자바스크립트 자료구조

자바스크립트에서의 자료구조는 데이터를 효율적으로 저장하고, 관리하며, 접근하는 방법을 정의하는 데 사용되는 구조를 의미합니다. 다양한 내장 자료구조를 활용해 복잡한 데이터를 다루고 알고리즘을 구현할 수 있습니다.

자료구조	설명
배열(Array)	순서가 있는 요소의 집합으로, 고정된 크기를 가지거나 동적으로 크기가 변할 수 있습니다. 각 요소는 인덱스를 통해 접근할 수 있으며, 순차적인 데이터 저장과 접근에 적합합니다.
객체(Object)	키-값 쌍의 모음으로, 데이터를 구조화하는 데 사용됩니다. 속성과 메서드 형태로 데이터와 동작을 캡슐화할 수 있으며, 다양한 타입의 키와 값 사용이 가능합니다.
스택(Stack)	LIFO(Last In, First Out) 원리를 따르는 선형 구조로, 마지막에 추가된 요소가 먼저 제거됩니다. 함수 호출, 실행 취소 등에서 사용됩니다.
큐(Queue)	FIFO(First In, First Out) 원리를 따르는 선형 구조로, 처음에 추가된 요소가 먼저 제거됩니다. 데이터 스트림 처리, 대기열 관리 등에 사용됩니다.
맵(Map)	키-값 쌍을 저장하는 컬렉션으로, 객체와 달리 키로 다양한 타입을 사용할 수 있습니다. 삽입 순서가 유지되며, 빠른 조회, 삽입, 삭제가 가능합니다.
셋(Set)	중복을 허용하지 않는 유니크한 요소의 집합으로, 각 요소는 한 번만 존재합니다. 데이터의 유일성을 보장할 때 유용하며, 요소의 존재 여부를 빠르게 확인할 수 있습니다.
연결 리스트(Linked List)	각 노드가 데이터와 다음 노드를 가리키는 참조를 갖는 선형 구조입니다. 배열과 달리 동적 크기 조정이 용이하며, 삽입과 삭제가 배열에 비해 간단합니다.
트리(Tree)	노드들이 부모-자식 관계를 가진 계층적 구조입니다. 이진 트리, AVL 트리, 붉은-검은 트리 등 다양한 종류가 있으며, 계층적 데이터와 정보를 효율적으로 관리하는 데 사용됩니다.
그래프(Graph)	노드들이 간선으로 연결된 구조로, 네트워크 모델링, 경로 탐색 등에 사용됩니다. 방향성이 있거나 없는 그래프, 가중치가 있는 그래프 등 다양한 형태가 있습니다.

```
// Array
let fruits = ['apple', 'banana', 'orange'];
console.log(fruits[1]); // 'banana'

// Object
let person = {
  name: 'Alice',
  age: 25,
  greet: function() { console.log('Hello!'); }
};
console.log(person.name); // 'Alice'
person.greet(); // 'Hello!'

// Stack
let stack = [];
stack.push(1); // 스택에 추가
stack.push(2);
```

```
console.log(stack.pop()); // 마지막 요소 제거하고 반환 (2)

// Queue
let queue = [];
queue.push(1); // 큐에 추가
queue.push(2);
console.log(queue.shift()); // 첫 번째 요소 제거하고 반환 (1)

// Map
let map = new Map();
map.set('key1', 'value1');
map.set('key2', 'value2');
console.log(map.get('key1')); // 'value1'
console.log(map.has('key2')); // true
map.delete('key1');


// Set
let set = new Set();
set.add('apple');
set.add('banana');
set.add('apple'); // 중복된 'apple'은 추가되지 않음
console.log(set.has('banana')); // true
set.delete('banana');


// Linked List
function ListNode(val) {
  this.val = val;
  this.next = null;
}

let head = new ListNode(1);
head.next = new ListNode(2);

// Tree
function TreeNode(val) {
  this.val = val;
  this.left = this.right = null;
}

let root = new TreeNode(1);
root.left = new TreeNode(2);
root.right = new TreeNode(3);

// Graph
let graph = {};
graph['a'] = ['b', 'c'];
graph['b'] = ['a', 'd'];
graph['c'] = ['a', 'd'];
graph['d'] = ['b', 'c'];
```