



LIDNUG

Typescript for the C# developer

AKA: LESSONS Learned on the FrontClient Line



Peter "Shawty" Shaw

Who am I

- Microsoft UK community leader
- Lidnug group manager (UK & Europe)
- Independent Systems Consultant
- Original 80's UK back bedroom programmer
- Established Technical Author
- 30+ years in I.T covering everything from radio communications to enterprise software development

What's this talk about?

- I'll be telling you what TypeScript is and giving you a few small code samples.
- I'll be explaining to you why TypeScript is important in today's developer environment.
- I'll be showing you some TypeScript code and explaining its similarities to C#

What this talk is not about?

- I won't be teaching "TypeScript" as a language
- I won't be showing direct conversions from C# to TypeScript or how to convert projects.
- If you are already an established TS developer you probably won't benefit much from this talk.
- **But first.....**

Please Note:

- **I am (Not) Partially Deaf.....**
- If you ask a question, and I don't respond please don't be offended, unless your very close to me, and speaking fairly loud, it's very likely I won't hear you.
- My hearing aids are directional and room acoustics can effect them drastically, so there is still a chance I won't hear you, I'm sure if we need to we can get the room monitor to wave at me to get my attention.

If you want to contact me...

- Via Linked-in:
 - <https://www.linkedin.com/in/petershaw08>
- My Blog:
 - <http://shawtyds.wordpress.com/>
- Via E-Mail:
 - lidnug@shawty.me.uk
 - shawty.d.ds@gmail.com
- Via Twitter:
 - [@shawty_ds](https://twitter.com/shawty_ds)

Would You Like To Play Along?

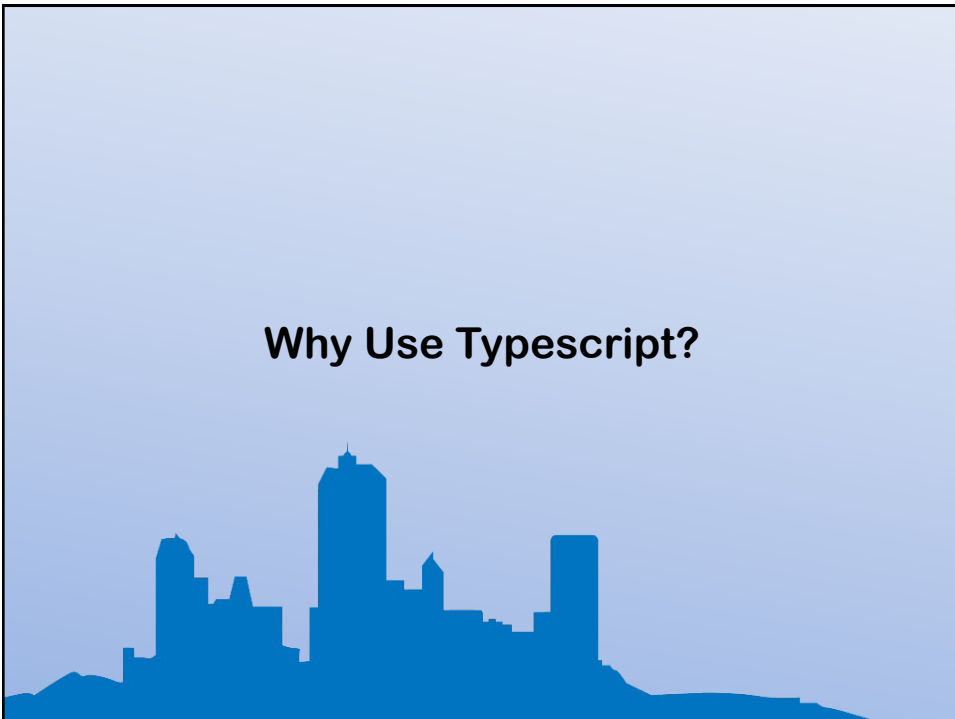
If you go to the typescript website, you'll find an online playground.

<http://www.typescriptlang.org/play/>

Code entered in the left pane, will be converted to JavaScript in real time and displayed in the right pane, complete with syntax checking.

TypeScript

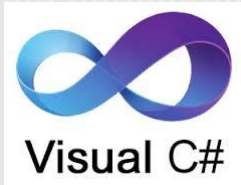
Why Use Typescript?



Why Use Typescript?

To Answer that question, we first have to ask another...

