# Exercise (var, let, const)

### **Exercise 1: Block Scope with Let**

**Question:**

運行下面 JavaScript 的程式會否出現錯誤?  
如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

let count = 10;

if (count > 5) {

let count = 5; // New block-scoped variable

console.log("Inside block:", count);

}

console.log("Outside block:", count);

**Answer:** No error occurs.

Success message:   
 *Inside block: 5  
 Outside block: 10*

————————————————————————————————————————

### **Exercise 2: Reassignment of Const**

**Question:**

運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

const maxScore = 100;

if (maxScore < 150) {

maxScore = 120; // Attempting to reassign a constant variable

}

console.log(maxScore);

**Answer:**

Error occurs.

Reason: Cannot assign value to constant variable.

————————————————————————————————————————

### 

### **Exercise 3: Variable Reassignment with Var**

**Question:** 運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

var name = "John";

if (true) {

var name = "Doe"; // Reassigning a variable declared with var

}

console.log(name);

**Answer:** No error occurs.  
 Success message: "Doe"

————————————————————————————————————————

### **Exercise 4: Const in Loops**

const values = [1, 2, 3];

for (const value of values) {

value = value + 1; // Attempting to reassign a constant variable

}

console.log(values);

**Answer:** Error occurs.  
 Cannot assign value to constant variable.

————————————————————————————————————————

### 

### **Exercise 5: Var Hoisting**

**Question:** 運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

console.log(a); // Hoisted declaration

var a = 5;

console.log(a);

**Answer:** No error occurs.  
 Success message: "undefined" and "5"

————————————————————————————————————————

### **Exercise 6: Let and Const in Loops**

**Question:** 運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

const arr = [];

for (let i = 0; i < 3; i++) {

const value = i \* 2;

arr.push(value);

}

console.log(arr);

**Answer:** No error occurs.  
 Success message: [0, 2, 4]

————————————————————————————————————————

### 

### **Exercise 7: Const and Object Mutability**

**Question:** 運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

const obj = { key: "value" };

obj.key = "newValue"; // Mutating the object

console.log(obj);

**Answer:** No error occurs.  
 Success message: { key: 'newValue' }

————————————————————————————————————————

# 

# Exercise (JavaScript Numbers)

### **Exercise 1: Basic Arithmetic Operations**

**Question:** 運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

const a = 10;

const b = 5;

const sum = a + b;

const product = a \* b;

console.log("Sum:", sum);

console.log("Product:", product);

**Answer:** No error occurs.  
 Success message: Sum: 15  
 Product: 50

————————————————————————————————————————

### **Exercise 2: Division by Zero**

**Question:** 運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

const numerator = 20;

const denominator = 0;

const result = numerator / denominator;

console.log("Result:", result);

**Answer:** No error occurs.  
 Success message: Result: Infinity

————————————————————————————————————————

### 

### **Exercise 3: NaN and Type Checking**

**Question:** 運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

const value = "hello" \* 2;

console.log("Value:", value);

console.log("Is NaN:", isNaN(value));

**Answer:** No error occurs.  
 Success message: Value: NaN  
 Is NaN: true

————————————————————————————————————————

### **Exercise 4: Rounding Numbers**

**Question:** 運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

const floatNumber = 3.14159;

const roundedNumber = Math.round(floatNumber);

console.log("Rounded Number:", roundedNumber);

**Answer:** No error occurs.  
 Success message: Rounded Number: 3

————————————————————————————————————————

### **Exercise 5: Random Number Generation**

**Question:** 運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

const randomNum = Math.random() \* 100;

console.log("Random Number:", randomNum);

**Answer:** No error occurs.  
 Success message: Random Number: (a number between 0 and 100)

————————————————————————————————————————

# Exercise (Regular Functions (常規函數) vs. Arrow Functions (箭頭函數))

### **Exercise 1: Basic Function Declaration (常規函數)**

**Question:** 將以下常規函數轉換為箭頭函數並運行，運行結果是什麼?

function greet(name) { // 常規函數

return "Hello, " + name + "!";

}

console.log(greet("Alice"));

**Answer:** No error occurs.  
 Success message: Hello, Alice!

const greet = (name) => "Hello, " + name + "!";

