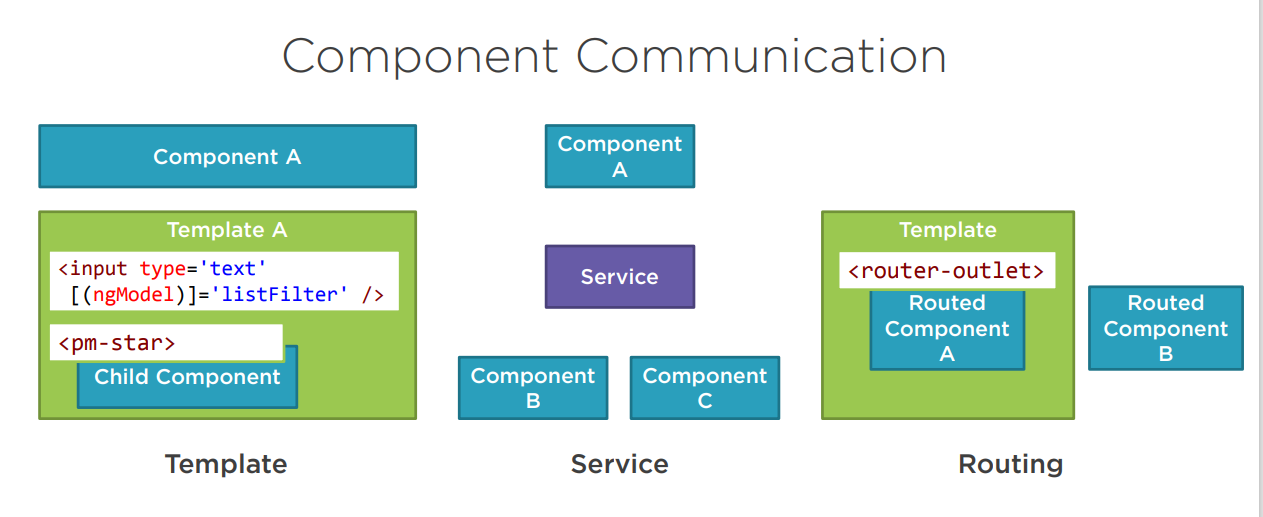
The Component is the main building block of an [Angular App](https://www.tektutorialshub.com/angular/angular-create-first-application/). A typical [Angular application](https://www.tektutorialshub.com/angular/angular-create-first-application/) consists of a lot of components.

Each component handles a small part of the UI.

These components must interact or communicate together to produce the complete user interface of the application



* How [Component Communication](https://www.tektutorialshub.com/angular/angular-component-communication-sharing-data/#component-communication) happen?

Here are the three Possible scenario   
(1)Parent to Child;  Pass data from parent to child component using **@Input** decorators and property binding. @Input Decorator

(2) Child to Parent; Using **EventEmitter** and passing data from child component to parent in the form of events with the help of **@Output** decorators.

(3) Use Child components methods and properties; The **@ViewChild** decorator can be used to access the child component’s public properties and methods.

(4) One Component to Many or Other non-relational components by using RxJS **BehaviorSubject** and **Subject** observable services.

*  Browser Session Storage
* : We can also store data in the browser memory to access data access sessions in the persisted formats using **localStorage** and **sessionStorage.**
* [Parent to Child Communication](https://www.tektutorialshub.com/angular/angular-component-communication-sharing-data/#parent-to-child-communication)
  + [Using @Input Decorator to Pass Data](https://www.tektutorialshub.com/angular/angular-component-communication-sharing-data/#using-input-decorator-to-pass-data)
  + [Listen for Input Changes](https://www.tektutorialshub.com/angular/angular-component-communication-sharing-data/#listen-for-input-changes)
* [Child to Parent Communication](https://www.tektutorialshub.com/angular/angular-component-communication-sharing-data/#child-to-parent-communication)
  + [Listens to Child Event](https://www.tektutorialshub.com/angular/angular-component-communication-sharing-data/#listens-to-child-event)
  + [Uses Local Variable to access the child](https://www.tektutorialshub.com/angular/angular-component-communication-sharing-data/#uses-local-variable-to-access-the-child)
  + [Uses a @ViewChild to get the reference to the child component](https://www.tektutorialshub.com/angular/angular-component-communication-sharing-data/#uses-a-viewchild-to-get-the-reference-to-the-child-component)
* [Communication when there is no relation](https://www.tektutorialshub.com/angular/angular-component-communication-sharing-data/#communication-when-there-is-no-relation)

**Passing Data Between Components**

**What Angular input decorator is?**

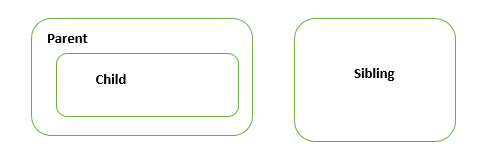
Angular provides **@Input** and **@Output** APIs to communicate between components. @Input in Angular belongs to the TypeScript family and as far as **@Input() decorator** is concerned we all are familiar with TypeScript and how flexible it is.

