3주차 (4/20)

▼ 목차

1번 실습

Hint

```
Menu 객체 {
  handleEvent 메서드(이벤트) {
    //mousedown -> onMousedown으로 바꿔주는 코드
    ...
  }

  onMousedown() {
    //mousedown event가 발생 할 때
    ...
  }

  onMouseUp() {
    //mouseup event가 발생 할 때
  }
}
```

▼ 실습 정답 코드

```
class Menu {
  handleEvent(event) {
    let method = "on" + event.type[0].toUpperCase() + event.type.slice(1);
    this[method](event);
  }

  onMousedown() {
    elem.innerHTML = "마우스 버튼을 눌렀습니다.";
  }

  onMouseup() {
    elem.innerHTML = "마우스 버튼을 뗐습니다.";
  }
}

let menu = new Menu();
const elem = document.getElementById("elem");

elem.addEventListener("mousedown", menu);
elem.addEventListener("mouseup", menu);
```

2번 실습

map()

`map()` 메서드는 배열 내의 모든 요소 각각에 대하여 주어진 함수를 호출한 결과를 모아 새로운 배열을 반환합니다.

```
const array1 = [1, 4, 9, 16];

// pass a function to map
const map1 = array1.map(x => x * 2);

console.log(map1);
// expected output: Array [2, 8, 18, 32]
```

reduce()

`reduce()` 메서드는 배열의 각 요소에 대해 주어진 **리듀서**(reducer) 함수를 실행하고, 하나의 결과값을 반환합니다.

reduce()에는 누산기가 포함되어 있기 때문에, 배열의 각 요소에 대해 함수를 실행하고 누적된 값을 출력할 때 용이합니다.

가장 기본적인 예제로는 모든 배열의 합을 구하는 경우가 있습니다.

`array .reduce( function(total, currentValue, currentIndex, arr),initialValue )`

```
const arr = [1, 2, 3, 4, 5];
const result = arr.reduce((acc, cur) => { return acc += cur; });
console.log(result);  // 15
```

filter()

`filter()` 메서드는 주어진 함수의 테스트를 통과하는 모든 요소를 모아 새로운 배열로 반환합니다.

```
const words = ['나', '는', '야', '멋쟁이', '개발자'];

const result = words.filter(word => word.length > 2);

console.log(result);
// expected output: Array ["멋쟁이", "개발자"]
```

Hint

```
const arrayFunctions = {
  map(array, func) {
    // 새로운 배열 생성

    //for문을 활용하여 매개변수로 받은 배열의 각 요소에 대해 콜백 함수를 실행
```

```
    {
      //매개변수로 받은 배열의 i 번째 요소에 매개변수로 받은 함수를 적용하여 새로운 배열에 저장
    }

    //새로운 배열 반환

  },

  filter(array, func) {
    //새로운 배열 생성

    //for문을 활용하여 매개변수로 받은 배열의 각 요소에 대해 콜백 함수를 실행

    {
      //만약 매개변수로 받은 함수를 매개변수로 받은 배열의 i 번째 요소에 적용하여 true를 반환하면 새로운 배열에 저장
    }

    //새로운 배열 반환

  },

  reduce(array, func, initialValue) {
    //새로운 배열 생성

    //for문을 활용하여 매개변수로 받은 배열의 각 요소에 대해 콜백 함수를 실행

    {
      //매개변수로 받은 함수안에 새로운 배열과 매개변수로 받은 배열의 i 번째 요소를 전달하여 새로운 배열에 저장

    }

    //새로운 배열 반환

  },
};

export default arrayFunctions;
```

▼ 실습 정답 코드

```
const arrayFunctions = {
  map(array, func) {
    // 새로운 배열 생성
    let newArray = [];
    //for문을 활용하여 매개변수로 받은 배열의 각 요소에 대해 콜백 함수를 실행
    for (let i = 0; i < array.length; ++i) {
      //매개변수로 받은 배열의 i 번째 요소에 매개변수로 받은 함수를 적용하여 새로운 배열에 저장
      newArray.push(func(array[i]));
    }
    //새로운 배열 반환
    return newArray;
  },

  filter(array, func) {
    //새로운 배열 생성
    let newArray = [];
    //for문을 활용하여 매개변수로 받은 배열의 각 요소에 대해 콜백 함수를 실행
    for (let i = 0; i < array.length; ++i) {
      //만약 매개변수로 받은 함수를 매개변수로 받은 배열의 i 번째 요소에 적용하여 true를 반환하면 새로운 배열에 저장
```

```
      if (func(array[i])) newArray.push(array[i]);
    }
    //새로운 배열 반환
    return newArray;
  },

  reduce(array, func, initialValue) {
    //새로운 배열 생성,InitialValue를 초기값으로 사용
    let result = initialValue;
    //for문을 활용하여 매개변수로 받은 배열의 각 요소에 대해 콜백 함수를 실행
    for (let i = 0; i < array.length; ++i) {
      //매개변수로 받은 함수안에 새로운 배열과 매개변수로 받은 배열의 i 번째 요소를 전달하여 새로운 배열에 저장
      result = func(result, array[i]);
    }
    //새로운 배열 반환
    return result;
  },
};

export default arrayFunctions;
```

