

# หลักการเขียนโปรแกรม

## Google Apps Script

2 เม.ย. 2020 (พิมพ์ครั้งที่ 3)

เรียบเรียงโดย  
วสันต์ คุณดิลกเศวต

พิมพ์ครั้งที่ 1 : 12 พ.ย. 2019

พิมพ์ครั้งที่ 2 : 1 ธ.ค. 2019

พิมพ์ครั้งที่ 3 : 2 เม.ย. 2020

# หลักการเขียนโปรแกรม Google Apps Script

เรียบเรียงโดย  
วสันต์ คุณดิลกเศวต

wasankds@gmail.com  
Line ID : wasankds  
08-1459-8343  
[www.poeclub.org](http://www.poeclub.org)

# สารบัญ

สารบัญ.....	4
คำนำ.....	11
บทที่ 1 Hello World.....	13
1.1. Google Apps Script แบบฝังและแบบ Stand alone .....	14
1.1.ก.) แบบฝังอยู่ในไฟล์ Google Apps(14)	
1.1.ข.) แบบ Stand alone (ไฟล์โดด) (14)	
1.2. โค้ดแรก และ คีย์ลัดที่สำคัญในการเขียนโค้ด .....	15
1.2.ก.) โค้ดแรก (15)	
1.2.ข.) คอมเมนต์ (16)	
1.2.ค.) คีย์ลัด (16)	
1.2.ง.) รันโค้ด (17)	
1.2.จ.) เปิดดู Logger (17)	
1.2.ฉ.) Logs ของ Chrome V8 (18)	
1.3. Apps Script Dashboard .....	19
บทที่ 2 ตัวแปร โอเปอเรเตอร์ และ ฟังก์ชัน.....	21
2.1. ที่มา .....	22
2.2. Variables และ Data Types .....	22
2.3. Operators .....	23
2.3.ก.) Operators ตัวเลข (23)	
2.3.ข.) Operators ข้อความ (24)	
2.3.ค.) Operators เปรียบเทียบ (25)	
2.3.ง.) Logical Operators (25)	
2.4. ประกาศตัวแปรตามเงื่อนไข (Ternary Operator) .....	26
2.5. ฟังก์ชัน (Functions) .....	27
2.5.ก.) โครงสร้างของฟังก์ชัน (27)	
2.5.ข.) Globally declared และ Locally declared (27)	
2.5.ค.) ตัวอย่าง – ฟังก์ชันที่มีการส่งผ่าน Argument 2 ตัว (28)	
2.5.ง.) ตัวอย่าง - เรียกใช้ฟังก์ชันอื่น จากอีกฟังก์ชัน (28)	
2.5.จ.) ตัวอย่าง – เก็บฟังก์ชันไว้ในตัวแปร (28)	
2.5.ฉ.) ตัวอย่าง – Argument ที่เป็นฟังก์ชัน (29)	
2.5.ช.) สร้างฟังก์ชันไว้ใช้งานใน Google Sheet (29)	
2.5.ซ.) ทำให้เรียกใช้งานได้เหมือนกับฟังก์ชัน Built-in (30)	

2.6. คำเฉพาะ .....	30
2.6.ก.) undefined      (30)	
2.6.ข.) null      (31)	
2.7. this .....	31
<b>บทที่ 3 ตัวแปรวัตถุ.....</b>	<b>33</b>
3.1. ตัวแปร Object (วัตถุ) .....	34
3.2. get, set และ Object method - Chrome V8 .....	34
3.2.ก.) get และ function ใน Object   (34)	
3.2.ข.) set      (35)	
3.3. เพิ่ม, ลบ, แทรก สมาชิกให้ Object .....	36
3.4. Object Constructors - Chrome V8 .....	36
3.5. Object Prototypes .....	37
3.6. Object Method key() .....	38
<b>บทที่ 4 เงื่อนไขและลูป.....</b>	<b>39</b>
4.1. if – else if - else .....	40
4.1.ก.) โครงสร้างของ if   (40)	
4.1.ข.) โครงสร้างของ if – else   (40)	
4.1.ค.) โครงสร้างของ if – else if - else      (41)	
4.2. switch .....	41
4.3. for .....	42
4.4. for in .....	44
4.5. while .....	45
4.6. do while .....	46
4.7. try catch .....	47
4.7.ก.) try catch      (47)	
ผล   (47)	
4.7.ข.) throw   (48)	
4.7.ค.) finally   (48)	

<b>บทที่ 5 อาเรย์.....</b>	<b>51</b>
5.1. อาเรย์คืออะไร .....	52
5.2. พื้นฐานข้อมูลชนิดอาเรย์ .....	52
5.3. อาเรย์ 2 มิติ .....	52
5.3.ก.) การเขียนอาเรย์ 2 มิติ (53)	
5.3.ข.) การเข้าถึงสมาชิกในอาเรย์ 2 มิติ (53)	
5.3.ค.) การนับจำนวนสมาชิกในอาเรย์ (53)	
<b>บทที่ 6 อาเรย์เมธอด.....</b>	<b>55</b>
6.1. push() .....	56
6.2. unshift() .....	56
6.3. pop() .....	57
6.4. shift() .....	57
6.5. splice() .....	58
6.6. slice() .....	59
6.7. ตัวอย่างการใช้งานอาเรย์เมธอดกับอาเรย์ 2 มิติ .....	60
6.8. join() .....	60
6.9. concat() .....	61
6.10. indexOf() .....	62
6.11. lastIndexOf() .....	63
6.12. reverse() .....	63
6.13. includes() - Chrome V8 .....	64
<b>บทที่ 7 อาเรย์เมธอด ตระกูลวนรูป.....</b>	<b>65</b>
7.1. map() .....	66
7.1.ก.) map() และ อาเรย์ 1 มิติ(66)	
7.1.ข.) map() และ อาเรย์ 2 มิติ(67)	
7.2. filter() .....	68
7.2.ก.) ตัวอย่างที่ 1 (68)	
7.2.ข.) ตัวอย่างที่ 2 – ใส่ฟังก์ชันเป็น Argument (69)	
7.2.ค.) ตัวอย่างที่ 3 - กรองตัวเลข (69)	
7.2.ง.) ตัวอย่างที่ 4 - กรองข้อความ (70)	
7.3. sort() .....	70
7.3.ก.) ข้อมูลเป็นตัวเลขทั้งหมด (70)	
7.3.ข.) ข้อมูลเป็นข้อความทั้งหมด (71)	
7.3.ค.) ข้อมูลผสม (73)	
7.3.ง.) เรียงข้อมูลในอาเรย์ 2 มิติ(75)	
7.3.จ.) เรียงโดยใช้ 2 คีย์ (หรือมากกว่า) – ใช้อาเรย์อย่างเดียว (76)	
7.3.ฉ.) เรียงโดยใช้ 2 คีย์ (หรือมากกว่า) – ใช้ Object ช่วย (77)	

<b>7.4. every() และ some()</b> .....	<b>78</b>
7.4.ก.) ตัวอย่างที่ 1 : อาเรย์ 1 มิติ	(78)
7.4.ข.) ตัวอย่างที่ 2 : อาเรย์ 2 มิติ	(79)
<b>7.5. forEach() Method</b> .....	<b>79</b>
7.5.ก.) โครงสร้างการใช้งาน	(79)
7.5.ข.) ตัวอย่างที่ 1 : การใช้งานแบบ 1 Argument	(80)
7.5.ค.) ตัวอย่างที่ 2 : การใช้งานแบบ 2 Arguments	(80)
7.5.ง.) ตัวอย่างที่ 3 : การใช้งานแบบ 3 Arguments	(80)
7.5.จ.) ตัวอย่างที่ 4 : ตัวอย่างอื่นๆ	(81)
<b>7.6. reduce() Method</b> .....	<b>81</b>
7.6.ก.) โครงสร้างการใช้งาน	(82)
7.6.ข.) ตัวอย่างที่ 1 : รวมค่าของสมาชิกในอาเรย์	(82)
7.6.ค.) ตัวอย่างที่ 2 : ลดมิติของอาเรย์	(83)
7.6.ง.) ตัวอย่างที่ 3 : หาค่าเฉลี่ยของสมาชิกในอาเรย์	(83)
7.6.จ.) ตัวอย่างที่ 4 : หาค่าตัวซ้ำ	(84)

## **บทที่ 8 อาเรย์เทคนิค.....87**

<b>8.1. จับแถวและคอลัมน์ใน Google Sheets มาใส่อาเรย์</b> .....	<b>88</b>
<b>8.2. จับเร้นจีใส่ในอาเรย์ จับอาเรย์ใส่ในเร้นจ (Google Sheets)</b> .....	<b>88</b>
8.2.ก.) จับเร้นจีใน Google Sheet มาใส่ในอาเรย์	(89)
8.2.ข.) จับค่าในอาเรย์ ไปใส่ใน Google Sheet	(89)
<b>8.3. นำคอลัมน์ในชีทมาคำนวณ แล้ววางไว้อีกคอลัมน์หนึ่ง (Google Sheets)</b> .....	<b>90</b>
8.3.ก.) พัฒนาการที่ 1	(90)
8.3.ข.) พัฒนาการที่ 2	(91)
<b>8.4. กรองแถวว่าง หรือ แถวที่มีเซลล์ว่างทิ้ง</b> .....	<b>92</b>
<b>8.5. การกรองตัวซ้ำในอาเรย์ ให้เหลือแต่ตัวที่ไม่ซ้ำ</b> .....	<b>93</b>
8.5.ก.) เทคนิคที่ 1 - สั้นแต่มีประสิทธิภาพสูง	(93)
8.5.ข.) เทคนิคที่ 2 - แยกสมาชิกในอาเรย์ที่ซ้ำและไม่ซ้ำออกเป็นอาเรย์ 2 ก้อน	(94)
<b>8.6. การกรองตัวซ้ำในอาเรย์ ให้เหลือแต่ตัวที่ซ้ำ</b> .....	<b>94</b>
<b>8.7. เปรียบเทียบอาเรย์ 2 ก้อน</b> .....	<b>96</b>
8.7.ก.) เปรียบเทียบอาเรย์ 2 ก้อน เก็บตัวที่ไม่ซ้ำกันไว้	(96)
<b>8.8. กรองเซลล์ว่างทิ้ง (ข้อมูล 1 คอลัมน์)</b> .....	<b>96</b>
<b>8.9. ใช้ filter() กรองตารางข้อมูลใน Google Sheet</b> .....	<b>97</b>
8.9.ก.) พัฒนาการที่ 1 : จับค่าในคอลัมน์มากรอง	(98)
8.9.ข.) พัฒนาการที่ 2 : ใส่ผลการกรองกลับไปชี้ที่	(98)
8.9.ค.) พัฒนาการที่ 3 : กรอง 2 เงื่อนไข	(99)

8.10. ทำ VLOOKUP ด้วยอาเรย์ .....	100
8.11. ทำ Lookup แบบคืนหลายค่า .....	101
8.12. วนลูปเช็คข้อมูลลงใน Google Sheets .....	102
8.13. แทรกคอลัมน์ลงในอาเรย์ 2 มิติ .....	103
8.14. จับคอลัมน์ออกมาจากอาเรย์ 2 มิติ .....	104
<b>บทที่ 9 รู้จักกับ Google Services.....</b>	<b>105</b>
9.1. สรุป OOP .....	106
9.2. Google Services (G Suite services) .....	106
9.3. การเรียกใช้ Google Services .....	107
9.4. Enum หรือ Enumeration .....	108
9.5. Interfaces .....	111
<b>บทที่ 10 คลาส Logger.....</b>	<b>115</b>
10.1. คลาส Logger .....	116
10.2. log() .....	116
<b>บทที่ 11 Triggers.....</b>	<b>117</b>
11.1. Triggers คืออะไร .....	118
11.2. Simple Triggers .....	118
11.2.ก.) ตัวอย่าง เมื่อคอลัมน์ A เปลี่ยน พิมพ์เวลาที่คอลัมน์ B(118)	
11.3. Event Objects .....	119
11.4. Installable Triggers .....	120
11.5. newTrigger() Method .....	122
<b>บทที่ 12 การใช้โปรเจ็ค Google Apps Script ในไฟล์อื่น.....</b>	<b>123</b>
12.1. การนำโปรเจ็ค Google Apps Script ไปใช้ในไฟล์อื่น .....	124
12.1.ก.) ปัญหาการใช้งาน 1 โปรเจ็ค(แบบฝัง) ในหลายไฟล์ (124)	
12.1.ข.) การนำโปรเจ็ค Google Apps Script ไปใช้ในไฟล์อื่น (124)	
12.1.ค.) การนำโปรเจ็คไปใช้ในไฟล์อื่นแบบข้าม Account และ ข้าม Domain (127)	
12.1.ง.) สรุป (128)	







# คำนำ

ใครที่ต้องการเริ่มต้นเขียนโปรแกรมด้วย Google Apps Script หนังสือเล่มนี้ เหมาะสำหรับการเริ่มต้น

ผู้เขียน เขียนหนังสือเล่มนี้ “หลักการเขียนโปรแกรม Google Apps Script” จุดประสงค์ดั้งเดิม ก็คือ เก็บไว้  
อ่านเอง

เมื่อผู้เขียนศึกษาเรื่องอะไร ก็จะไปเรียนรู้จากสื่อออนไลน์ในอินเทอร์เน็ต ทั้งคอร์สออนไลน์ วิดีโอ หรือ เอกสาร  
ทั้งในแบบฟรีและเสียเงิน

แต่ยุคนี้ เราต้องเรียนรู้อะไรให้เร็ว โดยเฉพาะเรื่องของ IT ผู้เขียนจึงตั้งใจจะดูวิดีโอเพียงรอบเดียว จึงดูไป พิมพ์  
สรุปไปด้วย เวลาจำอะไรไม่ได้ มาดูจากที่พิมพ์สรุปไว้ ง่ายกว่าการไปย้อนดูจากวิดีโอ นอกจากนี้ก็ยังนำมาทบทวนได้  
ง่าย ในอนาคตสามารถเพิ่มเติมเสริมแต่งเนื้อหาได้เรื่อยๆด้วย

หนังสือเล่มนี้ ความตั้งใจดั้งเดิมของผู้เขียน ก็คือ ตั้งใจเก็บไว้อ่านเองคนเดียว อย่างไรก็ตาม อดสำหรับพิมพ์ไว้เป็น  
หนังสือแล้ว จะเก็บไว้อ่านคนเดียวก็รู้สึกเสียดาย ฉะนั้น ผู้เขียนจึงนำออกมาแบ่งปัน

เนื่องด้วย ผู้เขียนให้ความสำคัญกับประเด็นด้านลิขสิทธิ์มาก ฉะนั้นจึงขอแจ้งไว้ ณ ที่นี้ ตั้งแต่ต้น  
ก็คือ

1. เนื้อหาในหนังสือ ผู้เขียนรวบรวมมาจากแหล่งต่างๆในอินเทอร์เน็ต โดยจะพยายามให้มากที่สุด ที่จะบอก  
ลิงค์หรือแหล่งที่มาในแต่ละหัวข้อ เพราะหนังสือเล่มนี้ถูกเขียนไว้นานแล้ว บางเรื่องลืมก็อปปีลิงค์มาแปะไว้
2. ผู้เขียนรวบรวมเนื้อหาจากแหล่งต่างๆ และเขียนเพิ่มเติมไปด้วย
3. หนังสือเล่มนี้แจกฟรี ผู้เขียนไม่มีรายได้จากหนังสือเล่มนี้

หนังสือเล่มนี้ยังไม่จบเสียทีเดียว หากผู้เขียนว่าง จะมาเขียนเพิ่มเติมเรื่อยๆ ให้ดูเวอร์ชันตามวันที่ที่ปล่อย  
หนังสือ

วสันต์ คุณดิลกเสวต  
wasankds@gmail.com  
081-459-8343  
Line ID : wasankds



# บทที่ 1

## Hello World

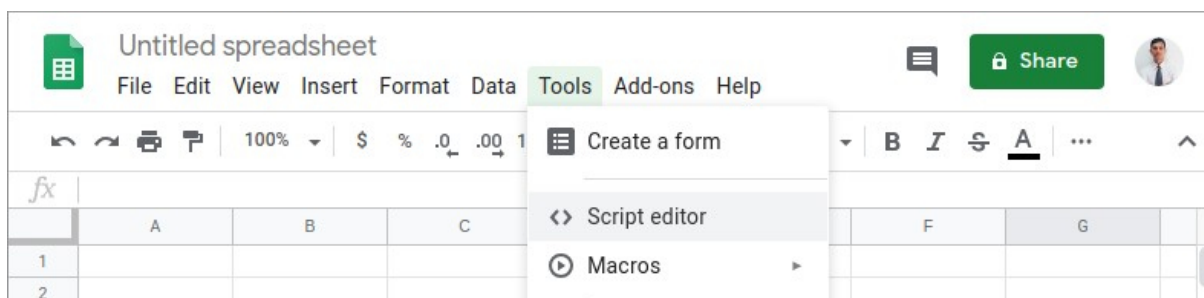


## 1.1. Google Apps Script แบบฝังและแบบ Stand alone

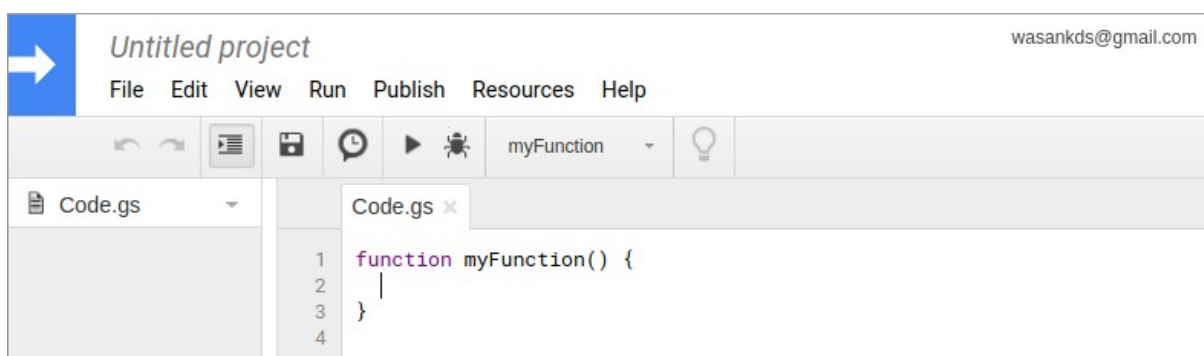
Google Script มี 2 ลักษณะใหญ่ๆ ก็คือ แบบฝังอยู่ในไฟล์ กับ แบบ Stand alone

### 1.1.ก.) แบบฝังอยู่ในไฟล์ Google Apps

ขณะทำงานอยู่กับไฟล์ Google Apps เช่น Google Sheet, Google Docs หรือ Google Slide เป็นต้น เราสามารถสร้างโปรเจกต์ Google Apps Script ได้โดยไปที่เมนู Tools → Script Editor ตามภาพ



จะปรากฏหน้าต่างเว็บสำหรับเขียนโค้ด ซึ่งเราเรียกหน้าต่างนี้ว่า หน้าโปรเจกต์ Google Apps Script ตามภาพ



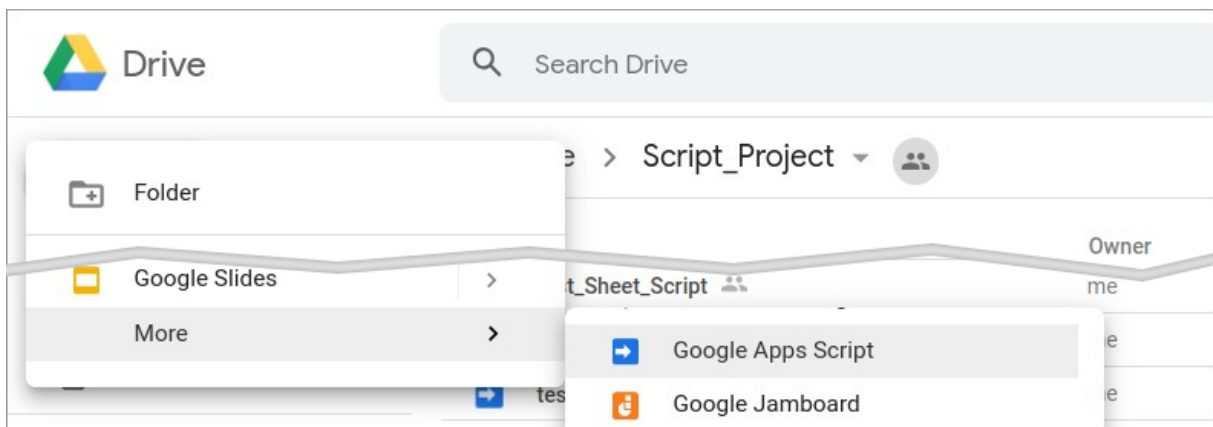
Google Apps Script แบบฝังอยู่ในไฟล์ Google Apps จะมองไม่เห็นเป็นชื่อไฟล์ใน Google Drive เพราะฝังอยู่กับไฟล์ของ Google Apps

### 1.1.ข.) แบบ Stand alone (ไฟล์โดด)

เราสามารถสร้าง Google Apps Script แบบ Stand alone ได้ที่ Google Drive

โดย ที่หน้าแรก Google Drive ไปที่ New → More → Google Apps Script (ตามภาพถัดไป) จะเป็นการสร้างโปรเจกต์และไฟล์ Google Apps Script ขึ้นมา

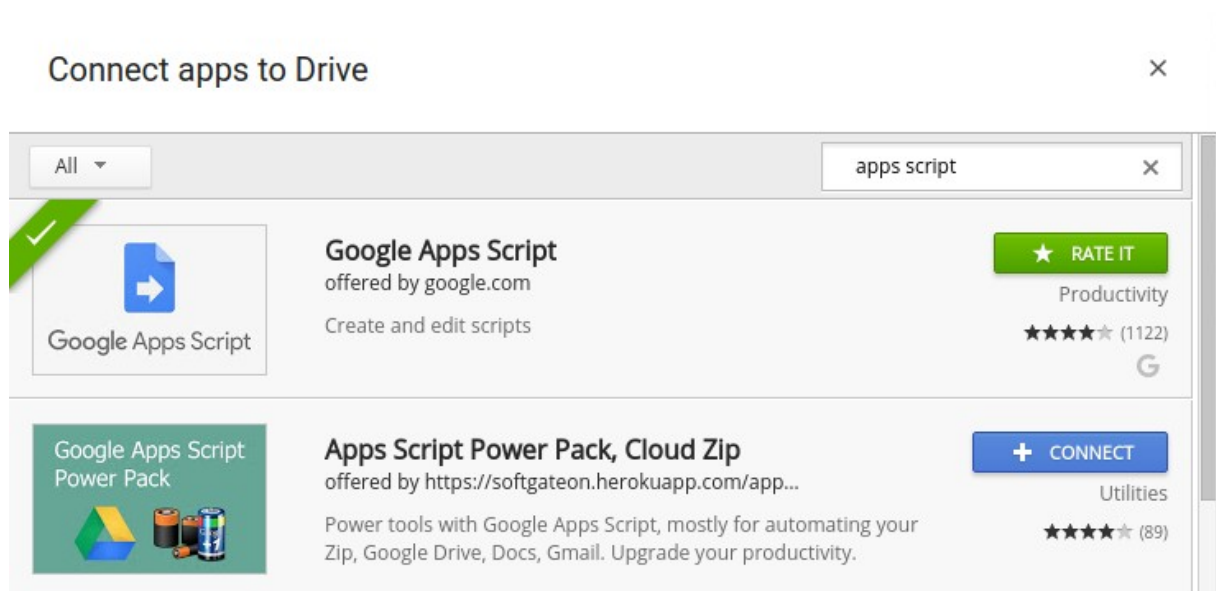
โปรเจกต์ Google Apps Script แบบนี้ สามารถมองเห็นเป็นไฟล์ใน Google Drive ด้วยเหตุนี้จึงเรียกว่า Stand alone (ไฟล์โดด)



### หมายเหตุสำคัญ !!!

หากเมนู New → More → Google Apps Script ไม่มี ให้เราเชื่อมต่อ Google Drive กับแอป Google Apps Script ก่อน โดย ...

ที่หน้าแรก Google Drive ไปที่เมนู New → More → Connection more apps จะปรากฏหน้าต่าง(ตามภาพ) ให้ค้นหาด้วยคำว่า apps script จะพบแอปที่ชื่อ Google Apps Script (ตามภาพ) จากนั้นให้คลิกที่ Connect เพื่อเชื่อมต่อกับแอปดังกล่าว



## 1.2. โค้ดแรก และ ศัพท์สำคัญในการเขียนโค้ด

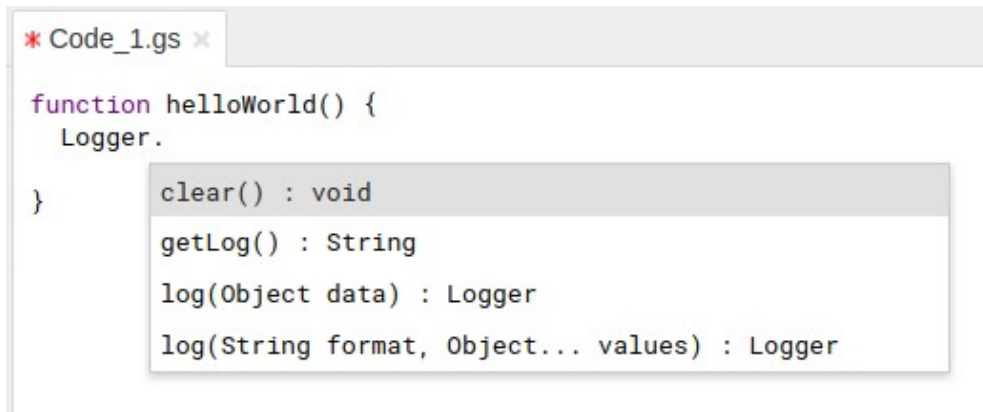
เมื่อสร้างโปรเจ็ค Google Apps Script มาแล้ว ให้เรามาเริ่มต้น เขียนโค้ดแรกกันก่อน เพื่อทำความเข้าใจกับระบบโดยรวมก่อน

### 1.2.ก.) โค้ดแรก

เขียนโค้ดดังต่อไปนี้

```
function myFunction(){
  Logger.log("Hello, World !!!") ;
}
```

ตอนพิมพ์ `Logger` ตามด้วย `.` (จุด) จะปรากฏ AutoComplete ซึ่งเป็นรายการของ Method หรือ Properties ของคลาส `Logger` ให้เลือก ก็ช่วยให้เราเขียนโค้ดง่ายขึ้น



```
* Code_1.gs x
function helloWorld() {
  Logger.
}
```

The dropdown menu shows the following options:

- `clear() : void`
- `getLog() : String`
- `log(Object data) : Logger`
- `log(String format, Object... values) : Logger`

### 1.2.ข.) คอมเมนต์

**คอมเมนต์** ก็คือ บรรทัดของโค้ดที่เราไม่ต้องการให้รัน แต่มีไว้สำหรับให้ผู้เขียนโค้ดเขียนอธิบาย

ใช้เครื่องหมาย `// คอมเมนต์` เพื่อมาร์คว่าบรรทัดนี้เป็นคอมเมนต์

ใช้เครื่องหมาย `/* คอมเมนต์ */` เพื่อมาร์คว่าภายในบล็อกเป็นคอมเมนต์

ตัวอย่าง

```
function myFunction(){
  // พิมพ์ข้อความ Hellow, World !!! ที่หน้าต่าง Logger
  Logger.log("Hellow, World !!!") ;
  /*
    โค้ดในบล็อกนี้ ไม่ถูกรัน
  */
}
```

### 1.2.ค.) คีย์ลัด

`<Ctrl></>` = ใส่เครื่องหมาย `//` หน้าบรรทัด ก็คือ บรรทัดนั้นจะเป็น Comment

`<Alt></>` = ใส่ AutoComplete กับชื่อตัวแปร

`<Ctrl><Space>` = ใส่ AutoComplete เพื่อดูรายการ Method/Properties

(ตอนพิมพ์ `.` (จุด) หลังคลาสหรือตัวแปร แล้วขึ้น Method/Properties อัตโนมัติ)

`<Shift><Tab>` = ล่นย่อหน้าอัตโนมัติ

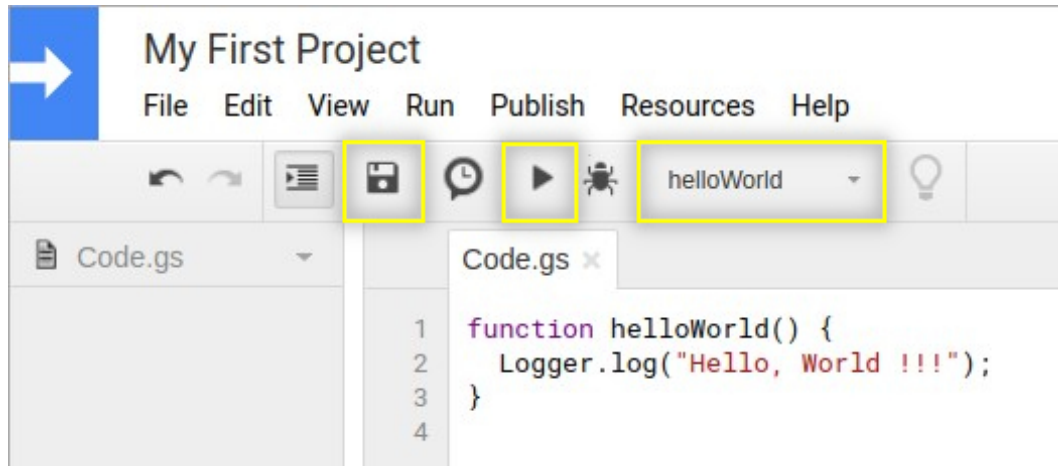


### 1.2.ง.) รันโค้ด

โค้ดถูกเขียนอยู่ในบล็อกของฟังก์ชัน เวลารันโค้ดเราจะรันที่ฟังก์ชัน ซึ่งสามารถรันได้ดังนี้

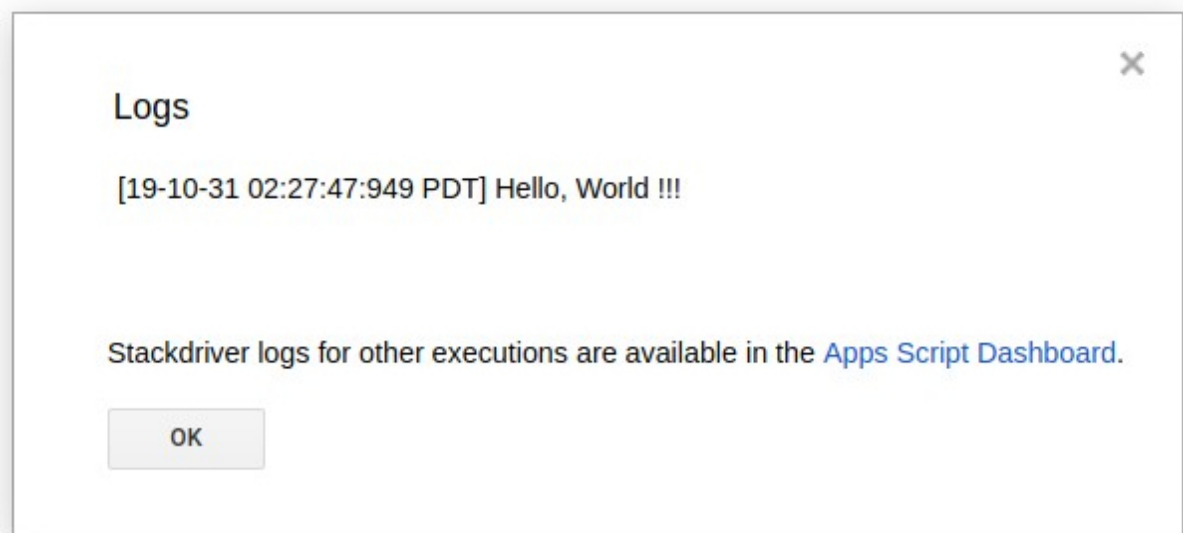
#### ขั้นตอนการรันโค้ด

1. บันทึกโปรเจกต์ - คลิกที่ปุ่ม Save หรือกด **<Ctrl><S>**
2. เลือกฟังก์ชันที่จะรัน - เช่น ฟังก์ชันชื่อ helloWorld
3. รันโค้ด - คลิกปุ่ม Run หรือกด **<Ctrl><R>**



