

알아 두셔야 할 사항

- 책은 타입스크립트 버전 3.7.4를 기준으로 만들어 졌지만, 이 PPT는 버전 4.3.5를 기준으로 작성되었습니다. 이는 책 코드의 5% 정도가 타입스크립트 버전 4에서 정상적으로 컴파일 되지 않기 때문입니다.
- PPT 내용 중 비주얼 스튜디오 코드 화면을 캡처 한 것은 버전 4들 대상으로 코드가 변한 것을 반영한 것이고, PDF를 캡처한 것은 책의 코드가 버전 4에서도 정상적으로 컴파일 된다는 것을 반영한 것입니다.
- 타입스크립트는 tsconfig.json 파일 내용에 매우 민감합니다. 그리고 tsc --init으로 생성된 tsconfig.json 파일은 가장 엄격하게 코드를 컴파일 합니다. 이는 버그가 있는 자바스크립트 코드들을 컴파일 시점에서 잡아주게 하기 위해서 입니다.
- 이에 따라 가능한 이 책의 02-2절에 기술된 타입스크립트 초보자용 tsconfig.json 파일을 사용하시기 바랍니다.

타입스크립트 개발자를 찾고 있어요!!

- 프론트엔드, 백엔드 개발자가 연봉을 올릴 수 있는 방법, 타입스크립트!
- 타입스크립트 = 자바스크립트의 문법을 그대로 수용 + 선택적 정적 타이핑, 클래스 선언, 모듈 지원 등의 기능을 추가한 형태
- 자바스크립트의 단점
 - 명시적인 타입을 기술할 수 없어 타이핑 실수를 코드 작성 단계에서 찾아내기 어렵다.
 - 다른 사람이 작성한 코드를 어떻게 사용해야 하는지 파악하기 어렵다.

TypeScript	JavaScript
TypeScript is an Object-Oriented language	JavaScript is a Scripting language
It has a feature known as Static typing	It does not have static typing
TypeScript gives support for modules	JavaScript does not support modules
It supports optional parameter function	It does not support optional parameter function

타입스크립트로 함수형 프로그래밍 도전

- 함수형 설계 방식으로 작성한 코드는 객체지향 방식의 코드보다 간결하고 논리 정연하다.
- 함수형 프로그래밍의 장점
 - 불변성이라는 원칙에 따라 값이 변하는 것을 최대한 배제하므로 프로그램 검증과 최적화, 그리고 동시에 여러 스레드에서 문제없이 동작하는 프로그램을 쉽게 작성할 수 있다.
 - 함수를 하나의 값처럼 다룰 수 있어서 재사용이 수월하고, 값을 미리 계산하지 않고 꼭 필요할 때만 계산하므로 메모리 절약과 프로그램의 성능에도 긍정적인 영향을 준다.
 - 코드를 함수형 프로그래밍으로 작성하면 전체 개발 시간을 줄이고 유지보수를 쉽게 해서 프로젝트의 생산성을 높일 수 있다.
- 선언형 프로그래밍, 함수 조합, 제네릭, 모나드 등 네 가지 방식의 함수형 프로그래밍을 타입스크립트로 구현하는 방법을 배울 예정

01

타입스크립트 와 개발 환경 만들기

01-1 타입스크립트란 무엇인가?

01-2 타입스크립트 주요 문법 살펴보기

01-3 타입스크립트 개발 환경 만들기

01-1 타입스크립트란 무엇인가?

- 이 절의 목적

- 세상에는 3종류의 자바스크립트 언어가 있다는 것 알기
- 타입스크립트는 타입 기능이 추가된 자바스크립트라는 것 알기
- 개발자들이 타입스크립트를 선호하는 이유 알기

01-1 타입스크립트란 무엇인가?

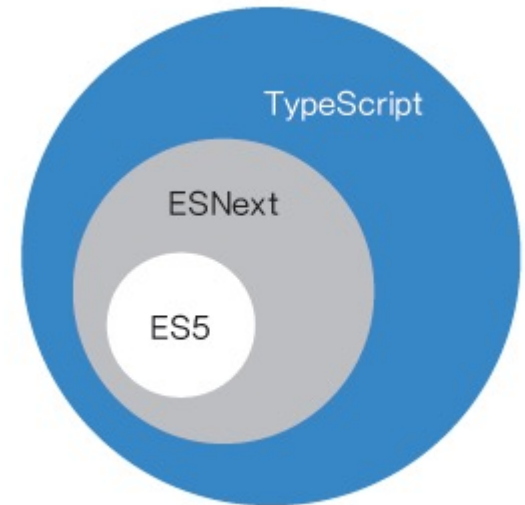
세 종류의 자바스크립트

- es5 - 웹 브라우저에서 동작하는 표준 자바스크립트(ECMAScript version 5)
- esnext - 2015년부터 매년 새로운 기능이 누적된 자바스크립트(즉, es2015 + es2016 + ...)
- typescript - esnext 자바스크립트에 타입(type) 기능이 추가된 것

(MS는 요즘 타입스크립트를 **typed javascript**로 표현하고 있음)

타입스크립트는 누가 만들었나?

- 마이크로소프트가 개발하는 오픈소스 프로그래밍 언어
- 2012년 말 처음 발표
- 아네르스 하일스베르(Anders Hejlsberg, Object Pascal 과 C# 언어 창시자) 주도



ES5와 ESNext, 타입스크립트의 관계도

01-1 타입스크립트란 무엇인가?

자바스크립트에 타입 기능이 있으면 좋은 이유

- 자바스크립트 -
 - 코드 사용자가 코드 작성자의 구현 의도를 정확히 알기 어려움
 - 런타임 때 비로소 파악되므로 시간 낭비 초래
- 타입스크립트 -
 - 코드 사용자가 코드 작성자의 구현 의도를 정확하게 알 수 있음
 - 코드 작성 시점에서 타입스크립트 컴파일러에 의해 오류가 실시간으로 파악되므로 시간 절약

```
1 function makePerson(name: string, age: number) {}  
2 makePerson("Jack", "32")
```

터미널 문제 1 출력 디버그 콘솔 필터(예

TS test.ts 1

⊗ 'string' 형식의 인수는 'number' 형식의 매개 변수에 할당될 수 없습니다. ts(2345) [2, 20]



vscode 내장 타입스크립트 컴파일러가
코드 작성시점에서 오류를 잡아준 예

01

타입스크립트 와 개발 환경 만들기

01-1 타입스크립트란 무엇인가?

