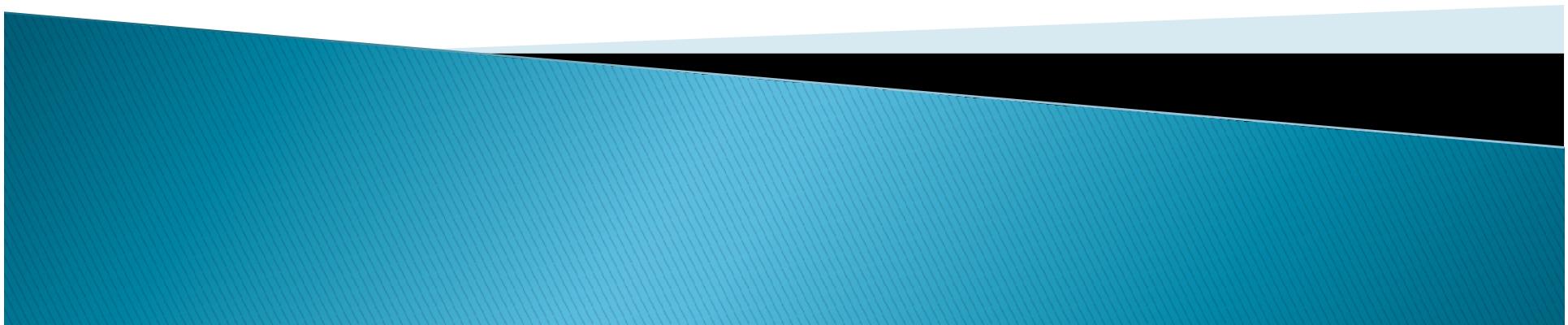


# พัฒนา Modern Web Application ด้วย Node.js



โค้ชเอก  
[Codingthailand.com](http://Codingthailand.com)



# หัวข้อการเรียน

- ▶ ซอฟต์แวร์ที่ต้องติดตั้ง
- ▶ พื้นฐานภาษา JavaScript เวอร์ชันใหม่ ES2015 หรือ ES6 ขึ้นไป



# ซอฟต์แวร์ที่ต้องติดตั้ง

- ▶ Node.js และ npm
  - ▶ <https://nodejs.org/en/>
- ▶ Visual Studio Code (Editor)
  - ▶ <https://code.visualstudio.com/>
- ▶ Git
  - ▶ <https://git-scm.com/>



# เส้นทางการพัฒนาตัวเองสำหรับ Web Developer



[www.codingthailand.com](http://www.codingthailand.com)

Git & GitHub  
(npm/yarn, webpack/parcel,  
nodemon, PM2, Mongoose, mocha etc.)

Node/Front-end Framework

(Meteor/Koa.js/Sails.js/Angular/Vue.js  
React/Hapi.js etc.)

Mobile Framework  
(Ionic 3, React Native etc.)

HTML

CSS

+ SASS

Ja

css framework

(Bootstrap 4 / Bulma etc.)

Docker for Web Developers

JS OOP/TypeScript/  
GraphQL/Design Patterns

Electron

(Build cross platform desktop apps)

Ubuntu/CentOS, CI/CD, AWS, Firebase

# ภาพใหญ่ของการพัฒนา Web App ด้วย Node.js

- ▶ รูปแบบที่ 1 ใช้ไลบรารี JavaScript
  - ▶ Vanilla JavaScript หรือ jQuery + Node.js
- ▶ รูปแบบที่ 2 ใช้ frontend framework สมัยใหม่
  - ▶ Angular หรือ React หรือ Vue.js + Node.js
- ▶ รูปแบบที่ 3 เขียนแยกเป็น backend (APIs) ส่วน frontend เป็นอะไรก็ได้
  - ▶ any + Node.js



# พื้นฐานการใช้งาน npm

- ▶ ดูเวอร์ชันของ npm ใช้คำสั่ง

```
npm -v
```

- ▶ อัปเดตเวอร์ชันของ npm

```
npm install npm@latest -g
```

- ▶ คำสั่ง npm init เพื่อสร้างไฟล์ package.json

- ▶ ติดตั้ง Package โดยการระบุชื่อ

```
npm install <package_name>
```

หรือ npm install <package\_name> --save

หรือ npm i



# พื้นฐานการใช้งาน npm

- ▶ ติดตั้ง Package ทั้งหมด ในไฟล์ package.json ใช้คำสั่ง
- ▶ npm install
  
- ▶ ติดตั้ง npm ได้ที่  
<https://docs.npmjs.com/getting-started/installing-node>
  
