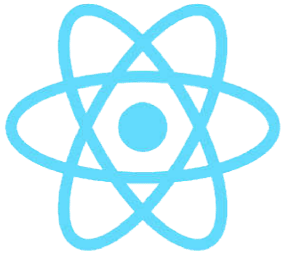


React Native

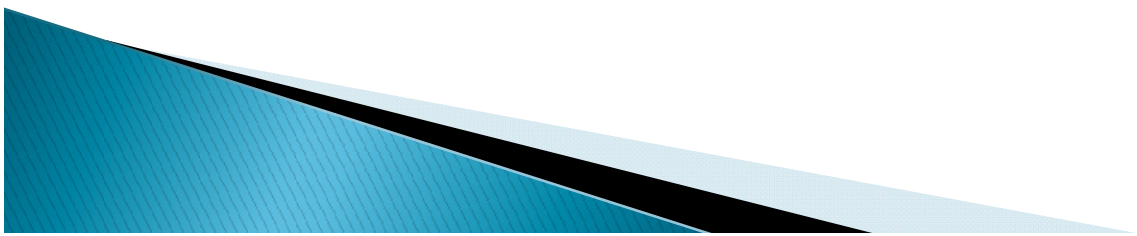
พัฒนา Mobile Apps ด้วย React Native

โค้ชเอก

Codingthailand.com

พื้นฐาน JavaScript สมัยใหม่ (ES6/ES2015 ขึ้นไป)

Variables, Constants and Data Types



Variables and Constants

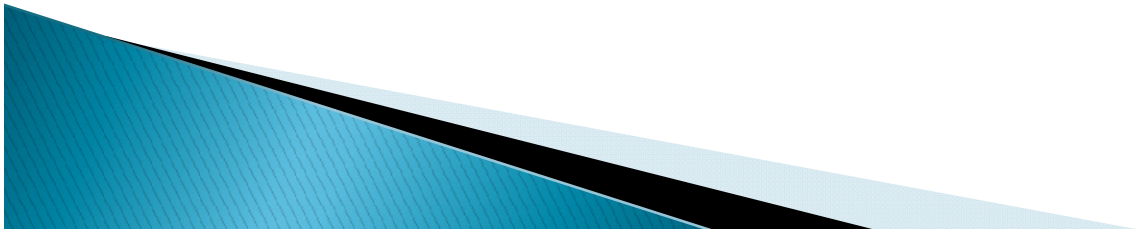
- Variables หรือตัวแปร คือ ชื่อที่เราตั้งขึ้นเพื่อเก็บข้อมูล

```
let currentTempC = 22;  // degrees Celsius
```

- เมื่อเราประกาศตัวแปรแบบไม่ได้กำหนดค่าเริ่มต้น เช่น

```
let targetTempC;  // equivalent to "let targetTempC = undefined";
```

  - ค่าเริ่มต้นของตัวนั้นจะถูกกำหนดให้ undefined



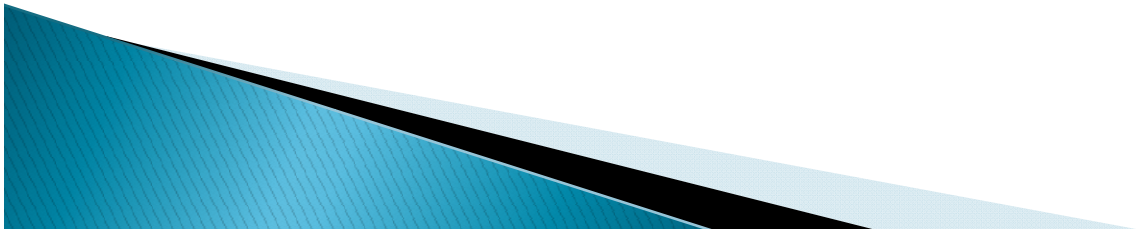
Variables and Constants

- การประกาศตัวแปรหลายตัวในบรรทัดเดียว

```
let targetTempC, room1 = "conference_room_a", room2 = "lobby";
```

- *constant* (new in ES6) ตัวแปรแบบค่าคงที่ ไม่ได้สามารถเปลี่ยนได้ในภายหลัง

```
const ROOM_TEMP_C = 21.5, MAX_TEMP_C = 30;
```



