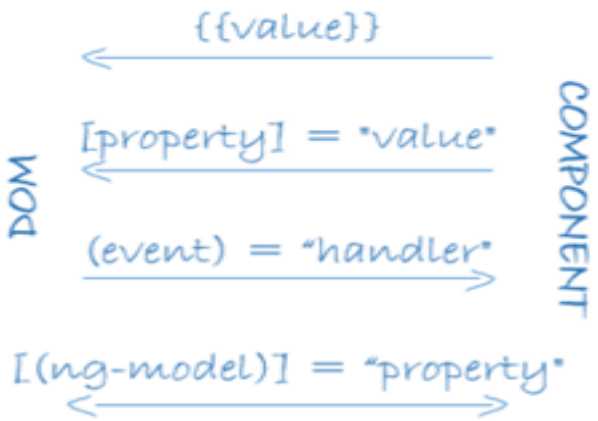
Data binding techniques

These techniques describe data transfer between component and DOM.The following diagram demonstrates it.



The following are data binding techniques implemented by angular framework which can reduce writing lot of code to push and pull data.

Interpolation

Property binding

Event binding

Two-way binding

Interpolation

The easiest way to display a component property is to bind the property name through interpolation. With interpolation, you put the property name in the view template, enclosed in double curly braces:

This one-way data binding from component to DOM.

Syntax:

{{ variable }}

Property binding

Property binding flows a value in one direction, from a component's property into a target element property.

You can't use property binding to read or pull values out of target elements. Similarly, you cannot use property binding to call a method on the target element.

Use property binding to *set* properties of target elements or directive `@Input()` decorators.

Syntax:

[attribute-of- element] = variable

Event binding

If the element raises events, we can listen to them with an event binding. We can use "angular event bindings" to respond to any "DOM events". Many DOM events are triggered by user input. Binding to these events provides a way to get input from the user.

To bind to a DOM event, surround the DOM event name in parentheses and assign a quoted template statement to it.

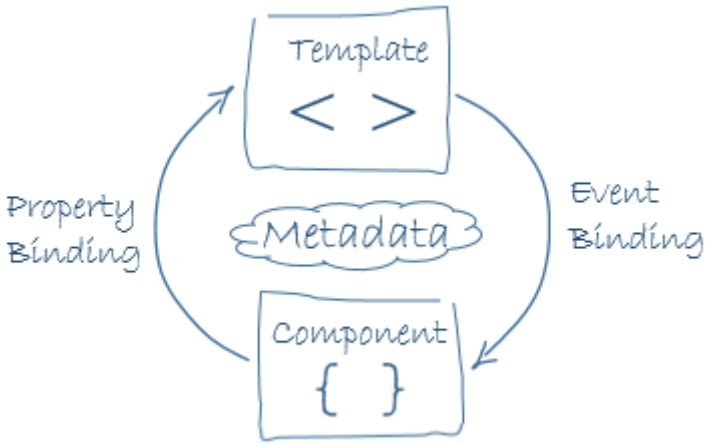
Syntax:

(DOM event)=" template statement"

Two way binding

Two-way data binding combines property and event binding in a single notation.

In two-way binding, a data property value flows to the input box from the component as with property binding. The user's changes also flow back to the component, resetting the property to the latest value, as with event binding.



Syntax:

[(ngModel)]= "template syntax"

Note: Here, "ngModel" is a selector of "NgModel" directive. Creates a `FormControl` instance from a domain model and binds it to a form control element.

The `FormControl` instance tracks the value, user interaction, and validation status of the control and keeps the view synced with the model. If used within a parent form, the directive also registers itself with the form as a child control.

This directive is used by itself or as part of a larger form. Use the `ngModel` selector to activate it.