세상의 속도를
따라잡고 싶다면

Do it!

기초부터 함수형 코드까지 챙기는 실속 코스!

타입스크립트 프로그래밍

10만 건의 빅데이터 배치 프로그램 만들기
Node.js + MongoDB + Express + React.js를 활용한 리액트 웹 애플리케이션 만들기

전예홍 지음

TS

이지스퍼블리싱

자바스크립트 기술 스택을 끌어올리는 타입스크립트와 함수형 프로그래밍을 배운다!

전예홍 yehongj2020@gmail.com

연세대학교 전자공학과를 졸업하고 카이스트 대학원에서 전자공학 석사를 받았다. 1990년부터 지금까지 30년 동안 프로그램을 개발해 왔다. 한국 마이크로소프트에서 수석 개발 컨설턴트 리드(ADC Lead)로 근무했으며 Node.js 환경에서 동작하는 프로그램 개발에 능통하다. 특히 리액트와 리액트 네이티브, 앵귤러 프레임워크의 전문가다. 그동안 개발한 제품으로 대통령상을 받았고 정보통신부 장관상을 3회 받았다. 아시아 100대 보안 기업 제품 가운데 1등으로 뽑혀 말레이시아 국왕이 주는 대상을 받기도 했다.

Do it!

기초부터 함수형 코드까지 챙기는 실속 코스!

타입스크립트 프로그래밍

전자책 1쇄 발행 • 2020년 3월 3일 (종이책 초판 1쇄 기준)

지은이 • 전예홍
펴낸이 • 이지연
펴낸곳 • 이지스퍼블리싱(주)
출판사 등록번호 • 제313-2010-123호
주소 • 서울특별시 마포구 잔다리로 109 이지스빌딩 4층(우편번호 04003)
대표전화 • 02-325-1722 / 팩스 • 02-326-1723
홈페이지 • www.easyspub.co.kr / 이메일 • service@easyspub.co.kr
기획 및 책임 편집 • 이인호 / 표지 및 내지 디자인 • 트인글터 / 본문 전산편집 • 트인글터
교정교열 • 박명희 / 마케팅 • 박정현 / 인쇄 및 제본 • 보광문화사
영업 • 이주동(nlrose@easyspub.co.kr)

전자책 ISBN 979-11-6303-149-9 15000
전자책 가격 16,250원 (종이책 가격 25,000원)

저는 타입스크립트를 좋아합니다. 최고죠.

라이언 달(Ryan Dahl, Node.js 창시자)

타입스크립트는 C#처럼 우수한 개발 경험을 제공함으로써
팀의 지식을 재사용하고 개발 속도를 유지하는 데
도움이 되었습니다.
자바스크립트보다 훨씬 향상된 기능을 제공합니다.

발리오 스토이체프(Valio Stoychev, NativeScript PM 리드)

타입스크립트를 사용한 후 내 인생의 품질이 달라졌어요.
타입스크립트는 프런트엔드,
백엔드 개발자 모두에게 필요합니다.

니꼴라스 세라노 아레발로(Nicolás Serrano Arévalo , 노마드 코더 운영자)

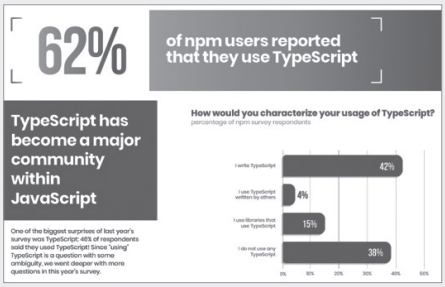
대규모 프로젝트를 진행하는 기업에서 타입스크립트 개발자를 찾고 있어요

프런트엔드, 백엔드 개발자가 연봉을 올릴 수 있는 가장 확실한 방법, 타입스크립트를 시작하세요!

타입스크립트는 2012년 등장했습니다. 마이크로소프트가 자바스크립트의 문법을 그대로 수용하면서 그에 없는 선택적 정적 타이핑, 클래스 선언, 모듈 지원 등 기능을 추가한 형태로 선보였습니다. 그리고 5년 후 구글은 타입스크립트를 사내 프런트엔드 표준 언어로 승인했습니다. 구글이 대규모 웹 애플리케이션 개발 프로젝트에 자바스크립트 이상의 생산성을 보장하는 프로그래밍 언어로 타입스크립트를 인정했다는 뜻입니다.

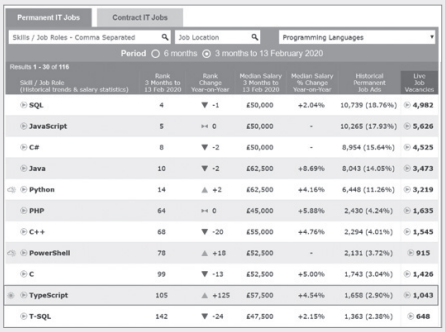
자바스크립트는 코드에 명시적으로 타입을 기술할 수 없으므로 타이핑 실수를 코드 작성 단계에서 찾아내기 어렵고, 다른 사람이 작성한 코드를 어떻게 사용해야 하는지 파악하기도 쉽지 않습니다. 하지만 타입스크립트는 코드에 타입을 명시적으로 기술할 수 있으므로 코드 작성자와 사용자가 다를 때 생기는 혼란을 줄일 수 있습니다.

타입스크립트의 이런 장점 때문에 오늘날 많은 오픈 소스 라이브러리가 타입스크립트로 작성되고 있습니다. 특히 여러 사람이 참여하는 기업용 프로젝트는 대부분 타입스크립트를 사용하고 있습니다. 프로그래밍 언어 인기 순위를 발표하는 사이트에서 타입스크립트는 매년 상위권에 올라 있으며, 각종 취업 포털에서도 자바스크립트 개발자보다 더 많은 연봉을 받는 것으로 조사되고 있습니다.



자바스크립트 개발자 62%가 타입스크립트를 사용한다는 조사 결과(출처: 「Enterprise JavaScript in 2019」 보고서)

이 책은 자바스크립트 개발자가 타입스크립트의 핵심을 빠르게 학습할 수 있게 설계되었습니다. ES6+ 자바스크립트의 문법과 기능을 타입스크립트로는 어떻게 작성하는지 알려줍니다. 또 타입스크립트만의 고유한 문법과 기능도 알려줍니다. 이 책에서 제시하는 코드는 단순히 문법을 소개하는 수준을 넘어 타입스크립트의 장점을 최대한 살렸습니다. 간결하고 구조적이며 읽기가 쉽습니다. 따라서 코드의 안전성과 성능, 유지보수성을 고려하는 실무에서 바로 적용할 수 있습니다.



타입스크립트 개발자 평균 연봉이 자바스크립트보다 1,100만원가량 더 높다는 조사 결과(출처: itjobswatch.co.uk)

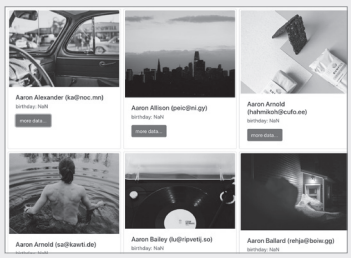
이제 자바스크립트는 익숙하잖아요?
다음은 타입스크립트로 함수형 프로그래밍까지 도전해 보세요!

함수형 설계 방식으로 작성한 코드는 객체지향 방식의 코드보다 간결하고 논리 정연합니다. 함수형 프로그래밍에서는 불변성이라는 원칙에 따라 값이 변하는 것을 최대한 배제하므로 프로그램 검증과 최적화, 그리고 동시에 여러 스레드에서 문제없이 동작하는 프로그램을 쉽게 작성할 수 있습니다. 또한, 함수를 하나의 값처럼 다룰 수 있어서 재사용이 수월하고, 값을 미리 계산하지 않고 꼭 필요할 때만 계산하므로 메모리 절약과 프로그램의 성능에도 긍정적인 영향을 줍니다. 따라서 코드를 함수형 프로그래밍으로 작성하면 전체 개발 시간을 줄이고 유지보수를 쉽게 해서 프로젝트의 생산성을 높일 수 있습니다.

