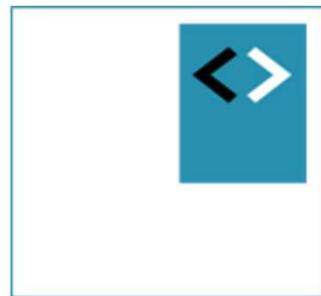


WARMTEBOUW.

Angular Advanced
New Template Syntax & @defer



Peter Kassenaar
info@kassenaar.com



Flow Control statements

New options for Template Syntax and Conditional Rendering

Control Flow in components: @if

- Previously: *ngIf="..."
- New: @if (condition) { ... }

Invalid

```
<div @if="showTitle">  
  {{ title }}  
</div>
```

Valid

```
@if (showTitle){  
  <div>  
    {{ title }}  
  </div>  
}
```

Alternative @else

- Conditional Control Flow: @if-@else
- Note: no nesting!

Invalid

```
@if (showTitle){  
  <div>  
    {{ title }}  
  </div>  
  @else{  
    <div>...</div>  
  }  
}
```

Valid

```
@if (showTitle) {  
  <div>  
    {{ title }}  
  </div>  
} @else {  
  <div>Please set a title...</div>  
}
```

Repeating items: @for

- Previously: *ngFor="let item of items"
- New: @for (item of items; track item.id) {...}

```
cities : City[] = [
  {id: 1, name: 'Amsterdam', country: 'NL'},
  {id: 2, name: 'Berlin', country: 'GER'},
  {id: 3, name: 'Tokyo', country: 'JAP'},
]
```

```
<ul>
  @for (city of cities; track city.id) {
    <li>{{ city.id }} - {{ city.name }}</li>
  }
</ul>
```

- 1 - Amsterdam
- 2 - Berlin
- 3 - Tokyo

Mandatory – tracking property

- Set a *unique tracking property*.
 - **Optional** in Angular 16 and lower (function track by)
 - **Mandatory** in Angular 17+ (=better rendering performance)
 - So, for instance @for (os of operatingSystems; **track os.id**)

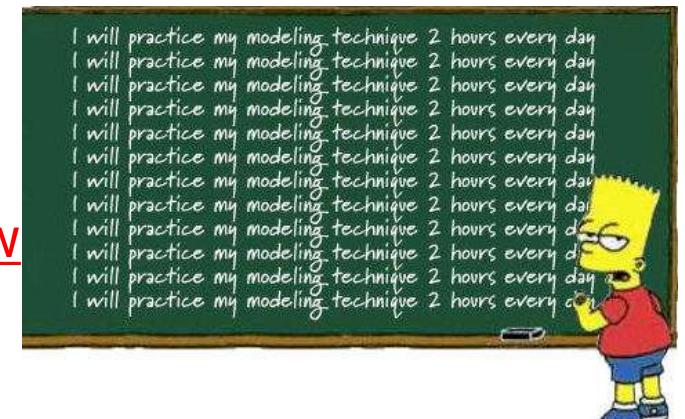
The screenshot shows a web-based Angular tutorial interface. At the top left, there's a navigation menu with 'Learn Angular' and a dropdown menu 'Control flow - @for'. To the right are navigation arrows (left, right, plus), a code editor area with a placeholder 'Type your code here', and a pencil icon for editing. Below the navigation is a large title 'Control Flow in Components - **@for**'. A explanatory text block follows: 'Often when building web applications, you need to repeat some code a specific number of times - for example, given an array of names, you may want to display each name in a `</>` tag'. At the bottom right is a red link: <https://angular.dev/tutorials/learn-angular/control-flow-for>.

Unchanged in Angular 17+

- Property binding
 - ``
- Event binding
 - `<button (click) = " clickHandler () " >`
- Input binding
 - `@Input () userID = "... "`
- Output binding
 - `@Output () increment = new EventEmitter<number>();`

Workshop

- Check in your own projects if there are any loops or conditionals that can be changed
- 1. Create a small Array of items/objects, loop over them in the template using the new `@for()` syntax.
 - Correctly use the `track` function, or use `$index`.
- 2. Check the Angular schematic for automatically refactoring your projects: `ng generate @angular/core:control-flow`
- 3. Read the official documentation on Control flow syntax at angular.dev/guide/templates/control-flow



