Seoul Software ACademy

웹 개발자 부트캠프 과정

SeSAC x CODINGOn

With. 팀 뤼쳐드

TypeScript

TypeScript란?

- *"Typescript is a **typed superset** of Javascript that compiles to plain Javascript"*

  타입스크립트는 자바스크립트로 컴파일 되는, 자바스크립트의 **타입이 있는 상위집합**이다.

- 2012년에 발표된 오픈 소스 프로그래밍 언어로 Microsoft에 의해 개발됨

https://www.typescriptlang.org/docs/handbook/typescript-from-scratch.html#a-typed-superset-of-javascript

TypeScript란?

- 타입이 있는 자바스크립트
- 즉, JavaScript 의 기본 문법에 자료형 체크하는 기능을 추가한 것
- 자바스크립트가 자의적으로 type 해석을 하고 코드를 실행시켰을 때, 의도와 다른 방식으로 쓰이지 않도록 방지
- 정적 파일 언어

  -> 실행하지 않고도 코드 상의 에러를 알려줌 (실시간 디버깅)

TypeScript는 JavaScript Superset!

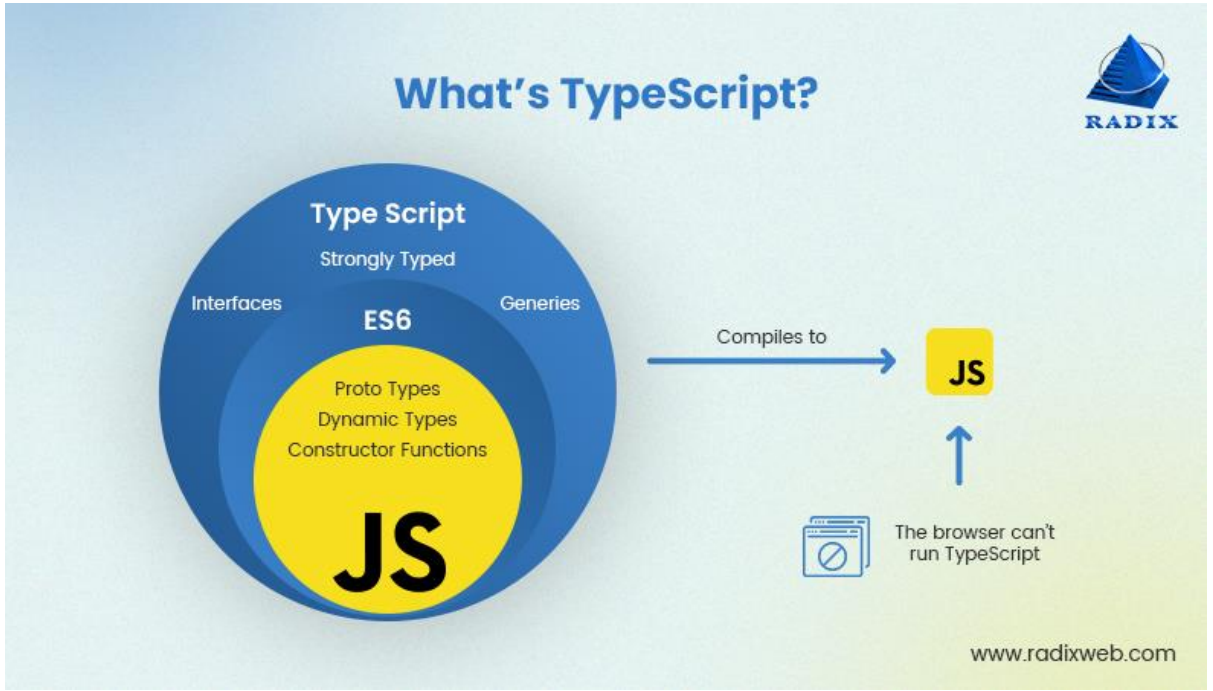
- JavaScript에서 지원하지 않는 기능을 지원!

- TypeScript 대표 기능
  - 엄격한 타입 관리 (Strongly Typed)
  - 제너릭 (Generics)
  - 인터페이스 (Interface)
  - …

트랜스파일러 (Transpiler)

- TS는 웹 브라우저에서 바로 해석될 수 없음
- TS가 JS로 변환되어야 브라우저가 해석 할 수 있음!

- TS가 JS로 출력되기에 트랜스파일러(Transpiler) 라고 부름



JavaScript 의 자료형

- number
- string
- boolean
- null
- undefined
- object

타입과 함께 선언하는 typescript

변수나 함수를 만들어줄 때 타입까지 명시해서 선언

```
let a: number = 1; // 명시적으로 number 타입 지정
let b: string = "안녕하세요"; // 명시적으로 string 타입 지정
let c: object = {
    name: “gildong",
    friends: null,
}; // 명시적으로 object 타입 지정
```

let 변수이름:타입; 으로 선언할 수 있어요! (물론 const 로도 가능하겠죠? ^^)

TS 사용

- 웹 브라우저는 ts 파일을 읽을 수 없기 때문에 ts →js 의 변환 과정이 필요합니다.