### 1.2.จ.) เปิดดู Logs

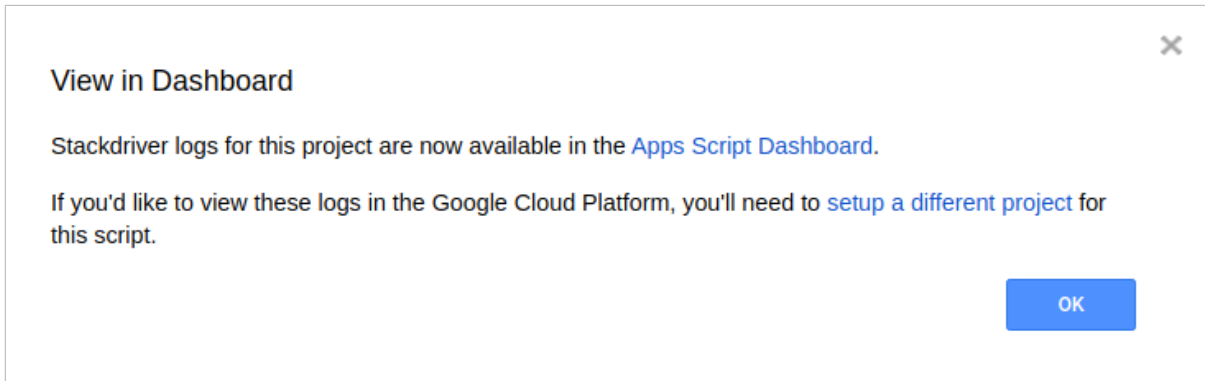
Logs เป็นหน้าต่างที่ใช้รับผลจากการรันโค้ดมาแสดง เมื่อรันโค้ดแล้ว ถ้าโค้ดของเราเรียกใช้บริการของ Logger เราสามารถเปิดดู Logs ได้โดยไปที่ เมนู View → Logs หรือกด **<Ctrl><Enter>**



## 1.2.ค.) Logs ของ Chrome V8

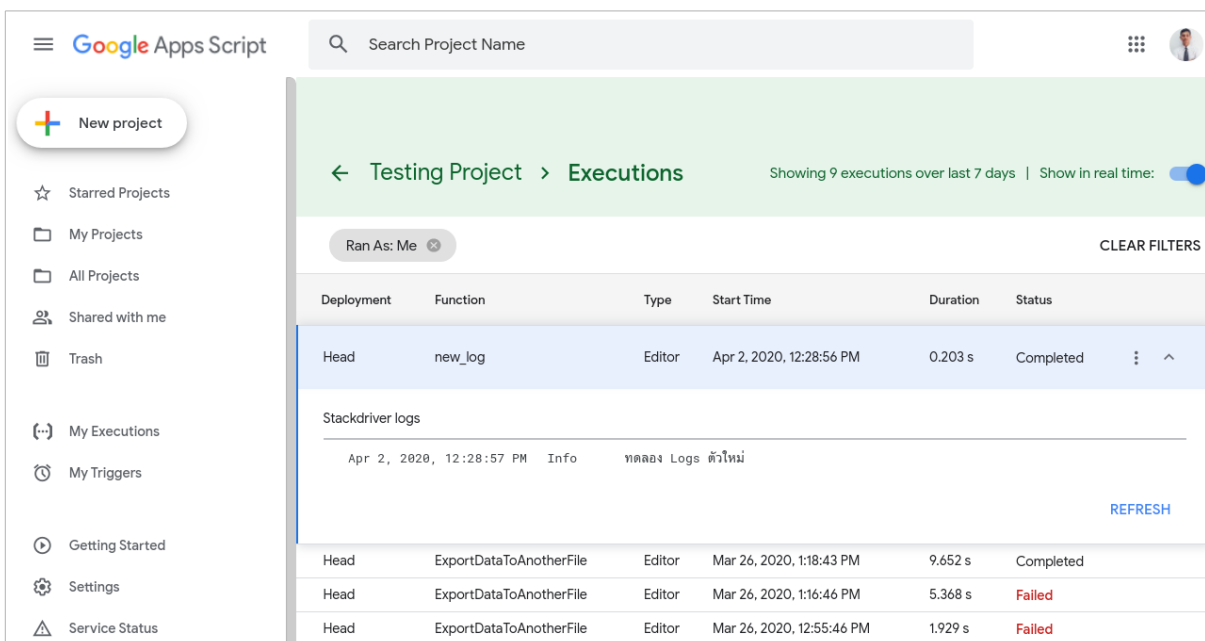
Google Apps Script รุ่นล่าสุดใช้เครื่องย่นต์ Chrome V8 ได้เพิ่ม Logs ระบบใหม่ ก็คือ **Stackdriver logging** เมื่อจะรันสคริปต์แล้วกด **<Ctrl><Enter>** เพื่อดู Logs แบบเดิมก็ได้ (แต่ช้ากว่าเดิมมาก)

หรือจะดู Logs ที่ Stackdriver logging ก็ได้ โดยไปที่เมนู **View → Stackdriver logging** จะปรากฏหน้าตามภาพ



คลิกที่ **Apps Script Dashboard** จะปรากฏหน้าเว็บแสดง Logs ตามภาพถัดไป

เมื่อรันสคริปต์ ผลจาก Logs จะปรากฏที่นี่ โดย Logs ต่างๆที่เราเคยรันไว้จะเก็บไว้ที่นี่ด้วย สามารถดูย้อนหลังได้ นี่เป็นข้อดีของ Logs แบบนี้



Deployment	Function	Type	Start Time	Duration	Status
Head	new_log	Editor	Apr 2, 2020, 12:28:56 PM	0.203 s	Completed

Stackdriver logs

Time	Level	Message
Apr 2, 2020, 12:28:57 PM	Info	ทดลอง Logs ตัวใหม่

### 1.3. Apps Script Dashboard

เมื่อสร้างโปรเจ็ค Google Apps Script อะไรก็ตาม ไม่ว่าจะแบบฝังในไฟล์ หรือ แบบไฟล์โดด เราสามารถดูโปรเจ็ค Google Apps Script ทั้งหมดได้ที่ Apps Script Dashboard ตามลิงค์ดังต่อไปนี้

<https://script.google.com/home/my>

สังเกตที่ไอคอนหน้าชื่อโปรเจ็ค จะเห็นว่าเป็นแบบฝังในไฟล์อะไร หรือเป็นแบบ Stand alone

Project	Owner	Last modified
test Sheet Script	Me	2:06 PM
test Standalone Script	Me	Oct 29, 2019
test_Doc_Script	Me	Oct 29, 2019
MM Email	Me	Oct 29, 2019



บทที่ 2  
ตัวแปร โอเปอเรเตอร์  
และ ฟังก์ชัน



## 2.1. ที่มา

ที่มาของบทนี้ หลักๆผู้เขียนนำมาจาก คอร์ส Apps Script Blastoff (เรียนฟรี) เหมาะมากสำหรับผู้เริ่มต้น(ภาษาอังกฤษทั้งหมด) แต่พอเขียนไปเรื่อยๆก็เอาเนื้อหาจากที่นู่นที่นี้มาแปะเพิ่มเติม เนื้อหาจากที่ไหนผู้เขียนจะแปะลิงค์ไว้ นอกจากนี้ผู้เขียนก็เขียนเพิ่มลงไปเองด้วย โดยเฉพาะตัวอย่าง

คอร์ส Apps Script Blastoff (เรียนฟรี)

<https://courses.benlcollins.com/courses/enrolled/435404>

JavaScript Data Types

[https://www.w3schools.com/js/js\\_datatypes.asp](https://www.w3schools.com/js/js_datatypes.asp)

## 2.2. Variables และ Data Types

Variables หรือ ตัวแปร ใช้เก็บค่าต่างๆ โดย Variables มีหลายชนิดแล้วแต่ว่าจะใช้เก็บอะไร ฉะนั้น Variables จึงจำแนกได้หลายชนิด ซึ่งเราเรียกคุณสมบัตินี้ว่า Data Types หรือ ชนิดข้อมูล

ให้ทดสอบพิมพ์โค้ดต่อไปนี้ แล้วดูที่ Logs เพื่อเรียนรู้เรื่องชนิดของข้อมูล

ใช้ **typeof (ชนิดตัวแปร)** เพื่อดูว่า ตัวแปรดังกล่าวมี **ชนิดข้อมูล** เป็นอะไร

```
// 1. String (text values) – ข้อความ – ให้ใช้ Single quotes ครอบ =====
var myName = 'Wasan Khunnadiloksawet' ;
Logger.log(myName) ;           // พิมพ์ : Wasan Khunnadiloksawet
Logger.log(typeof myName) ;    // พิมพ์ : String

// 2. Number – ตัวเลข =====
var myAge = 41 ;
Logger.log(myAge);             // พิมพ์ : 41
Logger.log(typeof myAge);      // พิมพ์ : number

// 3. Boolean – บูลีน มี 2 ค่า true/false =====
var result = true;
Logger.log(result);            // พิมพ์ : true
Logger.log(typeof result);     // พิมพ์ : boolean

// ไม่ได้ประกาศตัวแปร แบบนี้ไม่ดี
alpha = 10;
Logger.log(alpha);
Logger.log(typeof alpha);

// 4. Function – ตัวแปรแบบฟังก์ชัน =====
var newFunc = function(a,b){
    return a+b ;
};
Logger.log(newFunc);           // พิมพ์ : ( Code ของ newFunc )
Logger.log(typeof newFunc);    // พิมพ์ : function

// ตัวอย่างการเรียกใช้งานฟังก์ชัน
Logger.log(newFunc(5,11));      // พิมพ์ : 16
```

```
// 5. Object – ตัวแปรวัตถุ =====
var myDetail = {
    name : 'Wasan' ,
    lastname : 'KDS' ,
    age : 43
};
Logger.log(myDetail);           // พิมพ์ : {name=Wasan, age=43.0, lastname=KDS}
Logger.log(typeof myDetail);    // พิมพ์ : object

// 6. Array – อาร์เรย์ – ก้อนข้อมูล
var myArray = [ 1 , 2 , 3 , 4 , "AAA" ];
Logger.log(myArray);           // พิมพ์ : [1.0, 2.0, 3.0, 4.0, AAA]
Logger.log(typeof myArray);    // พิมพ์ : object *** Array is an object data type.
```

## 2.3. Operators

JavaScript Arithmetic Operators

[https://www.w3schools.com/js/js\\_operators.asp](https://www.w3schools.com/js/js_operators.asp)

Operators หรือ ตัวดำเนินการ เช่น เครื่องหมาย +, -, \*, / เป็นต้น แล้วแต่ว่าการคำนวณของเราเป็นสูตรแบบไหน สูตรตัวเลข ข้อความ บูลีน อาร์เรย์ เป็นต้น ก็จะมีเครื่องหมายโอเปอเรเตอร์ให้เลือกใช้ต่างกัน

ทดสอบพิมพ์โค้ดต่อไปนี้ แล้วดูที่ Logger

### 2.3.ก.) Operators ตัวเลข

```
function mathOperators() {
    var a = 10 ;
    var b = 2 ;

    // โอเปอเรเตอร์พื้นฐาน =====
    Logger.log(a+b);    // พิมพ์ : 12.0
    Logger.log(a-b);    // พิมพ์ : 8.0
    Logger.log(a*b);    // พิมพ์ : 20.0
    Logger.log(a/b);    // พิมพ์ : 5.0

    // หารเอาเศษ =====
    Logger.log(a%b);    // พิมพ์ : 0
    Logger.log(5%2);    // พิมพ์ : 1.0
    Logger.log(1%0);    // พิมพ์ : NaN

    // ยกกำลัง =====
    // Math object ( ดูเพิ่มเติมข้อ บทที่ 9 รู้จักกับ Google Services หน้า 105 )

    var c = Math.pow(a, b) ;
    Logger.log(c);

    // เพิ่มค่าหรือลดค่าด้วย 1 =====
    var i=0 ;    Logger.log(i) ; // พิมพ์ : 0
    i++ ;    Logger.log(i) ;    // พิมพ์ : 1
    i-- ;    Logger.log(i) ;    // พิมพ์ : 0
```

```
// เพิ่มค่าหรือลดค่า ด้วยค่าอีกตัว =====
var x = 1
var y = 50
var z = 5
var w = 100

for ( var i = 1 ; i < 5 ; i++){
  x += i ;    // x = x + i
  y -= i ;    // y = y + i
  z *= i ;    // z = z + i
  w /= i ;    // w = w + i
}

Logger.log(x) ;    // พิมพ์ : 11
Logger.log(y) ;    // พิมพ์ : 40
Logger.log(z) ;    // พิมพ์ : 120
Logger.log(w) ;    // พิมพ์ : 4.166666666666667
}
```

ดูการเปลี่ยนแปลงครั้ง ตามตารางต่อไปนี้

ลูป	i	x	y	x	w
1	1	2	49	5	100
2	2	4	47	10	50
3	3	7	44	30	16.6666668
4	4	11	40	120	4.16666667

## 2.3.ข.) Operators ข้อความ

เครื่องหมาย + ใช้เชื่อมข้อความ

```
function stringOperators(){
  // ใช้เครื่องหมาย Single quotes หรือ Doubles ครอบค่าที่เป็นข้อความ
  // ตัวไหนก็ได้ แต่ต้องเหมือนกันทั้งบรรทัด

  var string1 = "Wasan";           // Double quotes
  var string2 = 'Khunnadiloksawet' ; // Single quotes
  var string3 = 'Napabhorn';       // Single quotes

  // ใช้เครื่องหมาย + ในการเชื่อมข้อความ

  var fullname = string3+" & "+" string1 + " " + string2 ;
  Logger.log(fullname);
  // พิมพ์ : Napabhorn & Wasan Khunnadiloksawet
}
```



### 2.3.ค.) Operators เปรียบเทียบ

Operators เปรียบเทียบ ใช้เปรียบเทียบได้ทั้งตัวเลขและข้อความ เช่น เครื่องหมาย =, <, > เป็นต้น  
เน้นทำความเข้าใจกับเครื่องหมายเท่ากับ เพราะมีถึง 3 แบบ ก็คือ =, ==, ===

```
function comparisonOperators(){  
  
    var a=10 ; // Number  
    var b='10' ; // Text – เป็นข้อความ แต่หน้าตาเป็นตัวเลข  
  
    Logger.log(a) ; // พิมพ์ : 10.0  
    Logger.log(typeof a) ; // พิมพ์ : number  
    Logger.log(b) ; // พิมพ์ : 10  
    Logger.log(typeof b) ; // พิมพ์ : string  
  
    // 1. Single = : เครื่องหมาย = อันเดียว เอาค่าจากขวาไปใส่ซ้าย  
    a = b;  
    Logger.log(a) ;  
    Logger.log(typeof a) ;  
  
    // 2. Double == : เปรียบเทียบเฉพาะค่า  
    Logger.log(a==b) ; // พิมพ์ : true - แม้ชนิดข้อมูลไม่ตรงกัน  
  
    // 3. Triple === : เปรียบเทียบค่าและชนิดข้อมูล  
    Logger.log(a===b) ; // พิมพ์ : false - ตรวจสอบทั้งค่าและชนิดของข้อมูล  
  
    // 4. Not equal != : เปรียบเทียบเฉพาะค่า  
    Logger.log(a != b) ; // พิมพ์ : false - เพราะ ค่าตรงกัน  
  
    // 5. Not equal !== : เปรียบเทียบทั้งค่าและชนิด  
    Logger.log(a !== b) ; // พิมพ์ : true - เพราะ ชนิดข้อมูล ไม่ตรงกัน  
  
    // 6. Greater than, Less than operators  
    var c=50 ;  
    var d=100 ;  
  
    Logger.log(c>d); // พิมพ์ : false  
    Logger.log(c>=d); // พิมพ์ : false  
    Logger.log(c<d); // พิมพ์ : true  
    Logger.log(c<=d); // พิมพ์ : true  
}
```

### 2.3.ง.) Logical Operators

JavaScript Comparison and Logical Operators

[https://www.w3schools.com/js/js\\_comparisons.asp](https://www.w3schools.com/js/js_comparisons.asp)

Logical Operators หรือ ตัวดำเนินการตรรกะ มี 3 ตัวก็คือ && (และ) , || (หรือ) และ ! (ไม่)

Operators	Description	Example
&&	and	(x < 10 && y > 1)
	or	(x == 5    y == 5)
!	not	!(x == y)

ตัวอย่าง

```
function myFunction() {  
    var num = 62 ;  
    if(num > 0 && num < 100) {    // คะแนนมากกว่า 0 และ น้อยกว่า 100  
        Logger.log("คะแนนของคุณ คือ " + num) ;  
    } else {  
        Logger.log("Error") ;  
    } ;    // End – if  
} // End – function
```

ตัวอย่างที่ 2

```
function myFunction2() {  
    var num = 62 ;    var yyy ;  
    if(xxx) {  
        Logger.log(xxx) ;    // 62  
    }  
    if(yyy) {  
        Logger.log(yyy) ;  
    }else{  
        Logger.log(typeof yyy) ;    // undefined  
    }  
} // End – function
```

---

## 2.4. ประกาศตัวแปรตามเงื่อนไข (Ternary Operator)

---

JavaScript Comparison and Logical Operators

[https://www.w3schools.com/js/js\\_comparisons.asp](https://www.w3schools.com/js/js_comparisons.asp)

ข้อนี้ ไม่รู้ว่าควรจะจัดอยู่ในหัวข้อ **ตัวแปร** หรือ **Operator** ดี เพราะมันคาบเกี่ยวกันทั้ง 2 อย่าง

เราสามารถประกาศตัวแปร ตามเงื่อนไขได้ด้วย โดยใช้ Operator **?** ตามโครงสร้างต่อไปนี้

```
variablename = (condition) ? value1 : value2  
//                      true    false
```

ถ้าเงื่อนไขเป็น **true** ตัวแปร จะเก็บค่า **value1** แต่ถ้าเป็น **false** จะเก็บค่า **value2** ไว้

ตัวอย่าง

```
function MyFunction(){  
    var voteable, age = 19 ;  
    var voteable = (age < 18) ? "Too young" : "Old enough" ;  
    Logger.log(voteable) ;    // พิมพ์ : Old enough  
}
```

## 2.5. ฟังก์ชัน (Functions)

ฟังก์ชัน (Functions) เป็นบล็อกของโค้ด ที่ช่วยให้เราชอยโค้ดเป็นบล็อกต่างๆ ทำให้โค้ดอ่านง่าย และสามารถรียูสโค้ดได้ง่ายด้วย เพราะฟังก์ชันที่เขียนไว้แล้ว เราสามารถเก็บไว้ใช้ได้ตลอด เพียงแต่ต้องทราบว่าต้องส่งอะไรไปให้ฟังก์ชัน และฟังก์ชันคืนอะไรกลับมาให้เรา

### 2.5.ก.) โครงสร้างของฟังก์ชัน

Argument (ตัวแลกเปลี่ยน) เป็นวัตถุที่เราต้องส่งไปให้ฟังก์ชันประมวลผล

```
function functionName(argument1, argument2, argument3) {  
    // โค้ด ที่จะทำงานเมื่อฟังก์ชันถูกเรียกใช้งาน  
    // argument1, argument2, argument3 คือวัตถุที่จะนำมาประมวลผลในฟังก์ชัน  
}
```

ตัวอย่าง – การสร้างฟังก์ชันอย่างง่าย ที่ไม่มีการส่งผ่าน Argument (ตัวแลกเปลี่ยน)

```
function firstFunction() {  
    Logger.log("Hello world!");  
}
```

### 2.5.ข.) Globally declared และ Locally declared

การประกาศตัวแปร ในฟังก์ชัน(Locally declared) และ นอกฟังก์ชัน(Globally declared) ต่างกัน เช่น

ตัวแปร `price` ประกาศอยู่ในฟังก์ชัน `myPrice` เป็นตัวแปรแบบ Locally declared สามารถใช้งานได้เฉพาะในฟังก์ชัน `myPrice()` ไม่สามารถเรียกใช้จากนอกฟังก์ชันได้

ตัวแปร `Const` ประกาศอยู่นอกฟังก์ชัน เป็นตัวแปรแบบ Globally declared ตัวแปร `Const` สามารถถูกเรียกใช้จากฟังก์ชันใดก็ได้

```
var Const = 100 ;  
Logger.log(price); // Error เพราะตัวแปร price อยู่ในฟังก์ชัน แต่เรียกใช้งานข้างนอก  
  
function myPrice() { // รั้นฟังก์ชันนี้  
    var price = 9.99;  
    Logger.log(price); // พิมพ์ : 9.99  
    Logger.log(Const); // พิมพ์ : 100  
}  
  
Logger.log(price); // Error เพราะตัวแปร price อยู่ในฟังก์ชัน แต่เรียกใช้งานข้างนอก
```

## 2.5.ค.) ตัวอย่าง – ฟังก์ชันที่มีการส่งผ่าน Argument 2 ตัว

ตัวอย่างโค้ดนี้ส่งผ่าน Argument 2 ตัว เมื่อฟังก์ชันประมวลผลแล้ว จะคืนค่ากลับมา ตามบรรทัด

return

บรรทัด `Logger.log(addFunction(3,4));` ที่อยู่นอกฟังก์ชัน เป็นการเรียกใช้ฟังก์ชัน

```
function addFunction(firstNum, secondNum) {  
    return firstNum + secondNum ;  
}  
  
function call() {  
    Logger.log(addFunction(3,4));    // พิมพ์ : 7  
}
```

## 2.5.ง.) ตัวอย่าง - เรียกใช้ฟังก์ชันอื่น จากอีกฟังก์ชัน

เราเรียกใช้ฟังก์ชันอื่น จากอีกฟังก์ชันได้ ซึ่งจะเห็นแบบนี้บ่อยมากในการเขียนโค้ด

```
function addFunction(firstNum, secondNum) {  
    return firstNum + secondNum ;  
}  
  
function runOtherFunction() {    // รันฟังก์ชันนี้  
    var answer = addFunction(100,93) ;  
    // เรียกใช้ฟังก์ชัน และ เก็บการคืนค่าจากฟังก์ชันไว้ในตัวแปร  
    Logger.log(answer) ;  
}
```

## 2.5.จ.) ตัวอย่าง – เก็บฟังก์ชันไว้ในตัวแปร

เราสามารถเก็บฟังก์ชันไว้ในตัวแปรได้ เวลาเรียกใช้งาน จะเรียกจากชื่อตัวแปร

```
var mult = function(a,b) {  
    return a * b;  
}  
  
var power = function(a,b) {  
    return Math.pow(a,b);  
}  
  
function call() {  
    // เรียกใช้ฟังก์ชัน mult  
    Logger.log(mult);    // พิมพ์ : (โค้ดของฟังก์ชัน)  
    Logger.log(mult(5,10));    // พิมพ์ : 50  
    Logger.log(power(3,4));    // พิมพ์ : 81  
}
```

## 2.5.ฉ.) ตัวอย่าง – Agrument ที่เป็นฟังก์ชัน

Agrument ที่ใช้ส่งผ่านกันระหว่างฟังก์ชัน สามารถส่งผ่านเป็นฟังก์ชันได้

```
function superFunc(functionArgument, a, b) {  
  return functionArgument(a,b);  
}  
  
function call() {  
  Logger.log(superFunc(power,2,3));           // พิมพ์ : 8.0  
  Logger.log(superFunc(mult,2,3));             // พิมพ์ : 6.0  
  Logger.log(superFunc(addFunction,2,3));      // พิมพ์ : 5.0  
}
```

## 2.5.ช.) สร้างฟังก์ชันไว้ใช้งานใน Google Sheet

เขียนโค้ด Google Apps Script ต่อไปนี้เก็บไว้ จากนั้นบันทึกโค้ดไว้

```
function mulByX(x,y) {  
  return x*y;  
}
```

การใช้งานไม่ต้องรันโค้ด แต่ให้เรียกใช้จาก Google Sheet สามารถเรียกใช้ได้เหมือนการใช้งานฟังก์ชันทั่วไปของ Google Sheet อย่าง SUM ตัวอย่างการใช้งานเช่น

**=mulByTwo(5,8)**

หลังกด <Enter> เพื่อประมวลผล จะปรากฏข้อความ Loading เพราะต้องส่งโค้ดไปประมวลผลที่เซิร์ฟเวอร์ของ Google

fx	=mulByX(5,8)
	A
1	40

## 2.5.ช.) ทำให้เรียกใช้งานได้เหมือนกับฟังก์ชัน Built-in

(ลองแล้วยังไม่ Work - แต่เขียนเก็บไว้ก่อน)

รายละเอียดเพิ่มเติม

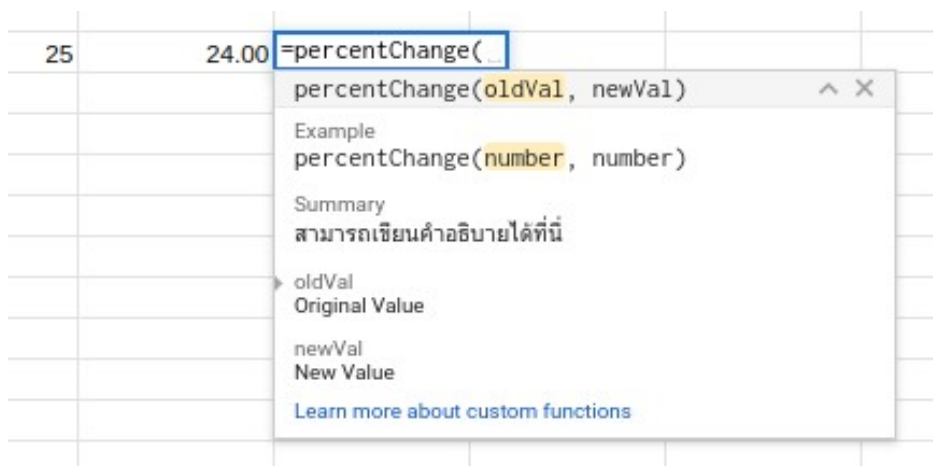
<https://developers.google.com/apps-script/guides/sheets/functions>

<https://yagisanatode.com/2018/08/24/google-apps-script-how-to-make-a-custom-function-to-use-in-google-sheets/>

ใส่ Comment ในโค้ดโดยระบุค่าตัวแปรต่างๆ ดังต่อไปนี้

```
/*สามารถเขียนคำอธิบายได้ที่นี้
 * @param {number} oldVal Original Value
 * @param {number} newVal New Value
 * @return The percent change between new and old value.
 * @customfunction */

function percentChange(oldVal,newVal) {
  return (newVal-oldVal)/oldVal;
}
```



## 2.6. คำเฉพาะ

### 2.6.ก.) undefined

JavaScript Global Reference

[https://www.w3schools.com/jsref/jsref\\_obj\\_global.asp](https://www.w3schools.com/jsref/jsref_obj_global.asp)

JavaScript undefined Property

[https://www.w3schools.com/jsref/jsref\\_undefined.asp](https://www.w3schools.com/jsref/jsref_undefined.asp)

**undefined** ก็คือไม่มี หรือไม่ได้จำกัดความหมาย เช่น ตัวแปรที่ยังไม่ได้ระบุค่า

```
var x ;
Logger.log(x) ; // พิมพ์ : undefined

if (typeof x === "undefined") {
  txt = "x is undefined" ; // ทำบรรทัดนี้
} else {
  txt = "x is defined" ;
}
```

## 2.6.ข.) null

**null** คล้ายกับ **undefined** ไม่มีข้อมูลเหมือนกัน แต่มันที่ชนิด

ตัวอย่าง

```
typeof undefined    // undefined
typeof null         // object

null === undefined  // false
null == undefined   // true
```

ตัวอย่าง

```
var person = {
    firstName : "John" ,
    lastName  : "Doe" ,
    age       : 50 ,
    eyeColor  : "blue"
};

person = undefined ; // Now both value and type is undefined

var person = {
    firstName : "John" ,
    lastName  : "Doe" ,
    age       : 50 ,
    eyeColor  : "blue"
};

person = null ; // Now value is null, but type is still an object
```

## 2.7. this

The JavaScript this Keyword

[https://www.w3schools.com/js/js\\_this.asp](https://www.w3schools.com/js/js_this.asp)

**this** เป็นคีย์เวิร์ดที่หมายถึง วัตถุตัวเอง **this** มีหลากหลายค่าขึ้นอยู่กับว่า ถูกเรียกใช้ที่ไหน

ตัวอย่าง **this** โดดๆ ก็คือโปรเจ็ค Google Apps Script

```
function myFunction() {
    var x = this ;
    Logger.log(x) ;
}
```

ตัวอย่าง **this** ในฟังก์ชัน ก็คือโปรเจ็ค Google Apps Script เหมือนกับข้างบน

```
function call() {  
  Logger.log(inFunc());  
}  
  
function inFunc() {  
  return this ;  
}
```

ตัวอย่าง **this** อยู่ใน Method ของวัตถุ

```
function myFunction() {  
  
  var person = {  
    firstName: "John" ,  
    lastName : "Doe" ,  
    id       : 5566 ,  
    fullName : function() {  
      return this.firstName + " " + this.lastName ;  
    } // Close - function  
  } ; // Close - person  
  
  Logger.log(person.firstName) ; // พิมพ์ : John  
  Logger.log(person.lastName) ;  // พิมพ์ : Doe  
  Logger.log(person.id) ;        // พิมพ์ : 5566.0  
  Logger.log(person.fullName()) ; // พิมพ์ : John Doe  
}
```



# บทที่ 3

## ตัวแปรวัตถุ



---

### 3.1. ตัวแปร Object (วัตถุ)

---

JavaScript Objects

[https://www.w3schools.com/js/js\\_object\\_definition.asp](https://www.w3schools.com/js/js_object_definition.asp)

JavaScript Object Accessors

[https://www.w3schools.com/js/js\\_object\\_accessors.asp](https://www.w3schools.com/js/js_object_accessors.asp)

ตัวแปร Object ประกอบไปด้วย **keys**(หรือ **Properties**) และ **values** โดยทั้ง **keys** และ **values** เราเรียกรวมกันว่า **items** (ดูตัวอย่างได้จากโค้ดด้านล่าง)

