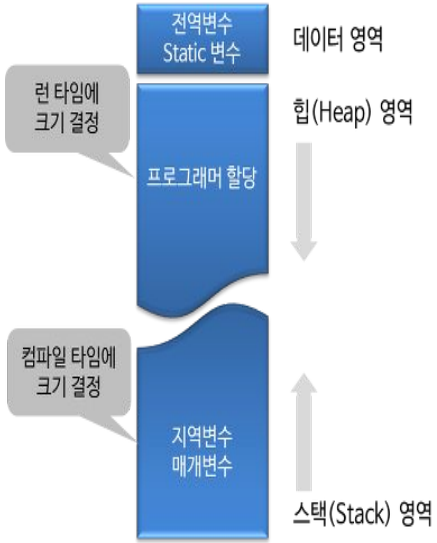
Getter, Setter

접근자 프로퍼티는 'getter(획득자)'와 'setter(설정자)' 메서드로 표현됩니다.
객체 리터럴 안에서 getter와 setter 메서드는 `get`과 `set`으로 나타낼 수 있습니다.

```
let obj = {
 get propName() {
   // getter, obj.propName을 실행할 때 실행되는 코드
 },

 set propName(value) {
   // setter, obj.propNAme = value를 실행할 때 실행되는 코드
 }
};
```



Class

객체 지향 프로그래밍에서 특정 객체를 생성하기 위해 변수와 메소드를 정의하는 일종의 틀로,
객체를 정의하기 위한 상태(멤버 변수)와 메서드(함수)로 구성된다.

```
class Obj {
   constructor() {
      // 생성자를 constructor()로 사용해야 한다.
      this.name = "test"
      this.major = "electronics"
   }
}
```

```
// 클래스 Obj에서 불러오는것. obj가 아닌
console.log(Obj.getStaticTest( isTest: true))
```

Heap영역의 obj가 아닌 데이터영역인 Static변수를 불러오는것으로
클래스인 Obj에서 호출한다.

ParseInt,Float

parseFloat()는 문자열을 실수로 바꾸는 함수
parseInt()는 문자열을 정수로 바꾸는 함수
-parseInt(string, n) ---> string을 n 진법 값으로 바꾼다.

배열을 수정하는 메서드

push 메서드 : 배열의 마지막에 새로운 요소를 추가한 후, 변경된 배열의 길이를 반환
Pop 메서드 : 배열의 마지막 요소를 제거한 후, 제거한 요소를 반환
Unshift 메서드 : 배열의 첫 번째 자리에 새로운 요소를 추가한 후, 변경된 배열의 길이를 반환
Shift 메서드 : 배열의 첫 번째 요소를 제거한 후, 제거한 요소를 반환
Reverse 메서드 : 배열의 순서를 반전

```
var arr = [1, 2, 3]
// 배열의 마지막 요소 제거, 제거된 요소 리턴
arr.pop()  // 3을 제거 ----현재arr 배열 상태 [1,2]

// 배열 마지막에 요소 추가, 배열의 크기 리턴
arr.push("new")  // 현재arr 배열 상태 [1,2]  -----new를 추가
console.log(arr)  //-> [ 1, 2, 'new' ]

// 배열의 첫번째 요소 제거, 제거된 요소 리턴
arr.shift()  // 현재 arr 배열의 상태 [1,2,'new']----1을 제거

// 배열의 처음에 요소 추가, 배열의 크기 리턴
arr.unshift("new")  // 현재 arr 배열의 상태 [2,'new'] ---첫번째 배열에 추가
console.log(arr)  //-> [ 'new', 2, 'new' ]

// 요소 순서를 반전 시킴
var arr = [1,3,5,7]
arr.reverse() //-> [7,5,3,1]
```

배열을 수정하는 메서드

Join 메서드 : 배열의 모든 요소를 연결해 하나의 문자열로 만듬
Splice 메서드 : 기존 요소를 제거하거나 새 요소를 추가하여 배열의 내용을 변경
Slice 메서드 : 원본 배열을 얇게 복사하여 새로운 배열 객체로 반환 따라서 원본 내용이 변경되지 않는다.(splice 와 slice 차이)
Concat메서드 : 인자로 주어진 배열이나 값들을 기존 배열에 합쳐서 새 배열로 반환
var arr = [1, 2, 3]
join
arr.join(+) // +추가 ----> 1+2+3

splice
arr.splice(start:1,deleteCount:2,items:5,8) start는 시작인덱스, deleteCount는 삭제할 수, items 는 삭제된 자리에 추가할 배열값
// [1,2,3] ---> [1,5,8]

slice
arr.slice(1) // [2,3] 이지만 원본 arr [1,2,3]은 변형되지 않는다.

concat
test = arr.concat("실험") // --->test [1,2,3,"실험"] --- 원본 arr[1,2,3] 의 내용은 변하지 않는다.

*sort 메서드: 배열 내부의 요소를 정렬하는데 사용

var arr = [10,1,100,30,50]
arr.sort() [1,10,30,50,100]이 예상되지만 [1,10,110,30,50] 으로 나타남
*원인 내부적으로 숫자를 문자열로 변환 후 값을 비교하기 때문이다.
그렇다면 이를 해결할 방법은 ??
sortedArr = arr.sort((a,b) =>a-b) 로 함수를 전달하면 된다.

비교함수는 두개의 인자(a,b)를 받으며 각각 첫번째, 두번째 요소가 된
    결과값이 0보다 작으면 a가 낮은 색인으로 정렬
    결과값이 0일 경우 순서가 바뀌지 않는다.
    결과값이 0보다 크면 b가 낮은 색인으로 정렬된다.

이를 이용하여 b-a로 전달하면 오름차순으로 정렬할 수 있다.
이름의 길이, 알파벳순으로도 가능

```
var items = [10, 30, 2, 20];

// 오름차순
items.sort((a, b) => a - b);
console.log(items); // [2, 10, 20, 30]

// 내림차순
items.sort((a, b) => b - a);
console.log(items); // [30, 20, 10, 2]

// 이름의 길이순으로 정렬하기
var names = ['Kittie', 'John', 'Sally', 'Einstein'];
names.sort((a, b) => b.length - a.length);
console.log(names); // ['Einstein', 'Kittie', 'Sally', 'John']
```