ทฤษฎี บทที่ 4

4.1 หลักการเบื้องต้นเกี่ยวกับภาษา Script บน Client

4.1.1 หลักการรันภาษา Script ใน HTML

ภาษา JavaScript เป็นภาษาสำหรับออกแบบและเขียนโปรแกรมบนระบบอินเทอร์เน็ต มีความสามารถในด้านงานคำนวณ, การเปรียบเทียบ, ข้อความ, รูปภาพ, เสียง, วันที่, เวลา และยังสนับสนุนคุณสมบัติพื้นฐานของภาษา Java ได้เป็นอย่างดี

คำสั่ง <SCRIPT>…..</SCRIPT>

เป็นการกำหนดขอบเขตของกลุ่มคำสั่งของภาษา JavaScript มีรูปแบบดังนี้

```
<SCRIPT>
      กลุ่มคำสั่งของ JavaScript
      .....
</SCRIPT>
```

การกำหนดตำแหน่ง <SCRIPT>…</SCRIPT> ในเอกสาร HTML

จะมี 2 รูปแบบ คือ

- กำหนดในส่วนคำสั่ง <BODY>….</BODY>

```
<HTML>
<HEAD>
<TITLE> ทดสอบ 1 </TITLE>
</HEAD>
<BODY>
      ข้อความเอกสาร HTML
<SCRIPT>
      กลุ่มคำสั่งของ JavaScript
      .....
</SCRIPT>
</BODY>
</HTML>
```

- กำหนดในส่วนคำสั่ง <HEAD>….</HEAD

```
<HTML>
<HEAD>
<TITLE> ทดสอบ 2 </TITLE>
  <SCRIPT>
      กลุ่มคำสั่งของ JavaScript
      .....
  </SCRIPT>
</HEAD>
<BODY>
      ข้อความเอกสาร HTML
</BODY>
</HTML>
```

<u>การกำหนดหมายเหตุ</u>

สำหรับการกำหนดหมายเหตุมีประโยชน์ทั้งในเอกสาร HTML และคำสั่งภายใน <script> ในกรณีที่บราวเซอร์มรุ่นเก่า ๆ ไม่สนับสนุนการทำงานของคำสั่ง <script> แบ่งออกเป็น 2 ชนิดด้วยกันดังต่อไปนี้

1. หมายเหตุของเอกสาร HTML เนื่องจากบราวเซอร์บางตัวไม่สนับสนุน JavaScript จึงมองไม่เห็นชุดคำสั่งดังกล่าวภายใต้หมายเหตุของ HTML

```
<!- -  ข้อความหมายเหตุ  - ->
```

2. หมายเหตุของ JavaScript แบ่งเป็น
   - หมายเหตุบรรทัดเดียว // **ข้อความหมายเหตุ**
   - หมายเหตุไม่จำกัดบรรทัด **/* ข้อความหมายเหตุ */**

<u>คำสั่งการพิมพ์เบื้องต้นของ JavaScript</u>

ใช้สำหรับพิมพ์ข้อความใด ๆ และการพิมพ์คำสั่งของเอกสาร HTML

รูปแบบ

```
document.write("ข้อมูล");
```

โดยกำหนดให้ ข้อมูล หมายถึง ข้อความใด ๆ หรือคำสั่งใด ๆ ของ JavaScript ท้ายคำสั่งต้องปิดด้วยเครื่องหมายเซมิโคลอน(;)

<u>การนำคำสั่งของภาษา HTML เข้ามาร่วมกับงานพิมพ์ใน JavaScript</u>

ใช้กำหนดข้อคำสั่งที่ต้องการอยู่ภายใต้เครื่องหมายคำพูด (“คำสั่ง”) หรือถ้าต้องการเชื่อมคำสั่งให้ใช้เครื่องหมาย + คั่นระหว่างคำสั่งภายใต้คำสั่งการพิมพ์ document.write(“….”);

ตัวอย่าง

```
document.write("<H4> JavaScript is not Java </H4>");     หรือ
document.write("<H2>"+"JavaScript is a scripting language"+"</H2>");
```

ในกรณีที่คำสั่งหรือข้อมูลที่อยู่ภายใต้คำสั่ง document.write(“…”) มีเครื่องหมายคำพูด (“…”) ซ้อนกันอยู่ภายใน เช่น คำสั่งกำหนดแบบอักษร <font face = “CordiaNew”> ให้แก้ไขโดยเปลี่ยนเครื่องหมาย (“) ให้เป็น (‘) ที่คำสั่งคู่ใดก็ได้ดังนี้

```
document.write(" <font face = 'CordiaNew'> ");     หรือ
document.write(' <font face = "CordiaNew"> ');
```

4.1.2 คำสั่งนิพจน์

ตำดำเนินการยูนารีนีเกชัน ใช้แปลงค่าให้ตรงกันข้ามกับค่าเดิมโดยใช้เครื่องหมายลบ (-) กระทำต่อตัวแปรที่อยู่ทางซ้ายและค่าของข้อมูลที่อยู่ทางขวา

ตัวอย่าง

X = -5; ดังนั้น unary negation ของ –x จะเท่ากับ 5 คือ -x = 5;

4.1.3 ตัวแปรและชนิดของข้อมูล

ตัวแปรใน JavaScript

1. ขึ้นต้นด้วยตัวอักษรในภาษาอังกฤษ ตามด้วยตัวอักษรหรือตัวเลขใด ๆ ก็ได้
2. ห้ามเว้นช่องว่าง
3. ห้ามใช้สัญลักษณ์พิเศษ ยกเว้นขีดล่าง (_) และ ดอลล่าร์ ($)
4. สำหรับความยาว ยามเท่าใดก็ได้ แต่ที่นิยมใช้ไม่เกิน 20 ตัวอักษร
5. การตั้งชื่อต้องไม่ซ้ำกับคำสงวน (Reserve Word)
6. ตัวอักษรต่างกันระหว่างพิมพ์เล็กและพิมพ์ใหญ่ (Case Sensitive) อย่างเช่น Bird ต่างกันกับ BIRD และต่างกันกับ BirD
7. ควรตั้งชื่อโดยให้ชื่อนั้นสื่อความหมายให้เข้ากับข้อมูล อ่านและเข้าใจได้ง่าย

คำสงวน (Reserve Word)

เป็นคำที่มีความหมายเฉพาะตัวในภาษา JavaScript สงวนไม่ให้มีการตั้งชื่อซ้ำกับชื่อโปรแกรม, ฟังก์ชั่น, ตัวแปร, ค่าคงที่ และคำสั่ง ได้แก่คำต่อไปนี้

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

ชนิดข้อมูลของตัวแปร (Data Type)

เป็นการกำหนดประเภทค่าข้อมูลของตัวแปรเพื่อให้เหมาะสมกับการอ้างอิงข้อมูลจากตัวแปร มี 4 ชนิดคือ

- Number ข้อมูลชนิดตัวเลข ประกอบด้วย จำนวนเต็ม (Integer) และ จำนวนจริง (Floating)
- Logical ข้อมูลทางตรรกะ มี 2 สถานะ คือ True และ False
- String ข้อมูลที่เป็นข้อความ ต้องกำหนดภายในเครื่องหมายคำพูด (“…”)
- Null ไม่มีค่าข้อมูลใด ๆ ค่า null ใช้สำหรับยกเลิกพื้นที่เก็บค่าของตัวแปรออกจากหน่วยความจำ

การประกาศค่าตัวแปร (Declaration)

เป็นการกำหนดชื่อและชนิดข้อมูลให้กับตัวแปร เพื่อนำไปใช้ในโปรแกรม โดยการตั้งชื่อจะต้องคำนึงถึงค่าของข้อมูลที่อ้างอิงและควรตั้งให้สื่อความหมายและสอดคล้องกับข้อมูล และข้อควรระวังคือ ชื่อที่ประกาศนี้ให้กำหนดพอดีกับการใช้งาน และอักษรของชื่อจะจำแนกแตกต่างระหว่าง อักษรตัวพิมพ์เล็กกับอักษรตัวพิมพ์ใหญ่

