**Everything there is to know about components - Angular**

**The structure of a component –**

Any component should be composed from number of parts, the first part is the \*component-name\*.component.ts file, this file will have an import of the component from the angular/core, a component selector that gets an object that contains kind of important data to the component and export of the component as a class. It looks like that:



**selector –** it’s the HTML tag that we will use when we want to use our new component. The selector mechanism is like the css one, but selecting by ID or pseudo selector won't work.

**templateUrl / template –** the actual HTML code that we insert into the component and will be shown whenever we use the new component we are creating. If we want to use a file we will choose - templateUrl else for inline HTML style we will use - template.

**stylesUrl/styles –** same as the template selector but only with CSS.

**How to create a new component?**

There are 2 ways to create a new component, manually or with the CLI, let's go through them:

**The manual way:**

In this way, we will create the files ourselves, first of all lets create a new folder inside our app folder, the folder name will be our component name. Example: warning-alert, inside that folder lets create these files: warning-alert.component.ts/html/css in each of them we will find only the relevant pieces of code as we expect by the file name.

After that, we should import our new component to the app.module.ts file to the declarations array.

**The CLI way:**

we can you the following commands:

ng generate component \*component-name\* / ng g c \*component-name\*.

**Component Lifecycle:**

Inside the export class we can see that our new component implements OnInit, and has a constructor. Inside the constructor, we can write code that will run at the point of time when this component is being created by Angular.

**How to Send & Receive data from components:**

To send and receive data from components we will use custom property & event binding to our components.

**Custom Property Binding:**

By default, all properties of components are only accessible inside the component itself and not outside. If we want to expose the property to the world, we need to use decorators.

**@Input –** First we will need to import this decorator from angular/core, second, we will add the decorator itself near the property we want to expose like this:



Now, every parent component that uses our component will be able to bind to our element property like this:



And that’s how you can pass data down from a father to his child by property data binding.

**Custom Event Binding:**

**@Output -** To pass data from component outside (from child to parent component), we would use event so we need to create a custom event. First lets import Output from angular/core so the event will be exposed to the parent component, second lets import the EventEmitter (from angular/core also) so that we could create the custom event.

In the export part of the component we will add for each event we want to create a property with the @Output decorator in the start of the line, then we will assign a new EventEmitter to it from the type of the object we want to pass outside – this is the actual place we decide how our data will come out from this event.

Than we will create functions that will use the properties of the events and assign the values we want to pass up. Here is an example:

In the parent component