



JavaScript for React

Nuttachai Kulthammanit



บทวนเนื้อหา Basic JavaScript จาก คอร์ส HTML

01

บทวนเนื้อหา Basic JavaScript
จาก คอร์ส HTML

การประกาศตัวแปร (Declaring variables)

var เป็น JavaScript เวอร์ชันเก่า
let และ const มาใน ES6 (เวอร์ชันใหม่)

```
var foo;  
var bar;
```

ES5

```
const foo  
let bar
```

ES6

การประกาศตัวแปร (Declaring variables)

- **const** ใช้กับตัวแปรที่เปลี่ยนค่าไม่ได้ และต้องใส่ค่าเริ่มต้นให้เสมอ
- **var** ใช้กับตัวแปรที่เปลี่ยนค่าได้ (แต่เราจะใช้ **let** แทน **var**)
- **let** ใช้กับตัวแปรที่เปลี่ยนค่าได้ ไม่จำเป็นต้องใส่ค่าเริ่มต้นให้



การประกาศตัวแปร (Declaring variables)



```
// แบบนี้จะเกิด Error ขึ้น  
const foo  
// ถ้าเป็น const ต้องกำหนดค่าเริ่มต้นด้วย  
const foo = 'bar'
```

การประกาศตัวแปร (ตัวอย่าง - const)



การประกาศตัวแปร (Declaring variables)

```
// สามารถประกาศแบบไม่กำหนดค่าเริ่มต้นได้  
let foo  
// สามารถมาใส่ค่าทีหลังได้  
foo = 'bar'  
// หรือจะประกาศและใส่ค่าไปเลยก็ได้  
let message = 'Hello World!'
```

การประกาศตัวแปร (ตัวอย่าง - let)



ประเภทของข้อมูล (Data types)

5 ประเภท

ใน JavaScript
มีข้อมูลที่เป็นชนิดพื้นฐาน

Number

ข้อมูลประเภทตัวเลข

String

ข้อมูลประเภทข้อความ

Boolean

ข้อมูลที่มีแค่ true และ false

null

ค่าว่าง

undefined

ข้อมูลที่ยังไม่ได้ใส่ค่า



JavaScript สำหรับ React

02

ฟังก์ชัน (Function)

ฟังก์ชัน (Function)

- เป็นการเขียน Code ที่สามารถนำโค้ดนั้น ๆ มาใช้ซ้ำได้
- ไว้สำหรับเขียน Code ที่เราต้อง**ใช้บ่อย ๆ**
- มี build-in function (ฟังก์ชันที่เค้าเขียนมาให้) เช่น alert, prompt

```
function ชื่อฟังก์ชัน(parameters) {  
    // body  
}
```

ฟังก์ชัน (Function)

- วิธีเรียกใช้ function
- เขียนชื่อและตามด้วยวงเล็บเปิดและปิด ใส่ parameters ด้วย (ถ้ามี)

```
function showMessage() {  
    console.log( 'Hello everyone!' );  
}  
  
showMessage();
```

Parameters

- ฟังก์ชันบางฟังก์ชันต้องรับค่าจากข้างนอกมาเพื่อทำงาน
- ซึ่งจะรับค่าผ่านตัวแปร
- ตัวแปรนั้นเรียกว่า Parameters หรืออีกชื่อคือ function arguments

```
function showMessage(from, text) { // arguments: from, text
  console.log(from + ': ' + text);
}
```

```
showMessage('Ann', 'Hello!'); // Ann: Hello!
showMessage('Ann', "What's up?"); // Ann: What's up?
```

การคืนค่าของฟังก์ชัน

- function สามารถคืนค่าได้ออกได้
- เขียนตามหลังคำว่า return

```
function sum(a, b) {  
    return a + b;  
}  
  
let result = sum(1, 2);  
console.log( result ); // 3
```

การคืนค่าของฟังก์ชัน

- ถ้าฟังก์ชันนั้นมีการเขียน return; จะออกจากฟังก์ชันและไม่คืนค่าใด ๆ ออกจากฟังก์ชัน

```
function showMovie(age) {  
  if (age < 18) {  
    return;  
  }  
  
  console.log("Show Movie");  
}
```

การคืนค่าของฟังก์ชัน

- ไม่มีการเขียน return
- หรือมีการเขียน return;
- ฟังก์ชันนั้นจะคืนค่าเป็น undefined

```
function doNothing() {  
    /* empty */  
}  
  
console.log( doNothing() ); // undefined
```

```
function doNothing() {  
    return;  
}  
  
alert( doNothing() ); // undefined
```


การประกาศฟังก์ชันและใส่ในตัวแปร

- การประกาศฟังก์ชันปกติ (Function declaration)

```
function sayHi() {  
    console.log( "Hello" );  
}
```

การประกาศฟังก์ชันและใส่ในตัวแปร

- Function สามารถเก็บค่าใส่ตัวแปรได้เช่นเดียวกับ ข้อมูลประเภทอื่น ๆ (number, boolean, etc.) ฟังก์ชันนี้จะเรียกว่า "Function expression"

```
let sayHi = function () {  
  console.log( "Hello" );  
}
```

การเรียกใช้ฟังก์ชัน

- การเรียกใช้ฟังก์ชันจะต้องมี () ทุกครั้ง
- ถ้าไม่มีการเขียน () หลังชื่อฟังก์ชัน Function ก็จะไม่ถูกรับ

```
let showMessage = function () {  
    return 'Hello World!'  
}  
  
/* จะไม่มีการเรียก showMessage แต่จะแสดงรายละเอียดของฟังก์ชัน */  
console.log( showMessage );  
  
/* จะมีการเรียก showMessage() และ จะส่ง 'Hello World!' ออกมาให้ console.log ไปรัน  
ต่อ */  
console.log( showMessage() );
```

03

Arrow Function

Arrow function

- เป็นการเขียนฟังก์ชันอีกรูปแบบหนึ่ง
- ที่ไวยากรณ์สั้นจะเรียกว่า Arrow Function

```
let func = (arg1, arg2, ...argN) => expression
```

Arrow function

- ถ้าเปรียบเทียบกับ การประกาศฟังก์ชันปกติ
- Arrow function ก็คือการเขียน ฟังก์ชันแบบย่อ

```
let func = (arg1, arg2, ...argN) => expression
```

```
let func = function(arg1, arg2, ...argN) {  
    return expression;  
};
```

Arrow Function

ตัวอย่างที่ 1

```
// arrow function อันนี้เหมือนกับการเขียนแบบนี้
//
//    let sum = function(a, b) {
//        return a + b;
//    };

let sum = (a, b) => a + b;

console.log( sum(1, 2) ); // 3
```

Arrow Function

ตัวอย่างที่ 2

- จากตัวอย่างที่ 1 มี arguments 2 ตัว คือ a และ b
- ถ้ามี arguments แค่ตัวเดียว เราสามารถ ละวงเล็บได้ อีกด้วย

```
// arrow function นี้เหมือนกับการประกาศฟังก์ชันด้านล่าง
//   let double = function (n) {
//       return n * 2
//   }

let double = n => n * 2;

console.log(double(3)); // 6
```


