

รีวิวฟรีเจอร์ใหม่ใน

จาวาสคริปต์มาตรฐาน ES7

(ECMAScript 2016)

แก้ไขครั้งที่ 2

เขียนโดย แอดมินโฮ โอน้อยออก

JAVASCRIPT

(ECMAScript)

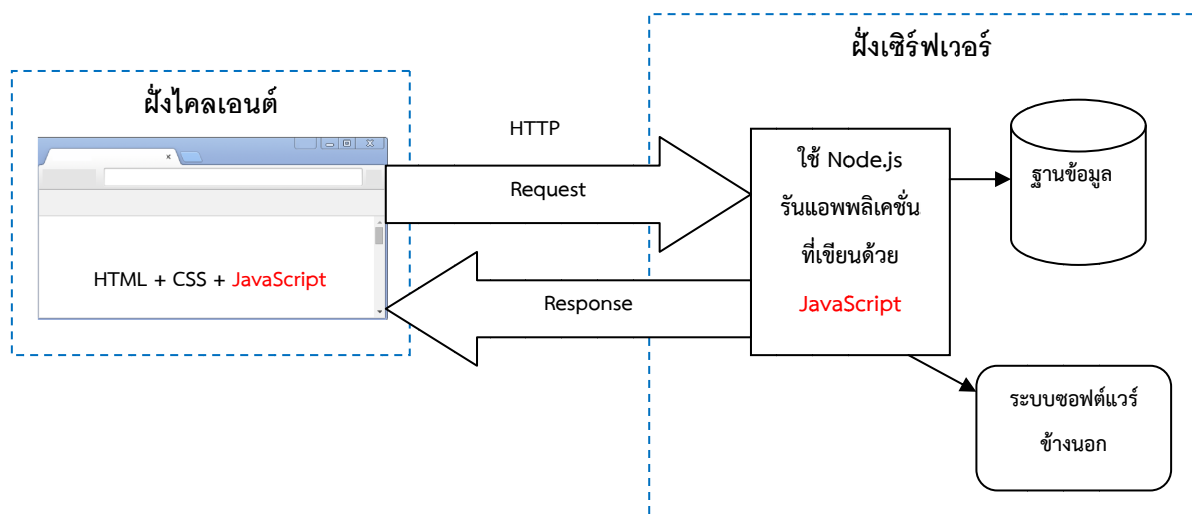
*EBook เล่มนี้สงวนลิขสิทธิ์ตามกฎหมาย ห้ามมิให้ผู้ใด นำไปเผยแพร่ต่อสาธารณะ เพื่อประโยชน์ในการค้า หรืออื่นๆ โดย  
ไม่ได้รับความยินยอมเป็นลายลักษณ์อักษรจากผู้เขียน*

## ให้ความรู้เพิ่มเติมนิดหนึ่ง เพื่อคนไม่รู้จักภาษา JavaScript

- JavaScript เป็นภาษาเขียนโปรแกรมที่**โคตรจะอินดี้** ถ้าศึกษาอย่างผิวเผินก็จะคิดว่าง่ายง่าย แต่เมื่อศึกษาลงลึก ๆ แล้ว ก็จะพบว่ามันเป็นภาษาปราบเซียน จนคนไม่ค่อยเข้าใจกันมากสุดภาษาหนึ่งในโลก
- JavaScript ไม่ใช่ภาษา Java นะครับ คนละภาษา (คนมักสับสนกัน)

แวนน์ ไบซ์ แวน จันใด  
JavaScript ก็ไบซ์ Java จันนั้น

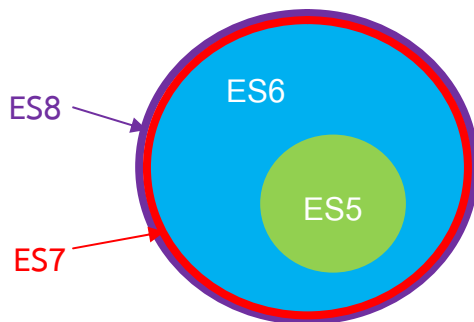
- คนส่วนใหญ่รู้แค่ว่าใช้ JavaScript ร่วมกับภาษา HTML กับ CSS เพื่อให้เว็บมันไดนามิก ฟังก์ชัน กรังกิง (มันดังในฝั่ง Font-end มานาน)
- แต่ปัจจุบันนี้ **JavaScript สมัยใหม่** มันก้าวหน้าไปไกลมาก ๆ ๆ ๆ เพราะสามารถทำงานอยู่ฝั่งเซิร์ฟเวอร์ได้ (Back-end) ด้วย Node.js แม้แต่เอาไปทำแอปบนโมบาย หรือแม้แต่เว็บอ็อท ก็ยังทำได้ด้วย ....อวยยะ