이 책은 선언형 프로그래밍, 함수 조합, 제네릭, 모나드 등 네 가지 방식의 함수형 프로그래밍을 타입스크립트로 구현하는 방법을 알려줍니다. 독자는 이 책을 통해 타입스크립트와 함수형 프로그래밍의 여러 가지 개념과 기법을 배울 수 있고, 자바스크립트 생태계에서 가장 미래가 밝은 기술로 인정받는 타입스크립트와 함수형 프로그래밍을 자신의 기술 스택에 장착할 수 있습니다.

빅데이터 배치 프로그램을 만들어 보세요!
데이터베이스와 API 서버를 구축하고 웹 애플리케이션까지 개발해 보세요!

이 책의 마지막 장에서는 타입스크립트가 실무에 어떻게 적용되는지를 보여줍니다. 사실 지금까지 Node.js 환경에서 엄청난 분량의 데이터를 다루는 배치 프로그램을 만드는 것은 어려웠지만, 이 책의 코드를 사용하면 빅데이터를 처리하는 배치 프로그램을 매우 쉽게 작성할 수 있습니다. 책에서는 10만 건의 데이터만을 대상으로 하지만 데이터가 1,000만 건, 1억 건이 되더라도 최소한의 메모리만 소비하면서 동작합니다. 그리고 이 데이터를 몽고DB에 저장한 후 익스프레스 API 서버와 리액트를 이용해 서비스하는 과정까지 실습합니다.



이 책에서 만드는 웹 애플리케이션 모습

고마운 사람들

오랜 집필 기간 동안 포기하지 않고 마침내 결실을 볼 수 있게 해주신 이지스퍼블리싱 출판사의 이지연 대표님과 이인호 편집자님, 그리고 직원분들께 감사의 마음을 보냅니다. 또한 저와 많은 프로젝트를 함께 진행하면서 동료와 함께 개발하는 즐거움을 느끼게 해준 김성경 님께도 감사의 마음을 전합니다.

전예홍 드림(yehongj2020@gmail.com)

타입스크립트 기본기부터 탄탄하게!

01장
•
타입스크립트와 개발 환경 만들기

타입스크립트를 소개하고 이 책에서 다루는 핵심 문법을 요약해서 보여줍니다. 그리고 타입스크립트와 Node.js, 비주얼스튜디오 코드 등을 설치해 실습 환경을 구성합니다. 윈도우에서 각종 프로그램을 설치할 때 복잡성을 줄이고 버전 관리를 쉽게 하고자 윈도우용 커맨드 기반의 패키지 관리자인 scoop을 이용합니다.

02장
•
타입스크립트 프로젝트 생성과 관리

타입스크립트로 첫 번째 프로젝트를 만들어 봅니다. 그러면서 모듈이라는 개념을 살펴보고 import와 export로 모듈을 활용하면서 프로젝트를 구성하는 방법을 익힙니다. 그리고 타입스크립트를 처음 시작할 때 장애물 중 하나로 알려진 tsconfig.json 파일을 뜯어봅니다.

03장
•
객체와 타입

자바스크립트는 타입이 없는 객체를 만들 수 있지만, C++나 Java, Python과 같은 전통적인 객체지향 언어와 비교해 독특한 특징이 있습니다. 3장에서는 자바스크립트 관점에서 객체지향 프로그래밍의 특징을 살펴보고, 타입스크립트에서 interface 키워드를 이용해 객체를 구현하는 방법을 살펴봅니다.

04장
•
함수와 메서드

놀랍게도 함수형 언어에서 함수는 객체입니다. 다른 언어를 경험한 독자라면 이 부분이 낯설 텐데요. 04장에서는 함수형 프로그래밍의 핵심인 객체로서의 함수, 즉 '일등 함수'를 다룹니다.

05장
•
배열과 튜플

함수형 프로그래밍은 명령형 프로그래밍 대신 선언형 프로그래밍 방식을 사용합니다. 선언형 프로그래밍의 핵심은 데이터를 배열로 만들고 map이나 filter, reduce와 같은 메서드를 통해 가공하는 것입니다. 05장에서는 함수형 프로그래밍 관점에서 배열을 다루는 방법을 '순수 함수' 방식과 함께 다룹니다.

06장
•
반복기와 생성기

for...in, for...of와 같은 구문은 모두 반복기와 반복기 제공자, 생성기 구문과 결합된 형태로 동작합니다. 06장에서는 이런 방식의 동작 원리를 설명합니다.

함수형 프로그래밍 기본기를 다지고, 10만 건의 빅데이터를 활용한 리액트 웹 애플리케이션 개발까지!

07장
•
Promise와 async/await 구문

동기/비동기 방식의 특징을 알아보고 비동기 방식으로 구현하기 위해 타입스크립트의 Promise 객체를 async/await 구문으로 어떻게 다루는지 설명합니다.

08장
•
함수 조합의 원리와 응용

함수형 프로그래밍에서 자주 등장하는 함수 조합을 다룹니다. 함수 조합이 가능하려면 프로그래밍 언어 문법이 일등 함수 기능을 제공해야 합니다. 그리고 일등 함수 기능은 고차 함수와 클로저, 그리고 커리와 같은 또 다른 문법 기능을 제공해야 동작합니다. 08장은 이런 내용을 설명합니다.

09장
•
람다 라이브러리

lodash는 자바스크립트 분야에서 널리 사용되는 유틸리티 패키지입니다. 그런데 lodash는 함수 조합 방식으로 사용하는 데 불편함이 있어서 주로 ramda 패키지를 사용합니다. 09장은 ramda 패키지가 제공하는 다양한 기능을 소개합니다.

10장
•
제네릭 프로그래밍

제네릭은 타입에 무관한 클래스나 함수를 만들 때 사용합니다. 제네릭은 보통 객체지향 언어의 기능으로 인식되지만 함수형 프로그래밍을 할 때도 사용됩니다. 10장은 함수형 프로그래밍 관점에서 제네릭을 설명합니다.

11장
•
모나드

타입스크립트 관점에서 모나드를 어떻게 설계하고 이용하는지 설명합니다. 비교적 널리 알려진 Identity, Maybe, Validation, IO 모나드를 직접 제작해 가는 방식으로 설명합니다.

12장
•
타입스크립트 함수형 프로그래밍 실습

타입스크립트가 실무에서 어떻게 활용될 수 있는지 구체적으로 알아봅니다. 생성기 문법을 사용해 데이터를 10만 개 생성한 후 이를 CSV 파일 포맷으로 저장하고, 이 파일을 읽어서 몽고DB에 저장합니다. 그다음 API 서버를 만들고 이를 리액트로 동작하는 웹 애플리케이션을 제작하는 실습을 진행합니다.

궁금한 내용은 저자에게 질문해 보세요

책을 읽다가 궁금한 내용이 있으면 다음 메일 주소로 저자에게 질문해 보세요. 몇 쪽에서 어떤 점이 궁금한지 자세히 적어야 정확하고 빠른 답변을 받을 수 있습니다

- **저자 메일 주소**: yehongj2020@gmail.com

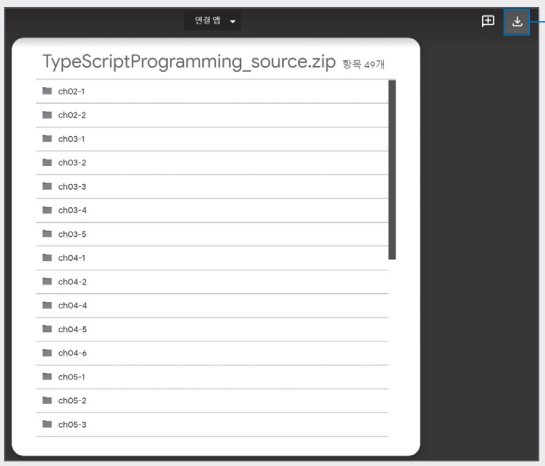
코드를 입력할 때 참고해 보세요 — 전체 소스 무료 공개

내가 입력한 코드가 잘 입력되었는지 확인하고 싶다면 이지스퍼블리싱 자료실에서 제공하는 전체 소스를 비교하며 공부해 보세요.

