Chapter 3

JavaScript (JS)

Theerayut Thongkrau

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
  - ขณะโหลดหน้าเว็บเสร็จแล้ว
  - ขณะคลิกที่ปุ่ม
  - 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 ECMAScript 2015 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>** ภายใต้แท็ก <head> เมื่อต้องการให้โหลด หรือทำงานก่อนการแสดงผล หรือภายใต้แท็ก <body> เมื่อต้องการให้ทำงานในช่วงแสดงผล

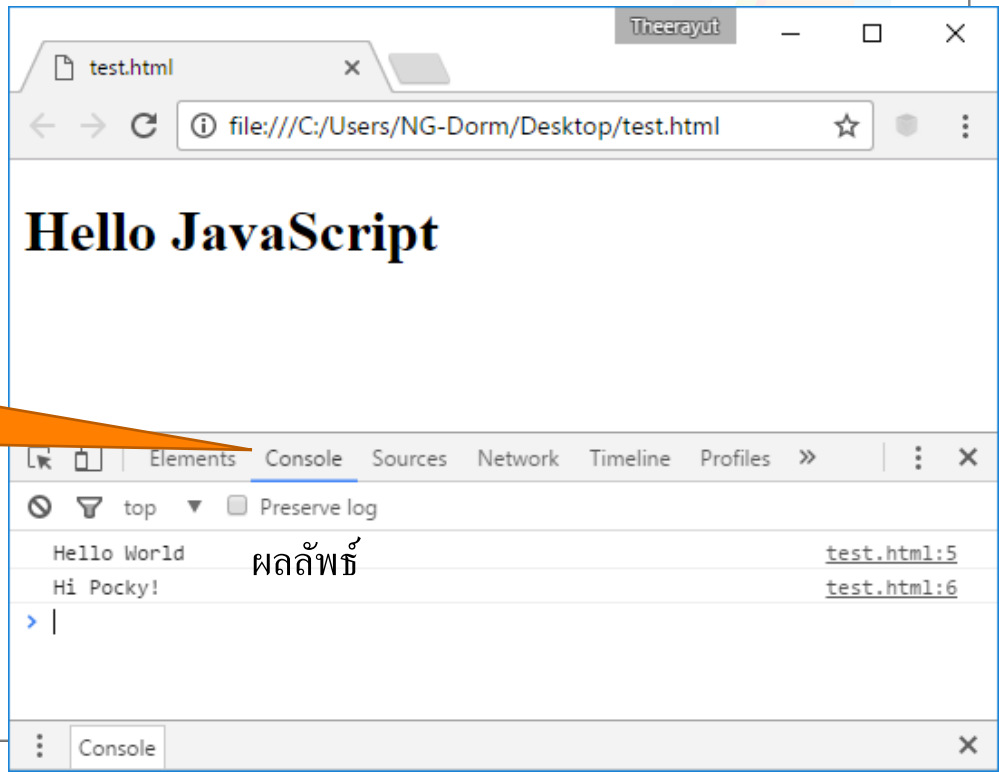
- **External Script** - นำคำสั่ง JavaScript **ในไฟล์แยกต่างหาก** แล้วอ้างถึงไฟล์ในแท็ก <script> ซึ่งอยู่ภายใต้แท็ก <head> เหมาะกับการแชร์คำสั่งหรือฟังก์ชันให้กับเว็บหลายหน้า

Internal Script

```
<!doctype html>
<html>
<head>
    <script>
        console.log('Hello World')
        console.log('Hi Pocky!')
    </script>
</head>
<body>
    <h1>Hello JavaScript</h1>
</body>
</html>
```

เพิ่มคำสั่ง JavaScript ในการแสดงข้อความออกทาง Console

กดปุ่ม F12 และเลือกที่แท็บ Console



ผลลัพธ์

External Script

```
<!doctype html>
<html>
<head>
    <script src="myscript.js"></script>
</head>
<body>
    <h1>Hello JavaScript</h1>
</body>
</html>
```

