิดขึ้นตอน parent ของ component มีการ update

*จะถูกแทนที่ด้วย `getDerivedStateFromProps` ใน React 17



getDerivedStateFromProps

เกิดขึ้นตอน parent ของ component มีการ update hook นี้จะ return ข้อมูลเพื่อ update state ถ้าไม่มีอะไร update ให้ return null

เหมาะสำหรับ

- setState โดยอิงจาก props



getDerivedStateFromProps (ช่วง Updation)



ดูได้ใน `getderivedstatefromprops.html`

```
static getDerivedStateFromProps(nextProps, prevState) {  
  if (nextProps.reset === true) {  
    return { score: 0 };  
  }  
  return null;  
}
```


shouldComponentUpdate (ใช้บ่อย)

เกิดขึ้นหลังจาก hook `getDerivedStateFromProps` หรือ หลังจากสั่ง `setState` ใช้เพื่อกำหนดว่า component ควรจะ update หรือไม่ โดย default แล้วจะ return เป็น true

ใช้สำหรับ

- optimize performance



shouldComponentUpdate



ดูได้ใน shouldcomponentupdate.html

```
shouldComponentUpdate(nextProps, nextState) {  
  if (nextProps.values !== this.props.values) {  
    return true;  
  } else {  
    return false;  
  }  
}
```

componentWillUpdate

เกิดขึ้นหลังจาก hook shouldComponentUpdate แต่ก่อนจะ render

*จะถูกเอาออกใน React 17

*ใช้ getSnapshotBeforeUpdate หรือ componentDidUpdate แทนดีกว่า



render (ช่วง Updation)

เกิดขึ้นหลังจาก shouldComponentUpdate



getSnapshotBeforeUpdate

เกิดขึ้นหลัง render แต่ก่อน component จะ update hook นี้สามารถ return ค่าไปให้ hook componentDidUpdate ได้

เหมาะสำหรับ

- เก็บค่าต่างๆของ DOM เช่น scrollHeight เพื่อนำไป update ที่หลัง

componentDidUpdate

เกิดขึ้นหลังจาก `getSnapshotBeforeUpdate`
เหมาะสำหรับ

- action ที่มี side effect เช่น Fetch
- กำหนดค่า DOM

componentDidUpdate



ดูได้ใน
componentdidupdate.html

```
state = {  
  user: null  
}  
  
async componentDidUpdate(prevProps, prevState, snapshot) {  
  if (prevProps.willFetch === false && this.props.willFetch === true) {  
    const response = await fetch('https://randomuser.me/api/');  
    const data = await response.json();  
    this.setState({ user: data.results[0] });  
  }  
}
```

Hooks

Unmouting



componentWillUnmount

เกิดก่อนที่ component จะถูกลบออก
เหมาะสำหรับ

- ลบ timer ต่างๆ
- Unsubscribe Event ต่างๆ
- Cancel Ajax Call



componentWillUnmount



ดูได้ใน componentdidmount-setinterval.html

```
componentWillUnmount() {  
  clearInterval(this.counter);  
}
```


Hooks

Error Catching



componentDidCatch

เกิดเมื่อ children component เกิด error
เหมาะสำหรับ

- ทำ Fallback UI เมื่อเกิด Error
- Log Error

componentDidCatch

ดูได้ใน componentdidcatch.html



```
class ErrorBoundary extends React.Component {  
  state = {  
    error: null,  
    errorInfo: null  
  }  
  
  componentDidCatch(error, errorInfo) {  
    this.setState({  
      error: error,  
      errorInfo: errorInfo  
    });  
  }  
}
```

componentDidCatch



ดูได้ใน componentdidcatch.html

```
render() {  
  if (this.state.error) {  
    return ( // Fallback UI if an error occurs  
      <h2>{"Oh-no! Something went wrong"}</h2>  
    );  
  }  
  return this.props.children;  
}
```

componentDidCatch



ดูได้ใน componentdidcatch.html

```
class App extends React.Component {  
  render() {  
    return (  
      <ErrorBoundary>  
        <BuggyButton />  
      </ErrorBoundary>  
    );  
  }  
}
```

Life Cycle Simulators

สามารถทดลองเล่น Life Cycle ได้ที่

<https://reactarmory.com/guides/lifecycle-simulators>



Lab 1: Component Life Cycle

เปิดไฟล์ lab1.html แล้วลองสังเกตการทำงานของ Life Cycle

- ทดลองใช้ setInterval , setTimeout ใน ComponentDidMount
- ทดลองใช้ Fetch เพื่อดึงข้อมูลใน ComponentDidMount
- ทดลองเล่น shouldComponentUpdate



