

การใช้ Array กับ JSX

CHAPTER 1

การ Render ข้อมูล Array ใน JSX

การ Render ข้อมูล Array ใน JSX

หัวข้อที่จะได้เรียนใน Chapter นี้

- ทบทวนการใช้งานฟังก์ชัน Map
- ตัวอย่างการใช้งานฟังก์ชัน Map
- การ Render ข้อมูล Array ใน JSX
- ตัวอย่างการ Render ข้อมูล Array ใน JSX
- การเก็บ Array เป็น State

การ Render ข้อมูล Array ใน JSX

ก่อนหน้านี้เวลาเราต้องการจะใส่ StudentItem 1 อันเข้าไปในหน้าเว็บ เราก็ใส่ `<StudentItem />` เข้าไป 1 อัน

```
<div className="App">
  <NewStudentItem onAddStudent={addStudentHandler} />
  <StudentItem name={studentList[0].name} surname={studentList[0].surname} age={studentList[0].age} />
  <StudentItem name={studentList[1].name} surname={studentList[1].surname} age={studentList[1].age} />
  <StudentItem name={studentList[2].name} surname={studentList[2].surname} age={studentList[2].age} />
  <StudentItem name={studentList[3].name} surname={studentList[3].surname} age={studentList[3].age} />
  <h3>Status: {status}</h3>
  <button onClick={clickEventHandler}>Click me</button>
</div>
```

การ Render ข้อมูล Array ใน JSX

แต่การทำแบบนี้เป็นการทำแบบ Hardcode ซึ่ง เพราะว่าถ้ามี Student เพิ่มเข้ามาใน StudentList อีก 1 อัน เราก็ต้องมานั่งเพิ่มใน JSX เอง ทำให้ Code มันไม่ Dynamic

```
<div className="App">
  <NewStudentItem onAddStudent={addStudentHandler} />
  <StudentItem name={studentList[0].name} surname={studentList[0].surname} age={studentList[0].age}/>
  <StudentItem name={studentList[1].name} surname={studentList[1].surname} age={studentList[1].age}/>
  <StudentItem name={studentList[2].name} surname={studentList[2].surname} age={studentList[2].age}/>
  <StudentItem name={studentList[3].name} surname={studentList[3].surname} age={studentList[3].age}/>
  <h3>Status: {status}</h3>
  <button onClick={clickEventHandler}>Click me</button>
</div>
```

การ Render ข้อมูล Array ใน JSX

studentItem ไม่ได้เพิ่มเองอัตโนมัติตามจำนวนสมาชิกใน `StudentList`

```
<div className="App">
  <NewStudentItem onAddStudent={addStudentHandler} />
  <StudentItem name={studentList[0].name} surname={studentList[0].surname} age={studentList[0].age}/>
  <StudentItem name={studentList[1].name} surname={studentList[1].surname} age={studentList[1].age}/>
  <StudentItem name={studentList[2].name} surname={studentList[2].surname} age={studentList[2].age}/>
  <StudentItem name={studentList[3].name} surname={studentList[3].surname} age={studentList[3].age}/>
  <h3>Status: {status}</h3>
  <button onClick={clickEventHandler}>Click me</button>
</div>
```

การ Render ข้อมูล Array ใน JSX

เราจะใช้ฟังก์ชันของ Array เข้ามาช่วย

```
<div className="App">
  <NewStudentItem onAddStudent={addStudentHandler} />
  <StudentItem name={studentList[0].name} surname={studentList[0].surname} age={studentList[0].age}/>
  <StudentItem name={studentList[1].name} surname={studentList[1].surname} age={studentList[1].age}/>
  <StudentItem name={studentList[2].name} surname={studentList[2].surname} age={studentList[2].age}/>
  <StudentItem name={studentList[3].name} surname={studentList[3].surname} age={studentList[3].age}/>
  <h3>Status: {status}</h3>
  <button onClick={clickEventHandler}>Click me</button>
</div>
```

ทบทวนการใช้งานฟังก์ชัน Map

แต่ก่อนอื่นต้องขอทบทวนฟังก์ชันของ Array ก่อน

- Map
- Filter
- Find
- FindIndex

ฟังก์ชันของ Array นั้นมีหลายอัน แต่ที่เราจะใช้ในการ Render ข้อมูล Array ก็คือฟังก์ชัน Map

ทบทวนการใช้งานฟังก์ชัน Map

แต่ก่อนอื่นต้องขอทบทวนฟังก์ชันของ Array ก่อน

- Map
- Filter
- Find
- FindIndex

ฟังก์ชันของ Array นั้นมีหลายอัน แต่ที่เราจะใช้ในการ Render ข้อมูล Array ก็คือฟังก์ชัน Map



บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1

```
numberArray = [ 1, 2, 3, 4, 5 ];
```



บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งที่ต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

2





บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

2

บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งที่ต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

2


บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

2 4



บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งที่ต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

2 4

บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

2 4

บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งที่ต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

2 4 6



บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งที่ต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

2 4 6

บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

2 4 6

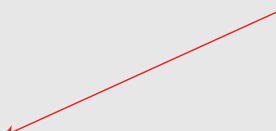
บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งที่ต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

2 4 6 8



ทบทวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

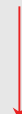
```
mapResult = numberArray.map(e => e*2)
```

2 4 6 8

ทบทวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งที่ต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1

```
numberArray = [ 1, 2, 3, 4, 5 ];
```



```
mapResult = numberArray.map(e => e*2)
```

2 4 6 8

ทบทวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

2 4 6 8 10



บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งที่ต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

```
2      4      6      8      10
```

บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งที่ต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

```
[ 2 , 4 , 6 , 8 , 10 ]
```

บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งที่ต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

```
mapResult = [ 2, 4, 6, 8, 10 ]
```

บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งที่ต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1 (สรุป)

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

```
mapResult = [ 2, 4, 6, 8, 10 ]
```

บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งที่ต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1 (สรุป)

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

```
mapResult = [ 2, 4, 6, 8, 10 ]
```

บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งที่ต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1 (สรุป)

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

```
mapResult = [ 2, 4, 6, 8, 10 ]
```

บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1 (สรุป)

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

```
mapResult = [ 2, 4, 6, 8, 10 ]
```


บททวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง (Transform) สมาชิกใน Array ให้ออกเป็นสิ่งที่ต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 1 (สรุป) - ตัวเลขแปลงเป็นตัวเลข

