Phần 3: Thao tác với number, string, array, object và class trong Javascript

1. Bài tập

1. Áp dụng callback, viết một function giải quyết 3 bài toán nhỏ dưới đây.
    - Hãy tìm các số tự nhiên bé hơn 10 và là số lẻ.
    - Hãy tìm các số tự nhiên bé hơn 20 và là số chẵn.
    - Hãy tìm các số tự nhiên bé hơn 30 và là số nếu chia 3 thì dư 2.

    Gợi ý cách giải thông thường (không phải dùng callback)

```
const findNumber1 = () => {
  const result = []
  for (let i = 0; i < 10; i++) {
    if (i % 2 === 1) {
      result.push(i)
    }
  }
  return result
}
const findNumber2 = () => {
  const result = []
  for (let i = 0; i < 20; i++) {
    if (i % 2 === 0) {
      result.push(i)
    }
  }
  return result
}
const findNumber3 = () => {
  const result = []
  for (let i = 0; i < 30; i++) {
    if (i % 3 === 2) {
      result.push(i)
    }
  }
  return result
}
findNumber1()
findNumber2()
findNumber3()
```

2. Tương tự bài 1 nhưng áp dụng kỹ thuật currying
3. Viết hàm tìm giá trị lớn nhất trong mảng 1 chiều các số thực
4. Viết hàm kiểm tra trong mảng các số nguyên có tồn tại giá trị chẵn nhỏ hơn 2004 hay không
5. Viết hàm sắp xếp mảng 1 chiều các số thực tăng dần
6. Viết hàm tính tổng các phần tử trong mảng
7. Viết hàm nhận vào 2 mảng a, b. Return về 1 mảng mới chứa các giá trị chỉ xuất hiện 1 trong 2 mảng.
8. Viết hàm nhận vào 1 string và trả về một Capitalize string. Ví dụ: 'du thanh duoc' -> 'Du Thanh Duoc'
9. Viết hàm nhận vào 1 string và trả về một reverse string. Ví dụ: 'duoc' -> 'coud'
10. Viết hàm kiểm tra nhận vào 2 chuỗi và kiểm tra có phải là reverse string của nhau hay không

11. Viết hàm tìm chữ lặp lại nhiều nhất trong đoạn string.
12. Viết hàm thay thế những chữ không muốn trong 1 đoạn. Ví dụ `replace('du thanh duoc', ['duoc', 'thanh'], 'hello')` thì return về `du hello hello`
13. Viết hàm xử lý

```
// input
const input = [
  { name: 'A', age: 1 },
  { name: 'B', age: 2 },
  { name: 'C', age: 3 }
]
// output
const output = { name: ['A', 'B', 'C'], age: [1, 2, 3] }
```

14. Viết hàm tính tổng tiền lương, nếu object rỗng thì tổng là 0

```
const input = {
  An: 100,
  My: 200,
  Nga: 300,
  Huy: 150
}
const output = 750
```

15. Viết hàm trả về 1 mảng sắp xếp danh sách tăng dần theo chỉ số như name, price. Ví dụ `handle(input, 'name')`

```
const input = [
  {
    name: 'Tivi',
    price: 500
  },
  {
    name: 'Dien thoai',
    price: 100
  },
  {
    name: 'Quan ao',
    price: 80
  }
]
```

16. Viết lại bài trên mà không mutate input

17. Viết hàm trả về 1 mảng mới chứa những bất động sản có giá trên 100

```
const input = [
  {
    name: 'Khu 1',
    price: 500
  },
  {
    name: 'Khu 2',
    price: 100
  },
  {
    name: 'Khu 3',
    price: 80
  }
]
```

2. Đáp án

1. Áp dụng callback, viết một function giải quyết 3 bài toán nhỏ dưới đây.
   - Hãy tìm các số tự nhiên bé hơn 10 và là số lẻ.
   - Hãy tìm các số tự nhiên bé hơn 20 và là số chẵn.
   - Hãy tìm các số tự nhiên bé hơn 30 và là số nếu chia 3 thì dư 2.

```
const findNumber = (num, func) => {
  const result = []
  for (let i = 0; i < num; i++) {
    if (func(i)) {
      result.push(i)
    }
  }
  return result
}
findNumber(10, (number) => number % 2 === 1)
findNumber(20, (number) => number % 2 === 0)
findNumber(30, (number) => number % 3 === 2)
```

2. Tương tự bài 1 nhưng áp dụng kỹ thuật currying

```
const findNumber = (num) => (func) => {
  const result = []
  for (let i = 0; i < num; i++) {
    if (func(i)) {
      result.push(i)
    }
  }
  return result
}
findNumber(10)((number) => number % 2 === 1)
findNumber(20)((number) => number % 2 === 0)
findNumber(30)((number) => number % 3 === 2)
```

3. Viết hàm tìm giá trị lớn nhất trong mảng 1 chiều các số thực

```
const findMax = (array) => {
  let max = array[0]
  for (let i = 1; i < array.length; ++i) {
    if (array[i] > max) {
      max = array[i]
    }
  }
  return max
}

findMax([0, 30, 12, 22, 40, 31])
```

4. Viết hàm kiểm tra trong mảng các số nguyên có tồn tại giá trị chẵn nhỏ hơn 2004 hay không

```
const findMax = (array) => {
  for (let i = 0; i < array.length; ++i) {
    if (array[i] % 2 === 0 && array[i] < 2004) {
      return true
    }
  }
  return false
}

findMax([10000, 2025, 2005, 4000])
```

