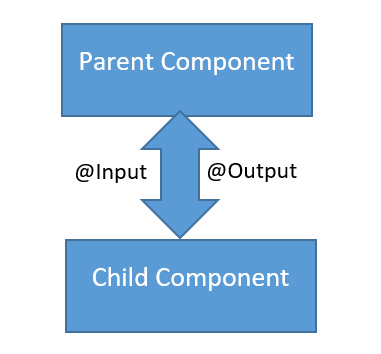
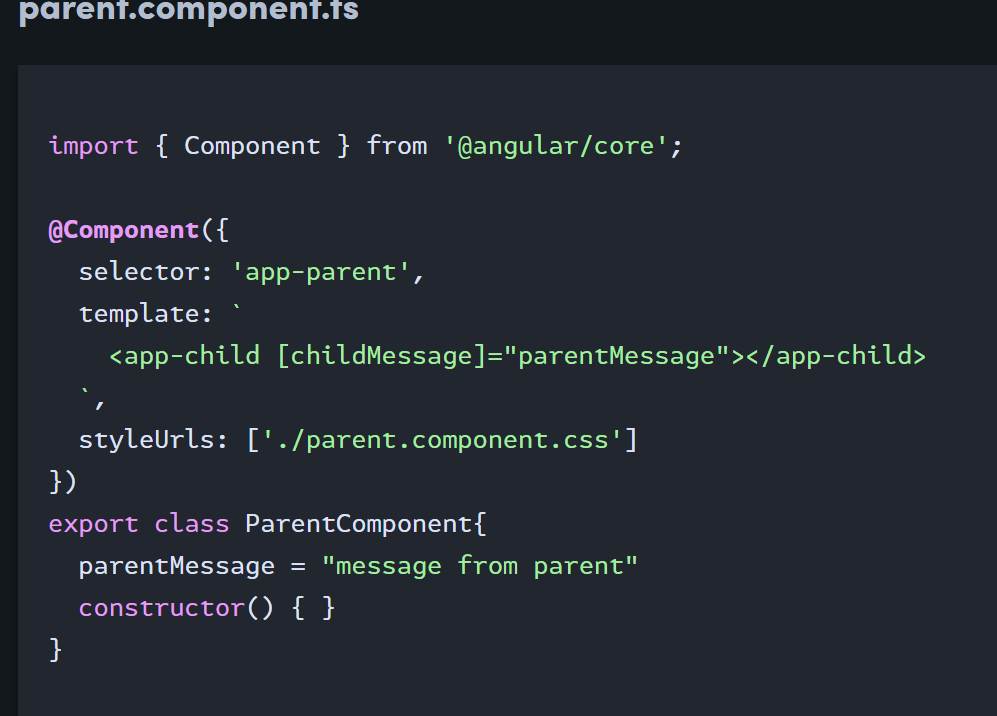
In Angular, a component can share data and information with another component by passing data or events. A component can be used inside another component, thus creating a component hierarchy. The component being used inside another component is known as the child component and the enclosing component is known as the parent component.  Components can communicate to each other in various ways, including:

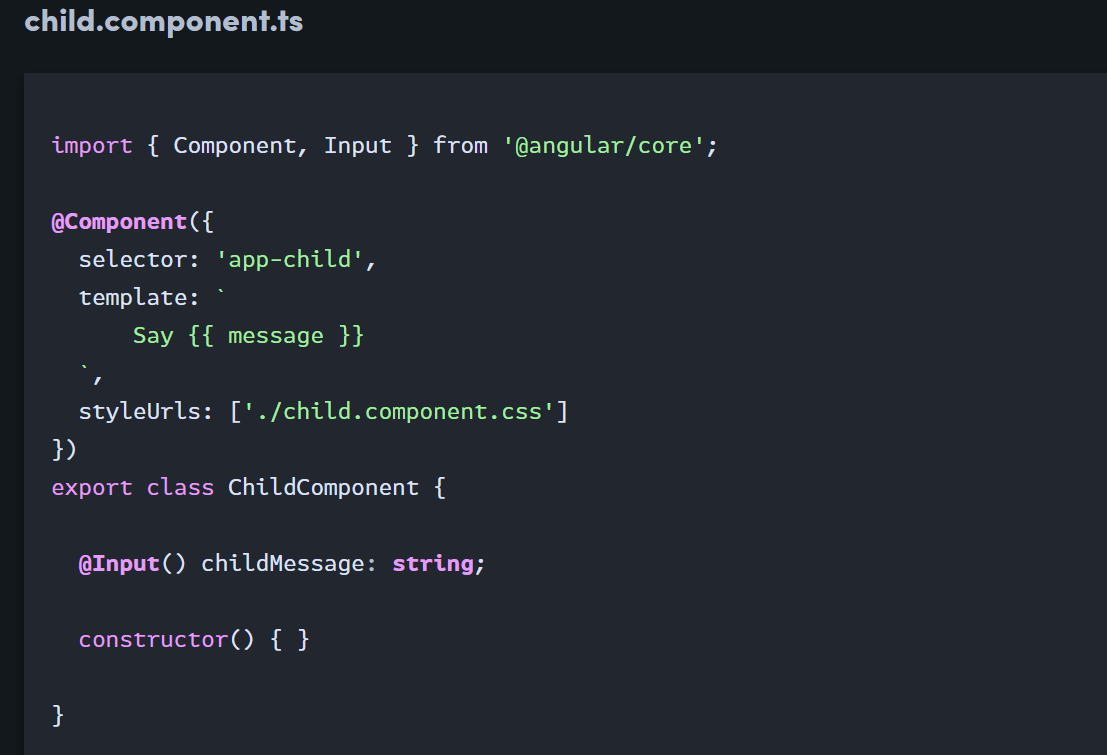
1. Using @Input()
2. Using @Output()
3. Using Services
4. Parent component calling ViewChild
5. Parent interacting with a child using a local variable



**Parent to Child: Sharing Data via Input**

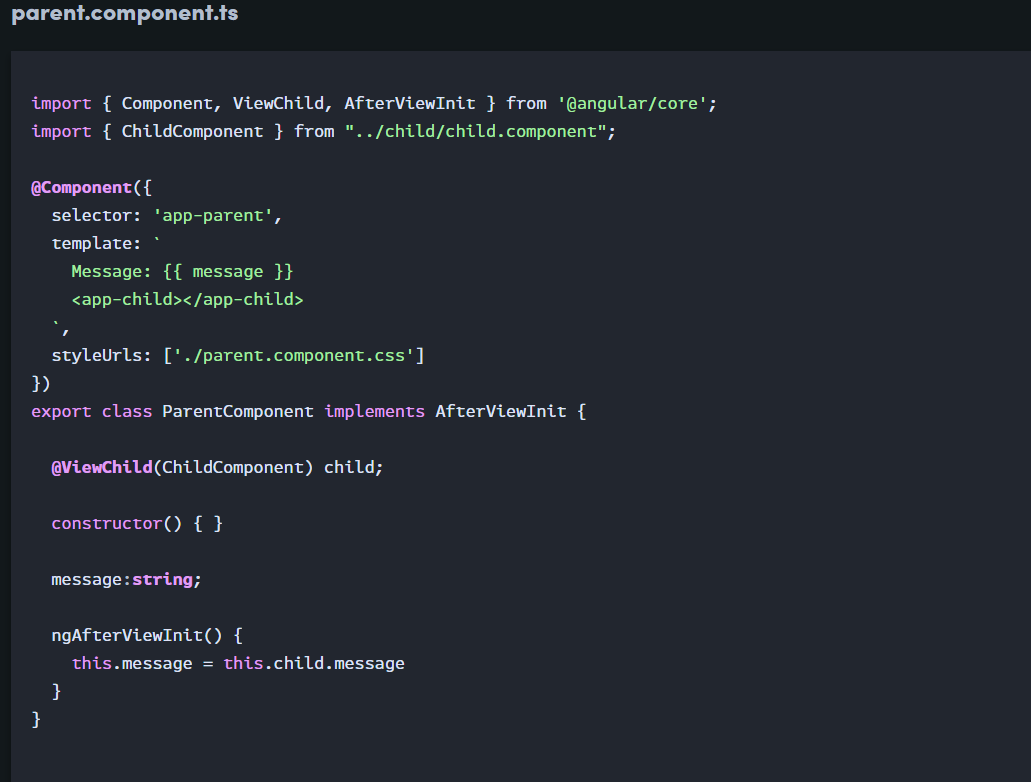
This is probably the most common and straightforward method of sharing data. It works by using the @Input() decorator to allow data to be passed via the template.

