interface

정의

인터페이스는 객체가 구현해야 하는 속성과 메서드를 정의해 사용자 타입을 만드는 방법이다.

```
interface IComplexType {
    id: number;
    name: string;
}

let complexType: IComplexType;
complexType = { id: 1, name: "test"};
complexType = { id: 1}; // TS2741: Property 'name' is missing in type '{ id: number; }' but required in type 'IComplexType'.  index.ts
```

왜?

타입스크립트를 지원하는 ide나 편집기 사용시 자동 완성이나 인텔리센스 옵션 사용이 가능 하기 때문에

인터페이스 컴파일

인터페이스는 타입스크립트의 컴파일 시점 언어 기능이다.

타입스크립트 컴파일러는 프로젝트에 포함된 인터페이스를 자바스크립트 코드로 생성하지 않음.

인터페이스는 컴파일 단계에서 타입 확인을 위해 컴파일러에서만 사용

역할

- 타입 체크를 위해 사용되며, 변수, 함수, 클래스에 사용할수 있다.
- 여러가지 타입을 갖는 프로퍼티로 이루어진 새로운 타입을 정의하는 것과 유사
- 인터페이스에 선언된 프로퍼티 또는 메소드의 구현을 강제하여 일관성을 유지 할 수 있도록 하는 것

특징

- ES6는 인터페이스를 지원하지 않지만 타입스크립트는 인터페이스 지원

선택적 속성

```
interface IOptionProp {
  id: number;
  name?: string;
}

let idOnly: IOptionProp = { id : 1};
```

```
let idAndName: IOptionProp = {id:1 , name: 'idAndName'};
idAndName = idOnly;
```

인터페이스 활용예제

```
interface Persistence<T> {
    load(): T;
    connection(): void;
    disConnection(): void;
    save(): void;
}

class DbPersistence implements Persistence<string> {
    load(): string {
        return "";
    }
    connection(): void {
        console.log("connection");
    }
    disConnection(): void {
        console.log("disconnection");
    }
    save(): void {
        console.log("save");
    }
}

class RedisPersistence implements Persistence<string> {
    load(): string {
        return "";
    }
    connection(): void {
        console.log("connection");
    }
    disConnection(): void {
        console.log("disconnection");
    }
    save(): void {
        console.log("save");
    }
}

let p1: Persistence<string> = new DbPersistence();
let p2: Persistence<string> = new RedisPersistence();
```