- 이지스퍼블리싱 공식 홈페이지(easyspub.co.kr)에 회원가입하여 [자료실]에서 'Do it! 타입스크립트 프로그래밍'을 검색해 보세요.

❶ Do it! 타입스크립트 프로그래밍 검색하기

❷ [예제 파일 다운받기] 클릭



❸ 다운로드 아이콘 클릭

책을 통해 스스로 발전하는 지적인 독자를 만나보세요 — Do it! 스터디룸 카페

같은 고민을 하는 친구들과 함께 공부해 보세요. 내가 잘 이해한 내용은 남을 도와주고 내가 잘 이해하지 못한 내용은 도움을 받아가며 공부하면 복습 효과도 누릴 수 있습니다. 서로서로 코드와 개념 리뷰를 하며 훌륭한 개발자로 성장해 보세요 (회원 가입과 등업은 필수).

https://cafe.naver.com/doitstudyroom

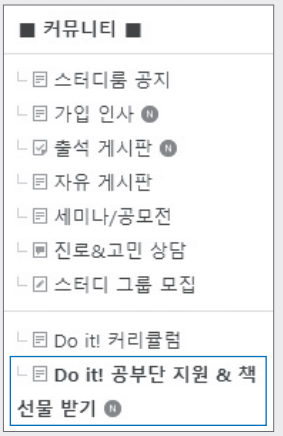
스터디 노트도 쓰고 책 선물도 받고! — Do it! 공부단 상시 모집 중

혼자 공부하다 보면 계획을 세워도 잘 지켜지지 않죠? Do it! 스터디룸에서 운영하는 공부단에 지원해 보세요! 공부단에서는 자기 목표를 공유하고 매일 공부한 내용을 스터디 노트로 작성합니다. 꾸준히 공부할 수 있어 목표 달성하기가 훨씬 수월하겠죠? 스터디 노트를 쓰며 책을 완독하면 원하는 책 한 권을 선물로 드립니다. 공부단 지원이나 진행에 관한 자세한 방법은 다음 설명을 참고해 주세요.

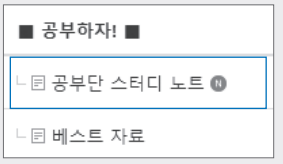
'Do it! 스터디룸' 카페 회원 가입 필수, 회원 등급 '두잇 독자(게시글 1, 댓글 10, 출석 2회)' 부터 이용 가능!

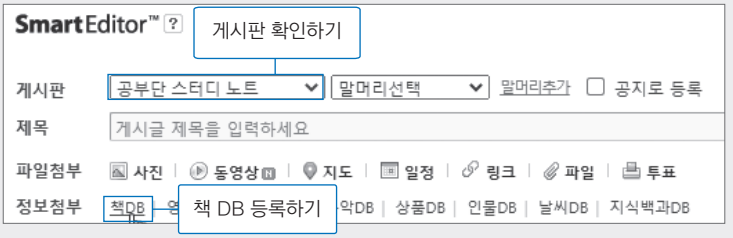
1. Do it! 스터디룸 카페에 방문하면 '■커뮤니티■' 메뉴에 'Do it! 공부단 지원 & 책 선물 받기' 게시판이 있습니다. 게시판에 입장하여 [글쓰기]를 누른 다음 공부단 지원 글 양식에 맞춰 작성해 주세요. 자세한 방법은 아래 링크를 참고하세요.

https://cafe.naver.com/doitstudyroom/13482



2. 스터디 노트는 '■공부하자!■' 메뉴의 '공부단 스터디 노트' 게시판을 이용해 주세요. 이때 글 맨 위에 '책 DB'를 등록해 주세요.





강의 진도표

도전! 30일 완성

날짜와 공부한 범위를 진도표에 기록하여 계획에 맞게 공부했는지 확인해 보세요.

월 일	월 일	월 일	월 일	월 일
공부한 범위 (13 ~)	공부한 범위 (~)	공부한 범위 (~)	공부한 범위 (~)	공부한 범위 (~)
월 일	월 일	월 일	월 일	월 일
공부한 범위 (~)	공부한 범위 (~)	공부한 범위 (~)	공부한 범위 (~)	공부한 범위 (~)
월 일	월 일	월 일	월 일	월 일
공부한 범위 (~)	공부한 범위 (~)	공부한 범위 (~)	공부한 범위 (~)	공부한 범위 (~)
월 일	월 일	월 일	월 일	월 일
공부한 범위 (~)	공부한 범위 (~)	공부한 범위 (~)	공부한 범위 (~)	공부한 범위 (~)
월 일	월 일	월 일	월 일	월 일
공부한 범위 (~)	공부한 범위 (~)	공부한 범위 (~)	공부한 범위 (~)	공부한 범위 (~)
월 일	월 일	월 일	월 일	월 일
공부한 범위 (~)	공부한 범위 (~)	공부한 범위 (~)	공부한 범위 (~)	공부한 범위 (~ 402)

01

타입스크립트와 개발 환경 만들기

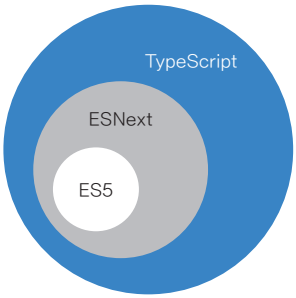
이 장에서는 타입스크립트에 관해 알아보고, 타입스크립트 개발 관련 프로그램 설치와 사용 방법을 살펴보겠습니다.

01-1 타입스크립트란 무엇인가?

세 종류의 자바스크립트

자바스크립트는 현재 세 가지 종류가 있습니다. 웹 브라우저에서 동작하는 표준 자바스크립트인 ES5(ECMAScript 5)와 2015년부터 매년 새로운 버전을 발표하는 ESNext, 그리고 ESNext에 타입(type) 기능을 추가한 타입스크립트(TypeScript)입니다.

다음 그림은 ES5, ESNext, 타입스크립트 간의 관계를 보여줍니다.



ES5와 ESNext, 타입스크립트의 관계도

ESNext는 ES5의 모든 문법을 포함하고, 타입스크립트는 ESNext의 모든 문법을 포함합니다. 따라서 타입스크립트로 개발했더라도 타입 기능을 사용하지 않는다면 ESNext 소스나 마찬가지입니다.

ESNext 자바스크립트란?

자바스크립트의 공식 표준은 ECMAScript(줄여서 ES)입니다. 2009년 발표된 ES5 버전이 있었는데 2015년에 발표된 ES6 버전에서 큰 변화가 있었습니다. 그래서 ES6 이후 버전을 통틀어 가리킬 때는 '새로운 자바스크립트'라는 뜻에서 'ESNext'라고 합니다.
그리고 2015년에 ECMAScript 공식 버전 표기법이 바뀌었습니다. ES6부터는 발표 연도를 붙여 'ECMAScript 2015(줄여서 ES2015)'처럼 부르기로 했습니다. 또한, 1년 주기로 새로운 버전을 발표하기로 해서 2019년 말 현재 ECMAScript 2019까지 나왔습니다. 따라서 이 책에서 ESNext라고 하면 ECMAScript 2015~ECMAScript 2019까지를 의미합니다.

타입스크립트는 누가 만들었나?

타입스크립트는 마이크로소프트가 개발하고 유지하고 있는 오픈소스 프로그래밍 언어로 2012년 말 처음 발표되었습니다. 타입스크립트는 C# 언어를 창시한 아네르스 하일스베르(Anders Hejlsberg)가 핵심 개발자로 참여하고 있습니다. 구글의 Angular.js 팀이 앵귤러(Angular) 버전 2를 만들면서 타입스크립트를 채택한 이후부터 널리 알려졌습니다. 요즘은 앵귤러의 경쟁 프레임워크인 리액트(React.js)나 뷰(Vue.js)조차도 타입스크립트를 사용해 개발되고 있습니다.