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map(e => e*2)
```

```
mapResult = [ 2, 4, 6, 8, 10 ]
```

บทวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง(Transform)สมาชิกใน Array ให้ออกเป็นสิ่งต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 2 - ลองหยุดวิดีโอแล้วลองทำดูก่อนนะครับ ว่า mapResult จะออกมาเป็นอะไร

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map((e) => "Number " + e);
```

บทวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง(Transform)สมาชิกใน Array ให้ออกเป็นสิ่งต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 2 (สรุป)

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map((e) => "Number " + e);
```

```
mapResult = [ "Number 1", "Number 2", "Number 3", "Number 4", "Number 5" ];
```

บทวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง(Transform)สมาชิกใน Array ให้ออกเป็นสิ่งต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 2 (สรุป)

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map((e) => "Number " + e);
```

```
mapResult = [ "Number 1", "Number 2", "Number 3", "Number 4", "Number 5" ];
```

บทวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง(Transform)สมาชิกใน Array ให้ออกเป็นสิ่งต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 2 (สรุป)

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map((e) => "Number " + e);
```

```
mapResult = [ "Number 1", "Number 2", "Number 3", "Number 4", "Number 5" ];
```

บทวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง(Transform)สมาชิกใน Array ให้ออกเป็นสิ่งต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 2 (สรุป)

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map((e) => "Number " + e);
```

```
mapResult = [ "Number 1", "Number 2", "Number 3", "Number 4", "Number 5" ];
```

บทวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง(Transform)สมาชิกใน Array ให้ออกเป็นสิ่งต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 2 (สรุป)

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map((e) => "Number " + e);
```

```
mapResult = [ "Number 1", "Number 2", "Number 3", "Number 4", "Number 5" ];
```

บทวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง(Transform)สมาชิกใน Array ให้ออกเป็นสิ่งต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 2 (สรุป)

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map((e) => "Number " + e);
```

```
mapResult = [ "Number 1", "Number 2", "Number 3", "Number 4", "Number 5" ];
```


บทวนการใช้งานฟังก์ชัน Map

ฟังก์ชัน Map คือการแปลง(Transform)สมาชิกใน Array ให้ออกเป็นสิ่งต่าง ๆ ที่เราต้องการ เช่น ตัวอย่างที่ 2 (สรุป) - การแปลงตัวเลขเป็นสตริง

```
numberArray = [ 1, 2, 3, 4, 5 ];
```

```
mapResult = numberArray.map((e) => "Number " + e);
```

```
mapResult = [ "Number 1", "Number 2", "Number 3", "Number 4", "Number 5" ];
```

ทบทวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3 ที่เป็นอาเรย์ของ Object ที่มี name กับ age เป็น Properties

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

บททวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3 - ลองทำเองดูก่อนนะครับ

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```

บททวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```


```
mapResult = studentArray.map((e) => e.name);
```

ทบทวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```




บททวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```


```
mapResult = studentArray.map((e) => e.name);
```



บททวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```



```
mapResult = studentArray.map((e) => e.name);
```

ทบทวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```

"A"

ทบทวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```

"A"

ทบทวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```

"A"

ทบทวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```

"A"

ทบทวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```

"A"

ทบทวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```

"A"

"B"

บททวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```

"A"

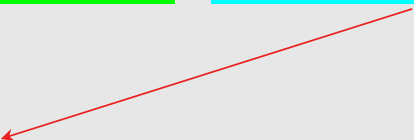
"B"

ทบทวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```



"A"

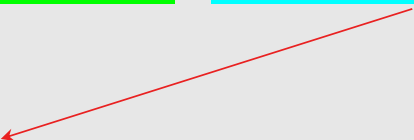
"B"

ทบทวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```



"A"

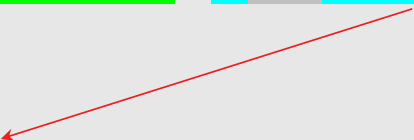
"B"

ทบทวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```



"A"

"B"

ทบทวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```

"A"

"B"

"C"

ทบทวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```

"A"

"B"

"C"

บททวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```

```
[ "A" , "B" , "C" ]
```

บททวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```

```
mapResult = [ "A", "B", "C" ];
```

บททวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3 (สรุป)

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```

```
mapResult = [ "A", "B", "C" ];
```

บททวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3 (สรุป)

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```

```
mapResult = [ "A", "B", "C" ];
```

บททวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3 (สรุป)

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```

```
mapResult = [ "A", "B", "C" ];
```


บททวนการใช้งานฟังก์ชัน Map

แต่ใน Array อาจจะเป็น Object ก็ได้ เช่น ตัวอย่างที่ 3 (สรุป) - การแปลง Object เป็นสตริง (name)

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => e.name);
```

```
mapResult = [ "A", "B", "C" ];
```

การ Render ข้อมูล Array ใน JSX

Mini workshop: นกทวนฟังก์ชัน Map

Note: Lab-j3-l1

ตัวอย่างการ Render ข้อมูล Array ใน JSX

เวลาเราจะ Render ข้อมูล Array เป็น JSX เราก็จะใช้ฟังก์ชัน Map ในการแปลงสมาชิกแต่ละตัวให้เป็น JSX เนี่ยแหละ

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4 - ลองทำดูก่อนนะครับ

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```



```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```



```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```



```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```


ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
<div>A</div>
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
<div>A</div>
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```



```
<div>A</div>
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
<div>A</div>
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
<div>A</div>
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
<div>A</div> <div>B</div>
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
<div>A</div>    <div>B</div>
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```



```
<div>A</div>    <div>B</div>
```


ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```



```
<div>A</div>    <div>B</div>
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```



```
<div>A</div>    <div>B</div>
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
<div>A</div>    <div>B</div>    <div>C</div>
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
[ <div>A</div> , <div>B</div> , <div>C</div> ]
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4 - สรุป

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
[ <div>A</div> , <div>B</div> , <div>C</div> ]
```

ที่เราต้องใส่ปีกกาเพราะว่า
e.name เป็นตัวแปร (variable)

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4 - สรุป

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
mapResult = [ <div>A</div>, <div>B</div>, <div>C</div> ];
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4 - สรุป

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
mapResult = [ <div>A</div>, <div>B</div>, <div>C</div> ];
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 4 - สรุป

