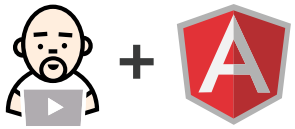

egghead.io presents

Core Injectable Services

CORE INJECTABLE SERVICES

- `$anchorScroll` - checks current value of `$location.hash()` and scrolls to the related element
- `$cacheFactory(cacheName)` - constructs Cache objects and gives access to them
- `$compile(html)(scope)` - Compiles an HTML string or DOM into a template and produces a template function
- `$controller(constructor, locals)` - responsible for instantiating controllers
- `$exceptionHandler(exception[, cause])` - uncaught exception in angular expressions is delegated to this service
- `$filter(name)` - used for formatting data displayed to the user
- `$http[(options)]` - facilitates communication with the remote HTTP servers
- `$parse(expression)` - Converts Angular expression into a function

$animate - provides rudimentary DOM manipulation functions

Methods:

- `enter(element, parent, after, [done])` - Inserts the element into the DOM either after the after element or as the first child within the parent element
- `leave(element, [done])` - Removes the element from the DOM
- `move(element, parent, after, [done])` - Moves the position of the provided element within the DOM to be placed either after the after element or inside of the parent element
- `addClass(element, className, [done])` - Adds the provided className CSS class value to the provided element
- `removeClass(element, className, [done])` - Removes the provided className CSS class value from the provided element
- `setClass(element, add, remove, [done])` - Adds and/or removes the given CSS classes to and from the element

$q - promise/deferred implementation inspired by kris kowal's q

Methods:

- `defer()` - Creates deferred object to expose the associated Promise instance as well as APIs that can be used for signaling the successful or unsuccessful completion, as well as the status of the task
- `all(promisesArray|promisesHash)` - Combines multiple promises into a single promise that is resolved when all of the input promises are resolved

the deferred object

Methods:

- `resolve(value)` – resolves the derived promise with the value
- `reject(reason)` – rejects the derived promise with the reason
- `notify(value)` - provides updates on the status of the promise's execution

Properties:

`promise` - promise object associated with the deferred (see below)

the promise object

- `then(successCallback, errorCallback, notifyCallback)` - calls one of the success or error callbacks asynchronously as soon as the result is available
- `catch(errorCallback)` – shorthand for promise.then(null, errorCallback)
- `finally(callback)` – allows you to observe either the fulfillment or rejection of a promise, but to do so without modifying the final value. This is useful to release resources or do some clean-up.

$rootScope - every application has a single root scope. All other scopes are descendant scopes of the root scope

$sce - provides strict contextual escaping services to angularjs

Methods:

- `isEnabled()` - Returns a boolean indicating if SCE is enabled.
- `parse(type, expression)` - Converts Angular expression into a function
- `trustAs(type, value)` - Delegates to $sceDelegate.trustAs
- `trustAsHtml(value)` - Shorthand method. $sce.trustAsHtml(value) → $sceDelegate.trustAs($sce.HTML, value)
- `trustAsUrl(value)` - Shorthand method. $sce.trustAsUrl(value) → $sceDelegate.trustAs($sce.URL, value)
- `trustAsResourceUrl(value)` - Shorthand method. $sce.trustAsResourceUrl(value) → $sceDelegate.trustAs($sce.RESOURCE_URL, value)
- `trustAsJs(value)` - Shorthand method. $sce.trustAsJs(value) → $sceDelegate.trustAs($sce.JS, value)
- `getTrusted(type, maybeTrusted)` - Delegates to $sceDelegate.getTrusted
- `getTrustedHtml(value)` - Shorthand method. $sce.getTrustedHtml(value) → $sceDelegate.getTrusted($sce.HTML, value)
- `getTrustedCss(value)` - Shorthand method. $sce.getTrustedCss(value) → $sceDelegate.getTrusted($sce.CSS, value)
- `getTrustedUrl(value)` - Shorthand method. $sce.getTrustedUrl(value) → $sceDelegate.getTrusted($sce.URL, value)
- `getTrustedResourceUrl(value)` - Shorthand method. $sce.getTrustedResourceUrl(value) → $sceDelegate.getTrusted($sce.RESOURCE_URL, value)
- `getTrustedJs(value)` - Shorthand method. $sce.getTrustedJs(value) → $sceDelegate.getTrusted($sce.JS, value)
- `parseAsHtml(expression)` - Shorthand method. $sce.parseAsHtml(expression string) → $sce.parseAs($sce.HTML, value)
- `parseAsCss(expression)` - Shorthand method. $sce.parseAsCss(value) → $sce.parseAs($sce.CSS, value)
- `parseAsUrl(expression)` - Shorthand method. $sce.parseAsUrl(value) → $sce.parseAs($sce.URL, value)
- `parseAsResourceUrl(expression)` - Shorthand method. $sce.parseAsResourceUrl(value) → $sce.parseAs($sce.RESOURCE_URL, value)
- `parseAsJs(expression)` - Shorthand method. $sce.parseAsJs(value) → $sce.parseAs($sce.JS, value)

$templateCache - the first time a template is used, it is loaded into the template cache.

browser wrappers - Set of wrappers to access key components of the browser as injectable objects. Very useful for testing.

- `$interval(fn, delay, [count], [invokeApply])` - wrapper for window.setInterval
- `$location` - parses the URL in the browser address bar (based on the window.location)
- `$log` - safely writes a message into the browser's console (if present)
- `$timeout(fn[, delay][, invokeApply])` - wrapper for window.setTimeout
- `$window` - reference to the browser's window object
- `$document` - wrapper for the browser's window.document object