How to do Everything in AngularJS

Prevent Route Change

1. Add `target="_self"` to all elements
2. Create new directive to prevent the defaults:
3. ```
   app.directive('a', function() {
     return {
       restrict: 'E',
       link: function(scope, elem, attrs) {
         if (attrs.ngClick || attrs.href === '' || attrs.href === '#') {
           elem.on('click', function(e) {
             e.preventDefault();
           });
         }
       }
     };
   });
   ```

Updating Model Within Directives

```
App.directive('myDirective', function ($parse) {
  return {
    require: 'ngModel',
    link: function (scope, elm, attrs, ctrl) {
      ctrl.$setViewValue(newValue);
      ctrl.$render();
      e.preventDefault();
      scope.$apply();
    });
  };
});
```
```

Making AngularJS Work with Bootstrap Vertical Button Group

```
<div class="btn-group-vertical" ng-class="{'active':selected}">
    <input type="checkbox" autocomplete="off" ng-checked="selected" />
</div>
```

<Enter> Key Event Directive

JS:

```
app.directive('ngEnter', function () {
  return function (scope, element, attrs) {
    element.bind("keydown keypress", function (event) {
      if(event.which === 13) {
        scope.$apply(function (){
          scope.$eval(attrs.ngEnter);
        });
        event.preventDefault();
      }
    });
  };
});
```

HTML Usage:

```
<div ng-app="" ng-controller="MainCtrl">
    <input type="text" ng-enter="doSomething()">
</div>
```

Passing Functions to Directives

HTML:

```
<test color1="color1" update-fn="updateFn(msg)"></test>
```

JS:

```
var app = angular.module('dr', []);

app.controller("testCtrl", function($scope) {
  $scope.color1 = "color";
    $scope.updateFn = function(msg) {
```

```
    alert(msg);
  }
});

app.directive('test', function() {
  return {
    restrict: 'E',
    scope: {
      color1: '=',
      updateFn: '&'
    },
    // object is passed while making the call
    template: "<button ng-click='updateFn({msg : \"Hello World!\"})'>Click</button>",
    replace: true,
    link: function(scope, elm, attrs) {
    }
  }
});
```

Angular JS Best Practices

1. Do not put the main controller into the main module, instead the main controller should be declared in a new module, this will make the application more modular e.g.

```
angular.module('app', ['Controller'])
angular.module('Controller', []).controller('Controller', function($scope) {
  $scope.something = 100
})
```

2. Do not call functions for ng-show and ng-hide directives, as it may degrade performance

3. Using too much $scope.$watch is going to degrade performance, try only watch what you really need to and remove the watchers when it's not needed anymore

---

Revision #1
Created 5 April 2017 21:52:32 by Tingwai
Updated 5 April 2017 22:10:10 by Tingwai