As A C#/Backend developer, why should I even care about Frontend/Client development?



Why Go Frontend?

- More and more code these days is based in the browser.
- JavaScript is “Eating the World”, it’s everywhere.
- Apps increasingly need to be delivered online, and are expected to work offline.
- Backend is becoming more and more API orientated.....

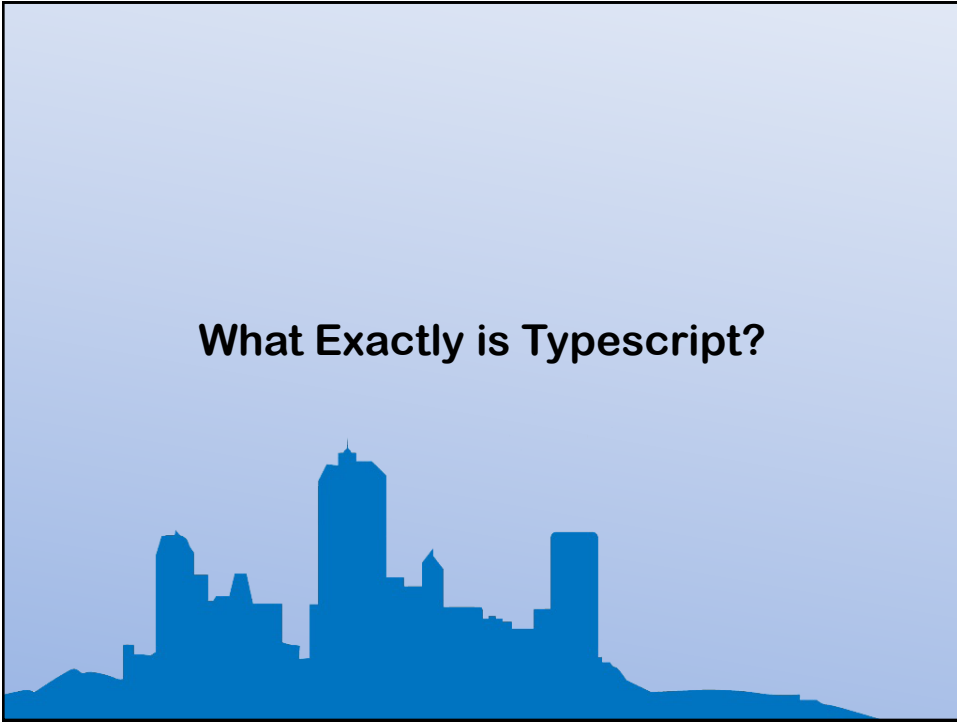
Why Use Typescript?

- Typescript was developed by Anders Hejlsberg's team at Microsoft, Anders is a veteran of backend languages.
- Typescript was designed with the object orientated developer in mind.
- It produces plain old JavaScript that runs on anything that regular JS code runs on.

Why Use Typescript?

- It's amazingly productive once you get into the swing of things.
- It's syntax is very similar to that of C#
- There's a huge community built around it.
- It has a zero cost, to get started
- You can use ANY tooling you want to work with it.

What Exactly is Typescript?



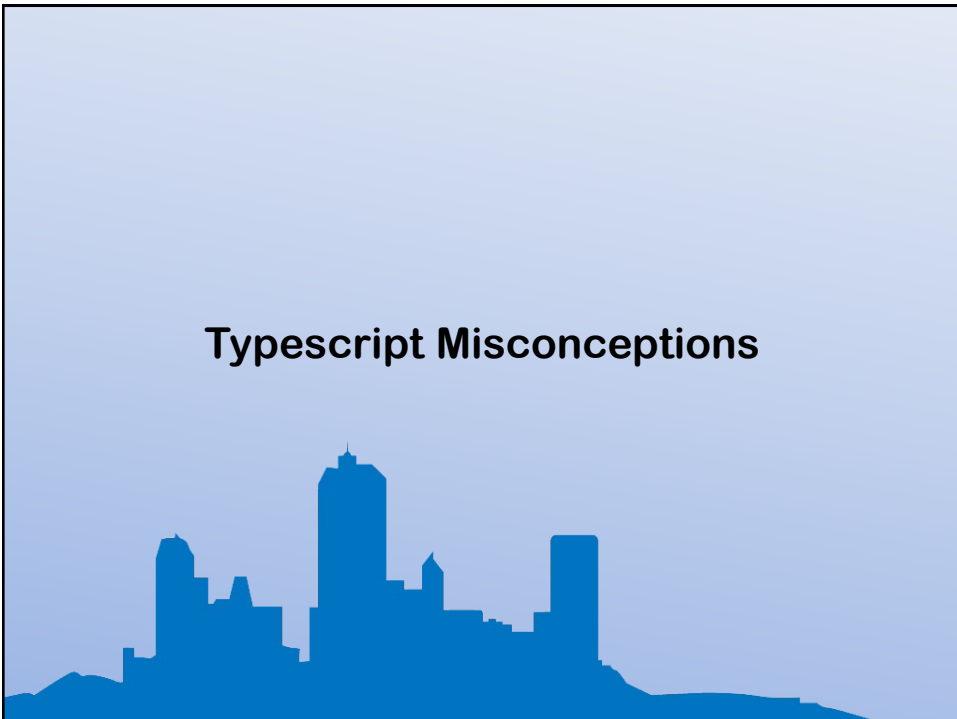
What exactly is Typescript?