- ▶ ติดตั้ง Node.js ได้ที่  
<http://goo.gl/2Lm4xe>



# พื้นฐานการใช้งาน npm

- ▶ Semantic Versioning การระบุตัวเลขเวอร์ชัน สำหรับโปรแกรมเมอร์

CODE STATUS	STAGE	RULE	EXAMPLE #
First Release	New Product	Start with 1.0.0	1.0.0
Bug fixes, other minor changes	Patch Release	Increment the third digit	1.0.1
New Features that don't break existing features	Minor release	Increment the middle digit	1.1.0
Changes that break backward compatibility	Major release	Increment the first digit	2.0.0

- ▶ การระบุเวอร์ชันสำหรับผู้ที่เรียกใช้
- ▶ Patch releases: 1.0 or 1.0.x or ~1.0.4 or 1.0.\*
- ▶ Minor releases: 1 or 1.x or ^1.0.4
- ▶ Major releases: \* or x



# พื้นฐานการใช้งาน npm (คำสั่งอื่นๆที่น่าสนใจ)

- ▶ npm ls -g --depth=0 หรือ npm ls -g --depth=0 --json
- ▶ หรือ npm ll -g --depth=0 (ดูรายละเอียดได้)
  
- ▶ npm config list -l
  
- ▶ npm config set init-author-name "Akenarin Komkoon"
  
- ▶ npm config set save true
  
- ▶ npm home <ชื่อ package> หรือ npm repo <ชื่อ package>
  
- ▶ npm visnup หรือ npm xmas



# ทดลองสร้างโปรเจคใหม่ด้วย npm

- ▶ ทดลองติดตั้ง และใช้งาน nodemon

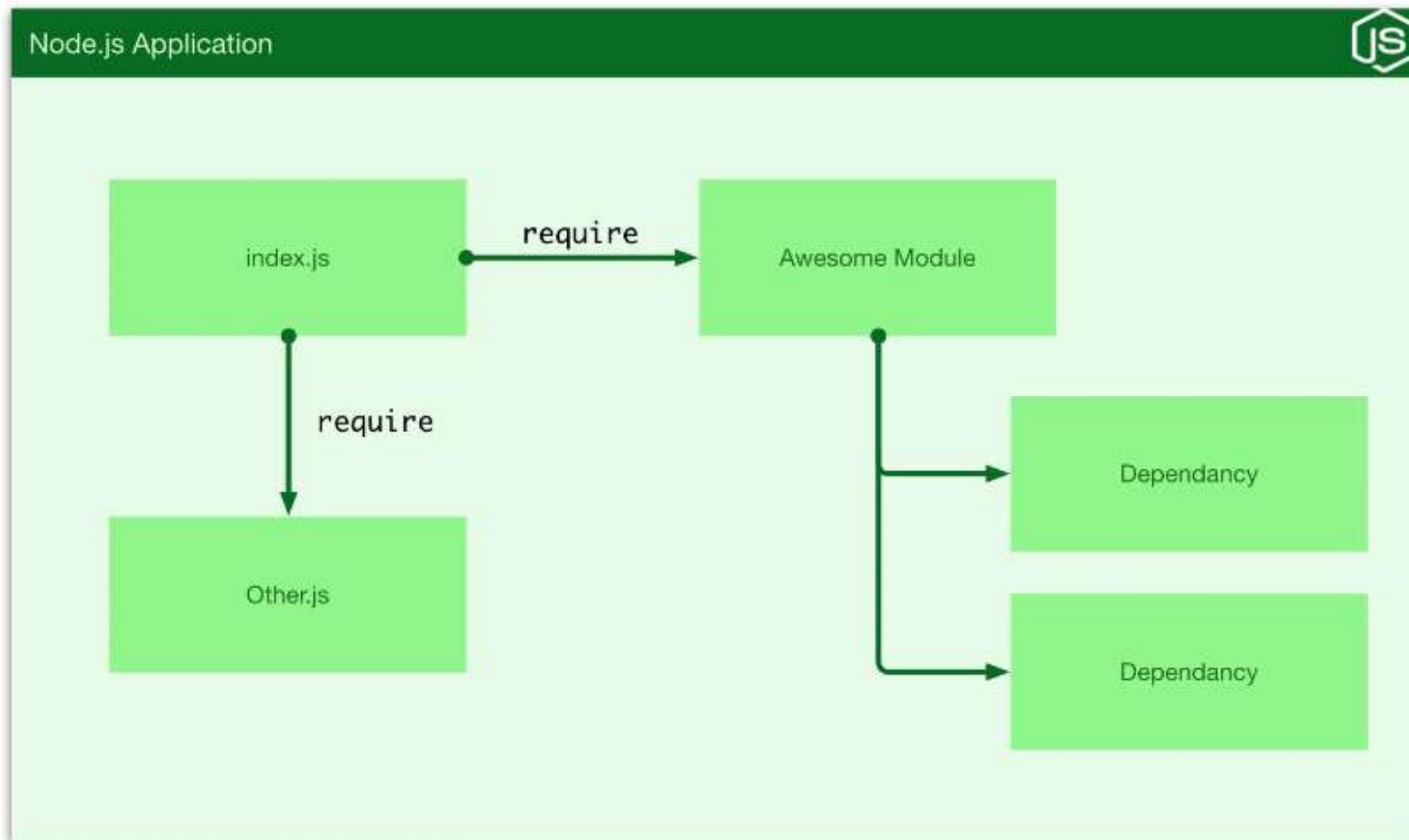
- ▶ ทดลองติดตั้ง และใช้งาน lib

```
npm i seedrandom --save
```

```
npm install moment --save
```

