

JavaScript

เนื้อหา:

- JavaScript คืออะไร
- ความสามารถของ JavaScript
- การเขียน JavaScript
- รูปแบบไวยากรณ์ (Syntax) ของภาษา JavaScript
- ประเภทของข้อมูลใน JavaScript
- การประกาศตัวแปรและการกำหนดค่า
- ตัวดำเนินการ (Operators)
- คำสั่ง if, if else, else if
- คำสั่ง switch case
- คำสั่ง while loop และ do while loop

- คำสั่ง for loop
- คำสั่ง break และ continue
- ฟังก์ชัน
- Event
- การใช้งาน Array
- Regular Expression
- Exceptions และ Error
- JavaScript DOM
- JavaScript กับ JSON
- Ajax

JavaScript คืออะไร

JavaScript คือภาษาคอมพิวเตอร์สำหรับการสร้างและพัฒนาเว็บไซต์ใช้ร่วมกับ HTML และ CSS เพื่อให้เว็บไซต์สามารถตอบสนองผู้ใช้งานได้มากขึ้น JavaScript เป็นภาษาที่ถูกถอดแบบมาจากภาษา Java โดยตัดส่วนจุกจิกออกไปเพื่อให้่ายต่อการเขียนโปรแกรม

ความสามารถของ JavaScript

- สามารถเขียนโปรแกรมแบบง่ายๆ ได้ โดยไม่ต้องพึ่งภาษาอื่น
- มีคำสั่งที่ตอบสนองต่อผู้ใช้งาน
- ใช้ตรวจสอบข้อมูลได้ และแจ้งเตือนเมื่อผู้ใช้กรอกข้อมูลไม่ถูกต้อง
- ใช้ตรวจสอบผู้ใช้ได้ เช่น ตรวจสอบว่าผู้ใช้งานใช้ Browser อะไร
- สร้าง Cookies ได้
- สร้างลูกเล่นให้เว็บไซต์ได้

การเขียน JavaScript

การเขียน JavaScript สามารถเขียนได้ 2 แบบ คือ เขียนไว้ในไฟล์ HTML โดยเขียนไว้ในแท็ก <script> หรือเขียนแยกเป็นไฟล์ .js แล้วเรียกใช้ก็ได้ ซึ่งการเขียนแยกโค้ดออกเป็นไฟล์จะมีข้อดีคือ ช่วยให้โค้ด HTML กระชับดูง่าย, สามารถแยกโค้ด JavaScript ที่ใช้งานบ่อยเป็นไฟล์ เพื่อนำไปใช้ในหลายๆ पेจได้ และยังช่วยให้การแก้ไขโค้ดภายหลังทำได้ง่าย

ตัวอย่างการเขียน JavaScript ไว้ในไฟล์ HTML

HTML Code	Output
<pre><!DOCTYPE html> <html> <head> <meta charset="utf-8"> </head> <body> <h2>JavaScript</h2> <script> document.write("การเขียน JavaScript ไว้ในไฟล์ HTML"); </script> </body> </html></pre>	<p>Javascript</p> <p>การเขียน Javascript ไว้ในไฟล์ HTML</p>

ตัวอย่างการเขียน JavaScript แยกไว้ในไฟล์ .js

HTML Code	Output
<pre><!DOCTYPE html> <html> <head> <meta charset="utf-8"> <script src="assets/js/js01.js"></script> </head> <body> <h2>JavaScript</h2> <script>showMsg();</script> </body> </html></pre>	JavaScript การเขียน JavaScript แยกไว้ในไฟล์ .js
File: js01.js	
<pre>function showMsg() { document.write("การเขียน JavaScript แยกไว้ในไฟล์ .js"); }</pre>	

JavaScript Output

การแสดงผลของ JavaScript จะมี 4 วิธี ดังนี้

- แสดงออกทาง HTML element โดยใช้ property innerHTML
- แสดงออกทางเว็บเพจ โดยใช้คำสั่ง document.write()
- แสดงออกทาง alert box โดยใช้ฟังก์ชัน alert()
- แสดงออกทาง browser console โดยใช้คำสั่ง console.log()

ตัวอย่างการใช้ Output แบบต่างๆ

HTML Code	Output
<pre> <!DOCTYPE html> <html> <body> <h3>JavaScript Output</h3> <p id="demo"></p> <script> let s = "ข้อความที่แสดงด้วย property innerHTML"; document.getElementById("demo").innerHTML = s; document.write("<p>ข้อความจากคำสั่ง document.write</p>"); console.log("ข้อความจากคำสั่ง console.log"); function showAlert() { alert('ข้อความจากฟังก์ชัน alert()'); } </script> <button type="button" onclick="showAlert();">Show alert box</button> </body> </html> </pre>	<p>JavaScript Output</p> <p>ข้อความที่แสดงด้วย property innerHTML</p> <p>ข้อความจากคำสั่ง document.write</p> <p>Show alert box</p> <p>ตัวอย่าง alert box</p> <div> <p>This page says</p> <p>ข้อความจากฟังก์ชัน alert()</p> <p>OK</p> </div>

สำหรับการดูผลลัพธ์ที่แสดงทาง console ให้กด F12 แล้วดูที่ Tab Console ดังตัวอย่างภาพ



รูปแบบไวยากรณ์ (Syntax) ของภาษา JavaScript

การเขียนภาษา JavaScript มีรูปแบบไวยากรณ์ที่สำคัญดังนี้

- ทุก statement ต้องลงท้ายด้วยเครื่องหมาย ; เช่น `a = b + 2;`
- แต่ละบรรทัดสามารถเขียนได้มากกว่า 1 statement เช่น `a = 1; b = 2; x = 0;` แต่ไม่นิยม เพราะจะทำให้ดูได้ยาก
- การเขียนคอมเมนต์ เพื่ออธิบายโปรแกรม หรือเพื่อปิดโค้ดในส่วนที่ไม่ต้องการให้ทำงาน จะใช้เครื่องหมาย `//` นำหน้าข้อความคอมเมนต์ หรือ `/*` ข้อความคอมเมนต์ `*/` เช่น

JavaScript Code

```
/*  
File: script01.js  
Create by MIS-BOY, 2023-03-02  
*/  
  
//Script หลักสำหรับเว็บ winwin.co.th  
  
let n = 0; // กำหนดค่าเริ่มต้นตัวแปร n
```

- White space หรือช่องว่าง ไม่มีผลกับการประมวลผลของ JavaScript การเว้นช่องทาง 1 ช่องหรือมากกว่า มีค่าเท่ากัน
- การกำหนดค่าให้ตัวแปรประเภทสตริง (String) จะใช้เครื่องหมาย `" "` (Double quotes) หรือ `' '` (Single quotes) เช่น `let lang = "JavaScript";`
- ชื่อตัวแปร หรือชื่อฟังก์ชัน ตัวอักษรเล็ก/ใหญ่ มีความสำคัญ (Case Sensitive) หมายความว่าไม่ใช่ตัวเดียวกัน เช่น `let num = 5;` `let Num = 10;` ตัวแปร `num` กับ `Num` ไม่ใช่ตัวเดียวกัน

ประเภทของข้อมูลใน JavaScript

ประเภทข้อมูล (Data Types) ในภาษา JavaScript มีประเภทข้อมูลพื้นฐานอยู่ 6 ประเภท ได้แก่

- Number: เป็นข้อมูลชนิดตัวเลข ประกอบด้วย เลขจำนวนเต็ม (Integer) และเลขจำนวนจริง (float)
- String: เป็นข้อมูลที่เป็นข้อความ ซึ่งจะต้องกำหนดไว้ในเครื่องหมายคำพูด ("...")
- Boolean: เป็นข้อมูลทางตรรกะ มี 2 สถานะ คือ จริง (true) กับเท็จ (false)
- BigInt: เป็นข้อมูลชนิดตัวเลขขนาดใหญ่ที่มีค่าน้อยกว่า หรือมากกว่า Number
- Object: คือ ข้อมูลที่เป็น ออบเจ็ค เช่น Array, Date
- Null: เป็น object ชนิดหนึ่ง ที่ไม่มีค่า
- Undefined: คือ ไม่มีค่า ไม่ได้กำหนดค่า

การประกาศตัวแปรและการกำหนดค่า

ตัวแปร คือ สิ่งที่เรากำหนดขึ้นเพื่อใช้เก็บค่าหรือข้อมูล เพื่อใช้ในโปรแกรม การตั้งชื่อของตัวแปร สามารถประกอบไปด้วย ตัวอักษร ตัวเลข สัญลักษณ์

Underscore (_) และดอลลาร์ (\$) แต่ชื่อของตัวแปรนั้นจะขึ้นต้นด้วยตัวเลขไม่ได้ และต้องไม่ซ้ำกับคำสงวนของ JavaScript ด้วย การประกาศตัวแปรภาษา

JavaScript จะมีคำสั่งที่ใช้ประกาศตัวแปรอยู่ 3 คำสั่ง คือ var, let และ const โดยที่

- var เป็นการประกาศตัวแปรระดับ Function และ Global สามารถแก้ไขค่าของตัวแปรได้
- let เป็นการประกาศตัวแปรระดับ Block สามารถแก้ไขค่าของตัวแปรได้
- const ย่อมาจาก constant หรือค่าคงที่ เป็นการประกาศตัวแปรระดับ Block เหมือนกับ let แต่ไม่สามารถแก้ไขค่าของตัวแปรภายหลังได้ ยกเว้นเป็น object เช่น array/map จะสามารถแก้ไขค่าภายหลังได้