자바스크립트에 타입 기능이 있으면 좋은 이유

오늘날 소프트웨어는 상당히 복잡하므로 보통 여러 사람이나 팀이 협력해 하나의 제품을 개발합니다. 그런데 이런 상황에서는 항상 코드를 작성한 쪽과 사용하는 쪽 사이에 커뮤니케이션이 중요합니다. 예를 들어, A라는 개발자가 다음과 같은 코드를 만들었다고 가정해 봅시다.

```
function makePerson(name, age) {}
```

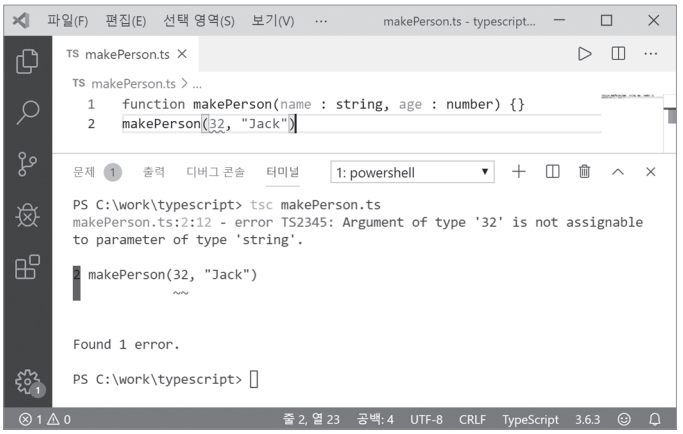
B라는 개발자가 이 코드를 이용하려고 다음 코드를 만들어 실행했을 때 오류가 발생했다면, B 개발자는 오류의 원인이 무엇인지 찾기가 어렵습니다.

```
makePerson(32, "Jack")
```

그런데 처음 코드를 다음처럼 타입스크립트의 타입 기능을 이용해 구현했다면 이러한 문제는 발생하지 않았을 것입니다.

```
function makePerson(name: string, age: number) {}
```

그리고 타입스크립트 컴파일러는 다음 화면처럼 문제의 원인이 어디에 있는지 친절하게 알려주므로 코드를 좀 더 수월하게 작성할 수 있습니다.



오류의 원인을 알려주는 타입스크립트 컴파일러

이 때문에 개발자들은 대규모 소프트웨어를 개발할 때 자바스크립트보다 타입스크립트를 선호하게 되었습니다.

트랜스파일

ESNext 자바스크립트 소스코드는 바벨(Babel)이라는 트랜스파일러(transpiler)를 거치면 ES5 자바스크립트 코드로 변환됩니다. 바벨과 유사하게 타입스크립트 소스코드는 TSC(TypeScript compiler)라는 트랜스파일러를 통해 ES5 자바스크립트 코드로 변환됩니다.

여기서 트랜스파일러란, 어떤 프로그래밍 언어로 작성된 소스코드를 또 다른 프로그래밍 언어로 된 소스코드로 바꿔주는 프로그램을 말합니다. 트랜스파일러는 텍스트로 된 소스코드를 바이너리 코드로 바꿔주는 컴파일러(compiler)와 구분하기 위해 생긴 용어입니다.

트랜스파일과 컴파일은 '무엇인가를 또 다른 무엇으로 바꿔준다'는 관점에서는 크게 차이가 없으므로, 요즘은 둘을 구분하지 않는 경향이 있습니다. 따라서 이 책에서는 트랜스파일이라는 용어보다는 컴파일이라는 용어를 주로 사용합니다.

01-2 타입스크립트 주요 문법 살펴보기

지금까지 타입스크립트가 무엇인지 간략하게 알아보았습니다. 이제부터는 타입스크립트의 주요 문법을 살펴보겠습니다. 여기서 소개하는 문법은 뒤에서 더 자세하게 다룰 예정이므로 지금은 가볍게 읽고 넘어가도 괜찮습니다.

타입스크립트는 ESNext 문법을 대부분 지원하므로 타입스크립트를 다루려면 ESNext 문법을 알아야 합니다. 그리고 타입스크립트에만 고유한 문법도 있습니다. 두 가지 문법을 구분해서 알아봅시다.

ESNext의 주요 문법 살펴보기

(1) 비구조화 할당

ESNext는 '비구조화 할당(destructuring assignment)'이라고 하는 구문을 제공합니다. 비구조화 할당은 객체와 배열에 적용할 수 있습니다. 다음 코드는 비구조화 할당을 적용한 예입니다.

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
```

02행을 살펴보면 person 객체의 name과 age 속성을 비구조화 할당을 통해 쉽게 각 멤버를 얻습니다. 05행은 배열에 비구조화 할당을 적용한 예입니다. 배열에서 첫 번째 요소와 나머지 요소를 비구조화 할당을 통해 쉽게 분리해서 얻습니다. 08행은 두 변수 a와 b의 값을 서로 교환(swap)하는 예입니다.

비구조화 할당은 03장에서 자세하게 다룹니다.

(2) 화살표 함수

자바스크립트에서 함수를 선언할 때는 다음 코드의 01행처럼 function 키워드를 사용합니다. 그런데 ESNext에서는 function 키워드 외에도 화살표(=>)로 함수를 선언할 수 있습니다. 02행에서는 화살표로 add2 함수를 만들었는데, 이처럼 화살표로 만든 함수를 '화살표 함수(arrow function)'라고 합니다.

화살표 함수 예

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

코드는 될 수 있으면 간결하게 작성해야 읽기 좋습니다. 화살표 함수를 사용하면 function 키워드 방식보다 코드를 간결하게 만들 수 있습니다. 참고로 이 책의 코드는 대부분 화살표 함수 형태로 구현되었습니다.

화살표 함수는 04장에서 자세하게 다룹니다.

(3) 클래스

ESNext에서는 클래스라는 기능을 제공해 C++나 Java 언어에서 보던 객체지향 프로그래밍을 지원합니다. 객체지향 프로그래밍은 프로그래밍 언어가 '캡슐화(encapsulation)'와 '상속(inheritance)', '다형성(polymorphism)'이라는 세 가지 요소를 지원합니다. 다음 코드는 객체지향 프로그래밍의 세 가지 요소를 모두 보여줍니다.

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"]
```

이 코드는 C++나 자바와 같은 언어로 객체지향 프로그래밍을 경험해 본 독자라면 낯설지 않을 것입니다. ES5 자바스크립트로는 이러한 형태로 코드를 작성하지 못합니다.

클래스는 03장에서 자세하게 다룹니다.

(4) 모듈

모듈을 사용하면 코드를 여러 개 파일로 분할해서 작성할 수 있습니다. 변수나 함수, 클래스 등에 `export` 키워드를 사용해 모듈로 만들면 다른 파일에서도 사용할 수 있습니다. 그리고 이렇게 만든 모듈을 가져오고 싶을 때는 `import` 키워드를 사용합니다. 이처럼 ES5와 다르게 ESNext는 파일을 분할해서 구현할 수 있게 해주는 모듈 기능을 제공합니다.

모듈은 02장에서 자세하게 다룹니다.

모듈 예

```
01: import * as fs from 'fs'
02: export function writeFile(filepath: string, content: any) { }
```

(5) 생성기

타입스크립트는 물론 파이썬이나 PHP와 같은 몇몇 프로그래밍 언어는 `yield`라는 특별한 키워드를 제공합니다. `yield` 문은 '반복자'를 의미하는 반복기(iterator)를 생성할 때 사용합니다. 그런데 반복기는 독립적으로 존재하지 않고 반복기 제공자(iterable)를 통해 얻습니다. 이처럼 `yield` 문을 이용해 반복기를 만들어 내는 반복기 제공자를 '생성기(generator)'라고 부릅니다.

생성기는 `function` 키워드에 별표(*)를 결합한 `function*`과 `yield` 키워드를 이용해 만듭니다. 타입스크립트에서 `yield`는 반드시 `function*`으로 만들어진 함수 내부에서만 사용할 수 있습니다.

생성기 예

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

코드에서 01행의 function*을 생성기라고 합니다. 이 생성기 덕분에 02행에서 yield라는 키워드를 사용할 수 있습니다. 코드에서 yield가 호출되면 프로그램 실행이 02행에서 일시 정지한 후 점프해서 04행을 실행합니다. 그리고 04행 실행을 마치면 다시 02행을 실행하고, 이 과정을 02행의 배열 [1, 2]의 요소를 모두 순회할 때까지 반복합니다. 이처럼 yield는 특별한 실행 흐름을 보여줍니다.