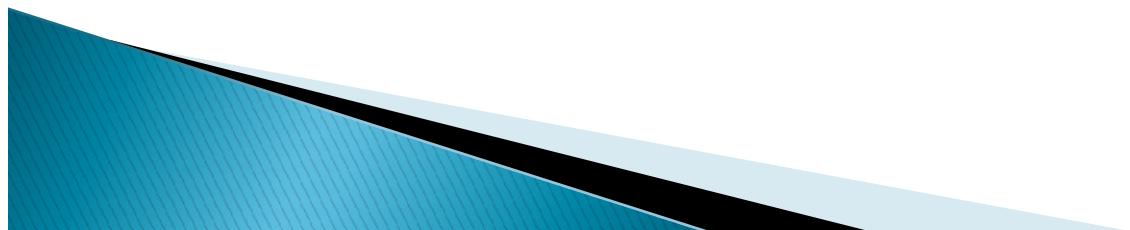


# เราจะใช้งานไลบรารี ไฟล์ หรือ โมดูลได้อย่างไร





พื้นฐาน JavaScript



# 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;
```

# Identifier Names

- ▶ ต้องขึ้นต้นด้วยตัวอักษร หรือ \$ หรือ \_ ก็ได้
- ▶ จะต้องประกอบด้วย ตัวอักษร ตัวเลข dollar sign และ
- ▶ ตัวอักษรประเภท Unicode ก็ตั้งได้เหมือนกัน เช่น Π or ö
- ▶ ต้องตั้งให้ไม่ตรงกับคำส่วนของ js (reserved word)
- ▶ ตั้งชื่อในรูปแบบ Camel case เช่น currentTempC, anIdentifierName



# Primitive Types and Objects

- ▶ Primitive types คือ ชนิดข้อมูลatyตัว ไม่เปลี่ยนรูปจะเป็นพวktัวเลข 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 ( $3.0 \times 10^6 = 3,000,000$ )
const e = -1.6e-19;       // exponential ( $-1.6 \times 10^{-19} = 0.0000000000000000000016$ )
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 'Sam said \"hello\".'  
when not necessary)

---

\` 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;
```

- ▶ New in ES6 are *symbols* เป็นชนิดข้อมูล

## Symbols

```
const RED = Symbol();  
const ORANGE = Symbol("The color of a sunset!");  
RED === ORANGE // false: every symbol is unique
```



# null and undefined

- ▶ เป็นชนิดข้อมูลแบบพิเศษ `null` และ `undefined`
- ▶ ใช้ `undefined` เมื่อเราจงใจที่จะไม่กำหนดค่าให้กับตัวแปรนั้น อย่างลึมว่า เมื่อเราประกาศตัวแปรเฉยๆ ไม่ได้กำหนดค่า ค่า `default` ของตัวแปรนั้นคือ `undefined`
- ▶ ใช้ `null` เมื่อเราไม่แน่ใจหรือไม่รู้ว่าอนาคตตัวแปรนี้จะกำหนดค่าเป็นอะไร

```
let currentTemp;           // implicit value of undefined
const targetTemp = null;    // target temp null -- "not yet known"
currentTemp = 19.5;         // currentTemp now has value
currentTemp = undefined;   // currentTemp appears as if it had never
                           // been initialized; not recommended
```