Refs



Refs

refs ย่อมาจาก references เราใช้เพื่ออ้างถึง reference ของ component instance / dom element โดยตรง



Refs: use case

- ใช้จัดการ focus, text selection, media playback
- ใช้ร่วมกับ third party dom library



การใช้ Refs

เราแค่ใส่ props ref แล้วกำหนดให้ค่าเป็น callback function เพื่อรับ instance กลับมา



ตัวอย่าง Refs: Focus

ดูได้ใน ref-focus.html



```
handleClick = () => {  
  this.textInput.focus();  
}  
  
render() {  
  return (  
    <div>  
      <input ref={(input) => this.textInput = input}/>  
      <button onClick={this.handleClick}>set input Focus</button>  
    </div>  
  );  
}
```


ตัวอย่าง Refs: Value

ดูได้ใน ref-value.html



```
handleClick = () => {  
  console.log(this.textInput.value);  
}  
  
render() {  
  return (  
    <div>  
      <input ref={(input) => this.textInput = input}/>  
      <button onClick={this.handleClick}>get value</button>  
    </div>  
  );  
}
```

ตัวอย่าง Refs: Component

ดูได้ใน ref-component.html

```
class User extends React.Component {  
  showAlert = (value) => {  
    alert(`show ${value} from function in user Component`);  
  }  
  render() {  
    return (<div>User Component</div>);  
  }  
}
```

ตัวอย่าง Refs: Component

ดูได้ใน ref-component.html

```
class App extends React.Component {  
  handleClick = () => {  
    this.user.showAlert('Secret');  
  }  
  render() {  
    return (  
      <div>  
        <User ref={ (user) => this.user = user} />  
        <button onClick={this.handleClick}>Show Alert</button>  
      </div>  
    )  
  }  
}
```

createRef

มีใน React \geq 16.3



createRef

ในอดีต React มีวิธีใช้ refs อยู่ 2 แบบคือ string ref และ callback ref แต่ว่า string ref นั้นถึงจะใช้ง่ายแต่ก็มีข้อด้อยอยู่หลายสาเหตุ คนจึงนิยมใช้ callback ref มากกว่า

แต่ใน React 16.3 ได้มีการเพิ่ม createRef เข้ามาเพื่อกำจัดข้อด้อยของ string ref และคงความใช้ง่ายเอาไว้

*ดังนั้นในอนาคต string ref จะถูกเอาออก



createRef

เราจะสร้าง property ของ class แล้วเรียกใช้ `React.createRef()` ก่อนเพื่อเก็บ refs หลังจากนั้นเราจะใส่ refs ลงไปใน props ที่ชื่อ ref ของ element / component

หลังจากนั้นเราจะเข้าถึง refs ได้จาก `this.[ชื่อ property].current`



ตัวอย่าง createRef: Focus

ดูได้ใน `createref-focus.html`



```
textInput = React.createRef();  
handleClick = () => {  
  this.textInput.current.focus();  
}  
render() {  
  return (  
    <div>  
      <input ref={this.textInput}/>  
      <button onClick={this.handleClick}>set input Focus</button>  
    </div>  
  );  
}
```

ตัวอย่าง createRef: Value

ดูได้ใน createref-value.html



```
textInput = React.createRef();  
handleClick = () => {  
  console.log(this.textInput.current.value);  
}  
render() {  
  return (  
    <div>  
      <input ref={this.textInput}/>  
      <button onClick={this.handleClick}>get value</button>  
    </div>  
  );  
}
```


ตัวอย่าง createRef: Component

ดูได้ใน createref-component.html

```
class User extends React.Component {  
  showAlert = (value) => {  
    alert(`show ${value} from function in user Component`);  
  }  
  render() {  
    return (<div>User Component</div>);  
  }  
}
```

ตัวอย่าง createRef: Component

ดูได้ใน createref-component.html



```
class App extends React.Component {  
  user = React.createRef();  
  handleClick = () => {  
    this.user.current.showAlert('Secret');  
  }  
  render() {  
    return (  
      <div>  
        <User ref={this.user} />  
        <button onClick={this.handleClick}>Show Alert</button>  
      </div>  
    )  
  }  
}
```

