# ฟังก์ชันย่อย

# SubFunction

ผู้ช่วยศาสตราจารย์สมเกียรติ ช่อเหมือน (tko@webmail.npru.ac.th)

สาขาวิชาวิศวกรรมซอฟต์แวร์ คณะวิทยาศาสตร์และเทคโนโลยี

# บทนำ

- ฟังก์ชัน
- พารามิเตอร์
- การส่งค่า
- Arrow Function
- Anonymous Function
- Callback Function
- Refactor code

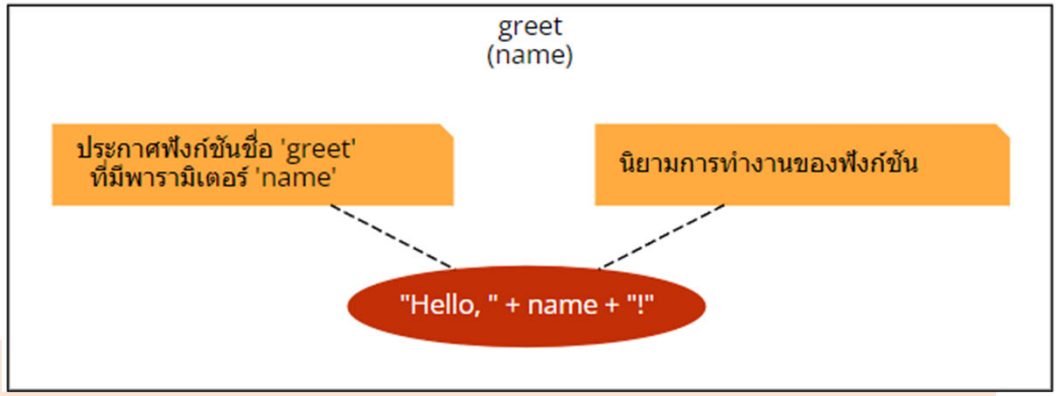
# ความหมายและประโยชน์ของฟังก์ชัน

ฟังก์ชัน **(Function)**

- หน่วยของโค้ดที่จัดรวบรวมการทำงานเฉพาะเจาะจง
- สามารถเรียกใช้งานได้ซ้ำ ๆ
- ทำให้โปรแกรมมีโครงสร้างที่ดี
- ง่ายต่อการจัดการ
- ลดการซ้ำซ้อนของโค้ด

# การประกาศและการนิยามฟังก์ชัน

- ประกาศและนิยามด้วยคำหลัก function
- ตามด้วยชื่อฟังก์ชัน
- รายการพารามิเตอร์ (ถ้ามี)
- บล็อกโค้ด



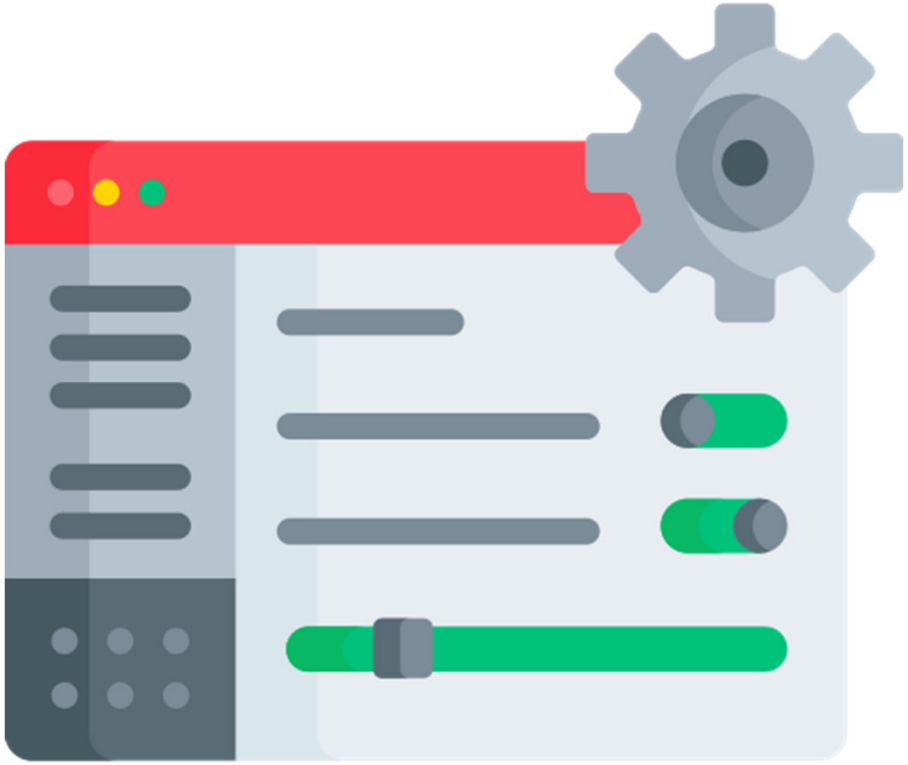

```
function greet(name) {
// ประกาศฟังก์ชันชื่อ 'greet' ที่มีพารามิเตอร์ 'name'
    return "Hello, " + name + "!";  // นิยามการทำงานของฟังก์ชัน
}
```