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
mapResult = [ <div>A</div>, <div>B</div>, <div>C</div> ];
```


ตัวอย่างการ Render ข้อมูล Array ใน JSX

ที่นี้เดี๋ยวเรามาดูตัวอย่างการใช้ใน JSX กันนะครับ
โดยเราจะใช้ array และ map function จาก ตัวอย่างที่ 4 นะครับ

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
mapResult = [ <div>A</div>, <div>B</div>, <div>C</div> ];
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 5

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];  
  
mapResult = studentArray.map((e) => <div>{e.name}</div>);  
  
mapResult = [ <div>A</div>, <div>B</div>, <div>C</div> ];
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 5

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];  
  
mapResult = studentArray.map((e) => <div>{e.name}</div>);  
  
mapResult = [ <div>A</div>, <div>B</div>, <div>C</div> ];
```

React

```
function Test() {  
  const studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];  
  return (  
    <div>  
      <h1>Hello React</h1>  
      {studentArray.map(e => <div>{e.name}</div>)}  
      <h2>Hello JS</h2>  
    </div>  
  );  
}
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 5

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
mapResult = [ <div>A</div>, <div>B</div>, <div>C</div> ];
```

React

```
function Test() {  
  const studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];  
  return (  
    <div>  
      <h1>Hello React</h1>  
      {studentArray.map(e => <div>{e.name}</div>)}  
      <h2>Hello JS</h2>  
    </div>  
  );  
}
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 5

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];  
  
mapResult = studentArray.map((e) => <div>{e.name}</div>);  
  
mapResult = [ <div>A</div>, <div>B</div>, <div>C</div> ];
```

React

```
function Test() {  
  const studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];  
  return (  
    <div>  
      <h1>Hello React</h1>  
      {studentArray.map(e => <div>{e.name}</div>)}  
      <h2>Hello JS</h2>  
    </div>  
  );  
}
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 5

หลังจากการรัน Code แล้ว ตัว **สีเหลือง** จะกลายเป็นสีเขียว

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];  
  
mapResult = studentArray.map((e) => <div>{e.name}</div>);  
  
mapResult = [ <div>A</div>, <div>B</div>, <div>C</div> ];
```

React

```
function Test() {  
  const studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];  
  return (  
    <div>  
      <h1>Hello React</h1>  
      {studentArray.map(e => <div>{e.name}</div>)}  
      <h2>Hello JS</h2>  
    </div>  
  );  
}
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 5

หลังจากการรัน Code แล้ว ตัว **สีเหลือง** จะกลายเป็น **สีเขียว**

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];  
  
mapResult = studentArray.map((e) => <div>{e.name}</div>);  
  
mapResult = [ <div>A</div>, <div>B</div>, <div>C</div> ];
```

React

```
function Test() {  
  const studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];  
  return (  
    <div>  
      <h1>Hello React</h1>  
      {studentArray.map(e => <div>{e.name}</div>)}  
      <h2>Hello JS</h2>  
    </div>  
  );  
}
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 5

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
mapResult = [ <div>A</div>, <div>B</div>, <div>C</div> ];
```

React

แปลว่าตัวนี้จะกลายเป็นสี่เหลี่ยมด้วย

```
function Test() {  
  const studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];  
  return (  
    <div>  
      <h1>Hello React</h1>  
      {studentArray.map(e => <div>{e.name}</div>)}  
      <h2>Hello JS</h2>  
    </div>  
  );  
}
```


ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 5

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
mapResult = [ <div>A</div>, <div>B</div>, <div>C</div> ];
```

React

แปลว่าตัวนี้จะกลายเป็นสี่เหลี่ยมด้วย

```
function Test() {  
  const studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];  
  return (  
    <div>  
      <h1>Hello React</h1>  
      {[ <div>A</div>, <div>B</div>, <div>C</div> ]}  
      <h2>Hello JS</h2>  
    </div>  
  );  
}
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 5

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
mapResult = [ <div>A</div>, <div>B</div>, <div>C</div> ];
```

React

แต่หลายคนอาจจะสงสัยว่าเอ เราใส่เป็น Array ของ Element แบบนี้ได้ด้วยหรอ

```
function Test() {  
  const studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];  
  return (  
    <div>  
      <h1>Hello React</h1>  
      {[ <div>A</div>, <div>B</div>, <div>C</div> ]}  
      <h2>Hello JS</h2>  
    </div>  
  );  
}
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 5

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
mapResult = [ <div>A</div>, <div>B</div>, <div>C</div> ];
```

React

คำตอบคือได้ครับ

```
function Test() {  
  const studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];  
  return (  
    <div>  
      <h1>Hello React</h1>  
      {studentArray.map(e => <div>{e.name}</div>)}  
      <h2>Hello JS</h2>  
    </div>  
  );  
}
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 5

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
mapResult = [ <div>A</div>, <div>B</div>, <div>C</div> ];
```

React

สุดท้ายแล้วมันจะกลายเป็นแบบนี้ครับ จาก Array ก็กลายเป็นที่เรียงติดกันเลย

```
<div>
  <h1>Hello React</h1>
  {[ <div>A</div>, <div>B</div>, <div>C</div> ]}
  <h2>Hello JS</h2>
</div>
```

```
<div>
  <h1>Hello React</h1>
  <div>A</div>
  <div>B</div>
  <div>C</div>
  <h2>Hello JS</h2>
</div>
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 5

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
mapResult = [ <div>A</div>, <div>B</div>, <div>C</div> ];
```

React

```
<div>
  <h1>Hello React</h1>
  {[ <div>A</div>, <div>B</div>, <div>C</div> ]}
  <h2>Hello JS</h2>
</div>
```

```
<div>
  <h1>Hello React</h1>
  <div>A</div>
  <div>B</div>
  <div>C</div>
  <h2>Hello JS</h2>
</div>
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 5

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
mapResult = [ <div>A</div>, <div>B</div>, <div>C</div> ];
```

React

```
<div>  
  <h1>Hello React</h1>  
  { [ <div>A</div>, <div>B</div>, <div>C</div> ] }  
  <h2>Hello JS</h2>  
</div>
```

```
<div>  
  <h1>Hello React</h1>  
  <div>A</div>  
  <div>B</div>  
  <div>C</div>  
  <h2>Hello JS</h2>  
</div>
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 5

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
mapResult = [ <div>A</div>, <div>B</div>, <div>C</div> ];
```

React

```
<div>  
  <h1>Hello React</h1>  
  {[ <div>A</div>, <div>B</div>, <div>C</div> ]}  
  <h2>Hello JS</h2>  
</div>
```

