JavaScript เบื้องต้น

ร่วมกับ HTML5 CSS3 [ฉบับปรับปรุง 2020]

สำหรับใช้งานร่วมกับ HTML CSS

JavaScript คืออะไร

เป็นภาษาคอมพิวเตอร์ที่ใช้ในการพัฒนาเว็บร่วมกับ HTML เพื่อให้เว็บมีลักษณะแบบไดนามิก คือ เว็บสามารถตอบสนองกับผู้ใช้งานหรือแสดงเนื้อหาที่แตกต่างกันไปโดยจะอ้างอิงตามเว็บบราวเซอร์ที่ผู้เข้าชมเว็บใช้งานอยู่

เป็นภาษาที่ทำงานฝั่งผู้ใช้ (Client Side Script)โดยเว็บเบราว์เซอร์จะทำหน้าที่ประมวลผลคำสั่งที่ถูกเขียนขึ้นมาและตอบสนองต่อผู้ใช้ได้ทันที เช่น การแสดงข้อความแจ้งเตือน (Alert) การตรวจสอบข้อมูลที่ผู้ใช้ป้อน (Validation) เป็นต้น

ความสามารถของ JavaScript

- สามารถเปลี่ยนแปลงรูปแบบการแสดงผลของ HTML,CSS ได้
- ตรวจสอบความถูกต้องของข้อมูลได้
- ตรวจสอบ Browser ของผู้ใช้ได้
- เก็บข้อมูลผู้ใช้ได้ เช่น การใช้ Cookie , Local Storage เป็นต้น

รูปแบบการเขียน JavaScript

1.แบบ Internal คือ กำหนด JavaScript ไว้ในส่วนของ <head></head> หรือ <body></body>

```
<script type="text/javascript">
        ..... Statement.....
    </script>
```

```
<script type="text/javascript">
        document.write("Kong Ruksiam");
</script>
```

รูปแบบการเขียน JavaScript

2. แบบ External คือ กำหนด JavaScript ไว้เป็นไฟล์ด้านนอกที่มีนามสกุล .js จากนั้นก็นำเข้ามาทำงานในหน้าเว็บ หรือ HTML ไฟล์

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

```
document.write("KongRuksiam");
document.write("<br>");
document.write("JavaScript เบื้องต้น");
```

การแสดงข้อมูล

- **document.write(“ข้อความที่ต้องการแสดง”)** แสดงเป็นข้อความ ตัวเลข ตัวแปร หรือแท็ก HTML ก็ได้ในหน้าเว็บ
- **alert(“ข้อความแจ้งเตือน”)** สำหรับแจ้งเตือนผู้ใช้ในหน้าเว็บ
- **Console.log(“ข้อความ หรือ ตัวแปร”)** สำหรับ debug ค่าต่างๆ แต่จะไม่แสดงผลในหน้าเว็บ

การเขียนคำอธิบาย (Comment)

วิธีที่ 1 โดยใช้เครื่องหมาย Slash (/) ใช้ในการอธิบาย**คำสั่งสั้นๆในรูปแบบบรรทัดเดียว**

วิธีที่ 2 เขียนคำอธิบายไว้ในเครื่องหมาย /* ... */ ใช้ในการอธิบายคำสั่ง**ยาวๆหรือแบบหลายบรรทัด**

ตัวแปรและชนิดข้อมูล

ตัวแปร คือ **ชื่อที่ถูกนิยามขึ้นมา**เพื่อใช้เก็บค่าข้อมูลสำหรับนำไปใช้งานในโปรแกรม โดยข้อมูลอาจจะประกอบด้วย ข้อความ ตัวเลข ตัวอักษรหรือผลลัพธ์จากการประมวลผลข้อมูล

รูปแบบการตั้งชื่อ

1. ขึ้นต้นด้วยตัวอักษรในภาษาอังกฤษตามด้วยตัวอักษรหรือตัวเลข
2. ห้ามขึ้นต้นด้วยตัวเลขหรือสัญลักษณ์พิเศษ
3. ขึ้นต้นด้วย $ (dollar sign) และ _ (underscore) ได้
4. มีลักษณะเป็น case sensitive คือ ตัวพิมพ์เล็กพิมพ์ใหญ่จะมีความหมายที่แตกต่างกัน
5. ไม่ซ้ำกับคำสงวน (Keyword)

