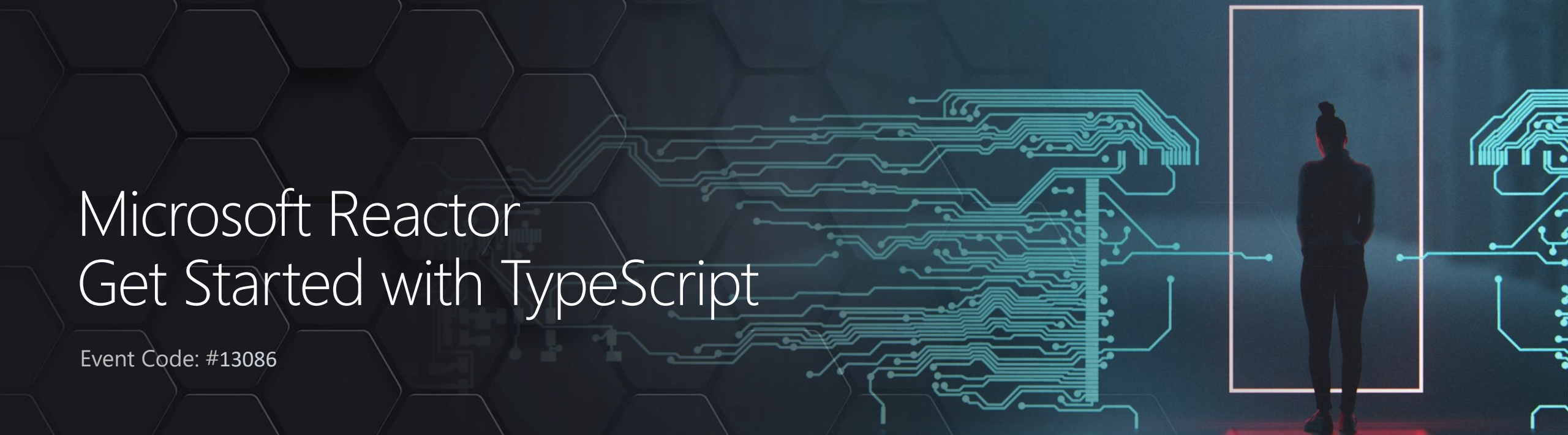
Microsoft

Reactor



Microsoft Reactor
Get Started with TypeScript

Event Code: #13086

FAQ's

What are the Reactors?

Reactors are community spaces where technology professionals meet, learn, and connect—to both their local peers as well as industry-leading ideas and technology from Microsoft, partners, and the open source community.

With a diverse mix of hands-on workshops, expert panels, and community events, there's something for everyone—whether you're just getting started or working on complex projects.

Reactor programming is always free and inclusive of a broad set of products, tools, and technologies. You will find:

- First party content designed to help developers learn new skills in high-demand fields such as Machine Learning, AI, and Data Science
- New content around breakthrough technologies and concepts such as quantum computing and blockchain
- Resources, talks and classes from Microsoft for Startups to deliver what startups need to succeed
- Cloud Advocates who deep-dive into their technical specialties and the Industry Experiences team to provide tailored content around applications for Manufacturing, Healthcare, Retail, and other verticals
- Reactors always welcome community groups, Meetups, partners and MVPs to use our space

Map


Stockholm
London
Redmond
Toronto
New York
San Francisco
Tel Aviv
Shanghai
Abu Dhabi
Bangalore
São Paulo
Sydney

Our Code of Conduct

Microsoft is dedicated to empowering every person and every organization on the planet to achieve more.

This includes Microsoft Reactor events where we seek to provide a respectful, friendly, professional experience for everyone, regardless of gender, sexual orientation, physical appearance, disability, age, race or religion.

We do not tolerate any behaviour that is harassing or degrading to any individual, in any form. Individuals are responsible for knowing and abiding by these standards. We encourage everyone to assist in creating a welcoming and safe environment.

Be aware of others

Be friendly and patient

Be welcoming and respectful

Be open to all questions and viewpoints

Be understanding of differences

Be kind and considerate to others

Our Code of Conduct

Microsoft is dedicated to empowering every person and every organization on the planet to achieve more.

This includes Microsoft Reactor events where we seek to provide a respectful, friendly, professional experience for everyone, regardless of gender, sexual orientation, physical appearance, disability, age, race or religion.

We do not tolerate any behaviour that is harassing or degrading to any individual, in any form. Individuals are responsible for knowing and abiding by these standards. We encourage everyone to assist in creating a welcoming and safe environment.

Our Ask of You

Be aware of others

Be friendly and patient

Be welcoming and respectful

Be open to all questions and viewpoints

Be understanding of differences

Be kind and considerate to others

Get Started with TypeScript

Allen Sanders

Principal Cloud Architect

Speaker Bio: Allen Sanders is a Principal Cloud Architect with over 25 years of experience in software engineering, architecture and design delivering technology strategy and business solutions in multiple verticals. As a Microsoft Certified Professional, he has taught, led and mentored teams of varying sizes in multiple application transformation efforts. He has a passion for learning – both his own and for others.