ตัวอย่างการประกาศตัวแปร

JavaScript Code
<pre>//ประกาศด้วยคำสั่ง var var num; var _amount; var \$name; var VAT; var total_sale; var str1;</pre>
<pre>//ประกาศด้วยคำสั่ง let let num; let _amount; let \$name; let VAT; let total_sale; let str1;</pre>

การกำหนดค่า สามารถกำหนดค่าตอนประกาศตัวแปร หรือกำหนดค่าทีหลังก็ได้ เช่น

JavaScript Code
<pre>//ประกาศตัวแปร และกำหนดค่าเริ่มต้น let SUM = 0; let weight = 52.4; let new_user = true; const \$model = "ATTO 3"; const VAT = 0.07; const Discount = 0.1; //ประกาศตัวแปรไว้ก่อน แล้วกำหนดค่าภายหลัง let salary; salary = 30000;</pre>

การประกาศตัวแปร สามารถใช้คำสั่งเดียว ประกาศตัวแปรหลายตัวได้ โดยใช้เครื่องหมาย comma (,) คั่น เช่น

JavaScript Code

```
let min = 0, max = 100;
```

การประกาศตัวแปร หลายตัวพร้อมกัน โดยกำหนดค่าเริ่มให้เป็นค่าเดียวกัน เช่น

JavaScript Code

```
let sum1 = sum2 = sum3 = 0;
```

จากตัวอย่างนี้ตัวแปร sum1, sum2 และ sum3 จะมีค่าเริ่มต้นเป็น 0

คำสงวนในภาษา JavaScript

คำสงวน คือคำที่ใช้เป็นคำสั่งเฉพาะของ JavaScript การเขียนโปรแกรม จะไม่สามารถใช้คำสงวนเหล่านี้ ตั้งชื่อตัวแปร หรือฟังก์ชันได้

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

การเปลี่ยนประเภทข้อมูล

การแปลงข้อความเป็นตัวเลข

การแปลงข้อความเป็นตัวเลข มีฟังก์ชันที่สามารถใช้งานได้ 3 ฟังก์ชันคือ Number(), parseFloat() หรือ parseInt() ดังตัวอย่างต่อไปนี้

JavaScript Code

```
let a = Number("2.34");  
let b = parseFloat("15.25");  
let c = parseInt("32");
```

การแปลงตัวเลขเป็นข้อความ

สามารถทำได้ 2 วิธี คือใช้ฟังก์ชัน String() หรือใช้ method toString() ดังตัวอย่างต่อไปนี้

JavaScript Code

```
let n = 55;  
let a = String(n);  
let b = n.toString();  
let c = (n * 2).toString();
```

การแปลงวันที่เป็นตัวเลข

สามารถใช้ฟังก์ชัน Number() แปลงได้ ดังตัวอย่างต่อไปนี้

JavaScript Code

```
let d = new Date();  
let n = Number(d) //Ex. 1680776128140
```

การแปลงวันที่เป็นข้อความ

สามารถทำได้ 2 วิธี คือใช้ฟังก์ชัน String() หรือใช้ method toString() ดังตัวอย่างต่อไปนี้

JavaScript Code

```
let d = new Date();  
let x = String(d);  
let y = d.toString(); //Ex. "Thu Apr 06 2023 17:24:40 GMT+0700 (Indochina Time)"
```

การแปลงข้อมูล Boolean เป็นตัวเลข

สามารถใช้ฟังก์ชัน Number() ในการแปลง โดยจะได้ค่า เป็น 0 หรือ 1 ดังตัวอย่างนี้

JavaScript Code
<pre>let a = Number(true); // a = 1 let b = Number(false); // b = 0</pre>

การแปลงข้อมูล Boolean เป็นข้อความ

สามารถทำได้ 2 วิธี คือใช้ฟังก์ชัน String() หรือใช้ method toString() ดังตัวอย่างต่อไปนี้

JavaScript Code	Output
<pre>let a = true; let b = false; document.write("String(a) = " + String(a) + "
"); document.write("String(b) = " + String(b) + "
"); document.write("a.toString() = " + a.toString() + "
"); document.write("b.toString() = " + b.toString() + "
");</pre>	<pre>String(a) = true String(b) = false a.toString() = true b.toString() = false</pre>

ตัวดำเนินการ (Operators)

ตัวดำเนินการ (Operator) คือสัญลักษณ์ของภาษา JavaScript ที่ใช้ในคำนวณ เปรียบเทียบ หรือดำเนินการกับข้อมูลเพื่อให้ได้ผลลัพธ์ใหม่ ตัวดำเนินการจะใช้งานกับตัวแปร ซึ่งอาจมีตั้งแต่หนึ่งตัวหรือหลายตัวก็ได้ ในภาษา JavaScript นั้นมีตัวดำเนินการอยู่หลายประเภท และแต่ละประเภทมีหน้าที่การทำงานที่แตกต่างกัน ดังต่อไปนี้

1. ตัวดำเนินการกำหนดค่า

ตัวดำเนินการกำหนดค่า (Assignment operator) คือตัวดำเนินการที่ใช้สำหรับกำหนดหรืออัปเดตค่าให้กับตัวแปรหรือค่าคงที่ โดยจะใช้เครื่องหมาย = ดังตัวอย่างต่อไปนี้

JavaScript Code

```
let num = 0;  
let weight = 52.4;  
let name = "BOY";  
let isban = false;  
let x, y, z;  
  
x = 5;  
y = 10;  
z = 20;
```

2. ตัวดำเนินการทางคณิตศาสตร์

ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic operators) คือ ตัวดำเนินการที่ใช้ในการคำนวณข้อมูลที่เป็นตัวเลข เช่น การบวก ลบ คูณ หาร

ตัวดำเนินการ	หน้าที่	ตัวอย่างการใช้งาน
+	การบวก	$a + b$
-	การลบ	$a - b$
*	การคูณ	$a * b$
/	การหาร	a / b
%	การหารเอาเศษ	$a \% b$
**	การยกกำลัง	$a ** b$

ตัวอย่างการใช้งานตัวดำเนินการในทางคณิตศาสตร์

JavaScript Code	Output
<pre>let a = 5, b = 3; document.write("a = 5, b = 3"); document.write("
a + b = " + (a + b)); document.write("
a - b = " + (a - b)); document.write("
a * b = " + (a * b)); document.write("
a / b = " + (a / b)); document.write("
a % b = " + (a % b)); document.write("
a ** b = " + (a ** b));</pre>	<pre>a = 5, b = 3 a + b = 8 a - b = 2 a * b = 15 a / b = 1.6666666666666667 a % b = 2 a ** b = 125</pre>

3. ตัวดำเนินการทางตรรกศาสตร์

ตัวดำเนินการทางตรรกศาสตร์ (Logical operators) ใช้ดำเนินการทางตรรกศาสตร์ แล้วให้ผลลัพธ์ออกมาเป็นค่า true (จริง) หรือ false (เท็จ)

ตัวดำเนินการ	หน้าที่	ตัวอย่างการใช้งาน
&&	AND	a && b
	OR	a b
!	NOT	!a

จากตัวอย่างการใช้งาน

a && b จะได้ค่าเป็น true ถ้า a และ b มีค่าเป็น true ไม่เช่นนั้นจะได้ false

a || b จะได้ค่าเป็น true ถ้า a หรือ b มีค่าเป็น true ไม่เช่นนั้นจะได้ false

!a จะได้ค่าตรงข้ามกับ a เช่น ถ้า a มีค่าเป็น true จะได้ค่าเป็น false แต่ถ้า a มีค่าเป็น false จะได้ค่า true

4. ตัวดำเนินการเปรียบเทียบค่า

ตัวดำเนินการเปรียบเทียบ (Comparison operators) คือตัวดำเนินการที่ใช้สำหรับเปรียบเทียบระหว่างค่าสองค่า โดยจะให้ผลลัพธ์ออกมาเป็นค่า true (จริง) หรือ false (เท็จ)

ตัวดำเนินการ	หน้าที่	ตัวอย่างการใช้งาน
==	เท่ากับ	a == b
!=	ไม่เท่ากับ	a != b
<	น้อยกว่า	a < b
>	มากกว่า	a > b
<=	น้อยกว่าหรือเท่ากับ	a <= b
>=	มากกว่าหรือเท่ากับ	a >= b
===	เท่ากันทั้งค่าและประเภทข้อมูล	a === b
!==	ไม่เท่ากันทั้งค่าและประเภทข้อมูล	a !== b
?	ตรวจสอบเงื่อนไขแบบสั้น	a == b ? true : false;

5. ตัวดำเนินการกับข้อความ

ตัวดำเนินการที่ใช้กับข้อความจะมีอยู่ตัวเดียวคือ ตัวดำเนินการ + ใช้สำหรับการรวมข้อความเข้าด้วยกัน ดังตัวอย่าง

JavaScript Code	Output
<pre>let str1 = "ABC", str2 = "XYZ"; let str3 = str1 + str2; document.write("str1 = " + str1); document.write("
str2 = " + str2); document.write("
str3 = " + str3);</pre>	<pre>str1 = ABC str2 = XYZ str3 = ABCXYZ</pre>

