# Basics of Attributes & Properties

Before we dive into understanding how data is displayed and bound, we should know the difference between HTML Attribute Vs DOM Property

Attributes are defined by HTML.   
Properties are defined by the DOM (Document Object Model).

## What is DOM ?

The HTML DOM is a standard for how to get, change, add, or delete HTML elements.

### What does the DOM do ?

It defines:

* The HTML elements as objects
* The **properties** of all HTML elements
* The methods to access all HTML elements
* The **events** for all HTML elements

### Two important points about HTM DOM:

1. HTML DOM methods are **actions** you can perform (on HTML Elements).
2. HTML DOM properties are **values** (of HTML Elements) that you can set or change.

**The HTML DOM can be accessed with JavaScript or with other programming languages like Angular.**

## Example of DOM:

<html>  
<body>  
  
<p id="myApp"></p>

<img id="myImage" src="" />

<table>

<tr>

<td colspan="2"></td>

</tr>

<tr> <td><button id=”myBtn” onclick="displayDate()">Try it</button></td>  
<td></td>

</tr>

</table>  
<script language="javascript">

document.getElementById("myApp").innerHTML = "Hello Angular 2!"; document.getElementById("myImage").src = "/angularproject/src/app/assets";  
var imgId = document.getElementById("myImage").id ;

document.getElementById("myBtn").onclick = displayDate;

function displayDate() {

document.getElementById("myApp").innerHTML = Date();

}

</script>  
  
</body>  
</html>

In the example above, getElementById is a method, while innerHTML is a property.

In Angular Template binding works with properties and events, not attributes.

# Data Binding and Syntax

The syntax used in the html template to accept values from and return values to, its corresponding component class is called “Template Syntax”

## Template Syntax:

3 Types:

### Interpolation

The component property is enclosed in 2 curly braces.

Syntax:

1. Interpolation used between HTML element tags

<h3> {{title}} // title is the name of the property in the component class

1. Within attribute assignments

<img src="{{heroImageUrl}}" style="height:30px"> </h3>  
// heroImageUrl is the name of the property in the component class

### Template Expressions:

A template expression produces a value. Angular executes the **expression** and assigns the value to a **DOM property** of a binding target.   
The target might be an HTML element, a component, or a directive.

**Syntax:**

[property]="expression"

The DOM Property is marked in **square brackets**

Expression can be:

* Properties in component class
  + <img [src]="**myImageUrl**">
* Template reference variables
  + <input **#phone** placeholder="phone number"> ----- -------  
    <button (click)="callPhone(**phone**.value)">Call</button>
* Template input variables
  + <div \*ngFor="let **course** of courses">{{**course**.name}}</div>

Property can be:

* A DOM property
* <img **[src]**="myImageUrl">  
  *src is a DOM property for the img element.*
* A Directive
* <div **[ngClass]**="classes"> </div>  
  *The ngClass is an angular defined directive. Its function is to add/remove classes on the target element.*
* A Model
* <product-detail **[product]**="currentProduct"></product-detail>  
  *product-detail is a reference to child-component within a parent component’s html template  
  product in the square brackets is a property in that product-detail component.*

### Template Statements:

A template statement responds to an event raised by a binding target. The target might be an HTML element, a component, or a directive.

(event)="statement"

The DOM Event is enclosed in **parantheses**.

Event can be:

* The DOM event like “click, mouseover, mouseout, drag, load..”
  + <button **(click)** ="addToCart()">Add Item </button>
* A custom event from a custom directive.
  + <div **(myClick)** ="clickMessage=$event" clickable>click with myClick</div><p>{{clickMessage}}</p>

Statement can be:

* A method
  + <button (click)= **"addToCart()"**>Add Item </button>  
    *addToCart is a method in the component class.*

The method can take input values like below:

* Using an $event object:   
  <button (click)="onSave($event)">Save</button>
* Using a template input variable:   
  <button \*ngFor="let hero of heroes" (click)="deleteHero(hero)">{{hero.name}}</button>
* Using a template reference variable:   
  <form #heroForm (ngSubmit)="onSubmit(heroForm)"> ... </form>