생성기는 06장에서 자세하게 다룹니다.

(6) Promise와 async/await 구문

ES5로 비동기 콜백 함수(asynchronous callback function)를 구현하려면 코드가 상당히 복잡하고 번거로워집니다. 이 때문에 '콜백 지옥(callback hell)'이라는 표현이 있을 정도입니다. Promise는 웹 브라우저와 노드제이에스(Node.js)에서 모두 제공하는 기본 타입으로 비동기 콜백 함수를 상대적으로 쉽게 구현할 목적으로 만들어졌습니다. ESNext는 C# 4.5 버전의 async/await 구문을 빌려서 여러 개의 Promise 호출을 결합한 좀 더 복잡한 형태의 코드를 간결하게 구현할 수 있게 합니다.

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]
```

코드에서 01행의 함수는 async 함수 수정자(function modifier)를 사용합니다. async를 사용한 함수는 본문에서 await 키워드를 사용할 수 있습니다. await는 Promise 객체를 해소(resolve)해 줍니다. 따라서 아래 get 함수는 [1, 2, 3] 값을 Promise 형태로 반환합니다. get이 반환한 Promise 객체는 then 메서드를 호출해 실제 값을 얻을 수 있습니다.

Promise와 async/await 구문은 07장에서 자세하게 다룹니다.

타입스크립트 고유의 문법 살펴보기

(1) 타입 주석과 타입 추론

다음 코드에서 01행의 변수 n 뒤에는 콜론(:)과 타입 이름이 있습니다. 이것을 '타입 주석(type annotation)'이라고 합니다.

타입 주석과 타입 추론 예

```
01: let n: number = 1
02: let m = 2
```

그런데 타입스크립트는 02행처럼 타입 부분을 생략할 수도 있습니다. 타입스크립트는 변수의 타입 부분이 생략되면 대입 연산자(=)의 오른쪽 값을 분석해 왼쪽 변수의 타입을 결정합니다. 이를 '타입 추론(type inference)'이라고 합니다. 타입스크립트의 타입 추론 기능은 자바스크립트 소스코드와 호환성을 보장하는 데 큰 역할을 합니다. 타입 추론 덕분에 자바스크립트로 작성된 '.js' 파일을 확장자만 '.ts'로 바꾸면 타입스크립트 환경에서도 바로 동작합니다.

(2) 인터페이스

다음 코드의 인터페이스 구문은 타입이 있는 언어를 경험해 본 독자라면 낯설지 않을 것입니다.

인터페이스 구문은 03장에서 자세하게 다룹니다.

인터페이스 예

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

(3) 튜플

파이썬과 같은 몇몇 프로그래밍 언어에는 튜플(tuple)이라는 타입이 있습니다. 튜플은 물리적으로는 배열과 같습니다. 다만, 배열에 저장되는 아이템의 데이터 타입이 모두 같으면 배열, 다르면 튜플입니다.

튜플은 05장에서 자세하게 다룹니다.

배열과 튜플

```
01: let numberArray: number[ ] = [1, 2, 3]  // 배열
02: let tuple: [boolean, number, string] = [true, 1, 'Ok']  // 튜플
```

(4) 제네릭 타입

제네릭 타입은 다양한 타입을 한꺼번에 취급할 수 있게 해줍니다. 다음 코드에서 Container 클래스는 value 속성을 포함합니다. 이 클래스는 Container<number>, Container<string>, Container<number[]>, Container<boolean>처럼 여러 가지 타입을 대상으로 동작할 수 있는데 이를 '제네릭 타입(generic type)'이라고 합니다.

제네릭 타입은 05장, 08장, 10장에서 자세하게 다룹니다.

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: Containenr<string> = new Container<string>('Hello world')
```

(5) 대수 타입

ADT란, 추상 데이터 타입(abstract data type)을 의미하기도 하지만 대수 타입(algebraic data type)이라는 의미로도 사용됩니다. 대수 타입이란, 다른 자료형의 값을 가지는 자료형을 의미합니다. 대수 타입에는 크게 합집합 타입(union 또는 sum type)과 교집합 타입(intersection 또는 product type) 두 가지가 있습니다. 합집합 타입은 '|' 기호를, 교집합 타입은 '&' 기호를 사용해 다음 코드처럼 여러 타입을 결합해서 만들 수 있습니다.

대수 타입은 10장에서 자세하게 다룹니다.

대수 타입 예

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

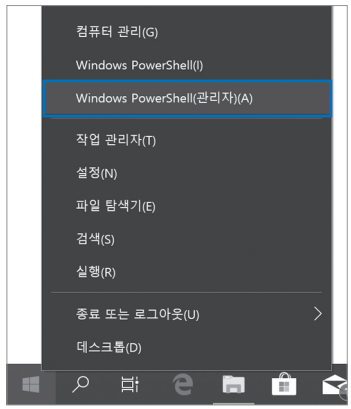
01-3 타입스크립트 개발 환경 만들기

scoop 프로그램 설치

타입스크립트 개발 환경은 노드제이에스 개발 환경과 똑같습니다. 즉, 노드제이에스를 설치하고 비주얼 스튜디오 코드와 크롬 브라우저를 설치하면 바로 개발할 수 있습니다. 이 프로그램들은 각 웹 사이트에 접속해서 내려받아 설치해야 합니다. 그런데 이렇게 설치하면 프로그램을 판올림할 때 같은 과정을 반복해야 합니다. 따라서 이 책은 scoop이라는 설치 프로그램을 사용하겠습니다. scoop으로 설치한 프로그램들은 `scoop update *` 명령으로 한꺼번에 가장 최신 버전으로 업데이트됩니다.

① 윈도우 파워셸 실행

먼저, 윈도우 10 바탕화면에서 왼쪽 아래에 있는 시작 버튼을 마우스 오른쪽으로 클릭하고 [Window PowerShell(관리자)] 메뉴를 선택합니다.



