

27장 배열

모던 자바스크립트 Deep Dive

배열이란

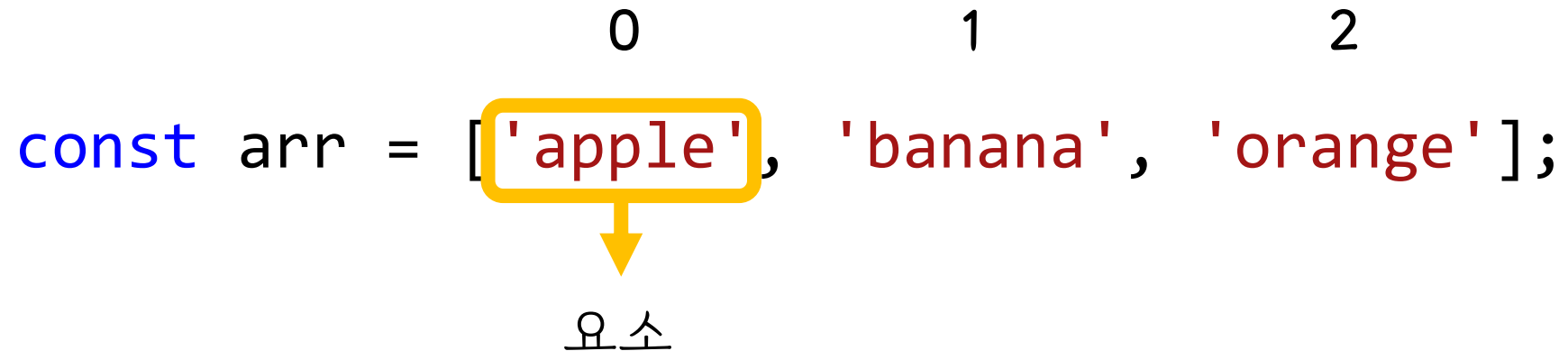
- 여러 개의 값을 순차적으로 나열한 자료구조

0 1 2

```
const arr = ['apple', 'banana', 'orange'];
```

↓

요소



배열이란

- 여러 개의 값을 순차적으로 나열한 자료구조

```
arr[0]; // 'apple'
```

```
arr[1]; // 'banana'
```

```
arr[2]; // 'orange'
```