01-2 타입스크립트 주요 문법 살펴보기

01-3 타입스크립트 개발 환경 만들기

01-2 타입스크립트 주요 문법 살펴보기

이 절의 목적

- 타입스크립트는 esnext 자바스크립트에 타입 기능을 추가한 자바스크립트 라는 것 알기
- 타입스크립트 개발은 jQuery 스타일의 es5 형태의 코드를 선호하지 않는 것 알기
- 새롭게 등장한 esnext 자바스크립트 문법에 대해 알아보기
- 타입스크립트 고유 문법에 대해 알아보기

01-2 타입스크립트 주요 문법 살펴보기

타입스크립트(typescript) 란?

- 타입스크립트 = Typed JavaScript
- 타입스크립트 = Typed **ESNEXT** JavaScript
- 결론적으로 타입스크립트를 배우려면 **최우선적으로 esnext 자바스크립트를 배워야 함**

01-2 타입스크립트 주요 문법 살펴보기

ESNext의 주요 문법 살펴보기

■ 비구조화 할당 구문(destructuring assignment)

- 객체와 배열에 적용 할 수 있음
- 코드 01~02는 객체에 적용한 비구조화 할당 구문
- 코드 04~05는 배열에 적용한 비구조화 할당 구문
- 코드 07~08은 변수 값을 서로 바꾸는(swap) 것에 비구조화 할당 구문을 적용한 것

비구조화 할당 예

```
01: let person = {name: "Jane", age: 22}
02: let {name, age} = person    // name = "jane", age = 22
03:
04: let array = [1, 2, 3, 4]
05: let [head, ...rest] = array // head = 1, rest = [2, 3, 4]
06:
07: let a = 1, b = 2;
08: [a, b] = [b, a]    // a = 2, b = 1
```

01-2 타입스크립트 주요 문법 살펴보기

ESNext의 주요 문법 살펴보기

■ 화살표 함수(arrow function)

- 자바스크립트는 function 키워드를 사용하여 함수를 만듦
- esnext에서는 => 기호(즉, 화살표)를 사용하여 함수를 만들 수 있게 함
- 화살표 함수는 주로 콜백함수를 구현할 때 사용
- 아래 코드는 filter, map, reduce 등의 메서드의 콜백 함수 호출에 화살표 함수를 적용한 예

화살표 함수 예

```
01: function add(a, b) {return a + b}  
02: const add2 = (a, b) => a + b
```

• array-method-chain.ts

```
01: const multiply = (result, val) => result * val // 07행에서 사용  
02:  
03: let numbers: number [] = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
04: let tempResult = numbers  
05:   .filter(val => val % 2 !== 0)  
06:   .map(val => val * val)  
07:   .reduce(multiply, 1)  
08: let result = Math.round( Math.sqrt(tempResult))  
09: console.log(result) // 945
```

01-2 타입스크립트 주요 문법 살펴보기

ESNext의 주요 문법 살펴보기

- **class, extends, abstract 키워드 도입**
 - 자바스크립트는 프로토타입(prototype) 기반 객체 지향 언어이므로, class 등 다른 OOP 언어에서 흔히 볼 수 있는 키워드가 없음
 - esnext는 다른 객체 지향 언어에서 흔히 볼 수 있는 class, extends, abstract 키워드를 도입하여 객체 지향 코드를 쉽게 작성할 수 있게 함

클래스 예

```
01: abstract class Animal {  
02:   constructor(public name?: string, public age?: number) { }  
03:   abstract say(): string  
04: }  
05: class Cat extends Animal {  
06:   say() {return '야옹'}  
07: }  
08: class Dog extends Animal {  
09:   say() {return '멍멍'}  
10: }  
11:  
12: let animals: Animal[] = [new Cat('야옹이', 2), new Dog('멍멍이', 3)]  
13: let sounds = animals.map(a =>a.say()) // ["야옹", "멍멍"]
```

01-2 타입스크립트 주요 문법 살펴보기

ESNext의 주요 문법 살펴보기

■ 모듈 개념 도입(import 와 export 키워드)

- Node.js 의 등장으로 자바스크립트 코드를 여러 개의 파일로 분할해서 구현할 수 있도록 모듈 개념을 도입
- import 키워드 - 다른 모듈의 기능을 '사용'하려 할 때 사용
- export 키워드 - 다른 모듈에 기능을 '제공'하려 할 때 사용

모듈 예

```
01: import * as fs from 'fs'
02: export function writeFile(filepath: string, content: any) {
03:   fs.writeFile(filepath, content, (err) => {
04:     err && console.log('error', err)
05:   }
06: }
```

01-2 타입스크립트 주요 문법 살펴보기

ESNext의 주요 문법 살펴보기

■ 생성기(function* 와 yield 키워드)

- 파이썬이나 PHP와 같은 몇몇 프로그래밍 언어는 yield라는 특별한 키워드를 제공
- yield는 문법적으로 연산자(operator)
- yield는 반복기(iterator)를 생성할 때 사용
- yield를 호출하는 함수는 반드시 function * 키워드로 만들어야 함
- yield는 yield* 형태로도 사용할 수 있음

생성기 예

```
01: function* gen() {  
02:   yield* [1,2]  
03: }  
04: for(let value of gen()) { console.log(value) } // 1, 2
```

01-2 타입스크립트 주요 문법 살펴보기

ESNext의 주요 문법 살펴보기

■ Promise와 async/await 구문

- Promise는 비동기 io(async io) 함수들의 콜백 함수를 대체한 것
- Promise 객체의 then 메서드를 호출하면 반드시 결과 값을 넘겨줌을 '약속' (즉, Promise)
- await는 Promise 객체를 피연산자로 받는 연산자

