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) {…}`

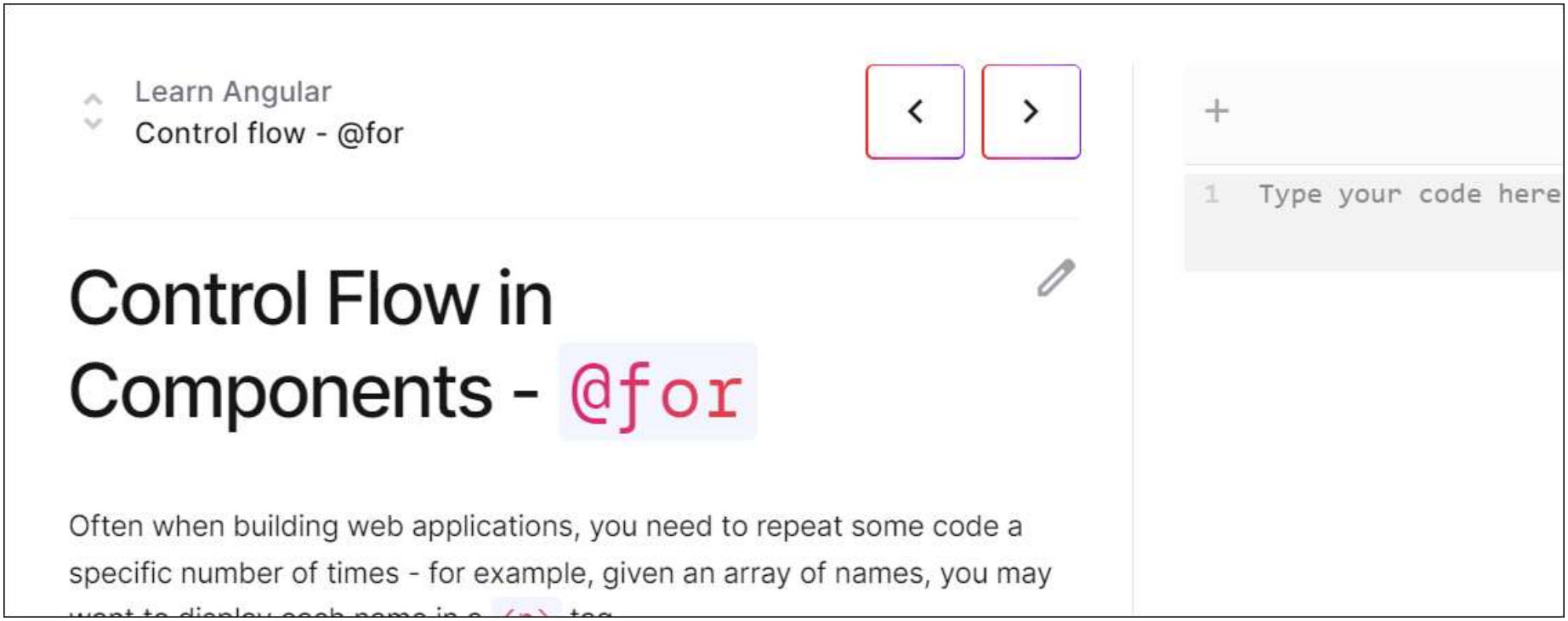
```
cities : City[] = [
  {id: 1, name: 'Amsterdam', country: 'NL'},
  {id: 2, name: 'Berlin', country: 'GER'},
  {id: 3, name: 'Tokyo', country: 'JAP'},
]
```

```
<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)`



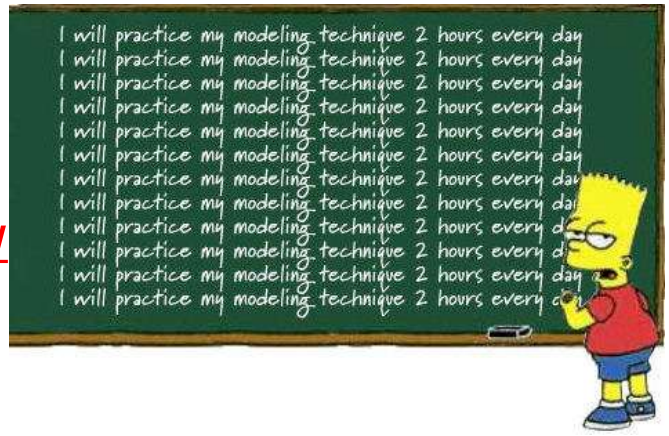
https://angular.dev/tutorials/learn-angular/control-flow-for

