Chapter 4. Going further than ES6

4.1. Dynamic, static and optional types

You may have heard that Angular apps can be written in ES5, ES6 or TypeScript. And you may be wondering what TypeScript is, or what it brings to the table.

JavaScript is dynamically typed. That means you can do things like:

```
let pony = 'Rainbow Dash';
pony = 2;
```

And it works. That's great for all sort of things, as you can pass pretty much any object to a function and it works, as long as the object has the properties the function needs:

```
const pony = { name: 'Rainbow Dash', color: 'blue' };
const horse = { speed: 40, color: 'black' };
const printColor = animal => console.log(animal.color);
// works as long as the object has a `color` property
```

This dynamic nature allows wonderful things but it is also a pain for a few others compared to more statically-typed languages. The most obvious might be when you call an unknown function in JS from another API, you pretty much have to read the doc (or, worse, the function code) to know what the parameter should look like. Take a look at our previous example: the method `printColor` needs a parameter with a `color` property. That can be hard to guess, and of course it is much worse in day-to-day work, where we use various libraries and services developed by fellow developers. One of Ninja Squad's co-founders is often complaining about the lack of types in JS, and finds it regrettable he can't be as productive and write as good code as he would in a more statically-typed environment. And he is not entirely wrong, even if he is sometimes ranting for the sake of it too! Without type information, IDEs have no real clue if you're doing something wrong, and tools can't help you find bugs in your code. Of course, we have tests in our apps, and Angular has always been keen on making testing easy, but it's nearly impossible to have a perfect test coverage.

That leads to the maintainability topic. JS code can become hard to maintain, despite tests and documentation. Refactoring a huge JS app is no easy task, compared to what could be done in other statically-typed languages. Maintainability is a very important topic, and types help humans and tools to avoid mistakes when writing and maintaining code. Google has always been keen to push new solutions in that direction: it's easy to understand as they have some of the biggest web apps of the world, with GMail, Google apps, Maps… So they have tried several approaches to front-end maintainability: GWT, Google Closure, Dart… All trying to help writing big webapps.

For Angular, the Google team wanted to help us writing better JS, by adding some type information to our code. It's not a very new concept in JS, it was even the subject of the ECMASCRIPT 4 specification, which was later abandoned. At first they announced AtScript, as a superset of ES6 with annotations (types annotations and another kind I'll discuss later). They also announced the support of TypeScript, the Microsoft language, with additional type annotations. And then, a few