Zones

Zone.js provides a mechanism, called zones, for encapsulating and intercepting asynchronous activities in the browser (e.g. `setTimeout` , , promises).

These zones are *execution contexts* that allow Angular to track the start and completion of asynchronous activities and perform tasks as required (e.g. change detection). Zone.js provides a global zone that can be forked and extended to further encapsulate/isolate asynchronous behaviour, which Angular does so in its **NgZone** service, by creating a fork and extending it with its own behaviours.

The **NgZone** service provides us with a number of Observables and methods for determining the state of Angular's zone and to execute code in different ways inside and outside Angular's zone.

It is important to know that Zone.js accomplishes these various interceptions by Monkey Patching common methods and elements in the browser, e.g. `setTimeout` and `HTMLElement.prototype.onclick` . These interceptions can cause unexpected behaviour between external libraries and Angular. In some cases, it may be preferential to execute third party methods outside of Angular's zone (see below).

In The Zone

NgZone exposes a set of Observables that allow us to determine the current status, or *stability*, of Angular's zone.

- *onUnstable* – Notifies when code has entered and is executing within the Angular zone.
- *onMicrotaskEmpty* - Notifies when no more microtasks are queued for execution. *Angular subscribes to this internally to signal that it should run change detection.*
- *onStable* – Notifies when the last `onMicroTaskEmpty` has run, implying that all tasks have completed and change detection has occurred.
- *onError* – Notifies when an error has occurred. *Angular subscribes to this internally to send uncaught errors to its own error handler, i.e. the errors you see in your console prefixed with 'EXCEPTION:'.*

To subscribe to these we inject **NgZone** into our components/services/etc. and subscribe to the public Observables.

```
import { Injectable, NgZone } from '@angular/core';

@Injectable()
export class OurZoneWatchingService() {
  constructor(private ngZone: NgZone) {
    this.ngZone.onStable.subscribe(this.onZoneStable);
    this.ngZone.onUnstable.subscribe(this.onZoneUnstable);
    this.ngZone.onError.subscribe(this.onZoneError);
  }

  onZoneStable() {
    console.log('We are stable');
  }

  onZoneUnstable() {
    console.log('We are unstable');
  }

  onZoneError(error) {
    console.error('Error', error instanceof Error ? error.message : error.toString());
  }
}
```

Subscribing to these can help you determine if your code is unexpectedly triggering change detection as a result of operations that do not affect application state.

Change Detection

Since all asynchronous code executed from within Angular's zone can trigger change detection you may prefer to execute some code outside of Angular's zone when change detection is not required.

To run code outside of Angular's context, **NgZone** provides a method aptly named **runOutsideAngular**. Using this method, Angular's zone will not interact with your code and will not receive events when the global zone becomes stable.

In this example you will see in the log what happens with Angular's zone when code is run in and outside of it.

You will notice that in both cases clicking the button causes the Angular zone to become unstable due to Zone.js patching and watching **HTMLElement.prototype.onclick**, however the **setInterval** executing outside of Angular's zone does not affect its stability and does not trigger change detection.

Debugging

Generally, exceptions thrown during a chain of asynchronous events will only include the current method in their stack trace.

With Zone.js tracking all of our asynchronous calls it can provide us a longer, more detailed, stack trace of the events and calls that occurred leading up to our exception.

To enable long stack traces in development, you should include the **long-stack-trace-zone** module in your code. It is a good idea not to include this in your production build but Angular will skip setting up longer stack traces when in production mode (`enableProdMode` from `@angular/core`).

Angular will take care of forking and extending its own zone to display more meaningful stack traces.

```
if (__PRODUCTION__) {
  enableProdMode();
} else {
  require('zone.js/dist/long-stack-trace-zone');
}
```

With the following code, we start by calling `startAsync` which triggers a chain of setTimeouts leading up to an uncaught error.

```
function startAsync() {
  setTimeout(stepOne, 100);
}

function stepOne() {
  setTimeout(stepTwo, 100);
}

function stepTwo() {
  throw new Error('Finished');
}
```

Simple Stack trace

This is a typical stack trace that you would see in this scenario, without Zone, showing only the function where the unhandled exception occurred.

```
Uncaught Error: Finished(…)
  stepTwo @ debugging.html:28
```

Detailed "Long" Stack trace

In the stack trace below, you can see the order of events that occurred within this asynchronous chain of function calls, '>>' has been added to point out our functions.

You'll notice this stack trace includes much more information, including Zone's own task management (e.g. `onScheduleTask`), as well as the time that elapsed between when the function was queued and when it was executed.

Having this longer stack trace may aide you with debugging which feature of Angular your code is interacting with asynchronously and help you narrow down where your problem is occuring.

```
debugging.html:16 Error: Finished
>>  at stepTwo (http://localhost:3030/examples/debugging.html:28:15)
    at ZoneDelegate.invokeTask (http://localhost:3030/node_modules/zone.js/dist/zone.j
s:265:35)
    at Zone.runTask (http://localhost:3030/node_modules/zone.js/dist/zone.js:154:47)
    at ZoneTask.invoke (http://localhost:3030/node_modules/zone.js/dist/zone.js:335:33
)
    at data.args.(anonymous function) (http://localhost:3030/node_modules/zone.js/dist
/zone.js:970:25)
  -------------   Elapsed: 101 ms; At: Wed Nov 16 2016 08:23:17 GMT-0500 (EST)   -----
--------
    at Object.onScheduleTask (http://localhost:3030/node_modules/zone.js/dist/long-sta
ck-trace-zone.js:83:18)
    at ZoneDelegate.scheduleTask (http://localhost:3030/node_modules/zone.js/dist/zone
.js:242:49)
    at Zone.scheduleMacroTask (http://localhost:3030/node_modules/zone.js/dist/zone.js
:171:39)
    at http://localhost:3030/node_modules/zone.js/dist/zone.js:991:33
    at setTimeout (eval at createNamedFn (http://localhost:3030/node_modules/zone.js/d
ist/zone.js:927:17), <anonymous>:3:37)
>>  at stepOne (http://localhost:3030/examples/debugging.html:23:9)
    at ZoneDelegate.invokeTask (http://localhost:3030/node_modules/zone.js/dist/zone.j
s:265:35)
    at Zone.runTask (http://localhost:3030/node_modules/zone.js/dist/zone.js:154:47)
  -------------   Elapsed: 105 ms; At: Wed Nov 16 2016 08:23:17 GMT-0500 (EST)   -----
--------
    at Object.onScheduleTask (http://localhost:3030/node_modules/zone.js/dist/long-sta
ck-trace-zone.js:83:18)
    at ZoneDelegate.scheduleTask (http://localhost:3030/node_modules/zone.js/dist/zone
.js:242:49)
    at Zone.scheduleMacroTask (http://localhost:3030/node_modules/zone.js/dist/zone.js
:171:39)
    at http://localhost:3030/node_modules/zone.js/dist/zone.js:991:33
    at setTimeout (eval at createNamedFn (http://localhost:3030/node_modules/zone.js/d
ist/zone.js:927:17), <anonymous>:3:37)
>>  at startAsync (http://localhost:3030/examples/debugging.html:33:9)
    at ZoneDelegate.invoke (http://localhost:3030/node_modules/zone.js/dist/zone.js:23
2:26)
    at Zone.run (http://localhost:3030/node_modules/zone.js/dist/zone.js:114:43)
```