Build an Excel Web Add-in in Minutes

Michael Olafusi

Microsoft MVP

www.UrBizEdge.com

Disclaimer

01 I am still at the learning phase of Excel (web) add-ins

02 I come from the VBA background and know only a few hours course worth of JavaScript

03 I might say some things that are technically incorrect so best to factcheck everything I say.

04 My main goal is to stimulate your interest in this new cool way of building Excel add-ins

Excel (web) Add-ins run inside Excel (desktop app & Excel online).

It interacts with the Excel document using JavaScript API.

- VBA
- COM Add-in
- Office Add-in
- Office Scripts

Excel (web) Add-in

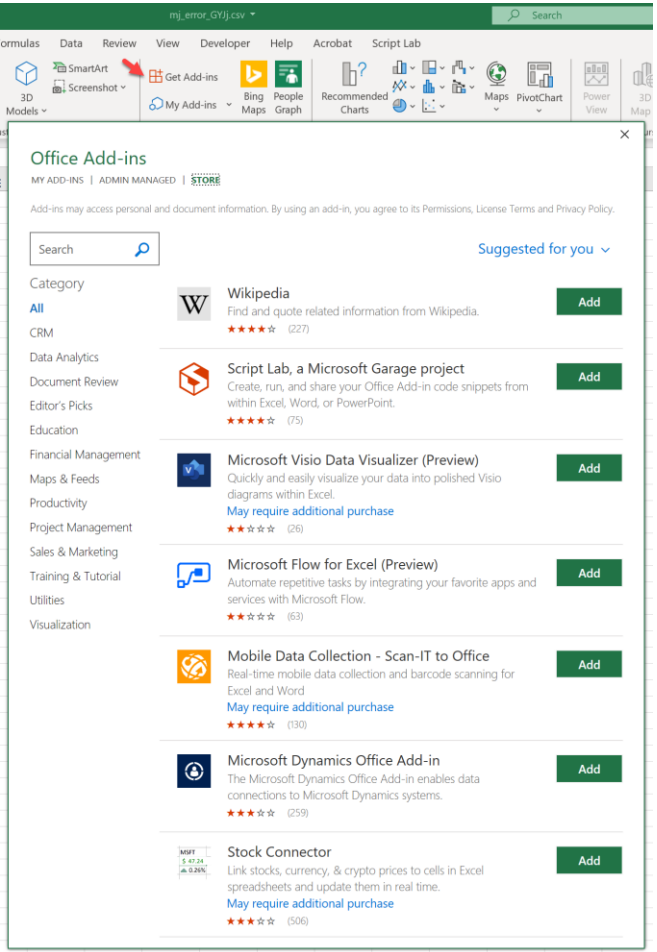
