

TypeScript

JaxDUG May 2013

About Us

- David Fekke
- Jyoti Chawla
- Software Engineers





Javascript
is highly unstructured
and highly functional

- [] + [];

- [] + [];

- => “”

- $[] + [];$

- $\Rightarrow " "$

- $[] + \{\};$

- `[] + [];`
 - `=> “”`
- `[] + {};`
 - `=> [object Object]`

- `[] + [];`
 - `=> “”`
- `[] + {};`
 - `=> [object Object]`
- `{ } + [];`

- `[] + [];`
 - $\Rightarrow \text{""}$
- `[] + {};`
 - $\Rightarrow [\text{object Object}]$
- `{} + [];`
 - $\Rightarrow 0$

- `[] + [];`
 - $\Rightarrow \text{""}$
- `[] + {};`
 - $\Rightarrow [\text{object Object}]$
- `{} + [];`
 - $\Rightarrow 0$
- `{} + {};`

- `[] + [];`
 - $\Rightarrow \text{""}$
- `[] + {};`
 - $\Rightarrow [\text{object Object}]$
- `{} + [];`
 - $\Rightarrow 0$
- `{} + {};`
 - $\Rightarrow \text{NaN}$



- `false == ""`

- `false == ""`

- `=> true`

- `false == ""`
 - \Rightarrow true
- `false === ""`

- `false == ""`
 - \Rightarrow true
- `false === ""`
 - \Rightarrow false

- `false == ""`
 - \Rightarrow true
- `false === ""`
 - \Rightarrow false
- `| == "|"`

- `false == ""`
 - \Rightarrow true
- `false === ""`
 - \Rightarrow false
- `| == "|"`
 - \Rightarrow true

- `false == ""`
 - \Rightarrow true
- `false === ""`
 - \Rightarrow false
- `| == "|"`
 - \Rightarrow true
- `| === "|"`

- `false == ""`
 - \Rightarrow true
- `false === ""`
 - \Rightarrow false
- `| == "|"`
 - \Rightarrow true
- `| === "|"`
 - \Rightarrow false

Is Javascript an Object-Oriented language?

What is TypeScript

TypeScript is a programming language that is a superset of JavaScript. It adds optional static typing to the language, allowing developers to catch errors at compile time instead of runtime. TypeScript is designed to be used with Node.js and can also be compiled to plain JavaScript for use in web browsers.

TypeScript's type system includes support for interfaces, classes, and modules, making it easier to write large-scale applications. It also includes features like function overloads, type guards, and type inference, which make it easier to work with complex data structures.

TypeScript is a popular choice for building large-scale web applications, particularly those that require a high level of code quality and maintainability. It is also used in many other domains, such as mobile development and server-side programming.

TypeScript is a powerful tool for modern web development, providing a way to write clean, maintainable code that is both safe and efficient. Whether you're a beginner or an experienced developer, TypeScript is definitely worth learning.

What is TypeScript

- Based on JavaScript

What is TypeScript

- Based on JavaScript
- Adds a Type system to JavaScript

What is TypeScript

- Based on JavaScript
- Adds a Type system to JavaScript
- Makes Javascript Structured

What is TypeScript

- Based on JavaScript
- Adds a Type system to JavaScript
- Makes Javascript Structured
- Can write trully OO Javascript

Transpiler

- Compiles down Javascript
- Dart and CoffeeScript both transpile to javascript

What is the Use Case

Use cases are a way to define what a system does.

Use cases are a way to define what a system does.

Use cases are a way to define what a system does.

Use cases are a way to define what a system does.

Use cases are a way to define what a system does.

Use cases are a way to define what a system does.

Use cases are a way to define what a system does.

Use cases are a way to define what a system does.

Use cases are a way to define what a system does.

Use cases are a way to define what a system does.

Use cases are a way to define what a system does.

Use cases are a way to define what a system does.

Use cases are a way to define what a system does.