- JavaScript === TypeScript (or to put it another way, JavaScript is TypeScript)
- People call TypeScript a compiler, however it's more of a "pre-processor"
- TypeScript in Visual Studio is a Language Provider Extension, in other environments it's a command line binary.

What exactly is Typescript?

- TypeScript is an ES5/6/7 transpiler (This is one of it's best features...)
- More than anything else in these bullet points however
- TypeScript is a "Safety Net" that let's you write enterprise scale JS apps without shooting yourself in the foot.

Typescript Misconceptions



Typescript Misconceptions

- TypeScript is SLOW
- **UNTRUE** – TypeScript produces standard JavaScript code, if the code it produces is slow, it's because the developer wrote poor TypeScript code in the first place.

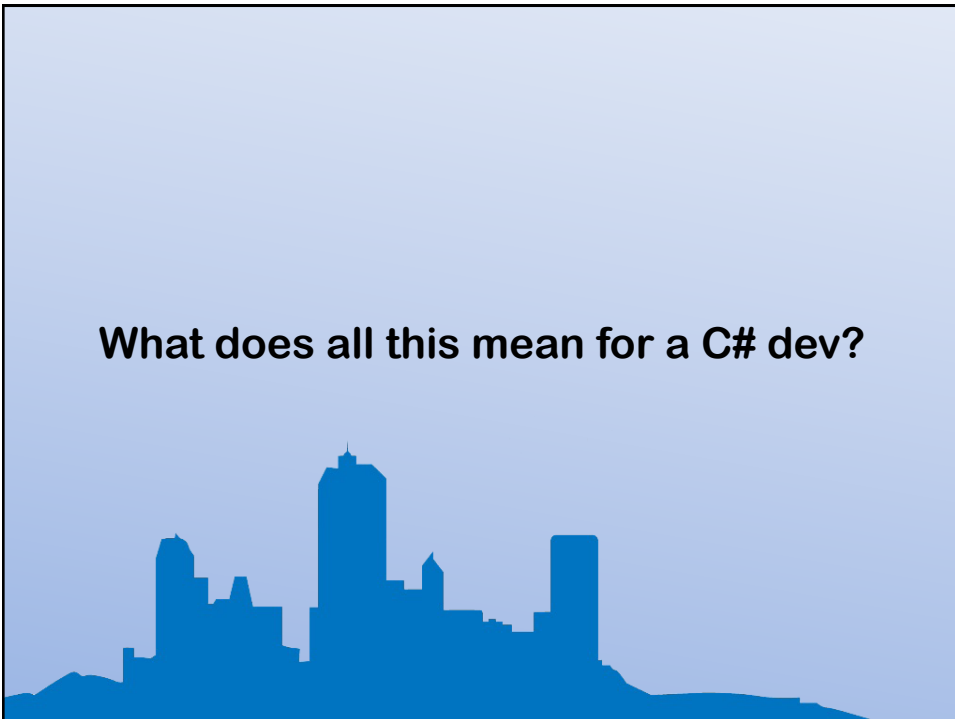
Typescript Misconceptions

- TypeScript requires expensive Microsoft tools to use it.
- **UNTRUE** – TypeScript does come shipped with most versions of Visual Studio, but as I've mentioned it's also available to install from NPM using standard JavaScript tools and editors you may already be using, meaning you can use it on Linux & Mac too.