ตัวแปรใน JavaScript เป็นรูปแบบ Dynamic Typing

- **ตัวแปรแบบ Dynamic Typing** คือชนิดตัวแปรจะเป็นอะไรก็ได้ตามค่าที่ตัวมันเก็บโดยไม่ต้องประกาศชนิดข้อมูล

- **ตัวแปรแบบ Static Typing** ต้องประกาศชนิดข้อมูลในตอนเริ่มต้น เช่น int, double, char เพื่อบอกว่าตัวแปรนี้จะเก็บข้อมูลชนิดไหน

การนิยามตัวแปร

var (เปลี่ยนแปลงค่าในตัวแปรได้)

var ชื่อตัวแปร;

var ชื่อตัวแปร = ค่าเริ่มต้น;

var ชื่อตัวแปร = ค่าเริ่มต้น,ชื่อตัวแปร = ค่าเริ่มต้น

```
var money;

var money=100;

money=200;


var a, b, c, d;
var x = 10, y = 20, z = 30;
```

***ตัวแปรที่ประกาศไว้แต่ยังไม่ได้กำหนดค่า จะมีค่าเป็น undefined โดยอัตโนมัติ

การนิยามตัวแปร (2015)

let (เปลี่ยนแปลงค่าในตัวแปรได้)

let ชื่อตัวแปร;

let ชื่อตัวแปร = ค่าเริ่มต้น;

let ชื่อตัวแปร = ค่าเริ่มต้น,ชื่อตัวแปร = ค่าเริ่มต้น

```
let money;
let money=100;
money=200;

let a, b, c, d;
let x = 10, y = 20, z = 30;
```

***ตัวแปรที่ประกาศไว้แต่ยังไม่ได้กำหนดค่า จะมีค่าเป็น undefined โดยอัตโนมัติ

การนิยามตัวแปร (2015)

const (ค่าคงที่)

const ชื่อตัวแปร = ค่าของตัวแปร;

เช่น

```
const money=100;
money=200;// เปลี่ยนแปลงค่าเดิมไม่ได้
```

Data Type	คำอธิบาย	รูปแบบข้อมูล
boolean	ค่าทางตรรกศาสตร์	True /False
number	ตัวเลขที่ไม่มีจุดทศนิยม	20
	ตัวเลขที่มีจุดทศนิยม	30.15
string	ข้อความ	“kongruksiam”
object	ข้อมูลเชิงวัตถุ	{firstName:”kong”, lastName:”ruksiam”, age:20};
array	ชุดข้อมูล	[”มะม่วง”, ”มะละกอ”, ”ส้ม”]

หัวข้อที่เกี่ยวข้องกับตัวแปร

- **typeof** คือ เช็คชนิดข้อมูล
- **null** คือ ไม่มีการกำหนดค่าถูกกำหนดค่าโดยผู้เขียน
- **undefined** ไม่มีการกำหนดค่า (เป็นค่าเริ่มต้นของโปรแกรม)

จัดการตัวเลข (Number)

```
let x ,y ;
x = 20; // integer
y = 20.15; // float
```

จัดการอักขระและข้อความด้วย string

การประกาศ string ขึ้นมาใช้ ต้องกำหนดเนื้อหาหรือค่าอยู่ในเครื่องหมาย ’ (single quote) หรือ ” (double quote)

```
let a = ’kongruksiam’;
let b = ”สอน javascript เบื้องต้น”;
let c = ’basic to advance’;
```

การแปลงชนิดข้อมูล (Type Conversion)

แปลงจาก String เป็น Number

- x = parseInt('1.2');
- x = parseFloat('1.2');
- ใช้เครื่องหมาย (+...) เพิ่มไปข้างหน้า

แปลงจาก Number เป็น String

- ใช้เครื่องหมาย " " + ตัวแปร หรือ ค่าที่เป็นตัวเลข
- ใช้ toString() เช่น x.toString()

อาร์เรย์ (Array) คืออะไร

ความหมายที่ 1 ชุดของตัวแปรที่อยู่ในรูปลำดับใช้เก็บค่าข้อมูลให้อยู่ในกลุ่มเดียวกัน ข้อมูลภายในอาร์เรย์จะถูกเก็บบนหน่วยความจำในตำแหน่งที่ต่อเนื่องกัน โดยขนาดของอาร์เรย์จะเล็กหรือใหญ่ขึ้นกับจำนวนมิติที่กำหนดขึ้น

