TypeScript Type

TypeScript

TypeScript Type

1. 타입스크립트 기본 타입

Boolean

JavaScript/TypeScript의 가장 기본적인 데이터 타입은 true/false 값을 가지는 boolean 이다.

```
let name: string = "HongGilDong"
```

Number

JavaScript에서와 마찬가지로 TypeScript의 모든 숫자는 부동 소수점 값이고 number 타입을 이다. TypeScript는 16 진수 및 10 진수 리터럴 외에도 ECMAScript 2015에 도입된 바이너리 및 8 진수를 지원한다.

```
let decimal: number = 10;
let hex: number = 0x6666;
let binary: number = 0b1010;
let octal: number = 0o744;
```

TypeScript

TypeScript Type

1. 타입스크립트 기본 타입

String

웹 페이지와 서버용 JavaScript에서 프로그램을 생성할 때 흔히 문자열 데이터를 많이 사용한다. 다른 언어와 마찬가지로, TypeScript에서 이러한 문자 데이터 유형을 표현하기 위해 string 타입 문자열을 사용한다.

JavaScript와 마찬가지로 TypeScript은 문자열 데이터를 표현하기 위해 큰 따옴표(") 또는 작은 따옴표(')를 사용한다.

문자열이 여러 행에 걸쳐 있고 표현식을 포함할 수 있는 template string을 사용할 수도 있다. 이러한 문자열은 백틱 / 백 쿼트(`) 문자로 문자를 감싸며, ${expr} 를 이용하여 표현식을 포함할 수 있다.

```
let name: string = 'HongGilDong';
let age: number = 20;
let info = `${name}님의 나이는 ${age + 10}입니다`;
```

TypeScript

TypeScript Type

1. 타입스크립트 기본 타입

Symbol

Symbol 은 ECMAScript 6 에서 추가되었다. Symbol은 유일하고 변경 불가능한 (immutable) 기본값 (primitive value) 이다. 또한, 객체 속성의 key 값으로도 사용될 수 있다. 몇몇 프로그래밍 언어에서는 Symbol을 atom 이라고 부른다. 또, C 언어의 이름있는 열거형 (enum)과도 비슷하다.

```
let key = Symbol();
```

enum

enum은 열거 형 데이터를 생성한다. number에서 확장된 타입으로 첫 번째 Enum 요소에는 숫자 0이 할당된다. 그 다음 값은 특별히 초기화 하지 않는 이상 1씩 증가한다.

```
enum week { Mon, Tue, Wed, Thu };
let today: week = week.Mon;
```

TypeScript

TypeScript Type

1. 타입스크립트 기본 타입

Any

값을 자신 또는 타사 라이브러리의 동적 콘텐츠에서 가져오는 것과 같이 프로그램을 작성할 때 알지 못하는 변수 유형을 설명해야 할 수도 있다. 이 경우 컴파일 시 타입 검사를 하지 않고 지나가도록 해야 한다. 이런 방법을 위해 any 타입을 사용한다.

any 타입은 기존 JavaScript와 같이 작업하는 강력한 방법 중 하나 이다. 컴파일하는 동안 타입 검사를 옵트 인(opt-in)하거나 옵트 아웃 (opt-out) 할 수 있다

```
let key: any;
```

Void

void</code>는 타입이 전혀 없다는 것에서 any의 반대 의미와 비슷하다. 일반적으로 값을 반환하지 않는 함수의 반환 유형으로 이 타입을 사용한다.

TypeScript

TypeScript Type

1. 타입스크립트 기본 타입

Null / Undefiend

TypeScript에서 undefined와 null은 실제로 각각 undefined와 null이라는 이름의 타입을 가지고 있다. void와 매우 비슷하게, 이 타입들은 그 자체로 매우 유용하지는 않다.

기본적으로 null과 undefined는 다른 모든 유형의 하위 유형이다. 즉, null과 undefined를 number와 같은 것에 할당할 수 있다.

--strictNullChecks 플래그를 사용할 경우 null과 undefined는 void와 각각의 타입 변수에만 할당 가능하다.
이러면 많은 일반적인 오류를 피할 수 있다. string또는 null또는 undefined를 전달 하고자 하는 경우, union 타입 string | null | undefined을 사용할 수 있다

TypeScript

TypeScript Type

1. 타입스크립트 기본 타입

Never

never 타입은 절대로 발생하지 않는 값의 타입을 나타낸다. 예를 들어, never는 함수 표현식의 리턴 타입이거나, 항상 예외를 던지는 화살표 함수 표현식이거나, 리턴 하지 않는 표현식이다.