윈도우 파워셸 실행

② scoop 설치

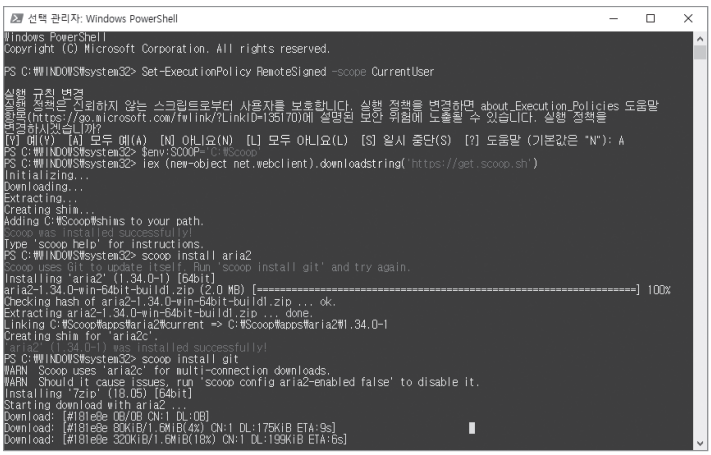
관리자 권한으로 실행한 윈도우 파워셸에서 다음과 같은 명령을 차례로 실행합니다.

```
> 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

첫 번째 명령은 세번 째 명령이 정상적으로 동작하도록 윈도우 실행 규칙을 변경합니다. 이때 실행 정책 변경을 묻는 질문에 A를 입력합니다. 두 번째 명령은 앞으로 scoop을 이용해 설치하는 모든 프로그램의 경로를 'C:\Scoop'으로 설정합니다. 세 번째 명령은 scoop을 설치합니다.

네 번째 명령은 scoop을 이용해 aria2를 설치합니다. aria2를 설치해 놓으면 scoop이 다중 내려받기를 할 수 있어서 프로그램 설치 시간이 절감됩니다. 마지막 명령은 git을 설치합니다. scoop은 내부적으로 다양한 설치 관련 정보를 깃허브(GitHub)라는 저장소에서 얻습니다.

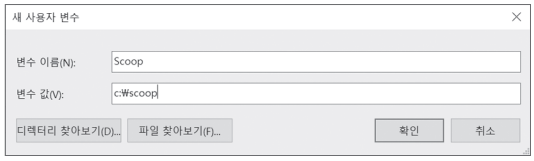


윈도우 파워셸에서 scoop 설치 명령 실행

모든 것이 정상적으로 설치되었다면 C:\Scoop 디렉터리가 만들어지고, 그 안에 몇 가지 하위 디렉터리가 만들어집니다. 그중 apps 디렉터리에 들어가 보면 앞서 설치한 프로그램의 이름으로 된 디렉터리가 있습니다. 만일, 특정 디렉터리를 지우면 사실상 해당 디렉터리의 프로그램이 제거(uninstall)됩니다. 또한, 관리자 모드 파워셸에서 `scoop update *` 명령을 실행하면 지금까지 설치한 모든 프로그램을 대상으로 한꺼번에 최신 버전으로 판올림할 수 있습니다.

윈도우 환경 변수 설정

`scoop`으로 프로그램을 설치할 때 앞에서는 `$env:Scoop` 명령으로 설치 경로를 지정했습니다. 만약, 프로그램을 설치할 때마다 경로를 지정하는 것이 번거롭다면, 윈도우 환경 변수에 설정해 주면 됩니다. 윈도우 검색에서 '시스템 환경 변수 편집'이라는 키워드로 찾으면 환경 변수를 설정할 수 있습니다. 새 사용자 변수 창에서 '변수 이름'에 "Scoop"을 입력하고, '변수 값'에 경로를 입력합니다. 이로써 Scoop이라는 이름의 환경 변수가 윈도우에 등록됩니다.



환경 변수 만들기

비주얼 스튜디오 코드 설치

비주얼 스튜디오 코드(Visual Studio Code, 이하 VSCode)는 마이크로소프트에서 만들어 무료로 배포하는 코드 편집기입니다. VSCode는 타입스크립트 지원에 가장 많은 공을 들인 편집기지만, 다른 프로그래밍 언어를 사용하는 개발자들도 매우 선호합니다.

① VSCode 설치

윈도우 파워셸에서 다음 명령으로 VSCode를 설치합니다.

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

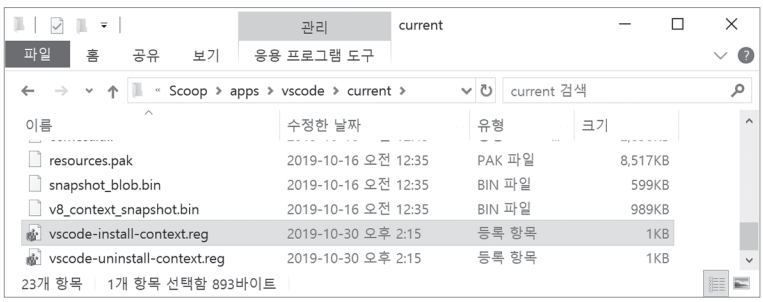
첫 번째 명령은 두 번째 명령에 필요한 scoop 부가 정보(extras)를 설치합니다. 그리고 두 번째 명령은 VSCode를 설치합니다.

② VSCode를 마우스 오른쪽 단축 메뉴에 등록

앞서 `scoop install vscode` 명령을 실행하면 맨 마지막에 다음과 같은 메시지가 출력됩니다.