รูปแบบ

```
var ชื่อตัวแปร ;                    รูปแบบประกาศค่าตัวแปรปกติ หรือ
var ชื่อตัวแปร = ข้อมูล ;           รูปแบบการกำหนดค่าเริ่มต้น หรือ
var ชื่อตัวแปร = ข้อมูล1, ข้อมูล2, ข้อมูล3 ;
```

การกำหนดค่าให้กับตัวแปร

จะใช้เครื่องหมาย = เป็นตัวกำหนดค่า มีรูปแบบ ดังนี้

```
ชื่อตัวแปร = ค่าของข้อมูล ;                หรือ
var ชื่อตัวแปร = ค่าของข้อมูล ;
```

ค่าของข้อมูลได้แก่

1. ข้อมูลที่เป็นตัวเลข โดยกำหนดตัวเลขลงไปได้เลย เช่น year = 19999 ;
2. ข้อมูลในทางตรรกะ ได้แก่ True หรือ False เช่น test = True;
3. ข้อมูลสตริง ให้กำหนดอยู่ในเครื่องหมายคำพูด (“…”) เช่น Book_code = “ST203”;

รหัสคำสั่งพิเศษ (Character escape code)

เป็นการกำหนดรหัสเพื่อควบคุมงานพิมพ์สตริง โดยใช้ \ (backslash) นำหน้าตัวอักษรที่ต้องการกำหนดเป็นรหัสเพื่อให้กลายเป็นคำสั่งพิเศษ มีรหัสดังนี้

\b	ลบไปทางซ้าย (Back Space)
\f	เลื่อนกระดาษไปอีก 1 หน้า
\n	ขึ้นบรรทัดใหม่ (New Line)
\r	ตรวจสอบการกด Enter และการเลื่อนขึ้นบรรทัดใหม่ของหน้ากระดาษ
\t	เลื่อนตำแหน่งไปทางขวา 1 ช่วงแท็บ (Tab)

นอกจากนี้เรายังสามารถใช้เครื่องหมาย \ (Backslash) นำหน้าตัวอักษรพิเศษที่ต้องการพิมพ์ ดังนี้

```
\ ตามด้วยเครื่องหมายพิเศษที่ต้องการ
```

4.1.4 โอเปอร์เรเตอร์

ตัวดำเนินการคณิตศาสตร์ (Arithmetic operator)

+	เครื่องหมายบวก
-	เครื่องหมายลบ
*	เครื่องหมายคูณ
/	เครื่องหมายหาร
%	เครื่องหมายหาเศษที่ได้จากการหาร (mod)
++	เครื่องหมายเพิ่มค่า (increment)
- -	เครื่องหมายลดค่า (decrement)

การเพิ่มค่า จะเขียนได้ 2 แบบ คือ

1. ใช้อินครีเมนต์วางไว้ข้างหน้าตัวแปรที่ถูกกระทำ เป็นการเพิ่มค่าให้กับตัวแปรที่ถูกกระทำก่อนกำหนดค่าให้กับตัวแปรที่อยู่ทางซ้ายมือ

   ตัวแปร = ++ตัวแปรที่ถูกกระทำ

2. ใช้อินครีเมนต์วางไว้ข้างหลังตัวแปรที่ถูกกระทำ เป็นการกำหนดค่าให้กับตัวแปรที่อยู่ทางซ้ายมือ ก่อนการเพิ่มค่าให้กับตัวแปรที่ถูกกระทำ

   ตัวแปร = ตัวแปรที่ถูกกระทำ++

ตัวอย่างเช่น

กำหนดให้ x = 5 ;
ถ้ากำหนดให้ y = ++x; จะมีผลให้ x = 6 ก่อน แล้วจึงให้ y = 6
ถ้ากำหนดให้ y = x++; จะมีผลให้ y = 5 ก่อน แล้วจึงให้ x = 6

การลดค่า จะเขียนได้ 2 แบบ คือ

1. ใช้ดีครีเมนต์วางไว้ข้างหน้าตัวแปรที่ถูกกระทำ เป็นการลดค่าให้กับตัวแปรที่ถูกกระทำก่อนกำหนดค่าให้กับตัวแปรที่อยู่ทางซ้ายมือ

   ตัวแปร = - -ตัวแปรที่ถูกกระทำ

2. ใช้ดีครีเมนต์วางไว้ข้างหลังตัวแปรที่ถูกกระทำ เป็นการกำหนดค่าให้กับตัวแปรที่อยู่ทางซ้ายมือ ก่อนการลดค่าให้กับตัวแปรที่ถูกกระทำ

   ตัวแปร = ตัวแปรที่ถูกกระทำ- -

ตัวอย่างเช่น

กำหนดให้ x = 5 ;
ถ้ากำหนดให้ y = - -x; จะมีผลให้ x = 4 ก่อน แล้วจึงให้ y = 4
ถ้ากำหนดให้ y = x- -; จะมีผลให้ y = 5 ก่อน แล้วจึงให้ x = 4

ตัวดำเนินการเชิงเปรียบเทียบ (Comparison operator)

เครื่องหมายในการเปรียบเทียบข้อมูล ผลลัพธ์ที่ได้จะมีค่าตรรกบูลีนเป็น True และ False ได้แก่

= =	เครื่องหมายเท่ากับ
+=	เครื่องหมายไม่เท่ากับ
>	เครื่องหมายมากกว่า
>=	เครื่องหมายมากกว่าหรือเท่ากับ
<	เครื่องหมายน้อยกว่า
<=	เครื่องหมายน้อยกว่าหรือเท่ากับ

ตัวดำเนินการกำหนดค่า (Assignment operator)

เครื่องหมายในการกำหนดค่าของข้อมูลทางขวาให้กับตัวแปรที่อยู่ทางซ้าย ได้แก่

=	เป็นการกำหนดค่าของข้อมูลทางขวาให้กับตัวแปรทางซ้ายมือ เช่นกำหนดค่า 20 ให้กับตัวแปร x xX = 20;
+=	เพิ่มค่าของข้อมูลทางขวามือให้กับตัวแปรทางซ้ายมือ เช่น กำหนดค่า 20 ให้กับตัวแปร x แล้วเพิ่มค่า 5 ให้กับตัวแปร x x = 20; x +=5; หมายถึง x = x + 5; ผลลัพธ์ที่ได้เป็น 25
-=	ลบค่าของข้อมูลทางขวามือให้กับตัวแปรทางซ้ายมือ เช่นกำหนดค่า 20 ให้กับตัวแปร x แล้วลบค่า 5 ออกจากตัวแปรx x = 20; x -= 5; หมายถึง x = x – 5; ผลลัพธ์ที่ได้เป็น 15
*=	คูณค่าของข้อมูลทางขวามือให้กับตัวแปรทางซ้ายมือ เช่น กำหนดค่า 20 ให้กับตัวแปร x แล้วคูณด้วย 2 ให้กับตัวแปร x x = 20; x *= 2; หมายถึง x = x * 2; ผลลัพธ์ที่ได้เป็น 40
/=	หารด้วยค่าของข้อมูลทางขวามือให้กับตัวแปรทางซ้ายมือเช่นกำหนดค่า 20 ให้กับตัวแปร x แล้วนำค่า 4 ไปการตัวแปร x x = 20; x /= 4; หมายถึง x = x /4 ; ผลลัพธ์ที่ได้เป็น 5
%=	หารด้วยค่าของข้อมูลทางขวามือให้กับตัวแปรทางซ้ายมือ แล้วเก็บเศษที่ได้จากการหาร เช่น x = 20; x %= 3; หมายถึง x = x % 3; ผลลัพธ์ที่ได้เป็น 2
&=	เก็บค่า AND ในระดับบิตให้กับตัวแปรทางซ้ายมือ เช่น เก็บค่า x AND y ให้กับตัวแปร x x &= y; หมายถึง x = x & y;
|=	เก็บค่า OR ในระดับบิตให้กับตัวแปรทางซ้ายมือ เช่น เก็บค่า x OR y ให้กับตัวแปร x x |= y; หมายถึง x = x | y;
^=	เก็บค่า XOR ในระดับบิตให้กับตัวแปรทางซ้ายมือ เช่น เก็บค่า x XOR y ให้กับตัวแปร x x ^= y; หมายถึง x = x ^ y;
<<=	เลื่อนบิตในตัวแปรทางซ้าย ไปทางซ้ายตามจำนวนบิตที่กำหนด เช่น เลื่อนบิตของตัวแปร x ไปทางซ้าย 2 บิต x <<= 2; หมายถึง x = x << 2
>>=	เลื่อนบิตในตัวแปรทางซ้าย ไปทางขวาตามจำนวนบิตที่กำหนด เช่น เลื่อนบิตของตัวแปร x ไปทางขวา 2 บิต x >>= 2; หมายถึง x = x >> 2
>>>=	เลื่อนบิตในแบบซีโรฟิลล์ในตัวแปรทางซ้าย ไปทางขวาตามจำนวนบิตที่กำหนด เช่น เลื่อนบิตซีโรฟิลล์ของตัวแปร x ไป

