TechRetros

1

Profiling

Change Detection in Angular with DevTools for Performance Optimization

Optimizing performance is crucial for creating fast and responsive Angular applications.

TechRetros

2

Understanding Change Detection in Angular

Angular uses a mechanism called change detection to keep the UI in sync with the application's state.

Profiling Change Detection with Angular DevTools

- Install Angular DevTools
- Activate DevTools
- Open DevTools
- Start Profiling
- Interact with Your App
- Stop Profiling
- Analyze the Results

TechRetros

3

Identifying Performance Bottlenecks

When analyzing the results from Angular DevTools, keep an eye out for the following performance bottlenecks.

- **Frequent Change Detection**
- **Large Component Trees**
- **Inefficient Functions**
- **Optimizing Performance**
- **Change Detection Strategy**
- **Memoization**
- **Async Pipe**
- **Lazy Loading**
- **Angular Universal**

TechRetros

4

Example

Here's an example of how to use the OnPush change detection strategy in an Angular component.

STEP 4

```
import { Component, ChangeDetectionStrategy } from '@angular/core';

@Component({
  selector: 'app-my-component',
  templateUrl: './my-component.component.html',
  styleUrls: ['./my-component.component.css'],
  changeDetection: ChangeDetectionStrategy.OnPush // Use OnPush strategy
})
export class MyComponent {
  // Component logic here
}
```