```
<div>  
  <h1>Hello React</h1>  
  <div>A</div>  
  <div>B</div>  
  <div>C</div>  
  <h2>Hello JS</h2>  
</div>
```

ตัวอย่างการ Render ข้อมูล Array ใน JSX

ตัวอย่างที่ 5

```
studentArray = [ { name: "A", age: 16 }, { name: "B", age: 17 }, { name: "C", age: 15 } ];
```

```
mapResult = studentArray.map((e) => <div>{e.name}</div>);
```

```
mapResult = [ <div>A</div>, <div>B</div>, <div>C</div> ];
```

React

!!แต่การทำแบบนี้จะมีเรื่อง Key ที่เราต้องเรียนรู้เพิ่มเติมด้วยครับ เดี๋ยวเราจะมาพูดถึงใน Chapter ถัดไปนะครับ

```
<div>
  <h1>Hello React</h1>
  {[ <div>A</div>, <div>B</div>, <div>C</div> ]}
  <h2>Hello JS</h2>
</div>
```

```
<div>
  <h1>Hello React</h1>
  <div>A</div>
  <div>B</div>
  <div>C</div>
  <h2>Hello JS</h2>
</div>
```


การ Render ข้อมูล Array ใน JSX

Mini workshop: การ Render ข้อมูล Array ใน JSX

โค้ดเริ่มต้น: <https://github.com/soncomqiq/ep4-initial-code>

Note: Lab-r3-l1

การ Render ข้อมูล Array ใน JSX

Mini workshop: การใช้ Array เป็นแบบ State

Note: Lab-r3-l2



CHAPTER 2

"Keys"

สำหรับการ Render Array ใน JSX

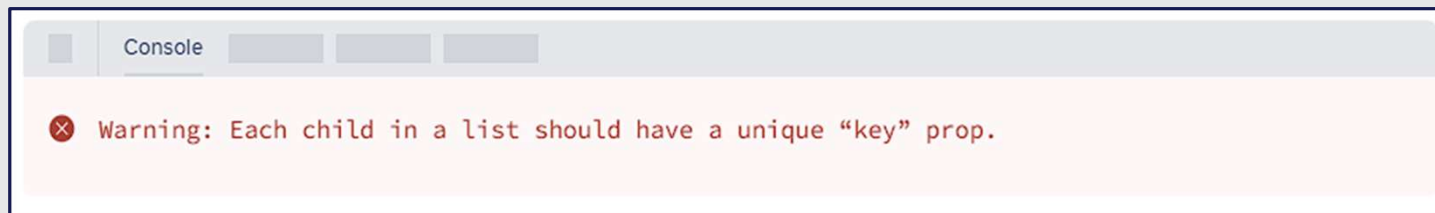
“Key” สำหรับการ Render Array ใน JSX

หัวข้อที่จะได้เรียนใน Chapter นี้

- ทบทวน Virtual DOM
- key สำคัญอย่างไร
- ตัวอย่างการใช้งาน key
- กฎของ key
- ตัวอย่างการใช้งานคู่กับ Array ใน JSX

“Key” คืออะไร

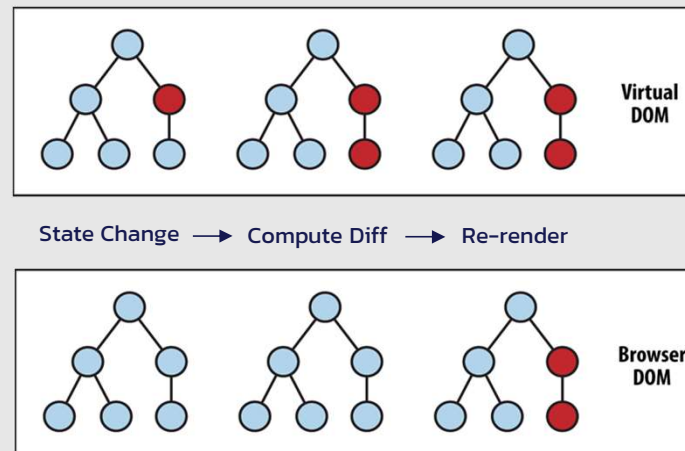
เราอาจจะเห็นข้อความนี้เวลากด Developer Tools ขึ้นมา



ข้อความนี้คืออะไร?

บทวน Virtual DOM

ต้องย้อนกลับมาที่ Concept การ Render ของ React ก่อน ว่าปกติแล้ว React จะใช้ Virtual DOM ในการ Track การเปลี่ยนแปลง แล้วนำมาเทียบว่าจุดไหนบ้างที่ถูกเปลี่ยนแปลง และค่อยไปเปลี่ยนที่ Real DOM เฉพาะจุดเหล่านั้น



ที่มา: https://i2.wp.com/programmingwithmosh.com/wp-content/uploads/2018/11/lnrn_0201.png?fit=1173%2C785&ssl=1

key สำคัญอย่างไร

ซึ่งการที่เราจะ Render Array ใน JSX แบบนี้

```
<div>
  <h1>Hello React</h1>
  {[ <div>A</div>, <div>B</div>, <div>C</div> ]}
  <h2>Hello JS</h2>
</div>
```

(ทวนเพื่อลื้ม) ผลลัพธ์สุดท้ายอันนี้
ก็ออกมาเป็น Array

```
<div>
  <h1>Hello React</h1>
  {studentArray.map((e) => <div>{e.name}</div>)}
  <h2>Hello JS</h2>
</div>
```

key สำคัญอย่างไร

ซึ่งการที่เราจะ Render Array ใน JSX แบบนี้

```
<div>
  <h1>Hello React</h1>
  {[ <div>A</div>, <div>B</div>, <div>C</div> ]}
  <h2>Hello JS</h2>
</div>
```

```
<div>
  <h1>Hello React</h1>
  {studentArray.map((e) => <div>{e.name}</div>)}
  <h2>Hello JS</h2>
</div>
```


key สำคัญอย่างไร

ตัว React จะไม่สามารถแยกความแตกต่างระหว่างสมาชิกใน Array ได้

```
<div>
  <h1>Hello React</h1>
  {[ <div>A</div>, <div>B</div>, <div>C</div> ]}
  <h2>Hello JS</h2>
</div>
```