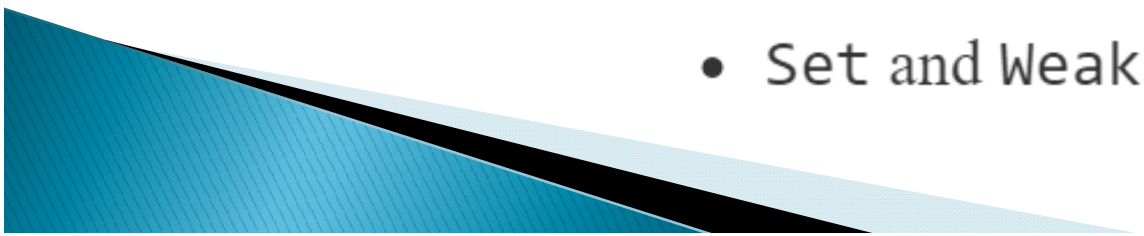
Primitive Types and Objects

- Primitive types คือ ชนิดข้อมูลตายตัว ไม่เปลี่ยนรูปจะเป็นพวกตัวเลข ตัวอักษร นั่นเอง

  - Number
  - String
  - Boolean
  - Null
  - Undefined
  - Symbol

Primitive Types and Objects

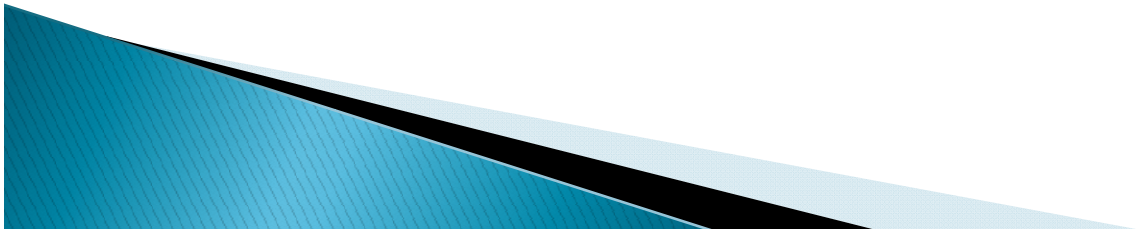
- built-in object types
  - `Array`
  - `Date`
  - `RegExp`
  - `Map` and `WeakMap`
  - `Set` and `WeakSet`



Numbers

- จะเป็นตัวเลข

```
let count = 10;           // integer literal; count is still a double
const blue = 0x0000ff;    // hexadecimal (hex ff = decimal 255)
const umask = 0o0022;     // octal (octal 22 = decimal 18)
const roomTemp = 21.5;    // decimal
const c = 3.0e6;          // exponential (3.0 × 10^6 = 3,000,000)
const e = -1.6e-19;       // exponential (-1.6 × 10^-19 = 0.00000000000000000016)
const inf = Infinity;
const ninf = -Infinity;
const nan = NaN;          // "not a number"
```

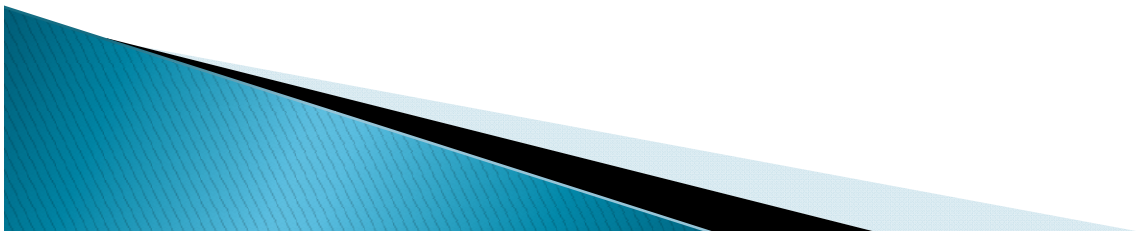


Strings

- ตัวอักษร

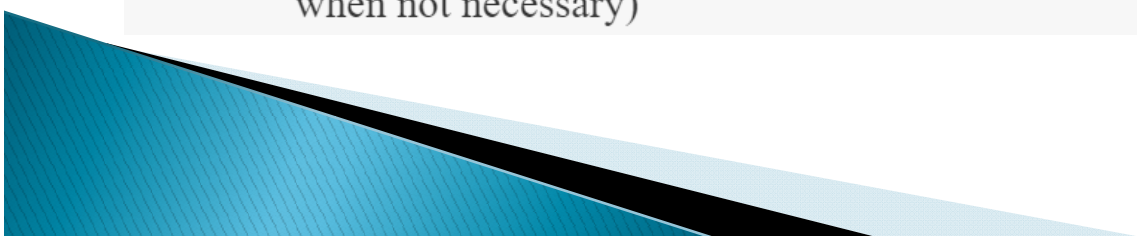
```
const dialog1 = "He looked up and said \"don't do that!\" to Max.";
const dialog2 = 'He looked up and said "don\'t do that!" to Max.';
```

```
const s = "In JavaScript, use \\ as an escape character in strings.";
```



Special Characters

Code	Description	Example
`\n`	Newline (technically a line feed character: ASCII/Unicode 10)	`"Line1\nLine2"`
`\r`	Carriage return (ASCII/Unicode 13)	`"Windows line 1\r\nWindows line 2"`
`\t`	Tab (ASCII/Unicode 9)	`"Speed:\t60kph"`
`\'`	Single quote (note that you can use this even when not necessary)	`"Don\'t"`



