**Angular Custom Property Binding Using @Input Decorator**

how to bind custom property in Angular using @Input decorator.

**Table of contents**

1. [Angular custom property binding](https://www.netjstech.com/2020/04/angular-custom-property-binding-using-input-decorator.html#AngularCustomBinding)
2. [Using @Input decorator in custom property binding](https://www.netjstech.com/2020/04/angular-custom-property-binding-using-input-decorator.html#InputDecorator)
3. [Angular custom property binding example](https://www.netjstech.com/2020/04/angular-custom-property-binding-using-input-decorator.html#AngularCustomBindingExp)

**Angular custom property binding**

Using custom property binding to set the model property of a custom component is a great way for parent and child components to communicate. For example following statement in the parent template shows the binding for the property “childItem” in the child component.

<app-child [childItem]=“parentItem”></app-child>

* Here **<app-child>** is the selector defined in the child component.
* childItem is a field in the child component.
* parentItem is a field in parent component.

By using custom property binding here, childItem is bound to the value of the parentItem. But that needs one more thing to be done, using @Input decorator.

**Using @Input decorator in custom property binding**

By default any property in a component is accessible with in the component where it is defined. If you want to expose any property outside of the component then that property has to be decorated with @Input decorator.

export class UserComponent {

@Input() usr: User;

...

...

}

With **usr** property decorated with @Input decorator, parent component can bind to this property of the child component.

@Input decorator marks a class field as an input property. The input property is bound to a DOM property in the template. During change detection, Angular automatically updates the data property with the DOM property's value.

**Angular custom property binding example**

The requirement is to show user details in a child component where each user instance is passed from the parent component.

**Creating a User class**

Create a Type script class user.model.ts to define a User class. If you are aware of MVC (Model View Controller) pattern then this class is the Model. There are 3 fields name, age and joinDate in the User class.

**user.model.ts**

export class User {

name : string;

age : number;

joinDate : Date;

constructor(name: string, age : number, joinDate : Date) {

this.name = name;

this.age = age;

this.joinDate = joinDate;

}

}

**app.component.ts (Parent component)**

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

import { User } from './user/user.model';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

users: User[];

constructor(){

//Adding User instances to users array

this.users = [new User('Jack', 56, new Date('2005-03-25')),

new User('Lisa', 32, new Date('2012-05-09')),

new User('Jayesh', 28, new Date('2014-10-21'))] ;

}

}

AppComponent uses User model so that is imported. An array of type User is defined and user instances are added to that array in the Constructor.

**app.component.html**

<div class="container">

<h3>User Details</h3>

<app-user \*ngFor="let user of users"

[usr]="user">

</app-user>

</div>

In the template with in the <app-user> selector users array is iterated using [ngFor directive](https://www.netjstech.com/2020/04/angular-ngfor-directive-with-examples.html" \t "_blank) and each user instance is bound to the usr property of the child component.

**user.component.ts (Child component)**

import {

Component, Input

} from '@angular/core';

import { User } from './user.model';

@Component({

selector: 'app-user',

templateUrl: './user.component.html'

})

export class UserComponent {

@Input() usr: User;

}

In the child component usr variable is decorated with @Input decorator indicating that parent component can bind to this property.

**User.component.html**

In the template user data is displayed using [string interpolation](https://www.netjstech.com/2020/04/angular-one-way-data-binding-string-interpolation.html).

<div class="container">

<div class="row">

<div class="col-xs-6">

<label>Name: </label> {{ usr.name }}

<label>Age: </label> {{ usr.age }}

<label>Join Date: </label> {{ usr.joinDate | date:'dd/MM/yyyy' }}

</div>

</div>

</div>