ข้อควรระวังเรื่อง Refs

การใช้ Refs เป็นการออกจาก data flow ของ react (parent จะติดต่อกับ children ผ่าน props) เราจึงควรใช้ refs เมื่อจำเป็นเท่านั้น

Refs ใช้กับ component ที่สร้างจาก function ไม่ได้เนื่องจาก มันไม่มี Instance



FlashBack

```
▼ {$$typeof: Symbol(react.element), type: "div", key: "1", ref: f, props: {...}, ...}  
  $$typeof: Symbol(react.element)  
  key: "1"  
  ► props: {firstName: "John", lastName: "Doe", children: "Hello React!"}  
  ► ref: f ref(div)  
    type: "div"  
    _owner: null  
  ► __proto__: Object
```


Lab 2: Refs

เปิดไฟล์ lab2.html ขึ้นมาแล้วลองเล่น ref ดูครับ

- หัดเขียน refs ด้วยวิธี callback และ createRef
- หัดใช้ refs กำหนด focus, รับ value
- หัดใช้ refs กับ component
- console.log ดูว่า refs มีหน้าตาอย่างไรทำอะไรได้บ้าง



Higher-Order Components

HOC



Higher-Order Components

Higher-Order Component หรือย่อว่า HOC เป็นเทคนิคขั้นสูงใน React ใช้เพื่อ reuse logic ของ component

ให้คิดง่ายๆว่า HOC เป็นแค่ Pattern ที่เรียกใช้ **Function** รับ Component เข้าไปแล้ว return **Component** อันใหม่ออกมา

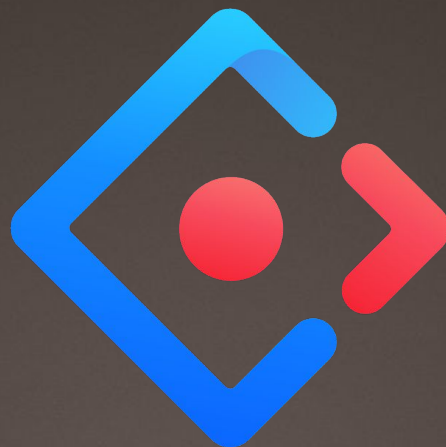
ส่วนใหญ่มักจะเห็น HOC กันบ่อยๆเวลาใช้ third Party Library เช่น Redux Recompose



ตัวอย่าง HOC



```
const OptimizedComponent = pure(ExpensiveComponent);  
  
const NewTodoList = connect()(TodoList);  
  
const NewButtons = withStyles(styles)(FlatButtons);
```



Ant Design



Ant Design

คือ Open source React UI library ที่รวม Component สำเร็จรูปไว้ เยอะ และมี star ใน github สูง

Homepage: <https://ant.design/>



การใช้งานกับ create-react-app



เปิด Terminal ใน โฟลเดอร์ Project แล้วพิมพ์

```
npm install --save antd
```

หรือ

```
yarn add antd
```

ดูฉบับเต็มได้ที่

<https://ant.design/docs/react/use-with-create-react-app>

การใช้งานเบื้องต้น

ให้เพิ่ม `@import '~antd/dist/antd.css';` ด้านบนสุดในไฟล์ global css ที่เราเรียกใช้งาน





```
import React, { Component } from 'react';
import Button from 'antd/lib/button';
import './App.css';

class App extends Component {
  render() {
    return (
      <div className="App">
        <Button type="primary">Button</Button>
      </div>
    );
  }
}
```

Grid System ใน Ant Design

ระบบ Grid ใน Ant Design จะแบ่งออกเป็น 24 ส่วน

ใช้ component `<Row>` ในการแบ่งแถว สามารถกำหนด gutter ได้ เช่น

`<Row gutter={8}>`

ใช้ component `<Col>` ในการแบ่งส่วน สามารถกำหนดส่วนได้ด้วยการส่ง props `span` เช่น `<Col span={12}>`

รองรับ Flex

รองรับ Responsive 5 ขนาดได้แก่ xs sm md lg xl

ดูเพิ่มเติมได้ที่ <https://ant.design/components/grid/>



Recharts



Recharts

คือ Open source charting library สร้างมาสำหรับ React ข้อดีคือมี star ใน github สูงและมีรูปแบบ chart ให้เลือกเยอะ โดยเราสามารถเลือก import component ที่ต้องการจาก library เข้ามาใช้ได้เลย