6. ตัวดำเนินการกำหนดค่าแบบรวม

ตัวดำเนินการกำหนดค่าแบบรวม (Compound assignment operators) คือการใช้งานตัวดำเนินการกำหนดค่า = ร่วมกับตัวดำเนินการประเภทอื่น เช่น $a += b$ จะมีค่าเท่ากับ $a = a + b$ ตัวดำเนินการประเภทยังมีอยู่หลายตัว ดังนี้

ตัวดำเนินการ	ตัวอย่างการใช้งาน	มีค่าเท่ากับ
+=	$a += b$	$a = a + b$
-=	$a -= b$	$a = a - b$
*=	$a *= b$	$a = a * b$
/=	$a /= b$	$a = a / b$
%=	$a \% = b$	$a = a \% b$
**=	$a ** = b$	$a = a ** b$

7. ตัวดำเนินการเพิ่มค่าและลดค่า

ตัวดำเนินการประเภทนี้จะใช้กับตัวแปรที่เป็นตัวเลข มีอยู่ 2 ตัว คือ ++ สำหรับเพิ่มค่า และ -- สำหรับลดค่า ดังตัวอย่างต่อไปนี้

JavaScript Code	Output
<pre>let n = 5, m = 10; document.write("n = " + n + ", m = " + m); n++; m--; document.write("
n = " + n); document.write("
m = " + m);</pre>	<pre>n = 5, m = 10 n = 6 m = 9</pre>

การใช้ตัวดำเนินการเพิ่มค่า ลดค่า สามารถจะใส่ไว้ข้างหน้าหรือข้างหลังก็ได้ หากใส่ไว้ด้านหน้าจะเป็นการเพิ่มหรือลดค่าก่อนที่จะใช้ค่าตัวแปร แต่ถ้าใส่ไว้ข้างหลังจะเป็นการใช้ค่าตัวแปรก่อนที่จะเพิ่มหรือลดค่า ดังตัวอย่างต่อไปนี้

JavaScript Code	Output
<pre>let a = 5, b = 10, c = 15, d = 20; document.write("ค่าเริ่มต้น a = " + a + ", b = " + b); document.write(", c = " + c + ", d = " + d); document.write("

ค่าของ ++a = " + ++a); document.write("
ค่าของ b++ = " + b++); document.write("
ค่าของ --c = " + --c); document.write("
ค่าของ d-- = " + d--); document.write("

ค่าปัจจุบัน a = " + a + ", b = " + b); document.write(", c = " + c + ", d = " + d);</pre>	<pre>ค่าเริ่มต้น a = 5, b = 10, c = 15, d = 20 ค่าของ ++a = 6 ค่าของ b++ = 10 ค่าของ --c = 14 ค่าของ d-- = 20 ค่าปัจจุบัน a = 6, b = 11, c = 14, d = 19</pre>

คำสั่ง if

เป็นคำสั่งตรวจสอบเงื่อนไข เพื่อบริการการทำงานของโปรแกรมให้ทำงานตามเงื่อนไขที่กำหนด

รูปแบบของคำสั่ง if

```
if (condition) {  
    // block of code to be executed if the condition is true  
}
```

ตัวอย่างการใช้คำสั่ง if

JavaScript Code	Output
<pre>let total = 45; if (total < 50) { document.write("total น้อยกว่า 50"); } let name = "สายัน"; if (name.length > 20) { document.write("ชื่อยาวเกินไป"); }</pre>	<p>total น้อยกว่า 50</p>

จากตัวอย่างโปรแกรมจะแสดงคำว่า "total น้อยกว่า 50" เพราะตัวแปร total มีค่า 45 และไม่แสดงคำว่า "ชื่อยาวเกินไป" เพราะตัวแปร name มีความยาวไม่เกิน 20 ตัวอักษร

คำสั่ง if else

คำสั่ง if โปรแกรมจะทำงานเมื่อเงื่อนไขเป็นจริง และถ้าเงื่อนไขไม่เป็นจริงโปรแกรมจะข้ามการทำงานในบล็อกนั้นไป เราสามารถกำหนดบล็อกของคำสั่ง else เพื่อให้โปรแกรมทำงานในกรณีที่เงื่อนไขของคำสั่ง if ไม่เป็นจริงได้ คำสั่ง else นั้นจะต้องใช้ร่วมกับคำสั่ง if เสมอ

รูปแบบของคำสั่ง if else

```
if (condition) {  
    // block of code to be executed if the condition is true  
} else {  
    // block of code to be executed if the condition is false  
}
```

ตัวอย่างการใช้คำสั่ง if else

JavaScript Code	Output
<pre>let total = 55; if (total < 50) { document.write("คุณสอบไม่ผ่าน"); } else { document.write("ยินดีด้วย คุณสอบผ่าน"); }</pre>	ยินดีด้วย คุณสอบผ่าน

จากตัวอย่างนี้ โปรแกรมจะแสดงคำว่า “ยินดีด้วย คุณสอบผ่าน” เพราะ total ไม่น้อยกว่า 50

คำสั่ง else if

กรณีที่ต้องการให้โปรแกรมเลือกการทำงานได้มากกว่า 2 ทางเลือก เราสามารถใช้คำสั่ง else if เพื่อเพิ่มเงื่อนไขได้

รูปแบบคำสั่ง else if

```
if (condition1) {  
    // block of code to be executed if condition1 is true  
} else if (condition2) {  
    // block of code to be executed if the condition1 is false and condition2 is true  
} else {  
    // block of code to be executed if the condition1 is false and condition2 is false  
}
```

ตัวอย่างการใช้คำสั่ง else if

JavaScript Code	Output
<pre>let n = -5; if (n > 0) { document.write("ตัวแปร n มีค่าเป็นบวก"); } else if (n < 0) { document.write("ตัวแปร n มีค่าติดลบ"); } else { document.write("ตัวแปร n มีค่าเป็น 0"); }</pre>	ตัวแปร n มีค่าติดลบ

คำสั่ง switch case

คำสั่ง switch case เป็นคำสั่งควบคุมการทำงานที่คล้ายกับคำสั่ง if แต่จะใช้สำหรับเปรียบเทียบโดยตรงกับค่าที่กำหนดเท่านั้น ในขณะที่คำสั่ง if สามารถสร้างเงื่อนไขจากตัวดำเนินการต่างๆ ได้

รูปแบบของคำสั่ง switch case

```
switch (input) {  
    case value1:  
        // code block  
        break;  
    case value2:  
        // code block  
        break;  
    default:  
        // code block  
}
```

การทำงานของคำสั่ง switch case โปรแกรมจะตรวจสอบค่าของ input ที่กำหนด หากค่าที่กำหนดตรงกับ case ไหน ก็จะเริ่มทำงานตามคำสั่งใน case นั้น ไปจนเจอคำสั่ง break; หากไม่เจอคำสั่ง break; โปรแกรมจะทำงานคำสั่งที่เหลือทั้งหมด

ตัวอย่างการใช้งานคำสั่ง switch case

JavaScript Code	Output
<pre>let sex = "M"; switch (sex) { case "M": document.write("เพศชาย"); break; case "F": document.write("เพศหญิง"); break; default: document.write("เพศทางเลือก"); }</pre>	เพศชาย

จากตัวอย่างนี้ หากไม่ใส่ break; ในแต่ละ case จะได้ผลลัพธ์เหมือนตัวอย่างนี้

JavaScript Code	Output
<pre>let sex = "M"; switch (sex) { case "M": document.write("เพศชาย"); case "F": document.write("เพศหญิง"); default: document.write("เพศทางเลือก"); }</pre>	เพศชายเพศหญิงเพศทางเลือก

การกำหนดเงื่อนไขแบบ OR

คำสั่ง switch case จะตรวจสอบค่าตัวแปรที่กำหนดว่าตรงกับแต่ละ case หรือไม่ ไม่สามารถกำหนดเงื่อนไขเหมือนคำสั่ง if ได้ แต่ถ้าช่วงข้อมูลไม่มากนัก และต้องการผลลัพธ์ที่เหมือนกัน สามารถกำหนดให้โปรแกรมทำหลายๆ case ได้ ดังตัวอย่างนี้

JavaScript Code	Output
<pre>let day = "Wen"; switch (day) { case "Mon": case "Tue": case "Wen": case "Thu": case "Fri": document.write("วันทำงาน"); break; case "Sat": case "Sun": document.write("วันหยุด"); break; default: document.write("ชื่อวันไม่ถูกต้อง"); }</pre>	วันทำงาน

คำสั่ง while loop และ do while loop

คำสั่ง while loop และ do while loop เป็นคำสั่งให้โปรแกรมวนทำงานใน Loop ซ้ำ จนกว่าเงื่อนไขจะเป็นเท็จ โดยที่คำสั่ง while loop โปรแกรมจะตรวจสอบเงื่อนไขก่อน ถ้าเป็นจริงก็จะทำตามคำสั่งใน Loop แต่คำสั่ง do while จะทำตามคำสั่งใน Loop ก่อน แล้วถึงจะตรวจเงื่อนไขถ้าเงื่อนไขเป็นเท็จ ก็จะออกจาก Loop

รูปแบบของคำสั่ง while loop

```
while (condition) {
    // code block to be executed
}
```

