



Software Park Thailand  
</Code Camp>

# JAVASCRIPT

---

Nuttachai Kulthammanit

Slide Design By Patteera Banlangthammas

# หัวข้อ

- ตัวเลข
  - ข้อความ
  - Array
  - Methods ของ Array
  - Iterable
- Map และ Set
  - Keys, Values, และ Entities



Software Park Thailand  
</Code Camp>

# ตัวเลข

# 1. ตัวเลข

## 1.1. วิธีการเขียนตัวเลข

- ถ้าต้องการเขียน 1 พันล้าน วิธีที่ชัดเจนที่สุดคือ

```
let billion = 1000000000;
```

- แต่ในการเขียน คนส่วนใหญ่จะหลีกเลี่ยงการเขียน 0 ยาว เพราะจะทำให้ผิดพลาดได้ง่าย

# 1. ตัวเลข

## 1.1. วิธีการเขียนตัวเลข

- ซึ่ง 1,000,000,000 สามารถเขียนเป็น “ พันล้าน ”
- ดังนั้น “7.3 พันล้าน” จะได้เป็น “ 7,300,000,000 ”
- เพราะว่า พันล้าน คือเลขที่มีหลักจำนวน 10 หลักนั่นเอง
- \*\*ใน Javascript ให้ตัวอักษร “e” แทนตัวเลขที่มีจำนวน 0 ต่อท้าย

```
let billion = 1e9; // 1 พันล้าน | ประกอบด้วย: 1 และ 0 เก้าตัว  
let billion = 7.3e9; // 7.3 พันล้าน (7,300,000,000)
```

# 1. ตัวเลข

## 1.1. วิธีการเขียนตัวเลข

- หรือเราจะกล่าวได้ว่า “e” คือ จำนวนที่อยู่ข้างหน้า “e” คูณกับ เลข 1 ที่ตามหลังด้วย 0 ตามจำนวนเลขที่เขียนเอาไว้หลัง “e”

$$1e3 = 1 * 1000$$

$$1.23e6 = 1.23 * 1000000$$

(หรือจะมองว่า e เป็น 10 ยกกำลัง เช่น  $1e3 = 1 * 10^3$  // 1 คูณ 10 ยกกำลัง 3)

# 1. ตัวเลข

## 1.1. วิธีการเขียนตัวเลข

- หรือเราจะกล่าวได้ว่า “e” คือ เลข 1 ที่ตามหลังด้วย 0 ตามเลขที่เขียนเอาไว้หลัง “e” คูณกับ จำนวนที่อยู่ข้างหน้า “e”

$$\begin{aligned} 1e3 &= 1 * 1000 & (=1,000) \\ 1.23e6 &= 1.23 * 1000000 & (=1,230,000) \end{aligned}$$

(หรือจะมองว่า e เป็น 10 ยกกำลัง เช่น  $2e3 = 2 * 10^3$  // 2 คูณ 10 ยกกำลัง 3)

# 1. ตัวเลข

## 1.1. วิธีการเขียนตัวเลข

- ในกรณีที่เขียนเลขลบ คือ จำนวนที่อยู่ข้างหน้า “e” หาร ด้วย เลข 1 ที่ตามหลังด้วย 0 ตามจำนวนเลขที่เขียนเอาไว้หลัง “e”

```
// -8 หารด้วย 1 ที่มีศูนย์ 8 ตัว  
1.23e-8 = 1.23 / 100000000    (=0.0000000123)
```

(หรือจะมองว่า e เป็น 10 ยกกำลัง เช่น  $12e-3 = 12 \times 10^{-3}$  // 12 คูณ 10 ยกกำลัง -3)



# 1. ตัวเลข

## 1.2. ตัวเลขฐาน 16(Hex), ฐาน 2(Binary), ฐาน 8(Octal)

- เลขฐาน 16 คือ ตัวเลขที่ใช้ใน Javascript เพื่อ แสดงสี, เข้ารหัสตัวอักษร และอื่นๆ















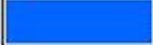

Color	Hexadecimal	Color	Hexadecimal
	FF0000		0000FF
	FF8000		7F00FF
	FFFF00		FF00FF
	7FFF00		FF0080
	00FF00		804000
	00FF80		7F7F7F
	00FFFF		FFFFFF
	007FFF		000000

Image credit : <https://catchascience.wordpress.com/igcse/igcse-computer-science/hexadecimal/>

# 1. ตัวเลข

## 1.2. ตัวเลขฐาน 16(Hex), ฐาน 8(Octal), ฐาน2(Binary)

- การเขียนเลขฐาน 16 ให้ใส่ 0x นำหน้าตัวเลข

```
alert( 0xff ); // 255
```

```
alert( 0xFF ); // 255 (เหมือนกันพิมพ์ใหญ่พิมพ์เล็กไม่สำคัญ)
```

- เลขฐาน 16 จะเขียนเป็นพิมพ์เล็กหรือพิมพ์ใหญ่ค่าจะออกมาเหมือนกัน

# 1. ตัวเลข

## 1.2. ตัวเลขฐาน 16(Hex), ฐาน 8(Octal), ฐาน2(Binary)

- การเขียนเลขฐาน 8 ให้ใส่ 0o นำหน้าตัวเลข
- การเขียนเลขฐาน 2 ให้ใส่ 0b นำหน้าตัวเลข

```
let a = 0b11111111; // binary form of 255  
let b = 0o377; // octal form of 255
```

```
alert( a == b ); // true, the same number 255 at both sides
```

# 1. ตัวเลข

## 1.3. toString(ประเภทเลขฐาน)

- ในชีวิตจริงนั้นเลขฐานที่ใช้กัน จะมีแค่เลขฐาน 2, 8, 10 และ 16
- ซึ่ง method `num.toString(base)` ทำให้สามารถแปลงเป็นฐานอะไรก็ได้
- ซึ่ง default คือ 10

```
let num = 255;
```

```
alert( num.toString(16) ); // ff  
alert( num.toString(2) ); // 11111111  
alert( num.toString() ); // 255
```

# 1. ตัวเลข

## 1.4. Rounding - การปัดเศษ

- วิธีการปัดเศษตัวเลขใน Javascript มีหลายวิธี เช่น
- Math.floor(ตัวเลข) ปัดเศษลงทั้งหมด

```
alert(Math.floor(3.6)); // output:3  
alert(Math.floor(3.1)); // output:3  
alert(Math.floor(-1.6)); // output:-2  
alert(Math.floor(-1.1)); // output:-2
```

# 1. ตัวเลข

## 1.4. Rounding - การปัดเศษ

- Math.ceil(ตัวเลข) ปัดเศษขึ้นทั้งหมด

```
alert(Math.ceil(3.6)); // output:4  
alert(Math.ceil(3.1)); // output:4  
alert(Math.ceil(-1.6)); // output:-1  
alert(Math.ceil(-1.1)); // output:-1
```

# 1. ตัวเลข

## 1.4. Rounding - การปัดเศษ

- Math.round(ตัวเลข) ตั้งแต่ .5 ปัดขึ้นทั้งหมด หากน้อยกว่า .5 ปัดลง

```
alert(Math.round(3.5)); // output:4  
alert(Math.round(3.1)); // output:3  
alert(Math.round(-1.6)); // output:-2  
alert(Math.round(-1.1)); // output:-1
```

# 1. ตัวเลข

## 1.4. Rounding - การปัดเศษ

- Math.trunc(ตัวเลข) ตัดทศนิยมทิ้ง

```
alert(Math.trunc(3.6)); // output:3  
alert(Math.trunc(3.1)); // output:3  
alert(Math.trunc(-1.6)); // output:-1  
alert(Math.trunc(-1.1)); // output:-1
```



# 1. ตัวเลข

## 1.4. Rounding - การปัดเศษ

- ลองทำ

```
let num = 1.23456;  
alert( Math.floor(num * 100) / 100 );
```

- คำถามคือจะได้ output ออกมาเป็นเท่าไร

เฉลยอยู่หน้าถัดไป

# 1. ตัวเลข

## 1.4. Rounding - การปัดเศษ

- คำถามคือจะได้ output ออกมาเป็นเท่าไร

```
let num = 1.23456;  
alert( Math.floor(num * 100) / 100 );
```

- เฉลย

$1.23456 * 100 \rightarrow 123.456$

$\text{Math.floor}(123.456) \rightarrow 123$

$123 / 100 \rightarrow 1.23$

คำตอบคือ 1.23

# 1. ตัวเลข

## 1.4. Rounding - การปัดเศษ

- Math.toFixed(จำนวนทศนิยม) เลือกจำนวนทศนิยมที่แสดง โดยจะปัดเศษแบบ round

```
let num = 12.34;  
alert( num.toFixed(1) ); // "12.3"
```

```
let num = 12.36;  
alert( num.toFixed(1) ); // "12.4"
```

```
let num = 12.34;  
alert( num.toFixed(5) ); // "12.34000"
```

# 1. ตัวเลข

## 1.5. Imprecise calculations (การคำนวณที่ไม่แม่นยำ)

```
alert( 0.1 + 0.2 == 0.3 ); // false
```

```
alert( 0.1 + 0.2 ); // 0.30000000000000004
```