Arrow Function

ตัวอย่างที่ 3

- ถ้าไม่มี arguments เลยให้ใส่วงเล็บว่างเปล่าไว้

```
let sayHi = () =>  
  console.log("Hello!");  
  
sayHi();
```

Arrow Function

Arrow function คืออะไร

- ถ้า body ใน arrow function มีมากกว่า 1 บรรทัด เราต้อง ใส่ {} ครอบ และต้องมีการ ใส่ return keyword ไว้

```
let sum = (a, b) => { // ต้องใช้ปีกกาถ้า code มีหลายบรรทัด
    let result = a + b;
    return result; // ถ้าใส่ปีกกาต้องบอกด้วยว่าจะ return อันไหน
};

console.log( sum(1, 2) ); // 3
```

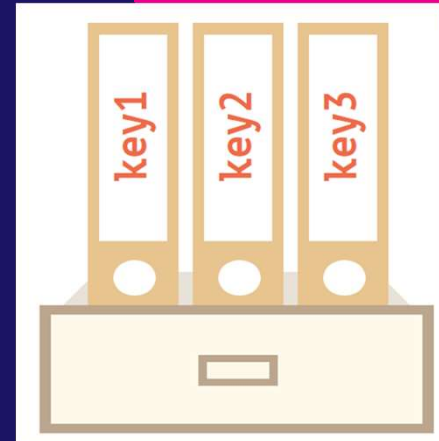
04

Object

Object

Object คืออะไร

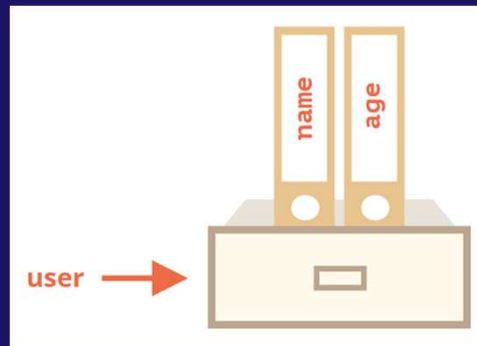
- ใน JavaScript จะมีข้อมูลทั้งหมด 6 ประเภท
- string, boolean, number, null, undefined และ Object
- ซึ่ง 5 ประเภทแรกจะเรียกว่า primitive



Object

Object คืออะไร

- Object จะประกอบด้วย properties
- ซึ่ง properties แต่ละอันจะประกอบด้วย key และ value

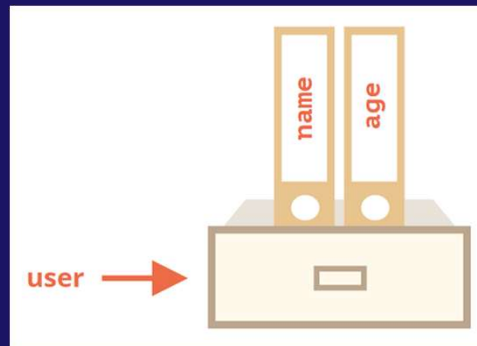


```
let user = {  
  name: "John",  
  age: 30  
};
```

Object

Object คืออะไร

- Object จะประกอบด้วย properties
- ซึ่ง properties แต่ละอันจะประกอบด้วย **key** และ value

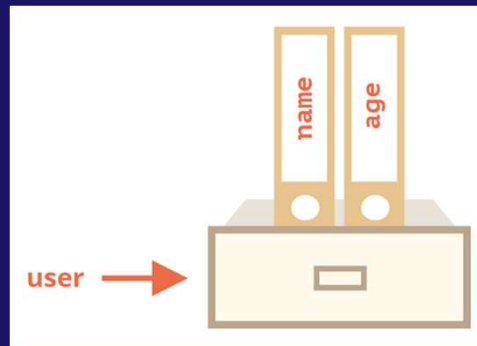


```
let user = {  
  name: "John",  
  age: 30  
};
```

Object

Object คืออะไร

- Object จะประกอบด้วย properties
- ซึ่ง properties แต่ละอันจะประกอบด้วย key และ value

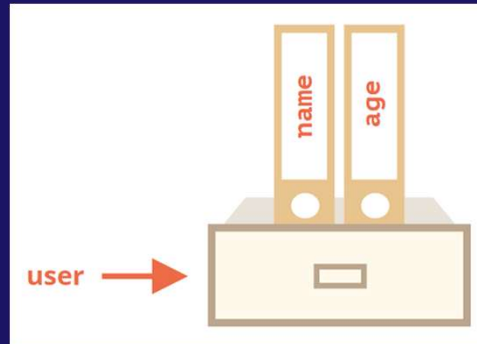


```
let user = {  
  name: "John",  
  age: 30  
};
```

Object

Object คืออะไร - ตัวอย่างที่ 1

- Key หรือเรียกอีกชื่อว่า property name
- Value หรือเรียกอีกชื่อว่า property value
- key และ value รวมกันเรียกว่า properties ของ Object

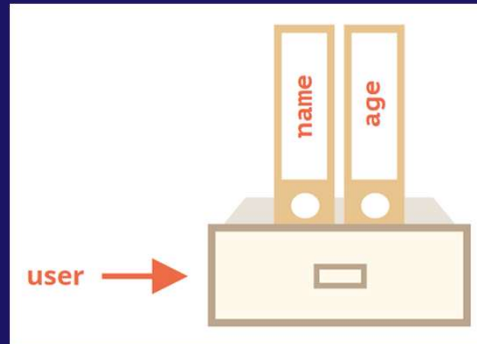


```
let user = {  
  name: "John",  
  age: 30  
};
```


Object

Object คืออะไร - ตัวอย่างที่ 1

- Key หรือเรียกอีกชื่อว่า property name
- Value หรือเรียกอีกชื่อว่า property value
- key และ value รวมกันเรียกว่า properties ของ Object

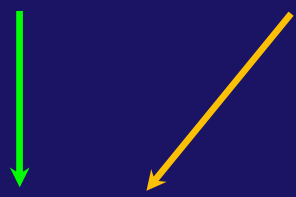


```
let user = {  
  name: "John",  
  age: 30  
};
```

Object

การอ่านค่า Properties ของ Object ออกมา (วิธีที่ 1)

- วิธีแรก คือ การใช้เครื่องหมายจุด
- **<ชื่อของ Object>.<ชื่อ Properties>**
ตัวอย่าง



```
alert( user.age ); // 30  
alert( user.name ); // "John"
```

```
let user = {  
  name: "John",  
  age: 30  
};
```