3번 실습

`...`

Spread 연산자는 연산자의 대상 배열 또는 이터러블(iterable)을 "개별" 요소로 분리해줍니다.

```
// 배열
console.log(...[1, 2, 3]); // -> 1, 2, 3

// 문자열
console.log(...'Daniel');  // D a n i e l

// Map과 Set
console.log(...new Map([['a', '1'], ['b', '2']]));  // [ 'a', '1' ] [ 'b', '2' ]
console.log(...new Set([1, 2, 3]));  // 1 2 3
```

Rest Parameter

Rest Parameter는 위의 Spread 연산자를 사용하여 함수의 parameter로 오는 값들을 배열로 전달받습니다. 사용 방법은 파라미터 앞에 (...)을 붙이면 됩니다.

```
function logger(...rest) {
  console.log(rest); // [ 1, 2, 3, 4, 5 ]
}

logger(1, 2, 3, 4, 5);
```

유의점:

- function foo(param1, param2, ...rest) 처럼 앞에 파라미터는 일반적인 파라미터로 받을 수 있고 그 뒤부터는 Rest 파라미터로 받을 수 있습니다.
- 다만, Rest파라미터는 항상 제일 마지막 파라미터로 있어야 합니다 예를들어 function foo(...rest, param1, param2){~}는 사용할 수 없습니다.

구조 분해 할당

객체나 배열을 변수로 분해할 수 있게 해주는 특별한 문법

```
// 이름과 성을 요소로 가진 배열
let arr = ["Daniel", "Kang"]

// 구조 분해 할당을 이용해
// firstName엔 arr[0]을
// surname엔 arr[1]을 할당하였습니다.
let [firstName, surname] = arr;

alert(firstName); // Bora
alert(surname);  // Lee
```

Hint

```
function mul(m, ...arr) {
  //if문으로 만약 arr가 0이면 m 반환

  //각각 head와 tail에 arr 배열의 첫 요소와 나머지 요소를 할당

  //mul 함수를 반환, 매개변수로 m * head와 tails 넘

}

function multiply(num, ...arr) {
  //새로운 배열 (newArray) 생성

  //for문을 이용, i가 배열의 길이보다 작다면, 위에 생성한 배열에 num * arr의 i 인덱스 요소 곱해서 넣어주기

  //새로운 배열 (newArray) 반환
}
```

▼ 실습 정답 코드

```
function sumArray(sum, ...arr) {
  // arr.length는 arr가 배열이므로, 매개변수 길이를 의미합니다
  if (arr.length === 0) return sum;

  //배열의rest operator는 나머지 변수를 다시 배열로 묶습니다.
  const [head, ...tail] = arr;

  //sumArray의 tail 변수는, 첫번째 원소 head를 제외한 나머지값 들을 다시 배열로 묶는다
  //tail은 하나씩 줄어들게 되며, 길이가 0이 되면 합을 반환한다.
  return sumArray(sum + head, ...tail);
}
sumArray(0, 1, 2, 3, 4, 5);

//지시사항 1
function mul(m, ...arr) {
    if (arr.length === 0) {
        return m;
    }

  const [head, ...tail] = arr;

  return mul(m * head, ...tail);
}

console.log(mul(3, 6, 9, 12, 15));
// expected output: 29160

console.log(mul(11, 13, 17, 19, 23, 29));
// expected output: 30808063

//지시사항 2
function multiply(num, ...arr) {
  let newArray = [];

  for (i = 0; i < arr.length; i++) newArray.push(num * arr[i]);

  return newArray;
}

let arr = multiply(12, 6, 4, 3, 2, 1);
console.log(arr);
// expected output: [ 72, 48, 36, 24, 12 ]
```

