

SeSAC 도봉 1기 웹 풀스택 과정

문자열 관련 메소드

let str = 'Happy day~!';

- `length` : 문자열의 길이를 반환(공백포함)
`str.length` 13
- `toUpperCase()`&`toLowerCase()` : 문자열 전체를 대문자, 혹은 소문자로 변경
`str.toUpperCase()` HAPPY DAY~! `str.toLowerCase()`
- `indexOf()` : 매개변수로 문자열을 받아서 몇번째 인덱스인지 숫자 반환
`str.indexOf('p')` 2
- `slice(startIdx[,endIdx])` : start 부터 end-1 까지 슬라이싱, 빼옴(?) 매개변수로 음수값도 가능
`str.slice(5, 9)` day
`str.slice(2)` ppy day~!

- `replace(문자열1, 문자열2)` : 문자열1을 문자열2로 변경
`str.replace('p', 'a')` 'Haapy day~! '
- `replaceAll(문자열1, 문자열2)` : 문자열1을 전부찾아서 문자열 2로 바꿔줌
`str.replaceAll('p', 'a')` 'Haaay day~! '
- `repeat(n)` : 문자열에 대해 n번 반복
`str.repeat(3)` Happy day~! Happy day~! Happy day~!
- `trim()` : 문자열의 양끝 공백 없애기
`str.trim()` 'Happy day~!'
- `split()` : 매개변수로 들어온 문자열을 기준으로 str을 쪼개서 배열로 저장

`str.split('')`

▶ (13) ['H', 'a', 'p', 'p', 'y', ' ', 'd', 'a', 'y', '~', '!', ' ', ' ']

`str.split(' ')`

▶ (4) ['Happy', 'day~!', '', '']

배열 관련 메소드

push() : 배열 끝에 추가

```
let days = ['월', '화', '수'];
```

```
days.push('목')
```

```
console.log(days) // ['월', '화', '수', '목']
```

pop() : 배열 끝 요소 제거

```
let days = ['월', '화', '수'];
```

```
days.pop()
```

```
console.log(days) // ['월', '화']
```

shift, unshift 배열 앞에 제거/추가

추가

```
days.unshift('일');  
console.log(days) // ['일', '월', '화', '수'];
```

제거

```
days.shift();  
console.log(days) // ['월', '화', '수'];
```


배열 함수, `includes`

- 해당 배열에 지정한 요소가 있는지 확인하는 메소드
- `[].include(요소);`

```
let numbers = [1, 2, 3, 4, 5, 6];

console.log(numbers.includes(3));
console.log(numbers.includes(7));

const words = ['spray', 'limit', 'elite', 'exuberant', 'destruction', 'present'];

console.log(words.includes('elite'));
console.log(words.includes('pororo'));
```

추가적인 배열관련 method

- `arr.length` : 배열의 길이 반환
- `arr.indexOf()` : 문자열에서의 `indexOf`와 마찬가지로 매개변수에 해당하는 배열의 인덱스를 받아옴. 단, 매개변수로 문자열만 넣을 수 있는 것은 아님!
- `arr.reverse()`; 배열 순서 뒤집어서 반환
- `arr.join()`; join 안의 문자열 기준으로 문자열로 병합

메소드 체이닝

메소드 체이닝

- 각각의 메소드를 연결해서 사용하는 개념!
- 단, 사용한 메소드가 반환(return) 값을 가지고 있는 경우에만 사용이 가능!
- `'hello'.split("")` → `['H', 'e', 'l', 'l', 'o']` 라는 배열이 반환 됨
- 배열에는 `reverse()` 라는 메소드가 존재
- `'hello'.split("").reverse()` 는 `['H', 'e', 'l', 'l', 'o'].reverse()` 와 동일한 것!
- `['H', 'e', 'l', 'l', 'o'].reverse()` → `['o','l','l','e','H']` 와 동일
- `'hello'.split("").reverse().join("")` → `['o','l','l','e','H'].join("")` 과 동일

```
let hello = "Hello~"
helloArr = hello.split("");
console.log(helloArr);

let reverseHello = helloArr.reverse();
console.log(reverseHello);

let helloAgain = reverseHello.join("");
console.log(helloAgain);

console.log(hello.split("").reverse().join(""));
```

```
► (5) ['H', 'e', 'l', 'l', 'o']
► (6) ['H', 'e', 'l', 'l', 'o', '~']
► (6) ['~', 'o', 'l', 'l', 'e', 'H']
```

```
~olleH
```

```
~olleH
```

배열에서의 반복

배열, 기본 for 문 사용하기

```
let numbers = [1, 2, 3, 4, 5, 6];
let fruits = ["사과", "바나나", "수박", "포도", "파인애플"];

for (let i = 0; i < numbers.length; i++) {
  console.log(numbers[i]);
}

for (let i = 0; i < fruits.length; i++) {
  console.log(fruits[i]);
}
```

배열, for of 반복문

```
let numbers = [1, 2, 3, 4, 5, 6];  
let fruits = ["사과", "바나나", "수박", "포도", "파인애플"];  
  
let numbersLength = numbers.length;  
let fruitsLength = fruits.length;  
  
for (let number of numbers) {  
  console.log(number);  
}  
  
for (let fruit of fruits) {  
  console.log(fruit);  
}
```


배열, [].forEach

```
let numbers = [1, 2, 3, 4, 5, 6];
let fruits = ["사과", "바나나", "수박", "포도", "파인애플"];

numbers.forEach(function (number, index, array) {
  console.log(number, index, array);
});

numbers.forEach((number, index, array) => {
  console.log(number, index, array);
});

fruits.forEach(function (fruit, i, arr) {
  console.log(fruit, i, arr);
});

fruits.forEach((fruit, i, arr) => {
  console.log(fruit, i, arr);
});
```

배열의 합

```
let numbers = [1, 2, 3, 4, 5, 6];
var sum1 = 0;
var sum2 = 0;
var sum3 = 0;

for (let i = 0; i < numbers.length; i++) {
  console.log(numbers[i]);
  sum1 = sum1 + numbers[i];
}

for (let num of numbers) {
  sum2 = sum2 + num;
}

numbers.forEach((num) => {
  sum3 = sum3 + num;
});

console.log(sum1, sum2, sum3);
```

```
const arr = [1, 2, 3, 4, 5];
```

배열에서의 기타 메소드

- arr.filter()

```
let arr_filter = arr.filter(function (a) {  
  return a > 3;  
});
```

```
arr_filter = [4, 5]
```

- 배열 내부에서 조건에 부합하는 요소만 찾아서 “배열로” 반환

- arr.find()

```
let val_find = arr.find(function (a) {  
  return a > 3;  
});
```

```
val_find = 4
```

- 배열 내부에서 조건에 부합하는 첫번째 요소를 찾아서 “값”으로 반환

- arr.map()

```
let arr_map = arr.map(function (b) {  
  return b + 3;  
});
```

```
arr_map = [4, 5, 6, 7, 8]
```

- 익명함수에 쓰여진 연산결과를 새로운 배열로 반환

- 위의 메소드들은 매개변수로 익명함수가 들어간다는 공통점이 있음

- 앞서 나온 `forEach((val)=>{})` 메소드도 마찬가지!