อาร์เรย์ (Array) คืออะไร

ความหมายที่ 2 เป็นตัวแปรที่ใช้ในการเก็บข้อมูลที่มีลำดับที่ต่อเนื่อง ซึ่งข้อมูลมีค่าได้หลายค่าโดยใช้ชื่ออ้างอิงได้เพียงชื่อเดียว และใช้หมายเลขกำกับ (index) ให้กับตัวแปรเพื่อจำแนกความแตกต่างของค่าตัวแปรแต่ละตัว

การสร้าง Array

วิธีที่ 1 สร้างโดยใช้คำสั่ง Array()

let ชื่ออาร์เรย์ = new Array( );

let ชื่ออาร์เรย์= Array(สมาชิกตัวที่1, สมาชิกตัวที่2, ...);

เช่น

```
let myArray = new Array( );
myArray[0] = 2000;
let days = Array("จันทร์", "อังคาร", "พุธ");
```

การสร้าง Array

วิธีที่ 2 สร้างโดยใช้เครื่องหมาย []

let ชื่ออาร์เรย์ = [สมาชิกตัวที่1, สมาชิกตัวที่2, ...];

เช่น

let color = [“แดง”, “น้ำเงิน”, “เหลือง”];

การเข้าถึงสมาชิก

ชื่ออาร์เรย์[เลขลำดับ]

```
let color = [“แดง”, “น้ำเงิน”, “เหลือง”];
color[0]
color[1]
```

ตัวดำเนินการ (Operator)

กลุ่มของเครื่องหมายหรือสัญลักษณ์ที่ใช้ในการเขียนโปรแกรม

A+B

1. ตัวดำเนินการ (Operator)
2. ตัวถูกดำเนินการ (Operand)

ตัวดำเนินการทางคณิตศาสตร์

Operator	คำอธิบาย
+	บวก
-	ลบ
*	คูณ
/	หาร
%	หารเอาเศษ

ตัวดำเนินการเปรียบเทียบ

**** ชนิดข้อมูล boolean

Operator	คำอธิบาย
==	เท่ากับ
!=	ไม่เท่ากับ
>	มากกว่า
<	น้อยกว่า
>=	มากกว่าเท่ากับ
<=	น้อยกว่าเท่ากับ

ตัวดำเนินการทางตรรกศาสตร์

Operator	คำอธิบาย
&&	AND
||	OR
!	NOT

ตัวดำเนินการทางตรรกศาสตร์

a	!a	a	b	a && b	a || b
true	false	false	false	false	false
false	true	false	true	false	true
		true	false	false	true
		true	true	true	true

ตัวดำเนินการเพิ่มค่า - ลดค่า

Operator	รูปแบบการเขียน	ความหมาย
++ (Prefix)	++a	เพิ่มค่าให้ a ก่อน 1 ค่าแล้วนำไปใช้
++ (Postfix)	a++	นำค่าปัจจุบันใน a ไปใช้ก่อนแล้วค่อยเพิ่มค่า
-- (Prefix)	--b	ลดค่าให้ b ก่อน 1 ค่าแล้วนำไปใช้
-- (Postfix)	b--	นำค่าปัจจุบันใน b ไปใช้ก่อนแล้วค่อยลดค่า

Compound Assignment

Assignment	รูปแบบการเขียน	ความหมาย
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

ลำดับความสำคัญของตัวดำเนินการ

ลำดับที่	เครื่องหมาย	ลำดับการทำงาน
1	( )	
2	++ , --	ซ้ายไปขวา
3	* , / , %	ซ้ายไปขวา
4	+ , -	ซ้ายไปขวา
5	< , <= , > , >=	ซ้ายไปขวา
6	== , !=	ซ้ายไปขวา
7	&&	ซ้ายไปขวา
8	||	ซ้ายไปขวา
9	= , += , -= , *= , /= , %=	ขวาไปซ้าย

กรณีศึกษา

1. 5+8 *9
2. 10 - 4+2
3. 10 - (2+1)
4. 5 * 2 – 40 / 5
5. 7+8/2+25

โครงสร้างควบคุม (Control Structure)

คือ กลุ่มคำสั่งที่ใช้ควบคุมการทำงานของโปรแกรม

- แบบลำดับ (Sequence)
- แบบมีเงื่อนไข (Condition)
- แบบทำซ้ำ (Loop)

