

# Array

Array.prototype

## Methods

**static** **Array.clear()** [{Array}](#)

 [AXUtil.js, line 2567](#)

Array를 빈 Array 로 변경합니다.

### Example

```
1. var a = [1,2,3];
2. trace(a);
3. // [1, 2, 3]
4. trace(a.clear());
5. // []
6. trace(a);
7. // []
```

**static** **Array.convertTree**(parentKey, childKey, hashDigit) [{Object}](#)

 [AXUtil.js, line 2916](#)

리스트형 데이터를 부모 참조키와 자식 참조키를 이용하여 트리형 데이터로 변환처리 합니다.

Name	Type	Default	Description
parentKey	<a href="#">String</a>		
childKey	<a href="#">String</a>		
hashDigit	<a href="#">String</a>	3	<small>optional</small> 트리의 주소값에 해당하는 hash 의 자릿수 단위 설정 (기본값 3)

### Example

```
1. var a = [
2.   {pno:0, no:1, name:"장기영"},
3.   {pno:1, no:2, name:"장기영"},
```

```

4.     {pno:1, no:3, name:"장기영"},
5.     {pno:3, no:4, name:"장기영"},
6.     {pno:3, no:5, name:"장기영"},
7.     {pno:5, no:6, name:"장기영"},
8.     {pno:5, no:7, name:"장기영"}
9. ];
10.
11. var myTree = a.convertTree("pno", "no");
12. trace(myTree);
13. // [{"pno":0, "no":1, "name":"장기영", "subTree":[{"pno":1, "no":2, "name":"장기영",
    "__subTreeLength":0, "subTree":[], "pHash":"000_000", "hash":"000_000_000"}, {"p
    no":1, "no":3, "name":"장기영", "__subTreeLength":2, "subTree":[{"pno":3, "no":4,
    "name":"장기영", "__subTreeLength":0, "subTree":[], "pHash":"000_000_001",
    "hash":"000_000_001_000"}, {"pno":3, "no":5, "name":"장기영",
    "__subTreeLength":2, "subTree":[{"pno":5, "no":6, "name":"장기영",
    "__subTreeLength":0, "subTree":[], "pHash":"000_000_001_001", "hash":"000_000_00
    1_001_000"}, {"pno":5, "no":7, "name":"장기영", "__subTreeLength":0, "subTree":[]
    , "pHash":"000_000_001_001", "hash":"000_000_001_001_001"}], "pHash":"000_000_00
    1", "hash":"000_000_001_001"}], "pHash":"000_000", "hash":"000_000_001"}], "__su
    bTreeLength":2, "pHash":"000", "hash":"000_000"}]

```

**static** **Array.first()** [{Object}](#)

 [AXUtil.js, line 2587](#)

Array의 첫번째 아이템을 반환합니다.

#### Example

```

1. var a = [1,2,3];
2. trace(a.first());
3. // 1
4.
5. var b = [{a:"엑시스제이"}, 2, 3];
6. trace(b.first());
7. // {"a":"엑시스제이"}

```

8.

9. `var c = [[1,2,3], 2, 3];`

10. `trace(c.first());`

11. `// [1, 2, 3]`

## **static** `Array.getIndex(context)` [{Object}](#)

 [AXUtil.js, line 2991](#)

조건에 맞는 아이템을 index 값과 함께 반환합니다.

Name	Type	Description
context	<a href="#">function</a>	

### Example

1. `var b = [1,2,3,4];`

2. `trace(b);`

3. `// [1, 2, 3, 4]`

4. `trace(b.getIndex(function(idx, item){`

5. `return (this.item >= 3);`

6. `}));`

7. `// {"item":3, "index":2}`

## **static** `Array.getMaxObject(key)` [{Object}](#)

 [AXUtil.js, line 2824](#)

Object Array의 키를 정렬한후 가장 큰 값을 반환합니다.

Name	Type	Description
key	<a href="#">String</a>	

### Example

1. `var myArray = [{a:2},{a:99},{a:1}];`

2. `myArray.getMaxObject("a");`

3. `// Object {a: 99}`

## static **Array.getMinObject**(key) [{Object}](#)

 [AXUtil.js, line 2800](#)

Object Array의 키를 정렬한후 가장 작은 값을 반환합니다.

