

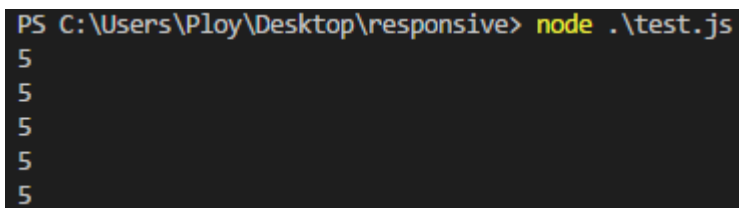
## Homework 4

```
for(var i = 0; i < 5; i++) {

  setTimeout(() => { console.log(i) }, 100)

}
```

1. จงอธิบายว่า console.log(i) ออกมาเป็นค่าอะไร ซึ่งถูกหรือผิด



```
PS C:\Users\Ploy\Desktop\responsive> node .\test.js
5
5
5
5
5
```

out put ผิด

2. จากข้อ 1. หาก ถูก ถูกเพราะอะไร และ หาก ผิด ผิดเพราะอะไร พร้อมแก้ไข Code ให้ถูกต้อง

ตอบ ผิด เพราะ เวลาใช้ var ที่เป็น function scope พอเจอคำสั่ง setTimeout ทำให้ถูกหน่วงเวลาเอาไว้ ลูป for จึงทำงานเสร็จก่อน ค่า i เลยจบที่ 5 และ log 5 ออกมาทุกตัว

```
for(let i = 0; i < 5; i++) {

  setTimeout(() => { console.log(i) }, 100)

}
```

### 3. จะเปลี่ยน function พวกนี้ให้กลายเป็น Arrow function

โดยเขียนลวดรูปให้สั้นที่สุดตาม pattern ของ Arrow function

```
function fn1() {}
```

```
const fn1 = () => {}
```

---

```
function fn2() {
```

```
  console.log('Hello')
```

```
  console.log('Arrow')
```

```
}
```

```
const fn2 = () => {
```

```
  console.log('Hello')
```

```
  console.log('Arrow')
```

```
}
```

---

```
function fn3() {
```

```
  console.log('before return')
```

```
  return 5
```

```
}
```

```
const fn3 = () => {
```

```
  console.log('before return')
```

```
  return 5
```

```
}
```

---

```
function fn4() {
```

```
  return 5
```

```
}
```

---

```
const fn4 = () => 5
```

```
function fn5(a) {
```

```
  return a + 5
```

```
}
```

```
const fn5 = a => a + 5
```

```
function fn6(a, b) {
```

```
  return a + b
```

```
}
```

```
const fn6 = (a,b) => a+b
```

#### 4. จะเปลี่ยน Javascript ต่อไปนี้ให้เป็น ES6

```
const obj1 = { a: 1, b: 2 }
```

```
const obj2 = { a: 3, c: 1, d: 'Deja' }
```

```
const obj3 = Object.assign({}, obj1, obj2);
```

```
const obj1 = { a: 1, b: 2 }
```

```
const obj2 = { a: 3, c: 1, d: 'Deja' }
```

```
const obj3 = { ...obj1, ...obj2 }
```

```
console.log(obj3);
```

```
const arr1 = [1,2,3]
```

```
const arr2 = [5,1]
```

```
const arr3 = arr1.concat(arr2);
```

```
const arr1 = [1,2,3]
```

```
const arr2 = [5,1]
```

```
const arr3 = [...arr1,...arr2]
```

```
console.log(arr3);
```

5. จงใช้ Destructuring เพื่อจะให้ได้สามารถ log ได้แบบนี้

```
console.log(age)
console.log(gender)
console.log(name)
console.log(firstName)
console.log(lastName)
```

```
let person = {
  age: 24,
  gender: 'male',
  name: {
    firstName: 'firstName',
    lastName: 'lastName'
  }
}
```

ตอบ

```
17 // หากใช้ Destructuring จะเหลือแค่นี้
18 let { age, gender, name } = person
19 let { firstName, lastName } = name
20
21 // แต่ในความเป็นจริงแล้ว name เป็นเพียงแค่ทางผ่าน
22 // เราไม่ต้องการมัน เราต้องการแค่ firstName และ lastName
23 // จึงใช้ Destructuring ซ้อนเข้าไปอีกครั้ง
24 // เพียงเท่านี้ตัวแปร name ก็จะไม่เกิดขึ้นมาให้รำคาญใจ
25 let { age, gender, name: { firstName, lastName } } = person
26
```

กำหนด class Animal ให้ดังนี้

6. 

```
class Animal {
  constructor(name) {
    this.isOrganism = true
    this.name = name
  }
}
```

จงสร้าง class Dog ที่สืบทอดคุณสมบัติของ class Animal สามารถทราบชื่อได้, เป็นสิ่งมีชีวิตหรือไม่, เสียงเห่า ดังนี้

```
const chalath = new Dog('Chalath')
console.log(chalath.name) // Chalath
console.log(chalath.isOrganism) // true
console.log(chalath.getBark()) // Hong Hong!! My name is Chalath
```

ตอบ

```
1  class Animal {
2    constructor(name){
3      this.isOrganism = true
4      this.name = name
5    }
6    getBark()
7      {
8        return 'Hong Hong!! My name is ' + this.name
9      }
10 }
11 class Dog extends Animal {
12   constructor(name){
13     super(name)
14   }
15 }
16 const chalath = new Dog('Chalath')
17 console.log(chalath.name)
18 console.log(chalath.isOrganism)
19 console.log(chalath.getBark())
```