แบบมีเงื่อนไข (Condition)

กลุ่มคำสั่งที่ใช้ตัดสินใจในการเลือกเงื่อนไขต่างๆ ภายในโปรแกรมมาทำงาน

- if
- Switch..Case

รูปแบบคำสั่งแบบเงื่อนไขเดียว

- **if statement**

เป็นคำสั่งที่ใช้กำหนดเงื่อนไขในการตัดสินใจทำงานของโปรแกรม

ถ้าเงื่อนไขเป็นจริงจะทำตามคำสั่งต่างๆ ที่กำหนดภายใต้เงื่อนไขนั้นๆ

```
if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง ;
}
```

รูปแบบคำสั่งแบบ 2 เงื่อนไข

```
if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง ;
}else{
    คำสั่งเมื่อเงื่อนไขเป็นเท็จ ;
}
```

ข้อควรระวังการเขียน if เพื่อตรวจสอบเงื่อนไข

```
if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง ;
}if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง ;
}
```

รูปแบบคำสั่งแบบหลายเงื่อนไข

```
if(เงื่อนไขที่ 1){
        คำสั่งเมื่อเงื่อนไขที่ 1 เป็นจริง ;
}else if(เงื่อนไขที่ 2){
        คำสั่งเมื่อเงื่อนไขที่ 2 เป็นจริง ;
}else if(เงื่อนไขที่ 3){
        คำสั่งเมื่อเงื่อนไขที่ 3 เป็นจริง ;
}else{
        คำสั่งเมื่อทุกเงื่อนไขเป็นเท็จ ;
}
```

if..else แบบลดรูป (Ternary Operator)

ตัวแปร = (เงื่อนไข) ? คำสั่งเมื่อเงื่อนไขเป็นจริง : คำสั่งเมื่อเงื่อนไขเป็นเท็จ;

```
if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง
}else{
    คำสั่งเมื่อเงื่อนไขเป็นเท็จ
}
```

การเขียน if ซ้อน if

```
if(เงื่อนไขที่ 1){
    if(เงื่อนไขที่ 2 ){
        คำสั่งเมื่อเงื่อนไขที่ 2 เป็นจริง ;
    }
}
```

แบบมีเงื่อนไข (Condition)

กลุ่มคำสั่งที่ใช้ตัดสินใจในการเลือกเงื่อนไขต่างๆ ภายในโปรแกรมมาทำงาน

- **Switch..Case**

Switch เป็นคำสั่งที่ใช้กำหนดเงื่อนไขคล้ายๆกับ if แต่จะเลือกเพียงหนึ่งทางเลือกออกมาทำงานโดยนำค่าในตัวแปรมากำหนดเป็นทางเลือกผ่านคำสั่ง case

รูปแบบคำสั่ง

```
switch(สิ่งที่ต้องการตรวจสอบ) {

        case ค่าที่ 1 : คำสั่งที่ 1;
                break;
         case ค่าที่ 2 : คำสั่งที่ 2;
                break;
                .....
         case ค่าที่ N : คำสั่งที่ N;
                break;

        default : คำสั่งเมื่อไม่มีค่าที่ตรงกับที่ระบุใน case
}
```

***คำสั่ง break จะทำให้โปรแกรมกระโดดออกไปทำงานนอกคำสั่ง switch ถ้าไม่มีคำสั่ง break โปรแกรมจะทำคำสั่งต่อไปเรื่อยๆ จนจบการทำงาน

รูปแบบคำสั่ง

```
switch(month) {

        case 1 : console.log(“มกราคม”);
            break;
        case 2: console.log(“กุมภาพันธ์”);
        break;
                .....
         case ค่าที่ N : คำสั่งที่ N;
                  break;
        default : console.log(“ไม่พบเดือน”);
}
```

กำหนดให้ตัวแปร month เก็บตัวเลข

Switch..Case VS if Statement

```
switch(month) {
        case 1 : console.log(“มกราคม”);
            break;
        case 2: console.log(“กุมภาพันธ์”);
    break;
            .....
     case ค่าที่ N : คำสั่งที่ N;
            break;
        default : console.log(“ไม่พบเดือน”);
}
```

```
if(month==1){
        console.log(“มกราคม”);
}elseif(month==2){
        console.log(“กุมภาพันธ์”);
}elseif(เงื่อนไขที่ 3){
        คำสั่งเมื่อเงื่อนไขที่ 3 เป็นจริง ;
}else{
        System.out.println(“ไม่พบเดือน”);
}
```

