Angular 8 : Online Class

Class – 6

By: Sahosoft Solutions

Presented by : Chandan Kumar

Interpolation

Before we get to know interpolation, we need to know the data Binding in Angular. Let's start with the data Binding.

The data binding is a powerful feature of Angular, that allows us to communicate between the component and its view. The data binding can be a one-way data binding [angular interpolation / string interpolation, linking properties, event linking] or a two-way data binding.
In the one-way data binding, the model value is inserted into an HTML element (DOM) and the model cannot be updated from the view. In the two-way binding, the automatic synchronization of data occurs between the Model and the View (each time the Model changes, it will be reflected in the View and vice versa)

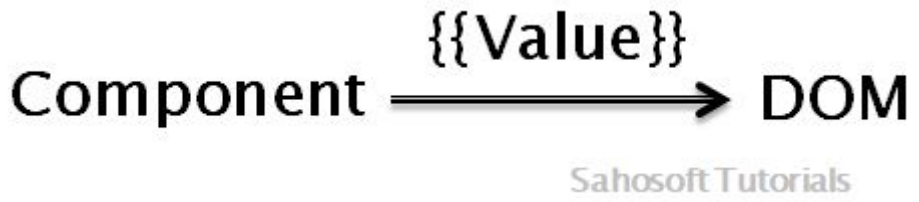
Interpolation

Angular has four types of Data binding

1. **Interpolation / String Interpolation** (one-way data binding)
2. **Property Binding** (one-way data binding)
3. **Event Binding** (one-way data binding)
4. **Two-Way Binding**

Interpolation

Interpolation is a technique that allows the user to bind a value to an element of the user interface. Interpolation binds data in one-way. This means that when you change the value of the bound field by interpolation, it is also updated on the page. The field value cannot be changed. An object of the component class is used as a data context for the component template. Therefore, the value to be bound in the view must be assigned to a field in the component class.



Selector

The selector attribute allows us to define how Angular is identified when the component is used in HTML.It tell Angular to create and insert an instance of this component where it finds the selector tag in the Parent HTML file in your angular app.

The selector accepts a string value, which is the CSS selector that Angular will use to identify the element. We will use it in html to place this component where we want. If you see the index.html file, you will find **<app-root></app-root>** component inside the body.

The component selector is a CSS selector that identifies how Angular finds this particular component in any HTML page. In general, we use element selectors **<app-root></app-root>**), but it can be any CSS selector, from a CSS class to an attribute as well.

The component is applied to the **<app-root></app-root>** tag in index.html. If index.html does not have that tag, Angular will fail at startup. You can check where the angular application will be played.

Selector

Note that it has a prefix with the app, which is added by the Angular CLI by default, unless otherwise specified.

The recommended practice when creating new components is to use element selectors (as we did with 'app-root'), but technically any other selector can be used.

You can make the selector as simple or complex as you want, but as a general rule, try to limit yourself to simple element selectors unless you have a very strong reason for not doing so.

The basic component only needs a selector (to tell Angular how to find the instances of the component being used) and a template (which Angular should render when it finds the element). All other attributes in the component decorator are optional. In the previous example, we defined that the AppComponent will be rendered whenever Angular finds the app-root selector and that the app.component.html file is displayed when it finds the item.

Selector

For example, here are some ways you can specify the selector attribute and how to use it in HTML

- element-name: Select by element name.
- .class: Select by class name.
- [attribute]: Select by attribute name.

Encapsulation

Angular Components

The Angular components are consist of three things:

- Component class
- Template
- Style

The combination of these three factors makes an angular component reusable in an application. In theory, when you create a component, you create a Web component in some way (the angular components are not Web components) to take advantage of the Shadow DOM. You can also use Angular with browsers, which are not compatible with Shadow DOM because Angular has its own emulation and can emulate Shadow DOM.

Encapsulation

To emulate the shadow DOM and encapsulate styles, Angular provides three types of view encapsulation. Are the following:

Emulated (default): The main HTML styles are propagated to the component. The styles defined in the component's @Component decorator are limited to this component only.

Native: The main HTML styles are not propagated to the component. The styles defined in the component's @Component decorator are limited to this component only.

None: the component styles are propagated to the main HTML and therefore are visible to all components of the page. with apps that have None and Native components in the application.. All components with encapsulation None will have their duplicate styles on all components with native encapsulation.

Encapsulation