Use cases are a way to define what a system does.

Use cases are a way to define what a system does.

Use cases are a way to define what a system does.

Use cases are a way to define what a system does.

What is the Use Case

- Type systems help us develop large structured applications

What is the Use Case

- Type systems help us develop large structured applications
- Single Page Apps

What is the Use Case

- Type systems help us develop large structured applications
- Single Page Apps
- Node JS Apps

What is the Use Case

- Type systems help us develop large structured applications
- Single Page Apps
- Node JS Apps
- Windows Store Apps

Compiler



Compiler

- Compiler written in TypeScript

Compiler

- Compiler written in TypeScript
- Runs in Node.js & Visual Studio 2012

Compiler

- Compiler written in TypeScript
- Runs in Node.js & Visual Studio 2012
- myapp.ts is TypeScript file

Compiler

- Compiler written in TypeScript
- Runs in Node.js & Visual Studio 2012
- myapp.ts is TypeScript file
- Compiles into myapp.js

Demo

- Compiler

TypeScript features



TypeScript features

- Parameter types

TypeScript features

- Parameter types
- Return types

TypeScript features

- Parameter types
- Return types
- Interfaces

TypeScript features

- Parameter types
- Return types
- Interfaces
- Classes

TypeScript features

- Parameter types
- Return types
- Interfaces
- Classes
- Modules

```
1 var Student = (function () {
2     function Student(firstname, middleinitial, lastname) {
3         this.firstname = firstname;
4         this.middleinitial = middleinitial;
5         this.lastname = lastname;
6         this.fullname = firstname + " " + middleinitial + " " + lastname;
7     }
8     return Student;
9 })();
10 function greeter(person) {
11     return "Hello, " + person.firstname + " " + person.lastname;
12 }
13 var user = new Student("Jane", "M.", "User");
14 document.body.innerHTML = greeter(user);
15
```

```
1 class Student {  
2     fullname : string;  
3     constructor(public firstname, public middleinitial, public lastname) {  
4         this.fullname = firstname + " " + middleinitial + " " + lastname;  
5     }  
6 }  
7  
8 interface Person {  
9     firstname: string;  
10    lastname: string;  
11 }  
12  
13 function greeter(person : Person) {  
14     return "Hello, " + person.firstname + " " + person.lastname;  
15 }  
16  
17 var user = new Student("Jane", "M.", "User");  
18  
19 document.body.innerHTML = greeter(user);
```

Basic Types



Basic Types

- **string**

Basic Types

- string
- number

Basic Types

- string
- number
- bool

Basic Types

- string
- number
- bool
- any

Basic Types

- string
- number
- bool
- any
- null

Basic Types

- string
- number
- bool
- any
- null
- undefined

Parameters Types



Parameters Types

- Primitives and user defined

Parameters Types

- Primitives and user defined
- string, number, bool or any.

Parameters Types

- Primitives and user defined
- string, number, bool or any.
- aFunc (b: string, c: number) {}

Parameters Types

- Primitives and user defined
- string, number, bool or any.
- aFunc (b: string, c: number) {}
- accept a function

Parameters Types

- Primitives and user defined
- string, number, bool or any.
- aFunc (b: string, c: number) {}
- accept a function
- aFunc (handler: (res: string) => any) {}

Return Types



Return Types

- Return primitives or user defined types

Return Types

- Return primitives or user defined types
- `function a(): string { return “”; }`

Interfaces



Interfaces

- Similar to interfaces in Java and C#

Interfaces

- Similar to interfaces in Java and C#
- interface IPerson { A: string; B?: string; }

Classes



Classes

- Use “class” keyword

Classes

- Use “class” keyword
- Can “extend” class or “implement” interfaces

Classes

- Use “class” keyword
- Can “extend” class or “implement” interfaces
- Allows encapsulation

Classes

- Use “class” keyword
- Can “extend” class or “implement” interfaces
- Allows encapsulation
- Use “constructor” as constructor function

