Static

- 인스턴스 프로퍼티(Instance property)와 정적 프로퍼티(Static property, Class propery)의 차이는?

Instance property
Javascript에서 class를 정의한 후 그 클래스의 객체를 생성한 다음 접근할 수 있는 프로퍼티

Static property
클래스 프로퍼티(class property)라고도 함
정적 프로퍼티는 이름처럼 클래스에 고정되어 있는 프로퍼티
=> 인스턴스를 생성해서는 접근할 수 없고 클래스 이름을 통해서 접근 가능

- 인스턴스 프로퍼티와 정적 프로퍼티는 언제 사용?

만약 특정 객체에 관련된 프로퍼티라면 => 인스턴스 프로퍼티
예) 각 객체가 키나 몸무게 등과 같은 특성을 지녀, 객체마다 다른 상태를 표현할 필요가 있는 경우

반대로 모든 인스턴스가 공유할 필요가 있는 기능이 필요하다면 => 정적 프로퍼티로 정해서 일괄적으로 관리
예) 모든 인스턴스에 '걷기'라는 기능이 공통적으로 필요할때

Extends

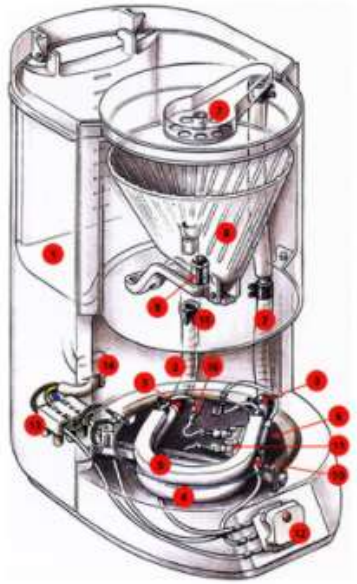
- Class는 extends를 이용하여 상속받을 수 있다.
- 자식 클래스가 부모 클래스를 상속받아서 부모클래스의 기능을 물려받아 사용할 수 있다.
- 부모함수를 참조할 때 super()를 사용
- Overriding a method : 부모의 함수를 받아서 재정의해서 사용할 수 있다.
- 자신의 함수가 없으면 부모의 함수를 실행
- 자바스크립트는 단일상속만을 허용 => 클래스는 클래스 하나만 상속받을 수 있다.

Public, Private, (Protected)

객체 지향 프로그래밍에서 가장 중요한 원리 중 하나는 '내부 인터페이스와 외부 인터페이스를 구분 짓는 것'

- public: 어디서든지 접근 가능. this.이름, 함수인자
- private: 클래스 내부에서만 접근 가능. # 이름
- protected: private과 비슷하지만, 자손 클래스에서도 접근이 가능하다는 점이 다르다. _이름

  ==> 자바스크립트 이외의 다수 언어에서 클래스 자신과 자손 클래스에서만 접근을 허용하는 'protected' 필드를 지원



Mixin

-믹스인(mixin) : 다른 클래스를 상속받을 필요 없이 이들 클래스에 구현되어 있는 메서드를 담고 있는 클래스

⇒ 특정 행동을 실행해주는 메서드를 제공하는데, 단독으로 쓰이지 않고 다른 클래스에 행동을 더해주는 용도로 사용

- 믹스인을 활용하면 다른 클래스를 상속받는 동시에, 믹스인에 구현된 추가 메서드도 사용가능

- 믹스인 안에서 믹스인을 상속받는 것도 가능

Mixin 예제

아래 예시의 믹스인 sayHiMixin은 User에게 '언어 능력'을 부여해줌

```
// mixin
let sayHiMixin = {
  sayHi() {
    alert(`Hello ${this.name}`);
  },
  sayBye() {
    alert(`Bye ${this.name}`);
  }
};

// usage:
class User {
  constructor(name) {
    this.name = name;
  }
}

// copy the methods
Object.assign(User.prototype, sayHiMixin);

// now User can say hi
new User("Dude").sayHi(); // Hello Dude!
```

References

https://blog.naver.com/magnking

https://javascript.info/class-inheritance

https://javascript.info/private-protected-properties-methods

https://javascript.info/mixins