```
<div>
  <h1>Hello React</h1>
  {studentArray.map((e) => <div>{e.name}</div>)}
  <h2>Hello JS</h2>
</div>
```

key สำคัญอย่างไร

ตัว React จะไม่สามารถแยกความแตกต่างระหว่างสมาชิกใน Array ได้

```
<div>
  <h1>Hello React</h1>
  {[ <div>A</div>, <div>B</div>, <div>C</div> ]}
  <h2>Hello JS</h2>
</div>
```

React จะไม่สามารถ
แยกความแตกต่าง
ของสมาชิกทั้งสามตัวได้

```
<div>
  <h1>Hello React</h1>
  {studentArray.map((e) => <div>{e.name}</div>)}
  <h2>Hello JS</h2>
</div>
```

key สำคัญอย่างไร

1. เมื่อเรา มีการ update array (การเพิ่ม การลบ การเรียงสมาชิกใหม่) แต่ React ไม่สามารถแยกความแตกต่าง ระหว่างสมาชิกใน Array ได้
2. ตัว Virtual DOM ก็จะ Track Array ทั้งก่อนว่ามีการเปลี่ยนแปลง
3. และเวลาแก้ไข Real DOM ก็ต้องไปแก้ไขในส่วนของ Array ทั้งก่อน (ซึ่งอาจส่งผลต่อ Performance)

```
<div>
  <h1>Hello React</h1>
  {[ <div>A</div>, <div>B</div>, <div>C</div> ]}
  <h2>Hello JS</h2>
</div>
```

```
<div>
  <h1>Hello React</h1>
  {studentArray.map((e) => <div>{e.name}</div>)}
  <h2>Hello JS</h2>
</div>
```

ตัวอย่างการใช้งาน key

ตัวอย่างที่ 1 เช่น มีการเพิ่ม `<div>X</div>` เข้ามาใน Array

```
<div>
  <h1>Hello React</h1>
  {[ <div>X</div> , <div>A</div>, <div>B</div>, <div>C</div> ]}
  <h2>Hello JS</h2>
</div>
```

ตัวอย่างการใช้งาน key

ตัวอย่างที่ 1 เช่น มีการเพิ่ม `<div>X</div>` เข้ามาใน Array

แต่เนื่อง React ไม่สามารถแยกความแตกต่างระหว่างสมาชิกทั้ง 4 ได้ ตัว Virtual DOM ก็ต้อง Track array ทั้งก้อนว่ามีการเปลี่ยนแปลง

```
<div>
  <h1>Hello React</h1>
  {[ <div>X</div> , <div>A</div>, <div>B</div>, <div>C</div> ]}
  <h2>Hello JS</h2>
</div>
```

ตัวอย่างการใช้งาน key

ตัวอย่างที่ 1 เช่น มีการเพิ่ม `<div>X</div>` เข้ามาใน Array

ซึ่งเวลา React มีการ Re-render ก็ทิ้งไปแก้ไข DOM ในส่วนของสมาชิกทั้งสี่ตัวใน Array นี้ ทำให้มีผลต่อ Performance (ช้า) - ในกรณีที่ array มีสมาชิกเยอะๆ

```
<div>
  <h1>Hello React</h1>
  {[ <div>X</div> , <div>A</div>, <div>B</div>, <div>C</div> ]}
  <h2>Hello JS</h2>
</div>
```

ตัวอย่างการใช้งาน key

วิธีที่จะทำให้ React สามารถแยกความแตกต่างระหว่างสมาชิกใน Array ได้ก็คือการใส่ props ที่ชื่อว่า "key"

```
<div>
  <h1>Hello React</h1>
  {[<div key="x">X</div>, <div key="a">A</div>, <div key="b">B</div>, <div key="c">C</div>]}
  <h2>Hello JS</h2>
</div>
```

ตัวอย่างการใช้งาน key

ซึ่งการทำแบบนี้จะทำให้ React สามารถแยกความแตกต่างระหว่างสมาชิกใน Array ได้

```
<div>
  <h1>Hello React</h1>
  {[<div key="x">X</div>, <div key="a">A</div>, <div key="b">B</div>, <div key="c">C</div>]}
  <h2>Hello JS</h2>
</div>
```


ตัวอย่างการใช้งาน key

ตัวอย่างที่ 2 - เช่น เรามีการสลับระหว่าง X กับ A (การ Re-order) ของ array

```
<div>
  <h1>Hello React</h1>
  {[<div key="x">X</div>, <div key="a">A</div>, <div key="b">B</div>, <div key="c">C</div>]}
  <h2>Hello JS</h2>
</div>
```

ตัวอย่างการใช้งาน key

ตัวอย่างที่ 2 - เช่น เรามีการสลับระหว่าง X กับ A (การ Re-order) ของ array

```
<div>
  <h1>Hello React</h1>
  {[<div key="a">A</div>, <div key="x">X</div>, <div key="b">B</div>, <div key="c">C</div>]}
  <h2>Hello JS</h2>
</div>
```

ตัว React ก็จะ Track เฉพาะสมาชิกที่มี key เป็น "a" และ "x" ใน Virtual DOM

ตัวอย่างการใช้งาน key

ตัวอย่างที่ 2 - เช่น เรามีการสลับระหว่าง X กับ A (การ Re-order) ของ array

```
<div>
  <h1>Hello React</h1>
  {[<div key="a">A</div>, <div key="x">X</div>, <div key="b">B</div>, <div key="c">C</div>]}
  <h2>Hello JS</h2>
</div>
```

หลังจากนั้นเวลาแก้ไขใน Real DOM ก็จะถูกแก้ไขแค่สมาชิก 2 ตัวนี้
(ไม่ได้แก้ไขทั้ง Array) ทำให้ Performance ดีขึ้น

กฎของ key

กฎของ Key ในการ Render มีทั้งหมด 2 ข้อ

1. ต้อง Unique ใน Array เดียวกัน เช่น

```
<div>
  <h1>Hello React</h1>
  {[<div key="a">A</div>, <div key="x">X</div>, <div key="b">B</div>, <div key="c">C</div>]}
  <h2>Hello JS</h2>
  {[<h1 key="a">A</h1>, <h1 key="x">X</h1>, <h1 key="b">B</h1>, <h1 key="c">C</h1>]}
</div>
```

กฎของ key

กฎของ Key ในการ Render มีทั้งหมด 2 ข้อ

1. ต้อง Unique ใน Array เดียวกัน เช่น จะเห็นว่าสมาชิกใน Array แรกจะมี **key** ที่ไม่ซ้ำกันเลย ได้แก่ a, x, b และ c