	ทางขวา 2 บิต x >>>= 2; หมายถึง x = x >>> 2

ตัวดำเนินการเชิงตรรกะ (Logical operator)

เป็นเครื่องหมายที่ให้ค่าจริง (True) และเท็จ (False) ในการเปรียบเทียบ ประกอบด้วยเครื่องหมาย

&&	และ(AND) จะเป็นจริงเมื่อค่าที่ใช้เปรียบเทียบทั้ง 2 ค่า เป็นจริงทั้งคู่
||	หรือ(OR) จะเป็นจริงเมื่อค่าที่ใช้เปรียบเทียบทั้ง 2 ค่าเป็นจริงทั้งคู่ หรือ จริงเพียงค่าใดค่าหนึ่ง
!	ปฏิเสธ(NOT) เป็นการแปลงค่าตรงกันข้าม จากจริงเป็นเท็จ และ จากเท็จเป็นจริง

ตัวดำเนินการระดับบิต (Bitwise operator)

เป็นการดำเนินการเชิงตรรกะในระดับบิตโดยจะใช้มุมมองในแบบเลขฐาน 2 มาจัดการกับข้อมูลนั่นคือ ข้อมูลตัวเลขนั้นจะถูกแปลงเป็นเลขฐานสองในหน่วยความจำในขณะที่มีการดำเนินการเชิงตรรกะในระดับบิต ซึ่งโดยปกติแล้วการกระทำใน JavaScript จะอยู่ในระดับตัวอักษร ที่เรียกว่า ระดับไบต์ (Byte) โดยที่ 1 ไบต์ จะมีค่าเท่ากับ 8 บิต ในระบบการเก็บข้อมูลในคอมพิวเตอร์ ตัวดำเนินการระดับบิตมีรายละเอียดดังนี้

x & y	การเทียบบิตแบบ AND ระหว่าง x กับ y
x | y	การเทียบบิตแบบ OR ระหว่าง x กับ y
x ^ y	การเทียบบิตแบบ XOR ระหว่าง x กับ y
~x	เพิ่มค่าบิตให้ 1 จากนั้นจะให้ผลลัพธ์ของบิตมีค่าตรงข้าม

ตัวดำเนินการเลื่อนบิต (Shift bib)

การย้ายตำแหน่งของบิตต่าง ๆ ในข้อมูลไปในทิศทางและจำนวนครั้งที่ต้องการเป็นผลให้ค่าข้อมูลนั้น ๆ เกิดการเปลี่ยนแปลงค่าไปตามการเลื่อนของบิต ประกอบด้วย

x << n	เลื่อนบิตในตัวแปร x ไปทางซ้าย n บิต
x >> n	เลื่อนบิตในตัวแปร x ไปทางขวา n บิต
x >>> n	เลื่อนบิตแบบซีโรฟิลล์ในตัวแปร x ไปทางขวา n บิต

ตัวดำเนินการเลื่อนบิต (Shift bib)

ลำดับที่	ตัวดำเนินการ
1	( )
2	++ -- ! ~
3	* / %
4	+ -
5	<< >> >>>

6	< <= > >=
7	== !=
8	&
9	^
10	|
11	&&
12	||
13	= += -= *= /= %= <<= >>= >>>= &= ^= !=

<u>คำสั่งการพิมพ์ของ JavaScript</u>

เป็นโค้ดคำสั่งจาวาสคริปต์สำหรับการพิมพ์ข้อความใด ๆ และการพิมพ์คำสั่งของเอกสาร HTML ภายใตขอบเขตของคำสั่งสคริปต์ มีรายละเอียดดังนี้

document.write(“ ข้อมูล ”);

ตัวอย่าง

```
<HTML>
<HEAD>
<TITLE> JavaScript </TITLE>
</HEAD>
<BODY>
  <SCRIPT LANGUAGE="JavaScript">
    var name = "Soraya";
    var age = 23;
    document.write("Miss "+name+"<BR>");
    document.write("Your age : "+age);
  </SCRIPT>
</BODY>
</HTML>
```

```
<HTML>
<HEAD>
<TITLE> JavaScript </TITLE>
</HEAD>
<BODY>
  <SCRIPT LANGUAGE="JavaScript">
    code = "CS101";
    desc = "Hyper Text Markup Language";
    price = 220;
    quant = 120;
    total = quant*price;
    document.write("Book code: "+code+"<BR>");
    document.write("Description: "+desc+"<BR>");
    document.write("Total: "+total);
  </SCRIPT>
</BODY>
</HTML>
```

คำสั่งการแสดงกรอบโต้ตอบ

เป็นการแสดงข้อมูลอีกวิธีหนึ่ง ซึ่งอยู่ในรูปแบบของกรอบโต้ตอบที่เรียกว่า Dialog Box โดยสามารถแสดงข้อมูลได้ เช่นเดียวกับคำสั่ง document.write("…");

```
alert(" ข้อมูล ");
confirm(" ข้อมูล ");
```

ปุ่ม OK คืนค่าจริง (True) ให้กับ JavaScript

ปุ่ม Cancel คืนเท็จ (False) ให้กับ JavaScript

```
<HTML>
<HEAD>
<TITLE> JavaScript </TITLE>
</HEAD>
<BODY>
  <SCRIPT LANGUAGE="JavaScript">
    Hello = "Hello JavaScript";
    Nice = "Nice to meet you";
    alert(Hello);
    confirm(Nice);
  </SCRIPT>
</BODY>
</HTML>
```

การรับข้อมูลทางแป้นพิมพ์

การรับข้อมูลทางแป้นพิมพ์ (Input Data) ในภาษา JavaScript จะเป็นลักษณะของการแสดงกรอบโต้ตอบเพื่อรอรับข้อมูลที่ป้อนจากแป้นพิมพ์ โดยจะมีปุ่ม OK เป็นปุ่มตอบรับข้อมูล และมีปุ่ม Cancel เป็นปุ่มในการยกเลิกข้อมูล

```
ตัวแปร = prompt("ข้อความนำ", "ค่าเริ่มต้น");
```

ตัวแปร ตัวแปรที่อ้างอิงเพื่อรอรับเก็บข้อมูลที่ป้อนได้จากแป้นพิมพ์

ข้อความนำ ข้อความที่ต้องการให้ปรากฏแสดงในกรอบโต้ตอบ เพื่อบอกว่าต้องการให้ทำอะไร

ค่าเริ่มต้น ค่าของข้อมูลหรือข้อความใด ๆ ที่ต้องการให้ปรากฏในช่องกรอกข้อมูล โดยจะกำหนดหรือไม่ก็ได้ แต่ถ้าไม่กำหนดภายในกรอบจะแสดงข้อความ undefined ออกมา