Object

การเพิ่ม Properties ของ Object

- $\text{<ชื่อของ Object>.<ชื่อ Properties> = <ค่าที่จะใส่ไปใน property นั้น>}$

ตัวอย่าง

`user.height = 176;`

`user.isSingle = true;`

Object

การเพิ่ม Properties ของ Object – ตัวอย่าง

- การเพิ่ม properties ของ user เพิ่ม 2 properties โดยเพิ่ม properties ที่มี key ชื่อ height และ value เป็น number ที่มีค่า 176 และ properties ที่มี key ชื่อ isSingle และ value เป็น boolean ที่มีค่าเป็น true

```
let user = {  
  name: "John",  
  age: 30  
};
```

Object

การเพิ่ม Properties ของ Object – ตัวอย่าง

- การเพิ่ม properties ของ user เพิ่ม 2 properties โดยเพิ่ม properties ที่มี key ชื่อ height และ value เป็น number ที่มีค่า 176 และ properties ที่มี key ชื่อ isSingle และ value เป็น boolean ที่มีค่าเป็น true

```
let user = {  
  name: "John",  
  age: 30  
};
```

user.height = 176;

```
let user = {  
  name: "John",  
  age: 30,  
  
};
```

Object

การเพิ่ม Properties ของ Object – ตัวอย่าง

- การเพิ่ม properties ของ user เพิ่ม 2 properties โดยเพิ่ม properties ที่มี key ชื่อ height และ value เป็น number ที่มีค่า 176 และ properties ที่มี key ชื่อ isSingle และ value เป็น boolean ที่มีค่าเป็น true

```
let user = {  
  name: "John",  
  age: 30  
};
```

`user.height = 176;`

```
let user = {  
  name: "John",  
  age: 30,  
  height: 176,  
};
```

Object

การเพิ่ม Properties ของ Object – ตัวอย่าง

- การเพิ่ม properties ของ user เพิ่ม 2 properties โดยเพิ่ม properties ที่มี key ชื่อ height และ value เป็น number ที่มีค่า 176 และ properties ที่มี key ชื่อ isSingle และ value เป็น boolean ที่มีค่าเป็น true

```
let user = {  
  name: "John",  
  age: 30  
};
```

user.height = 176;

```
let user = {  
  name: "John",  
  age: 30,  
  height: 176,  
};
```

Object

การเพิ่ม Properties ของ Object – ตัวอย่าง

- การเพิ่ม properties ของ user เพิ่ม 2 properties โดยเพิ่ม properties ที่มี key ชื่อ height และ value เป็น number ที่มีค่า 176 และ properties ที่มี key ชื่อ isSingle และ value เป็น boolean ที่มีค่าเป็น true

```
let user = {  
  name: "John",  
  age: 30  
};
```

```
user.height = 176;  
user.isSingle = true;
```

```
let user = {  
  name: "John",  
  age: 30,  
  height: 176,  
};
```


Object

การเพิ่ม Properties ของ Object – ตัวอย่าง

- การเพิ่ม properties ของ user เพิ่ม 2 properties โดยเพิ่ม properties ที่มี key ชื่อ height และ value เป็น number ที่มีค่า 176 และ properties ที่มี key ชื่อ isSingle และ value เป็น boolean ที่มีค่าเป็น true

```
let user = {  
  name: "John",  
  age: 30  
};
```

```
user.height = 176;  
user.isSingle = true;
```

```
let user = {  
  name: "John",  
  age: 30,  
  height: 176,  
  isSingle: true,  
};
```

Object

การเพิ่ม Properties ของ Object – ตัวอย่าง

- การเพิ่ม properties ของ user เพิ่ม 2 properties โดยเพิ่ม properties ที่มี key ชื่อ height และ value เป็น number ที่มีค่า 176 และ properties ที่มี key ชื่อ isSingle และ value เป็น boolean ที่มีค่าเป็น true

```
let user = {  
  name: "John",  
  age: 30  
};
```

```
user.height = 176;  
user.isSingle = true;
```

```
let user = {  
  name: "John",  
  age: 30,  
  height: 176,  
  isSingle: true,  
};
```

Object

การอ่านค่า Properties ของ Object ออกมา (วิธีที่ 2)

- วิธีที่สอง คือการใช้วงเล็บก้ามปู[]
- ใช้ในกรณีที่ key มีช่องว่างได้ด้วย
- **<ชื่อของ Object>**[**<ชื่อ Properties>**]
ตัวอย่าง

`alert(user["computer skill"]); // "JavaScript"`

```
let user = {  
  name: "John",  
  age: 30,  
  "computer skill": "JavaScript"  
};
```

Object

- การเพิ่ม Properties ก็สามารถทำได้แบบเดียวกับแบบจุด
 - วิธีแรก คือการใช้วงเล็บก้ามปู[]

```
let user = {  
  name: "John",  
  age: 30,  
};
```

Object

- การเพิ่ม Properties ก็สามารถทำได้แบบเดียวกับแบบจุด
 - วิธีแรก คือการใช้วงเล็บก้ามปู[]

```
user["likes birds"] = true;
```

```
let user = {  
  name: "John",  
  age: 30,  
  
};
```

Object

- การเพิ่ม Properties ก็สามารถทำได้แบบเดียวกับแบบจุด
 - วิธีแรก คือการใช้วงเล็บก้ามปู[]

```
user["likes birds"] = true;
```

```
let user = {  
  name: "John",  
  age: 30,  
  "likes birds": true,  
};
```

Object

Nested object – object ที่มี properties ที่เป็น object

- object ก็สามารถมี properties ที่เป็น object ได้เหมือนกัน (Object ซ้อน Object)

```
let user = {  
  name: "John",  
  sizes: {  
    height: 182,  
    width: 50  
  }  
};  
  
alert( user.sizes.height ); // 182
```

05

Array

Array

Array คืออะไร

- Array เป็นประเภทของข้อมูลที่ทำให้เราเก็บชุดของข้อมูลได้
- วิธีประกาศ Array มีสองวิธี แต่เราจะใช้แค่วิธีที่ไฮไลต์สีเหลืองในคอร์สนี้

```
let arr = new Array();  
let arr = [];
```

Array

การเรียกข้อมูลใน Array – ตัวอย่าง

- ในอาเรย์จะมีลำดับของข้อมูลโดยเริ่มต้นที่ 0 เรียกว่า index

```
let fruits = ["Apple", "Orange", "Plum"];
```

Array

การเรียกข้อมูลใน Array - ตัวอย่าง

- ในอาเรย์จะมีลำดับของข้อมูลโดยเริ่มต้นที่ 0 เรียกว่า index

Index 0

Index 1

Index 2

```
let fruits = ["Apple", "Orange", "Plum"];
```

Array

การเรียกข้อมูลใน Array - ตัวอย่าง

- วิธีเรียกข้อมูลใน Array คือ
- `<ชื่อของ Array>[<index>]`

```
let fruits = ["Apple", "Orange", "Plum"];
```

```
alert( fruits[0] ); // Apple  
alert( fruits[1] ); // Orange  
alert( fruits[2] ); // Plum
```

