This cheat sheet is for the course Learn C# Full Stack Development with Angular and ASP.NET by Jannick Leismann.

ANGULAR OBSERVABLE

An **Observable** in Angular is analogous to a data stream to which you may subscribe. It is a component of the **RxJS** library, which facilitates the management of asynchronous tasks like as **user input events** and **HTTP requests**.

Why Do We Use Observables?

Asynchronous Data

Handle data that comes in the future (like API responses).

Event Handling

Manage events like user clicks or input changes.

Real-time Updates

Stream data continuously, like stock prices or live chats.

Creating an observable using the RxJS library.

```
import { Observable } from 'rxjs';

const myObservable = new Observable(subscriber => {

  subscriber.next('Hello');

  subscriber.next('World');

  subscriber.complete();

});
```

Subscribing to an Observable

To get the data from an observable, you need to subscribe to it:

```
myObservable.subscribe({
  next(value) { console.log(value); },  // Called for each value
  error(err) { console.error(err); },   // Called if there's an error
  complete() { console.log('Done'); }   // Called when the observable completes
});
```

Using Observables with HttpClient

The HttpClient service in Angular processes HTTP requests and returns observables. Here's an easy illustration:

Create a service to fetch the data.

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Observable } from 'rxjs';
@Injectable({
  providedIn: 'root'
})
export class DataService {
  private apiUrl = 'https://api.example.com/data';
  constructor(private http: HttpClient) {}
  getData(): Observable<any> {
    return this.http.get<any>(this.apiUrl);
  }
}
```

Create a component to use the service.

```
import { Component, OnInit } from '@angular/core';

import { DataService } from './data.service';

@Component({

  selector: 'app-data',

  template: '<div *ngIf="data">{{ data | json }}</div>',

})

export class DataComponent implements OnInit {

  data: any;

  constructor(private dataService: DataService) {}

  ngOnInit(): void {

    this.dataService.getData().subscribe(response => {

      this.data = response;

    });

  }

}
```

You can effectively manage **asynchronous** actions and **data streams** in your Angular applications by utilizing **observables**.