ตัวอย่าง

```
<HTML>
<HEAD>
<TITLE> JavaScript </TITLE>
</HEAD>
<BODY>
  <SCRIPT LANGUAGE="JavaScript">
    name = prompt("Input your name : ");
    age = prompt("Input your age : ");
    document.write("Hello Khun : "+name+"<BR>");
    document.write("Your age is ..."+age);
  </SCRIPT>
</BODY>
</HTML>
```

<u>การแปลงตัวเลขสตริงให้คำนวณได้</u>

โดยปกติในการรับหรือป้อนข้อมูลใด ๆ ในภาษาจาวาสคริปต์ ข้อมูลเหล่านั้นจะมีชนิดข้อมูลเป็นสตริง (String Type) ถ้าเป็นตัวเลขจะใช้ในการคำนวณไม่ได้ด้วยเหตุนี้ ถ้าเราต้องการให้ข้อมูลตัวเลขที่ป้อนเข้าไปกำหนดค่าให้กับตัวแปรนั้นคำนวณได้ ก็ต้องมีฟังก์ชั่นช่วยในการแปลงค่าตัวเลขสตริงให้คำนวณได้ ได้แก่

- **eval(ตัวแปร)** แปลงค่าสตริงใด ๆ ให้เป็นตัวเลข แล้วคืนค่านั้นให้กับตัวแปรที่กำหนด แต่ถ้ามีการป้อนข้อมูลตัวอักษรใด ๆ ของสตริงที่ไม่สามารถแปลงเป็นตัวเลขคำนวณได้ ฟังก์ชั่นนี้จะแสดงข้อความเตือน "..is not defined"
- **parseFloat(ตัวแปร)** แปลงค่าสตริงใด ๆ ให้เป็นตัวเลขจำนวนจริง แล้วคืนค่านั้นให้กับตัวแปรที่กำหนด แต่ถ้ามีการป้อนตัวอักษรใด ๆ ของสตริงที่ไม่สามารถแปลงเป็นตัวเลขคำนวณได้ ฟังก์ชั่นนี้จะคืนค่า ๆ "NaN" ให้กับจาวาสคริปต์
- **parseInt(ตัวแปร, เลขฐาน)** แปลงค่าสตริงใด ๆ ให้เป็นตัวเลขจำนวนเต็ม แล้วคืนค่านั้นให้กับตัวแปรที่กำหนด ถ้าไม่ระเลขฐาน ค่าตัวเลขที่รับมานั้นจะมีค่าเป็นเลขฐาน 10 ทันที แต่ถ้าต้องการตัวเลขฐาน 8 และ ฐาน 16 ให้กำหนดที่ตำแหน่งเลขฐาน

```
<HTML>
<HEAD>
<TITLE> JavaScript </TITLE>
</HEAD>
<BODY>
  <SCRIPT LANGUAGE="JavaScript">
     a = prompt("Input value A : "); aa = eval(a);
     b = prompt("Input value B : "); bb = parseFloat(b);
     c = prompt("Input value C : "); cc = parseInt(c,8);
     document.write("Value A is : "+aa+"<BR>");
     document.write("Value B is : "+bb+"<BR>");
     document.write("Value C is : "+cc+"<BR>");
  </SCRIPT>
</BODY>
</HTML>
```

4.1.5 คำสั่งเงื่อนไข

การเขียนโปรแกรมแบบมีเงื่อนไข เป็นลักษณะของการเปรียบเทียบข้อมูลเพื่อหาหนทางการทำงานที่ถูกต้องให้เป็นไปตามความต้องการของเงื่อนไข หรือของทิศทางการทำงานที่ต้องการ โดยผลของการเปรียบเทียบข้อมูลหรือการทดสอบเงื่อนไขมีความเป็นไปได้ 2 ทิศทาง คือ ผลการเปรียบเทียบที่ให้ผลค่าเป็นจริง (True) และผลการเปรียบเทียบที่ให้ผลค่าเป็นเท็จ (False)

นิพจน์เงื่อนไข (Condition Expression)

เป็นวิธีการหนึ่งในการกำหนดเงื่อนไขเปรียบเทียบข้อมูลอย่างง่าย ในรูปแบบของประโยคคำสั่ง เพื่อหาทิศทางการทำงานให้เป็นไปตามเงื่อนไขที่กำหนดแล้วเก็บค่าที่ได้นั้นให้กับตัวแปร

รูปแบบ

```
ตัวแปร = (เงื่อนไขการเปรียบเทียบข้อมูล) ? ค่าที่1 : ค่าที่2 ;
```

ถ้าเงื่อนไขการเปรียบเทียบข้อมูลได้เป็นจริง ตัวแปรทางด้านซ้าย จะมีค่าเท่ากับ ค่าที่ 1 และถ้าเงื่อนไขการเปรียบเทียบข้อมูลได้เป็นเท็จ ตัวแปรทางด้านซ้าย จะมีค่าเท่ากับ ค่าที่ 2

การเปรียบเทียบข้อมูลด้วยคำสั่ง IF

รูปแบบ

```
if (เงื่อนไขการเปรียบเทียบข้อมูล)
    คำสั่งที่ 1 ;
else
    คำสั่งที่ 2 ;
```

ถ้าเงื่อนไขการเปรียบเทียบข้อมูลได้เป็นจริง จะทำงานตามทิศทางของคำสั่งที่ 1 แล้วจบประโยคคำสั่ง if ถ้าเงื่อนไขการเปรียบเทียบข้อมูลได้เป็นเท็จ จะทำงานตามทิศทางของคำสั่งที่ 2 แล้วจบประโยคคำสั่ง if ถ้าในกรณีที่ทิศทางการทำงานของการเปรียบเทียบเงื่อนไขที่เป็นเท็จไม่มี ก็ให้ละไม่ต้องเขียนประโยคคำสั่ง else

สำหรับในกรณีที่ประโยคคำสั่งสำหรับเงื่อนไขการเปรียบเทียบข้อมูลได้เป็นจริง หรือได้เป็นเท็จ มีมากกว่า 1 ประโยคคำสั่ง ให้เขียนกลุ่มประโยคคำสั่งเหล่านั้นอยู่ในบล็อกคำสั่งโดยเขียนภายใต้เครื่องหมายวงเล็กปีกกา มีรูปแบบ ดังนี้

```
if (เงื่อนไขการเปรียบเทียบข้อมูล)
    { คำสั่งที่ 1 เมื่อเป็นจริง;
      คำสั่งที่ 2 เมื่อเป็นจริง; }
else
    { คำสั่งที่ 1 เมื่อเป็นเท็จ ;
      คำสั่งที่ 2 เมื่อเป็นเท็จ; }
```

ตัวอย่าง

```
<HTML>
<HEAD>
<TITLE> JavaScript </TITLE>
</HEAD>
<BODY>
   <SCRIPT LANGUAGE="JavaScript">
      name = prompt("Input student name : ");
      score = prompt("Input score test : ");
      if(score < 50)
         test = "FAIL";
      else
         test = "PASS";
      document.write("Student name : "+name+"<BR>");
      document.write("Test is : "+test);
   </SCRIPT>
</BODY>
</HTML>
```

การเขียนคำสั่ง if ซ้อนเงื่อนไข

เป็นการเขียนโครงสร้างการเปรียบเทียบข้อมูลอย่างมีเงื่อนไขที่มีหลาย ๆ ชุดคำสั่งของประโยคเงื่อนไขซ้อนกัน รูปแบบ

```
if (เงื่อนไขการเปรียบเทียบข้อมูลที่ 1)
      คำสั่งที่ 1;
else if (เงื่อนไขการเปรียบเทียบข้อมูลที่ 2)
      คำสั่งที่ 2;
else if (เงื่อนไขการเปรียบเทียบข้อมูลที่ 3)
      คำสั่งที่ 3;
else..... ต่อ ๆ ไปจนจบเงื่อนไขที่ต้องการ
.....
```

