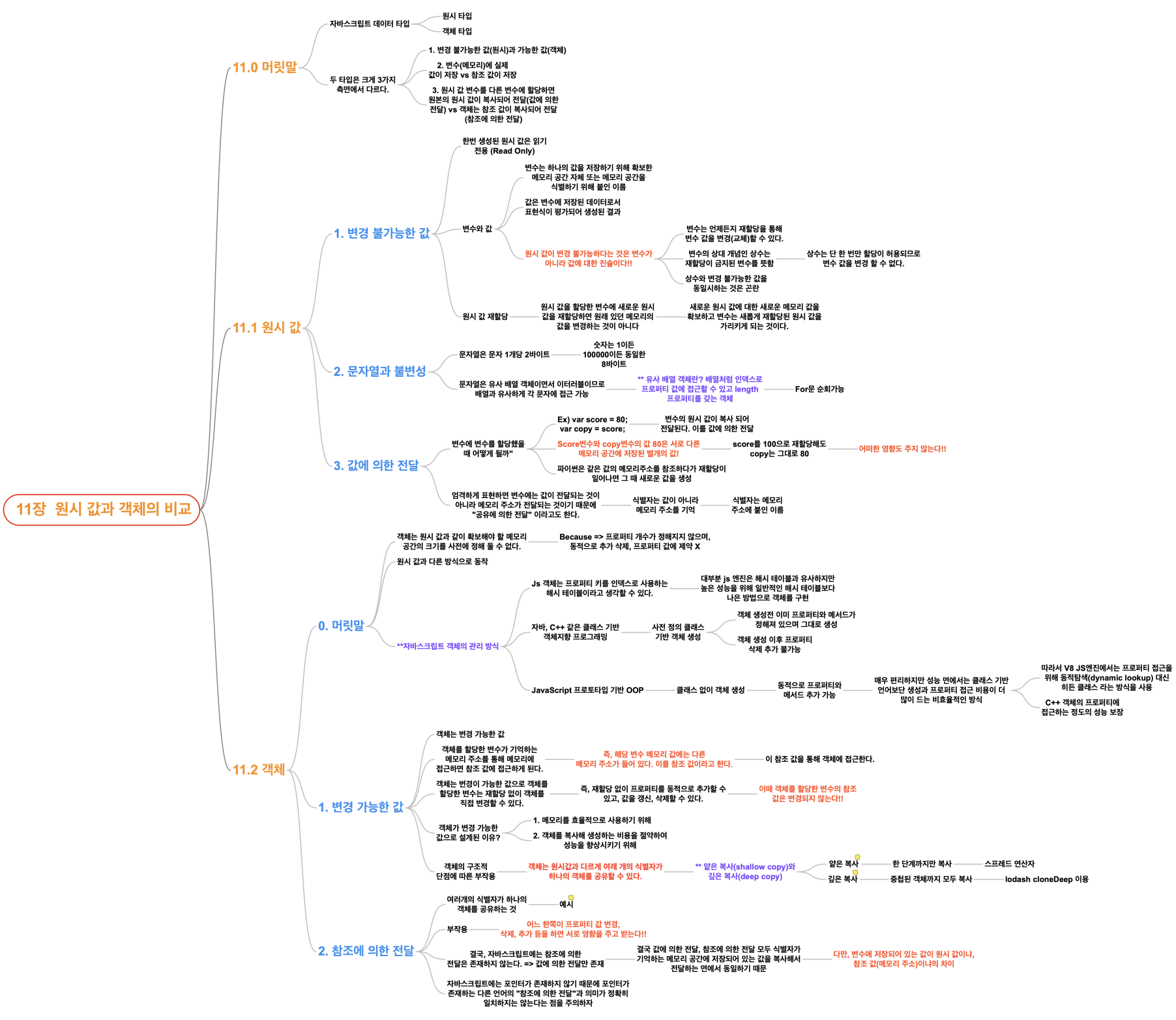
11장 원시 값과 객체의 비교
11.0 머릿말
자바스크립트 데이터 타입
원시 타입
객체 타입
두 타입은 크게 3가지 측면에서 다르다.
1. 변경 불가능한 값(원시)과 가능한 값(객체)
2. 변수(메모리)에 실제 값이 저장 vs 참조 값이 저장
3. 원시 값 변수를 다른 변수에 할당하면 원본의 원시 값이 복사되어 전달(값에 의한 전달) vs 객체는 참조 값이 복사되어 전달 (참조에 의한 전달)
11.1 원시 값
1. 변경 불가능한 값
한번 생성된 원시 값은 읽기 전용 (Read Only)
변수와 값
변수는 하나의 값을 저장하기 위해 확보한 메모리 공간 자체 또는 메모리 공간을 식별하기 위해 붙인 이름
값은 변수에 저장된 데이터로서 표현식이 평가되어 생성된 결과
원시 값이 변경 불가능하다는 것은 변수가 아니라 값에 대한 진술이다!!
변수는 언제든지 재할당을 통해 변수 값을 변경(교체)할 수 있다.
변수의 상대 개념인 상수는 재할당이 금지된 변수를 뜻함
상수는 단 한 번만 할당이 허용되므로 변수 값을 변경 할 수 없다.
