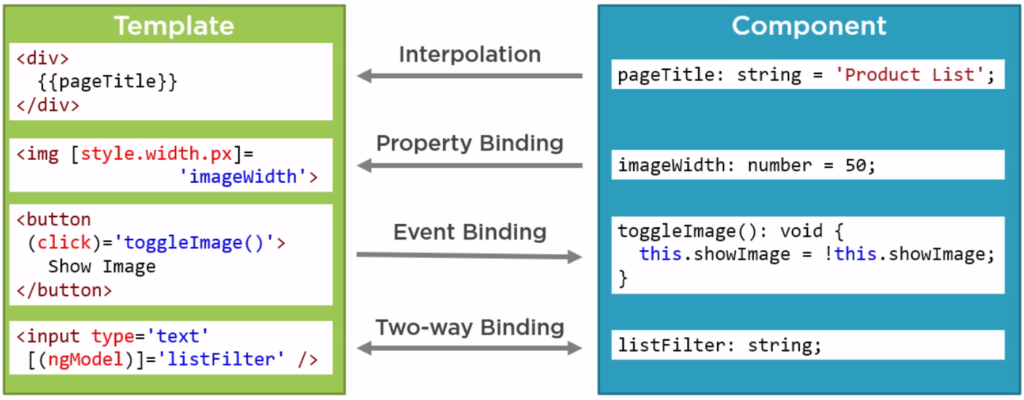
**Data Bindings**

1. **Angular Data Bindings**

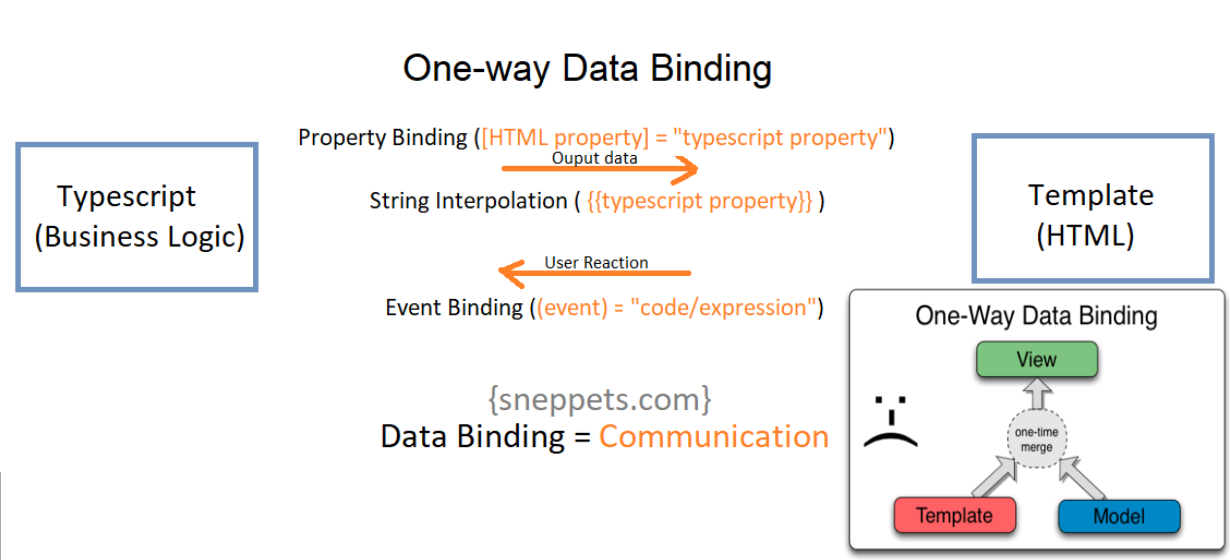
What is Angular Data Binding? Data binding is a technique, **where the data stays in sync between the component and the view**. Whenever the user updates the data in the view, Angular updates the component. When the component gets new data, the Angular updates the view.

Data binding automatically keeps your page up-to-date based on your application's state. You use data binding to specify things such as the source of an image, the state of a button, or data for a particular user.