```
<div>
  <h1>Hello React</h1>
  {[<div key="a">A</div>, <div key="x">X</div>, <div key="b">B</div>, <div key="c">C</div>]}
  <h2>Hello JS</h2>
  {[<h1 key="a">A</h1>, <h1 key="x">X</h1>, <h1 key="b">B</h1>, <h1 key="c">C</h1>]}
</div>
```

กฎของ key

กฎของ Key ในการ Render มีทั้งหมด 2 ข้อ

1. ต้อง Unique ใน Array เดียวกัน เช่น จะเห็นว่าสมาชิกใน Array แรกจะมี **key** ที่ไม่ซ้ำกันเลย ได้แก่ a, x, b และ c

แต่ถ้ามีการเปลี่ยน key ตัวนี้จาก "c" เป็น "b" key ก็จะไม่ unique กันทีเพราะว่าไปซ้ำกับของสมาชิกตัวอื่น

```
<div>
  <h1>Hello React</h1>
  {[<div key="a">A</div>, <div key="x">X</div>, <div key="b">B</div>, <div key="b">C</div>]}
  <h2>Hello JS</h2>
  {[<h1 key="a">A</h1>, <h1 key="x">X</h1>, <h1 key="b">B</h1>, <h1 key="c">C</h1>]}
</div>
```

กฎของ key

กฎของ Key ในการ Render มีทั้งหมด 2 ข้อ

1. ต้อง Unique ใน Array เดียวกัน เช่น จะเห็นว่าสมาชิกใน Array แรกจะมี key ที่ไม่ซ้ำกันเลย ได้แก่ a, x, b และ c ใน Array ที่สอง key ของสมาชิกทั้งหมดก็ Unique ใน Array นั้นเหมือนกัน

```
<div>
  <h1>Hello React</h1>
  {[<div key="a">A</div>, <div key="x">X</div>, <div key="b">B</div>, <div key="c">C</div>]}
  <h2>Hello JS</h2>
  {[<h1 key="a">A</h1>, <h1 key="x">X</h1>, <h1 key="b">B</h1>, <h1 key="c">C</h1>]}
</div>
```

กฎของ key

กฎของ Key ในการ Render มีทั้งหมด 2 ข้อ

1. ต้อง Unique ใน Array เดียวกัน เช่น จะเห็นว่าสมาชิกใน Array แรกจะมี key ที่ไม่ซ้ำกันเลย ได้แก่ a, x, b และ c ใน Array ที่สอง key ของสมาชิกทั้งหมดก็ Unique ใน Array นั้นเหมือนกัน ถึงแม้ Array แรก และ Array ที่สอง จะมี key "a" ที่ซ้ำกันข้าม Array ก็ไม่เป็นไร แต่ขอให้ใน Array เดียวกันไม่ซ้ำกันก็เพียงพอแล้ว

```
<div>
  <h1>Hello React</h1>
  {[<div key="a">A</div>, <div key="x">X</div>, <div key="b">B</div>, <div key="c">C</div>]}
  <h2>Hello JS</h2>
  {[<h1 key="a">A</h1>, <h1 key="x">X</h1>, <h1 key="b">B</h1>, <h1 key="c">C</h1>]}
</div>
```


กฎของ key

กฎของ Key ในการ Render มีทั้งหมด 2 ข้อ

1. ต้อง Unique ใน Array เดียวกัน เช่น จะเห็นว่าสมาชิกใน Array แรกจะมี key ที่ไม่ซ้ำกันเลย ได้แก่ a, x, b และ c ใน Array ที่สอง key ของสมาชิกทั้งหมดก็ Unique ใน Array นั้นเหมือนกัน ถึงแม้ Array แรก และ Array ที่สอง จะมี key "a" ที่ซ้ำกันข้าม Array ก็ไม่เป็นไร แต่ขอให้ใน Array เดียวกันไม่ซ้ำกันก็เพียงพอแล้ว

```
<div>
  <h1>Hello React</h1>
  {[<div key="a">A</div>, <div key="x">X</div>, <div key="b">B</div>, <div key="c">C</div>]}
  <h2>Hello JS</h2>
  {[<h1 key="a">A</h1>, <h1 key="x">X</h1>, <h1 key="b">B</h1>, <h1 key="c">C</h1>]}
</div>
```

กฎของ key

กฎของ Key ในการ Render มีทั้งหมด 2 ข้อ

1. ต้อง Unique ใน Array เดียวกัน
2. ต้องไม่ถูกเปลี่ยนแปลง เช่น

```
<div>
  <h1>Hello React</h1>
  {[<div key="a">A</div>, <div key="x">X</div>, <div key="b">B</div>, <div key="c">C</div>]}
  <h2>Hello JS</h2>
</div>
```

กฎของ key

กฎของ Key ในการ Render มีทั้งหมด 2 ข้อ

1. ต้อง Unique ใน Array เดียวกัน
2. ต้องไม่ถูกเปลี่ยนแปลง เช่น เราใช้ Math.random() ในการ Generate Key

```
<div>
  <h1>Hello React</h1>
  {[<div key={Math.random()}>A</div>, <div key="x">X</div>, <div key="b">B</div>, <div key="c">C</div>]}
  <h2>Hello JS</h2>
</div>
```

กฎของ key

กฎของ Key ในการ Render มีทั้งหมด 2 ข้อ

1. ต้อง Unique ใน Array เดียวกัน
2. ต้องไม่ถูกเปลี่ยนแปลง เช่น เราใช้ Math.random() ในการ Generate Key การทำแบบนี้จะทำให้ Key ของสมาชิกตัวแรกนี้เปลี่ยนทุกครั้งที่มีการ Re-render ทำให้ผิดกฎข้อสอง

```
<div>
  <h1>Hello React</h1>
  {[<div key={Math.random()}>A</div>, <div key="x">X</div>, <div key="b">B</div>, <div key="c">C</div>]}
  <h2>Hello JS</h2>
</div>
```

กฎของ key

กฎของ Key ในการ Render ทั้งหมด 2 ข้อ - สรุป

1. ต้อง Unique ใน Array เดียวกัน
2. ต้องไม่ถูกเปลี่ยนแปลง

key ต้อง Unique และไม่ถูกเปลี่ยนแปลง

ตัวอย่างการใช้งาน key

ตัวอย่างที่ 3 - การใช้กับ StudentList

