Array methods

- Arrays inherit properties from `Array.prototype`, which defines a rich set of array manipulation methods
  1. forEach() & shift()& unshift()
  2. map() & concat()
  3. every() & indexOf()
  4. some() & include()
  5. slice() & reverse()
  6. splice() (insert & replace & delete)
  7. fill() & find()
  8. push() & pop() & filter()
  9. sort() & findIndex()
  10. join() & reduce()

รายละเอียดการส่งงาน

1. Method Syntax (ทำเฉพาะส่วนที่พารามิเตอร์เป็น primitive types, array, object หรือ arrow function)
2. คำอธิบายความสามารถของ method กระชับและชัดเจน
3. วิธีการใช้ method นั้น ระบุรายการพารามิเตอร์ อธิบายความหมายและชนิดข้อมูล และ output ที่ได้จาก method)
4. ตัวอย่างการใช้งาน ของแต่ละ syntax ให้มีหลาย ๆ กรณีศึกษาที่แตกต่างกัน อย่างน้อย 3 ตัวอย่าง

reduce()

syntax

Arrow function

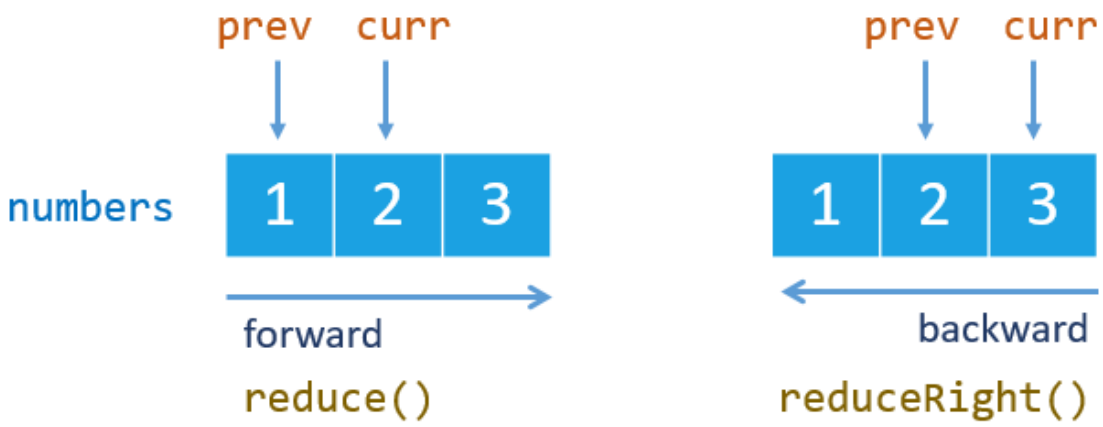
```
reduce((previousValue, currentValue) => { ... } )
reduce((previousValue, currentValue, currentIndex) => { ... } )
reduce((previousValue, currentValue, currentIndex, array) => { ... } )
reduce((previousValue, currentValue, currentIndex, array) => { ... }, initialValue)
```

Callback function

```
reduce(callbackFn)
reduce(callbackFn, initialValue)
```

Inline callback function

```
reduce(function callbackFn(previousValue, currentValue) { ... })
reduce(function callbackFn(previousValue, currentValue, currentIndex) { ... })
reduce(function callbackFn(previousValue, currentValue, currentIndex, array){ ... })
reduce(function callbackFn(previousValue, currentValue, currentIndex, array) { ... }, initialValue)
```



Ability

- มีการสร้างผลลัพธ์ ขึ้นมาใหม่ ไม่ส่งผลต่อ array ตั้งต้น
- มีการเรียกใช้ callback function ในแต่ละ element ที่อยู่ภายใน array และเก็บค่าผลลัพธ์ เพื่อไปดำเนินการใช้งานใน element ถัดไปได้
- ไม่มีการเปลี่ยนค่า เก่า ของ array
- ใช้หาผลลัพธ์ จาก array มาวิเคราะห์ข้อมูลต่างๆ ได้

reduce เป็น method ของ array ซึ่งจะ รับค่าเป็น function หรือ call back function โดยภายใน function และ 2 คือ ค่าตั้งต้นหรือ currentvalue แรก

ภายใน callback จะรับ parameter หลัก ๆ จะ เป็น 2 ค่า แต่มีตัวช่วยเพิ่ม มาอีก 2 ค่า โดยจะ loop โดย จำนวนใน array เป็น length - 1 (ถ้า ไม่ใส่ currentvalue (parameter ที่2))

1. prev คือ เริ่มต้น จะ ส่งผ่านไปเป็น ค่า prev ใน loop ต่อไป
2. current คือ ค่า ปัจจุบัน ที่ loop นั้น กำลังชี้อยู่
3. currentIndex คือ จำนวน loop ที่ทำไปแล้ว
4. array คือ array เดิมที่ใช้

แล้ว return ค่า prev สุดท้ายที่ได้รับมา

การใช้งาน ส่วนมากใช้ในการ **run operator** เพื่อหาค่าใดค่าหนึ่ง ภายใน array เป็นต้น ที่อาศัยการเช็คเป็นลำดับขั้นนั่นเอง

```
const example = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

let i = 0;
```