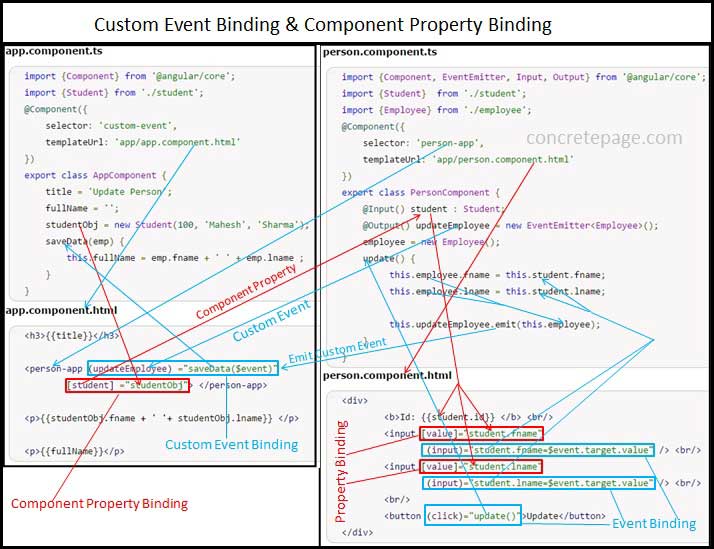
****

**There are two type of data binding one-way and two-way**

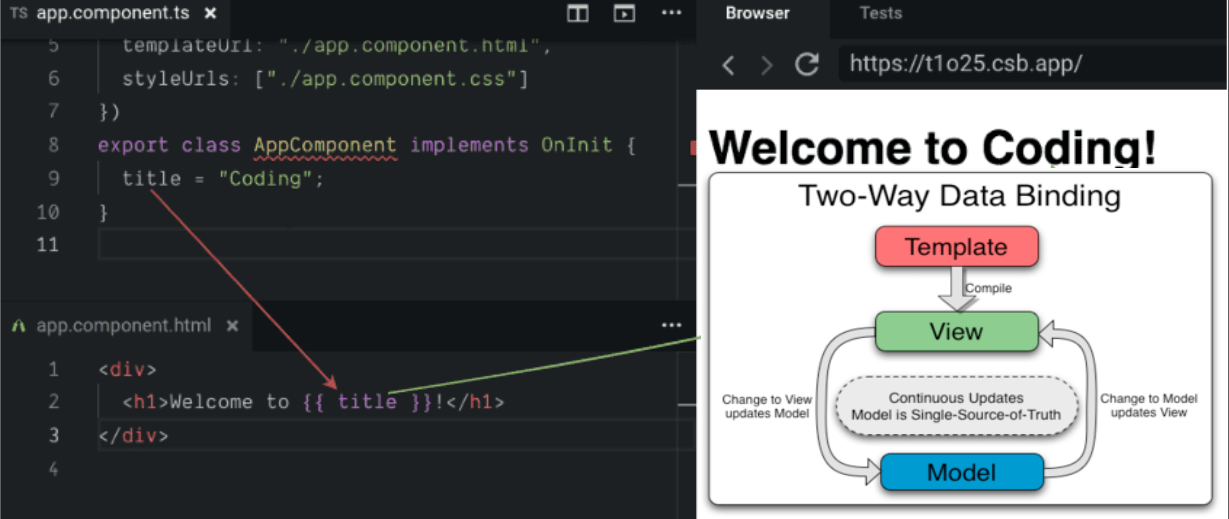
1. **One way**
2. **Two way**



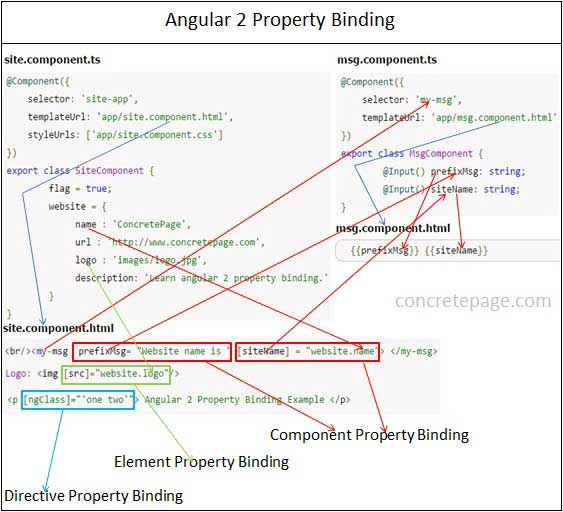
1. **Property, Event, Element and component binding example**

