

JavaScript



JavaScript (JS)

- ภาษาโปรแกรมสำหรับพัฒนา Web Application ที่สามารถประมวลผลบน Browser หรือบน Server ก็ได้
- ใช้พัฒนาได้ทั้งส่วน frontend และ backend ซึ่งระบบที่พัฒนาแบบนี้เรียกว่า Full Stack JavaScript (หรือ Isomorphic JavaScript)
- ในบทนี้จะกล่าวเฉพาะ JavaScript ที่ประมวลผลบน Browser เพื่อเพิ่ม Logic เข้าไปบนหน้าเว็บตามเหตุการณ์ต่างๆ เช่น
 - a) ขณะโหลดหน้าเว็บเสร็จแล้ว
 - b) ขณะคลิกที่ปุ่ม
 - c) กรอกข้อมูลในฟอร์มช่องนั้นเสร็จแล้ว ต้องการตรวจสอบบางอย่าง

ความเป็นมาของ JavaScript

- [พ.ศ. 2538](#) วิศวกรบริษัท Netscape สร้างภาษา LiveScript เพื่อใช้กับ Browser ชื่อ Netscape Navigator
- [พ.ศ. 2539](#) ได้ถูกเผยแพร่ด้วยชื่อ JavaScript เพื่อให้มีความคล้ายคลึงกับภาษา Java ที่กำลังเป็นที่นิยมในขณะนั้น
- Netscape ส่ง JavaScript ให้องค์กร Ecma International เป็นผู้กำหนดมาตรฐาน โดยตั้งชื่ออย่างเป็นทางการว่าภาษา ECMAScript รุ่น 1 หรือ ES1 ในปี [พ.ศ. 2540](#)
- ถูกพัฒนาเรื่อย ๆ มาถึง ECMAScript รุ่น 5 หรือ ES5 ในปี [พ.ศ. 2552](#) ซึ่งเป็นที่นิยมใช้อย่างแพร่หลายจนถึงปัจจุบัน
- ปี [พ.ศ. 2558](#) ออก ES6 ชื่อเต็มว่า ECMAScript2015 ชื่อเล่น ECMAScript Harmony หรือ ES6 Harmony
- ปี [พ.ศ. 2559 -2562](#) ออก ES ทุกปี จนถึงปัจจุบัน ล่าสุดคือ ES9

การแทรกคำสั่ง JavaScript บนเว็บเพจ

- **Internal Script** -แทรกคำสั่ง JavaScript **ในแท็ก<script>** ภายใต้อะแท็ก<head> เมื่อต้องการให้โหลด หรือทำงานก่อนการแสดงผล หรือภายใต้อะแท็ก<body> เมื่อต้องการให้ทำงานในช่วงแสดงผล
- **External Script** -นำคำสั่ง JavaScript **ในไฟล์แยกต่างหาก** แล้วอ้างอิงไฟล์ในแท็ก<script> ซึ่งอยู่ภายใต้อะแท็ก<head> เหมาะกับการแชร์คำสั่งหรือฟังก์ชันให้กับเว็บหลายหน้า

Internal Script

```
<!doctype html>
```

```
<html>
```

```
<head>
```

```
<script>
```

```
  console.log('Hello World')
```

```
  console.log('Hi Pocky!')
```

```
</script>
```

เพิ่มคำสั่ง *JavaScript* ในการ
แสดงข้อความออกทาง *Console*

```
</head>
```

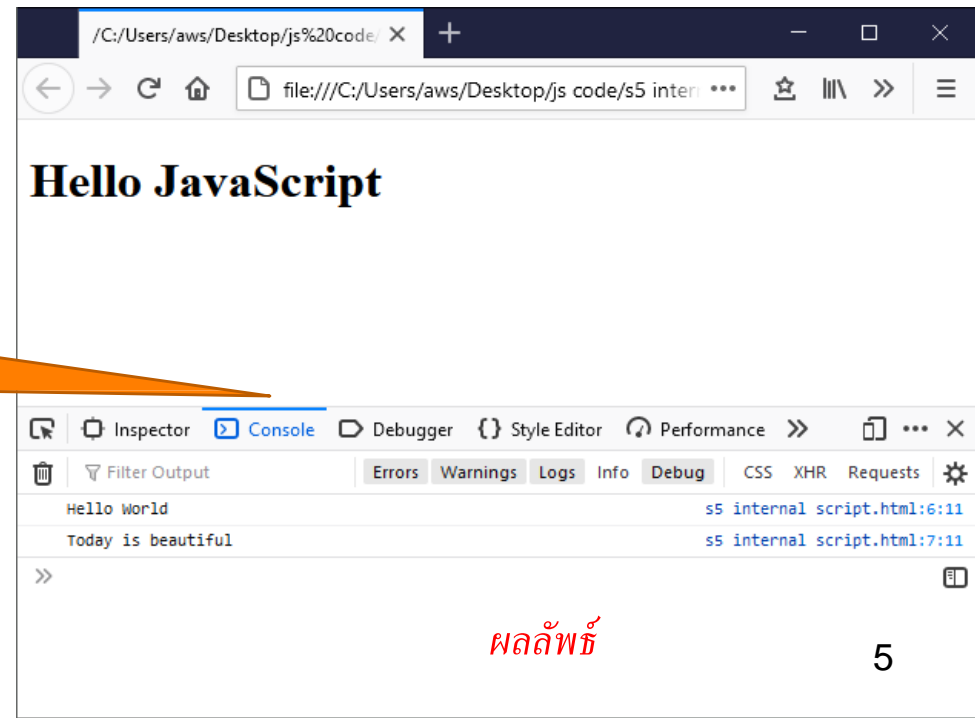
```
<body>
```

```
  <h1>Hello JavaScript</h1>
```

```
</body>
```

```
</html>
```

กดปุ่ม *F12* และเลือก
ที่แท็บ *Console*



ผลลัพธ์

External Script

```
<!doctype html>
```

```
<html>
```

```
<head>
```

อ้างอิงไฟล์โดยใช้ *Relative URL*

```
<script src="myscript.js"> </script>
```

```
</head>
```

ต้องมีแท็กปิดเสมอ

```
<body>
```

```
    <h1>Hello JavaScript</h1>
```

```
</body>
```

```
</html>
```

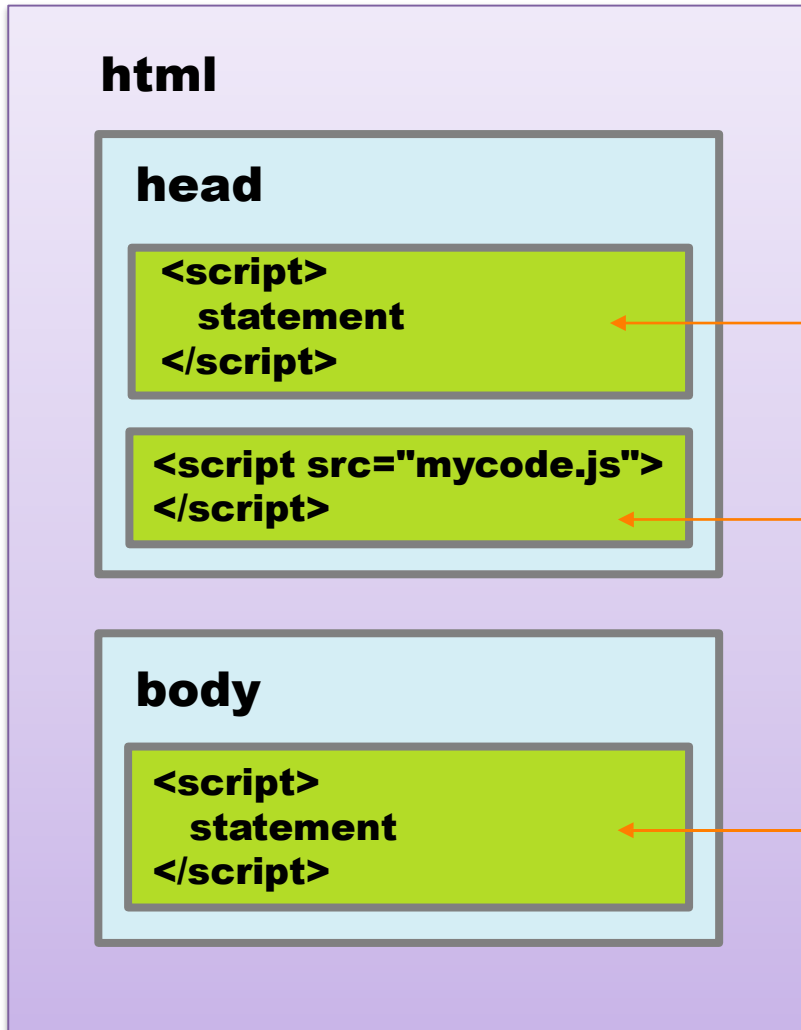
myscript.js

```
console.log('Hello World')
```

```
console.log('Hi Pocky!')
```