Object สามารถมี Method ของ Object ได้ด้วย (**Chrome V8**) โดย Object Method ก็คือ Property ที่บรรจุด้วยฟังก์ชัน เวลาเรียกใช้งานจึงเรียกใช้แบบ Method เช่น `Object.Method()`

```
function objectFunction1() {  
    // ใช้วงเล็บปีกกา ครอบค่าของตัวแปร Object  
  
    var newObj = {} ;                // สร้างตัวแปร Object ว่างๆ  
  
    // Object ประกอบไปด้วย key : value หลายตัว  
    // ลำดับของ key : value ไม่สำคัญ  
  
    var employee = {  
        name : 'Joe Bloggs' ,        // key ก็คือ name , value ก็คือ 'Joe Bloggs'  
        age : 25 ,                    // key ก็คือ age , value ก็คือ 25  
        title : 'Data Analyst'       // key ก็คือ title , value ก็คือ 'Data Analyst'  
        name_title : function() {    // Method ก็คือ name_title  
            return this.name + " " + this.title ;  
        } // Close - name_title  
    } // Close - employee  
  
    Logger.log(employee) ;            // พิมพ์ : {name=Joe Bloggs, title=Data Analyst, age=25.0}  
    Logger.log(typeof employee) ;    // พิมพ์ : object  
  
    Logger.log(employee.name) ;      // เข้าถึงข้อมูลใน Object วิธีที่ 1 - ผล Joe Bloggs  
    Logger.log(employee['name']) ;   // เข้าถึงข้อมูลใน Object วิธีที่ 2 - ผล Joe Bloggs  
  
    Logger.log(employee.title) ;     // เข้าถึงข้อมูลข้างใน Object - ผล Data Analyst  
  
    Logger.log(employee.name_title()) ; // เรียก Method name_title()  
                                        // พิมพ์ : Joe Bloggs Data Analyst  
}
```

---

### 3.2. get, set และ Object method - Chrome V8

---

#### 3.2.ก.) get และ function ใน Object

---

JavaScript Object Accessors

[https://www.w3schools.com/js/js\\_object\\_accessors.asp](https://www.w3schools.com/js/js_object_accessors.asp)

2 โค้ดต่อไปนี้จะให้ผลเหมือนกัน ตัวหนึ่งใช้ **function** สร้าง Method ให้กับ Object อีกตัวหนึ่งใช้ **get** สร้าง Property ให้กับ Object

**get** เป็นของใหม่ ช่วยให้โค้ดซับซ้อนน้อยลง

โค้ดที่ 1

```
function objectFunction() {  
  var person = {  
    firstName : "John" ,  
    lastName : "Doe" ,  
    fullName : function() {  
      return this.firstName + " " + this.lastName ;  
    }  
  }  
  Logger.log(person.fullName()) ;           // John Doe  
}
```

โค้ดที่ 2

```
function objectGet() {  
  var person = {  
    firstName : "John" ,  
    lastName : "Doe" ,  
    get fullName() {  
      return this.firstName + " " + this.lastName ;  
    }  
  }  
  Logger.log(person.fullName) ;           // John Doe  
}
```

### 3.2.ข.) set

ตัวอย่าง

```
function objectSet() {  
  var person = {  
    firstName : "John" ,  
    lastName : "Doe" ,  
    language : "" ,  
    set lang(lang) { this.language = lang.toUpperCase() ; }  
  } ;  
  person.lang = "en" ;  
  Logger.log(person.language) ;           // John Doe  
}
```

### 3.3. เพิ่ม, ลบ, แทรก สมาชิกให้ Object

JavaScript Object Properties

[https://www.w3schools.com/js/js\\_object\\_properties.asp](https://www.w3schools.com/js/js_object_properties.asp)

การเพิ่ม ลบ แทรก item ลงในตัวแปร Object สามารถทำได้ดังนี้

```
function objectFunction2() {  
  
  var employee = {  
    name: 'Joe Bloggs',  
    age: 25,  
    title: 'Data Analyst'  
  }  
  
  // 1. เพิ่ม item =====  
  
  employee['department'] = 'Web Analytics' ;  
  
  // หรือใช้แบบนี้ก็ได้  
  employee.department = 'Web Analytics' ;  
  
  Logger.log(employee);  
  // พิมพ์ : {name=Joe Bloggs, title=Data Analyst, department=Web Analytics, age=25.0}  
  
  // 2. ลบ item =====  
  
  delete employee.age ;  
  
  Logger.log(employee) ;  
  // พิมพ์ : {name=Joe Bloggs, title=Data Analyst, department=Web Analytics}  
  
}
```

### 3.4. Object Constructors - Chrome V8

JavaScript Object Constructors

[https://www.w3schools.com/js/js\\_object\\_constructors.asp](https://www.w3schools.com/js/js_object_constructors.asp)

Object Constructors ใช้สร้าง Blueprint เพื่อสร้าง Objects หลากยัวตัวที่มีชนิดเดียวกัน เช่น วัตถุพนักงาน ที่มี Properties และ Methods ข้างในเหมือนกัน

ข้างต้นสามารถทำได้โดยใช้ ฟังก์ชัน Object constructor

ตัวอย่างต่อไปนี้ ฟังก์ชัน Person เป็น ฟังก์ชัน Object constructor

```
function Person(first, last, age, eye) {  
  this.firstName = first ;  
  this.lastName = last ;  
  this.age = age ;  
  this.eyeColor = eye ;  
}
```

เราสามารถสร้างวัตถุตัวใหม่ชนิดเดียวกับ Object Person ได้โดยใช้คีย์เวิร์ด **new** เช่น

```
var MyFather = new Person("John", "Doe", 50, "blue") ;
```

### 3.5. Object Prototypes

JavaScript Object Prototypes

[https://www.w3schools.com/js/js\\_object\\_prototypes.asp](https://www.w3schools.com/js/js_object_prototypes.asp)

The JavaScript this Keyword

[https://www.w3schools.com/js/js\\_this.asp](https://www.w3schools.com/js/js_this.asp)

**Prototype** ใช้เพิ่ม Properties และ Methods ให้กับ Object constructor

```
// ฟังก์ชัน Object constructor
function employee(name, jobtitle, born) {
  this.name = name ;
  this.jobtitle = jobtitle ;
  this.born = born ;
}

function myFunction() {

  employee.prototype.salary = 2000 ;
  employee.country = 'Thailand' ;    // ไม่มีผล

  var fred = new employee("Fred Flintstone", "Caveman", 1970) ;
  Logger.log(fred) ;    // ดูผลที่ Logs ----- >

  var wasan = new employee("Wasan Kds", "Instructor", 1977) ;
  Logger.log(wasan) ;    // ดูผลที่ Logs ----- >
}
```

ผล

Logs

```
[ 01 ] { jobtitle=Caveman, born=1970.0, name=Fred Flintstone, salary=2000.0 }
[ 02 ] { jobtitle=Instructor, born=1977.0, name=Wasan Kds, salary=2000.0 }
```

---

### 3.6. Object Method key()

---

JavaScript ES5 Object Methods

[https://www.w3schools.com/js/js\\_object\\_es5.asp](https://www.w3schools.com/js/js_object_es5.asp)

ECMAScript 6 - ECMAScript 2015

[https://www.w3schools.com/js/js\\_es6.asp](https://www.w3schools.com/js/js_es6.asp)

key() ใช้จับ Keys ของ Object ใส่อาเรย์

ตัวอย่าง

```
function ex(){  
  var instrutor = {  
    fname: "Wasan" ,  
    lname: "Khunnsdiloksawet" ,  
    email: "wasankds@gmail.com",  
    zip: 67140  
  } ;  
  
  var keys = Object.key(instrutor) ;  
  Logger.log(keys) ;           // ดูผลที่ Logs --- >  
}
```

ผล

Logs

[ ] [fname, lname, email, zip]

# บทที่ 4

## เงื่อนไขและรูปแบบ



## 4.1. if – else if – else

JavaScript if else and else if

[https://www.w3schools.com/js/js\\_if\\_else.asp](https://www.w3schools.com/js/js_if_else.asp)

if-else if-else ใช้สร้างทางเลือกในการทำโค้ดตามเงื่อนไข เช่น ถ้าค่าเป็น + ทำโค้ดในบล็อกนี้ แต่ถ้าเป็น - ทำอีกบล็อกหนึ่ง เป็นต้น

### 4.1.ก.) โครงสร้างของ if

```
if (condition) {           // condition เป็น บูลีนคินค่า true/false
  // โค้ด – ทำโค้ดในบล็อกนี้ หากเงื่อนไขเป็น true ;
}
```

ตัวอย่าง

```
function myFunction() {
  var num = 51;
  if(num > 50 ) {           // ถ้ามากกว่า 50
    Logger.log("คะแนนของคุณคือ " + num + " : ผ่าน");      // ทำบรรทัดนี้
  } // End if
}
```

### 4.1.ข.) โครงสร้างของ if – else

```
if (condition) {
  // โค้ด หากเงื่อนไข condition เป็น true ;
} else {
  // โค้ด หากเงื่อนไข condition เป็น false ;
}
```

ตัวอย่าง

```
function myFunction() {
  var num = 49.5;
  if(num > 50 ) {
    Logger.log("คะแนนของคุณคือ " + num + " : ผ่าน") ;
  } else {
    Logger.log("คะแนนของคุณคือ " + num + " : ตก") ;      // ทำบรรทัดนี้
  } // End if
}
```



#### 4.1.ค.) โครงสร้างของ if – else if – else

```
if (condition1) {  
    // โค้ด หากเงื่อนไข condition1 เป็น true ;  
} else if (condition2) {  
    // โค้ด หากเงื่อนไข condition1 เป็น false แต่ condition2 เป็น true ;  
} else {  
    // โค้ด หากเงื่อนไข condition1 เป็น false แต่ condition2 เป็น false ;  
}
```

ตัวอย่าง - else if ใช้ซ้อนกันได้หลายตัว

```
function myFunction() {  
    var num = 62;  
    if(num > 80 ) {  
        Logger.log("คะแนนของคุณคือ " + num + " : A");  
    } else if (num > 70 ){  
        Logger.log("คะแนนของคุณคือ " + num + " : B");  
    } else if (num > 60 ){  
        Logger.log("คะแนนของคุณคือ " + num + " : C");    // ทำบรรทัดนี้  
    } else {  
        Logger.log("คะแนนของคุณคือ " + num + " : F");  
    }  
}
```

#### 4.2. switch

JavaScript Switch Statement

[https://www.w3schools.com/js/js\\_switch.asp](https://www.w3schools.com/js/js_switch.asp)

switch ใช้งานคล้าย if-else if-else ใช้ทำบล็อกของโค้ดตามเงื่อนไข

โครงสร้างของ switch

expression เป็นตัวที่ใช้ประเมินว่าจะไปลง case ไหน เช่นถ้า expression คำนวณกลับมาเป็น 2 จะไปทำ case 2 : (โค้ด) เป็นต้น

หลังจบ case จะปิด case ด้วย คำสั่ง break ;

```
switch(expression) {  
    case x :  
        // code block  
        break ;  
    case y :  
        // code block  
        break ;  
    default :  
        // ถ้าไม่มีที่ตรงกับ Case รหัสที่บล็อกนี้  
}
```

ตัวอย่าง

```
function swtichTest() {  
  var myVar ;  
  
  switch (2) {  
    case 0 : myVar= "ศูนย์" ;      break ;  
    case 1 : myVar= "หนึ่ง" ;      break ;  
    case 2 : myVar= "สอง" ;      break ;  
    default : myVar= "ไม่รู้" ;  
  }  
  Logger.log(myVar) ;  
}
```

ตัวอย่าง

```
function swtichTest() {  
  var day ;  
  
  switch (new Date().getDay()) { // คืนค่ากลับมาเป็น 0-6  
    case 0 : day= "Sunday" ;      break ;  
    case 1 : day= "Monday" ;      break ;  
    case 2 : day= "Tuesday" ;      break ;  
    case 3 : day= "Wednesday" ;    break ;  
    case 4 : day= "Thursday" ;     break ;  
    case 5 : day= "Friday" ;       break ;  
    case 6 : day= "Saturday" ;  
  }  
  Logger.log(day) ;  
}
```

### 4.3. for

JavaScript For Loop

[https://www.w3schools.com/js/js\\_loop\\_for.asp](https://www.w3schools.com/js/js_loop_for.asp)

for เป็นคำสั่งลูป ใช้วนรอบเพื่อทำโค้ดซ้ำๆ

โครงสร้าง

```
for (statement 1 ; statement 2 ; statement 3) {  
  // โค้ด  
}
```

**Statement 1** : ทำครั้งเดียว ก่อนเข้าไปทำโค้ดในบล็อก for

**Statement 2** : เงื่อนไขที่จะเข้าไปทำโค้ดในบล็อก for – ถ้ายังจริงทำต่อไปเรื่อยๆ

**Statement 3** : ทำหลังจากโค้ดในบล็อก for ทำแล้ว จะมาทำ statement3 ต่อ แล้วไปตรวจสอบเงื่อนไขใน statment2 ว่ายังจริงอยู่หรือไม่

## ตัวอย่างที่ 1

ทำโค้ดครั้งแรก ก็คือ  $i = 0$ , ครั้งที่ 2  $i = 1$ , ครั้งที่ 3  $i = 2$  และเมื่อ  $i = 3$  อยู่นอกเงื่อนไขใน statement2 จึงออกจากลูป for ไปโดยไม่ทำโค้ดเมื่อ  $i = 3$

```
function forTest1() {
  var text = "";
  for (i = 0 ; i < 3 ; i++) {
    text += "The number is " + i + "<br>" ;
    // หรือ text = text + "The number is " + i + "<br>" ;
  } // End for

  Logger.log(text);
  // พิมพ์ : The number is 0<br>The number is 1<br>The number is 2<br>
  //      i = 0          i = 1          i = 2
}
```

หมายเหตุ 1 : เครื่องหมาย += ก็คือ เอาข้อความด้านขวามาใส่ในตัวแปร Text ด้านซ้าย ลูป for จะทำให้ข้อความสะสมกันไปเรื่อยๆ

หมายเหตุ 2 :  $i++$  ก็คือ  $i = i + 1$  เป็นการเพิ่มค่าให้  $i$  ทีละ 1

วิธีการใช้ลูป for พลิกแพลงได้มาก เช่น ใส่หลายคำสั่งลงใน statement เดียว หรือ เว้นบาง statement ก็ได้ เป็นต้น

## ตัวอย่างที่ 2

statement1 มีการกำหนดค่าของตัวแปรหลายตัว ขึ้นด้วย " , " (คอมม่า) สังเกตว่าจบ statement1 ด้วยเครื่องหมาย " ; " (เซมิคอลอน)

```
function forTest2() {
  var cars = ["BMW", "Volvo", "Saab", "Ford"] ;
  var i, len, text ;

  for (i = 0, len = cars.length, text = "" ; i < len ; i++) {
    //      statement1

    text += cars[i] + "<br>";
  } // End for

  Logger.log(text) ;    // พิมพ์ : BMW<br>Volvo<br>Saab<br>Ford<br>
}
```

### ตัวอย่างที่ 3

ไม่ระบุ statement 1 และ 3 แต่ใช้วิธีกำหนดไว้นอก for หรือ โค้ดด้านใน for

```
function forLoop() {  
  
    var cars = ["BMW" , "Volvo" , "Saab" , "Ford" ] ;  
    var i = 0 , text = "" ;  
    var len = cars.length ;  
  
    for ( ; i < len ; ) {  
        text += cars[i] + "<br>" ;  
        i++ ;  
    } // End for  
  
    Logger.log(text);  
    // พิมพ์ : BMW<br>Volvo<br>Saab<br>Ford<br>  
  
}
```

---

#### 4.4. for in

---

JavaScript For Loop

[https://www.w3schools.com/js/js\\_loop\\_for.asp](https://www.w3schools.com/js/js_loop_for.asp)

for in ใช้กับตัวแปร Object เพื่อวนลูป ส่ง key ทีละตัว เข้าไปเป็นวัตถุติดให้กับโค้ดใน for

### ตัวอย่างที่ 1

```
function forIn_1() {  
  
    var oPerson = {  
        fname : "John",  
        lname : "Doe",  
        age : 25  
    } ;  
  
    var tText = "" ;  
    var x ;  
  
    for (x in oPerson) {  
        tText += oPerson[x] + " " ;  
    } // End - For/In  
  
    Logger.log(tText) ;    // พิมพ์ : John Doe 25  
  
}
```

## ตัวอย่างที่ 2

เอาตัวอย่างที่ 1 มาปรับแก้ อธิบายให้กระจ่างยิ่งขึ้นว่า **x** คือค่าอะไร - **x** ก็คือ **key** ที่ถูกส่งเข้าไป

```
function forIn_2() {  
    var oPerson = {  
        fname : "John",  
        lname : "Doe",  
        age : 25  
    };  
  
    var tText = "" ;  
    var x ;  
  
    for (x in oPerson) {  
        tText += oPerson[x]+" ";  
        Logger.log(x) ;           // x คือ Key ของ Object  
        Logger.log(tText) ;       // tText เก็บ Value ของ Key  
    } // End For In  
}
```

ผล

```
Logs  
[01] fname // Key 1  
[01] John // Value 1  
  
[01] lname // Key 2  
[01] Doe // Value 2  
  
[01] age // Key 3  
[01] 25 // Value 3
```

## 4.5. while

JavaScript While Loop

[https://www.w3schools.com/js/js\\_loop\\_while.asp](https://www.w3schools.com/js/js_loop_while.asp)

While เป็นคำสั่งลูปเช่นเดียวกับ for โดยจะทำโค้ดที่อยู่ใน while ไปเรื่อยๆ ตราบที่เงื่อนไขของ while เป็นจริง โครงสร้างการใช้งานดังต่อไปนี้

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

ตัวอย่างที่ 1

```
function while_1() {  
  var text = "" ;  
  var i = 0 ;  
  while ( i < 10 ) {  
    text += "The number is " + i + " ." ;  
    i++;  
  } // End while  
  Logger.log(text) ;  
}
```

ผล

#### Logs

[01] The number is 0 .The number is 1 .The number is 2 .The number is 3 .The number is 4 .The number is 5 .The number is 6 .The number is 7 .The number is 8 .The number is 9 .

---

## 4.6. do while

---

**do while** จะทำโค้ดที่อยู่ในบล็อก 1 ครั้งก่อน จากนั้นจึงเช็คเงื่อนไขที่ **while** และจะทำโค้ดที่อยู่ในบล็อกไปเรื่อยๆ ตราบที่เงื่อนไขเป็นจริง

โครงสร้างการใช้งาน **do while** ดังต่อไปนี้

```
do {  
  // โค้ดที่จะทำ  
} while (condition)
```

ตัวอย่าง

```
function DoWhile_1() {  
  var text = "" ;  
  var i = 11 ;  
  do {  
    text += "The number is " + i + " ." ;  
    i-- ;  
  } while(i >= 0) ;  
  Logger.log(text) ;  
}
```

ผล

#### Logs

[01] The number is 11. The number is 10. The number is 9. The number is 8. The number is 7. The number is 6. The number is 5. The number is 4. The number is 3. The number is 2. The number is 1. The number is 0.

## 4.7. try catch

### 4.7.ก.) try catch

JavaScript Errors - Throw and Try to Catch

[https://www.w3schools.com/js/js\\_errors.asp](https://www.w3schools.com/js/js_errors.asp)

**try** ใช้ทดสอบโค้ดขณะประมวลผล

**catch** ใช้จัดการกับ Error กรณีบล็อก **try** เกิด Error

```
try {  
    // Block of code to try  
}  
catch(err) {  
    // Block of code to handle errors  
}
```

ตัวอย่าง

```
function myFunction() {  
    adddler("Welcome guest!"); // ไม่มีฟังก์ชันชื่อ adddler  
                                // เกิด Error ระบบไม่ไปต่อ  
}
```

แก้เป็นแบบนี้ เพื่อป้องกันการเกิด Error แล้วโปรแกรมค้าง

```
function myFunction() {  
    try {  
        adddler("Welcome guest!");  
    }  
    catch(err) { // Error ถูกส่งมาให้ catch  
        Logger.log(err); // จับ Error มา Logs ดู  
                        // ---> ออกจาก Catch แล้วทำได้ต่อไป  
    }  
    Logger.log('Keep going');  
}
```

ผล

```
Logs  
[ ] ReferenceError: "adddler" is not defined.  
[ ] Keep going
```

## 4.7.ข.) throw

**throw** ใช้ในบล็อก **try** ใช้สร้าง Error เอง เหมาะมากกับการนำไปทำ Validation

```
// รับค่า x 5-10 เท่านั้น ถ้าใส่เป็นอย่างอื่นจะ catch error ออกมา

function myFunction() {

  var x = 'dddd' ;

  try {
    if(x == "")    throw "empty" ;           // ถ้า x เป็นค่าว่าง
    if(isNaN(x))   throw "not a number" ;     // ถ้า x ไม่ใช่ตัวเลข

    x = Number(x) ;                          // แปลง x เป็นตัวเลข
    if(x < 5)      throw "too low" ;          // ถ้า x น้อยกว่า 5
    if(x > 10)     throw "too high" ;         // ถ้า x มากกว่า 10
  }
  catch(err) {
    Logger.log("Input is " + err) ;           // Input is not a number
  }
}
```

ผล

Logs

[ ] Input is not a number

## 4.7.ค.) finally

**finally** ใช้ทำได้ต่อจาก **try** และ **catch**

```
// รับค่า x 5-10 เท่านั้น ถ้าใส่เป็นอย่างอื่นจะ catch error ออกมา

function myFunction() {

  var x = 'dddd' ;

  try {
    if(x == "")    throw "empty" ;
    if(isNaN(x))   throw "not a number" ;
    x = Number(x) ;
    if(x < 5)      throw "too low" ;
    if(x > 10)     throw "too high" ;
  }
  catch(err) {
    Logger.log("Input is " + err) ; // Input is not a number
  }
  finally {
    x = 5 ;
    Logger.log(x) ;
  }
}
```



ผล

Logs

[ ] Input is not a number

[ ] 5.0



# บทที่ 5 อาเรย์



## 5.1. อาร์เรย์คืออะไร

อาร์เรย์(Array) เป็นข้อมูลแบบก๊อ่น มีสมาชิกในก๊อ่นข้อมูลได้หลายตัว เก็บข้อมูลแบบมีมิติ เป็นเส้น(1 มิติ) เป็นตาราง(2 มิติ) เป็นต้น

อาร์เรย์ใช้บ่อยมากในการเขียนโค้ด ผู้เขียนจึงแยกออกมาเป็นบทใหญ่

## 5.2. พื้นฐานข้อมูลชนิดอาร์เรย์

สมาชิกในอาร์เรย์เก็บไว้ในวงเล็บ [ ] เช่น

```
var fruit = [ 'Apple' , 'Banana' , 'Pear' , 'Strawberry' ] ;
```

ข้างต้นเป็น อาร์เรย์ 1 มิติ หรือ อาร์เรย์แบบเส้น 1 แถว

เราสามารถอ้างอิงตำแหน่งการเก็บข้อมูลโดยระบุตัวเลขลงไปใน [ ] หรือ เรียกว่าดัชนีตำแหน่ง เพื่อดึงสมาชิกในอาร์เรย์ออกมาใช้ เช่น `fruitsArray[0]` ก็คือ `Apple`

ตัวอย่าง

```
function arrayFunction() {  
    // สร้างตัวแปรอาร์เรย์ว่างๆ  
    var newArray = [] ;  
  
    // ใส่ข้อมูลลงในอาร์เรย์      0      1      2      3  
    var fruitsArray = [ 'Apple' , 'Banana' , 'Pear' , 'Strawberry' ] ;  
  
    Logger.log(fruitsArray) ;    // พิมพ์ : [Apple, Banana, Pear, Strawberry]  
  
    // ลำดับข้อมูลในอาร์เรย์เริ่มจาก 0  
    // เข้าถึงข้อมูลในอาร์เรย์ได้โดยใช้สัญลักษณ์ดังนี้  
  
    Logger.log(fruitsArray[0]) ;    // พิมพ์ : Apple  
    Logger.log(fruitsArray[1]) ;    // พิมพ์ : Banana  
    Logger.log(fruitsArray[2]) ;    // พิมพ์ : Pear  
    Logger.log(fruitsArray[3]) ;    // พิมพ์ : Strawberry  
    Logger.log(fruitsArray[4]) ;    // พิมพ์ : undefined (ไม่มีอะไรที่ดัชนีลำดับที่ 4)  
}
```

## 5.3. อาร์เรย์ 2 มิติ

อาร์เรย์ 2 มิติ ในระบบของ Google Apps Script เราจะเขียนในลักษณะ อาร์เรย์แบบเส้นในอาร์เรย์แบบเส้น ซึ่งสามารถเขียนได้ดังนี้ [ [ , ] , [ , ] , [ , ] ]

### 5.3.ก.) การเขียนอาเรย์ 2 มิติ

ยกตัวอย่าง

```
var Arr2dim = [ ["Linda",27] , ["Lisa",35] , ["John",42] ] ;
```

หรือเขียนแบบนี้ เพื่อให้เข้าใจว่ามีกี่แถว กี่คอลัมน์

```
var Arr2dim = [ // คอลัมน์ที่ 1    // คอลัมน์ที่ 2
                [ "Linda" ,      27 ] , // แถวที่ 1
                [ "Lisa" ,       35 ] , // แถวที่ 2
                [ "John" ,       42 ]   // แถวที่ 3
            ] ;
```

อาเรย์ข้างต้นมี 3 แถว 2 คอลัมน์

### 5.3.ข.) การเข้าถึงสมาชิกในอาเรย์ 2 มิติ

`Arr2dim[0]` = อาเรย์เส้นแรก ก็คือ `["Linda",27]`

`Arr2dim[0][0]` = อาเรย์เส้นแรก สมาชิกตัวแรก ก็คือ `Linda`

```
function arrayFunction() {
    var Arr2dim = [
        ["Linda", 27] , // Arr2dim[0] ก็คือ อาเรย์เส้นนี้
        ["Lisa", 35] , // Arr2dim[1] ก็คือ อาเรย์เส้นนี้
        ["John", 42]   // Arr2dim[2] ก็คือ อาเรย์เส้นนี้
    ] ;
    Logger.log(Arr2dim); // พิมพ์ : [ ["Linda",27] , ["Lisa",35] , ["John",42] ]
    Logger.log(Arr2dim[0]); // พิมพ์ : ["Linda",27]
    Logger.log(Arr2dim[0][0]); // พิมพ์ : Linda
}
```

### 5.3.ค.) การนับจำนวนสมาชิกในอาเรย์

การนับจำนวนสมาชิกในอาเรย์ใช้ ทำได้โดยใช้ Method `length` ของอาเรย์

ตัวอย่าง - กรณี 1 มิติ

```
var fruitsArray = [ 'Apple' , 'Banana' , 'Pear' , 'Strawberry' ] ;
Logger.log(fruitsArray.length); // พิมพ์ : 4
```

ตัวอย่าง - กรณี 2 มิติ

```
function arrayFunction() {
    var Arr2dim = [ ["Linda", 27] ,
                    ["Lisa", 35] ,
                    ["John", 42] ] ;

    Logger.log(Arr2dim.length) ; // พิมพ์ : 3 - เส้นนอกมีสมาชิก 3
    Logger.log(Arr2dim[0].length) ; // พิมพ์ : 2 - เส้นใน(ตัวแรก - 0) มีสมาชิก 2 ตัว
    Logger.log(Arr2dim.length * Arr2dim[0].length) ; // พิมพ์ : 6 - สมาชิกทั้งหมด
}
```



# บทที่ 6

## อาเรย์เมกอด



## 6.1. push()

JavaScript Array push() Method

[https://www.w3schools.com/jsref/jsref\\_push.asp](https://www.w3schools.com/jsref/jsref_push.asp)

push() ใช้ใส่ข้อมูลลงในอาร์เรย์ ณ จุดสุดท้าย โดยเข้าไปเปลี่ยนแปลงอาร์เรย์ตั้งต้น และคืนค่ากลับมาเป็นจำนวนสมาชิกของอาร์เรย์

โครงสร้าง

```
array.push(item1, item2, ..., itemX)
```

พารามิเตอร์

Parameter	Description
item1, ..., itemX	(จำเป็น) สมาชิกใหม่ที่จะเพิ่มลงในอาร์เรย์

ตัวอย่าง

```
var fruits1 = [ "Banana" , "Orange" , "Apple" , "Mango" ] ;  
  
// push สมาชิกเข้าไป  
fruits1.push("Kiwi" , "Lemon" , "Pineapple") ;  
  
// Logs ดูอีกครั้ง จะพบว่าสมาชิกเปลี่ยนไป  
Logger.log(fruits1) ;           // [Banana, Orange, Apple, Mango, Kiwi, Lemon, Pineapple]  
  
var fruits2 = [ "Banana" , "Orange" , "Apple" , "Mango" ] ;  
  
// push สมาชิกเข้าไป - แต่ฝากไว้ในตัวแปรใหม่ด้วย  
var newfruits2 = fruits2.push("Kiwi" , "Lemon" , "Pineapple") ;  
  
// ตัวแปรใหม่เก็บ จำนวนสมาชิก ไม่ใช่อาร์เรย์ที่เปลี่ยนไป *****  
Logger.log(newfruits2) ;       // 7.0
```

## 6.2. unshift()

JavaScript Array unshift() Method

[https://www.w3schools.com/jsref/jsref\\_unshift.asp](https://www.w3schools.com/jsref/jsref_unshift.asp)

unshift() ใช้ใส่ข้อมูลลงในอาร์เรย์ ณ จุดแรก โดยเข้าไปเปลี่ยนแปลงอาร์เรย์ตั้งต้น และคืนค่ากลับมาเป็นจำนวนสมาชิกของอาร์เรย์

โครงสร้าง

```
array.unshift(item1, item2, ..., itemX)
```

พารามิเตอร์

Parameter	Description
item1, ..., itemX	(จำเป็น) สมาชิกใหม่ที่จะเพิ่มลงในอาร์เรย์



ตัวอย่าง

```
var fruits = [ "Banana" , "Orange" , "Apple" , "Mango" ] ;
fruits.unshift("Lemon" , "Pineapple") ;

Logger.log(fruits) ;           // [ Lemon, Pineapple, Banana, Orange, Apple, Mango ]

var fruits2 = [ "Banana" , "Orange" , "Apple" , "Mango" ] ;
var newfruits2 = fruits2.unshift("Lemon" , "Pineapple") ;

Logger.log(newfruits2) ;      // 6.0
```

### 6.3. pop()

JavaScript Array pop() Method

[https://www.w3schools.com/jsref/jsref\\_pop.asp](https://www.w3schools.com/jsref/jsref_pop.asp)

pop() ใช้ลบข้อมูลในอาเรย์ ตัวท้ายสุด และคืนค่ากลับมาเป็นข้อมูลตัวนั้น

โครงสร้าง

```
array.pop()
```

ตัวอย่าง

```
var fruits1 = [ "Banana" , "Orange" , "Apple" , "Mango" ] ;
fruits1.pop() ;

Logger.log(fruits1) ;           // [ Banana , Orange , Apple ]

var fruits2 = [ "Banana" , "Orange" , "Apple" , "Mango" ] ;
var newfruits2 = fruits2.pop() ;

Logger.log(newfruits2) ;       // Mango
```

### 6.4. shift()

JavaScript Array shift() Method

[https://www.w3schools.com/jsref/jsref\\_shift.asp](https://www.w3schools.com/jsref/jsref_shift.asp)

shift() ใช้ลบข้อมูลในอาเรย์ ตัวแรก และคืนค่ากลับมาเป็นข้อมูลตัวที่ถูกลบออก

โครงสร้าง

```
array.shift()
```

ตัวอย่าง

```
var fruits = [ "Banana" , "Orange" , "Apple" , "Mango" ] ;
fruits.shift() ;

Logger.log(fruits1) ;           // [ Orange, Apple, Mango ]

var fruits2 = [ "Banana" , "Orange" , "Apple" , "Mango" ] ;
var newfruits2 = fruits2.shift() ;

Logger.log(fruits) ;           // Banana
```