# การเรียกใช้ฟังก์ชันในโปรแกรม:

- การเรียกใช้ฟังก์ชันในโปรแกรม
- โดยการใส่ชื่อฟังก์ชันตามด้วยวงเล็บ
- พารามิเตอร์ (ถ้ามี) ภายในวงเล็บ

```
let greeting = greet("John");
// เรียกใช้ฟังก์ชัน 'greet' ด้วยพารามิเตอร์ 'John'
console.log(greeting);  // Output: Hello, John
```

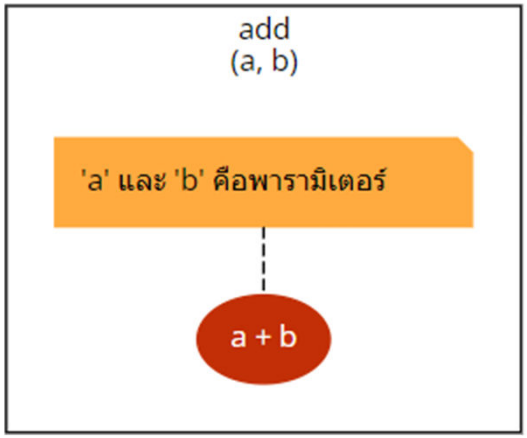
# พารามิเตอร์



# พารามิเตอร์และการใช้งาน

พารามิเตอร์ **(Parameter)**

- ตัวแปรที่ใช้ในการรับค่าที่ส่งมาจากภายนอกของฟังก์ชัน
- พารามิเตอร์ทำให้ฟังก์ชันมีความยืดหยุ่น
- สามารถใช้งานได้ในสถานการณ์ต่าง ๆ



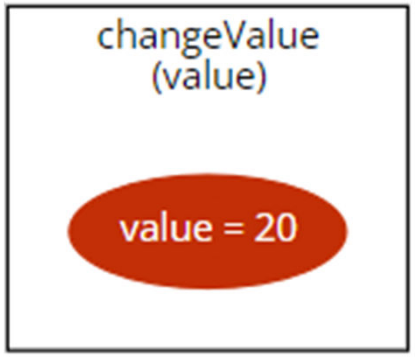

```
function add(a, b) {  // 'a' และ 'b' คือพารามิเตอร์
    return a + b;
}
```

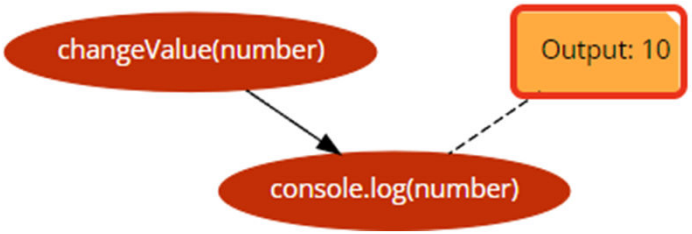
# การส่งค่าผ่านพารามิเตอร์แบบ By Value

**By Value**

- ค่าจริงของตัวแปรจะถูกคัดลอกไปยังพารามิเตอร์
- การเปลี่ยนแปลงในพารามิเตอร์ไม่ได้ส่งผลต่อตัวแปรต้นฉบับ

```
let number = 10;
function changeValue(value) {
    value = 20;
}
changeValue(number);
console.log(number);  // Output: 10
```