Promise와 async/await 예

```
01: async function get() {  
02:   let values = []  
03:   values.push(await Promise.resolve(1))  
04:   values.push(await Promise.resolve(2))  
05:   values.push(await Promise.resolve(3))  
06:   return values  
07: }  
08: get().then(values => console.log(values)) // [1, 2, 3]
```


01-2 타입스크립트 주요 문법 살펴보기

타입스크립트 고유 문법 살펴보기

■ 타입 주석과 타입 추론

- 타입 주석(type annotation) - 변수, 함수의 매개 변수, 함수의 반환 값에 타입을 부여하는 구문
 - 타입 주석 구문은 변수 이름 뒤에 콜론(:) 기호를 붙이고 타입을 명시
 - 코드 01줄은 변수 n의 타입을 number로 설정한 예
- 타입 추론(type inference) - 타입 주석을 생략하면 컴파일러가 생략된 타입을 추측(즉, 추론)

타입 주석과 타입 추론 예

```
01: let n: number = 1 // 타입 주석  
02: let m = 2 // 타입 추론
```

01-2 타입스크립트 주요 문법 살펴보기

타입스크립트 고유 문법 살펴보기

■ 인터페이스

- 자바스크립트는 프로토타입(prototype) 기반 객체 지향 언어이므로 클래스 없이도 객체를 만들 수 있음
- interface 키워드 - 객체의 타입을 만들어 주기 위해 도입된 키워드
- 옆 코드는 person 변수에 interface 키워드로 만든 Person 타입을 부여(즉, 타입 주석)한 예

인터페이스 예

```
01: interface Person {  
02:   name: string  
03:   age?: number  
04: }  
05:  
06: let person: Person = { name: "Jane" }
```

01-2 타입스크립트 주요 문법 살펴보기

타입스크립트 고유 문법 살펴보기

■ 배열(array)과 튜플(tuple)

- 배열 - 같은 타입의 값들의 집합
- 튜플 - 다른 타입의 값들의 집합
- 배열의 타입은 '아이템_타입[]' 타입
- 튜플의 타입은 '[아이템0_타입, 아이템1_타입]' 형태

배열과 튜플

```
01: let numberArray: number[ ] = [1, 2, 3] // 배열
```

```
02: let tuple: [boolean, number, string] = [true, 1, 'Ok'] // 튜플
```

01-2 타입스크립트 주요 문법 살펴보기

타입스크립트 고유 문법 살펴보기

■ 제네릭 타입(generic type)

- 제네릭(generics) - 여러 타입에 공통적으로 구현해야 할 기능을 한꺼번에 구현할 수 있게 하는 언어 구문
- 제네릭 타입 - 타입<타입_변수1, 타입_변수2> 형태로 만든 '새로운' 타입
- 코드는 Container<T> 형태의 제네릭 타입을 만든 뒤, Container<number>, Container<string>이라는 두 개의 새로운 타입을 만들고 있음

제네릭 타입 예

```
01: class Container<T> {  
02:   constructor(public value: T) { }  
03: }  
04: let numberContainer: Container<number> = new Container<number>(1)  
05: let stringContainer: Container<string> = new Container<string>('Hello world')
```

01-2 타입스크립트 주요 문법 살펴보기

타입스크립트 고유 문법 살펴보기

■ 대수 타입(algebraic data type)

- 함수형 언어(functional programming language)에서 사용하는 타입
- 대수 타입은 합집합 타입(union 또는 sum type)과 교집합 타입(intersection 또는 product type) 로 나뉨
- 합집합 타입 - '|' 기호를 사용하여 만든 타입(예: A | B)으로 A '또는(or)' B의 의미를 지닌 타입
- 교집합 타입 - '&' 기호를 사용하여 만든 타입(예: A & B)으로 A '이고(and)' B의 의미를 지닌 타입

대수 타입 예

```
01: type NumberOrString = number | string // 합집합 타입 예
02: type AnimalAndPerson = Animal & Person // 교집합 타입 예
```

01

타입스크립트 와 개발 환경 만들기

01-1 타입스크립트란 무엇인가?

01-2 타입스크립트 주요 문법 살펴보기

01-3 타입스크립트 개발 환경 만들기

01-3 타입스크립트 개발 환경 만들기

이 절의 목적

- 타입스크립트 개발 환경 = 타입스크립트 컴파일러가 설치된 Node.js 개발 환경 이해하기
- 개발자들이 가장 선호하는 편집기는 비주얼 스튜디오 코드(vscode)
- 타입스크립트 개발 환경은 윈도우일 때와 맥일 때가 다르다는 것 이해하기

01-3 타입스크립트 개발 환경 만들기

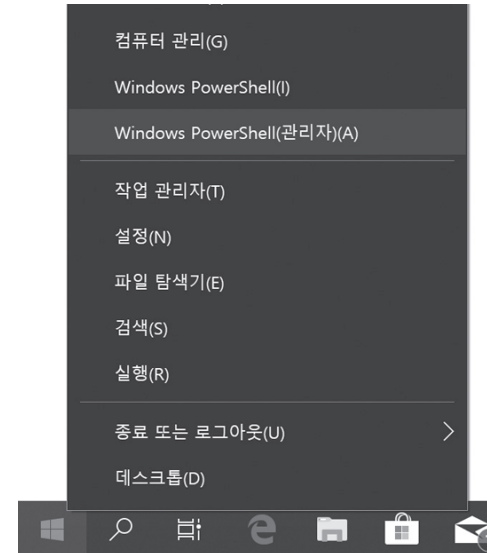
윈도우10 환경

■ scoop 프로그램 설치

- ① scoop는 node.js 와 비주얼 스튜디오 코드 등 다른 프로그램들을 쉽게 설치하게 해주는 프로그램 (<https://scoop.sh> 사이트 참조)
- ② scoop는 다른 프로그램을 설치해 주는 프로그램이므로 관리자 모드 파워셸에서 설치 필요
- ③ 관리자 모드 파워셸에서 다음 명령을 차례로 실행
- ④ 윈도우 환경 변수 설정 창에서 Scoop 변수 생성

```
> Set-ExecutionPolicy RemoteSigned -scope CurrentUser
> $env:SCOOP='C:\Scoop'
> iex (new-object net.webclient).downloadstring('https://get.scoop.sh')
> scoop install aria2
> scoop install git
```

