

---

# Table of Contents

扉页	1.1
介绍	1.2
who	1.2.1
vm	1.2.2
虚拟DOM	1.2.3
指令	1.3
插值表达式	1.3.1
ms-skip	1.3.2
ms-controller	1.3.3
ms-important	1.3.4
ms-attr	1.3.5
ms-css	1.3.6
ms-text	1.3.7
ms-html	1.3.8
ms-class	1.3.9
ms-active	1.3.10
ms-hover	1.3.11
ms-if	1.3.12
ms-visible	1.3.13
ms-for	1.3.14
ms-on	1.3.15
ms-duplex	1.3.16
ms-rules	1.3.17
ms-validate	1.3.18

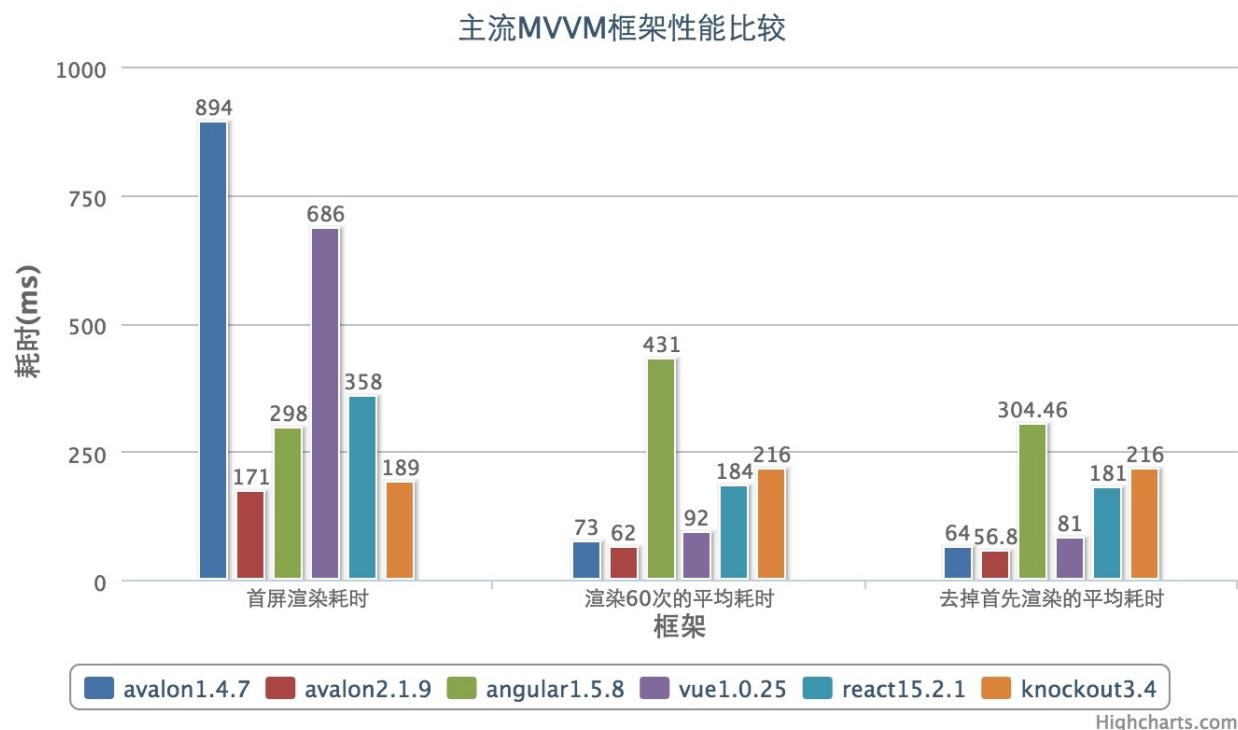
---

<a href="#">ms-effect</a>	1.3.19
<a href="#">ms-widget</a>	1.3.20
<a href="#">自定义标签</a>	1.3.21
<a href="#">组件</a>	1.4
<a href="#">过滤器</a>	1.5
<a href="#">类型转换器</a>	1.6
<a href="#">表单验证</a>	1.7
<a href="#">配置</a>	1.8
<a href="#">移动端支持</a>	1.9
<a href="#">常见问题</a>	1.10
<a href="#">与jQuery混用</a>	1.11
<a href="#">API</a>	1.12
<a href="#">更新日志</a>	1.13

---

# avalon 2

迷你、易用、高性能的前端MVVM框架



测试页面avalon仓库[perf](#)目录下的 `index.html`, `index.*.html` 文件

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间: 2016-07-27 21:35:06

## 简介

avalon2是一款基于虚拟DOM与属性劫持的 迷你、 易用、 高性能 的 前端 MVVM框架， 拥有超优秀的兼容性, 支持移动开发, 后端渲染, WEB Component式组件开发, 无需编译, 开箱即用。