- เกิดอะไรขึ้น?
- เนื่องจากคอมพิวเตอร์นั้น เก็บข้อมูลตัวเลขเป็นประเภทฐาน 2 ก่อนจะมาแปลงเป็นเลขฐาน 10 ให้เราเห็น
- ซึ่งการที่  $0.1 + 0.2$  ทำให้เกิด เลขฐาน 2 แบบไม่รู้จบ

# 1. ตัวเลข

## 1.5. Imprecise calculations (การคำนวณที่ไม่แม่นยำ)

- เราจะแก้ไขปัญหานี้ยังไง
- หนึ่งในวิธีที่เราใช้แก้ คือการใช้ `toFixed(n)`;

```
let sum = 0.1 + 0.2;  
alert( sum.toFixed(2) ); // 0.30
```

# 1. ตัวเลข

## 1.6. isFinite and isNaN

- NaN คือ “Not a Number” ไม่ใช่ตัวเลข
- ดังนั้น `isNaN(value)` คือฟังก์ชันที่เช็คว่าค่าที่ใส่มา ไม่ใช่ตัวเลข

```
alert( isNaN(NaN) ); // true
alert( isNaN("str") ); // true
alert( isNaN(123) ); // false เพราะเป็นตัวเลข
```

# 1. ตัวเลข

## 1.6. isFinite and isNaN

- Infinity (and -Infinity) คือจำนวนพิเศษที่มีค่ามากกว่า(น้อยกว่า)ทุกค่า
- isFinite(value) จะให้ค่าเป็น true ต่อเมื่อ value ไม่เป็นค่า NaN/Infinity

```
alert( isFinite("15") ); // true
alert( isFinite("str") ); // false, เพราะเป็น NaN
alert( isFinite(Infinity) ); // false, เพราะเป็น Infinity
```

# 1. ตัวเลข

## 1.7. parseInt and parseFloat

- parseInt(ข้อความ) - รับข้อความมาเปลี่ยนเป็นค่าจำนวนเต็ม
- parseFloat(ข้อความ) - รับข้อความมาเปลี่ยนเป็นค่าจำนวนจริง

```
alert( parseInt('100px') ); // 100  
alert( parseFloat('12.5em') ); // 12.5
```

```
alert( parseInt('12.3') ); // 12 ~ ส่งค่ากลับมาเฉพาะจำนวนเต็ม  
alert( parseFloat('12.3.4') ); // 12.3 ~ ส่งค่ากลับมาเฉพาะจุดตัวแรก
```



# 1. ตัวเลข

## 1.8. Other math functions

- **Math.random()** - จะ return ค่าจาก 0 ถึง 1 (ไม่นับ 1)

```
alert( Math.random() ); // 0.1234567899864  
alert( Math.random() ); // 0.5435258423432  
alert( Math.random() ); // ... (any random numbers)
```

# 1. ตัวเลข

## 1.8. Other math functions

- **Math.max(a, b, c...) / Math.min(a, b, c...)** - จะ return ค่าที่ มากที่สุด/น้อยที่สุด กลับมาจากค่าที่เราส่งไป

```
alert( Math.max(3, 5, -10, 0, 1) ); // 5  
alert( Math.min(1, 2, 5, 200) ); // 1
```

# 1. ตัวเลข

## 1.8. Other math functions

- **Math.pow(n, m)** - จะ return ค่า  $n$  กำลัง  $m$

```
alert( Math.pow(2, 10) ); // 2 ยกกำลัง 10 = 1024
```

# 1. ตัวเลข

## 1.9. แบบฝึกหัด

1. ให้เขียนฟังก์ชัน `random(min, max)` ที่จะ random เลข float ตั้งแต่ min จนถึง max มาให้เรา (ไม่รวม max)

```
alert( random(1, 5) ); // 1.2345623452  
alert( random(1, 5) ); // 3.7894332423  
alert( random(1, 5) ); // 4.3435234525
```



Software Park Thailand  
</Code Camp>



# หัวข้อ

- ตัวเลข
- ข้อความ
- Array
- Methods ของ Array
- Iterable

- Map และ Set
- Keys, Values, และ Entities



Software Park Thailand  
</Code Camp>

# ข้อมูล

# 2. ข้อความ (String)

## 2.1. Quotes

- การสร้างข้อความ(String) สามารถใช้ single quotes(' ')  
หรือ double quotes(" ") หรือ backticks(` `) ก็ได้

```
let single = 'single-quoted';  
let double = "double-quoted";  
  
let backticks = `backticks`;
```



# 2. ข้อความ (String)

## 2.1. Quotes

- ทั้งสามอย่างสามารถสร้างข้อความได้เหมือนกัน
- แต่ backticks จะอนุญาตให้เราใช้ `${...}` เพื่อใส่ค่า Expression

```
function sum(a, b) {  
  return a + b;  
}
```

```
alert(`1 + 2 = ${sum(1, 2)}.`); // 1 + 2 = 3.
```

Expression คือ “สิ่งที่ให้ส่งกลับมา” เช่น  $(1+1)*4$  จะส่งค่ากลับมาเป็น 8  
ซึ่งตัวแปรก็นับเป็น expression เช่นกัน เพราะมีการส่งค่ากลับมาเป็นสิ่งที่เก็บไว้ในตัวแปร  
function ก็เป็น expression เพราะมีการ return(ส่งค่ากลับมา) ยังตำแหน่งที่เรียกใช้

# 2. ข้อความ (String)

## 2.2. ตัวอักษรพิเศษ

- ตัวอักษรพิเศษ “\n” จะทำการสร้าง newline

```
let guestList = "Guests:\n * John\n * Pete\n * Mary";  
alert(guestList); //a multiline list of guests
```



Guests:  
\* John  
\* Pete  
\* Mary

ตกลง

# 2. ข้อความ (String)

## 2.2. ตัวอักษรพิเศษ

### - ตัวอักษรพิเศษอื่นๆ

Escape	Description	Example	Output String
\'	Single quote mark	"couldn't be"	couldn't be
\"	Double quote mark	"I \"think\" I \"am\""	I "think" I "am"
\\	Backslash	"one\\two\\three"	one\two\tthree
\n	New line	"I am\nI said"	I am I said
\r	Carriage return	"to be\r or not"	to be or not
\t	Tab	"one\ttwo\tthree"	one two three
\b	Backspace	"correctoin\b\b\bion"	correction
\f	Form feed	"Title A\fTitle B"	Title A then Title B

image credit : <https://medium.com/swlh/working-with-strings-in-javascript-34060a1c17a9>

# 2. ข้อความ (String)

## 2.3. ความยาวของข้อความ (String length)

- สามารถใช้ length ในการ บอกจำนวนตัวอักษร ในข้อความ

```
let str = "Happy New Year"
```

```
alert( str.length ); // 14
```

```
alert( "Hello World".length ); // 11
```

```
alert( `My\n`.length ); // 3
```

# 2. ข้อความ (String)

## 2.4. เข้าถึงตัวอักขระในข้อความ (Accessing characters)

- ก่อนจะเข้าถึงได้ ต้องเข้าใจก่อนว่า ข้อความ ใน javascript มีการเก็บ ตำแหน่งอักขระโดยเริ่มต้นที่ 0 ไม่ใช่ 1

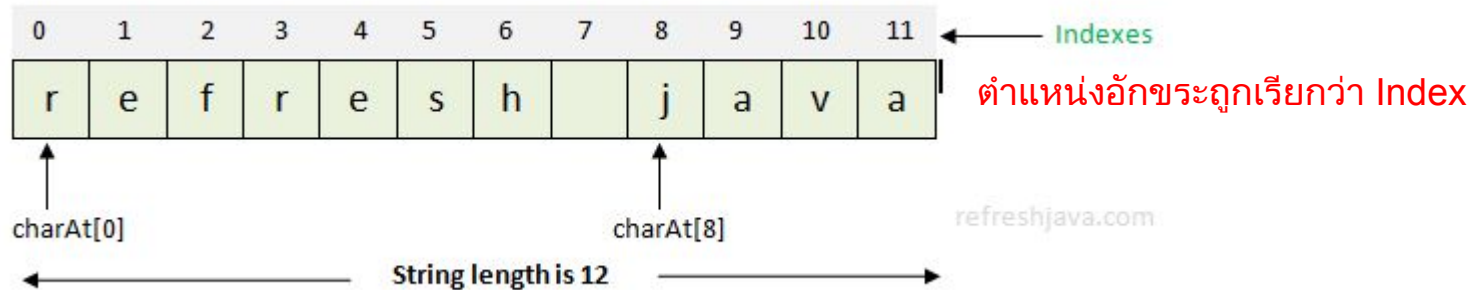


image credit : <https://refreshjava.com/java/string-class-methods>

# 2. ข้อความ (String)

## 2.4. เข้าถึงตัวอักขระในข้อความ (Accessing characters)

- วิธีเข้าถึงตัวอักขระในข้อความมี 2 วิธีคือ ใช้ brackets “[ ]” หรือ `charAt()`
- วิธีที่ 1 ใช้ brackets “[ ]” → วาง [] ไว้หลังข้อความ

```
let str = `Hello`;  
  
// เลือกตัวอักษรตัวแรก  
alert( str[0] ); // H
```



