Cheat Sheet – Http

Http & Observables

In Angular 2, reaching out to the web is very simple as the framework ships with its own Http Service (conveniently called 'Http').
This service offers methods for various Http methods, for example this one:

```
this.http.get('example.com/resource?myId=123');
```

However, this example would not return the fetched data directly. Instead, it sets up and configures **an Observable**.
Angular 2 by default uses Observables to handle Http access. This is done because Http operations are asynchronous by nature. Using Observables, the code execution may continue and the Subscriber of an Observable will eventually handle the response.
A method in a custom service may therefore return the Observable created with the above code.

```
@Injectable()
class MyCustomHttpService {
   getMyData() {
      return this.http.get('example.com/resource?myId=123');
   }
}
```

In your component, you may then subscribe to that Observable and use the returned response:

```
this.myCustomHttpService.getMyData()
     .subscribe(
          response => console.log(response),
          error => console.error(error),
          completed => console.log('Operation completed!')
     );
```

POST Requests

Post requests work like GET requests (see above). The only difference is, that they also require a body to be sent with the request. This body has to be a string and might (for example) be an object, turned into a string using 'JSON.stringify(obj)'.

Optional Http Arguments

You may set up Http calls with additional, optional argument. For example, you may want to add certain **Headers** to your Http call.
Therefore, the Http service methods take an optional argument in the form of a JS object. This object allows you to set up additional configuration.

```
myHeaders = new Headers();
myHeaders.append('Content-Type', 'application/json');


this.http.get('example.com/resource?myId=123', {
   headers: myHeaders
});
```

More about Observables

You could create a complete course about Observables. Since I haven't done that (yet?), here are some great resources, which allow you to dive deeper into them:
Official docs: http://reactivex.io/rxjs/
The introduction you have been missing:
https://gist.github.com/staltz/868e7e9bc2a7b8c1f754
Another introduction to Observables: https://medium.com/google-developer-experts/angular-introduction-to-reactive-extensions-rxjs-a86a7430a61f#.e0whi9jor