In **Angular**, *two-way data binding* allows you to **bind a property to the UI** and **update it automatically when the user changes the value**.

The most common way to achieve this is with the `[(ngModel)]` directive from `FormsModule`.

Example: Two-Way Binding with `[(ngModel)]`

```
// app.module.ts
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { FormsModule } from '@angular/forms'; // Required for ngModel
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule, FormsModule], // Import FormsModule here
  bootstrap: [AppComponent]
})
export class AppModule {}
```

```
// app.component.ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html'
})
export class AppComponent {
  username: string = 'John Doe';
}
```

```
<!-- app.component.html -->
<!-- Two-way binding with ngModel -->
<input [(ngModel)]="username" placeholder="Enter your name" />

<!-- Display the bound value -->
```

```
<p>Hello, {{ username }}!</p>
```

How It Works

1. `[(ngModel)]` is **syntactic sugar** for:

   ```
   [value]="username" (input)="username = $event.target.value"
   ```

2. The **square brackets** `[]` are for **property binding** (component → view).
3. The **parentheses** `()` are for **event binding** (view → component).
4. Together as `[()]`, they create **two-way binding**.

Important Notes

1. **Enable FormsModule**
   Without importing `FormsModule` in your module, `ngModel` will not work.
2. **For Reactive Forms**
   Use `FormControl` and `formControlName` instead of `ngModel`.
3. **Custom Components**
   You can implement two-way binding on your own components using:

   ```
   @Input() value: string;
   @Output() valueChange = new EventEmitter<string>();
   ```

   And bind with:

   ```
   <my-input [(value)]="username"></my-input>
   ```

✅ If you want, I can show you **how to implement two-way binding for a custom Angular component** so it works just like `[(ngModel)]`.
Do you want me to prepare that example?