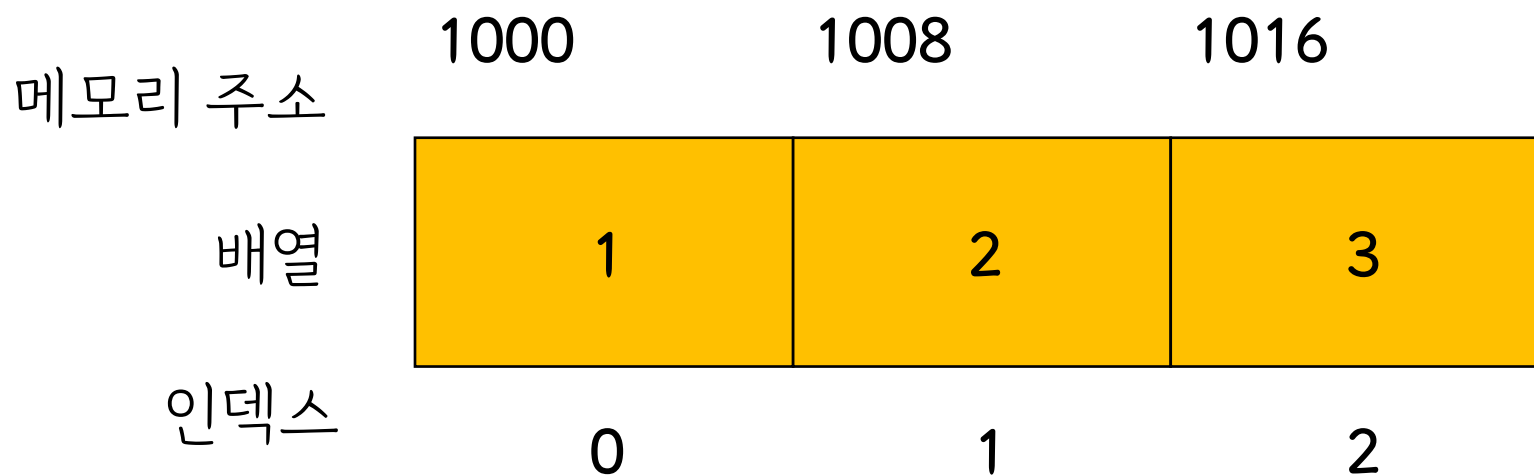
배열은 객체 타입

- 배열은 객체지만 일반 객체와 구별되는 특징이 존재

구분	객체	배열
구조	프로퍼티 키와 프로퍼티 값	인덱스와 요소
값의 참조	프로퍼티 키	인덱스
값의 순서	X	0
length 프로퍼티	X	0

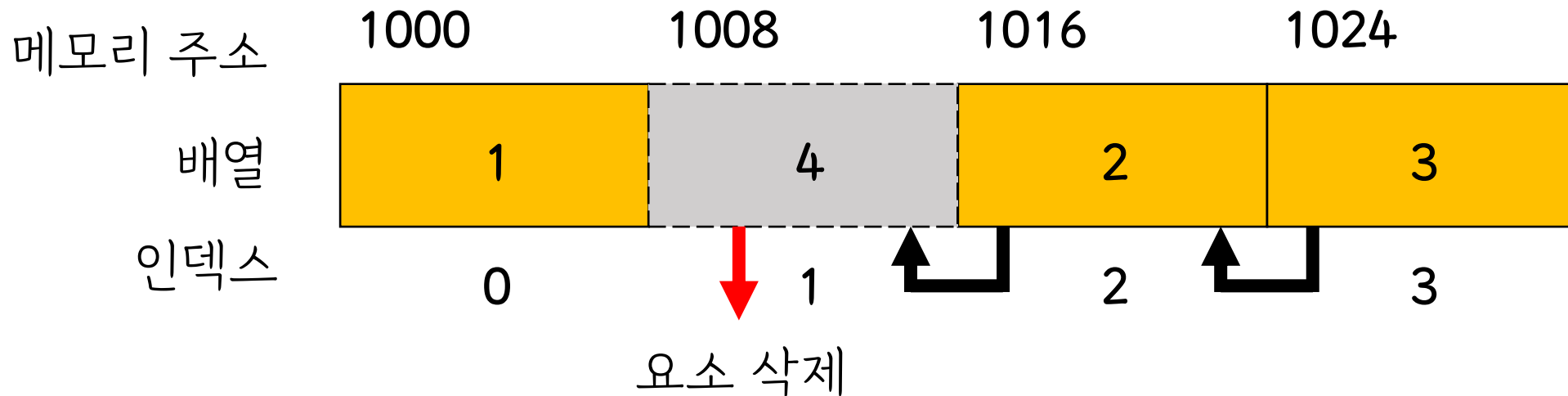
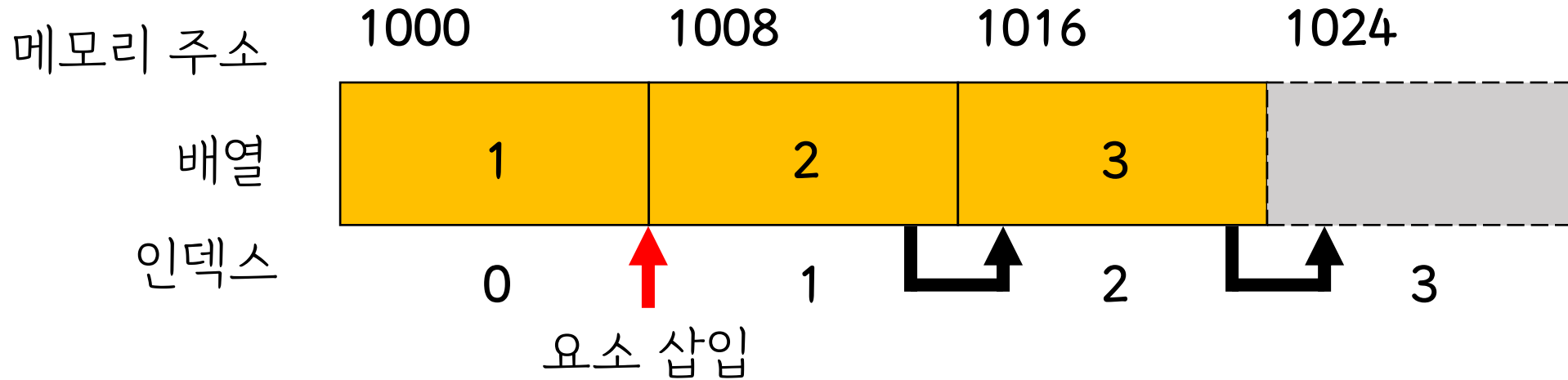
일반적인 배열과 자바스크립트 배열