```
const studentList = [  
  { id: 1, name: "Samuel", surname: "Jackson", age: 73 },  
  { id: 2, name: "Keanu", surname: "Reeves", age: 58 },  
  { id: 3, name: "Tom", surname: "Cruise", age: 60 },  
  { id: 4, name: "Johnny", surname: "Depp", age: 59 },  
];
```

```
<div>  
  {studentList.map(s => <StudentItem key={s.id} name={s.name} surname={s.surname} age={s.age}/>)}  
</div>
```

ตัวอย่างการใช้งาน key

ตัวอย่างที่ 3 - การใช้กับ StudentList

เราจึงใช้ **id** ที่มัน unique และไม่เปลี่ยนแปลงสำหรับ Student คนนั้น ๆ

```
const studentList = [
  { id: 1, name: "Samuel", surname: "Jackson", age: 73 },
  { id: 2, name: "Keanu", surname: "Reeves", age: 58 },
  { id: 3, name: "Tom", surname: "Cruise", age: 60 },
  { id: 4, name: "Johnny", surname: "Depp", age: 59 },
];
```

```
<div>
  {studentList.map(s => <StudentItem key={s.id} name={s.name} surname={s.surname} age={s.age}/>)}
</div>
```

ตัวอย่างการใช้งาน key

ตัวอย่างที่ 3 - การใช้กับ StudentList
เราก็จะได้ผลลัพธ์ออกมาแบบ slide ถัดไปนี้

```
const studentList = [
  { id: 1, name: "Samuel", surname: "Jackson", age: 73 },
  { id: 2, name: "Keanu", surname: "Reeves", age: 58 },
  { id: 3, name: "Tom", surname: "Cruise", age: 60 },
  { id: 4, name: "Johnny", surname: "Depp", age: 59 },
];
```

```
<div>
  {studentList.map(s => <StudentItem key={s.id} name={s.name} surname={s.surname} age={s.age}/>)}
</div>
```


ตัวอย่างการใช้งาน key

ตัวอย่างที่ 3 - การใช้กับ StudentList ผลลัพธ์ออกมาแบบนี้

```
const studentList = [  
  { id: 1, name: "Samuel", surname: "Jackson", age: 73 },  
  { id: 2, name: "Keanu", surname: "Reeves", age: 58 },  
  { id: 3, name: "Tom", surname: "Cruise", age: 60 },  
  { id: 4, name: "Johnny", surname: "Depp", age: 59 },  
];
```

