Lab Dependency Injection

Exercise 1: A service

1. Create a class (just a regular class) `contacts.service.ts`.
2. This class will be dependent on `Http`, inject this with the constructor.

```
@Injectable()
export class ContactsService {
    constructor(private http: Http) { }
```

3. Create the functions `getContacts()` and `addContact()`. Do not return the `Http` observable/promise, instead :
    - map the observable (`.map(c => c.json())`)

```
getContacts() {
    return this.http.get('url')
                    .map(c => c.json());
}
```

4. Mark the class with `@Injectable()` and register it as a provider in the `HomeModule`.

```
@NgModule({
    imports: [...],
    declarations: [...],
    exports: [...],
    providers: [..., ContactsService]
})
export class HomeModule { }
```

5. Inject the service and retrieve your contacts.

```
export class HomeComponent {
     constructor(private contactsService: ContactsService) {
         contactsService.getContacts().subscribe(contacts => {
             /* store */
         });
     }
}
```

Exercise 2: Mocking services

1. Create a copy of the service and call it `MockContactService`.
   Implement the same methods as `ContactService` (perhaps use an interface?) but return a fixed collection. This service can now be used for development/debugging/testing.
2. Change the provider registration so that it uses this service instead of the real one.

```
@NgModule({
    imports: [...],
    declarations: [...],
    exports: [...],
    providers: [..., { provide: ContactsService, useClass:
MockContactsService }]
})
export class HomeModule { }
```

3. Your component doesn't need changing. Angulars DI should now inject the `MockContactsService`, thus the data on the page should automatically reflect the mocked data.

Exercise 3: Wrap HTTP

Alright, back to a real situation. Disable your `MockContactsService`.

For our application, we want to send an authentication token along with every HTTP request. So let's write a wrapper that adds an HTTP header with this authentication token.

Of course, this wrapper doesn't know what that token is. We have an `AuthService` for that. Here it is:

```
@Injectable()
export class AuthService {
    getAuthToken() { return 'supersecret123'; }
}
```

The HTTP wrapper should be injected in whereever services ask for an `Http` instance, so no changes will have to be made to `ContactsService`.

Create this wrapper.