Http (Final)

By: Mosh Hamedani

Adding the Http script:

First you need to import **HttpModule** and **JsonpModule** into **AppModule** (or any modules that require HTTP services):

```
import { HttpModule, JsonpModule } from '@angular/http';


@NgModule({
    imports: [ HttpModule, JsonpModule ]
})
export class AppModule {}
```

Http Class

```
http.get(url)
     .map(response => response.json());


http.post(url, JSON.stringify(obj))
     .map(response => response.json());
```

Other methods:

```
http.put()
http.delete()
http.patch()
```

Http (Final)

By: Mosh Hamedani

Using the Http Class

```
import { Http } from '@angular/http';


@Injectable()
export class PostService {
    constructor(private _http: Http) {}

    getPosts() {
        return this._http.get(url).map(res => res.json());
    }
}
```

Component's Lifecycle Hooks

- **OnInit**: to execute logic after component's data-bound properties have been initialized.
- **OnChanges**: to execute logic if any bindings have changed
- **AfterContentInit**: to execute logic when a component's content (<ng-content>) has been fully initialized.
- **AfterContentChecked**: to execute logic after every check of component's content.
- **AfterViewInit**: to execute logic when a component's view has been fully initialized.
- **AfterViewChecked**: to execute logic after every check of component's view.
- **OnDestroy**: to execute clean up logic when a component is destroyed.

These interfaces have a method with the same name as the interface name, prefixed with "ng".

```
export class AppComponent implements OnInit {
    ngOnInit() {
    }
}
```

Http (Final)

By: Mosh Hamedani

Showing a Loader Icon

```
export class AppComponent {
    isLoading = true;

    ngOnInit() {
        this._postService.getPosts()
            .subscribe(posts => {
                this.posts = posts
                this.isLoading = false;
            });
    }
}
```

In the view:

```
<i *ngIf="isLoading" class="fa fa-spinner fa-spin fa-3x"></i>
```

Http (Final)

By: Mosh Hamedani

Jsonp Class

Only supports get().

Adding Custom Request Headers

```
var headers = new Headers({
    “key”: “value”
});

var options = new RequestOptions({ headers: headers });

http.get(url, options);
```