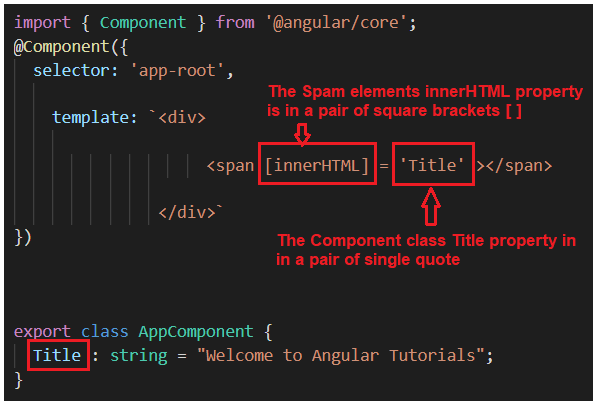
****

1. **Two Way Data Binding**



1. **Property & Event Bindings**

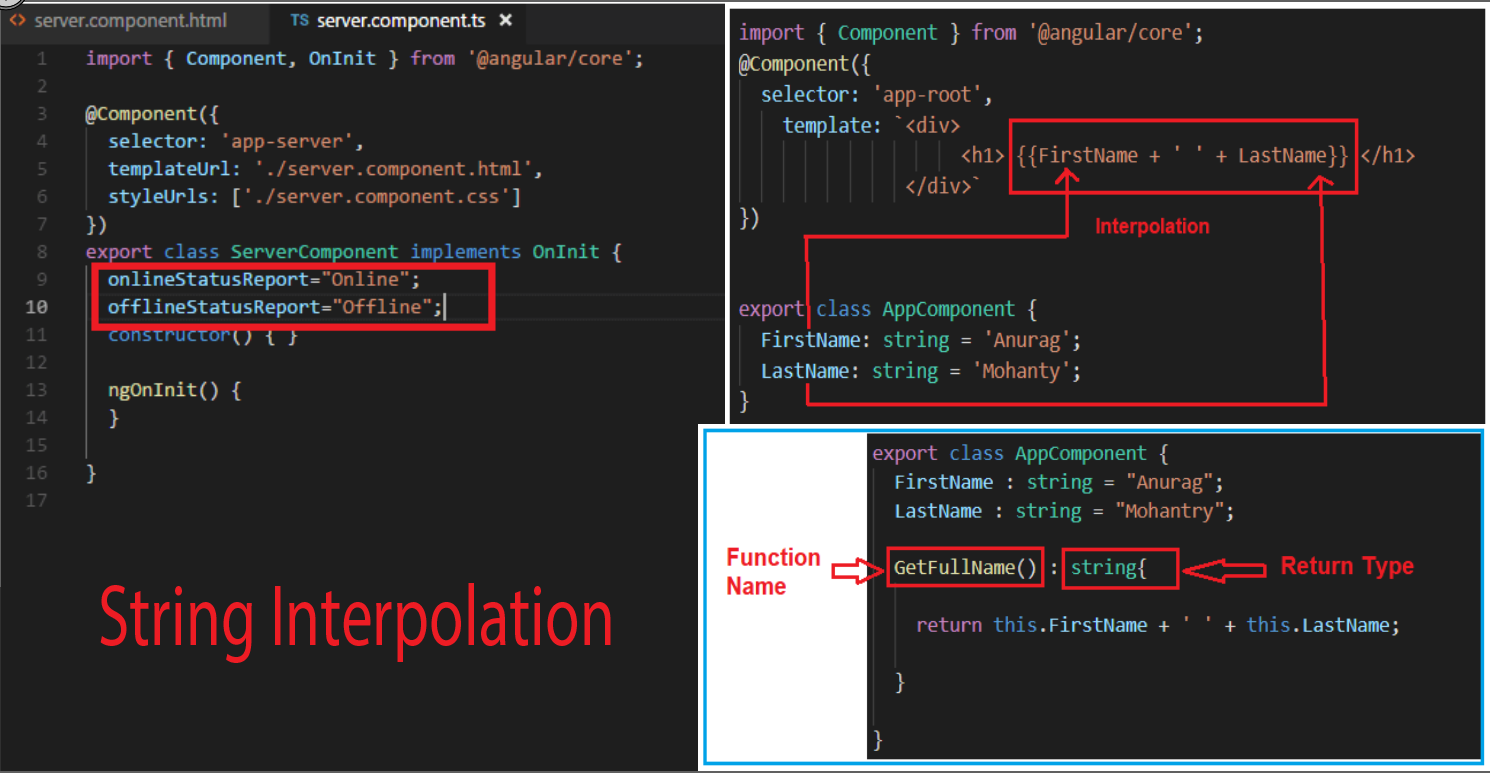
****

****

1. **String Interpolation**

Interpolation refers to embedding expressions into marked up text. By default, interpolation uses the double curly braces {{ and }} as delimiters.

To illustrate how interpolation works, consider an Angular component that contains a currentCustomer variable:

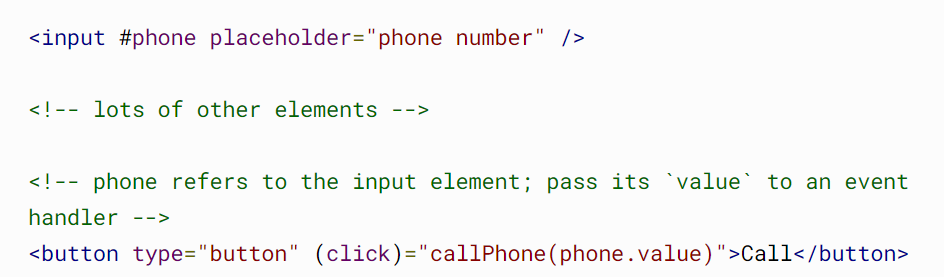


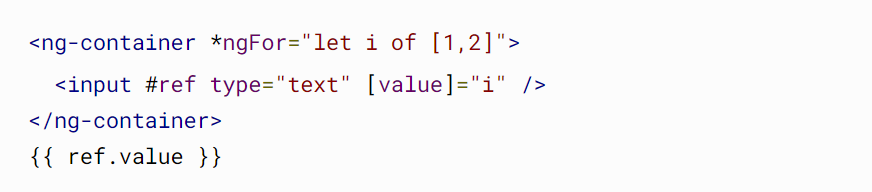
1. **Template Variables**

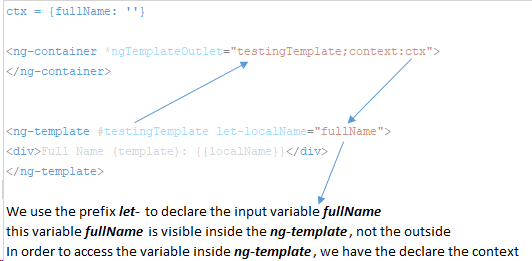
Template variables help you use data from one part of a template in another part of the template. Use template variables to perform tasks such as respond to user input or finely tune your application's forms.

A template variable can refer to the following:

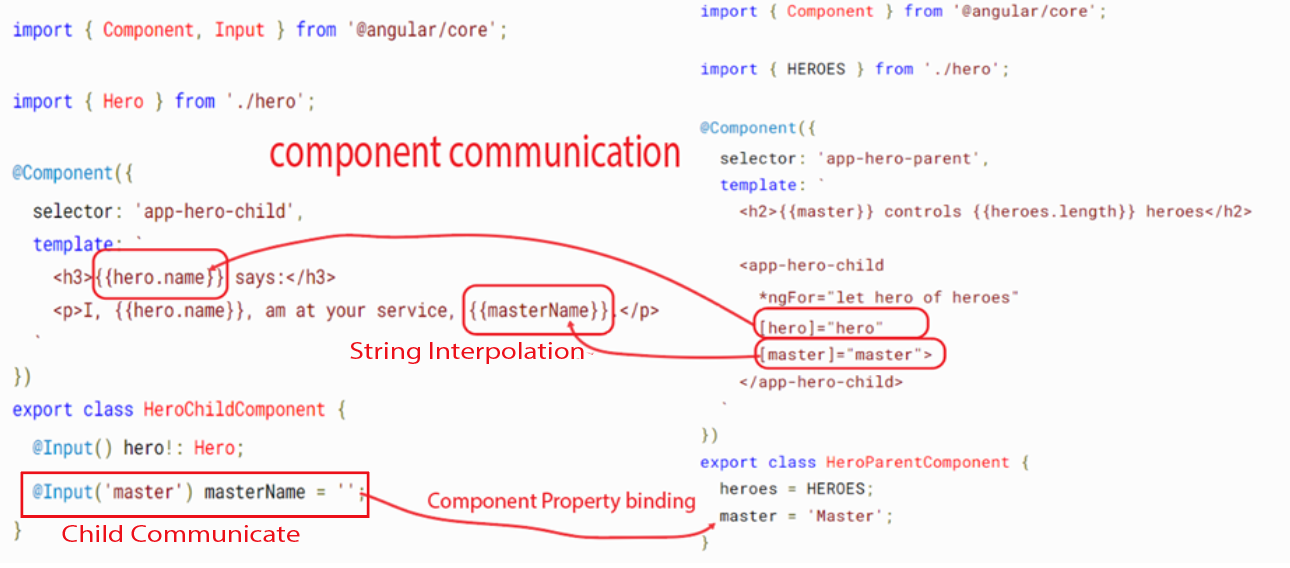
* a DOM element within a template
* a directive
* an element
* [TemplateRef](https://angular.io/api/core/TemplateRef)
* a [web component](https://developer.mozilla.org/en-US/docs/Web/Web_Components)