//ไม่ต้องใส่แท็ก**HTML** ใดๆในนี้

External และ Internal Script



แทรกไว้ในส่วนของ *<head>*
Script จะทำงานก่อนที่จะแสดงหน้าเว็บ

แยกเป็นไฟล์ต่างหาก ที่มีนามสกุล *.js*
แล้วอ้างอิงโดย *Relative Path*

แทรกไว้ในส่วน *<body>* จะทำงานขณะที่แสดง
หน้าเว็บตามลำดับจากบนลงล่าง

JavaScript Comment

การอธิบายโปรแกรมใช้รูปแบบเดียวกับภาษาซี

```
<html>
<head>
  <script>
    //แบบอธิบายจบภายในบรรทัดเดียว

    /* แบบอธิบายหลายๆ บรรทัด
    บรรทัดที่ 2
    บรรทัดที่ 3*/

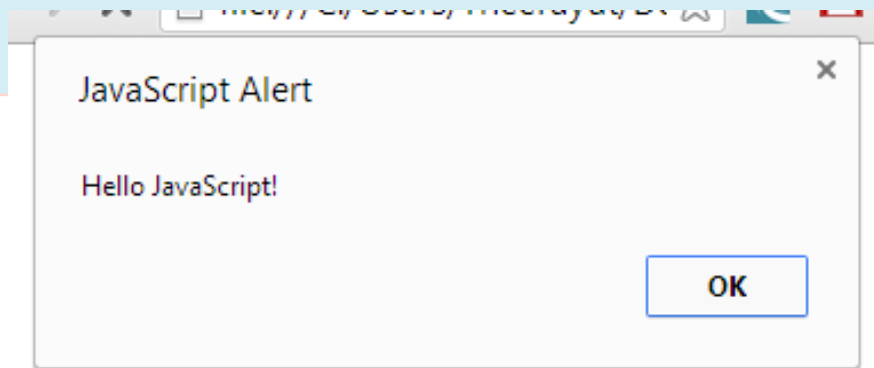
  </script>
</head>
  <body>
    ...
  </body>
</html>
```


การแสดงความแบบ Alert

```
<html>

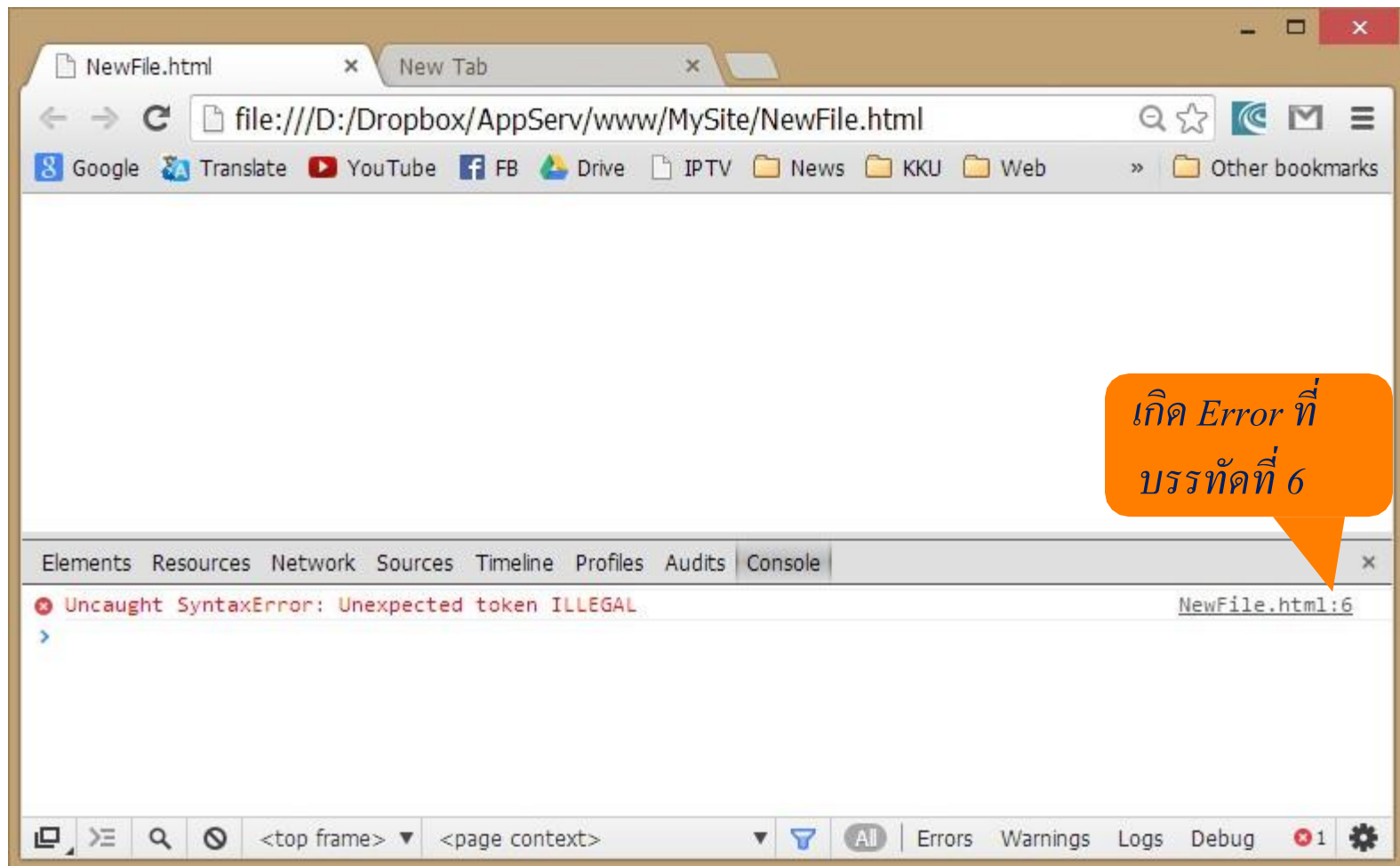
  <body>
    <script>
      alert('Hello JavaScript!')
    </script>
  </body>

</html>
```



การดู Error จาก Console

- บน Google Chrome กด F12 และเลือกแท็บ Console



การประกาศตัวแปรแบบ let

- ชื่อตัวแปรเป็นแบบ case-sensitive
- ตัวแปรใน JavaScript ไม่ต้องระบุชนิดของข้อมูล (Weakly Type) สามารถเปลี่ยนแปลงชนิดข้อมูลได้ตลอดเวลา
- การกำหนดค่าให้ตัวแปรจะใช้เครื่องหมาย =

let count = 2 *ตัวแปรชนิด integer*

let price = 53.50 *ตัวแปรชนิด floating point*

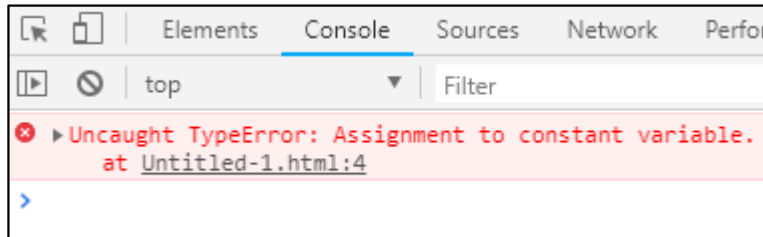
let name = 'Johnny'
let name2 = 'John' *ตัวแปรชนิด string (JavaScript ไม่มีชนิด char) และ String ไม่ได้หมายถึง Array ของ Character*

let isEmpty = false *ตัวแปรชนิด boolean มี 2 ค่า คือ true หรือ false*

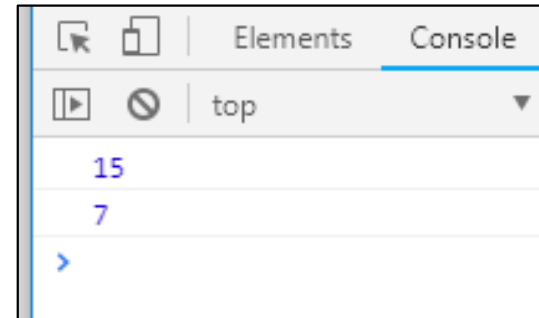
การประกาศตัวแปรแบบ const และ let

- ตัวแปรที่ประกาศแบบ const จะไม่สามารถกำหนดค่าใหม่ได้ แต่ตัวแปรแบบ let จะกำหนดค่าใหม่ได้

```
<html><head>
<script>
  const a = 15
  a = 69
  console.log(a)
</script>
</head></html>
```



```
<html><head>
<script>
  let a = 15
  console.log(a)
  a = 7
  console.log(a)
</script>
</head></html>
```



JS ไม่เหมือนภาษาอื่น...