# 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',  
        subfamily: 'Felinae',  
        genus: 'Felis',  
        species: 'catus',  
    },  
};
```

# Objects

- ▶ การเข้าถึงค่าของ objects

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

- ▶ Objects สามารถมีพิงก์ชันในตัวมันได้

```
sam3.speak = function() { return "Meow!"; };
```

```
sam3.speak();                      // "Meow!"
```

```
delete sam3.classification;          // the whole classification tree is removed  
delete sam3.speak;                  // the speak function is removed
```

# 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 = [
    "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 = [
    { name: "Ruby", hardness: 9 },
    { name: "Diamond", hardness: 10 },
    { name: "Topaz", hardness: 8 },
];
const a5 = [
    [1, 3, 5],
    [2, 4, 6],
];
```

// array containing objects

// array containing arrays



# 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]
```

# Dates

- ▶ the built-in Date object

```
const now = new Date();
now; // example: Thu Aug 20 2015 18:31:26 GMT-0700 (Pacific Daylight Time)

const halloween = new Date(2016, 9, 31); // note that months are
                                         // zero-based: 9=October

const halloweenParty = new Date(2016, 9, 31, 19, 0); // 19:00 = 7:00 pm

halloweenParty.getFullYear();           // 2016
halloweenParty.getMonth();            // 9
halloweenParty.getDate();             // 31
halloweenParty.getDay();              // 1 (Mon; 0=Sun, 1=Mon,...)
halloweenParty.getHours();            // 19
halloweenParty.getMinutes();          // 0
halloweenParty.getSeconds();          // 0
halloweenParty.getMilliseconds();     // 0
```

# อีนๆ

- ▶ Regular Expressions
- ▶ Maps and Sets



# Data Type Conversion การแปลงชนิดข้อมูล

- ▶ Converting to Numbers ให้ใช้ Number object constructor ถ้าแปลงไม่ได้จะคืนค่า NaN

- ▶ built-in `parseInt` or `parseFloat` functions

```
const a = parseInt("16 volts", 10); // the "volts" is ignored, 16 is  
// parsed in base 10  
const b = parseInt("3a", 16); // parse hexadecimal 3a; result is 58  
const c = parseFloat("15.5 kph"); // the "kph" is ignored; parseFloat  
// always assumes base 10
```

# Data Type Conversion การแปลงชนิดข้อมูล

- ▶ บางครั้งหากต้องการแปลง boolean ไปเป็น 1 กับ 0 ให้ใช้ *ternary operator*

```
const b = true;  
const n = b ? 1 : 0;
```

- ▶ Converting to String
- ▶ Object ทุกอย่างใน js ถ้าหากต้องการแปลงเป็น string ให้เรียกใช้ method **toString()** เช่น vatTotal.toString()



# Data Type Conversion การแปลงชนิดข้อมูล

- ▶ ตัวอย่างการแปลงให้เป็น string

```
const n = 33.5;  
n;                                // 33.5 - a number  
const s = n.toString();  
s;                                // "33.5" - a string
```

- ▶ การแปลง array ไปเป็น string ผลที่ได้จะถูกคั่นด้วย ,

```
const arr = [1, true, "hello"];  
arr.toString();                    // "1,true,hello"
```

# Data Type Conversion การแปลงชนิดข้อมูล

## ▶ Converting to Boolean

```
const n = 0;           // "falsy" value
const b1 = !!n;        // false
const b2 = Boolean(n); // false
```



# สรุป

- ▶ ภาษา JavaScript มีชนิดข้อมูล (แบบ primitive) 6 ตัว ได้แก่ string, number, boolean, null, undefined และ object
- ▶ ตัวเลขทุกอย่างใน js เป็น double
- ▶ Arrays เป็นชนิดข้อมูลพิเศษที่สามารถเก็บ type ได้ยืดหยุ่น และสะดาวกมากๆ
- ▶ ชนิดข้อมูลอื่นๆ เช่น dates, maps, sets, and regular expressions เป็นชนิดข้อมูล แบบ object



# 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



# JavaScript Operators



# Arithmetic Operators

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



# 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	<code>x &amp;&amp; y</code>
---	---	-----------------------------

false	false	false
-------	-------	-------

false	true	false
-------	------	-------

true	false	false
------	-------	-------

true	true	true
------	------	------

*for OR (`||`)*

x	y	<code>x    y</code>
---	---	---------------------

false	false	false
-------	-------	-------

false	true	true
-------	------	------

true	false	true
------	-------	------

true	true	true
------	------	------

*table for NOT  
(`!`)*

x	<code>!x</code>
---	-----------------

false	true
-------	------

true	false
------	-------

# Conditional Operator

- ▶ *ternary operator*
- ▶ ເຂີຍນຍ່ອ ໃຊ້ແທນ if...else