Special Characters

\"	Double quote (note that you can use this even when not necessary)	`'Sam said \"hello\".'`
\`	Backtick (or “accent grave”; new in ES6)	`` `New in ES6: \` strings.` ``
\$	Dollar sign (new in ES6)	`` `New in ES6: ${interpolation}` ``
\\	Backslash	`"Use \\\\ to represent \\!"`

Multiline Strings

```
const multiline = "line1\n\
line2";
```

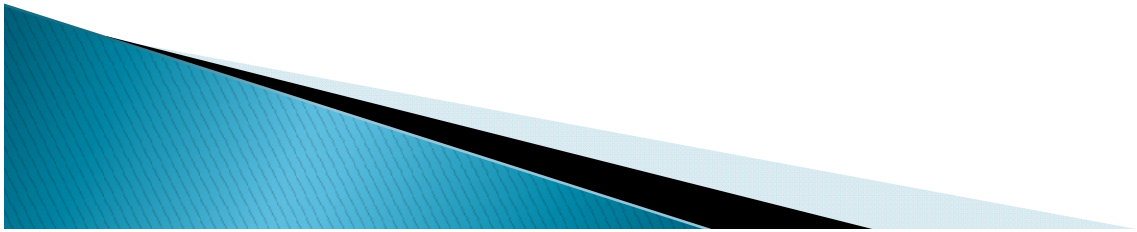
```
const multiline = "line1\n" +
    "line2\n" +
    "line3";
```

```
let info = `
    My Name is ${firstName}
    My Age is ${age}
`;

console.log(info);
```

Booleans

- true and false

```
let heating = true;
let cooling = false;
```

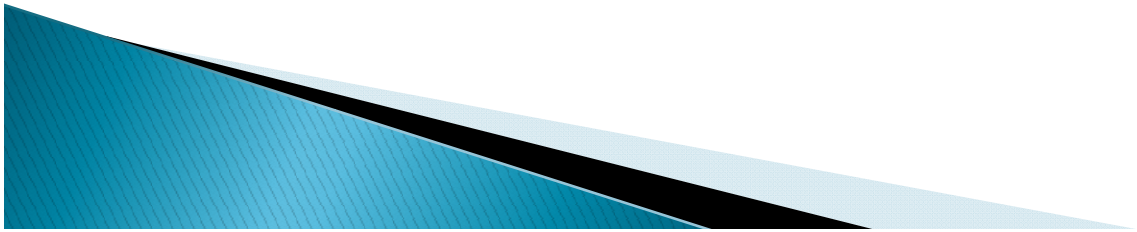


Objects

- Objects เป็นชนิดข้อมูลที่เปลี่ยนแปลงรูปได้ตลอดเวลา เก็บได้หลายค่า
- การประกาศ objects ว่าง

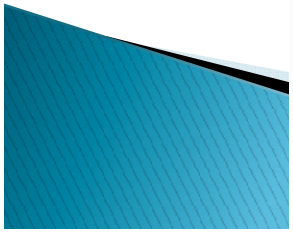
- *Objects ประกอบด้วย properties* (หรือ *members*)

```
const obj = {};
```



Objects

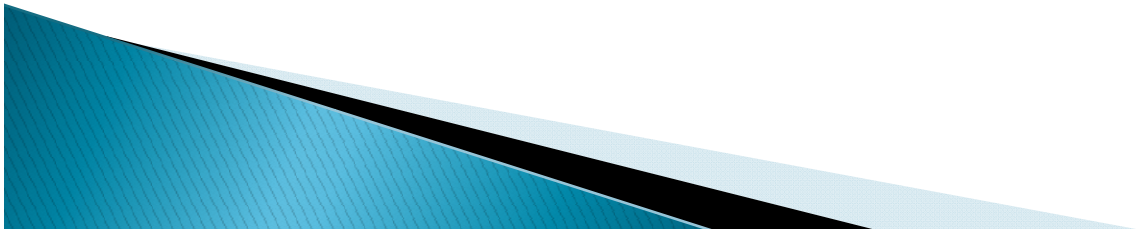
```
const sam1 = {
    name: 'Sam',
    age: 4,
};

const sam2 = { name: 'Sam', age: 4 };   // declaration on one line

const sam3 = {
    name: 'Sam',
    classification: {                   // property values can
        kingdom: 'Anamalia',            // be objects themselves
        phylum: 'Chordata',
        class: 'Mamalia',
        order: 'Carnivoria',
        family: 'Felidae',
        subfaimily: 'Felinae',
        genus: 'Felis',
        species: 'catus',
    },
};
```