แบบทำซ้ำ (Loop)

กลุ่มคำสั่งที่ใช้ในการวนรอบ (loop) โปรแกรมจะทำงานไปเรื่อยๆจนกว่าเงื่อนไขที่กำหนดไว้จะเป็นเท็จ จึงจะหยุดทำงาน

- While
- For
- Do..While

คำสั่งที่เกี่ยวข้องกับ Loop

- **break** ถ้าโปรแกรมพบคำสั่งนี้จะหลุดจากการทำงานในลูปทันที เพื่อไปทำคำสั่งอื่นที่อยู่นอกลูป

- **continue** คำสั่งนี้จะทำให้หยุดการทำงานแล้วย้อนกลับไปเริ่มต้นการทำงานที่ต้นลูปใหม่

คำสั่ง While

- **While Loop**

จะทำงานตามคำสั่งภายใน while ไปเรื่อยๆเมื่อเงื่อนไขที่กำหนดเป็นจริง

```
while(เงื่อนไข){
    คำสั่งที่จะทำซ้ำเมื่อเงื่อนไขเป็นจริง ;
}
```

คำสั่ง For

- **For Loop**

เป็นรูปแบบที่ใช้ในการตรวจสอบเงื่อนไข มีการกำหนดค่าเริ่มต้น และเปลี่ยนค่าไปพร้อมๆกัน เมื่อเงื่อนไขในคำสั่ง for เป็นจริงก็จะทำงานตามคำสั่งที่แสดงไว้ภายในคำสั่ง for ไปเรื่อยๆ

โครงสร้างคำสั่ง

```
for(ค่าเริ่มต้นของตัวแปร; เงื่อนไข; เปลี่ยนแปลงค่าตัวแปร) {
        คำสั่งเมื่อเงื่อนไขเป็นจริง;
}
```

```
for(let i = 1;i<=10;i++) {
    คำสั่งเมื่อเงื่อนไขเป็นจริง;
}
```

คำสั่ง Do..While

- **Do..While**

โปรแกรมจะทำงานตามคำสั่งอย่างน้อย 1 รอบ เมื่อทำงานเสร็จจะมาตรวจสอบเงื่อนไขที่คำสั่ง while ถ้าเงื่อนไขเป็นจริงจะวนกลับขึ้นไปทำงานที่คำสั่งใหม่อีกรอบ แต่ถ้าเป็นเท็จจะหลุดออกจากลูป

โครงสร้างคำสั่ง

```
do {
        คำสั่งต่างๆ เมื่อเงื่อนไขเป็นจริง;
 } while(เงื่อนไข);
```

ข้อแตกต่างและการใช้งาน Loop

- For ใช้ในกรณีรู้จำนวนรอบที่ชัดเจน
- While ใช้ในกรณีที่ไม่รู้จำนวนรอบ
- Do..while ใช้ในกรณีที่อยากให้ลองทำก่อน 1 รอบ แล้วทำซ้ำไปเรื่อยๆทราบเท่าที่เงื่อนไขเป็นจริง

ค่า null, undefined และ NaN

null คือตัวแปรที่ไม่มีค่าใดๆ เลย ไม่เท่ากับ 0 และไม่เท่ากับสตริงว่าง ไม่สามารถนำไปคำนวณใดๆ ได้ แต่หากนำไปเปรียบเทียบด้วยเงื่อนไขจะมีค่าเท่ากับค่า false

```
let a = null;

if(!a) {
alert(“a is null”);
} else {
  alert(“a is not null”);
}
```

ค่า null, undefined และ NaN

undefined คือ ตัวแปรที่ประกาศเอาไว้แต่ไม่ได้กำหนดค่าใดๆ ให้กับมัน ยกตัวอย่าง เช่น

```
let a;

alert(a);
```

ค่า null, undefined และ NaN

NaN (มาจาก Not a Number) หมายถึงการนำตัวแปรที่ไม่ใช่ตัวเลขไปคำนวณทางคณิตศาสตร์

```
let a = 10;
let b = “x”;
alert(10-b);
```