=

Manifest.xml

+

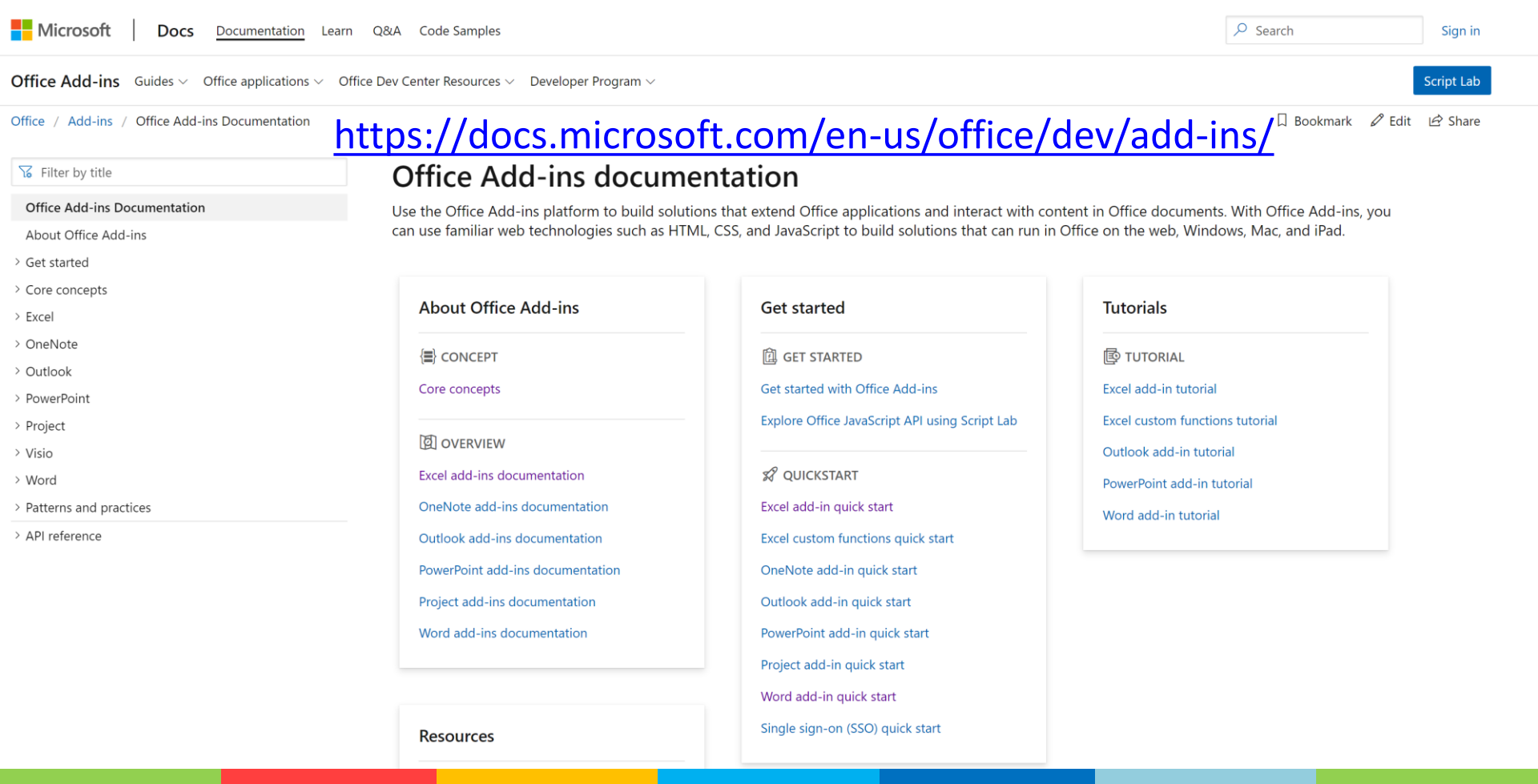
Web App

The Case for Excel (Web) Add-ins

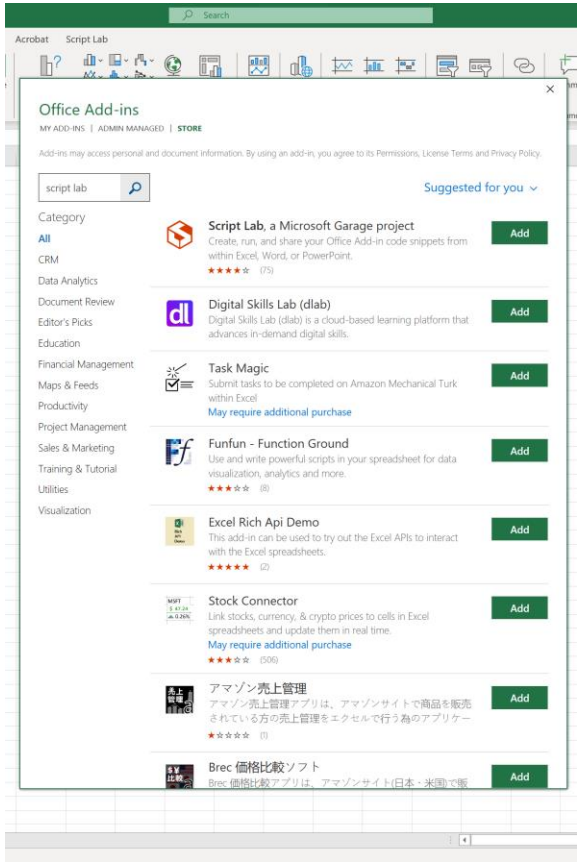


- Best native cross platform support: Excel on Windows, Excel on Mac, Excel on iPad and Excel on Web
- Easiest to deploy (especially updates) and distribute to end users. Thanks to AppSource Store
- Fixes most of the security worries and central admin preference for organizations
- Opens you and your add-in up to the rich goodness of JavaScript libraries and methods
- Microsoft is putting more current work into it than they are on the VBA and VSTO.
- Safe to say the future involves more of it and relatively less of the others.