ตัวเลขในช่อง bracket คือตำแหน่ง(index) ของข้อความ str ซึ่งตัวแรกเริ่มต้นที่ 0

# 2. ข้อความ (String)

## 2.4. เข้าถึงตัวอักขระในข้อความ (Accessing characters)

- ตัวอย่างการใช้ bracket เพิ่มเติม ในกรณีที่เราใส่ ตำแหน่งนอกเหนือจากที่ มีจะส่งค่ากลับเป็น undefined

```
let str = `Hello`;  
// เลือกตัวอักษร  
alert( str[-1] ); // undefined  
alert( str[0] ); // h  
alert( str[1] ); // e  
alert( str[2] ); // l  
alert( str[3] ); // l  
alert( str[4] ); // o  
alert( str[5] ); // undefined
```

# 2. ข้อความ (String)

## 2.4. เข้าถึงตัวอักขระในข้อความ (Accessing characters)

- วิธีที่ 2 ใช้ `charAt()` → วาง `.charAt()` ไว้หลังตัวแปรข้อความหรือข้อความ
- `charAt()` นั้นมีการใช้เหมือน bracket แต่เขียนในรูปแบบที่ต่างกัน

```
let str = `Hello`;  
  
// เลือกตัวอักษรตัวแรก  
alert( str[0] ); // H  
alert( str.charAt[0] ); // H
```



# 2. ข้อความ (String)

## 2.4. เข้าถึงตัวอักขระในข้อความ (Accessing characters)

- สิ่งที่ใช้ `charAt()` ต่าง กับ bracket คือ
- เมื่อใส่ ตำแหน่งนอกเหนือจากที่มี จะส่งค่ากลับเป็น “ ข้อความเปล่า ”

```
let str = `Hello`;
```

```
// เลือกตัวอักษรตัวแรก  
alert( str[30] ); // undefined  
alert( str.charAt[30] ); // “ ”
```

# 2. ข้อความ (String)

## 2.4. เข้าถึงตัวอักขระในข้อความ (Accessing characters)

- สามารถ ใช้ for....of ในการ เข้าถึงตัวอักษร ทีละตัวได้

```
for (let char of "Hello") {  
  alert(char); // H,e,l,l,o  
}
```

# 2. ข้อความ (String)

## 2.5. ข้อความไม่สามารถแก้ไขได้ (Strings are immutable)

- ข้อความ ในภาษา javascript ไม่สามารถแก้ไขได้
- ดังนั้นการแก้ไขอักขระ ในข้อความจึงทำไม่ได้

```
let str = 'Hi';
```

```
str[0] = 'h'; // error  
alert( str[0] ); // doesn't work
```

# 2. ข้อความ (String)

## 2.5. ข้อความไม่สามารถแก้ไขได้ (Strings are immutable)

- หากต้องการเปลี่ยนแปลงข้อความทำได้โดยการ สร้างข้อความใหม่ทับ  
ลงตัวแปรเดิม

```
let str = 'Hi';  
  
str = 'h' + str[1]; // replace the string  
  
alert( str ); // hi
```

# 2. ข้อความ (String)

## 2.6. เปลี่ยนแปลงอักขระ

- มี 2 Methods สำหรับเปลี่ยนแปลงอักขระ
- toLowerCase() ใช้เปลี่ยนตัวอักษรภาษาอังกฤษทุกตัวเป็นพิมพ์เล็ก  
`alert( 'Interface123'.toLowerCase() ); // interface123`
- toUpperCase() ใช้เปลี่ยนตัวอักษรภาษาอังกฤษทุกตัวเป็นพิมพ์ใหญ่  
`alert( 'Interface123'.toUpperCase() ); // INTERFACE123`

**Methods เหมือนกับฟังก์ชัน** แต่มีข้อจำกัดที่มากกว่า ตัวอย่างเช่น methods ของข้อความ จะเป็นฟังก์ชันที่ใช้ได้เฉพาะกับข้อความเท่านั้น ไม่สามารถใช้อย่างอื่นได้เช่น ตัวเลข `123.toUpperCase() // error`

# 2. ข้อความ (String)

## 2.7. ค้นหาข้อความ (Searching for a substring)

- มีหลากหลายวิธีในการค้นหาข้อความ แต่ในหัวข้อนี้จะยกตัวอย่าง 2 วิธี
- วิธีที่ 1 ใช้ ข้อความ.indexOf(substr) -> ใช้ในการหาข้อความย่อยตัวแรก ที่เจอและส่งตำแหน่งแรกของตัวที่เจอกลับมา
- ในกรณีที่ไม่พบ จะส่งค่า -1 กลับมา

```
let str = 'Widget with id';
```

```
alert( str.indexOf('Widget') ); // 0, พบที่ตำแหน่งแรก  
alert( str.indexOf('widget') ); // -1, ไม่พบ เพราะว่าตัว "W" กับ "w"  
                                     ไม่ใช่ตัวเดียวกัน  
alert( str.indexOf("id") ); // 1, "id" พบที่ตำแหน่งที่ 1
```

# 2. ข้อความ (String)

## 2.7. ค้นหาข้อความ (Searching for a substring)

- ใช้ ข้อความ.indexOf(substr, position) -> ใช้ในการหาข้อความย่อยเริ่มจากตำแหน่ง position
- จากสไลด์หน้าก่อน จะพบว่ามีคำว่า “id” อยู่ข้างหลังอีกอัน

```
let str = 'Widget with id';
alert( str.indexOf("id") ); // 1, "id" พบที่ตำแหน่งที่ 1
```
- ถ้าเราต้องการหาข้อความย่อยตัวที่ 2 โดยเริ่มหาจากตำแหน่งที่ 2

```
let str = 'Widget with id';
      ↑
alert( str.indexOf("id", 2) ); // 12, เริ่มค้นหาจากตำแหน่งที่ 2
                                พบ "id" ที่ตำแหน่งที่ 12
```

# 2. ข้อความ (String)

## 2.7. ค้นหาข้อความ (Searching for a substring)

- indexOf เหมือนกับ lastIndexOf , แต่ lastIndexOf หาตัวสุดท้าย
- ข้อความ.lastIndexOf(substr, position) -> ใช้ในการ หาข้อความย่อยตัวสุดท้าย ที่เจอและ ส่งตำแหน่งแรก ของตัวที่เจอกลับมา
- ในกรณีที่ไม่มีพบ จะส่งค่า -1 กลับมา

```
let str = 'Widget with id';  
alert( str.lastIndexOf("id") ); // 12, "id" พบที่ตำแหน่งที่ 12
```



# 2. ข้อความ (String)

## 2.8. ข้อความย่อย (Substring)

- มีข้อความ “Hello World” ถ้าหากเราอยากได้แค่คำว่า “Hello” ต้องทำอย่างไร?
- ในหัวข้อนี้ขอเสนอ 3 วิธีในการแยกข้อความย่อยออกมาจากข้อความหลัก

# 2. ข้อความ (String)

## 2.8. ข้อความย่อย (Substring)

- วิธีที่ 1 คือการใช้ ข้อความ.slice(start [, end])

```
let str = "stringify";  
alert( str.slice(0, 5) ); // 'strin', เลือกจากตำแหน่งที่ 0 ถึง 5 (แต่ไม่เอาตำแหน่งที่ 5)  
alert( str.slice(0, 1) ); // 's', เลือกจากตำแหน่งที่ 0 ถึง 1 (แต่ไม่เอาตำแหน่งที่ 1)
```

- หาก ไม่ใส่ end จะเริ่มต้นจาก ตัวแรกจนถึงสุดท้าย

```
let str = "stringify";  
alert( str.slice(2) ); // 'ringify', เลือกจากตำแหน่งที่ 2 จนถึงตัวสุดท้าย
```

slice สามารถทำตำแหน่งกลับหลังได้ `alert( str.slice(-4, -1) ); // 'gif'`

↑  
คือตัวที่ 4 นับจากตำแหน่งท้ายสุด

# 2. ข้อความ (String)

## 2.8. ข้อความย่อย (Substring)

- วิธีที่ 2 คือการใช้ ข้อความ.substring(start [, end]) เหมือน slice

```
let str = "stringify";  
alert( str.substring(0, 5) ); // 'strin'  
alert( str.slice(0, 5) ); // 'strin' ได้ผลลัพธ์เหมือนกัน
```

```
alert( str.substring(2) ); // 'ringify'  
alert( str.slice(2) ); // 'ringify' ได้ผลลัพธ์เหมือนกัน
```

**\*\*แต่ substring ไม่สามารถทำกลับหลังได้** alert( str.slice(6, 2) ); // "" (an empty string)

# 2. ข้อความ (String)

## 2.8. ข้อความย่อย (Substring)

- วิธีที่ 3 คือการใช้ ข้อความ.substr(start [, length])

```
let str = "stringify";
```

```
alert( str.slice(2) ); // 'ringify'  
alert( str.substr(2) ); // 'ringify' ได้ผลลัพธ์เหมือนกัน
```

```
alert( str.slice(1, 5) ); // 'trin'  
alert( str.substr(1, 4) ); // 'trin' ได้ผลลัพธ์เหมือนกัน
```

↑  
จำนวนตัวอักขระที่จะให้แสดงออกมา เริ่มที่ตำแหน่งที่ 1

# 2. ข้อความ (String)

## 2.9. เปรียบเทียบข้อความ (Comparing strings)

- ตัวภาษาอังกฤษ พิมพ์เล็ก มีค่า มากกว่า พิมพ์ใหญ่เสมอ

```
alert( 'a' > 'Z' ); // true
```

- สงสัยกันหรือเปล่าครับว่า Javascript ใช้อะไรเปรียบเทียบตัวอักขระมากกว่าน้อยกว่า?

