

# Angular Performance Optimizations





## Performance

Values are estimated and may vary. The [performance score is calculated](#) directly from these metrics. [See calculator.](#)

▲ 0–49    ■ 50–89    ● 90–100

### METRICS

[Expand view](#)

- First Contentful Paint

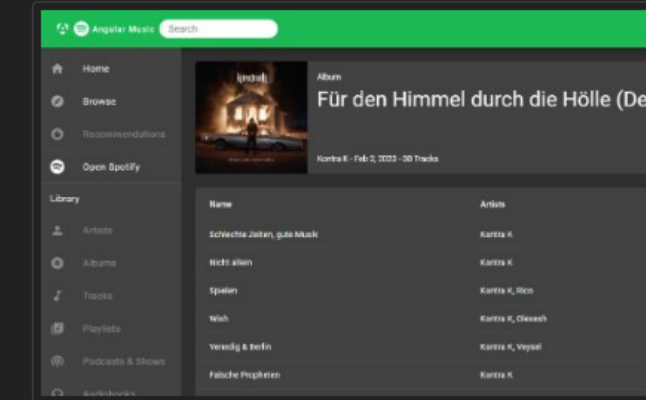
0.5 s

- Total Blocking Time

50 ms

- Speed Index

1.0 s



- Largest Contentful Paint

0.8 s

- Cumulative Layout Shift




0



# Angular Performance Optimizations

How to elevate the performance of an Angular App to the next level 🚀



 [github.com/sonallux](https://github.com/sonallux)  [@sonallux](https://twitter.com/sonallux)  [sonallux.github.io](https://sonallux.github.io)