Objects

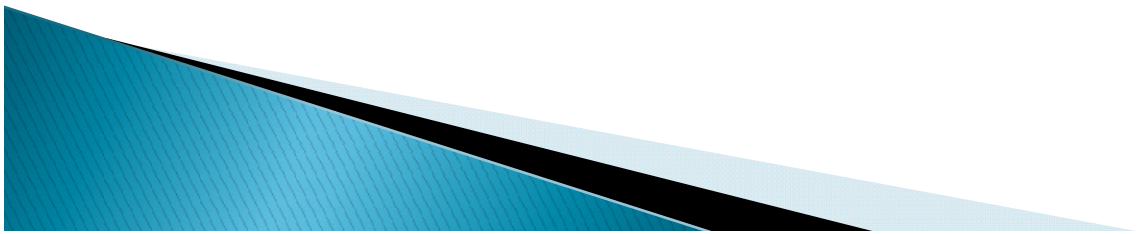
- การเข้าถึงค่าของ objects

```
sam3.classification.family;          // "Felinae"
sam3["classification"].family;       // "Felinae"
sam3.classification["family"];       // "Felinae"
sam3["classification"]["family"];    // "Felinae"
```



Arrays

- ขนาดของ array ไม่ได้ตายตัว (fixed) เราสามารถเพิ่มหรือลบสมาชิกได้ตลอดเวลา
- ชนิดข้อมูลของสมาชิกไม่จำเป็นต้องเหมือนกัน
- สมาชิกตัวแรกของ array เริ่มที่ 0



ตัวอย่างการประกาศ Arrays

```
const a1 = [1, 2, 3, 4];                    // array containing numbers
const a2 = [1, 'two', 3, null];             // array containing mixed types
const a3 = [                                // array on multiple lines
    "What the hammer?  What the chain?",
    "In what furnace was thy brain?",
    "What the anvil?  What dread grasp",
    "Dare its deadly terrors clasp?",
];
const a4 = [                                // array containing objects
    { name: "Ruby", hardness: 9 },
    { name: "Diamond", hardness: 10 },
    { name: "Topaz", hardness: 8 },
];
const a5 = [                                // array containing arrays
    [1, 3, 5],
    [2, 4, 6],
];
```

Arrays

- ใช้ property length สำหรับหาขนาดของ array

```
const arr = ['a', 'b', 'c'];
arr.length;                                    // 3
```

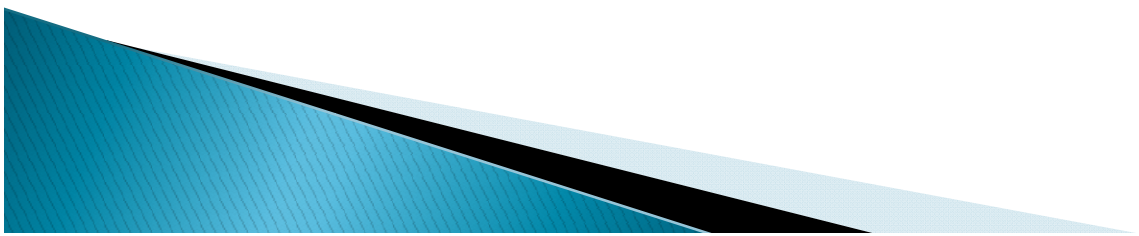
- การเข้าถึงค่าของสมาชิก Array

```
const arr = ['a', 'b', 'c'];

// get the first element:
arr[0];                                        // 'a'

// the index of the last element in arr is arr.length-1:
arr[arr.length - 1];                           // 'c'
```

- การกำหนดค่าให้กับสมาชิกที่มีอยู่แล้ว

```
const arr = [1, 2, 'c', 4, 5];
arr[2] = 3;      // arr is now [1, 2, 3, 4, 5]
```

Control Flow, Expressions and Operators



IF...ELSE STATEMENT

- ถ้าเงื่อนไขเป็นจริงจะรัน statement 1 นอกนั้นจะรัน statement2 (ถ้ามี else)

```
if(condition)
    statement1
[else
    statement2]
```

```
if(new Date().getDay() === 3) {
    totalBet = 1;
} else {
    if(funds === 7) {
        totalBet = funds;
    } else {
        console.log("No superstition here!");
    }
}
```

WHILE STATEMENT

- ขณะที่เงื่อนไขเป็นจริงจะรัน statements ต่อไปเรื่อยๆ

```
while(condition)
    statement
```

```
let funds = 50;     // starting conditions

while(funds > 1 && funds < 100) {
    // place bets

    // roll dice

    // collect winnings
}
```