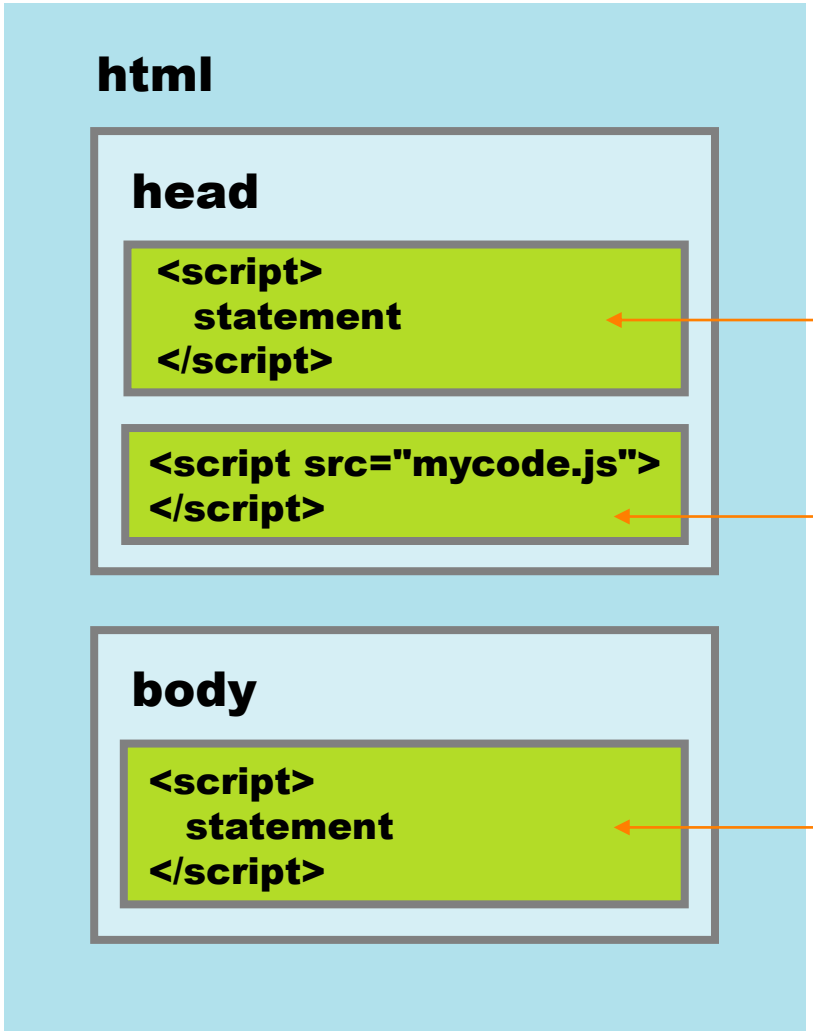
อ้างอิงโดยใช้ Relative URL

ต้องมีแท็กปิดเสมอ

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> จะทำงานขณะที่แสดงหน้าเว็บ
ทำตามลำดับจากบนลงล่าง

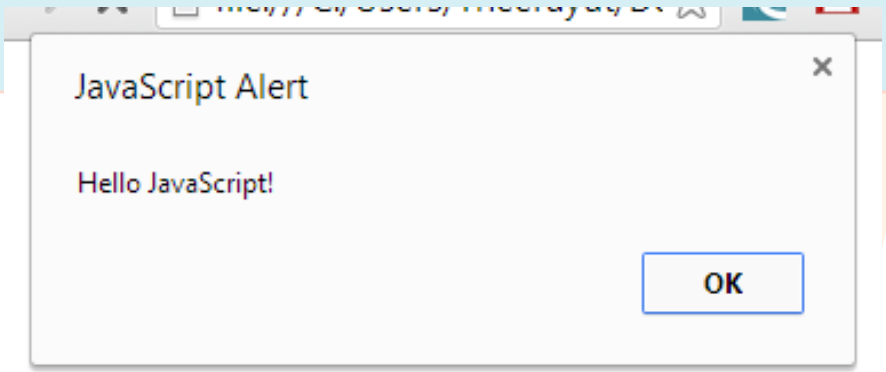
JavaScript Comment

- การอธิบายโปรแกรมใช้รูปแบบเดียวกับภาษาซี

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

    /* แบบอธิบายหลายๆ บรรทัด
      บรรทัดที่ 2
      บรรทัดที่ 3  */
</script>
</head>
<body>
    ...
</body>
</html>
```

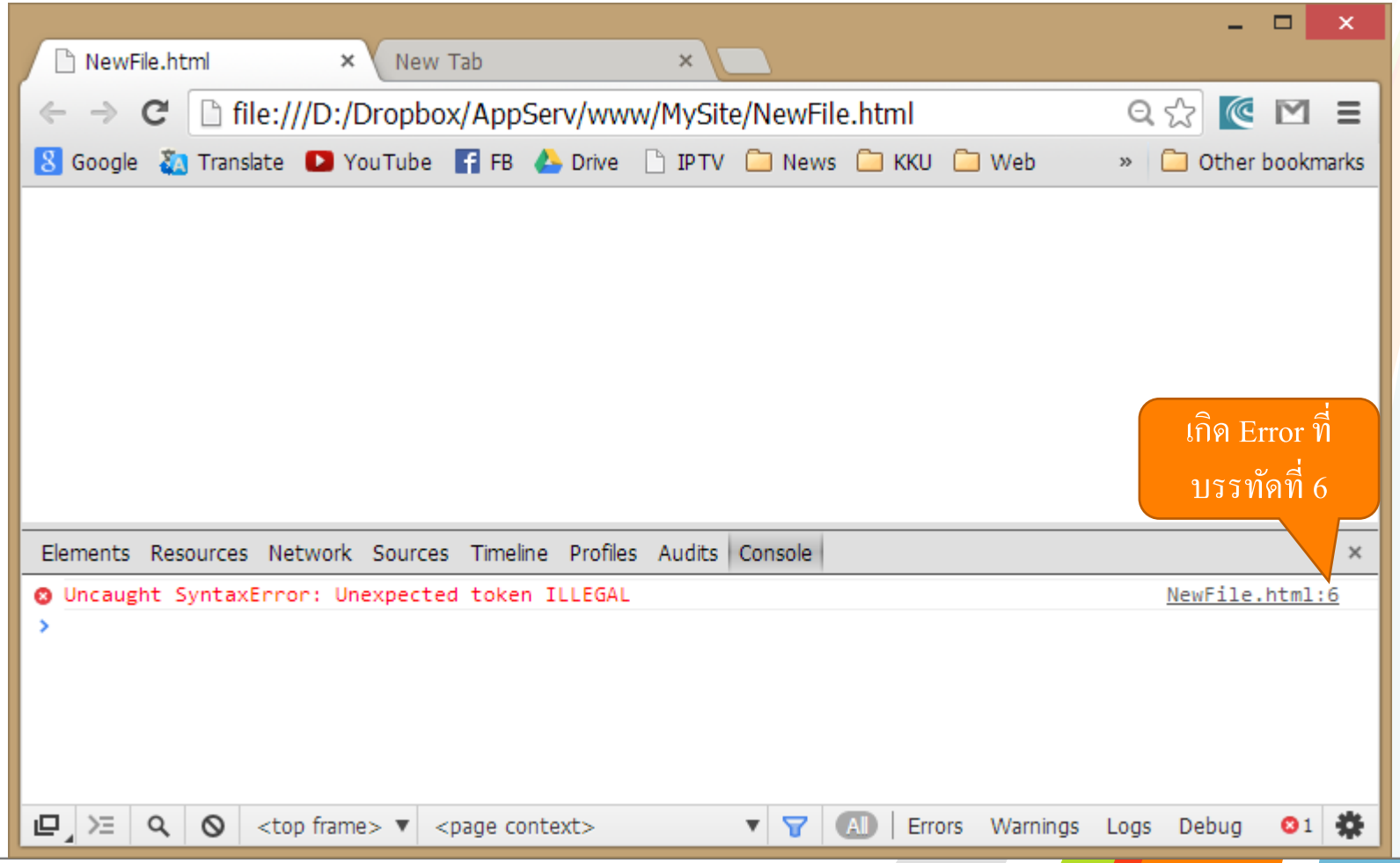
การแสดงข้อความแบบ Alert

