삼육대학교 SW융합교육원 김진호

자바스크립트 기초교육

<class>
</body>
<head>
<title>

9. array (배열)

10-1. array

10-2. 배열 메소드

9. array

9-1. array (배열)

9-1-1. array

배열은 여러 개의 값을 순차적으로 나열한 자료구조이다.
배열이 가지고 있는 값을 요소라 부르며 자바스크립트의 모든 값은 배열의 요소가 될 수 있다.

배열 리터럴

```
const arr = ['바나나', '복숭아', '키위'];
```

9. array

9-1. array (배열)

9-1-1. array

배열 생성자 함수

```
const arr2 = new Array();
const arr3 = new Array(10);
const arr4 = new Array(1, 2, 3);
```

- 전달 된 인수가 없을 경우 빈 배열을 생성한다.
- 전달 된 인수가 1개이고 숫자인 경우 length 프로퍼티 값이 인수인 배열을 생성한다.
- 전달 된 인수가 2개 이상이거나 숫자가 아닌 경우 인수를 요소로 갖는 배열을 생성한다.

9. array

9-1. array (배열)

9-1-1. array

Array.of 메소드

```
const arr5 = Array.of(10);
const arr6 = Array.of(1, 2, 3);
const arr7 = Array.of('hello', 'js');
```

- 전달 된 인수를 요소로 갖는 배열을 생성한다.

배열의 요소 접근

```
const arr = ['바나나', '복숭아', '키위'];
console.log(arr[0]);    // 바나나
console.log(arr[1]);    // 복숭아
console.log(arr[2]);    // 키위
```

- 배열의 요소는 자신의 위치를 나타내는 인덱스를 가지며 배열의 요소에 접근할 때 사용된다.
- 요소에 접근 시에는 대괄호 표기법을 사용한다.

9. array

9-1. array (배열)

9-1-1. array

length 프로퍼티, for문

```
const arr = ['바나나', '복숭아', '키위'];
console.log(arr.length);    // 3

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

- 배열은 요소의 개수, 즉 배열의 길이를 나타내는 length 프로퍼티를 갖는다.
- 배열은 인덱스와 length 프로퍼티를 갖기 때문에 for문을 통해 순차적으로 요소에 접근할 수 있다.

9. array

9-1. array (배열)

9-1-1. array

typeof 배열

```
const arr = ['바나나', '복숭아', '키위'];
console.log(typeof arr);    // object
```

■ 배열이라는 별도의 타입은 존재하지 않으며 배열은 객체 타입이다.

9. array

9-1. array (배열)

9-1-2. differences from regular array (일반 배열과의 차이점)

일반적인 의미의 배열은 각 요소가 동일한 데이터 크기를 가지며, 빈틈 없이 연속적으로 이어져 있어 인덱스를 통해 임의의 요소에 한 번에 접근할 수 있는 고속 동작을 장점으로 한다. 하지만 자바스크립트의 배열은 일반적인 배열의 동작을 흉내 낸 특수한 객체로 각각의 메모리 공간이 동일한 크기를 갖지 않아도 되고 연속적으로 이어져 있지 않을 수도 있다.

```
console.log(Object.getOwnPropertyDescriptors([1, 2, 3]));
/*
{
  '0': { value: 1, writable: true, enumerable: true, configurable: true },
  '1': { value: 2, writable: true, enumerable: true, configurable: true },
  '2': { value: 3, writable: true, enumerable: true, configurable: true },
  length: { value: 3, writable: true, enumerable: false, configurable: false }
}
*/
```

- 인덱스를 나타내는 문자열을 프로퍼티 키로 가지며, length 프로퍼티를 갖는 특수한 객체이다.

9. array

9-1. array (배열)

9-1-2. differences from regular array (일반 배열과의 차이점)

```
const arr = [
    '홍길동',
    20,
    true,
    null,
    undefined,
    NaN,
    Infinity,
    [],
    {},
    function(){}
];
```

- 자바스크립트의 모든 값이 객체의 프로퍼티 값이 될 수 있으므로 모든 값이 배열의 요소가 될 수 있다.
- 자바스크립트의 배열은 인덱스로 배열 요소에 접근하는 경우에는 일반적인 배열보다 느리지만 요소를 삽입, 삭제하는 경우에는 일반적인 배열보다 빠르다.

9. array

9-1. array (배열)

9-1-3. length property (length 프로퍼티)

```
console.log([].length);              // 0
console.log([1,2,3,4,5].length);     // 5
```

- length 프로퍼티는 요소의 개수를 나타내는 0 이상의 정수를 값으로 갖는다.