Unchanged in Angular 17+

- Property binding
  - `<img [src]="imgUrl" alt="… >`
- 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

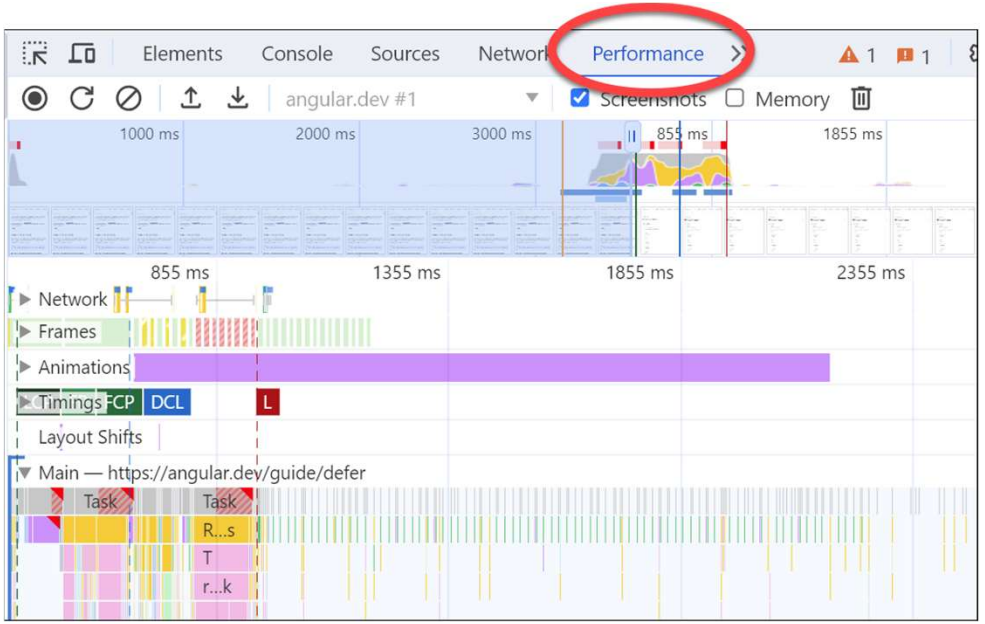


Deferrable Views

Loading content or components at a later time

Why deferrable views?

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

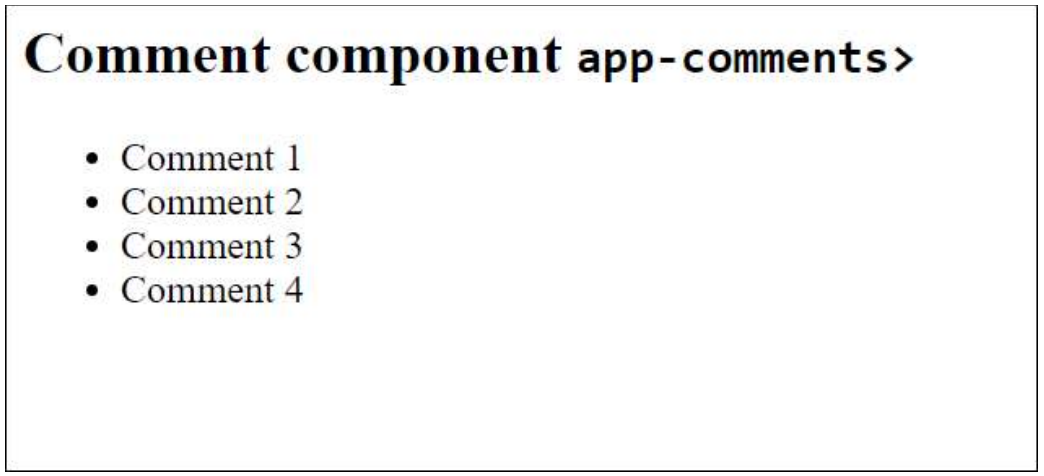


Deferrable views

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

