**Performance Optimizations in Angular**

**\*Tree Shaking to Reduce Bundle Size**

In Angular, Tree Shaking is used to decrease the size of JavaScript’s bundle.

Tree shaking removes unused code from the application and shifts it into a smaller, more efficient bundle. The underlying build system performs it like Angular CLI during the building process. In addition, the build system analyzes the code and detects unused code, functions, or classes not called by the application.Thus, by removing unused code, the small size of JavaScript helps to load faster and enhance overall performance. However, if you are using Angular CLI, you can make it default.

Syntax : ng build --prod --optimization=true --build-optimizer=true

**\*Lazy loading modules**

Lazy loading is a design approach that holds off on initializing an item until it is actually required.

The primary purpose of lazy loading is to decrease the main bundle size of the Angular application by loading only what the user needs to see at first.

It will reduce the initial load time of the Angular app.

**\*Lazy loading Images**

Loading images only when they are in the viewport will help reduce the loading time of the application.It is straightforward to implement Native lazy loading.

<img src="testImage.png" loading="lazy"/>

The loading attribute supports auto, eager, and lazy. When the loading attribute is set to lazy, the browser will not load the resource until it is in the viewport. Unfortunately,

even though this is super easy to implement, it is not supported by the Safari browser.

**\*Use pure pipes instead of methods or functions**

When we use a method in our template to do some calculations, the change detection will be triggered to re-render the component more frequently.

This will affect the performance when there are many interactions in the template and when the processing is heavy.

Pure pipes is a good solution for this issue since it is called only when function parameters change.

In the following, you can see the difference. The pipe is called only once after the Add Customer button is clicked

### \*Proper Usage of ngZone

### A ngZone service is a powerful tool of Angular for managing the change detection process. However, too much use of ngZone affects your application’s performance. Thus, it is crucial to use it accordingly to restrict unintended consequences

### \*Minimizing the Number of HTTP Requests

### This will improve Angular performance significantly. Also, one of the best practices to minimize HTTP requests is to use caching. By caching responses from earlier requests, you can avoid redundant requests and reduce the number of HTTP requests. In addition, combining multiple HTTP requests into a single request can enhance performance optimization in Angular.

### \*Avoiding Unnecessary Code Duplication

### In Angular, code duplication leads to inefficient code and negatively impacts application performance. A code duplication increases your application’s file size and creates functionality issues. For Angular app performance optimization, you should avoid unnecessary code duplication and must write clean, efficient code that is maintainable and reusable

### 

### \*Prevent Memory Leakage

### A memory leak in an Angular application occurs when objects in the application's memory are no longer needed but are not released by the garbage collector because they are still being referenced.

### In the context of observables, if subscription is not unsubscribed when the component is destroyed or removed from the DOM, the observable and its associated resources will continue to exist in memory, even though they are no longer needed. Over time, this can cause a buildup of unused memory, which can slow down the application, reduce its performance, and even crash the application in extreme cases.

Additionally, as memory leaks are often difficult to diagnose, they can be frustrating for developers to troubleshoot and resolve. Therefore, Angular developers must proactively prevent memory leaks by adequately unsubscribing from observables when they are no longer needed.

The most recommended way of subscribing to an observable is to use the [**Async Pipe**](https://angular.io/api/common/AsyncPipe) or the rxjs **[takeUntil](https://www.learnrxjs.io/learn-rxjs/operators/filtering/takeuntil)** operator.