- สังเกตว่า คำสั่ง (Statement) ไม่จำเป็นต้องปิดด้วย ; (semicolon) เรียกว่า Automatic Semicolon Insertion
- ในความเห็นของผู้สอน...กระบวนการดังกล่าวมีไว้เป็นทางเลือกแต่เพื่อความชัดเจนในการทำงาน ควรใส่ไว้
- มีความแตกต่างระหว่าง การประกาศตัวแปรด้วย var กับ let
 - Function Scope VS. Block Scope
 - <https://github.com/mbeaudru/modern-js-cheatsheet/blob/master/translations/th-TH.md>
 - https://www.w3schools.com/js/js_let.asp
- แนะนำให้ประกาศตัวแปรด้วย let ซึ่งจะทำให้ใช้งานตัวแปรได้เหมือนกับภาษาอื่นที่เคยเรียนมา

Reserved word

abstract	delete	goto	null	throws
as	do	if	package	transient
boolean	double	implements	private	true
break	else	import	protected	try
byte	enum	in	public	typeof
case	export	instanceof	return	use
catch	extends	int	short	var
char	false	interface	static	void
class	final	is	super	volatile
continue	finally	long	switch	while
const	float	namespace	synchronized	
debugger	for	native	this	with
default	function	new	throw	

Undefined และ Null

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

```
  let person
```

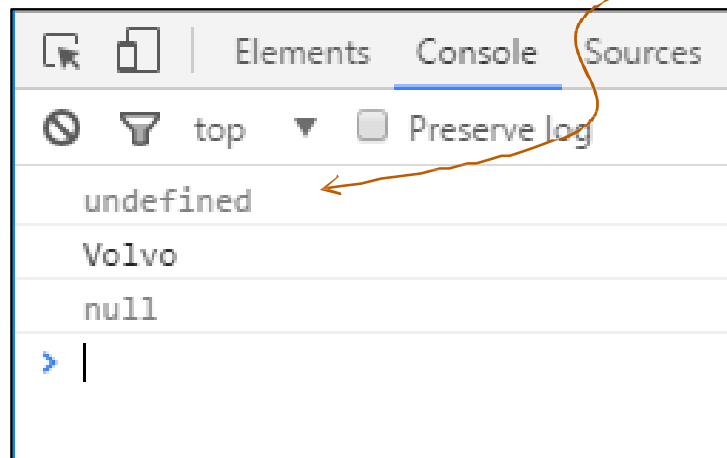
```
  let car = 'Volvo'
```

```
  console.log(person) ← ตัวแปรที่ยังไม่มีการกำหนดค่า จะแสดงเป็น Undefined  
  console.log(car)
```

```
  let car = null ← การกำหนดค่าว่างให้กับตัวแปรจะใช้ค่า null  
  console.log(car)
```

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

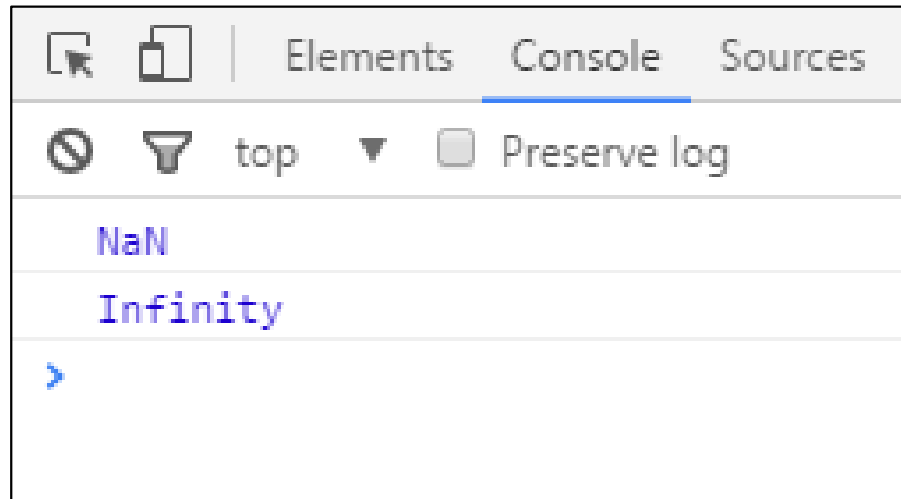
ผลลัพธ์ทาง console



NaN (Not-A-Number) และ Infinity

```
<html>
<body>
<script>
  let x = 'Joey' * 10    // NaN
  let y = 10 / 0         // Infinity
  console.log(x)
  console.log(y)
</script>
</body>
</html>
```

ผลลัพธ์ทาง console



ตัวแปรชนิด Array

```
<html>
```

```
<head>
```

```
<script>
```

การกำหนดค่าให้
สมาชิกแต่ละตัว

```
let age = new Array() •———— การประกาศตัวแปรอาร์เรย์ใหม่
```

```
age[0] = 10
```

```
age[1] = 20
```

```
age[2] = 30
```

```
console.log(age.length) // การขอจำนวนสมาชิกของอาร์เรย์
```

```
console.log(age) // การขอข้อมูลทั้งหมดในอาร์เรย์
```

```
console.log(age[1]) // การเข้าถึงข้อมูลสมาชิกแต่ละตัว
```

การกำหนดค่าให้
สมาชิกแต่ละตัว

```
let cars = [] •———— การประกาศตัวแปรอาร์เรย์ใหม่
```

```
cars[0] = 'Ford'
```

```
cars[1] = 'Volvo'
```

```
cars[2] = 'BMW'
```

```
console.log(cars[0])
```

```
</script>
```

```
</head>
```

```
</html>
```

ผลลัพธ์ทาง console

```
3
```

```
▶ (3) [10, 20, 30]
```

```
20
```

```
Ford
```

ประกาศ Array พร้อมกำหนดค่าเริ่มต้น

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<script>
```

```
let age = new Array(10, 20, 30, 40, 50);  
console.log(age);  
console.log(age[3]);
```

กำหนดค่าเริ่มต้นให้
อาร์เรย์ 5 ค่า

```
let score = [2, 4.5, 'three', 'two'];  
console.log(score[0]);  
console.log(score['0']);
```

กำหนดค่าเริ่มต้นให้อาร์เรย์
4 ค่า สามารถมีสมาชิกที่มี
ชนิดข้อมูลแตกต่างกันได้

```
</script>
```

```
</head>
```

```
<body>
```

```
</body>
```

```
</html>
```

ผลลัพธ์ทาง console

▶ (5) [10, 20, 30, 40, 50]
40
2
2

ตัวแปรชนิด Object

```
<html><head>
```

```
<script>
```

```
let person = {
```

```
  id: 69,
```

```
  fullname: 'John Smith',
```

```
  weight: 72.5,
```

```
  option: ['move', 'stop', 'slow']
```

```
}
```

```
console.log(person.weight)
```

```
// เข้าถึง property weight
```

```
console.log(person.option[2])
```

```
// เข้าถึง property option
```

```
person.fullname = 'Robert Smith' // กำหนดค่าใหม่
```

```
console.log(person.fullname)
```

```
</script>
```

```
</head></html>
```

การสร้าง object โดยขึ้นต้นด้วย { และลงท้ายด้วย }

รูปแบบ property

คั่นด้วย : ตามด้วย

value

คั่นสมาชิกด้วย comma

อาร์เรย์ซ้อนใน object

72.5

slow

Robert Smith

ผลลัพธ์ทาง console

Array ของ Object

```
<html>
<head>
<script>
  let student = [           // ประกาศตัวแปรอาร์เรย์
    {                       // สร้าง object แรก
      id: 62001,
      thainame: 'มานะ'
      fullname: 'mana'
    },                     // สร้าง object ที่สอง
    { id: 62002,
      fullname: 'manee'
    }
  ]
```

ผลลัพธ์ทาง console

mana
62002
phiti
>>