```
Add Visual Studio Code as a context menu option by running: "C:\Scoop\apps\vscode\
current\vscode-install-context.reg"
```

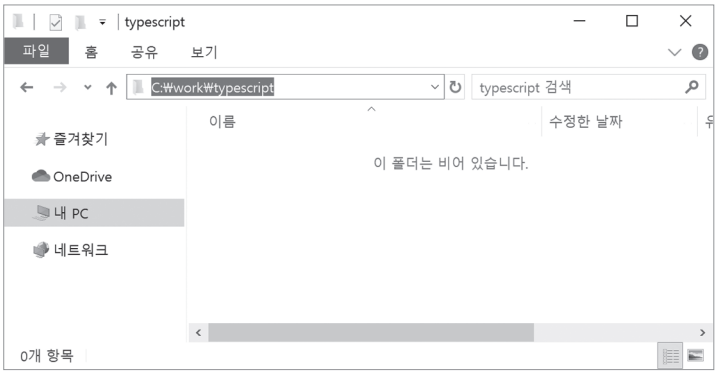
이 메시지는 C:\Scoop의 apps/vscode/current 디렉터리에 있는 vscode-install-context.reg 파일을 실행하라는 의미입니다. 이 파일을 실행하면 마우스 오른쪽 단축 메뉴에 VSCode 실행 메뉴가 나타납니다.



vscode-install-context.reg 파일 실행

③ 실습 디렉터리 만들기

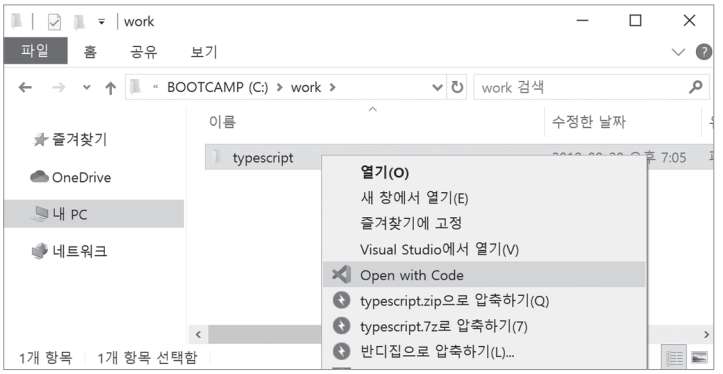
윈도우 탐색기를 열고 적당한 위치에 실습 디렉터리를 만듭니다. 필자는 C 드라이브 아래 work\typescript 디렉터리를 만들었습니다. 이 책의 모든 실습 파일을 이곳에 장별로 보관하기로 합니다.



실습 디렉터리 만들기

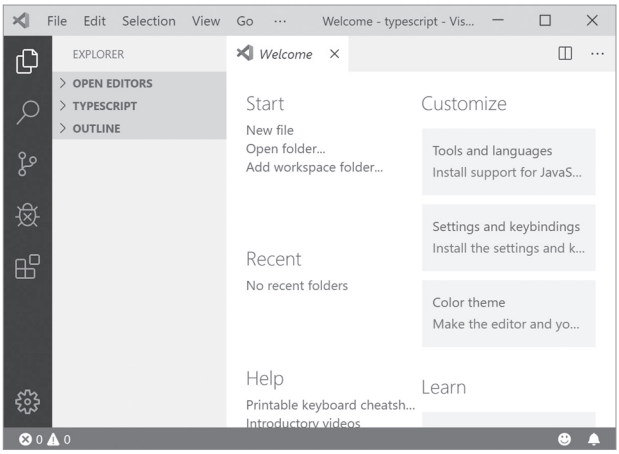
④ VSCode 실행하기

앞에서 만든 실습 디렉터리에 마우스 오른쪽을 누르고 단축 메뉴에서 [Open with Code]를 선택합니다.



실습 디렉터리에서 VSCode 실행

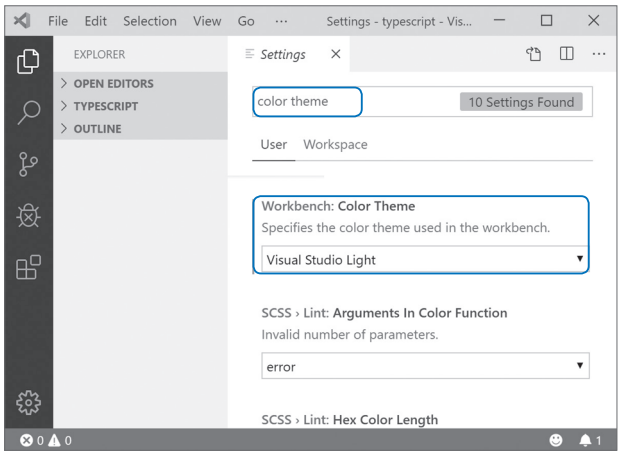
그러면 다음처럼 VSCode가 실행됩니다.



VSCode

VSCode 색상 테마 설정

VSCode를 처음 실행하면 색상 테마가 기본으로 어둡지만, 앞서 그림은 밝은 것으로 변경한 결과입니다. VSCode의 색상 테마를 변경하려면 [File → Preference → Settings] 메뉴를 선택하거나 키보드에서 Ctrl + , 를 누릅니다. Setting 창이 열리면 검색란에 'color theme'를 입력하고 설정 항목을 찾습니다.



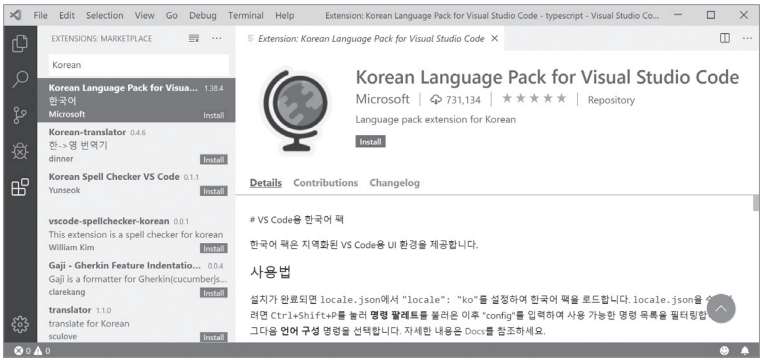
VSCode 색상 테마 설정

필자는 펼침 목록에서 'Visual Studio Light'를 선택했습니다. 자신의 취향에 맞게 골라서 사용합니다.

⑤ 한국어 언어 팩 설치하기

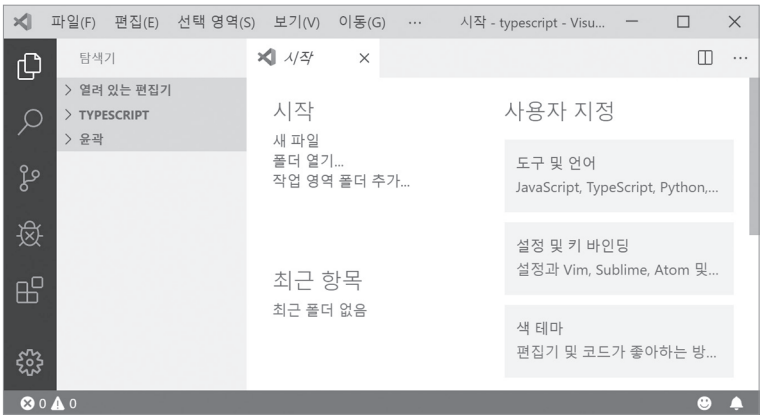
scoop으로 설치한 VSCode는 기본적으로 영문 버전입니다. 이를 한국어로 바꾸려면 한국어 언어 팩을 설치해야 합니다. 한국어 언어 팩은 VSCode의 확장(extension) 마켓플레이스에서

내려받아 설치합니다. 확장 마켓플레이스를 열려면 왼쪽 사이드 메뉴에서 마지막 아이콘(⊞)을 누르거나 [View → Extensions] 메뉴를 선택합니다. 또는 키보드에서 Ctrl + Shift + X를 누릅니다.



VSCode 확장 마켓플레이스

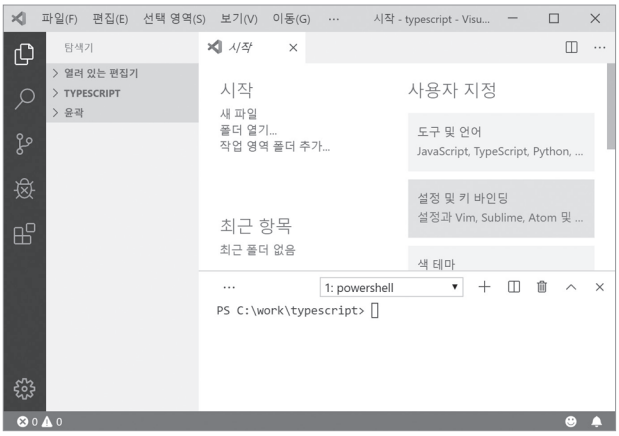
확장 마켓플레이스가 열리면 검색란에 “Korean”을 입력하고 목록에서 한국어 언어 팩을 찾아 〈Install〉 버튼을 누릅니다. 설치가 완료되면 오른쪽 아래에 팝업으로 나타나는 메시지에서 〈재시작〉 버튼을 누릅니다. 다음은 한국어 언어 팩이 적용된 VSCode입니다.



한국어 언어 팩이 적용된 VSCode

⑥ 터미널 열기

VSCode는 윈도우 파워셸 같은 터미널 기능도 지원합니다. VSCode에서 터미널 창을 보이게(또는 안 보이게) 하려면 [보기 → 터미널] 메뉴를 선택하거나 Ctrl + `를 누릅니다. 그러면 아래쪽에 다음과 같은 터미널 창이 나타납니다.



터미널 창 열기

VSCode는 이처럼 터미널을 제공함으로써 타입스크립트를 개발할 때 모든 작업을 VSCode 안에서 간편하게 할 수 있습니다. 터미널을 여러 개 실행하려면 터미널 창 오른쪽에 + 모양의 아이콘을 클릭하거나 Ctrl + Shift + ~를 누릅니다.

~ 키의 활용

키보드 왼쪽 Tab 키 위에는 ~ 키가 있습니다. ~ 키는 두 가지 문자를 입력할 수 있는데, 그냥 누르면 역따옴표(`, backtick)가, Shift 키와 함께 누르면 물결표(~, tilde)가 입력됩니다. ~ 키는 앞서 설명한 것처럼 VSCode에서 터미널을 보이거나 감추고 새 터미널을 실행하는 단축키로 사용됩니다. 그리고 타입스크립트 코드에서는 '템플릿 문자열(template string)'이라는 특별한 기능을 수행할 때도 사용됩니다.

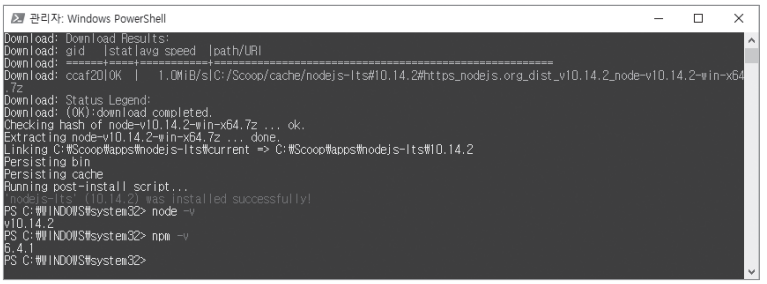
노드제이에스 설치

앞에서 VSCode를 설치했고 VSCode에서 터미널을 사용할 수 있다고 했지만, 윈도우 운영체제의 보안 정책에 따라 scoop과 같은 설치 프로그램은 반드시 관리자 모드로 동작해야 합니다. 따라서 윈도우 운영체제에서는 관리자 모드로 동작하는 터미널이 필요하므로, 앞으로 설치는 계속 관리자 모드로 동작하는 윈도우 파워셸에서 진행합니다. 만약, 리눅스나 macOS 운영체제라면 VSCode 내부 터미널에서 `apt`나 `brew`와 같은 명령으로 설치할 수 있습니다.

노드제이에스는 다음 명령으로 쉽게 설치할 수 있습니다.

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

첫 번째 명령은 scoop을 이용해 노드제이에스 안정 버전(long term support, LTS)을 설치합니다. 두 번째 명령은 설치된 노드제이에스의 버전을 확인합니다.



노드제이에스 설치 및 버전 확인

구글 크롬 브라우저 설치

웹 브라우저를 사용하는 개발에서 구글의 크롬(Chrome) 브라우저를 가장 많이 사용합니다. 크롬 브라우저가 웹 표준을 가장 많이 준수하는 이유도 있지만, 브라우저가 제공하는 개발자 기능이 매우 편리하기 때문입니다.

크롬 브라우저의 저작권은 구글에 있지만, 크로미움(Chromium)은 오픈소스 버전의 크롬 브라우저입니다. 저작권 차이만 있을 뿐 타입스크립트 개발 환경 관점에서는 큰 차이가 없으므로 이 책에서는 크로미움 브라우저를 설치합니다. 크로미움 브라우저는 윈도우 파워셸이나 VSCode 터미널에서 다음 명령으로 설치할 수 있습니다.