# 2. ข้อความ (String)

## 2.9. เปรียบเทียบข้อความ (Comparing strings)

- อักขระแต่ละตัวมีค่าของตัวเอง ซึ่งตัวเลขของค่าขึ้นอยู่กับมาตรฐาน unicode ที่ใช้

```
// หาค่าตัวเลขจากอักขระ
alert( "a".codePointAt(0) ); // 97
alert( "Z".codePointAt(0) ); // 90

// สร้างตัวขระจากค่าตัวเลข
alert( String.fromCharCode(97) ); // a
alert( String.fromCharCode(90) ); // Z
```

# 2. ข้อความ (String)

## 2.10. includes, startsWith, endsWith

- includes เป็นการเช็คว่ามี substring อยู่ใน string หลัก ไหม

```
alert( "Widget with id".includes("Widget") ); // true
```

```
alert( "Hello".includes("Bye") ); // false
```

# 2. ข้อความ (String)

## 2.10. includes, startsWith, endsWith

- includes สามารถกำหนดได้ด้วยว่าให้เริ่มค้นหาตั้งแต่ index ไหน

```
alert( "Widget".includes("id") ); // true  
alert( "Widget".includes("id", 3) ); // false, เริ่มเช็คตั้งแต่ index 3
```



## 2. ข้อความ (String)

### 2.10. includes, startsWith, endsWith

- startsWith คือการเช็คค่า string ดังกล่าวเริ่มด้วย substring นั้นไหม
- endsWith คือการเช็คค่า string ดังกล่าวจบด้วย substring นั้นไหม

```
alert( "Widget".startsWith("Wid") ); // true, "Widget" เริ่มด้วย "Wid"  
alert( "Widget".endsWith("get") ); // true, "Widget" จบด้วย "get"
```

# 2. ข้อความ (String)

## 2.11. แบบฝึกหัด

1. เขียนฟังก์ชัน `ucFirst(string)` โดยทำคืนค่าเป็น string เดิม แต่ตัวแรกของ string กลายเป็นพิมพ์ใหญ่

# 2. ข้อความ (String)

## 2.11. แบบฝึกหัด

2. เขียนฟังก์ชันที่ checkSpam โดยถ้าข้อความดังกล่าวมีคำว่า “xxx” หรือ “viagra” ให้คืนค่าเป็น true ถ้าไม่มีให้คืนค่าเป็น false

# 2. ข้อความ (String)

## 2.11. แบบฝึกหัด

3. เขียนฟังก์ชันที่ `truncate(str, maxlength)` โดยฟังก์ชันดังกล่าวจะเช็คค่า string ที่ถูกส่งเข้ามามีความยาวเกิน `maxlength` ไหม ถ้าเกินให้แทน ข้อความต่อจากนั้นด้วย “...”

```
truncate("What I'd like to tell on this topic is:", 20) = "What I'd like to te..."
```

```
truncate("Hi everyone!", 20) = "Hi everyone!"
```

## 2. ข้อความ (String)

### 2.11. แบบฝึกหัด

4. เขียนฟังก์ชันที่ `extractCurrencyValue(string, rate)` โดยที่ฟังก์ชันดังกล่าวจะแปลง `string` ที่เป็นค่าเงิน dollar ให้เป็น `number` ที่มีค่าเป็นเงินบาทไทย โดยอ้างอิง `rate` จาก `parameters` ตัวที่สอง ที่ส่งมาให้

```
alert( extractCurrencyValue('$120', 30.5) === 3660 ); // true
```



Software Park Thailand  
</Code Camp>



# หัวข้อ

- ตัวเลข
- ข้อความ
- **Array**
- Methods ของ Array
- Iterable

- Map และ Set
- Keys, Values, และ Entities



# Array



# 3. Array

## 3.1. Array คืออะไร

- Array เป็นประเภทของข้อมูลที่ทำให้เราเก็บ ชุดของข้อมูล ได้
- วิธีประกาศ Array มีสองวิธี

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

# 3. Array

## 3.1. Array คืออะไร - ตัวอย่าง

- Array ที่เก็บชื่อของผลไม้ที่เป็น String

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

# 3. Array

## 3.2. 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
```

# 3. Array

## 3.3. การแทนค่าข้อมูลใน Array - ตัวอย่าง

- วิธีแทนค่าข้อมูลใน Array คือ
- <ชื่อของ Array>[<index>] = <ค่าที่ต้องการจะแทน>



```
fruits[2] = 'Pear'; // now ["Apple", "Orange", "Pear"]
```

# 3. Array

## 3.4. การเพิ่มข้อมูลใน Array - ตัวอย่าง

- วิธีเพิ่มข้อมูลใน Array คือ
- <ชื่อของ Array>[<new index>] = <ค่าที่ต้องการจะเพิ่ม>



```
fruits[3] = 'Mongo'; // now ["Apple", "Orange", "Pear", "Mongo"]
```

# 3. Array

## 3.5. การหาความยาวของ Array - ตัวอย่าง

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

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

# 3. Array

## 3.6. Array สามารถเก็บข้อมูลได้หลายประเภท

- Array ไม่ได้เก็บข้อมูลได้เพียงประเภทเดียว แต่สามารถใส่ ข้อมูลกี่ประเภทก็ได้ ใน 1 Array

```
// value ใน Array ไม่ใช่ประเภทเดียวกัน
let arr = [ 'Apple', { name: 'John' }, true, function() { alert('hello'); } ];

// ตัว array index ที่ 1 แล้ว เรียก property name ออกมา
alert( arr[1].name ); // John

// ตัว array index ที่ 3 แล้ว เรียกใช้ function (เนื่องจาก array ใน index ที่ 3 เป็น function)
arr[3](); // hello
```

# 3. Array

## 3.7. Methods ของ Array

- array.push(item) คือการเพิ่มสมาชิกต่อท้าย array
- array.shift() คือการดึงสมาชิกข้างหน้าสุดมาใช้ และเลื่อนสมาชิกที่เหลือมาแทนตัวที่ถูกดึงออกไป

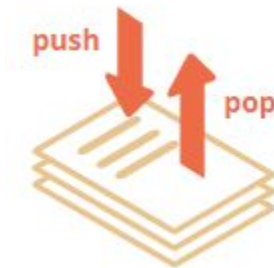




# 3. Array

## 3.7. Methods ของ Array

- `array.push(item)` คือการเพิ่มสมาชิกต่อท้าย array
- `array.pop()` คือการดึงสมาชิกข้างหลังสุดมาใช้ และลบสมาชิกตัวนั้น ออกไป



# 3. Array

## 3.7. Methods ของ Array

- `array.unshift(item)` คือการเพิ่มสมาชิกข้างหน้าสุดของ array

# 3. Array

## 3.7. Methods ของ Array

- ตัวอย่างของ pop()

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

alert( fruits.pop() ); // คัดลอกเป็น "Pear" และ ลบ "Pear" ออก

alert( fruits ); // Apple, Orange
```

# 3. Array

## 3.7. Methods ของ Array

- ตัวอย่างของ push(item)

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

fruits.push("Pear");

alert( fruits ); // Apple, Orange, Pear
```

# 3. Array

## 3.7. Methods ของ Array

- ตัวอย่างของ shift()

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

alert( fruits.shift() ); // remove Apple and alert it

alert( fruits ); // Orange, Pear
```

# 3. Array

## 3.7. Methods ของ Array

- ตัวอย่างของ unshift(item)

```
let fruits = ["Orange", "Pear"];

fruits.unshift('Apple');

alert( fruits ); // Apple, Orange, Pear
```

# 3. Array

## 3.7. Methods ของ Array

- `push(element)` และ `unshift(item)` สามารถเพิ่มสมาชิกหลาย ๆ ตัวได้ภายในทีเดียว

```
let fruits = ["Apple"];

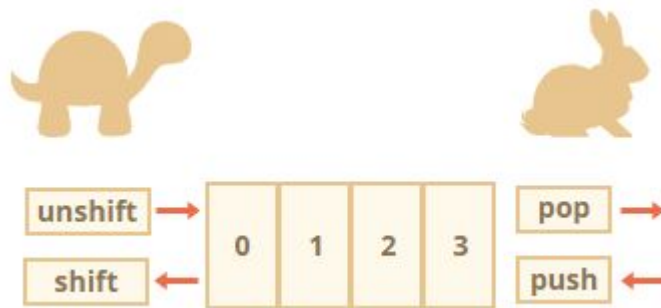
fruits.push("Orange", "Peach");
fruits.unshift("Pineapple", "Lemon");

// ["Pineapple", "Lemon", "Apple", "Orange", "Peach"]
alert( fruits );
```