Typescript Misconceptions

- You have to learn a completely new language to use TypeScript
 - **UNTRUE** – Any developer who's serious about using JavaScript these days, should be using Modern JS syntax anyway, and since TS embraces Modern JS, it's a match you should explore.
- you can include regular JS code in a TypeScript file and it will still compile/transpile.

What does all this mean for a C# dev?



What does this mean for a C# dev?

- It means your journey to Frontend dev is to be honest relatively painless, and quite straight forward.
- It means you can start writing browser and node.js based code while staying in reasonably familiar surroundings.
- There are even frameworks and toolkits that look and feel like desktop WinForms development.

What does this mean for a C# dev?

- Q - With Blazor being released, doesn't this mean TypeScript is no longer needed, if we can now do C# directly in the browser?
- Answer – Nope not at all, with Blazor you still need some JS to interface to browser API's and TypeScript is a perfect fit to help build those interfaces.

What does this mean for a C# dev?

- It means you can leverage existing development skills.
- It means you can challenge your OSS & JS peers, and advertise yourself as a “Full Stack Developer”
- It means you can spend more time developing and less time worrying about learning new things.

Types

AKA: Where/Why TypeScript started in the first place



Types

- TypeScript originally started as a way to strongly type JavaScript.
- TypeScript can often figure out what your doing and WILL enforce type safety
- There are 3 basic types provided
- ~~Beware of 'null' and 'undefined'~~ See the next slide :-)
- You can easily build your own types

The MADNESS that is null vs undefined

- I personally never use 'undefined'
- I try hard NOT to use 'null' and use it as rarely as I have to if I do have to use it
- If I do have to use either of them, I DO NOT use them in decision logic, I use simple truthy/falsy checks instead.
- Where possible I pre allocate variable contents similar to how you might do so in C or C++

The MADNESS that is null vs undefined

- <https://basarat.gitbooks.io/typescript/docs/javascript/null-undefined.html>
- <https://github.com/Microsoft/TypeScript/wiki/Coding-guidelines#null-and-undefined>
- <https://howtodoinjava.com/typescript/undefined-vs-null/>
- <https://jameshenry.blog/typescript-null-and-undefined-types/>

Defining Variable Types

- In C# types are typically defined before the variable name:
 - public `string` foo { get; set; }
 - `string` bar = "abc";
- In typescript the type comes after the variable name, separated by a colon:
 - public foo:`string`;
 - let bar:`string` = "abc";

Union Types

- TypeGuards – or the “or’ing” of types. (**Now known as ‘Union Types’**)

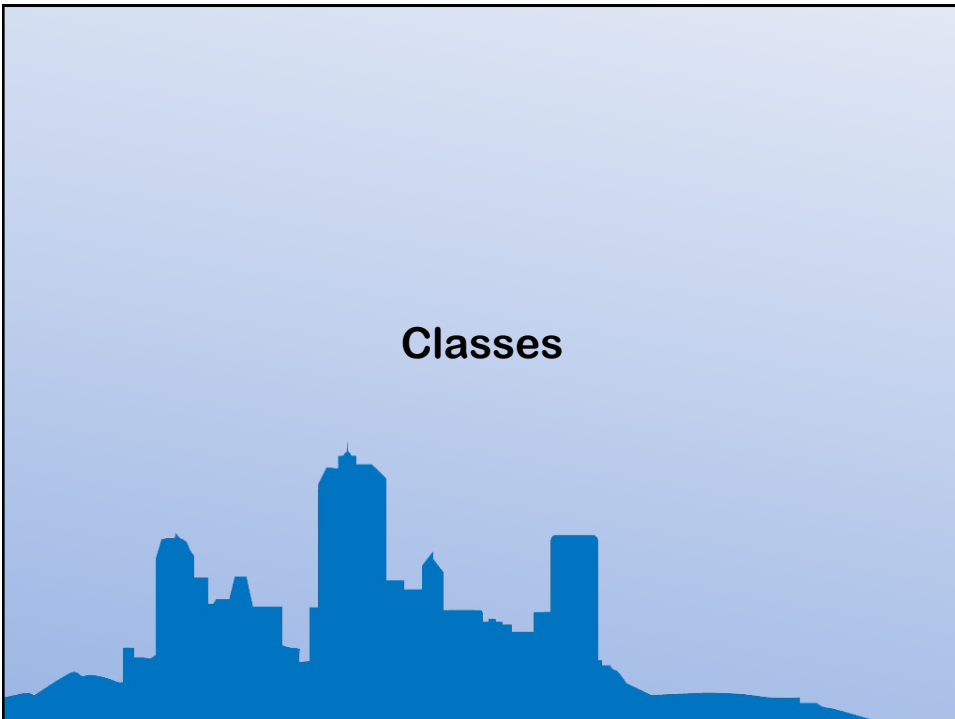
```
let myVar : string | number;
```

```
myVar = "Hello"; // Works
```

```
myVar = 2; // Works
```

```
myVar = true; // Fails
```