- **밀집 배열**: 배열의 요소가 하나의 데이터 타입으로 통일되어 있으며 서로 연속적으로 인접해 있음



검색 대상 요소의 메모리 주소 = 배열의 시작 메모리 주소 + 인덱스 * 요소의 바이트 수

일반적인 배열과 자바스크립트 배열



일반적인 배열과 자바스크립트 배열

- 희소 배열: 배열의 요소가 연속적으로 이어져 있지 않은 배열

```
const arr = [  
    'string', 10, true, null, undefined, NaN, Infinity,  
    [], {}, function () {},  
];
```

- 자바스크립트에서 사용할 수 있는 모든 값은 배열의 요소가 될 수 있음

일반적인 배열과 자바스크립트 배열

- 일반적인 배열

- ✓ 인덱스로 요소에 빠르게 접근 가능
- ✓ 특정 요소 검색, 삽입, 삭제하는 경우 효율 x

- 자바스크립트 배열

- ✓ 인덱스로 요소에 접근할 경우 상대적으로 성능 떨어짐
- ✓ 특정 요소 검색, 삽입, 삭제하는 경우 빠른 성능
- ✓ 배열을 일반 객체와 구별하여 더 배열처럼 동작하도록 최적화

length 프로퍼티와 희소 배열

- length 프로퍼티 값은 배열에 요소를 추가하거나 삭제하면 자동 갱신됨

```
const arr = [1, 2, 3];  
console.log(arr.length); // 3
```

```
arr.push(4);  
console.log(arr.length); // 4
```

```
arr.pop();  
console.log(arr.length); // 3
```

length 프로퍼티와 희소 배열

- length 프로퍼티 값에 임의의 숫자 값을 명시적으로 할당할 수 있음

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

```
arr.length = 3;
```

```
console.log(arr); // [1, 2, 3]
```

- 현재 length 프로퍼티 값보다 작은 숫자 값을 할당하면 배열의 길이가 줄어듦

length 프로퍼티와 희소 배열

- length 프로퍼티 값에 임의의 숫자 값을 명시적으로 할당할 수 있음

```
const arr = [1];  
arr.length = 3;
```

```
console.log(arr.length); // 3  
console.log(arr); // [1, empty x 2]
```

- 현재 length 프로퍼티 값보다 큰 숫자 값을 할당하는 경우, length 프로퍼티 값은 변경되지만 실제로 배열의 길이가 늘어나지는 않음

length 프로퍼티와 희소 배열

- 자바스크립트는 희소 배열을 문법적으로 허용함

```
const sparse = [, 2, , 4];
```

```
console.log(sparse.length); // 4
```

```
console.log(sparse); // [empty, 2, empty, 4]
```

- 희소 배열은 되도록 사용하지 않는 것이 좋으며, 배열에는 같은 타입의 요소를 연속적으로 위치시키는 것을 권장

배열 생성 - 배열 리터럴

- 배열 리터럴은 0개 이상의 요소를 쉼표로 구분하여 대괄호로 묶음

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

```
const arr2 = [1, , 3];
```

```
console.log(arr2[1]); // undefined
```