## 谁在使用avalon

## 使用方式

avalon2: <https://github.com/RubyLouvre/avalon/tree/2.1.5/dist>

CDN: <https://cdnjs.com/libraries/avalon.js> (推荐)

CDN: <http://www.bootcdn.cn/avalon.js/>

**注意**，不要使用avalon2.0s,avalon2.0b1,avalon2.0b2,那是很早期的beta版本

```
npm install avalon2
```

下面回来,你发现有一个dist目录,然后使用script标签引用当中的avalon.js或avalon.modern.js文件

avalon2是使用一份源码编译成N个版本:

avalon 支持IE6+及古老的W3C浏览器(判定标准是  
这些浏览器是否支持VBScript, \_\_defineSetter\_\_, \_\_defineGetter\_\_  
\_)

avalon.modern 支持IE10+及较新的W3C浏览器(判定标准是  
这些浏览器是否支持Object.defineProperty, addEventListener)

avalon.next 支持IE12+(edge)及chrome49, firefox49(判定标准是  
这些浏览器是否支持Proxy, document.registerElement)

## 学习资料

[avalon2-seed](#) 这是一个avalon项目的框架，可以将它作为avalon项目的工程  
初始化目录配置。

[webpack+avalon的学习教程](#)

[1.4的入门教程](#)

[1.4的仓库地](#)

[1.4的另一个更漂亮的教程](#)

[1.5的入门教程](#)

[1.5的仓库地址](#)

[1.\\*的视频教程](#)

[1.\\*的官网地址](#)

[前端乱炖网站上的avalon1.5学习教程](#)

[segmentfault网站上的avalon2学习教程](#)

[avalon论坛](#)

QQ学习群: 314247255

电子书下载：[avalon cookbook](#)

## 入门例子

```
<!DOCTYPE html>
<html>
  <head>
    <title>first example</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="./dist/avalon.js"></script>
    <script>
      var vm = avalon.define({
        $id: "test",
        name: "司徒正美",
        array: [11, 22, 33]
      })
      setTimeout(function(){
        vm.array.set(0, 444)
      }, 3000)
    </script>
  </head>

  <body ms-controller="test">
    <input ms-duplex="@name">
    <p>Hello, {{@name}}!</p>
    <ul>
      <li ms-for="($index,el) in @array">{{$index}}--{{el}}</li>
    </ul>
  </body>
</html>
```

这里面涉及一些知识点

1. vm, 使用`avalon.define`方法生成, 必须带\$id属性
2. 指令, 以ms-开头的属性及双花括号的插值表达式
3. 圈定作用域, 使用`ms-controller`告诉框架, 只处理这个范围内的标签
4. 引导符, 使用`@` 或 `##` 来告诉框架这些变量是来自vm的
5. 自动扫描机制

avalon2.1.15后还可以使用`:xxxx` 短指令.

```
<body :controller="test">
  <input :duplex="@name">
  <p>Hello,{{@name}}!</p>
  <ul>
    <li :for="($index,el) in @array">{{$index}}--{{el}}</li>
  </ul>
</body>
```

## 后端渲染

<https://github.com/RubyLouvre/avalon-server-render-example>

## avalon起源

avalon 是一个简单易用迷你的MVVM框架, 它最早发布于2012.09.15, 为解决同一业务逻辑存在各种视图呈现而开发出来的。事实上, 这问题其实也可以简单地利用一般的前端模板加jQuery 事件委托 搞定, 但随着业务的膨胀, 代码就充满了各种选择器与事件回调, 难以维护。因此彻底的将业务与逻辑分离, 就只能求助于架构。最初想到的是MVC, 尝试过backbone, 但代码不降反升, 很偶尔的机会, 碰上微软的WPF, 优雅的MVVM架构立马吸引住我, 我觉得这就是我一直寻找的解决之道。

avalon将所有前端代码彻底分成两部分，视图的处理通过绑定实现（angular有个更炫酷的名词叫指令），业务逻辑则集中在一个个叫VM的对象中处理。我们只要操作VM的数据，它就自然而然地神奇地同步到视图。显然所有神秘都有其内幕，C#是通过一种叫访问器属性的语句实现，那么JS也有对应的东西。感谢上帝，IE8最早引入这东西（`Object.defineProperty`），可惜有BUG，但带动了其他浏览器实现它，IE9+便能安全使用它。对于老式IE，我找了好久，实在没有办法，使用VBScript实现了。