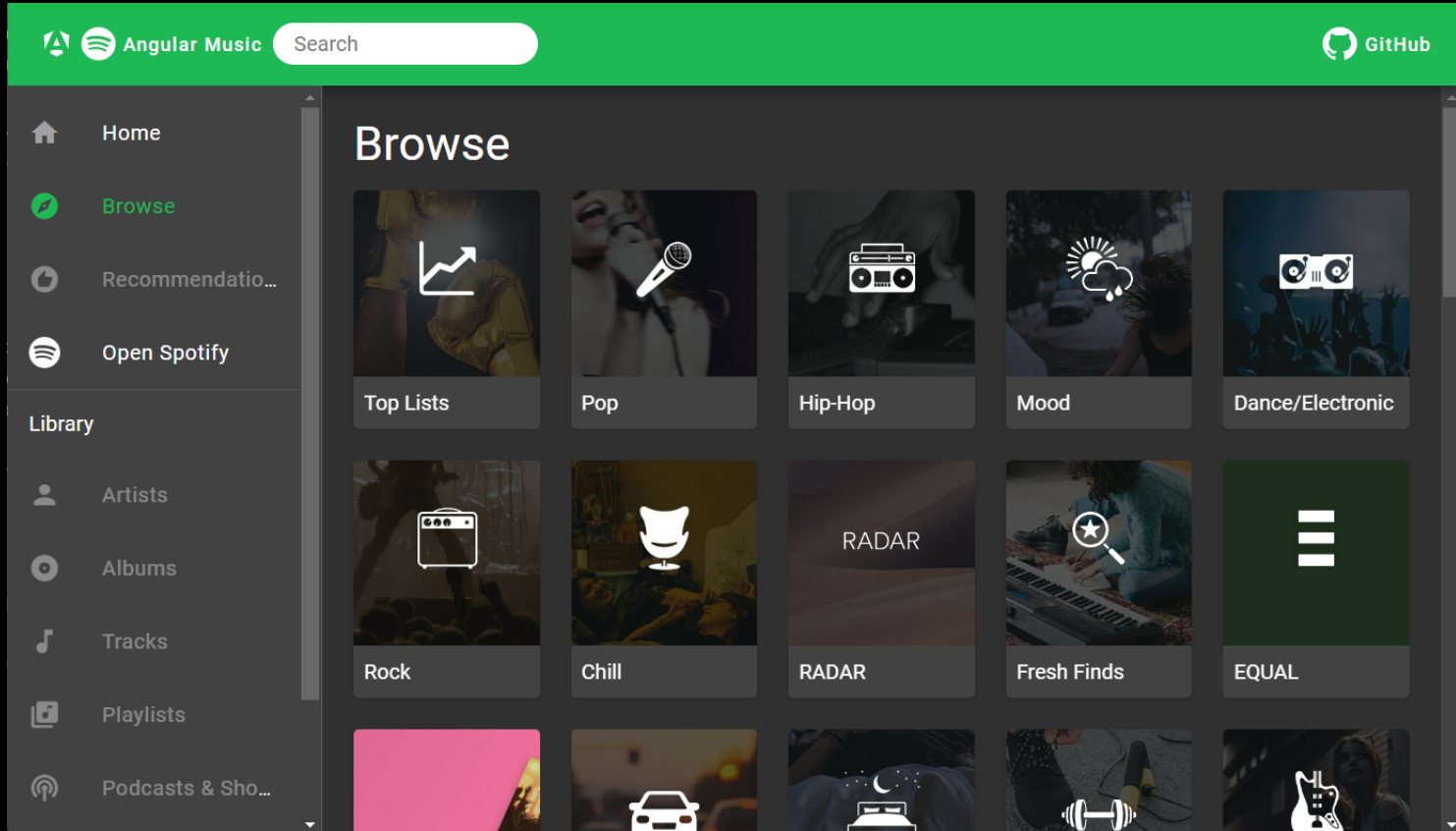


# Content

- Application
- Measurements
- Optimization steps
  - General
  - Standalone components
  - Lazy loading
  - Image loading
  - Built-in Control flow
  - esbuild
  - Server-side rendering
- Results



# Angular Music



# Measurements

- Lines of code
- Build duration
- Initial bundle size
- Lighthouse performance score
  - First Contentful Paint
  - Largest Contentful Paint
  - Total Blocking Time
  - Cumulative Layout Shift
  - Speed Index



# General

- Choose base components wisely ( `MatToolbar` , `MatCard` , `MatSidenav` )
- Do not use Components CSS files when using TailwindCSS
- Use CSS instead of JavaScript
- Reduce Backend request count and payload
- Cache JavaScript and CSS bundles



# Standalone components [1]

- Available since Angular 15

```
@Component({  
  selector: 'app-login',  
  templateUrl: './login.component.html',  
})  
export class LoginComponent {}  
  
@NgModule({  
  declarations: [LoginComponent],  
  imports: [CommonModule, MatButtonModule],  
  exports: [LoginComponent]  
})  
export class LoginModule {}
```

ts

```
@Component({  
  selector: 'app-login',  
  templateUrl: './login.component.html',  
  standalone: true,  
  imports: [NgIf, MatButtonModule],  
})  
export class LoginComponent {}
```

ts

1. <https://github.com/sonallux/angular-music/pull/106>





# Standalone components [1]

Stats	Relative change
Build time	+ 2%
Lines of code	- 2%
Initial bundle size	- 2%

1. <https://github.com/sonallux/angular-music/pull/106>



# Lazy loading <sup>[1]</sup>

- Lazy load routes
- Lazy load animations `provideAnimationsAsync()` ( $\geq$  Angular 17)
- Defer component loading with `@defer` (only works for standalone components) ( $\geq$  Angular 17)

```
@defer (on idle) {  
  <large-component />  
} @placeholder (minimum 500ms) {  
  <p>Placeholder content</p>  
}
```

html

1. <https://github.com/sonallux/angular-music/pull/107>



# Lazy loading <sup>[1]</sup>

Stats	Relative change
Build time	- 5%
Lines of Code	0%
Initial bundle size	- 33%

1. <https://github.com/sonallux/angular-music/pull/107>



# Image loading <sup>[1]</sup>

- Adjust image size to render size
- Add `preconnect` instructions
- Use `NgOptimizedImage` directive ( $\geq$  Angular 15)

Stats	Relative change
Build time	+ 2%
Lines of Code	+ 2%
Initial bundle size	+ 1%

1. <https://github.com/sonallux/angular-music/pull/108>



# Built-in control flow [1]

- Available since Angular 17 in developer preview
- Replaces the existing `NgIf`, `NgFor` and `NgSwitch` Directives

1. <https://github.com/sonallux/angular-music/pull/162>



# Built-in control flow - @if block

```
<h1 *ngIf="isLoggedIn; else loggedOut">html  
  Hello User!  
</h1>  
<ng-template #loggedOut>  
  Please log in!  
</ng-template>
```

```
@if (isLoggedIn) { html  
  <h1>Hello User!</h1>  
} @else {  
  <h1>Please log in!</h1>  
}
```