## 6.5. splice()

JavaScript Array splice() Method

[https://www.w3schools.com/jsref/jsref\\_splice.asp](https://www.w3schools.com/jsref/jsref_splice.asp)

splice() ใช้เพิ่มหรือลบ สมาชิกในอาร์เรย์ แบบกำหนดตำแหน่งได้ โดยเข้าไปเปลี่ยนแปลงอาร์เรย์ตั้งต้น และ คืนค่ากลับมาเป็นสมาชิกที่ถูกลบ ( กรณีคืนค่า อันนี้คนเขียนก็งง แปลตามเอกสาร แต่ทดลองแล้ว ไม่ได้ตามนั้น )

โครงสร้าง

```
array.splice(index, howmany, item1, ....., itemX)
```

พารามิเตอร์

Parameter	Description
index	(ต้องการ) เลข Integer ระบุตำแหน่งที่จะลบหรือแทรกสมาชิก ใช้เลขลบ เพื่อระบุตำแหน่งจากท้าย
howmany	(ไม่จำเป็น) จำนวนสมาชิกที่จะลบออก (0 = ไม่ลบ)
item1, ..., itemX	(ไม่จำเป็น) สมาชิกใหม่ที่จะเพิ่มลงในอาร์เรย์ (ไม่ระบุ = ไม่เพิ่ม)

ตัวอย่างที่ 1 – หากไม่ระบุ item จะเป็นการลบออก ตามตำแหน่งและจำนวนที่ระบุ

```
//                                ลบ      ลบ
var fruits1 = [ "Banana", "Orange", "Apple", "Mango", "Kiwi" ];
fruits1.splice(2, 2);
Logger.log(fruits1);           // [ Banana, Orange, Kiwi ]
var fruits2 = [ "Banana", "Orange", "Apple", "Mango", "Kiwi" ];
var newfruits2 = fruits1.splice(2, 2);
Logger.log(newfruits2);        // [Kiwi] - ??? งงกันไป
```

ตัวอย่างที่ 2 – ระบุพารามิเตอร์ item ด้วย ก็คือ สมาชิกที่จะเพิ่มลงไป

```
//              0          1          2          3
var fruits1 = [ "Banana", "Orange", "Apple", "Mango" ];
// เริ่มดรรชนีตำแหน่งที่ 2 ลบออก 1 ตัว >> เพิ่มลงไป 2 ตัวก็คือ "Lemon", "Kiwi"
fruits1.splice(2, 1, "Lemon", "Kiwi");
Logger.log(fruits1);           // [ Banana, Orange, Lemon, Kiwi, Mango ]

var fruits2 = [ "Banana", "Orange", "Apple", "Mango" ];
// คืนค่าเป็น แล้วจับใส่ตัวแปรตัวใหม่
var newfruits2 = fruits1.splice(2, 1, "Lemon", "Kiwi");
Logger.log(newfruits2);        // [ Lemon ] - ??? งงกันไป
```

## 6.6. slice()

JavaScript Array Methods - Slicing an Array

[https://www.w3schools.com/js/js\\_array\\_methods.asp](https://www.w3schools.com/js/js_array_methods.asp)

slice() ใช้ตัดสมาชิกบางส่วนของอาร์เรย์ ไปสร้างอาร์เรย์ใหม่

โครงสร้างการใช้งาน

```
array.slice(start, end)
```

พารามิเตอร์

Parameter	Description
start	(Optional) เลข integer ที่ใช้ระบุตำแหน่งเริ่มการเลือก ( - ตำแหน่งแรก เริ่มจาก 0 - ใช้เลขลบ เพื่อเลือกจากด้านหลัง - หากปล่อยว่างจะหมายถึง 0 )
end	(Optional) เลข integer ที่ใช้ระบุตำแหน่งจบการเลือก แต่ไม่รวมตัวนี้ ( - หากปล่อยว่างไว้ จะเริ่มจาก start ไปจนถึงตัวสุดท้าย - ใช้เลขลบ เพื่อเลือกจากด้านหลัง )

ตัวอย่าง

```
//           0       1       2       3       4
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"] ;
// เริ่มจากดัชนีตำแหน่งที่ 3 ไปจนจบ
var citrus = fruits.slice(3) ;                               // [ Apple, Mango ]
```

ตัวอย่างที่ 2

```
//           0       1       2       3       4
var fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"] ;
// เริ่มจากดัชนีตำแหน่งที่ 1 ไปจนถึงก่อนตัวที่ 3
var citrus = fruits.slice(1, 3) ;                             // ["Orange", "Lemon"]
```

## 6.7. ตัวอย่างการใช้งานอาเรย์เมธอดกับอาเรย์ 2 มิติ

อาเรย์ 2 มิติยังใช้ push, shift ได้ตามปกติ

ตัวอย่าง

```
var Arr2dim = [ ["Linda", 27] ,
                ["Lisa", 35] ,
                ["John", 42] ] ;

Arr2dim.push(["Wasan",42]) ;
Logger.log(Arr2dim) ;           // พิมพ์ : [ [Linda,27.0], [Lisa,35.0], [John,42.0], [Wasan, 42.0] ]

Arr2dim.pop() ;
Logger.log(Arr2dim) ;           // พิมพ์ : [ [Linda,27.0], [Lisa,35.0], [John,42.0] ]

Arr2dim[0].push("179") ;
Logger.log(Arr2dim) ;           // พิมพ์ : [ [Linda, 27.0, 179], [Lisa, 35.0], [John, 42.0] ]

// เปลี่ยนค่า
Arr2dim[0][1] = 53 ;
Logger.log(Arr2dim) ;           // พิมพ์ : [ [Linda, 53.0, 179], [Lisa, 35.0], [John, 42.0] ]
```

## 6.8. join()

JavaScript Array join() Method

[https://www.w3schools.com/jsref/jsref\\_join.asp](https://www.w3schools.com/jsref/jsref_join.asp)

ใช้รวมข้อมูลในอาเรย์ ให้กลายเป็นข้อมูล String (ข้อความ) ก้อนเดียวที่ขึ้นด้วยเครื่องหมาย , (คอมม่า) หรือกำหนดตัวขึ้นได้เอง

โครงสร้าง

```
array.join(separator)
```

พารามิเตอร์

Parameter	Description
separator	(เว้นว่างได้) ตัวขึ้น ถ้าปล่อยว่างไว้ จะใช้ตัวขึ้นเป็นคอมม่า

ตัวอย่าง

```
function arrayFunction() {
    var arr = [ 'Apple' , 'Banana' , 'Pear' , 'Strawberry' ] ;
    var joinArr = arr.join() ;
    Logger.log(joinArr) ;           // Apple,Banana,Pear,Strawberry

    var joinArr2 = arr.join(" - ") ;
    Logger.log(joinArr2) ;           // Apple - Banana - Pear - Strawberry
}
```

## 6.9. concat()

JavaScript Array concat() Method

[https://www.w3schools.com/jsref/jsref\\_concat\\_array.asp](https://www.w3schools.com/jsref/jsref_concat_array.asp)

concat ใช้รวมอาร์เรย์เข้าด้วยกัน Method นี้ไม่เปลี่ยนแปลงอาร์เรย์เดิม แต่จะสร้างอาร์เรย์ใหม่ ฉะนั้นให้สร้างตัวแปรมารองรับผลการรวมอาร์เรย์ด้วย

โครงสร้างการใช้งาน

```
array1.concat(array2, array3, ..., arrayX)
```

พารามิเตอร์

Parameter	Description
array2, array3, ..., arrayX	Required. The arrays to be joined

ตัวอย่างที่ 1 - อาร์เรย์ 1 มิติ

```
function testConcat(){  
    var arr1dim = [37, 39, 42, 46, 11, 33] ;  
    var moreCcol = ['A', 'B'] ;  
  
    arr1dim = arr1dim.concat(moreCcol) ;  
  
    Logger.log(arr1dim) ; // [37.0, 39.0, 42.0, 46.0, 11.0, 33.0, A, B]  
}
```

ตัวอย่างที่ 2 - อาร์เรย์ 2 มิติ

ตัวอย่างนี้ใช้บ่อยมาก เมื่อเราจับข้อมูลจาก Google Sheet มาได้ จะได้อาร์เรย์ 2 มิติมา ซึ่งจากนั้น เราอาจจะลดหรือเพิ่มคอลัมน์เข้าไป

```
function testArrayB(){  
    var arr2dim = [ ['A', 39, 12000 ] ,  
                    ['B', 46, 42321 ] ,  
                    ['C', 65, 22520 ] ] ;  
  
    // เพิ่มสมาชิกต่อท้ายที่ละแถว  
    arr2dim = arr2dim.map(function(row){  
        return row.concat( ['More column'] ) ;  
    });  
    Logger.log(arr2dim) ;  
  
    /*      [ [A, 39.0, 12000.0, More column] ,  
              [B, 46.0, 42321.0, More column] ,  
              [C, 65.0, 22520.0, More column]      ] */  
}
```

## 6.10. indexOf()

JavaScript Array indexOf() Method

[https://www.w3schools.com/jsref/jsref\\_indexof\\_array.asp](https://www.w3schools.com/jsref/jsref_indexof_array.asp)

ใช้ค้นหาในอาร์เรย์ โดยส่งค่าที่จะค้นหาเข้าไป โดยจะคืนค่ากลับมาเป็นดัชนีตำแหน่งในอาร์เรย์ เป็นเลข 0,1,2,3 ... แต่ถ้าค่าที่ใช้นั้นหาไม่มีในอาร์เรย์ จะคืนค่า -1 กลับมา

โครงสร้างการใช้งาน

```
array.indexOf(item, start)
```

พารามิเตอร์

Parameter	Description
item	(ต้องการ) ข้อมูลที่ใช้ค้นหา
start	(ไม่จำเป็น) จุดเริ่มต้นค้นหา โดยค่าลบ จะเริ่มนับตำแหน่งและค้นหาจากด้านหลัง

ตัวอย่างที่ 1

```
function arrayFunction() {  
  var fruits = ["Banana", "Orange", "Apple", "Mango", "Banana", "Orange", "Apple"] ;  
               // 0         1         2         3         4         5         6  
  
  var a = fruits.indexOf("Apple") ;  
  Logger.log(a) ; // 2  
  
  var b = fruits.indexOf("Apple", 4) ; // เริ่มค้นหาที่ดัชนีตำแหน่งที่ 4 (Banana)  
  Logger.log(b) ; // 6  
  
  var c = fruits.indexOf("broccoli") ;  
  Logger.log(c) ; // -1  
}
```

ตัวอย่างที่ 2 - ข้อมูลเป็นเวลา ให้ใช้ `valueOf` แปลงเป็นมิลลิวินาทีก่อน เพราะรูปแบบการเขียนวันที่ในภาษาที่คนอ่านได้ มีหลากหลายมาก

```
function testIndexOf2() {  
  
  var day = new Date("Wed Feb 5 2020 00:00:00 GMT+0700 (Indochina Time)").valueOf()  
  Logger.log(day.toString()) ; // 1580835600000  
  
  var disableDays = [  
    new Date("Tue Feb 11 2020 00:00:00 GMT+0700 (Indochina Time)").valueOf(),  
    new Date("Wed Feb 12 2020 00:00:00 GMT+0700 (Indochina Time)").valueOf(),  
    new Date("Wed Feb 5 2020 00:00:00 GMT+0700 (Indochina Time)").valueOf()  
  ] ;  
  
  // มี day ใน disableDays สักตัวหรือไม่  
  var aaa = disableDays.indexOf(day.valueOf()) > -1 ;  
  Logger.log(aaa) ; // true  
}
```

## 6.11. lastIndexOf()

JavaScript Array 6.11. lastIndexOf() Method

[https://www.w3schools.com/jsref/jsref\\_lastindexof\\_array.asp](https://www.w3schools.com/jsref/jsref_lastindexof_array.asp)

คล้ายกับ indexOf() แต่กรณีที่เจอค่าที่ค้นหาหลายค่า จะคืนค่าตัวสุดท้ายในระดับที่พบกลับมา

โครงสร้างการใช้งาน

```
array.indexOf(item, start)
```

พารามิเตอร์

Parameter	Description
item	(ต้องการ) ข้อมูลที่ใช้ค้นหา
start	(ไม่จำเป็น) จุดเริ่มต้นค้นหา โดยค่าลบ จะเริ่มนับตำแหน่งและค้นหาจากด้านหลัง

ตัวอย่าง

```
function arrayFunction() {  
  var fruits = [ "Banana", "Orange", "Apple", "Mango", "Banana", "Orange", "Apple" ] ;  
              //   0         1       2       3       4       5       6  
  
  var a = fruits.lastIndexOf("Apple") ;  
  Logger.log(a) ; // 6  
  
  var b = fruits.lastIndexOf("Apple", 4) ; // เริ่มค้นหาที่ตรงตำแหน่งที่ 4 (Banana)  
  Logger.log(b) ; // 2 ( 6-4 )  
  
  var c = fruits.lastIndexOf("broccoli") ;  
  Logger.log(c) ; // -1  
}
```

## 6.12. reverse()

JavaScript Array reverse() Method

[https://www.w3schools.com/jsref/jsref\\_reverse.asp](https://www.w3schools.com/jsref/jsref_reverse.asp)

ใช้กลับตำแหน่งสมาชิกเอาเรย์จากหน้าไปหลัง หลังไปหน้า

โครงสร้างการใช้งาน

```
array.reverse()
```

ตัวอย่าง

```
var fruits = [ "Banana" , "Orange" , "Apple" , "Mango" ] ;  
var n = fruits.reverse() ;  
Logger.log(n) ; // [Mango, Apple, Orange, Banana]
```

## 6.13. includes() - Chrome V8

JavaScript Array includes() Method

[https://www.w3schools.com/jsref/jsref\\_includes\\_array.asp](https://www.w3schools.com/jsref/jsref_includes_array.asp)

ใช้ตรวจสอบว่าอาร์เรย์ มีสมาชิกต่อไปนี้หรือไม่ โดยทำงานแบบ Case-sensitive และจะคืนค่ากลับมาเป็นบูลีน

โครงสร้างการใช้งาน

```
array.includes(element, start)
```

พารามิเตอร์

Parameter	Description
<a href="#">element</a>	(Required) ข้อมูลที่ใช้ค้นหา
<a href="#">start</a>	(Optional) ตำแหน่งที่จะเริ่มค้นหา ค่าปรียายคือ 0

ตัวอย่าง

```
function example() {  
  var fruits = [ "Banana", "Orange", "Apple", "Mango" ] ;  
  var n = fruits.includes("Banana", 3) ;  
  Logger.log(n) ;      // false  
}
```



บทที่ 7  
อาเรย์เมกอด  
ตระกูลวนรูป



## 7.1. map()

JavaScript Arrays & Map Method Tutorial - Google Apps Script Part 4

[https://youtu.be/WA8QotNEVc4?list=PLv9Pf9aNgemvD9NFa86\\_uDt-NWh37efmD](https://youtu.be/WA8QotNEVc4?list=PLv9Pf9aNgemvD9NFa86_uDt-NWh37efmD)

map() ใช้ส่งสมาชิกทุกตัวในอาเรย์ ไปทำอะไรสักอย่างในฟังก์ชัน เช่น คำนวณ, แปะหน้าแปะหลังด้วยข้อความ

กรณีอาเรย์ 1 มิติ สมาชิกจะเป็นค่าใดๆ แต่หากเป็นอาเรย์ 2 มิติ สมาชิกในอาเรย์ก็คือ เส้นอาเรย์ด้านใน

### 7.1.ก.) map() และ อาเรย์ 1 มิติ

โค้ดด้านล่าง ส่งสมาชิกในอาเรย์ไปที่ละตัว เพื่อใส่วงเล็บครอบ

```
function mapMethod() {
  var arr1dim = [ "Sara" , 24 , "Chaingmai" ] ;

  // วนลูปส่งสมาชิกในอาเรย์ไปทำบางอย่างในฟังก์ชันที่ระบุ
  var new_arr1dim = arr1dim.map(addBrackets) ;

  Logger.log(new_arr1dim) ;    // พิมพ์ : [ (Sara), (24), (Chaingmai) ]
}

function addBrackets(item) {   // item คือ สมาชิกในอาเรย์ โดยจะวนลูปส่งมาทีละตัว
  return "(" + item + ")" ;    // ใส่วงเล็บครอบ
}
```

สามารถ เขียนโค้ดโดยเอาฟังก์ชันยัดลงไปใน map เลยก็ได้ เหมาะกับโค้ดในฟังก์ชันที่ไม่ยาวเกินไป

```
function mapMethod() {

  var arr1dim = [ "Sara" , 24 , "Chaingmai" ] ;

  var new_arr1dim = arr1dim.map(function addXs(item){
    return "X" + item + "X" ;    // ใส่ X หน้าและหลัง
  }) ; // จบ - map

  Logger.log(new_arr1dim);      // พิมพ์ : [ XSaraX, X24X, XChaingmaiX ]
}
```

## 7.1.ข.) map() และ อาร์เรย์ 2 มิติ

กรณีเป็นอาร์เรย์ 2 มิติ สมาชิกในอาร์เรย์ก็คือ **เส้นอาร์เรย์ด้านใน** ฉะนั้นการนำไปประมวลผลหรือการคืนค่ากลับมา ต้องทำแบบอาร์เรย์

**ตัวอย่างที่ 1** นำสมาชิกที่เป็นเส้นอาร์เรย์ ไปทำบางอย่างในฟังก์ชัน

```
function mapMethod() {
    var arr2dim = [
        [ "Linda", 27, "Bangkok" ] ,      // สมาชิกตัวที่ 1
        [ "Lisa", 35, "Nontaburi" ] ,      // สมาชิกตัวที่ 2
        [ "John", 42, "Petchaboon" ]      // สมาชิกตัวที่ 3
    ];

    var new_arr2dim = arr2dim.map(doSomething);
    Logger.log(new_arr2dim) ;              // ดูผลที่ Logs --- >
}

function doSomething(row) {
    return [ row[0] , row[1]+1 ,row[2] ] ;
}
```

ผล – อาร์เรย์ใหม่ มีตัวเลขเปลี่ยนไปจากอาร์เรย์ดั้งเดิม โดยสมาชิกที่เป็นตัวเลข ถูกเพิ่มค่าไป 1

### Logs

```
[01] [ [Linda, 28.0, Bangkok], [Lisa, 36.0, Nontaburi], [John, 43.0, Petchaboon] ]
```

**ตัวอย่างที่ 2** เปลี่ยนมิติการคืนค่า

```
function mapMethod() {
    var arr2dim = [
        [ "Linda", 27, "Bangkok" ] ,      // สมาชิกตัวที่ 1
        [ "Lisa", 35, "Nontaburi" ] ,      // สมาชิกตัวที่ 2
        [ "John", 42, "Petchaboon" ]      // สมาชิกตัวที่ 3
    ];

    var new_arr2dim = arr2dim.map(doSomething);
    Logger.log(new_arr2dim) ;
}

function doSomething(row) {
    var textOutput = row[0] + " is from " + row[2] ;
    var newAge = row[1] + 1 ;

    return [ textOutput , newAge ] ;
}
```

ผล – ได้อาร์เรย์ขนาด 3 แถว x2 คอลัมน์

### Logs

```
[ 01 ] [ [Linda is from Bangkok, 28.0] , [Lisa is from Nontaburi, 36.0] , [John is from Petchaboon, 43.0] ]
```

โค้ดข้างต้น จับฟังก์ชันยัดลงใน map ก็ได้ ได้ผลแบบเดียวกัน

```
function mapMethod() {
  var arr2dim = [
    ["Linda", 27, "Bangkok" ] ,      // สมาชิกตัวที่ 1
    ["Lisa", 35, "Nontaburi" ] ,     // สมาชิกตัวที่ 2
    ["John", 42, "Petchaboon" ]      // สมาชิกตัวที่ 3
  ];

  var new_arr2dim = arr2dim.map(function doSomething(row) {
    var textOutput = row[0] + " is from " + row[2] ;
    var newAge = row[1] + 1 ;
    return [textOutput , newAge] ;
  });
  Logger.log(new_arr2dim) ;
}
```

## 7.2. filter()

JavaScript Filter Method Tutorial - Google Sheets Apps Scripts - Array Methods Part 7

[https://youtu.be/PT\\_TDhMhWsE?list=PLv9Pf9aNgemvD9NFa86\\_uDt-NWh37efmD](https://youtu.be/PT_TDhMhWsE?list=PLv9Pf9aNgemvD9NFa86_uDt-NWh37efmD)

filter() การใช้งานตามชื่อ ใช้กรองข้อมูลในอาเรย์ โดยที่เราสามารถเขียนเงื่อนไขการกรองเป็นฟังก์ชันได้เอง

### 7.2.ก.) ตัวอย่างที่ 1

ตัวแปร filterLogic เก็บฟังก์ชัน ที่เป็นตัวประมวลผลการกรอง พร้อมกับรับค่ากลับมาด้วย

```
function filterArr() {

  var arr = [ "txt1" , 4 , 8 , 33 , 2 , "txt2" , 9 ] ;
  var new_arr = arr.filter(filterLogic) ;

  Logger.log(arr);      // พิมพ์ : ["txt1", 4, 8, 33, 2, "txt2", 9]
  Logger.log(new_arr);  // พิมพ์ : [8.0, 33.0, 9.0] – เฉพาะค่ามากกว่า 6 – ข้อความถูกตัดทิ้ง
} // End function

var filterLogic = function(item) {  // ฟังก์ชันที่ใช้กรอง

  return item > 6 ;                // คืนเฉพาะตัวที่มีค่ามากกว่า 6
} // End function
```

### 7.2.ข.) ตัวอย่างที่ 2 – ใส่ฟังก์ชันเป็น Argument

เพื่อให้ได้ดั่งนั้น และหากฟังก์ชันการกรองไม่ยาวมาก เอาฟังก์ชันที่เป็นตัวกรอง ยัดลงไปใน filter() ก็ได้

```
function filterArr() {  
    var arr1 = [ 1 , 2 , 3 , 4 , 5 , 7 ] ;  
    var fData = arr1.filter(function(item){  
        return !(item >= 4) ; // ตัวที่ไม่ >= 4  
    }) ;  
    Logger.log(fData);    // พิมพ์ : [1.0, 2.0, 3.0]  
} // End function
```

### 7.2.ค.) ตัวอย่างที่ 3 - กรองตัวเลข

```
function filterArr() {  
    var arr = [ "txt1" , 4 , 8 , 33 , 2 , "txt2" , 9 ] ;  
    var new_arr = arr.filter(filterLogic) ;  
  
    Logger.log(arr);    // พิมพ์ : [ "txt1" , 4 , 8 , 33 , 2 , "txt2" , 9 ]  
    Logger.log(new_arr);    // พิมพ์ : [ 8.0 , 33.0 , 9.0]  
                           // เฉพาะค่ามากกว่า 6 – ตัวที่เป็นข้อความถูกตัดทิ้ง  
} // End function  
  
var filterLogic = function(item) {  
    if(item>6)  
    {  
        return true ;    // คืนค่า true - เมื่อค่ามากกว่า 6  
    }  
    else  
    {  
        return false ;  
    } ; // End if  
} // End function
```

## 7.2.ง.) ตัวอย่างที่ 4 - กรองข้อความ

ข้อมูลในอาเรย์มีทั้งตัวเลข และ ข้อความปะปนกันอยู่ ตัวอย่างนี้ทดสอบกรองข้อความ

```
function filterArr() {  
    var arr = [ "txt1" , 4 , 8 , 33 , 2 , "txt2" , 9 ] ;  
    var new_arr = arr.filter(filterLogic) ;  
  
    Logger.log(arr);           // พิมพ์ : [ "txt1" , 4 , 8 , 33 , 2 , "txt2" , 9 ]  
    Logger.log(new_arr);       // พิมพ์ : [ 4.0 , 8.0 , 33.0 , 2.0 , txt2 , 9.0 ]  
                                // อะไรที่ไม่ใช่ txt1 มาหมด  
} // End function  
  
var filterLogic = function(item) {  
    if(item == "txt1" )  
    {  
        return false ;  
    }  
    else  
    {  
        return true ;         // คืนค่า true – อะไรที่ไม่ใช่ "txt1" มาหมด  
    } ; // End if  
} // End function
```

## 7.3. sort()

JavaScript Arrays Sort Method Tutorial - Google Sheets Apps Scripts - Array Methods Part 9  
[https://youtu.be/hPCIOohFOFg?list=PLv9Pf9aNgemvD9NFa86\\_udt-NWh37efmD](https://youtu.be/hPCIOohFOFg?list=PLv9Pf9aNgemvD9NFa86_udt-NWh37efmD)

sort() เป็น Method ของอาเรย์ที่ใช้เรียงสมาชิกในอาเรย์

### 7.3.ก.) ข้อมูลเป็นตัวเลขทั้งหมด

ตัวอย่างที่ 1 - เรียงโดยไม่ใส่ Argument ใดๆ ลงไปใน sort()

```
function sortArr() {  
    var arr1 = [4,2,5,6,5,2];  
    var arr2 = [4,2,5,6,5,2,44,9,56];  
  
    Logger.log(arr1.sort()); // พิมพ์ : [2.0, 2.0, 4.0, 5.0, 5.0, 6.0]  
    Logger.log(arr2.sort()); // พิมพ์ : [2.0, 2.0, 4.0, 44.0, 5.0, 5.0, 56.0, 6.0, 9.0]  
} // End function
```

ตัวอย่างที่ 2 - เรียงตามค่าของตัวเลข ต้องใส่ Argument เป็นฟังก์ชันการกรองลงไปใน sort()

```
function sortArr() {  
    var arr2 = [4,2,5,6,5,2,44,9,56];  
    var dataSort = arr2.sort(function(a,b){  
        // จับมาทีละคู่แล้วเปรียบเทียบไปจบเรียงจบ a = 4  
        //                                     b = 2  
        if(a > b) { return 1 ; } // -1 หากต้องการเรียงมากไปน้อย  
        else if (a < b) { return -1 ; } // 1 หากต้องการเรียงมากไปน้อย  
        return 0 ;  
    }); // End sort  
    Logger.log(dataSort); // พิมพ์ : [2.0, 2.0, 4.0, 5.0, 5.0, 6.0, 9.0, 44.0, 56.0]  
} // End function
```

หากต้องการเรียงจากมากไปน้อย ก็ให้สลับตัว Return จาก 1 เป็น -1 อย่งไรก็ดี ไม่จำเป็นต้องเป็น 1 หรือ -1 เป็น 11 กับ -15 ก็ได้

ตัวอย่างที่ 3 - ทำให้สั้นกว่าตัวอย่างที่ 2 โดยไม่ต้องใช้ if

```
function sortArr() {  
    var arr2 = [4,2,5,6,5,2,44,9,56];  
    var dataSort = arr2.sort(function(a,b){  
        return a - b ; // เช่น 4-2 = 2 เป็น + ฉะนั้น a(4) ควรอยู่หลัง b(2)  
    }); // เช่น 2-5 = -3 เป็น - ฉะนั้น b(5) ควรอยู่หลัง a(2) .... ทำไปเรื่อยๆ  
    Logger.log(dataSort); // พิมพ์ : [2.0, 2.0, 4.0, 5.0, 5.0, 6.0, 9.0, 44.0, 56.0]  
} // End function
```

### 7.3.ข.) ข้อมูลเป็นข้อความทั้งหมด

ตัวอย่างที่ 1 - เรียงโดยไม่ใส่พารามิเตอร์ใดๆ ลงไปใน sort()

```
function sortArr() {  
    var arr3 = ["t","n","a","b","a","r","b","apple"];  
    // ข้อมูล "5" ชนิดเป็นข้อความ  
    var arr4 = ["t","n","5","A","b","a","r","B","Apple","apple","APple"] ;  
    Logger.log(arr3.sort()); // พิมพ์ : [a, a, apple, b, b, n, r, t]  
    Logger.log(arr4.sort()); // พิมพ์ : [5, A, APple, Apple, B, a, apple, b, n, r, t]  
} // End function
```

## ตัวอย่างที่ 2 - เรียงตามตัวอักษร ใช้วิธีเดียวกับข้อมูลตัวเลข

```
function sortArr() {  
  
    var arr4 = ["t","n","5","A","b","a","r","B","Apple","apple","APple"]; // 5 ชนิดเป็นข้อความ  
  
    var dataSort = arr4.sort(function(a,b){  
        if(a > b)      { return 1 ; }      // -1 หากต้องการเรียงมากไปน้อย  
        else if (a < b) { return -1 ; }     // 1 หากต้องการเรียงมากไปน้อย  
                        return 0 ;  
    }); // End sort  
  
    Logger.log(dataSort); // พิมพ์ : [5, A, APple, Apple, B, a, apple, b, n, r, t]  
  
} // End function
```

## ตัวอย่างที่ 3 - เรียงตามตัวอักษร –โดยไม่สนใจ Case

ตัวอย่างที่ 2 เรียงโดยเอา Case ของตัวอักษรมาใช้ในการเรียงด้วย หากต้องการเรียงโดยไม่สนใจ Case ให้ใช้ฟังก์ชัน toLowerCase() แปลงข้อความให้เป็นพิมพ์เล็กทั้งหมดก่อนเรียง

```
function sortArr() {  
  
    var arr4 = ["t","n","5","A","b","a","r","B","Apple","apple","APple"]; // 5 ชนิดเป็นข้อความ  
  
    var dataSort = arr4.sort(function(a,b){  
        a = a.toLowerCase();  
        b = b.toLowerCase();  
        if(a > b)      { return 1 ; }      // -1 หากต้องการเรียงมากไปน้อย  
        else if (a < b) { return -1 ; }     // 1 หากต้องการเรียงมากไปน้อย  
                        return 0 ;  
    }); // End sort  
  
    Logger.log(dataSort); // พิมพ์ : [5, A, a, Apple, apple, APple, b, B, n, r, t]  
  
} // End function
```



#### ตัวอย่างที่ 4 - กรณีข้อมูลมีตัวเลขผสมอยู่ในข้อความ

กรณีข้อมูลมีตัวเลขผสมอยู่ในข้อความ ให้แปลงเป็นชนิดข้อความก่อน โดยใช้ฟังก์ชัน `toString()` ไม่เช่นนั้นจะ Error แม้เป็นตัวเลขก็จะถูกเรียงแบบข้อความ

```
function sortArr() {  
    var arr6 = ["t","n","5","A",4,"b",44,"a","r","B","Apple","apple","APple"]; // 5 ชนิดเป็น  
    ข้อความ  
    var dataSort = arr6.sort(function(a,b){  
        a = a.toString().toLowerCase();  
        b = b.toString().toLowerCase();  
        if(a > b) { return 1 ; } // -1 หากต้องการเรียงมากไปน้อย  
        else if (a < b) { return -1 ; } // 1 หากต้องการเรียงมากไปน้อย  
        return 0 ;  
    }); // End sort  
    Logger.log(dataSort); // พิมพ์ : [4.0, 44.0, 5, A, a, Apple, apple, APple, b, B, n,  
    r, t]  
} // End function
```

#### 7.3.ค.) ข้อมูลผสม

ตัวอย่างที่ 1 - เรียงโดยไม่ใส่พารามิเตอร์ใดๆ ลงไปใน `sort()` - สังเกตว่าข้อมูลในอาร์เรย์ก็จะสะเปะสะปะ  
ปะหน้อย

```
function sortArr() {  
    var arr5 = var arr5 = ["t","","n","5","A",undefined,"",null  
        , "a",undefined,"r",1,66,9,"B","Apple","apple","APple"] ;  
    Logger.log(arr5.sort());  
    // พิมพ์ : [ , , 1.0, 5, 66.0, 9.0, A, APple, Apple, B, a, apple, n, null, r, t, null, null]  
} // End function
```