FOR STATEMENT

- ก่อนรันค่าเริ่มต้นจะทำงาน ถ้าเงื่อนไขยังเป็นจริงอยู่จะรัน statement จากนั้นจะรัน final-expression และตรวจสอบเงื่อนไขว่าเป็นจริงหรือไม่ ถ้าจริงก็รันต่อไป

```
for([initialization]; [condition]; [final-expression])
    statement
```

```
const hand = [];
for(let roll = 0; roll < 3; roll++) {
    hand.push(randFace());
}
```

switch Statements

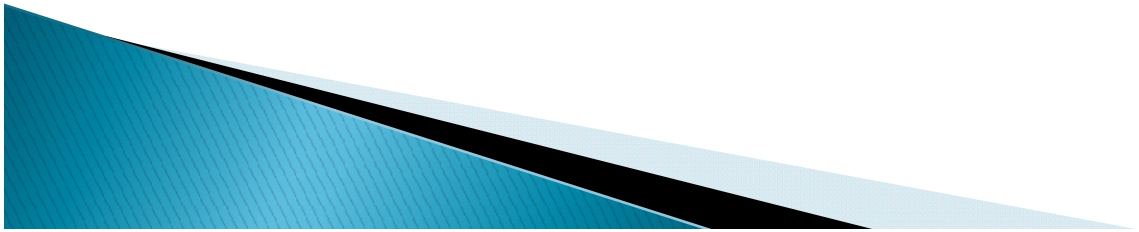
```
switch(expression) {
    case value1:
        // executed when the result of expression matches value1
        [break;]
    case value2:
        // executed when the result of expression matches value2
        [break;]
        ...
    case valueN:
        // executed when the result of expression matches valueN
        [break;]
    default:
        // executed when none of the values match the value of expression
        [break;]
}
```

switch Statements

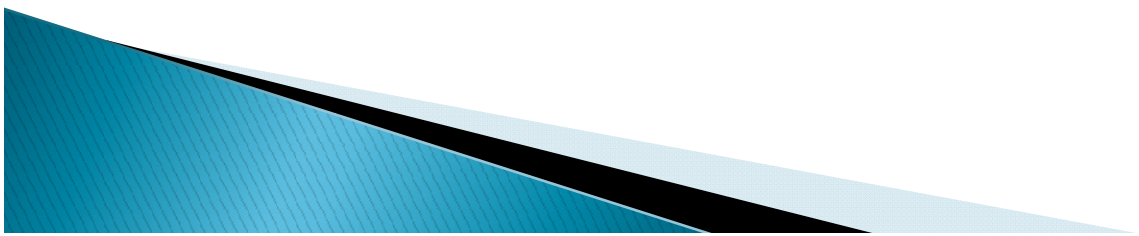
```
switch(totalBet) {
    case 7:
        totalBet = funds;
        break;
    case 11:
        totalBet = 0;
        break;
    case 13:
        totalBet = 0;
        break;
    case 21:
        totalBet = 21;
        break;
}
```

Control Flow Exceptions

- break ออกจาก loop
- continue ข้ามขั้นตอนใน loop
- return ออกจากฟังก์ชันปัจจุบัน
- throw เมื่อเกิดข้อผิดพลาด



JavaScript Operators



Arithmetic Operators

Operator	Description	Example
+	Addition (also string concatenation)	`3 + 2 // 5`
-	Subtraction	`3 - 2 // 1`
/	Division	`3/2 // 1.5`
*	Multiplication	`3*2 // 6`

Arithmetic Operators

%	Remainder	`3%2 // 1`
-	Unary negation	`-x // negative x; if x is 5, -x will be -5`
+	Unary plus	`+x // if x is not a number, this will attempt conversion`
++	Pre-increment	`++x // increments x by one, and evaluates to the new value`
++	Post-increment	`x++ // increments x by one, and evaluates to value of x`
--	Pre-decrement	`--x // decrements x by one, and evaluates to the new value`
--	Post-decrement	`x-- // decrements x by one, and evaluates to value of x before the decrement`

Comparison Operators

- เป็นตัวดำเนินการเปรียบเทียบค่า

```
3 > 5;      // false
3 >= 5;     // false
3 < 5;      // true
3 <= 5;     // true

5 > 5;      // false
5 >= 5;     // true
5 < 5;      // false
5 <= 5;     // true
```

```
const n = 5;
const s = "5";
n === s;
n !== s;
```

Comparing Numbers

- การเปรียบเทียบตัวเลข อย่างแรกที่ควรรู้คือค่า NaN จะไม่เท่ากันกับทุกๆ อย่าง (เป็น false หมด)

String Concatenation

- ใช้เครื่องหมาย + ในการเชื่อม string

```
3 + 5 + "8"          // evaluates to string "88"
"3" + 5 + 8          // evaluates to string "358"
```