```
console.log(student[0].fullname) // เข้าถึง object แรก property fullname
console.log(student[1].id)       // เข้าถึง object ที่สอง property id
student[1].fullname = 'phiti' // กำหนดค่าใหม่
console.log(student[1].fullname)

</script>
</head>
</html>
```

ตัวดำเนินการ (Operator)

เช่น กำหนดค่าเริ่มต้นให้ $y=5$

Operator	คำอธิบาย	ตัวอย่าง	ค่า x	ค่า y
+	การบวก	$x=y+2$	7	5
-	การลบ	$x=y-2$	3	5
*	การคูณ	$x=y*2$	10	5
/	การหาร	$x=y/2$	2.5	5
%	การหารเอาเศษ (Modulo)	$x=y\%2$	1	5
++	เพิ่มค่าหนึ่งค่าให้กับตัวแปร	$x=++y$	6	6
		$x=y++$	5	6
--	ลดค่าหนึ่งค่าให้กับตัวแปร	$x=--y$	4	4
		$x=y--$	5	4

การใช้ + กับ String

- การต่อ String

```
txt1 = 'What a very'  
txt2 = 'nice day'  
txt3 = txt1 + txt2  
console.log(txt3)
```

What a verynice day

หรือ

```
txt1 = 'What a very'  
txt2 = 'nice day'  
txt3 = txt1 + ' ' + txt2  
console.log(txt3)
```

What a very nice day

- ใช้ + ระหว่าง String และตัวเลข

```
x = 5 + 5 //10  
y = '5' + 5 //55  
z = 'Hello' + 5 //Hello5
```

ตัวดำเนินการกำหนดค่า (Assignment Operators)

เช่น กำหนดค่าเริ่มต้นให้ $x=10$ และ $y=5$

Operator	ตัวอย่าง	เขียนแบบเต็ม	ผลลัพธ์
$=$	$x = y$		$x=5$
$+=$	$x += y$	$x = x + y$	$x=15$
$-=$	$x -= y$	$x = x - y$	$x=5$
$*=$	$x *= y$	$x = x * y$	$x=50$
$/=$	$x /= y$	$x = x / y$	$x=2$
$\% =$	$x \% = y$	$x = x \% y$	$x=0$

ตัวดำเนินการเปรียบเทียบ (Comparison Operators)

เช่น กำหนดค่าเริ่มต้นให้ $x=5$

Operator	คำอธิบาย	Comparing	ผลลัพธ์
==	เท่ากัน	$x==8$	<i>false</i>
		$x==5$	<i>true</i>
===	เท่ากันทั้งค่าและชนิดข้อมูล (exactly equal to)	$x==='5'$	<i>false</i>
		$x===5$	<i>true</i>
!=	ไม่เท่ากัน	$x!=8$	<i>true</i>
!==	ไม่เท่ากัน ค่าต่างกัน หรือ ชนิดข้อมูลต่างกัน	$x!== '5'$	<i>true</i>
		$x!==5$	<i>false</i>
>	มากกว่า	$x>8$	<i>false</i>
<	น้อยกว่า	$x<8$	<i>true</i>
>=	มากกว่าเท่ากับ	$x>=8$	<i>false</i>
<=	น้อยกว่าเท่ากับ	$x<=8$	<i>true</i>

ตัวดำเนินการตรรกะ (Logical Operators)

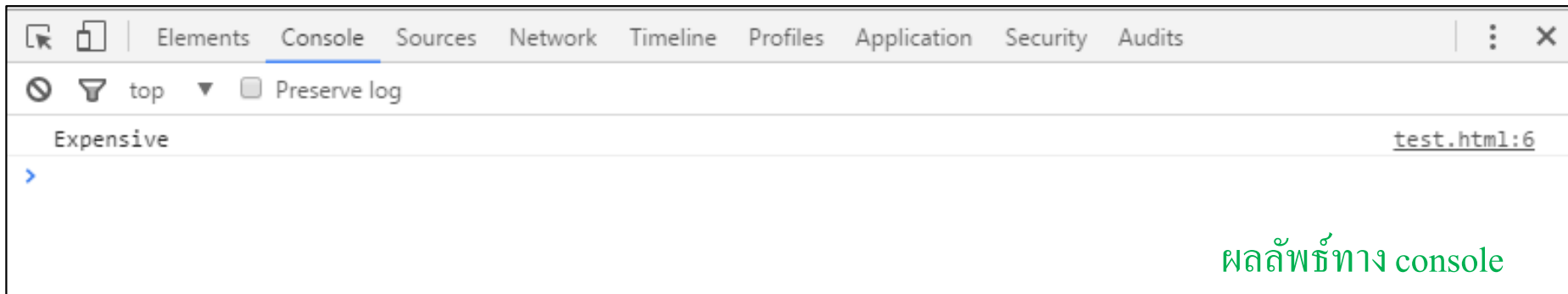
เช่น กำหนดค่าเริ่มต้นให้ $x=6$ และ $y=3$

Operator	ความหมาย	ตัวอย่าง
&&	and	$(x < 10 \ \&\& \ y > 1)$ is true
	or	$(x==5 \ \ y==5)$ is false
!	not	$!(x==y)$ is true

คำสั่งเงื่อนไข if

```
if (condition) {  
    statement  
}
```

```
<html>  
<body>  
<script>  
    let price = 1500  
    if (price > 1000) {  
        console.log('Expensive')  
    }  
</script>  
</body>  
</html>
```



ผลลัพธ์ทาง console

คำสั่งเงื่อนไข if...else

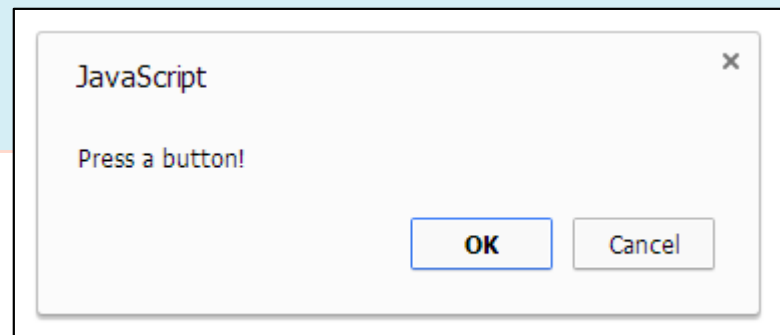
```
if (condition) {  
    statement1  
  
} else {  
    statement2  
}
```

```
<html>  
<body>  
<script>  
    let price = 1500  
    if (price > 1000) {  
        console.log('Expensive')  
    } else {  
        console.log('Cheap')  
    }  
</script>  
</body>  
</html>
```

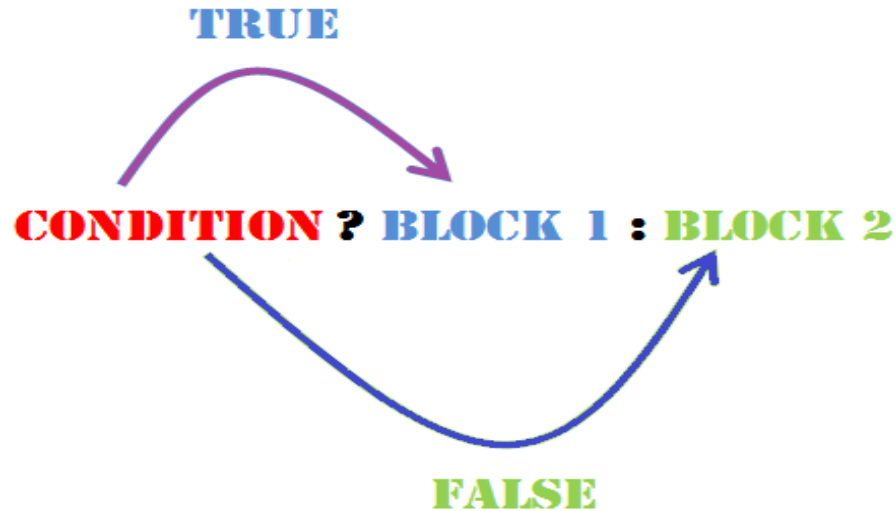
ตัวอย่าง

```
<html>
<body>
<script>
  let r = confirm('Press a button!')
  if (r == true)
    console.log('OK')
  else
    console.log('cancel')
</script>
</body>
</html>
```

การใช้กล่องยืนยัน (Confirm Box) จะ return ค่า *true* เมื่อผู้ใช้กด OK, *false* เมื่อผู้ใช้กด Cancel



เงื่อนไข if...else แบบย่อ



```
<html>
<body>
<script>
  let age = 15
  let level = (age<18) ? 'Young' : 'Old'
  console.log(level)
</script>
</body>
</html>
```

คำสั่งเงื่อนไข if...else if...