ตัวอย่างการใช้งานคำสั่ง while loop

JavaScript Code	Output
<pre>let n = 1, sum = 0; document.write("while loop"); while (n < 10) { sum += n; document.write("
วน Loop รอบที่ " + n + ", sum = " + sum); n++; }</pre>	<pre>while loop วน Loop รอบที่ 1, sum = 1 วน Loop รอบที่ 2, sum = 3 วน Loop รอบที่ 3, sum = 6 วน Loop รอบที่ 4, sum = 10 วน Loop รอบที่ 5, sum = 15 วน Loop รอบที่ 6, sum = 21 วน Loop รอบที่ 7, sum = 28 วน Loop รอบที่ 8, sum = 36 วน Loop รอบที่ 9, sum = 45</pre>

รูปแบบของคำสั่ง do while loop

```
do {  
    // code block to be executed  
} while (condition);
```

ตัวอย่างการใช้งานคำสั่ง do while loop

JavaScript Code	Output
<pre>let n = 1, sum = 0; document.write("do while loop"); do { sum += n; document.write("
วน Loop รอบที่ " + n + ", sum = " + sum); n++; } while (n < 10);</pre>	<pre>do while loop วน Loop รอบที่ 1, sum = 1 วน Loop รอบที่ 2, sum = 3 วน Loop รอบที่ 3, sum = 6 วน Loop รอบที่ 4, sum = 10 วน Loop รอบที่ 5, sum = 15 วน Loop รอบที่ 6, sum = 21 วน Loop รอบที่ 7, sum = 28 วน Loop รอบที่ 8, sum = 36 วน Loop รอบที่ 9, sum = 45</pre>

คำสั่ง do while loop จะมีการทำงานในบล็อกก่อน 1 รอบ แล้วจึงตรวจสอบเงื่อนไข ต่างจากคำสั่ง while loop ที่จะตรวจสอบเงื่อนไขก่อน ดังนั้นคำสั่ง while loop อาจจะไม่ทำคำสั่งในบล็อกเลย ถ้าเงื่อนไขไม่เป็นจริง

คำสั่ง for loop

คำสั่ง for loop เป็นคำสั่งควบคุมการทำงานแบบวนซ้ำที่ใช้สำหรับควบคุมเพื่อให้โปรแกรมทำงานบางอย่างซ้ำๆ ในขณะที่เงื่อนไขเป็นจริง โดยทั่วไปแล้วเรามักใช้คำสั่ง for loop ในกรณีลูปที่จำนวนการวนรอบที่แน่นอน

รูปแบบของคำสั่ง for loop

```
for (initialize; condition; changes) {  
    // code block to be executed  
}
```

โดยที่ initialize คือการประกาศตัวแปรและค่าเริ่มต้นสำหรับใช้ภายใน Loop

condition คือเงื่อนไขในการวน Loop

changes คือการเปลี่ยนแปลงค่าตัวแปรที่ประกาศ

ตัวอย่างการใช้งานคำสั่ง for loop

JavaScript Code	Output
<pre>let rand; let min = 100, max = 0; document.write("for loop"); for (i = 1; i <= 10; i++) { rand = Math.floor(Math.random() * 100) + 1; if (rand < min) min = rand; if (rand > max) max = rand; document.write("
" + rand); } document.write("<hr>Min = " + min + ", Max = " + max);</pre>	<pre>for loop 55 33 48 57 93 100 74 93 6 38 Min = 6, Max = 100</pre>

ตัวอย่างนี้เป็นการใช้คำสั่ง for loop วน loop 10 รอบ โดยประกาศตัวแปร i มีค่าเริ่มต้นเป็น 1 และเพิ่มค่า i ขึ้น รอบละ 1 ในแต่ละรอบจะสุ่มตัวเลขขึ้นมา 1 ตัว มีค่าอยู่ระหว่าง 1 - 100 แสดงค่าที่สุ่มออกมาทางหน้าจอ พร้อมทั้งหาค่าต่ำสุด และค่าสูงสุดของตัวเลขที่สุ่มได้ แสดงออกมาทางหน้าจอ หลังจากวน loop ครบแล้ว

การใช้คำสั่ง for loop ในส่วนของการกำหนดตัวแปร และส่วนของการเปลี่ยนแปลงค่า สามารถประกาศตัวแปรได้มากกว่า 1 ตัวแปร และเปลี่ยนแปลงค่าที่ละหลายค่าได้ โดยใช้เครื่องหมาย comma (,) คั่นแต่ละตัวแปร ดังตัวอย่างต่อไปนี้

JavaScript Code	Output
<pre>document.write("for loop"); for (i = 1, j = 10; i < j; i++, j--) { document.write("
i = " + i + ", j = " + j); } document.write("
Loop ended");</pre>	<pre>for loop i = 1, j = 10 i = 2, j = 9 i = 3, j = 8 i = 4, j = 7 i = 5, j = 6 Loop ended</pre>

การใช้คำสั่ง for ซ้อน for

เราสามารถเขียนคำสั่ง for ซ้อนอยู่ในคำสั่ง for อีกตัวได้ เรียกว่า for ซ้อน for เช่นเดียวกันคำสั่ง while loop, do while loop ก็สามารถทำได้เช่นกัน ดังตัวอย่างต่อไปนี้

JavaScript Code	Output
<pre>for (a = 1; a <= 6; a++) { for (b = 1; b <= a; b++) { document.write("O"); } document.write("
"); }</pre>	<pre>O OO OOO OOOO OOOOO OOOOOO</pre>

การใช้คำสั่ง break และ continue

คำสั่ง break และ continue จะใช้กับ for loop, while loop หรือ do while loop โดยที่

- break จะใช้สำหรับออกจาก loop หยุดการวน loop
- continue จะใช้สำหรับให้โปรแกรมกลับไปเริ่มต้น loop ใหม่ โดยไม่ต้องทำคำสั่งที่เหลืออยู่

ตัวอย่างการใช้คำสั่ง break

JavaScript Code	Output
<pre>for (i = 1; i <= 10; i++) { document.write(i + "
"); if (i == 5) { break; } } document.write("Loop ended");</pre>	<pre>1 2 3 4 5 Loop ended</pre>

ตัวอย่างนี้ กำหนดให้วน loop 10 รอบ ใน loop แต่ละรอบ มีการเช็คตัวแปร i ถ้า i เท่ากับ 5 ให้ break ออกจาก loop ทำให้โปรแกรมนี้ วน loop แค่ 5 รอบ

ตัวอย่างการใช้คำสั่ง continue

JavaScript Code	Output
<pre>let i = 0; do { i++; if (i % 2 == 1) { continue; } document.write(i + "
"); } while (i < 10); document.write("Loop ended");</pre>	<pre>2 4 6 8 10 Loop ended</pre>

ตัวอย่างนี้ กำหนดให้วน loop 10 รอบ ในแต่ละรอบ มีการเช็คตัวแปร i ด้วย ถ้า i หาร 2 แล้วเหลือเศษ 1 จะให้โปรแกรมกลับไปเริ่มต้น loop ใหม่ ทำให้โปรแกรมไม่แสดงตัวเลขที่เป็นเลขคี่ออกมา

ฟังก์ชันคืออะไร

ฟังก์ชัน (Function) คือกลุ่มของชุดคำสั่งที่ถูกรวมเข้าด้วยกัน สำหรับการทำงานบางอย่าง ฟังก์ชันสามารถรับพารามิเตอร์เพื่อนำข้อมูลเข้ามาใช้งานและส่งค่ากลับได้ โดยปกติการทำงานบางอย่างที่ต้องใช้งานบ่อย หรือใช้หลายๆ ที่ เราจะแยกคำสั่งเหล่านั้นออกมาเป็น ฟังก์ชันไว้ เพื่อความสะดวกในการเขียนโปรแกรม ทำให้โค้ดสั้นลง ดูง่ายขึ้น

การประกาศฟังก์ชัน

ก่อนที่จะใช้งานฟังก์ชัน มันจะต้องถูกประกาศหรือสร้างขึ้นมาก่อน การประกาศฟังก์ชันจะใช้คำสั่ง `function` ตามด้วยชื่อของฟังก์ชัน `name` การตั้งชื่อของฟังก์ชันนั้นจะเหมือนกับตัวแปร

รูปแบบของการประกาศฟังก์ชัน

```
Function name(parameter1, parameter2, ...) {  
    // code to be executed  
    return value;    //Optional  
}
```

ตัวอย่างการใช้งานฟังก์ชัน

JavaScript Code	Output
<pre>function show(str) { document.write(str); } show("<p>Front-End content:</p>"); show("HTML
"); show("CSS
"); show("JavaScript
");</pre>	Front-End content: HTML CSS JavaScript

ตัวอย่างนี้เป็นการประกาศฟังก์ชัน show() ขึ้นมา โดยมีการส่ง parameter เข้าไป 1 ตัว เพื่อให้ฟังก์ชันนี้แสดงข้อความที่กำหนด ออกมาทางหน้าจอ

การส่งค่ากลับจากฟังก์ชัน

หากต้องการส่งค่ากลับมาจากฟังก์ชัน จะใช้คำสั่ง return ในการส่ง ดังตัวอย่างต่อไปนี้

JavaScript Code	Output
<pre>function random(min, max) { let rand = Math.floor(Math.random() * (max - min)) + min; return rand; } document.write("Random number is: " + random(1,10));</pre>	Random number is: 6

ตัวอย่างนี้ ฟังก์ชัน random() ทำหน้าที่สุ่มตัวเลขที่มีค่าอยู่ในช่วง 1 – 10 แล้วส่งค่ากลับออกมา