การบวกเลข ทำโดยจะ รันค่าหาผลลัพธ์ไปทีละ ค่าเป็น current ในแต่ละ callback จะเก็บเป็น prev ในก่อนหน้า โดย prev แรก จะใช้เป็น index ตัวที่ 0 ของ array แต่ถ้า เรา มี paremeter ที่ 2 จะใช้ ตัวนั้น เป็น prev แรก

1. กรณี prev เริ่ม ที่ index แรก คือ 1 loop 9 รอบ เพราะ ไม่นับ index แรก

```
const testReduce = example.reduce((prev, current, currentIndex, array) => {
  console.log(prev, current, currentIndex);
  return prev + current;
});
```

```
[Running] node "d:\IT3\INT201\tempCodeRunnerFile.js"
1 2 1
3 3 2
6 4 3
10 5 4
15 6 5
21 7 6
28 8 7
36 9 8
45 10 9

[Done] exited with code=0 in 0.113 seconds
```

2. กรณี prev เริ่ม ที่ ค่า 10 แล้ว loop 10 รอบ ตามจำนวน length

```
const testReduce2 = example.reduce((prev, current, currentIndex, array) => {
  console.log(prev, current, currentIndex);
  return prev + current;
}, 10);
```

```
[Running] node "d:\IT3\INT201\tempCodeRunnerFile.js"
10 1 0
11 2 1
13 3 2
16 4 3
20 5 4
25 6 5
31 7 6
38 8 7
46 9 8
55 10 9

[Done] exited with code=0 in 0.098 seconds
```

3. ประยุกต์ หารผลรวม ค่า mod 2 ของ ทุก value ภายใน array

```
const testArray = [1];

const mathFunction = (prev, current) => (prev += current % 2);

const testReduce3 = example.reduce(mathFunction, 0);

console.log({ testReduce });
console.log({ testReduce2 });
console.log({ testReduce3 });
```

```
[Running] node "d:\IT3\INT201\tempCodeRunnerFile.js"
1 2 1
3 3 2
6 4 3
10 5 4
15 6 5
21 7 6
28 8 7
36 9 8
45 10 9
10 1 0
11 2 1
13 3 2
16 4 3
20 5 4
25 6 5
31 7 6
38 8 7
46 9 8
55 10 9
{ testReduce: 55 }
{ testReduce2: 65 }
{ testReduce3: 5 }

[Done] exited with code=0 in 0.103 seconds
```

join()

syntax

join()

join(separator)

Ability

- ใช้แสดง string จาก ค่าภายใน array ทุก element มารวมกันเป็น string เดียว โดยสามารถ ระบุว่าจะใช้ string อะไรคั่นระหว่าง element ได้
- มีการสร้างผลลัพธ์ ขึ้นมาใหม่ ไม่ส่งผลต่อ array ตั้งต้น
- ไม่มีการเปลี่ยนค่า เก่า ของ array

join() ใช้สำหรับ การนำค่าแต่ละค่าในตัวแปร array มารวมกันเป็นข้อความ และส่งค่ากลับเป็นข้อความ ที่มีตัวคั่นค่าตัวแปรแต่ละค่าตามที่กำหนด หากไม่ได้กำหนดจะเป็นเครื่องหมาย (,) ให้โดยอัตโนมัติและseparator เป็นตัวคั่นต่าง ๆ เช่น , - _ / \

```
let names = ['A', 'B', 'C', 'D']; //ตัวแปรใน array
let nums = ['1', '2', '3', '4']; //ตัวแปรใน array
```

1. ไม่ได้กำหนดตัวคั่นตัวแปร

ผลคือการนำตัวแปรใน array มาเรียงต่อกันและคั่นด้วย (,) โดยจะเป็น A,B,C,D เนื่องจากถ้าไม่ได้กำหนดตัวคั่นไว้จะเป็น (,) โดยอัตโนมัติ

```
console.log(names.join());
```

```
[Running] node "C:\Users\User\AppData\Local\Temp\tempCodeRunnerFile.javascript"
A,B,C,D
```

2. กำหนดค่าตัวคั่นกลางไว้เป็น (_)

โดยผลลัพธ์ก็จะเป็น 1_2_3_4 ตามที่เรากำหนดไว้ใน num.join('_')

```
console.log(nums.join('_'));
```

```
[Running] node "C:\Users\User\AppData\Local\Temp\tempCodeRunnerFile.javascript"
1_2_3_4
```

3. ใช้ .filter เพื่อกรอกเอาค่า x ที่มากกว่าเท่ากับ 2 มา จากนั้นใช้ Function join เพื่อคั่นกลางระหว่างตัวแปรที่ปริ้นออกมา

```
console.log(
  nums
    .filter((x) => {
      return parseInt(x) >= 2;
    })
    .join('||')
);
```

```
[Running] node "C:\Users\User\AppData\Local\Temp\tempCodeRunnerFile.javascript"
2||3||4
```

git link : https://github.com/study-in-sit/s2_group10