배열

배열: 본격적인 학습

배열에 대해 본격적으로 공부해보겠습니다. 우선 배열이 무엇인지 이해하는 것이 중요합니다.

1. 배열이란 무엇인가?

배열은 여러 개의 데이터를 한 곳에 모아두는 **긴 상자**로 생각할 수 있습니다. 각 데이터는 **인덱스**라는 번호로 구분되며, 배열의 인덱스는 항상 0부터 시작합니다. 배열에 저장된 데이터는 **인덱스**를 통해 접근하고 조작할 수 있습니다.

2. 인덱스란?

배열의 인덱스는 0부터 시작하는 번호로, 배열 안에 있는 각각의 데이터 위치를 나타냅니다. 배열에 있는 데이터는 모두 인덱스를 통해 접근할 수 있습니다. 예를 들어, 배열에서 첫 번째 데이터는 `index 0`, 두 번째 데이터는 `index 1`에 저장됩니다.

예시: 강아지 종류 배열

```
// 배열 생성
const dogs = ['비숑프리제', '포메라니안', '푸들', '시바견', '말티즈'];

// 배열에서 첫 번째 데이터 읽기
console.log(dogs[0]); // 비숑프리제

// 배열에서 세 번째 데이터 읽기
console.log(dogs[3]); // 시바견
```

3. 배열의 기본 특징

배열은 CRUD 기능을 가지고 있습니다. 즉, 배열 안의 데이터를 생성(Create), 읽기(Read), 수정(Update), 삭제(Delete)할 수 있습니다.

4. 배열의 CRUD 기능

(1) 배열의 생성 (Create)

배열을 생성하는 것은 데이터를 모아서 배열에 저장하는 과정입니다. 앞서 작성한 강아지 배열 예시처럼 배열에 데이터를 넣으면 배열이 생성됩니다.

```
const dogs = ['비숑프리제', '포메라니안', '푸들', '시바견', '말티즈'];
```

(2) 배열의 읽기 (Read)

배열 안에 있는 특정 데이터를 인덱스를 사용해 읽을 수 있습니다. 예를 들어, `dogs[0]`는 배열의 첫 번째 데이터를 반환합니다.

```
console.log(dogs[1]); // 포메라니안
```

(3) 배열의 수정 (Update)

배열의 특정 인덱스에 접근하여 값을 수정할 수 있습니다. 예를 들어, 2번째 강아지를 '허스키'로 바꿀 수 있습니다.

```
dogs[2] = '허스키';
console.log(dogs); // ['비숑프리제', '포메라니안', '허스키', '시바견', '말티즈']
```

(4) 배열의 삭제 (Delete)

배열에서 데이터를 삭제하는 방법은 여러 가지가 있지만, 배열의 끝에서부터 삭제할 때 자주 사용하는 것이 `pop()` 메소드입니다.

```
dogs.pop(); // 마지막 요소인 '말티즈'가 삭제됨
console.log(dogs); // ['비숑프리제', '포메라니안', '허스키', '시바견']
```

배열에서 중간 데이터를 삭제하는 경우 `splice()` 메소드를 사용할 수 있습니다.

```
dogs.splice(1, 1); // 인덱스 1에 있는 '포메라니안' 삭제
console.log(dogs); // ['비숑프리제', '허스키', '시바견']
```

5. 배열의 메소드 정리

- **push()**: 배열의 끝에 데이터를 추가합니다.
- **pop()**: 배열의 마지막 데이터를 삭제합니다.
- **splice()**: 배열의 특정 위치에서 데이터를 삭제하거나 추가할 수 있습니다.

예시: 다양한 배열 조작

```
// 배열의 끝에 '도베르만' 추가 (Create)
dogs.push('도베르만');
console.log(dogs); // ['비숑프리제', '허스키', '시바견', '도베르
만']

// 배열의 중간에서 데이터 삭제 (Delete)
dogs.splice(1, 1); // '허스키' 삭제
console.log(dogs); // ['비숑프리제', '시바견', '도베르만']

// 배열의 첫 번째 데이터 수정 (Update)
dogs[0] = '골든리트리버';
console.log(dogs); // ['골든리트리버', '시바견', '도베르만']
```

이렇게 배열은 데이터의 순서와 인덱스를 이용해 여러 가지 조작을 할 수 있는 유용한 자료 구조입니다. 배열을 다루는 기본 메소드를 잘 활용하면 다양한 상황에서 배열을 쉽게 관리할 수 있습니다.