Classes



Classes

- In C# classes are usually declared in the following manner:

```
public class test
{
    public string foo { get; set; }
}
```

- In typescript a class is declared in exactly the same way but without the modifier:

```
class test (or 'export class test')
{
    public foo:string = "";
}
```

Classes

- C# allows you to extend classes:

```
public class test::classToExtend
{}
```

- In typescript a class is extend using the extend keyword:

```
class test extends classToExtend
{}
```


Classes

- TypeScript classes can also have a constructor, using the constructor keyword :

```
class test
{
  constructor()
  {}
}
```

Classes

- If your extending a class you **MUST** call super in your constructor to call the base constructor otherwise you'll get a build error:

```
class test extends foo
{
  constructor()
  {
    super();
  }
}
```

Classes

- Property access must be performed using `'this'`, there is no default. This applies to all local AND abstracted members (Data and functions)
- TypeScript also has something similar to .NET namespaces. These are called modules, and the syntax is as shown in the next slide:

Classes

```
module myNameSpace
{
  export class test
  {
  }
}
```

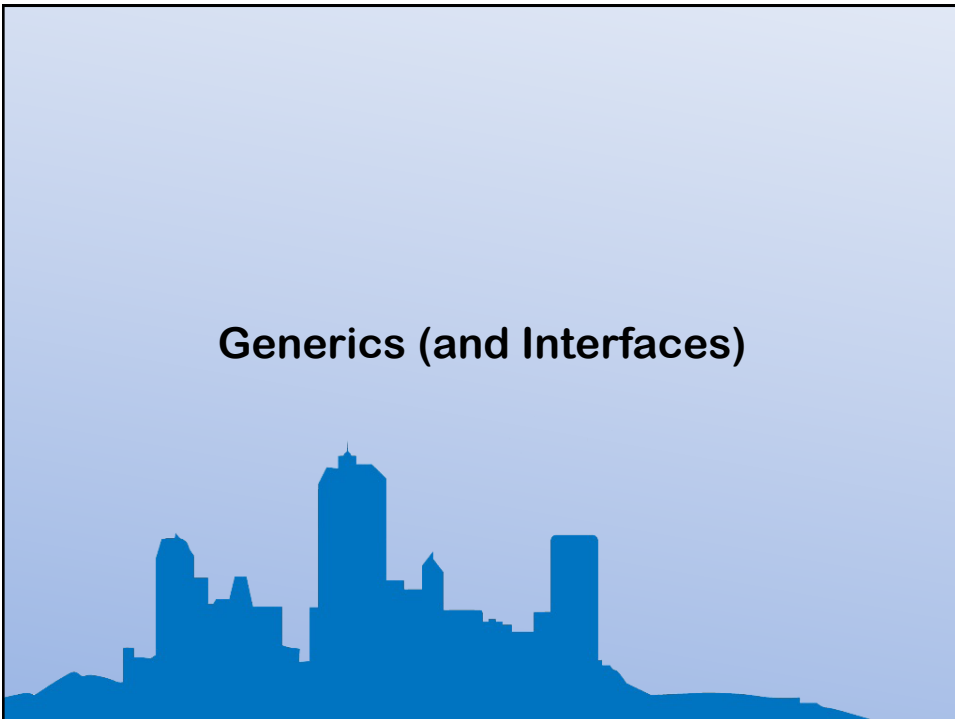
To Access or use this class you would use `'myNameSpace.test'`, as the export makes it public.