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

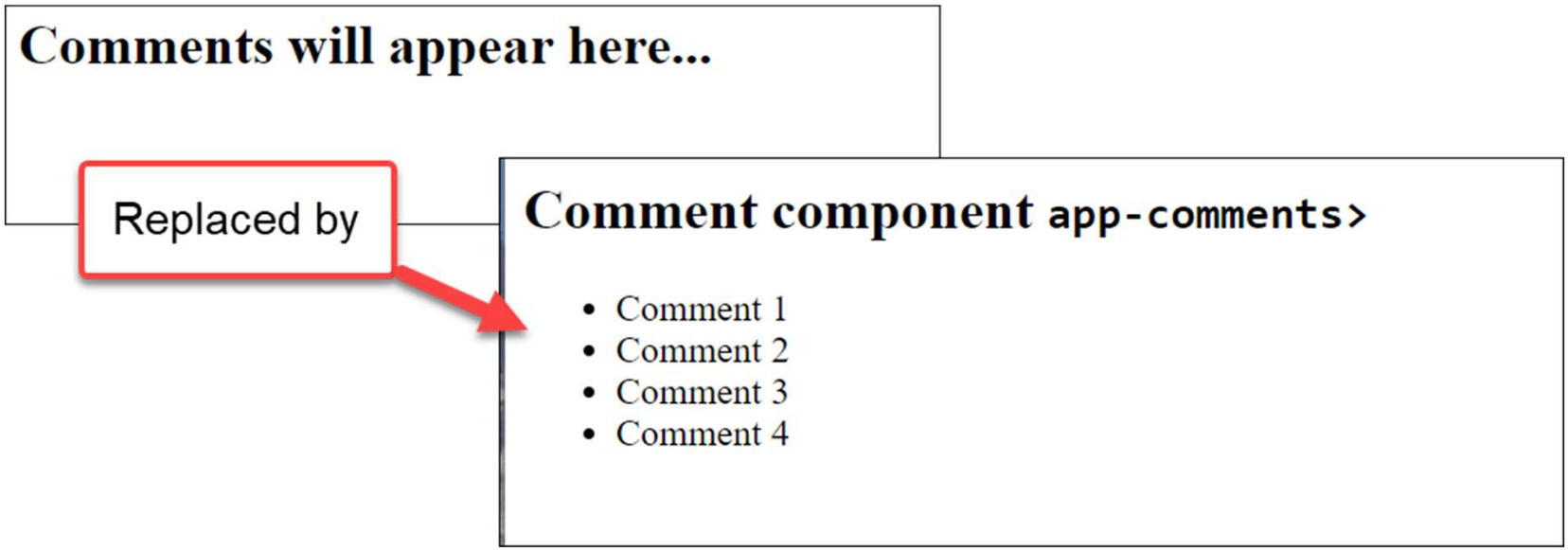
- By default: `@defer` will load `<app-comments>` when the browser is idle



Adding a placeholder

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

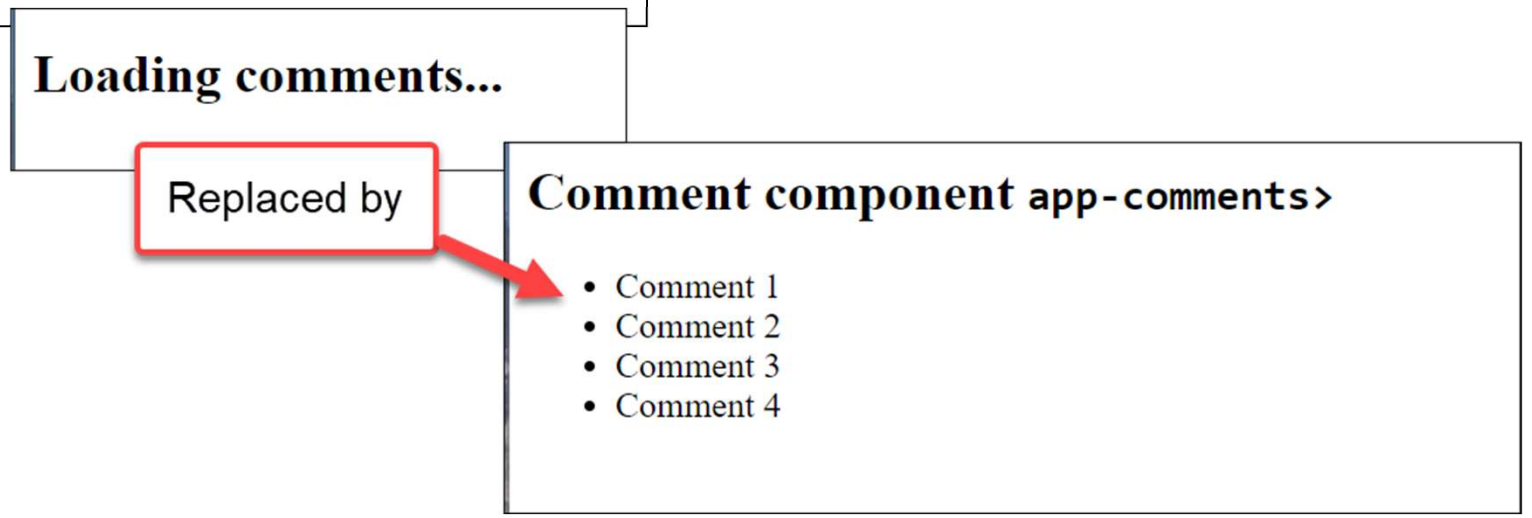
```
@defer {
  <app-comments/>
} @placeholder{
  <h2>Comments will appear here...</h2>
}
```