Name	Type	Description
key	<a href="#">String</a>	

### Example

```
1. var myArray = [{a:99},{a:2},{a:1}];
2. myArray.getMinObject("a");
3. // Object {a: 1}
```

## static **Array.getToSeq**(seq) [{Object}](#)

 [AXUtil.js, line 2633](#)

인자값에 해당하는 인덱스의 아이템을 반환합니다.

Name	Type	Description
seq	<a href="#">Number</a>	

### Example

```
1. var a = [1,2,3];
2. trace(a.getToSeq(1));
3. // 2
4.
5. var a = [1,{a:2},3];
6. trace(a.getToSeq(1));
7. // {"a":2}
```

## static **Array.hasObject**(callback) [{Object}](#)

 [AXUtil.js, line 2762](#)

사용자가 정의한 조건에 맞는 아이템을 한 개만 반환합니다.

Name	Type	Description
callBack	function	

### Example

```

1.  var a = [1,2,3,4];
2.  trace(a);
3.  // [1, 2, 3, 4]
4.  trace(a.has(function(idx, item){
5.      return (item == 3);
6.  }));
7.  // 3
8.
9.  var b = [1,2,3,4];
10. trace(b);
11. // [1, 2, 3, 4]
12. trace(b.has(function(idx, item){
13.     return (this.item == 3);
14. }));
15. // 3

```

**static** **Array.last()** [{Object}](#)

 [AXUtil.js, line 2610](#)

Array의 마지막 아이템을 반환합니다.

### Example

```

1.  var a = [1,2,3];
2.  trace(a.last());
3.  // 1
4.

```

```
5. var b = [1, 2, {a:"엑시스제이"}];
```

```
6. trace(b.last());
```

```
7. // {"a": "엑시스제이"}
```

```
8.
```

```
9. var c = [1, 2, [1,2,3]];
```

```
10. trace(c.last());
```

```
11. // [1, 2, 3]
```

**static** **Array.remove(callback)** [{Array}](#)

 [AXUtil.js, line 2670](#)

사용자가 정의한 조건에 맞는 아이템을 제거한 Array 를 반환합니다.

Name	Type	Description
callback	<a href="#">function</a>	remove 처리할 대상에 return true; 하면 true 인 대상이 제거 됩니다.

#### Example

```
1. var a = [1,2,3,4];
```

```
2. trace(a);
```

```
3. // [1, 2, 3, 4]
```

```
4. a = a.remove(function(idx, item){
```

```
5.     return (item == 3);
```

```
6. });
```

```
7. trace(a);
```

```
8. // [1, 2, 4]
```

```
9. * var b = [1,2,3,4];
```

```
10. trace(b);
```

```
11. // [1, 2, 3, 4]
```

```
12. b = b.remove(function(){
```

```
13.     return (this.item == 3 || this.index == 0);
14.   });
15.   trace(b);
16.   // [2, 4]
```

## static **Array.search**(callback) [{Number}](#)

 [AXUtil.js, line 2704](#)

사용자가 정의한 조건에 맞는 아이템 갯수를 반환합니다.

Name	Type	Description
callback	<a href="#">function</a>	

### Example

```
1.   var a = [1,2,3,4];
2.   trace(a);
3.   // [1, 2, 3, 4]
4.   trace(a.search(function(idx, item){
5.       return (item < 3);
6.   }));
7.   // 2
```

## static **Array.searchObject**(callback) [{Array}](#)

 [AXUtil.js, line 2729](#)

사용자가 정의한 조건에 맞는 아이템을 모두 반환합니다.

Name	Type	Description
callback	<a href="#">function</a>	

### Example

```
1.   var a = [1,2,3,4];
2.   trace(a);
3.   // [1, 2, 3, 4]
```

4. `trace(a.searchObject(function(idx, item){`

5.  `return (item < 3);`

6.  `}));`

7.  `// [1, 2]`

8.

9. `var b = [1,2,3,4];`

10. `trace(b);`

11.  `// [1, 2, 3, 4]`

12. `trace(b.searchObject(function(idx, item){`

13.  `return (this.item < 3);`

14.  `}));`

15.  `// [1, 2]`