# 3. Array

## 3.7. Methods ของ Array

- push/pop จะเร็วกว่า shift/unshift

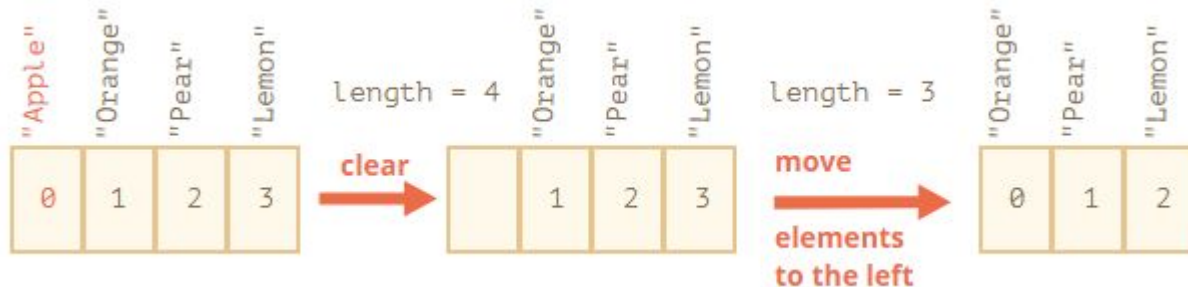




# 3. Array

## 3.7. Methods ของ Array

- ตัวอย่างของ shift()



# 3. Array

## 3.7. Methods ของ Array

- ตัวอย่างของ pop()



# 3. Array

## 3.8. Loop กับ Array

- ตัวอย่าง

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

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

# 3. Array

## 3.8. Loop กับ Array

- ตัวอย่าง - แบบใหม่ for each

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

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

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

for (let fruit of fruits) {
  alert( fruit );
}
```

# 3. Array

## 3.8. Loop กับ Array

- ตัวอย่าง - แบบ Object

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

for (let key in arr) {
  alert( arr[key] ); // Apple, Orange, Pear
}
```

# 3. Array

## 3.9. Array แบบ Object

- Array สามารถประกาศแบบ Object ได้

```
let arr = new Array("Apple", "Pear", "etc");
```

# 3. Array

## 3.10. Array หลายมิติ

- Array สามารถประกาศแบบหลายมิติได้

```
let matrix = [  
  [1, 2, 3],  
  [4, 5, 6],  
  [7, 8, 9]  
];  
  
alert( matrix[1][1] ); // 5, ตัวกลาง
```

# 3. Array

## 3.11. แบบฝึกหัด

### 1. ผลลัพธ์ของความยาว array คืออะไร

```
let fruits = ["Apples", "Pear", "Orange"];

let shoppingCart = fruits;
shoppingCart.push("Banana");

alert( fruits.length ); // ?
```



# 3. Array

## 3.11. แบบฝึกหัด

### 2. ให้ทำตามขั้นตอนต่อไปนี้

- a. สร้าง array ชื่อ styles ที่มี items ชื่อ “Jazz” และ “Blues”
- b. เพิ่ม “Rock-n-Roll” ต่อท้าย
- c. นำค่า Classics ไปทับค่าตรงกลางของ Array
- d. นำ items ตัวแรกออกมาและลบ items ตัวนั้นออกจาก array
- e. เพิ่ม “Rap” และ “Reggae” ไปข้างหน้าของ Array

# 3. Array

## 3.11. แบบฝึกหัด

### 2. ให้ทำตามขั้นตอนต่อไปนี้

- a. สร้าง array ชื่อ styles ที่มี items ชื่อ “Jazz” และ “Blues”
- b. เพิ่ม “Rock-n-Roll” ต่อท้าย
- c. นำค่า Classics ไปทับค่าตรงกลางของ Array
- d. นำ items ตัวแรกออกมาและลบ items ตัวนั้นออกจาก array
- e. เพิ่ม “Rap” และ “Reggae” ไปข้างหน้าของ Array

# 3. Array

## 3.11. แบบฝึกหัด

3. เขียนฟังก์ชัน `sumInput()` ที่
  - a. ใช้ prompt รับ value มาเก็บใน array
  - b. หยุดถามเมื่อเจอค่าที่ไม่ใช่ ตัวเลข
  - c. คำนวณผลรวมของตัวเลขทั้งหมดใน Array

# 3. Array

## 3.11. แบบฝึกหัด

### 4. Maximal contiguous subarray (\*\*Optional\*\*)

ให้เขียนฟังก์ชัน getMaxSubSum(arr) ที่ return ผลรวมของ subarray ที่มากที่สุดที่ติดกัน

```
getMaxSubSum([-1, 2, 3, -9]) == 5 (the sum of highlighted items)
```

```
getMaxSubSum([2, -1, 2, 3, -9]) == 6
```

```
getMaxSubSum([-1, 2, 3, -9, 11]) == 11
```

```
getMaxSubSum([-2, -1, 1, 2]) == 3
```

```
getMaxSubSum([100, -9, 2, -3, 5]) == 100
```

```
getMaxSubSum([1, 2, 3]) == 6 (take all)
```



Software Park Thailand  
</Code Camp>



# หัวข้อ

- ตัวเลข
- ข้อความ
- Array
- **Methods ของ Array**
- Iterable

- Map และ Set
- Keys, Values, และ Entities



# Methods ของ Array

# 4. Methods ของ Array

## 4.1. Methods สำหรับ เพิ่ม/ลบ items

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



# 4. Methods ของ Array

## 4.2. splice การลบ items ตำแหน่งที่ไม่ใช่ขอบ

- Syntax ของ splice

```
let arr = ["I", "study", "JavaScript"];

arr.splice(1, 1); // เริ่มลบที่ index 1, ลบทั้งหมด 1 element

alert( arr ); // ["I", "JavaScript"]
```

# 4. Methods ของ Array

## 4.2. splice การลบ items ตำแหน่งที่ไม่ใช่ขอบ

- ตัวอย่างการลบ และ แทนที่ด้วย elements ใหม่

```
let arr = ["I", "study", "JavaScript", "right", "now"];

// ลบ 3 elements แรก และแทนที่ด้วย "Let's" กับ "dance"
arr.splice(0, 3, "Let's", "dance");

alert( arr ) // now ["Let's", "dance", "right", "now"]
```

# 4. Methods ของ Array

## 4.2. splice การลบ items ตำแหน่งที่ไม่ใช่ขอบ

- การลบโดยใช้ splice จะได้ค่าของ elements ที่ถูกลบไป

```
let arr = ["I", "study", "JavaScript", "right", "now"];

// ลบ 2 elements แรก
let removed = arr.splice(0, 2);
console.log(removed); <--- ผลลัพธ์คือ array ของ elements ที่ถูกลบ
```

# 4. Methods ของ Array

## 4.2. splice การลบ items ตำแหน่งที่ไม่ใช่ขอบ

- การใช้ splice แทรก elements ไประหว่างการ
- โดยการกำหนด deleteCount เป็น 0

```
let arr = ["I", "study", "JavaScript"];

// จาก index 2
// delete ทั้งหมด 0 ตัว (deleteCount = 0)
// เพิ่ม "complex" และ "language" เข้าไป
arr.splice(2, 0, "complex", "language");

alert( arr ); // "I", "study", "complex", "language", "JavaScript"
```

# 4. Methods ของ Array

## 4.3. slice

- Syntax

```
arr.slice([start], [end])
```

# 4. Methods ของ Array

## 4.3. slice

- Syntax
- ฟังก์ชัน slice จะคืนค่าจาก `arr` เป็น array ที่ประกอบด้วย elements ของ `arr` ตั้งแต่ ตำแหน่งที่ `start` ถึง `end` (index ที่ `end` ไม่เอามา)
- `slice` จะไม่มีผลต่อ `arr` (original array)
- index แบบกลับหลัง(index ตีกลับ)สามารถใช้ได้

```
arr.slice([start], [end])
```

# 4. Methods ของ Array

## 4.3. slice

- ตัวอย่าง

```
let arr = ["t", "e", "s", "t"];

alert( arr.slice(1, 3) ); // e,s (คัดลอกตั้งแต่ 1 ถึง 3)

alert( arr.slice(-2) ); // s,t (คัดลอกตั้งแต่ -2 จนจบ)
```

# 4. Methods ของ Array

## 4.4. concat

- การต่อ array - ใช้เมื่อเราต้องการรวม array ทั้งหลายอันเข้าด้วยกัน
- Syntax

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



# 4. Methods ของ Array

## 4.4. concat

- 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
```

# 4. Methods ของ Array

## 4.5. forEach

- การเรียกใช้ for each แบบใส่ callback
- Syntax

```
arr.forEach(function(item, index, array) {  
    // ...  
});
```

# 4. Methods ของ Array

## 4.5. forEach

- ตัวอย่าง

```
[ "Bilbo", "Gandalf", "Nazgul" ].forEach((item, index, array) => {  
    alert(`${item} is at index ${index} in ${array}`);  
});
```

# 4. Methods ของ Array

## 4.6. find และ findIndex

- ปัญหา
- อยากได้ชื่อของคนที่มี id เท่ากับ 1

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

# 4. Methods ของ Array

## 4.6. find และ findIndex

- วิธีแก้ไข find

```
let users = [  
  {id: 1, name: "John"},  
  {id: 2, name: "Pete"},  
  {id: 3, name: "Mary"}  
];  
  
let user = users.find(item => item.id == 1);  
  
alert(user.name); // John
```

# 4. Methods ของ Array

```
let students = [  
  { id: 1, name: "Mr. A" },  
  { id: 2, name: "Mr. B" },  
  { id: 3, name: "Mr. C" },  
  { id: 4, name: "Mr. D" },  
]  
  
students.find(function(item) {  
  return item.id === 3  
})
```

Input

```
function(item) {  
  return item.id === 3  
}
```

false

```
{ id: 3, name: "Mr. C" }
```

Output

# 4. Methods ของ Array

## 4.6. find และ findIndex

- arr.find(fn)
- syntax

