# class

ES6 버전부터 신규로 추가된 객체 설계 문법으로 Prototype이 발전된 형태로 이해하면 됩니다.

Java, C/C++, C# 등의 다른 프로그래밍 언어에서 말하는 Class와 동일한 개념이기 때문에 대부분의 프로그래밍 언어에서 객체지향 개념을 공부하신 경우는 매우 쉽게 접근할 수 있는 개념입니다.

> Class,객체가 포함해야 하는 기능을 설계하는 프로그래밍 패러다임

# #01. 객체 (Object)

- 사전적 의미 : 어떠한 물건이나 대상
- 프로그래밍에서의 의미 : 프로그램에서 표현하고자 하는 기능을 묶기 위한 단위

## 객체를 구성하는 단위

- 객체를 이루는 것은 데이터와 기능이다.
- 데이터는 변수로 표현된다.
    - 객체 안에 포함된 변수를 멤버변수 혹은 프로퍼티라 한다.
- 기능은 메서드(=함수)로 표현된다.

# #02. 클래스 (Class)

객체의 설계도 역할을 하는 프로그램 소스

공장에서 하나의 설계도를 사용하여 여러 개의 제품을 생산할 수 있는 것처럼 하나의 클래스를 통해 동일한 구조를 갖는 객체를 여러 개 생성할 수 있다.

## 클래스의 가장 기본적인 코드 형식

클래스 이름은 명사들의 조합으로 이루어지며 첫 글자는 대문자로 지정하는 것이 관례이다.

```
class 클래스이름 {
    // 생성자 --> 멤버변수 선언 및 기타 초기화
    // getter, setter
    // 메서드
}
```

## 클래스를 통한 객체 생성하기

new 예약어를 사용한다.

```
var|let|const 변수이름 = new 클래스이름();
```

일반적으로 JS에서의 객체 선언은 **const** 키워드를 사용함.

위와 같이 정의하면 변수는 클래스 안에 정의된 모든 기능을 부여받은 특수한 형태의 변수가 되는데 이를 객체라고 하고, 객체는 자신에게 부여된 기능을 점(.)을 통해 접근할 수 있다.

```
객체.멤버변수 = 값;
객체.메서드();
```

## 클래스의 작성 패턴

1. 변수만 정의
2. 메서드만 정의
3. 변수와 메서드를 함께 정의

한대상에 대한 설명이되는 것들 묶음 > 멤버변수만 포함
비슷비슷한 함수를 한 덩어리로 묶음 > 메서드만포함 멤버변수는 클래스안에서 함수끼리 공유가 가능함.

객체라는 개념은 배열이 같은 종류의 변수들만 그룹화 하는 한계를 벗어나 서로 다른 종류의 변수를 그룹화 하는데서 출발한다. (이 상태를 C언어의 구조체라고 한다.)

그렇게 그룹화 해 놓은 변수들간의 관계를 구현하기 위해 메서드를 함께 포함하는 형태로 발전된 것이다.

!number 0 !string 빈문자열 !boolean false ! null, undefined

## 변수만 정의한 클래스

생성자 함수를 정의하고 생성자 함수 안에서 **this**키워드를 통해 객체 안에 탑제될 변수값들을 생성한다.

생성자 함수는 **`constructor`**로 이름이 미리 약속되어져 있다.

### JSON 형식의 객체와 차이

class나 prototype을 통해 new로 생성된 객체는 구조는 동일하지만 각각 독립적인 값을 갖는다.

JSON으로 생성되는 객체는 싱글톤으로서 존재하므로 단순 복사만으로는 동일한 형식을 갖는 두 개의 데이터를 생성할 수 없다.

싱글톤: 프로그램에서 유일하게 하나만 존재하는 형태의 객체

서로 독립적인 데이터를 보유하려면 동일한 JSON 코드를 한 번 더 작성해야 한다.

## 메서드만 정의한 클래스

용도나 목적이 같은 메서드들을 별도의 클래스로 묶어둔다.