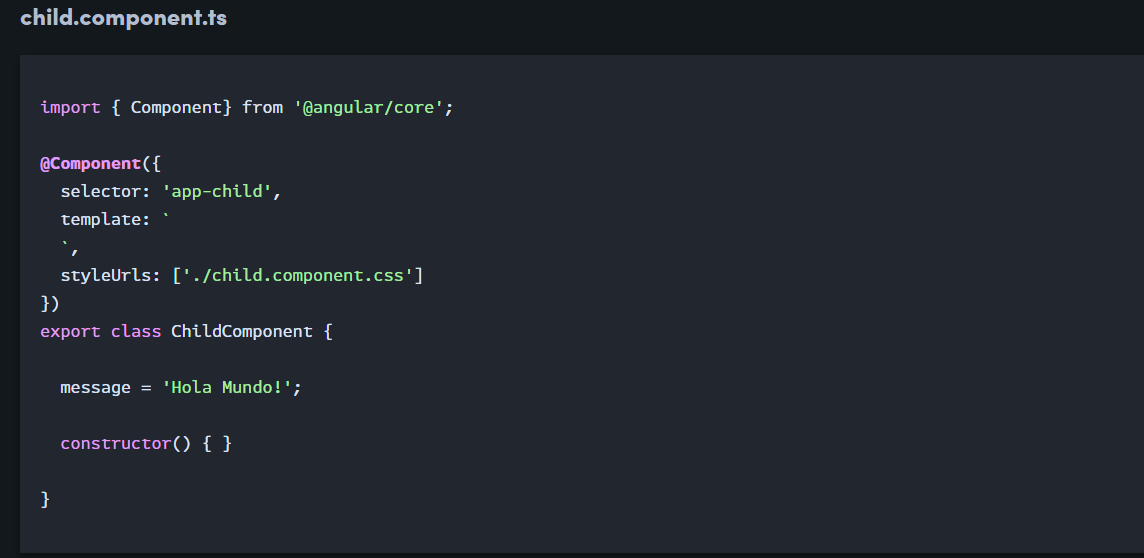




**Child to Parent: Sharing Data via ViewChild**

[**ViewChild**](https://angular.io/api/core/ViewChild) allows a one component to be injected into another, giving the parent access to its attributes and functions. One caveat(warning), however, is that child won’t be available until after the view has been initialized. This means we need to implement the AfterViewInit lifecycle hook to receive the data from the child.





**Child to Parent: Sharing Data via Output() and EventEmitter**

Another way to share data is to emit data from the child, which can be listened to by the parent. This approach is ideal when you want to share data changes that occur on things like button clicks, form entires, and other user events.

In the parent, we create a function to receive the message and set it equal to the message variable.

In the child, we declare a messageEvent variable with the Output decorator and set it equal to a new event emitter. Then we create a function named sendMessage that calls emit on this event with the message we want to send. Lastly, we create a button to trigger this function.

The parent can now subscribe to this messageEvent that’s outputted by the child component, then run the receive message function whenever this event occurs.