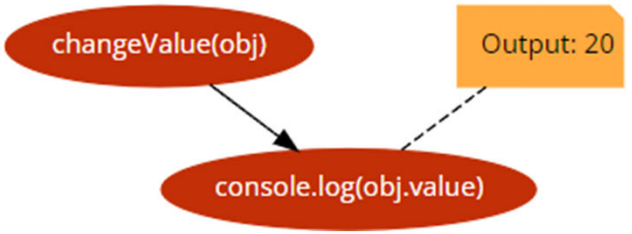



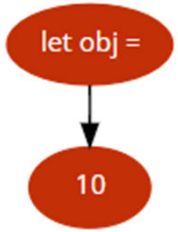
# การส่งค่าผ่านพารามิเตอร์แบบ By Reference

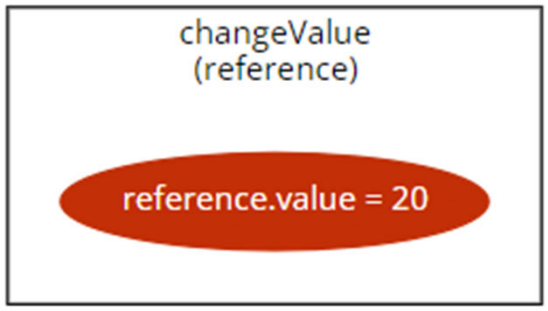
By Reference

- พารามิเตอร์จะได้รับการอ้างอิงไปยังตัวแปร
- การเปลี่ยนแปลงในพารามิเตอร์จะส่งผลต่อตัวแปรต้นฉบับ

```
let obj = { value: 10 };
function changeValue(reference) {
    reference.value = 20;
}
changeValue(obj);
console.log(obj.value);  // Output: 20
```



# การกำหนดค่าเริ่มต้นให้กับพารามิเตอร์

ค่าเริ่มต้นกับพารามิเตอร์

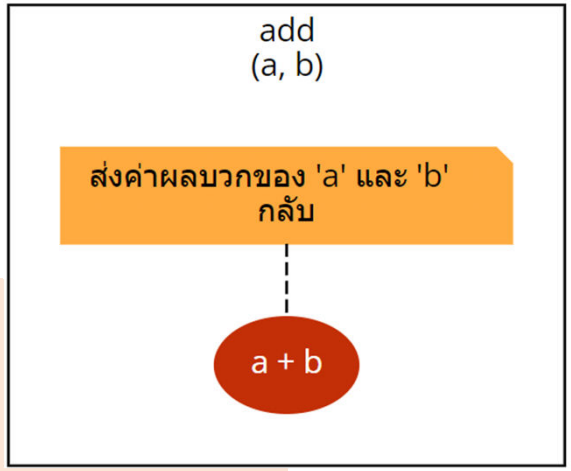
- ถ้าไม่มีการส่งค่าให้กับพารามิเตอร์ **name**
- ค่าเริ่มต้น **"Anonymous"** จะถูกใช้แทน


```
function greet(name = "Anonymous") {
// กำหนดค่าเริ่มต้นให้กับพารามิเตอร์ 'name' เป็น "Anonymous"
    return "Hello, " + name + "!";
}
console.log(greet());  // Output: Hello, Anonymous!
console.log(greet("John"));  // Output: Hello, John!
```

# การส่งค่า

# การส่งค่ากลับจากฟังก์ชัน

- การส่งค่ากลับจากฟังก์ชัน โดยใช้คำสั่ง **return**
- ค่าที่ส่งกลับไว้ใน**ตัวแปร**หรือ**ใช้ต่อ**ในโปรแกรม




```
function add(a, b) {
    return a + b;  // ส่งค่าผลบวกของ 'a' และ 'b' กลับ
}
let sum = add(5, 3);
// เรียกฟังก์ชัน 'add' และเก็บค่าที่ส่งกลับไว้ในตัวแปร 'sum'
console.log(sum);  // Output: 8
```