Classes

- Use “class” keyword
- Can “extend” class or “implement” interfaces
- Allows encapsulation
- Use “constructor” as constructor function
- “super” to call base functions and properties

Modules



Modules

- Modules are like packages or libraries

Modules

- Modules are like packages or libraries
- Use “export” keyword to make members available

Modules

- Modules are like packages or libraries
- Use “`export`” keyword to make members available
- `module a { export b():string { return “”; } }`

File types



File types

- *.ts files are typescript files

File types

- *.ts files are **typescript** files
- *.js files are **compiled JavaScript**

File types

- *.ts files are typescript files
- *.js files are compiled JavaScript
- *.d.ts files are interface representations of libraries. Helps visual studio with intellisence

Node compiler



Node compiler

- npm install -g typescript

Node compiler

- `npm install -g typescript`
- `tsc myfile.ts`
generates a `myfile.js`

Node compiler

- `npm install -g typescript`
- `tsc myfile.ts`
generates a `myfile.js`
- “`tsc --declaration myfile.ts`” creates `*.d.ts`

Visual Studio 2012



Visual Studio 2012

- Plugin support

Visual Studio 2012

- Plugin support
- TypeScript project type

Visual Studio 2012

- Plugin support
- TypeScript project type
- Add ts files to existing projects

Visual Studio 2012

- Plugin support
- TypeScript project type
- Add ts files to existing projects
- Full Intellisense

Visual Studio 2012

- Plugin support
- TypeScript project type
- Add ts files to existing projects
- Full Intellisense
- `///<reference path="jquery.d.ts" />`

