# JavaScript Array Methods

- description : JavaScript Array Methods
- author : 오션
- email : shlim@repia.com
- lastupdate : 2021-04-27

## The source of this article

JavaScript Array Methods
JavaScript 메소드 `toString()`은 쉼표로 구분된 배열을 배열 값의 문자열로 변환합니다.

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.toString();  //
Banana,Orange,Apple,Mango
```

`join()` 메서드는 또한 모든 배열 요소를 문자열로 결합합니다.

`join()` 메서드는 `toString()`처럼 동작하지만 추가로 구분자(separator)를 지정할 수 있습니다.

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits.join(" * ");
// Banana * Orange * Apple * Mango
```

## Popping and Pushing

배열로 작업할 때, 쉽게 요소를 제거하고 새 요소를 추가 할 수 있습니다.

이것이 popping과 pushing 입니다.

배열에서 항목을 꺼내거나(popping), 배열로 항목을 넣습니다(pushing).

## Popping

`pop()` 메서드는 배열에서 마지막 요소를 제거합니다.

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
```

```
document.getElementById("demo1").innerHTML = fruits;  //
Banana,Orange,Apple,Mango
fruits.pop();                                          // remove Mango
document.getElementById("demo2").innerHTML = fruits;  // Banana,Orange,Apple
```

pop() 메서드는 빠져나온(poped out) 마지막 요소를 반환합니다.

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = fruits;         //
Banana,Orange,Apple,Mango
document.getElementById("demo2").innerHTML = fruits.pop();  // Mango
document.getElementById("demo3").innerHTML = fruits;         //
Banana,Orange,Apple
```

## Pushing

push() 메서드는 배열의 마지막에 새 요소를 추가합니다.

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits;

function myFunction() {
  fruits.push("Kiwi");
  document.getElementById("demo").innerHTML = fruits;  //
Banana,Orange,Apple,Mango,Kiwi
}
```

push() 메서드는 새로운 배열의 길이를 반환합니다:

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = fruits;  //
Banana,Orange,Apple,Mango

function myFunction() {
  document.getElementById("demo2").innerHTML = fruits.push("Kiwi");  // 5
  document.getElementById("demo1").innerHTML = fruits;   //
Banana,Orange,Apple,Mango,Kiwi
}
```

## Shifting Elements

Shifting은 popping과 동일하며, 마지막 요소 대신에 첫 번째 요소를 작업합니다.

shift() 메서드는 첫 번째 배열 요소를 제거하고, 다른 모든 요소들을 더 낮은 인덱스로(왼쪽으로) “ 이동 ” 시킵니다.

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = fruits;  //
Banana,Orange,Apple,Mango
fruits.shift();      // remove first element "Banana" from fruits
document.getElementById("demo2").innerHTML = fruits;  // Orange,Apple,Mango
```

shift() 메서드는 첫 번째 배열 요소가 제거된 문자열을 반환합니다:

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = fruits;          //
Banana,Orange,Apple,Mango
document.getElementById("demo2").innerHTML = fruits.shift();  // Banana
document.getElementById("demo3").innerHTML = fruits;          //
Orange,Apple,Mango
```

unshift() 메서드는 배열의 시작 부분에 새로운 요소를 추가하고, 기존의 요소들을 배열의 오른쪽으로 이동시킵니다(“unshifts” older elements):

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits;
//Banana,Orange,Apple,Mango

function myFunction() {
  fruits.unshift("Lemon");
  document.getElementById("demo").innerHTML = fruits; //
Lemon,Banana,Orange,Apple,Mango
}
```

unshift() 메서드는 새로운 배열의 길이를 반환합니다.

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = fruits;    //
Banana,Orange,Apple,Mango
document.getElementById("demo2").innerHTML = fruits.unshift("Lemon"); // 5
document.getElementById("demo3").innerHTML = fruits;    //
Lemon,Banana,Orange,Apple,Mango
```

## Changing Elements

**인덱스 번호(index number)**를 사용하여 배열의 요소들에 액세스합니다.

배열의 인덱스는 0 부터 시작합니다. [0]은 첫 번째 배열 요소이고, [1]은 두 번째, [2]은 세 번째...

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = fruits;  //
Banana,Orange,Apple,Mango
fruits[0] = "Kiwi";      // index[0]의 Banana를 "Kiwi"로 교체
document.getElementById("demo2").innerHTML = fruits;  //
Kiwi,Orange,Apple,Mango
```

length 속성은 배열에 새로운 요소를 쉽게 추가하는 방법을 제공합니다.

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits; //
Banana,Orange,Apple,Mango

function myFunction() {
  fruits[fruits.length] = "Kiwi"; // fruits 배열에 새 요소 'Kiwi'를 배열의 마지막
에 추가
  document.getElementById("demo").innerHTML = fruits; //
Banana,Orange,Apple,Mango,Kiwi
}
```

## Deleting Elements

JavaScript 배열은 오브젝트이므로, JavaScript 연산자 delete를 사용하여 요소를 삭제할 수 있습니다:

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = "The first fruit is: " +
fruits[0];  // The first fruit is: Banana
delete fruits[0];   // index[0]의 Banana를 삭제
document.getElementById("demo2").innerHTML = "The first fruit is: " +
fruits[0];  // The first fruit is: undefined
console.log(fruits);  // (4) [empty, "Orange", "Apple", "Mango"]
```

delete를 사용하는 것은 배열에 정의되지 않은 구멍(holes)를 남길 수도 있습니다. 대신에 pop() 또는 shift()를 사용하세요.

## Splicing an Array

splice() 메서드는 배열에 새로운 항목(items)들을 추가하는데 사용할 수 있습니다.

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = "Original Array: " + fruits;
// Original Array: Banana,Orange,Apple,Mango
function myFunction() {
  fruits.splice(2, 0, "Lemon", "Kiwi");
  document.getElementById("demo2").innerHTML = "New Array: " + fruits;  //
New Array: Banana,Orange,Lemon,Kiwi,Apple,Mango
}
```

첫 번째 매개변수 (2)는 새 요소가 추가 (연결) 되어야 하는 위치를 정의합니다.

