The idea of Angular input and output is to exchange data. Angular components are able to send and receive information via input and output properties. Any given component can have as many input and output properties as needed for the application component. If a component needs to receive data you would use the @input() property by binding the binding values to the corresponding input. The input doesn’t need to know who it’s receiving the data from, it just needs to know what to do with it after receiving it. When the component needs to send data, it can use event emitter objects through the @output(). The beauty of the output is that it doesn’t need to know where to send the data. Whoever needs the data can subscribe to the events that a component emits.

Components tend to have a hierarchy structure to them as you build your application. With this structure the @input() decorator typically receives its data from parent components as the opposite it true for @output sends data from the current component to the parent ones. For example a shopping cart can receive data via @input() from the store front component and @output data to the checkout component. The following is an example of and @input() decorator: @Input() myName: string;. Here is now an example of an @Output(): @Output myEvent: EventEmitter = new EventEmitter();

Sitepoint.com/angular-2-components-inputs-outputs/

Medium.com/@foolishnew/understanding-input-output-and-eventemitter-in-angular-c1aeb9fff594