AngularJS 1.x “Strict Mode” API draft

Currently, “strict mode” entails disabling the automatic annotation of functions in Angular’s injector. Prototype implementation is at <https://github.com/angular/angular.js/pull/6719>  
  
**Raison d’etre:**

When AngularJs 1.x code is minified, any method invoked by the injector, which does not explicitly annotate itself using the $inject property or array notation, will break the application. This can be difficult to debug, as you will typically only see this when the app is minified, and names are mangled.  
  
**Requirements:**

1. The application (and not its required modules) should decide whether or not the injector is created in “strict” mode.  
     
   **Constraints/Caveats:**
   1. When this behaviour is a feature of the module API (loader.js), it is difficult to know what the “root” module actually is. The bootstrap API may know how to do this, when it creates the injector. However, this feels clunky, since the set of modules passed to createInjector() is not a tree-like structure, and bootstrap implicitly prepends several modules to it.
2. This should be declarative. One strategy is to add special attributes which define module flags for the bootstrapped (root) module
   1. an existing implementation of this is `ng-app-strict`, which can be seen at <http://goo.gl/wLu0Is>.
   2. an alternative would be to provide a second attribute that is read by ng-app, say `require-annotations=”true”`  
        
      **Constraints/Caveats:**
   3. To use @petebacondarwin’s suggested module.requireAnnotations() API in the current implementation of angular-mocks, it’s not really possible because mocks never give you the module object from loader.js. This would require a new helper function to be added to angular-mocks, which might be called in a beforeEach() block.  
        
      This is unfortunately awkward, and the ordering of when this is called can matter. So, while it is a simple and fairly elegant API, there are some complexities that it can introduce.
3. Tests should also be able to opt for this behaviour using angular-mocks. Currently, a test suite can opt into this by calling `angular.mock.module(‘moduleName’, function moduleFn() { … }, true)` --- in short, by passing the boolean value `true` to the mock. This is, needless to say, not ideal.
4. The mock inject() routine should be exempt from the rules of requiring explicit annotations, for simplicity, and to avoid breaking tests.

Other considerations

1. How will this change impact users of Closures @ngInject annotation
2. How will this change impact users of ngMin

<html ng-app="myApp" ng-config="{requireAnnotations: false}">

<html ng-app="myApp" ng-require-annotations>

<html ng-app="myApp" ng-strict={object}>

angular.bootstrap(rootElement, [...modules], {config1: 'val1'})

beforeEach(inject.requireAnnotations)

it('should do...', inject(function($http) {

));

function inject(spec) {

injector.annotate(spec);

injector.invoke(

….

}

…

if (currentSpec.$injectorNoMagic) {

// If the injector is strict / noMagic, and the spec wants to inject using automatic

// annotation, then annotate the function here.

injector.annotate(blockFns[i]);

}

…

angular.injector.$$annotate(fn)

angular.module ('myApp', [])

.hasAnnotations() / .requireAnnotations()

.factory()

.etc

The bootstrapper would only look at the root module to decide whether to check annotations.

(Or maybe it could only check for annotations if all modules declare they have annotations. )

Inside tests you could use a helper...

beforeEach(checkAnnotations);