Event

หัวข้อนี้จะพูดถึงการใช้ HTML Event เรียกคำสั่ง JavaScript ขึ้นมาทำงาน ตัวอย่างเช่น ให้ตรวจสอบข้อมูลที่ผู้ใช้ป้อนตอน submit form เป็นต้น

HTML Events ทัวไป

Event	Description
onchange	ทำงานเมื่อมีการเปลี่ยนแปลงค่า ข้อมูลของ Element ที่กำหนด
onclick	ทำงานเมื่อมีการคลิกที่ Element ที่กำหนด
onmouseover	ทำงานเมื่อเลื่อนเมาส์มาที่ Element ที่กำหนด
onmouseout	ทำงานเมื่อเลื่อนเมาส์ออกจาก Element ที่กำหนด
onkeydown	ทำงานเมื่อมีการกดแป้นคีย์บอร์ด
onload	ทำงานเมื่อโหลดหน้าเพจเสร็จแล้ว
onsubmit	ทำงานเมื่อมีการ submit form

ตัวอย่างการใช้งาน Event

JavaScript Code	Output
<pre><!DOCTYPE html> <html> <body> <script> function checkForm() { let f = document.getElementById("form1"); let num = f.num.value; if (num == "") { alert("กรุณาป้อนจำนวน"); return false; } if (isNaN(num)) { alert("กรุณาป้อนตัวเลขเท่านั้น"); return false; } alert("คุณป้อนตัวเลข " + num); f.submit(); } </script> <h2>HTML Event</h2> <form id="form1" name="form1" method="post"> <label>กรุณาป้อนจำนวน:</label> <input type="text" id="num" name="num" maxlength="5">

 <input type="button" value="Submit" onclick="checkForm();"> </form> </body> </html></pre>	<p>HTML Event</p> <p>กรุณาป้อนจำนวน: <input type="text"/></p> <p><input type="button" value="Submit"/></p>

ตัวอย่างนี้ กำหนด Event onclick ที่ปุ่ม Submit ให้เรียกใช้ฟังก์ชัน checkForm() ของ JavaScript เพื่อตรวจสอบจำนวนที่ผู้ใช้ป้อน ถ้าผู้ใช้ไม่ป้อน หรือป้อนข้อมูลที่ไม่ใช่ตัวเลข จะมีข้อความแจ้งเตือน และหยุดทำงาน แต่ถ้าผู้ใช้ป้อนตัวเลข จะมีข้อความบอกว่าป้อนตัวเลขอะไร และ submit form

การใช้งาน Array

อาร์เรย์ (Array) คือ ชุดของข้อมูลที่ถูกเรียงต่อกันเป็นลำดับ เหมือนข้อมูลที่อยู่ในตาราง เราสามารถใช้ Array เก็บข้อมูลอื่นได้ทุกประเภท

การประกาศตัวแปร Array

```
let color = []; //ประกาศตัวแปร array เปล่า
```

```
let cars = ["BMW","VOLVO","TESLA","BYD","MG"]; //ประกาศตัวแปร Array พร้อมกำหนดข้อมูลใน Array 5 ตัว
```

การใช้ข้อมูลใน Array

ข้อมูลใน Array จะเก็บเรียงตาม Index และ Index ของ Array เริ่มต้นจาก 0 การใช้ข้อมูลใน Array จะอ้างอิงตาม Index ดังตัวอย่างต่อไปนี้

JavaScript Code	Output
<pre>let cars = ["BMW","VOLVO","TESLA","BYD","MG"]; document.write("ข้อมูลใน Array cars: " + cars); document.write("
cars[0] = " + cars[0]); document.write("
cars[1] = " + cars[1]); document.write("
cars[2] = " + cars[2]); document.write("
cars[3] = " + cars[3]); document.write("
cars[4] = " + cars[4]);</pre>	<pre>ข้อมูลใน Array cars: BMW,VOLVO,TESLA,BYD,MG cars[0] = BMW cars[1] = VOLVO cars[2] = TESLA cars[3] = BYD cars[4] = MG</pre>

การหาขนาดของ Array

หากต้องการทราบว่า Array มีข้อมูลอยู่ที่ตัว สามารถใช้ Property length หาได้ ดังตัวอย่างนี้

JavaScript Code	Output
<pre>let cars = ["BMW", "VOLVO", "TESLA", "BYD", "MG"]; document.write("ข้อมูลใน Array cars: " + cars); document.write("
มีข้อมูลทั้งหมด " + cars.length + " ตัว");</pre>	<p>ข้อมูลใน Array cars: BMW,VOLVO,TESLA,BYD,MG มีข้อมูลทั้งหมด 5 ตัว</p>

การเพิ่มข้อมูลใน Array

การเพิ่มข้อมูลใน Array มี 2 เมธอด ที่ใช้ได้ ดังนี้

- push ใช้สำหรับเพิ่มข้อมูลไปยังตำแหน่งท้ายสุดของ Array
- unshift ใช้สำหรับเพิ่มข้อมูลไปยังตำแหน่งแรกของ Array

ตัวอย่างการเพิ่มข้อมูลใน Array

JavaScript Code	Output
<pre>let cars = ["BMW", "VOLVO", "TESLA", "BYD", "MG"]; document.write("ข้อมูลใน Array cars: " + cars); cars.push("NETA"); document.write("<hr>ข้อมูลใน Array cars: " + cars); cars.unshift("BENZ"); document.write("<hr>ข้อมูลใน Array cars: " + cars);</pre>	<p>ข้อมูลใน Array cars: BMW,VOLVO,TESLA,BYD,MG</p> <hr/> <p>ข้อมูลใน Array cars: BMW,VOLVO,TESLA,BYD,MG,NETA</p> <hr/> <p>ข้อมูลใน Array cars: BENZ,BMW,VOLVO,TESLA,BYD,MG,NETA</p>

การลบข้อมูลออกจาก Array

หากต้องการลบข้อมูลออกจาก Array จะมี 2 เมธอด ที่ทำได้ ดังนี้

- pop ใช้สำหรับลบข้อมูลใน Array ตำแหน่งสุดท้ายออก และส่งค่าที่ลบออกกลับมา
- shift ใช้สำหรับลบข้อมูลใน Array ตำแหน่งแรกออก และส่งค่าที่ลบออกกลับมา

ตัวอย่างการลบข้อมูลใน Array

JavaScript Code	Output
<pre>let cars = ["BMW", "VOLVO", "TESLA", "BYD", "MG"]; document.write("ข้อมูลใน Array: " + cars); let item = cars.pop(); document.write("<hr>เมธอด pop ลบ " + item + " ออก"); document.write("
ข้อมูลใน Array: " + cars); item = cars.shift(); document.write("<hr>เมธอด shift ลบ " + item + " ออก"); document.write("
ข้อมูลใน Array: " + cars);</pre>	<p>ข้อมูลใน Array: BMW,VOLVO,TESLA,BYD,MG</p> <hr/> <p>เมธอด pop ลบ MG ออก ข้อมูลใน Array: BMW,VOLVO,TESLA,BYD</p> <hr/> <p>เมธอด shift ลบ BMW ออก ข้อมูลใน Array: VOLVO,TESLA,BYD</p>

การใช้ for loop กับ Array

หากข้อมูลใน Array มีจำนวนมาก และต้องการใช้งานทุกตัว การเขียนคำสั่งเข้าถึงทีละตัวจะไม่สะดวก ใช้คำสั่ง for loop แทนจะดีกว่า ดังตัวอย่างต่อไปนี้

JavaScript Code	Output
<pre>let number = [45, 32, 16, 25, 50]; for (i = 0; i < number.length; i++) { document.write(number[i] + "
"); }</pre>	45 32 16 25 50

การใช้ for in, for of กับ Array

คำสั่ง for in, for of เป็นคำสั่งที่วน Loop ตามจำนวนข้อมูลใน Array ที่สามารถใช้ได้เหมือน for loop แต่การใช้งานจะง่ายกว่า ดังตัวอย่างต่อไปนี้

JavaScript Code	Output
<pre>let number = [45, 32, 16, 25, 50]; for (let x in number) { document.write(number[x] + "
"); } document.write("<hr>"); for (let x of number) { document.write(x + "
"); }</pre>	45 32 16 25 50 <hr/> 45 32 16 25 50

คำสั่ง for of จะวนรอบสมาชิกทั้งหมดใน Array และนำค่าของการวนแต่ละรอบกำหนดไว้ในตัวแปร x

การรวม Array

สามารถใช้ Method concat() ในการรวม (Merge) Array ได้ ดังตัวอย่างต่อไปนี้

JavaScript Code	Output
<pre>let fe = ["HTML", "CSS", "JavaScript"]; let be = ["PHP", "NodeJS"]; let fsp = fe.concat(be); document.write(fsp);</pre>	HTML,CSS,JavaScript,PHP,NodeJS

การแปลง Array เป็น String

เราสามารถแปลง Array เป็น String ได้โดยใช้ method toString() โดย JavaScript จะใช้ comma (,) กั้นข้อมูลใน Array แต่ละตัว ดังตัวอย่างนี้

JavaScript Code	Output
<pre>let arr = [5,4,8,3,2,9]; let str = arr.toString(); document.write(str + "
"); document.write("typeof str is " + typeof str);</pre>	5,4,8,3,2,9 typeof str is string

