

Starting Up

```
@NgModule({
  declarations: [],
  imports: [HttpModule],
  bootstrap: [],
  providers: []
})
```

Using Angular HTTP

```
constructor(private http: Http) {}
```

Http Verbs

GET `get(url: string, options?: RequestOptionsArgs) : Observable<Response>` Performs a request with get http method.

POST `post(url: string, body: any, options?: RequestOptionsArgs) : Observable<Response>` Performs a request with post http method.

PUT `put(url: string, body: any, options?: RequestOptionsArgs) : Observable<Response>` Performs a request with put http method.

DELETE `delete(url: string, options?: RequestOptionsArgs) : Observable<Response>` Performs a request with delete http method.

Http Verbs (cont)

PATCH `patch(url: string, body: any, options?: RequestOptionsArgs) : Observable<Response>` Performs a request with patch http method.

HEAD `head(url: string, options?: RequestOptionsArgs) : Observable<Response>` Performs a request with head http method.

OPTIONS `options(url: string, options?: RequestOptionsArgs) : Observable<Response>` Performs a request with options http method.

RequestOptionsArgs

```
interface RequestOptionsArgs {
  url : string
  method : string | RequestMethod
  search : string | URLSearchParams
  headers : Headers
  body : any
  withCredentials : boolean
  responseType : ResponseContentType
}
```

Headers

```
class Headers {
  static fromResponse(HeaderString(headersString): string) : Headers
  constructor(headers?: Headers | { [name: string]: any })
  append(name: string, value: string) : void
  delete(name: string) : void
  forEach(fn: (values: string[], name: string, headers: Map<string, string[]>) => void) : void
  get(name: string) : string
  has(name: string) : boolean
  keys() : string[]
  set(name: string, value: string | string[]) : void
  values() : string[][]
  toJSON() : {[name: string]: any}
  getAll(name: string) : string[]
  entries()
}
```

Observables - Flow



By **Nathan** (Nathane2005)

cheatography.com/nathane2005/

Published 17th October, 2016.

Last updated 17th October, 2016.

Page 1 of 3.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

Sample Delegate

```
import { Injectable } from
 '@angular/core';
import {Http, Response, Headers,
 RequestOptions} from " @an -
 gular/ http ";
import 'rxjs/Rx';
import {Observable} from " rxj -
 s";
import {Post} from "./http /po -
 st.c lass";
@Injectable()
export class HttpSe rvice {
    con str uct or( private http:
 Http) { }
    private reques tUrl: string =
 'http: //l oca lho st: 400 0/p -
 osts';
    //Do all methods and
 observable options
    get Data(id : number) :
 Observabl e<P ost> {
        return this.h ttp.$get(
 his.re que sTUr l} /${id})
        .ma p(t his.ma pRe -
 sponse)
        .ca tch (th is.h an -
 dle Error)
    }
    han dle Err or( error: any):
 Observabl e<a ny> {
        con sol e.e rro r('An
 error occurred', error);
        return Observabl e.t -
 hro w(e rro r.j son() || 'Server
 error');
    }
    map Res pon se( res ponse :
 Response) : Post {
        return respon se.j son();
    }
}
```

Sample Delegate (cont)

```
> addData(body : Post) : Observable<Post> {
    let bodyString = JSON.stringify(body);
    let header = new Headers({
        'Content-Type': 'application/json'
    });
    let options = new RequestOptions({
        headers : header
    });
    return this.http.post(${this.re que sTUr
 l}, bodyString, options)
        .map(this.mapResponse)
        .catch(this.handleError);
}
updateData(body : Post) : Observable<Post>
{
    let bodyString = JSON.stringify(body);
    let header = new Headers({
        'Content-Type': 'application/json'
    });
    let options = new RequestOptions({
        headers : header
    });
    return this.http.put(${this.re que sTUr
 l} /${ bod y.id}, bodyString, options)
        .map(this.mapResponse)
        .catch(this.handleError);
}
```

Sample Delegate (cont)

```
> }
deleteData(body : Post) : Observable<P-
ost> {
    return this.http.delete(${this.re que s
 tU r l} /${ bod y.id})
        .map(this.mapResponse)
        .catch(this.handleError);
}
}
```

Rx - Map

```
getData(id : number) :
Observable<Post> {
    return this.h ttp.$get(
 his.re que sTUr l} /${id})
        .ma p(t his.ma pRe -
 sponse)
        .ca tch (th is.h an -
 dle Error)
    }
    map Res pon se( res ponse :
 Response) : Post {
        return respon se.j son();
    }
}
```

The map() function takes in a lambda function or a reference to a function that will execute the procedure and return the mapped result.
Accepts (res : Response) and returns a result.



By Nathan (Nathane2005)

Published 17th October, 2016.

Last updated 17th October, 2016.

Page 2 of 3.

Sponsored by CrosswordCheats.com

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>

Rx - catch

```

getData(id : number) :
Observable<Post> {
    return this.http.$get(
        this.requestUrl}/${id})
        .map(this.mapResponse)
        .catch(this.handleError)
    }
    handleError(error: any):
Observable<any> {
        console.error('An
error occurred', error);
        return Observable.throw(
            new Error('Server
error'));
    }
}

```

The catch reference is there to handle exceptions that are thrown. This gives you an opportunity to handle them in a graceful manner.

All rx operations return an observable.

Observable - Subscribe

```

observeOnNext PartialObserver<T> | ((value:
T) => void)

```

```

error (error: any) => void

```

```

complete () => void

```

```

this.service.getData(10).subscribe(
    (data : Post) => {
        this.result = data;
    },
    (error : any) => {
        console.error(error);
    }
)

```

Reference

```

private requestUrl: string =
'http://localhost:4000/posts';
getData(id : number) :
Observable<Post> {
    return this.http.$get(
        this.requestUrl}/${id})
        .map(this.mapResponse)
        .catch(this.handleError)
    }
}

```

Using the back ticks to specify internal references ``
referencing the content in \${}



By **Nathan** (Nathane2005)

cheatography.com/nathane2005/

Published 17th October, 2016.

Last updated 17th October, 2016.

Page 3 of 3.

Sponsored by **CrosswordCheats.com**

Learn to solve cryptic crosswords!

<http://crosswordcheats.com>