————————————————————————————————————————

### **Exercise 2: Implicit Return in Arrow Function (箭頭函數)**

**Question:** 將以下常規函數轉換為箭頭函數，並使用隱式返回。運行結果是什麼?

function square(x) { // 常規函數

return x \* x;

}

console.log(square(5));

**Answer:**

const square = (x) => x \* x; // 箭頭函數

console.log(square(5)); // output: 25

————————————————————————————————————————

### 

### **Exercise 3: Arrow Function with No Parameters (無參數的箭頭函數)**

**Question:** 將以下常規函數轉換為箭頭函數。運行結果是什麼?

function print() {

var x = n \* n;

console.log(x);

var n = 50;

var z = n \* n;

console.log(z);

}

print();

**Answer:**

const print = () => {

var x = n \* n;

console.log(x);

var n = 50;

var z = n \* n;

console.log(z);

};

// NaN

// 2500

————————————————————————————————————————

### 

### **Exercise 4: Return in Arrow Function (箭頭函數)**

**Question:** 將以下常規函數轉換為箭頭函數，並使用隱式返回。運行結果是什麼?

function addSquares(a, b) {

function square(x) {

return x \* x;

}

return square(a) + square(b);

}

addSquares(1, 2);

**Answer:**

const addSquares = (a, b) => {

function square(x) {

return x \* x;

}

return square(a) + square(b);

};

// Output is 5

————————————————————————————————————————

### **Exercise 5: Return in Arrow Function (箭頭函數)**

**Question:** 將以下常規函數改寫為箭頭函數，並簡化為1行返回。

function addSquares(a, b) {

function square(x) {

return x \* x;

}

return square(a) + square(b);

}

addSquares(1, 2);

**Answer:**

// arrow function

const addSquares = (a, b) => {

function square(x) {

return x \* x;

}

return square(a) + square(b);

}

// step 1

const addSquares = (a, b) => {

const square = (x) => x \* x; // arrow function

return square(a) + square(b);

}

// step 2

const addSquares = (a, b) => {

const square = ((x) => x \* x);

return square(a) + square(b);

}

// step 3

const addSquares = (a, b) => {

return ((x) => x \* x)(a) + ((x) => x \* x)(b); // convert arrow function variant to IIFE

}

// step 4

const addSquares = (a, b) => ((x) => x \* x)(a) + ((x) => x \* x)(b);

————————————————————————————————————————

### **Exercise 6: Return in Regular Function (常規函數)**

**Question:** 將以下箭頭函數改寫為常規函數。

const squareArray = arr => arr.map(num => num \* num);

**Answer:**

function squareArray(arr) {

return arr.map(num => num \* num);

}

————————————————————————————————————————

### **Exercise 7: Return in Regular Function (常規函數)**

**Question:** 將以下箭頭函數改寫為常規函數。

function squareArray(arr) {

return arr.map(num => num \* num);

}

**Answer:**

function squareArray(arr) {

return arr.map(function(num) {

return num \* num;

});

}

————————————————————————————————————————

# Exercise (For in loop)

### **Exercise 1: Basic for...in Loop**

**Question:** 運行結果是什麼?

const person = {

name: "Alice",

age: 30,

city: "New York"

};

for (const key in person) {

console.log(key);

}

**Answer:**

name

age

city

————————————————————————————————————————

### 

### **Exercise 2: Accessing Property Values**

**Question:**運行結果是什麼?

const car = {

make: "Toyota",

model: "Camry",

year: 2020

};

for (const key in car) {

console.log(`${key}: ${car[key]}`);

}

**Answer:**

make: Toyota

model: Camry

year: 2020

————————————————————————————————————————

### **Exercise 3: Counting Properties**

**Question:** 運行結果是什麼?

const student = {

name: "Bob",

grade: "A",

subjects: ["Math", "Science"]

};

let count = 0;

for (const key in student) {

count++;

}

console.log(`Number of properties: ${count}`);

**Answer:**

Number of properties: 3

————————————————————————————————————————

### 

### **Exercise 4: Iterating Over Array Indices**

**Question:**

運行結果是什麼?

const fruits = ["apple", "banana", "cherry"];

for (const index in fruits) {

console.log(index);

