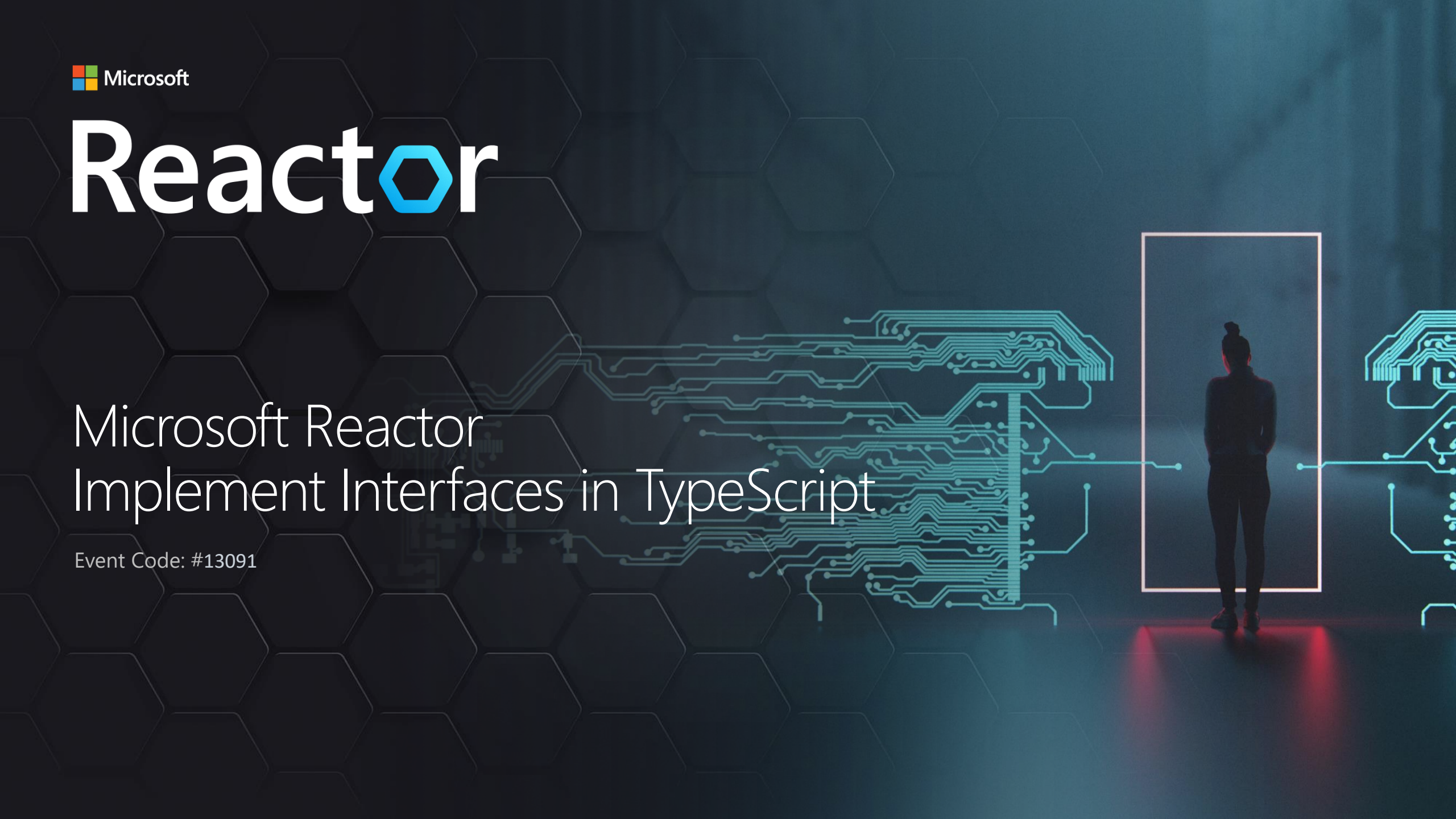
Microsoft

Reactor



Microsoft Reactor
Implement Interfaces in TypeScript

Event Code: #13091

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

Implement Interfaces in 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	Introduction
2	Interfaces in TypeScript
3	Extending Interfaces in TypeScript
4	Using Interfaces in TypeScript
5	Knowledge Check
6	Summary
7	
8	
9	
10	

Introduction

https://aka.ms/InterfacesTypeScript

Prerequisites

- Knowledge of TypeScript helpful
- Familiarity with JavaScript
- Familiarity with functions & arrays in JavaScript

Software Used

- Git
- Visual Studio Code
- Node.js
- TypeScript
- TypeScript Playground (https://www.typescriptlang.org/play)

Installing TypeScript

- Use *npm install -g typescript* to install globally
- Run *tsc --version* from terminal to confirm

Interfaces in TypeScript

What is an Interface?

- "Code contract" that defines shape of a type
- Defines properties and methods (names and signatures) provided
- However, does not include implementation

What is an Interface?