****

1. **Component Communications**



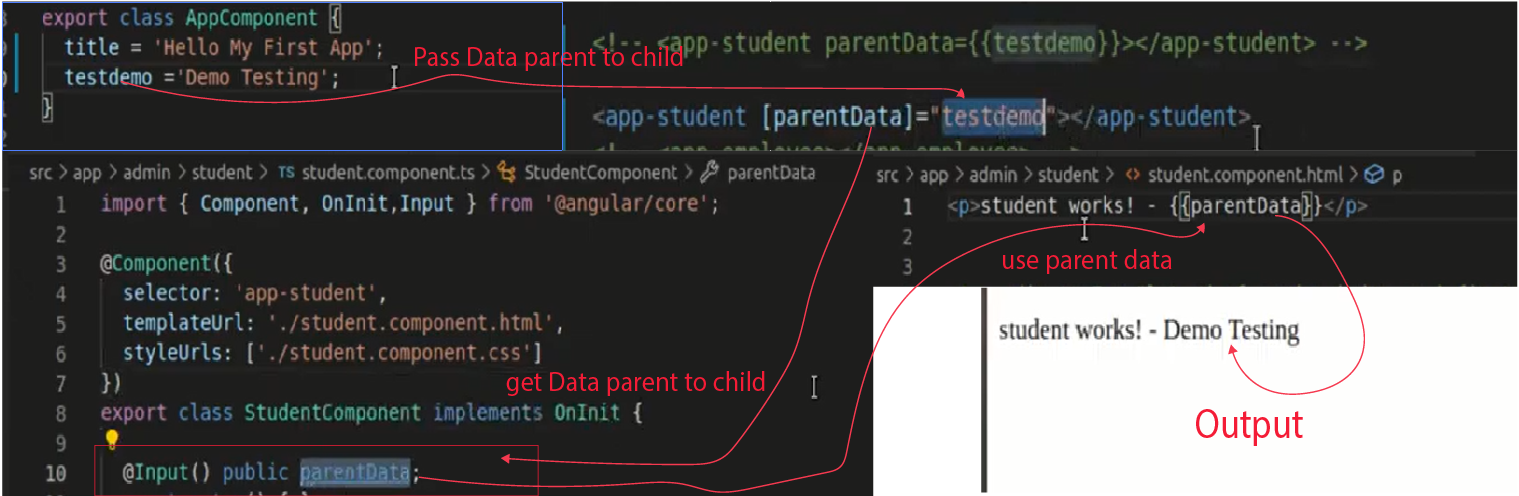
1. **Input and Output**

A common pattern in Angular is sharing data between a parent component and one or more child components. Implement this pattern with the @[Input](https://angular.io/api/core/Input)() and @[Output](https://angular.io/api/core/Output)() decorators.

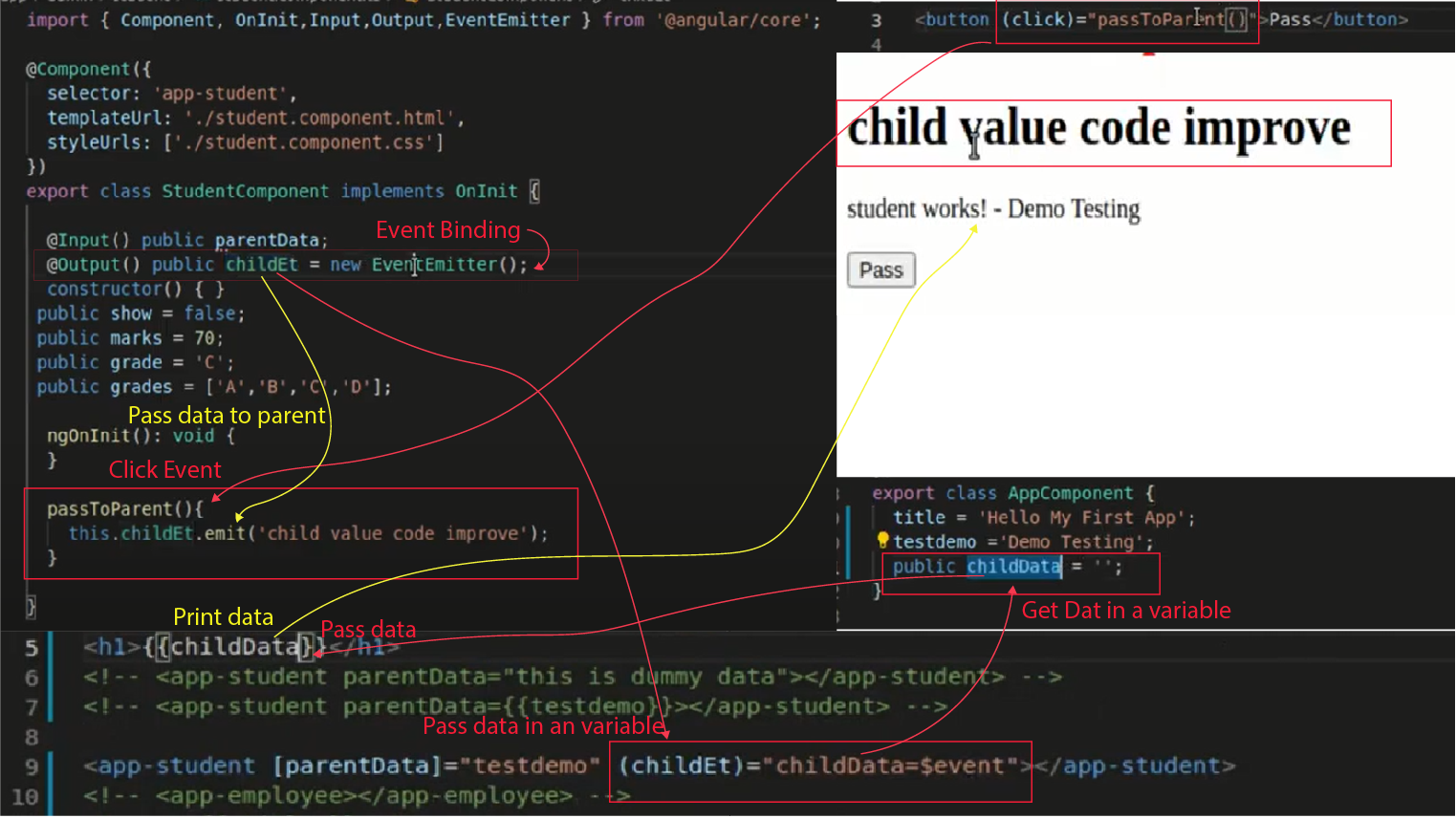
The <parent-component> serves as the context for the <child-component>.