```
const doIt = false;  
const result = doIt ? "Did it!" : "Didn't do it.";
```

T

F



# Typeof Operator

- ▶ มีไว้ตรวจสอบชนิดข้อมูล โดยจะคืนค่ามาเป็น string
- ▶ ตรวจสอบชนิดข้อมูลได้ ดังนี้ undefined, null, boolean, number, string, symbol, and object
- ▶ มี bug นิดนึงถ้าเรา typeof null จะไม่ได้ null จะได้ object แทน



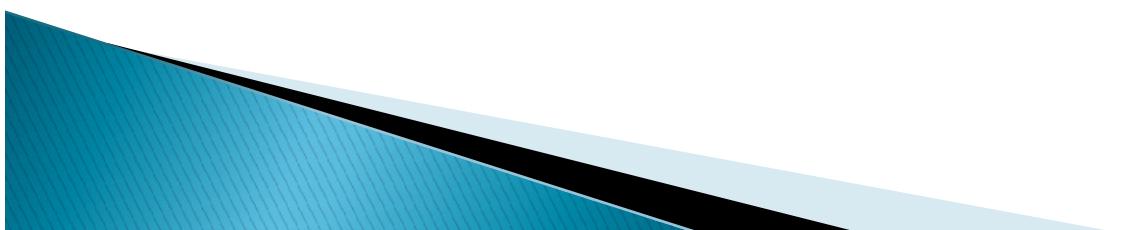
# Typeof Operator

# 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);
```



# JavaScript: Functions and Scope



# Functions

- ▶ เป็นการเขียนโค้ดรวมกลุ่มคำสั่งเพื่อช่วยในการเรียกใช้ได้ง่ายขึ้น ใช้ซ้ำได้ และสามารถเรียกใช้ได้เพียงบรรทัดเดียว

```
function sayHello() {  
    // this is the body; it started with an opening curly brace...  
  
    console.log("Hello world!");  
    console.log("¡Hola mundo!");  
    console.log("Hallo wereld!");  
    console.log("Привет мир!");  
  
    // ...and ends with a closing curly brace  
}
```

sayHello();

# Return Values

- ▶ การ return เป็นการหยุดการทำงานของฟังก์ชันแล้วส่งค่า  
บางอย่างกลับออกไป

```
function getGreeting() {  
    return "Hello world!";  
}
```

```
getGreeting();
```



# Calling vs Referencing

```
getGreeting();           // "Hello, World!"  
getGreeting;            // function getGreeting()
```

- ▶ assign a function to a variable

```
const f = getGreeting;  
f();                  // "Hello, World!"
```

- ▶ assign a function to an object property

```
const o = {};  
o.f = getGreeting;  
o.f();                // "Hello, World!"
```



# Function Arguments

- ▶ เราสามารถใส่ arguments ให้กับฟังก์ชัน และตั้งพารามิเตอร์ได้

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



# 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();
```



# Functions as Properties of Objects

```
const o = {
  name: 'Wallace',                      // primitive property
  bark: function() { return 'Woof!'; },   // function property (method)
}
```

- ▶ ES6 introduces a new **shorthand syntax** for methods

```
const o = {
  name: 'Wallace',                      // primitive property
  bark() { return 'Woof!'; },           // function property (method)
}
```



# The this Keyword

- ▶ the **this** keyword relates to functions that are properties of objects

```
const o = {  
  name: 'Wallace',  
  speak() { return `My name is ${this.name}!`; },  
}  

```

## 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; }
```

- ▶ ลองแปลงเป็น arrow notation ดู



# Scope

- ▶ ขอบเขตในการสร้างและใช้งานตัวแปร, constants, และ arguments

```
function f(x) {  
    return x + 3;  
}  
f(5);      // 8  
x;          // ReferenceError: x is not defined
```

# Global scope

- ▶ เป็น scope ที่ scope อื่นๆ สามารถเรียกใช้ได้ทุกที่

```
let name = "Irena";      // global
let age = 25;            // global

function greet() {
    console.log(`Hello, ${name}!`);
}

function getBirthYear() {
    return new Date().getFullYear() - age;
}
```



# Block Scope

- ▶ Scope ที่อยู่เครื่องหมายปีกกา ถ้าประกาศตัวแปร ค่าคงที่ต่างๆ ในรูปแบบ block scope ก็แสดงว่ามันจะใช้ได้ภายในปีกกาเท่านั้น

```
console.log('before block');
{
  console.log('inside block');
  const x = 3;
  console.log(x);           // Logs 3
}
console.log(`outside block; x=${x}`); // ReferenceError: x is not defined
```