```
if (condition 1){  
    statement1  
  
} else if (condition 2){  
    statement2  
  
} else if (condition n){  
    statement3  
  
} else {  
    other_statement  
}
```

```
<html>  
<body>  
<script>  
    let score = 65  
    if (score >= 80)  
        console.log('A')  
    else if (score >= 70)  
        console.log('B')  
    else if (score >= 60)  
        console.log('C')  
    else if (score >= 50)  
        console.log('D')  
    else  
        console.log('F')  
</script>  
</body>  
</html>
```

คำสั่งเงื่อนไข switch-case

```
switch (ตัวแปรชนิดใดก็ได้)
```

```
{
```

Colon

```
case <ตัวเลข,String>:
```

```
    statement
```

```
    break
```

คำสั่ง break

```
case <ตัวเลข,String>:
```

```
    statement
```

```
    break
```

```
...
```

```
default:
```

```
    statement
```

```
}
```

```
<html><body>
```

```
<script>
```

```
let test = 'male'
```

```
switch (test) {
```

```
case 1:
```

```
    console.log('Number!!')
```

```
    break
```

```
case 3.14:
```

```
    console.log('Floating Point!!')
```

```
    break
```

```
case 'male':
```

```
    console.log('String!!')
```

```
    break
```

```
case 'female':
```

```
    console.log('String!!')
```

```
    break
```

```
default:
```

```
    console.log('Other!!')
```

```
}
```

```
</script>
```

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

คำสั่งวนซ้ำ while

```
while (condition) {  
    statement  
}
```

```
<html><head>
```

```
<script>
```

```
let cars = [ 'Ferrari', 'Benz', 'BMW', 'Mazda', 'Toyota', 'Honda' ]
```

```
let i = 0, length = cars.length
```

```
while (i < length) {
```

```
    console.log(cars[i])
```

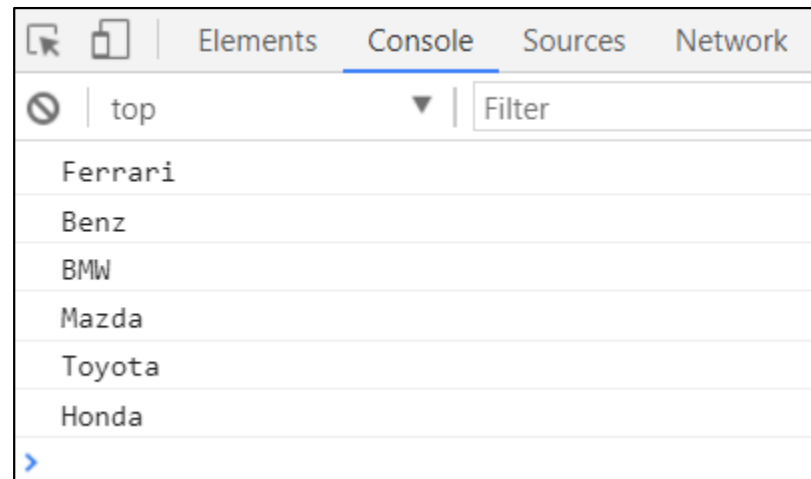
```
    i++
```

```
}
```

```
</script>
```

```
</head></html>
```

ผลลัพธ์ทาง console



คำสั่งวนซ้ำ for

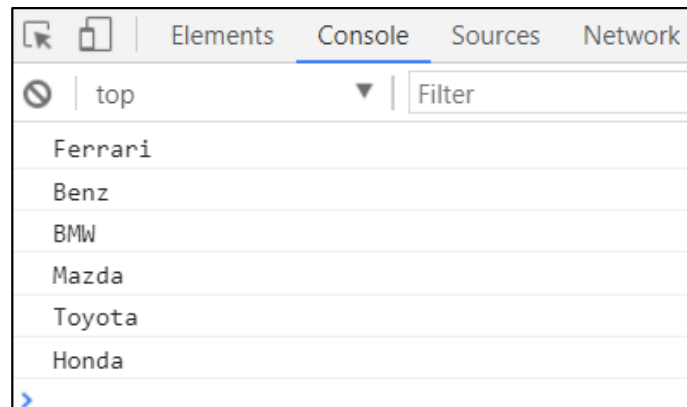
```
for ( กำหนดค่าเริ่มต้น; เงื่อนไข; คำสั่งเพิ่ม/ลดค่า ) {  
    statement  
}
```

หรือ

```
for ( ตัวแปรควบคุม in ชื่ออาร์เรย์ ) {  
    statement  
}
```

```
<html><head>  
<script>  
    let cars =['Ferrari','Benz','BMW',  
                'Mazda','Toyota','Honda']  
    for (let i = 0; i<cars.length; i++){  
        console.log(cars[i])  
    }  
</script>  
</head></html>
```

```
<html><head>  
<script>  
    let cars =['Ferrari','Benz','BMW',  
                'Mazda','Toyota','Honda']  
    for(let i in cars){  
        console.log(cars[i])  
    }  
</script>  
</head></html>
```



ผลลัพธ์ทาง console

การสร้างฟังก์ชัน

```
<html>  
<head>  
<script>
```

ฟังก์ชันควรเขียนในส่วน *<head>* เสมอ

```
function ชื่อฟังก์ชัน(พารามิเตอร์1, พารามิเตอร์2, พารามิเตอร์N, ...) {
```

```
    ชุดคำสั่งต่างๆ
```

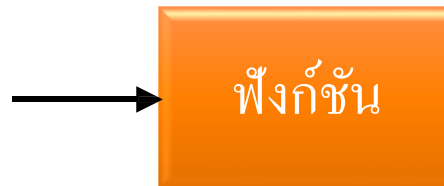
```
    return [ชื่อตัวแปรที่ส่งค่ากลับ]
```

```
}
```

```
</script>  
</head>
```

```
<body>  
    ...  
</body>  
</html>
```

ตัวแปรรับเข้า



ฟังก์ชัน

ค่าส่งกลับ

การเรียกใช้ฟังก์ชัน

เรียกใช้เมื่อโหลดเว็บ

```
<html>
<head>
<script>
  function myFunction() {
    ...
  }

  myFunction() //เรียกใช้ฟังก์ชัน
</script>
</head>
<body>
  Hello world
  <script>
    myFunction() //เรียกใช้ฟังก์ชัน
  </script>
</body>
</html>
```

เรียกใช้ตามเหตุการณ์

```
<html>
<head>
<script>
  function myFunction() {
    ...
  }
</script>
</head>
<body>
  Hello world
  <!-- เรียกใช้ฟังก์ชันในเหตุการณ์ต่างๆ -->

  <a href="#" onclick="myFunction()">Execute</a>
  <input type="button" onclick="myFunction()" value="Click">
  <h2 onmouseover="myFunction()">Hello</h2>
</body>
</html>
```

การเรียกใช้ฟังก์ชัน จะใช้ชื่อฟังก์ชันที่นิยามตามด้วยวงเล็บ ()
สามารถใส่ค่า หรือตัวแปรที่จะส่งให้ในวงเล็บ คั่นด้วย comma

ฟังก์ชันรูปแบบต่างๆ

แบบไม่รับและไม่ส่งกลับข้อมูล

hello

แบบรับข้อมูล แต่ไม่ส่งข้อมูลกลับ

name

printGreeting

แบบรับและส่งกลับข้อมูล

num1

num2

num3

findAverage

ค่าเฉลี่ย

แบบไม่รับและไม่ส่งกลับข้อมูล

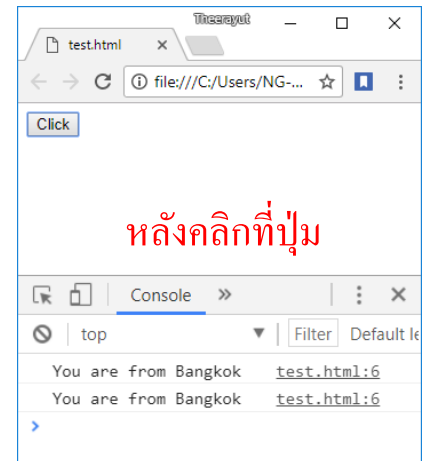
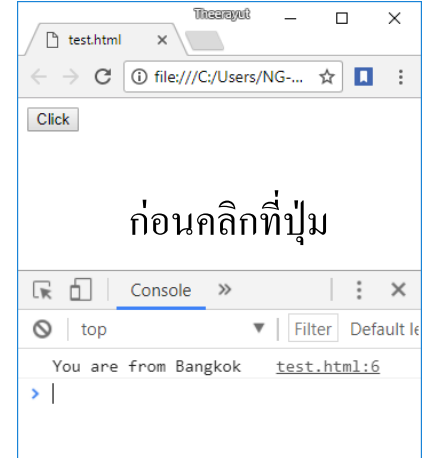
```
<html>
<head>
<script>
  let address = 'Bangkok' // ตัวแปรชนิด Global