- ES6 (ECMAScript 2015) เป็นมาตรฐานใหม่ล่าสุดของ JavaScript ประกาศออกมาเมื่อกลางเดือนมิถุนายนปี 2558 ซึ่งถือว่า**เปลี่ยนแปลงเวอร์ชันครั้งใหญ่ที่สุดในประวัติศาสตร์ของภาษานี้** หลังจากไม่ได้เปลี่ยนมาเกือบ 6 ปี (เวอร์ชันเก่าที่เราคุ้นเคยกันดี ก็คือ ES5)



- ปีค.ศ. 2016 เวอร์ชันใหม่ ES7 (ECMAScript 2016) ก็ออกมาแหละ ส่วนปีหน้า 2017 ก็จะเป็นคิวของเวอร์ชัน ES8 (ECMAScript 2017) จะออกมาเช่นกัน
- ต้องเข้าใจอย่างนั้นะครัช เนื่อง ES6 มันใหญ่โตล้งการงานสร้างมาก คึ้นรอบปล่อยออกมาหมดทีเดียว ก็คงรอหลายชาติภพ อาจทำให้มีเสียงบ่นตามมาได้ ด้วยเหตุนี้เข้าถึงเพิ่มฟีเจอร์เล็กยับ ๆ ย่อย ๆ มาใส่ไว้ในเวอร์ชันหลัง ๆ แทน
- โดยคาดว่าจากนี้ไป จะมีการประกาศเวอร์ชันใหม่ทุก ๆ ปี โดยให้คิดเสียว่า ES6 เหมือนโปรแกรมหลัก ส่วนเวอร์ชันที่ออกตามทีหลัง ไม่ได้ว่าจะเป็น ES7, ES8 และ ESXXXXX มันก็คือการอัปเดตซอฟต์แวร์ อะไรมะประมาณนี้



- API ที่ใช้ติดต่อกับ DOM หรือใช้งานร่วมกับ HTML5, CSS3 ใน ES6 เขาไม่ได้เปลี่ยนแปลงอะไรเลย
- ES6, ES7, ES8 มันเป็นแค่มาตรฐานใหม่สด ๆ ชิง ๆ ดังนั้นการใช้งานโดยตรงบนเว็บเบราว์เซอร์ (ปัจจุบันที่ผมเขียนอยู่นี้) ก็ยังไม่ support ทุกฟีเจอร์ ต้องมีตัวคอมไพล์ช่วยก่อน (ยังมีข้อจำกัดบางประการ) ...หรือถ้าใครใช้ Node.js เวอร์ชัน 6 ก็ยังรองรับ ES6 ได้แค่ 93 % (ES7 รองรับได้บางส่วน)

ชื่อเวอร์ชัน	ชื่อมาตรฐานเต็ม	ปีที่ออก
ES6	ECMAScript 2015	2015
ES7	ECMAScript 2016	2016
ES8	ECMAScript 2017	2017
เวอร์ชันต่อไปนี้จะอัปเดตจาก ES6 ไปเรื่อย ๆ		

### ข้อตกลงเวลาอ่านเอกสารชุดนี้

- จะกล่าวถึงเฉพาะฟีเจอร์ที่เพิ่มเข้ามาใน ES7 และ สิ่งที่เปลี่ยนแปลงไปในเวอร์ชันดังกล่าว
- สำหรับความรู้จาวาสคริปต์มาตรฐานใหม่ ES6 แบบเจาะลึกถึงขั้วหัวใจ (เนื้อหาเยอะมาก) ท่านสามารถอ่านได้จากหนังสือที่อ้างอิง [1] เพราะความรู้จาวาสคริปต์แบบเก่า (ES5) ที่คุณรู้จัก ...**นับวันใกล้หมดอายุลงเต็มทน**

