Do you still think that NgZone (zone.js) is required for change detection in Angular?

A deep dive into NgZone implementation and its usage

Max Koretskyi aka Wizard Follow
Oct 5, 2017 · 7 min read

> **We're organizing the first Angular conference in Kiev, Ukraine as part of Angular In Depth community. Read more about it here.** I'll be happy to see you at the conference. We'll have experts from Angular team and the community and we made tickets affordable for most developers (around $150), even if you pay out of your own pocket.

Most articles I have seen strongly associate `Zone` (`zone.js`) and `NgZone` with change detection in Angular. And although they are definitely related, technically they are not part of one whole. Yes, `Zone` and `NgZone` is used to automatically trigger change detection as a result of async operations. But since change detection is a separate mechanism it can successfully work without `Zone` and `NgZone` . In the first chapter I will show how Angular can be used without `zone.js` . Second part of the article explains how Angular and zone.js interact together through `NgZone` . In the end I'll also show why automatic change detection sometimes doesn't work with 3rd party libraries like Google API Client Library (gapi).

I've written many in-depth articles on change detection in Angular and this one completes the picture. If you're looking for a comprehensive overview of how change detection works I recommend reading all of them starting with this one These 5 articles will make you an Angular Change Detection expert. **Please note that this article is not about Zones (zone.js), but about how Angular uses Zones in the implementation of NgZone and its relationship with change detection mechanism.** To learn more about Zones read I reverse-engineered Zones (zone.js) and here is what I've found

> I work as a developer advocate at **ag-Grid**. If you're curious to learn about data grids or looking for the ultimate Angular data grid solution, give it a try with the guide "**Get started with Angular grid in 5 minutes**". I'm happy to answer any questions you may have. **And follow me to stay tuned!**

• • •

Using Angular without Zone (zone.js)

To demonstrate that Angular can successfully work without Zone I was initially planning to provide a mock zone object that simply doesn't do anything. But the upcoming version 5 of Angular simplified things for me. It now provides a way to use a noop Zone that doesn't do anything through configuration.

To do that let's first remove dependency on `zone.js`. I'll use stackblitz to demo the application and since it uses Angular-CLI I will remove the following import from `polyfils.ts` file:

```
* Zone JS is required by Angular itself. */

import 'zone.js/dist/zone';  // Included with Angular CLI.
```

After that I'll configure Angular to use the noop Zone implementation like this:

```
platformBrowserDynamic()
    .bootstrapModule(AppModule, {
        ngZone: 'noop'
    });
```

If you now run the application you will see that change detection is fully operational and renders `name` component property in the DOM.

Now if we update this property using `setTimeout`:

```
export class AppComponent  {
    name = 'Angular 4';

    constructor() {
        setTimeout(() => {
            this.name = 'updated';
        }, 1000);
    }
```

You can see that the change is not updated. And it's expected since `NgZone` is not used and hence change detection is not triggered automatically. Yet it still works fine if we trigger it manually. This can be done by injecting ApplicationRef and triggering `tick` method to start change detection:

```
export class AppComponent  {
    name = 'Angular 4';

    constructor(app: ApplicationRef) {
        setTimeout(()=>{
            this.name = 'updated';
            app.tick();
        }, 1000);
    }
```

Now you can see that the update is successfully rendered.

To summarize, the point of the above demonstration is to show you that `zone.js` and `NgZone` in particular are not part of change detection implementation. It's a **very convenient** mechanism to trigger change detection automatically by calling `app.tick()` instead of doing it manually at certain points. We will see in a minute what those points.

. . .

How NgZone uses Zones

In my previous article on Zone (zone.js) I described in depth the inner working and API that Zone provides. There I explained the core concepts of forking a zone and running a task in a particular zone. I'll be referring to those concepts here.

I also demonstrated two capabilities that Zone provides—context propagation and outstanding asynchronous tasks tracking. Angular implements NgZone class that relies heavily on the tasks tracking mechanism.

NgZone is just a wrapper around a forked child zone:

```
function forkInnerZoneWithAngularBehavior(zone:
NgZonePrivate) {
    zone._inner = zone._inner.fork({
        name: 'angular',
        ...
```

The forked zone is kept in the `_inner` property and is usually referred to as Angular zone. This is the zone that is used to run a callback when you execute `NgZone.run()` :

```
run(fn, applyThis, applyArgs) {
    return this._inner.run(fn, applyThis, applyArgs);
}
```

The current zone at the moment of forking the Angular zone is kept in the `_outer` property and is used to run a callback when you execute `NgZone.runOutsideAngular()` :

```
runOutsideAngular(fn) {
    return this._outer.run(fn);
}
```

This method is often used to run performance heavy operations outside Angular zone to avoid constantly triggering change detection.