  function hello() {      // ไม่มีการรับข้อมูล
    console.log('You are from ' + address)
  }

  hello() // เรียกใช้ฟังก์ชัน hello() ก่อนโหลดหน้าเว็บ
</script>
</head>

<body>
  <input type="button" onclick="hello()" value="Click">
</body>
</html>
```

เรียกใช้ฟังก์ชัน *hello()* เมื่อเกิดเหตุการณ์คลิก



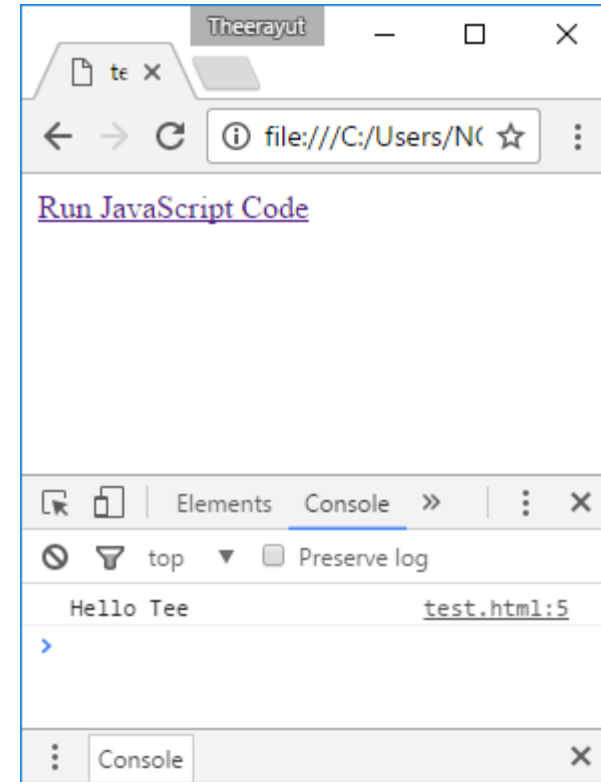
แบบรับข้อมูล แต่ไม่ส่งข้อมูลกลับ

```
<html>
<head>
<script>
    function printGreeting(name) {
        console.log('Hello ' + name)
    }
</script>
</head>
<body>
    <a href="#" onclick="printGreeting('Tee')">
        Run JavaScript Code
    </a>
</body>
</html>
```

รับข้อมูลเข้า

ส่งข้อมูลไปยังฟังก์ชัน

เรียกใช้ฟังก์ชัน



แบบรับและส่งกลับข้อมูล

```
<html>
```

```
<head>
```

```
<script>
```

```
function findAverage(num1, num2, num3) {
```

```
    let sum = num1 + num2 + num3
```

```
    let avg = sum/3
```

```
    return avg
```

```
}
```

```
let ans = findAverage(3, 7, 6)
```

```
alert(ans)
```

```
</script>
```

```
</head>
```

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

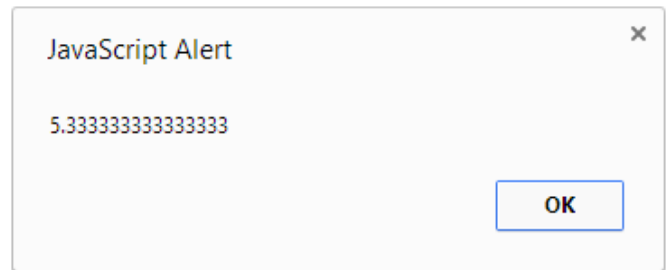


รับข้อมูลเข้า (parameter)

ข้อมูลที่ส่งกลับ

เรียกใช้ฟังก์ชัน

ส่งข้อมูลไปยังฟังก์ชัน (argument)



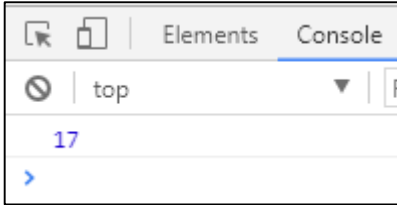
การสร้างฟังก์ชันเก็บในตัวแปร

- ฟังก์ชันใน JavaScript สามารถนิยาม และเก็บผลลัพธ์ไว้ในตัวแปรได้ ซึ่งจะ
เป็นฟังก์ชันที่ไม่มีชื่อ เรียกว่า **Anonymous Function**

```
<html>
<head>
  <script>
    let p = function(a, b) {
      return a + b
    }
  </script>
</head>
<body>
  <script>
    let result = p(8, 9)
    console.log(result)
  </script>
</body>
</html>
```

เก็บฟังก์ชันไว้ในตัวแปร *p*

เรียกฟังก์ชันผ่านชื่อตัวแปร *p* พร้อมกับส่ง *argument*



The screenshot shows a web browser's developer console with the 'Console' tab selected. It displays a single log entry with the value '17' in purple text, indicating the result of the function call p(8, 9).

การสร้างฟังก์ชันเก็บใน object

- ฟังก์ชันใน JavaScript สามารถนิยาม และเก็บไว้ในตัวแปรชนิด object ได้ และสามารถเรียกใช้ ผ่านชื่อ object

```
<html>
<head>
  <script>
    let t = {
      id:69,
      fname:'mana',
      lname:'pakpian',
      say: function(name) { return'Hello ' + name }
    }
    console.log( t.id + ' ' + t.fname+ ' ' + t.lname)
    console.log( t.say('phiti') )
  </script>
</head>
<body></body>
</html>
```

เก็บฟังก์ชันไว้
ใน key ชื่อ say



เรียกฟังก์ชันผ่านชื่อ object พร้อมกับส่ง argument

Built-in Object

- Built-in Object คือ object มาตรฐานในภาษา JavaScript ภายในประกอบด้วย

1. **Property** ใช้ในการกำหนดหรือขอค่าคุณสมบัติของ built-in object

รูปแบบการเรียกใช้

[ชื่อ Object].[ชื่อ property]

2. **Method** คือ ฟังก์ชัน หรือชุดคำสั่งพร้อมใช้สำหรับทำงานกับ built-in object นั้น

รูปแบบการเรียกใช้

ฟังก์ชัน = เมธอด (Method)

[ชื่อ Object].[ชื่อฟังก์ชัน] (รายการ argument)

ตัวอย่าง Built-in Object

- **Number** - parseInt(), parseFloat(), isNaN(), isInteger()
- **Math** - cos(), exp(), log(), max(), min(), sqrt()
- **String** - search(), substr(), replace()
- **Date** - getDate(), getHours(), getMinutes()
- **Array** ประกอบด้วย property เช่น length ใช้ในการขอจำนวนสมาชิกของ Array

ดูรายการ Built-in Object พร้อมตัวอย่างการใช้ได้ที่

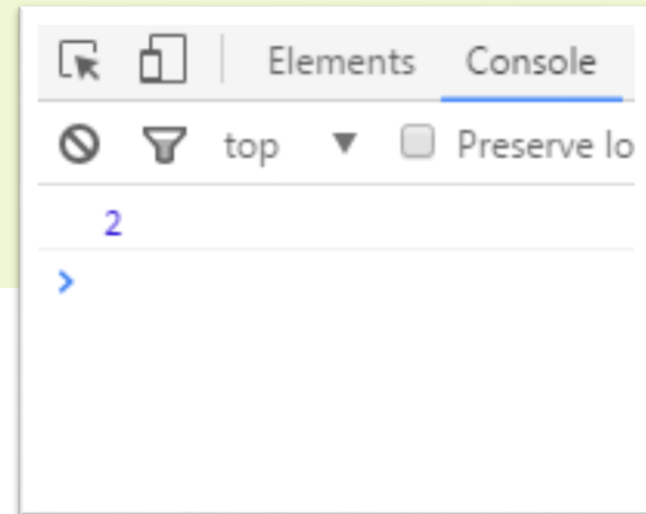
<http://devdocs.io/javascript>

Math Object Example