Index 0

Index 1

Index 2

```
let fruits = ["Apple", "Orange", "Plum"];
```

Array

การแก้ไขข้อมูลใน Array

- วิธีแก้ไขค่าข้อมูลใน Array คือ
- `<ชื่อของ Array>[<index>] = <ค่าที่ต้องการจะแทน>`

`fruits[2] = 'Pear';`

Index 0

Index 1

Index 2

```
let fruits = ["Apple", "Orange", "Plum"];
```

Array

การแก้ไขข้อมูลใน Array

- วิธีแก้ไขค่าข้อมูลใน Array คือ
- `<ชื่อของ Array>[<index>] = <ค่าที่ต้องการจะแทน>`

`fruits[2] = 'Pear';`

```
let fruits = ["Apple", "Orange", "Plum"];
```

Index 0

Index 1

Index 2

Array

การแก้ไขข้อมูลใน Array

- วิธีแก้ไขค่าข้อมูลใน Array คือ
- `<ชื่อของ Array>[<index>] = <ค่าที่ต้องการจะแทน>`

`fruits[2] = 'Pear';`

Index 0

Index 1

Index 2

```
let fruits = ["Apple", "Orange", "Plum"];
```

Array

การเพิ่มข้อมูลใน Array

- `array.push(item)` คือการเพิ่มสมาชิกต่อท้าย array

fruits

```
["Apple", "Orange", "Plum"];
```

```
let fruits = ["Apple", "Orange", "Plum"];
```


Array

การเพิ่มข้อมูลใน Array

- `array.push(item)` คือการเพิ่มสมาชิกต่อท้าย array

fruits

```
["Apple", "Orange", "Plum"];
```

```
let fruits = ["Apple", "Orange", "Plum"];  
fruits.push("Banana")
```

Array

การเพิ่มข้อมูลใน Array

- `array.push(item)` คือการเพิ่มสมาชิกต่อท้าย array

fruits

```
["Apple", "Orange", "Plum" "Banana"];
```

```
let fruits = ["Apple", "Orange", "Plum"];  
fruits.push("Banana")
```

Array

การหาความยาวของ Array – ตัวอย่าง

- วิธีหาความยาวของ Array คือ
- **<ชื่อของ Array>.length**

```
let fruits = ["Apple", "Orange", "Plum"];  
  
alert( fruits.length ); // 3
```

Array

Loop กับ Array

- ตัวอย่าง

```
let arr = ["Apple", "Orange", "Pear"];

for (let i = 0; i < arr.length; i++) {
  console.log( arr[i] );
}
```

```
let fruits = ["Apple", "Orange", "Plum"];

for (let fruit of fruits) {
  console.log( fruit );
}
```

06

Method ប្រើប្រាស់ Array

Method ของ Array

Methods สำหรับ เพิ่ม/ลบ items ของ Array

- `arr.push(...items)` - เพิ่ม items ไปข้างหลัง
- `arr.pop()` - นำค่าข้างหลังออกมา
- `arr.shift()` - นำค่าข้างหน้าออกมา
- `arr.unshift(...items)` - เพิ่ม items ไปข้างหน้า

Method ของ Array

การรวม Array หลายอันเป็นอันเดียว

- การต่อ array - ใช้เมื่อเราต้องการรวม array ทั้งหลายอันเข้าด้วยกัน
- Syntax

```
arr.concat(arg1, arg2...)
```

Method ของ Array

การรวม Array หลายอันเป็นอันเดียว – ตัวอย่าง

```
let arr = [1, 2];

// สร้าง Array ใหม่ จาก arr and [3,4]
alert( arr.concat([3, 4]) ); // 1,2,3,4

// สร้าง Array ใหม่ จาก arr and [3,4] and [5,6]
alert( arr.concat([3, 4], [5, 6]) ); // 1,2,3,4,5,6

// สร้าง Array ใหม่ จาก arr and [3,4], และเพิ่ม elements 5 กับ 6
alert( arr.concat([3, 4], 5, 6) ); // 1,2,3,4,5,6
```


Method ของ Array

การรวม Array หลายอันเป็นอันเดียว – ตัวอย่าง

```
let arr = [1, 2];

// สร้าง Array ใหม่ จาก arr and [3,4]
console.log( arr.concat([3, 4]) ); // [1,2,3,4]

// สร้าง Array ใหม่ จาก arr and [3,4] and [5,6]
console.log( arr.concat([3, 4], [5, 6]) ); // [1,2,3,4,5,6]

// สร้าง Array ใหม่ จาก arr and [3,4], และเพิ่ม elements 5 กับ 6
console.log( arr.concat([3, 4], 5, 6) ); // [1,2,3,4,5,6]
```

Method ของ Array

find - การค้นหาสมาชิกใน Array

- ในกรณีที่เรต้องการหาว่าสมาชิกที่เราต้องการ เรา
จะใช้ฟังก์ชัน find เช่น

```
#javascript  
Shapes = [● ▲ ■ ▲ ▲]  
Shapes.find(shape => shape === ▲)  
[▲]
```

<https://www.freecodecamp.org/news/compare-array-find-vs-find-index/>
@UbanTheBuilder

Method ของ Array

find - การค้นหาสมาชิกใน Array

- ส่วนใหญ่เราจะใช้กับ Array ที่เก็บ Object ไว้
- โดยฟังก์ชัน find จะรับฟังก์ชันที่ใช้สำหรับการเช็คเงื่อนไข
เช่น อยากได้ชื่อของคนที่มี id เท่ากับ 2

```
let users = [  
  {id: 1, name: "John"},  
  {id: 2, name: "Pete"},  
  {id: 3, name: "Mary"}  
];
```

Method ของ Array

find – อยากได้ชื่อของคนที่ id เท่ากับ 2

- ฟังก์ชัน find จะรับฟังก์ชันที่ใช้สำหรับการเช็คเงื่อนไข และส่ง สมาชิกที่ตรงเงื่อนไขตัวแรก

```
let users = [  
  { id: 1, name: "John" },  
  { id: 2, name: "Pete" },  
  { id: 3, name: "Mary" }  
];  
  
users.find(function (e) {  
  return e.id === 2  
});
```

ฟังก์ชันสำหรับ
การเช็คเงื่อนไข

Method ของ Array

find – อยากได้ชื่อของคนที่ id เท่ากับ 1

```
let users = [  
  { id: 1, name: "John" },  
  { id: 2, name: "Pete" },  
  { id: 3, name: "Mary" }  
];
```

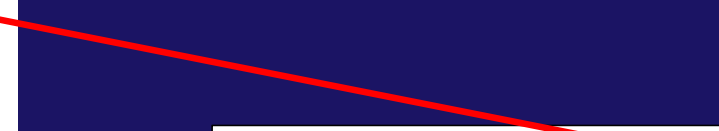
```
users.find(function (e) {  
  return e.id === 2  
});
```

Method ของ Array

find – อยากได้ชื่อของคนที่ id เท่ากับ 1

```
let users = [  
  { id: 1, name: "John" },  
  { id: 2, name: "Pete" },  
  { id: 3, name: "Mary" }  
];
```

```
users.find(function (e) {  
  return e.id === 2  
});
```