# Built-in control flow - @for block

```
<ul>  
  <li *ngFor="let item of items">{{ item.name }}</li>  
  <li *ngIf="items.length === 0">There are no items</li>  
</ul>
```

html

```
<ul>  
  @for (item of items; track item.name) {  
    <li>{{ item.name }}</li>  
  } @empty {  
    <li>There are no items</li>  
  }  
</ul>
```

html



# Built-in control flow - @switch block

```
<ng-container [ngSwitch]="orderStatus"> html
  <span *ngSwitchCase="'PLACED'">
    Order received, order processing started
  </span>
  <span *ngSwitchCase="'SHIPPED'">
    Order shipped
  </span>
  <span *ngSwitchCase="'DELIVERED'">
    Order delivered! Enjoy your purchase
  </span>
  <span *ngSwitchCase="'CANCELED'">
    Order canceled
  </span>
  <span *ngSwitchDefault>
    Invalid order status: {{orderStatus}}
  </span>
</ng-container>
```

```
@switch (orderStatus) { html
  @case ('PLACED') {
    <span>Order received, order processing started</span>
  }
  @case ('SHIPPED') {
    <span>Order shipped</span>
  }
  @case ('DELIVERED') {
    <span>Order delivered! Enjoy your purchase</span>
  }
  @case ('CANCELED') {
    <span>Order canceled</span>
  }
  @default {
    <span>Invalid order status: {{orderStatus}}</span>
  }
}
```





# Built-in control flow <sup>[1]</sup>

Stats	Relative change
Build time	- 2%
Lines of Code	+ 1%
Initial bundle size	+ 6%

1. <https://github.com/sonallux/angular-music/pull/162>



# esbuild <sup>[1]</sup>

- Switch bundler from webpack to esbuild
- Use `browser-esbuild` as drop-in replacement or `application` builder ( $\geq$  Angular 17)