```
<html>
<body>

<script>
  let a = Math.min(5, 10, 8, 7, 2, 6)
  console.log(a)
</script>

</body>
</html>
```



String Object Example1/3

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

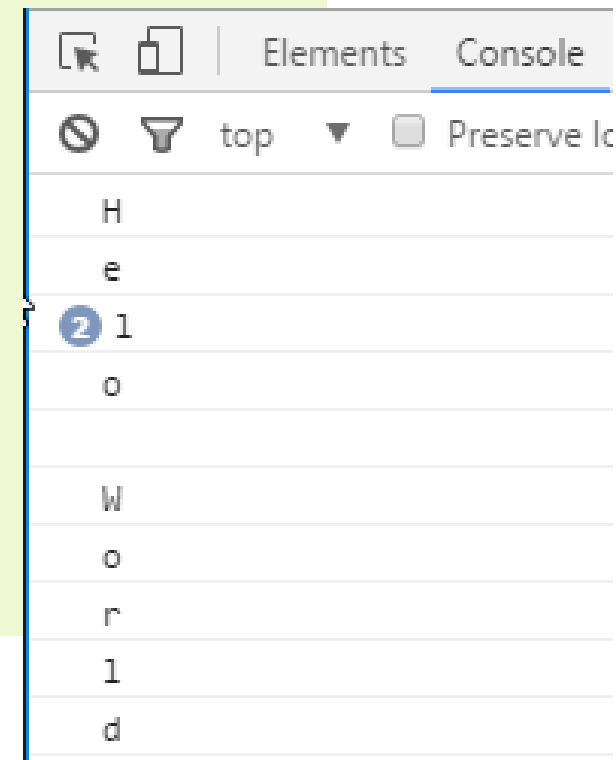
```
let str = 'Hello World'
```

```
for (let i=0; i<str.length; i++) {
  console.log(str.charAt(i))
}
```

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

*Property length ของ String Object ใช้
เป็นเงื่อนไขในลูป*

*charAt() ของ String Object ใช้ขออักขระ
แต่ละตัวภายใน String ซึ่งระบุ argument
เป็นตำแหน่งอักขระ เริ่มจากตำแหน่งที่ 0*



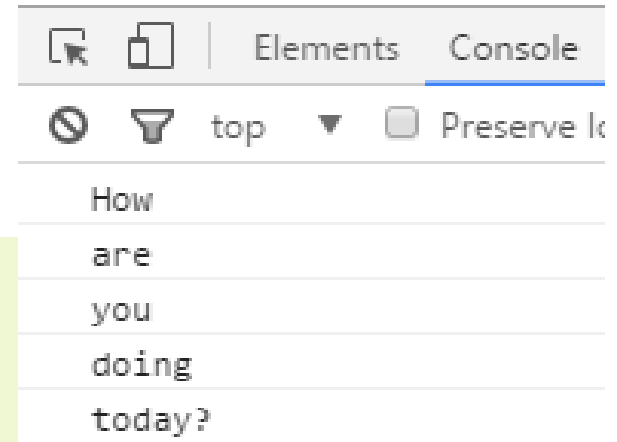
String Object Example2/3

```
<html>
<body>

<script>
  let str = 'How are you doing today?'
  let result = str.split(' ')

  for (let i=0; i<result.length; i++)
    { console.log(result[i])
  }
</script>

</body>
</html>
```



split() ของ String Object ใช้ในการแยก String ที่คั่นด้วยข้อความที่กำหนด จากตัวอย่างคือ แยก String ทั้งหมดที่คั่นด้วยช่องว่าง มีการส่งกลับเป็น Array ของ String

String Object Example3/3

```
<html>
<body>
```

```
<script>
```

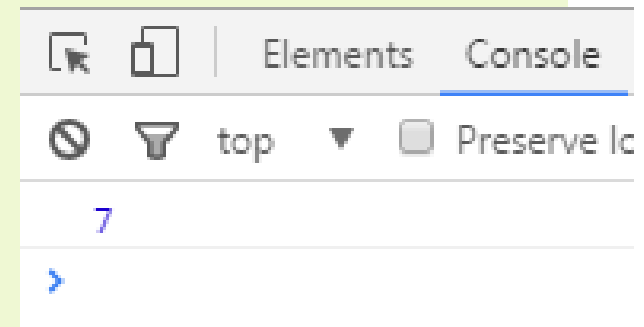
```
  let str = 'เด็กภาคคอม!'
  let n = str.search('คอม')
  console.log(n)
```

```
</script>
```

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

ตัวแปรต่าง ๆ ของ JavaScript ที่เป็น Object
จะสามารถเข้าถึง method หรือ property ได้ทันที

`search()` ของ String Object ใช้ค้นหา
String โดยระบุ argument เป็น String ที่
ต้องการค้นหา และมีการส่งกลับเป็น
ตำแหน่งอักขระแรกที่พบ



Workshop

จงศึกษา built-in object ชื่อ **Date** จาก <http://devdocs.io/javascript/>
แล้วสร้างสคริปต์เพื่อแสดงวันที่ และเวลาปัจจุบันออกจาก Console ดังนี้

```
<html>
<body>
<script>
    let d = new Date()

    _____

    _____

    _____

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

วันนี้ 15/9/2563

เวลา 9.52 น.

> |

เราใช้ JavaScript (JS) ในงาน Front End เพื่อทำอะไร?

- JavaScript can ...
 - change all the HTML elements in the page
 - change all the HTML attributes in the page
 - change all the CSS styles in the page
 - remove existing HTML elements and attributes
 - add new HTML elements and attributes
 - reaction to all existing HTML events in the page
- ทั้งนี้เพื่อลดการติดต่อกับ Backend

Top 10 JavaScript Frameworks to Learn in 2020

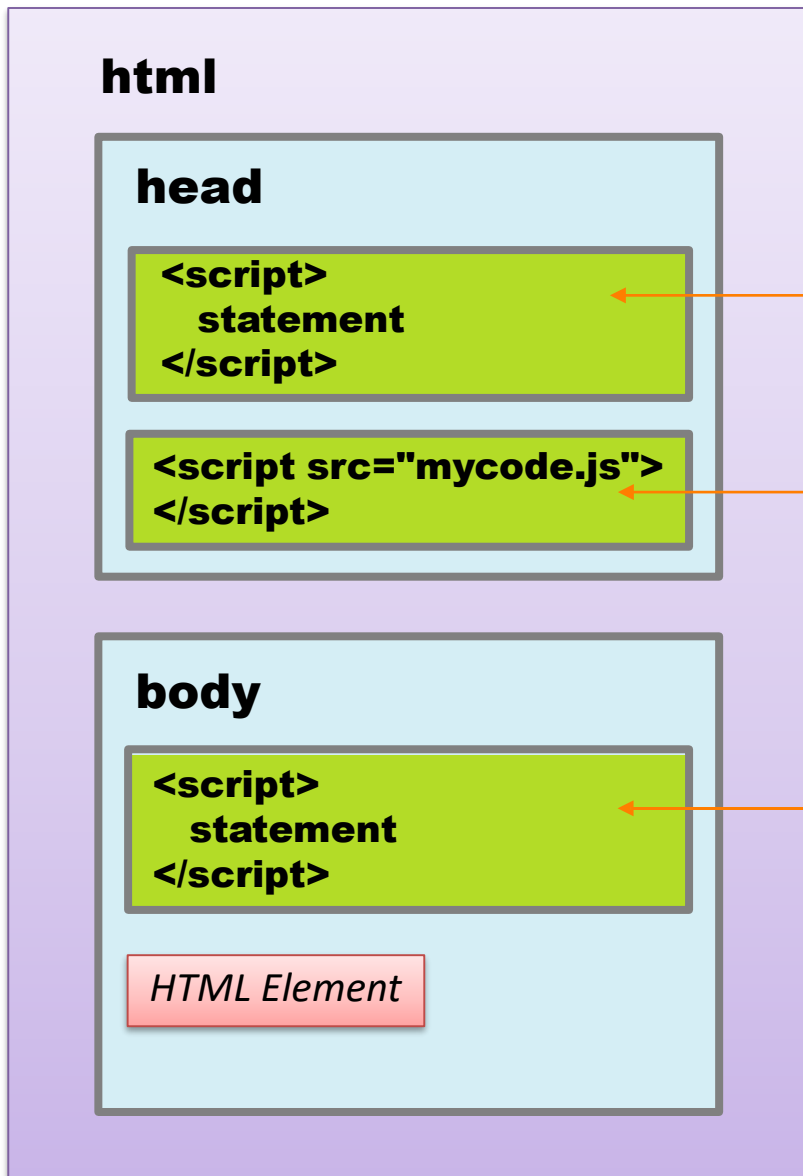


ฝึกฝนเพิ่มทักษะต่อยอดไปยัง Frameworks ต่างๆ ได้ เมื่อต้อง Implement งานที่มี Scale ใหญ่ขึ้น

JS กับงาน Front End

- JS ต้องเข้าถึงโครงสร้างของหน้าเว็บได้จึงจะควบคุมงาน Front End ได้
- DOM = Document Object Model
- JS อ้างถึง HTML Element ด้วย ค่า id
- JS จะอ้างถึง HTML Element ใดได้ก็ต่อเมื่อ Element นั้นถูกโหลดมาเรียบร้อยแล้ว ดังนั้นหากพิจารณาตำแหน่งของโค้ด JS แล้ว....จะพบว่า

ตำแหน่งของโค้ด JS....



