ANGULAR 2

Angular v2.2.1

[Home]

[Why Angular2?]

[Components]

[Inputs]

[Outputs]

[Lifecycle]

[Templates]

[Events]

[Forms]

[ViewChild]

ES6/TypeScript

TEMPLATES

[Suggest improvements]

[Tweet]

Templates are very similar to templates in Angular 1, though there are many small syntactical changes that make it more clear what is happening.

A SIMPLE TEMPLATE

Let's start with a very simple template that shows our name and our favorite thing:

```
<div>
  Hello my name is {{name}} and I like {{thing}} quite a l
</div>
```

`{}`: RENDERING

To render a value, we can use the standard double-curly syntax:

```
My name is {{name}}
```

Pipes, previously known as “Filters,” transform a value into a new value, like localizing a string or converting a floating point value into a currency representation:

`[]` : BINDING PROPERTIES

To resolve and bind a variable to a component, use the `[]` syntax. If we have `this.currentVolume` in our component, we will pass this through to our component and the values will stay in sync:

```
<video-control [volume]="currentVolume"></video-control>
```

`()` : HANDLING EVENTS

To listen for an event on a component, we use the `()` syntax

```
<my-component (click)="onClick($event)"></my-component>
```

`[()]`: TWO-WAY DATA BINDING

To keep a binding up to date given user input and other events, use the `[()]` syntax. Think of it as a combination of handling an event and binding a property:

```
<input [(ngModel)]="myName">
```

The `this.myName` value of your component will stay in sync with the input value.

`*`: THE ASTERISK

`*` indicates that this directive treats this component as a template and will not draw it as-is. For example, `ngFor` takes our `<my-component>` and stamps it out for each `item` in `items`, but it never renders our initial `<my-component>` since it's a template:

```
<my-component *ngFor="#item of items">
</my-component>
```

Other similar directives that work on templates rather than rendered components are `*ngIf` and `*ngSwitch`.