Method ของ Array

find – อยากได้ชื่อของคนที่มี id เท่ากับ 1

```
let users = [  
  { id: 1, name: "John" },  
  { id: 2, name: "Pete" },  
  { id: 3, name: "Mary" }  
];
```

```
users.find(function(e) {  
  return e.id === 2  
});
```

Method ของ Array

find – อยากได้ชื่อของคนที่มี id เท่ากับ 1

```
let users = [  
  { id: 1, name: "John" },  
  { id: 2, name: "Pete" },  
  { id: 3, name: "Mary" }  
];
```

```
users.find(function(e) {  
  return e.id === 2  
});
```

ยังไม่ตรงเงื่อนไข เช็คตัวถัดไป

Method ของ Array

find – อยากได้ชื่อของคนที่ id เท่ากับ 1

```
let users = [  
  { id: 1, name: "John" },  
  { id: 2, name: "Pete" },  
  { id: 3, name: "Mary" }  
];
```

```
users.find(function (e) {  
  return e.id === 2  
});
```

Method ของ Array

find – อยากได้ชื่อของคนที่ id เท่ากับ 1

```
let users = [  
  { id: 1, name: "John" },  
  { id: 2, name: "Pete" },  
  { id: 3, name: "Mary" }  
];
```

```
users.find(function(e) {  
  return e.id === 2  
});
```

Method ของ Array

find – อยากได้ชื่อของคนที่มี id เท่ากับ 1

```
let users = [  
  { id: 1, name: "John" },  
  { id: 2, name: "Pete" },  
  { id: 3, name: "Mary" }  
];
```

```
users.find(function (e) {  
  return e.id === 2  
});
```

Method ของ Array

find – อยากได้ชื่อของคนที่ id เท่ากับ 1

```
let users = [  
  { id: 1, name: "John" },  
  { id: 2, name: "Pete" },  
  { id: 3, name: "Mary" }  
];
```

```
users.find(function (e) {  
  return e.id === 2  
});
```

ตรงเงื่อนไขแล้ว

Method ของ Array

find – อยากได้ชื่อของคนที่ id เท่ากับ 1

```
let users = [  
  { id: 1, name: "John" },  
  { id: 2, name: "Pete" },  
  { id: 3, name: "Mary" }  
];
```

```
users.find(function (e) {  
  return e.id === 2  
});
```

ตรงเงื่อนไขแล้ว

```
{ id: 2, name: "Pete" }
```

เราจึงได้ตัวนี้ออกมาและไม่หาตัวถัดไปแล้ว

Method ของ Array

find – อยากได้ชื่อของคนที่ id เท่ากับ 2

สรุป tUser ก็จะได้เป็น { id: 2, name: "Pete" }

```
let users = [  
  { id: 1, name: "John" },  
  { id: 2, name: "Pete" },  
  { id: 3, name: "Mary" }  
];  
  
const tUser = users.find(function (e) {  
  return e.id === 2  
});
```

Method ของ Array

find – อยากได้ชื่อของคนที่ id เท่ากับ 2
เราสามารถเขียนโดยใช้ Arrow Function ก็ได้

```
let users = [  
  { id: 1, name: "John" },  
  { id: 2, name: "Pete" },  
  { id: 3, name: "Mary" }  
];  
  
const tUser = users.find(function (e) {  
  return e.id === 2  
});
```

Method ของ Array

find – อยากได้ชื่อของคนที่ id เท่ากับ 2
เราสามารถเขียนโดยใช้ Arrow Function ก็ได้

```
let users = [  
  { id: 1, name: "John" },  
  { id: 2, name: "Pete" },  
  { id: 3, name: "Mary" }  
];  
  
const tUser = users.find(function (e) {  
  return e.id === 2  
});
```

```
let users = [  
  { id: 1, name: "John" },  
  { id: 2, name: "Pete" },  
  { id: 3, name: "Mary" }  
];  
  
const tUser = users.find(e => e.id === 2);
```


Method ของ Array

find – อยากได้ชื่อของคนที่มี id เท่ากับ 2
จะเห็นได้ว่าอ่านง่ายและสั้นลงเยอะเลย

```
let users = [  
  { id: 1, name: "John" },  
  { id: 2, name: "Pete" },  
  { id: 3, name: "Mary" }  
];  
  
const tUser = users.find(function (e) {  
  return e.id === 2  
});
```

```
let users = [  
  { id: 1, name: "John" },  
  { id: 2, name: "Pete" },  
  { id: 3, name: "Mary" }  
];  
  
const tUser = users.find(e => e.id === 2);
```

Method ของ Array

filter – การคัดกรองสมาชิกใน Array

- filter จะคล้ายๆ กับ find โดยจะเอาสมาชิกที่ตรงกับเงื่อนไข
- แต่ find จะเอาตัวแรกตัวเดียว แต่ filter จะเอาทุกตัวที่ตรงกับเงื่อนไข

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];  
  
const tUser = users.find(e => e.age === 18);
```

Method ของ Array

filter – อยากได้เฉพาะคนที่มีอายุเท่ากับ 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.find(e => e.age === 18);
```

Method ของ Array

filter – อยากได้เฉพาะคนที่มีอายุเท่ากับ 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.find(e => e.age === 18);
```

Method ของ Array

filter – อยากได้เฉพาะคนที่มีอายุเท่ากับ 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

`users.find(e => e.age === 18);`

Method ของ Array

filter – อยากได้เฉพาะคนที่มีอายุเท่ากับ 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.find(e => e.age === 18);
```

ตัวนี้ยังไม่ตรงเงื่อนไข เช็คตัวถัดไป

Method ของ Array

filter – อยากได้เฉพาะคนที่มีอายุเท่ากับ 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.find(e => e.age === 18);
```

Method ของ Array

filter – อยากได้เฉพาะคนที่มีอายุเท่ากับ 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

`users.find(e => e.age === 18);`

Method ของ Array

filter – อยากได้เฉพาะคนที่มีอายุเท่ากับ 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

`users.find(e => e.age === 18);`

Method ของ Array

filter – อยากได้เฉพาะคนที่มีอายุเท่ากับ 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.find(e => e.age === 18);
```

ตัวนี้ตรงเงื่อนไข นำมาเก็บไว้

Method ของ Array

filter – อยากได้เฉพาะคนที่มีอายุเท่ากับ 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.find(e => e.age === 18);
```

ตัวนี้ตรงเงื่อนไข นำมาเก็บไว้

```
[  
  { id: 2, name: "Pete", age: 18 },  
];
```



FutureSkill

Method ของ Array

filter – อยากได้เฉพาะคนที่มีอายุเท่ากับ 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.find(e => e.age === 18);
```

```
[  
  { id: 2, name: "Pete", age: 18 },  
];
```



Method ของ Array

filter – อยากได้เฉพาะคนที่มีอายุเท่ากับ 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

