How do we use Angular's **$timeout and $interval** services in our unit tests?

$timeout

Angular's wrapper for window.setTimeout

AngularJS $timeout Service

```
$timeout([fn], [delay], [invokeApply], [pass]);
```

AngularJS $timeout Service

```
$timeout(function() {
  console.log('I was delayed');
}, [delay], [invokeApply], [pass]);
```

AngularJS $timeout Service

```
$timeout(function() {
  console.log('I was delayed');
}, 1000, [invokeApply], [pass]);
```

AngularJS $timeout Service

```
$timeout(function() {
  console.log('I was delayed');
}, 1000, true, [pass]);
```

AngularJS $timeout Service

```
$timeout(function() {
  console.log('I was delayed');
}, 1000, false, [pass]);
```

AngularJS $timeout Service

```
$timeout(function() {
  $scope.counter += 1;
}, 1000, false, [pass]);
```

AngularJS $timeout Service

```
$timeout(function(msg) {
  console.log(msg);
}, 1000, false, 'I was delayed');
```

AngularJS $timeout Service

```
var delayedMessage = $timeout(function() {
  console.log('I was delayed');
}, 3000);


$timeout.cancel(delayedMessage);
```

Unit Testing Challenges with $timeout

Unit Testing Challenges with $timeout

- Asynchronous code is difficult to test

Unit Testing Challenges with $timeout

- Asynchronous code is difficult to test
- Delays will make test execution slow

$timeout | ngMock's wrapper for $timeout

ngMock $timeout Service

```
$timeout.flush([delay]);
```

ngMock $timeout Service

```
$timeout.flush([delay]);

$timeout.verifyNoPendingTasks();
```

$interval

Angular's wrapper for window.setInterval

AngularJS $interval Service

```
$interval(fn, delay, [count], [invokeApply], [pass]);
```

AngularJS $interval Service

```
$interval(function() {
  console.log('I run at intervals');
}, delay, [count], [invokeApply], [pass]);
```

AngularJS $interval Service

```
$interval(function() {
  console.log('I run at intervals');
}, 1000, [count], [invokeApply], [pass]);
```

AngularJS $interval Service

```
$interval(function() {
  console.log('I run at intervals');
}, 1000, 3, [invokeApply], [pass]);
```

AngularJS $interval Service

```
$interval(function() {
  console.log('I run at intervals');
}, 1000, 3, false, [pass]);
```

AngularJS $interval Service

```
$interval(function(msg) {
  console.log(msg);
}, 1000, 3, false, 'I run at intervals');
```

AngularJS $interval Service

```
var interval = $interval(function() {
  console.log('I run at intervals');
}, 1000);


$interval.cancel(interval);
```

Unit Testing Challenges with $interval

Unit Testing Challenges with $interval

- Asynchronous code is difficult to test

Unit Testing Challenges with $interval

- Asynchronous code is difficult to test
- Delays will make test execution slow

Unit Testing Challenges with $interval

- Asynchronous code is difficult to test
- Delays will make test execution slow
- How to advance through time?

$interval

ngMock's mock implementation of the $interval service

AngularJS $interval Service

```
$interval.flush([millis]);
```