JavaScript Array Iteration

- description : JavaScript Array Iteration
- author : 오션
- email : shlim@repia.com
- lastupdate : 2021-05-06

The Source of this article

JavaScript Array Iteration
배열 반복 함수(Array iteration methods)는 모든 배열 항목에서 작동합니다.

Array.forEach()

forEach() 메서드는 각 배열 요소에 대해 한 번씩 함수(콜백 함수, a callback function)를 호출합니다.

Example

```
let txt = "";
let numbers = [45, 4, 9, 16, 25];
numbers.forEach(myFunction);
document.getElementById("demo").innerHTML = txt;

function myFunction(value, index, array) {
  txt = txt + value + "\, " // 45, 4, 9, 16, 25,
}
```

상기 예제의 함수는 다음과 같은 3개의 인수(arguments)를 가집니다.

- 항목의 값 (The item value)
- 항목의 인덱스 (The item index)
- 배열 자체 (The array itself)

위의 예제에서는 값 매개변수(value parameter)만을 사용합니다. 다음과 같이 다시 작성할 수 있습니다.

Example

```
let txt = "";
let numbers = [45, 4, 9, 16, 25];
```

```
    numbers.forEach(myFunction);
    document.getElementById("demo").innerHTML = txt;

    function myFunction(value) {
      txt = txt + value + "\, " // 45, 4, 9, 16, 25,
    }
```

Array.map()

map() 메서드는 각 배열 요소에 대해 함수를 실행하여 새 배열을 만듭니다.

map() 메서드는 값이 없는 배열 요소에 대해 함수를 실행하지 않습니다.

map() 메서드는 기존 배열을 변경하지 않습니다.

다음 예제에서는 각 배열 값에 2를 곱합니다.

Example

```
    let numbers1 = [45, 4, 9, 16, 25];
    let numbers2 = numbers1.map(myFunction);

    document.getElementById("demo").innerHTML = numbers2;
    // 90,8,18,32,50
    function myFunction(value, index, array) {
      return value * 2;
    }
    console.log(numbers1);  // [45, 4, 9, 16, 25]
    console.log(numbers2);  // [90, 8, 18, 32, 50]
```

이 함수는 3 개의 인수를 가집니다.

- 항목 값 (The item value)
- 항목 색인 (The item index)
- 배열 자체 (The array itself)

콜백 함수가 value 매개변수만 사용하는 경우, index 및 array 매개변수를 생략할 수 있습니다.

Example

```
    let numbers1 = [45, 4, 9, 16, 25];
    let numbers2 = numbers1.map(myFunction);

    document.getElementById("demo").innerHTML = numbers2;
    // 90,8,18,32,50
```

```
function myFunction(value) {
  return value * 2;
}
console.log(numbers1);  // [45, 4, 9, 16, 25]
console.log(numbers2);  // [90, 8, 18, 32, 50]
```

Array.filter()

filter() 메서드는 테스트를 통과한 배열 요소로 새 배열을 만듭니다.

다음 예제는 값이 18보다 큰 요소에서 새 배열을 만듭니다.

Example

```
let numbers = [45, 4, 9, 16, 25];
let over18 = numbers.filter(myFunction);

document.getElementById("demo").innerHTML = over18;
// 45,25
function myFunction(value, index, array) {
  return value > 18;
}

console.log(numbers); //  [45, 4, 9, 16, 25]
console.log(over18);  //  [45, 25]
```

이 함수는 3 개의 인수를 가집니다.

- 항목 값 (The item value)
- 항목 색인 (The item index)
- 배열 자체 (The array itself)

위의 예제에서, 콜백 함수는 인덱스 및 배열 매개 변수를 사용하지 않으므로 생략할 수 있습니다.

Example

```
let numbers = [45, 4, 9, 16, 25];
let over18 = numbers.filter(myFunction);

document.getElementById("demo").innerHTML = over18;
// 45,25
function myFunction(value) {
  return value > 18;
}

console.log(numbers); //  [45, 4, 9, 16, 25]
```

```
    console.log(over18);  //  [45, 25]
```

Array.reduce()

reduce() 메서드는 각 배열 요소에서 함수를 실행하여, 단일 값을 생성(축소)합니다.

reduce() 메서드는 배열에서 왼쪽에서 오른쪽으로 작동합니다. reduceRight()도 참조하십시오.

reduce() 메서드는 기존 배열을 줄이지 않습니다.

다음 예제는 배열에 있는 모든 숫자의 합계를 찾습니다.