## ตัวอย่างการเขียน ES6 กับ ES7

ตัวอย่างนี้จะแสดงการเขียน JavaScript บนเว็บเบราว์เซอร์ โดยใช้ Traceur ทำตัวเป็น transpiler (คอมไพเลอร์) เพื่อแปลงซอร์สโค้ดให้อยู่ในรูปแบบเวอร์ชัน ES5 จากนั้นเว็บเบราว์เซอร์ถึงจะทำงานได้ (หาอ่านเพิ่มเติมได้ตามหนังสือ [1])

```
<!-- ไฟล์ index.html -->
<!DOCTYPE html>
<html>
<head>

<!-- Traceur (เป็นตัว transpiler) -->
<script src="https://google.github.io/traceur-compiler/bin/traceur.js"></script>
<script src="https://google.github.io/traceur-compiler/bin/BrowserSystem.js"></script>
<script src="https://google.github.io/traceur-compiler/src/bootstrap.js"></script>
</head>
<body>
<h1 id="element1"></h1>
<script type="module">

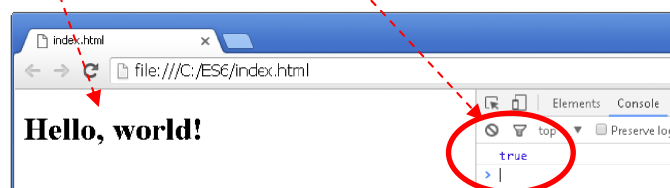
    class Chat{                                // class ไวยากรณ์ใหม่ของ ES6
        constructor(message) {                // constructor ไวยากรณ์ใหม่ของ ES6
            this.message = message;
        }
        say(){
            let element = document.querySelector('#element1');
            element.innerHTML = this.message;
        }
    }

    let chat = new Chat("Hello, world!");      // let ไวยากรณ์ใหม่ของ ES6
    chat.say();

    let array = ["A", "B", "C"];                // let ไวยากรณ์ใหม่ของ ES6
    console.log(array.includes("A"));           // true    -- เมธอดของอาร์เรย์ที่เพิ่มเข้ามาใน ES7

</script>
</body>
</html>
```

ถ้าซอร์สโค้ดดังกล่าวเซฟเป็นไฟล์เก็บไว้ที่ "C:/ES6/index.html" เมื่อดับเบิลคลิกเปิดมันขึ้นมา ก็จะปรากฏดังรูป



หมายเหตุ ถ้าอ่านจากหนังสือ [1] วิธีการใช้ ES6 กับ ES7 จะต่างกับซอร์สโค้ดที่เห็นในตัวอย่างนี้เล็กน้อย เพราะ traceur มันมีการเปลี่ยนแปลง ด้วยการเพิ่ม BrowserSystem.js เข้าไปใน <head> ...</head> ดังนี้

```
<script src="https://google.github.io/traceur-compiler/bin/BrowserSystem.js"></script>
```

(ที่มา <https://github.com/google/traceur-compiler/wiki/Getting-Started>)

ตัวอย่างต่อไปนี้จะแสดงการเขียน ES6 กับ ES7 บน node.js ดังนี้

```
class Chat{                                     // class ไวยากรณ์ใหม่ของ ES6
    constructor(message) {                     // constructor ไวยากรณ์ใหม่ของ ES6
        this.message = message;
    }
    say(){
        console.log(this.message);
    }
}
let chat = new Chat("Hello, world!");          // let ไวยากรณ์ใหม่ของ ES6
chat.say();                                    // "Hello, world!"

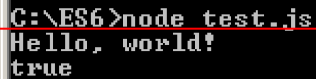
let array = ["A", "B", "C"];
console.log(array.includes("A"));              // true    -- เมธอดของอาร์เรย์ที่เพิ่มมาใน ES7
```

ถ้าซอร์สโค้ดดังกล่าวเซฟเป็นไฟล์เก็บไว้ที่ "C:/ES6/test.js" และมีโครงสร้างโปรเจกต์ดังนี้