```
<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

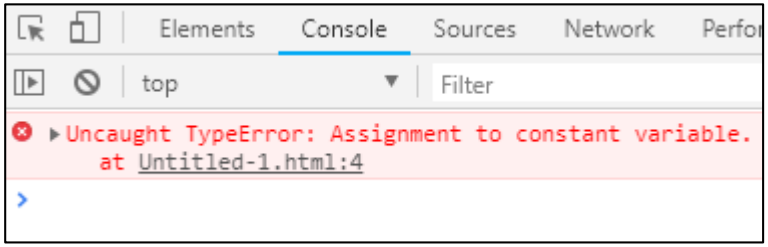
```
let name = 'Johny'
let name2 = 'John'
```
ตัวแปรชนิด string (JavaScript ไม่มีชนิด char) และ String ก็ไม่ได้หมายถึง Array ของ Character

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

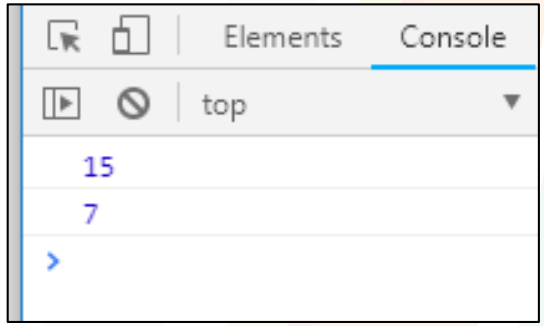
การประกาศตัวแปรแบบ const และ let

- 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>
```



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	

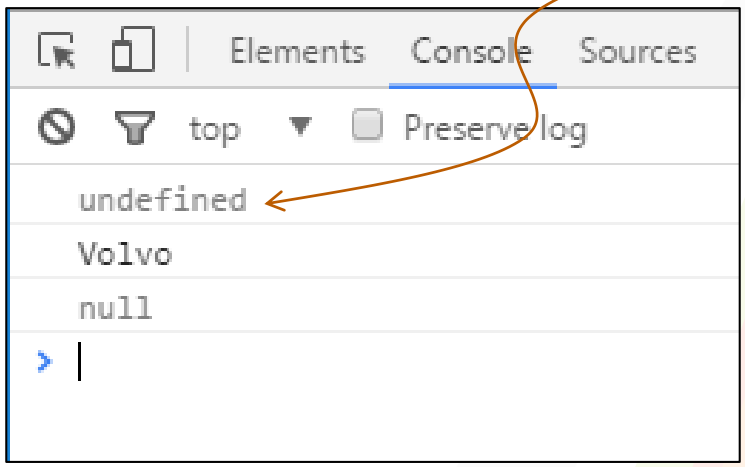
Undefined และ Null

```
<html>
<body>
<script>
    let person
    let car = 'Volvo'
    console.log(person)
    console.log(car)

    let car = null
    console.log(car)
</script>
</body></html>
```

ตัวแปรที่ยังไม่มีการกำหนดค่า จะแสดงเป็น Undefined

การกำหนดค่าว่างให้กับตัวแปรจะใช้ null

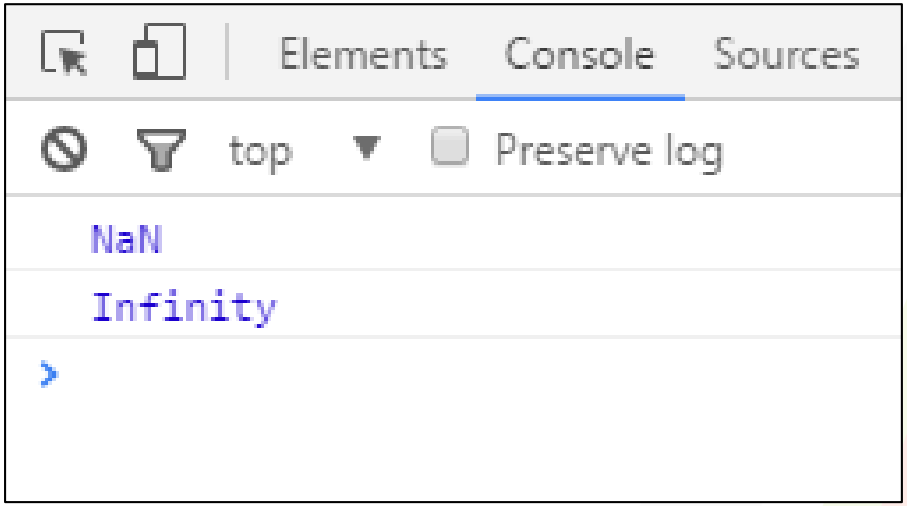
ผลลัพธ์ทาง console



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>
        let age = new Array()
        age[0] = 10
        age[1] = 20
        age[2] = 30
        console.log(age.length) // การขอจำนวนสมาชิกของอาเรย์
        console.log(age)        // การขอข้อมูลทั้งหมดในอาเรย์
        console.log(age[1])     // การเข้าถึงข้อมูลสมาชิกแต่ละตัว

        let cars = []
        cars[0] = 'Ford'
        cars[1] = 'Volvo'
        cars[2] = 'BMW'
        console.log(cars[0])
   </script>
</head>
</html>
```