Interesting Resource to Help with Getting Started



An Easy Way To Try Out Creating Your First Office Add-In



Apps > **Script Lab, a Microsoft Garage project**

Script Lab, a Microsoft Garage project

save for later

Microsoft Corporation

★★★★☆ 4.0 (75)

Overview Reviews

GET IT NOW

Pricing
Free

Products
Excel
PowerPoint
Word

Publisher
Microsoft Corporation

Acquire Using
Work or school account
Microsoft account

Version
1.5.0.0

Updated
12/11/2019

Categories
IT & Management Tools
Productivity
Training & Tutorial

Products supported
Excel 2013+
Excel 2016+
Excel 2016 for Mac
Excel Online
PowerPoint 2013+
PowerPoint 2016+
PowerPoint 2016 for Mac
PowerPoint Online
Word 2013+
Word 2016+
Word 2016 for Mac
Word Online

Create, run, and share your Office Add-in code snippets from within Excel, Word, or PowerPoint.

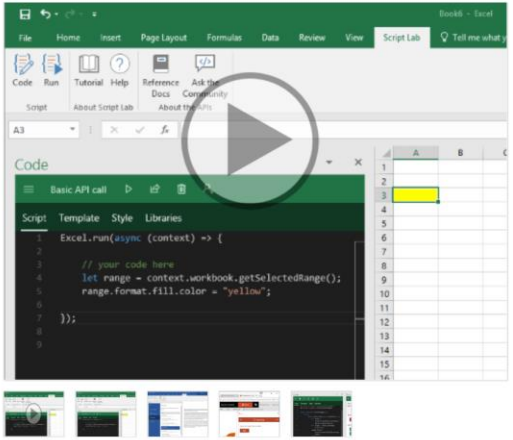
Experiment with the Office JavaScript API without ever leaving Excel, Word, or PowerPoint! You can use Script Lab to:

- Create and edit code snippets that can include JavaScript, HTML, CSS, and references to libraries and data on the web
- Run the snippets and instantly see the result in a task pane and in the Office document
- Get started quickly with a selection of samples
- Share and save your snippets with GitHub gists
- Learn the JavaScript API with the help of IntelliSense while you edit
- Try TypeScript: make use of arrow functions, template strings, or even the latest TypeScript 2.0+ features like async/await
- Prototype an add-in you're planning to develop

There's no need to install any other software or configure your environment: as long as you have Office, you can get started in seconds.

Script Lab, a Microsoft Garage project, works in Excel, Word, and PowerPoint (Office 2013 and later, Office Online, and Office for Mac).

The source code for Script Lab is open to the community at https://github.com/OfficeDev/script-lab. We welcome your contributions!



Add-in capabilities

When this add-in is used, it

- Can read and make changes to your document
- Can send data over the Internet

https://appsource.microsoft.com/en-US/product/office/wa104380862

Building A Proper Office Add-in Using Yeoman generator

Filter by title

- Office Add-ins Documentation
- About Office Add-ins
- Get started
  - Start Here! A guide for beginners
  - Transition Here! A guide for VSTO add-in developers
  - Building Office Add-ins
  - Quick starts
    - Excel add-in
    - Excel custom functions
    - OneNote add-in
    - Outlook add-in
    - PowerPoint add-in
    - Project add-in
    - Word add-in
    - Single sign-on (SSO)
  - Explore Office JS APIs using Script Lab
  - Set up your development environment
- Core concepts
- Excel
- OneNote
- Outlook
- PowerPoint
- Project
- Visio
- Word
- Patterns and practices
- API reference

Create the add-in project

Run the following command to create an add-in project using the Yeoman generator:

command line — Copy

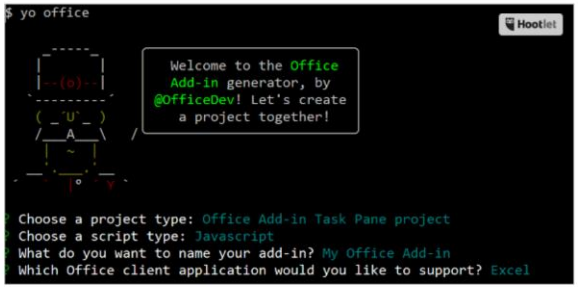
```
yo office
```

Note

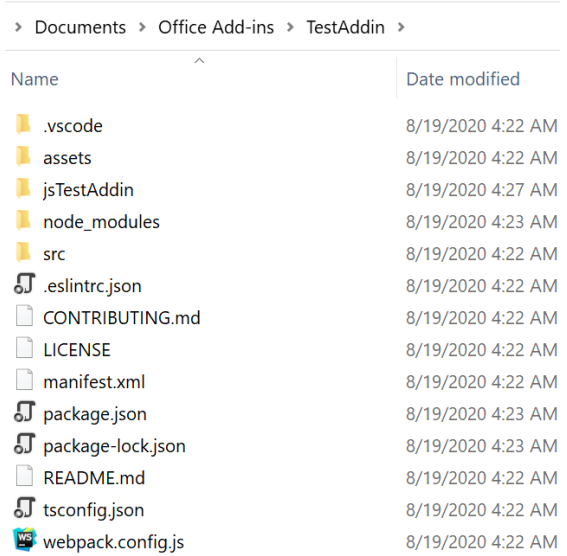
When you run the `yo office` command, you may receive prompts about the data collection policies of Yeoman and the Office Add-in CLI tools. Use the information that's provided to respond to the prompts as you see fit.

When prompted, provide the following information to create your add-in project:

- **Choose a project type:** `Office Add-in Task Pane project`
- **Choose a script type:** `Javascript`
- **What do you want to name your add-in?** `My Office Add-in`
- **Which Office client application would you like to support?** `Excel`