การจัดเรียงข้อมูลใน Array

การจัดเรียงข้อมูล String ใน Array จะใช้ method sort() เพื่อจัดเรียงจากน้อยไปมาก ดังตัวอย่างนี้

JavaScript Code	Output
<pre>let arr = ["Dog", "Bee", "Pig", "Ant", "Cat"]; arr.sort(); document.write(arr);</pre>	Ant,Bee,Cat,Dog,Pig

การจัดเรียงข้อมูลจากมากไปน้อย

การจัดเรียงข้อมูล String จากมากไปน้อย จะใช้ method sort() และ reverse() ร่วมกัน ดังตัวอย่างนี้

JavaScript Code	Output
<pre>let arr = ["Dog", "Bee", "Pig", "Ant", "Cat"]; arr.sort(); arr.reverse(); document.write(arr);</pre>	Pig,Dog,Cat,Bee,Ant

การจัดเรียงข้อมูลตัวเลข

โดยปกติ Method sort() จะจัดเรียงข้อมูลแบบ String โดยจะเทียบตัวอักษรทีละตัว ไม่ได้จัดเรียงแบบตัวเลข ดังตัวอย่างนี้

JavaScript Code	Output
<pre>let arr = [4,25,2,30,1,20,45,3,5,15]; arr.sort(); document.write(arr);</pre>	1,15,2,20,25,3,30,4,45,5

หากต้องการจัดเรียงข้อมูลแบบตัวเลขจะต้องใช้ compare function ช่วย ดังตัวอย่างต่อไปนี้

JavaScript Code	Output
<pre>let arr = [4,25,2,30,1,20,45,3,5,15]; arr.sort(function(a, b){return a - b}); document.write(arr);</pre>	1,2,3,4,5,15,20,25,30,45

ตัวอย่างการจัดเรียงข้อมูลแบบตัวเลข จากมากไปน้อย

JavaScript Code	Output
<pre>let arr = [4,25,2,30,1,20,45,3,5,15]; arr.sort(function(a, b){return b - a}); document.write(arr);</pre>	45,30,25,20,15,5,4,3,2,1

การหาค่าต่ำสุด หรือมากที่สุด ใน Array

หาต้องการหาค่าต่ำสุด หรือมากที่สุด ใน Array จะใช้ method Math.min.apply() หรือ Math.max.apply() หาได้ ดังตัวอย่างต่อไปนี้

JavaScript Code	Output
<pre>let arr = [20,5,31,16,65,86,45]; let min = Math.min.apply(null,arr); let max = Math.max.apply(null,arr); document.write(arr); document.write("
Min: " + min + ", Max: " + max);</pre>	20,5,31,16,65,86,45 Min: 5, Max: 86

Regular Expression

Regular Expressions คือ รูปแบบหรือกลุ่มคำ (pattern) ที่เรากำหนดขึ้นเพื่อเอาไว้ค้นหาข้อความหรือตัวอักษรต่างๆ เพื่อเช็คตรงตามเงื่อนไขที่กำหนดหรือไม่ เราสามารถใช้ Regular Expressions ในการตรวจสอบข้อมูลในฟอร์มที่ถูกคำบั่น หากมีอะไรไม่ถูกต้อง ก็แจ้งเตือน ก่อนที่จะ submit form ดังตัวอย่างนี้

JavaScript Code	Output
<pre><!DOCTYPE html> <html> <body> <script> function checkForm() { let f = document.getElementById("form1"); let name = f.name.value; let pattern = /^[\\w-]+\$/; if (!pattern.test(name)) { alert("ชื่อไม่ถูกต้อง"); return false; } alert("สวัสดี " + name); f.submit(); } </script> <h3>Regular Expressions</h3> <form id="form1" name="form1" method="post"> <label>คุณชื่ออะไร:</label> <input type="text" id="name" name="name" maxlength="20">

 <input type="button" value="Submit" onclick="checkForm();"> </form> </body> </html></pre>	<p>Regular Expressions</p> <p>คุณชื่ออะไร: <input type="text"/></p> <p><input type="button" value="Submit"/></p>

จากตัวอย่าง ในฟังก์ชัน checkForm() มีการตรวจสอบชื่อที่ผู้ใช้ป้อนโดยกำหนด pattern ให้ป้อนได้เฉพาะตัวอักษร ตัวเลข หรือ _ ถ้าผิดไปจาก pattern นี้จะ มีการแจ้งเตือนให้ลูกค้าทราบว่า “ชื่อไม่ถูกต้อง” แต่ถ้าชื่อที่ป้อนถูกต้องตาม pattern จะแสดงข้อความว่า “สวัสดิ์ “ + ชื่อที่ผู้ใช้ป้อน ขึ้นมา พร้อมกับ submit form

การกำหนด pattern

Pattern จะประกอบไปด้วยสัญลักษณ์หรือเครื่องหมายต่างๆ ดังนี้

สัญลักษณ์	ความหมาย
.	แทนตัวอักษรอะไรก็ได้
^	ต้องขึ้นต้นด้วย
\$	ต้องลงท้ายด้วย
[]	เรียกว่า bracket expression หมายถึง กลุ่มของตัวอักษรในที่นี้เท่านั้นที่ต้องการ เช่น <ul style="list-style-type: none"> - [abc] หมายถึง ตัวอักษร a, b หรือ c - [a-z] หมายถึง ตัวอักษร a ถึง z - [abcx-z] หมายถึง ตัวอักษร a, b, c, x, y, z - [a-zA-Z] หมายถึง ตัวอักษร a ถึง z และ A ถึง Z - [0-9] หมายถึง ตัวเลข 0 ถึง 9 - [+_] หมายถึง เครื่องหมาย + _ หรือ . - [ก-๙] หมายถึง พยัญชนะ สระ และตัวเลขไทยทั้งหมด
[^]	ตรงข้ามกับ [] คือ ไม่อยู่ในกลุ่มที่กำหนด เช่น [^abc] หมายถึง ไม่ใช่ตัวอักษร a, b หรือ c
()	จัดให้อยู่ในกลุ่มเดียวกัน
	ใช้แทน หรือ เช่น a b หมายถึง ตัวอักษร a หรือ b
\	ใช้กำหนดให้สัญลักษณ์ กลายเป็นตัวอักษรปกติ เช่น * จะหมายถึง ตัวดอกจัน 1 ตัว ไม่ใช่สัญลักษณ์ใน pattern
\w	แทนตัวอักษร, ตัวเลข หรือ _ เทียบได้กับ [a-zA-Z0-9_]

\W	ตรงข้ามกับ \w คือ ไม่ใช่ตัวอักษร, ตัวเลข และ _ เทียบได้กับ [^a-zA-Z0-9_]
\a	แทนตัวอักษร เทียบได้กับ [a-zA-Z]
\s	แทนช่องว่าง หรือ tab
\S	แทนข้อมูล ที่ไม่ใช่ ช่องทาง หรือ tab
\d	ตัวเลข มีค่าเท่ากับ [0-9]
\D	แทนข้อมูลที่ไม่ใช่ตัวเลข เทียบได้กับ [^0-9]
*	แทนอะไรก็ได้ โดยที่จะมี หรือไม่มี ก็ได้ และจะมีกี่ตัวก็ได้
?	แทนอะไรก็ได้ 0 หรือ 1 ตัว
+	แทนอะไรก็ได้ อย่างน้อย 1 ตัว
{min,max}	กำหนดจำนวนตั้งแต่ min ถึง max เช่น <ul style="list-style-type: none"> - {6} คือ ต้องมี 6 ตัว - {5,10} คือ ต้องมี 5 ถึง 10 ตัว - {4,} คือ ต้องมีอย่างน้อย 4 ตัว

ตัวอย่าง pattern

สัญลักษณ์	ความหมาย
^\d+\$	ต้องเป็นตัวเลขเท่านั้น ก้ตัวก็ได้
^[1-9]\d+\$	ต้องเป็นตัวเลขที่ขึ้นต้นด้วยเลข 1-9
^\d{3,6}\$	ต้องเป็นตัวเลข 3 ถึง 6 ตัว
^\w+\$	ต้องเป็นตัวอักษรภาษาอังกฤษ ตัวเลข หรือ _
^\[\w\-\u\]+\u\$	ต้องเป็นตัวอักษรภาษาอังกฤษ ภาษาไทย ตัวเลข หรือ _

<code>^[a-zA-Z0-9ก-๙]+\$</code>	ต้องเป็นตัวอักษรภาษาอังกฤษ ภาษาไทย หรือ ตัวเลข
<code>^(fb\@ gg\@)\d+\$</code>	ต้องเป็นขึ้นต้นด้วย fb@ หรือ gg@ และตามด้วยตัวเลข
<code>^[a-zA-Z0-9ก-๙/().,\-]+\$</code>	ต้องเป็นตัวอักษรภาษาอังกฤษ ภาษาไทย ตัวเลข เครื่องหมาย / () . , - หรือ ช่องว่าง

Exceptions และ Error

Exception คือ รูปแบบของการจัดการข้อผิดพลาด (Error) ที่เกิดขึ้นในโปรแกรม เพื่อไม่ให้โปรแกรมหยุดการทำงาน โดยปกติแล้วเมื่อเกิดข้อผิดพลาดขึ้นในขณะที่โปรแกรมทำงาน โปรแกรมหยุดทำงานทันที เราสามารถป้องกันเหตุการณ์นี้ได้โดยการตรวจจับและจัดการกับข้อผิดพลาดดังกล่าว ซึ่งวิธีการนี้เรียกว่า Exception handling ในภาษา JavaScript เราสามารถตรวจสอบข้อผิดพลาดที่เกิดขึ้นเพื่อจัดการกับมันได้ โดยใช้คำสั่ง try catch ซึ่งมีรูปแบบการใช้งานดังนี้

```
Try {
    // Statements
} catch (error) {
    // Handling errors
}
```