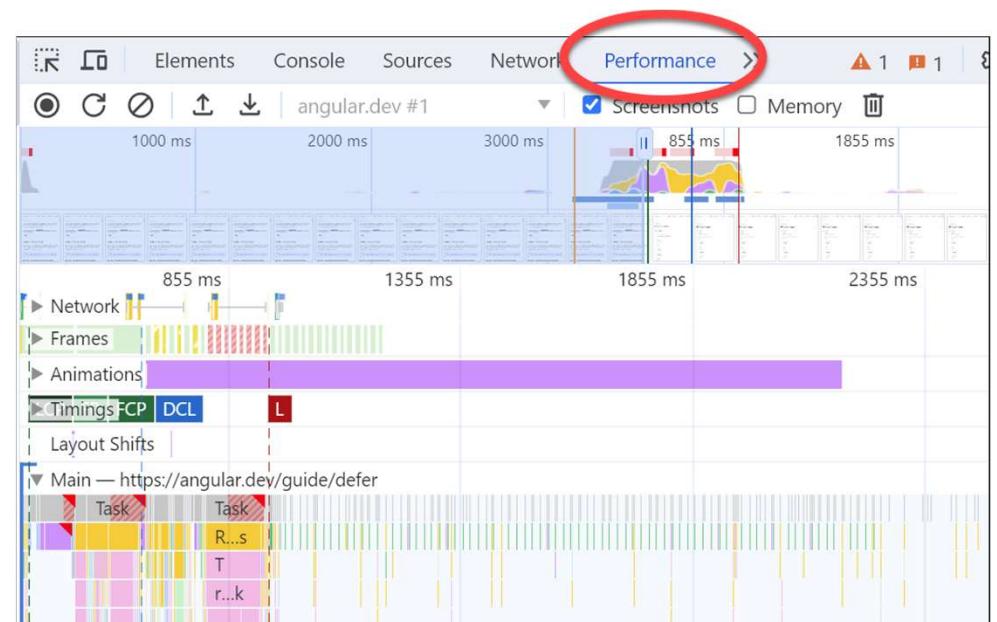


Deferrable Views

Loading content or components at a later time

Why deferrable views?

- Performance!
 - Reduce initial bundle size
 - Faster initial load
 - Improve Largest Contentful Pain (LCP)
 - Improve Time To First Byte (TTFB)
 - So in general – higher SEO-rankings
- Chrome DevTools for rendering details



Deferrable views

- Loading components, or resources **at a later time**
- Example:

```
@defer {  
  <app-comments />  
}
```

- By default: `@defer` will load `<app-comments>` when the browser is idle

Comment component `app-comments`

- Comment 1
- Comment 2
- Comment 3
- Comment 4

Adding a placeholder

- Use a `@placeholder {...}` block that contains html that will show *before* the deferred loading starts

```
@defer {  
  <app-comments/>  
} @placeholder{  
  <h2>Comments will appear here...</h2>  
}
```

Comments will appear here...

