# Lab 1 - Nodejs basic

Nodejs เป็น Opensource ที่ Cross-Platform Runtime Environment ใช้ภาษา Javascript ในการทำงานทั้งฝั่ง Client และ ฝั่ง Server ในการทดลองนี้จะอธิบาย nodejs ในส่วนของความสามารถและความแตกต่าง ที่เพิ่มขยายมาจาก ภาษาโปรแกรมแบบเดิม

### **Nodejs installation**

ติดตั้งจาก <https://nodejs.org/en/download/>

### 

### **Javascript block**

ทดลองประกาศตัวแปร และสังเกตค่าต่อไปนี้

|  |
| --- |
| > x=1  1  > x  1  > {let x=2}  undefined  > x  1  > {var x=3}  undefined  > x  3  > {x=4}  4  > with ( {x:5}) {x}  5  > x  4 |

**ข้อสังเกต:** ค่าตัวแปร x ที่อยู่ใน {} และ{} จะเห็นได้ว่า การใช้ let กับ with เป็นการบังคับให้ตัวแปรมีค่าใน local scope {} แต่ var มันจะบังคับให้ตัวแปรอยู่ใน local scope {} ไม่ล่าย

### 

### **Variable type**

ทดลองกำหนดค่าให้กับตัวแปรที่เป็น primitive type ต่าง ๆ ได้แต่ Boolean, Number, String, Special value, Object, Arrays รวมไปถึง ฟังก์ชัน

เช่น a = false, age = 45, name = “Bob”, w = null, x = undefined, y = NaN , object = {}, myarray = [],

ตรวจสอบชนิดตัวแปร: typeof(null), typeof(undefined)

//consol.log(typeof(…))

จงประกาศ Object ชื่อว่า person (ใน 1 คำสั่ง) มีคุณสมบัติ ประกอบด้วย

* name เก็บเป็น Bob
* age เก็บเป็น 45
* birthday เก็บเป็น object ของ วัน เดือน ปี วันที่ 5/11/1900
* max เก็บเป็น function max() รับค่า 2 ค่า และส่งค่าที่มากกว่ากลับมา

**ตอบ**

person = {

name:"Bob",

age:45,

birthday: { day:5, month:11, year:1900},

max: (x,y) => (x>y)?x:y //คล้าย if-else ถ้าใช่ก็ตอบ x ไม่ตอบ y

}

จงเพิ่มคำอธิบายของฟังก์ชัน max.description มีค่าเป็น “Find max” เก็บเป็นคำอธิบายของฟังก์ชันนั้น

**ตอบ:**

person.max.description = "Find max" //แค่เพิ่มคำอธิบายฟังก์ชัน

### **การรับค่าจาก standard input**

เรียกใช้ Standard input จาก process object เมื่อรับข้อมูลแล้วให้ แสดงผลโดยใช้ callback ฟังก์ชัน จากการดักจับ event “data” ใน add listener ดังนี้

|  |
| --- |
| let stdin = process.openStdin()  stdin.addListener("data", function (str) {  console.log(str.toString().trim()) //แสดงค่า  stdin.destroy() //ล้างค่า  }); |

### **การจัดการเกี่ยวกับข้อความ**

ทดลองใช้คำสั่งจัดการเกี่ยวกับข้อความดังต่อไปนี้

console.log(str.length) //วัดความยาวของข้อความ

console.log(str.substring(7, 13) ) //ดึงคำที่อยู่ในช่วงตำแหน่งที่ 7-13

console.log(str.replace("Banana", "Grape")) //แทนที่

console.log(str.toUpperCase() ) //เปลี่ยนเป็นพิมพ์ใหญ่ทั้งหมด

console.log(str.toLowerCase() )//เปลี่ยนเป็นพิมพ์เล็กทั้งหมด

console.log(str.concat(", ", "Grape") ) //ต่อท้ายด้วย “ , ” กับ “ Grape ”

console.log(str.charAt(0)) แสดง char ตำแหน่งในวงเล็บ

console.log(str[0]) // แสดงค่าในตำแหน่งที่กำหนด

console.log(str.indexOf("Kiwi")) //แสดงตำแหน่งเริ่มต้นของคำในวงเล็บ

var str = "Please locate where 'locate' occurs!";

var pos = str.lastIndexOf("locate"); // แสดงเลขตำแหน่งสุดท้ายของคำที่กำหนดในวงเล็บ

console.log( pos + '\n') // print 21

var str = "Please locate where 'locate' occurs!";