Logical Operators

- AND, OR, and NOT

for AND (&&)

x	y	x && y
false	false	false
false	true	false
true	false	false
true	true	true

for OR (||)

x	y	x || y
false	false	false
false	true	true
true	false	true
true	true	true

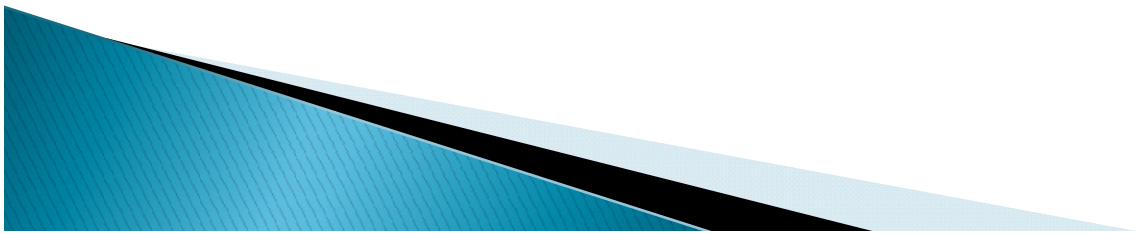
table for NOT (!)

x	!x
false	true
true	false

Function Arguments

- เราสามารถใส่ arguments ให้กับฟังก์ชัน และตั้งพารามิเตอร์ได้

```
function avg(a, b) {
    return (a + b)/2;
}
```



Arrow Notation (Arrow Function)

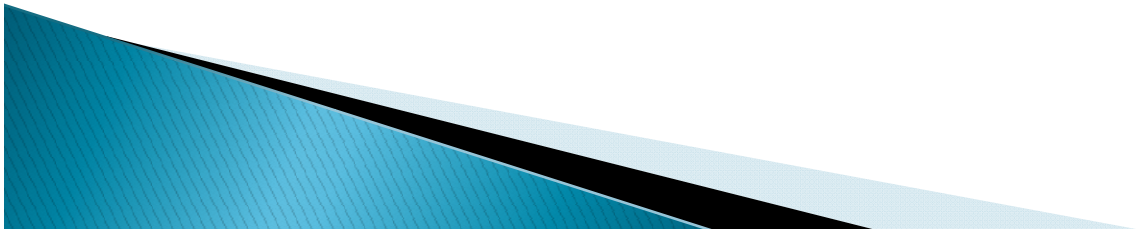
- ES6 introduces a new and welcome syntax called ***arrow notation (fat arrow)***

```
const f1 = function() { return "hello!"; }
```

```
const f2 = function(name) { return `Hello, ${name}!`; }
```

```
const f3 = function(a, b) { return a + b; }
```

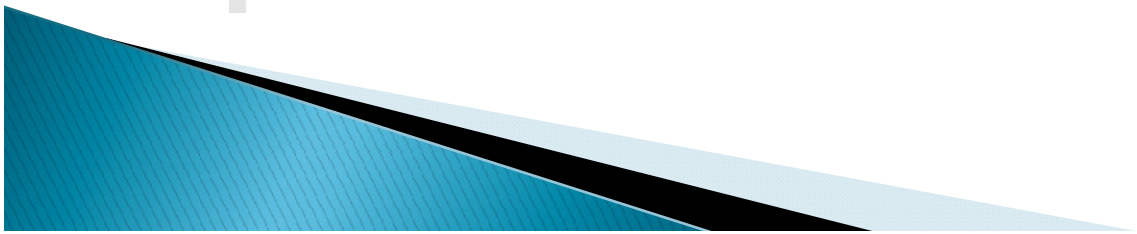
- *ลองแปลงโค้ดด้านบน เป็น arrow function*



Destructuring Assignment

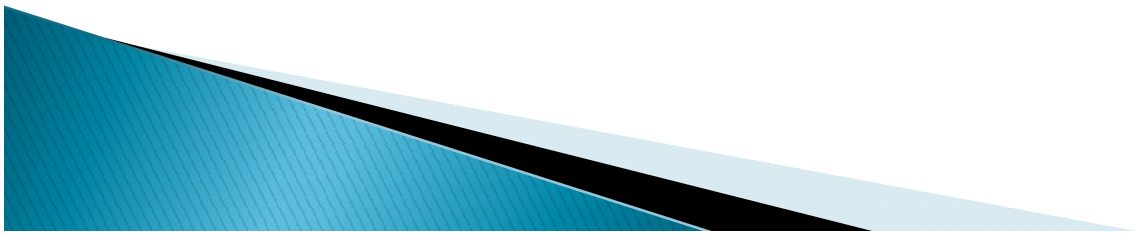
- New in ES6 เราสามารถเปลี่ยน object หรือ array ให้กลับมาเป็นตัวแปรได้ (แยกเป็นรายตัว)

```
// a normal object
const obj = { b: 2, c: 3, d: 4 };

// object destructuring assignment
const {a, b, c} = obj;
console.log(a);
console.log(b);
console.log(c);
console.log(d);
```