배열 생성 - Array 생성자 함수

- Array 생성자 함수는 전달된 인수의 개수에 따라 다르게 동작함

```
const arr = new Array(10);
```

```
console.log(arr); // [empty x 10]
```

```
console.log(arr.length); // 10
```

- 전달된 인수가 1개이고 숫자인 경우 length 프로퍼티 값이 인수인 배열 생성

배열 생성 - Array 생성자 함수

- Array 생성자 함수는 전달된 인수의 개수에 따라 다르게 동작함

```
const arr = new Array(1, 2, 3); // [1, 2, 3]
```

```
const arr2 = new Array({}); // [{}]
```

- 전달된 인수가 2개 이상이거나 숫자가 아닌 경우, 인수를 요소로 갖는 배열을 생성함

배열 생성 - Array.of

- Array.of 메서드는 전달된 인수를 요소로 갖는 배열을 생성함

```
const arr = Array.of(1); // [1]
```

```
const arr2 = Array.of(1, 2, 3); // [1, 2, 3]
```

```
const arr3 = Array.of('string'); // ['string']
```

- Array 생성자와 다르게 전달된 인수가 1개이고 숫자이더라도 인수를 요소로 갖는 배열 생성

배열 생성 - Array.from

- Array.from 메서드는 유사 배열 객체 또는 이터러블 객체를 인수로 전달받아 배열로 변환하여 반환함

```
const arr = Array.from({ length: 2, 0: 'a', 1: 'b' });  
// ['a', 'b']
```

```
const arr2 = Array.from('Hello');  
// ['H', 'e', 'l', 'l', 'o']
```

배열 생성 - Array.from

- Array.from을 사용하면 두 번째 인수로 전달한 콜백 함수를 통해 값을 만들면서 요소를 채울 수 있음

```
const arr = Array.from({ length: 3 });  
// [undefined, undefined, undefined]
```

```
const arr2 = Array.from({ length: 3 }, (_, i) => i);  
// [0, 1, 2]
```

배열 요소의 추가

- 존재하지 않는 인덱스를 사용해 값을 할당하면 새로운 요소 추가됨

```
const arr = [0];  
arr[1] = 1;  
console.log(arr); // [0, 1]
```

```
arr[100] = 100;  
console.log(arr); // [0, 1, empty x 98, 100]
```

배열 요소의 갱신

- 이미 요소가 존재하는 요소에 값을 재할당하면 요소값이 갱신됨

```
const arr = [0, 1, 2];
```

```
arr[1] = 10;
```

```
arr['2'] = 20;
```

```
console.log(arr); // [0, 10, 20]
```

배열 요소의 갱신

- 인덱스는 반드시 0 이상의 정수 (또는 정수 형태의 문자열)를 사용해야 함

```
const arr = [];
```

```
arr['foo'] = 3;
```

```
arr.bar = 4;
```

```
arr[1.1] = 5;
```

```
arr[-1] = 6;
```

프로퍼티 추가

```
console.log(arr); // [foo: 3, bar: 4, '1.1': 5, '-1': 6]
```

```
console.log(arr.length); // 2
```

배열 요소의 갱신

- 자바스크립트에서 배열의 요소에 접근할 때 음수를 사용할 수 없음

```
console.log(arr[arr.length - 1]);
```

```
console.log(arr.at(-1));
```

- at() 메서드는 정수 값을 받아, 배열에서 해당 값에 해당하는 인덱스의 요소를 반환.
양수와 음수 모두 지정 가능

배열 요소의 삭제

- 배열의 특정 요소를 삭제함

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

```
delete arr[1];
```

```
console.log(arr); // [1, empty, 3]
```

```
console.log(arr.length); // 3
```

- delete 연산자를 사용해 요소를 삭제하면 희소 배열이 되므로 사용하지 않는 것이
좋음

배열 요소의 삭제

- `Array.prototype.splice`(삭제를 시작할 인덱스, 삭제할 요소 수)

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

```
arr.splice(1, 1); → arr[1]부터 1개의 요소 제거
```

```
console.log(arr); // [1, 3]
```