**Component Data Binding in Angular**

Data binding in Angular facilitates the seamless flow of information between your component's data (TypeScript code) and the HTML template that represents its view. It eliminates the need for manual DOM manipulation, making your application more reactive and easier to maintain.

**Types of Data Binding:**

1. **Property Binding:**
   * **Direction:** One-way, from component to view.
   * **Syntax:** [property]="expression"
   * **Example:**

HTML

<input type="text" [value]="name">

This binds the value of the name property in your component to the value attribute of the input element. Changes in the name property will automatically update the input field's value.

1. **Event Binding:**
   * **Direction:** One-way, from view to component.
   * **Syntax:** (event)="handler()"
   * **Example:**

HTML

<button (click)="onClick()">Click Me</button>

This binds the click event of the button to the onClick() method in your component. When the button is clicked, the onClick() method will be executed.

1. **Two-Way Binding:**
   * **Direction:** Two-way, between component and view.
   * **Syntax:** [(ngModel)]="property"
   * **Example:**

HTML

<input type="text" [(ngModel)]="name">

This combines property and event binding to create a synchronized relationship. Changes in the input field's value are reflected in the name property, and vice versa.

**Key Benefits:**

* **Improved Readability:** Data binding makes your templates cleaner and more declarative, improving code readability.
* **Enhanced Maintainability:** Changes to data are automatically reflected in the view, reducing the risk of inconsistencies.
* **Increased Responsiveness:** Applications become more reactive to user interactions and data changes.
* **Reduced Boilerplate Code:** Data binding eliminates the need for manual DOM manipulation, reducing the amount of code you need to write.

**Example:**

TypeScript

import { Component } from '@angular/core';

@Component({

selector: 'app-my-component',

template: `

<h2>Hello, {{ name }}!</h2>

<input type="text" [(ngModel)]="name">

`,

})

export class MyComponent {

name: string = 'John Doe';

}

In this example:

* {{ name }} uses interpolation to display the value of the name property in the heading.
* [(ngModel)]="name" enables two-way binding between the input field and the name property.

By mastering data binding, you can create dynamic and user-friendly Angular applications that effectively interact with the user and respond to data changes.