**$watches** can be used to watch any value, and trigger a function call when that value changes. A $watch can be set up from any $scope by calling $scope.$watch() as shown below.

**By Expression**  
  
  
The following will watch 'foo'. Which is an expression evaluated against $scope.

1. //$scope.$watch(<function/expression>, <handler>);
3. $scope.$watch('foo', function(newVal, oldVal) {
4. console.log(newVal, oldVal);
5. });

**By Function**  
  
To set up a $watch by function, you can do the following, which is *technically* the same as what is shown above:

1. $scope.$watch(function() {
2. return $scope.foo;
3. }, function(newVal, oldVal) {
4. console.log(newVal, oldVal);
5. });

**Facts about $watch:**

* A watcher can evaluate *any value.*
* A watcher's handler can execute *anything* when aforementioned value has changed.
* All watchers are evaluated when $digest() is called.
* If the first argument of a $watch is a string, it is $eval'ed into a function prior to registration. It's functionally equivalent to passing a function as the first argument, just with an extra step internally.

### What is $digest?

At it's core, the important thing to know about $digest is that it loops through all watchers **on the scope it was called on and it's child scopes.** and evaluates them to see if they've changed, executing their handlers if they have. That's the important part you need to know.

#### How to call $digest:

1. $scope.$digest();

### What is $apply?

Simply put, it's a wrapper around $rootScope.$digest that evaluates any expression passed to it prior to calling $digest(). That's it.  So, if you're calling it by itself without passing an argument to it, you may as well just call $digest().

#### How to call $apply:

1. $scope.$apply('foo = "test"');
2. //or
3. $scope.$apply(function(scope) {
4. scope.foo = 'test';
5. });
6. //or
7. $scope.$apply(function(){
8. $scope.foo = 'test';
9. });

### So when does a $digest/$apply happen?

At key moments defined by the framework. The Angular framework has built in calls to $digest and $apply to it's services and directives as well as some internal calls. But basically it's like this, things like $timeout perform a setTimeout, then call $apply or $digest (it actual does a little more than that, but that's the basic idea). $http.get(), same deal, makes an AJAX call, returns it, then queues up a $digest.  Then there are directives, like inputs with ngModel for example. Updates to the input will also trigger a $digest. You get the idea.

### How do $watch, $digest, and $apply relate to updating my view?

* The directive registers a $watch that looks for a change in the model on the $scope. The handler will update the DOM element's value.
* The directive registers an event handler of some sort in the DOM that will get a value from the DOM and apply it to the model in $scope. It will also call $apply or $digest.
* When you update the model in the scope via some in-framework call... $http.get() for example, it kicks off a $digest after it completes.
* The $digest checks the $watch the directive registered, sees the change and fires the handler associated to it, updating the DOM element.

### Some Guidelines For Use:

* $watch
  + **DO**use $watch in directives to update the DOM when a $scope value changes.
  + **DON'T** use $watch in a controller. It's hard to test and completely unnecessary in almost every case. Use a method on the scope to update the value(s) the watch was changing instead.
* $digest/$apply
  + **DO**use $digest/$apply in directives to let Angular know you've made changes after an asynchronous call, such as a DOM event.
  + **DO**use $digest/$apply in services to let Angular know some asynchronous operation has returned, such as a WebSocket update, or an event from a 3rd party library like Facebook API.
  + **DON'T** use $digest/$apply in a controller. This will make your code harder to test, and asynchronous operations outside of the Angular framework don't belong in your controllers. They belong in services and directives.

$scope.getMessage = function() {

setTimeout(function() {

$scope.message = 'Fetched after two seconds';

console.log('message:' + $scope.message);

$scope.$apply(); //this triggers a $digest

}, 2000);

};

Example :

<div ng-controller="myController">

{{data.time}}

<br/>

<button ng-click="updateTime()">update time - ng-click</button>

<button id="updateTimeButton" >update time</button>

</div>

<script>

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

var myController1 = module.controller("myController", function($scope) {

$scope.data = { time : new Date() };

$scope.updateTime = function() {

$scope.data.time = new Date();

}

document.getElementById("updateTimeButton")

.addEventListener('click', function() {

console.log("update time clicked");

$scope.data.time = new Date();

});

});

</script>