}

**Answer:**

0

1

2

————————————————————————————————————————

### 

### **Exercise 5: Filtering Array Values**

**Question:**運行結果是什麼?

const mixedArray = [1, "hello", true, "world", 5];

for (const index in mixedArray) {

if (typeof mixedArray[index] === 'string') {

console.log(mixedArray[index]);

}

}

**Answer:**

hello

world

————————————————————————————————————————

### **Exercise 6: Modifying Array Values**

**Question:**運行結果是什麼?

const messages = ["Hi", "Hello", "Goodbye"];

for (const index in messages) {

messages[index] += '!';

}

console.log(messages);

**Answer:**

[ 'Hi!', 'Hello!', 'Goodbye!' ]

————————————————————————————————————————

### 

### **Exercise 7: Nested Array Elements**

**Question:**運行結果是什麼?

const nestedArray = [

["a", "b"],

["c", "d"],

["e", "f"]

];

for (const index in nestedArray) {

console.log(nestedArray[index]);

console.log(`Element ${index}: ${nestedArray[index]}`);

}

**Answer:**

[ 'a', 'b' ]

Element 0: a,b

[ 'c', 'd' ]

Element 1: c,d

[ 'e', 'f' ]

Element 2: e,f

————————————————————————————————————————

# Exercise (JS Sum of an Array)

### **Exercise 1: Sum of an Array with Negative Numbers**

**Question:**運行結果是什麼?

const mixedNumbers = [10, -5, 3, 7];

let sum = 0;

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

sum += mixedNumbers[i];

}

console.log(sum);

**Answer:**

15

————————————————————————————————————————

### **Exercise 2: Sum of an Array with Mixed Data Types**

**Question:**運行結果是什麼?

const mixedArray = [1, "hello", 2, true, 3];

let sum = 0;

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

if (typeof mixedArray[i] === 'number') {

sum += mixedArray[i];

}

}

console.log(sum);

**Answer:**

6

————————————————————————————————————————

### 

### **Exercise 3: Sum of an Array with Mixed Data Types**

**Question:**運行結果是什麼? 如果有錯誤, 請說明會出現錯誤的原因。

const mixedArray = [1, "hello", 2, true, 3];

let sum = 0;

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

sum += mixedArray[i];

}

console.log(sum);

**Answer:**

1hello2true3

————————————————————————————————————————

### **Exercise 4: Sum of an Empty Array**

**Question:**運行結果是什麼? 如果有錯誤, 請說明會出現錯誤的原因。

const emptyArray = [];

let sum = 0;

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

sum += emptyArray[i];

}

console.log(sum);

**Answer:**

0

————————————————————————————————————————

### 

### **Exercise 5: Sum of an Array with Objects**

**Question:**運行結果是什麼?

const items = [

{ name: "item1", price: 10 },

{ name: "item2", price: 15 },

{ name: "item3", price: 5 }

];

let totalSum = 0;

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

totalSum += items[i].price;

}

console.log(totalSum);

**Answer:**

30

### **Exercise 6: Sum of Mixed Array Elements**

**Question:**運行結果是什麼?

const mixedArray = [1, "hello", 2, true, 3, null, 4.5, "world", 5];

let sum = 0;

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

sum += mixedArray[i];

}

console.log(sum);

**Answer:**

1hello2true3null4.5world5

————————————————————————————————————————

### **Exercise 7: Sum of Mixed Array Elements**

**Question:**運行結果是什麼?

const mixedArray = [1, 10, true, false, "hello", 2, false, 3, undefined, null, 4.5, "world", 5];

let sum = 0;

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

sum += mixedArray[i];

}

console.log(sum);

**Answer:**

12hello2false3undefinednull4.5world5

————————————————————————————————————————

### **Exercise 8: Sum of Mixed Array Elements**

**Question:**運行結果是什麼?

const mixedArray = [undefined, 1, 10, true, false, "hello", 2, false, 3, undefined, null, 4.5, "world", 5];

let sum = 0;

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

sum += mixedArray[i];

}

console.log(sum);

**Answer:**

NaNhello2false3undefinednull4.5world5

————————————————————————————————————————

### **Exercise 9: Sum of Mixed Array Elements**

**Question:**運行結果是什麼?

const mixedArray = [null, 1, 10, true, false, "hello", 2, false, 3, undefined, null, 4.5, "world", 5];

let sum = 0;