```
const arr = [1,2,3,4,5];
arr.push(6);                 // push : 인자로 전달한 요소 추가
console.log(arr.length);     // 6
arr.pop();                   // pop : 마지막 요소 제거
console.log(arr.length);     // 5
```

- length 프로퍼티 값은 배열에 요소를 추가하거나 삭제하면 자동 갱신된다.

9. array

9-1. array (배열)

9-1-3. length property (length 프로퍼티)

```
const arr = [1,2,3,4,5];
arr.length = 3;
console.log(arr);           // [ 1, 2, 3 ]
arr.length = 10;
console.log(arr);           // [ 1, 2, 3, <7 empty items> ]
console.log(arr.length);    // 10
console.log(Object.getOwnPropertyDescriptors(arr));
/*
{
  '0': { value: 1, writable: true, enumerable: true, configurable: true },
  '1': { value: 2, writable: true, enumerable: true, configurable: true },
  '2': { value: 3, writable: true, enumerable: true, configurable: true },
  length: { value: 10, writable: true, enumerable: false, configurable: false }
}
*/
```

- length 프로퍼티에 임의의 숫자 값을 명시적으로 할당할 수도 있다.
- 현재 length 프로퍼티보다 작은 숫자 값을 할당하면 배열의 길이가 줄어든다.
- 현재 length 프로퍼티보다 큰 숫자 값을 할당하면 length 프로퍼티의 값은 변경되지만 배열의 길이가 늘어나지는 않는다.

9. array

9-1. array (배열)

9-1-3. length property (length 프로퍼티)

```
const sparse = [, 2, , 4];
console.log(sparse);              // [ <1 empty item>, 2, <1 empty item>, 4 ]
console.log(sparse.length);       // 4
console.log(Object.getOwnPropertyDescriptors(sparse));
/*
{
  '1': { value: 2, writable: true, enumerable: true, configurable: true },
  '3': { value: 4, writable: true, enumerable: true, configurable: true },
  length: { value: 4, writable: true, enumerable: false, configurable: false }
}
*/
```

- 자바스크립트는 배열의 요소가 연속적으로 위치하지 않고 일부가 비어있는 희소 배열을 문법적으로 허용한다.
- 일반적인 배열의 length는 배열의 요소의 개수, 즉 배열의 길이와 언제나 일치하지만 희소 배열의 length와 배열 요소의 개수는 일치하지 않는다는 것에 유의한다. 자바스크립트 문법이 허용하긴 하지만 배열은 같은 타입의 요소를 연속적으로 위치 시키는 것이 효율적으로 동작한다.

9. array

9-2. 배열 메소드

```
const arr = [];
// 배열의 생성자 함수는 Array
console.log(arr.constructor === Array);                          // true
// 배열의 프로토타입 객체는 Array.prototype
console.log(Object.getPrototypeOf(arr) === Array.prototype);   // true
// => Array.prototype은 배열을 위한 빌트인 메서드를 제공한다.
```

9. array

9-2. 배열 메소드

9-2-1. 배열 메소드

Array.prototype.indexOf, Array.prototype.lastIndexOf, Array.prototype.includes

- indexOf : 배열에서 요소가 위치한 인덱스를 리턴
- lastIndexOf : 배열의 요소가 위치한 마지막 인덱스를 리턴
- includes : 배열에 해당 요소 포함 여부 리턴