Classes

- Be aware of your module and class name length.
- Unlike C# there is **NO** equivalent to the 'using' statement, so if you have deeply nested classes, you'll end up doing this:

```
Let myClass = new myModule.anotherModule.myClass();
```

Generics (and Interfaces)



Generics

- TypeScript has interfaces, they work the same way as C# interfaces, the syntax is the same too, except for the TS class naming.
- TypeScript has generics, and the syntax is very similar to that of C# but not quite identical
- To use generics, you need to use TypeScript's interface specification in order to define the type to use.
- Once the interface is defined, you can then use it as you would expect, in a similar way to C#

Generics

```
interface myGenericsType<T>
{
  myMethod(someParameter: string): T;
}

class myGenericsClass implements myGenericsType<string>
{
  myMethod(someParameter: string)
  {
    return "[" + someParameter + "];";
  }
}

let testClass = new myGenericsClass();
alert(testClass.myMethod());
```

ES6 The Next Generation



ES6 The Next Generation

- Many of the newer ES6 features have been implemented in TypeScript.
- I'm not going to go through all of them (There's far too many), there's a great compatibility chart available here:

<http://kangax.github.io/compat-table/es6/>

ES6 (Highlights)

- **Arrow Functions (Lambdas)**

```
class Person{
  constructor(public age: number) {
  }
  growOld = (amount) => { this.age = this.age + amount; }
}

setTimeout(() => {
  alert("I will show after 5 seconds...");
}, 5000);
```

ES6 (Highlights)

- **Default Parameters**

```
class myClass{
  public doSomething(text: string, aFlag = false) {
    if (!aFlag) {
      alert(text);
    } else {
      alert("Not allowed to display Text as flag is set");
    }
  }
}

let aThing = new myClass();
aThing.doSomething("Hello World");
aThing.doSomething("Hello World", true);
```

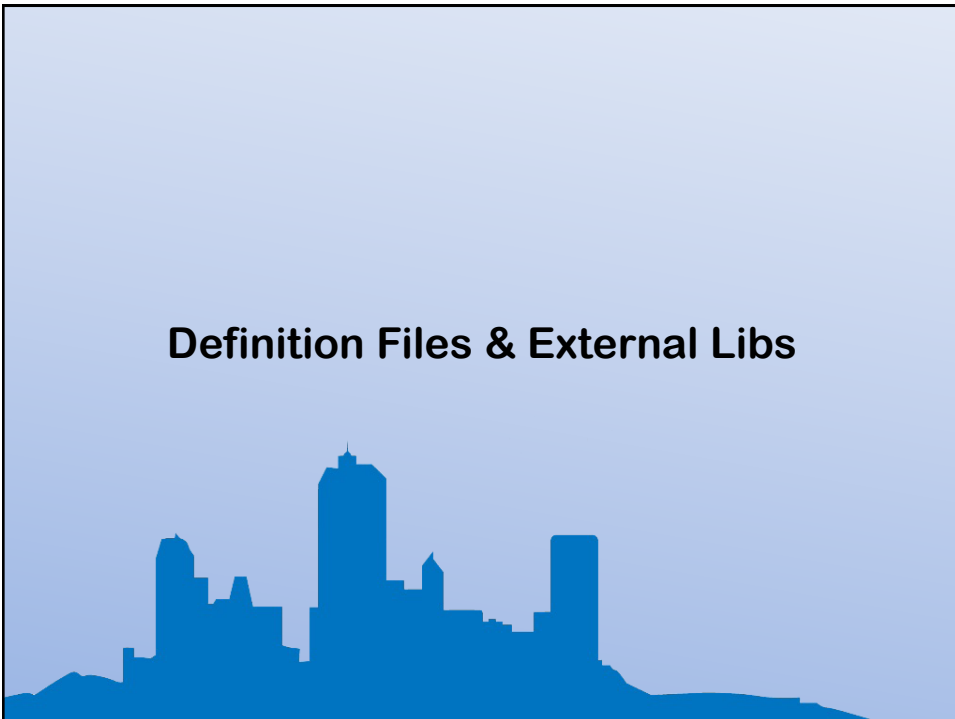
ES6 (Highlights)

- **Optional Parameters**

```
class myClass{
  public doSomething(text: string, aFlag?: boolean) {
    if (!aFlag) { // the check now becomes truthy rather than bool
      alert(text);
    } else {
      alert("Not allowed to display Text as flag is set");
    }
  }
}

Let aThing = new myClass;
aThing.doSomething("Hello World");
aThing.doSomething("Hello World", true);
```