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

sum += mixedArray[i];

}

console.log(sum);

**Answer:**

12hello2false3undefinednull4.5world5

————————————————————————————————————————

### **Exercise 10: Sum of Mixed Array Elements**

**Question:**運行結果是什麼?

const mixedArray = [-20, null, 1, 10, true, false, "hello", NaN, 2, false, 3, undefined, null, 4.5, "world", 5];

let sum = 0;

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

sum += mixedArray[i];

}

console.log(sum);

**Answer:**

-8helloNaN2false3undefinednull4.5world5

————————————————————————————————————————

### **Exercise 11: Sum of Mixed Array Elements**

**Question:**運行結果是什麼?

const mixedValues = [10, '5', -3, '2.5', true, null, undefined, 4, false, 0, 'hello', 7];

let total = 0;

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

if (mixedValues[i]) {

total += mixedValues[i]; // This will coerce and sum only truthy values

}

}

console.log(total);

**Answer:**

105-32.5true4hello7

————————————————————————————————————————

### **Exercise 12: Sum of Mixed Array Elements**

**Question:**運行結果是什麼?

const values = [15, '10', null, 3, undefined, '4', -2, true, NaN, 5, '0', 8];

let totalSum = 0;

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

if (values[i] || values[i] === 0) { // Include 0 as a valid number

totalSum += values[i]; // Use unary plus to coerce to number

} else {

console.log(`Skip: i: ${i}, ${values[i]}`);

}

}

console.log(totalSum);

**Answer:**

Skip: i: 2, null

Skip: i: 4, undefined

Skip: i: 8, NaN

151034-2true508

————————————————————————————————————————

# Exercise (Looping)

### Exercise 1

**Question:** 運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

let sum = 0;

for (let i = 5; i >= 1; i--) {

sum += i;

}

console.log(sum);

**Answer:**15————————————————————————————————————————

### Exercise 2

**Question:** 運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

const grid = [

[1, 2, 3],

[4, 5, 6]

];

let total = 0;

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

for (let j = 0; j < grid[i].length; j++) {

if (grid[i][j] % 2 === 0) {

total += grid[i][j] \* 2;

} else {

total += grid[i][j] - 1;

}

}

}

console.log(total);

**Answer:**30————————————————————————————————————————

### 

### Exercise 3

**Question:** 運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

const numbers = [4, 7, 10, 1, 3, 0, -5];

const transformed = [];

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

if (numbers[i] > 0) {

transformed.push(numbers[i] \* 3);

} else if (numbers[i] < 0) {

transformed.push(numbers[i] + 5);

}

}

let totalSum = 0;

for (let j = 0; j < transformed.length; j++) {

totalSum += transformed[j];

}

console.log(totalSum);

**Answer:**75————————————————————————————————————————

# 

# Exercise (JavaScript Array)

**Exercise 1: Adding Elements to an Array**

**Question:** 運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

const colors = ["Red", "Green", "Blue"];

colors[3] = "Yellow";

colors[5] = "Purple";

let cLen = colors.length;

let result = "";

for (let i = 0; i < cLen; i++) {

result += colors[i] + "<br>";

}

console.log(result);

**Answer:**

Red<br>Green<br>Blue<br>Yellow<br>undefined<br>Purple<br>

————————————————————————————————————————

### **Exercise 4: Sum Lengths with Undefined Values**

**Question:** 運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

const fruits = ["Banana", undefined, "Apple"];

let totalLength = 0;

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

if (fruits[i]) {

totalLength += fruits[i].length;

}

}

console.log("Total Length:", totalLength);

**Answer:**

Total Length: 11

————————————————————————————————————————

### **Exercise 5: Sum Lengths with Undefined Values**

**Question:** 運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

const fruits = ["Banana", undefined, "Apple"];

let totalLength = 0;

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

totalLength += fruits[i].length;

}

console.log("Total Length:", totalLength);

**Answer:**

TypeError: Cannot read properties of undefined (reading 'length')