ตัวอย่างที่ 2 - แยกข้อมูลเป็น 2 ก้อน ก้อนข้อความ และ ตัวเลข ด้วยอาเรย์ว่างๆ 2 ตัว จากนั้น จึงเรียง  
แล้วเอามารวมกันด้วย concat

```
function sortArr() {  
  
    var arr7 = ["t","","n","5","A",undefined,"",null,  
                "a",undefined,"r",1,66,9,"B","Apple","apple","APple"] ;  
  
    var numberArr =[];  
    var textArr =[];  
    var otherArr =[];  
  
    arr7.map(function(item){  
        if(typeof item === "number"){  
            numberArr.push(item) ;  
        } else if (typeof item === "string"){  
            textArr.push(item.toString());  
        } else {  
            otherArr.push(item);  
        }; // End if  
    }); // End map  
  
    Logger.log(numberArr);  
    // พิมพ์ : [7.0, 66.0, 19.0]  
  
    Logger.log(textArr);  
    // พิมพ์ : [t, , n, 5, A, , a, r, B, Apple, apple, APple]  
  
    Logger.log(otherArr);  
    // พิมพ์ : [null, null, null]  
  
    // รวมอาเรย์ที่เรียงแล้วเข้าด้วยกัน  
    var comArr = numberArr.sort(sortNumber).concat(textArr.sort(sortAlpha));  
  
    Logger.log(comArr);  
    // พิมพ์ : [7.0, 19.0, 66.0, , , 5, A, APple, Apple, B, a, apple, n, r, t]  
  
    } // End function  
  
function sortAlpha(a,b){  
    if(a > b)      { return 1 ; }  
    else if (a < b) { return -1 ; }  
                    return 0 ;  
}  
// End function  
  
function sortNumber(a,b) {  
    return a-b ;  
}  
// End function
```

### 7.3.ง.) เรียงข้อมูลในอาเรย์ 2 มิติ

<https://stackoverflow.com/questions/16096872/how-to-sort-2-dimensional-array-by-column-value>

ตัวอย่างอย่างนี้ เป็นการเรียงอาเรย์แบบ 2 มิติ ซึ่งมิติของอาเรย์มีลักษณะเดียวกันกับการจับข้อมูลมาจากเร็นจิใน Google Sheets

```
function sortArr2Dim(){
    var arr2dim = [
        [ 12 , 'AAA' ] ,
        [ 58 , 'BBB' ] ,
        [ 28 , 'CCC' ] ,
        [ 18 , 'DDD' ]
    ] ;

    Logger.log(sortBy1stColumn(arr2dim)) ;
    /*
       [
         [12.0, AAA ] ,
         [18.0, DDD ] ,
         [28.0, CCC ] ,
         [58.0, BBB ]
       ]
    */
}

function sortBy1stColumn(arr2dim){
    arr2dim.sort(function(a, b) {
        if (a[0] === b[0])
        {
            return 0 ;
        }
        else
        {
            return (a[0] < b[0]) ? -1 : 1 ;
        }
    }) ;

    return arr2dim ;
}
```

### 7.3.จ.) เรียงโดยใช้ 2 คีย์ (หรือมากกว่า) – ใช้อาเรย์อย่างเดียว

ปรับแต่งจาก

<https://gomakethings.com/sorting-an-array-by-multiple-criteria-with-vanilla-javascript/>

ตัวอย่าง

```
function sortBy2Keys(){  
  // อาเรย์ 1 มิติ แต่สมาชิกแต่ละตัวเป็น Objects  
  var votes = [  
    [ 'Apple' , 1 , 100 ] ,  
    [ 'B'      , 2 , 109 ] ,  
    [ 'Carrot' , 3 , 102 ] ,  
    [ 'Banana' , 2 , 103 ] ,  
    [ 'B'      , 2 , 99  ] ,  
    [ 'A'      , 2 , 105 ] ,  
    [ 'B'      , 2 , 777 ]  
  ] ;  
  
  votes.sort(function (vote1, vote2) {  
    // คีย์ที่ 1 เรียงตาม คอลัมน์ที่ 2 - ก่อนไปหลัง  
    if (vote1[1] > vote2[1]) return -1 ;  
    if (vote1[1] < vote2[1]) return 1 ;  
  
    // คีย์ที่ 1 เรียงตาม คอลัมน์ที่ 1 - มากไปน้อย  
    if (vote1[0].toString() > vote2[0].toString()) return 1 ;  
    if (vote1[0].toString() < vote2[0].toString()) return -1 ;  
  
    // คีย์ที่ 1 เรียงตาม คอลัมน์ที่ 3 - น้อยไปมาก  
    if (vote1[2] > vote2[2]) return 1 ;  
    if (vote1[2] < vote2[2]) return -1 ;  
  }) ; // Close - sort  
  
  Logger.log(votes) ;  
  
  /*   [ [ Carrot, 3.0, 102.0 ] ,  
        [ A,      2.0, 105.0 ] ,  
        [ B,      2.0, 99.0  ] ,  
        [ B,      2.0, 109.0 ] ,  
        [ B,      2.0, 777.0 ] ,  
        [ Banana, 2.0, 103.0 ] ,  
        [ Apple , 1.0, 100.0 ] ]   */  
}
```

### 7.3.ฅ.) เรียงโดยใช้ 2 คีย์ (หรือมากกว่า) – ใช้ Object ช่วย

ปรับแต่งจาก

<https://gomakethings.com/sorting-an-array-by-multiple-criteria-with-vanilla-javascript/>

ตัวอย่างต่อไปนี้ เรียงตามตัวเลขก่อน(Vote) จากนั้นจึงเรียงตามชื่อ(Title) และ ตามคิว(que) สังเกตว่าข้อมูลเป็นอาเรย์ 1 มิติ แต่สมาชิกข้างในเป็น Objects ที่มีได้หลาย Properties

```
function sortBy2Keys(){
    // อาเรย์ 1 มิติ แต่สมาชิกแต่ละตัวเป็น Objects
    var votes = [
        { title: 'Apple' ,   vote: 1 ,   que : 100 } ,
        { title: 'B' ,       vote: 2 ,   que : 109 } ,
        { title: 'Carrot' ,  vote: 3 ,   que : 102 } ,
        { title: 'Banana' ,  vote: 2 ,   que : 103 } ,
        { title: 'B' ,       vote: 2 ,   que : 99 } ,
        { title: 'A' ,       vote: 2 ,   que : 105 } ,
        { title: 'B' ,       vote: 2 ,   que : 777 }
    ] ;

    votes.sort(function (vote1, vote2) {
        // เรียงตาม Votes
        // ถ้าค่าของตัวแรก มากกว่า ย้ายไปด้านล่าง
        // ถ้าค่าของตัวแรก น้อยกว่า ย้ายไปด้านบน
        if (vote1.vote > vote2.vote) return -1 ;
        if (vote1.vote < vote2.vote) return 1 ;

        // ถ้าค่าของ Votes ซ้ำกัน เรียงตามตัวอักษรของ Title ต่อ
        // ถ้าลำดับของตัวแรก มาก่อน ย้ายไปด้านบน - อย่างอื่น ย้ายไปด้านล่าง
        if (vote1.title.toString() > vote2.title.toString()) return 1 ;
        if (vote1.title.toString() < vote2.title.toString()) return -1 ;

        // เรียงโดยใช้คีย์ที่ 3 - จากน้อยไปมาก
        if (vote1.que > vote2.que) return 1 ;
        if (vote1.que < vote2.que) return -1 ;
    }) ; // Close - sort

    Logger.log(votes) ; // ทดสอบ Log หลังจากเรียงแล้ว

    // กรอกข้อมูลลง Google Sheets
    var sheet3 = SpreadsheetApp.getActiveSpreadsheet().getSheetByName('Sheet3') ;

    // จับใส่อาเรย์ 2 มิติ - เตรียมสำหรับกรอกลง Google Sheets
    var arr2dim = [] ;

    votes.forEach(function(obj){
        arr2dim.push([ obj.title , obj.vote , obj.que ] ) ;
    }) ; // Logger.log(d) ;

    sheet3.getRange(1,1,arr2dim.length,arr2dim[0].length).setValues(arr2dim) ;
}
```

	A	B	C
1	Carrot	3	102
2	A	2	105
3	B	2	99
4	B	2	106
5	B	2	777
6	Banana	2	103
7	Apple	1	100

#### 7.4. every() และ some()

JavaScript Every & Some Array Methods Tutorial - Google Sheets Apps Scripts - Array Methods Part 8  
[https://youtu.be/gaC290XzPX4?list=PLv9Pf9aNgemvD9NFa86\\_uDt-NWh37efmD](https://youtu.be/gaC290XzPX4?list=PLv9Pf9aNgemvD9NFa86_uDt-NWh37efmD)

**every()** ใช้ตรวจสอบสมาชิกทุกตัวในอาเรย์ว่าเป็นไปตามเงื่อนไขหรือเปล่า เช่น มากกว่า 0 ทุกตัวหรือเปล่า ส่วน **some()** ก็คล้ายกัน ถ้ามีเพียงบางตัวจริงตามเงื่อนไข ก็จะคืนค่าจริงกลับมา

##### 7.4.ก.) ตัวอย่างที่ 1 : อาเรย์ 1 มิติ

```
function every_some() {

    var arr1 = [1,2,3,4,5,7] ;

    var resultsEvery = arr1.every(function(num){
        return num > 3 ; // [ 1 , 2 , 3 , 4 , 5 , 7]
                        // T T F F F F =>
                        // มี false 1 ตัวคืนค่า false
    });

    Logger.log(resultsEvery); // พิมพ์ : false

    var resultsSome = arr1.some(function(num){
        return num > 3 ; // [ 1 , 2 , 3 , 4 , 5 , 7]
                        // T T F F F F => มี true 1 ตัวคืนค่า true
    });

    Logger.log(resultsSome); // พิมพ์ : true

} // End function
```

## 7.4.ข.) ตัวอย่างที่ 2 : อาเรย์ 2 มิติ

```
function every_some() {  
  
    var arr2 = [ ["a" , 6] ,  
                 ["b" , 9] ,  
                 ["c" , 6]   ] ;  
  
    var resultsEvery = arr2.every(function(row){  
        return row[0] === "b" ;    // [ a , b , c ]  
                                   // F T F    => มี false 1 ตัวคืนค่า false  
    });  
  
    Logger.log(resultsEvery);    // พิมพ์ : false  
  
    var resultsSome = arr2.some(function(num){  
                                   return num[1] > 7 ; // [ 6 , 9 , 6 ]  
                                   // T F T  
                                   // มี true 1 ตัวคืนค่า true  
    });  
  
    Logger.log(resultsSome) ;    // พิมพ์ : true  
} // End function
```

## 7.5. forEach() Method

JavaScript Array forEach() Method

[https://www.w3schools.com/jsref/jsref\\_foreach.asp](https://www.w3schools.com/jsref/jsref_foreach.asp)

forEach ใช้วนลูปส่งสมาชิกในอาเรย์ทีละตัว ไปประมวลผลในฟังก์ชัน

## 7.5.ก.) โครงสร้างการใช้งาน

โครงสร้างการใช้งาน

```
array.forEach(function(currentValue, index, arr), thisValue )
```

พารามิเตอร์ ( ไม่มีเวลาแปลจำ .... ขออภัยด้วย ดูตัวอย่างจะเข้าใจได้ไม่ยาก)

Parameter	Description
function(currentValue, index, arr)	(ต้องการ) A function to be run for each element in the array. Function arguments : (ฟังก์ชันที่จะใช้รันสมาชิกแต่ละตัวในอาเรย์)
	ArgumentDescription
	currentValue(ต้องการ) The value of the current element (ค่าของสมาชิกตัวปัจจุบัน)
	index(เว้นไว้ได้) The array index of the current element (ดรรชนีตำแหน่งสมาชิกตัวปัจจุบัน)
	arr(เว้นไว้ได้) The array object the current element belongs to (อาเรย์ที่สมาชิกตัวปัจจุบันอยู่)
thisValue	(เว้นไว้ได้) A value to be passed to the function to be used as its "this" value. If this parameter is empty, the value "undefined" will be passed as its "this" value

## 7.5.ข.) ตัวอย่างที่ 1 : การใช้งานแบบ 1 Argument

เอาตัวเลขในอาเรย์มาบวกกัน

```
function forEach_1() {  
    var sum = 0;  
    var numbers = [65, 44, 12, 4]; // ตัวแปรอาเรย์ที่ระบุสมาชิก  
    numbers.forEach(myFunction); // จบ forEach  
    function myFunction(item) { // 1 Argument  
        sum += item; // เอาค่าของสมาชิกมาบวกกันไปเรื่อยๆ แล้วเก็บไว้ใน sum  
    }  
    Logger.log(sum); // พิมพ์ : 125.0  
}
```

## 7.5.ค.) ตัวอย่างที่ 2 : การใช้งานแบบ 2 Arguments

เอาข้อความในอาเรย์มาต่อกัน

```
function forEach_2() {  
    var fruits = ["apple", "orange", "cherry"]; // item  
        // 0    1    2    // index  
    var text = "" ;  
    fruits.forEach(function(item, index) { // 2 Arguments  
        text += index + ":" + item + "<br>" ;  
    }) ; // จบ forEach  
    Logger.log(text) ; // พิมพ์ 0:apple<br>1:orange<br>2:cherry<br>  
}
```

## 7.5.ง.) ตัวอย่างที่ 3 : : การใช้งานแบบ 3 Arguments

เอาตัวเลขในอาเรย์ทั้งหมดมาคูณด้วยสิบ

```
function forEach_3() {  
    var numbers = [65, 44, 12, 4];  
    numbers.forEach(function(item, index, arr) { // 3 Arguments  
        // เอาสมาชิกในอาเรย์ numbers มาประมวลผล แล้วเก็บไว้ในอาเรย์ arr (ตัวเดิมชื่อใหม่)  
        arr[index] = item * 10;  
    } // จบ forEach  
    Logger.log(numbers);  
    // พิมพ์ : [650.0, 440.0, 120.0, 40.0]  
    // เอาทุกค่าอาเรย์ numbers มาคูณกับ 10  
}
```



### 7.5.จ.) ตัวอย่างที่ 4 : ตัวอย่างอื่นๆ

ตัวอย่างนี้ แปลงสมาชิกในอาเรย์ที่ อุณหภูมิเป็นองศา C ไปเป็นเป็น F แล้วเก็บไว้ในอาเรย์ว่างๆที่เตรียมไว้

```
function forEachLoopTemp() {  
    var celciusArray = [ 0 , 13 , 19 , 11 , 24 , 29 , 34 , 17 , 4 ] ;  
    var fahrenheitArray = [] ;    // เตรียมอาเรย์ว่างๆไว้เก็บผลลัพธ์  
    celciusArray.forEach(function(celciusTemp) {  
        // เรียกใช้ฟังก์ชัน convertCtoF(t)  
        // โดยส่งสมาชิกไปประมวลผลทีละตัว  
        var fahrenheitTemp = convertCtoF(celciusTemp) ;  
        // เก็บผลลัพธ์ไว้ในอาเรย์ว่างๆที่เตรียมไว้  
        fahrenheitArray.push(fahrenheitTemp) ;  
    });  
    Logger.log('Celcius Array:') ;    // ดูผลที่ Logs ----- > [ 01 ]  
    Logger.log(celciusArray) ;        // ดูผลที่ Logs ----- > [ 02 ]  
    Logger.log('Fahrenheit Array') ;    // ดูผลที่ Logs ----- > [ 03 ]  
    Logger.log(fahrenheitArray) ;      // ดูผลที่ Logs ----- > [ 04 ]  
}  
  
function convertCtoF(t) {  
    return (t * 9/5) + 32 ;  
}
```

ผล

```
Logs  
[01] Celcius Array:  
[02] [0.0, 13.0, 19.0, 11.0, 24.0, 29.0, 34.0, 17.0, 4.0]  
[03] Fahrenheit Array  
[04] [32.0, 55.4, 66.2, 51.8, 75.2, 84.2, 93.2, 62.6, 39.2]
```

### 7.6. reduce() Method

JavaScript Array forEach() Method

[https://www.w3schools.com/jsref/jsref\\_reduce.asp](https://www.w3schools.com/jsref/jsref_reduce.asp)

Javascript reduce example

<https://appdividend.com/2018/10/16/javascript-reduce-example-tutorial/>

**reduce** เป็น Method ของอาเรย์ ใช้วนลูปเพื่อนำสมาชิกในอาเรย์มาประมวลผลด้วยฟังก์ชัน เพื่อลดสมาชิกในอาเรย์ โดยจะคืนผลลัพธ์สุดท้ายมาตัวเดียว

## 7.6.ก.) โครงสร้างการใช้งาน

โครงสร้างการใช้งาน

```
array.reduce(function(total, currentValue, currentIndex, arr), initialValue)
```

พารามิเตอร์

Parameter	Description	
function(total, currentValue, index, arr)	(Required) ฟังก์ชันที่ใช้รันสมาชิกในอาเรย์แต่ละตัว	
	Argument	Description
	total	(Required) ค่า initialValue(หากมีการกำหนด) หรือเป็นค่าที่คืนกลับมาก่อนหน้า โดยฟังก์ชัน
	currentValue	(Required) ค่าของสมาชิกอาเรย์ตัวปัจจุบัน
	currentIndex	(Optional) ดรรชนีลำดับของสมาชิกอาเรย์ตัวปัจจุบัน
	arr	(Optional) อาเรย์ที่ใช้เก็บผลลัพธ์
initialValue	(Optional) ค่าที่ส่งให้กับฟังก์ชัน ที่ใช้เป็นค่าเริ่มต้น	

## 7.6.ข.) ตัวอย่างที่ 1 : รวมค่าของสมาชิกในอาเรย์

ตัวอย่างนี้ นำค่าของสมาชิกในอาเรย์มารวมกัน

```
function test_sum_array(){
  var data = [5, 10, 15, 20, 25] ; // Sum = 75
  var res = data.reduce(function(total, currentValue) {
    return total + currentValue ;
  });
  Logger.log(res) ; // พิมพ์ : 75
}
```

ดูการรวมรูปของโค้ดข้างต้น

ลูปครั้งที่	total	currentValue	return
1	0	5	5
2	5	10	15
3	15	15	30
4	30	20	50
5	50	25	75

### 7.6.ค.) ตัวอย่างที่ 2 : ลดมิติของอาเรย์

ตัวอย่างนี้ ลดมิติของอาเรย์จาก 2 เหลือ 1 สังเกตว่ามีการใช้พารามิเตอร์ `initialValue` ด้วยก็คือ []

```
function test_concat_array(){
  var data = [ [1, 2, 3], [4, 5, 6], [7, 8, 9] ];
  var flatValues = data.reduce(function(total, value) {
    return total.concat(value);
  }, []); // ใส่ initialValue

  Logger.log(flatValues); // พิมพ์ [1, 2, 3, 4, 5, 6, 7, 8, 9]
}
```

ดูการวนลูปของโค้ดข้างต้น

ลูปครั้งที่	total	value	return total.concat(value)
1	[] - initialValue	[1, 2, 3]	[1, 2, 3]
2	[1, 2, 3]	[4, 5, 6]	[1, 2, 3, 4, 5, 6]
3	[1, 2, 3, 4, 5, 6]	[7, 8, 9]	[1, 2, 3, 4, 5, 6, 7, 8, 9]

### 7.6.ง.) ตัวอย่างที่ 3 : หาค่าเฉลี่ยของสมาชิกในอาเรย์

ตัวอย่างนี้ หาค่าเฉลี่ยของสมาชิกในอาเรย์ ข้อสังเกตของโค้ดที่สำคัญ ก็คือ พารามิเตอร์ `array` (พารามิเตอร์ตัวที่ 4) เป็นอาเรย์ที่ใช้เก็บผลลัพธ์จากการประมวลผลของฟังก์ชัน มีมิติของอาเรย์เท่ากับ อาเรย์ที่นำมาประมวลผล

```
function test_avg_array(){
  var pounds = [ 11, 21, 16 ];
  var avg = pounds.reduce(function(total, amount, index, array){
    total += amount;
    if (index === array.length - 1) {
      return total / array.length;
    } else {
      return total;
    }
  });

  Logger.log(avg); // 16
}
```

ดูการวนลูปของโค้ดข้างต้น

ลูป	total+= amount	amount	index	array.length-1	return
1	0+11	11	0	3-1	11
2	11+21	21	1	3-1	33
3	33+16	16	2	3-1	16 (48/3)

## 7.6.จ.) ตัวอย่างที่ 4 : หาตัวซ้ำ

ตัวอย่างนี้ หาค่าเฉลี่ยของสมาชิกในอาเรย์ ข้อสังเกตของโค้ดที่สำคัญ ก็คือ พารามิเตอร์ `array` (พารามิเตอร์ตัวที่ 4) เป็นอาเรย์ที่ใช้เก็บผลลัพธ์จากการประมวลผลของฟังก์ชัน มีมิติของอาเรย์เท่ากับ อาเรย์ที่นำมาประมวลผล

```
function test_findDup_array(){
    var pounds = [11, 21, 16, 19, 46, 29, 46, 19, 21] ;
    var count = pounds.reduce(function(data, pound){
        data[pound] = (data[pound] || 0) + 1 ;
        return data ;
    }, {} ) ;

    Logger.log(count) ;    // {11=1.0, 46=2.0, 16=1.0, 29=1.0, 19=2.0, 21=2.0}
}
```

ดูการวนลูปของโค้ดข้างต้น

ลูป 1 `data = {}`  
`pound = 11`  
`return = data[11]=(data[11] || 0) + 1 → data = {11:1}`

ลูป 2 `data = {11:1}`  
`pound = 21`  
`return = data[21]=(data[21] || 0) + 1 → data = {11:1, 21:1}`

ลูป 3 `data = {11:1, 21:1}`  
`pound = 16`  
`return = data[16]=(data[16] || 0) + 1 → data = {11:1, 21:1, 16:1}`

ลูป 4 `data = {11:1, 21:1, 16:1}`  
`pound = 19`  
`return = data[19]=(data[19] || 0) + 1 → data = {11:1, 21:1, 16:1, 19:1}`

ลูป 5 `data = {11:1, 21:1, 16:1, 19:1}`  
`pound = 46`  
`return = data[46]=(data[46] || 0) + 1 → data = {11:1, 21:1, 16:1, 19:1, 46:1}`

ลูป 6 `data = {11:1, 21:1, 16:1, 19:1, 46:1}`  
`pound = 29`  
`return = data[29]=(data[29] || 0) + 1 → data = {11:1, 21:1, 16:1, 19:1, 46:1, 29:1}`

ลูป 7 `data = {11:1, 21:1, 16:1, 19:1, 46:1, 29:1}`  
`pound = 46`  
`return = data[46]=(data[46] || 0) + 1 → data = {11:1, 21:1, 16:1, 19:1, 46:2, 29:1}`

ลูป 8 `data = {11:1, 21:1, 16:1, 19:1, 46:2, 29:1}`  
`pound = 19`  
`return = data[19]=(data[19] || 0) + 1 → data = {11:1, 21:1, 16:1, 19:2, 46:2, 29:1}`

ลูป 9 `data = {11:1, 21:1, 16:1, 19:2, 46:2, 29:1}`  
`pound = 21`  
`return = data[21]=(data[21] || 0) + 1 → data = {11:1, 21:2, 16:1, 19:2, 46:2, 29:1}`

`return` สุดท้ายคือ `{11:1, 21:2, 16:1, 19:2, 46:2, 29:1}`

### หมายเหตุ 1 :

ปกติเราจะเห็น Properties ของตัวแปร Object เป็น Text(ข้อความ) แต่โค้ดต่อไปนี้ Properties เป็น Number(ตัวเลข) ซึ่งสอดคล้องกับโค้ดในตัวอย่างก่อนหน้านี้

```
function add_member_to_object(){  
    var count = { 111:1 , 222:2 } ; // Properties : Value เป็นตัวเลขทั้งคู่  
    count[333] = 3 ;                // เพิ่ม Property และ Value ตัวใหม่ ให้กับตัวแปร Object  
    Logger.log(count) ;              // { 111:1 , 222:2 , 333:3}  
}
```

### หมายเหตุ 2 :

ถ้าค่าของตัวแปรยังไม่มี(undefind) หรือยังไม่ถูกระบุ หากนำไปคำนวณจะเกิด Error ก็คือ NaN สามารถป้องกันได้โดยใช้ || (หรือ) เพื่อสร้างเงื่อนไขการคำนวณ เช่น a || 5 ถ้า a เป็น undefind จะนำ 5 มาใช้คำนวณแทน ตัวอย่างตามโค้ดต่อไปนี้

```
function pipe_in_(){  
    var A = 1, B = 2;  
    var M = -1, N = 0 ;  
    var X = null, Y = undefind ;  
    var Z ;  
  
    Logger.log((A || 0) + 1) ;    // 1+1      = 2  
    Logger.log((B || 0) + 1) ;    // 2+1      = 3  
    Logger.log((M || 0) + 1) ;    // 3+1      = 0  
    Logger.log((N || 0) + 1) ;    // 0+1      = 1  
    Logger.log((X || 0) + 1) ;    // 0(null)+1 = 1  
    Logger.log((Y || 0) + 1) ;    // 0(undefind)+1 = 1  
  
    Logger.log(null + 1) ;        // 1  
    Logger.log(undefind + 1) ;    // NaN  
    Logger.log(Z + 1) ;          // NaN  
}
```



# บทที่ 8

## อาเรย์เทคนิค



## 8.1. จับแถวและคอลัมน์ใน Google Sheets มาใส่อาเรย์

ตัวอย่างนี้มีการเรียกใช้เซอร์วิส SpreadsheetApp ฉะนั้นให้สร้างไฟล์ Google Sheets จากนั้นพิมพ์ข้อมูลตัวอย่างลงไป ตัวอย่างตามภาพ

	A	B	C	D
1		XXX	YYY	ZZZ
2	AAA			
3	BBB			
4	CCC			
5	DDD			

เมื่อเตรียมข้อมูลใน Google Sheet พร้อมแล้ว ให้สร้างโปรเจ็ค Google Apps Script แบบฝังไฟล์ จากนั้นพิมพ์โค้ดต่อไปนี้ลงไป โดยโค้ดนี้จะจับข้อมูลข้างต้นมาใส่อาเรย์แล้ว Logs ดู เพื่อดูว่า Google Apps Script จับข้อมูลมาเป็นอาเรย์อย่างไร

```
function getData() {  
    // จับชีทที่เปิดไว้ในไฟล์ Google Sheet ใส่ตัวแปร sheet  
    var sheet = SpreadsheetApp.getActiveSpreadsheet().getActiveSheet();  
  
    // จับข้อมูลในเรนจ์ A2:A5 - 1 คอลัมน์  
    var d1Col = sheet.getRange("A2:A5").getValues();  
  
    // จับข้อมูลในเรนจ์ B1:D1 - 1 แถว  
    var d1Row = sheet.getRange("B1:D1").getValues(); // จับข้อมูล 1 แถว  
  
    Logger.log(d1Col); // ดูผลที่ Logs ต่อ ----- > [ 01 ]  
    Logger.log(d1Row); // ดูผลที่ Logs ต่อ ----- > [ 02 ]  
}
```

ผล

### Logs

```
[ 01 ] [[AAA], [BBB], [CCC], [DDD]] // 1 คอลัมน์ - เป็นอาเรย์ 2 มิติ  
[ 02 ] [[XXX, YYY, ZZZ]] // 1 แถว - เป็นอาเรย์ 2 มิติ
```

## 8.2. จับเรนจ์ใส่ในอาเรย์ จับอาเรย์ใส่ในเรนจ์ (Google Sheets)

ข้อนี้ใช้เซอร์วิส SpreadsheetApp และ Method ของคลาสต่างๆจำนวนหนึ่ง แต่ก็ เป็น Method พื้นๆที่ใช้อย่าง

ให้สร้างโปรเจ็ค แล้วลองพิมพ์โค้ดเพื่อทำความเข้าใจไปก่อน



## 8.2.ก.) จับเร้นจ้ใน Google Sheet มาใส่ในอาเรย์

ข้อมูลในไฟล์ Google Sheet ที่ชี้ทื่อ Sheet2 มีดังต่อไปนี้

	A	B
1	มะนาว	12
2	มะพร้าว	35
3	ส้มโอ	45
4	แตงโม	50

เราสามารถจับข้อมูลทั้งหมดไปใส่ในตัวแปรอาเรย์ได้ดังนี้

```
function getData() {  
    var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Sheet2");  
    //          เริ่มที่ A1      4 แถว      2 คอลัมน์  
    var data = sheet.getRange(1, 1,  
sheet.getLastRow(),sheet.getLastColumn()).getValues();  
  
    Logger.log(data);    // ข้อมูลที่ Logs ต่อ ----- > [ 01 ]  
}
```

ผล - ได้อาเรย์ 2 มิติ เป็นแบบอาเรย์เส้น ในอาเรย์เส้น

### Logs

```
[ 01 ] [[มะนาว, 12.0], [มะพร้าว, 35.0], [ส้มโอ, 45.0], [แตงโม, 50.0]]
```

## 8.2.ข.) จับค่าในอาเรย์ ไปใส่ใน Google Sheet

ข้อนี้ตรงข้ามกับ ข้อก่อนหน้า เราจะนำค่าในอาเรย์ ไปใส่ในไฟล์ Google Sheet แทน

```
function setData() {  
    var Arr2dim = [  
        ["Linda", 27] ,    // Arr2dim[0] ก็คือ อาเรย์เส้นนี้  
        ["Lisa", 35] ,    // Arr2dim[1] ก็คือ อาเรย์เส้นนี้  
        ["John", 42]      // Arr2dim[2] ก็คือ อาเรย์เส้นนี้  
    ] ;  
    var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Sheet2") ;  
  
    //          เริ่มที่ D1      3 แถว      2 คอลัมน์  
    var data = sheet  
        .getRange(1, 4, Arr2dim.length, Arr2dim[0].length)  
        .setValues(Arr2dim) ;  
}
```

ผล - ดูที่เซลล์ D1:E3

	A	B	C	D	E
1	มะนาว	12		Linda	27
2	มะพร้าว	35		Lisa	35
3	ส้มโอ	45		John	42
4	แตงโม	50			

### 8.3. นำคอลัมน์ในชีตมาคำนวณ แล้ววางไว้อีกคอลัมน์หนึ่ง (Google Sheets)

ข้อนี้ ต่อเนื่องจากข้อก่อนหน้า แต่จะเป็นการนำ map มาใช้คำนวณใน Google Sheets โดย นำคอลัมน์ในชีตมาคำนวณ แล้ววางไว้อีกคอลัมน์หนึ่ง

ข้อมูลในชีตมีดังต่อไปนี้

	A	B	C
1	No	คอร์ส	ค่าใช้จ่าย
2	1	LibreOffice Calc	7,600
3	2	LibreOffice Base	9,000
4	3	Google Apps	7,600
5	4	Google Sheet	7,600
6	5	Google Apps Script	9,000

#### 8.3.ก.) พัฒนาการที่ 1

จับข้อมูลใน Google Sheet มา Log ดูก่อน เพื่อดูว่าจับข้อมูลมาถูกหรือเปล่า

```
function calcColumn() {  
    var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Sheet2") ;  
    var data = sheet.getRange(2,1, sheet.getLastRow()-1,  
        sheet.getLastColumn()).getValues() ;  
    Logger.log(data) ; // ดูผลที่ Logs ----- >  
}
```