Homepage: <http://recharts.org>



การใช้งานกับ create-react-app



เปิด Terminal ใน โฟลเดอร์ Project แล้วพิมพ์

```
npm install --save recharts
```

หรือ

```
yarn add recharts
```



```
import { LineChart, Line, CartesianGrid, XAxis, YAxis, Tooltip, Legend } from
'recharts';

class App extends Component {
  state = {
    data: [
      { name: 'Page A', value: 4000, value2: 2400 },
      { name: 'Page B', value: 3000, value2: 1398 },
      { name: 'Page C', value: 2000, value2: 9800 },
      { name: 'Page D', value: 2780, value2: 3908 },
      { name: 'Page E', value: 1890, value2: 4800 },
      { name: 'Page F', value: 2390, value2: 3800 },
      { name: 'Page G', value: 3490, value2: 4300 },
    ]
  }
}
```



```
render() {  
  return (  
    <LineChart width={730} height={250} data={this.state.data}  
      margin={{ top: 5, right: 30, left: 20, bottom: 5 }}>  
      <CartesianGrid strokeDasharray="3 3" />  
      <XAxis dataKey="name" />  
      <YAxis />  
      <Tooltip />  
      <Legend />  
      <Line type="monotone" dataKey="value" stroke="#8884d8" />  
      <Line type="monotone" dataKey="value2" stroke="#82ca9d" />  
    </LineChart>  
  );  
}
```

การบ้าน #1 - สร้าง App ดูราคา Bitcoin

- ใช้ Create React App ในการสร้าง Project
- สร้าง App ดูราคา Bitcoin โดยต้องมี **Feature** ดังนี้
- ใช้ **Ant Design** เป็น UI library
- ตอนโหลด app ขึ้นมาจะมีการ **fetch** ราคา bitcoin จาก coindesk แล้วแสดงผล (ค่าเริ่มต้นคือ USD)
- มี **option** ให้เลือกสกุลเงิน USD GBP EUR
- ระหว่างที่โหลดข้อมูลตอนเริ่มต้นต้องมี **icon spinner** บอกว่ากำลังโหลดข้อมูลอยู่
- **automatic** โหลดข้อมูลใหม่ทุกๆ 5 วินาที

API <https://api.coindesk.com/v1/bpi/currentprice.json>



การบ้าน #2 - สร้าง App ดูราคา Bitcoin ภาค 2

- จากการบ้านข้อที่ 1
- เพิ่ม Feature ดังนี้
- ให้ใช้ **Recharts** เพื่อแสดง chart ราคา bitcoin ย้อนหลังตามสกุลเงินที่เลือก
- ระหว่างโหลด data ใหม่ต้องมี **spinner** บน chart

API <https://api.coindesk.com/v1/bpi/historical/close.json?currency=สกุล>



การบ้าน #2 - สร้าง App ดูราคา Bitcoin ภาค 2



ดูภาพขนาดใหญ่ได้ที่ <https://ibb.co/hRfBvS>

การบ้าน #3 - Ant Design Form

- ใช้ Create React App ในการสร้าง Project
- ใช้ **Ant Design** เป็น UI library
- สร้างหน้า register user ขึ้นมาโดยกำหนดให้ Form มีสิ่งที่ต้องการดังนี้
- email, password, confirm password, phone number, ปุ่ม submit
- พร้อมกับ feature **validation**
- email ต้องตรวจ **pattern** ด้วยว่าใช้ email จริงไหม
- password ต้องมากกว่า 6 ตัวอักษรขึ้นไป
- ตรวจสอบ **confirm password** ว่า ตรงกับ password ไหม
- phone number จะต้องกรอกด้วย **pattern** 089112222 เท่านั้น ห้ามมี - หรือ ตัวหนังสือ
- หาก validate ผ่านให้ **console.log** ค่าของ Form ออกมา



สำหรับหาความรู้เพิ่มเติม

Update on Async Rendering

<https://reactjs.org/blog/2018/03/27/update-on-async-rendering.html>

componentDidCatch Lifecycle Method

<https://medium.com/@sgroff04/2-minutes-to-learn-react-16s-componentdidcatch-lifecycle-method-d1a69a1f753>

Understanding React—Component life-cycle

<https://medium.com/@baphemot/understanding-reactjs-component-life-cycle-823a640b3e8d>