\*\*\* Note: template statement supports = in the statement part of the syntax. *The template expressions do not support = in the expression part:*  
<div (myClick) ="clickMessage=$event" clickable>click with myClick</div><p>{{clickMessage}}</p>

## Data Binding

### Data Flow and Syntax

3 Ways of Data flow:

1. Data flows from data-source (component class) to html template ( the html elements) . The template syntax is 2 types:
   1. Interpolation {{expression}}
   2. Template Expressions [target]=”expression”
2. Data flows from html template to data-source. The template syntax is of 1 type:
   1. Template Statements. (target)=”statement”
3. Data flows in both directions. Combination of the above syntaxes: ([target])=”expression”

### Types of Binding:

Property Binding

The property binding has been discussed in section 2.1.2.1

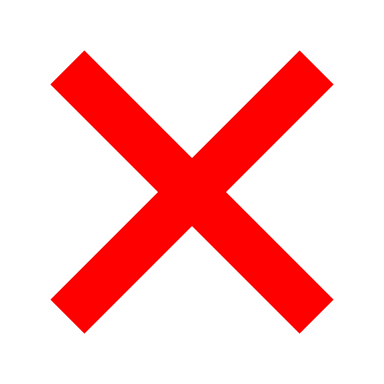
Attribute Binding

In this binding, you can create and set a html attribute. We need attribute property when there is no corresponding dom property for that html attribute. For example:

<tr><td **[attr.colspan]**="1 + 1">One-Two</td></tr>

We need to do it the above way using attr, bcoz there is no corresponding DOM property for colspan html attribute.   
And hence, a normal attribute setting for colspan will throw an error.

*Is good:* <img src={{myImagePath}} />

*Gives error:* <tr><td colspan="{{1 + 1}}">Three-Four</td></tr>

Style Binding

Syntax: **style.style-property**

**Examples:**

* <button [style.color]="isSpecial ? 'red': 'green'">Red</button>
* <button [style.background-color]="canSave ? 'cyan': 'grey'" >Save</button>
* <button [style.font-size.em]="isSpecial ? 3 : 1" >Big</button>

Class Binding

Syntax: **class.class-name**

Examples:

<div [class.specialStyle]="isSpecial">The class binding is special</div>

The isSpecial is a component property. If this expression "isSpecial" evaluates to true, the class "specialStyle" is added to the div element. If it is false it is removed.

Event Binding

Syntax: **(event)="statement"**

In this binding the flow data is from a html element to a component.

The target event can be a simple event belonging to an element or a property of a custom directive.

Example:

If the target is an event of an element:   
<button (click)="onSave()">Save</button>

If the target is an output property of a directive:  
<div (myClick)="Save()">Click Me</div>

Event binding results in an event-object $event. It carries info about the event and data values.

**For example:**

|  |
| --- |
| **HTML Template:**  <input [value]="productItem" (input)="productName=$event.target.value" >  <input [value]="productItem" (input)="displayInput\_method1($event)" /> |  <input [value]="productItem" #myInput (input)="displayInput\_method2(myInput.value)" />  <hr/>  <div>  {{productName}}  {{InputProductItem\_method1}} |  {{InputProductItem\_method2}}  </div>  --------------------------------------------------------------------------  **Component class:**  ProductName="Bangles"  displayInput\_method1(event: KeyboardEvent) {  if (event) {  let targetElement = <HTMLInputElement>event.target  this.InputProductItem\_method1 = targetElement.value  }  }  displayInput\_method2(value: string) {  this.InputProductItem\_method2 = value  } |

Two Way Binding

We need two way binding when we want to both display a data property and update that property when the user makes changes.

Syntax**: [(x)]**

Example:

In the below example, the ngModel data property sets the element's value property and the ngModelChange event property listens for changes to the element's value.

|  |
| --- |
| <div>  <input [value]="productItem" (input)="productItem=$event.target.value"><br/> {{productItem}}  <hr/> Simpler way of writing:  <input [ngModel]="productItem" (ngModelChange)="displayInput\_method4($event)" />  <br> From two way binding input: {{productItem}}  <hr/> Even Simpler way of writing:  <input [(ngModel)]="productItem" />  <br> From two way binding input: {{productItem}}  </div> |