ในบล็อกของคำสั่ง try เป็นส่วนที่เราจะเขียนโค้ดที่อาจทำให้เกิดข้อผิดพลาดได้ และถ้าเกิดข้อผิดพลาดขึ้น โปรแกรมจะข้ามการทำงานไปทำตามคำสั่งบล็อก catch แทน พร้อมส่งตัวแปร error ซึ่งเป็นออบเจกต์ของข้อผิดพลาด ให้เราสามารถตรวจสอบได้ แต่ถ้าไม่เกิดข้อผิดพลาด โปรแกรมจะข้ามการทำงานคำสั่งในบล็อก catch ไป

ตัวอย่างการใช้คำสั่ง try

JavaScript Code	Output
<pre>let total = 0; try { total = sum(5,8,10,16,9); document.write("<p>Total = " + total + "</p>"); } catch (err) { document.write("<p>Error: " + err.message + "</p>"); }</pre>	Error: sum is not defined

ตัวอย่างนี้ไม่ได้มีการประกาศฟังก์ชัน sum มาก่อนเลยทำให้เกิด error ขึ้น

การใช้คำสั่ง finally

การใช้คำสั่ง finally จะใช้ร่วมกับคำสั่ง try catch เพื่อให้โปรแกรมทำงานบางอย่าง เมื่อการจัดการข้อผิดพลาดเสร็จแล้ว เช่น การคืนทรัพยากร การปิดไฟล์ การปิดการเชื่อมต่อฐานข้อมูล เป็นต้น คำสั่ง finally มีรูปแบบการใช้งานดังนี้

```
Try {
    // Statements
} catch (error) {
    // Handling errors
} finally {
    // Statements
}
```

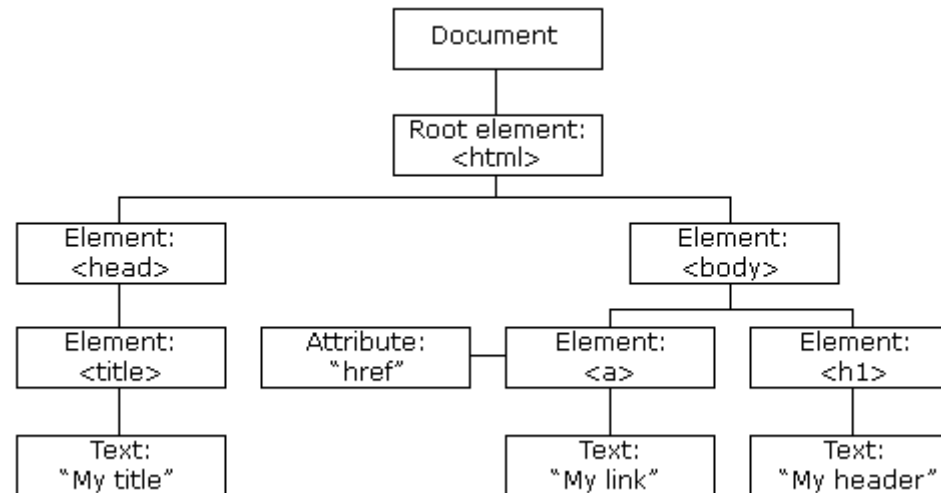
ตัวอย่างการใช้คำสั่ง finally

JavaScript Code	Output
<pre>function sum() { if (arguments.length == 0) { throw new Error("Null parameter."); } let \$sum = 0; for (i = 0; i < arguments.length; i++) { if (typeof arguments[i] != "number") { throw new Error("Invalid input data."); } \$sum += arguments[i]; } return \$sum; } let total = 0; try { //total = sum(); total = sum(5,8,10,16,9); document.write("<p>Total = " + total + "</p>"); } catch (err) { document.write("<p>Error: " + err.message + "</p>"); } finally { document.write("<p>Process done.</p>"); }</pre>	<p>กรณีที่ไม่มี error</p> <p>Total = 48</p> <p>Process done.</p> <p>กรณีที่มี error</p> <p>Error: Null parameter.</p> <p>Process done.</p>

ตัวอย่างนี้มีการประกาศฟังก์ชัน sum() แล้ว ถ้าส่งข้อมูลให้ฟังก์ชันนี้เป็นตัวเลข จะไม่ error จะได้ยอดรวมออกมาตามตัวอย่าง แต่ถ้าไม่ส่งข้อมูลให้ฟังก์ชันหรือไม่ใช่ตัวเลข จะเกิด error ในฟังก์ชัน sum() มีการใช้คำสั่ง throw เพื่อบังคับให้เกิด error ขึ้น เพราะข้อมูลที่รับไม่ถูกต้อง

JavaScript DOM

DOM ย่อมาจาก Document Object Model เป็นการมองส่วนประกอบต่างๆ ของหน้าเว็บให้เป็น object ทำให้สามารถแยกแยะ และเข้าถึงแต่ละ object ได้ง่ายขึ้น เมื่อเว็บเพจถูกโหลดเสร็จแล้ว Browser จะสร้าง Document Object Model ของเพจ หรือ HTML DOM โดยจะมีโครงสร้างเป็นแบบต้นไม้ ดังภาพ



JavaScript สามารถใช้ HTML DOM ในการจัดการเว็บเพจ ได้ เช่น

- สามารถเปลี่ยนคุณสมบัติ หรือสไตล์ ของ Element ต่างๆ
- สามารถเปลี่ยนเพิ่มหรือลบ Element ในเพจได้
- สามารถตอบสนองต่อ event ต่างๆ ในเพจ
- สามารถสร้าง event ใหม่ในเพจได้

การเลือก HTML Element

การเลือก HTML Element ในเพจจะใช้ method ที่สำคัญดังนี้

Method	Description
document.getElementById(id)	ใช้สำหรับเลือก Element ตาม ID
document.getElementsByTagName(name)	ใช้เลือก Element ตามชื่อแท็ก
document.getElementsByClassName(name)	ใช้เลือก Element ตามชื่อคลาส

การเปลี่ยนแปลง HTML Element

เราสามารถที่จะเปลี่ยนแปลงข้อความ คุณสมบัติหรือรูปแบบของ Element ต่างๆ ในเพจได้ ดังนี้

Property	Description
element.innerHTML = new html content	ใช้สำหรับเปลี่ยนข้อความใน Element
element.attribute = new value	ใช้สำหรับเปลี่ยนค่า attribute ของ Element
element.style.property = new style	ใช้สำหรับเปลี่ยนสไตล์ของ Element

ตัวอย่างการเลือกและการเปลี่ยนแปลง HTML Element

JavaScript Code	Output
<pre><!DOCTYPE html> <html lang="en"> <body> <form id="form1" method="post"> <p id="title">JavaScript DOM</p> <label for="fName">fname</label>
 <input type="text" id="fName" maxlength="25">
 <label for="lName">lname</label>
 <input type="text" id="lName" maxlength="25">

 <input type="button" class="btn" value="Send"> <input type="reset" class="btn" value="Reset"> </form> </body> </html> <script> let p = document.getElementById("title"); let fName = document.getElementById("fName"); let lName = document.getElementById("lName"); let label = document.getElementsByTagName("label"); let btn = document.getElementsByClassName("btn"); p.style.color = "blue"; p.style.fontSize = "20pt"; label[0].innerHTML = "ชื่อ:"; label[1].innerHTML = "นามสกุล:"; fName.placeholder = "ป้อนชื่อไม่เกิน 25 ตัว"; lName.placeholder = "ป้อนนามสกุลไม่เกิน 25 ตัว"; btn[0].value = "ส่งข้อมูล"; btn[1].value = "ล้างข้อมูล"; </script> </body> </html></pre>	<p>JavaScript DOM</p> <p>ชื่อ: <input type="text" value="ป้อนชื่อไม่เกิน 25 ตัว"/></p> <p>นามสกุล: <input type="text" value="ป้อนนามสกุลไม่เกิน 25 ตัว"/></p> <p><input type="button" value="ส่งข้อมูล"/> <input type="button" value="ล้างข้อมูล"/></p>

การเพิ่ม หรือลบ Element

Method สำหรับการเพิ่มหรือลบ Element จะมีดังนี้


Method	Description
document.createElement(element)	ใช้สำหรับสร้าง HTML Element
document.removeChild(element)	ใช้สำหรับลบ HTML Element
document.appendChild(element)	ใช้สำหรับเพิ่ม HTML Element
document.replaceChild(new, old)	ใช้สำหรับแทนที่ HTML Element

ตัวอย่างการเพิ่ม/ลบ Element

JavaScript Code	Output
<pre><html> <body> <ul id="menu"> Home Product Service About Us <script> let menu = document.getElementById('menu'); menu.removeChild(menu.lastElementChild); let li = document.createElement('li'); li.textContent = "Report"; menu.appendChild(li); let newNode = document.createTextNode("Member"); menu.children[1].replaceChild(newNode, menu.children[1].childNodes[0]); </script> </body> </html></pre>	<ul style="list-style-type: none">• Home• Member• Service• Report