การประกาศตัวแปรอาร์เรย์ใหม่

การกำหนดค่าให้สมาชิกแต่ละตัว

การประกาศตัวแปรอาร์เรย์ใหม่

การกำหนดค่าให้สมาชิกแต่ละตัว

ผลลัพธ์ทาง console

```
3
▶ (3) [10, 20, 30]
20
Ford
```

ประกาศ Array พร้อมกำหนดค่าเบื้องต้น

```
<!DOCTYPE html>
<html>
<head>
    <script>
        let age = new Array(10, 20, 30, 40, 50)
        console.log(age)
        console.log(age[3])

        let score = [2, 4.5, 'three', 'two']
        console.log(score[0])
        console.log(score['0'])
    </script>
</head>
<body>
</body>
</html>
```

คั่นสมาชิกด้วย comma

กำหนดค่าเริ่มต้นให้อาเรย์ 5 ค่า

กำหนดค่าเริ่มต้นให้อาเรย์ 4 ค่า
สามารถมีสมาชิกที่มีชนิดข้อมูลแตกต่างกันได้

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,
            fullname: 'ธีจัง'
        },
        {                    // สร้าง object ที่สอง
            id: 62002,
            fullname: 'โต๋ยจัง'
        }
    ]

    console.log(student[0].fullname) // เข้าถึง object แรก property fullname
    console.log(student[1].id)   // เข้าถึง object ที่สอง property id

    student[1].fullname = 'พี่โต๋ย'   // กำหนดค่าใหม่
    console.log(student[1].fullname)
</script>
</head>
</html>
```

ผลลัพธ์ทาง console

```
ธีจัง
62002
พี่โต๋ย
```

ตัวดำเนินการ (Operator)

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)

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)

กำหนดค่าเริ่มต้นให้ x=5

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

ตัวดำเนินการตรรกะ (Logical Operators)

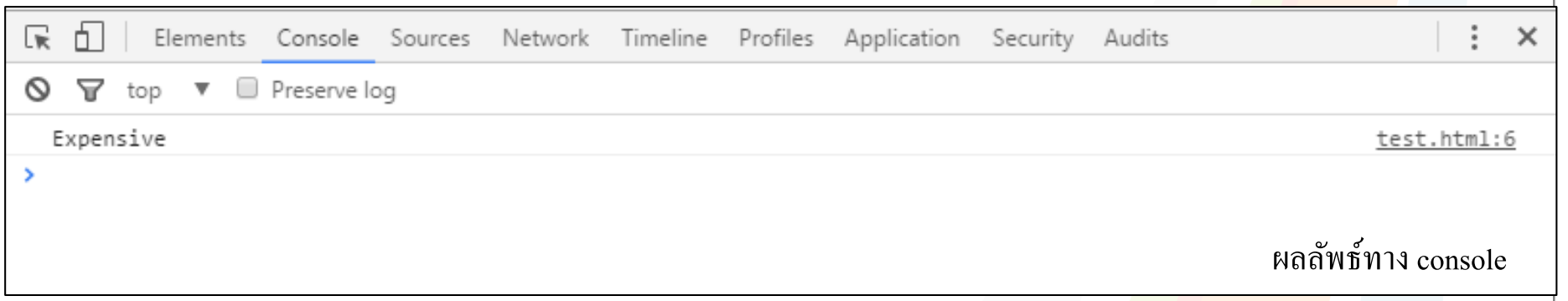
กำหนดค่าเริ่มต้นให้ x=6 และ y=3

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

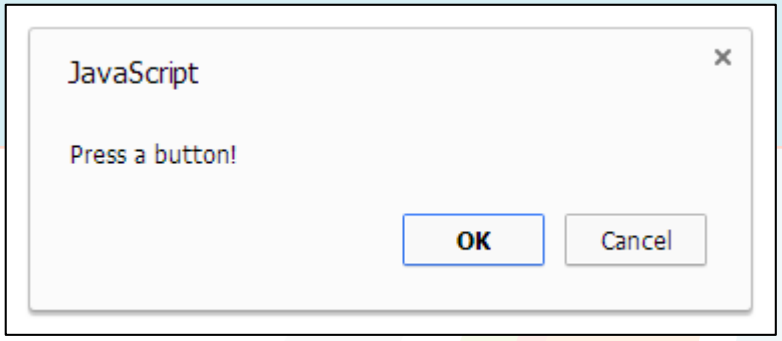
```
if (condition){
      statement1

} else {
      statement2
}
```

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

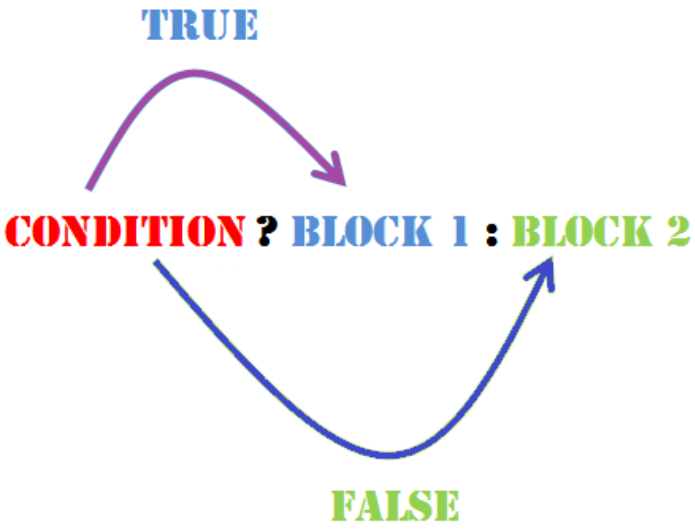
ตัวอย่าง