ฟังก์ชั่น คืออะไร

ความหมายที่ 1:

ชุดคำสั่งที่นำมาเขียนรวมกันเป็นกลุ่มเพื่อให้เรียกใช้งานตามวัตถุประสงค์ที่ต้องการ และลดความซ้ำซ้อนของคำสั่งที่ใช้งานบ่อยๆ ฟังก์ชั่นสามารถนำไปใช้งานได้ทุกที่และแก้ไขได้ในภายหลัง ทำให้โค้ดในโปรแกรมมีระเบียบและใช้งานได้สะดวกมากยิ่งขึ้น

ความหมายที่ 2 :

โปรแกรมย่อยที่นำเข้ามาเป็นส่วนหนึ่งของโปรแกรมหลัก เพื่อให้สามารถเรียกใช้งานได้โดยไม่จำเป็นต้องเขียนโค้ดคำสั่งใหม่ทั้งหมด

รูปแบบของฟังก์ชั่น

1.ฟังก์ชั่นที่ไม่มีการรับและส่งค่า

```
function ชื่อฟังก์ชั่น(){
    // คำสั่งต่างๆ
}
```

การเรียกใช้งานฟังก์ชั่น

```
ชื่อฟังก์ชั่น ();
```

รูปแบบของฟังก์ชั่น

2.ฟังก์ชั่นที่มีการรับค่าเข้ามาทำงาน

```
function ชื่อฟังก์ชั่น(parameter1,parameter2,.....){

// กลุ่มคำสั่งต่างๆ

}
```

อาร์กิวเมนต์ คือ ตัวแปรหรือค่าที่ต้องการส่งมาให้กับฟังก์ชัน (ตัวแปรส่ง)
พารามิเตอร์ คือ ตัวแปรที่ฟังก์ชันสร้างไว้สำหรับรับค่าที่จะส่งเข้ามาให้กับฟังก์ชัน (ตัวแปรรับ)

การเรียกใช้งานฟังก์ชั่น

```
ชื่อฟังก์ชั่น (argument1,argument2,.....);
```

รูปแบบของฟังก์ชั่น

3.ฟังก์ชั่นที่มีส่งค่าออกมา

```
function ชื่อฟังก์ชั่น(){
    return ค่าที่จะส่งออกไป
}
```

รูปแบบของฟังก์ชั่น

4.ฟังก์ชั่นที่มีการรับค่าเข้ามาและส่งค่าออกไป

```
function ชื่อฟังก์ชั่น(parameter1,parameter2,.....){
        retrun ค่าที่จะส่งออกไป
}
```

ฟังก์ชั่นแบบกำหนดค่าเริ่มต้น

```
function ชื่อฟังก์ชั่น (name=”kongruksiam”,parameter2,.....){
    // คำสั่งต่างๆ
}
```

ขอบเขตตัวแปร

- **local variable** ตัวแปรที่ทำงานอยู่ในฟังก์ชั่นมีขอบเขตการทำงานตั้งแต่จุดเริ่มต้นไปจนถึงจุดสิ้นสุดของฟังก์ชั่น

- **global variable** ตัวแปรที่ทำงานอยู่นอกฟังก์ชั่นมีขอบเขตการทำงานตั้งแต่จุดเริ่มต้นไปจนถึงจุดสิ้นสุดของไฟล์ที่ประกาศใช้

Array Properties & Function

หาจำนวนสมาชิกและเรียงลำดับ

```
let color = [“แดง”, “น้ำเงิน”, “เหลือง”];
let x = color.length;
let y = color.sort();
```

สมาชิกตัวแรกและตัวสุดท้าย

```
let first = color[0];
let last = color[color.length-1];
```

การเพิ่มสมาชิก

```
color.push(”สีเทา”);
```

เข้าถึงสมาชิกด้วย For Loop

```
let color = [“แดง”, “น้ำเงิน”, “เหลือง”];
let count = color.length;


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

เข้าถึงสมาชิกด้วย ForEach

```
let color = [“แดง”, “น้ำเงิน”, “เหลือง”];
color.forEach(myData);


function myData(item) {
        console.log(item);
}
```

แปลง Array เป็น String

- .toString() //แปลงเป็น String
- .join(" * "); // นำค่าแต่ละค่าในตัวแปร array มารวมกันเป็นข้อความ และส่งค่ากลับเป็นข้อความที่มีตัวคั่นค่าตัวแปรแต่ละค่าตามที่กำหนด
- color.pop(); // เอาตัวสุดท้ายออก
- let x = color.pop(); //เอาตัวท้ายออกแล้วเก็บในตัวแปร x

การรวม Array