```
let result = arr.find(function(item, index, array) {  
    // ถ้า function return true เมื่อไหร่, item นั้นจะถูก returned ไป  
    ให้ result iteration จะหยุด  
});
```

# 4. Methods ของ Array

## 4.6. find และ findIndex

- findIndex ก็เหมือนกับ find แต่คืนเป็น index แทนการคืน element



# 4. Methods ของ Array

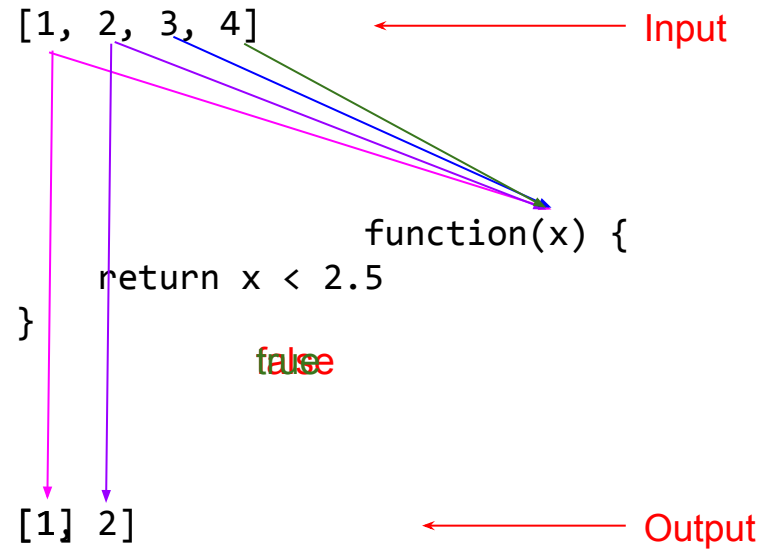
## 4.7. filter

- find จะหาค่าตัวแรกที่เจอแล้วคืนกลับมา
- ถ้าต้องการหลาย ๆ ตัวต้องใช้ filter

```
let users = [  
  {id: 1, name: "John"},  
  {id: 2, name: "Pete"},  
  {id: 3, name: "Mary"}  
];  
  
// ค้นหาตัวที่ id น้อยกว่า 3  
// ผลลัพธ์ที่ได้จะเก็บไว้ใน someUsers  
let someUsers = users.filter(item => item.id < 3);  
  
alert(someUsers.length); // 2
```

# 4. Methods ของ Array

```
[1,2,3,4].filter(function(x) {  
    return x < 2.5  
})
```



# 4. Methods ของ Array

## 4.8. map

- การแปลงร่าง elements ของ array
- ปัญหา อยากหาความยาวของ ชื่อและเก็บใส่ไว้ใน array ใหม่

```
let studentsList = ["Bilbo", "Gandalf", "Nazgul"]
```

# 4. Methods ของ Array

## 4.8. map

- ใช้ map ในการแปลงร่าง elements ที่เป็น string ทั้งหมดของ array เป็น number ที่เก็บค่าความยาวของ string นั้น

```
let studentsList = ["Bilbo", "Gandalf", "Nazgul"]  
  
let ความยาวชื่อ = studentsList.map(item => item.length);  
alert(ความยาวชื่อ); // 5,7,6
```

# 4. Methods ของ Array

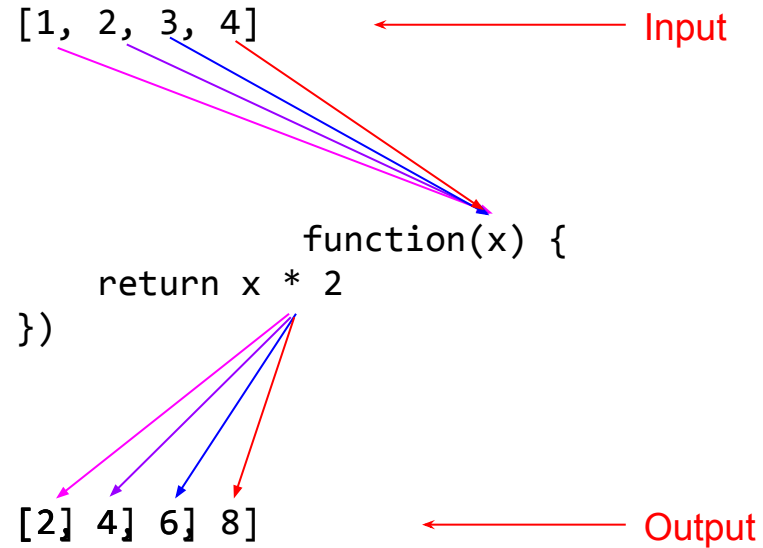
## 4.8. map

- syntax

```
let result = arr.map(function(item, index, array) {  
    // returns the new value instead of item  
});
```

# 4. Methods ของ Array

```
[1,2,3,4].map(function(x) {  
    return x * 2  
})
```



# 4. Methods ของ Array

## 4.9. sort(fn)

- ใช้สำหรับเรียงลำดับของ elements

```
let arr = [ 1, 2, 15 ];  
  
// method sort จะเรียงลำดับ elements ของ array ใหม่  
arr.sort();  
  
alert( arr ); // 1, 15, 2
```

# 4. Methods ของ Array

## 4.9. sort(fn)

- เนื่องจากการเรียงลำดับของ Array จะเรียงแบบ string
- วิธีแก้ เพิ่ม compare function  
เข้าไปให้ sort ใช้งาน

```
function compareNumeric(a,
b) {
    if (a > b) return 1;
    if (a == b) return 0;
    if (a < b) return -1;
}

let arr = [ 1, 2, 15 ];

arr.sort(compareNumeric);
alert(arr); // 1, 2, 15
```



# 4. Methods ของ Array

## 4.9. reverse()

- การทำงานเหมือน sort แต่กลับด้านกัน

```
let arr = [1, 2, 3, 4, 5];  
arr.reverse();  
  
alert( arr ); // 5,4,3,2,1
```

# 4. Methods ของ Array

## 4.10. split และ join

- split จะแบ่ง string ออกเป็น array ด้วยตัวคั่นที่เราระบุ

```
let names = 'Bilbo, Gandalf, Nazgul';

let arr = names.split(', ');

for (let name of arr) {
  alert( `A message to ${name}.` ); // A message to Bilbo (และชื่ออื่น ๆ)
}
```

# 4. Methods ของ Array

## 4.10. split และ join

- split สามารถระบุความยาว array ที่ต้องการได้ด้วย

```
let arr = 'Bilbo, Gandalf, Nazgul, Saruman'.split(', ', 2);  
  
alert(arr); // Bilbo, Gandalf
```

# 4. Methods ของ Array

## 4.10. แบบฝึกหัด

ให้สร้าง array2 จาก array1 ตามที่โจทย์กำหนด โดยใช้ฟังก์ชัน `Array.map()`

1.1    `array1 = [1, 2, 30, 400]`  
       `array2 [2, 4, 60, 800]`

1.2    `array1 = [1, 2, 3, 4]`  
       `array2 ["1", "2", "3", "4"]`

1.3    `array1 = [1, "1", 2, {}]`  
       `array2 ["number", "string", "number", "object"]`

1.4    `array1 = ["apple", "banana", "orange"]`  
       `array2 ["APPLE", "BANANA", "ORANGE"]`

# 4. Methods ของ Array

## 4.10. แบบฝึกหัด

```
1.5 array1 = [  
    { name: "apple", age: 14 },  
    { name: "banana", age: 18 },  
    { name: "watermelon", age: 32 },  
]  
array2 ["apple", "banana", "watermelon"]
```

```
1.6 array1 = [  
    { name: "apple", age: 14 },  
    { name: "banana", age: 18 },  
    { name: "watermelon", age: 32 },  
]  
array2 [14, 18, 32]
```

# 4. Methods ของ Array

## 4.10. แบบฝึกหัด

```
1.7 array1 = [  
    { name: "apple", surname: "London" },  
    { name: "banana", surname: "Bangkok" },  
    { name: "watermelon", surname: "Singapore" },  
]  
array2 ["apple London", "banana Bangkok", "watermelon Singapore"]
```

# 4. Methods ของ Array

## 4.10. แบบฝึกหัด

```
1.8 array1 = [1,3,4,5,6,7,8]
    array2 ["odd", "odd", "even", "odd", "even", "odd", "even"]
```

```
1.9 array1 = [1, -3, 2, 8, -4, 5]
    array2 [1, 3, 2, 8, 4, 5]
```

```
1.10 array1 = [100, 200.25, 300.84, 400.3]
    array2 ["100.00", "200.25", "300.84", "400.30"]
```

# 4. Methods ของ Array

## 4.10. แบบฝึกหัด

```
1.11 array1 = [  
    { name: "apple", birth: "2000-01-01" },  
    { name: "banana", birth: "1990-10-01" },  
    { name: "watermelon", birth: "1985-12-01" },  
]  
array2 [  
    { name: "apple", birth: "2000-01-01", age: 19 },  
    { name: "banana", birth: "1990-10-01", age: 29 },  
    { name: "watermelon", birth: "1985-12-01", age: 33 },  
]
```



# 4. Methods ของ Array

## 4.10. แบบฝึกหัด