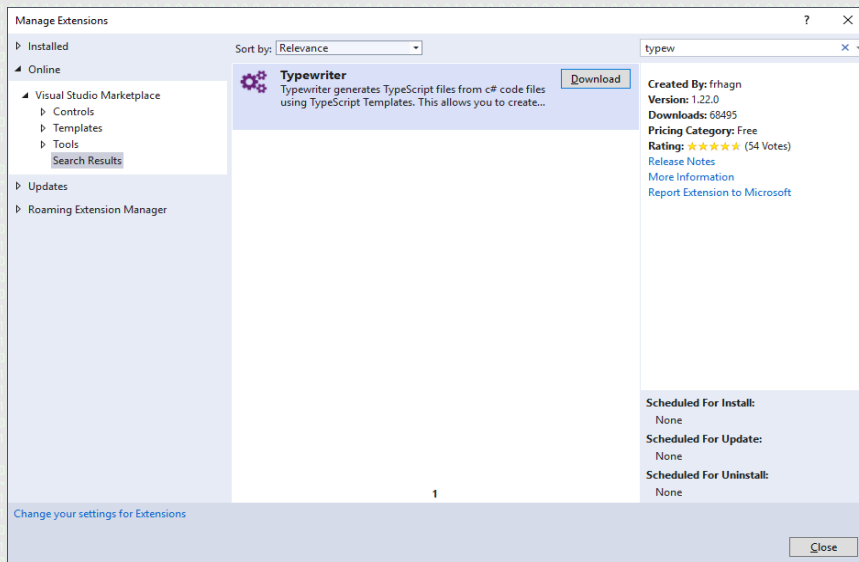
Definition Files & External Libs



Definition Files & External Libs

- TypeScript can use files with a '.d.ts' extension, these files will have a special meaning
- You can use existing JavaScript libraries, by creating a definition file to describe to TypeScript what the JS lib looks like.

Definition Files & External Libs



Definition Files & External Libs

- Many of the most common libraries in use today, have already been created and are available from:

<http://definitelytyped.org/>

```
npm install tsd
```

```
npm install typings
```

```
npm install @types/<package name>
```

Definition Files & External Libs

- If your using Visual Studio, you can also find them in NuGet using 'tag:typescript'
- If your using web essentials, right clicking on a JS file in your project and left clicking on 'Search for Typings'

Summary

Summary

- TypeScript is very C# developer friendly.
- TypeScript removes a vast amount of the pain associated with JavaScript development.
- It costs nothing to get started, esp if you already have Visual Studio

Summary

- TypeScript is a route to modern and future JavaScript standards, a simple configuration or command line parameter change, allows you to change from ES3/ES5 to ES6, while still writing everything in ES6

Summary

- TypeScript has a fantastic community supporting it.
- Even the “Angular” team have decided to support it by writing Anugular2 onwards completely in TypeScript
- The Aurelia team support it by writing all of the Aurelia framework using it.

Summary

- TypeScript is supported in ALL the tools your already using.
- There is NO easier way for a back-end C# dev to make the move into front-end development.

Thanks for Listening

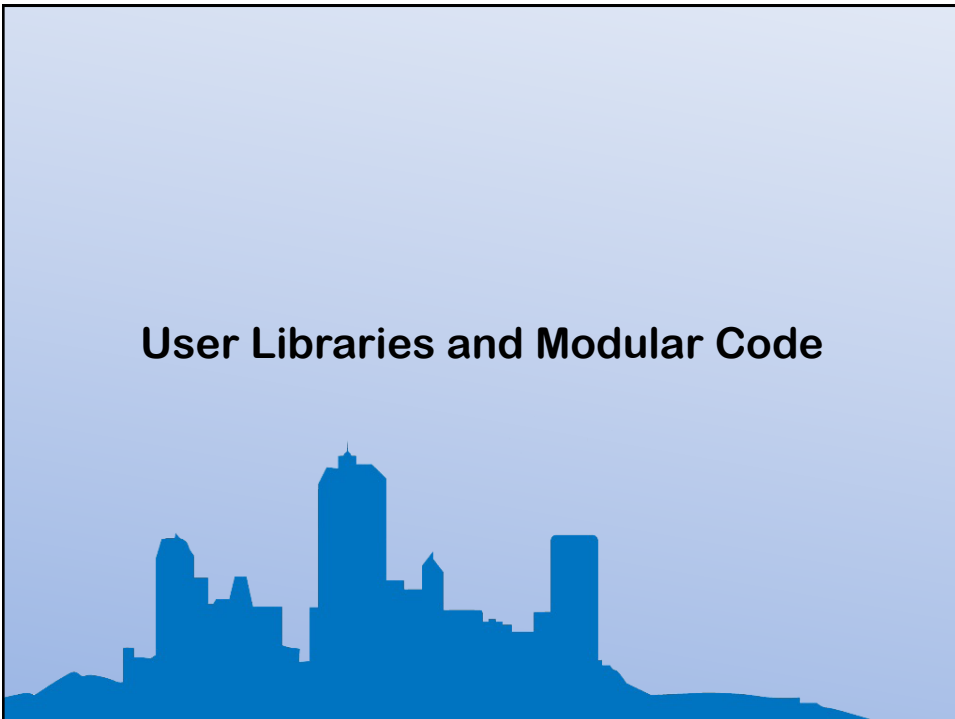
If your still awake, you did better than me, the person you see on stage now is a remote controlled robot, powered by TypeScript ;-)