```
let fruits = ["ส้ม", "องุ่น"];
let vegetables = ["คะน้า", "ผักชี", "ผักกาด"];
    let hardware = ["เม้าส์","คีย์บอร์ด"];

let carts = fruits.concat(vegetables,computer);
```

เรียงลำดับใน Array

- let fruits = ["ส้ม", "องุ่น"];
- fruits.sort();
- fruits.reverse();

เรียงลำดับใน Array แบบตัวเลข (น้อยไปมาก)

```
let points = [20, 100, -100, 5, -25, 10];
points.sort(function(a, b){
    return a - b
});
```

a คือ ค่าตัวเลขที่มีค่าลบจะถูกเรียงก่อน

b คือ ค่าตัวเลขที่มีค่าบวกจะถูกเรียงทีหลัง

เรียงลำดับใน Array แบบตัวเลข (มากไปน้อย)

```
let points = [20, 100, -100, 5, -25, 10];
points.sort(function(a, b){
    return b - a
});
```

b คือ ค่าตัวเลขที่มีค่าบวกจะถูกเรียงก่อน

a คือ ค่าตัวเลขที่มีค่าลบจะถูกเรียงทีหลัง

JavaScript Object

let ชื่อวัตถุ = {*propertyName*:value}
ยกตัวอย่าง เช่น

```
let user = {
 name:"kong", age:20, email:"kong@gmail.com"
};
let product ={name:"มะม่วง",price:150,category:"ผลไม้"}
```

JavaScript Object

การเข้าถึงข้อมูล

objectName.propertyName

objectName[“propertyName”]

ยกตัวอย่าง เช่น

user.name

user[“name”]

JavaScript Object (Method)

```
let user = {
        name:"kong",
        age:20,
        email:"kong@gmail.com",
        getUser:function(){
                return this.name + " " + this.email;
         }
    };
 };
```

การเรียกใช้งาน

objectName.methodName();

```
let data = user.getUser();
```

ความแตกต่างของ Array และ Object

- Array มี Index เป็นตัวเลข , Object กำหนดเป็นชื่อ
- Array ใช้ [] , ส่วน Object ใช้ {}

การยืนยันด้วย confirm( )

เป็นหน้าต่างที่ต้องการสอบถามการยืนยันจากผู้ใช้ ก่อนที่จะทำการใดๆ ต่อไป

confirm(“ข้อความ”);

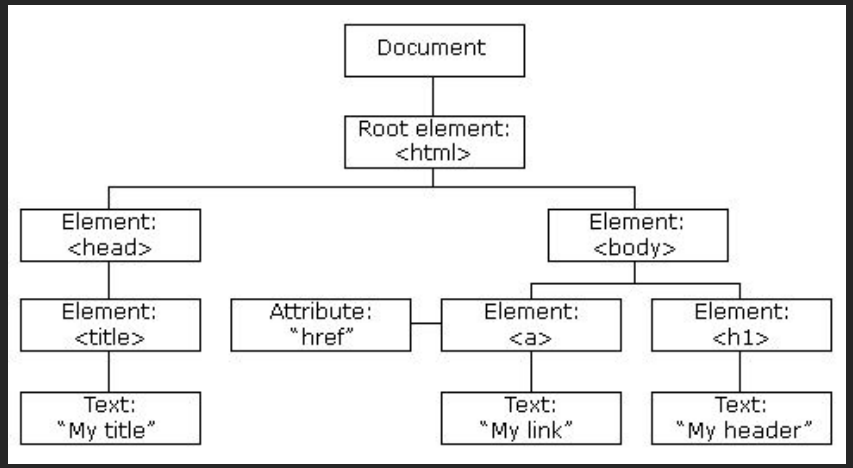
โดยผลลัพธ์จะมีค่าทางตรรกศาสตร์

- มีค่าเป็น true เมื่อผู้ใช้คลิก Ok
- มีค่าเป็น false เมื่อผู้ใช้คลิก cancel

HTML DOM (Document Object Model)

เมื่อหน้าเว็บโหลดเสร็จเรียบร้อย Web Browser มันจะสร้าง DOM ของหน้านั้นขึ้นมา โดยมอง HTML เป็นโครงสร้างต้นไม้ (ก้อน Object) หรือ**เรียกว่า DOM**

