```
@Component({ selector: 'ns-home' })
class HomeComponent {

  constructor(@Optional() hello: HelloService) {
    logger.log(hello);
  }

}
```

The `@Component` decorator is added to the class `Home`. When Angular loads our app, it will find the class `Home` and will understand that it is a component, based on the metadata the decorator will add. Cool, huh? As you can see, a decorator can also receive parameters, here a configuration object.

I just wanted to introduce the raw concept of decorators; we'll look into every decorator available in Angular all along the book.

I have to point out that you can use decorators with Babel as a transpiler instead of TypeScript. There is even a plugin to support all the Angular decorators: angular2-annotations. Babel also supports class properties, but not the type system offered by TypeScript. You can use Babel, and write "ES6+" code, but you will not be able to use the types, and they are very useful for the dependency injection for example. It's completely possible, but you'll have to add more decorators to replace the types.

So my advice would be to give TypeScript a try! All my examples from here will use it. It's not very intrusive, as you can use it just where it's useful and forget about it for the rest. If you really don't like it, it will not be very difficult to switch to ES6 with Babel or Traceur, or even ES5, if you are slightly crazy (but honestly, an Angular app in ES5 has pretty ugly code).