var pos = str.search("locate"); //หาคำที่กำหนดในวงเล็บว่าเริ่มต้นอยู่ที่ตำแหน่งไหน

console.log( pos + '\n') // print 7

var str = "Apple, Banana, Kiwi";

var res = str.slice(7, 13); //ดึงคำเฉพาะในช่วงที่กำหนดมาแสดง

console.log( res + '\n') // print Banana

var str = "Apple, Banana, Kiwi";

var res = str.slice(-12, -6);

console.log( res + '\n') // print Banana

var res = str.slice(7);

console.log( res + '\n') // print Banana, Kiwi

เครื่องหมาย + ใช้ในการต่อข้อความ

var n1 = '2';

var n2= '3';

console.log(n1+n2) // 23

ถ้าต้องการให้ตัวเลขบวกกันต้องแปลงเป็น Integer

var n3 = parseInt(n1)+ parseInt(n2);

console.log(n3) // 5

* สามารถใช้ +’1’ แปลงจาก string เป็น number ได้

ตรวจสอบว่าเป็นตัวเลขหรือไม่ จากกรณีต่อไปนี้

isNaN(123) //false

isNaN(-1.23) //false

isNaN(5-2) //false

isNaN(0) //false

isNaN('123') //false

isNaN('Hello') //true

isNaN('2005/12/12') //true

isNaN('') //false

isNaN(true) //false

isNaN(undefined) //true

isNaN('NaN') //true

isNaN(NaN) //true

isNaN(0 / 0) //true

not a number

### **Loop**

การใช้ for แบบดั้งเดิม ท่ามาตรฐาน

let array = [0,1,2,3]

for(let i=0;i<array.length;i++)

console.log(array[i])

การใช้ for in กับ array

for (let key in array)

console.log ( array[key] )

การใช้ for in กับ object

let obj = { 'a':1, 'b':2 , 'c':3}

for (key in obj)

console.log ( key + ': ' + obj[key] )

### 

### **Comparison operator**

let str1 = 'OK';

let str2 = new String('OK');

if (str1 == str2 )

console.log('equal');

else

console.log('Not equal');

ทดลองเปลี่ยน == เป็น === หรือ !== แล้วสังเกตผลการทำงาน

ทดลองเปลี่ยน str1 เป็น 1, ‘1’, “1” สังเกตการทำงาน ผลลัพธ์จากการเปรียบเทียบค่าในแต่ละแบบ

จงบอกข้อแตกต่างระหว่าง == กับ ===

**ตอบ**

== จะทำ type conversion ให้ เช่น 1 = ‘1’ ==”1” ในขณะที่ === จะไม่มีการทำ type conversion และ ต้องเป็นค่าที่ตำแหน่ง memory เดียวกัน ทำให้ ‘OK’ กับ new String(‘OK’) มีค่าไม่เท่ากัน

การเทียบ == ระหว่าง new String(‘OK’) กับ new String(‘OK’) ได้ผลว่าไม่เท่ากัน แม้จะทำ type conversion ให้ ต้องใช้ localCompare

### **Function**

Anonymous functions make great callbacks เป็นฟังก์ชันที่ไม่มีชื่อ ใช้ในการทำ Callback

|  |
| --- |
| setTimeout(function() {  console.log("done");  }, 10000) |

Curried function เป็นฟังก์ชันที่ กำหนดภายในฟังก์ชันอีกที

|  |
| --- |
| function CurriedAdd(x){  return function(y){ return x+y}  };  g = CurriedAdd(2);  g(3) |

Variable number of arguments การกำหนดอาร์กิวเม้นท์แบบไม่ตายตัว

|  |
| --- |
| function sumAll() {  var total=0;  for (var i=0; i< sumAll.arguments.length; i++)  total+=sumAll.arguments[i];  return(total);  }  sumAll(3,5,3,5,3,2,6); |

จงเขียนฟังก์ชันที่กำหนดต่อไปนี้ ในรูปแบบ Curry function

function sum3(x, y, z) {

return x + y + z;

}

console.log(sum3(1, 2, 3) // 6

**ตอบ**

function sum3(x) {

return (y) => {

return (z) => {

return x + y + z;

};

};

}

console.log(sum3(1)(2)(3)) // 6

### **Traditional Function vs. Arrow function**

การเขียน Javascript แบบ ES6 นั้นสามารถเขียนฟังก์ชันในรูปย่อ เรียกว่า arrow function ทำให้โค๊ดสั้นลง ก่อนอื่น ให้มาดูการทำงานของฟังก์ชันปรกติ (ยังไม่ใช้ arrow function) ก่อน