แทรกไว้ในส่วนของ **<head>**

Script จะทำงานก่อนที่จะแสดงหน้าเว็บ

แยกเป็นไฟล์ต่างหาก ที่มีนามสกุล *.js*
แล้วอ้างอิงโดย *Relative Path*

แทรกไว้ในส่วน **<body>** จะทำงานขณะที่แสดง
หน้าเว็บตามลำดับจากบนลงล่าง

ตำแหน่งของโค้ด JS

- “JS จะอ้างอิงถึง *HTML Element* ใดได้ก็ต่อเมื่อ *Element* นั้นถูกโหลดมาเรียบร้อยแล้ว”
- เขียนปนกันในส่วน Body จะทำให้โค้ดดูไม่เป็นสัดส่วน ลำบากต่อการปรับปรุงในอนาคต
- เขียนในส่วน head tag ปัญหาคือ...
- วิธีการแก้ปัญหาคือกำหนดการกระทำที่ต้องการ ให้เกิดขึ้นหลังจาก *HTML Elements* ทั้งหมดของหน้าเว็บถูกโหลดมาจนครบแล้ว
- โดยกำหนดไว้ใน Event ชื่อ window.onload

```
<script>
    document.getElementById('today').innerHTML='<b>Today is beautiful</b>';
</script>

<body>
    Hello <span id="today"></span>
</body>
```

JS จะอ้างถึง HTML Element ใดได้ก็ต่อเมื่อ Element นั้นถูกโหลดมา
เรียบร้อยแล้ว

```
<body>
    Hello <span id="today"></span>
    <script>
        document.getElementById('today').innerHTML='<b>Today is beautiful</b>';
    </script>

</body>
```

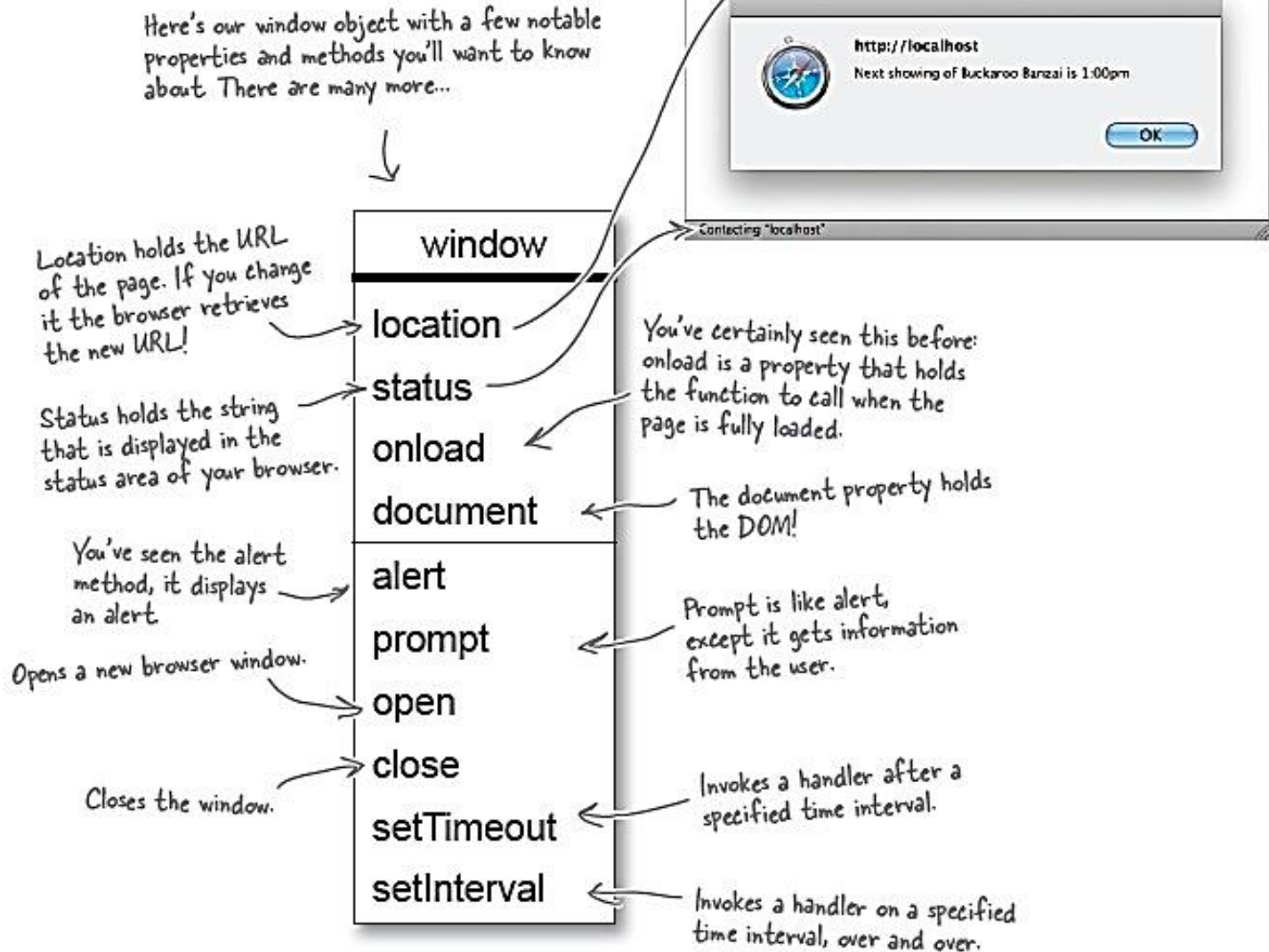
เขียนปนกันในส่วน **Body** จะทำให้โค้ดดูไม่เป็นสัดส่วน ลำบากต่อการปรับปรุงในอนาคต

window.onload

- หากเขียนโค้ดแยกเป็นไฟล์ต่างหาก เหมือนกับเขียน แยก CSS ไว้เรียกใช้ ก็
จะเกิดข้อผิดพลาด เนื่องจาก <head> มาก่อน <body>
- ใช้ Event ชื่อ window.onload

```
<html>
<head>
  <meta charset="UTF-8">
  <script>
    window.onload = function() {
      document.getElementById('today').innerHTML='<b>Today is beautiful</b>';
    }
  </script>
</head>
<body>
  Hello <span id="today"></span>
</body>
</html>
```

Window Object



Window Property

```
<html>
<head>
  <script>
    window.location = 'http://www.google.com/'
  </script>
</head>
<body></body>
</html>
```



กำหนดค่าใหม่ให้กับ *property location* จะเกิดการส่ง
ต่อ (*Redirect*) ไปยัง URL ใหม่

open() ของ Window object

ใช้เมธอด *open* ใน *window object* เพื่อเปิดหน้าต่างเว็บบน
หน้าต่างใหม่ ตาม URL ที่กำหนด

```
<html>
<head>
<script>
  function openPopup() {
    window.open('http://www.google.com', '', 'width=600,height=250')
  }
</script>
</head>

<body>
  <input type="button" value="Open!!" onClick="openPopup()">
</body>

</html>
```

Window Method

ใช้ `open()` ของ `window object` เพื่อเปิดหน้าเว็บบนหน้าตาใหม่ แล้วอ้างถึง `object` นั้นด้วยตัวแปร `winObj`

```
<html>
<head>
<script>
  function openPopup() {
    let winObj = window.open('', '', 'width=300,height=150')
    winObj.document.write('<html><body><h1>Sample Text</h1></body>')
  }
</script>
</head>
<body>
  <input type="button" value="Open!!" onClick="openPopup()">
</body>
</html>
```

เข้าถึง `document object` ซึ่งอยู่ภายใต้ `window object` และเรียกฟังก์ชัน `write` เพื่อเขียนคำสั่ง `HTML` ลงบนหน้าตาใหม่