상수와 변경 불가능한 값을 동일시하는 것은 곤란
원시 값 재할당
원시 값을 할당한 변수에 새로운 원시 값을 재할당하면 원래 있던 메모리의 값을 변경하는 것이 아니다
새로운 원시 값에 대한 새로운 메모리 값을 확보하고 변수는 새롭게 재할당된 원시 값을 가리키게 되는 것이다.
2. 문자열과 불변성
문자열은 문자 1개당 2바이트
숫자는 1이든 100000이든 동일한 8바이트
문자열은 유사 배열 객체이면서 이터러블이므로 배열과 유사하게 각 문자에 접근 가능
** 유사 배열 객체란? 배열처럼 인덱스로 프로퍼티 값에 접근할 수 있고 length 프로퍼티를 갖는 객체
For문 순회가능
3. 값에 의한 전달
변수에 변수를 할당했을 때 어떻게 될까"
Ex) var score = 80; var copy = score;
변수의 원시 값이 복사 되어 전달된다. 이를 값에 의한 전달
Score변수와 copy변수의 값 80은 서로 다른 메모리 공간에 저장된 별개의 값!
score를 100으로 재할당해도 copy는 그대로 80
어떠한 영향도 주지 않는다!!
파이썬은 같은 값의 메모리주소를 참조하다가 재할당이 일어나면 그 때 새로운 값을 생성
엄격하게 표현하면 변수에는 값이 전달되는 것이 아니라 메모리 주소가 전달되는 것이기 때문에 "공유에 의한 전달" 이라고도 한다.
식별자는 값이 아니라 메모리 주소를 기억
식별자는 메모리 주소에 붙인 이름
11.2 객체
0. 머릿말
객체는 원시 값과 같이 확보해야 할 메모리 공간의 크기를 사전에 정해 둘 수 없다.
Because => 프로퍼티 개수가 정해지지 않으며, 동적으로 추가 삭제, 프로퍼티 값에 제약 X
원시 값과 다른 방식으로 동작
**자바스크립트 객체의 관리 방식
Js 객체는 프로퍼티 키를 인덱스로 사용하는 해시 테이블이라고 생각할 수 있다.
대부분 js 엔진은 해시 테이블과 유사하지만 높은 성능을 위해 일반적인 해시 테이블보다 나은 방법으로 객체를 구현
자바, C++ 같은 클래스 기반 객체지향 프로그래밍
사전 정의 클래스 기반 객체 생성
객체 생성전 이미 프로퍼티와 메서드가 정해져 있으며 그대로 생성
객체 생성 이후 프로퍼티 삭제 추가 불가능
JavaScript 프로토타입 기반 OOP
클래스 없이 객체 생성
동적으로 프로퍼티와 메서드 추가 가능
매우 편리하지만 성능 면에서는 클래스 기반 언어보단 생성과 프로퍼티 접근 비용이 더 많이 드는 비효율적인 방식
따라서 V8 JS엔진에서는 프로퍼티 접근을 위해 동적탐색(dynamic lookup) 대신 히든 클래스 라는 방식을 사용
C++ 객체의 프로퍼티에 접근하는 정도의 성능 보장
1. 변경 가능한 값
객체는 변경 가능한 값
객체를 할당한 변수가 기억하는 메모리 주소를 통해 메모리에 접근하면 참조 값에 접근하게 된다.
즉, 해당 변수 메모리 값에는 다른 메모리 주소가 들어 있다. 이를 참조 값이라고 한다.
이 참조 값을 통해 객체에 접근한다.
객체는 변경이 가능한 값으로 객체를 할당한 변수는 재할당 없이 객체를 직접 변경할 수 있다.
즉, 재할당 없이 프로퍼티를 동적으로 추가할 수 있고, 값을 갱신, 삭제할 수 있다.
이때 객체를 할당한 변수의 참조 값은 변경되지 않는다!!
객체가 변경 가능한 값으로 설계된 이유?
1. 메모리를 효율적으로 사용하기 위해
2. 객체를 복사해 생성하는 비용을 절약하여 성능을 향상시키기 위해
객체의 구조적 단점에 따른 부작용
객체는 원시값과 다르게 여래 개의 식별자가 하나의 객체를 공유할 수 있다.
** 얕은 복사(shallow copy)와 깊은 복사(deep copy)
얕은 복사
한 단계까지만 복사
스프레드 연산자
깊은 복사
중첩된 객체까지 모두 복사
lodash cloneDeep 이용
2. 참조에 의한 전달
여러개의 식별자가 하나의 객체를 공유하는 것
예시
부작용
어느 한쪽이 프로퍼티 값 변경, 삭제, 추가 등을 하면 서로 영향을 주고 받는다!!
결국, 자바스크립트에는 참조에 의한 전달은 존재하지 않는다. => 값에 의한 전달만 존재
결국 값에 의한 전달, 참조에 의한 전달 모두 식별자가 기억하는 메모리 공간에 저장되어 있는 값을 복사해서 전달하는 면에서 동일하기 때문
다만, 변수에 저장되어 있는 값이 원시 값이나, 참조 값(메모리 주소)이냐의 차이
자바스크립트에는 포인터가 존재하지 않기 때문에 포인터가 존재하는 다른 언어의 "참조에 의한 전달"과 의미가 정확히 일치하지는 않는다는 점을 주의하자