Angular HttpClient

IT Akademija

HTTP Protocol

The most common communication method between client applications and the back end is the HTTP protocol. A RESTful API typically exposes data from the Back End, which can then be consumed by the client and presented to the user.

Browsers have two native APIs for making HTTP Requests:

- XMLHttpRequest - a default in every browser, introduced in 2002.
- window.fetch or just fetch() - a more powerful, flexible API - available on current browsers. Implements control for common web security standards, like CORS.

HttpClient

Angular provides `import { HttpClient} from ‘@angular/common/http’` via the `HttpClientModule`. It is a wrapper fover the `XMLHttpRequest` interface. The HttpClient has an API, which provides easy access to the most common HTTP methods. It mainly uses RxJS Observables for output, which makes operating with responses very easy. Most frequently used HTTP verbs are implemented as simple methods:

- httpClient.get()
- httpClient.post()
- httpClient.delete()
- httpClient.put()

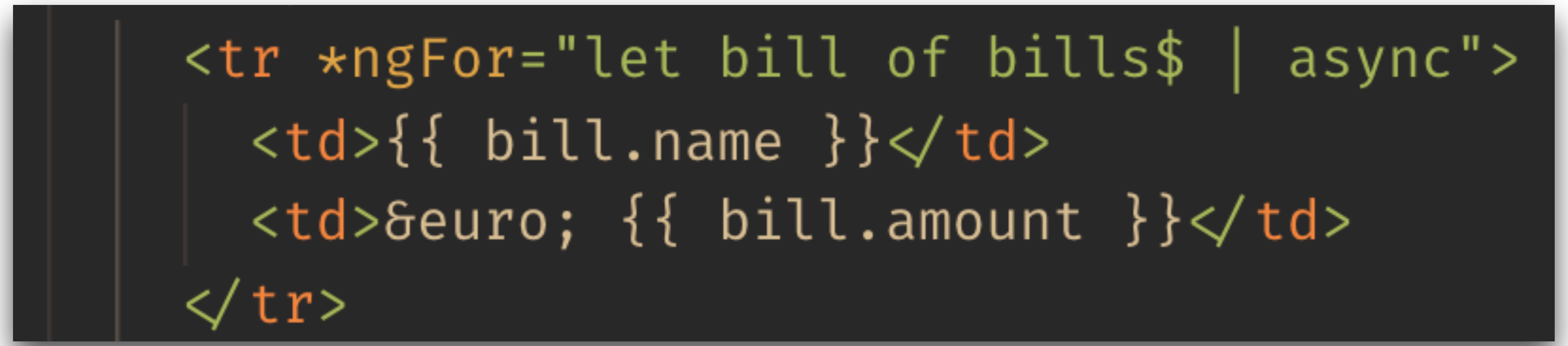
```
import { HttpClientModule } from '@angular/common/http';
@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
  ],
  providers: [],
  bootstrap: [AppComponent],
})
```

```
constructor(private httpClient: HttpClient) {}

loadExpenses() {
  return this.httpClient.get<Expense[]>('/api/expenses');
}
```

AsyncPipe

- Angular provides a way to subscribe to Observables in the HTML template - called the AsyncPipe.



```
<tr *ngFor="let bill of bills$ | async">
  <td>{{ bill.name }}</td>
  <td>&euro; {{ bill.amount }}</td>
</tr>
```

Task #1: HTTP Client

Get your expense list and upcoming bills from an API.

- All this time - you had a backend running in the background - time to put it to use!
- Use HttpClient in `expenses.service.ts` to load your expenses from the `/api/expenses` URL. (Don't forget to provide the type of the response as the generic).
- Use the service in the `expenses.component.ts` to add the expenses to your expenses object.
- Use AsyncPipe in the HTML to subscribe to that observable and show them in the template.
- (Optional) Do the same for upcoming bills in `home.component.ts` and get them from `/api/upcomingBills`.