Example

```
    let numbers = [45, 4, 9, 16, 25];
    let sum = numbers.reduce(myFunction);

    document.getElementById("demo").innerHTML = "The sum is " + sum;  // The
sum is 99

    function myFunction(total, value, index, array) {
      return total + value;
    }

    console.log(numbers); // [45, 4, 9, 16, 25]
    console.log(sum);     // 99
```

이 함수는 4 개의 인수를 사용합니다.

- 합계 (초기 값 / 이전에 반환 된 값) (The total(the initial value / previously returned value))
- 항목 값 (The item value)
- 항목 색인 (The item index)
- 배열 자체 (The array itself)

위의 예제에서는 인덱스 및 배열 매개변수를 사용하지 않습니다. 다음과 같이 다시 작성할 수 있습니다:

Example

```
    let numbers = [45, 4, 9, 16, 25];
    let sum = numbers.reduce(myFunction);

    document.getElementById("demo").innerHTML = "The sum is " + sum;  // The
sum is 99

    function myFunction(total, value) {
```

```
      return total + value;
    }

    console.log(numbers); // [45, 4, 9, 16, 25]
    console.log(sum);     // 99
```

reduce() 메서드는 초기 값(initial value)을 받을 수 있습니다.

Example

```
    let numbers = [45, 4, 9, 16, 25];
    let sum = numbers.reduce(myFunction, 100);

    document.getElementById("demo").innerHTML = "The sum is " + sum;  // The sum is 199

    function myFunction(total, value) {
      return total + value;
    }

    console.log(numbers); // [45, 4, 9, 16, 25]
    console.log(sum);     // 199
```

Array.reduceRight()

reduceRight() 메서드는 각 배열 요소에서 함수를 실행하여 단일 값을 생성 (축소)합니다.

reduceRight()는 배열에서 오른쪽에서 왼쪽으로 작동합니다. reduce()도 참조하십시오.

reduceRight() 메서드는 기존 배열을 줄이지 않습니다.

다음 예제는 배열에 있는 모든 숫자의 합계를 찾습니다.

Example

```
    let numbers = [45, 4, 9, 16, 25];
    let sum = numbers.reduce(myFunction);

    document.getElementById("demo").innerHTML = "The sum is " + sum;  // The sum is 99

    function myFunction(total, value, index, array) {
      return total + value;
    }

    console.log(numbers); // [45, 4, 9, 16, 25]
```

```
    console.log(sum);     // 99
```

이 함수는 4 개의 인수를 사용합니다.

- 합계 (초기 값 / 이전에 반환 된 값)
- 항목 값
- 항목 색인
- 배열 자체

위의 예제에서는 인덱스 및 배열 매개변수를 사용하지 않습니다. 다음과 같이 다시 작성할 수 있습니다.

Example

```
    let numbers = [45, 4, 9, 16, 25];
    let sum = numbers.reduce(myFunction);

    document.getElementById("demo").innerHTML = "The sum is " + sum;  // The
sum is 99

    function myFunction(total, value) {
      return total + value;
    }

    console.log(numbers); // [45, 4, 9, 16, 25]
    console.log(sum);     // 99
```

Array.every()

every() 메서드는 모든 배열 값이 테스트를 통과하는지 확인합니다.

다음 예제에서는 모든 배열 값이 18보다 큰지 확인합니다.

Example

```
    let numbers = [45, 4, 9, 16, 25];
    let allOver18 = numbers.every(myFunction);

    document.getElementById("demo").innerHTML = "All over 18 is " +
allOver18;
    // All over 18 is false

    function myFunction(value, index, array) {
      return value > 18;
    }
    console.log(numbers);   // [45, 4, 9, 16, 25]
```

```
    console.log(allOver18); // false
```

이 함수는 3 개의 인수를 가집니다.

- 항목 값
- 항목 색인
- 배열 자체

콜백 함수가 첫 번째 매개변수 (값)만 사용하는 경우, 다른 매개 변수는 생략 할 수 있습니다.

Example

```
    let numbers = [45, 4, 9, 16, 25];
    let allOver18 = numbers.every(myFunction);

    document.getElementById("demo").innerHTML = "All over 18 is " +
allOver18;
    // All over 18 is false

    function myFunction(value) {
      return value > 18;
    }
    console.log(numbers);   // [45, 4, 9, 16, 25]
    console.log(allOver18); // false
```

Array.some()은 Internet Explorer 8 이하를 제외한 모든 브라우저에서 지원됩니다.
(Chrome Yes, Edge 9.0, Firefox Yes, Safari Yes, Opera Yes)

Array.some()

some() 메서드는 일부 배열 값이 테스트를 통과하는지 확인합니다.

다음 예제에서는 일부 배열 값이 18보다 큰지 확인합니다.