ผล

#### Logs

[[1.0, LibreOffice Calc, 7600.0], [2.0, LibreOffice Base, 9000.0], [3.0, Google Apps, 7600.0], [4.0, Google Sheet, 7600.0], [5.0, Google Apps Script, 9000.0]]

### 8.3.ข.) พัฒนาการที่ 2

เราต้องการนำคอลัมน์ C เท่านั้นมาคำนวณ (คอลัมน์ "ค่าใช้จ่าย" ) แล้ววางลงไปคอลัมน์ที่ D

อันดับแรก คำนวณก่อนข้อมูลที่จับมาจาก Google Sheet โดยใช้ `map()` ดึงสมาชิกมาคำนวณ จากนั้นเอาผลลัพธ์ทั้งก้อน ใส่กลับไป Google Sheets

```
function calcColumn() {  
  var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName("Sheet2") ;  
  var data = sheet.getRange(2,1, sheet.getLastRow()-1,sheet.getLastColumn()).getValues() ;  
  
  /* [  
    [1.0,    LibreOffice Calc,    7600.0],  
    [2.0,    LibreOffice Base,    9000.0],  
    [3.0,    Google Apps,        7600.0],  
    [4.0,    Google Sheet,       7600.0],  
    [5.0,    Google Apps Script, 9000.0]  ]  */  
  
  var calcData = data.map(function(row){  
    // คำนวณเฉพาะคอลัมน์ที่ 3 - หาดด้วย 8 แล้วเก็บค่าไว้ในตัวแปร  
    var cost = row[2]/8 ;  
    // return [ row[0] , row[1] , cost ] ;  
    // คืนกลับทั้งหมด โดยคำนวณบางคอลัมน์  
  
    return [cost] ;          // คืนกลับเฉพาะคอลัมน์ที่คำนวณ  
  }) /* จบ map / ;  
  
  // Logger.log(calcData);  
  /* [ มี 5 แถว 1 คอลัมน์  
    [950.0],  
    [1125.0],  
    [950.0],  
    [950.0],  
    [1125.0]  ]  */  
  
  //          คอลัมน์สุดท้ายของชีท+1 จำนวนสมาชิก จำนวนสมาชิกตัวแรก(ในอาเรย์)  
  //          2          4          5          1  
  sheet.getRange(2, sheet.getLastColumn()+1, calcData.length, calcData[0].length)  
    .setValues(calcData) ;  
}
```

ผล - ดูคอลัมน์ D

	A	B	C	D
1	No	คอร์ส	ค่าใช้จ่าย	
2	1	LibreOffice Calc	7,600	950
3	2	LibreOffice Base	9,000	1,125
4	3	Google Apps	7,600	950
5	4	Google Sheet	7,600	950
6	5	Google Apps Script	9,000	1,125

## 8.4. กรองแถวว่าง หรือ แถวที่มีเซลล์ว่างทิ้ง

ตัวอย่างข้อมูล ใน Google Sheets มีดังต่อไปนี้

	A	B	C	D	E	F
1	ID	Course	Expense	Categories	Instructor	Course hours
2	1	LibreOffice Writer : Basic	7,600	LibreOffice	Por	6
3	2	LibreOffice Writer : Advance	9,000	LibreOffice	WK	6
4	3	LibreOffice Calc : Basic	7,600	LibreOffice		6
5	4	LibreOffice Calc : Advance	9,000	LibreOffice	WK	6
6	5	LibreOffice Calc : Data Analysis		LibreOffice	WK	6
7	6	LibreOffice Base : Flat DB	9,000	LibreOffice	WK	6
8	7	LibreOffice Base : RDBMS	9,000		WK	6
9	8	Gimp for Photographer	15,000	Graphics	Wasankds	12
10	9	Gimp for Cartoonist	15,000	Graphics	Wasankds	12
11						
12	10	Inkscape	15,000	Graphics	Wasankds	12
13	11	Ubuntu Basic for Beginner	7,600		WK	6
14	12	Google Apps Basic for Work	7,600	Google Suite	PoE Club	6
15						
16	13	Google Sheet : Basic	7,600	Google Suite	PoE Club	6
17	14	Google Sheet : Data Analysis	9,000	Google Suite	PoE Club	6
18	15	Google Apps Script	21,000		PoE Club	
19	16	DIY Home Automation	7,600	IoT	Wasan	6

แถวว่าง เมื่อจับข้อมูลมาเป็นอาร์เรย์ จะได้เป็นอาร์เรย์ 2 มิติว่างๆ ก็คือ [ [], [], [], [], [], [] ]

โค้ดต่อไปนี้ กรองแถวว่างออก แล้ววางข้อมูลที่กรองแล้วกลับไปทีเดิม

```
function filterEmptyRows() {  
  var ss = SpreadsheetApp.getActiveSpreadsheet() ;  
  var sheetData = ss.getSheetByName("Sheet3") ;  
  var rawData = sheetData.getRange(2,1,sheetData.getLastRow()-1, sheetData.getLastColumn())  
    .getValues() ;  
  
  // ===== ตัวตรวจสอบและกรองข้อมูล =====  
  var fData = rawData.filter(function(row){  
    return !row.every(function(cell){ // กรองตัวที่ไม่ใช่(!) ที่ผ่านเงื่อนไขมาจาก every-จับตัวไม่ว่าง  
      return cell == "" ; // ตรวจสอบว่าทุกเซลล์เป็นค่าว่างหรือไม่  
    }); // End - every  
  }); // End - filter  
  
  sheetData.activate().getRange(2,1,sheetData.getLastRow()-1,sheetData.getLastColumn())  
    .clearContent(); // จับไปที่ชี้ที่เดิมที่มีข้อมูล จากนั้นล้างตั้งแต่แถวที่ 2 ทิ้งไป  
  
  sheetData.getRange(2,1,fData.length,fData[0].length).setValues(fData);  
  // Logger.log(fData);  
} // End function
```

หากต้องการกรองบรรทัดที่มีเซลล์ว่างเพียงบางเซลล์ทิ้ง ให้เปลี่ยนจาก every เป็น some

```
var fData = rawData.filter(function(row){
  return !row.some(function(cell){ // filter ตัวที่ไม่ใช่(!) ที่ผ่านเงื่อนไขมาจาก every
    return cell == "" ;           // ตรวจสอบว่าทุกเซลล์เป็นค่าว่างหรือไม่
  }); // End – every
}); // End – filter
```

ผล - กรองบรรทัดที่มีเซลล์ว่างทิ้ง

	A	B	C	D	E	F
1	ID	Course	Expense	Categories	Instructor	Course hours
2	1	LibreOffice Writer : Basic	7,600	LibreOffice	Por	6
3	2	LibreOffice Writer : Advance	9,000	LibreOffice	WK	6
4	4	LibreOffice Calc : Advance	9,000	LibreOffice	WK	6
5	6	LibreOffice Base : Flat DB	9,000	LibreOffice	WK	6
6	8	Gimp for Photographer	15,000	Graphics	Wasankds	12
7	9	Gimp for Cartoonist	15,000	Graphics	Wasankds	12
8	10	Inkscape	15,000	Graphics	Wasankds	12
9	12	Google Apps Basic for Work	7,600	Google Suite	PoE Club	6
10	13	Google Sheet : Basic	7,600	Google Suite	PoE Club	6
11	14	Google Sheet : Data Analysis	9,000	Google Suite	PoE Club	6
12	16	DIY Home Automation	7,600	IoT	Wasan	6

## 8.5. การกรองตัวซ้ำในอาเรย์ ให้เหลือแต่ตัวที่ไม่ซ้ำ

ข้อนี้ เป็นการกรองตัวซ้ำในอาเรย์ ให้เหลือแต่ตัวที่ไม่ซ้ำ

งานลักษณะที่เราต้องกรองตัวซ้ำภายในอาเรย์ออก เป็นงานที่เราต้องทำบ่อยๆ เช่น จับคอลลัมน์จากในชีท แล้วกรองตัวซ้ำออก เป็นต้น

หมายเหตุ : ดูเพิ่มเติมเรื่อง Remove duplicates (เมนู Tools → Remove duplicates) เป็นเครื่องมือใน Google Sheets ที่ใช้กรองตัวซ้ำ

### 8.5.ก.) เทคนิคที่ 1 - สั้นแต่มีประสิทธิภาพสูง

โค้ดต่อไปนี้ เป็นลोजิกที่ใช้กรองตัวซ้ำที่เขียนโค้ดได้สั้น แต่มีประสิทธิภาพสูง

```
var arr = [ "A" , "1" , 2 , 3 , 4 , 5 , 5 , 5 , 7 , 8 , 2 , 3 , 4 , 4 , 3 , 4 , 4 ] ;
var result = [] ;

result = arr.filter(function(item,i){
  return arr.indexOf(item) == i ;
// ตัวอย่าง
//      [ "A" , "1" , 2 , 3 , 4 , 5 , 5 , 5 , 7 , 8 , 2 , 3 , 4 , 4 , 3 , 4 , 4 ] ;
// indexOf(2)      2 (true)      10(false)
// i              0   1   2  3  4  5  6  7  8  8  10 11 12 13 14 15 16
});

Logger.log(result);           // พิมพ์ : [ A , 1 , 2.0 , 3.0 , 4.0 , 5.0 , 7.0 , 8.0 ]
```

## 8.5.ข.) เทคนิคที่ 2 - แยกสมาชิกในอาเรย์ที่ซ้ำและไม่ซ้ำออกเป็นอาเรย์ 2 ก้อน

โค้ดต่อไปนี้จะแยกสมาชิกในอาเรย์ที่ซ้ำและไม่ซ้ำออกเป็นอาเรย์ 2 ก้อน

```
var arr = [37, 39, 42, 46, 46, 11, 33, 11] ;

var uniqueDays = [] ;
var dupDays = [] ;

arr.forEach(function(d){
    if(uniqueDays.indexOf(d) === -1) {
        uniqueDays.push(d) ;
    }else{
        dupDays.push(d) ;
    }
}); // Close forEach
Logger.log(uniqueDays) ; // [ 37.0, 39.0, 42.0, 46.0, 11.0, 33.0 ] // ก้อนที่ไม่ซ้ำ
Logger.log(dupDays) ; // [ 46, 11 ] // ก้อนที่ซ้ำ
```

ขั้นตอน

ลูป	d	uniqueDays	uniqueDays.indexOf(d)	uniqueDays.push(d)
1	37	[]	-1 (false)	[37]
2	39	[37]	-1 (false)	[37, 39]
3	42	[37, 39]	-1 (false)	[37, 39, 42]
4	46	[37, 39, 42]	-1 (false)	[37, 39, 42, 46]
5	46	[37, 39, 42, 46]	3 (true)	[37, 39, 42, 46]
6	11	[37, 39, 42, 46]	-1 (false)	[37, 39, 42, 46, 11]
7	33	[37, 39, 42, 46, 11]	-1 (false)	[37, 39, 42, 46, 11, 33]
8	11	[37, 39, 42, 46, 11, 33]	4 (true)	[37, 39, 42, 46, 11, 33]

## 8.6. การกรองตัวซ้ำในอาเรย์ ให้เหลือแต่ตัวที่ซ้ำ

ปรังแต่งจาก

<https://stackoverflow.com/questions/840781/get-all-non-unique-values-i-e-duplicate-more-than-one-occurrence-in-an-array>

กลับกันกับข้อก่อนหน้า ข้อนี้เป็นการกรองอาเรย์ให้เหลือแต่ตัวที่ซ้ำ

```
var input = [ 'A', 'B', 'C', 'A', 'C', 'A' ] ;
var duplicates = input.reduce(function(acc, el, i, arr) {
    if (arr.indexOf(el) !== i && acc.indexOf(el) < 0) {
        acc.push(el) ;
    } // Close - if
    return acc ;
}, [] ) ; // Close - reduce

Logger.log(duplicates) ; // ผล : [ A, C ]
```

การวนลูปของโค้ดข้างต้น

ลูป	acc	el	i	arr	arr.indexOf(el) !== i && acc.indexOf(el) < 0	acc.push(el)	return
1	[]	A	0	[ 'A', 'B', 'C', 'A', 'C', 'A' ]	arr.indexOf('A') !== 0 && acc.indexOf('A') < 0  0 !== 0 && -1 < 0  0 && 1  0 (false)	[]	[]
2	[]	B	1	[ 'A', 'B', 'C', 'A', 'C', 'A' ]	arr.indexOf('B') !== 1 && acc.indexOf('B') < 0  1 !== 1 && -1 < 0  0 && 1  0 (false)	[]	[]
3	[]	C	2	[ 'A', 'B', 'C', 'A', 'C', 'A' ]	arr.indexOf('C') !== 2 && acc.indexOf('C') < 0  2 !== 2 && -1 < 0  0 && 1  0 (false)	[]	[]
4	[]	A	3	[ 'A', 'B', 'C', 'A', 'C', 'A' ]	arr.indexOf('A') !== 3 && acc.indexOf('A') < 0  0 !== 3 && -1 < 0  0 && 1  1 (true)	[A]	[A]
5	[A]	C	4	[ 'A', 'B', 'C', 'A', 'C', 'A' ]	arr.indexOf('C') !== 4 && acc.indexOf('C') < 0  2 !== 4 && -1 < 0  1 && 1  1 (true)	[A,C]	[A,C]
6	[A,C]	A	5	[ 'A', 'B', 'C', 'A', 'C', 'A' ]	arr.indexOf('A') !== 5 && acc.indexOf('A') < 0  0 !== 4 && 0 < 0  1 && 0  1 (false)	[A,C]	[A,C]

## 8.7. เปรียบเทียบอาเรย์ 2 ก้อน

### 8.7.ก.) เปรียบเทียบอาเรย์ 2 ก้อน เก็บตัวที่ไม่ซ้ำกันได้

ตัวอย่าง

```
// อาเรย์ก้อนที่ 1 – ถูกเปรียบเทียบ - จับจากเร้นจี่ใน Google Sheets 1 แถว
var arrData2Dims = [ [ 'D' , 'C' , 'X' , 'B' ] ] ;
var arrData1Row = arrData2Dims[0] ;           //   [ D , C , X , B ]

// อาเรย์ก้อนที่ 2 – ใช้เปรียบเทียบ - จับจากเร้นจี่ใน Google Sheets 1 เซลล์
var findArr = [ [ 'X' ] ] ;
var findVal = findArr[0][0] ;                 //   X

var filterOut = [] ;
arrData1Row.forEach(function(item){
    if(item != findVal){
        filterOut.push(item);
    }
}) ; // Close - forEach

Logger.log(filterOut) ;                       // ผล : [ D, C, B ]
```

## 8.8. กรองเซลล์ว่างทิ้ง (ข้อมูล 1 คอลัมน์)

ข้อมูลใน Google Sheets ขี้หมีดังต่อไปนี้

	A
1	หมวด
2	OTC
3	DGG
4	DGB
5	CTG
6	CTB
7	OPT
8	DCC
9	
10	
11	
12	



โค้ดต่อไปนี้จะจับค่าในเซลล์ A2:F12 แต่เซลล์ A9:A12 เป็นเซลล์ว่าง ซึ่งเราจะกรองค่าว่างที่จับมาได้ทั้งไป

```
function filterEmpCell(){
    var valCatCodes = ss.getRange(' A2:A12')
        .getValues() // [[OTC], [DGG], [DGB], [CTG], [CTB], [OPT], [DCC], [], [], [], []]
        .reduce(function(total, value) {
            return total.concat(value) ;
        },[]) // [OTC, DGG, DGB, CTG, CTB, OPT, DCC, , , , ]
        .filter(function(e){
            return e ; // [OTC, DGG, DGB, CTG, CTB, OPT, DCC]
        });

    Logger.log(valCatCodes) // [OTC, DGG, DGB, CTG, CTB, OPT, DCC]
}
```

## 8.9. ใช้ filter() กรองตารางข้อมูลใน Google Sheet

JavaScript Filter Method Tutorial - Google Sheets Apps Scripts - Array Methods Part 7

[https://youtu.be/PT\\_TDhMhWsE?list=PLv9Pf9aNgemvD9NFa86\\_udt-NWh37efmD](https://youtu.be/PT_TDhMhWsE?list=PLv9Pf9aNgemvD9NFa86_udt-NWh37efmD)

ตารางข้อมูลที่จะใช้ทดสอบ มีข้อมูลดังต่อไปนี้

	A	B	C	D	E	F
1	ID	Course	Expense	Categories	Instructor	Course hours
2	1	LibreOffice Writer : Basic	7,600	LibreOffice	Por	6
3	2	LibreOffice Writer : Advance	9,000	LibreOffice	WK	6
4	3	LibreOffice Calc : Basic	7,600	LibreOffice	Por	6
5	4	LibreOffice Calc : Advance	9,000	LibreOffice	WK	6
6	5	LibreOffice Calc : Data Analysis	9,000	LibreOffice	WK	6
7	6	LibreOffice Base : Flat DB	9,000	LibreOffice	WK	6
8	7	LibreOffice Base : RDBMS	9,000	LibreOffice	WK	6
9	8	Gimp for Photographer	15,000	Graphics	Wasankds	12
10	9	Gimp for Cartoonist	15,000	Graphics	Wasankds	12
11	10	Inkscape	15,000	Graphics	Wasankds	12
12	11	Ubuntu Basic for Beginer	7,600	OS	WK	6
13	12	Google Apps Basic for Work	7,600	Google Suite	PoE Club	6
14	13	Google Sheet : Basic	7,600	Google Suite	PoE Club	6
15	14	Google Sheet : Data Analysis	9,000	Google Suite	PoE Club	6
16	15	Google Apps Script	21,000	Google Suite	PoE Club	18
17	16	DIY Home Automation	7,600	IoT	Wasan	6

### 8.9.ก.) พัฒนาการที่ 1 : จับค่าในคอลัมน์มากรอง

ก่อนอื่นๆ จับค่าในคอลัมน์ มากรองตามเงื่อนไข ออกมา Log ดูก่อน

ตัวอย่างโค้ดนี้ เอาเฉพาะค่าในคอลัมน์ "Instructor" (E) ที่มีค่าเป็น **Por** ออกมา Log ดู

```
function filterArr() {  
    var ss = SpreadsheetApp.getActiveSpreadsheet() ;  
    var sCrS = ss.getSheetByName("OurCourse") ;  
    var dCrS = sCrS.getRange(2,1,sCrS.getLastRow()-1,sCrS.getLastColumn()).getValues() ;  
    var nInstr = "Por" ;  
  
    // จับฟังก์ชันที่ใช้กรองยัดลงไป filter เลย  
    var fData = dCrS.filter(function(row){  
        return row[4] === nInstr ;  
    }) ;  
    Logger.log(fData) ;  
} // ดูผลที่ Logs ----- >
```

ผล - มี 2 บรรทัดที่ผ่านการกรอง

```
Logs  
[01] [ [1.0, LibreOffice Writer : Basic, 7600.0, LibreOffice, Por, 6.0],  
        [3.0, LibreOffice Calc : Basic, 7600.0, LibreOffice, Por, 6.0 ] ]
```

### 8.9.ข.) พัฒนาการที่ 2 : ใส่ผลการกรองกลับไปที่ชีท

เมื่อจับค่าที่ต้องการและกรองเรียบร้อยแล้ว ถัดมาจะเป็นการใส่กลับไปใน Google Sheet ที่ชีทใหม่ที่สร้างโดยใช้โค้ด

```
function filterArr() {  
    var ss = SpreadsheetApp.getActiveSpreadsheet();  
    var sCrS = ss.getSheetByName("OurCourse");  
    var dCrS = sCrS.getRange(2,1,sCrS.getLastRow()-1,sCrS.getLastColumn()).getValues();  
    var nInstr = "Por" ;  
  
    var fData = dCrS.filter(function(row){ return row[4] === nInstr ; });  
  
    // สร้างชีทใหม่สำหรับเก็บผลการกรอง  
    var sTarget = ss.insertSheet("Courses by " + nInstr);  
  
    // ใส่ข้อมูลที่กรองแล้ว ลงไปในชีท  
    sTarget.getRange(2, 1 ,fData.length, fData[0].length).setValues(fData);  
}
```

ผล

	A	B	C	D	E	F
1						
2	1	LibreOffice Writer : Basic	7600	LibreOffice	Por	6
3	3	LibreOffice Calc : Basic	7600	LibreOffice	Por	6

### 8.9.ค.) พัฒนาการที่ 3 : กรอง 2 เงื่อนไข

โค้ดนี้จะเป็นการกรองแบบ 2 เงื่อนไข ก็คือ กรองผู้สอนเป็น Wasankds และ ค่าใช้จ่ายต่อคอร์ส มากกว่า 10,000 บาท

```
function filterArr() {  
    var ss = SpreadsheetApp.getActiveSpreadsheet();  
    var sCrS = ss.getSheetByName("OurCourse");  
    var dCrS = sCrS.getRange(2,1,sCrS.getLastRow()-1,sCrS.getLastColumn()).getValues();  
    var nInstr = "Wasankds" ;  
  
    var fData = dCrS.filter(function(row){ // กรอง 2 เงื่อนไข  
        return row[4].toLowerCase() === nInstr.toLowerCase() && row[2] > 10000 ;  
    });  
    // Logger.log(fData);  
  
    // สร้างชีตใหม่สำหรับเก็บผลการกรอง  
    var sTarget = ss.insertSheet("Courses by "+ nInstr);  
  
    // ใส่ข้อมูลที่กรองแล้ว ลงไปในชีต  
    sTarget.getRange(2,1,fData.length,fData[0].length).setValues(fData);  
}
```

ผล

	A	B	C	D	E	F
1						
2	8	Gimp for Photographer	15000	Graphics	Wasankds	12
3	9	Gimp for Cartoonist	15000	Graphics	Wasankds	12
4	10	Inkscape	15000	Graphics	Wasankds	12

#### หมายเหตุ 1

กรณีกรองแล้ว ไม่มีบรรทัดไหนผ่านการกรองเลย ให้เราใช้ if ตรวจสอบด้วย ไม่เช่นนั้น โค้ดจะ Error ในขั้นตอนเขียนผลการกรองกลับไป Google Sheets ให้แก้โค้ดเป็นดังต่อไปนี้

```
if(fData.length > 0) { // ตรวจสอบจำนวนสมาชิกในอาร์เรย์ผลการกรอง  
    sTarget.getRange(2,1,fData.length,fData[0].length).setValues(fData);  
} else {  
    sTarget.getRange(2,1,1,1).setValue("ไม่มีข้อมูลผ่านการกรอง");  
} ;
```

## หมายเหตุ 2

การกรองแบบ 2 เงื่อนไข กรณีเชื่อมการกรองแบบ Or ใช้ Operator เป็น || (Double Pipes) การใช้  
งานยกตัวอย่างเช่น

```
var fData = dCrs.filter(function(row){  
    return row[4] === "Por" || row[4] === "PoE Club" ;    // กรอง 2 เงื่อนไขแบบ Or  
});
```

ผล - กรองโดยเชื่อมเงื่อนไขแบบ Or

	A	B	C	D	E	F
1						
2	1	LibreOffice Writer : Basic	7600	LibreOffice	Por	6
3	3	LibreOffice Calc : Basic	7600	LibreOffice	Por	6
4	12	Google Apps Basic for Work	7600	Google Suite	PoE Club	6
5	13	Google Sheet : Basic	7600	Google Suite	PoE Club	6
6	14	Google Sheet : Data Analysis	9000	Google Suite	PoE Club	6
7	15	Google Apps Script	21000	Google Suite	PoE Club	18

## 8.10. ทำ VLOOKUP ด้วยอาเรย์

ในข้อนี้ เป็นตัวอย่างการทำ Lookup ที่ให้ผลลัพธ์คล้ายกับฟังก์ชัน VLOOKUP ก็คือ ส่งตัวค้นหาไปยัง  
ตารางข้อมูล จากนั้นค้นหาในคอลัมน์ที่ระบุ หากเจอข้อมูลที่ตรงกันให้คืนค่าคอลัมน์ตามที่ระบุในแถวที่พบ  
ข้อมูลกลับมา

ตัวอย่าง - สีเขียวคือตารางข้อมูล เซลล์ F2 คือเซลล์ที่จะใช้ Lookup ถ้า Lookup ในตารางข้อมูลแล้ว  
แล้วเจอค่าตรงกัน จะคืนค่าในแถวเดียวกัน คอลัมน์ที่ 3 กลับมาใส่ในเซลล์ G2

	A	B	C	D	E	F	G
1	รหัส	หมวด	สินค้าหรือบริการ	ราคาต่อหน่วย		รหัส	สินค้าหรือบริการ
2	OTC0020	OTC	Test 0020	100		DGG1520	อาบน้ำและตัดขนหมา (15-20 กก.)
3	OTC0021	OTC	Test 0019	50			
4	OTC0022	OTC	Other 1	50			
5	OTC0023	OTC	Other 2	50			
6	DGG1520	DGG	อาบน้ำและตัดขนหมา (15-20 กก.)	550			
7	DGG0715	DGG	อาบน้ำและตัดขนหมา (7-15 กก.)	350			
8	DGG0017	DGG	อาบน้ำและตัดขนหมา (1- 7 กก.)	250			
9	DCC0050	DCC	ส่วนลด 50 บาท	50			
10	DCC0040	DCC	ส่วนลด 40 บาท	40			
11	DCC0030	DCC	ส่วนลด 30 บาท	30			
12	DCC0020	DCC	ส่วนลด 20 บาท	20			
13	DCC0010	DCC	ส่วนลด 10 บาท	10			

```

var ss = SpreadsheetApp.getActiveSpreadsheet() ;
var sheet = ss.getActiveSheet() ;

// (1.) จับค่าในคอลัมน์ที่จะไปค้นหา >> แปลงเป็นอาเรย์ 1 มิติ - เพราะ indexOf ใช้ได้กับอาเรย์ 1 มิติเท่านั้น
var columnToLookup = sheet.getRange('A2:A13').getValues() ;
columnToLookup = columnToLookup.map(function(r){
    return r[0] ;
});
Logger.log(columnToLookup)
// [OTC0020, OTC0021, OTC0022, OTC0023, DGG1520, DGG0715, DGG0017, DCC0050, DCC0040, DCC0030, DCC0020, DCC0010]

// (2.) จับค่าที่ใช้เป็นตัวค้นหา เช่น DGG1520
var lookupValue = ss.getRange('F2').getValue() ;
Logger.log(lookupValue) ; // DGG1520

// (3.) ใช้ค่าที่เป็นตัวค้นหา ไปค้นในคอลัมน์ที่จับมา >> ได้ดรรชนีตำแหน่งกลับมา (ถ้าไม่เจอได้ -1)
var index = columnToLookup.indexOf(lookupValue) ; // *****
Logger.log(index) ; // 4.0 (แถวที่ 4+1)

// (4.) กรณีเจอค่าที่ค้นหา >> จับข้อมูลทั้งแถวที่ตรงกันออกมา
var rowData = sheet.getRange('A2:D13').getValues()[index] ;
Logger.log(rowData) ; // [DGG1520, DGG, อาบน้ำและตัดขนหมา (15-20 กก.) , 550.0]

// (5.) กรณีเจอค่าที่ค้นหา >> จับเซลล์ในคอลัมน์ที่ข้อมูลตรงกันออกมา
var columnToGet = 3 ;
var cellData = sheet.getRange('A2:D13').getValues()[index][columnToGet-1] ;
Logger.log(cellData) ; // อาบน้ำและตัดขนหมา (15-20 กก.)

// (6.) เซ็ตค่าที่จับมาได้ ไปที่เซลล์ G2
var lookupValue = ss.getRange('G2').setValues([[cellData]]) ;

```

### 8.11. ทำ Lookup แบบคืนหลายค่า

ปรับแต่งจาก

<https://stackoverflow.com/questions/36631641/javascript-indexof-method-with-multiple-values>

**ตัวอย่างที่ 1** - ในอาเรย์มีค่าที่ซ้ำกันหลายตัว เราจะ Lookup เพื่อจับตำแหน่งว่า ข้อมูลที่เราค้นหา อยู่ในตำแหน่งที่เท่าไรบ้าง

```

function Lookup_return_multiple(){
    var array = [ "test234", "test9495", "test234", "test93992", "test234" ] ;
    var newArr = [] ;
    for ( i=0 ; i<array.length ; i++) {
        if (array[i].indexOf("test234") >= 0 ) {
            newArr.push(i) ;
        }
    }
    Logger.log(newArr) ; // [0.0, 2.0, 4.0]
}

```

ตัวอย่างที่ 2 – พัฒนาจากตัวอย่างก่อนหน้านี้ เราจะจับค่าที่ซ้ำกันออกมา

```
function Lookup_return_multiple(){  
    var array = [ "test234", "test9495", "test234", "test93992", "test234" ] ;  
    var newArrData = [] ;  
    for ( i=0 ; i<array.length ; i++) {  
        if (array[i].indexOf("test234") >= 0 ) {  
            newArrData.push(array[i]) ;  
        }  
    }  
    Logger.log(newArrData) ;    // [test234, test234, test234]  
}
```

หรือใช้ forEach แทน for ก็ได้

```
function Lookup_return_multiple_using_forEach(){  
    var array = [ "test234", "test9495", "test234", "test93992", "test234" ] ;  
    var newArrData = [] ;  
    array.forEach(function(item,i){  
        if (array[i].indexOf("test234") >= 0 ) {  
            newArrData.push(array[i]) ;  
        } ; // Close - if  
    }) ; // Close - forEach  
    Logger.log(newArrData) ;    // [test234, test234, test234]  
}
```

---

## 8.12. วนลูปเช็คข้อมูลลงใน Google Sheets

---

วิธีที่ง่ายในการเช็คข้อมูลลงใน Google Sheets ก็คือใช้ Method **setValues** แต่ต้องจับมิติของเร้นจ์ให้พอดี กับมิติของข้อมูลที่จะเช็คลงไป

เช่น

เราจะเช็คข้อมูลลง Name range ที่มีมิติ 7x30 แต่ข้อมูลที่จะเช็คลงไปมีมิติ 7x18 **setValues** จะใช้งานไม่ได้ วิธีแก้ก็คือ ใช้ **for** 2 ลูปวนเช็คข้อมูลลงไปทีละตัว โดยใช้ **setValue** (ไม่มี s)

ตัวอย่าง – จับค่าใน Name range จากนั้นกรองเซลล์ว่างทิ้ง แล้วเช็คกลับไปยังตำแหน่งเดิมใหม่

การใช้งานโค้ดนี้ สามารถประยุกต์ใช้ในการโหลดข้อมูล ไปยังอีกชีทหรืออีกไฟล์ได้

```

var sheet = SpreadsheetApp.getActiveSpreadsheet().getSheetByName('Test') ;
var nr_dcItemsTable = 'dcItemsTable' // eg. sheet1!A13:G42

// จับข้อมูลใน Name range + กรองแถวว่างทิ้ง
var dcItemsTable = ss.getRangeByName(nr_dcItemsTable)
    .getRangeByName(nr_dcItemsTable)
    .getValues()
    .filter(function(row){
        return !row.every(function(cell){ // กรองตัวที่ไม่ใช่(!) -
            return cell == "" ; // ถ้าว่างทุกเซลล์ใน 1 แถว ค็นค่า จริง
        }) ; // close - every
    }) ; // close - filter

// จับเซลล์แรกใน Name range
var nr_dcItemsTable_1st = ss.getRangeByName(nr_dcItemsTable).getCell(1, 1) ; // A13
var nr_dcItemsTable_1st_col = nr_dcItemsTable_1st.getColumn() ; // 13
var nr_dcItemsTable_1st_row = nr_dcItemsTable_1st.getRow() ; // 1

// วนลูปเช็คค่าลงไปทีละเซลล์ - โดยเริ่มจากเซลล์แรกใน Name range
for(var r = 0 ; r < dcItemsTable.length ; r++){ // แต่ละแถว
    for(var c = 0 ; c < dcItemsTable[r].length ; c++){ // แต่ละคอลัมน์ในแต่ละแถว
        sheet.getRange(nr_dcItemsTable_1st_row + r , nr_dcItemsTable_1st_col + c )
            .setValue(dcItemsTable[r][c]) ;
    } // close - for #2 - inner
} // close - for #1 - outer