```
1.12 array1 = [  
    { name: "apple", birth: "2000-01-01" },  
    { name: "banana", birth: "1990-10-10" },  
    { name: "watermelon", birth: "1985-12-30" },  
]  
array2 [  
    "<tr>  
        <td>apple</td>  
        <td>01 jan 2000</td>  
    </tr>",  
    "<tr> <td>banana</td> <td>10 oct 1990</td> </tr>",  
    "<tr> <td>watermelon</td> <td>30 dec 1985</td> </tr>",  
]
```

# 4. Methods ของ Array

## 4.10. แบบฝึกหัด

ให้สร้าง array2 จาก array1 ตามที่โจทย์กำหนด โดยใช้ฟังก์ชัน `Array.filter()`

2.1 `array1 = [1, 2, 30, 400]`  
`array2 [30, 400] // filter เลขที่มากกว่า 10`

2.2 `array1 = [1, 2, 3, 4]`  
`array2 [1, 3] // filter เลขคี่`

2.3 `array1 = [1, "1", 2, {}]`  
`array2 [1, 2] // filter Number`

2.4 `array1 = ["apple", "banana", "orange", "pineapple", "watermelon"]`  
`array2 ["pineapple", "watermelon"] // filter ตัวอักษร > 6`

# 4. Methods ของ Array

## 4.10. แบบฝึกหัด

```
2.5 array1 = [  
    { name: "apple", age: 14 },  
    { name: "banana", age: 18 },  
    { name: "watermelon", age: 32 },  
    { name: "pineapple", age: 16 },  
    { name: "peach", age: 24 },  
]  
array2 [  
    { name: "apple", age: 14 },  
    { name: "pineapple", age: 16 },  
] // filter age < 18
```

# 4. Methods ของ Array

## 4.10. แบบฝึกหัด

```
2.6 array1 = [  
    { name: "apple", age: 14 },  
    { name: "banana", age: 18 },  
    { name: "watermelon", age: 32 },  
    { name: "pineapple", age: 16 },  
    { name: "peach", age: 24 },  
]  
array2 [  
    { name: "apple", age: 14 },  
    { name: "banana", age: 18 },  
    { name: "pineapple", age: 16 },  
    { name: "peach", age: 24 },  
] // filter ไม่เอาคนที่อายุ 32
```

# 4. Methods ของ Array

## 4.10. แบบฝึกหัด

2.7 array1 = [1, -3, 2, 8, -4, 5]  
array2 [1, 2, 8, 5] // filter เลขบวก

2.8 array1 = [1,3,4,5,6,7,8]  
array2 [3, 6] // filter เลขหาร 3 ลงตัว

2.9 array1 = ["peach", 1, -3, "2", {}, []]  
array2 ["peach", "2"] // filter string

2.10 array1 = ["APPLE", "appLE", "PEACH", "PEach"]  
array2 = ["APPLE", "PEACH"] // filter คำที่เป็นอักษรใหญ่ทุกตัว

# 4. Methods ของ Array

## 4.10. แบบฝึกหัด

```
2.11 array1 = [  
    { name: "apple", birth: "2001-01-01" },  
    { name: "banana", birth: "1990-10-10" },  
    { name: "watermelon", birth: "1985-12-30" },  
    { name: "peach", birth: "2002-10-13" },  
]  
array2 [  
    { name: "banana", birth: "1990-10-10" },  
    { name: "peach", birth: "2002-10-13" },  
] // filter คนเกิดเดือน 10
```

# 4. Methods ของ Array

## 4.10. แบบฝึกหัด

```
2.12 array1 = [  
    { name: "apple", birth: "2001-01-01" },  
    { name: "banana", birth: "1990-10-10" },  
    { name: "watermelon", birth: "1985-12-30" },  
    { name: "peach", birth: "2002-10-13" },  
]  
array2 [  
    { name: "banana", birth: "1990-10-10" },  
    { name: "watermelon", birth: "1985-12-30" },  
] // filter คนเกิดก่อนปี 2000
```



Software Park Thailand  
</Code Camp>





# หัวข้อ

- ตัวเลข
- ข้อความ
- Array
- Methods ของ Array
- Iterable

- Map และ Set
- Keys, Values, และ Entities



Software Park Thailand  
</Code Camp>

# Iterable

# 5. Iterable

## 5.1. Iterable คืออะไร

- Iterable คือ ความสามารถในการเข้าถึงรายตัวได้
- ตัวที่ Iterable ได้ก็สามารถเรียกใช้ for...of ได้

# 5. Iterable

## 5.2. String

- String เป็น Iterable

```
for (let char of "test") {  
  // alert 4 ครั้ง: แต่ละครั้งคือ character แต่ละตัว  
  alert( char ); // t, e, s, t  
}
```

# 5. Iterable

## 5.3. Array

- Array ก็เป็น Iterable เช่นเดียวกัน

```
for (let num of [1, 3, 8, 7]) {  
  // alert 4 ครั้ง: แต่ละครั้งคือ number แต่ละตัว  
  alert(num); // 1, 3, 8, 7  
}
```

# 5. Iterable

## 5.4. Object

- แต่ Object ไม่เป็น Iterable

```
let obj = {  
  name: "Golf",  
  age: 19,  
  skill: "nodeJS",  
  isAdmin: false  
}  
  
for (let element of obj) { // ERROR  
  console.log(element)  
}
```



Software Park Thailand  
</Code Camp>



# หัวข้อ

- ตัวเลข
- ข้อความ
- Array
- Methods ของ Array
- Iterable

- Map และ Set
- Keys, Values, และ Entities





Software Park Thailand  
</Code Camp>

# Map และ Set

# 6. Map และ Set

## 6.1. Map คืออะไร

- Map เหมือนกับ Object ทุกประการ
- แต่มันต่างกันตรงที่ Map ตัว key จะเป็นอะไรก็ได้ ไม่ได้จำกัดแค่ String

# 6. Map และ Set

## 6.2. Methods และ Properties ของ Map

```
new Map() - สร้าง Map ขึ้นมาใหม่  
map.set(key, value) - เก็บ key และ value ลง Map  
map.get(key) - คืนค่า value ของ key ใน Map, undefined ถ้า key นั้นไม่มีอยู่ใน Map  
map.has(key) - เป็น true ถ้า Map นั้นมี key  
map.delete(key) - ลบ value ที่มี key ออกจาก Map  
map.clear() - ลบทุกอย่างออกจาก Map  
map.size - คืนค่าเป็นขนาดของ Map
```

# 6. Map และ Set

## 6.2. Methods และ Properties ของ Map

### - ตัวอย่าง

```
let map = new Map();

map.set('1', 'str1'); // a string key
map.set(1, 'num1');   // a numeric key
map.set(true, 'bool1'); // a boolean key

// ถ้าเป็น Object มันจะแปลง key เป็น string ทั้งหมด
// Map จะเก็บค่าเป็นประเภทนั้นไว้เลย
alert( map.get(1) ); // 'num1'
alert( map.get('1') ); // 'str1'

alert( map.size ); // 3
```

# 6. Map และ Set

## 6.2. Methods และ Properties ของ Map

### - ตัวอย่าง

```
let map = new Map();

map.set('1', 'str1'); // a string key
map.set(1, 'num1');   // a numeric key
map.set(true, 'bool1'); // a boolean key

// ถ้าเป็น Object มันจะแปลง key เป็น string ทั้งหมด
// Map จะเก็บค่าเป็นประเภทนั้นไว้เลย
alert( map.get(1) ); // 'num1'
alert( map.get('1') ); // 'str1'

alert( map.size ); // 3
```

# 6. Map และ Set

## 6.3. การ set properties Map

- \*\*\*\* ห้าม \*\*\*\*
- เพิ่ม Entities โดยใช้ก้ามปู

```
let names = new Map();

names.set('CEO', 'PeeJak');
names.set('COO', "P'Kem");
names.set('CFO', "P'Earth");

names["CTO"] = "P'Oak"; // Wrong

console.log(names.get("CTO")); // undefined
```

# 6. Map และ Set

## 6.3. การ set properties Map

- \*\*\*\* ห้าม \*\*\*\*
- เพิ่ม Entities โดยใช้ก้ามปู

```
let names = new Map();

names.set('CEO', 'PeeJak');
names.set('COO', "P'Kem");
names.set('CFO', "P'Earth");

names.set('CTO', "P'OAK");

console.log(names.get("CTO")); // "P'Oak"
```

# 6. Map และ Set

## 6.4. key ของ Map เป็น Object ได้

- เราสามารถ set key ของ Map เป็น Object ได้

```
let john = { name: "John" };

let visitsCountMap = new Map();

visitsCountMap.set(john, 123);

alert( visitsCountMap.get(john) ); // 123
```



# 6. Map และ Set

## 6.5. Iteration ของ Map

```
map.keys() - คืค่า keys ที่ Iterable  
map.values() - คืค่า values ที่ Iterable,  
map.entries() - คืค่า entries ที่ Iterable [key, value]
```

# 6. Map และ Set

## 6.5. Iteration ของ Map

- ตัวอย่าง

```
let recipeMap = new Map([
  ['cucumber', 500],
  ['tomatoes', 350],
  ['onion', 50]
]);

// iterate แต่ละ keys (ผัก)
for (let vegetable of recipeMap.keys()) {
  alert(vegetable); // cucumber, tomatoes, onion
}

