Instructors

Anders Hejlsberg - Anders is a prominent Danish software engineer who co-designed several popular and commercially successful programming languages and development tools. He was the original author of Turbo Pascal and the chief architect of Delphi. He currently works for Microsoft as the lead architect of C# and core developer on TypeScript.

Dan Wahlin - Dan Wahlin founded The Wahlin Group (http://www.TheWahlinGroup.com) which provides consulting and training services on Web technologies such as JavaScript, TypeScript, jQuery, AngularJS, SPAs, HTML5, ASP.NET and SharePoint. Dan is a Microsoft Regional Director and has been awarded Microsoft's MVP award multiple times for ASP.NET, Connected Systems and Silverlight. Dan blogs at http://weblogs.asp.net/dwahlin and writes regular columns for various technical magazines. Follow Dan on Twitter @DanWahlin.

Prerequisites

Working knowledge of JavaScript and experience coding in an object-oriented programming language.

Time Commitment and Schedule

3-5 hours per week for 6 weeks. For students taking this course “live” content will be released twice per week typically on Tuesday and Thursday. Also note that all deadlines and release times are in UTC. This course will cover the following topics:

- Introduction to the TypeScript language
- Setting up your environment to work with TypeScript
- Understanding basic types
- Functions in TypeScript
- TypeScript classes (basics, constructors and inheritance, and properties)
- Interfaces in TypeScript
- Generics in TypeScript

Deadlines and Grading

Exercises are not graded but students will be asked to confirm that they have completed each homework assignment successfully as a part of the module assessment and will receive points for affirming that they have completed the exercises. Students who answer “Yes” to completing the homework assignment will receive 10 points. All other assessment questions are graded at 1 point each. In order to earn a certificate for this course, the student must complete all homework assignments (exercises) and earn at least 50% of the total points available.

All exercises and assessments must be completed by the end of the course.

About the Course and Learning Objectives

This course will provide an overview of TypeScript, a statically typed superset of JavaScript that compiles to plain JavaScript. The TypeScript language features an innovative structural type system that incorporates gradual typing and type inference. In this course, you'll learn how to use features of TypeScript such as optional static types, classes, interfaces, and modules to JavaScript, enabling IDE productivity features such as statement completion, refactoring, and code navigation. You'll learn how to make it easier for teams to communicate requirements and build applications safely using these advanced features through Microsoft Visual Studio. The TypeScript project is open-source and hosted on GitHub.

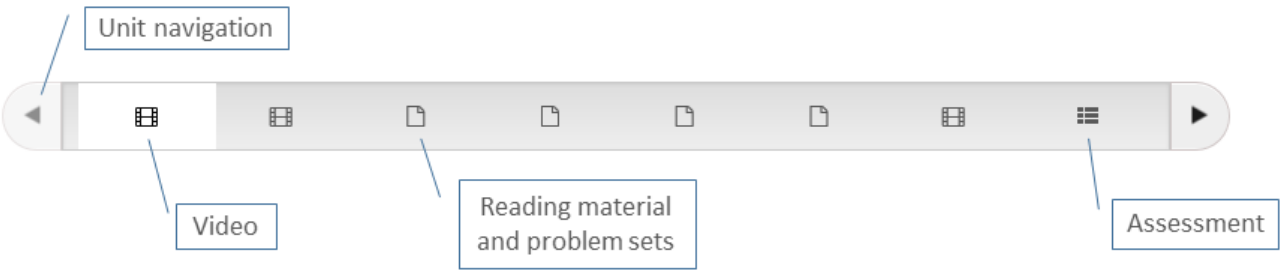
This course will cover the very basics of TypeScript. We make an assumption that you have a general understanding of either JavaScript or another programming language. This course will not teach you JavaScript or the fundamentals of programming so if you haven't programmed before, be sure to set aside extra time so you can explore concepts and skills that may be foreign to you. If know another language like Java, Python, or C#, this course will cover many programming constructs already very familiar to you (like classes and inheritance) but will show you how to implement them in TypeScript. If you know JavaScript well, many of the concepts in TypeScript will be familar but we'll cover some concepts that may be new to you and show you how TypeScript can enhance your JavaScript programming.