Destructuring Arguments

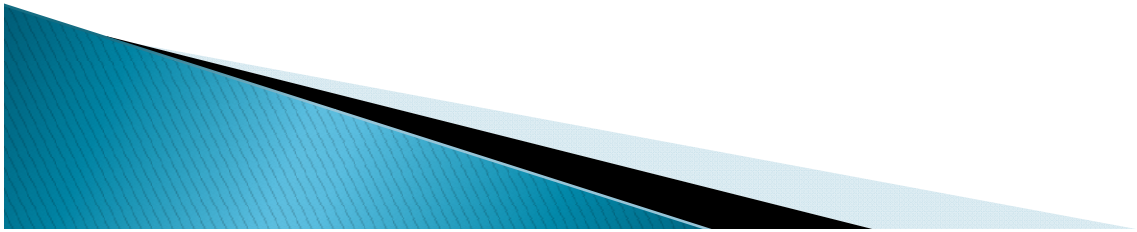
```
function getSentence({ subject, verb, object }) {
    return `${subject} ${verb} ${object}`;
}

const o = {
    subject: "I",
    verb: "love",
    object: "JavaScript",
};

getSentence(o);          // "I love JavaScript"
```



Default Arguments

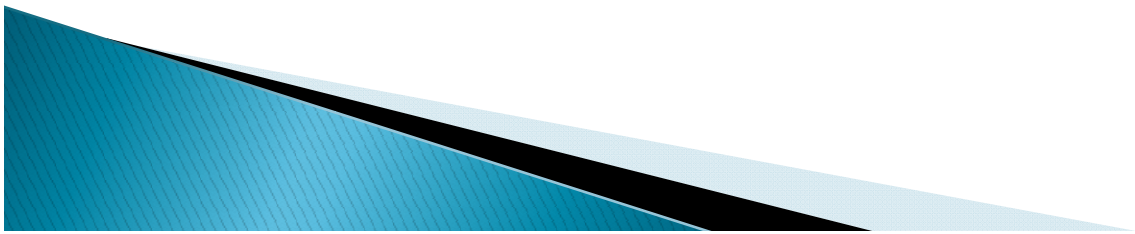
- New in ES6

```
function f(a, b = "default", c = 3) {
   return `${a} - ${b} - ${c}`;
}

f(5, 6, 7);
f(5, 6);
f(5);
f();
```



Spread Operator

```
function myFunction(x, y, z) {
    return x+y+z;
}

var args = [0, 1, 2];

console.log(myFunction(...args));
```



Rest Parameter

แปลง เป็น array

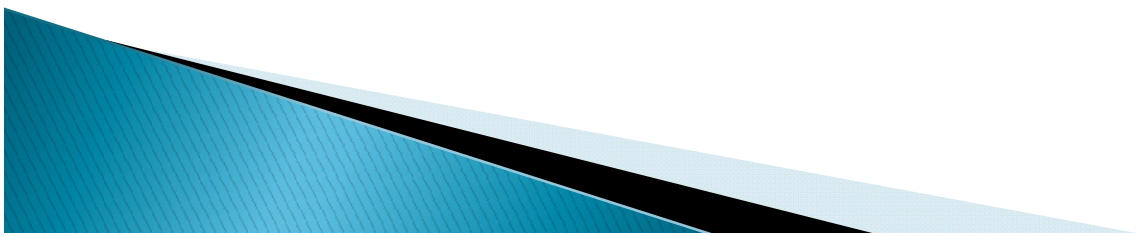
function f(a, b, ...theArgs) {

//

}

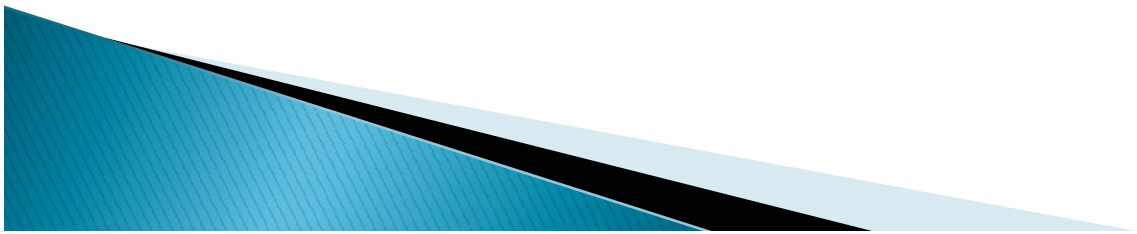
```
function sum(...theArgs)
{
    return theArgs.reduce((previous, current) => {
        return previous + current;
    });
}

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

JavaScript: Arrays and Array Processing



Arrays

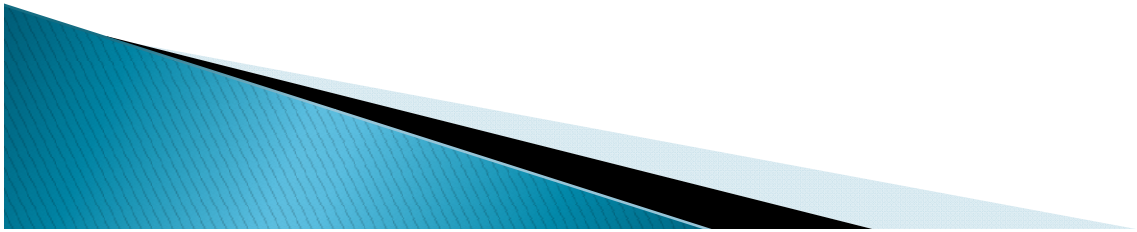
```
// array literals
const arr1 = [1, 2, 3];                                    // array of numbers
const arr2 = ["one", 2, "three"];                          // nonhomogeneous array
const arr3 = [[1, 2, 3], ["one", 2, "three"]];             // array containing arrays
const arr4 = [                                             // nonhomogeneous array
   { name: "Fred", type: "object", luckyNumbers = [5, 7, 13] },
   [
      { name: "Susan", type: "object" },
      { name: "Anthony", type: "object" },
   ],
   1,
   function() { return "arrays can contain functions too"; },
   "three",
];
```



Accessing array elements

- ถ้าอยากเข้าถึงสมาชิกตัวแรก ก็ให้เริ่มต้นที่ index 0