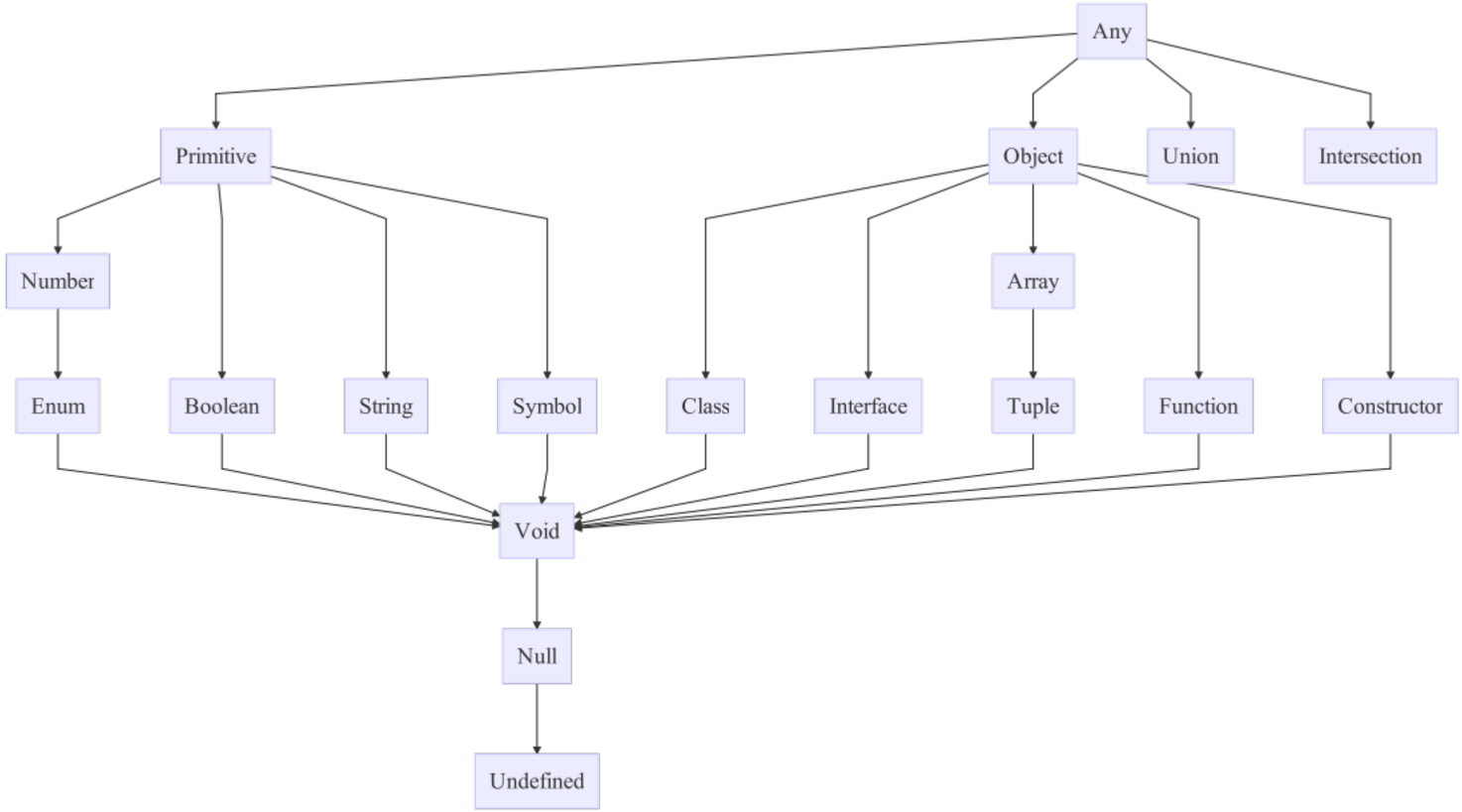
타입은 모든 타입의 서브 타입이며, 모든 타입에 assign 가능합니다. 하지만 어떤 타입도 never (never 자체 제외)의 하위 타입이 아니며 assign 할 수 없다. 어떤 타입도 'never'에 assign되지 않는다.

```
function errFn(error: Error): never {
    throw error;
}
```

TypeScript

TypeScript Type

1. 타입스크립트 기본 타입



TypeScript

TypeScript 선언

2. let / var / const

동일 변수 명의 재 선언

var: 허용함

let: 허용 안함.

const: 허용 안함.

```
var name: number = 10;
var name: number = 20;

let age: number = 10;
let age: number = 20;  //error
```

변수의 값 변경

var: 변경 가능

let: 변경 가능.

const: 변경 불가.

```
const name: string = "Hong";
name = "NolBu";     //error

let age: number = 10;
age = 30;
```

블록 Scope

var: scope를 갖지 않음

let: scope를 갖는다.

const: 변경 불가.

```
let age: number = 10;
if(true){
        let age: number = 20;
}
console.log(age);    => 10
```

TypeScript

TypeScript for

3. for / for in / for of

for

```
let numAry: number[] = [10, 11, 100, 101, 1000];

for(let i = 0; i < numAry.length; i++){
        console.log(`${i}: ${numAry[i]}`);          i는 index를 참조
}

for(let i in numAry) {
        console.log(`${i}: ${numAry[i]}`);          i는 index를 참조
}

for(let i of numAry) {
        console.log(`${i}: ${numAry[i]}`);          i는 value를 참조
}
```