Agenda

1	What is TypeScript?
2	TypeScript Development Environment
3	Compiling TypeScript Files
4	Creating a TypeScript Project
5	Types in TypeScript
6	*enum*, *any*, and *unknown* Types
7	Union, Intersection, and Literal Types
8	Collection Types
9	Knowledge Check
10	Summary

What is TypeScript?

https://aka.ms/GetStartedTypeScript

https://aka.ms/VariablesTypeScript

What is TypeScript?

- Open-source language developed by Microsoft
- Represents superset of JavaScript
- Brings types and static type checking to JavaScript development

Type Hints

- TypeScript allows assignment of data type to variable using type hint
- Improves readability and enables error checking at compile time
- Also supports type inference, inferring data type from assignment

Additional Features

- Interfaces
- Namespaces
- Generics
- Abstract classes
- Data modifiers
- Optionals
- Function overloading
- Decorators
- Type utils
- readonly keyword

TypeScript & JavaScript

- TypeScript is fully compatible with JavaScript (strict superset)
- However, browsers understand JavaScript only
- TypeScript gets “transpiled” into clean JavaScript

TypeScript Development Environment

Developing in TypeScript

- Couple of options:
  - TypeScript Playground
  - Full-featured IDE like Visual Studio Code
- In your browser, search for “download vs code” to download/install
- Supported on Windows, macOS, and Linux

Install the Transpiler

- Install Node.js and npm
- Run *npm install -g typescript* to install globally
- Can also install local to a specific project by removing the *-g* flag
- Run *tsc* in the terminal to confirm install

Compiling TypeScript Files

Compiling TypeScript Files

- TypeScript files have a .ts extension
- To “transpile” a .ts file to JavaScript, use *tsc <filename>.ts*
- Results in a .js JavaScript file

Compiler Options

- Multiple options available to manage how JavaScript is generated
- https://www.typescriptlang.org/docs/handbook/compiler-options.html
- Can be used on command-line or defined in a *tsconfig.json* file

Compiler Options

- Common compiler options include:
  - *--noImplicitAny* – compiler raises errors on expressions/declarations with implied any type
  - *--noEmitOnError* – disable emitting of JS files if any type checking errors reported
  - *--target* – specifies target ECMAScript version for JS file

Creating a TypeScript Project

VS Code

- In VS Code, create a workspace, a project folder, and add .ts files
- Use *tsc --init* to generate a tsconfig.json file
- Run *tsc filename.ts* to transpile a specific TypeScript file

VS Code

- Run *tsc* without a filename argument to:
  - Load the config
  - Transpile all .ts files in the folder
- Resulting .js files will be added to *outDir* folder if configured