```
> scoop install chromium
> chrome
```

첫 번째 명령은 크로미움 브라우저를 설치합니다. 두 번째 명령은 설치가 정상적으로 끝났는지 알기 위해 브라우저를 실행해 보는 것입니다.

타입스크립트 컴파일러 설치

이제 타입스크립트 컴파일러를 설치할 차례입니다. VSCode를 실행하고 터미널에 다음 명령을 입력해 typescript 패키지를 설치합니다.

```
> npm i -g typescript
> tsc -v
```

첫 번째 명령은 typescript 패키지를 설치합니다. 앞에서 설치한 프로그램들과 달리 타입스크립트는 노드제이에스 환경에서만 동작합니다. 따라서 scoop이 아닌 노드제이에스의 패키지 관리자인 npm을 사용해서 설치합니다. 참고로 `i`는 install, `-g`는 global, 즉 전역 공간에 설치하라는 의미입니다.

typescript 패키지는 서버와 클라이언트로 동작하는 두 개의 프로그램을 포함하고 있습니다. 따라서 타입스크립트 컴파일러 이름은 패키지 이름과 달리 'tsc'입니다. 즉, 타입스크립트 컴파일러(typescript compiler)와 클라이언트(client)라는 의미가 동시에 있습니다.

두 번째 명령은 설치된 컴파일러의 버전을 확인하는 명령으로 다음 화면에서 보듯 3.7.4 버전의 타입스크립트가 설치되었습니다.

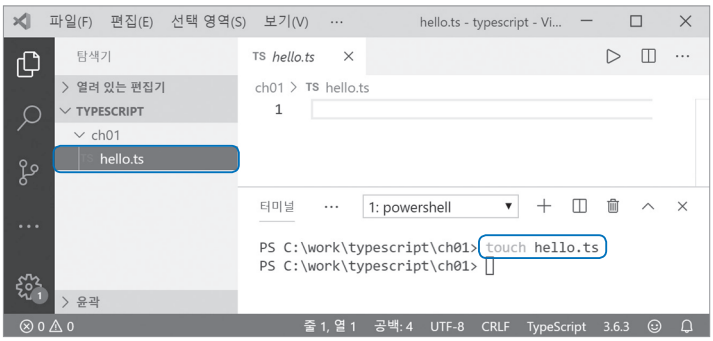
타입스크립트 컴파일러 설치 및 버전 확인

touch 프로그램 설치

리눅스 계열 운영체제에는 touch라는 프로그램이 기본으로 설치되어 있습니다. touch 프로그램은 파일을 생성할 때 지정한 이름의 파일이 이미 있으면 무시하고, 없으면 해당 이름으로 파일을 만들어 줍니다. 윈도우 운영체제에서 touch 프로그램은 다음 명령으로 설치할 수 있습니다.

```
> scoop install touch
```

VSCode의 터미널에서 **touch** 명령으로 hello.ts 파일을 생성합니다. 그러면 VSCode에서 왼쪽에 보이는 탐색기에 새로 생성된 hello.ts가 보입니다. 다음 그림은 이 파일을 선택해 편집 창이 열린 모습입니다.



touch로 파일 생성 및 열기

타입스크립트 컴파일과 실행

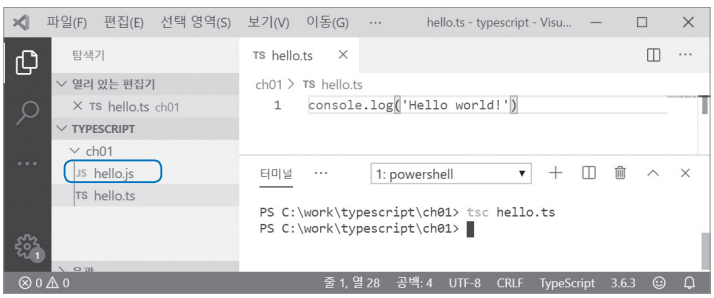
이제 hello.ts 파일에 다음 코드를 입력하고 Ctrl + S 키를 눌러 저장합니다.

• ch01/hello.ts

```
01: console.log('Hello world!')
```

그리고 다음처럼 터미널에서 명령을 실행하면 hello.js 파일이 생기는 것을 확인할 수 있습니다.

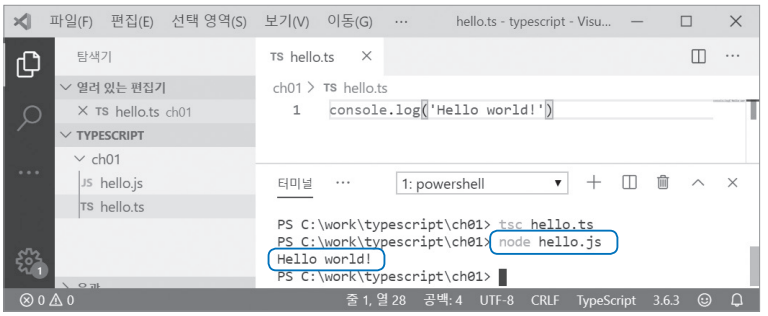
```
> tsc hello.ts
```



ts 파일 컴파일 후 js 파일 생성

즉, 타입스크립트 소스(hello.ts)가 TSC에 의해 트랜스파일되어 hello.js 파일이 생성되었습니다. 이제 노드제이에스로 hello.js 파일을 실행해 봅시다.

```
> node hello.js
Hello world!
```



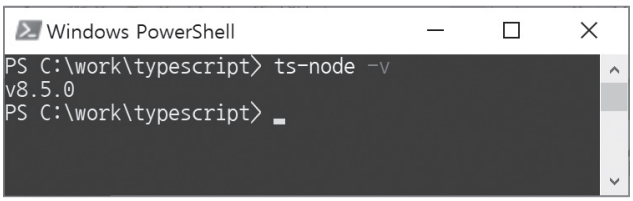
js 파일 실행 결과

ts-node 설치

tsc는 타입스크립트 코드를 ES5 형식의 자바스크립트 코드로 변환만 할 뿐 실행하지는 않습니다. 만약, 타입스크립트 코드를 ES5로 변환하고 실행까지 동시에 하려면 ts-node라는 프로그램을 설치해야 합니다. ts-node는 다음 명령으로 설치할 수 있습니다.

```
> npm i -g ts-node
```

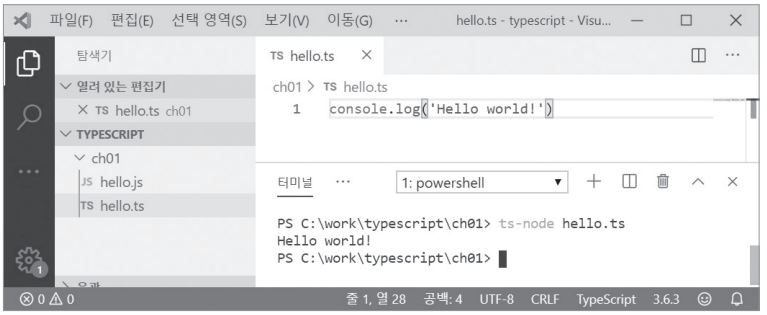
ts-node를 설치한 뒤, -v 옵션으로 프로그램의 버전을 확인합니다.



ts-node 버전 확인

이제 VSCode 터미널에서 다음 명령으로 컴파일과 실행을 동시에 진행해 봅시다.

```
> ts-node hello.ts
Hello world!
```



ts-node로 컴파일과 실행 동시 진행

이제 타입스크립트를 개발할 준비가 되었습니다. 다음 장에서는 타입스크립트 프로젝트를 생성하고 관리하는 내용을 살펴보겠습니다.