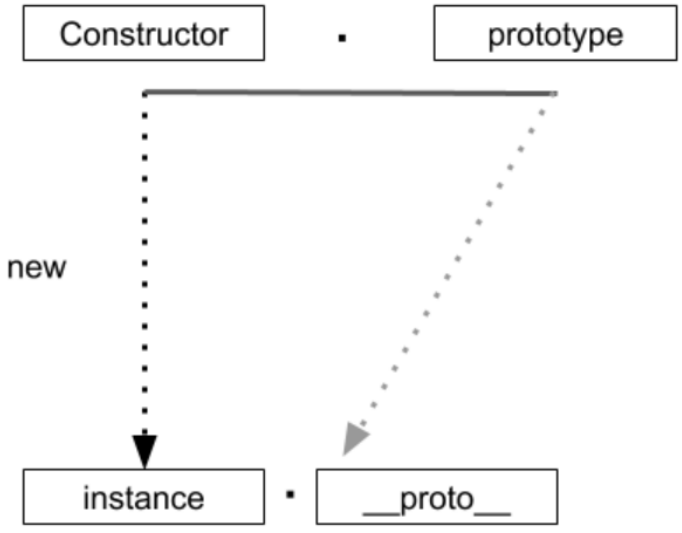
코어 JS - 6단원

06. 프로토타입

- JS는 클래스 기반이 아닌 프로토타입 기반 언어임
- 프로토타입 기반 언어는 어떤 객체를 원형(prototype)으로 삼고 이를 복제함 ≒ 클래스의 상속

1. 프로토타입의 개념 이해

1. constructor, prototype, instance



- 실무에서는 __proto__ 대신 Object.getPrototypeOf() / Object.create() 등을 이용
- __proto__ 는 생략 가능한 프로퍼티

  → 생성자 함수의 prototype에 어떤 메서드나 프로퍼티가 있다면 인스턴스에서도 자신의 것처럼 해당 메서드나 프로퍼티에 접근할 수 있음 / 프포퍼티 내부에 있자 않은 메서드들은 인스턴스가 직접 호출할 수 없으며 Array 생성자 함수에서 직접 접근해야 실행이 가능함(ex. from, isArray 등)

2. constructor 프로퍼티

생성자 함수 프로퍼티 prototype 객체 내부에는 constructor라는 프로퍼티가 있음

→ 인스턴스의 __proto__ 객체 내부에도 마찬가지임

constructor는 생정자 함수(자기 자신)를 참조함

→ 인스턴스로부터 그 원형이 무엇인지를 알 수 있는 수단이 됨

constructor 변경 == 참조하는 대상이 변경됨

- 이미 만들어진 인스턴스의 원형이 바뀐다거나 데이터 타입이 변하는 것은 아님

→ 인스턴스의 생성자 정보를 알아내기 위해 constructor 프로퍼티에 의존하는 게 항상 안전하지는 않음 but, 그렇기 때문에 클래스 상속을 흉내 내는 등이 가능해짐

II. 프로토타입 체인

1. 메서드 오버라이드

인스턴스가 동일한 이름의 프로퍼티 or 메서드를 가지고 있다면?

- 메서드 오버 라이드: 원본을 교체하는 것이 아닌 원본이 그대로 있는 상태에서 다른 대상을 그 위에 얹는 것

2. 프로토타입 체인

prototype 객체가 '객체'임

- 기본적으로 **__proto__**에는 Object.prototype이 연결됨

프로토타입 체인

- 어떤 데이터의 **__proto__** 프로퍼티 내부에 다시 **__proto__** 프로퍼티가 연쇄적으로 이어진 것
- 이 체인을 따라가며 검색하는 것 - 프로토타입 체이닝

3. 객체 전용 메서드의 예외 사항

어떤 생성자 함수이든 prototype은 반드시 객체

→ Object.prototype이 언제나 프로토타입 체인의 최상단에 존재함

- 객체만을 대상으로 동작하는 객체 전용 메서드들은 Object.prototype이 아닌 Object에 스태틱 메서드(static method)로 부여해야함

- 생성자 함수인 Object와 인스턴스인 객체 리터럴 사이에는 this를 통한 연결이 불가능 → 메서드명 앞의 대상이 곧 this가 되는 방식 X → this의 사용을 포기하고 대상 인스턴스를 직접 주입해야하함

```
예외적으로 Object.create를 이용하면 Object.prototype의 메서드에 접근할
Object.create(null)은 __proto__가 없는 객체를 생성함
```

4. 다중 프로토타입 체인

- 프로토타입 체인은 무한대의 단계를 생성할 수 있음 - 7장 클래스에서 다룸
- __proto__가 가리키는 대상, 즉 생성자 함수의 prototype이 연결하고자 하는 상위 생성자 함수의 인스턴스를 바라보게끔 해주면 됨