```
const foodList = ['물회', '삼계탕', '냉면', '수박', '물회'];

console.log(`foodList.indexOf('물회') : ${foodList.indexOf('물회')}`);                  // 0
console.log(`foodList.indexOf('물회', 1) : ${foodList.indexOf('물회', 1)}`);            // 4
console.log(`foodList.indexOf('삼겹살') :  ${foodList.indexOf('삼겹살')}`);             // -1

console.log(`foodList.lastIndexOf('물회') : ${foodList.lastIndexOf('물회')}`);          // 4
console.log(`foodList.lastIndexOf('물회', 1) : ${foodList.lastIndexOf('물회', 1)}`);    // 0
console.log(`foodList.lastIndexOf('삼겹살') : ${foodList.lastIndexOf('삼겹살')}`);      // -1

console.log(`foodList.includes('물회') : ${foodList.includes('물회')}`);               // true
console.log(`foodList.includes('삼겹살') : ${foodList.includes('삼겹살')}`);          // false
```

9. array

9-2. 배열 메소드

9-2-1. 배열 메소드

Array.prototype.push, Array.prototype.pop

- push : 배열의 맨 뒤에 요소 추가
- pop : 배열의 맨 뒤에 요소 제거

```
const chineseFood= ['짜장면', '짬뽕', '우동'];

console.log(`push 전 chineseFood : ${chineseFood}`);          // 짜장면,짬뽕,우동

chineseFood.push('탕수육');
chineseFood.push('양장피');
// chineseFood.push('탕수육', '양장피');

console.log(`push 후 arr :  ${chineseFood}`);                 // 짜장면,짬뽕,우동,탕수육,양장피

console.log(`chineseFood.pop() : ${chineseFood.pop()}`);      // 양장피
console.log(`chineseFood.pop() : ${chineseFood.pop()}`);      // 탕수육
console.log(`chineseFood.pop() : ${chineseFood.pop()}`);      // 우동

console.log(`pop 후 chineseFood :  ${chineseFood}`);          // 짜장면,짬뽕
```

9. array

9-2. 배열 메소드

9-2-1. 배열 메소드

Array.prototype.unshift, Array.prototype.shift

- unshift : 배열의 맨 앞에 요소 추가
- shift : 배열의 맨 앞 요소 제거 후 반환

```
const chickenList = ['양념치킨', '후라이드', '파닭'];

console.log(`unshift 전 chickenList : ${chickenList}`);          // 양념치킨,후라이드,파닭

chickenList.unshift('간장치킨');
chickenList.unshift('마늘치킨');
// chickenList.unshift('간장치킨', '마늘치킨');

console.log(`unshift 후 chickenList : ${chickenList}`);          // 마늘치킨,간장치킨,양념치킨,
후라이드,파닭

console.log(`chickenList.shift() : ${chickenList.shift()}`);    // 마늘치킨
console.log(`chickenList.shift() : ${chickenList.shift()}`);    // 간장치킨
console.log(`chickenList.shift() : ${chickenList.shift()}`);    // 양념치킨

console.log(`shift 후 chickenList : ${chickenList}`);           // 후라이드,파닭
```

9. array

9-2. 배열 메소드

9-2-1. 배열 메소드

Array.prototype.concat

- concat : 두 개 이상의 배열을 결합

```
const idol1 = ['아이브', '오마이걸'];
const idol2 = ['트와이스', '에스파'];
const idol3 = ['블랙핑크', '레드벨벳'];

// 배열명.concat(배열명1, 배열명2, ...)
const mix = idol1.concat(idol2);
const mix2 = idol3.concat(idol1, idol2);

console.log(`idol1 기준으로 idol2 배열을 concat : ${mix}`);
// 아이브,오마이걸,트와이스,에스파
console.log(`idol3 기준으로 idol1, idol2 배열을 concat : ${mix2}`);
// 블랙핑크,레드벨벳,아이브,오마이걸,트와이스,에스파
```

9. array

9-2. 배열 메소드

9-2-1. 배열 메소드

Array.prototype.slice, Array.prototype.splice

- slice : 배열의 요소 선택 잘라내기
- splice : 배열의 index 위치의 요소 제거 및 추가