```
<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)
{
    case <ตัวเลข,String>:
        statement
        break
    case <ตัวเลข,String>:
        statement
        break
     …

    default:
        statement
}
```

Colon

คำสั่ง break

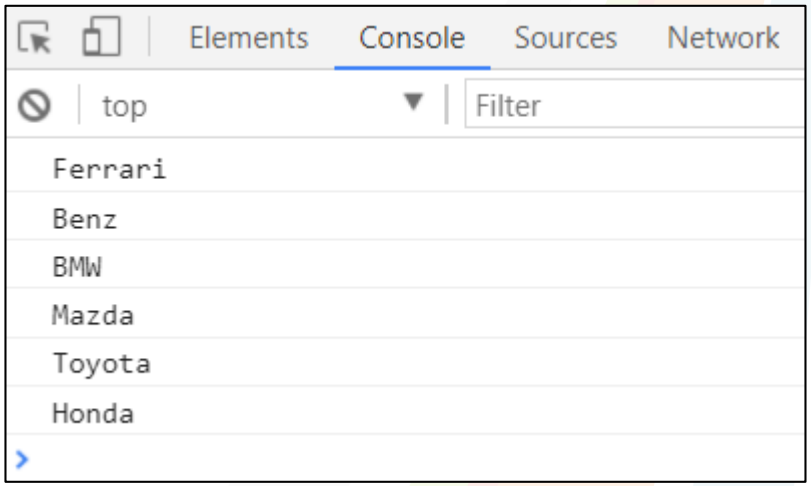
```
<html><body>
<script>
    let test = 'male'
    switch (test) {
      case 1:
        console.log('Number!!')
        break
      case 3.14:
        console.log('Floating Point!!')
        break
      case 'male':
        console.log('String!!')
        break
      case 'female':
        console.log('String!!')
        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
}
```

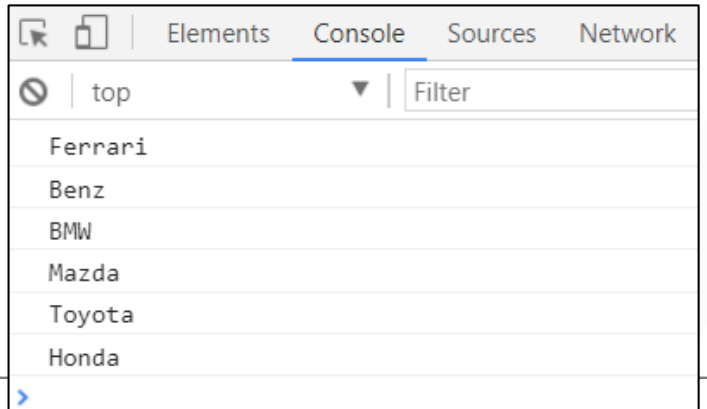
หรือ

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

```
<html><head>
<script>
    let cars =['Ferrari','Benz','BMW',
        'Mazda','Toyota','Honda']
    for (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>
    function ชื่อฟังก์ชัน(พารามิเตอร์1, พารามิเตอร์2, พารามิเตอร์N, ...)  {
        ชุดคำสั่งต่างๆ
        return  [ชื่อตัวแปรที่ส่งกลับ]
    }
</script>
</head>

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

ฟังก์ชันควรเขียนในส่วน <head> เสมอ

การเรียกใช้ฟังก์ชัน

เรียกใช้เมื่อโหลดเว็บ

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

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

เรียกใช้ตามเหตุการณ์

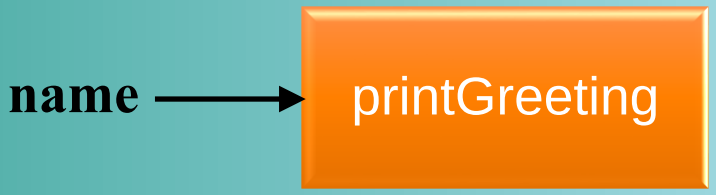
```
<html>
<head>
<script>
   function myFunction() {
      …
   }
</script>
</head>
<body>
    Hello world
    <!-- เรียกใช้ฟังก์ชันในเหตุการณ์ต่างๆ -->
    <a href="#" onclick="myFunction()">Execute</a>
    <input type="button" onclick="myFunction()" value="Click">
    <h2 onmouseover="myFunction()">Hello</h2>
</body>
</html>
```

การเรียกใช้ฟังก์ชัน จะใช้ชื่อฟังก์ชันที่นิยามตามด้วยวงเล็บ ( )
สามารถใส่ค่า หรือตัวแปรที่จะส่งให้ในวงเล็บ คั่นด้วย comma