# ฟังก์ชันที่ส่งค่ากลับและไม่ส่งค่ากลับ

**ฟังก์ชันที่มีการส่งค่ากลับ (Returning Function)**

- ส่งค่ากลับหลังจากที่ประมวลผลเสร็จสิ้น
- ค่านี้สามารถใช้งานต่อไปในโปรแกรม

```
function multiply(a, b) {
    return a * b;
}

let product = multiply(4, 5);
// 'product' ได้รับค่า 20 จากการเรียก 'multiply'
```

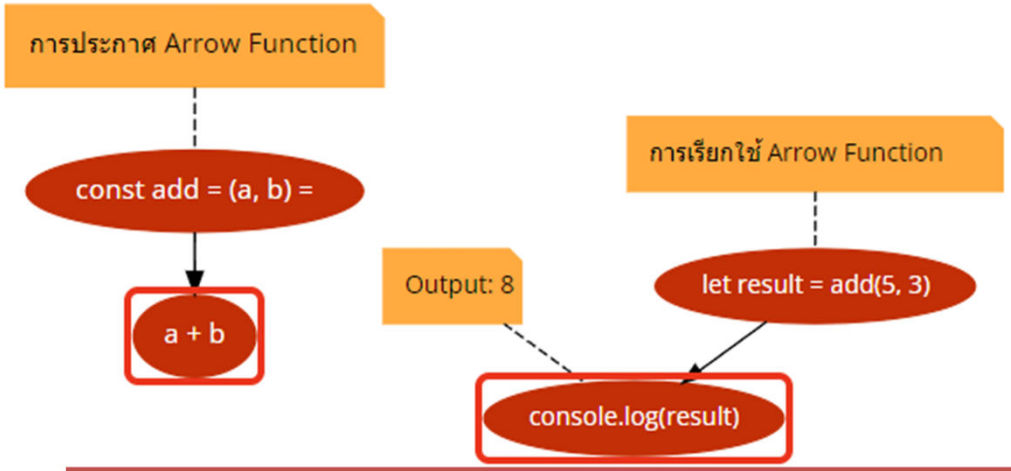
**ฟังก์ชันที่ไม่มีการส่งค่ากลับ (Non-returning Function)**

- ฟังก์ชันนี้จะไม่ส่งค่าใด ๆ กลับหลังจากประมวลผล
- มักใช้เพื่อแสดงผลหรือทำงานที่ไม่ต้องการค่าผลลัพธ์

```
function greet(name) {
    console.log("Hello, " + name + "!");
}

greet("Alice");  // Output: Hello, Alice!
```

# Arrow Function

- รูปแบบการประกาศฟังก์ชัน js ที่มีรูปแบบสั้นและง่ายต่อการเขียน
- โดยใช้สัญลักษณ์ => แทน function
- ไม่ผูกกับ this context ของตัวเอง

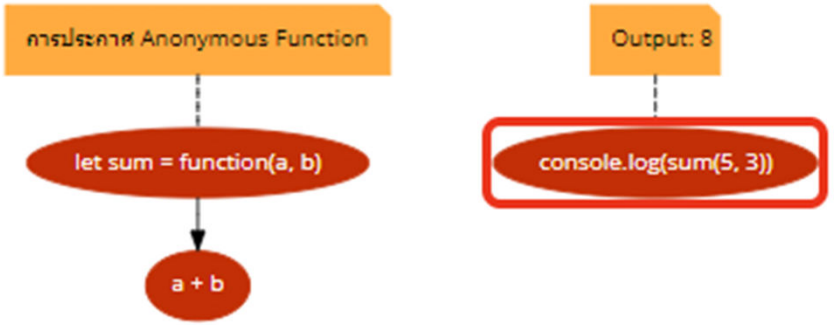



```
// การประกาศ Arrow Function
const add = (a, b) => a + b;

// การเรียกใช้ Arrow Function
let result = add(5, 3);
console.log(result); // Output: 8

// Arrow Function ที่ไม่มีพารามิเตอร์
const greet = () => console.log('Hello, World!');
greet(); // Output: Hello, World!

// Arrow Function ที่มีพารามิเตอร์เดียว ไม่ต้องใส่วงเล็บ
const square = x => x * x;
console.log(square(4)); // Output: 16
```