กำหนด function foo มีการทำงานดังนี้

foo = function () { console.log("Foo") }

foo() // เรียกใช้งาน

ผลลัพท์: (หากเรียกใช้ใน terminal (REPL) จะติด undefined มาด้วย)

Foo

undefined // เพราะไม่มีการ return ค่าออกมาจาก console.log

foo = function () { return "Foo" }

foo()

ผลลัพท์: จะแสดงข้อความ Foo ไม่มี undefined เพราะมีการ return ค่าออกมา

**Change to arrow function**

การเขียน function ใน ES6 สามารถย่อ โดยการลบคำว่า function กับ เครื่องหมาย {} ออก และเปลี่ยนเครื่องหมาย = เป็น => แทน ดังนี้

|  |
| --- |
| foo = function () { return "Foo" } เปลี่ยนเป็น foo = () => “Foo” |

ในกรณีที่ฟังก์ชันนั้นมีเพียงคำสั่งเดียว หากไม่ใส่ เครื่องหมาย {} จะถือว่าเป็นการ return ค่าออกไป

เช่น

|  |
| --- |
| foo = () => x=4 หมายถึง มีการ return x    foo = () => { x=4 } หมายถึง ไม่มีการ return x ออกไป |

Arrow function เป็นที่นิยมใช้กันมาก ทั้งในส่วนของ front-end และ back-end web framework การใช้ arrow function ยังมีข้อแตกต่างอื่น ๆ กับ การประกาศฟังก์ชันแบบปรกติ ซึ่งจะกล่าวต่อไปในภายหลัง

จงเปลี่ยน arrow function ต่อไปนี้ ให้เป็น function แบบ non-arrow curry function และ traditional function โดยผลลัพธ์ของการทำงานยังคงเหมือนเดิม

let add = x => y => x+y

add(2)(3) //5

**ตอบ**

function add(x) {

return function (y) {

return x+y

}

}

หรือ

function add(x,y) { return x+y }

จงเปลี่ยน ฟังก์ชันต่อไปนี้ เป็นแบบ arrow function

1. setTimeout(function() { console.log("done"); }, 10000)
2. function CurriedAdd(x){

return function(y){ return x+y}

};

3. function sum3(x) {

return (y) => {

return (z) => {

return x + y + z;

};

};

}

console.log(sum3(1)(2)(3)) // 6

**ตอบ:**

1. setTimeout( () => console.log('done'), 1000)
2. CurriedAdd = (x) => (y) => x+y
3. sum3 = (x) => (y) => (z) => x+y+z

### **Loop กับ callback**

การใช้ map กับ array มี callback function

array.map( (value) => console.log(value ))

การใช้ foreach กับ object มี callback function (ทำให้เราสามารถจัดการแต่ละ element ได้โดยสะดวก)

let obj = { 'a':1, 'b':2 , 'c':3}

Object.keys(obj).forEach( (key) => console.log( key + ': ' + obj[key]) )

จงเขียนโปรแกรมเพื่อรับข้อความเข้ามา 1 ข้อความ และตรวจสอบว่า ข้อความนั้นเป็น Palindrome หรือไม่

* กรณี case sensitive กับ non case sensitive

**ตอบ**

|  |
| --- |
| let stdin = process.openStdin()  stdin.addListener("data", (d) => {  let str = d.toString().trim()  console.log('Palindrome: ' + palindrome(str))  stdin.end()  });  let palindrome = (str) => {  for(let i=0;i<str.length/2;i++)  if( str[i] !== str[str.length-i-1] ) {  return false  }  return true  } |

### 

### **การจัดการ module (library)**

สามารถแบ่ง ฟังก์ชันย่อย ๆ ออกเป็น module แยกไฟล์กันดังนี้

**lib.js**

exports = f1 = () => console.log('f1');

**testLib.js**

require('./lib.js');

f1();

ทดสอบ: node testLib.js

หรือ export ผ่าน module.export

**lib.js**

module.exports.myfoo = () => 'myfoo';’

**testLib.js**

let mo = require('./lib.js');

console.log(mo.myfoo());

จงเขียนโปรแกรม แสดงผลสูตร คูณ, คำนวนเกรดเฉลี่ย และ อื่น ๆ มาอย่างน้อย 3 โปรแกรม