การเพิ่ม Event

เราสามารถเพิ่ม Event ให้กับ Element ที่ต้องการได้ ดังตัวอย่างต่อไปนี้

JavaScript Code	Output
<pre><!DOCTYPE html> <html> <head> <meta charset="utf-8"> <title>JavaScript Add Event</title> </head> <body> <h3>JavaScript</h3> <p id="p1">Click Me</p> <script> let p1 = document.getElementById('p1'); p1.onclick = function () { alert("สวัสดี User"); } </script> </body> </html></pre>	

JavaScript กับ JSON

การทำงานกับข้อมูล JSON ต้องแปลงให้เป็น JavaScript Object ก่อน โดยใช้ฟังก์ชัน JSON.parse() ดังตัวอย่างต่อไปนี้

JavaScript Code

```
let str = '{"code": "000","message": "success","products": [' +  
  '{"name": "ดินสอ","price": "45","qty": "59","unit": "แพค"},' +  
  '{"name": "ปากกา","price": "15","qty": "160","unit": "ด้าม"},' +  
  '{"name": "ยางลบ","price": "10","qty": "142","unit": "ก้อน"}]]';  
let obj = JSON.parse(str);  
document.write("code: " + obj.code + ", message: " + obj.message + "<hr>");  
for(let p of obj.products) {  
  document.write(p.name + " ราคา " + p.price + " บาท/" + p.unit + "<br>");  
}
```

Output

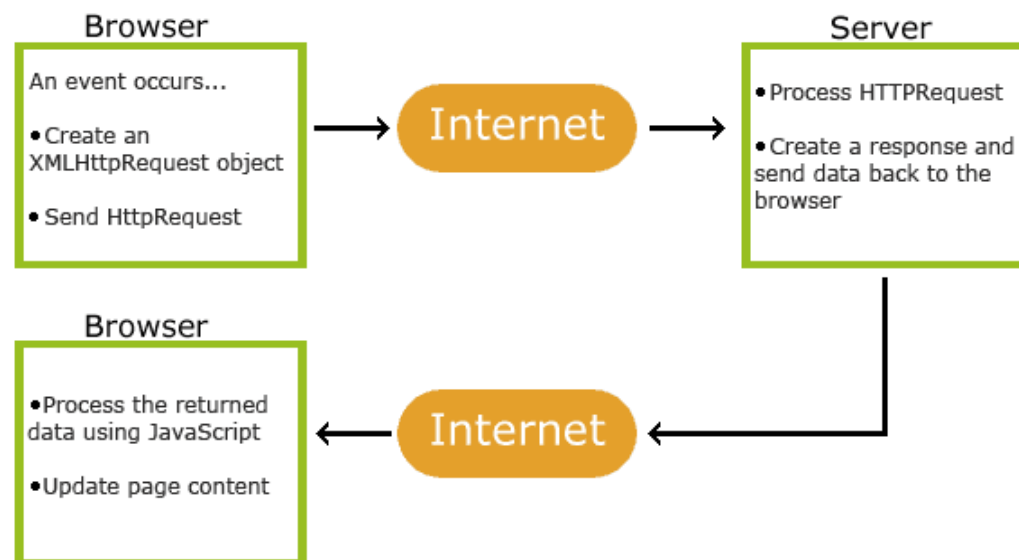
code: 000, message: success

ดินสอ ราคา 45 บาท/แพค
ปากกา ราคา 15 บาท/ด้าม
ยางลบ ราคา 10 บาท/ก้อน

AJAX

AJAX ย่อมาจาก Asynchronous JavaScript and XML คือ การพัฒนาเว็บที่ประมวลผลในเบื้องหลัง เป็นเทคนิคในการทำเว็บแอปพลิเคชันเพื่อให้สามารถโต้ตอบกับผู้ใช้ได้ดีขึ้น ทำให้ความรู้สึกเหมือนการใช้งานเดสก์ท็อปแอปพลิเคชัน

ปกติ JavaScript จะประมวลผลแบบเป็นลำดับ (synchronous) คือ ต้องทำคำสั่งแรกให้เสร็จสิ้นก่อนแล้วถึงจะทำงานในคำสั่งถัดไป แต่กระบวนการทำงานแบบ AJAX เมื่อ Browser ร้องขอข้อมูลไปยัง Server Browser จะไปทำงานคำสั่งถัดไปทันที (asynchronous) โดยไม่รอการตอบกลับจาก Server ทำให้การตอบสนองต่อผู้ใช้งานรวดเร็วขึ้น และเมื่อ Server ประมวลผลเสร็จแล้วถึงจะส่งข้อมูลกลับมาที่ AJAX และให้ AJAX ทำงานกับข้อมูลที่ส่งกลับมาอีกที



XMLHttpRequest Object

เราสามารถใช้งาน XMLHttpRequest Object เพื่อแลกเปลี่ยนข้อมูลกับ Server ที่อยู่เบื้องหลังได้ ทำให้สามารถอัปเดตหน้าเว็บได้โดยไม่ต้องโหลดเว็บเพจซ้ำทั้งหน้า ดังตัวอย่างต่อไปนี้

JavaScript Code	Output
<pre><!DOCTYPE html> <html> <head> <meta charset="utf-8"> </head> <body> <h2>XMLHttpRequest Object</h2> <hr> <div id="demo"> <button type="button" onclick="loadInfo()">Load content</button> </div> <script> function loadInfo() { let xhttp = new XMLHttpRequest(); let demo = document.getElementById("demo"); xhttp.onreadystatechange = function () { if (this.readyState == 4 && this.status == 200) { demo.innerHTML = this.responseText; } }; let url = "/assets/course_fsp.txt"; xhttp.open("GET", url, true); xhttp.send(); } </script> </body> </html></pre>	<h2>XMLHttpRequest Object</h2> <hr/> <div>Load content</div>
	<h2>XMLHttpRequest Object</h2> <hr/> <h3>Full Stack Programmer</h3> <ol style="list-style-type: none">1. Concept of Computer Programming2. Software Development Life cycle3. Software Design4. Data Structure and Algorithm5. HTTP6. Client-Side Programming (Front-End)7. Server-Side Programming (Back-End)8. Database9. API10. Linux Server11. Security

XMLHttpRequest Object Methods

Method	Description
new XMLHttpRequest()	สร้าง XMLHttpRequest Object
abort()	ยกเลิกการ Request ปัจจุบัน
getAllResponseHeaders()	คืนค่า Header ที่ส่งมา
getResponseHeader(header)	คืนค่า Header ที่กำหนดพารามิเตอร์ header ค่า header ที่ต้องการอ่าน
open(method,url,async,user,psw)	เปิดการติดต่อกับ Server โดยมี parameter ดังนี้ <ul style="list-style-type: none">- method คือวิธีการรับส่งข้อมูล จะมี 2 ตัว คือ GET และ POST- url: ระบุ url ของไฟล์หรือเว็บ- async: วิธีการประมวลผล true (asynchronous) หรือ false (synchronous)- user: username (ไม่บังคับ)- psw: password (ไม่บังคับ)
send()	ส่งข้อมูลไปยังปลายทาง สำหรับ method GET
send(string)	ส่งข้อมูลไปยังปลายทาง สำหรับ method POST
setRequestHeader(header, value)	กำหนด header ไปยังปลายทาง

XMLHttpRequest Object Properties

Method	Description
onreadystatechange	กำหนดฟังก์ชันที่จะให้ทำงานเมื่อสถานะ readyState มีการเปลี่ยนแปลง
readyState	ตัวเลขแสดงสถานการณ์ทำงาน โดยที่ 0 คือ ยังไม่เริ่มต้น 1 คือ เชื่อมต่อเซิร์ฟเวอร์แล้ว 2 คือ ส่งข้อมูลเรียบร้อยแล้ว 3 คือ กำลังประมวลผล 4 คือ ประมวลผลเสร็จสิ้นและพร้อมตอบสนองแล้ว
responseText	ข้อมูลที่ส่งมาจาก Server ในรูปข้อความ
responseXML	ข้อมูลที่ส่งมาจาก Server ในรูปเอกสาร XML
status	เลขสถานะการทำงาน 200: "OK" 403: "Forbidden" 404: "Not Found"
statusText	ข้อความที่ส่งมาจาก Server เพื่อแสดงสถานะการทำงาน

ตัวอย่างการใช้ AJAX ดึงข้อมูลจากเซิร์ฟเวอร์

JavaScript Code	Output
<pre><!DOCTYPE html> <html> <head> <meta charset="utf-8"> </head> <body> <h2>AJAX Request</h2> <hr> <div id="demo"> <button type="button" onclick="request()">Send Request</button> </div> <script> function request() { let xhttp = new XMLHttpRequest(); let demo = document.getElementById("demo"); xhttp.onreadystatechange = function () { if (this.readyState == 4 && this.status == 200) { demo.innerHTML = "Server response: " + this.responseText; } }; let url = "https://test.winwin.co.th/api/test.php"; xhttp.open("GET", url, true); xhttp.setRequestHeader('Content-type', 'application/x-www-form-urlencoded'); xhttp.send(); } </script> </body> </html></pre>	<div><h2>AJAX Request</h2><hr/><div>Send Request</div></div> <div><h2>AJAX Request</h2><hr/><p>Server response: Hello Developer</p></div>