————————————————————————————————————————

### **Exercise 6: Sum Lengths with Undefined Values**

**Question:** 運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

const fruits = ["Banana", undefined, "Apple"];

let totalLength = 0;

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

totalLength += fruits[i]?.length;

}

console.log("Total Length:", totalLength);

**Answer:**

Total Length: NaN

————————————————————————————————————————

# 

# Exercise (express, MongoDB with restful APIs)

<https://medium.com/@vitaliykorzenkoua/node-js-mongodb-tutorial-11a51e11f4ee>

1. How many APIs? What api? What does each API do?
2. What is the port of MongoDB service?
3. What is the port of the web service?
4. How many packages do you use? What packages do you use?
5. Give the database name, and give the database service name?
6. Give me the url of Mongoose connection.
7. How many models? What models to use? What collection names are in use?

**Answer:**

1. How many APIs? What api? What does each API do in detail?

There are 5 APIs in total:

* **Create a User (POST):** Creates a new user in the database using the details provided in the request body. Returns the newly created user with a 201 status code.
  + app.post('/users');
* **Read Users (GET):** Retrieves all users from the database. Returns an array of user objects.
  + app.get('/users');
* **Read a Single User (GET):** Retrieves a single user based on the provided user ID in the request parameters. Returns the user object if found.
  + app.get('/users/:id');
* **Update User (PATCH):** Updates the details of an existing user identified by the user ID in the request parameters. Returns the updated user object.
  + app.patch('/users/:id');
* **Delete a User (DELETE):** Deletes a user from the database based on the provided user ID in the request parameters. Returns the deleted user object.
  + app.delete('/users/:id');

2. What is the port of MongoDB service?

* **MongoDB Service Port**: Default is 27017 (not explicitly mentioned in the website).

3. What is the port of the web service?

* **Web Service Port**: Configurable, default set to 3000 (as per process.env.PORT || 3000).

4. How many packages do you use? What packages do you use?

* **Total Packages Used**: 3
  + express: Web framework for Node.js.
  + mongoose: ODM library for MongoDB and Node.js.
  + dotenv: Loads environment variables from a .env file.

5. Give the database name, and give the database service name?

**Database Name**: mydatabase (from the connection string).

**Database Service Name**: MongoDB.

6. Give me the url of Mongoose connection.

**Mongoose Connection URL**:

mongodb+srv://<username>:<password>@[cluster0.mongodb.net/mydatabase?retryWrites=true&w=majority](http://cluster0.mongodb.net/mydatabase?retryWrites=true&w=majority)

7. How many models? What models to use? What collection names are in use?

Only 1 model in use. Model Names: 'User', Collection names: 'users'.

————————————————————————————————————————

<https://medium.com/@diego.coder/the-perfect-combination-node-js-mongodb-mongoose-cbd21a2e1a26>

1. How many APIs? What api? What does each API do in detail?
2. What is the port of MongoDB service?
3. What is the port of the web service?
4. How many packages do you use? What packages do you use?
5. Give the database name, and give the database service name?
6. Give me the url of Mongoose connection.
7. How many models? What models to use? What collection names are in use?

**Answer:**

1. How many APIs? What api? What does each API do in detail? What function is in use?

There are 5 APIs in total:

* **Get message API (GET):** Get a string message with Hello world.
  + - **API:** app.get('/');
* **Read authors**: Retrieves all authors from the database and returns them as a JSON response.
  + **API:** app.get('/authors);
* **Create an authors**: Creates a new author in the database using the provided data in the request body. It validates the input and, upon success, returns the created author as a JSON response.
  + **API:** app.post('/authors);
* **Read books**: Retrieves all books from the database and returns them as a JSON response.
  + **API:** app.get('/books');
* **Create a book**: Creates a new book in the database using the details provided in the request body. It performs validation and, upon successful creation, returns the newly created book as a JSON response.
  + **API:** app.post('/books');

2. What is the port of MongoDB service?

* **MongoDB Service Port**: Default is 27017.

3. What is the port of the web service?

* **Web Service Port**: Set to 3000.

4. How many packages do you use? What packages do you use?

* **Total Packages Used:** 2
  + express: Web framework for Node.js.
  + mongoose: ODM library for MongoDB and Node.js.