- Becomes a new type
- Variable, function, or class can utilize/satisfy
- By doing so, agrees to implement defined contract (properties & methods)

Structural Type System

- TypeScript is a structural vs. nominal type system
- Means TypeScript only considers members on type for comparison
- Two constructs with same structure are interchangeable

See https://www.typescriptlang.org/play#example/structural-typing

Interfaces in TypeScript

- Have no run-time representation; compile-time construct only
- Useful as part of TypeScript's static type checking
- Brings readability to code and enables polymorphism

vs. Type Alias

- Has similarities with interface

```
interface Employee {
    firstName: string;
    lastName: string;
    fullName(): string;
}
```

VS

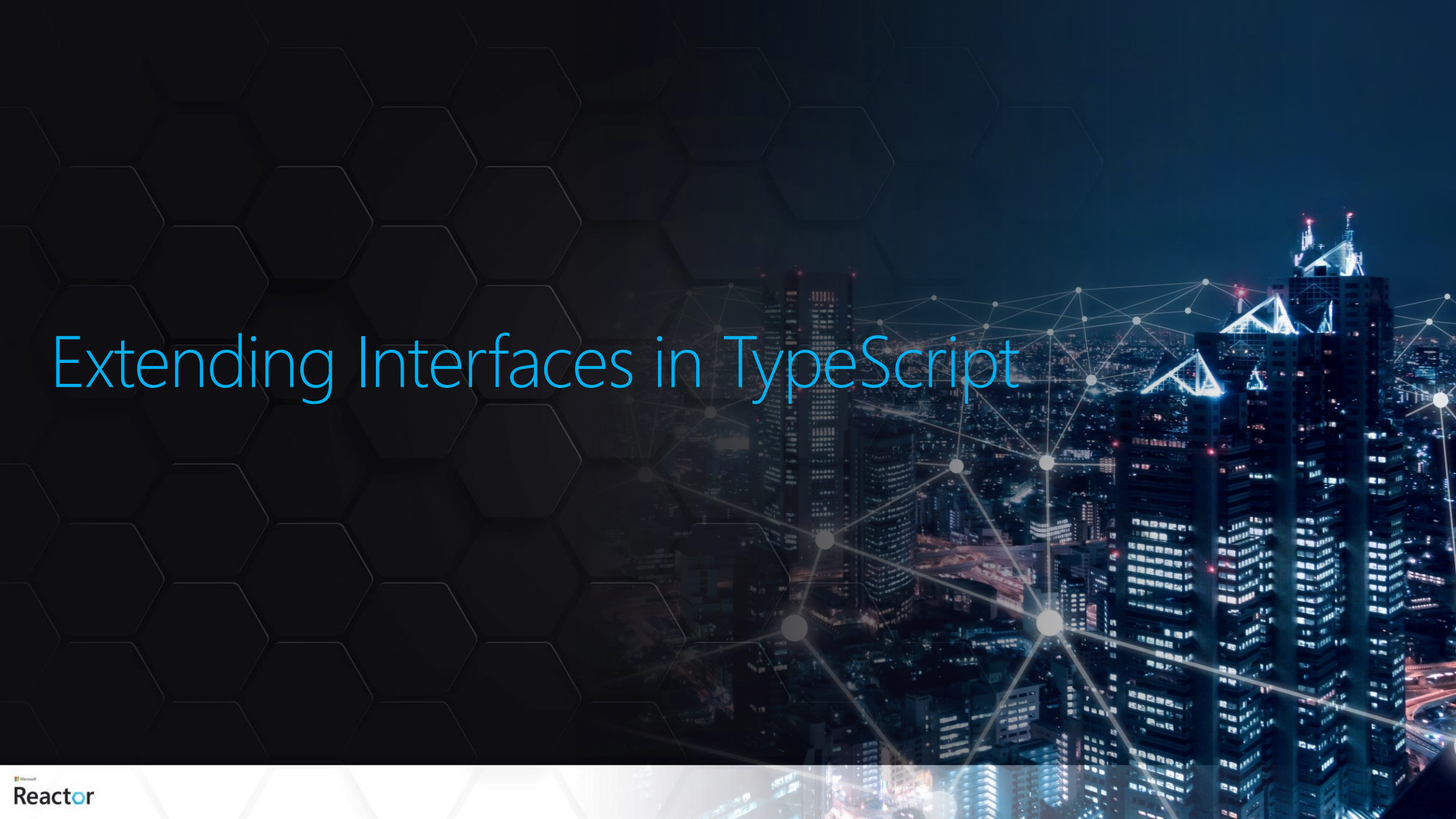
```
type Employee = {
    firstName: string;
    lastName: string;
    fullName(): string;
}
```

- Key difference is that interfaces are extendable

Property Options

- Interface properties in TypeScript can be:
  - Required – default unless specified otherwise
  - Optional – prevents error in type system if omitted; uses a ?
  - Read only – only modifiable on initial object creation

Extending Interfaces in TypeScript



Extending Interfaces

- In TypeScript, an interface can extend 1 or more other interfaces
- Allows copy of members from one interface into another
- Can be used to compose a more sophisticated object hierarchy
- Must implement all required properties from all extended interfaces

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

Using Interfaces in TypeScript

Additional Uses

- In addition to standard usage, can use interfaces in other ways as well:
  - Creating indexable types
  - Describing an API

Creating Indexable Types

- Interfaces can be used to create custom array type that is indexable
- Defines an index signature
- Describe type used to index and type returned from indexing

Creating Indexable Types

- Type is reusable but lacks regular array properties (unless exposed)
- Also, supports use of non-numeric index
- Enables assignment using key-value pairs (like an object)

Describing an API

- Interface can be used to define the shape of an API request/response
- If structure matches JSON format, can map automatically
- Helps bring readability and contract enforcement to API integration

Knowledge Check

Summary

What Did We Cover?

- Interfaces in TypeScript
- Extending interfaces in TypeScript
- Additional uses for interfaces in TypeScript

Next Steps / Further Study

- https://aka.ms/InterfacesTypeScript
- https://www.typescriptlang.org/play#example/structural-typing
- https://www.logicbig.com/tutorials/misc/typescript/indexable-types.html

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 13091 at the start of survey

Microsoft