[**Angular中ngModel的$render的详解**](http://www.cnblogs.com/HeJason/p/5486991.html)

**Render的作用：将modelValue更新到viewValue. 但是，我们可以重写该方法，从而实现个性化的VIEW展示，比如：自定义表单元素**

**以下是详细分析：**

　在我开始着手ngModel的领域时候，有一个问题很令我纠结，那就是$render()到底是做什么的呢？查了很多资料都只是简单的描述一下，这就令我很纠结了，终于在一个阳光明媚的晚上，我终于解决了这个大问题

　　那么这个$render方法到底是干什么的呢？他的用处就是在$viewValue改变的时候可以重新绑定model数据，但是我们要注意一点(**$viewValue和DOM节点的value是不同的**)，我觉得他们的区别有点类似setTimeout和$timeout的区别，但是又不太一样。ps:**其实modelValue和绑定的数据也可以不同**

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

Input里面模型的值：{{vm.modelTest}}

<input type="text" ng-model="vm.modelTest" model-render>

http://images.cnblogs.com/OutliningIndicators/ExpandedBlockStart.gif

[复制代码](javascript:void(0);)

.directive('modelRender', function () {

return {

require: 'ngModel',

link: function (scope, iElm, iAttrs, ngModelCtrl) {

iElm.on('mouseenter', function () {

//尝试注释

iElm.val(1);

console.log(ngModelCtrl);

//尝试注释

ngModelCtrl.$setViewValue(11);

console.log(ngModelCtrl);

//尝试注释

ngModelCtrl.$render();

console.log(ngModelCtrl);

})

}

}

})

[复制代码](javascript:void(0);)

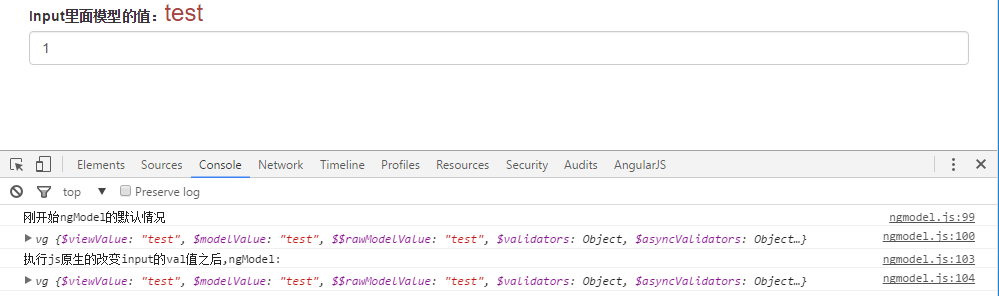
 我们分几种情况分析

这是鼠标没有经过指令的时候的样子



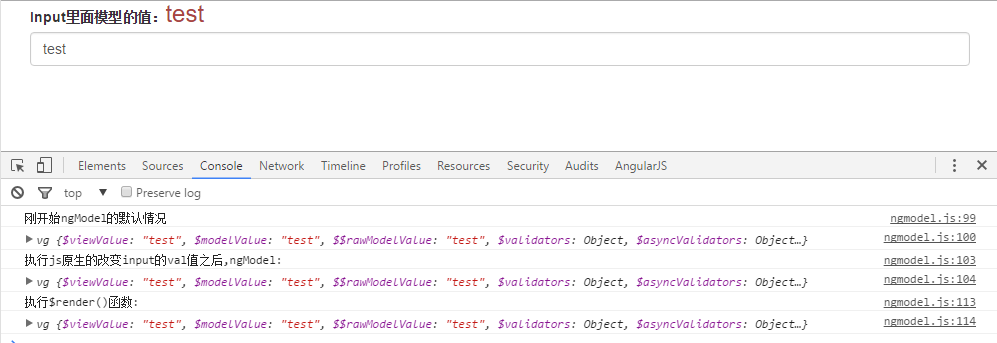
　　1.当我们使用js原生方法设置input的val值的时候，并且不执行$render函数，我们可以看到input里面的model值是没有变化的，但是input的的value是变成了1，而且我们看到不仅model值没有变化，ngModel的$viewValue和$modelValue同样也没有变化。我们可以得出结论 (input的value值不一定等于$viewValue)

　　　结果是这样的



-----------------------------------------------------------------------------------------------------------------------

　　然后，我们尝试在执行js原生改变value值之后，执行$render函数是个什么样的状况，



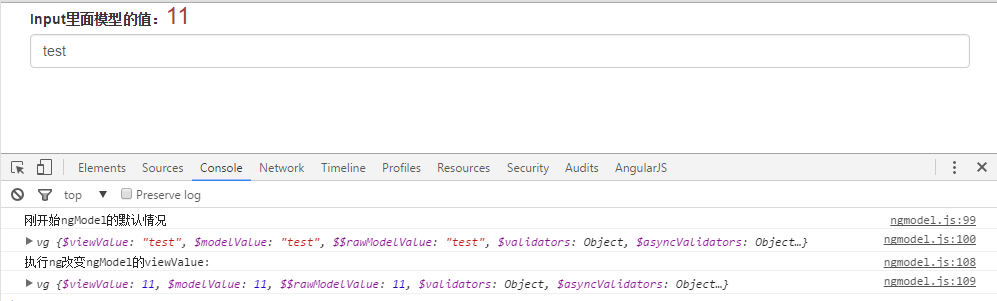
　　看完上面的实验之后我们发现input的value值并没有发生变化，也就是说js原生改变input的value值是无效的，那么在这里我们就可以看到$render的功能了。

