04

O papel de um carregador de módulos

Transcrição

Vamos alterar o arquivo `tsconfig.json` e indicar para o TypeScript que ele deve usar o sistema de módulos do `System.js`:

```
{
    "compilerOptions": {
        "target": "es6",
        "outDir": "app/js",
        "noEmitOnError": true,
        "noImplicitAny": true,
        "removeComments": true,
        "module": "system"
    },
    "include": [
        "app/ts/**/*"
    ]
}
```

Agora, quando nossos arquivos são compilados, eles possuem essa estranha estrutura:

```
System.register(["../views/NegociacoesView", "../views/MensagemView", "../models/Negociacoes", "
    "use strict";
    var __moduleName = context_1 && context_1.id;
    var NegociacoesView_1, MensagemView_1, Negociacoes_1, Negociacao_1, NegociacaoController;
    return {
   // código posterior omitido
```

Por fim, vamos importar o loader utilizá-lo para carregar `js/app/js`. É a partir dele que os demais serão carregados.

```
    <div id="negociacoesView"></div>
    <script src="lib/jquery.min.js"></script>

    <script src="lib/system.js"></script>
    <script>
         System.defaultJSExtensions = true;
        System.import('js/app.js').catch(err => console.error(err));
    </script>
```

Contudo, isso não é suficiente. Loaders usam XMLHttpRequest, ou seja, realizam requisições Ajax para baixar os módulos e para isso precisamos de um servidor que disponibiliza nossa aplicação para o browser.