```
<div>  
  <StudentItem key={1} name={"Samuel"} surname={"Jackson"} age={73}/>  
  <StudentItem key={2} name={"Keanu"} surname={"Reeves"} age={58}/>  
  <StudentItem key={3} name={"Tom"} surname={"Cruise"} age={60}/>  
  <StudentItem key={4} name={"Johnny"} surname={"Depp"} age={59}/>  
</div>
```

ลิงค์สำหรับอ่านเพิ่มเติม

สำหรับอ่านเพิ่มเติม

<https://react.dev/learn/rendering-lists>

<https://stackoverflow.com/questions/28329382/understanding-unique-keys-for-array-children-in-react-js/43892905#43892905>

การ Render ข้อมูล Array ใน JSX

Mini workshop: Key สำหรับ Array ใน JSX

Note: Lab-r3-l3

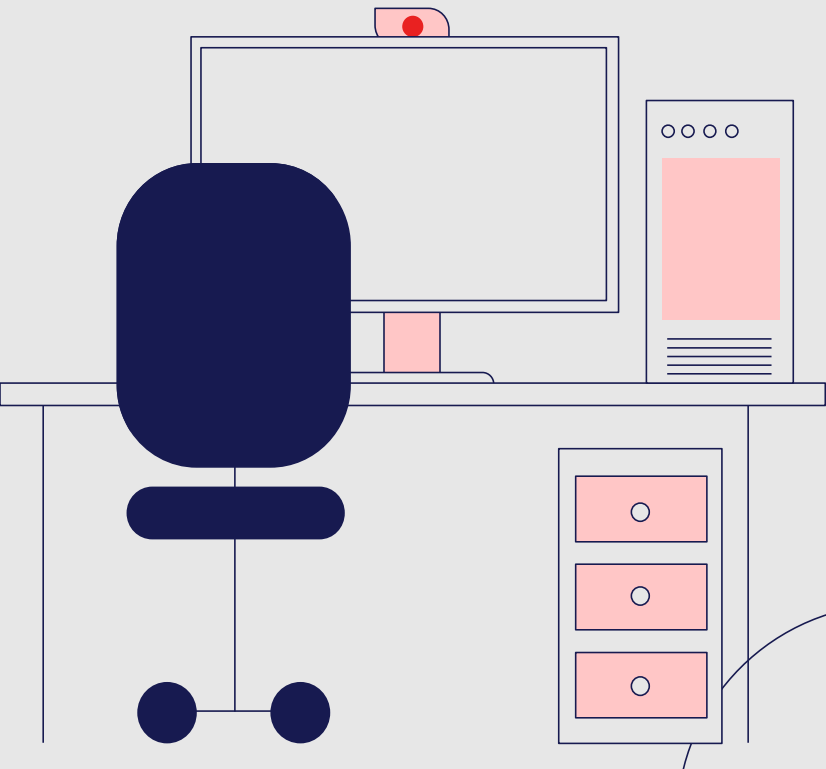
การ Render ข้อมูล Array ใน JSX

Mini workshop: ทำให้ Filter ใช้งานได้

Note: Lab-r3-l4

CHAPTER 3

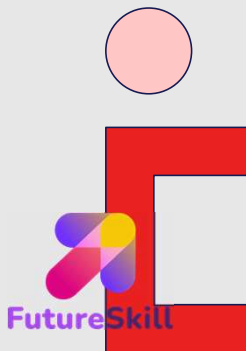
การแสดงผล ตามเงื่อนไข



การแสดงผลตามเงื่อนไข

หัวข้อที่จะเรียนใน Chapter นี้

- การแสดงผลตามเงื่อนไข
 - การใช้ Ternary operator
 - การใช้ && operator
 - การใช้ If-else
 - Return หลายอันใน 1 คอมโพเนนต์
 - CSS Style ตามเงื่อนไข
- ตัวอย่างการแสดงผลตามเงื่อนไข
 - การแสดง/ซ่อน Form
 - การเพิ่มปุ่ม Delete
 - การเพิ่มปุ่ม Edit

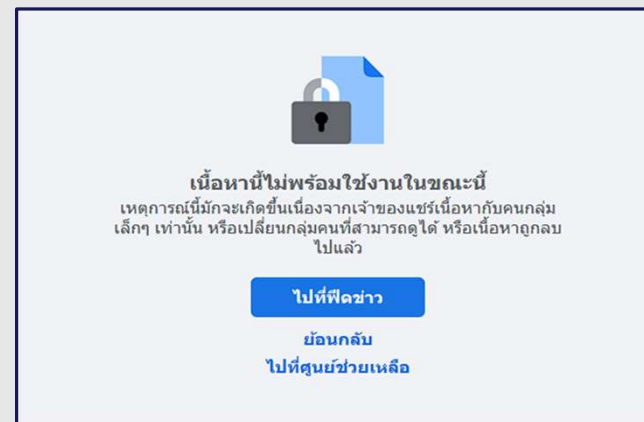


การแสดงผลตามเงื่อนไข

การแสดงผลตามเงื่อนไข คือ การกำหนดให้ Content นั้น Render ที่แตกต่างกันไปตามเงื่อนไข เช่น

การแสดงผล Not Found

- ถ้ามีให้แสดง Content
- ถ้าไม่มีข้อมูลให้แสดงว่า Not Found



การแสดงผลตามเงื่อนไข

การแสดงผลตามเงื่อนไขใน React มีหลายวิธี
แต่ในคอร์สนี้จะสอนคร่าว ๆ 3 วิธี

1. การใช้ Ternary Operator
2. การใช้ && Operator
3. การใช้ If - else

การแสดงผลตามเงื่อนไข

Mini workshop: การแสดงผลตามเงื่อนไขเบื้องต้น

Note: Lab-r3-l5

การแสดงผลตามเงื่อนไข

Mini workshop: การแสดงผลตามเงื่อนไขกับคอมโพเนนต์
NewStudentItem

Note: Lab-r3-l6

การแสดงผลตามเงื่อนไข

Mini workshop: เพิ่มปุ่ม Delete

Note: Lab-r3-l7

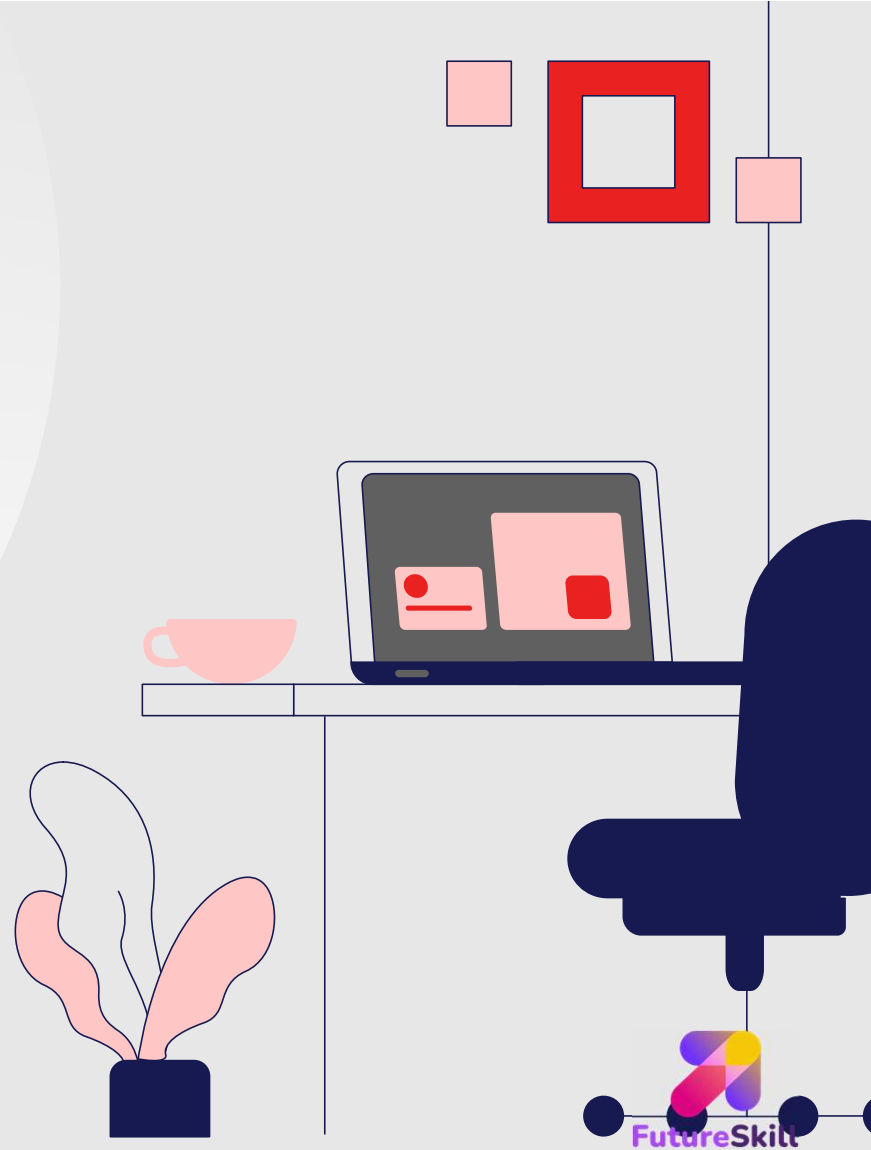
การแสดงผลตามเงื่อนไข

Mini workshop: เพิ่มปุ่ม Edit

Note: Lab-r3-l8

CHAPTER 4

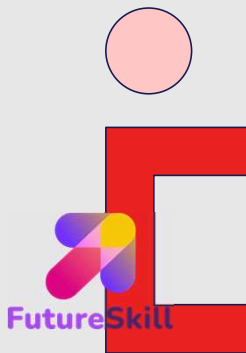
Full workshop



Full Workshop – Todo List

สิ่งที่จะทำเพิ่มใน Todo List

- ทำให้ Todo List มี Filter ตามปีได้
- สามารถซ่อนกล่องเพิ่ม Todo ได้
- แสดงข้อความเมื่อไม่มี Todo
- สามารถลบ Todo ได้
- สามารถแก้ไข Todo ได้



Full Workshop – Todo List

Link: <https://github.com/soncomqiq/workshop-todolist-2023/tree/start-ep-3>