NgZone has `isStable` property that denotes whether there are no outstanding micro or macro tasks. It also defines four events:

```
+------------------+---------------------------------------
-------+
|      Event       |                     Description
|
+------------------+---------------------------------------
-------+
| onUnstable       | Notifies when code enters Angular Zone.
|
|                  | This gets fired first on VM Turn.
|
|                  |
|
| onMicrotaskEmpty | Notifies when there is no more
microtasks     |
|                  | enqueued in the current VM Turn.
|
|                  | This is a hint for Angular to do change
|
|                  | detection which may enqueue more
microtasks.  |
|                  | For this reason this event can fire
multiple  |
|                  | times per VM Turn.
|
|                  |
|
| onStable         | Notifies when the last
`onMicrotaskEmpty` has |
|                  | run and there are no more microtasks,
which   |
|                  | implies we are about to relinquish VM
turn.   |
|                  | This event gets called just once.
|
|                  |
|
| onError          | Notifies that an error has been
delivered.    |
+------------------+---------------------------------------
-------+
```

And Angular uses the `onMicrotaskEmpty` event inside ApplicationRef to automatically trigger change detection:

```
this._zone.onMicrotaskEmpty.subscribe(
    {next: () => { this._zone.run(() => { this.tick(); });
}});
```

If you remember from the previous section the `tick()` method is used to run change detection for the application.

. . .

How NgZone implements onMicrotaskEmpty event

Now let’s see how `NgZone` implements the `onMicrotaskEmpty` event. The event is emitted from the checkStable function:

```
function checkStable(zone: NgZonePrivate) {
  if (zone._nesting == 0 && !zone.hasPendingMicrotasks &&
!zone.isStable) {
    try {
      zone._nesting++;
      zone.onMicrotaskEmpty.emit(null); <-------------------
```

And this function is regularly triggered from three Zone hooks:

- onHasTask
- onInvokeTask
- onInvoke

As explained in the article about Zones when the last two hooks are triggered there’s a possibility of a change in the microtask queue so Angular has to run the `stable` check every time the hooks are triggered. The `onHasTask` hook is also used to perform a check since it tracks entire queue changes.

. . .

Common pitfalls

One of the most frequent questions on stackoveflow related to change detection is why sometimes when using 3rd party libraries the changes in a component are not applied. Here is an example of such question involving Google API Client Library (gapi). The common solution to such problems is to simply run a callback inside Angular zone like this:

```
gapi.load('auth2', () => {
    zone.run(() => {
        ...
```

However, an interesting question is *why* Zone **doesn't** register the request which leads to **not** having a notification in one of the hooks? And without notification `NgZone` doesn't automatically trigger change detection.

To learn that I just dig into `gapi` **minified** sources and found that it uses JSONP to make a network request. This approach doesn't use common AJAX APIs like XMLHttpRequest or Fetch API which are patched and tracked by Zones. Instead, it creates a `script` tag with a source URL and defines a global callback to be triggered when the requested script with data is fetched from the server. This can't be patched or detected by Zones and hence the frameworks remains oblivious to requests performed using this technique.

Here is the relevant snipped from gapi minimized version for curious:

```
Ja = function(a) {
    var b = L.createElement(Z);
    b.setAttribute(“src”, a);
    a = Ia();
    null !== a && b.setAttribute(“nonce”, a);
    b.async = “true”;
    (a = L.getElementsByTagName(Z)[0]) ?
        a.parentNode.insertBefore(b, a) :
        (L.head || L.body ||
L.documentElement).appendChild(b)
}
```

The `Z` variable is equal to `"script"` and the parameter `a` holds the request URL:

```
https://apis.google.com/_.../cb=gapi.loaded_0
```

The last segment of the URL is `gapi.loaded_0` global callback:

```
typeof gapi.loaded_0
“function”
```

. . .

Thanks for reading! If you liked this article, hit that clap button below 👏. It means a lot to me and it helps other people see the story.

For more insights follow me on Twitter and on Medium.