`users.find(e => e.age === 18);`

```
[  
  { id: 2, name: "Pete", age: 18 },  
];
```



Method ของ Array

filter – อยากได้เฉพาะคนที่มีอายุเท่ากับ 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.find(e => e.age === 18);
```

```
[  
  { id: 2, name: "Pete", age: 18 },  
];
```



Method ของ Array

filter – อยากได้เฉพาะคนที่มีอายุเท่ากับ 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.find(e => e.age === 18);
```

```
[  
  { id: 2, name: "Pete", age: 18 },  
];
```

Method ของ Array

filter – อยากได้เฉพาะคนที่มีอายุเท่ากับ 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.find(e => e.age === 18);
```

ตัวนี้ก็ตรงเงื่อนไข นำมาเก็บไว้เหมือนกัน

```
[  
  { id: 2, name: "Pete", age: 18 },  
];
```



Method ของ Array

filter – อยากได้เฉพาะคนที่มีอายุเท่ากับ 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.find(e => e.age === 18);
```

```
[  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
];
```

Method ของ Array

filter – อยากได้เฉพาะคนที่มีอายุเท่ากับ 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.find(e => e.age === 18);
```

```
[  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
];
```

Method ของ Array

filter – อยากได้เฉพาะคนที่มีอายุเท่ากับ 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.find(e => e.age === 18);
```

```
[  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
];
```

Method ของ Array

filter – อยากได้เฉพาะคนที่มีอายุเท่ากับ 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.find(e => e.age === 18);
```

ตัวนี้ไม่ตรงเงื่อนไข ไปตัวถัดไป

```
[  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
];
```

Method ของ Array

filter – อยากได้เฉพาะคนที่มีอายุเท่ากับ 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.find(e => e.age === 18);
```

แต่เนื่องจากตัวถัดไปไม่มีแล้ว Array
ข้างล่างนี้จะเป็นผลลัพธ์

```
[  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
];
```

Method ของ Array

filter – สรุ

ตัว `tUser` จะมีค่าเท่ากับ

```
[  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
];
```

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];  
  
const tUser = users.find(e => e.age === 18);
```

Method ของ Array

map – การแปลงสมาชิกใน Array

- ฟังก์ชัน map จะใช้สำหรับการแปลงสมาชิกใน Array
- โดยฟังก์ชัน map จะรับฟังก์ชันเหมือนกับ find และ filter เลยแต่ ฟังก์ชันนี้จะใช้สำหรับ การแปลงสมาชิก

Method ของ Array

map – การแปลงสมาชิกใน Array

ตัวอย่าง

- เราอยากได้เฉพาะ id ของ users ทั้งหมด

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];  
  
const tUser = users.map(e => e.id);
```

ฟังก์ชันที่ใช้สำหรับการแปลงค่า

Method ของ Array

filter – อยากรู้เฉพาะ id ของ users ที่มากกว่า

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.map(e => e.id);
```

Method ของ Array

filter – อยากรู้เฉพาะ id ของ users ทั้งหมด

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

`users.map(e => e.id);`

Method ของ Array

filter – อยากได้เฉพาะ id ของ users ทั้งหมด

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.map(e => e.id);
```

e.id มีค่าเท่ากับ 1 จึงเก็บไว้ใน array ผลลัพธ์ก่อน และไปตัวถัดไป

Method ของ Array

filter – อยากได้เฉพาะ id ของ users ทั้งหมด

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.map(e => e.id);
```

e.id มีค่าเท่ากับ 1 จึงเก็บไว้ใน array ผลลัพธ์ก่อน และไปตัวถัดไป

```
[ 1 ]
```

Method ของ Array

filter – อยากรู้เฉพาะ id ของ users ที่มากกว่า

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.map(e => e.id);
```

```
[ 1 ]
```

Method ของ Array

filter – อยากรู้เฉพาะ id ของ users ทั้งหมด

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

`users.map(e => e.id);`

`[1]`

Method ของ Array

filter – อยากได้เฉพาะ id ของ users ทั้งหมด

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.map(e => e.id);
```

e.id มีค่าเท่ากับ 2 จึงเก็บไว้ใน array ผลลัพธ์ และไปตัวถัดไป

```
[ 1 ]
```

Method ของ Array

filter – อยากได้เฉพาะ id ของ users ทั้งหมด

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

`users.map(e => e.id);`

e.id มีค่าเท่ากับ 2 จึงเก็บไว้ใน array ผลลัพธ์ และไปตัวถัดไป

`[1 , 2]`

Method ของ Array

filter – อยากรู้เฉพาะ id ของ users ที่ > 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.map(e => e.id);
```

```
[ 1 , 2 ]
```

Method ของ Array

filter – อยากรู้เฉพาะ id ของ users ที่มากกว่า

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.map(e => e.id);
```

```
[ 1 , 2 ]
```

Method ของ Array

filter – อยากรู้เฉพาะ id ของ users ที่มากกว่า

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

`users.map(e => e.id);`

`[1 , 2]`

Method ของ Array

filter – อยากได้เฉพาะ id ของ users ทั้งหมด

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.map(e => e.id);
```

e.id มีค่าเท่ากับ 3 จึงเก็บไว้ใน array ผลลัพธ์ และไปตัวถัดไป

```
[ 1 , 2 ]
```

Method ของ Array

filter – อยากได้เฉพาะ id ของ users ทั้งหมด

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.map(e => e.id);
```

e.id มีค่าเท่ากับ 3 จึงเก็บไว้ใน array ผลลัพธ์ และไปตัวถัดไป

```
[ 1 , 2 , 3 ]
```

Method ของ Array

filter – อยากได้เฉพาะ id ของ users ที่ > 18

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.map(e => e.id);
```

```
[ 1 , 2 , 3 ]
```

Method ของ Array

filter – อยากได้เฉพาะ id ของ users ทั้งหมด

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.map(e => e.id);
```

```
[ 1 , 2 , 3 ]
```

Method ของ Array

filter – อยากได้เฉพาะ id ของ users ที่ > 3

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

`users.map(e => e.id);`

`[1 , 2 , 3]`

Method ของ Array

filter – อยากได้เฉพาะ id ของ users ทั้งหมด

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.map(e => e.id);
```

e.id มีค่าเท่ากับ 4 จึงเก็บไว้ใน array ผลลัพธ์ และไปตัวถัดไป

```
[ 1 , 2 , 3 , 4 ]
```

Method ของ Array

filter – อยากได้เฉพาะ id ของ users ทั้งหมด

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];
```

```
users.map(e => e.id);
```

แต่ไม่มีตัวถัดไปแล้ว Array ของล่างจึงเป็นผลลัพธ์

```
[ 1 , 2 , 3 , 4 ]
```

Method ของ Array

map – การแปลงสมาชิกใน Array – สรุป

- ตัว `tUser` จึงมีค่าเท่ากับ `[1, 2, 3, 4]`