Executing the Code

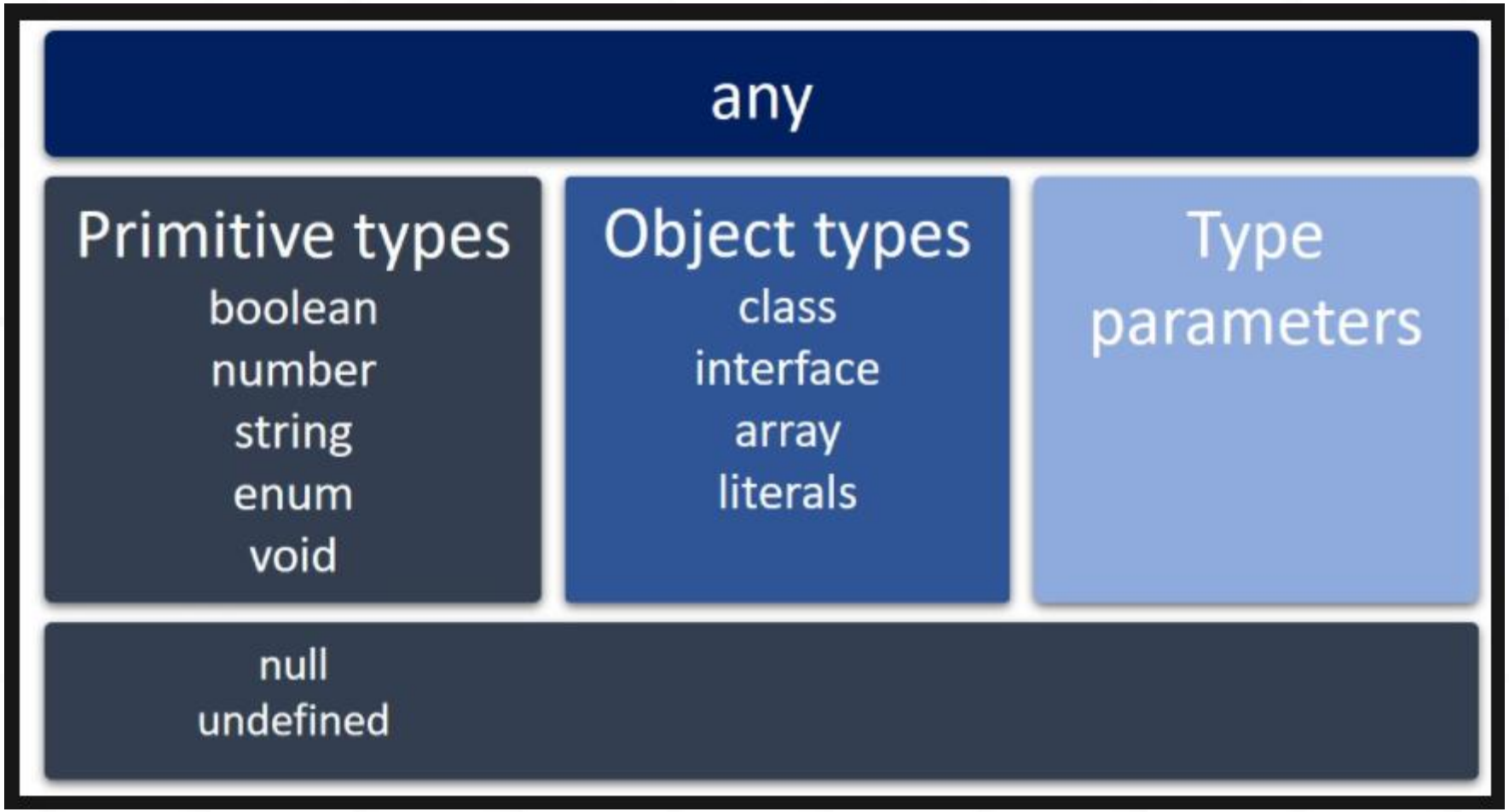
- Resulting .js file can be executed a couple of different ways:
  - In the terminal using *node*
  - Embedded as a script in an HTML file in browser

Types in TypeScript

Declaring Variables

- *let variableName: type;*
- Also, *let variableName: type = value;*
- Or *const constName: type = value;*

Types Available in TypeScript



enum, *any*, and *unknown* Types

enum Type

- Mechanism for working with sets of related constants
- Provides a symbolic name for a set of named, numeric values
- Created using the *enum* keyword

enum Type

- Becomes a new type that can be used in variable declarations
- Provide access to both numeric value and symbolic name
- Help to improve readability of code

any Type

- Base type for all other types in TypeScript
- Can represent any JavaScript value
- Useful when dynamic typing is still required in TypeScript code
- Should be used sparingly

unknown Type

- Similar to the *any* type but safer
- Any value is assignable but members of unknown type inaccessible
- Supports type guards using *typeof*
- Couple with type assertion to access underlying properties/functions

Union, Intersection, and Literal Types

Union Types

- Like the any type, union types can hold values of multiple types
- Using | operator, able to restrict valid values to a discrete list of types
- For example, *let variableName: type1 | type2 | type3;*

Intersection Types

- Combines 2 or more types into a new type with all properties
- Provide a way to build new types through composition
- Uses the & operator: *type typeName = type1 & type2;*
- Often used with interfaces to build composed structures

Literal Types

- TypeScript supports *string*, *number*, and *boolean* literal types
- Uses | to define a new type that supports a discrete set of literal values
- If used as type on new variable declaration, variable can only take on 1 of the allowed literal values

Collection Types

Arrays

- Arrays are supported in TypeScript
- Declared using 1 of 2 formats:
  - variableName: type[]
  - variableName: Array<type>

Tuples

- Instead of array of same type, provides array of mixed types
- *variableName: [type1, type2]*
- Types are fixed – valid values for assignment in form *[type1, type2]*

Knowledge Check

Summary

What Did We Cover?

- An overview of TypeScript
- Setting up our TypeScript development environment
- Multiple types – both primitive and more sophisticated types

Next Steps / Further Study

- https://aka.ms/GetStartedTypeScript
- https://aka.ms/VariablesTypeScript
- https://www.typescriptlang.org/
- https://www.typescriptlang.org/play
- https://www.typescriptlang.org/docs

Microsoft

Reactor

Thank You!

Q&A

https://www.meetup.com/pro/microsoft-reactor

@MSFTReactor

http://www.youtube.com/c/MicrosoftReactor

aka.ms/ReactorEmailSignUp

Microsoft

Reactor

We are constantly striving to create excellent content
and would appreciate if you could take this brief survey.

Survey Link: https://aka.ms/Reactor/Survey

Please enter the event code 13086 at the start of survey

Microsoft