What you'll learn:

- You will learn what TypeScript is and how it can add value to your JavaScript Development
- You will learn how to write TypeScript code and how to use the TypeScript API
- You will learn how to use TypeScript to manage your JavaScript code base

Navigating the Course

Introduction to TypeScript is laid out in modules and units. Each module contains 1 or more units that cover the material for that module. The module is the topic area and the units has all the learning material you'll need to learn that topic. To navigate throught the module's units, use the navigation bar at the top of the module. You can use the left and right arrows to move through each unit.



Each unit will generally have some **video content**, **reading material and problem sets**, and a set of **assessment questions** to test your knowledge. Some units also have a **discussion board** where you can post questions and other material so fellow students can provide feedback. The units are meant to be taken in order but you can skip around if you already have mastered the material in a particular unit. In order to do well on the assessments, you should learn the material in all the units. **Hover over the unit with your mouse to see the unit title**.

Exercises

As a software developer, you know that there's no better way to learn how to code than to actually try things on your own. Throughout this course, you will be asked to complete a number of exercises to help you lock in concepts and learn how to program in TypeScript. Please keep the following in mind as you prepare for the exercises.

- **TypeScript Playground:** This course uses a version of the TypeScript Playground which is a web-based editor that provides real-time formatting, compiling, and error checking of TypeScript code. Many of the exercises in this course use this editor and it will be available right in the unit that has the exercise.
- **Editors and Platforms:** While TypeScript can be used in a variety of editors, some, more complex exercises are written for Visual Studio and we provide starter code in the form of a Visual Studio solution. To take advantage of the starter code and to complete the exercise, you will need access to a Windows computer to install and run the free Visual Studio Community Edition. It may be possible to complete the exercise in other tools and platforms by looking at the starter code in a text editor.
- **Solution Code:** We do not provide solution code for any of the exercises. This means you will need to work out the problem sets on your own and discover the answer.
- **Difficulty**: Some of the exercises are very basic (particularly in the early modules) and are designed to get you started with the language. As the modules progress, the exercises will get progressively difficult. If you don't find the earlier exercises challenging enough, we encourage you to do some coding on your own using the Playground or Visual Studio until you have the concepts for that module clear. On the flip side, if you find the later exercises challenging, work at them until you discover the answer. Your work will pay off!
- **JavaScript and TypeScript**: Because TypeScript compiles to plain JavaScript, it will be possible to complete the exercises using only JavaScript. However we encourage you to complete all the exercises using the features of TypeScript being taught in the exercises. Avoid taking shortcuts

by writing plain JavaScript so you can learn how TypeScript works. Also, if you need more of a challenge in earlier exercises, try to convert some of your existing JavaScript code to TypeScript.

Being a Self-paced Learner

By taking this course, you have entered into an exciting world of self-paced, online learning! If online education is new to you, there are some things you should be aware of that may help you be more successful with this style of learning.

Staying on track. Because there isn't a live instructor to guide you through the material, you'll need to work out a system on your own so you can keep up with the schedule. You should plan on logging in a couple of times a day to check the discussion boards and check for new material. **You will not get regular email or announcements when things have changed** so be sure to check in regularly. Generally new material will be published once or twice a week (for those taking the course live) but your fellow students will be active in the discussion boards and you'll want to participate in the conversation.

Working on problem sets. It will be up to you to complete the problem sets in each of the modules. We will not provide answer keys for the problem sets or assessments so you'll have to work on the problems until you find the answer. While it may be possible to get solutions from other students or on the internet, we encourage you to work out the problems on your own and use your fellow students or internet resources to provide clues to solving the problem sets and assessments. This will enrich your learning experience and help lock in the concepts. If you get stuck, don't give up! There is enough material in the modules to help you solve the problem and your hard work will pay off.

Instructor feedback. Because of the nature of MOOC-style instruction (Massive Open Online Course), instructors aren't able to provide active feedback to individual students. Most MOOCs have thousands of students enrolled at a time and engaging personally with each student is not possible. However, we encourage you to use the module discussion boards or the general discussion board and seek help and feedback from your fellow students who are enrolled in the course.

Stick with it! We encourage you to take the course from beginning to end to get the full learning experience. Some modules may be very easy for you and others will be harder. But each module should offer something of value. There's nothing like that feeling of accomplishment when you complete something from beginning to end so hang in there and enjoy the course!

Other Resources

edX Student FAQ: This FAQ answers a lot of questions about the edX student experience. You should at least read over the questions so if you encounter any issues during the course, you can refer back to this document for help.

DemoX: A fun and interactive course designed to help you explore the edX learning experience. Perfect to take before you start your course.