If you want to contact me...

- Via Linked-in:
 - <https://www.linkedin.com/in/petershaw08>
- My Blog:
 - <http://shawtyds.wordpress.com/>
- Via E-Mail:
 - lidnug@shawty.me.uk
 - shawty.d.ds@gmail.com
- Via Twitter:
 - [@shawty_ds](https://twitter.com/shawty_ds)
- A copy of the slides for the session are available as a PDF at:
 - <http://files.digital-solutions.me.uk/presentations/typescript.pdf>

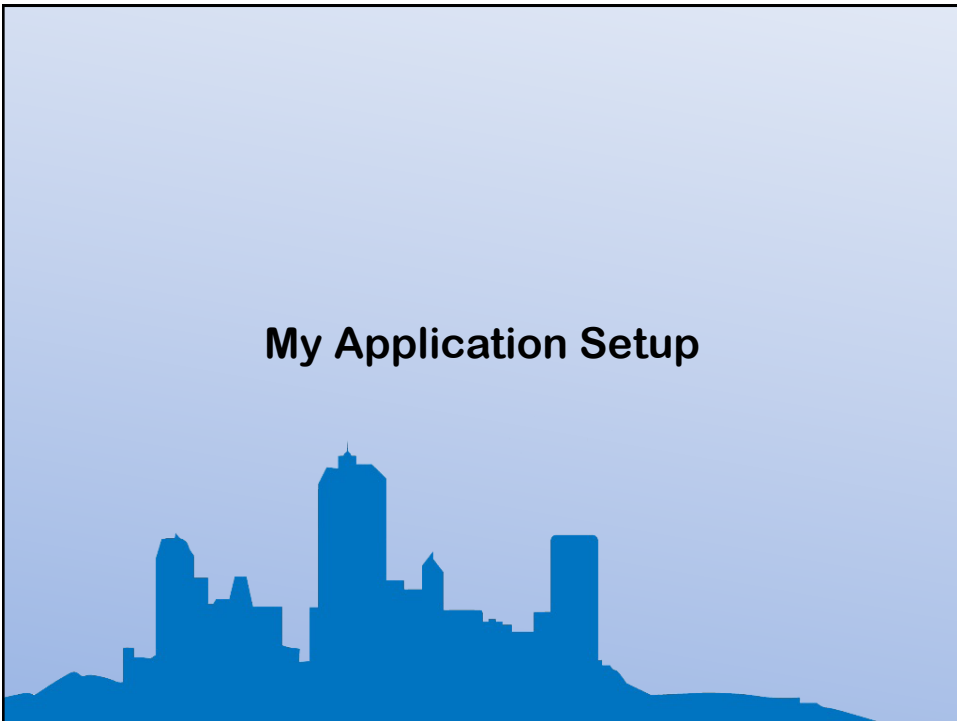
User Libraries and Modular Code



Beware of the optimizer



My Application Setup



My Application Setup

- During this talk, I'll be showing you some of the things I've learned using an application set up and framework, I've built for my own work with TypeScript.
- You don't have to use it, but a lot of the lessons I've learned doing typescript are based on this framework.

My Application Setup

- If you want a copy of it, you can download an earlier version of it (Which I used in a Talk for .NET Notts). It's available on my Git-Hub page at:
 - <https://github.com/shawty/dotnetnotts15>

It will be worth keeping an eye on this project, as I will get round to updating the code soon, with the more up to date version that I use in production.