1. TypeScript 설치 (전역 설치를 희망한다면 –g 옵션)
   `npm install typescript` (Mac : `sudo npm install typescript`)
2. 설치가 잘 되었는지 version 확인
   `npx tsc -v`
3. tsconfig 파일 생성
   `npx tsc --init`
4. ts 파일을 만들고 js로 변환하고 싶을 때
   `npx tsc 파일이름.ts`

- 실제 사용은 변환된 js 파일을 사용하면 됩니다. (`node 파일이름.js`)

변환 + 실행 자동화

- ts 파일을 일일이 변환한 후에, js 파일을 실행하는게 귀찮아요!
- ts-node 모듈 설치

  `npm install ts-node`

  - package.json 에 ts-node 모듈이 잘 설치되어있는지 확인하기
- 실행은

  `npx ts-node 파일이름.ts`

Ts type 알아보기2 (JS 에서는 없던 type들)

- Tuple
- Enum
- never
- void
- any

Tuple

- JS에서는 배열과 같습니다. (단, 요소의 길이와 타입이 고정된 특수한 배열)
- 순서와 개수가 정해져 있는 배열 (요소의 길이와 타입 고정)
- 일반 배열과 다른 점은 배열의 각각의 타입에 모두 type을 지정해줘야 합니다!
- 순서와 규칙이 있는 배열이 있다면 Tuple을 이용!

```
// 튜플 타입 선언
let drink: [string, number];

// 튜플 초기화
drink = ["cola", 1 ];
```

배열의 element에게 개별적으로 type 선언

drink 라는 배열은 2개의 요소를 가지며
첫번째 요소는 string, 두번째 요소는 number

(즉, 요소들의 타입이 모두 같을 필요는 없음을 나타냄)

Tuple 과 readonly

- 길이와 데이터 타입이 정해진 배열인 tuple
- Readonly ? 읽기만 가능한 data type!

```
let drink3: readonly [string, number] = ["cola", 2500];
```

  - Readonly로 만들어진 tuple 은 데이터를 변경할 수 없음

Enum (열거형)

- 숫자 열거형과 문자 열거형 (특정 값들의 집합)
- 값들에 미리 이름을 정의하고 사용하는 타입

```
enum Auth {
    admin = 0, // 관리자를 0으로 표현
    user = 1,  // 회원은 1로 표현
    guest = 2  // 게스트는 2로 표현
}
```

- 위는 권한 (관리자/사용자/게스트) 별로 숫자값으로 관리를 하는것

Enum (열거형)

```
// 관리자 여부를 숫자로 체크한다.
if (userType !== 0) {
    alert("관리자 권한이 없습니다");
}
```

```
// 회원 권한을 enum으로 정의
enum Auth {
    admin = 0, // 관리자
    user = 1,  // 회원
    guest = 2  // 게스트
}

// Auth.admin(==0) 으로 의미있게 값 체크 가능
if (userType !== Auth.admin) {
    alert("관리자 권한이 없습니다");
}
```

- 0,1,2 로 비교하는 코드를 짜야하지만 개발자가 0,1,2 의 의미를 모두 외우고 있어야 가능함
- 문자열이나 숫자에 미리 의미를 지정해 두고 그룹화할 수 있는 속성
  - Enum으로 정의된 타입 Auth(그룹)
  - Auth에 정의된 0,1,2 를 사용할 때에는 점 접근으로 사용할 수 있음
    Auth.admin / Auth.user / Auth.guest

Enum (열거형) 문법

- JS 의 오브젝트와 유사하지만 선언 이후로는 내용을 추가하거나 삭제할 수 없음
- enum 의 value로는 문자열 혹은 숫자만 허용
- 값을 넣지 않고 선언한다면 숫자형 Enum. 가장 위의 요소부터 0으로 할당돼서 1씩 늘어남

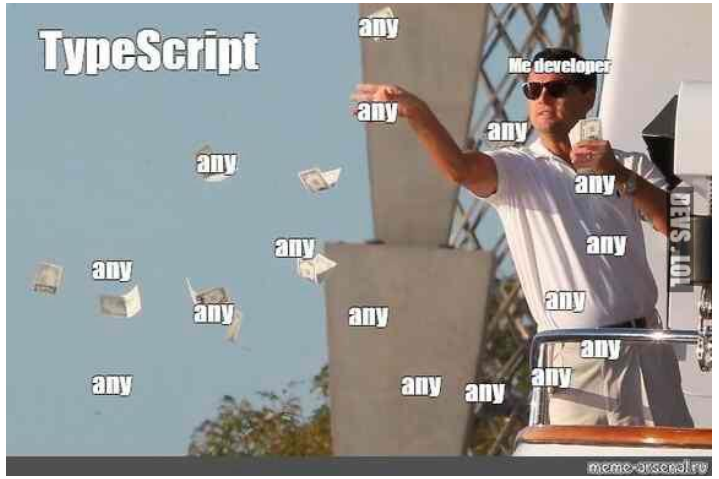
```
enum Cake {
    choco,
    vanilla,
    strawberry,
}
```

```
console.log(Cake.choco); //0
console.log(Cake.vanilla); //1
console.log(Cake.strawberry); //2
```

any (어떤 타입이든)

```
let val:any;
```

- 어떤 타입이든 상관 없이 오류가 나지 않아요!



- 빈 배열을 먼저 선언하고 싶을 때
- 받아오는 데이터 타입이 확실하지 않을 때
- 어떤 타입을 할당해야 할지 알지 못하는 경우 (외부 라이브러리를 사용하거나 동적 콘텐츠를 사용하는 경우)

하지만 정말 어쩔 수 없는 경우가 아니라면 지양!