// iterate แต่ละ values (จำนวน)
for (let amount of recipeMap.values()) {
  alert(amount); // 500, 350, 50
}

// iterate แต่ละ [key, value] entries
for (let entry of recipeMap) { // เหมือนกับการใช้ of recipeMap.entries()
  alert(entry); // cucumber,500 (และไปเรื่อย ๆ)
}
```

# 6. Map และ Set

## 6.5. Iteration ของ Map

- ตัวอย่าง

```
let recipeMap = new Map([
  ['cucumber', 500],
  ['tomatoes', 350],
  ['onion', 50]
]);

// iterate แต่ละ keys (ผัก)
for (let vegetable of recipeMap.keys()) {
  alert(vegetable); // cucumber, tomatoes, onion
}

// iterate แต่ละ values (จำนวน)
for (let amount of recipeMap.values()) {
  alert(amount); // 500, 350, 50
}

// iterate แต่ละ [key, value] entries
for (let entry of recipeMap) { // เหมือนกับการใช้ of recipeMap.entries()
  alert(entry); // cucumber,500 (และไปเรื่อย ๆ)
}
```

# 6. Map และ Set

## 6.5. Iteration ของ Map

- forEach กับ Map

```
// runs the function for each (key, value) ทุกคู่  
recipeMap.forEach( (value, key, map) => {  
    alert(`${key}: ${value}`); // cucumber: 500 etc  
});
```

# 6. Map และ Set

## 6.6. Object.entries

- เป็นการแปลง Object ให้เป็น Array ของ key-value
- array ของ key-value คืออะไร

```
// Object  
let user = {  
  name: 'sonter',  
  age: 19,  
  skill: "NodeJS"  
}
```

`Object.entries(user);`

```
// Array ของ key-value  
let user = [  
  ['name', 'sonter'],  
  ['age', 19],  
  ['skill', "NodeJS"]  
]
```

# 6. Map และ Set

## 6.6. Object.entries

- การจะแปลง Object เป็น Map ต้องใช้ Object.entries
- เพื่อแปลง Object ให้เป็น array ของ key-value และส่งลงไป  
ให้ Constructor ของ Map

```
// array ของ [key, value]
let map = new Map([
  ['1', 'str1'],
  [1, 'num1'],
  [true, 'bool1']
]);

alert( map.get('1') ); // str1
```

# 6. Map และ Set

## 6.6. Object.entries - ตัวอย่าง

- เนื่องจากตัว Map จะรับ Array ของ key-value เท่านั้น เราจะต้องแปลง Object เป็น Array ของ key-value ด้วย Object.entries เสียก่อน

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

```
[ ["name", "John"], ["age", 30] ]
```

```
let map = new Map(Object.entries(obj));  
  
alert( map.get('name') ); // John
```

# 6. Map และ Set

## 6.7. Object.fromEntries

- การแปลง array ของ key-value เป็น Object
- เป็นกระบวนการ ตรงกันข้าม กับ Object.entries

```
// Object  
let user = {  
  name: 'sonter',  
  age: 19,  
  skill: "NodeJS"  
}
```

Object.entries(user);

Object.fromEntries(user);

```
// Array ของ key-value  
let user = [  
  ['name', 'sonter'],  
  ['age', 19],  
  ['skill', "NodeJS"]  
]
```



# 6. Map และ Set

## 6.7. Object.fromEntries

- ตัวอย่าง

```
let prices = Object.fromEntries([
  ['banana', 1],
  ['orange', 2],
  ['meat', 4]
]);

// ตอนนี้ prices = { banana: 1, orange: 2, meat: 4 }

alert(prices.orange); //
```

# 6. Map และ Set

## 6.7. Object.fromEntries

- ตัวอย่าง แปลง Map เป็น Object

```
let map = new Map();
map.set('banana', 1);
map.set('orange', 2);
map.set('meat', 4);

let obj = Object.fromEntries(map.entries()); // สร้าง plain object (*)

// เรียบร้อย!
// obj = { banana: 1, orange: 2, meat: 4 }

alert(obj.orange); // 2
```

# 6. Map และ Set

## 6.8. Set คืออะไร

- Set เป็น Collection ชนิดพิเศษที่ เก็บเฉพาะ values ไม่มี key
- Methods ที่สำคัญของ Set

```
new Set(iterable) - สร้าง set ขึ้นมาใหม่, และถ้ามี Iterable ใส่เข้ามา(ส่วนใหญ่จะเป็น Array), มันจะ copy value ใส่ set ให้
set.add(value)    - เพิ่ม value เข้าไปใน set และคืนค่าเป็น set นั้นออกมา
set.delete(value) - ลบ value ใน set, คืนค่าเป็น true ถ้ามีให้ลบ
set.has(value)    - คืนค่าเป็น true ถ้ามี value นั้นอยู่ใน set.
set.clear()       - การเคลียร์ value ทั้งหมดออกจาก set.
set.size          - จำนวนสมาชิกของ set.
```

# 6. Map และ Set

## 6.8. Set คืออะไร

- Set จะ ไม่เก็บค่าที่ซ้ำ (unique)
- ถ้าเรา เพิ่มค่าที่ซ้ำ เข้าไป ค่านั้นจะไม่ถูกเพิ่ม

# 6. Map และ Set

## 6.8. Set คืออะไร

- ตัวอย่าง

```
let set = new Set();

let john = { name: "John" };
let pete = { name: "Pete" };
let mary = { name: "Mary" };

// visits, some users come multiple times
set.add(john);
set.add(pete);
set.add(mary);
set.add(john);
set.add(mary);

// เนื่องจาก set เก็บแค่ค่าที่ unique
alert( set.size ); // 3

for (let user of set) {
    alert(user.name); // John (then Pete and Mary)
}
```

# 6. Map และ Set

## 6.9. Iteration กับ Set

- forEach และ for...of จะให้ค่าเหมือนกัน

```
let set = new Set(["oranges", "apples", "bananas"]);

for (let value of set) alert(value);

// เหมือนกับ forEach:
set.forEach((value, valueAgain, set) => {
  alert(value);
});
```

# 6. Map และ Set

## 6.9. Iteration กับ Set

- เนื่องจาก set มีแค่ value เท่านั้น set.key() หรือ set.values() ก็จะให้ value ออกมา

```
set.keys()      - คืนค่าเป็น an iterable object สำหรับ values
set.values()    - เหมือนกับ set.keys(), มีไว้สำหรับใช้กับ with Map
set.entries()   - returns an iterable object สำหรับ entries
[value, value], มีไว้สำหรับใช้กับ with Map
```

# 6. Map และ Set

## 6.10. แบบฝึกหัด

### 1. ให้ arr เป็น Array

สร้าง function ชื่อ unique(arr) ให้คืนค่าเป็น unique items ของ arr

```
function unique(arr) {  
  /* your code */  
}  
  
let values = ["Hare", "Krishna", "Hare", "Krishna", "Krishna", "Krishna", "Hare", "Hare", ":-0" ];  
  
alert( unique(values) ); // Hare, Krishna, :-0
```



# 6. Map และ Set

## 6.10. แบบฝึกหัด

2. Anagram เป็นตัวอักษรที่มีจำนวนตัวอักษรแต่ละตัวที่เท่ากัน แต่เรียงไม่เหมือนกัน ( Optional )

```
nap - pan  
ear - are - era  
cheaters - hectares - teachers
```

```
let arr = ["nap", "teachers", "cheaters", "PAN", "ear", "era", "hectares"];  
  
alert( aclean(arr) ); // "nap,teachers,ear" or "PAN,cheaters,era"
```

# 6. Map และ Set

## 6.10. แบบฝึกหัด

- เราได้ array จาก map.keys() แต่ไม่สามารถใช้ push ได้  
เราจะทำยังไงให้ keys.push สามารถทำงานได้

```
let map = new Map();

map.set("name", "John");

let keys = map.keys();

// Error: keys.push is not a function
keys.push("more");
```



Software Park Thailand  
</Code Camp>



# หัวข้อ

- ตัวเลข
- ข้อความ
- Array
- Methods ของ Array
- Iterable

- Map และ Set
- Keys, Values, และ Entities



# Key, Values and Entities

# 7. Keys, Values และ Entities

## 7.1. Keys, Values และ Entities

- ทั้ง `map.keys()`, `map.values()` และ `map.entities()` นั้นไม่ได้ใช้ได้เฉพาะใน Map เท่านั้น แต่สามารถใช้ได้ใน  
Map, Set, Array

# 7. Keys, Values และ Entities

## 7.2. แบบฝึกหัด

1. กำหนดให้ salaries เป็น Object

ให้เขียนฟังก์ชัน sumSalaries(salaries) ที่คืนค่าเป็นผลรวมของเงินเดือน ถ้า salaries ไม่มีสมาชิก ให้คืนค่าเป็น 0

```
let salaries = {  
  "John": 100,  
  "Pete": 300,  
  "Mary": 250  
};  
  
alert( sumSalaries(salaries) ); // 650
```

# 7. Keys, Values และ Entities

## 7.2. แบบฝึกหัด

2. ให้เขียนฟังก์ชัน `count(obj)` ที่คืนค่าเป็นจำนวน properties ใน object

```
let user = {  
  name: 'John',  
  age: 30  
};  
  
alert( count(user) ); // 2
```