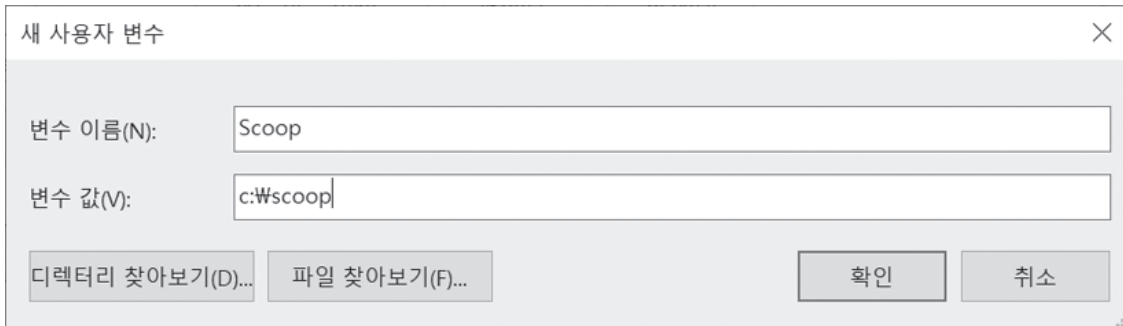
명령 실행 후 [A] 입력



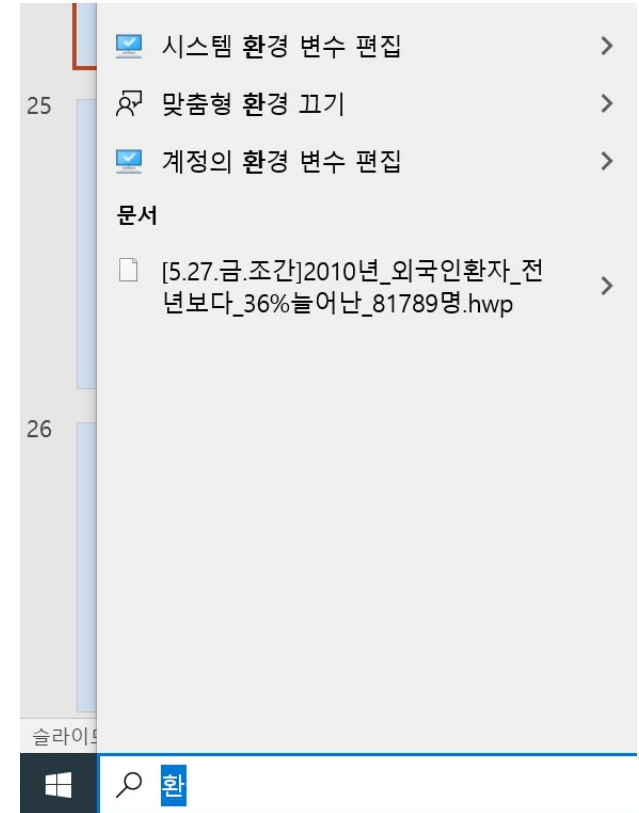
01-3 타입스크립트 개발 환경 만들기

[참고] scoop 사용시 번거로운 점 해결

- scoop 프로그램 사용시 번거로운 점은 다른 프로그램 설치를 위해 새로운 관리자 모드 쉘을 실행할 때 마다 `$env:Scoop='c:\WScoop'` 명령을 실행시켜 줘야 한다는 것
- 이 번거로움은 다음처럼 [시스템 환경 변수 편집] 메뉴를 통해 [새 사용자 변수]를 등록해 주면 없앨 수 있음



환경 변수 설정 모습

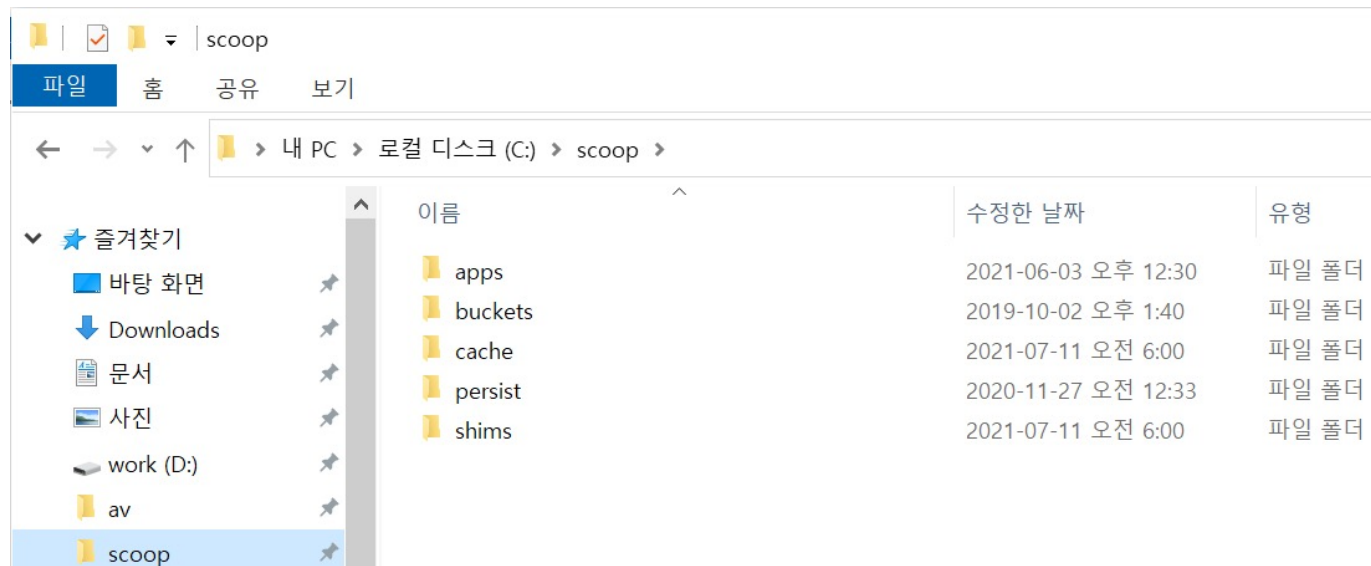


[시스템 환경 변수 편집 메뉴]실행 모습

01-3 타입스크립트 개발 환경 만들기

윈도우10 환경

- scoop 프로그램 설치 완료 모습



scoop 가 c:/Scoop 디렉터리에 설치된 모습

01-3 타입스크립트 개발 환경 만들기

윈도우10 환경

■ 비주얼 스튜디오 코드(vscode) 설치

① 관리자 모드 파워셸에서 다음 명령을 차례로 실행

```
> scoop bucket add extras  
> scoop install vscode
```