```
let users = [  
  { id: 1, name: "John", age: 17 },  
  { id: 2, name: "Pete", age: 18 },  
  { id: 3, name: "Mary", age: 18 },  
  { id: 4, name: "Sara", age: 17 }  
];  
  
const tUser = users.map(e => e.id);
```

07

Spread Syntax

Spread Syntax

Spread syntax ของ Array

- Spread operator คือการกระจายสมาชิกของ Array ออกมาใช้งาน เช่น การรวมของกันของ Array ใช้แทน Concat ได้

```
let arr = [3, 5, 1];  
let arr2 = [8, 9, 15];
```

Spread Syntax

Spread syntax ของ Array

- Spread operator คือการกระจายสมาชิกของ Array ออกมาใช้งาน เช่น การรวมของกันของ Array ใช้แทน Concat ได้

```
let arr = [3, 5, 1];  
let arr2 = [8, 9, 15];
```

arr ก็คือตัว array ที่ highlight ไว้

```
let merged = [0, ...arr, 2, ...arr2];  
// merged = [0, 3, 5, 1, 2, 8, 9, 15];
```

```
alert(merged); // 0,3,5,1,2,8,9,15 (0, then arr, then 2, then arr2)
```

Spread Syntax

Spread syntax ของ Array

- Spread operator คือการกระจายสมาชิกของ Array ออกมาใช้งาน เช่น การรวมของกันของ Array ใช้แทน Concat ได้

```
let arr = [3, 5, 1];  
let arr2 = [8, 9, 15];    การใส่ ...arr ก็เปรียบเสมือนการทำให้โลโก้ข้างล่าง  
  
let merged = [0, ...arr, 2, ...arr2];  
// merged = [0, 3, 5, 1, 2, 8, 9, 15];  
  
alert(merged); // 0,3,5,1,2,8,9,15 (0, then arr, then 2, then arr2)
```

Spread Syntax

Spread syntax ของ Array

- Spread operator คือการกระจายสมาชิกของ Array ออกมาใช้งาน เช่น การรวมของกันของ Array ใช้แทน Concat ได้

```
let arr = [3, 5, 1];  
let arr2 = [8, 9, 15];  
  
let merged = [0, ...arr, 2, ...arr2];  
// merged = [0, 3, 5, 1, 2, 8, 9, 15];  
  
alert(merged); // 0,3,5,1,2,8,9,15 (0, then arr, then 2, then arr2)
```

arr2 ก็เช่นเดียวกัน คือตัว array ที่ highlight ไว้

Spread Syntax

Spread syntax ของ Array

- Spread operator คือการกระจายสมาชิกของ Array ออกมาใช้งาน เช่น การรวมของกันของ Array ใช้แทน Concat ได้

```
let arr = [3, 5, 1];  
let arr2 = [8, 9, 15];  
  
let merged = [0, ...arr, 2, ...arr2];  
// merged = [0, 3, 5, 1, 2, 8, 9, 15];  
  
alert(merged); // 0,3,5,1,2,8,9,15 (0, then arr, then 2, then arr2)
```

การใช้ ...arr2 ก็เปรียบเสมือนการทำให้รูปแบบที่ไฮไลท์ข้างล่าง

Spread Syntax

Spread syntax ของ Array

- Spread operator คือการกระจายสมาชิกของ Array ออกมาใช้งาน เช่น การรวมของกันของ Array ใช้แทน Concat ได้

```
let arr = [3, 5, 1];  
let arr2 = [8, 9, 15];  
  
let merged = [0, ...arr, 2, ...arr2];  
// merged = [0, 3, 5, 1, 2, 8, 9, 15];  
  
alert(merged); // 0,3,5,1,2,8,9,15 (0, then arr, then 2, then arr2)
```

ผลลัพธ์ของการทำก็จะได้เหมือนตัวด้านล่าง

Spread Syntax

Spread syntax ของ Array

- Spread syntax คือการกระจายสมาชิกของ Array ออกมาใช้งาน เช่น การรวมของกันของ Array ใช้แทน Concat ได้

```
let arr = [3, 5, 1];  
let arr2 = [8, 9, 15];  
  
let merged = [0, ...arr, 2, ...arr2];  
// merged = [0, 3, 5, 1, 2, 8, 9, 15];  
  
alert(merged); // 0,3,5,1,2,8,9,15 (0, then arr, then 2, then arr2)
```

อันนี้เป็นแบบแยกสีไว้

Spread Syntax

Spread syntax ของ Object

- Spread syntax ในกรณีของ Object ก็คือการกระจาย Properties ของ Object ออกมาใช้งาน เช่น การรวมของกันของ Object

```
const user = {  
  name: "Sonter",  
  age: 18  
}  
  
const details = {  
  skill: "React",  
  language: "JavaScript"  
}  
  
const instructor = { ...user, ...details }
```

Spread Syntax

Spread syntax ของ Object – ตัวอย่าง

```
const user = {  
  name: "Sonter",  
  age: 18  
}  
  
const details = {  
  skill: "React",  
  language: "JavaScript"  
}
```

Object ที่ชื่อ user ก็จะประกอบไปด้วย name และ age

```
const instructor = { ...user, ...details }
```

Spread Syntax

Spread syntax ของ Object – ตัวอย่าง

```
const user = {  
  name: "Sonter",  
  age: 18  
}
```

```
const details = {  
  skill: "React",  
  language: "JavaScript"  
}
```

Object ที่ชื่อ details ก็จะประกอบไปด้วย name และ age

```
const instructor = { ...user, ...details }
```

Spread Syntax

Spread syntax ของ Object – ตัวอย่าง

```
const user = {  
  name: "Sonter",  
  age: 18  
}  
  
const details = {  
  skill: "React",  
  language: "JavaScript"  
}
```

การนำ `{...user, ...details}` ก็เปรียบเสมือนการนำ properties ทั้งสอง object มารวมกัน

```
const instructor = { ...user, ...details }
```

Spread Syntax

Spread syntax ของ Object – ตัวอย่าง

```
const user = {  
  name: "Sonter",  
  age: 18  
}
```

```
const details = {  
  skill: "React",  
  language: "JavaScript"  
}
```

การนำ `{...user, ...details}` ก็เปรียบเสมือนการนำ properties ทั้งสอง object มารวมกัน

```
const instructor = { ...user, ...details }
```


Spread Syntax

Spread syntax ของ Object – ตัวอย่าง

```
const user = {  
  name: "Sonter",  
  age: 18  
}  
  
const details = {  
  skill: "React",  
  language: "JavaScript"  
}
```

การนำ { ...user, ...details } ก็เปรียบเสมือนการนำ properties ทั้งสอง object มารวมกัน

```
const instructor = { ...user, ...details }
```

Spread Syntax

Spread syntax ของ Object – ตัวอย่าง

```
const user = {  
  name: "Sonter",  
  age: 18  
}  
  
const details = {  
  skill: "React",  
  language: "JavaScript"  
}
```

การนำ {...user,...details} ก็เปรียบเสมือนการนำ properties ทั้งสอง object มารวมกัน