## parent to Child Communication

see how we can share data between the components.

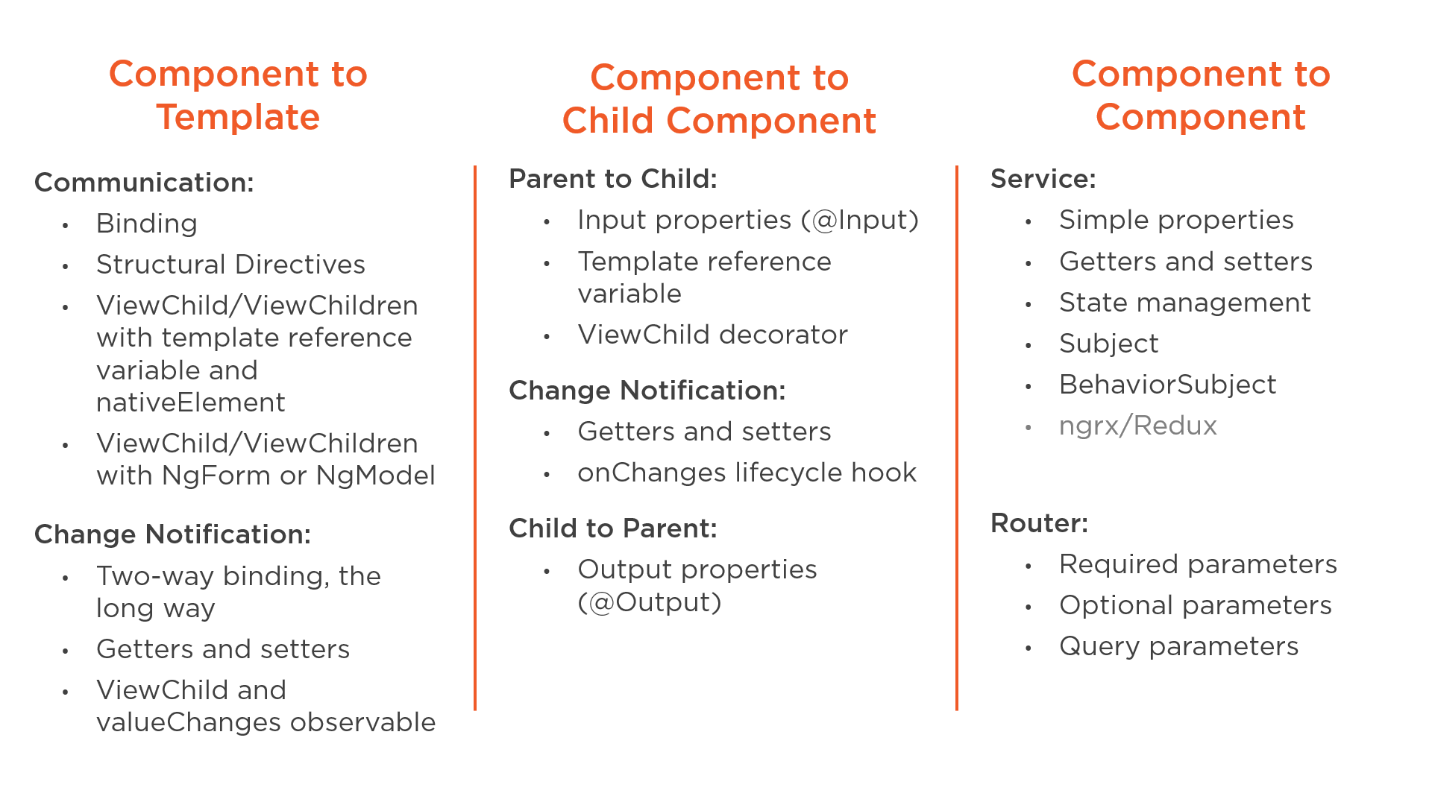
There are basically four ways to pass data between components; let's look at them one-by-one.

Before that, let’s see what we mean by parent and child components. The parent component is the component which acts as the container for other components, and a sibling is a component which is not in the container but which is not related to the other components.



The following points are the ways we can pass the data between parent and child components.

If the Components have a parent-child relationship then, then the parent component can pass the data to the child using the [@input](https://www.tektutorialshub.com/angular/angular-input-output-eventemitter/#input) Property.



### Using [@Input Decorator](https://www.tektutorialshub.com/angular/angular-passing-data-child-component/#the-child-component-with-input-decorator) to Pass Data

## ****Angular @Input****

Angular input decorator is just telling Angular, hey, when you find a property binding with this name, map it to my component property of this other name. Or, if I don’t give you an alias, use my component property name.

Create a property (someProperty) in the Child Component and decorate it with @Input(). This will mark the property as input property

export class ChildComponent {

@Input() someProperty: number;

}

In the Parent Component

1. Bind the Child component property in the Parent Component when instantiating the Child

In the Child Component

1. Import the [@Input](https://www.tektutorialshub.com/angular/angular-input-output-eventemitter/) module from @angular/Core Library
2. Mark those property, which you need data from the parent as input property using [@Input](https://www.tektutorialshub.com/angular/angular-input-output-eventemitter/) decorator

### Listen for Input Changes

The Child Component can get the values from the someProperty. But it also important for the child component to get notification when the values changes.