② c:/Scoop/apps/vscode/current 디렉터리의 vscode-install-context.reg 더블클릭

로컬 디스크 (C:) > scoop > apps > vscode > current

이름	수정된 날짜	유형	크기
tools	2021-07-08 오후 4:12	파일 폴더	
swiftshader	2021-07-08 오후 4:12	파일 폴더	
resources	2021-07-08 오후 3:55	파일 폴더	
locales	2021-07-08 오후 3:55	파일 폴더	
bin	2021-07-08 오후 3:55	파일 폴더	
vulkan-1.dll	2021-07-08 오후 4:12	응용 프로그램 확장	724KB
vk_swiftshader.dll	2021-07-08 오후 4:12	응용 프로그램 확장	4,544KB
libGLSLv2.dll	2021-07-08 오후 4:12	응용 프로그램 확장	7,721KB
libEGL.dll	2021-07-08 오후 4:12	응용 프로그램 확장	443KB
ffmpeg.dll	2021-07-08 오후 4:12	응용 프로그램 확장	2,279KB
d3dcompiler_47.dll	2021-07-08 오후 4:12	응용 프로그램 확장	4,428KB
Code.exe	2021-07-08 오후 4:13	응용 프로그램	121,397KB
vscode-uninstall-context.reg	2021-07-11 오전 6:00	등록 항목	1KB
vscode-install-context.reg	2021-07-11 오전 6:00	등록 항목	1KB

01-3 타입스크립트 개발 환경 만들기

윈도우10 환경

- Node.js 설치

- ① 관리자 모드 파워셸에서 다음 명령 실행

```
> scoop install nodejs-lts  
> node -v
```

```
Installing 'nodejs-lts' (14.17.0) [64bit]  
Loading node-v14.17.0-win-x64.7z from cache.  
Checking hash of node-v14.17.0-win-x64.7z ... ok.  
Extracting node-v14.17.0-win-x64.7z ... done.  
Linking C:\scoop\apps\nodejs-lts\current => C:\scoop\apps\nodejs-lts\14.17.0  
Persisting bin  
Persisting cache  
Running post-install script...  
'nodejs-lts' (14.17.0) was installed successfully!  
PS C:\WINDOWS\system32> node -v  
v14.17.0
```

01-3 타입스크립트 개발 환경 만들기

윈도우10 환경

- touch 프로그램 설치

- ① 관리자 모드 파워셸에서 다음 명령 실행

```
> scoop install touch
```

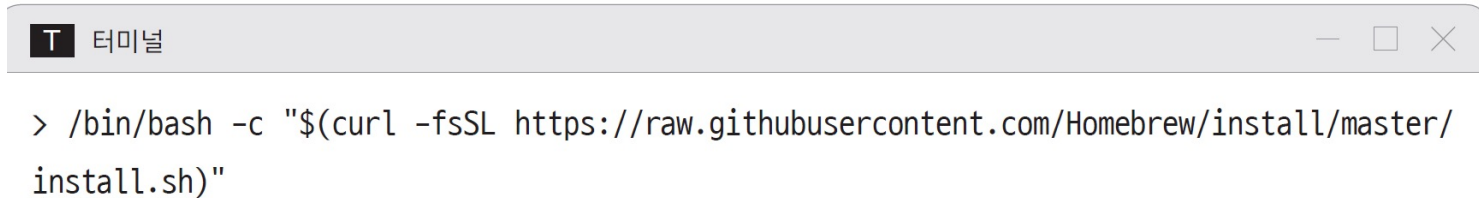
- ② touch 는 파일을 생성할 때 지정한 이름의 파일이 이미 있으면 무시하고, 없으면 해당 이름으로 파일을 만들어 주는 유틸리티(맥, 리눅스는 기본 설치 되어 있음)

01-3 타입스크립트 개발 환경 만들기

맥 환경

- Homebrew 프로그램 설치

- ① homebrew는 맥에서 node.js 와 비주얼 스튜디오 코드를 쉽게 설치하게 해주는 프로그램 (https://brew.sh/index_ko 사이트 참조)
- ② 맥에서 터미널을 하나 열고 다음 명령 실행



```
T 터미널
> /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

01-3 타입스크립트 개발 환경 만들기

맥 환경

- 비주얼 스튜디오 코드(vscode) 설치

- ① 터미널에서 다음 명령 실행



```
T 터미널
> brew install cask
> brew cask install visual-studio-code
```

01-3 타입스크립트 개발 환경 만들기

맥 환경

- Node.js 설치

- ① 터미널에서 다음 명령 실행