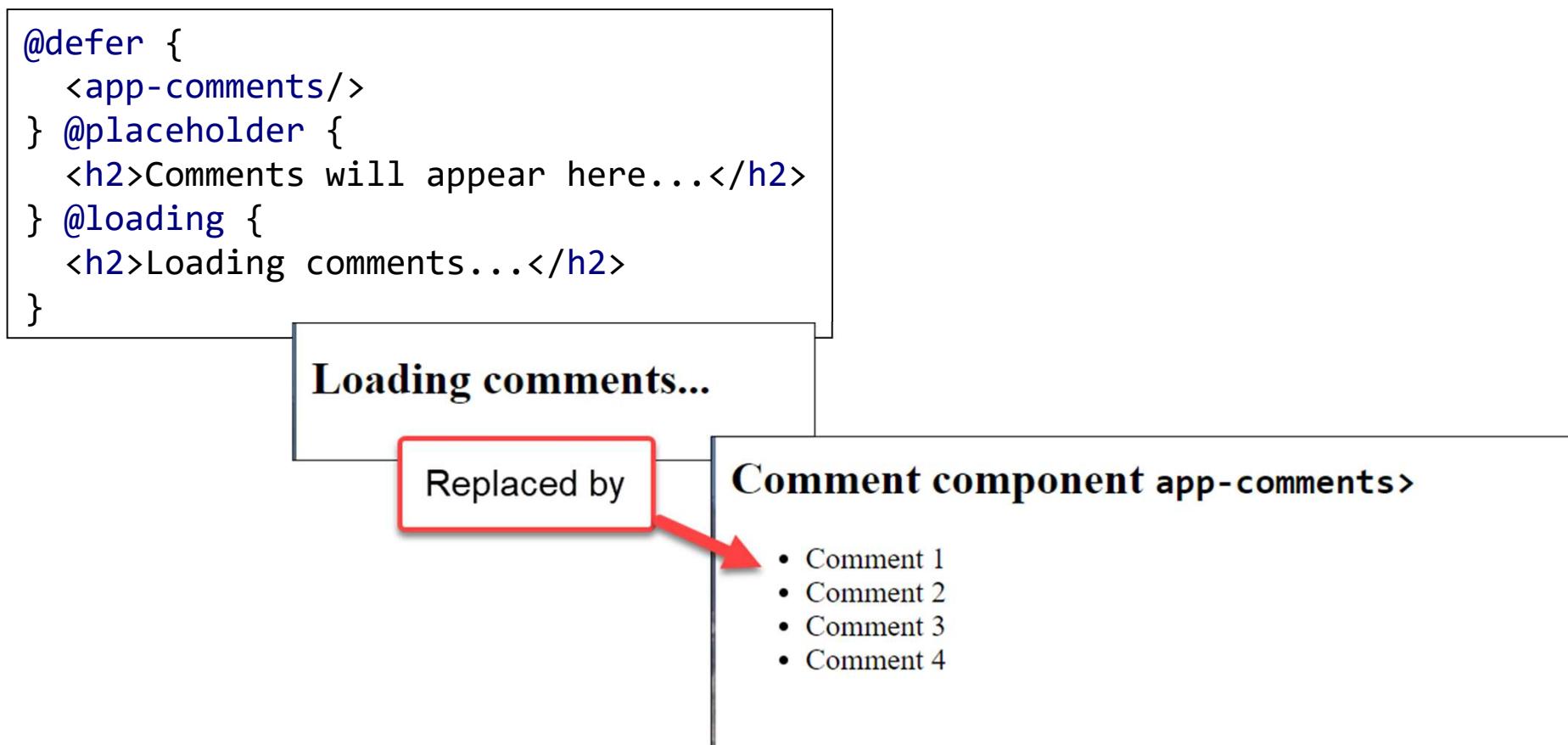
Replaced by

Comment component `app-comments`>

- Comment 1
- Comment 2
- Comment 3
- Comment 4

Adding a loading block

- Add `@loading { ... }` to `@defer`.
- The `@loading` block is shown *while* the deferred content is actively being loaded.



Parameters for blocks

- Minimum duration for @placeholder and @loading
 - Optional parameter – prevent flickering of screen
 - Example: @loading (minimum 2s)
 - “Make sure loading content is visible for at least 2 seconds”
- Trigger on viewport
 - Only load/show if you scroll / when content enters the viewport
 - @defer (on viewport)

```
<div>Lorem ipsum dolor sit amet, consectetur adipisicing  
expedita fugiat fugit, harum in inventore maiores  
</div>  
  
@defer (on viewport) {  
  <article>  
    <p>Angular is my favorite framework, and this is  
    defer loading content the easiest and most ergo-  
    with amazing contributors and experts that cre-  
    it really is the best community out there.</p>  
    <p>I can't express enough how much I enjoy work-  
    had. I love that the Angular team puts their o-  
    want Angular to be the best framework it can b-  
    comes from my heart and is not at all copied a-  
    few times.</p>  
    <p>Angular is my favorite framework, and this is  
    defer loading content the easiest and most ergo-  
    with amazing contributors and experts that cre-  
    it really is the best community out there.</p>
```

More triggers

- More triggers for your @defer block:

The screenshot shows a code editor with the following code snippet:

```
<div>
  @defer (on_) {
    <article> <?viewport>
      <p>A <?hover>
        de <?idle>
        wi <?immediate>
        it <?interaction>
      <p>] <?timer>
        ha
```

A code completion dropdown is open over the word `on_` in the `@defer` block. The dropdown lists several trigger types, with `viewport` highlighted in blue:

- viewport
- hover
- idle
- immediate
- interaction
- timer

Below the dropdown, a message says "Press Enter to insert, Tab to replace". To the right of the code editor, there is some text from a presentation slide.

Note!

- Parameter (on `viewport`) is only valid if the `@defer` block has a `@placeholder` block.
- Block `@placeholder` must have *exactly one root element*:

Invalid

```
@placeholder (minimum 5s) {  
  <h3>Text will appear here...</h3>  
    
}
```

Valid

```
@placeholder (minimum 5s) {  
  <div>  
    <h3>Text will appear here...</h3>  
      
  </div>  
}
```

Official Documentation on @defer blocks

The screenshot shows the Angular documentation interface. On the left, there's a sidebar with various links like 'Introduction', 'What is Angular?', 'Essentials', 'Start coding!', 'Components', 'Template Syntax', 'Directives', 'Dependency Injection', 'Signals', 'NgModule', 'Routing', 'Forms', 'Server-side Rendering', 'Build-time prerendering', 'Hydration', 'Deferrable views' (which is circled in red), 'Image Optimization', 'Testing', 'Internationalization', 'Security', 'Pipes', and 'Http Client'. The main content area is titled 'Deferrable Views' and contains a table of contents under 'On this page' with items such as 'Overview', 'Why use Deferrable Views?', 'Which dependencies are defer-loadable?', 'Blocks', '@defer', '@placeholder', '@loading', '@error', 'Triggers', 'on idle', 'on viewport', 'on interaction', 'on hover', 'on immediate', 'on timer', 'Prefetching', 'Testing', 'Behavior with Server-side rendering (SSR) and Static site generation (SSG)', 'Behavior with NgModule', and 'Nested @defer blocks and avoiding cascading loads'. Two red arrows point from the 'Deferrable views' link in the sidebar to the '@defer' and 'Triggers' entries in the table of contents.

<https://angular.dev/guide/defer>

@defer Mental model

@defer (when | on | prefetch ...)

```
<some-expensive-component />
```

@placeholder

Loading...

@loading

Spinner...

@error

Failed to load

Angular then does:

1. Create a **separate chunk** of code
2. Load the code when the trigger fires (on viewport, etc)
3. **Swaps** the placeholder with real content when ready

Workshop

- Create a small example using one of the triggers for loading deferred content
- See `../105-defer` as an example