두 번째 매개변수 (0)는 제거해야 하는 요소 수를 정의합니다.

나머지 매개 변수 (“Lemon”, “Kiwi”)는 추가할 새 요소를 정의합니다.

splice() 메서드는 삭제된 항목이 있는 배열을 반환합니다.

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo1").innerHTML = "Original Array: " + fruits; //
Original Array: Banana,Orange,Apple,Mango

function myFunction() {
  let removed = fruits.splice(2, 2, "Lemon", "Kiwi");
  document.getElementById("demo2").innerHTML = "New Array: " + fruits;  //
New Array: Banana,Orange,Lemon,Kiwi
  document.getElementById("demo3").innerHTML = "Removed Items: " + removed;
// Removed Items: Apple,Mango
}
```

## Using splice() to Remove Elements

영리한 매개변수 설정으로, splice()를 사용하여 배열에 “ 구멍 ” 을 남기지 않고, 요소를 제거할 수 있습니다.

```
let fruits = ["Banana", "Orange", "Apple", "Mango"];
document.getElementById("demo").innerHTML = fruits;   //
Banana,Orange,Apple,Mango
```

```
function myFunction() {
  fruits.splice(0, 1);
  document.getElementById("demo").innerHTML = fruits; // Orange,Apple,Mango
}
```

첫 번째 매개변수 (0)는 새 요소가 추가 (연결)되어야 하는 인덱스의 위치를 정의합니다.

두 번째 매개 변수 (1)는 제거해야 하는 요소의 개수를 정의합니다.

나머지 매개 변수는 생략됩니다. 새로운 요소가 추가되지 않습니다.

# Merging (Concatenating) Arrays

concat() 메서드는 기존의 배열을 병합(연결)하여 새로운 배열을 반듭니다.

## Merging Two Arrays

```
let myGirls = ["Cecilie", "Lone"];
let myBoys = ["Emil", "Tobias", "Linus"];
let myChildren = myGirls.concat(myBoys);

document.getElementById("demo").innerHTML = myChildren; //
Cecilie,Lone,Emil,Tobias,Linus
```

''concat()'' 메서드는 기존 배열을 변경하지 않습니다. 항상 새 배열을 반환합니다.

''concat()'' 메서드는 어떤한 배열 인수도 사용할 수 있습니다.

## Merging Three Arrays

```
let arr1 = ["Cecille", "Lone"];
let arr2 = ["Emil", "Tobias", "Linus"];
let arr3 = ["Robin", "Morgan"];

let myChildren = arr1.concat(arr2, arr3);

document.getElementById("demo").innerHTML = myChildren;
```

```
//Cecille,Lone,Emil,Tobias,Linus,Robin,Morgan
```

concat() 메서드는 문자열을 인수로 사용할 수도 있습니다.

### Merging an Array with Values

```
let myChildren = arr1.concat("Peter");
document.getElementById("demo").innerHTML = myChildren; //
Emil,Tobias,Linus,Peter
```

## Slicing an Array

slice() 메서드는 배열의 일부를 새 배열로 분할합니다.

다음 예제는 배열 요소 1 (“Orange”)에서 시작하는 배열의 일부를 잘라냅니다.

```

```

''slice()'' 메서드는 새 배열을 만듭니다. 소스 배열에서 어떠한 요소도 제거하지 않습니다.

다음 예제는 배열 요소 3 (“Apple”)에서 시작하는 배열의 일부를 분할합니다.

```

```

slice() 메서드는 slice(1, 3)처럼 두 개의 인수를 사용할 수 있습니다.

그런 다음 메서드는 시작 인수에서 끝 인수까지 (포함하지 않음) 요소를 선택합니다.

```

```

첫 번째 예제처럼, end 인수가 생략되면 slice() 메서드는 배열의 나머지 부분을 잘라냅니다.

```

```

# Automatic toString()

JavaScript는 원시 값(primitive value)이 예상될 때, 배열을 쉼표로 구분된 문자열로 자동 변환합니다.

배열을 출력하려고 할 때 항상 이렇게 됩니다.

아래의 두 가지 예제는 동일한 결과를 생성합니다.

```

```

```

```

```
모든 %%JavaScript%% 오브젝트에는 %%toString()%% 메서드가 있습니다.
```

# Finding Max and Min Values in an Array

JavaScript 배열에서 가장 높은 값 또는 가장 낮은 값을 찾기 위한 기본 제공 함수는 없습니다.

다음 챕터(JS Array Sort)에서 이 문제를 해결하는 방법을 배웁니다.

# Sorting Arrays

배열 정렬(sorting arrays)은 다음 챕터(JS Array Sort)에서 다룹니다.

# Complete Array Reference

전체 참조를 보려면 Complete JavaScript Array Reference로 이동하십시오.

이 참조에는 모든 배열 속성 및 메서드에 대한 설명과 예제가 포함되어 있습니다.

오션, Javascript, Array, Methods

From:
http://rwiki.repia.com/ - **문제를 잘 정의하는 것은 문제를 절반 해결한 것이다. - 2023.12**

Permanent link:
**http://rwiki.repia.com/doku.php?id=wiki:javascript:javascript_note:js_array_methods&rev=1619492241**

Last update: **2022/03/10 19:52**