```
brew install node@14  
node -v
```


01-3 타입스크립트 개발 환경 만들기

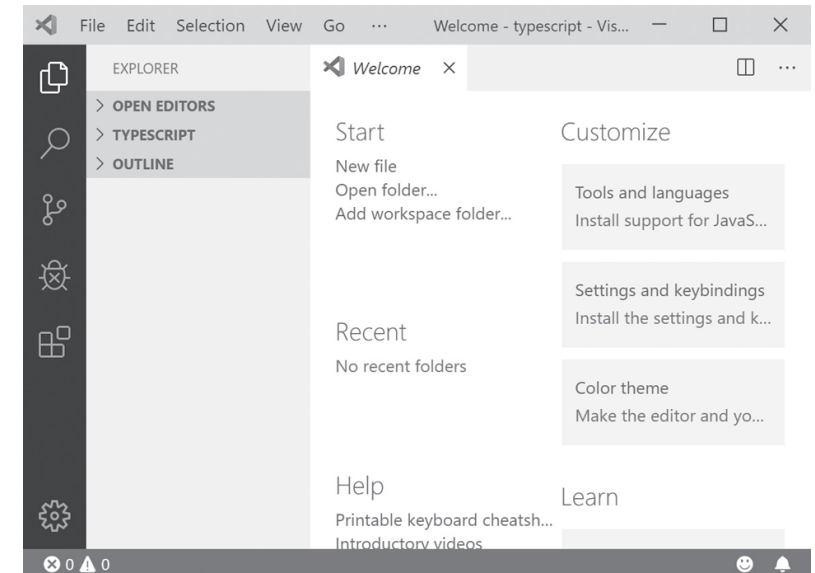
vscode 실행

■ 윈도우

- ① 파일 탐색기에서 특정 디렉터리를 대상으로 [Open with Code] 명령 실행

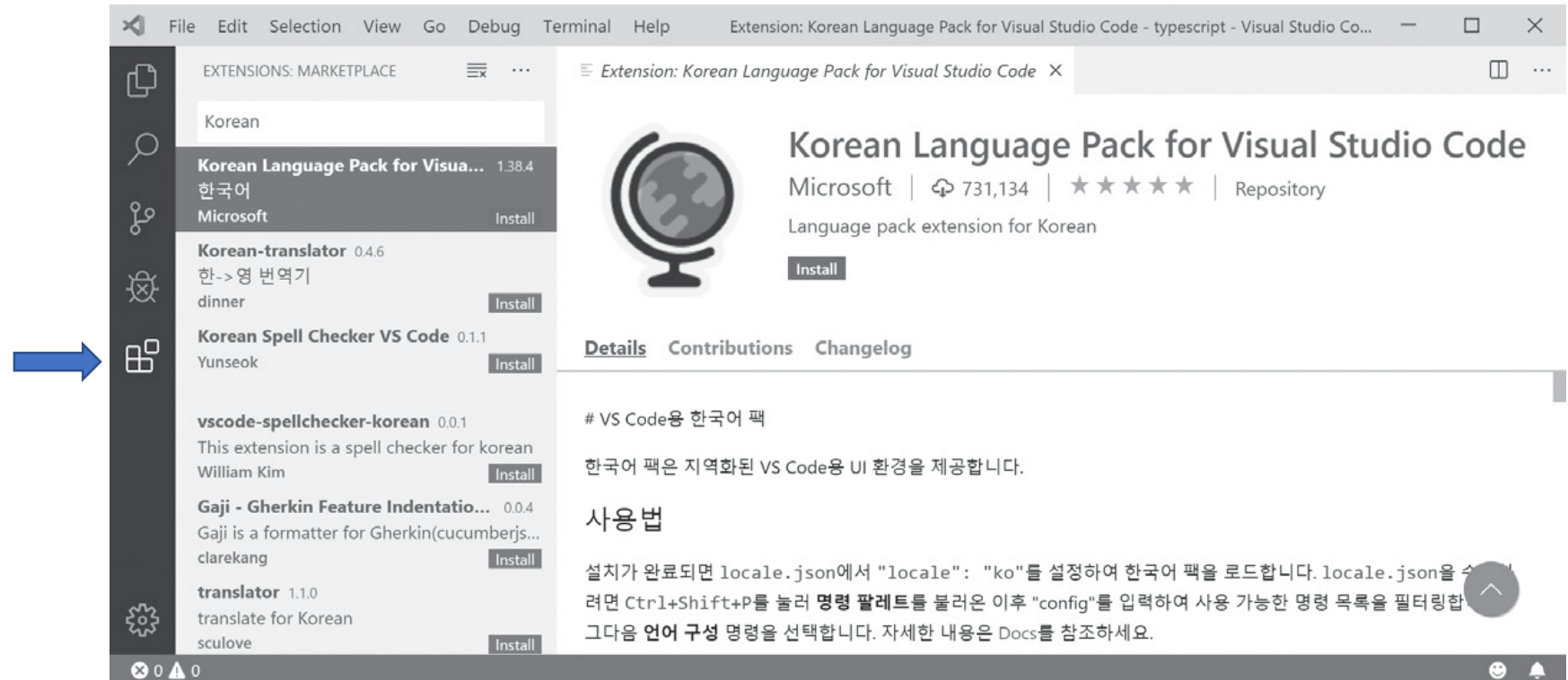
■ 맥

- ① 터미널에서 특정 디렉터리로 이동한 다음 'code .' 명령 실행



01-3 타입스크립트 개발 환경 만들기

vscode 에 한글 언어 확장 팩 설치



