AngularJS $timeout and $interval

Jakob Jenkov

Last update: 2014-11-19

AngularJS has two timer services, `$timeout` and `$interval`, which you can use to call functions in your application. The `$timeout` and `$interval` services are similar in their functionality to JavaScript's `setTimeout()` and `setInterval()` functions (actually belonging to the `window` object). The functionality of these services is also similar, so I will cover both in this text.

$timeout

The `$timeout` service can be used to call another JavaScript function after a given time delay. The `$timeout` service only schedules a single call to the function. For repeated calling of a function, see `$interval` later in this text.

Injecting $timeout

To use the `$timeout` service you must first get it injected into a controller function. Here is an example that injects the `$timeout` service into a controller function:

```
var myapp = angular.module("myapp", []);

myapp.controller("MyController", function($scope, $timeout){

});
```

Notice the `$timeout` parameter of the controller function. Into this parameter the `$timeout` service will be injected by AngularJS, just like any other AngularJS service you would want to use in your controller function.

Scheduling a Function Call

Once the `$timeout` service is injected into your controller function, you can use it to schedule function calls. Here is an example that used the `$timeout` service to schedule a function call 3 seconds later:

```
var myapp = angular.module("myapp", []);

myapp.controller("MyController", function($scope, $timeout){

    $timeout(callAtTimeout, 3000);

});

function callAtTimeout() {
    console.log("Timeout occurred");
}
```

This example schedules a function call to `callAtTimeout()` after 3 seconds (3000 milliseconds).

If you want to call a function on the `$scope` object, you can do so like this:

```
var myapp = angular.module("myapp", []);

myapp.controller("DIController", function($scope, $timeout){

    $scope.callAtTimeout = function() {
        console.log("$scope.callAtTimeout - Timeout occurred");
    }

    $timeout( function(){ $scope.callAtTimeout(); }, 3000);
});
```

Notice the function passed to the `$timeout` service. This function calls the `callAtTimeout()` function on the `$scope` object.

$interval

The `$interval` service is similar in function to the `$timeout` service, except it schedules a function for repeated execution with a time interval in between.

Injecting $interval

To use the `$interval` service you must have it injected into a controller function. Here is an example that injects the `$interval` service into a controller function:

```
var myapp = angular.module("myapp", []);

myapp.controller("MyController", function($scope, $interval){

});
```

As you can see, it is very similar to how you inject any other service in AngularJS.

Scheduling a Repeated Function Call

Once the `$interval` service is injected into your controller function, you can use it to schedule repeated function calls. Here is an example that used the `$interval` service to schedule a function call every 5 seconds:

```
var myapp = angular.module("myapp", []);

myapp.controller("MyController", function($scope, $interval){

    $interval(callAtInterval, 5000);

});

function callAtInterval() {
    console.log("Interval occurred");
}
```

This example schedules a function call to `callAtInterval()` every 5 seconds (5000 milliseconds).

If you want to call a function on the `$scope` object instead, you can do so like this:

```
var myapp = angular.module("myapp", []);

myapp.controller("DIController", function($scope, $interval){

    $scope.callAtInterval = function() {
        console.log("$scope.callAtInterval - Interval occurred");
    }

    $interval( function(){ $scope.callAtInterval(); }, 3000);
});
```

The function passed to the `$interval` service calls the `callAtInterval()` function on the `$scope` object.

Executing $digest() After the Scheduled Function Call

If the function you schedule for execution makes changes to variables in the `$scope` object, or make changes to any other variable which your application is watching, your application needs to execute`$scope.$digest()` after the scheduled function call finishes. Why that is necessary is explained in my tutorial about **$watch(), $digest() and $apply()**.

By default AngularJS already calls `$digest()` after the scheduled function call finishes, so you don't have to do that explicitly. You can, however, specify if AngularJS should *not* call `$digest()` after the scheduled function call. If, for instance, your scheduled function call only updates an animation but does not change any `$scope` variables, then it is a waste of CPU time to call `$digest()` after the function finishes.

Both `$timeout` and `$interval` have a third, optional parameter which can specify if the `$digest()` method is to be executed after the scheduled function finishes.

Actually, the third parameter specifies if the call to the scheduled function should be done inside an `$apply()` call. Here is an example of how to use this third parameter:

```
$interval( function(){ $scope.callAtInterval(); }, 3000, true);

$interval( function(){ $scope.callAtInterval(); }, 3000, false);
```

These two `$interval` examples both have a third parameter passed to the `$interval` service. This parameter can be either `true` or `false`. A value of `true` means that the scheduled function should be called inside an `$apply()` call. A value of `false` means that it should not be called inside an `$apply()` call (meaning `$digest()` will not get called after the scheduled function finishes).