```

### 8.13. แทรกคอลัมน์ลงในอาเรย์ 2 มิติ

ตัวอย่างต่อไปนี้ แทรกคอลัมน์แรกเพิ่มให้กับอาเรย์ 2 มิติ

```

var arr2dim = [ // 1 2 3
    ["Linda", 27, "Bangkok" ] ,
    ["Lisa", 35, "Nontaburi" ] ,
    ["John", 42, "Petchaboon" ]
] ;

// แทรกคอลัมน์ลงไปที่ตำแหน่งแรกสุด - ใช้ unshift
arr2dim.map(function(row) {
    return row.unshift('1stCol') ;
}) ;

Logger.log(arr2dim) ;

```

ผล - สังเกตผลลัพธ์ที่ Log map เข้าไปเปลี่ยนอาเรย์ดั้งเดิม แต่ตัวที่เข้าไปเปลี่ยนคือ unshift

Logs

```

[ // 1 2 3 4
  [ 1stCol, Linda, 27.0, Bangkok ] ,
  [ 1stCol, Lisa, 35.0, Nontaburi ] ,
  [ 1stCol, John, 42.0, Petchaboon ]
]

```

#### 8.14. จับคอลัมน์ออกมาจากอาเรย์ 2 มิติ

ตัวอย่างต่อไปนี้ จับคอลัมน์ออกมาจากอาเรย์ 2 มิติ ใช้เทคนิคง่ายๆ แค่เพียง return ค่ากลับมา  
โค้ด

```
var arr2dim = [ // 1 2 3
                ["Linda", 27, "Bangkok" ] ,
                ["Lisa", 35, "Nontaburi" ] ,
                ["John", 42, "Petchaboon" ]
              ];

arr2dim.map(function(row) {
    return row[1] ;
}) ;

Logger.log(arr2dim) ; // ดูผลที่ Logs ----- >

var col1 = arr2dim.map(function(row) {
    return row[0] ;
}) ;

Logger.log(col1) ; // ดูผลที่ Logs ----- >
```

ผล

```
[O1] [ [Linda, 27.0, Bangkok] , [Lisa, 35.0, Nontaburi] , [John, 42.0, Petchaboon] ]
[O2] [ Linda, Lisa, John ]
```



บทที่ 9  
รู้จักกับ  
Google Services



---

## 9.1. สรุป OOP

---

Google Apps Script มีโครงสร้างการเขียนโปรแกรมเหมือนกับ Java Script

โครงสร้างการเขียนโปรแกรมของ Google Apps Script เป็นแบบ OOP(Object-Oriented Programming) หรือ การเขียนโปรแกรมเชิงวัตถุ

ผู้เขียนพยายามจะอธิบายเรื่องของ OOP ให้ง่ายที่สุด ดังนี้

Programming ก็คือ การเข้าไปควบคุมหรือสั่งงาน โดยสิ่งที่เราจะเขียนโปรแกรมเข้าไปควบคุม จะถูกมองว่าเป็นวัตถุ(Objects) เช่น ไฟล์, โฟลเดอร์, ตารางคำนวณ, เอกสาร เป็นต้น

วัตถุมี **คุณสมบัติ(Properties)** หรือ **คุณลักษณะ(Attributes)** เช่น สี, ขนาด เป็นต้น และวัตถุสามารถทำสิ่งต่างๆได้ตามคำสั่ง (**Methods**) เช่น ย้าย, จับ, หมุน เป็นต้น วัตถุสามารถประกอบไปด้วยวัตถุย่อยๆได้อีก เป็นวัตถุลูกหลาน ซึ่งก็มี Properties และ Methods ด้วยเช่นเดียวกัน

เมื่อ Google ออกแบบอะไรมาให้เราใช้งาน เราจะเรียกว่า **เซอร์วิส(Services)** หรือ **บริการ** ซึ่งภายในบริการจะประกอบไปด้วยวัตถุต่าง ๆ นานา

ตัวอย่างที่มักจะยกขึ้นมาเปรียบเทียบกัน ก็คือ บริการรถยนต์

บริการรถยนต์ มี วัตถุหลักที่เราจะควบคุม ก็คือ ตัวรถยนต์ มี **คุณสมบัติ(Properties)** สี, น้ำหนัก, ยี่ห้อ เป็นต้น และรถยนต์สามารถทำตามคำสั่งได้(**Methods**) เช่น เร่งความเร็ว, เหยียบเบรก เป็นต้น ตัวรถยนต์ประกอบไปด้วยวัตถุย่อยๆมากมาย เช่น ประตู, ครีซ, คันเร่ง, ล้อ เป็นต้น

เมื่อค่ายรถยนต์ทำการมาขาย เขาได้ทำรถยนต์มาเรียบร้อยแล้ว หรือ มีบริการเตรียมไว้แล้ว ในส่วนของเราหรือผู้ใช้งาน ก็คือ **เลือกรถที่ใช่รุ่นที่ชอบ** จากนั้นก็ นำรถมาใช้งานตามที่รถยนต์จะทำได้

---

## 9.2. Google Services (G Suite services)

---

G Suite services

<https://developers.google.com/apps-script/reference>

Google Services ก็คล้ายกับการทำรถยนต์ออกมาขาย

Google Services หรือ บริการจาก Google สำหรับนักพัฒนาหรือนักเขียนโปรแกรม เพื่อควบคุม การใช้งาน Google Apps มีมากกว่า 30 เซอร์วิส(Services) เช่น เซอร์วิส Spreadsheet, เซอร์วิส Document เป็นต้น โดยแต่ละเซอร์วิสประกอบไปด้วย **วัตถุ(Objects)** มากมายหลายตัว ซึ่งบ่อยครั้งเรียกวัตถุว่า **คลาส(Class)** เช่น เซอร์วิส Spreadsheet ประกอบไปด้วย คลาส Range, คลาส Sheet เป็นต้น

**เซอร์วิสของ Google เป็นสิ่งที่ Google จัดเตรียมไว้ให้แล้ว** ที่เหลือเราก็แค่เลือกและนำมาโปรแกรมให้ตรงตามวัตถุประสงค์ของเรา โดยการเขียนโปรแกรมเข้าไปควบคุม ตาม Methods หรือ Properties ที่ Objects หรือ Class มีให้

เช่น

เซอร์วิส Spreadsheet เริ่มต้นจากคลาส SpreadsheetApp ซึ่งมี Method เช่น openById() ใช้สำหรับเปิดไฟล์ Google Sheet โดยระบุ ID ของไฟล์ลงใน () นอกจากนี้คลาส SpreadsheetApp มี Property เช่น BandingTheme ซึ่งเก็บชื่อของรูปแบบของ Alternating Colors ไว้

### 9.3. การเรียกใช้ Google Services

การเขียนโปรแกรมเพื่อใช้บริการ Google Services อาศัยการเข้าไปจับวัตถุเริ่มต้นก่อน จากนั้นจึงค่อยๆ ลงลึกลงไป จนถึงวัตถุที่เราจะควบคุม ซึ่งเราเรียกว่า Chaining(การร้อยโซ่)

เมื่อจับสิ่งที่ต้องการได้แล้ว ก็กำหนด Properties หรือใช้ Methods สั่งงานให้วัตถุทำอะไรๆ ไป

Google Services มีโครงสร้างการเรียกใช้งาน ได้ดังนี้

```
GlobalObjectName.methodName(argument1, argument2, ..., argumentN) ;
```

GlobalObjectName เป็นชื่อคลาสในเซอร์วิส เช่น MailApp, SpreadsheetApp, DocumentApp เป็นต้น

ตัวอย่าง โค้ดต่อไปนี้ เรียกใช้ Method ชื่อ sendMail() ในคลาส GmailApp เพื่อส่งอีเมล

```
GmailApp.sendEmail('claire@example.com', 'Subject line', 'This is the body.');
```

ตัวอย่างวัตถุรถยนต์

การเรียกใช้งานเซอร์วิส เพื่อเข้าไปควบคุมวัตถุหรือคลาส เพื่อให้เข้าใจง่าย ขอยกตัวอย่าง วัตถุรถยนต์ ตัวอย่างต่อไปนี้ เราจะสั่งงานให้รถยนต์ขับที่ความเร็ว 80 ซึ่งสามารถเขียนเป็นโค้ดได้ดังต่อไปนี้

```
// Object.Object.Method(Value)  
รถยนต์.คันเร่ง.เหยียบคันเร่ง(80) ;
```

ตัวอย่างการเรียกใช้คลาส Math

คลาส Math เป็น Built-in Services ของ Google ที่มีการใช้งานเหมือนคลาส Math ของ JavaScript ตัวอย่างโค้ดต่อไปนี้ เรียกใช้ Method ชื่อ pow() ของคลาส Math เพื่อคำนวณเลขยกกำลัง จากนั้นเก็บค่าไว้ในตัวแปร numPow

Math object ของ JavaScript

[https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global\\_Objects/Math](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math)

```
var numPow = Math.pow(2,3) ;  
Logger.log(numPow) ; // พิมพ์ : 8
```

## เรียกใช้บริการในคลาสย่อย

กรณี Method ที่เราเรียกใช้บริการ คื้นค่ากลับมาเป็นคลาสอื่น ยกตัวอย่าง

```
DocumentApp.create('New document') ;
```

Method ชื่อ create() คื้นค่ากลับมาเป็นคลาส Document เราจึงสามารถร้อย Method ของคลาสเข้าด้วยกันในบรรทัดเดียว(Chaining method) โดยใช้ . (Period หรือ จุด) ต่อกันไปเรื่อยๆ เพื่อเข้าถึงวัตถุหรือคลาสที่เราต้องการจะควบคุม

ตัวอย่าง โค้ดต่อไปนี้จะสร้างไฟล์ Google Docs ไฟล์ใหม่ โดยตั้งชื่อไฟล์ว่า **New document** และพิมพ์ย่อหน้าที่มีข้อความ **New paragraph** ลงไป

```
// สร้างไฟล์ Google Docs ชื่อ New document ... แล้วจับใส่ตัวแปรชื่อ doc ไว้
var doc = DocumentApp.create('New document') ;

// จับไปที่ Body ของเอกสาร New document ใส่ตัวแปรชื่อ body ไว้
var body = doc.getBody() ;

// พิมพ์ต่อท้าย ด้วยย่อหน้าที่มีข้อความ New paragraph
body.appendParagraph('New paragraph.') ;
```

โค้ดต่อไปนี้จะให้ผลลัพธ์เหมือนกับข้างบน แต่เขียนโค้ดโดยร้อยกันเป็นบรรทัดเดียว

```
DocumentApp.create('New document').getBody().appendParagraph('New paragraph.') ;
```

โค้ดข้างต้น เขียนไว้คนละบรรทัดก็ได้ ดังนี้

```
DocumentApp.create('New document')
    .getBody()
    .appendParagraph('New paragraph.') ;
```

## 9.4. Enum หรือ Enumeration

Enum (Enumeration) เป็นการแจกแจงคุณสมบัติของวัตถุ ว่าวัตถุมีคุณสมบัติอะไรบ้าง เพื่อที่เราจะสามารถเข้าไปเปลี่ยนหรือดึงค่าของคุณสมบัตินั้นออกมาได้

ยกตัวอย่าง

**วัตถุลักษณะ** มีคุณสมบัติอะไรบ้าง ผู้ผลิตก็จะแจกแจงมาให้เลย เช่น สี, ขนาด, ราคา เป็นต้น จากนั้นก็มารวมกันไว้ แล้วตั้งชื่อว่า **ENUM** ลักษณะ เป็นต้น

ในกรณีของ Google Apps วัตถุต่างชนิดกัน สามารถมี Enum ตัวเดียวกันได้ เพราะคุณสมบัติใกล้เคียงกันมาก จึงไม่จำเป็นต้องมี Enum เฉพาะวัตถุ เพราะ Enum จะมากเกินไป เช่น ตัวอักษร(Characters) และย่อหน้า(Paragraph) ใช้ Enum ตัวเดียวกันได้ แต่คุณสมบัติใน Enum บางตัวอาจไม่มี

## ยกตัวอย่าง Enum Attribute ในคลาส Document

Enum Attribute(คุณลักษณะ) เป็นการแจกแจงคุณลักษณะของวัตถุ เช่นคุณสมบัติ BOLD ก็คือคุณสมบัติตัวหนาของอักษร เป็นต้น

เราสามารถเข้าถึง Enum Attribute ได้โดยใช้โค้ด ที่มีโครงสร้างดังนี้

```
// Class Enum Property  
DocumentApp.Attribute.[Property]
```

โดย Properties ใน Enum Attribute สามารถดูได้จากตารางต่อไปนี้

( ที่มา <https://developers.google.com/apps-script/reference/document/attribute> )

Property	Description
BACKGROUND_COLOR	The background color of an element (Paragraph, Table, etc) or document.
BOLD	The font weight setting, for rich text.
BORDER_COLOR	The border color, for table elements.
BORDER_WIDTH	The border width in points, for table elements.
CODE	The code contents, for equation elements.
FONT_FAMILY	The font family setting, for rich text.
FONT_SIZE	The font size setting in points, for rich text.
FOREGROUND_COLOR	The foreground color setting, for rich text.
HEADING	The heading type, for paragraph elements (for example, DocumentApp.).
HEIGHT	The height setting, for image elements.
HORIZONTAL_ALIGNMENT	The horizontal alignment, for paragraph elements (for example, DocumentApp.).
INDENT_END	The end indentation setting in points, for paragraph elements.
INDENT_FIRST_LINE	The first line indentation setting in points, for paragraph elements.
INDENT_START	The start indentation setting in points, for paragraph elements.
ITALIC	The font style setting, for rich text.
GLYPH_TYPE	The glyph type, for list item elements.
LEFT_TO_RIGHT	The text direction setting, for rich text.
LINE_SPACING	The line spacing setting as a multiplier, for paragraph elements.
LINK_URL	The link URL, for rich text. The default link style (foreground color, underline) is automatically applied.
LIST_ID	The ID of the encompassing list, for list item elements.
MARGIN_BOTTOM	The bottom margin setting in points, for paragraph elements.
MARGIN_LEFT	The left margin setting in points, for paragraph elements.
MARGIN_RIGHT	The right margin setting in points, for paragraph elements.
MARGIN_TOP	The top margin setting in points, for paragraph elements.
NESTING_LEVEL	The item nesting level, for list item elements.
MINIMUM_HEIGHT	The minimum height setting in points, for table row elements.
PADDING_BOTTOM	The bottom padding setting in points, for table cell elements.
PADDING_LEFT	The left padding setting in points, for table cell elements.
PADDING_RIGHT	The right padding setting in points, for table cell elements.
PADDING_TOP	The top padding setting in points, for table cell elements.
PAGE_HEIGHT	The page height setting in points, for documents.
PAGE_WIDTH	The page width setting in points, for documents.
SPACING_AFTER	The bottom spacing setting in points, for paragraph elements.
SPACING_BEFORE	The top spacing setting in points, for paragraph elements.
STRIKETHROUGH	The strike-through setting, for rich text.
UNDERLINE	The underline setting, for rich text.
VERTICAL_ALIGNMENT	The vertical alignment setting, for table cell elements.
WIDTH	The width setting, for table cell and image elements.

## ตัวอย่าง

ให้สร้างไฟล์ Google Docs จากนั้น สร้างโปรเจ็ค Google Apps Script แบบฝัง และเขียนโค้ดดังต่อไปนี้ลงไป

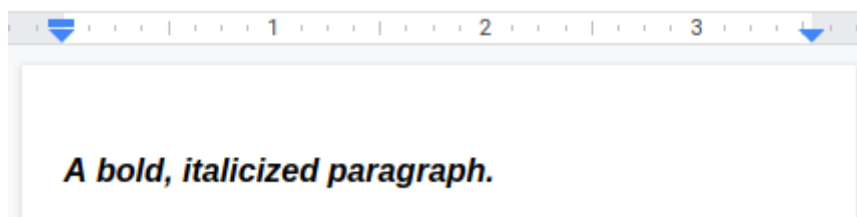
```
function getAttributes_test() {  
    // จัปไปที่ส่วน Body ของไฟล์ Google Docs ที่โค้ดฝังอยู่  
    var body = DocumentApp.getActiveDocument().getBody();  
  
    // พิมพ์ย่อหน้าที่มีข้อความต่อท้ายสุด + ทำตัวหนา + ทำตัวเอียง  
    var par = body.appendParagraph('A bold, italicized paragraph.');
```

`par.setBold(true);`    // ทำตัวหนา  
`par.setItalic(true);`    // ทำตัวเอียง

```
    // จัป Attributes ของย่อหน้าบางบน ใสตัวแปร atts  
    var atts = par.getAttributes();  
  
    Logger.log(atts);    // Logs ดู Attributes ของย่อหน้า  
    // ดูผลที่ Logs ----- >  
  
    // วนลูป Logs ดู Attributes(Property และ Value) ของย่อหน้า ที่เก็บอยู่ในตัวแปร atts  
    for (var att in atts) {  
        //        Property    Value  
        Logger.log(att + ":" + atts[att]);    // ดูผลที่ Logs ----- >  
    }    // จบลูป for  
}    // จบฟังก์ชัน
```

ผลหลังรันโค้ด

พิมพ์ย่อหน้าที่มีข้อความ **A bold, italicized paragraph.** และ กำหนดตัวอักษรเป็นตัวหนา และเอียง  
ทั้งย่อหน้า ตามภาพ



ผลจากบรรทัด `Logger.log(atts);` ได้ผลลัพธ์ดังต่อไปนี้ - จะเห็นว่า `atts` เป็นตัวแปร Object หากจะจับแต่ละ Item ในตัวแปร Object ออกมาดู จึงต้องใช้ `for/in` วนลูปเข้าไป

### Logs

```
[ 01 ] {FONT_SIZE=null, ITALIC=true, HORIZONTAL_ALIGNMENT=Left,  
INDENT_END=0.0, INDENT_START=0.0, LINE_SPACING=1.15, LINK_URL=null,  
UNDERLINE=null, BACKGROUND_COLOR=null, INDENT_FIRST_LINE=0.0,  
LEFT_TO_RIGHT=true, SPACING_BEFORE=0.0, HEADING=Normal,  
SPACING_AFTER=0.0, STRIKETHROUGH=null, FOREGROUND_COLOR=null, BOLD=true,  
FONT_FAMILY=null}
```

ผลจากวงเล็บ `for/in` เพื่อดู Attributes(Property และ Value) ของย่อหน้าได้ผลลัพธ์ดังต่อไปนี้

Properties ต่างๆที่ปรากฏใน Logs มีอยู่ในตาราง Enum Attribute (แต่ไม่ครบทั้งหมด) นอกจากนี้แต่ละ Property มี Value(ค่าของคุณสมบัติ) ระบุไว้ด้วย เช่น **ITALIC:true** ก็คือ เป็นอักษรตัวเอียง(จริง)

#### Logs

```
[01] FONT_SIZE:null
[02] ITALIC:true           // โค้ดตั้งเป็นตัวเอียงไว้
[03] HORIZONTAL_ALIGNMENT:Left
[04] INDENT_END:0
[05] INDENT_START:0
[06] LINE_SPACING:1.15
[07] LINK_URL:null
[08] UNDERLINE:null
[09] BACKGROUND_COLOR:null
[10] INDENT_FIRST_LINE:0
[11] LEFT_TO_RIGHT:true
[12] SPACING_BEFORE:0
[13] HEADING:Normal
[14] SPACING_AFTER:0
[15] STRIKETHROUGH:null
[16] FOREGROUND_COLOR:null
[17] BOLD:true           // โค้ดตั้งเป็นตัวหนาไว้
[18] FONT_FAMILY:null
```

หมายเหตุ : ที่มาของโค้ดข้างต้นก็คือ

<https://developers.google.com/apps-script/reference/document/body#getattributes>

`getAttributes()` เป็น Method ในคลาส Body/Text ใช้จับ Attributes ของวัตถุ โดยจะคืนค่ากลับมาเป็น Enum Attribute

---

## 9.5. Interfaces

Interfaces เป็นบริการหรือเป็นส่วนที่บอกว่าวัตถุนั้นๆ สามารถทำอะไรได้บ้าง หรือเป็นกลุ่มของเมธอด (Methods) ของวัตถุ Interfaces มักใช้เรียก Methods รวมๆของวัตถุ

โดยปกติ ผู้เขียนมักไม่ค่อยใช้คำว่า Interfaces ในการสื่อถึง Methods ของวัตถุหรือคลาส แต่จะใช้คำว่า Method อย่างเดียวอยู่บ่อยๆ

แต่ที่ต้องทำความเข้าใจกับคำว่า Interfaces เพราะ Method ของวัตถุบางตัว คืนค่ากลับมาเป็น Interfaces ของวัตถุอีกตัวหนึ่ง ก็คือ คืนค่ากลับมาเป็น Methods ของวัตถุอีกตัวหนึ่งนั่นเอง ฉะนั้น เราจะได้ไม่ต้องงงว่า Interfaces คืออะไร

อย่าสับสนระหว่าง User Interfaces และ Interfaces

User Interfaces เป็นส่วนของหน้าต่างของโปรแกรม เช่น เมนูหรือไชด์บาร์ เป็นต้น ส่วน Interfaces ที่อธิบายในข้อนี้ เป็นกลุ่ม Methods ของวัตถุหรือคลาส

ยกตัวอย่าง Interface Element ของเซอร์วิส DocumentApp

<https://developers.google.com/apps-script/reference/document/element>

เอกสาร Google Docs ประกอบไปด้วย Elements ต่างๆ เช่น [ListItem](#)(หัวข้ออัตโนมัติ เช่น Bullets หรือ Numbering), [Paragraph](#)(ย่อหน้า) และ [Table](#)(ตาราง) เป็นต้น ทุกตัวก็คือ Element ซึ่งมี Methods กลุ่มหนึ่งที่ซ้ำกัน ก็เลยถูกจับมามีตรรกะร่วมกัน (Inherit หรือสืบทอดสิ่งที่ซ้ำกันออกมา) กลายเป็น Interface Element หรือ เป็นกลุ่ม Methods ของวัตถุ Element

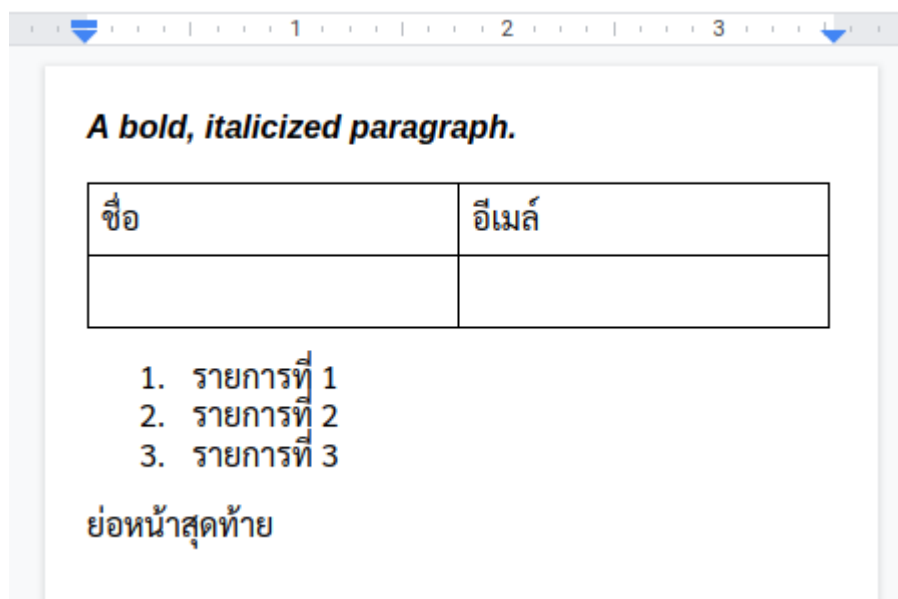
Methods ใน Interface Element ยกตัวอย่างเช่น `getAttributes()`, Method ตระกูล `as` เช่น `asParagraph()` เป็นต้น

อย่าสับสนกับ คลาส Element(ตามลิงค์ด้านล่าง) เพราะเป็นคนละตัวกัน คลาส Element เป็นวัตถุ ส่วน Interface Element เป็นกลุ่มของ Methods ของวัตถุ Element

( Class Element <https://developers.google.com/apps-script/reference/xml-service/element> )

ยกตัวอย่าง

องค์ประกอบในส่วน Body ในเอกสาร Google Docs มีตามภาพ



โค้ดต่อไปนี้ จับ Elements ใน Body ของเอกสาร Google Docs มา Log ดูว่ามีชนิดเป็นอะไรบ้าง

```
function testElem() {  
  var body = DocumentApp.getActiveDocument().getBody();  
  var elemInBody = body.getNumChildren();  
  
  for(var x = 0 ; x < elemInBody ; x++){  
    Logger.log(body.getChild(x).getType());  
  };  
}
```



Logs

```
[ 01 ] PARAGRAPH
[ 02 ] TABLE
[ 03 ] LIST_ITEM
[ 04 ] LIST_ITEM
[ 05 ] LIST_ITEM
[ 06 ] PARAGRAPH
```

หมายเหตุ

ดูรายละเอียดเพิ่มเติมได้จากลิงค์ต่อไปนี้

**getType()** - Method ใน Interface Element

<https://developers.google.com/apps-script/reference/document/element#gettype>

ใช้จับชนิดของ Element คืนค่ากลับมาเป็น Enum ElementType

**Enum ElementType** – แจกแจงชนิดของ Elements

<https://developers.google.com/apps-script/reference/document/element-type.html>

**getChild** – Method ในคลาส ContainerElement/Element

<https://developers.google.com/apps-script/reference/document/container-element#getchildchildindex>

<https://developers.google.com/apps-script/reference/xml-service/element#getchildname>

ใช้จับ Element ตามดัชนีลำดับ เช่น **getChild(0)** เป็นต้น คืนค่ากลับมาเป็น Element

**getNumChildren()** – Method ในคลาส ContainerElement

<https://developers.google.com/apps-script/reference/document/container-element#getnumchildren>

ใช้จับจำนวน Elements คืนค่ากลับมาเป็น เลขจำนวนเต็ม



# บทที่ 10

## คลาส Logger



## 10.1. คลาส Logger

Class Logger

<https://developers.google.com/apps-script/reference/base/logger?hl=th>

คลาส Logger เป็นหน้าต่าง Console ที่ใช้แสดงผลจากการรันโค้ด

Methods ของคลาส Logger มีดังต่อไปนี้

Method	Return type	Brief description
<a href="#">clear()</a>	void	Clears the log.
<a href="#">getLog()</a>	String	Returns a complete list of messages in the current log.
<a href="#">log(data)</a>	<a href="#">Logger</a>	Writes the string to the logging console.
<a href="#">log(format,values)</a>	<a href="#">Logger</a>	Writes a formatted string to the logging console, using the format and values provided.