There are two ways in which we can achieve that.

1. Using OnChanges life Cycle hook or
2. Using a Property Setter on Input Property

**1. Parent Component to Child Component**

There will be always a need to send data between the parent component and the child component. Let’s see step by step how we can achieve this.

@Input:

This decorator is used to **obtain data from the component**. To achieve this, I have made changes to the component. Let’s add two components, namely a master and a child component, in the application by using Angular CLI's ng generate command.

When the components are added look at the code snippet for the child component (child.component.ts)

Example-1

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

**@Component({**

**selector: 'child-component',**

**template: `<h2>Child Component</h2>**

**current count is {{ count }}**

**`**

**})**

**export class ChildComponent {**

**@Input() count: number;**

**}**

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

**@Component({**

**selector: 'app-root',**

**template: `**

**<h1>Welcome to {{title}}!</h1>**

**<button (click)="increment()">Increment</button>**

**<button (click)="decrement()">decrement</button>**

**<child-component [count]=Counter></child-component>` ,**

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

**})**

**export class AppComponent {**

**title = 'Component Interaction';**

**Counter = 5;**

**increment() {**

**this.Counter++;**

**}**

**decrement() {**

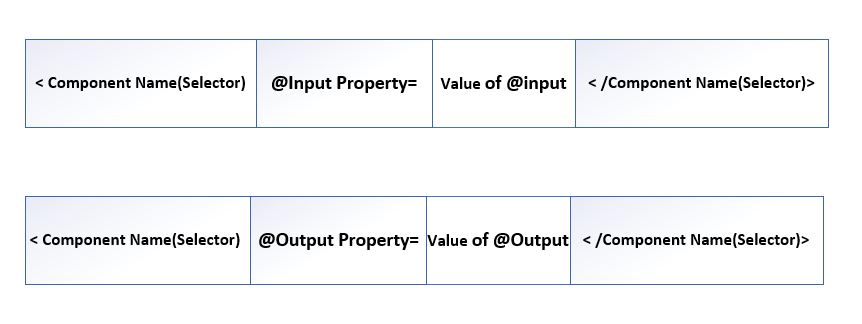
**this.Counter--;**

**}**

**}**

**Naming Pattern**

Below shows the Naming Pattern for **@Input**and **@Output**decorator with Component,



**Angular 12 @Input Decorator Tutorial with Example**

In this topic I will be teaching you how you can send data from parent component to child component using @Input Angular decorator.

In this demo tutorial, I will create a basic Angular application from scratch. Then I will create a child component, and after that, I will be sending data from the parent component to child component using the @Input decorator.

**Example02-Setting Up Basic Angular Project for Data Interaction Demo**

**01. Prerequisite**

In order to create this demo app you must have **Node JS development environment** set up in your machine.

**02. Install Angular CLI**

Angular projects are developed using Angular CLI, it’s an official tool. Hit the given below command to install the Angular CLI, ignore if Angular CLI is already installed.

npm install @angular/cli -g

**03. Set up Angular Project from Scratch**

I will be creating an employee record management system with Angular, in this demo app i will consume RESTful API via HttpClient service.

It’s time to setup Angular project, run the following command in Angular CLI.

ng new input-angular

It will ask you the following questions…

**Would you like to add Angular routing?**  
Select y and Hit Enter.

**Which stylesheet format would you like to use?** (Use arrow keys)  
Choose CSS and hit Enter

After that your project will start creating, Once the project is created then don’t forget to get into the project’s folder.

cd input-angular

**Let’s create a child component to send data from parent component.**

ng g c child

Now you are all set to play with **@Input() Decorator in Angular**.

**Send Data From Parent Component to Child Component Using @Input() Decorator**

This is going to be our **parent component** where i will create a fruits array and this fruit array’s data i will send to child component.

Go to app.component.ts file, and paste the below code.

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

@Component({

selector: 'app-root',

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

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

})

export class AppComponent {

fruits = ['Mengo', 'Orange', 'Banana'];

constructor() { }

addFruit(item) {

this.fruits.push(item);

}

}

In our **app component’s html** file i am using a simple form which is helping me to add a **fruit** into the fruits array. And that <app-child> selector from child component taking that data and showing in it self using @Input decorator.

Go to **app.component.html file**, and paste the below code.

<input #newFruit type="text" placeholder="Enter a new fruit">

<button (click)="addFruit(newFruit.value)">Add Fruit</button>

<app-child [data]="fruits"></app-child>//property binding

Markup

**Setting up Data in Child Component using @Input() Decorator**

Go to child.component.ts file, and the following code.

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

@Component({

selector: 'app-child',

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

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

})

export class ChildComponent {

@Input() data: string[];

}

TypeScript

Go to **child.component.ts** file, and the following code.

<h1>Fruit List</h1>

<ul>

<li \*ngFor="let item of data">{{item}}</li>

</ul>