5. Viết hàm sắp xếp mảng 1 chiều các số thực tăng dần

```
const sortArray = (array) => {
  return array.sort((a, b) => a - b)
}
```

6. Viết hàm tính tổng các phần tử trong mảng

```
const sum = (array) => {
  let result = 0
  array.forEach((item) => {
    result += item
  })
  return result
}
```

7. Viết hàm nhận vào 2 mảng a, b. Return về 1 mảng mới chứa các giá trị chỉ xuất hiện 1 trong 2 mảng.

```
const handle = (array1, array2) => {
  const result = []
  array1.forEach((item) => {
    if (!array2.includes(item)) {
      result.push(item)
    }
  })
  array2.forEach((item) => {
    if (!array1.includes(item)) {
      result.push(item)
    }
  })
  return result
}
```

8. Viết hàm nhận vào 1 string và trả về một Capitalize string. Ví dụ: 'du thanh duoc' -> 'Du Thanh Duoc'

```
const capitalize = (value) =>
  value
    .split(' ')
    .map((item) => item.charAt(0).toUpperCase() + item.slice(1))
    .join(' ')
```

9. Viết hàm nhận vào 1 string và trả về một reverse string. Ví dụ: 'duoc' -> 'coud'

```
const reverse1 = (str) => {
  if (str.length === 0) {
    return str
  }
  return str.charAt(str.length - 1) + reverse1(str.slice(0, str.length - 1))
}
const reverse2 = (str) => [...str].reverse().join('')
const reverse3 = (str) => {
  let result = ''
  for (let i = 0; i < str.length; i++) {
    result += str.charAt(str.length - i - 1)
  }
  return result
}
```

10. Viết hàm kiểm tra nhận vào 2 chuỗi và kiểm tra có phải là reverse string của nhau hay không

```
const compare = (str1, str2) => {
  const length1 = str1.length
  const length2 = str2.length
  if (length1 === length2) {
    for (let i = 0; i < length1; i++) {
      if (str1[i] !== str2[length1 - i - 1]) {
        return false
      }
    }
    return true
  }
  return false
}
const compare2 = (str1, str2) => {
  if (str1.length === str2.length) {
    return [...str1].every(
      (char, index) => char === str2.charAt(str2.length - index - 1)
    )
  }
  return false
}
```

11. Viết hàm tìm chữ lặp lại nhiều nhất trong đoạn string.

```
const handle = (str) => {
  const obj = {}
  let result = ''
  str.split(' ').forEach((item, index) => {
    if (index === 0) {
      result = item
    }
    if (!obj[item]) {
      obj[item] = 1
    } else {
      obj[item] += 1
    }
    if (obj[item] > obj[result]) {
      result = item
    }
  })
  return result
}
```

12. Viết hàm thay thế những chữ không muốn trong 1 đoạn. Ví dụ

`replace('du thanh duoc', ['duoc', 'thanh'], 'hello')` thì return về `du hello hello`

```
const replace = (string, array, target) => {
  const tempArray = array.map(item => item.toLowerCase())
  const stringToArray = string.split(' ')
  stringToArray.map((item, index) => {
    if (tempArray.includes(item.toLowerCase())) {
      stringToArray[index] = target
    }
  })
  return stringToArray.join(' ')
```

Áp dụng Regex

```
const replace = (string, array, target) => {
  return string.replace(new RegExp(array.join('|'), 'gi'), target)
}
```

13. Viết hàm xử lý

```
// input
const input = [
  { name: 'A', age: 1 },
  { name: 'B', age: 2 },
  { name: 'C', age: 3 }
]
// output
const output = { name: ['A', 'B', 'C'], age: [1, 2, 3] }


const handle = (input) => {
  const result = { name: [], age: [] }
  input.forEach((item) => {
    result.name.push(item.name)
    result.age.push(item.age)
  })
  return result
}
```

14. Viết hàm tính tổng tiền lương, nếu object rỗng thì tổng là 0

```
const input = {
  An: 100,
  My: 200,
  Nga: 300,
  Huy: 150
}
const output = 750


const handle = (input) => {
  let result = 0
  for (const key in input) {
    result += input[key]
  }
  return result
}
```

15. Viết hàm trả về 1 mảng sắp xếp danh sách tăng dần theo chỉ số như name, price. Ví dụ `handle(input, 'name')`

```
const input = [
  {
    name: 'Tivi',
    price: 500
  },
  {
    name: 'Dien thoai',
    price: 100
  },
  {
    name: 'Quan ao',
    price: 80
  }
]


const handle = (input, key) => {
  return input.sort((a, b) => {
    if (isNaN(a[key])) {
      if (a[key] > b[key]) {
        return 1
      }
      if (a[key] < b[key]) {
        return -1
      }
      return 0
    }
    return a[key] - b[key]
  })
}
```

16. Viết lại bài trên mà không mutate input

```
const handle = (input, key) => {
  return [...input].sort((a, b) => {
    if (isNaN(a[key])) {
      if (a[key] > b[key]) {
        return 1
      }
      if (a[key] < b[key]) {
        return -1
      }
      return 0
    }
    return a[key] - b[key]
  })
}
```

17. Viết hàm trả về 1 mảng mới chứa những bất động sản có giá trên 100

```
const input = [
  {
    name: 'Khu 1',
    price: 500
  },
  {
    name: 'Khu 2',
    price: 100
  },
  {
    name: 'Khu 3',
    price: 80
  }
]


const handle = (input) => {
  return input.filter((item) => {
    return item.price > 100
  })
}
```