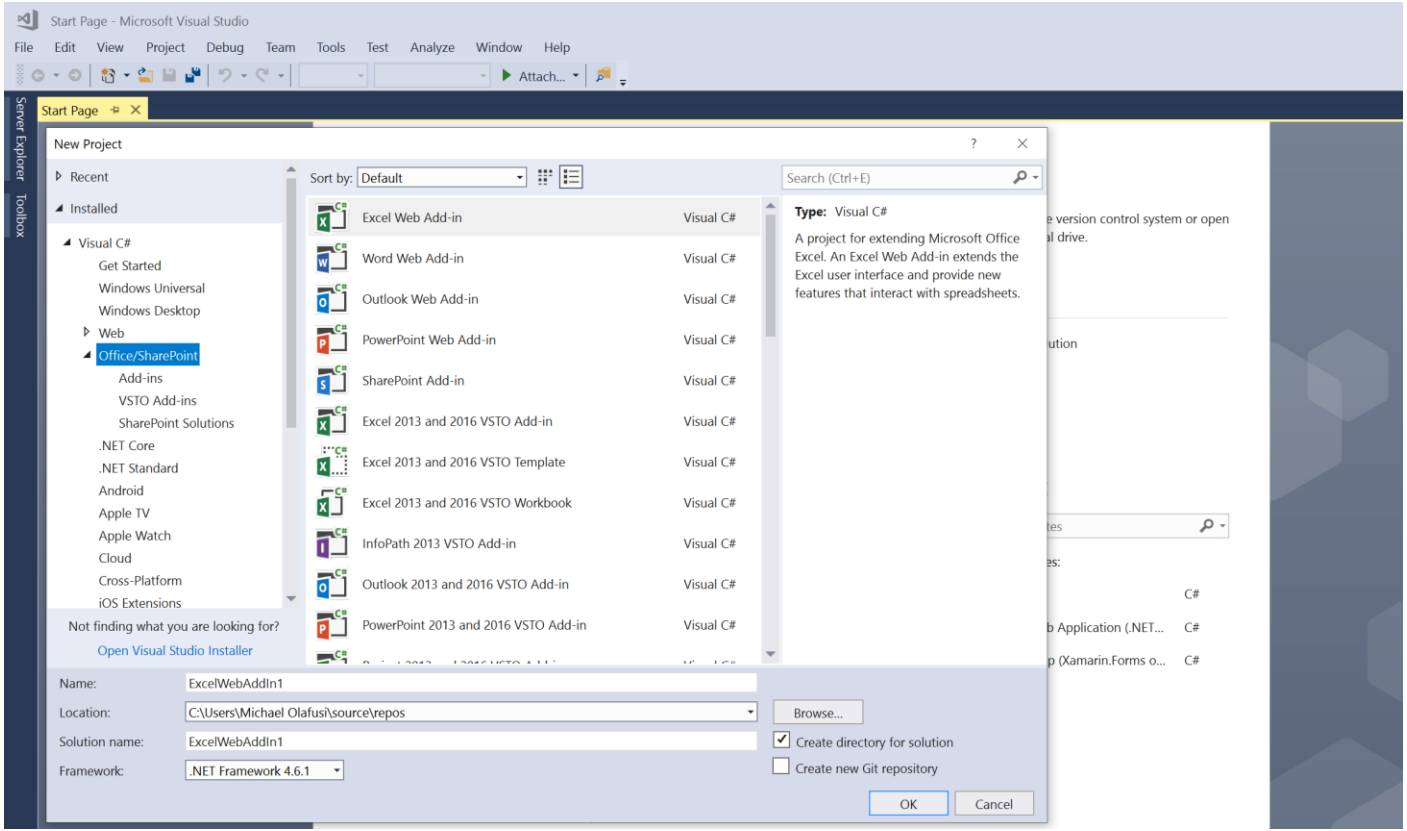


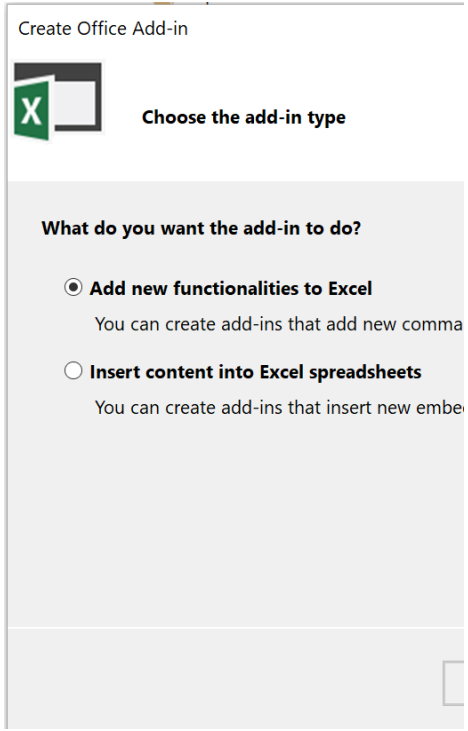
After you complete the wizard, the generator creates the project and installs supporting Node components.



https://docs.microsoft.com/en-us/office/dev/add-ins/quickstarts/excel-quickstart-jquery?tabs=yeomangenerator

Building A Proper Office Add-in Using Visual Studio





https://docs.microsoft.com/en-us/office/dev/add-ins/quickstarts/excel-quickstart-jquery?tabs=visualstudio

A Crash Course on JavaScript

WHAT YOU'LL LEARN

https://www.codecademy.com/learn/introduction-to-javascript

1 Introduction
In this course, you will learn about JavaScript data types, built-in methods, and variables.
View Details | Start

2 Conditionals
Learn how to use if, else if, else, switch, and ternary syntax to control the flow of a program in JavaScript.
View Details | Start

3 Functions
Learn about JavaScript function syntax, passing data to functions, the return keyword, ES6 arrow functions, and concise body syntax.
View Details | Start

4 Scope
Learn about global and block level scope in JavaScript.
View Details | Start

5 Arrays
In this course, you will learn about arrays, a data structure in JavaScript used to store lists of data.
View Details | Start

+ 9 more lessons

https://javascript.info/

PART 1

The JavaScript language

Here we learn JavaScript, starting from scratch and go on to advanced concepts like OOP.

We concentrate on the language itself here, with the minimum of environment-specific notes.

An introduction

1.1 An Introduction to JavaScript
1.2 Manuals and specifications
1.3 Code editors
1.4 Developer console

JavaScript Fundamentals

2.1 Hello, world!
2.2 Code structure
2.3 The modern mode, "use strict"
2.4 Variables
2.5 Data types
2.6 Interaction: alert, prompt, confirm
2.7 Type Conversions
2.8 Basic operators, maths
2.9 Comparisons
2.10 Conditional branching: if, '?'
2.11 Logical operators
2.12 Nullish coalescing operator '??'
2.13 Loops: while and for
2.14 The "switch" statement
2.15 Functions
2.16 Function expressions
2.17 Arrow functions, the basics
2.18 JavaScript specials

Code quality

3.1 Debugging in Chrome
3.2 Coding Style
3.3 Comments
3.4 Ninja code
3.5 Automated testing with Mocha
3.6 Polyfills

Objects: the basics

4.1 Objects
4.2 Object copying, references
4.3 Garbage collection
4.4 Object methods, "this"
4.5 Constructor, operator "new"
4.6 Optional chaining '?.'
4.7 Symbol type
4.8 Object to primitive conversion