ตัวอย่าง

```
<SCRIPT LANGUAGE="JavaScript">
   name = prompt("Input employee name : ");
   salary = prompt("Input salary : ");
   if(salary < 8000)
      tax = 0.03 * salary;
   else if(salary < 15000)
      tax = 0.05 * salary;
   else
      tax = 0.07 * salary;
   net = salary - tax;
   document.write("Employee name : "+name+"<br>");
   document.write("Get salary : "+salary+"<br>");
   document.write("Tax : "+tax+"<br>");
   document.write("Net-salary : "+net+"<br>");
</SCRIPT>
```

การคัดเลือกข้อมูลด้วยคำสั่ง switch

ใช้ในการแก้ปัญหาการเปรียบเทียบข้อมูลด้วยคำสั่ง if ในกรณีที่มีหลายชุดคำสั่งมีการเปรียบเทียบจากค่าคงที่ซ้อนกันอยู่ภายในโครงสร้างซึ่งการเขียนด้วยประโยคคำสั่ง if มีผลทำให้รูปแบบการเขียนโปรแกรมดูยาว อาจทำให้เกิดการสับสน การคัดเลือกข้อมูลด้วยคำสั่ง switch จะทำการเปรียบเทียบค่าของตัวแปรว่าตรงกับค่าคงที่ใด และค่าคงที่นี้จะไม่อยู่ในลักษณะของช่วงข้อมูล

รูปแบบ

```
switch (ตัวแปร)
{
      case ค่าคงที่ 1 : คำสั่ง 1 ;
            break ;
      case ค่าคงที่ 2 : คำสั่ง 2 ;
            break ;
      ..... ;
      default : คำสั่งสุดท้าย ;
}
```

เมื่อผลจากการทดสอบตัวแปรที่อยู่ในวงเล็บหลังคำสั่ง switch ให้ผลลัพธ์หรือค่าของข้อมูลตรงกับค่าคงที่ของตัวเลือก case ใด ๆ ก็จะไปดำเนินงานตามคำสั่งนั้น แต่ถ้าผลการทดสอบไม่ตรงกับค่าใด ๆ เลยโปรแกรมจะทำงานตามคำสั่งสุดท้ายที่กำหนดไว้ใน default

คำสั่ง break เป็นคำสั่งบอกการจบสิ้นของคำสั่ง case โดยจะไม่มีการทดสอบค่าใด ๆ อีก แต่ถ้าไม่มีคำสั่ง break จะมีผลให้ประโยคคำสั่งจะดำเนินงานจากคำสั่งที่ตรงตามค่าคงที่ของตัวเลือก case ลงมาทำคำสั่งอื่น ๆ ที่เหลืออยู่ถัดไป

```
<HTML>
<HEAD>
<TITLE> JavaScript </TITLE>
</HEAD>
<BODY>
  <SCRIPT LANGUAGE="JavaScript">
    num = eval(prompt('Input value : '));
    document.write('Test value is : <br>');
    switch(num)
    {
      case 1 : document.write('ONE');
               break;
      case 2 : document.write('TWO');
               break;
      case 3 : document.write('THREE');
               break;
      default : document.write('Another...');
    }
  </SCRIPT>
</BODY>
</HTML>
```

4.1.6 คำสั่งวนรอบ

การวนรอบทำงาน (loop) เป็นลักษณะการทำงานของคำสั่งที่มีการทำซ้ำกับงานที่ทำไปแล้วตามจำนวนครั้ง หรือตามเงื่อนไขการเปรียบเทียบของข้อมูลให้ดำเนินการวนรอบทำงานตามที่กำหนด การวนรอบทำงานจะช่วยลดความจำเจของการพิมพ์คำสั่งในกรณีที่ต้องเขียนคำสั่งนั้นซ้ำและทำงานเหมือนเดิมอยู่เรื่อย ๆ จึงช่วยให้การเขียนโปรแกรมสั้นลงและมีผลทำให้การประมวลผลของโปรแกรมในคอมพิวเตอร์ทำงานได้เร็วขึ้น

<u>การใช้คำสั่ง for</u>

เป็นคำสั่งให้โปรแกรมทำงานซ้ำ ๆ กัน ตามจำนวนรอบที่กำหนด เป็นการกำหนดค่าล่วงหน้าให้กับโปรแกรม เนื่องจากเราจะรู้จำนวนรอบการทำงานที่แน่นอน โดยกำหนดค่าเริ่มต้นให้กับตัวแปร จากนั้นให้ตัวแปรทำการปรับปรุงค่าเพื่อวนรอบการทำงานกลับมาทดสอบค่าตามเงื่อนไขที่กำหนด จนจบค่าสุดท้ายที่ตั้งไว้จึงหยุดการทำงาน

รูปแบบ

```
for (ค่าเริ่มต้น; เงื่อนไขเช็คค่าปลาย; ค่าเพิ่ม)
    คำสั่งที่ต้องการดำเนินการ ;
```

- ค่าเริ่มต้น (initial expression) เป็นการกำหนดค่าเริ่มต้นให้กับตัวแปร
- เงื่อนไขเช็คค่าปลาย (condition) เป็นการเช็คเงื่อนไขเปรียบเทียบค่าปลายของตัวแปร หากนิพจน์หารเปรียบเทียบมีค่าเป็นจริง โปรแกรมจะดำเนินงานทำตามคำสั่งที่กำหนดจนจบ แล้ววนกลับมาเช็คเงื่อนไขเดิมอีก ทำเช่นนี้ยนกว่านิพจน์การเปรียบเทียบนั้น จะให้ค่าเป็นเท็จ จึงออกจากโครงสร้างของของคำสั่ง for
- ค่าเพิ่ม (Update expression) เป็นการเพิ่มค่าหรือปรับปรุงค่าให้กับข้อมูลของตัวแปร สำหรับการวนรอบทำงานแต่ละครั้ง

สำหรับในกรณีที่ประโยคคำสั่งของการวนรอบทำงานมีมากกว่า 1 ประโยคคำสั่ง ให้เขียนกลุ่มประโยคคำสั่งเหล่านั้นอยู่ในบล็อคคำสั่ง โดยเขียนภายใต้เครื่องหมายวงเล็บปีกกาดังนี้

```
{ กลุ่มของประโยคคำสั่ง }
```

<u>Ex</u> ต้องการพิมพ์อนุกรมของตัวเลข 1 2 3 4 5 6 7 8 9 10 จะเขียนรูปแบบคำสั่งของ for ได้ดังนี้

```
for(a =1; a <= 10; a++) {
    document.write(a+“ ”);
}
```

โดยกำหนดให้ตัวแปร a ทำหน้าที่วนรอบการทำงานพิมพ์ค่าของ a ตั้งแต่ 1 ถึง 10

<u>Ex</u> ต้องการพิมพ์อนุกรมของตัวเลข 10 9 8 7 6 5 4 3 2 1 จะเขียนรูปแบบคำสั่งของ for ได้ดังนี้

```
for(b =10; b >= 10; b- -) {
    document.write(b+“ ”);
}
```

โดยกำหนดให้ตัวแปร b ทำหน้าที่วนรอบการทำงานพิมพ์ค่าของ b ตั้งแต่ 10 ลดลงถึง 1

ตัวอย่าง

```
<HTML>
<HEAD>
<TITLE> JavaScript </TITLE>
</HEAD>
<BODY>
  <SCRIPT LANGUAGE="JavaScript">
    for(loop = 1; loop <= 10; loop++){
      document.write("Hello JavaScript Loop : "+loop+"<br>");
    }

  </SCRIPT>
</BODY>
</HTML>
```