# 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 = [
  { 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

- ▶ ถ้าอยากรอเข้าถึงสมาชิกตัวแรก 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);  
}
```



# Array Content Manipulation

- ▶ การจัดการกับค่าของ array การเพิ่ม หรือลบสมาชิก 1 ตัวที่ array ตัวแรกและตัวสุดท้าย ได้แก่ push, pop, unshift, shift
- ▶ push เพิ่มสมาชิกเข้าไปที่ตัวสุดท้าย
- ▶ pop ลบสมาชิกตัวสุดท้ายนั้นออกไป
- ▶ unshift เพิ่มสมาชิกตัวแรกเข้าไป
- ▶ shift ลบสมาชิกตัวแรกนั้น



# Array Content Manipulation

การเพิ่มสมาชิกทีละหลายๆ ตัว (ต่อท้ายสุด) ให้ใช้ method concat()

```
const arr = [1, 2, 3];
arr.concat(4, 5, 6);
arr.concat([4, 5, 6]);
arr.concat([4, 5], 6);
arr.concat([4, [5, 6]]);
```



# Array Content Manipulation

- ▶ การตัดสมาชิก array ให้ใช้ `slice()` เลือกจากตำแหน่งจุดเริ่มต้น (`begin`) ไปยังจุดสิ้นสุด (`end`) (โดยไม่รวมตำแหน่งของจุดสิ้นสุด)  
อาร์เรย์ต้นฉบับจะไม่ถูกแก้ไข

```
const arr = [1, 2, 3, 4, 5];
arr.slice(3);
arr.slice(2, 4);
arr.slice(-2);
arr.slice(1, -2);
arr.slice(-2, -1);
```

# Array Content Manipulation

ใช้ splice() เพิ่มหรือลบสมาชิก (ได้ทุกตำแหน่ง)

```
array.splice(start, deleteCount[, item1[, item2[, ...]]])
```

```
const arr = [1, 5, 7];
arr.splice(1, 0, 2, 3, 4);      // returns []; arr is now [1, 2, 3, 4, 5, 7]
arr.splice(5, 0, 6);          // returns []; arr is now [1, 2, 3, 4, 5, 6, 7]
arr.splice(1, 2);            // returns [2, 3]; arr is now [1, 4, 5, 6, 7]
arr.splice(2, 1, 'a', 'b');   // returns [5]; arr is now [1, 4, 'a', 'b', 6, 7]
```

```
arr2.splice(1,1,1000);
```

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

1,000 คือตัวเลขที่ต้องการแทนที่เข้าไป

# Array Content Manipulation

- ▶ Copying and Replacing Within an Array
- ▶ ES6 brings a new method, `copyWithin()`

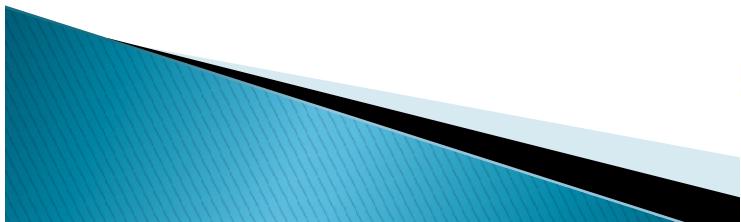
```
arr.copyWithin(target, start[, end = this.length])
```

```
const arr = [1, 2, 3, 4];
arr.copyWithin(1, 2);           // arr is now [1, 3, 4, 4]
arr.copyWithin(2, 0, 2);        // arr is now [1, 3, 1, 3]
arr.copyWithin(0, -3, -1);      // arr is now [3, 1, 1, 3]
```

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

```
arr2.copyWithin(0,2);
```

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



# Array Content Manipulation

## ▶ Reversing and Sorting Arrays

```
const arr = [1, 2, 3, 4, 5];
arr.reverse();                                // arr is now [5, 4, 3, 2, 1]
```