5. Give the database name, and give the database service name?

**Database Name**: test(from the connection string).

**Database Service Name**: MongoDB.

6. Give me the url of Mongoose connection.

**Mongoose Connection URL**:

mongodb://localhost:27017/test

**7. How many models? What models to use? What collection names are in use?**2 models in use. Model Names: 'Book','Author'; Collection names: 'books', 'authors'.

————————————————————————————————————————

**app.get('/', ...);**:

* This sets up a route handler for HTTP GET requests to the root URL (/) of the application.

**function (req, res) { ... }**:

* This is a callback function that gets executed whenever a GET request is made to the root URL.
* It takes two parameters:
  + req: This represents the request object, which contains information about the HTTP request (like headers, URL, etc.).
  + res: This represents the response object, which is used to send a response back to the client.

**res.redirect('/user');**:

* This line tells the server to respond to the client with a redirect to a specific URL ('/user'), which can be either a page link or an API endpoint.
* When a user accesses the root URL (/), they are automatically redirected to the /user page.

### **Page Link vs. API Link**

1. **Page Link**:  
   * A page link typically points to a route that serves HTML content to be rendered in a web browser.
   * **Example**: Redirecting to a user profile page.
     + **Code**: res.redirect('/user/profile');
     + **Use Case**: This would send the user to a webpage where they can view or edit their profile.
2. **API Link**:  
   * An API link usually refers to a route that returns data in a format like JSON or XML, often used for programmatic access.
   * **Example**: Redirecting to an API endpoint that retrieves user data.
     + **Code**: res.redirect('/api/user');
     + **Use Case**: This might return user data in JSON format for frontend applications to display.

### **Summary**

* **Page Link:** Directs users to a web page (e.g., /user/profile).
* **API Link:** Directs requests to an endpoint that returns data (e.g., /api/user).

**res.render(...)**:

* This method is used to render a view template and send it back to the client as a response.
* It combines the specified template with data provided in the second argument to produce HTML.
* Usage:
  + **Example (**app.js**):**
  + // app.js
  + const express = require('express');
  + const app = express();
  + app.set('view engine', 'ejs');
  + app.get('/', (req, res) => {
  + res.render('index', { title: 'Home', user: { name: 'Alice' }, message: 'Hello!' });
  + });
  + 'Index': This is the name of the template file to be rendered. It usually corresponds to a file like index.ejs, index.pug
  + .ejs stands for **Embedded JavaScript**. It is a templating engine used in Node.js applications to generate HTML markup with dynamic content.
  + **Example (**index.ejs**):**
  + **// index.ejs**
  + <!DOCTYPE html>
  + <html lang="en">
  + <head>
  + <meta charset="UTF-8">
  + <meta name="viewport" content="width=device-width, initial-scale=1.0">
  + <title><%= title %></title>
  + </head>
  + <body>
  + <h1>Welcome, <%= user.name %>!</h1>
  + <p>Your message: <%= message %></p>
  + </body>
  + </html>
  + **Reminder: You should install the EJS package!**
  + npm install ejs

————————————————————————————————————————

### Exercise (break and continue)

### **Exercise 1: Skip Negative Numbers**

**Question:**

運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

const numbers = [5, -3, 8, -1, 0, 12, -7];

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

if (numbers[i] < 0) {

continue;

}

console.log(numbers[i]);

}

**Answer:**

5

8

0

12

————————————————————————————————————————

### 

### **Exercise 2: Fibonacci Sequence with Break**

**Question:**

運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

let a = 0, b = 1;

while (true) {

let next = a + b;

if (next > 10) {

break;

}

console.log(next);

a = b;

b = next;

}

**Answer:**

1

2

3

5

8

————————————————————————————————————————

### 

### **Exercise 3: Count Down and Skip**

**Question:**

運行下面 JavaScript 的程式會否出現錯誤?  
 如果有錯誤, 請說明會出現錯誤的原因。如果沒有錯誤, 請寫下成功信息。

let count = 10;

while (count > 0) {

if (count === 5) {

count--;

continue; // Skip printing when count is 5

}

console.log(count);

if (count === 1) {

break; // Stop the loop when count is 1

}

count--;

}

**Answer:**

10

9

8

7

6

4

3

2

————————————————————————————————————————