```
var arr = ['this is the first element', 'this is the second element'];
console.log(arr[0]);              // logs 'this is the first element'
console.log(arr[1]);              // logs 'this is the second element'
console.log(arr[arr.length - 1]); // logs 'this is the second element'
```



Loop over an Array

- for

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

- forEach

```
const arr1 = ['apple', 'banana'];

arr1.forEach((item, index, arr) => {
    console.log(index + ' '+ item + '\n');
});
```

- for...of

```
let arr2 = [1,2,3];

for (const myarr of arr2) {
    console.log(myarr);
}
```

สรุป array สำหรับจัดการ content

When you need to...	Use...	In-place or copy
Create a stack (last-in, first-out [LIFO])	push (returns new length), pop	In-place
Create a queue (first-in, first-out [FIFO])	unshift (returns new length), shift	In-place
Add multiple elements at end	concat	Copy
Get subarray	slice	Copy
Add or remove elements at any position	splice	In-place
Cut and replace within an array	copyWithin	In-place
Fill an array	fill	In-place
Reverse an array	reverse	In-place
Sort an array	sort (pass in function for custom sort)	In-place

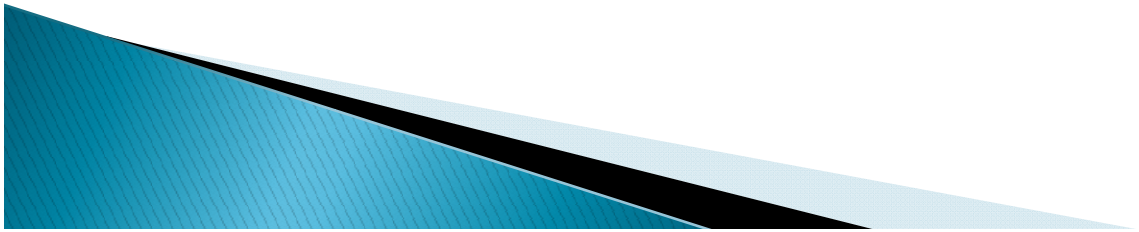
สรุป array เกี่ยวกับการค้นหา

When you want to know/find...	Use...
The index of an item	indexOf (simple values), findIndex (complex values)
The last index of an item	lastIndexOf (simple values)
The item itself	find
Whether the array has an item that matches some criteria	some
Whether all elements in the array matches some criteria	every

Import and Export

- https://developer.mozilla.org/th/docs/Web/JavaScript/Reference/Statements/export

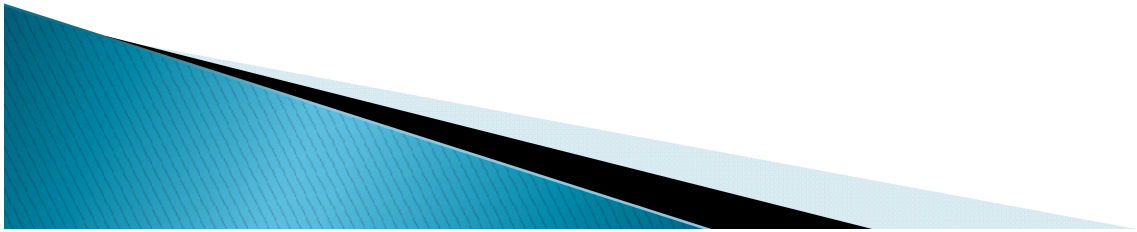
- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/import



JavaScript ES7 Async/await

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/await



The end