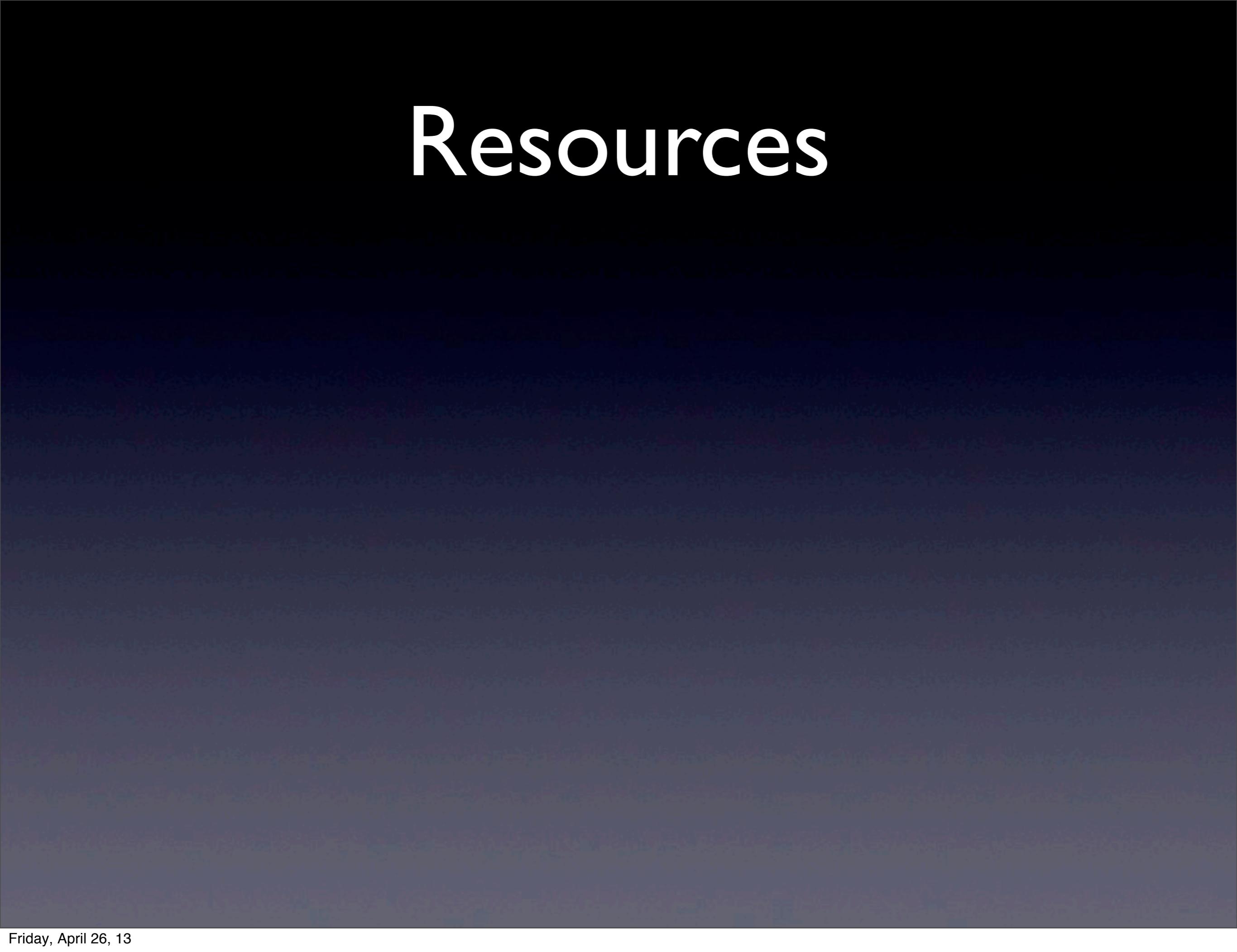
```
<PropertyGroup Condition="'$(Configuration)' == 'Debug'">
  <TypeScriptTarget>ES5</TypeScriptTarget>
  <TypeScriptIncludeComments>true</TypeScriptIncludeComments>
  <TypeScriptSourceMap>true</TypeScriptSourceMap>
</PropertyGroup>
<PropertyGroup Condition="'$(Configuration)' == 'Release'">
  <TypeScriptTarget>ES5</TypeScriptTarget>
  <TypeScriptIncludeComments>false</TypeScriptIncludeComments>
  <TypeScriptSourceMap>false</TypeScriptSourceMap>
</PropertyGroup>
<Import Project="$(MSBuildExtensionsPath32)\Microsoft\VisualStudio\v$(
(VisualStudioVersion)\TypeScript\Microsoft.TypeScript.targets" />
```

Add this to your VS Project, *.csproj

Demo

- TypeScript

Resources



Resources

- typescriptlang.org

Resources

- typescriptlang.org
- <http://github.com/davidfekke/PeopleKODB>

Resources

- typescriptlang.org
- [http://github.com/davidfekke/PeopleKODB](https://github.com/davidfekke/PeopleKODB)
- [http://github.com/davidfekke/tsexamples](https://github.com/davidfekke/tsexamples)

Resources

- typescriptlang.org
- <http://github.com/davidfekke/PeopleKODB>
- <http://github.com/davidfekke/tsexamples>
- Channel 9 interview with Anders Heljsberg

Contact

Contact

- davidfekke at gmail dot com

Contact

- davidfekke at gmail dot com
- veganjoti at gmail dot com

Contact

- [davidfekke at gmail dot com](mailto:davidfekke@gmail.com)
- [veganjoti at gmail dot com](mailto:veganjoti@gmail.com)
- twitter.com/davidfekke

Contact

- [davidfekke at gmail dot com](mailto:davidfekke@gmail.com)
- [veganjoti at gmail dot com](mailto:veganjoti@gmail.com)
- twitter.com/davidfekke
- <http://www.fekke.com/blog/>

Contact

- [davidfekke at gmail dot com](mailto:davidfekke@gmail.com)
- [veganjoti at gmail dot com](mailto:veganjoti@gmail.com)
- twitter.com/davidfekke
- <http://www.fekke.com/blog/>
- github.com/davidfekke