01-3 타입스크립트 개발 환경 만들기

vscode 에서 터미널 열기

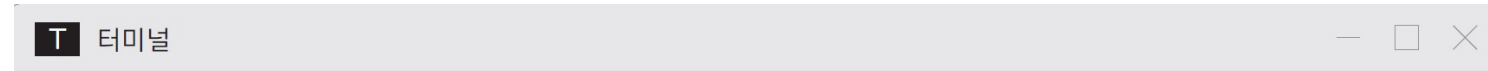
- ① [터미널/새 터미널] 메뉴 실행
- ② Ctrl 키와 ` 키를 동시에 누름



01-3 타입스크립트 개발 환경 만들기


타입스크립트 컴파일러 설치

- vscode 의 터미널 혹은 파워셸 에서 다음 명령 실행



> npm i -g typescript ts-node

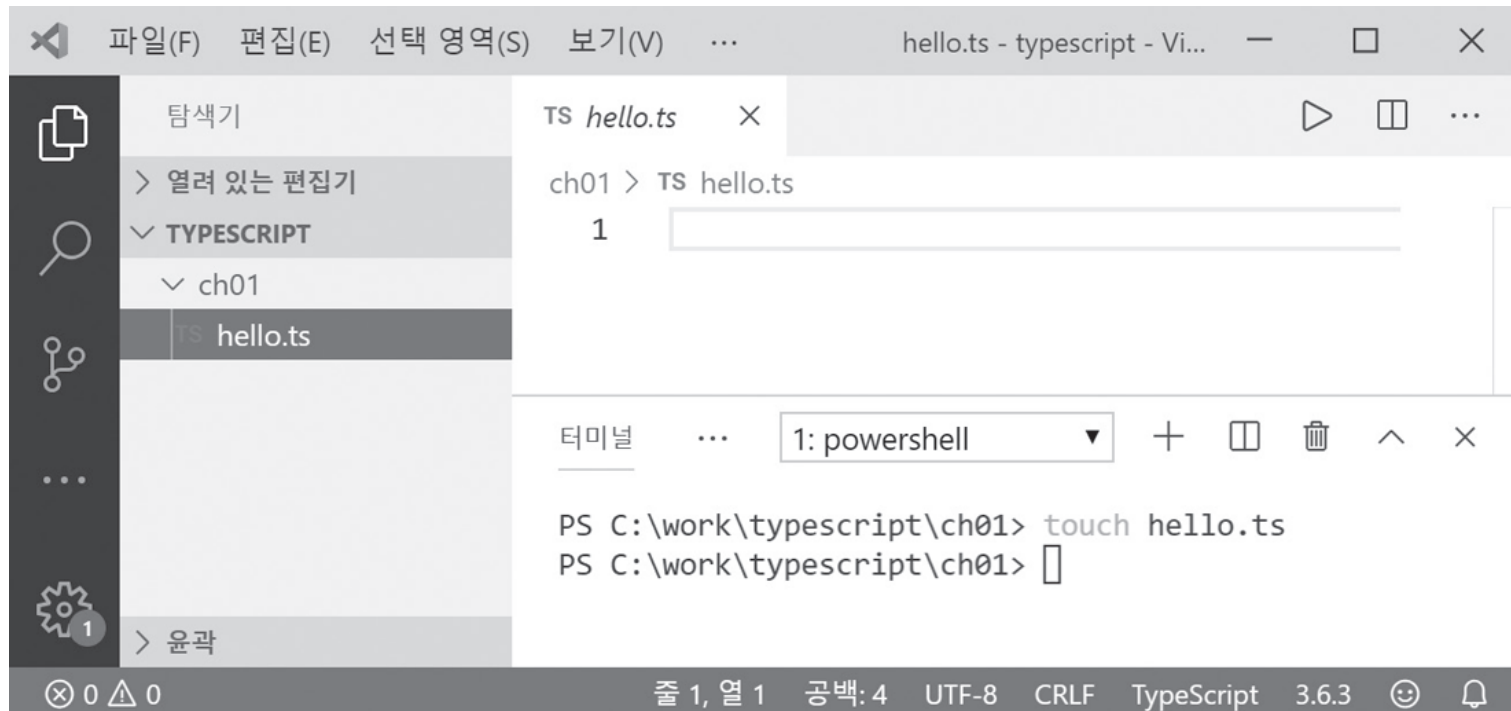
- 다음 명령으로 타입스크립트 컴파일러 버전 확인

 관리자: Windows PowerShell