C:/ES6/

|-- test.js

เมื่อรันคอมมานด์บนวินโดวส์ (ให้ Node.js มาอ่านและประมวลผลไฟล์จาวาสคริปต์) ก็จะได้ผลลัพธ์ดังภาพ



```
C:\ES6>node test.js
Hello, world!
true
```

หมายเหตุ สำหรับวิธีติดตั้งและรันจาวาสคริปต์บน Node.js สามารถอ่านเพิ่มเติมได้ที่ (เป็น PDF แจกฟรี)

- <http://www.patanasongsivilai.com/javascript.html>

## ฟีเจอร์ใหม่ที่เพิ่มเข้ามาใน ES7 (นิตเดี่ยวเอง)

### เพิ่มการใช้งานโอเปอเรเตอร์ยกกำลัง (Exponentiation Operator)

โอเปอเรเตอร์ยกกำลังจะใช้สัญลักษณ์เป็น **\*\*** (ดอกจันทั้งสองอันวางติดกัน) เพื่อแทนการคำนวณตัวเลขแบบยกกำลัง โดยไม่ต้องใช้เมธอด `Math.pow()` ซึ่งจะมีตัวอย่างการใช้งานดังนี้

```
let ans = 10 ** 2;           // นำเลข 10 มายกกำลัง 2 ( 102 )
console.log(ans);           // 100

// เหมือนใช้เมธอด Math.pow() ดังนี้
console.log(ans === Math.pow(10, 2)); // true
```

#### ลำดับของโอเปอเรเตอร์ \*\*

โอเปอเรเตอร์ **\*\*** จะถือว่ามีลำดับความสำคัญสูงกว่าโอเปอเรเตอร์ทางคณิตศาสตร์ตัวอื่น ๆ

```
let ans = 3 * 10 ** 2;
console.log(ans); // 300
```

จากตัวอย่างเดิมจะเหมือนมีวงเล็บมาครอบนิพจน์  $(10 ** 2)$  ดังตัวอย่างซอร์สโค้ดข้างล่าง

```
let ans = 3 * (10 ** 2);
console.log(ans); // 300
```

#### ข้อห้ามของโอเปอเรเตอร์ \*\*

โอเปอเรเตอร์ยกกำลังไม่สามารถใช้งานร่วมกับโอเปอเรเตอร์พวก unary expression เช่น `-` (เครื่องหมายลบ ไม่ใช้การลบ) , `+` (เครื่องหมายบวก ไม่ใช้การบวก), `void`, `delete` และ `typeof` เป็นต้น โดยจะให้ดูตัวอย่างต่อไปนี้ประกอบ

```
let ans = -10 ** 2; // syntax error
```

ตัวอย่างที่ยกมานี้จะเกิด error เพราะตรงนิพจน์  $-10 ** 2$  มันกำกวม เนื่องจากอาจหมายถึง

- $-(10 ** 2)$
- $(-10) ** 2$

จากตัวอย่างเดิม ถ้าลองนำวงเล็บมาครอบเพื่อกำหนดลำดับการทำงานเสียใหม่ ก็จะไม่เกิด error ดังตัวอย่างหน้าถัดไป

```
let ans = - (10 ** 2); // -100
```

จากตัวอย่างเดิมเช่นกัน ถ้าลองเปลี่ยนการครอบวงเล็บเสียใหม่ ก็จะได้ผลการทำงานที่แตกต่างกันดังนี้

```
let ans = (-10) ** 2; // 100
```

ขณะเดียวกันโอเปอเรเตอร์ยกกำลังก็มีข้อยกเว้น มันสามารถใช้ได้กับ ++ หรือ -- (เป็น unary expression) โดยไม่ต้องใช้วงเล็บครอบ

ลองพิจารณาการใช้โอเปอเรเตอร์ยกกำลังร่วมกับโอเปอเรเตอร์ ++ ดังตัวอย่าง

```
let value1 = 9, value2 = 10;

// ใช้งานโอเปอเรเตอร์++ แบบprefix
// ค่าของ value1 ถูกบวกด้วยหนึ่ง ก่อนที่จะยกกำลัง 2
console.log(++value1 ** 2); // 100
console.log(value1); // 10

// ใช้งานโอเปอเรเตอร์++ แบบpostfix
// หลังจากยกกำลัง 2 ไปแล้ว ค่าของ value2 จึงถูกบวกด้วยหนึ่งที่หลัง
console.log(value2++ ** 2); // 100
console.log(value2); // 11
```