**我们可以大胆的预计$render的功能跟$apply的功能是一致的，我们在上一章讲过，$apply是以viewValue为主，让modelValue变成viewValue，也就是modelValue -> viewValue，那么$render是不是以modelValue为主，让viewValue->modelValue呢？**

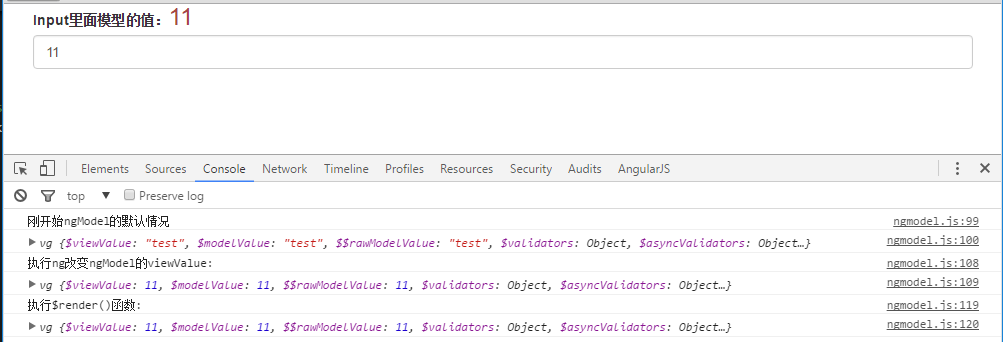
-------------------------------------------------------------------------------------------------------------------------

　　2.接下来我们尝试，使用ng原生改变 也就是说$setviewValue,是如何表现的呢？

　　　现在我们注释掉js原生改变value的方法，而去使用$setViewValue，并且不执行$render函数，直接上结果，我们看到，执行完$setViewValue之后，无论是viewValue和modelValue都是已经同步了，但是input里面的值却依然是test,在这里我们再次验证了那个说法(**$viewValue和DOM节点的value是不同的**)



　　现在我们在$setViewValue之后使用，$render()看看是什么效果，



　　大家发现了吧，$render的功能和$apply的功能极为相似，但是是不是很多人在讲$render的时候，都会说model同步到view，我觉得这个说法不太对，我测试过在**click事件**用非常规手段改变controller中model的值，发现就算controller的值已经改变了，但是ngModel的值无论是viewValue还是modelValue都没有变化，然后尝试用$modelValue的属性强行改变$modelValue，结果还是没作用。我们下面来看看$render的源码

ctrl.$render = function() {

element.val(ctrl.$isEmpty(ctrl.$viewValue) ? '' : ctrl.$viewValue);

};

　　这是其中一个，$render在不同的指令下的代码都不太一样，但是其作用基本一致，但是从这里我们就可以看出$render的到底在干什么事了。那么$render什么时候触发呢？其实看你自己想什么时候调用它，你可以覆盖他的方法，重写，在$watch也好，$viewChange也好。默认的触发事件一些特别input的value改变的时候例如单选，还有rollbackView()的时候

　　另外一个真正体现$render执行事件的源代码在这里，里面我写了注释，大家应该都能懂

[复制代码](javascript:void(0);)

$scope.$watch(function ngModelWatch() {

　　//解析ngModel的表达式，获取内容

var modelValue = ngModelGet($scope);

// if scope model value and ngModel value are out of sync

// TODO(perf): why not move this to the action fn?

　　//判断表达式的值是否跟modelValue一致

if (modelValue !== ctrl.$modelValue &&

// checks for NaN is needed to allow setting the model to NaN when there's an asyncValidator

(ctrl.$modelValue === ctrl.$modelValue || modelValue === modelValue)

) {

　　　　//更新modelValue的值

ctrl.$modelValue = ctrl.$$rawModelValue = modelValue;

parserValid = undefined;

　　　　//获取管道信息

var formatters = ctrl.$formatters,

idx = formatters.length;

var viewValue = modelValue;

while (idx--) {

viewValue = formatters[idx](viewValue);

}

　　　　//如果viewValue和ModelValue不一致

if (ctrl.$viewValue !== viewValue) {

ctrl.$viewValue = ctrl.$$lastCommittedViewValue = viewValue;

ctrl.$render();

ctrl.$$runValidators(modelValue, viewValue, noop);

}

}

　　//返回解析的表达式

return modelValue;

});

}];

[复制代码](javascript:void(0);)

**好了，$render就已经讲完了，这是作者的原创帖，如转载 请注明，最后贴段代码感受下**