This is a tough decision and there are many different ways to decide on these factors, but in most cases, it will depend simply on the purpose and the size of the application. For the time being, our challenge is to define an initial strategy that allows the team to evolve and enhance the architecture alongside application development. The answers related to deciding on the factors will certainly keep coming up as time goes on, but we should be able to perform some refactoring activities to keep the architecture healthy and up to date.

Four ways to organize the code

There are many ways, tendencies, and techniques to organize the project's code within files and directories. However, it would be impossible to describe all of them in detail, and we will present the most used and discussed styles in the JavaScript community.

Throughout the book, we will apply each of the following styles to our project as far as it evolves.

The inline style

Imagine that you need to develop a fast and disposable application prototype. The purpose of the project is just to make a presentation or to evaluate a potential product idea. The only project structure that we may need is the old and good `index.html` file with inline declarations for the scripts and style:

```
app/                    -> files of the application
  index.html            -> main html file
  angular.js            -> AngularJS script
```

If the application is accepted, based on the prototype evaluation, and becomes a new project, it is highly recommended that you create a whole structure from scratch based on one of the following styles.

The stereotyped style

This approach is appropriate for small apps with a limited number of components such as controllers, services, directives, and filters. In this situation, creating a single file for each script may be a waste. Thus, it could be interesting to keep all the components in the same file in a stereotyped way as shown in the following code:

```
app/                  -> files of the application
  css/                -> css files
    app.css           -> default stylesheet
  js/                 -> javascript application components
```

```
    app.js              -> main application script
    controllers.js      -> all controllers script
    directives.js       -> all directives script
    filters.js          -> all filters script
    services.js         -> all services script
  lib/                  -> javascript libraries
    angular.js          -> AngularJS script
  partials/             -> partial view directory
    login.html          -> login view
    parking.html        -> parking view
    car.html            -> car view
  index.html            -> main html file
```

With the application growing, the team may choose to break up some files by shifting to the specific style step by step.

The specific style

Keeping a lot of code inside the same file is really hard to maintain. When the application reaches a certain size, the best choice might be to start splitting the scripts into specific ones as soon as possible. Otherwise, we may have a lot of unnecessary and boring tasks in the future. The code is as follows:

```
app/                    -> files of the application
  css/                  -> css files
    app.css             -> default stylesheet
  js/                   -> javascript application components
    controllers/        -> controllers directory
      loginCtrl.js      -> login controller
      parkingCtrl.js    -> parking controller
      carCtrl.js        -> car controller
    directives/         -> directives directory
    filters/            -> filters directory
    services/           -> services directory
    app.js              -> main application script
  lib/                  -> javascript libraries
    angular.js          -> AngularJS script
  partials/             -> partial view directory
    login.html          -> login view
    parking.html        -> parking view
    car.html            -> car view
  index.html            -> main html file
```

In this approach, if the number of files in each directory becomes oversized, it is better to start thinking about adopting another strategy, such as the domain style.

The domain style

With a complex domain model and hundreds of components, an enterprise application can easily become a mess if certain concerns are overlooked. One of the best ways to organize the code in this situation is by distributing each component in a domain-named folder structure. The code is as follows:

```
app/                    -> files of the application
  application/          -> application module directory
    app.css             -> main application stylesheet
    app.js              -> main application script
  login/                -> login module directory
    login.css           -> login stylesheet
    loginCtrl.js        -> login controller
    login.html          -> login view
  parking/              -> parking module directory
    parking.css         -> parking stylesheet
    parkingCtrl.js      -> parking controller
    parking.html        -> parking view
  car/                  -> car module directory
    car.css             -> car stylesheet
    carCtrl.js          -> car controller
    car.html            -> car view
  lib/                  -> javascript libraries
    angular.js          -> AngularJS script
  index.html            -> main html file
```

Summary

Since the creation of the Web, many technologies related to the use of HTML and JavaScript have evolved. These days, there are lots of great frameworks such as AngularJS that allow us to create really well-designed web applications.

In this chapter, you were introduced to AngularJS in order to understand its purposes. Also, we created our first application and took a look at how to organize the code.

In the next chapter, you will understand how the AngularJS directives can be used and created to promote reuse and agility in your applications.