Adding a loading block

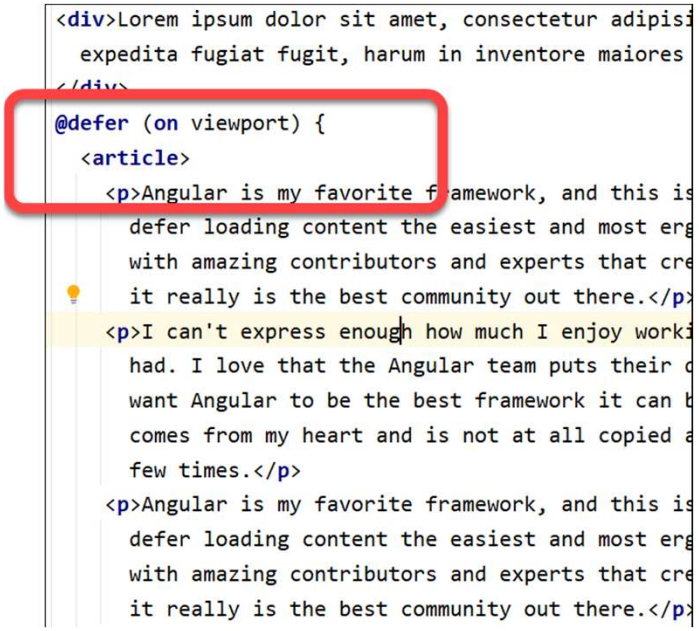
- Add `@loading { … }` to `@defer`.
- The `@loading` block is shown *while* the deferred content is actively being loaded.

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



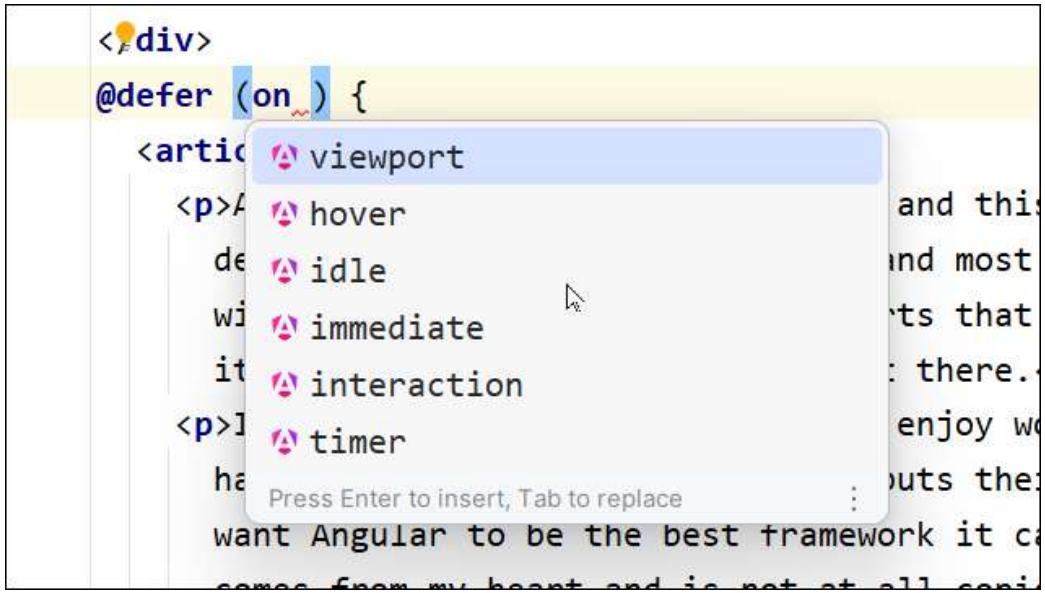
Parameters for blocks

- 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)`



More triggers

- More triggers for your @defer block:



Note!

