Zone less change detection

With Angular 18 the intermediary layer occupied by Zone.js can be removed. This was a process initiated in 17 with the introduction of Signals. However now instead of relying on Signal Components this new hybrid detection system will automatically trigger change detection by any invokers of markForCheck. Inside or outside of zone.js.

Leaving aside the 12k payload reduction if you go all in, there are plenty more benefits to be gained with zoneless change detection. The most immediate being performance with an immediate and probably large reduction in CD cycles and recalculation of values. There will also be:

* No more checking every binding on every mousemove event.
* No more runOutsideAngular() calls to reduce the amount of CD cycles
* No more juggling with detach() and detectChanges()
* Safer use of async/await with the absence of filled promises by Zone.js

You can completely remove Zone.js with two steps:

First by adding “provideExperimentalZonelessChangeDetection()” to your providers array:

bootstrapApplication(AppComponent, {  
 providers: [  
 // Toodles Zone.js by adding the below  
 provideExperimentalZonelessChangeDetection(),  
 ],  
});

Second — remove zone.js from your polyfills

{  
 "projects": {  
 "app": {  
 "architect": {  
 "build": {  
 "options": {  
 "polyfills": [  
 "zone.js" // <- This must go  
 ],  
 }  
 }  
 }  
 }  
 }  
}

You will also have to confirm that your templates are reactive / OnPush compatible. E.G. you may have to go from

@Component({  
 template: `  
 <div> {{ testString }} </div>  
 <child [value]="testValue"/>  
`  
})  
export class ExampleComponent {  
 name = "Test1";  
 value = "ABC";  
}

To

@Component({  
 template: `  
 <div> {{ testString() }} </div>  
 <child [value]="testValue | async"/>  
`  
})  
export class ExampleComponent {  
 name = signal("Test1");  
 value = new BehaviorSubject("ABC");  
}

If you have applications requiring custom change detection behaviours such as updates to their state outside the zone without triggering any change detection, customization options are available by using the schedulingMode option:

bootstrapApplication(App, {  
 providers: [  
 provideZoneChangeDetection({   
 // Outside zone state changes  
 schedulingMode: NgZoneSchedulingMode.NgZoneOnly   
 }),  
 ],  
});

You can also trigger signal updates even with code running outside of the zone.

class DemoComponent {  
 zone = inject(NgZone);  
 example = signal(0);  
  
 ngOnInit() {  
 this.zone.runOutsideAngular(() => {  
 setInterval(() => {  
 this.example.update(n => (n + 1 \* 42)  
 }, 5000);  
 })  
 }  
}

Angular Material was also refactored to not depend on Zone.js so the latest versions should be safe to use.

So, if your app is using ChangeDetectionStrategy.OnPush, Observables with async pipe or Signals to render data in templates 18 may be an easy migration with significant benefits.