```
const arr = [5, 3, 2, 4, 1];
arr.sort();                                    // arr is now [1, 2, 3, 4, 5]
```



# Array Searching

- ▶ การค้นหาเราจะใช้ indexOf() ถ้าหาเจอจะคืนค่าเป็นเลขตำแหน่ง index
- ▶ ถ้าหาไม่เจอจะคืนค่า -1

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

let result = arr2.indexOf(30);

if (result > 0) {
  console.log(result);
} else {
  console.log(result);
  console.log('not found');
}
```

```
const o = { name: "Jerry" };
const arr = [1, 5, "a", o, true, 5, [1, 2], "9"];
arr.indexOf(5);                                // returns 1
arr.lastIndexOf(5);                            // returns 5
arr.indexOf("a");                             // returns 2
arr.lastIndexOf("a");                           // returns 2
arr.indexOf({ name: "Jerry" });                // returns -1
arr.indexOf(o);                               // returns 3
arr.indexOf([1, 2]);                            // returns -1
arr.indexOf("9");                             // returns 7
arr.indexOf(9);                               // returns -1

arr.indexOf("a", 5);                          // returns -1
arr.indexOf(5, 5);                           // returns 5
arr.lastIndexOf(5, 4);                      // returns 1
arr.lastIndexOf(true, 3);                    // returns -1
```

# Array Searching

- ▶ `findIndex()` คล้ายกับ `indexOf()`

```
const arr = [{ id: 5, name: "Judith" }, { id: 7, name: "Francis" }];
arr.findIndex(o => o.id === 5);           // returns 0
arr.findIndex(o => o.name === "Francis"); // returns 1
arr.findIndex(o => o === 3);             // returns -1
arr.findIndex(o => o.id === 17);          // returns -1
```

- ▶ `find()` ค้นไม่ต้องสนใจ index ถ้าไม่เจอจะ `return null` ถ้าเจอจะ `return` ผลลัพธ์

```
let arr2 = [{id: 1, name: 'john'}, { id: 2, name: 'mary'}];

let result = arr2.find(n => n.name === 'mary');
if (result === undefined) {
  console.log('not found');
} else {
  console.log(result);
}
```

# Array: map

- ▶ map() เป็นการแปลงรูปแบบของ array ไปเป็นรูปแบบที่เราต้องการ การทำงานของมันคือ มันจะ copy สมาชิกมาใช้ (แต่ไม่ได้แก้ไขต้นฉบับ)

```
const cart = [ { name: "Widget", price: 9.95 }, { name: "Gadget", price: 22.95 }];  
const names = cart.map(x => x.name); // ["Widget", "Gadget"]  
const prices = cart.map(x => x.price); // [9.95, 22.95]  
const discountPrices = prices.map(x => x*0.8); // [7.96, 18.36]  
const lcNames = names.map(String.toLowerCase); // ["widget", "gadget"]
```

---

```
const items = ["Widget", "Gadget"];  
const prices = [9.95, 22.95];  
const cart = items.map((x, i) => ({ name: x, price: prices[i]}));  
// cart: [{ name: "Widget", price: 9.95 }, { name: "Gadget", price: 22.95 }]
```

# Array: reduce

- ▶ ตัวอย่างการหาผลรวม array

```
const arr = [5, 7, 2, 4];
const sum = arr.reduce((a, x) => a += x, 0);
```

▶ การรีดูร์เรดูส์จะวนลูปทุกๆ ตัวใน array และคำนวณผลรวม

```
let arr2 = [10, 10, 30];
```

```
let sum = arr2.reduce((sumValue, currValue) => sumValue += currValue, 0);
```

```
console.log(sum);
```



# String Joining

- ▶ เชื่อมค่าสมาชิกของ array (string)

```
const arr = [1, null, "hello", "world", true, undefined];
delete arr[3];
arr.join();          // "1,,hello,,true,"
arr.join('');       // "1hellotrue"
arr.join(' -- ');   // "1 -- -- hello -- -- true --"
```