การใช้คำสั่ง while

เป็นอีกวิธีหนึ่งของการวนรอบการทำงาน ที่มีความยืดหยุ่นและมีลูกเล่นมากกว่าคำสั่ง for มีความสามารถวนรอบการทำงานโดยทำการตรวจสอบการเปรียบเทียบข้อมูลกับเงื่อนไขที่กำหนด โดยกำหนดลักษณะการวนรอบการทำงานว่าจะทำงานในขณะที่การเปรียบเทียบข้อมูลจากเงื่อนไขที่กำหนดนั้นมีค่าเป็นจริง และจะออกจากข้อคำสั่ง while เมื่อผลจากการทดสอบข้อมูลกับเงื่อนไขนั้นมีค่าเป็นเท็จ

รูปแบบ

```
while( เงื่อนไขการเปรียบเทียบข้อมูล ) {
        คำสั่ง ;
}
```

ตัวอย่าง

```
<HTML>
<HEAD>
<TITLE> JavaScript </TITLE>
</HEAD>
<BODY>
  <SCRIPT LANGUAGE="JavaScript">
    loop = 1;
    while(loop <= 10){
      document.write("Hello JavaScript loop : "+loop+"<br>");
      loop++;
    }
  </SCRIPT>
</BODY>
</HTML>
```

การใช้คำสั่ง break

ใช้สำหรับขัดจังหวะของการวนรอบทำงานของคำสั่ง for และ while เพื่อให้ออกจากโครงสร้างการทำงานภายในวนรอบการทำงานนั้น ๆ ตามความต้องการทันที

ตัวอย่าง โปรแกรมพิมพ์ข้อความ Hello JavaScript เป็นจำนวน 10 รอบ โดยให้ตัวแปร loop ทำหน้าที่วนรอบการทำงานจากนั้นให้ใช้คำสั่ง break ขัดจังหวะการทำงานของการวนรอบทำงานเมื่อพิมพ์ข้อความไปแล้ว 5 ครั้ง

```
<HTML>
<HEAD>
<TITLE> JavaScript </TITLE>
</HEAD>
<BODY>
  <SCRIPT LANGUAGE="JavaScript">
    loop = 1;
    while(loop <= 10){
      document.write("Hello JavaScript <br>");
      if(loop == 5)
        break;
      loop++;
    }
    document.write("Stop when loop is :"+loop);
  </SCRIPT>
</BODY>
</HTML>
```

การใช้คำสั่ง continue

เป็นการขัดจังหวะของการวนรอบการทำงานของคำสั่ง for และ while เพื่อสั่งให้โปรแกรมกลับไปเริ่มต้นทำงานใหม่จากจุดเริ่มต้นวนรอบการทำงานทันที โดยไม่ต้องรอให้โปรแกรมทำงานต่อไปจนครบรอบการทำงานปกติ

ตัวอย่าง โปรแกรมพิมพ์ข้อความ Hello JavaScript เป็นจำนวน 10 รอบ โดยให้ตัวแปร loop ทำการวนรอบการทำงานแล้วพิมพ์ค่าของ loop ในแต่ละรอบจากนั้นให้ใช้คำสั่ง continue ขัดจังหวะการทำงานของการวนรอบทำงานเมื่อพิมพ์ข้อความไปแล้ว 5 ครั้ง

```
<HTML>
<HEAD>
<TITLE> JavaScript </TITLE>
</HEAD>
<BODY>
  <SCRIPT LANGUAGE="JavaScript">
    for(loop = 1; loop <=10; loop++){
      if(loop ==5)
        continue;
      document.write("Hello JavaScript loop : "+loop+"<br>");
    }
  </SCRIPT>
</BODY>
</HTML>
```

ตัวแปรอาร์เรย์ (Array)

ตัวแปรอาร์เรย์ หรือ ตัวแปรชุด เป็นการจองพื้นที่ในหน่วยความจำให้กับตัวแปร เพื่อให้ตัวแปรนั้นสามารถรับข้อมูลได้มากกว่า 1 ค่า โดยค่าแต่ละค่านั้นจะถูกเก็บลงยังพื้นที่ของหน่วยความจำที่แตกต่างกัน พื้นที่ในหน่วยความจำจะใช้หมายเลขลำดับเป็นตัวจำแนกความแตกต่าง ของค่าตัวแปรแต่ละพื้นที่ด้วยการระบุหมายเลขลำดับที่เรียกว่า ดัชนี(index) กำกับตามหลังชื่อตัวแปร โดยใช้หมายเลขลำดับตั้งแต่ 0, 1, 2, ….. เป็นต้นไป

การกำหนดตัวแปรอาร์เรย์ในภาษาจาวาสคริปต์ จะมีความยืดหยุ่นมากในเรื่องของการกำหนดจำนวนสมาชิกของอาร์เรย์ นั่นคือจำนวนสมาชิกหรือจำนวนพื้นที่ในการจองสำหรับการรับข้อมูลจะถูกกำหนดขนาดให้โดยอัตโนมัติ สามารถเพิ่มจำนวนสมาชิกได้ แต่ลดจำนวนลงไม่ได้ และมีสิ่งหนึ่งที่ควรจำก็คือ ยิ่งมีการกำหนดจำนวนสมาชิกหรือการจองพื้นที่ในการรับข้อมูลมากเท่าใด ก็ยิ่งสิ้นเปลืองเนื้อที่ในหน่วยความจำมากเท่านั้น และการทำงานก็จะช้าลงตามไปด้วย

การสร้างตัวแปรอาร์เรย์

ก่อนที่จะมีการใช้งานอาร์เรย์นั้นเราจะต้องทำการประกาศตัวแปรที่มีลักษณะเป็นอาร์เรย์เสียก่อน เพื่อให้โปรแกรมรู้จักชื่อของตัวแปรที่จะกำหนดเป็นอาร์เรย์ พร้อมถึงการกำหนดขนาดของพื้นที่ในหน่วยความจำสำหรับเก็บค่าของข้อมูล

การประกาศตัวแปรอาร์เรย์

รูปแบบ

ชื่อตัวแปรอาร์เรย์ = new Array(จำนวนสมาชิก);

- จำนวนสมาชิก หมายถึง การกำหนดการจองพื้นที่ในหน่วยความจำ ให้กับตัวแปรเพื่อรองรับข้อมูลที่กำหนด โดยปกติจะกำหนดหรือไม่ก็ได้ เพราะอาร์เรย์ของจาวาสคริปต์มีความยืดหยุ่นมากสำหรับในการรับจำนวนสมาชิก

การกำหนดค่าให้กับตัวแปรอาร์เรย์

รูปแบบ

ตัวแปร [index] = ข้อมูล ;

- index หมายถึง หมายเลขลำดับของพื้นที่ที่เก็บข้อมูลโดยเริ่มนับตั้งแต่ 0, 1, 2, … เป็นต้น

ตัวอย่าง

```
<HTML>
<HEAD>
<TITLE> JavaScript </TITLE>
</HEAD>
<BODY>
   <SCRIPT LANGUAGE="JavaScript">
      myarray = new Array(3);
      myarray[0] = "first";
      myarray[1] = "second";
      myarray[2] = "third";
      for(i=0; i<=2; i++)
         document.write(myarray[i]+"<br>");
   </SCRIPT>
</BODY>
</HTML>
```

การรับข้อมูลให้กับตัวแปรอาร์เรย์

การรับข้อมูลให้กับตัวแปรอาร์เรย์ เป็นวิธีการในการกำหนดค่าให้กับตัวแปรอาร์เรย์ โดยป้อนข้อมูลที่ต้องการผ่านทางแป้นพิมพ์ โดยใช้คำสั่ง prompt(….) ในการรับข้อมูล

รูปแบบ

ตัวแปร [index] = prompt(“ข้อความนำ”, “ค่าเริ่มต้น”);