```
const instructor = {  
  name: "Sonter",  
  age: 18,  
  ...details  
}
```

Spread Syntax

Spread syntax ของ Object – ตัวอย่าง

```
const user = {  
  name: "Sonter",  
  age: 18  
}  
  
const details = {  
  skill: "React",  
  language: "JavaScript"  
}
```

การนำ { ...user, ...details } ก็เปรียบเสมือนการนำ properties ทั้งสอง object มารวมกัน

```
const instructor = {  
  name: "Sonter",  
  age: 18,  
  ...details  
}
```

Spread Syntax

Spread syntax ของ Object – ตัวอย่าง

```
const user = {  
  name: "Sonter",  
  age: 18  
}  
  
const details = {  
  skill: "React",  
  language: "JavaScript"  
}
```

ผลลัพธ์ที่ได้ก็คือ Object อันใหม่

```
const instructor = {  
  name: "Sonter",  
  age: 18,  
  skill: "React",  
  language: "JavaScript"  
}
```

Spread Syntax

Spread syntax ของ Object – ตัวอย่าง

```
const user = {  
  name: "Sonter",  
  age: 18  
}  
  
const details = {  
  name: "React",  
  language: "JavaScript"  
}
```

ในกรณีที่ชื่อ property มีการซ้ำกัน ค่าที่มาทีหลังจะเป็นที่ถูกลเลือก

```
const instructor = {  
  ...user,  
  ...details  
}
```

Spread Syntax

Spread syntax ของ Object – ตัวอย่าง

```
const user = {  
  name: "Sonter",  
  age: 18  
}
```

```
const details = {  
  name: "React",  
  language: "JavaScript"  
}
```

จะเห็นว่ามี name ทั้งสองอันคือ Sonter และ React แต่ว่า object ที่ชื่อว่า details อยู่ข้างหลัง เพราะฉะนั้นตัว name ใน object ใหม่ที่ชื่อว่า instructor ก็จะมีค่าเป็น React (ของ Details)

```
const instructor = {  
  ...user,  
  ...details  
}
```

Spread Syntax

Spread syntax ของ Object – ตัวอย่าง

```
const user = {  
  name: "Sonter",  
  age: 18  
}  
  
const details = {  
  name: "React",  
  language: "JavaScript"  
}
```

จะเห็นว่า มี name ทั้งสองอันคือ Sonter และ React แต่ว่า object ที่ชื่อว่า details อยู่ข้างหลัง เพราะฉะนั้นตัว name ใน object ใหม่ที่ชื่อว่า instructor ก็จะมีค่าเป็น React (ของ Details)

```
const instructor = {  
  name: "Sonter",  
  age: 18,  
  name: "React",  
  language: "JavaScript"  
}
```


Spread Syntax

Spread syntax ของ Object – ตัวอย่าง

```
const user = {  
  name: "Sonter",  
  age: 18  
}  
  
const details = {  
  name: "React",  
  language: "JavaScript"  
}
```

ตัวมาก่อนจะถูกตัวหลังทับ

```
const instructor = {  
  name: "Sonter",  
  age: 18,  
  name: "React",  
  language: "JavaScript"  
}
```



Spread Syntax

Spread syntax ของ Object – ตัวอย่าง

```
const user = {  
  name: "Sonter",  
  age: 18  
}  
  
const details = {  
  name: "React",  
  language: "JavaScript"  
}
```

ตัวมาก่อนจะถูกตัวหลังกับ

```
const instructor = {  
  name: "React",  
  age: 18,  
  language: "JavaScript"  
}
```

Spread Syntax

Spread syntax ของ Object – ตัวอย่าง

```
const user = {  
  name: "Sonter",  
  age: 18  
}  
  
const details = {  
  name: "React",  
  language: "JavaScript"  
}
```

ตัวมาก่อนจะถูกตัวหลังทับ

```
const instructor = {  
  name: "React",  
  age: 18,  
  language: "JavaScript"  
}
```



08

Destructuring

Destructuring

Destructuring คืออะไร

- ตัวถูกใช้เยอะใน JavaScript ก็คือ array และ object
- บางทีการเราไม่ได้ต้องการใช้สมาชิกทั้งหมดใน array หรือ object นั้น แต่ต้องการใช้แค่บางตัวเท่านั้น
- การกำหนดค่าแบบ Destructuring จะช่วยให้สามารถเอาสมาชิกที่อยู่ใน array หรือ object ออกมาให้อยู่ในตัวแปรได้
- ซึ่งสิ่งนี้จะทำให้เราสะดวกมาก ๆ

Destructuring

Destructuring คืออะไร – Destructuring Array

- ตัวอย่างการใช้กับ Array
- จะเห็นได้ว่าจะง่ายกว่าที่เราต้องไปดึงมาใส่ตัวแปรโดยใช้ index
- ทั้งสองแบบจะให้ผลลัพธ์เหมือนกัน

```
// เรามี array ที่ข้างในประกอบไปด้วย name และ surname
let arr = ["Ilya", "Kantor"];

// destructuring assignment
// กำหนด firstName = arr[0]
// และ surname = arr[1]
let [firstName, surname] = arr;

alert(firstName); // Ilya
alert(surname);  // Kantor
```

```
// let [firstName, surname] = arr;
let firstName = arr[0];
let surname = arr[1];
```

Destructuring

Destructuring คืออะไร – Destructuring Object

- ตัวอย่างการใช้กับ Object
- เราต้องการดึงค่าใน Object ในฟังก์ชัน มาใส่ในตัวแปร ในฟังก์ชันที่มีชื่อตัวแปรต้องตรงกับ properties ใน object

```
let { var1, var2 } = { var1: ..., var2: ... }
```

Destructuring

Destructuring คืออะไร – Destructuring Object ตัวอย่าง

- ทั้งสองข้างให้ผลลัพธ์เหมือนกัน

```
let options = {  
  title: "Menu",  
  width: 100,  
  height: 200  
};  
  
let {title, width, height} = options;  
  
alert(title); // Menu  
alert(width); // 100  
alert(height); // 200
```

```
let options = {  
  title: "Menu",  
  width: 100,  
  height: 200  
};  
  
let {title, width, height} = options;  
  
alert(options.title); // Menu  
alert(options.width); // 100  
alert(options.height); // 200
```

Destructuring

```
let options = {
  title: "Menu",
  width: 100,
  height: 200
};

// { sourceProperty: targetVariable }
let { width: w, height: h, title } = options;

// width -> w
// height -> h
// title -> title

alert(title); // Menu
alert(w);     // 100
alert(h);     // 200
```

Destructuring คืออะไร – Destructuring Object

- เราเปลี่ยนชื่อ variable ได้ด้วยการใช้ colon
- เปลี่ยน width ให้มีชื่อ w
- เปลี่ยน height ให้มีชื่อ h
- title มีชื่อแบบเดิม