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

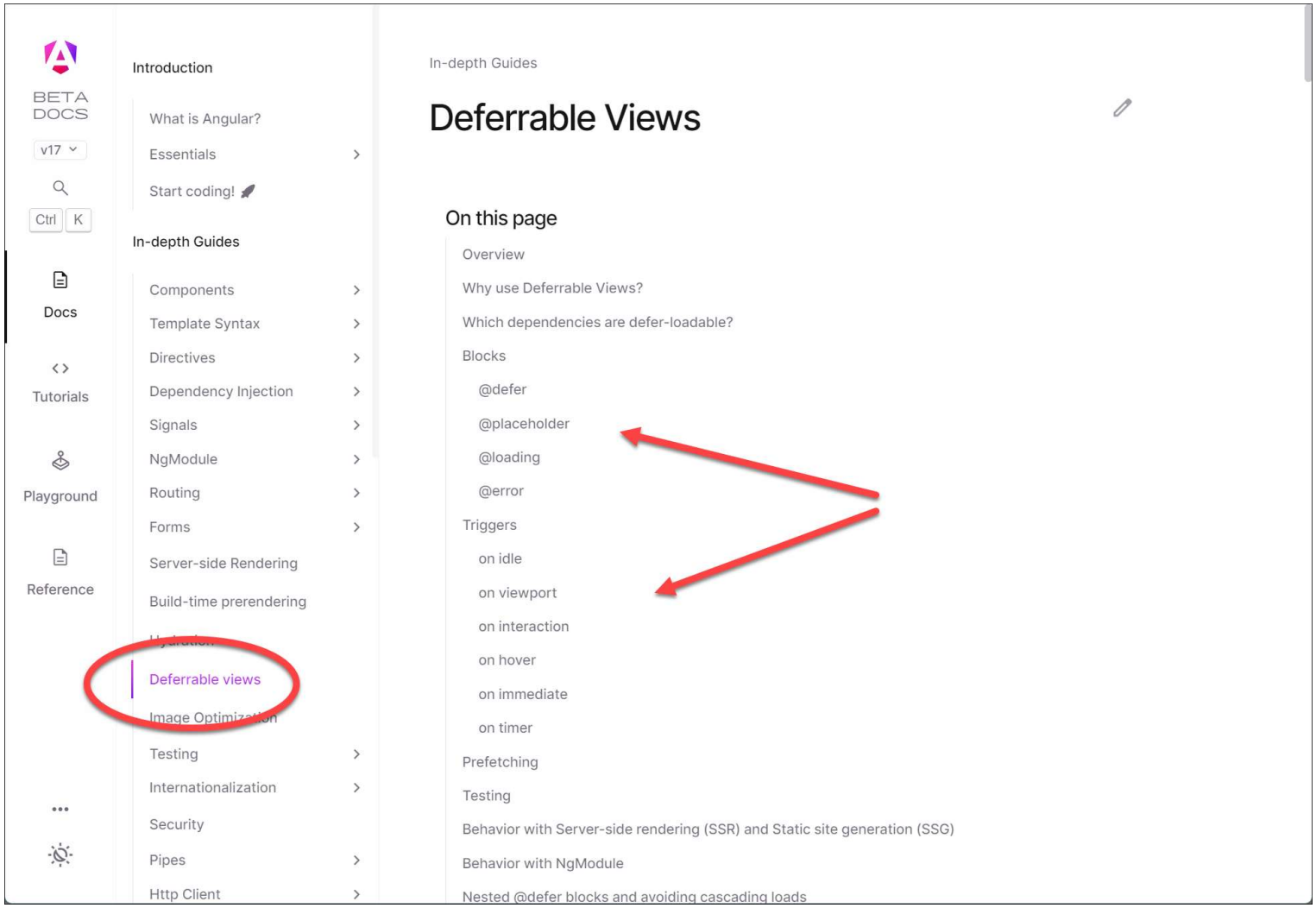
Invalid

```
@placeholder (minimum 5s) {
    <h3>Text will appear here...</h3>
    <img src="assets/loading.gif" alt="loading...">
}
```

Valid

```
@placeholder (minimum 5s) {
  <div>
    <h3>Text will appear here...</h3>
    <img src="assets/loading.gif" alt="loading...">
  </div>
}
```

Official Documentation on `@defer` blocks



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 deffered content
- See `../105-defer` as an example