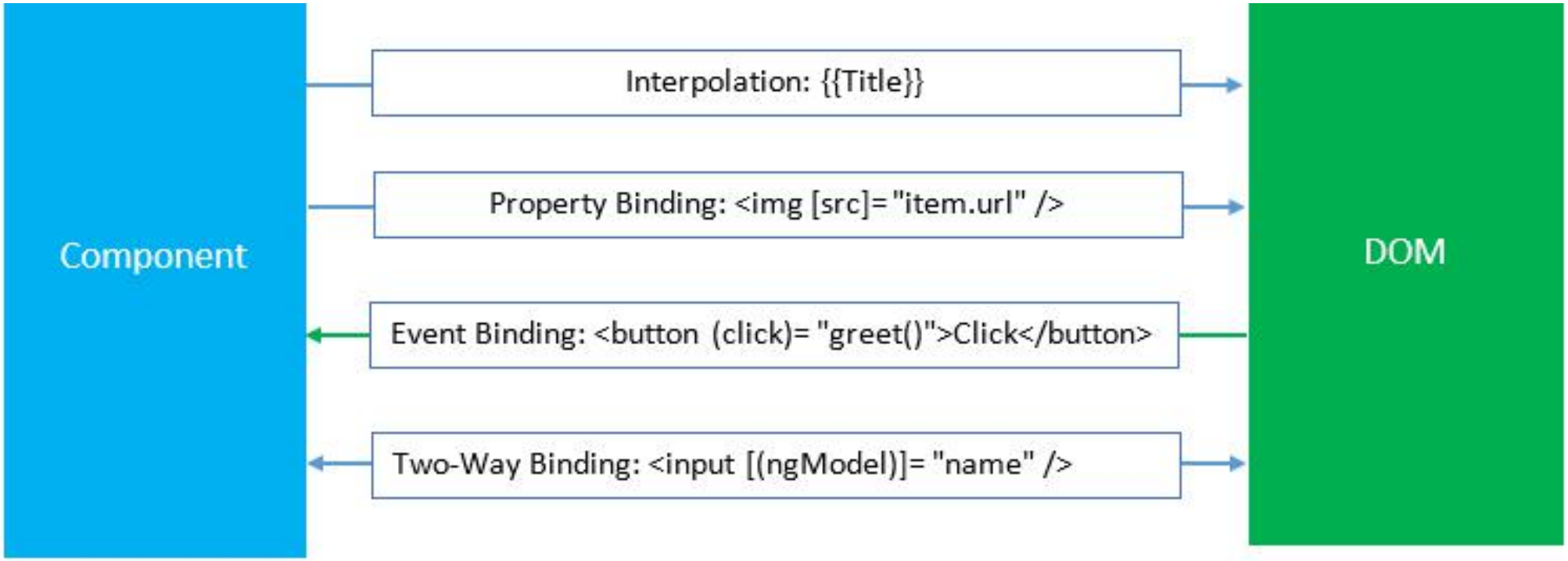
# Databinding

# Databinding

- A mechanism for binding data values to HTML elements and turning user activities into actions
- Responsible for data value updates
- Support following Databinding:
  - Interpolation
  - Property Binding
  - Two-way Binding
  - Event Binding

# Data Binding Syntax



# Directives

- Used to extend the power of existing Html markup
- Change the appearance or behavior of a DOM element
- There are four kinds of directives in Angular2
  - Components – A directive with a template. @Component decorator extends the @Directive decorator with template-oriented features
  - Structural directives – Alter layout by adding, removing, and replacing elements in DOM Eg. *ngIf, *ngFor, *ngSwitch
  - Attribute directives – Alter the appearance or behavior of an existing element. Eg. ngModel, ngStyle, ngClass
  - Custom directives – Your own directive

# Custom Directive

- @Directive decorator is used to create a custom directive
- Custom directive can be used as an attribute or class

```
@Directive({
  selector: '[Highlight],.Highlight'
})
export class HighlightDirective {
  constructor(private el: ElementRef) { }
  @HostListener('mouseover') onmouseover() {
    this.el.nativeElement.style.backgroundColor = 'yellow';
  }
  @HostListener('mouseleave') onmouseleave() {
    this.el.nativeElement.style.backgroundColor = null;
  }
}
```

# Pipes

- Used to transform bound properties before displaying
- Angular built-In Pipes are :
  - Lowercase
  - Uppercase
  - Date
  - Currency
  - Json
  - Decimal
  - Async etc.

# Custom Pipe

- A custom pipe can be used for applying you own logic to transform text
- @Pipe decorator is used to create a custom pipe

```
@Pipe({
  name: 'reverse'
})
export class ReversePipe implements PipeTransform {
  transform(value: any, args?: any): any {
    let result = '';
    for (let i = 0; i < value.length; i++) {
      result = value[i] + result;
    }
    return result;
  }
}
```