Closure 복습

```
```

4번 실습

split()

split() 메서드는 받은 문자열을 인수에 있는 문자열로 나눠서 배열로 만듭니다.

```
let str = '나는 감자다.';
console.log(str.split(' '));
// ['나는' '감자다.']
```

reverse()

reverse() 메서드는 배열의 순서를 반대로 만들어 줍니다.

```
let array = [1,2,3,4,5]
console.log(array.reverse());
// [ 5, 4, 3, 2, 1 ]

let str = '나는 감자다.';
console.log(str.split(' ').reverse());
// ['감자다.' '나는']
```

join()

join() 메서드는 배열의 값 사이에 인자값을 넣은 문자열을 만들어 줍니다.

```
let str = '나는 감자다.';
console.log(str.split(' ').join(' 맛있는 '));
```

toLocaleString()

`toLocaleString()` 메서드는 배열의 요소를 나타내는 문자열을 반환합니다. 요소는 `toLocaleString` 메서드를 사용하여 문자열로 변환되고 이 문자열은 locale 고유 문자열(가령 쉼표 “,”)에 의해 분리됩니다.

링크와 함수의 이름에서 알 수 있다시피 toLocaleString은 말 그대로 특정 자료가 들어왔을 때 설정해 놓은 지역에서 읽는 형태로 바꿔는 함수이다. 우리 주변에 흔한 1,000단위로 끊는 숫자 표기법은 영미 문화권에서 쓰는 방식이다. 그리고 이 함수의 기본값은 미국으로 돼있다. 해서 이런식으로 하면 쉽게 원하는 결과값을 얻을 수 있다.

```
let testNumber = 100000000.0235809;
console.log(testNumber.toLocaleString());
//it returns 100,000,000.024
```

꼭 숫자를 변수에 담은 뒤에 실행하도록 하자. 변수 안에 숫자를 넣으면 생성되는 Number객체 안에 있는 함수를 사용하는 것이기 때문에 그냥 상수 옆에 때려박으면 실행되지 않는다.

만약에 소수점 아래 숫자를 세자리 이상 출력하고 싶으면 이와 같이 하면 된다.

```
var testNumber = 100000000.0235809;
console.log(testNumber.toLocaleString(undefined, {maximumFractionDigits: 5}));
//it returns 100,000,000.02358
```

substr()

substr("시작 위치", "길이") 또는 **substr("시작 위치")**

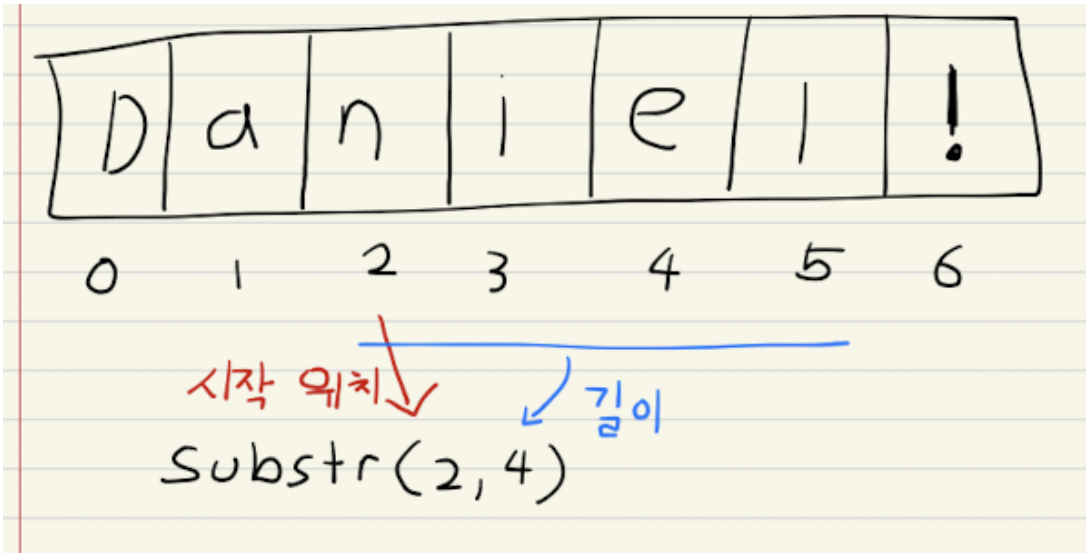
substr() 함수는 시작 위치부터 해당 길이만큼 문자열을 자르는 아주 기본적인 함수이다. "길이" 부분을 생략하면 시작 위치부터 문자열 끝까지 자른다.