@[Input](https://angular.io/api/core/Input)() and @[Output](https://angular.io/api/core/Output)() give a child component a way to communicate with its parent component. @[Input](https://angular.io/api/core/Input)() lets a parent component update data in the child component. Conversely, @[Output](https://angular.io/api/core/Output)() lets the child send data to a parent component.

@input



@Output

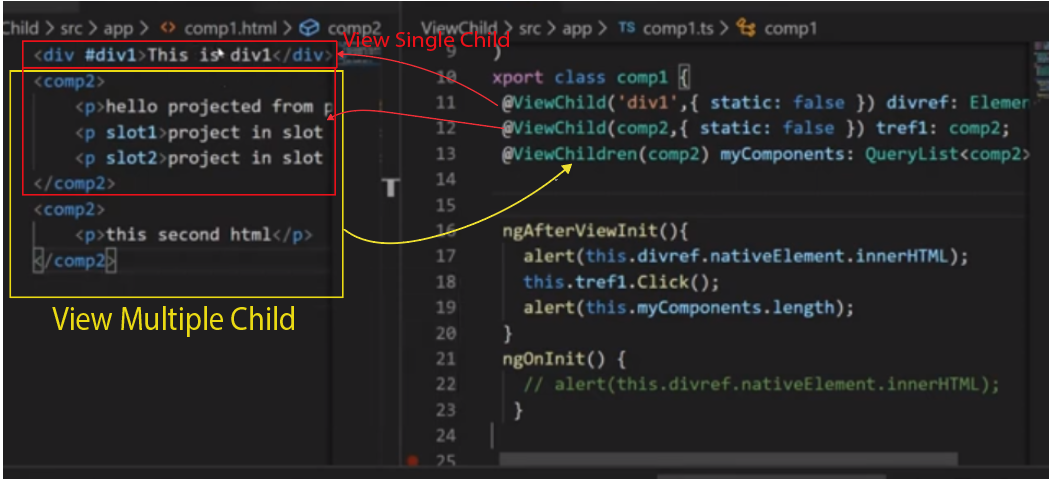


**Read more** [**https://tutorialslink.com/Articles/Exploring-@Input()-and-@Output()-Decorator-in-Angular-/1137**](https://tutorialslink.com/Articles/Exploring-@Input()-and-@Output()-Decorator-in-Angular-/1137)

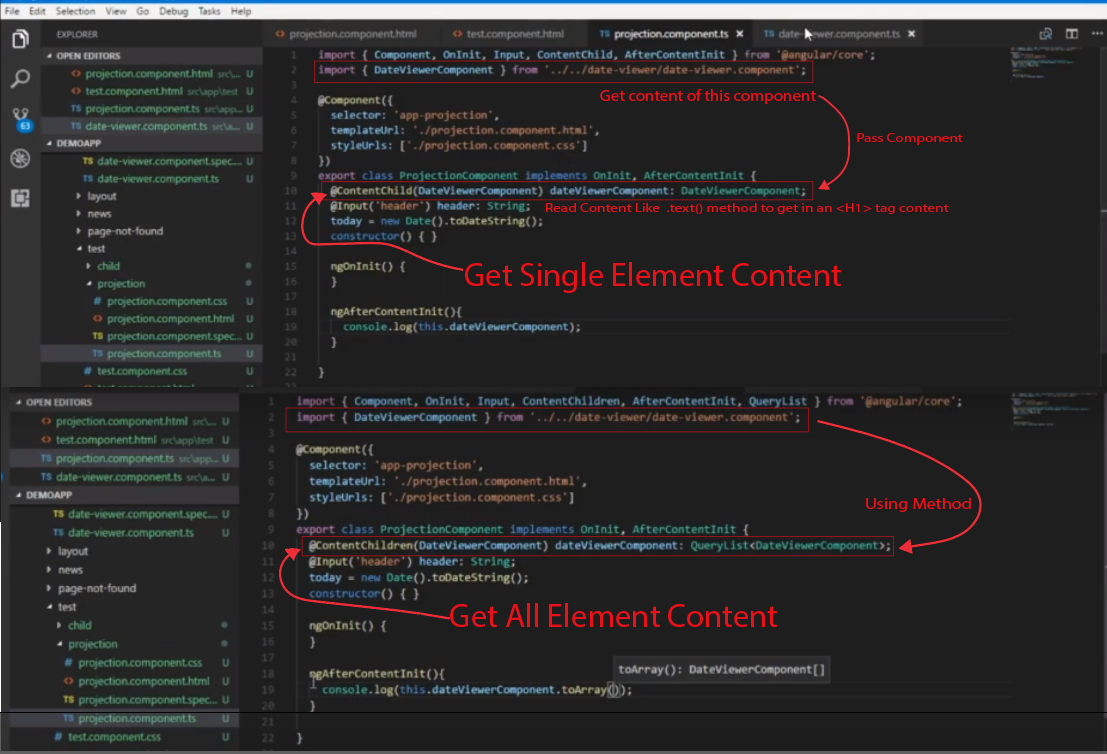
[**https://angular.io/guide/inputs-outputs**](https://angular.io/guide/inputs-outputs)

