Understanding ViewChild, ViewChildren & Querylist in Angular

The ViewChild or ViewChildren decorators are used to Query and get the reference of the DOM element in the Component. ViewChild returns the first matching element and ViewChildren returns all the matching elements as a QueryList of items. We can use these references to manipulate element properties in the component.

Change Detection strategy:

Whenever we talk about performance in Angular based application, Change Detection Strategy plays an important role.

## What is change detection?

Updating the DOM whenever user Data is changed is known as the Change Detection technique. There are two Strategies available for Change Detection in Angular.

# If the template or model file is changed, it's necessary to update the view. This process of syncing the template with the data is called "Change Detection”

## How change detection works?

Every Process has its work cycle, for Change Detection it's mainly divided into two parts: The angular side and developer side.

1.Developer side: Developer's updates application model.

**Eg:Event bindings,** which can be added using the () syntax, can be used to capture a browser event or component output to execute some function on a component or a directive. So they often trigger the first phase.

2.Angular side: Sync the updated state model and re-render the view.

**Eg:Property bindings,** which can be added using the [] syntax, should be used only for reflecting the state of the model in the view.

## Let’s take a brief look at this process

Step 1: When Developer updates the template page or Data models like changing something in data binding or something.

Step 2: Changes are made in the state of your application, now Angular will detect thechanges.

Step 3: Change detection uses the Tree method to re-render yourapplication/component,ChangeDetection will check every component of your application in tree form from top to bottom to check if any corresponding modelhas changed or not.

Step 4:If there is any change, it will update the component view and re-render it.

It will be easy to understand it with some graphical representation:

Default-strategy1:

You can see Angular components and it Changes Detector (CD) for every component which was created during your application bootstrap process in this image. Those change detectors compare the current value with the previouss one. If ChangeDetector detects the value is changed, it will set Property is Changed==true.

## Zone.js

Normally, any asynchronous task is intercepted and tracked by zone.The zone has three phases:

1. Phase 1: when the zone starts, it is in a stable situation.
2. Phase 2: It will become unstable if any task is run in the Zone.
3. Phase 3: After the Task is completed it will return to a Stable situation.

Change Detection will be triggered by the angular framework if one of these events is fired:

1. Any browser event like a click, keyup, keydown, OnChange, etc.
2. setInterval() or setTimeout()
3. Any HTTP request using XMLHttpRequest

Angular uses its zone called ngZone.There is only one ngZone and in this zone Change detection will be called, only for async operations.

## Change Detection strategies

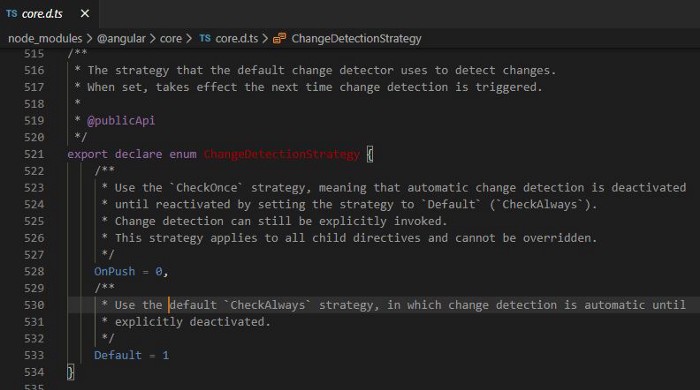
#### **Default ChangeDetection strategy**

Angular uses ChangeDetectionStrategy.Defaultas a default change detection strategy. Every time an event is fired like user event, XHR, timer, etc. The default strategy will check every component in the component tree from top to bottom. It will check every component without making any assumption on component's dependencies which may create some issue that's why it's called dirty checking. It can badly influence your application's performance if you have a large application that contains many components.

#### **OnPush()**

As we saw in default Change Detection strategy, if you have a large application default strategy will check all component, we will affect your performance.so to overcome that problem you can use OnPush()ChangeDetection strategy.

ChangeDetectionStrategy.Default (**CheckAlways**) and ChangeDetectionStrategy.OnPush( **CheckOnce**).



If we look at angular documentation or comments for each enum as in the angular code below, it says when the strategy value is 1 (Default), the change detection is automatic but when the value is 0(OnPush) then **automatic detection is deactivated but can be invoked explicitly**.

Unlike in Default mode, the change detection in OnPush mode doesn’t automatically update all of its child components change unless otherwise marked for dirty or change detection triggered explicitly.

## **Invoking Change Detection Explicitly:**

[ChangeDectectorRef](https://angular.io/api/core/ChangeDetectorRef#changedetectorref) class allows us to mark a specific view dirty so that the view can be re-rendered in the next change-detection cycle.

This can be done in 2 lifecycle hooks 1)ngOnChanges 2)ngDoCheck

We can mark a view as dirty using the [ChangeDetectionRef](https://angular.io/api/core/ChangeDetectorRef" \l "changedetectorref) abstract class methods. It exposes the following few methods:

1. **markForCheck**: It can be used to mark a component dirty i.e., in need of re-rendering. The view is updated only when the next change detection runs.

2. **detach**: To detach from change detection even if they are marked dirty until they are attached back

3. **detectChanges**: Runs the change detection for the corresponding component and all its children.

4. **checkNoChanges**: Can be used to check if there any changes associated with the component and throw an exception if any changes were identified

5. **reattach**: Attach the view back to the change detector tree.