ลองพิจารณาการใช้โอเปอเรเตอร์ยกกำลังร่วมกับโอเปอเรเตอร์ -- ดังตัวอย่าง

```
let value1 = 11, value2 = 10;

// ใช้งานโอเปอเรเตอร์-- แบบprefix
// ค่า value1 ถูกลบด้วยหนึ่ง ก่อนที่จะยกกำลัง 2
console.log(--value1 ** 2); // 100
console.log(value1); // 10

// ใช้งานโอเปอเรเตอร์-- แบบpostfix
// หลังจากยกกำลัง 2 ไปแล้ว ค่าของ value2 จึงถูกลบด้วยหนึ่งที่หลัง
console.log(value2-- ** 2); // 100
console.log(value2); // 9
```

**หมายเหตุ** โอเปอเรเตอร์ \*\* ตามสเปค ES7 ผมยังหาจาวาสคริปต์เอ็นจินรองรับการรันทดสอบไม่ได้เลย (เศร้าจัง) สรุปข้อสงสัยที่ได้เห็นในตัวอย่างที่ผ่าน ๆ มา ขาดการทดสอบจริงจัง ดังนั้นถ้าในอนาคตสามารถทดสอบได้ เดี่ยวมาปรับแก้เนื้อหาใหม่ ตอนนี้อาจคอนเซปต์ให้เห็นไปก่อนแล้วกันนะ!

## เพิ่มเมธอด Array.prototype.includes()

สำหรับ ES6 นั้น สตริงทุกตัวจะมีเมธอด includes() และเช่นเดียวกันใน ES7 ก็ได้เพิ่มเมธอดดังกล่าวให้กับอาร์เรย์ โดยมีจุดประสงค์ใช้ค้นหาข้อมูลในอาร์เรย์ ถ้าเจอข้อมูลที่ต้องการหาก็จะรี턴เป็น true ถ้าไม่เจอก็จะได้เป็น false ดังตัวอย่าง (ทำงานแบบเดียวกับ includes() ของสตริงบที่ 5 ในหนังสือ [1])

```
let array = ["A", "B", "C"]; // ประกาศอาร์เรย์

console.log(array.includes("A")); // true
console.log(array.includes("Z")); // false
```

ในตัวอย่างนี้จะค้นหาตัวอักษร "A" เจอในอาร์เรย์ แต่ไม่สามารถค้นหา "Z" พบ เพราะมันไม่มีอยู่ในอาร์เรย์

ปกติแล้วเมธอด includes() จะเริ่มค้นหาที่ตำแหน่งอินเด็กซ์เป็น 0 โดยดีฟอลต์ ดังนั้นถ้าจะเปลี่ยนตำแหน่งอินเด็กซ์ที่ใช้ค้นหา ก็สามารถทำได้ดังตัวอย่าง

```
let array = ["A", "B", "C"]; // ประกาศอาร์เรย์
// เริ่มค้นหา "B" จากอินเด็กซ์คือ 2 ซึ่งจะพบว่าหาไม่เจอ
console.log(array.includes("B", 2)); // false

// แต่ถ้าเปลี่ยนมาเริ่มค้นหาจากอินเด็กซ์เป็น 1 ก็เจอ "B" เจอ
console.log(array.includes("B", 1)); // true
```

ในตัวอย่างดังกล่าวจะเห็นว่าเมธอด includes รับค่าอาร์กิวเมนต์ตัวที่สอง เพื่อระบุตำแหน่งเริ่มต้นของอินเด็กซ์ที่จะใช้ค้นหาข้อมูลในอาร์เรย์

### ข้อควรระวัง includes()

เมธอด includes() จะเหมือนใช้โอเปอเรเตอร์ === เปรียบเทียบว่ามีสมาชิกที่ต้องค้นหาหรือไม่ แต่ทว่าเวลามันเห็นข้อมูลเป็น NaN ก็จะมองว่ามีค่าเท่ากัน (เปรียบเทียบแล้วได้ true) ซึ่งจะแตกต่างจาก indexOf ซึ่งจะเหมือนใช้ === เช่นกัน ซึ่งเวลามันเห็น NaN จะมองว่ามีค่าต่างกัน (เปรียบเทียบแล้วได้ false) ดังตัวอย่าง

```
let array = [0, NaN, 1];

console.log(array.indexOf(NaN)); // -1 -- ไม่เจอสมาชิกที่ต้องการ
console.log(array.includes(NaN)); // true
```