## 메서드와 멤버변수를 함께 갖는 클래스

멤버변수의 스코프는 클래스 내의 모든 메서드에서 식별 가능하다. 결국 멤버변수는 모든 메서드가 공유하는 전역 변수의 개념이 된다.

같은 클래스에 속한 멤버변수나 함수끼리는 예약어 **this**를 통해서만 접근 가능하다.

### getter, setter

객체를 통한 멤버변수로의 직접 접근이 소스코드 보안에 좋지 않기 때문에 멤버변수에 대한 직접 접근을 제한하려는 목적으로 (혹은 getter, setter 이름과 구분을 위한 목적으로) 멤버변수의 이름을 언더바(_)로 시작하게 지정한다.

그 후에 멤버변수에 대한 간접적인 할당, 반환을 수행하는 메서드를 별도로 만드는 형태를 getter, setter라고 한다.

```
function MemberClass(){
    this.userName = null;
    this.email = null;
}
class MemberClass(){
    constructor(){
        this.userName = null;
        this.email = null;
    }
}
```

# 클래스 상속(확장 extends)

어떤 클래스의 기능을 다른 클래스에 상속시킨 후 추가적인 기능을 명시하여 원래의 기능을 확장하는 방법.

class를 정의할 때 클래스 이름 뒤에 extends 키워드를 명시하고 상속받고자 하는 부모 클래스의 이름을 지정한다.

### 기능의 확장으로서의 상속

### 여러 클래스간의 공통 기능을 모아 놓는 의미로서의 상속

여러 개의 클래스가 포함하는 기능 중 일부가 동일한 경우 각 클래스로부터 공통되는 부분을 독립적인 클래스로 추출하고 그 클래스를 상속하여 공유하는 처리 기법

#### 공통기능을 정의하는 부모 클래스

#### 부모를 상속받는 자식 클래스(들) 정의

#### 자식 클래스에 대한 객체 생성

부모가 생성자 파라미터를 통해 초기화를 수행하고 있다면 그 생성자는 자식 클래스에게도 상속된다.

그러므로 자식 클래스를 통한 객체 생성시에도 부모가 요구하는 생성자 파라미터를 전달해야 한다.

### 메서드 오버라이드(Override)

> 자바스크립트에서 잘 사용되지 않는다.

클래스 간에 부모-자식 관계가 형성되었을 때 자식 클래스에서 부모 클래스가 갖는 메서드와 동일한 이름의 메서드를 정의하는 기법.

자식이 정의한 메서드에 의해 부모 메서드는 가려지게 된다.

상속 후 자식이 메서드를 추가하는 것이 기능의 확장이라면 메서드 오버라이드는 부모의 기능을 수정하는 개념이다.

## super 키워드

### Override 이전의 원본 기능 호출하기

**this** 키워드가 현재 클래스나 부모로부터 상속 받은 자원을 가리키는 예약어인 반면, **super** 키워드는 부모의 메서드를 Override 하고 있는 자식 클래스 안에서 부모의 원래 기능을 호출하고자 하는 경우에 사용한다.

### 부모 클래스의 생성자

**super** 키워드를 메서드처럼 사용할 경우 부모 클래스의 생성자를 의미한다.

자신의 생성자를 통해 전달받은 파라미터와 추가적으로 가공된 파라미터를 부모의 생성자로 전달하여 객체를 생성하는 방법에 변화를 주고자 할 경우 사용한다.

# #04. 정적 멤버변수, 정적 메서드

클래스에 속한 변수나 함수에 **static** 키워드를 접목하면 객체 생성에 상관 없이 클래스 이름을 통해 항상 접근할 수 있는 정적 기능을 정의할 수 있다.

이렇게 정의된 정적 기능은 각 객체간의 공유 자원이 된다.

setter: 값을 변경한다.

자바스크립트는 이름이 똑같고 메서드가 다르게 정의하는 경우가 불가능하다.