Stats	Relative change
Build time	- 46%
Lines of Code	0%
Initial bundle size	+ 8%

1. <https://github.com/sonallux/angular-music/pull/109>

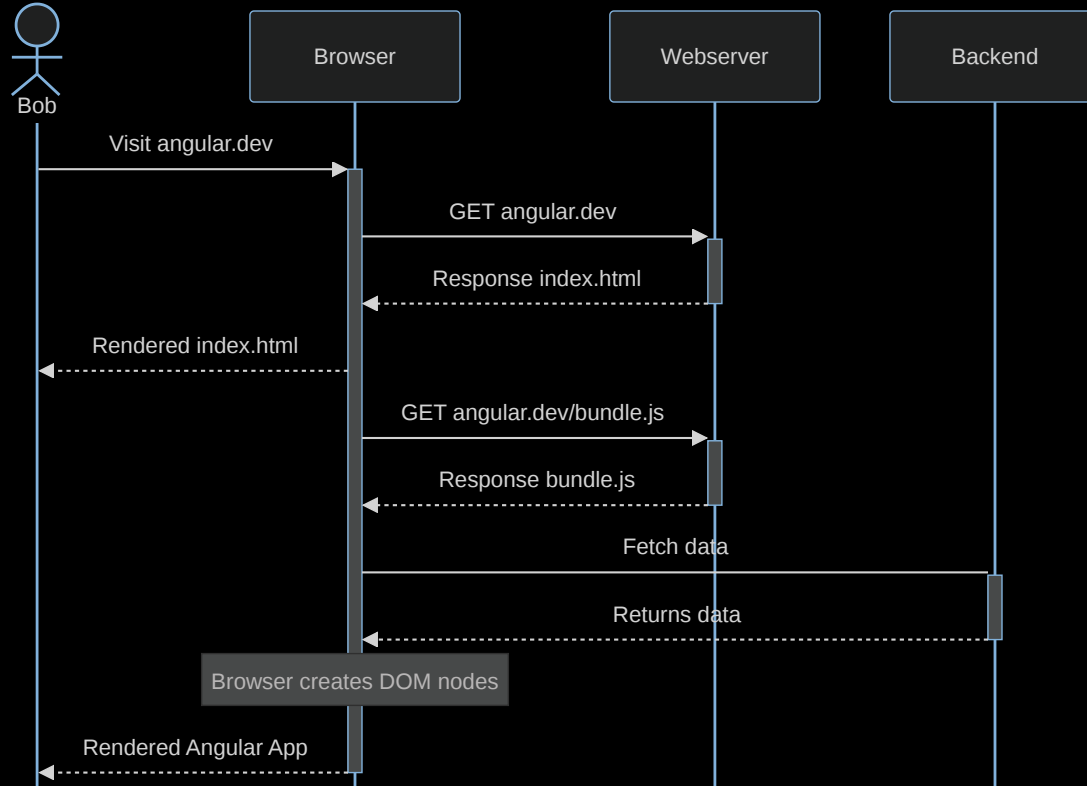


# Server-side rendering (SSR) [1]

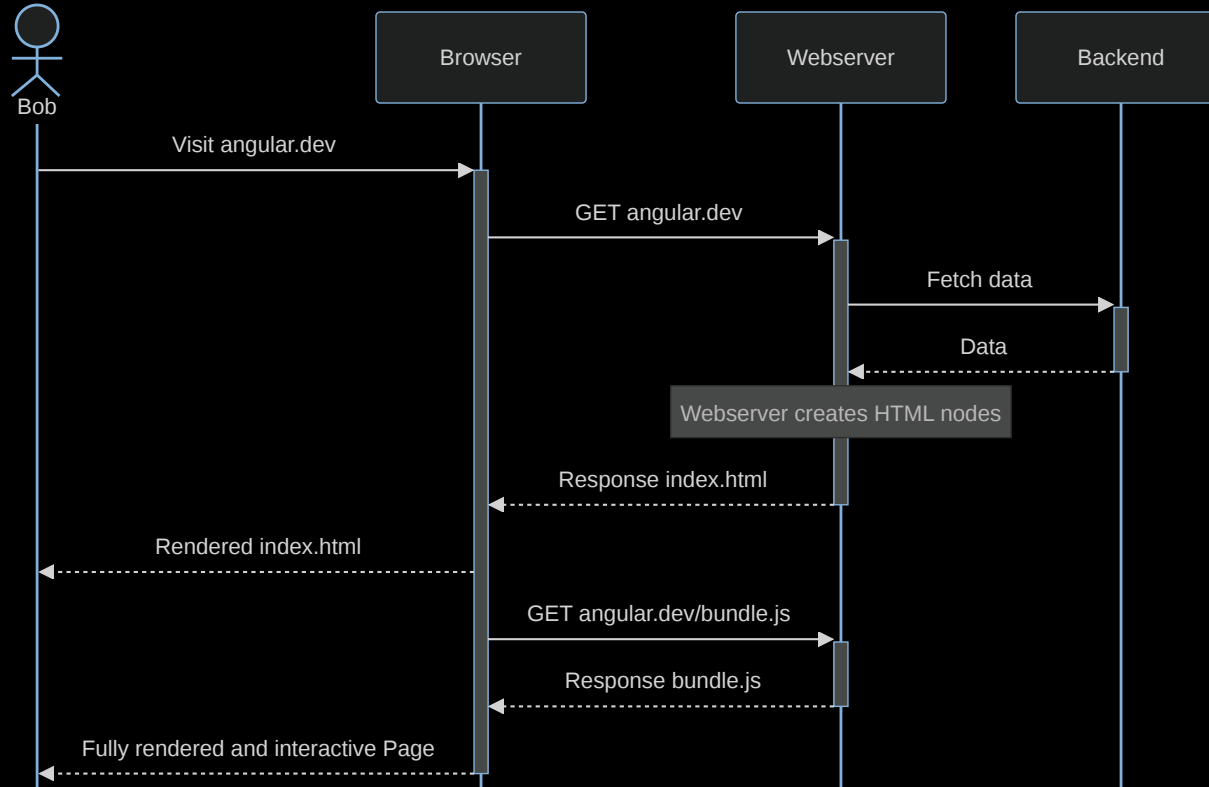
1. <https://github.com/sonallux/angular-music/pull/110>