แต่ถ้าข้อมูลเป็น +0 กับ -0 จะมองว่าเท่ากัน (เปรียบเทียบแล้วได้เป็น true) ทั้ง includes() กับ indexOf() ดังตัวอย่าง

```
let array = [-0, NaN, 1];

console.log(array.indexOf(+0)); // 0 -- เจอค่า -0 อยู่ในอาร์เรย์ที่ตำแหน่งอินเด็กซ์ 0
console.log(array.includes(+0)); // true
```



## TypedArray.prototype.includes()

ในอาร์เรย์ระดับบิต (TypedArray บทที่ 12 ของหนังสือ [1]) ก็จะมีเมธอด includes() ให้ใช้งานเหมือนกับอาร์เรย์ในหัวข้อก่อนหน้านี้ทุกประการเด๊ะ ดังตัวอย่าง

```
let uint8 = new Uint8Array([1, 2, 3, 4, 5]);
console.log(uint8.includes(1)); // true
console.log(uint8.includes(5)); // true
console.log(uint8.includes(10)); // false
```

## สิ่งที่เปลี่ยนแปลงไปของ ES7 เมื่อเทียบกับ ES6 (นิดเดียวเอง)

หัวข้อก่อนหน้านี้ได้กล่าวถึงฟีเจอร์ที่เพิ่มมาใหม่ใน ES7 แต่หัวข้อนี้จะกล่าวถึงฟีเจอร์ที่เปลี่ยนไปจาก ES6 ดังนี้

- 1) trap ที่เป็น enumerate() ของพร็อกซี (บทที่ 14 ของหนังสือ [1]) ถูกเอาออกไปใน ES7 เรียบร้อยแล้ว
- 2) เจอเนอเรเตอร์ (บทที่ 13 ของหนังสือ [1]) ไม่มี [[Construct]] ถ้าเรียก new จะเกิด error ขึ้นมาดังตัวอย่าง

```
function * generator() {}
let iterator = new generator(); // throws "TypeError: f is not a constructor"
```

## สิ่งที่คาดว่าจะเพิ่มเข้ามาใน ES8 (ECMAScript 2017) (มีนิดเดียว)

- Object.values()
- Object.entries()

## อ้างอิง

- [1] หนังสือ “พัฒนาเว็บแอปพลิเคชันด้วย JavaScript” จะอธิบายถึงมาตรฐานตัวใหม่ **ECMAScript 2015** หรือเรียกสั้น ๆ ว่า “**ES6**” หรือ “**ES6 Harmony**” โดยเล่มนี้ตีพิมพ์และจัดจำหน่ายโดยทีเอ็ด



[2] [https://developer.mozilla.org/en-](https://developer.mozilla.org/en-US/docs/Web/JavaScript/New_in_JavaScript/ECMAScript_Next_support_in_Mozilla)

[US/docs/Web/JavaScript/New\\_in\\_JavaScript/ECMAScript\\_Next\\_support\\_in\\_Mozilla](https://developer.mozilla.org/en-US/docs/Web/JavaScript/New_in_JavaScript/ECMAScript_Next_support_in_Mozilla)

[3] <https://github.com/nzakas/understandings6/blob/master/manuscript/B-ECMAScript-7.md>

[4] <https://tc39.github.io/ecma262/2016/>

### ก่อนจากกัน

- PDF จะอยู่ที่นี้ครับ <http://www.patanasongsivilai.com/javascript.html> (ถ้าไปดาวน์โหลดจากที่อื่นอาจไม่ได้เวอร์ชันใหม่ล่าสุด)
- สนใจอยากติดตามเพจเกี่ยวกับไอที ก็กด Like ได้ที่ <https://www.facebook.com/programmerthai/>

เขียนโดย

แอดมินโฮ ोन้อยออก

(จตุรพัชร พัฒนทรงศิริไธ)

30 พ.ค. 2559