ฟังก์ชันรูปแบบต่างๆ

แบบไม่รับและไม่ส่งกลับข้อมูล

แบบรับข้อมูล แต่ไม่ส่งข้อมูลกลับ

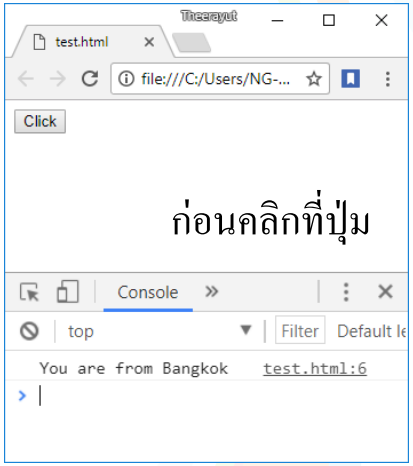


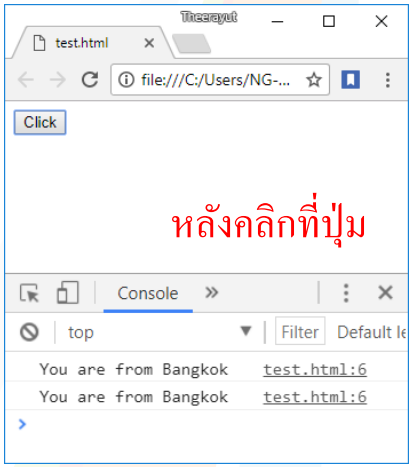
แบบรับและส่งกลับข้อมูล

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





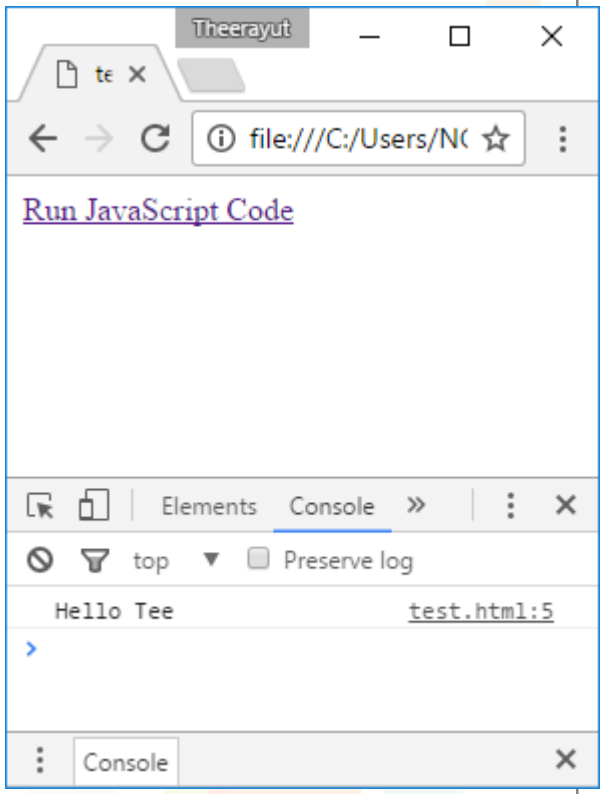
แบบรับข้อมูล แต่ไม่ส่งข้อมูลกลับ

```
<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

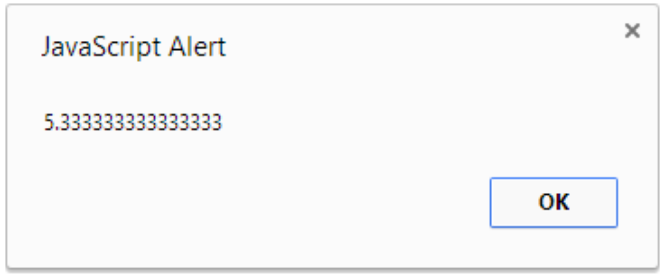

```
<html>
<head>
<script>
    function findAverage(num1, num2, num3) {
        let sum = num1 + num2 + num3
        let avg = sum/3
        return avg
    }
    let ans = findAverage(3, 7, 6)
    alert(ans)
</script>
</head>
<body></body></html>
```

รับข้อมูลเข้า (parameter)

ข้อมูลที่ส่งกลับ

ส่งข้อมูลไปยังฟังก์ชัน (argument)

เรียกใช้ฟังก์ชัน



กิจกรรม

$$\text{BMI} = \frac{Weight\ (kg)}{Height^2\ (m^2)}$$

- จงสร้างฟังก์ชันที่มีรูปแบบดังนี้

  function findBMI(weight, height)

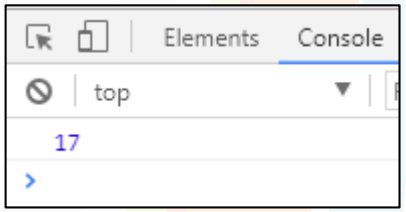
- เรียกใช้ฟังก์ชันที่สร้างขึ้น โดยทดสอบส่งค่าใดๆไปยังฟังก์ชัน

การสร้างฟังก์ชันเก็บในตัวแปร

- ฟังก์ชันใน JavaScript สามารถนิยาม และเก็บไว้ในตัวแปรได้ ซึ่งจะเป็นฟังก์ชันที่ไม่มีชื่อ (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

การสร้างฟังก์ชันเก็บใน object

- ฟังก์ชันใน JavaScript สามารถนิยาม และเก็บไว้ในตัวแปรชนิด object ได้ และสามารถเรียกใช้ได้ ผ่านชื่อ object

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

เก็บฟังก์ชันไว้ใน key ชื่อ say

เรียกฟังก์ชันผ่านชื่อ object พร้อมกับส่ง argument

Built-in Object

- Built-in Object คือ object มาตรฐานในภาษา JavaScript ภายในประกอบด้วย
  - **Property** ใช้ในการกำหนดหรือขอค่าคุณสมบัติของ built-in object

    รูปแบบการเรียกใช้

    [ชื่อ Object].[ชื่อ property]
  - **Method** คือ ฟังก์ชัน หรือชุดคำสั่งพร้อมใช้สำหรับทำงานกับ built-in object นั้น

    รูปแบบการเรียกใช้

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

ตัวอย่าง Built-in Object

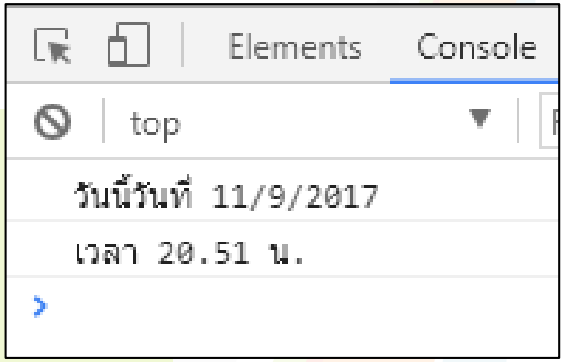
- **Number** - parseInt(), parseFloat(), isNaN(), isInteger()
- **Math** - cos(), exp(), log(), max(), min(), sqrt()
- **String** - search(), substr(), replace()
- **Date** - getDate(), getHours(), getMinutes()
- **Array** ประกอบด้วย property เช่น length ใช้ในการขอจำนวนช่องของ Array

ดูรายการ Built-in Object พร้อมตัวอย่างการใช้ได้ที่

http://devdocs.io/javascript

กิจกรรม

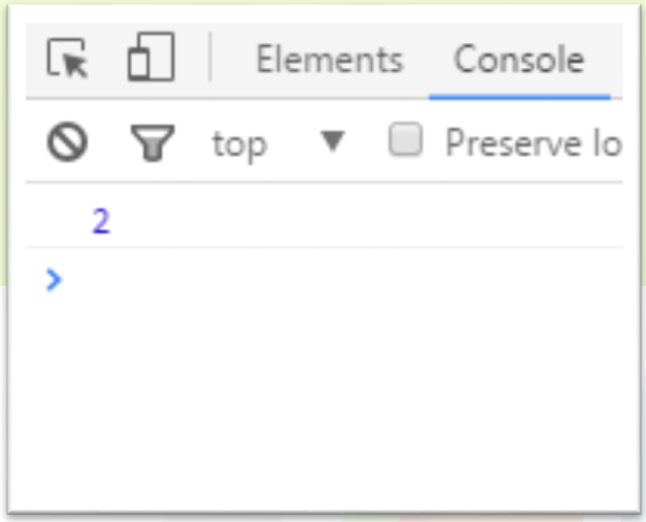
- ศึกษา built-in object ชื่อ Date จาก http://devdocs.io/javascript/ แล้วเรียกฟังก์ชันพร้อมใช้ เพื่อแสดงวันที่ และเวลาปัจจุบันออกทาง Console




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

Math Object