# 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

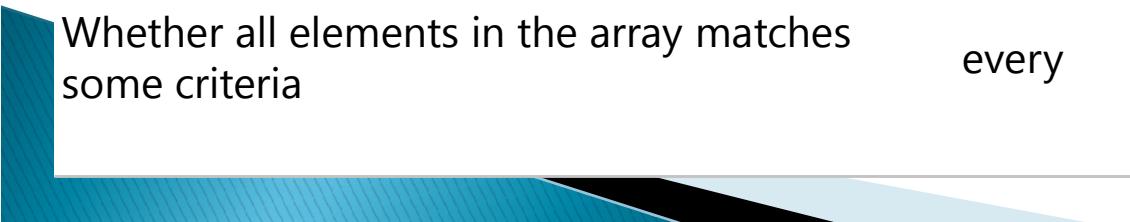
find

Whether the array has an item that matches some criteria

some

Whether all elements in the array matches some criteria

every



# สรุป array เกี่ยวกับการเปลี่ยนรูป *transformation*

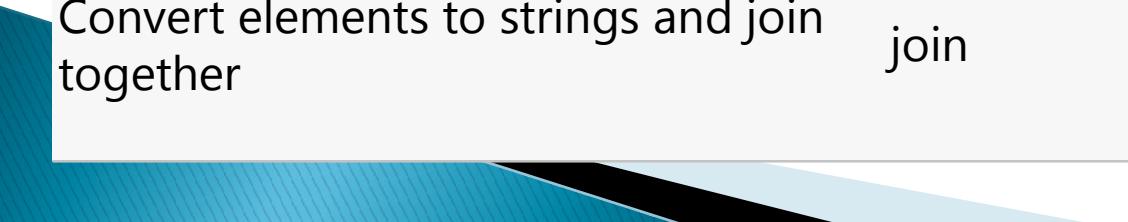
When you want to...	Use...	In-place or copy
---------------------	--------	------------------

Transform every element in an array	map	Copy
-------------------------------------	-----	------

Eliminate elements from an array based on some criteria	filter	Copy
--	--------	------

Transform an entire array into another data type	reduce	Copy
---	--------	------

Convert elements to strings and join together	join	Copy
--	------	------



# Math

- ▶ JavaScript's built-in Math object
- ▶ ถ้าอยากรู้วิธีคำนวณแบบต่างๆ ก็ลองดูใน Math นะ
- ▶ <http://mathjs.org/>



# Formatting Numbers

## ▶ Fixed Decimals

```
const x = 19.51;  
x.toFixed(3);      // "19.510"  
x.toFixed(2);      // "19.51"  
x.toFixed(1);      // "19.5"  
▶ x.toFixed(0);    // "20"
```

```
const x = 12;  
x.toString();      // "12"  (base 10)  
x.toString(10);   // "12"  (base 10)  
x.toString(16);   // "c"   (hexadecimal)  
x.toString(8);    // "14"  (octal)  
x.toString(2);    // "1100" (binary)
```



# Advanced Number Formatting

- ▶ ແນະໜ້າ <http://numeraljs.com/>

## Format

Numbers can be formatted to look like currency, percentages, times, or even plain old numbers with decimal places, thousands, and abbreviations.

```
var string = numeral(1000).format('0,0');
// '1,000'
```

## Numbers

Number	Format	String
10000	'0,0.0000'	10,000.0000
10000.23	'0,0'	10,000
10000.23	'+0,0'	+10,000
-10000	'0,0,0'	-10,000.0
10000.1234	'0.000'	10000.123
10000.1234	'0[,]00000'	10000.12340
-10000	'(0,0.0000)'	(10,000.0000)

# JavaScript: Exceptions and Error Handling



# The Error Object

- ▶ JavaScript has a built-in **Error object**

```
const err = new Error('invalid email');
```

```
function validateEmail(email) {
  return email.match(/@/) ?
    email :
    new Error(`invalid email: ${email}`);
}
```



# The Error Object

- ▶ เราสามารถใช้ instanceof เพื่อตรวจสอบ Errors ได้ และใช้ .message เพื่อแสดงข้อความข้อผิดพลาด

```
function validateEmail(email) {  
    return email.match(/@/) ?  
        email :  
        new Error(`invalid email: ${email}`);  
}  
  
const email = "jane@doe.com";  
  
const validatedEmail = validateEmail(email);  
if(validatedEmail instanceof Error) {  
    console.error(`Error: ${validatedEmail.message}`);  
} else {  
    console.log(`Valid email: ${validatedEmail}`);  
}
```

# Exception Handling with try and catch

```
const email = null; // whoops

try {
  const validatedEmail = validateEmail(email);
  if(validatedEmail instanceof Error) {
    console.error(`Error: ${validatedEmail.message}`);
  } else {
    console.log(`Valid email: ${validatedEmail}`);
  }
} catch(err) {
  console.error(`Error: ${err.message}`);
}
```



# Throwing Errors

```
function billPay(amount, payee, account) {  
    if(amount > account.balance)  
        throw new Error("insufficient funds");  
    account.transfer(payee, amount);  
}
```

- ▶ ข้อดีของการ throw ก็คือ เมื่อเกิด errors การทำงานจะหยุด และออกจากจุดอยู่นั้นทันที



## try...catch...finally

```
try {  
    console.log("this line is executed...");  
    throw new Error("whoops");  
    console.log("this line is not...");  
} catch(err) {  
    console.log("there was an error...");  
} finally {  
    console.log("...always executed");  
    console.log("perform cleanup here");  
}
```

The end