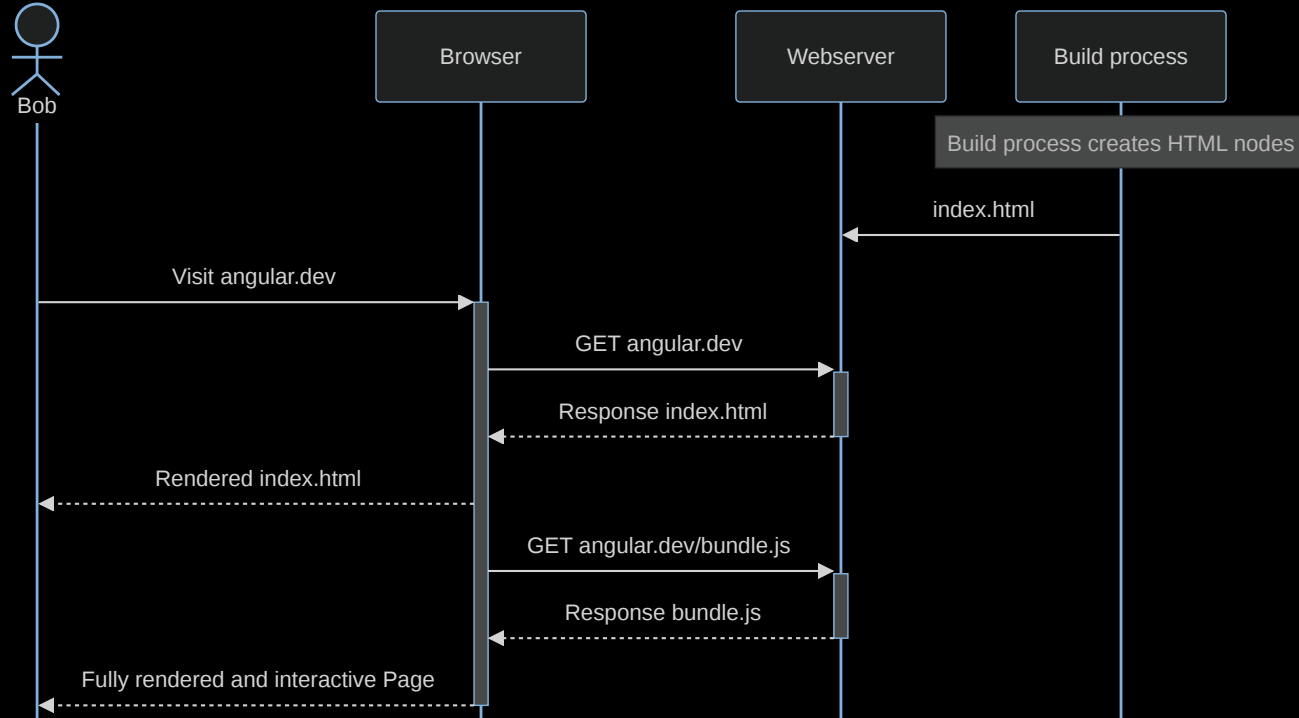
# Client-side rendering



# Server-side rendering (SSR)



# Static Site generation (SSG)



# Server-side rendering (SSR) <sup>[1]</sup>

Stats	Relative change
Build time	+ 55%
Lines of Code	+ 3%
Initial bundle size	+ 2%

1. <https://github.com/sonallux/angular-music/pull/110>



# Lighthouse Score

Page	Baseline	Standalone	Lazy loading	Image loading	Control flow	esbuild	SSR
Home	89	88	87	88	89	89	89
Browse	61	69	63	82	76	76	75
Category	87	81	88	87	87	87	79
Playlist	93	93	93	93	93	93	95
Album	93	93	93	93	93	93	98
Artist	90	91	93	91	92	91	97
Average	86	86	86	89	88	88	89















# Next steps


- OnPush change detection
- Angular signals
- Zoneless change detection



# Fragen ?

-  [Angular Music App](#)
-  [Core Web Vitals](#)
-  [Deferrable Views](#)
-  [NgOptimizedImage](#)
-  [Built-in control flow](#)
-  [Server-side rendering](#)
-  [Hydration](#)
-  [Prerendering \(SSG\)](#)
-  [Angular Movies App](#)

 [github.com/sonallux](https://github.com/sonallux)

 [@sonallux](#)

 [sonallux.github.io](https://sonallux.github.io)