substr 함수로 문자열 자르는 방법

```
let str = 'Daniel!';

let result1 = str.substr(2, 4);

console.log(result1)
//expected result: niel
```



▼ 실습 정답 코드

```
/*
1.
getReverse 함수를 return 문 한 줄로 작정하세요.
input: "Happy Thanksgiving!"
output: "!gnivigsknahT yppaH"
*/
function getRevese(s) {
  return s.split("").reverse().join("");
}

console.log(getRevese("Happy Thanksgiving!"));

/*
2.
groupByCommas 함수는 입력값의 3자리 단위씩 컴마로 묶어서 반환합니다.
return 문 한 줄로 작정하세요.
intput: 2443243
output: 2,443,243
*/
function groupByCommas(n) {
  return n.toLocaleString();
}

console.log(groupByCommas(2443243));

/*
3.
getMiddle 함수는 input String s의 중간에 있는 문자열을 반환합니다.
이 때 s의 길이가 짝수라면 가운데 두글자를 반환하고 홀수라면 문자 1개만 반환합니다.
return 문 한 줄로 작정하세요.
intput:eliceacademy
output: ac
intput:christmas
output: c
*/
function getMiddle(s) {
  return s.substr(Math.ceil(s.length / 2) - 1, s.length % 2 === 0 ? 2 : 1);
}

console.log(getMiddle("christmas"));
```

5번 실습

isNaN()

isNaN() - 매개변수가 숫자인지 검사하는 함수입니다.(NaN은 Not a Number입니다.)

문법

```
isNaN( value )
```

- value : 검사할 값을 입력합니다.
- 매개변수가 숫자가 아니면 true, 숫자이면 false를 반환합니다.

Hint

```
isTriangle(a,b,c){
  s = 삼각형의 반둘레 구하기 (모든 변의 길이를 더하고 2로 나눠준다)
  넓이를 구한다 (구하는 공식은 아래에), root = Math.sqrt()
  만약 넓이가 정수가 아니면 false
  만약 0보다 크면 true
}
```

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

Number.isNaN과 isNan은 같나요?

`isNaN` 으로 전달된 값이 숫자가 아닌지 또는 숫자로 변환할 수 없는지 여부를 확인하면 반면, `Number.isNaN`는 인자가 NaN이 아닌지 여부만 확인합니다

```
Number.isNaN({});
// <- false, {} is not NaN
Number.isNaN('ponyfoo')
// <- false, 'ponyfoo' is not NaN
Number.isNaN(NaN)
// <- true, NaN is NaN
Number.isNaN('pony'/'foo')
// <- true, 'pony'/'foo' is NaN, NaN is NaN

isNaN({});
// <- true, {} is not a number
isNaN('ponyfoo')
// <- true, 'ponyfoo' is not a number
isNaN(NaN)
// <- true, NaN is not a number
isNaN('pony'/'foo')
// <- true, 'pony'/'foo' is NaN, NaN is not a number
```

▼ 실습 정답 코드

```
function isTriangle(a, b, c) {
  const semiperimeter = (a + b + c) / 2;
  const area = Math.sqrt(
    semiperimeter *
      (semiperimeter - a) *
      (semiperimeter - b) *
      (semiperimeter - c)
  );

  if (Number.isNaN(area)) return false;
  return area > 0;
}

console.log(isTriangle(2, 4, 6)); //false
```

```
// Function Export
module.exports = { isTriangle };
```

7번 실습

항상 falsy 한 값

- false
- 0
- `''` 또는 `""`
- null
- undefined
- NaN

Falsy를 활용한 코드

위의 값들은 항상 falsy한 값이니, 앞에 `!`를 추가해주면 true로 전환됩니다.

```
function print(person) {
  if (!person) {
    console.log('person이 없네요');
    return;
  }
  console.log(person.name);
}

const person = null;
print(person);
```

Number.isInteger()

인수의 값이 정수인지 아닌지 반환 해줍니다.

```
Number.isInteger(1) //true
Number.isInteger(0) //true
Number.isInteger(-1) //true
Number.isInteger("string") //false
```

▼ 정답 실습 코드

```
//배열의 모든 요소가 square number(정수의 제곱)인지 확인하는 함수를 작성하십시오.
const isSquare = (array) =>
  array.length > 0
    ? array.every((number) => Number.isInteger(Math.sqrt(number)))
    : undefined;

console.log(isSquare([1, 4, 9, 81, 36, 1024]));

module.exports = { isSquare };
```