```
<html>
<body>

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

</body>
</html>
```

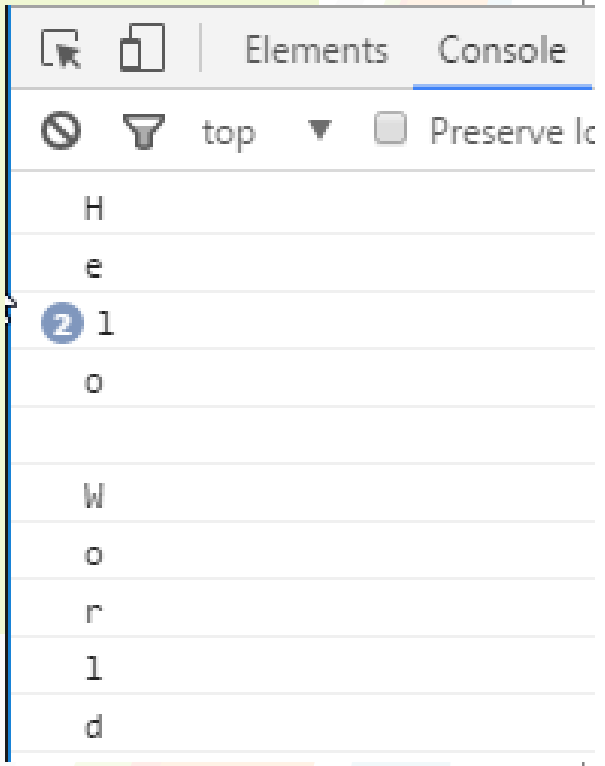


String Object

```
<html>
<body>
<script>
        let str = 'Hello World'

        for (i=0; i<str.length; i++) {
            console.log(str.charAt(i))
        }
</script>
</body>
</html>
```

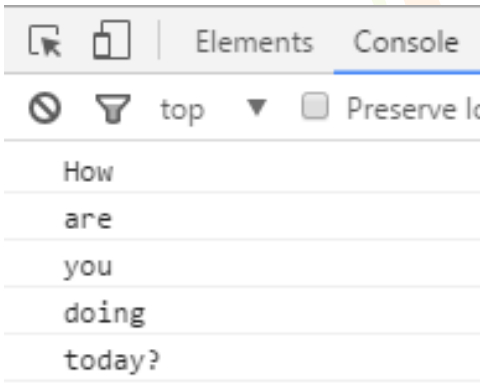
Property length ของ String Object ใช้เป็นเงื่อนไขในลูป

ฟังก์ชัน charAt ของ String Object ใช้ขออักขระแต่ละตัวภายใน String มี argument เป็นตำแหน่งอักขระ ซึ่งเริ่มจากตำแหน่งที่ 0



String Object

