





Ben Hoelting

In truth, he's just a big kid. He loves designing systems that solve real world problems. There is nothing more satisfying than seeing something you helped develop being used by the end users. Ben is also involved in the technology community and runs the South Colorado .NET user group. He also enjoys speaking at tech groups and events around the country.





Agenda

- TypeScript Defined
- What's wrong with JavaScript
- What's right with TypeScript
- TypeScript Basics
- TypeScript Beyond the Basics
- TypeScript Demo



Atwood's Law:

"Everything that can be written in JavaScript, will eventually be written in JavaScript"



TypeScript Defined:

"A coding language that compiles to JavaScript and provides strong typing and other modern coding language features"



What's Wrong with JavaScript



Issues with JavaScript

- Dynamic Typing Leads to Run Time Errors
- Lack of Maintainability, Scalability
- Fine for Small Apps
- Takes Effort to Write and Learn how to Write Clean Code
- Client Side JavaScript Coding is Different from what C# Developers are used to.



What's Right with TypeScript



TypeScript Rocks!

- "Compiles" to JavaScript
- Provides Intent at Design Time
- Any Browser, Any Host (Node.js), Any OS
- Open Source + Tooling Support
- Static Typed
- Provides Encapsulation
 - Interfaces
 - Classes
 - Modules and Namespaces



TypeScript Rocks More!

- Implementing ECMAScript 6 Standards now!
- TypeScript + Angular 2
- Type Definition Files
 - https://github.com/borisyankov/DefinitelyTyped



TypeScript Basics



The Basics - Ambient Declares

• "declare" Keyword

```
declare var document;
document.title = "Hello";
```



The Basics - Types Definitions

• ":" To Force a Type Definition

```
var MakePoint: () => {
    x: number; y: number;
};
```



The Basics – Function Signatures

```
function vote(candidate: string, callback: (result: string) => any) {
    // ...
}

vote("BigPig",
    function(result: string) {
        if (result === "BigPig") {
            // ...
        }
     }
}
```



TypeScript Classes



Interfaces

```
interface Friend {
    name: string;
    favoriteColor?: string;
}

function add(friend: Friend) {
    var name = friend.name;
}
```

Visual Studio LIVE

Classes

```
class BankAccount {
    constructor(public balance: number) {
         deposit(credit: number) {
             this.balance += credit;
             return this.balance;
        }
}

class CheckingAccount extends BankAccount {
    constructor(balance: number) {
             super(balance);
        }
        writeCheck(debit: number) {
             this.balance -= debit;
        }
}
```



TypeScript Beyond The Basic



Namespaces and Modules

- Namespaces are used for internal modules.
- Modules are used for external modules and require Node.js (CommonJS), require.js (AMD), isomorphic (UMD), SystemJS, or ECMAScript 2015 native modules (ES6) module-loading systems.



Configuration

- tsconfig.json defines compiler options.
- This file is used when the compiler(tsc) is called with out params.

```
"compilerOptions":
    "module": "system",
    "noImplicitAny": true,
    "reserveComments": true,
    "preserveConstEnums": true,
    "outFile": ".././built/local/tsc.js",
    "sourceMap": true
},
"include": [
    "arc/**/*"
],
"exclude": [
    "node modules",
    "**/*.spec.ts"
]
}
```

Visual Studio LIVE

Demo



Generics

 Why Generics when JavaScript is already a dynamic language? Can't we just use JavaScript any?



Demo



Type Definitions

- These are like DLLs for your favorite JavaScript libraries.
- Pretty much every library has one. Check for sure at https://aka.ms/types
- Installed with NPM
 - npm install –save @types/jquery
- Import them like any module
 - import * as \$ from "jquery"



Demo



TypeScript And Angular



AngularJS Uses TypeScript Heavily

Google adopted TS while working on version 2.
 They now support ES2016/5 but they still use mostly TS.

```
src/app/hero-list.component.ts (class)

export class HeroListComponent implements OnInit {
    heroes: Hero[]:
    selectedHero: Hero;

constructor(private service: HeroService) { }

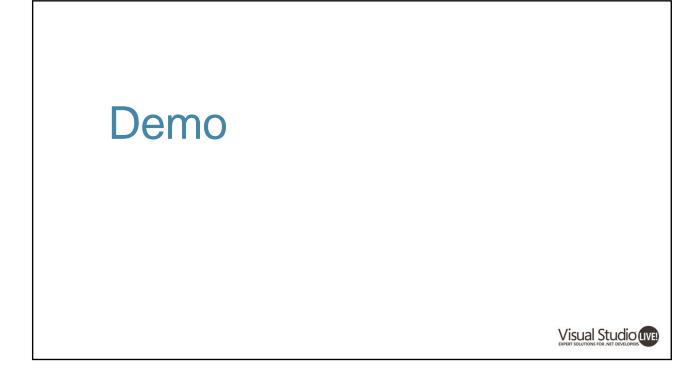
    ngonInit() {
    this.heroes = this.service.getHeroes();
    }

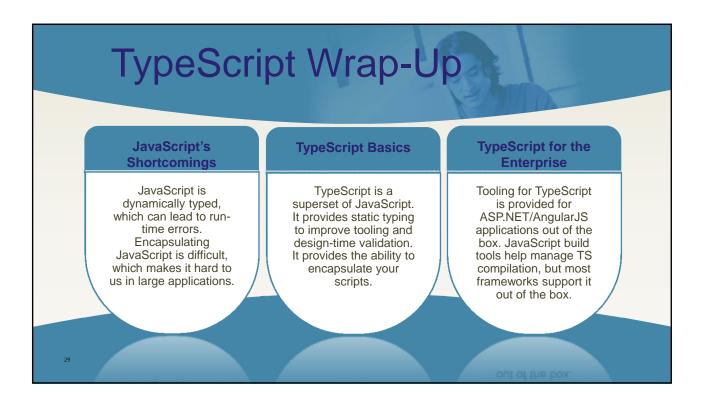
    selectHero(hero: Hero) { this.selectedHero = hero; }
}

Visual Studio

UVE

EMPRI SOLUTIONS FOR ART DIVILOPSES
```





Resources

- http://www.typescriptlang.org/docs/home.html
- Pluralsight Course
- https://github.com/Microsoft/TypeScript
- https://github.com/Microsoft/TypeScriptSamples
- https://angular.io/tutorial
- http://visualstudiomagazine.com/blogs/vs-livevideo/2013/05/typescript-video.aspx
- http://vslive.com/blogs/live-fromvslive/2013/03/typescript.aspx







Ben Hoelting

In truth, he's just a big kid. He loves designing systems that solve real world problems. There is nothing more satisfying than seeing something you helped develop being used by the end users. Ben is also involved in the technology community and runs the South Colorado .NET user group. He also enjoys speaking at tech groups and events around the country.





Ben Hoelting @benhnet b.hoelting@aspenware.com

