Using TypeScript To Build Better JavaScript Apps

Kurt Wiersma
@kwiersma
kwiersma@mac.com

Who am I?

- From Minneapolis, MN
- Have been developing web apps for over 12 years
- Have used Python, Groovy, Java, C#, ColdFusion
- Favorites: Python/django, C#/MVC, AngularJS, TypeScript

Agenda

- What is TypeScript?
  - Syntax
- Getting Started
- Comparisons
- Why would you want types?
- Working with JavaScript libraries
- Tooling

What is TypeScript?

- TypeScript lets you write JavaScript the way you really want to.
- TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.
- Any browser. Any host. Any OS. Open Source.
- http://typescriptlang.org

Wait, M$ and Open Source?

Are pigs flying now?

TypeScript | Select... | Share | Run | JavaScript

```
1 function greeter(person) {
2     return "Hello, " + person;
3 }
4
5 var user = "Jane User";
6
7 document.body.innerHTML = greeter(user);
8
9
```

```
1 function greeter(person) {
2     return "Hello, " + person;
3 }
4
5 var user = "Jane User";
6
7 document.body.innerHTML = greeter(user);
8
```

JavaScript is Valid TypeScript

TypeScript Syntax

```typescript
/// <reference path='../_all.ts' />

module djleague {

    export class FantasyTeamService {

        public teams: FantasyTeam[];

        private httpService: ng.IHttpService;

        constructor ($http: ng.IHttpService) {
            this.httpService = $http;
        }

        getTeams(): ng.IPromise<FantasyTeam[]> {
            return this.httpService.get('/api/teams')
                .then(function (response) {
                    var data = response.data;
                    this.teams = new Array<FantasyTeam>();

                    for (var i = 0; i < data.length; i++) {
                        var team: FantasyTeam = new FantasyTeam();
                        team.id = data[i].id;
                        team.name = data[i].name;
                        team.draftorder = data[i].draftorder;
                        team.owner = data[i].owner;
                        this.teams.push(team);
                    }

                    return this.teams;
                });
        }

    }
}
```

TypeScript

```javascript
/// <reference path='../_all.ts' />
var djleague;

(function (djleague) {
    var FantasyTeamService = (function () {

        function FantasyTeamService($http) {
            this.httpService = $http;
        }

        FantasyTeamService.prototype.getTeams = function () {
            return this.httpService.get('/api/teams').then(function (response) {
                var data = response.data;
                this.teams = new Array();

                for (var i = 0; i < data.length; i++) {
                    var team = new djleague.FantasyTeam();
                    team.id = data[i].id;
                    team.name = data[i].name;
                    team.draftorder = data[i].draftorder;
                    team.owner = data[i].owner;
                    this.teams.push(team);
                }

                return this.teams;
            });
        };

        return FantasyTeamService;
    })();

    djleague.FantasyTeamService = FantasyTeamService;

})(djleague || (djleague = {}));
```

JavaScript

Features

- Classes
- Modules
- Interfaces
- Generics
- Arrow Functions
- References
- Type Definitions
- Better “this” by default

```
1    /// <reference path='../_all.ts' />
2
3    module djleague {
4
5        export class FantasyTeamService {
6
7            public teams: FantasyTeam[];
8
9            private httpService: ng.IHttpService;
10
11           constructor ($http: ng.IHttpService) {
12               this.httpService = $http;
13           }
14
15           getTeams(): ng.IPromise<FantasyTeam[]> {
16               return this.httpService.get('/api/teams')
17                   .then(function (response) {
18                       var data = response.data;
19                       this.teams = new Array<FantasyTeam>();
20
21                       for (var i = 0; i < data.length; i++) {
22                           var team: FantasyTeam = new FantasyTeam();
23                           team.id = data[i].id;
24                           team.name = data[i].name;
25                           team.draftorder = data[i].draftorder;
26                           team.owner = data[i].owner;
27                           this.teams.push(team);
28                       }
29
30                       return this.teams;
31                   });
32           }
33
34       }
35   }
```

Pros & Cons

Pros

- Syntax much like JS
- Targets ES 6
- Optional Types
- Classes, Generics, Interfaces
- Fixes “this”

Cons

- Compile step
- Debugging (use source maps)
- Another language / tool to learn

Comparisons

	TYPESCRIPT	COFFESCRIPT	DART
COMPILED	X	X	X
TYPED	X		X
WHITE SPACE		X	
JS LIKE SYNTAX	X		1/2
CLASSES	X	X	X
USE JS LIBRARIES	X	X	

Why would you want types?

- Structure for large code bases and/or teams
- Catch errors early
- Provide a structured API
- Tooling can provide better code completion & refactoring

What about existing JavaScript Libraries?

- DefinitelyTyped provides TS definitions for tons of JS libraries
  - JQuery, Angular, Node, Ember, Backbone, ect.
  - http://definitelytyped.org
- You can write you own definitions easily

Custom Definitions

```
1   declare class Pusher {
2       constructor (key: string);
3       subscribe(channelName: string): Channel;
4   }
5
6   interface Channel {
7       bind(eventName: string, callback: Function);
8   }
```

pusher.d.ts

```
102         $scope.PUSHER_ENABLED = true;
103         if ($scope.PUSHER_ENABLED) {
104             this.pusher = new Pusher('');
105             this.channel = this.pusher.subscribe('draftedPlayers');
106             this.channel.bind('playerDrafted', function(data) {
107                 //console.log("playerDraft notification received");
108                 //console.dir(data);
109
110                 $scope.$apply(function(scope) {
111                     $scope.picks = data;
112                     $scope.vm.processPicks();
113                 });
114             });
115         }
```

Useage

Getting Started

- Install:
  - `npm -g typescript`
- Compile:
  - `tsc --sourcemap --out js/Application.js js/_all.ts`

Tooling

- CLI: grunt with grunt-ts
- IDEs:
  - WebStorm / IntelliJ
  - Visual Studio 2012+
  - Eclipse
- Editors:
  - Sublime
  - Atom

Making My App Better

- Gradually moved over my JS to TS
- Added types and better structure to code along the way
- Made working with Angular's API easier to learn

Key Points

- Javascript is valid TypeScript
- TypeScript is following ECMA Script 6
- Types are optional
- TypeScript does NOT force you to do Classes and Interfaces!
- Refactoring! Tooling!

References

- http://gilamran.blogspot.co.il/2013/07/typescript-on-production.html
- http://gilamran.blogspot.com/2014/04/typescript-for-javascripters.html
- http://visualstudiomagazine.com/articles/2013/10/01/calling-web-services-with-typescript.aspx

Questions?

Kurt Wiersma (kwiersma@mac.com)
@kwiersma