Data types

5.1 Methods of primitives
5.2 Numbers
5.3 Strings
5.4 Arrays
5.5 Array methods
5.6 Iterables
5.7 Map and Set
5.8 WeakMap and WeakSet
5.9 Object.keys, values, entries
5.10 Destructuring assignment
5.11 Date and time
5.12 JSON methods, toJSON

Advanced working with functions

6.1 Recursion and stack
6.2 Rest parameters and spread syntax
6.3 Variable scope, closure
6.4 The old "var"
6.5 Global object
6.6 Function object, NFE
6.7 The "new Function" syntax
6.8 Scheduling: setTimeout and setInterval
6.9 Decorators and forwarding, call/apply
6.10 Function binding
6.11 Arrow functions revisited

Object properties configuration

7.1 Property flags and descriptors
7.2 Property getters and setters

A Crash Course on TypeScript

TypeScript Documentation

https://www.typescriptlang.org/docs/

Get Started

Quick introductions based on your background or preference.

- TS for the New Programmer
- TypeScript for JS Programmers
- TS for Java/C# Programmers
- TS for Functional Programmers
- TypeScript Tooling in 5 minutes

Handbook

A good first read for your daily TS work.

- The TypeScript Handbook
- Basic Types
- Interfaces
- Functions
- Literal Types
- Unions and Intersection Types
- Classes
- Enums
- Generics

We also have an epub and pdf version of the Handbook.

Handbook Reference

Deep dive reference materials.

- Advanced Types
- Utility Types
- Decorators
- Declaration Merging
- Iterators and Generators
- JSX
- Mixins
- Modules
- Module Resolution
- Namespaces
- Namespaces and Modules
- Symbols
- Triple-Slash Directives
- Type Compatibility
- Type Inference
- Variable Declaration

Tutorials

Using TypeScript in several environments.

- ASP.NET Core
- Gulp
- DOM Manipulation
- Migrating from JavaScript
- Using Babel with TypeScript

Declaration Files

Learn how to write declaration files to describe existing JavaScript. Important for DefinitelyTyped contributions.

- Introduction
- Declaration Reference
- Library Structures
- Do's and Don'ts
- Deep Dive
- Publishing
- Consumption

JavaScript

How to use TypeScript-powered JavaScript tooling.

- JS Projects Utilizing TypeScript
- Type Checking JavaScript Files
- JSDoc Reference
- Creating .d.ts Files from .js files

Learn X in Y minutes

Share this page

Select theme: light dark

Where X=TypeScript

Get the code: learntypescript.ts

TypeScript is a language that aims at easing development of large scale applications written in JavaScript. TypeScript adds common concepts such as classes, modules, interfaces, generics and (optional) static typing to JavaScript. It is a superset of JavaScript: all JavaScript code is valid TypeScript code so it can be added seamlessly to any project. The TypeScript compiler emits JavaScript.

This article will focus only on TypeScript extra syntax, as opposed to JavaScript.

To test TypeScript's compiler, head to the Playground where you will be able to type code, have auto completion and directly see the emitted JavaScript.

```
// There are 3 basic types in TypeScript
let isDone: boolean = false;
let lines: number = 42;
let name: string = "Anders";

// But you can omit the type annotation if the variables are derived
// from explicit literals
let isDone = false;
let lines = 42;
let name = "Anders";

// When it's impossible to know, there is the "Any" type
let notSure: any = 4;
notSure = "maybe a string instead";
notSure = false; // okay, definitely a boolean

// Use const keyword for constants
const numLivesForCat = 9;
numLivesForCat = 1; // Error

// For collections, there are typed arrays and generic arrays
let list: number[] = [1, 2, 3];
// Alternatively, using the generic array type
let list: Array<number> = [1, 2, 3];

// For enumerations:
enum Color { Red, Green, Blue };
let c: Color = Color.Green;

// Lastly, "void" is used in the special case of a function returning nothing
function bigHorribleAlert(): void {
```

https://learnxinyminutes.com/docs/typescript/

Thank You!

Twitter: @olafusimichael
Web: www.urbizedge.com
LinkedIn: olafusimichael (https://www.linkedin.com/in/olafusimichael/)
Blog: www.olafusimichael.com