

ng

function

[angular.bind](#)
[angular.bootstrap](#)
[angular.copy](#)
[angular.element](#)
[angular.equals](#)
[angular.errorHandlingConfig](#)
[angular.extend](#)
[angular.forEach](#)
[angular.fromJson](#)
[angular.identity](#)
[angular.injector](#)
[angular.isArray](#)
[angular.isDate](#)
[angular.isDefined](#)
[angular.isElement](#)
[angular.isFunction](#)
[angular.isNumber](#)
[angular.isObject](#)
[angular.isString](#)
[angular.isUndefined](#)
[angular.merge](#)
[angular.module](#)
[angular.noop](#)

\$setValidity(validationErrorKey, isValid);

Change the validity state, and notify the form.

This method can be called within `$parsers/$formatters` or a custom validation implementation. However, in most cases it should be sufficient to use the `ngModel.$validators` and `ngModel.$asyncValidators` collections which will call **\$setValidity** automatically.

Parameters

Param	Type	Details
validationErrorKey	string	Name of the validator. The <code>validationErrorKey</code> will be assigned to either <code>\$error[validationErrorKey]</code> or <code>\$pending[validationErrorKey]</code> (for unfulfilled <code>\$asyncValidators</code>), so that it is available for data-binding. The <code>validationErrorKey</code> should be in camelCase and will get converted into dash-case for class name. Example: <code>myError</code> will result in <code>ng-valid-my-error</code> and <code>ng-invalid-my-error</code> classes and can be bound to as <code>{{ someForm.someControl.\$error.myError }}</code> .
isValid	boolean	Whether the current state is valid (true), invalid (false), pending (undefined), or skipped (null). Pending is used for unfulfilled <code>\$asyncValidators</code> . Skipped is used by AngularJS when validators do not run because of parse errors and when <code>\$asyncValidators</code> do not run because any of the <code>\$validators</code> failed.

Properties