1. **View Child**

**View Child, View Children**



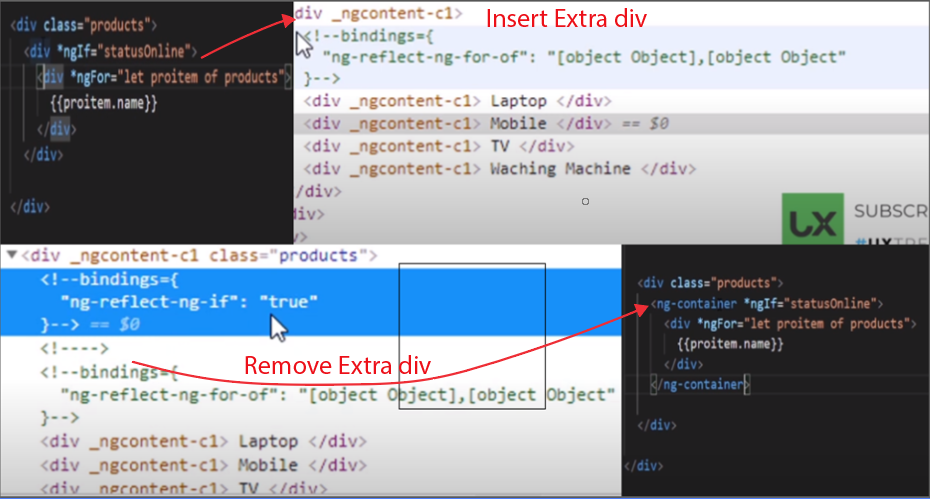
**Content Child, Content Children**



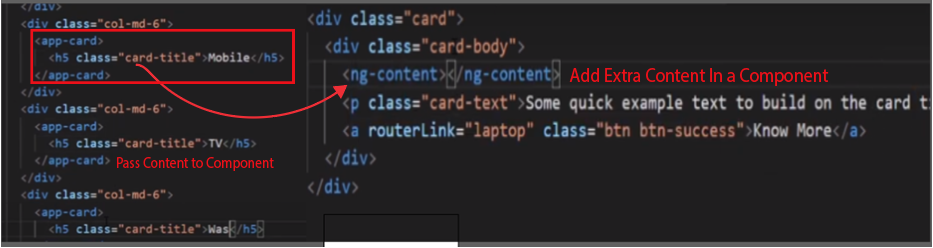
1. **Content Projection**

**Content projection mean in the component add content with externally. Externally content addition method**

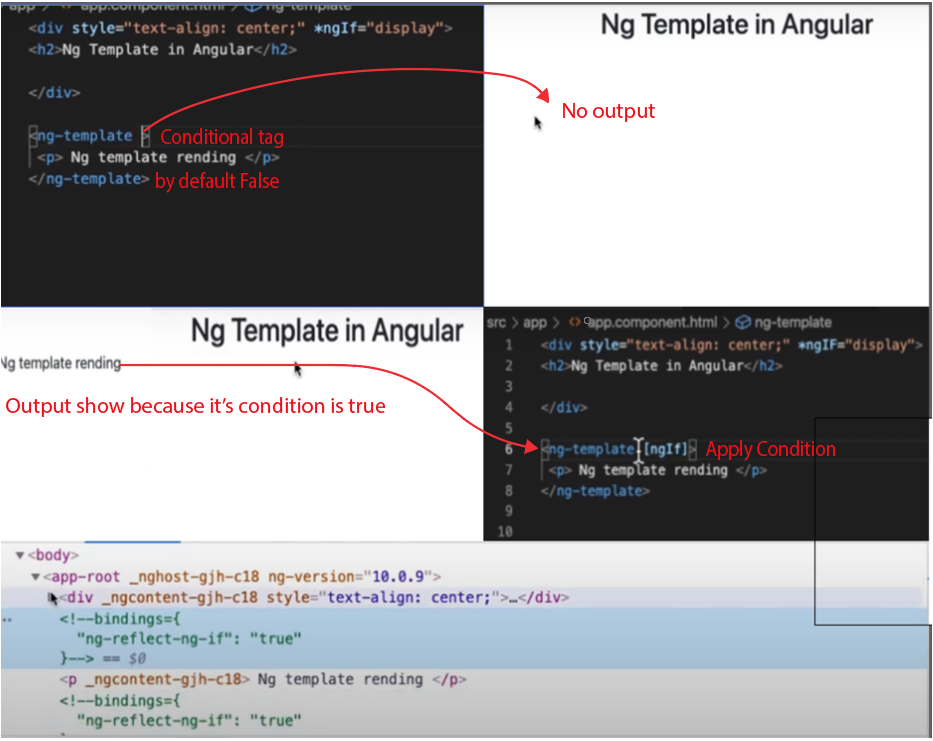
**Ng-Container**



**Ng-Content**

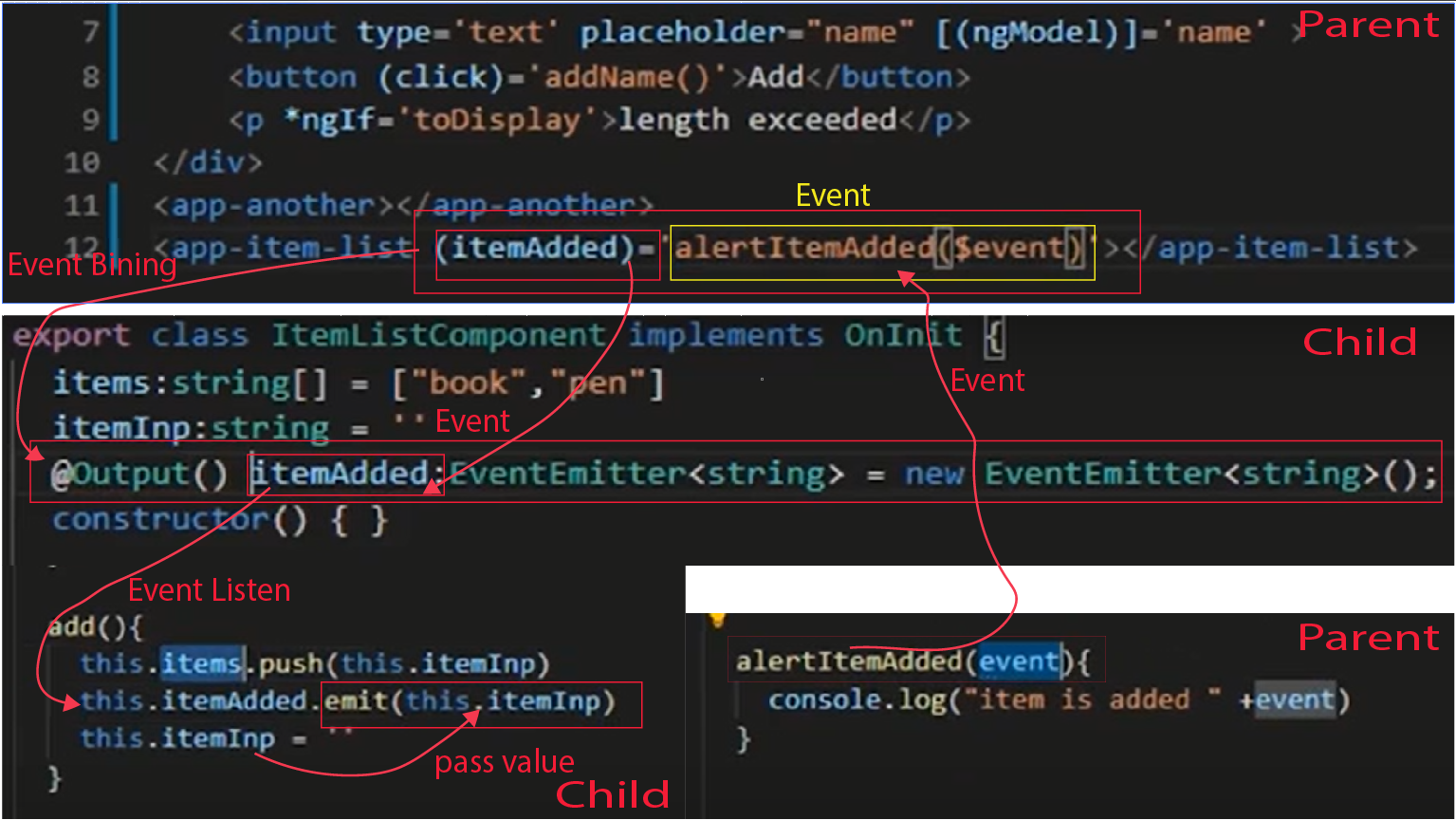


**Ng-template**



1. **­** **Events and Event Emitters**

**Event Mean just an action like click, trigger Emitter mean bind even to another component. Like that**



1. **Listening to Event Emitters**
2. **Container & Presentational Components**

**Container mean a main component and this container component pass data to other component. So the other component are Presentational Components**

1. **Smart & Dumb Components**

**Container behave like a smart component and presentational component behave like a dumb component**