Angular 2 Http Cheat Sheet

by Nathan (Nathane2005) via cheatography.com/28056/cs/8480/

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.

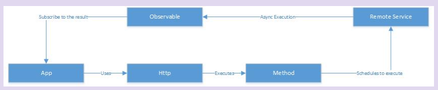
RequestOptionsArgs

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

Headers

```
class Headers {
  staticfromResponseHeaderString(h
eadersString: 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



Sample Delegate

```
import { Injectable } from
'@angular/core';
import {Http, Response, Headers,
RequestOptions} from
"@angular/http";
import 'rxjs/Rx';
import {Observable} from "rxjs";
import {Post} from
"./http/post.class";
@Injectable()
export class HttpService {
  constructor(private http: Http) {
}
  private requestUrl: string =
'http://localhost:4000/posts';
  //Do all methods and observable
options
  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(error.json() ||
'Server error');
  }
  mapResponse(response : Response)
: Post {
    return response.json();
  }
```

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

Sample Delegate (cont)

```
  }
  deleteData(body : Post) :
Observable<Post> {
    return
this.http.delete(${this.requestUr
l}/${body.id})
      .map(this.mapResponse)
      .catch(this.handleError);
  }
}
```

Rx - Map

```
getData(id : number) :
Observable<Post> {
    return
this.http.get(${this.requestUrl}/$
{id})
      .map(this.mapResponse)
      .catch(this.handleError)
  }
  mapResponse(response : Response)
: Post {
    return response.json();
  }
```

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 : Respone) and returns a result.

Angular 2 Http Cheat Sheet

by Nathan (Nathane2005) via cheatography.com/28056/cs/8480/

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(error.json() ||
'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

observerOrNext	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 ${}