# Anonymous Function

- **Anonymous Function** เป็นฟังก์ชันที่ไม่มีชื่อ สามารถประกาศและใช้งานได้ทันที

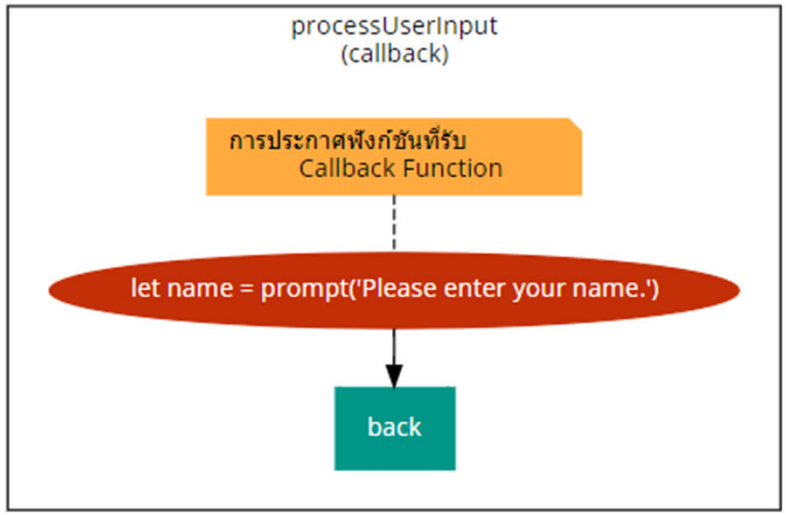
```
// การประกาศ Anonymous Function
let sum = function(a, b) {
  return a + b;
};

console.log(sum(5, 3)); // Output: 8
```



# Callback Function

- **Callback Function** เป็นฟังก์ชันที่ถูกส่งเป็นอาร์กิวเมนต์ให้ฟังก์ชันอื่น และถูกเรียกใช้เมื่อมีการดำเนินการเสร็จสิ้น

```
// การประกาศฟังก์ชันที่รับ Callback Function
function processUserInput(callback) {
  let name = prompt('Please enter your name.');
  callback(name);
}

// การเรียกใช้ฟังก์ชันพร้อม Callback Function
processUserInput(function(name) {
  console.log('Hello ' + name);
});
```



# Refactor code

- การปรับปรุงโค้ดให้มีโครงสร้างที่ชัดเจน
- การแบ่งโค้ดออกเป็นฟังก์ชันย่อย

```
// ฟังก์ชันหลักสำหรับประมวลผลรายการธุรกรรม
function processTransactions(transactions) {
  for (let transaction of transactions) {
    processTransaction(transaction);
  }
}

processTransactions(transactions);
```

```
// ฟังก์ชันย่อยสำหรับประมวลผลธุรกรรม
function processDeposit(amount) {
  console.log(`Deposited: $${amount}`);
}

function processWithdrawal(amount) {
  console.log(`Withdrew: $${amount}`);
}

function processTransaction(transaction) {
  switch (transaction.type) {
    case 'deposit':
      processDeposit(transaction.amount);
      break;
    case 'withdrawal':
      processWithdrawal(transaction.amount);
      break;
}
```

# สรุปท้ายบท

- บล็อกของโค้ดที่ออกแบบมาเพื่อทำงานเฉพาะ ช่วยลดการซ้ำซ้อน
- ประกาศฟังก์ชันด้วยคำหลัก function  ชื่อฟังก์ชันและ parameter
- การเรียกใช้ฟังก์ชันโดยการใช้ชื่อ อาร์กิวเมนต์ตามที่ต้องการ
- ตัวแปรที่ใช้ในการรับค่าเข้าสู่ฟังก์ชันแบบ By Value
- การส่งค่ากลับจากฟังก์ชันทำได้โดยคำหลัก return
- ฟังก์ชันที่มีการส่งค่ากลับสามารถนำผลลัพธ์ไปใช้งาน
- Arrow Function,  anonymous function, callback รวมถึงการ refactor code ในการแบ่งออกเป็นฟังก์ชันย่อย

# ถาม-ตอบ