```
PS C:\WINDOWS\system32> tsc -v
Version 4.3.5
PS C:\WINDOWS\system32>
```

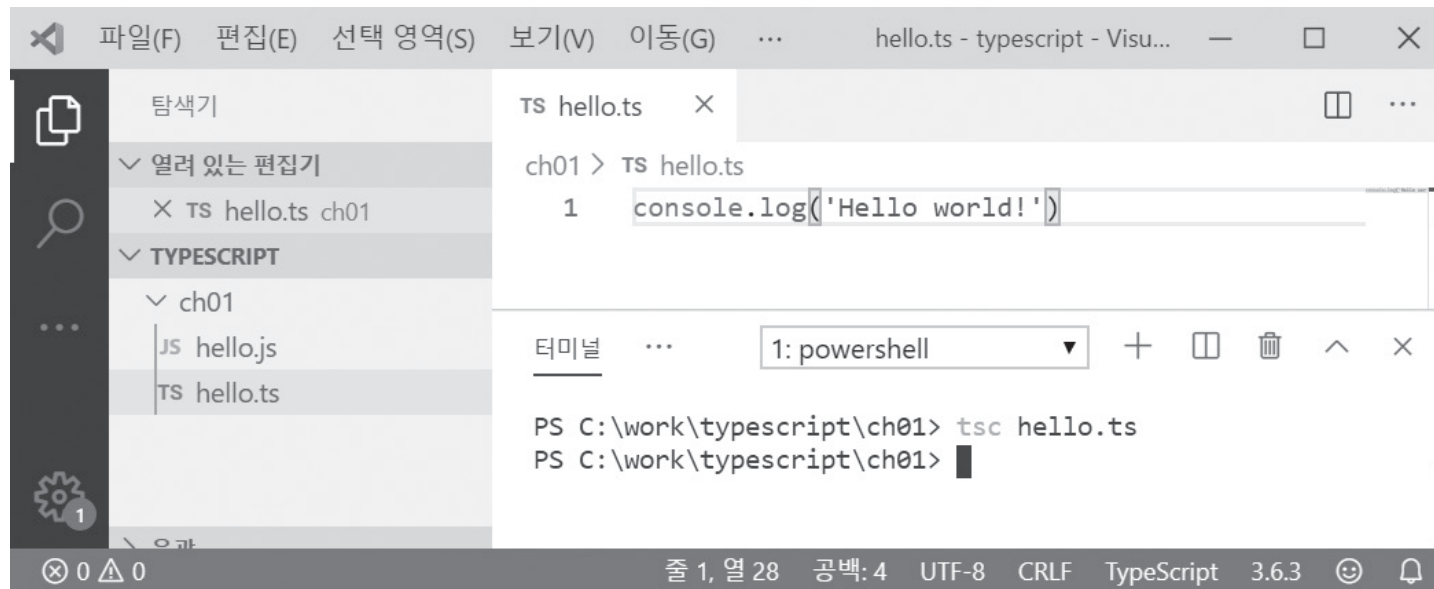
01-3 타입스크립트 개발 환경 만들기

vscode 터미널에서 touch 명령으로 파일 생성하기



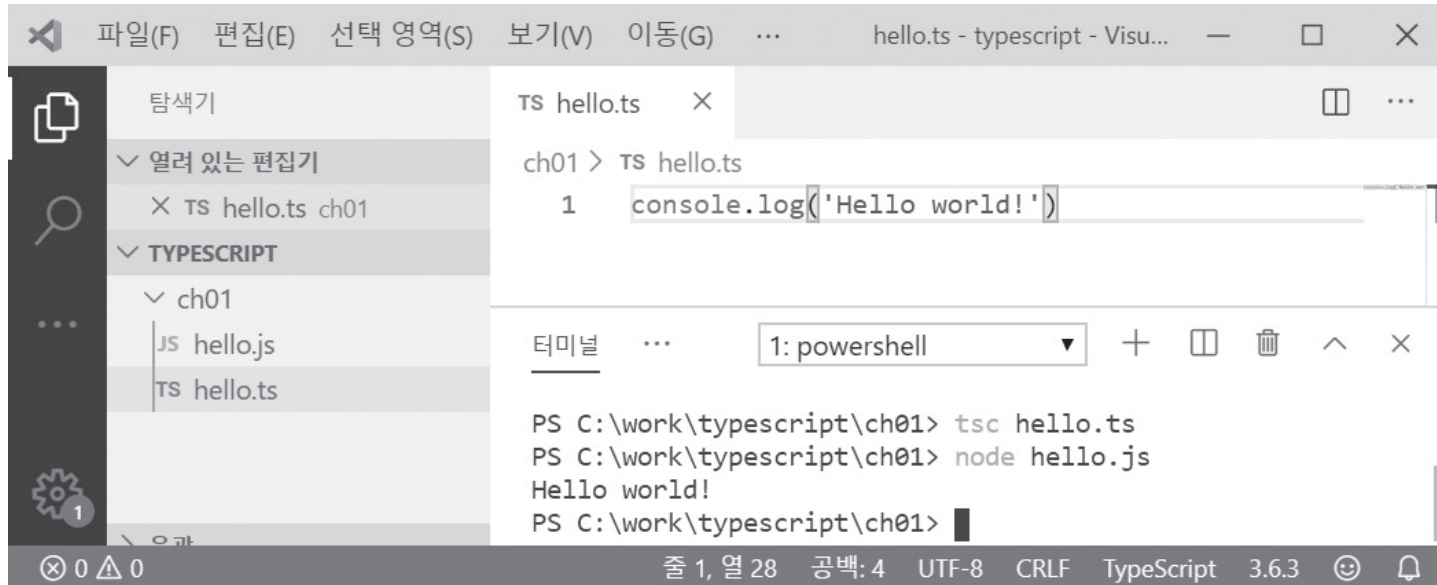
01-3 타입스크립트 개발 환경 만들기

vscode 편집기에서 hello.ts 파일에 코드 쓰고 컴파일하기



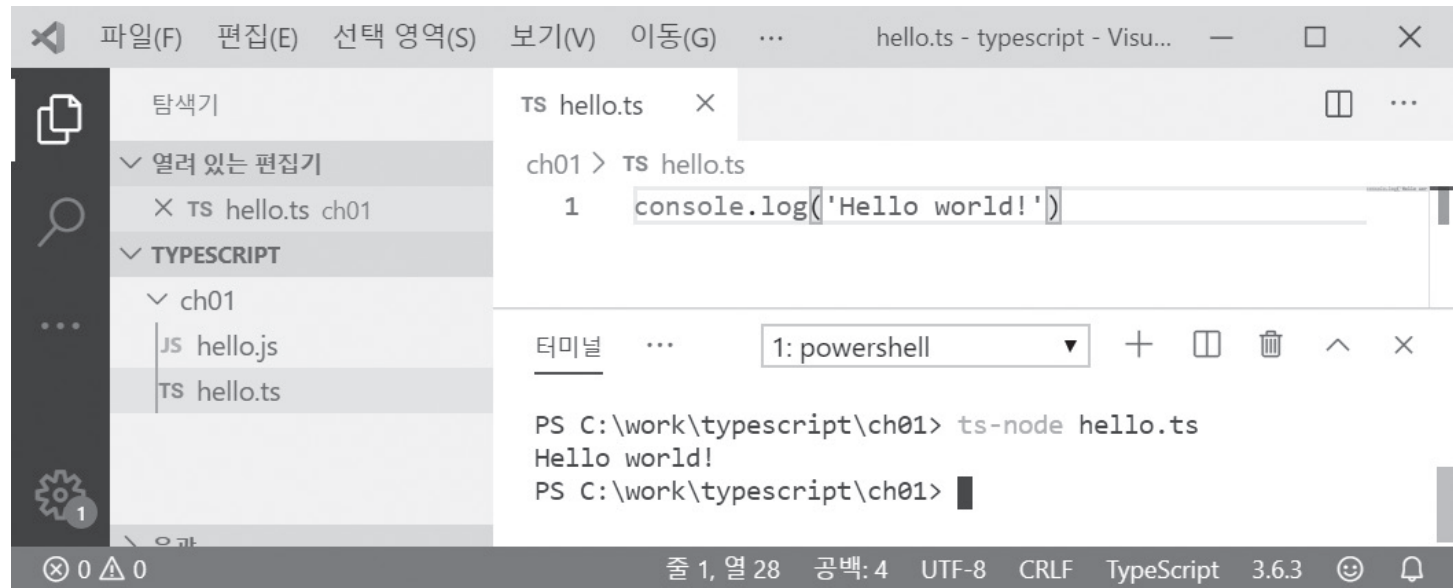
01-3 타입스크립트 개발 환경 만들기

컴파일되어 만들어진 **hello.js** 파일을 **node**로 실행하기



01-3 타입스크립트 개발 환경 만들기

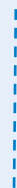
ts-node로 컴파일과 실행을 한꺼번에 하기





감사합니다

저는 타입스크립트를 좋아합니다. 최고죠.



라이언 달(Ryan Dahl, Node.js 과 Deno 창시자)