```
const front = ['HTML', 'CSS', 'JavaScript', 'jQuery'];

// slice(시작인덱스, 종료인덱스)
console.log(`front.slice(1, 3) : ${front.slice(1, 3)}`);
// CSS,JavaScript
console.log(`front : ${front}`);
// HTML,CSS,JavaScript,jQuery

// splice(index, 제거수, 추가값1, 추가값2, ...)
console.log(`front.splice(3, 1, "React") : ${front.splice(3, 1, "React")}`);
// jQuery
console.log(`front : ${front}`);
```

9. array

9-2. 배열 메소드

9-2-1. 배열 메소드

Array.prototype.join

- join : 배열을 구분자로 결합하여 문자열로 반환

```
const snackList = ['사탕', '초콜렛', '껌', '과자'];
console.log(`snackList.join() : ${snackList.join()}`);
// 사탕,초콜렛,껌,과자
console.log(`snackList.join('/') : ${snackList.join('/')}`);
// 사탕/초콜렛/껌/과자
```

Array.prototype.reverse

- reverse : 배열의 순서를 뒤집음

```
console.log(`[1, 2, 3, 4, 5].reverse() :
${[1, 2, 3, 4, 5].reverse()}`);     // 5,4,3,2,1
```

9. array

9-2. 배열 메소드

9-2-2. 배열 고차 함수

고차 함수 : 함수를 인수로 전달 받거나 함수를 반환하는 함수

Array.prototype.sort

- 배열을 정렬 기준으로 정렬한다.

```
let numbers = [];

for (let i = 0; i < 10; i++) {
    numbers[i] = Math.floor(Math.random() * 100) + 1;
}

console.log(`정렬 전 numbers : ${numbers}`);
numbers.sort();
console.log(`정렬 후 numbers : ${numbers}`);
```

```
// 숫자 오름차순 정렬
numbers.sort((a, b) => a - b);

// 숫자 내림차순 정렬
numbers.sort((a, b) => b - a);
```

- 오름차순 정렬이 기본이며 정렬 후 정렬 순서를 유지한다.
- 배열은 기본적으로 문자열 정렬이 되므로 일부 올바르지 않게 정렬 되는 모습을 확인할 수 있다.
- 다른 정렬 기준을 사용하려면 매개변수로 compare 함수 전달해야 한다.

9. array

9-2. 배열 메소드

9-2-2. 배열 고차 함수

Array.prototype.forEach

- for를 대체할 수 있는 고차함수이다.

```
/*
배열.forEach(function(item, index, array){
    // 배열 요소 각각에 실행할 기능 작성
});
*/
numbers = [1, 2, 3, 4, 5];
// 각 요소 별로 * 10 한 값을 콘솔에 출력
numbers.forEach(item => console.log(item * 10));
```

9. array

9-2. 배열 메소드

9-2-2. 배열 고차 함수

Array.prototype.map

- 배열 요소 전체를 대상으로 콜백 함수 호출 후 반환 값들로 구성 된 새로운 배열 반환한다.

```
/*
배열.map(function(item, index, array){
    // 배열 요소 각각에 반환할 새로운 값
});
*/
const types = [true, 1, 'text'].map(item => typeof item);
console.log(`types : ${types}`);            // boolean,number,string

const lengths = ['apple', 'banana', 'cat', 'dog', 'egg'].map(item => item.length);
console.log(`lengths : ${lengths}`);        // 5,6,3,3,3
```

9. array

9-2. 배열 메소드

9-2-2. 배열 고차 함수

Array.prototype.filter

- 배열 요소 전체를 대상으로 콜백 함수 호출 후 반환 값이 true인 요소로만 구성 된 새로운 배열 반환한다.

```
const odds = numbers.filter(item => item % 2);
console.log(odds);        // [ 1, 3, 5 ]
```

9. array

9-2. 배열 메소드

9-2-2. 배열 고차 함수

Array.prototype.reduce

- 배열을 순회하며 각 요소에 대하여 이전의 콜백함수 실행 반환값을 전달하여 콜백 함수를 실행하고 그 결과를 반환한다.