```
<html>
<body>

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

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

</body>
</html>
```



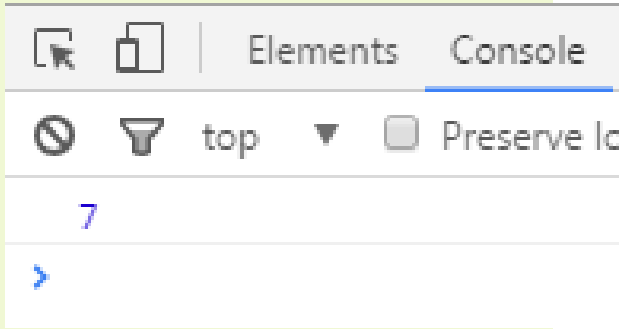
เมธอด split ของ String Object ใช้ในการแยก String ที่คั่นด้วยข้อความที่กำหนด เช่น จากตัวอย่างคือ แยก String ทั้งหมดที่คั่นด้วยช่องว่าง มีการส่งกลับเป็น Array ของ String

String Object

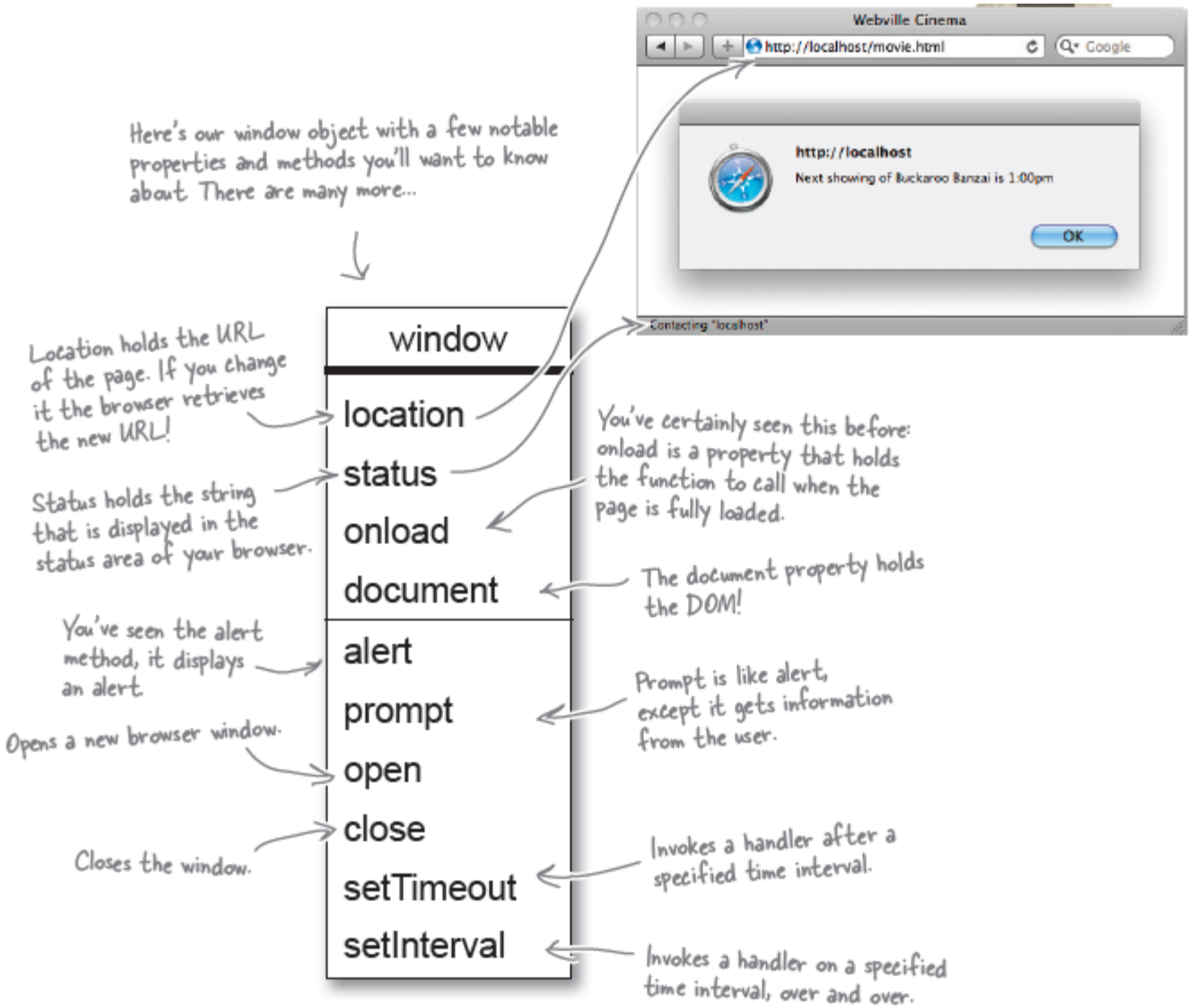
```
<html>
<body>

<script>
    let str = 'เด็กภาคคอม!'
    let n = str.search('คอม')
    console.log(n)
</script>

</body>
</html>
```

JavaScript มองตัวแปรต่าง ๆ เป็น Object ซึ่งสามารถเข้าถึง method หรือ property ได้ทันที

ฟังก์ชัน search ของ String Object ใช้ค้นหา String มี argument เป็น String ที่ต้องการค้นหา และมีการส่งกลับเป็นตำแหน่งอักขระแรกที่เจอ



Window Object



Window Property

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

กำหนดค่าใหม่ให้กับ property location จะเกิดการส่งต่อ (Redirect) ไปยัง URL ใหม่

ฟังก์ชัน open บน Window object

ใช้เมธอด open ใน window object เพื่อเปิดหน้าเว็บบนหน้าต่างใหม่ 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

```
<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>
```

ใช้ฟังก์ชัน open ใน window object เพื่อเปิดหน้าเว็บบนหน้าต่างใหม่ แล้วนำ object ของหน้าต่างนั้นไว้ในตัวแปร winObj

เข้าถึง document object ซึ่งอยู่ภายใต้ window object และเรียกฟังก์ชัน write เพื่อเขียนคำสั่ง HTML ลงบนหน้าต่างใหม่