Example

```
    let numbers = [45, 4, 9, 16, 25];
    let someOver18 = numbers.some(myFunction);

    document.getElementById("demo").innerHTML = "Some over 18 is " +
someOver18;
    // Some over 18 is true

    function myFunction(value, index, array) {
      return value > 18;
    }
```

```
    console.log(numbers);   // [45, 4, 9, 16, 25]
    console.log(someOver18); // true
```

이 함수는 3 개의 인수를 취합니다.

- 항목 값
- 항목 색인
- 배열 자체

Array.some()은 Internet Explorer 8 이하를 제외한 모든 브라우저에서 지원됩니다.
(Chrome Yes, Edge 9.0, Firefox Yes, Safari Yes, Opera Yes)

Array.indexOf()

indexOf() 메서드는 배열에서 요소 값을 검색하고 해당 위치를 반환합니다.

Note: 첫 번째 항목의 위치는 0이고, 두 번째 항목의 위치는 1입니다.

Example

배열에서 “Apple”항목 검색:

```
    let fruits = ["Apple", "Orange", "Apple", "Mango"];
    let a = fruits.indexOf("Apple");
    document.getElementById("demo").innerHTML = "Apple is found in position
" + a;
    // Apple is found in position 0
    console.log(fruits);  // ["Apple", "Orange", "Apple", "Mango"]
    console.log(a);       // 0
```

Array.indexOf()는 Internet Explorer 8 이하를 제외한 모든 브라우저에서 지원됩니다.
(Chrome Yes, Edge 9.0, Firefox Yes, Safari Yes, Opera Yes)

Syntax

```
array.indexOf(item, start)
```

item	**필수. 검색할 항목**
start	선택. 검색을 시작할 위치. 음수 값은 끝에서부터 계산하여 지정된 위치에서 시작하고 끝까지 검색합니다.

Array.indexOf()는 항목을 찾을 수 없는 경우 -1을 반환합니다.

항목이 두 번 이상 있으면, 첫 번째 발생 위치를 반환합니다.

Array.lastIndexOf()

Array.lastIndexOf()는 Array.indexOf()와 동일하지만, 지정된 요소의 마지막 발생 위치를 반환합니다.

Example

배열에서 “Apple”항목 검색:

```
    let fruits = ["Apple", "Orange", "Apple", "Mango"];
    let a = fruits.lastIndexOf("Apple");
    document.getElementById("demo").innerHTML = "Apple is found in position
" + (a + 1);
    // Apple is found in position 3
```

Array.lastIndexOf()는 Internet Explorer 8 이하를 제외한 모든 브라우저에서 지원됩니다.
(Chrome Yes, Edge 9.0, Firefox Yes, Safari Yes, Opera Yes)

Syntax

```
array.lastIndexOf(item, start)
```

item	필수. 검색할 항목
start	선택. 검색을 시작할 위치. 음수 값은 끝에서부터 계산하여 지정된 위치에서 시작하고 처음부터 검색

Array.find()

find() 메서드는 테스트 함수를 통과하는 첫 번째 배열 요소의 값을 반환합니다.

다음 예제는 18보다 큰 첫 번째 요소를 찾습니다 (값 반환).

Example

```
    let numbers = [4, 9, 16, 25, 29];
    let first = numbers.find(myFunction);

    document.getElementById("demo").innerHTML = "First number over 18 is " +
first;

    function myFunction(value, index, array) {
      return value > 18;
    }
```

이 함수는 3 개의 인수를 가집니다.

- 항목 값
- 항목 색인
- 배열 자체

Array.find()는 이전 브라우저에서 지원되지 않습니다. 완전히 지원되는 첫 번째 브라우저 버전은 다음과 같습니다.
(Chrome 45, Edge 12, Firefox 25, Safari 8, Opera 32)

Array.findIndex()

findIndex() 메서드는 테스트 함수를 통과하는 첫 번째 배열 요소의 인덱스를 반환합니다.

다음 예제에서는 18보다 큰 첫 번째 요소의 인덱스를 찾습니다.

Example

```
    let numbers = [4, 9, 16, 25, 29];
    let first = numbers.findIndex(myFunction);

    document.getElementById("demo").innerHTML = "First number over 18 has
index " + first;
    // First number over 18 has index 3
    function myFunction(value, index, array) {
      return value > 18;
    }
```

이 함수는 3 개의 인수를 가집니다.

- 항목 값
- 항목 색인
- 배열 자체

Array.findIndex()는 이전 브라우저에서 지원되지 않습니다. 완전히 지원되는 첫 번째 브라우저 버전은 다음과 같습니다.
(Chrome 45, Edge 12, Firefox 25, Safari 8, Opera 32)

오션, Javascript, Array, Iteration, Methods