In Angular, an Observable is a powerful tool that represents a stream of data that can be observed over time. It is part of the Reactive Extensions for JavaScript (RxJS) library, which is widely used in Angular for handling asynchronous operations and events. Observables are a key part of the reactive programming paradigm.

Here are the key concepts associated with Observables:

1. **Definition:**

- An Observable is a representation of any set of values or events over time.
- It can emit multiple values over time and can be observed by multiple parts of an application.

2. **Observable Creation:**

- Observables can be created using various methods. Common methods include:
  - `of`: Creates an observable from a sequence of values.
  - `from`: Converts an array, promise, or iterable into an observable.
  - `interval`: Creates an observable that emits sequential numbers at a specified interval.
  - `http.get`: Returns an observable for HTTP requests.

3. **Subscription:**

- Observables need to be subscribed to for their values or events to be observed.
- The `subscribe` method is used to attach observers to the observable.
- A subscription represents the execution of an observable, and it can be used to handle the emitted values.

4. **Operators:**

- RxJS provides a variety of operators that can be used to transform, filter, combine, and manipulate the data emitted by observables.
- Examples include `map`, `filter`, `mergeMap`, `concatMap`, and many more.

5. **Hot vs Cold Observables:**

- Cold observables start emitting values when a subscription is made.
- Hot observables emit values regardless of whether there are subscribers.

6. **Async Operations:**

- Observables are commonly used to handle asynchronous operations such as HTTP requests, user inputs, and timers.
- They provide a clean and consistent way to manage asynchronous code.

7. **Error Handling:**

- Observables can handle errors using the `catchError` and `retry` operators.
- Error handling is an essential aspect of building robust applications.

Example of Using Observables in Angular:

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

@Component({
  selector: 'app-example',
  template: `
    <h2>Observable Example</h2>
    <div *ngFor="let value of values | async">{{ value }}</div>
  `,
})
export class ExampleComponent implements OnInit {
  values: Observable<number[]>;

  ngOnInit() {
    // Creating an observable using the 'of' method
    this.values = Observable.of([1, 2, 3, 4, 5]);

    // Subscribing to the observable (not necessary when using the 'async' pipe
in the template)
    // this.values.subscribe(data => console.log(data));
  }
}
```

In this example:

- An observable (`values`) is created using the `of` method from RxJS, emitting an array of numbers.
- The `async` pipe in the template automatically subscribes to the observable and updates the view whenever new values are emitted.

Observables provide a declarative approach to handling asynchronous and event-based programming, making it easier to manage complex scenarios such as user interactions, data fetching, and real-time updates in Angular applications.