ตัวอย่าง โปรแกรมรับชื่อของบุคคล 3 คน ลงในตัวแปรอาร์เรย์ name จากนั้นให้พิมพ์ชื่อของบุคคลทั้ง 3

```
<HTML>
<HEAD>
<TITLE> JavaScript </TITLE>
</HEAD>
<BODY>
   <SCRIPT LANGUAGE="JavaScript">
      name = new Array(3);
      alert("Please enter to begin... Input name 3 persons : ");
      for(i=0; i<=2; i++)
         name[i] = prompt("Input name : ", "Noname");
      document.write("Print name : "+"<br><br>");
      for(i=0;i<=2; i++)
         document.write(name[0]+"<br>");
   </SCRIPT>
</BODY>
</HTML>
```

การกำหนดค่าเริ่มต้นให้กับอาร์เรย์

การกำหนดค่าเริ่มต้นให้กับอาร์เรย์ (initial value) เป็นการกำหนดค่าโดยตรงให้กับตัวแปรอาร์เรย์ ในขณะที่มีการประกาศเริ่มใช้ตัวแปรอาร์เรย์ ค่าที่กำหนดนี้ถือว่าเป็นค่าเริ่มต้นในการกำหนดค่าใด ๆ ให้กับตัวแปรอาร์เรย์ ค่าเหล่านี้มักใช้ในกรณีเหมือนค่าคงที่ ใช้สำหรับเปรียบเทียบหรือใช้สำหรับอ้างอิงกับข้อมูลอื่น ๆ และโดยทั่วไปก็ยังคงมีการใช้งานและกำหนดค่าเหมือนตัวแปรอาร์เรย์ปกติ มีรูปแบบดังนี้

รูปแบบ

ชื่อตัวแปรอาร์เรย์ = new Array(ข้อมูล 1, ข้อมูล 2, …..);

ตัวอย่าง โปรแกรมกำหนดชื่อวิชา Internet, HyperText Markup Language และ JavaScript โดยให้เป็นค่าเริ่มต้นกับตัวแปรอาร์เรย์ subject

```
<HTML>
<HEAD>
<TITLE> JavaScript </TITLE>
</HEAD>
<BODY>
   <SCRIPT LANGUAGE="JavaScript">
      subject = new Array("Internet","HyperText Markup Language","JavaScript");
      for(i=0; i<=2; i++)
         document.write("Subject : "+subject[i]+"<br>");
   </SCRIPT>
</BODY>
</HTML>
```

4.1.7 ฟังก์ชั่น

ฟังก์ชั่นใน JavaScript มีด้วยกันสองแบบ คือ

- ฟังก์ชันมาตรฐาน (Standard function)เป็นแบบชื่อของฟังก์ชันที่มีอยู่แล้วในภาษา JavaScript เราสามารถนำเอาไปใช้งานได้ทันที
- ฟังก์ชันสร้างขึ้นเอง (User-defined function) เป็นแบบชื่อของฟังก์ชันที่ผู้ใช้สร้างขึ้นมาใช้เอง เพื่อกำหนดให้ทำงานใดงานหนึ่งจนสำเร็จ

ฟังก์ชันมาตรฐานในภาษา JavaScript

abs	acos	alert	anchor	asin
atan	back	big	blink	blur
bold	ceil	charAt	clear	clearTimeout
click	close(document)	close(window)	confirm	cos
escape	eval	exp	fixed	floor
focus	fontcolor	fontsize	forward	getDate
getDay	getHours	getMinutes	getMonth	getSeconds
getTime	getTimezoneOffsets	getYear	go	indexOf
isNaN	italics	lastindexOf	link	log
max	min	open (document)	open (window)	parse
parseFloat	parseInt	pow	promt	random
round	select	setDate	setHours	setMinutes
setMounth	setSeconds	setTime	setTimeOut	setYear
sin	small	sqrt	strike	sub
submit	substring	sup	tain	tan
toGMTString	toLocaleString	toLowerCase	toUpperCase	toString
unescape	untain	UTC	write	writeln

การเรียกใช้ฟังก์ชัน

การเรียกใช้ฟังก์ชัน เป็นการกำหนดทิศทางการทำงานของคำสั่ง หรือ กำหนดส่วนของโปรแกรมให้ทำงานใดงานหนึ่งจนสำเร็จ โดยอ้างอิงชื่อฟังก์ชันของการทำงานที่ต้องการผลของการเรียกใช้ฟังก์ชันจะมีการส่งค่าคืนกลับไปยังโปรแกรมส่วนที่เรียกโดยใช้ชื่อฟังก์ชันเป็นตัวเก็บค่าเปรียบเสมือนหนึ่งกับเป็นตัวแปรมีรูปแบบการเรียกใช้ดังนี้

รูปแบบ

```
ตัวแปร = ชื่อฟังก์ชัน( );
```

ตัวอย่าง

```
<HTML>
<HEAD>
<TITLE> JavaScript </TITLE>
</HEAD>
<BODY>
   <SCRIPT LANGUAGE="JavaScript">
      a = 125.75;
      b = parseInt(a);
      document.write("Value b : "+b+"<br>");
      document.write("Test function : "+parseInt(125.75));
   </SCRIPT>
</BODY>
</HTML>
```

การสร้างฟังก์ชันขึ้นใช้เอง

การสร้างฟังก์ชันขึ้นใช้เอง เป็นฟังก์ชันที่ผู้ใช้สร้างขึ้นมาเพื่อกำหนดให้ทำงานใดงานหนึ่งจนสำเร็จ การสร้างฟังก์ชันจะขึ้นต้นด้วยคำว่า function ตามด้วยชื่อของฟังก์ชันที่ต้องการ มีรายละเอียดดังนี้

รูปแบบ

```
function  ชื่อฟังก์ชัน ( พารามิเตอร์1, พารามิเตอร์2, .....)
{
      ข้อคำสั่ง ;
      ..............................
}
```

- ชื่อฟังก์ชัน (function name) ชื่อของฟังก์ชันที่สร้างขึ้น เพื่อใช้ในการอ้างอิงเรียกใช้งานให้ทำงานใดงานหนึ่งจนสำเร็จแล้วส่งผลลัพธ์ที่ได้กลับไปยังโปรแกรมตัวเรียกที่เรียกใช้งานฟังก์ชัน
- พารามิเตอร์ (parameter) ค่าของข้อมูลหรือตัวแปรที่อ้างอิง (argument) ในการส่งผ่านค่าจากโปรแกรมตัวเรียกมายังชื่อฟังก์ชันที่กำหนด เพื่อนำค่านั้นมาใช้งานในฟังก์ชัน
- ข้อคำสั่ง (statements) คำสั่งต่าง ๆ ที่ประกอบกันขึ้นเพื่อให้ดำเนินงานภายในฟังก์ชัน ต้องกำหนดคำสั่ง ต่าง ๆ ภายใต้เครื่องหมายวงเล็บปีกกา {…}

การวางตำแหน่งฟังก์ชัน

สำหรับการวางตำแหน่งฟังก์ชั่นในภาษาจาวาสคริปต์ ก็มีลักษณะเดียวกับการวางตำแหน่งสคริปต์ นั่นคือจะวางไว้ในส่วนของ <HEAD> หรือ <BODY> ก็ได้ ขึ้นอยู่กับว่าต้องการให้ฟังก์ชันนั้นถูกโหลดใช้งานก่อนหรือหลังตามลำดับการเรียกใช้งานอย่างไร