```
/*
배열.reduce(function(previousValue, currentValue, currentIndex, array){
    previousValue: 이전 콜백의 반환값
    currentValue : 배열 요소의 값
    currentIndex : 인덱스
    array        : 메소드를 호출한 배열, 즉 this
});
*/

// 합산
const sum = numbers.reduce(function (previousValue, currentValue) {
  return previousValue + currentValue; // 결과는 다음 콜백의 첫번째 인자로 전달된다
});
console.log(`sum : ${sum}`);        // 15

// 최대값
const max = numbers.reduce(function (pre, cur) {
  return pre > cur ? pre : cur;
});
console.log(`max : ${max}`);        // 5
```

9. array

9-2. 배열 메소드

9-2-2. 배열 고차 함수

Array.prototype.some

- 배열 내 일부 요소가 콜백 함수의 테스트를 통과하는지 확인하여 그 결과를 boolean으로 반환한다.

```
// 배열 내 요소 중 10보다 큰 값이 1개 이상 존재하는지 확인
let result = [1, 5, 3, 2, 4].some(item  => item > 10);
console.log(`result : ${result}`); // false
// 배열 내 요소 중 3보다 큰 값이 1개 이상 존재하는지 확인
result = [1, 5, 3, 2, 4].some(item  => item > 3);
console.log(`result : ${result}`);  // true

// 배열 내 요소 중 특정 값이 1개 이상 존재하는지 확인
result = ['apple', 'banana', 'cat', 'dog'].some(item  => item === 'egg');
console.log(`result : ${result}`); // false
result = ['apple', 'banana', 'cat', 'dog'].some(item  => item === 'dog');
console.log(`result : ${result}`);  // true
```

9. array

9-2. 배열 메소드

9-2-2. 배열 고차 함수

Array.prototype.every

- 배열 내 모든 요소가 콜백함수의 테스트를 통과하는지 확인하여 그 결과를 boolean으로 반환한다.

```
result = [1, 5, 3, 2, 4].every(item  => item > 3);
console.log(`result : ${result}`); // false
// 배열 내 요소 중 3보다 큰 값이 1개 이상 존재하는지 확인
result = [1, 5, 3, 2, 4].every(item  => item > 0);
console.log(`result : ${result}`);  // true
```

9. array

9-2. 배열 메소드

9-2-2. 배열 고차 함수

Array.prototype.find, Array.prototype.findIndex

- find : 배열을 순회하며 각 요소에 대하여 인자로 주어진 콜백함수를 실행하여 그 결과가 참인 첫번째 요소를 반환
  - 참인 요소가 존재하지 않는다면 undefined를 반환
- findIndex : 배열을 순회하며 각 요소에 대하여 인자로 주어진 콜백함수를 실행하여 그 결과가 참인 첫번째 요소의 인덱스를 반환
  - 참인 요소가 존재하지 않는다면 -1을 반환

```
const students = [
    { name : '유관순', score : 90 },
    { name : '홍길동', score : 80 },
    { name : '장보고', score : 70 }
];
result = students.find(item => item.name === '유관순');
console.log(result);    // { name: '유관순', score: 90 }
result = students.findIndex(item => item.name === '유관순');
console.log(result);    // 0
result = students.find(item => item.name === '신사임당');
console.log(result);    // undefined
result = students.findIndex(item => item.name === '신사임당');
console.log(result);    // -1
```

9. array

9-2. 배열 메소드

9-2-2. 배열 고차 함수

- find, findIndex는 일치하는 요소를 찾으면 더 이상 탐색하지 않고 하나의 요소, 인덱스만 반환하므로 일치하는 여러 요소를 반환 받지는 못한다.
- 이 때 filter를 이용하면 콜백함수의 실행 결과가 true인 배열 요소의 값만을 추출한 새로운 배열을 반환 받을 수 있다.

```
result = students.find(item => item.score >= 60);
console.log(result);    // { name: '유관순', score: 90 }
result = students.filter(item => item.score >= 60);
console.log(result);    // 배열 반환
```