## 10.2. log()

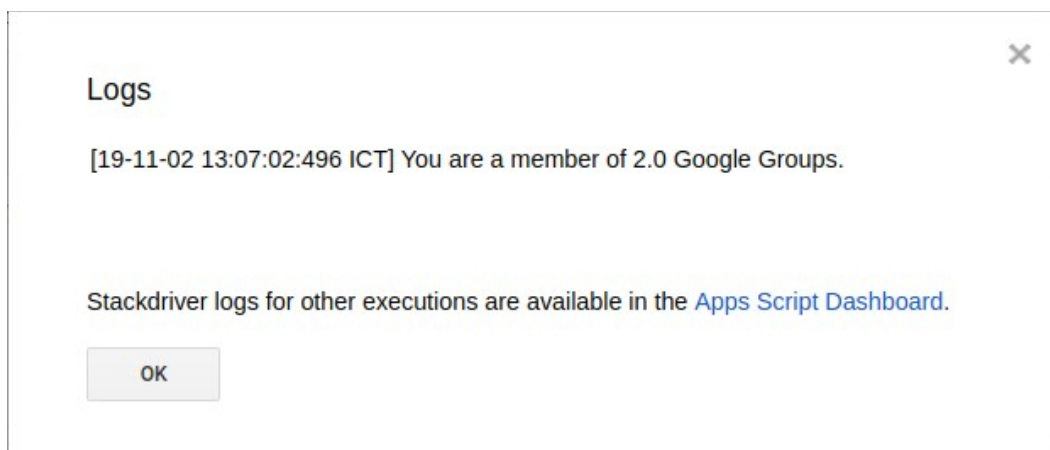
log() เป็น Method ที่ใช้เขียนผลจากการรันโค้ด ให้อยู่ในรูปแบบข้อความลงในหน้าต่าง Console (ขณะอยู่ในหน้าโปรเจกต์ Google Apps Script กด **<Ctrl><Enter>** เพื่อเปิด Logs )

สามารถใช้ **%s** ใส่ลงใน log() เพื่อรับค่าจากตัวแปรมาใส่ หรือ จะใส่ตัวแปรตรงๆลงใน log() ก็ได้ แต่ก็จะมีความแตกต่างกัน ให้สังเกตจากตัวอย่างต่อไปนี้

ตัวอย่าง

```
var groups = GroupsApp.getGroups();
Logger.log('You are a member of %s Google Groups.', groups.length);
```

ผล



อีกตัวอย่าง

```
var values = [1,2,3];
Logger.log('Values is %s' , values) ; // พิมพ์ใน Logger : Values is [1.0, 2.0, 3.0]
Logger.log('Values is' + values) ; // พิมพ์ใน Logger : Values is 1,2,3
```

# บทที่ 11

## Triggers



### 11.1. Triggers คืออะไร

Triggers ใช้รันฟังก์ชันแบบอัตโนมัติ เช่น เมื่อมีเหตุการณ์เกิดขึ้น เมื่อเปิดไฟล์ เป็นต้น

Triggers มี 2 แบบด้วยกันก็คือ Simple Triggers และ Installable Triggers

### 11.2. Simple Triggers

Simple Triggers

<https://developers.google.com/apps-script/guides/triggers>

Simple Triggers เป็นกลุ่มฟังก์ชันสวอน เช่น ฟังก์ชันที่ใช้ชื่อ `onOpen(e)` ชื่อนี้สวอนไว้สำหรับ Trigger โดยเฉพาะ ฟังก์ชันจะทำงานอัตโนมัติเมื่อไฟล์ Google Apps อย่าง Docs, Sheets, Slides หรือ Form ถูกเปิด อื่นๆมีดังต่อไปนี้

`onOpen(e)` รันเมื่อ ผู้ใช้งานเปิดไฟล์ Sheet, Docs, Sheets, Slides หรือ Form

`onEdit(e)` รันเมื่อ ยูสเซอร์เปลี่ยนค่าในเซลล์ใน Google Sheets

`onInstall(e)` รันเมื่อ ยูสเซอร์ติดตั้ง Addon ([add-on](#))

`doGet(e)` รันเมื่อ ยูสเซอร์เปิด Web App หรือเมื่อโปรแกรมส่ง HTTP GET request ไปที่ Web App (<https://developers.google.com/apps-script/guides/web>)

`doPost(e)` รันเมื่อ โปรแกรมส่ง HTTP POST request ไปที่ Web App

พารามิเตอร์ `e` ก็คือ **Event Object** ที่ส่งให้ฟังก์ชัน ซึ่งบรรจุไปด้วยข้อมูลและเนื้อหาเกี่ยวกับ Event ที่เกิดขึ้น

#### 11.2.ก.) ตัวอย่าง เมื่อคอลัมน์ A เปลี่ยน พิมพ์เวลาไว้ที่คอลัมน์ B

โค้ดต่อไปนี้เป็น Google Apps Script แบบฝังในไฟล์ Google Sheet เมื่อเปลี่ยนแปลงค่าในเซลล์ในคอลัมน์ A เซลล์ข้างๆในคอลัมน์ B ที่ติดกัน จะกรอกเวลาลงไปอัตโนมัติ

โค้ดนี้ถ้ารันปกติจะ Error เพราะไม่มี `e` ส่งผ่านมาให้บันทึกโปรเจ็คของ Google Apps Script ก็ใช้ได้ แล้ว จากนั้นให้ไปลองคีย์ค่าต่างลงในคอลัมน์ A

```
function onEdit(e) {  
  Logger.log(e); // ดูว่า e มีอะไร ----->  
  Logger.log(e.source.getActiveSheet().getName()); // ดูชื่อชีทที่จับมา ----->  
  Logger.log(e.range.getA1Notation()); // ดูเซลล์ที่จับมา ----->  
  var formattedDate = Utilities.formatDate(new Date(), "GMT", "dd-MMM-yyyy HH:mm:ss") ;  
  if (e.range.getColumn() == 1) { // ถ้าเป็นคอลัมน์ A  
    e.range.offset(0, 1).setValue(formattedDate) ;  
  }  
}
```

`formatDate`

<https://developers.google.com/apps-script/reference/utilities/utilities.html#formatdate, -timezone, -format>

ผล

	A	B
1	Change !!!	10-Nov-2019 1:54:47
2		

หากดูผลการ Logs เพื่อดูว่า e เป็นอะไรจะได้ Logs ดังนี้

```
[ 01 ] { authMode=LIMITED ,  
        range=Range ,  
        source=Spreadsheet ,  
        user=wasankds@gmail.com ,  
        value=ccc }  
[ 02 ] Sheet1  
[ 03 ] A1
```

### 11.3. Event Objects

#### Event Objects

<https://developers.google.com/apps-script/guides/triggers/events>

เมื่อ Triggers ทำงาน Google Apps Script จะส่งผ่านสิ่งที่เรียกว่า Event Object ซึ่งบรรจุไปด้วยข้อมูลและเนื้อหาเกี่ยวกับ Event ที่เกิดขึ้น เช่น เมื่อ Submit form (คลิกปุ่ม SUBMIT เพื่อส่งสิ่งที่กรอกลงในฟอร์ม) ก็จะส่งผ่าน Event Object ชื่อ Form Submit

#### ตัวอย่าง

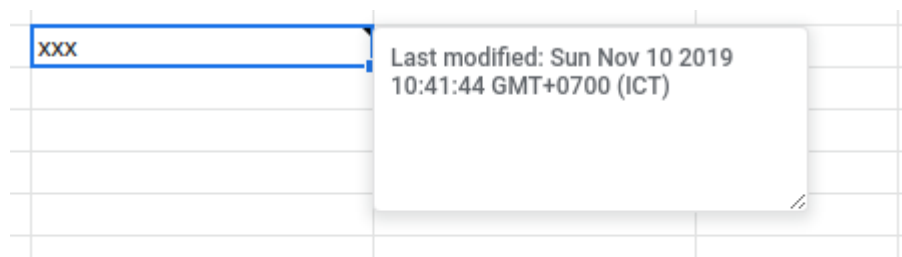
โค้ดต่อไปนี้จะใส่ Note ลงในเซลล์ที่มีการเปลี่ยนแปลงโดยอัตโนมัติ โดย Note ที่ใส่เป็น วันเวลาที่แก้ไขค่าในเซลล์

โปรเจ็ค Google Apps Script ตัวนี้ให้สร้างแบบฝังในไฟล์ Google Sheet

โค้ดนี้ถ้ารันปกติ จะ Error เพราะไม่มี e ส่งผ่านมาให้ฟังก์ชัน การใช้งานฟังก์ชันนี้ให้บันทึกโปรเจ็ค Google Apps Script โค้ดก็ใช้ได้แล้ว โดยฟังก์ชันจะรันเอง เมื่อมีการเปลี่ยนแปลงค่าใน Google Sheet

```
function onEdit(e){  
    var range = e.range ;  
    range.setNote('Last modified: ' + new Date()) ;  
}
```

ผล



## 11.4. Installable Triggers

Installable Triggers

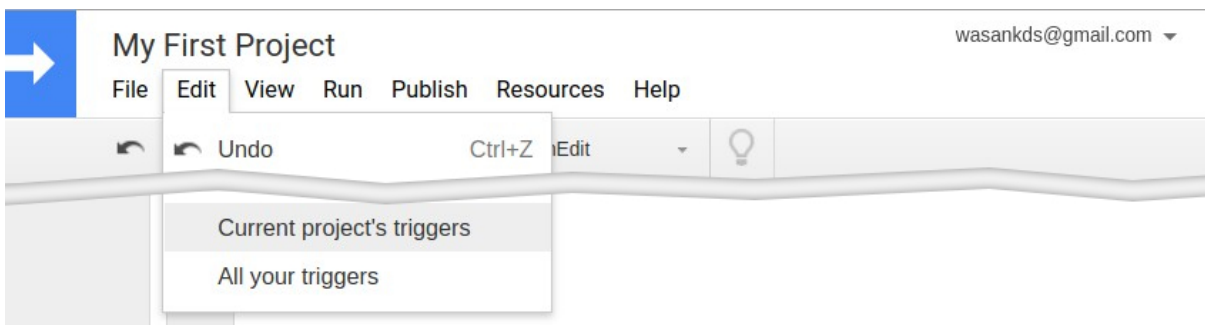
<https://developers.google.com/apps-script/guides/triggers/installable>

Installable Triggers ต้องมีการเซตก่อน เพื่อผูกฟังก์ชันที่จะรันกับ Event ตัวอย่างเช่น ให้ทำฟังก์ชัน `sendMail` เมื่อมีบรรทัดข้อมูลเพิ่มเข้ามาใน Google Sheets เมื่อคลิกปุ่ม Submit แบบฟอร์ม เป็นต้น

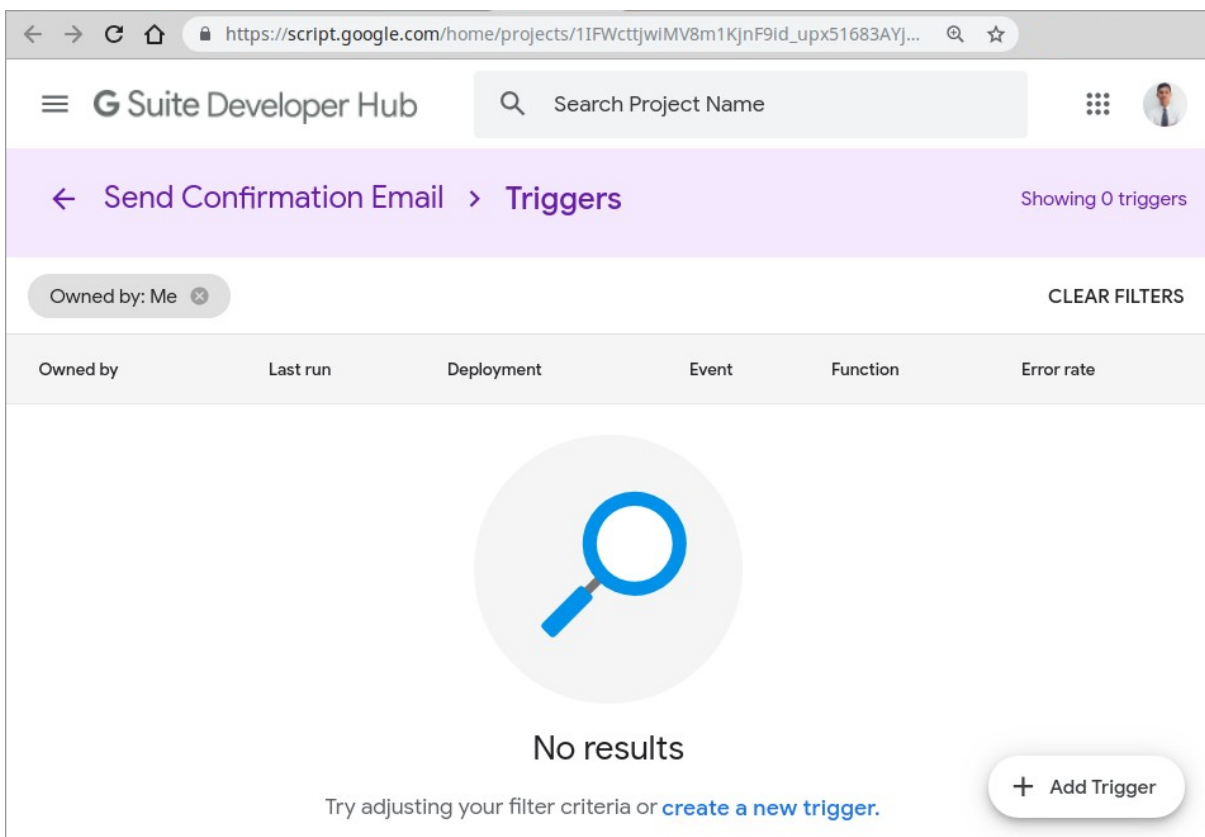
เราสามารถสร้าง Installable Triggers ได้โดย

ที่หน้าโปรเจกต์ Google Apps Script ไปที่เมนู **Edit → Current Project's Triggers** จะเข้าสู่หน้า

Triggers



ที่หน้า **Triggers** ตามภาพ คลิกที่ปุ่ม **Add Trigger** จะปรากฏหน้าต่างมาให้ตั้งค่า Trigger ตามภาพถัดไป





ช่อง [Choose which function to run](#)

ใช้เลือกฟังก์ชันในโปรเจกต์ Google Apps Script ที่จะรัน

ช่อง [Select event source](#)

ใช้เลือกว่าแหล่งของ event จะมาจากไหน เช่น Form ก็คือ มาจาก Form

ช่อง [Select event type](#)

ใช้เลือก event เช่น **On form submit** ก็คือ เมื่อคลิกที่ปุ่ม Submit เพื่อส่งคำตอบในการกรอกแบบฟอร์ม

**Add Trigger for Send Confirmation Email**

Choose which function to run  
sendEmail

Choose which deployment should run  
Head

Select event source  
From form

Select event type  
On form submit

Failure notification settings  
Notify me daily

Cancel Save

เมื่อคลิกปุ่ม Save จะได้ Trigger ตามภาพ

← Send Confirmation Email > Triggers Showing 1 trigger

Owned by: Me CLEAR FILTERS

Owned by	Last run	Deployment	Event	Function	Error rate
Me	-	Head	From form - On form submit	sendEmail	-

---

## 11.5. newTrigger() Method

---

How to create an automated calendar with Google Apps Script with open source on top  
<https://opensource.com/article/19/1/automate-calendar>

`deleteTrigger(trigger)`

[https://developers.google.com/apps-script/reference/script/script-app#deleteTrigger\(trigger\)](https://developers.google.com/apps-script/reference/script/script-app#deleteTrigger(trigger))

คลาส Class ScriptApp

[https://developers.google.com/apps-script/reference/script/script-app#newTrigger\(String\)](https://developers.google.com/apps-script/reference/script/script-app#newTrigger(String))

ใช้จัดการการเผยแพร่สคริปต์และ Triggers ใช้สร้าง Triggers และปล่อยสคริปต์ในลักษณะ Script as

a service

คลาส TriggerBuilder

<https://developers.google.com/apps-script/reference/script/trigger-builder.html>

`newTrigger(functionName)` – Method ในคลาส ScriptApp

[https://developers.google.com/apps-script/reference/script/script-app#newTrigger\(functionName\)](https://developers.google.com/apps-script/reference/script/script-app#newTrigger(functionName))

`newTrigger()` ใช้สร้าง Installable trigger โดยพารามิเตอร์ `functionName` คือ ฟังก์ชันที่จะรันเมื่อ

Trigger ทำงาน โดยจะคืนค่ากลับมาเป็นคลาส `TriggerBuilder`

ตัวอย่าง – โค้ดต่อไปนี้จะรันฟังก์ชัน `updateEvents` ทุก 5 นาที โดยเมื่อรันโค้ดนี้ Installable trigger จะถูกสร้างด้วย

```
function createTrigger() {  
    ScriptApp.newTrigger('updateEvents')  
        .timeBased()  
        .everyMinutes(5)  
        .create() ;  
}
```

ตัวอย่าง – โค้ดต่อไปนี้จะลบ Trigger ทั้งหมด ภายในโปรเจ็ค

```
var triggers = ScriptApp.getProjectTriggers();  
for (var i = 0; i < triggers.length; i++) {  
    ScriptApp.deleteTrigger(triggers[i]);  
}
```

บทที่ 12  
การใช้โปรเจ็ค  
Google Apps Script  
ในไฟล์อื่น



## 12.1. การนำโปรเจกต์ Google Apps Script ไปใช้ในไฟล์อื่น

How to export / import a Google Apps Script as a file into a new spreadsheet?

<https://webapps.stackexchange.com/questions/57920/how-to-export-import-a-google-apps-script-as-a-file-into-a-new-spreadsheet>

### 12.1.ก.) ปัญหาการใช้งาน 1 โปรเจกต์(แบบฝัง) ในหลายไฟล์

กรณีโปรเจกต์ Google Apps Script ของเราเป็นแบบฝังไฟล์ หากโปรเจกต์ ฝังอยู่ในไฟล์ไหน ก็จะสามารถเรียกใช้ได้เฉพาะในไฟล์นั้น

หากเราก็อปปี้ไฟล์ Google Apps เช่นก็อปปี้ไฟล์ Google Sheets มาอีกไฟล์ โปรเจกต์ที่สร้างไว้จะติดมาด้วย แต่โปรเจกต์ที่ติดมาด้วย แม้จะยังใช้ชื่อเดียวกัน แต่ได้กลายเป็นคนละโปรเจกต์ไปแล้ว

เมื่อเป็นแบบดังกล่าว ก็เกิดปัญหาการซ้ำซ้อนของโปรเจกต์ และเวอร์ชันของโปรเจกต์ ฉะนั้น 1 โปรเจกต์สำหรับ Google Apps 1 ไฟล์ จึงไม่เวิร์ค

ฉะนั้น จึงต้องมีวิธี ในการนำโปรเจกต์ Google Apps Script ไปใช้ในหลายๆไฟล์ การสร้าง Addon เป็นทางออกที่ดี(แต่ผ่านการอนุมัติโคตรยาก) อย่างไรก็ตาม มีวิธีการแชร์โปรเจกต์(คนละอย่างกับการแชร์สิทธิการใช้งานโปรเจกต์) ไปให้ไฟล์อื่นเรียกใช้งานได้ โดยยังไม่ต้องไปถึงขั้นการสร้าง Addon

### 12.1.ข.) การนำโปรเจกต์ Google Apps Script ไปใช้ในไฟล์อื่น

#### ขั้นตอน

1. (ที่ไฟล์หลักที่เราสร้างโปรเจกต์ Google Apps Script) สร้างเวอร์ชันให้กับโปรเจกต์

1.1 ที่โปรเจกต์ Google Apps Script ไปที่เมนู File → Manage Versions

1.2 (ที่หน้าต่าง Manage Versions ) บันทึกเวอร์ชัน

ก.) ตั้งชื่อเวอร์ชัน อะไรก็ได้ เช่น 1.0 Begin

ข.) คลิกปุ่ม Save new version จะได้เวอร์ชันของโปรเจกต์ตามภาพ

ค.) คลิกปุ่ม OK

**Manage versions**

A version is a snapshot of your project that can be used as a library in other projects.

Save new version

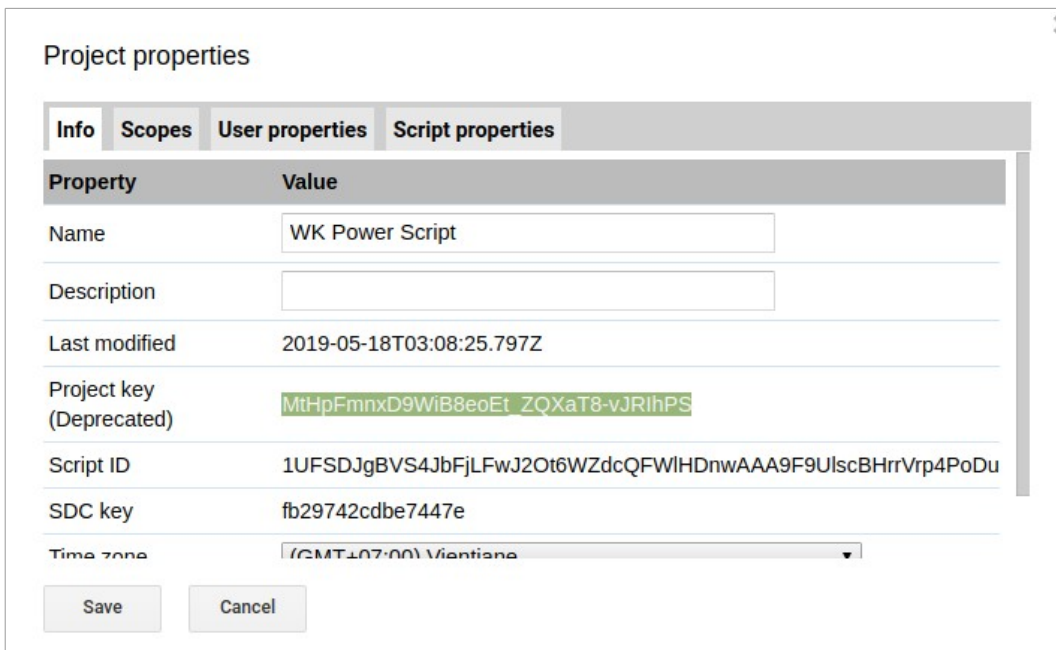
Version	Description	Saved by	Date
1	1.0 Begin	wasankds@gmail.com	2019-05-13 11:59

OK

## 2. (กลับมาที่โปรเจ็ค Google Apps Script) ดูคีย์ของโปรเจ็ค

2.1 ไปที่เมนู File → Project Properties

2.2 ที่หน้าต่าง Project Properties แท็บ Info หัวข้อ [Product key \(Deprecated\)](#) จดคีย์เอาไว้



Property	Value
Name	WK Power Script
Description	
Last modified	2019-05-18T03:08:25.797Z
Project key (Deprecated)	MtHpFmnxD9WiB8eoEt_ZQXaT8-vJRIhPS
Script ID	1UFSDJgBVS4JbFjLFwJ2Ot6WZdcQFWlHDnwAAA9F9UiscBHrrVrp4PoDu
SDC key	fb29742cdbe7447e
Time zone	(GMT+07:00) Vientiane

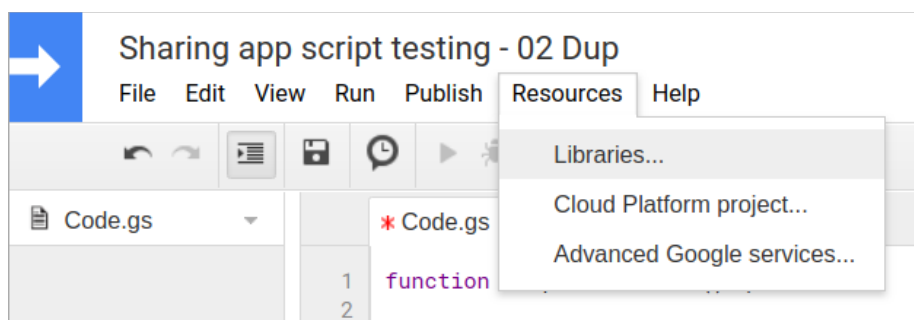
Save Cancel

## 3. ที่ไฟล์ Google Apps(ไฟล์ใหม่) ที่ต้องการจะเรียกใช้ Script – ลิงค์ Resource

3.1 สร้างโปรเจ็ค Google Apps Script ( Tools → Script Editor )

3.2 (ที่โปรเจ็ค Google Apps Script) ระบุที่แหล่งที่มาของโปรเจ็คที่จะเรียกโค้ดมาใช้งาน

ก.) ไปที่เมนู Resources → Libraries จะปรากฏหน้าต่าง Libraries



### 3.3 (ที่หน้าต่าง Libraries) เพิ่ม Resource ให้กับโปรเจ็ค

(โค้ดของโปรเจ็คที่เป็น Resource จะกลายเป็น Library ของโปรเจ็คนี้)

ก.) ช่อง Add Libraries ใส่ [Product Key](#) ลงไป (คีย์ของโปรเจ็คในไฟล์หลัก)

ข.) คลิกปุ่ม Add – แหล่งที่มาของโปรเจ็คจะถูกเพิ่มเป็น Library

ค.) ช่อง Version เลือก เวอร์ชันของโปรเจ็ค เช่น 1.0 begin

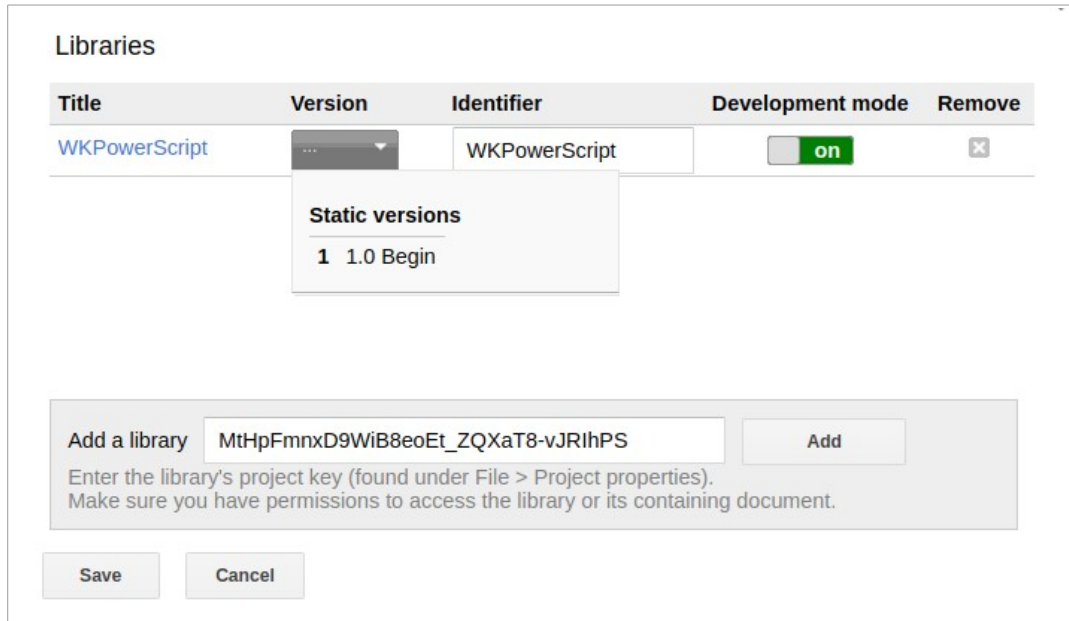
ง.) ช่อง Identifier **จำชื่อไว้** ตามภาพก็คือ WKPowerScript เพราะต้องนำไปใช้เรียกใช้โค้ดจากโปรเจ็คต้นทาง

จ.) Development mode เลือกเป็น On/Off แล้วแต่เรา

ถ้าต้นทางบันทึกเวอร์ชันนี้ใหม่ เช่น 1.0 Begin

ถ้าเลือกเป็น On เราจะเรียกใช้โค้ดล่าสุดได้ด้วย

ถ้าเลือกเป็น Off เราจะเรียกใช้เฉพาะโค้ดตอนที่ลิงค์ Resource โค้ดที่พัฒนาใหม่จะไม่ลิงค์มา



4. ที่ไฟล์ Google Apps(ไฟล์ใหม่) ที่ต้องการจะเรียกใช้ Script

4.1 เขียนโค้ดเพื่อเรียกใช้ฟังก์ชันจาก Resource

ตัวอย่างที่ 1

```
function myFunction() {  
  // เรียกใช้ฟังก์ชัน onOpen() จาก Resource WKPowerScript ที่ลิงค์กันอยู่  
  WKPowerScript.onOpen();  
}
```

ตัวอย่างที่ 2

```
// ฟังก์ชันนี้ ส่งผ่าน Agrument และ รับค่ากลับมา  
function BAHTTEXT(number) {  
  return WKPowerScript.BAHTTEXT(number);  
}
```

#### หมายเหตุ

ปัจจุบัน Google ได้สร้างฟังก์ชัน BAHTTEXT ใน Google Sheets ให้เป็นฟังก์ชัน Built-in แล้ว  
สามารถเรียกใช้ได้เลย ไม่ต้องเขียนสคริปต์อีกแล้ว

Resource มีฟังก์ชันอะไรให้เรียกใช้บ้าง สามารถทำ AutoComplete ได้เหมือนกับคลาสของ Object ทั่วๆไป



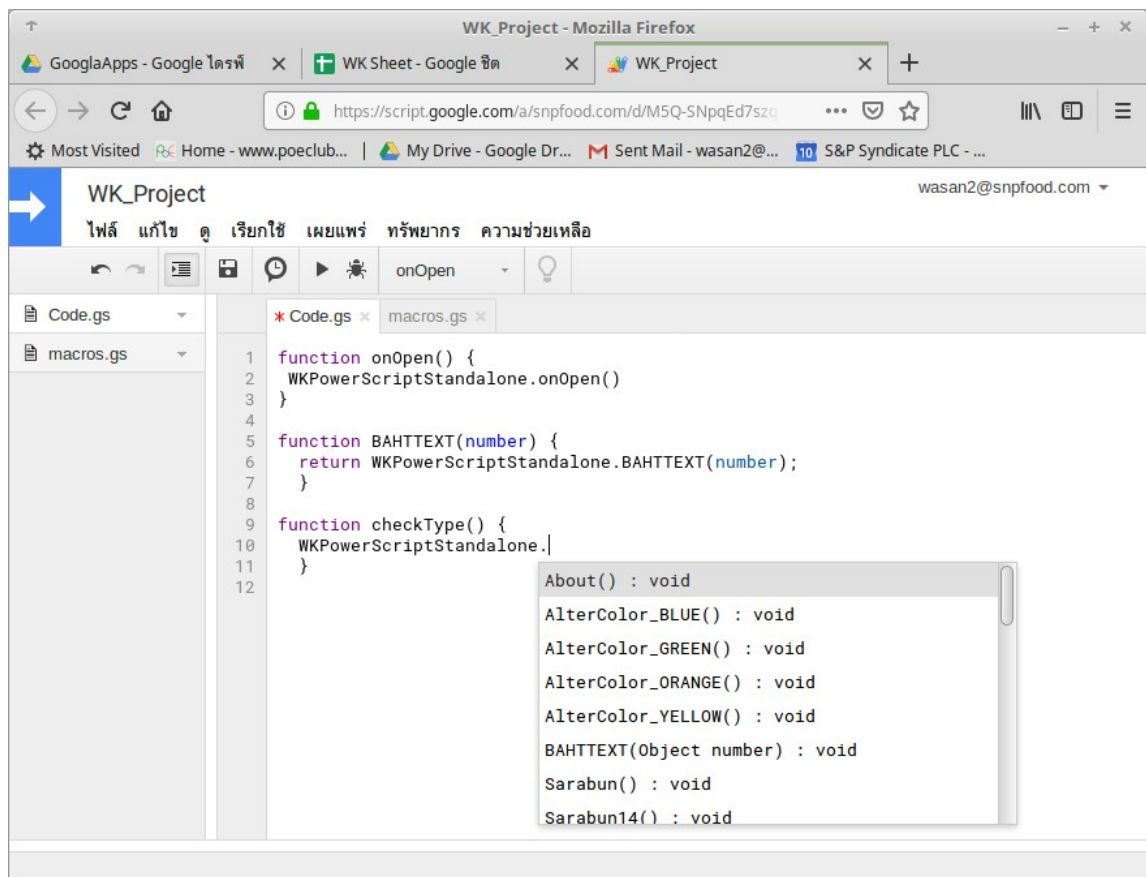
```
function onOpen() {
  WKPowerScript.onOpen();
}

function myFunction() {
  WKPowerScript.
}
```

About() : void  
AlterColor\_BLUE() : void  
AlterColor\_GREEN() : void  
AlterColor\_ORANGE() : void  
AlterColor\_YELLOW() : void  
BAHTTEXT(Object number) : void  
Sarabun() : void  
Sarabun14() : void

### 12.1.ค.) การนำโปรเจ็คไปใช้ในไฟล์อื่นแบบข้าม Account และ ข้าม Domain

การนำโปรเจ็คไปใช้ในไฟล์อื่นแบบข้าม Account และ ข้าม Domain สามารถทำได้โดย แชรโค้ดไปให้ บัญชี Google อื่นๆ หรือคนอื่นๆนอกโดเมนได้ เช่น จาก @gmail แชรไปให้ @snpfood ใช้ได้ เพียงแต่โปรเจ็ค Google Apps Script ที่ใช้เก็บ Library ของโค้ด ต้องเป็น Standalone



WK\_Project - Mozilla Firefox

GoogleApps - Google ไดรฟ์ WK Sheet - Google ชิต WK\_Project

https://script.google.com/a/snpfood.com/d/M5Q-SNpqEd7szq

WK\_Project wasan2@snpfood.com

ไฟล์ แก๊ช ดู เรียกใช้ เผยแพร่ ทรพยากร ความช่วยเหลือ

Code.gs macros.gs

```
1 function onOpen() {
2   WKPowerScriptStandalone.onOpen()
3 }
4
5 function BAHTTEXT(number) {
6   return WKPowerScriptStandalone.BAHTTEXT(number);
7 }
8
9 function checkType() {
10  WKPowerScriptStandalone.
11 }
12
```

About() : void  
AlterColor\_BLUE() : void  
AlterColor\_GREEN() : void  
AlterColor\_ORANGE() : void  
AlterColor\_YELLOW() : void  
BAHTTEXT(Object number) : void  
Sarabun() : void  
Sarabun14() : void

### 12.1.ง.) สรุป

วิธีการใช้งานโปรเจ็ค Google Apps Script 1 โปรเจ็คหลายไฟล์ ตามที่ได้อธิบายในข้อนี้ มีข้อดีที่เราสามารถเขียนโค้ดได้ในโปรเจ็คหลักตัวเดียว แล้วโปรเจ็คอื่นๆสามารถเรียกใช้โค้ดจากโปรเจ็คหลักได้

อย่างไรก็ดี **ไฟล์ที่เรียกใช้ก็ต้องสร้างโปรเจ็คอยู่ดี**

หากโปรเจ็คของเรา ยูสเซอร์จำเป็นต้องใช้โปรเจ็ค Google Apps Script กันเป็นจำนวนมาก และไม่อยากจะยุ่งยากกับการสร้างโปรเจ็คอีก ทางออก ก็คือ **การสร้าง Addon**