```
import { Component } from '@angular/core';

@Component({
  selector: 'ns-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'ponyracer';
}
```

Our application itself is a simple component. To tell Angular that it is a component, we use the `@Component` decorator. And to be able to use this decorator, we have to import it as you can see at the top of the file.

When you write new components, don't forget to import the `Component` decorator. You may forget to do so at the beginning, but it won't last, as the compiler will yell at you! ;)

You'll see that most of the things we need are in the `@angular/core` module, but that's not always the case. For example, when dealing with HTTP, we'll use imports from `@angular/http`; or, if we use the router, we'll import from `@angular/router`, etc.

```
import { Component } from '@angular/core';

@Component({
})
export class AppComponent {
  title = 'ponyracer';
}
```

The `@Component` decorator is expecting a configuration object. We'll see later in details what you can configure here, but for now only one property is expected: the `selector` one. It will tell Angular what to look for in our HTML pages. Every time Angular finds an element in our HTML which matches the selector of the component, it will create an instance of the component, and replace the content of the element by the template of the component.

```
import { Component } from '@angular/core';

@Component({
  selector: 'ns-root',
})
export class AppComponent {
  title = 'ponyracer';
}
```

So, here, every time our HTML will contain an element like `<ns-root></ns-root>`, Angular will instantiate a new instance of our `AppComponent` class.