Control flow

Allows you to use certain JavaScript-like structures directly in your HTML. They start with an @ symbol followed by the code.

@if block

The @else is optional.

```
@if ( isValid() ) {
  <p>Form valid</p>
}
@else {
  <p>Form invalid</p>
}
```

@for block

This repeats a block of the template once for each element in an array.

```
@for(r of recipes; track r.id) {
  <div>
    <h1>{{ r.name }}</h1>
    <p>Cost: {{ r.cost | currency }}</p>
  </div>
}
```

track

Track provides a key used to associate array items with elements in the DOM. It will increase performance by helping Angular match up pre-existing elements as items are added, removed, etc.

The key must be a unique and unchanging value for each element in the collection. Thus, database IDs are perfect. If there isn't anything good to use for the key, you can use $index–particularly if items are not being added, removed or rearranged in the middle of the array.

```
@for (option of options; track $index) {
  <option [value]="option.value">{{option.text}}</option>
}
```

Built in variables

These variables can be used within the @for block. When combined with a let, you can make variables to hold the information associated with the current iteration in the loop.

- $count: Number of items in a collection iterated over
- $index: Index of the current row
- $first: Whether the current row is the first row
- $last: Whether the current row is the last row
- $even: Whether the current row index is even
- $odd: Whether the current row index is odd

```
@for(r of recipes; track r.id; let ind = $index) {
  <div>
    <h1>{{ ind + 1 }}. {{ r.name }}</h1>
    <p>Cost: {{ r.cost | currency }}</p>
  </div>
}
```

```
fontColor:string = "blue";
```

Responding to events

Use the () syntax to respond to events such as the user clicking a button or filling out a form field.

- Can be added to any HTML tag
- Some common ones
    - (click)
    - (input)
    - (ngSubmit)
- Examples
-

```
<button (click)="myMethod()">Click me</button>
<!-- Expressions and variables can be used as method arguments. -->
<p (click)="myMethod('my value')">Sweet jam!</p>
<li (click)="myMethod(x)">{{x}}</li>
```

Angular Directives

Built-in classes that can adjust either the structure or the style of the website. Angular 17 is moving towards Statement Context and Class and Style Bindings so we won't use these, but you will still find them in a lot of Angular code.

*ngFor

- Used to loop through
- Add *ngFor=" " within any HTML tag (including other components)
- Within the " " add "let x of y"
    - x is the name of each element in the array as its being used
    - y is the array name
    - Optional: add ;index as i
        - This will keep track of the index as you go along

*ngIf

- Used to display/hide html based on a condition
- Add *ngIf=" " within any HTML tag (including other components)
- Within the " " add "BooleanVariable" or "x == y"
    - It can accept anything considered a condition
        - Boolean variable
        - Actual conditions
- If the condition supplied is true, then it will display. If false, it will hide

[ngClass]

- Used to add/remove a css rule based on a condition
- Add [ngClass]=" " within any HTML tag (including other components)
- Within the " " add "BooleanVariable ? 'cssTrue' : 'cssFalse' " or "x==y ? 'cssTrue' : 'cssFalse' "
    - It can accept anything considered a condition
        - Boolean variable
        - Actual conditions
    - ? 'cssTrue' : 'cssFalse'
        - If the condition is true, it will use whatever css rule you put inside of the first ' '
        - If the condition is false, it will use whatever css rule you put inside of the second ' '

    -

[ngStyle]

- Used to apply multiple css rules based on multiple conditions
- Add [ngStyle]=” ” within any HTML tag (including other components)
- Within the “ ” add the variable you will create below
  - In your typescript create a variable named stylesList (Or whatever you want to call it)
  - stylesList: object = { }
  - Within the { } you can add multiple styles
    - For each style, add the line below and adjust it as needed
    - 'font-style': this.changed ? 'italic' : 'normal'
      - ‘Css rule’ : boolean ? ‘value if true’ : ‘value if false’
      - Make sure to put a comma between each if using multiple

Additional Resources

- https://angular.dev/ - Official Angular website for documentation. Try working through the tutorial!
- Template syntax (Angular Docs)