## Tips & Tricks

Use Track By For Quickly Displaying List Items.

You can control the frequency of rendering lists by using the Track By feature. Otherwise, displaying the list will require the DOM to reload for every item in the list.

See the below example.

<div ng-repeat="listitem in empList">

</div>

The above approach will make your code less efficient. Hence, you should be using the following code.

<div ng-repeat=" listitem in empList track by item.empid">

</div>

The code above still follows the same process. But it would display the list faster and make the application more efficient. As it is using the unique key instead of using a random one.

### Avoid Filters With Large Arrays.

Filters run on each digest loop and create a new array every time they execute. In such cases, prefer to watch your list and do filtering upon detecting a change.

See this example of a bad use of filters in a digest loop.

<li ng-repeat="item in heroList | myHeroSortFilter"></li>

Whereas, now see a good example.

#### **In Controller.**

$scope.filteredItems = $scope.items;

$scope.$watchCollection('items', function () {

// Apply filter on change detection

$scope.filteredItems = myHeroFilter($scope.items);

});

#### **In HTML.**

<li ng-repeat="item in heroList"></li>

### How To Detect If An Item Changes In A List.

There could be two ways to achieve it. First of all, let’s see the bad example. The downside is, it’ll create copies of the object with each comparison.

$scope.watch('empList',

function watchListener() {

// Print a message if the reporting of any employee changes.

}

}, true);

Now, let’s see the approach that you should use.

$scope.watchCollection('empList',

function watchListener() {

// Print a message if the reporting of any employee changes.

}

});

### Slow Down The Watchers From Being Called.

For delaying the execution of watchers, we can use the “debounce” property.

<input type="text" name="empName"

ng-model="emp.name"

ng-model-options="{ debounce: 100 }" /><br />

The above code will change model value only after waiting for 100 ms after the last change happened in input value.

### Select An Element By ID Or Class.

For selecting the element using ID, use the following code.

angular.element(document.querySelector('#id'))

For selecting the element based on class name, use this code.

angular.element(elem.querySelector('.classname'))

#### Make a particular HTML section compile only after some asynchronous data is fetched:

In Controller:

$http.get('data\_endpoint').then(function () {

$scope.isDataFetched = true;

})

In HTML:

<div ng-if="isDataFetched">

<!-- Some complex HTML that needs fetched data -->

</div>

#### Angular copy function copies the prototype properties of an object on the new object. BEWARE: If you watch such an object, you will probably get stuck in an infinite loop because $digest will always find the old and new objects different due to the abnormal behavior of angular.copy.

var obj = {a: 1};

var obj1 = Object.create(obj);

obj1.hasOwnProperty('a'); // Returns false

var newCopy = angular.copy(obj1);

newCopy.hasOwnProperty('a'); // Returns true. Property `a` is own the object itself now.