```
<html>
<head>
        <title>My title</title>
</head>
 <body>
<a href="#">My link</a>
<h1>My header</h1>
</body>
</html>
```



Tag ต่าง ๆ ใน HTML จะเรียกว่า Element

คุณสมบัติของ HTML DOM

- เข้าถึงและเปลี่ยนคุณสมบัติทั้งหมดของ Element ในหน้าเว็บได้
- ควบคุมและเปลี่ยนรูปแบบ CSS ได้
- สามารถตอบสนองกับทุกเหตุการณ์ที่เกิดขึ้นหน้าเว็บได้

เข้าถึง Element ผ่าน Id , Tag , Class

- document.getElementById ("ชื่อไอดี");
- document.getElementsByTagName ("ชื่อแท็ก");
- document.getElementsByClassName ("ชื่อคลาส");

DOM Document

- เปลี่ยนเนื้อหา HTML : element.innerHTML
- เปลี่ยนเนื้อหา Text : element.innerText
- เปลี่ยน style Element : element.style.properties = value

ดำเนินการผ่าน Method

- *element*.setAttribute(*attribute, value*)

DOM Nodes

- document.createElement(*element*) // สร้าง element ใหม่
- document.removeChild(*element*) // ลบ node ลูก
- document.appendChild(*element*) // นำ element ไปต่อใน node แม่
- document.replaceChild(*new*, *old*) แทนที่ element

DOM CSS Add & Remove Class

- element.classList.add(“class”); **// เพิ่ม class style**
- element.classList.remove(“class”); **// ลบ class style**
- element.classList.toggle(“class”); **// สลับ class style**
- element.classList.contains(“class”);**// เปรียบเทียบ class style**

DOM Event

คือ เหตุการณ์หรือการกระทำบางอย่างที่เกิดขึ้นกับอิลิเมนต์ เช่น การคลิกเมาส์ การเคลื่อนย้ายเมาส์ การกดปุ่มคีย์บอร์ด เป็นต้น

โดยผู้พัฒนาสามารถใช้อีเวนต์ที่เกิดขึ้นเป็นตัวกำหนดให้ตอบสนอง หรือกระทำบางอย่างได้ เช่น การคลิกแล้วแจ้งเตือน เป็นต้น

ชื่อ Event	ความหมาย	ทำงานร่วมกับแท็ก
onfocus=“ ”	เมื่ออิลิเมนต์นั้นได้รับการโฟกัส	select, text, textarea
onblur=“ ”	เมื่ออิลิเมนต์นั้นสูญเสียการโฟกัส หรือถูกย้ายโฟกัสไปยังอิลิเมนต์อื่น	select, text, textarea
onchange=“ ”	เมื่อผู้ใช้เปลี่ยนแปลงค่าในฟอร์มรับข้อมูล	select, text, textarea
onselect=“ ”	เมื่อผู้ใช้เลือกข้อความ (ใช้เมาส์ลาก) ในช่องข้อความ	text, textarea
onsubmit=“ ”	เมื่อผู้ใช้คลิกปุ่ม submit	form

ชื่อ Event	ความหมาย	ทำงานร่วมกับแท็ก
onMouseover=“ ”	เกิดเมื่อออบเจกต์นั้นถูกเลื่อน mouse pointer ไปทับ	a,div
onMouseout=“ ”	เกิดเมื่อออบเจกต์นั้นถูกเลื่อน mouse pointer ที่ทับอยู่ออกไป	a,div
onclick=“ ”	เกิดเมื่อออบเจกต์นั้นถูกคลิก	a, button, checkbox, radio, reset, submit
onload=“ ”	เกิดเมื่อโหลดเอกสารเสร็จ	body
onunload=“ ”	เกิดเมื่อยกเลิกการโหลด เช่น คลิกปุ่ม Stop	body

EventListener

คือ เหตุการณ์หรือการกระทำบางอย่างที่เกิดขึ้นกับอิลิเมนต์ แต่รูปแบบการเขียนจะเขียนในฝั่ง javascript ทั้งหมด

โครงสร้าง :

```
element.addEventListener(event,callback)
```