ในกรณีฟังก์ชันนั้นมีการถูกเรียกใช้บ่อยครั้งจากส่วนอื่น ๆ ของโปรแกรม แนะนำว่าควรจะกำหนดฟังก์ชันไว้ในส่วนของ <HEAD> เพราะเมื่อมีการเรียกใช้โหลดเว็บเพจขึ้นมา ฟังก์ชันต่าง ๆ ที่กำหนดในส่วน <HEAD> จะถูกโหลดเข้าเก็บไว้ในหน่วยความจำก่อน ทำให้เราสามารถเรียกใช้ฟังก์ชันจากตำแหน่ง ๆ ใด ๆ บนเอกสาร HTML หรือบนขอบเขต <SCRIPT> ได้อย่างต่อเนื่อง และนอกจากนี้ฟังก์ชันยังสามารถเรียกใช้ฟังก์ชันอื่น ๆ ที่กำหนดในส่วน <HEAD> ทำงานร่วมกันได้อีกด้วย

```
<HTML>
<HEAD><TITLE> ข้อความแถบเรื่อง </TITLE>
<SCRIPT Language = “JavaScript”>
        function  ชื่อฟังก์ชัน (พารามิเตอร์1, พารามิเตอร์2, …)
        {    ข้อคำสั่ง ;
             ……
        }
        กลุ่มโค้ดคำสั่งของจาวาสคริปต์
</SCRIPT>
</HEAD>
<BODY>
        ข้อความเอกสาร HTML
        ……
        <SCRIPT Language = “JavaScript”>
              กลุ่มโค้ดคำสั่งจาวาสคริปต์
       </SCRIPT>
</BODY>
</HTML>
```

ตัวอย่าง เขียนโปรแกรมให้มีฟังก์ชันชื่อ hello( ) เพื่อทำการพิมพ์ข้อความ Hello JavaScript… โดยตัวอย่างนี้ให้เรียกใช้ฟังก์ชัน hello( ) ในส่วนของ <HEAD>

```
<HTML>
<HEAD><TITLE> JavaScript </TITLE>
<SCRIPT LANGUAGE="JavaScript">
   function hello()
   {
      document.write("Hello JavaScript ..."+"<br>");
   }
   document.write("<center>");
   document.write("เรียกใช้ฟังก์ชัน ในส่วนของ HEAD"+"<br>");
   hello();
   hello();
   document.write("</center>");
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

ตัวอย่าง เขียนโปรแกรมโดยให้มีฟังก์ชันชื่อ hello( ) เพื่อทำการพิมพ์ข้อความ Hello JavaScript… เช่นเดิม โดยตัวอย่างนี้ให้เรียกใช้ฟังก์ชันในส่วนของ <BODY>

```
<HTML>
<HEAD><TITLE> JavaScript </TITLE>
<SCRIPT LANGUAGE="JavaScript">
   function hello()
   {
      document.write("Hello JavaScript ..."+"<br>");
   }
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
   document.write("เรียกใช้ฟังก์ชัน ในส่วนของ BODY"+"<br>");
   hello( );
   hello( );
</SCRIPT>
</BODY>
</HTML>
```

การส่งผ่านข้อมูลให้กับฟังก์ชัน

การส่งผ่านข้อมูลให้กับฟังก์ชัน (Pass by Value) เป็นการส่งผ่านค่าของข้อมูลหรือตัวแปรที่อ้างอิงที่เรียกว่า อาร์กิวเมนต์ (argument) โดยการส่งผ่านค่าจากโปรแกรม ตัวเรียก มายังชื่อฟังก์ชันที่กำหนดเพื่อนำค่านั้นมาใช้งานในฟังก์ชัน

รูปแบบการเขียนฟังก์ชัน

```
function  ชื่อฟังก์ชัน( พารามิเตอร์1, พารามิเตอร์2,…)
{  ข้อคำสั่ง ;
   ……
{
```

รูปแบบการเรียกใช้

```
ชื่อฟังก์ชัน (ข้อมูล1, ข้อมูล2, ……)
```

- *ข้อมูล* คือ ค่าของข้อมูลหรือตัวแปรที่เรียกว่า อาร์กิวเมนต์ ทำหน้าที่ส่งผ่านค่าให้กับพารามิเตอร์ที่กำหนดในชื่อฟังก์ชันที่เรียกใช้ โดยมีข้อกำหนดว่าจำนวนข้อมูลที่ส่งผ่านค่าให้กับพารามิเตอร์จะต้องมีจำนวนเท่ากัน โดยที่ข้อมูลจะส่งผ่านค่าให้กับพารามิเตอร์ตามลำดับที่กำหนดไว้ในฟังก์ชัน
- *พารามิเตอร์* คือ ตัวแปรที่ทำหน้าที่รับข้อมูลที่ส่งผ่านค่ามาทางฟังก์ชันจากข้อมูลหรือตัวแปรอาร์กิวเมนต์ ชื่อของพารามิเตอร์จะเป็นชื่อเดียวกับชื่อตัวแปรอาร์กิวเมนต์หรือไม่ก็ได้

ตัวอย่าง เขียนโปรแกรมการส่งผ่านข้อมูล เรียกใช้ฟังก์ชันชื่อ sub โดยส่งผ่านค่าจากตัวแปร text และ loop ที่ได้จากการป้อนข้อมูลจากแป้นพิมพ์ ส่งค่าไปยังพารามิเตอร์ในฟังก์ชันชื่อ descrip และ n ตามลำดับของข้อมูล ฟังก์ชัน sub จะทำหน้าที่พิมพข้อความของตัวแปรพารามิเตอร์ descrip ตามจำนวนครั้งของตัวแปรพารามิเตอร์ n

```
<HTML>
<HEAD><TITLE> JavaScript </TITLE>
<SCRIPT LANGUAGE="JavaScript">
   function sub(descrip,n)
   {
      document.write("Write "+descrip+"..."+n+" Loop <br>");
      for(i=1; i<=n; i++)
         document.write(descrip+"<br>");
   }
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
   text = prompt("Pleas enter data : ","JavaScript");
   loop = prompt("Enter loop : ","1");
   sub(text,loop);
</SCRIPT>
</BODY>
</HTML>
```

ฟังก์ชันแบบส่งค่าคืนกลับ

ฟังก์ชันแบบส่งค่าคืนกลับ (Pass by reference) เป็นการส่งผ่านข้อมูลใด ๆ จากตัวแปรที่กำหนดในฟังก์ชัน หรือจากชื่อของฟังก์ชันเองคืนกลับไปยังโปรแกรมตัวเรียก โดยใช้ประโยคคำสั่ง return เป็นการส่งค่าคืนกลับ มีรายละเอียดดังนี้

รูปแบบการเขียนฟังก์ชันแบบส่งค่าคืนกลับ

```
function  ชื่อฟังก์ชัน (พารามิเตอร์1, พารามิเตอร์2, .....)
{    ข้อคำสั่ง ;
     ...................
     return( ค่าคืนกลับ );
}
```

รูปแบบการเรียกใช้ฟังก์ชัน

```
ชื่อตัวแปร  =  ชื่อฟังก์ชัน( ข้อมูล1, ข้อมูล2, ......) ;
```

- ค่าคืนกลับ คือ ค่าของข้อมูลใด ๆ หรือค่าของพารามิเตอร์หรือค่าของผลลัพธ์ที่ได้จากฟังก์ชันที่ต้องการส่งค่าคืนกลับไปยังโปรแกรมตัวเรียก โดยการใช้ประโยคคำสั่ง return(ค่าคืนกลับ) ในการส่งผ่านข้อมูล

ตัวอย่าง เขียนโปรแกรมการส่งผ่านข้อมูล เรียกใช้ฟังก์ชันชื่อ power โดยส่งผ่านค่าจากตัวแปร text และ loop ที่ได้จากการป้อนข้อมูลจากแป้นพิมพ์ ส่งค่าไปยังพารามิเตอร์ในฟังก์ชันชื่อ base และ n ตามลำดับของข้อมูล ฟังก์ชัน pow จะทำหน้าที่คำนวณค่ายกกำลัง โดย base คือเลขฐาน ส่วน n คือเลขยกกำลัง

```
<HTML>
<HEAD><TITLE> JavaScript </TITLE>
<SCRIPT LANGUAGE="JavaScript">
   function power(base,n)
   {
      result = 1;
      if(n==0)
         result = 1;
      else{
         for(i=1; i<=n; i++){
            result = base*result;
         }
      }
      return(result);
   }
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
   text = prompt("Pleas enter base data : ","1");
   loop = prompt("Enter power : ","1");
   result = power(text,loop);
   document.write(result);
</SCRIPT>
</BODY>
</HTML>
```