`Object.defineProperty`或VBS的作用是将对象的某一个属性，转换一个setter与getter，我们只要劫持这两个方法，通过Pub/Sub模式就能偷偷操作视图。为了纪念WPF的指引，我将此项目以WPF最初的开发代号avalon来命名。它真的能让前端人员脱离DOM的苦海，来到数据的乐园中！

avalon的所有指令都是以`ms-*`命名，ms是用纪念我之前的一个框架 mass Framework！

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间：2016-09-04 21:08:17

## 谁在使用avalon

欢迎各位使用者到QQ群找作者提交你们公司的LOGO与链接





滴滴一下 美好出行



octmami









• **昆布师**



• **咖狗网**

• 有货 全球 就上咖狗网



Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间：2016-08-10 11:14:16

## vm

avalon的所有操作都是围绕vm进行。 vm, 亦即view model, 视图模型。只要我将一个JS对象添加一个\$id属性, 再放到avalon.define方法里面, 就能得到一个vm。

```
var vm = avalon.define({
    $id: "start",
    name: "test"
})
```

vm是一种利用Proxy或 Object.defineProperties或VBScript创建的特殊对象。

里面以\$带头的属性或放到\$skipArray, 都转换为访问器属性, 也就是其他语言的setter, getter。因此如果这个属性最初没有定义, 那么它就不会转换为访问器属性, 修改该属性, 就不会刷新视图。

avalon定义了的vm, 都可以在avalon.vmodels中查看到。我们可以在chrome控制台下看一下刚才的start vm的构造。

vm

```
▼ start: Observer
  ► $accessors: Object
    $element: null
  ► $events: Object
  ► $fire: function (expr, a, b)
    $hashcode: "$495412453029"
    $id: "start"
    $model: (...)

  ► get $model: function ()          访问器属性$model
  ► set $model: function ()
    $render: 0
    $track: "name"
  ► $watch: function ()
  ► hasOwnProperty: function hasOwnProperty(key)
    name: (...)

  ► get name: function get()
  ► set name: function (val)
  ► __proto__: Object
```

平时而言，vm是一种比较重型的对象。从占用内存角度来划分，浏览器中的四种对象排行如下：

1. 超轻量 Object.create(null)
2. 轻量 一般的对象 {}
3. 重量 带有访问器属性的对象, avalon VM对象
4. 超重量 各种节点或window对象

我们构建VM时只允许存在普通对象(不能是某个函数的实例),函数,数组,数字,字符串,布尔,其他一切不支持(undefined与null不能出现在定义VM时,只能用它们来赋值)

## 内部属性

VM中以\$开头的属性都是框架保留使用的特殊属性,大家为数据起名字时要小心避开

这些以\$开头的属性,目前除了\$id, \$events, \$watch, \$fire, \$model比较稳定外,其他系统属性在不同版本存在增删的情况.

1. \$id, vm的名字
2. \$watch, 用于添加监听函数
3. \$fire, 用于触发监听函数
4. \$events, 用于储存监听函数
5. \$model, 返回一个纯净的JS对象
6. \$hashcode, 2.0新增,由于\$id无法保证唯一性,使用这个作为UUID
7. \$accessors, 用于放置访问器的定义,出现在兼容片的avalon VM上.
8. \$render, 2.0新增, 用于生成虚拟DOM树.
9. \$element, 2.0新增, 当我们用ms-controller, ms-important指定一个VM的作用域,对应元素节点会放到这个属性上.
10. \$track, 1.5新增, 用于实现hasOwnProperty方法. VM的hasOwnProperty会对系统属性返回false

另外, avalon不允许在VM定义之后, 再追加新属性与方法, 比如下面的方式是错误的:

```
var vm = avalon.define({  
    $id: "test",  
    test1: "点击测试按钮没反应 绑定失败"  
})  
vm.one = function () { //不能再追加此方法  
    vm.test1 = "绑定成功"  
}
```

但我们可以以下方式, 实现添加子属性。

```

var vm = avalon.define({
    $id: "test",
    placeholder: {}
});
setTimeout(function () {
    vm.placeholder = { //我们必须通过 =， 直接添加一个对象来添加子属性
        // 不能
        aaa: 1, //vm.placeholder.aaa =1; vm.placeholder.bbb = 2这样分散地添加子属性
        bbb: 2
    }
}, 1000)

```

VM中的数据更新，只能通过 = 赋值方式实现。但要注意在IE6-8，由于VM是一个VBScript对象，为VM添加新属性会抛错，因此我们想批量更新属性要时格外小心了，需要用hasOwnProperty进行过滤。

注意在IE6-8 下，err是VBscript的关键字，VM中存在这个字段，就会将VM中的其他数组变成字符串，详见[这里](#)

为了性能起见，请确保你的对象结构足够扁平，套嵌层次不能太深，里面的数组不能太长。

## 监控属性

在VM中，改变它们会引起视图改变的属性。如果一个属性是\$开头，或在定义时放在\$skipArray数组中，或是函数或节点元素，它们都不会转换成监控属性。

此外，改变监控属性的值还会触发对应的\$watch监听回调。

## 监控数组

操作此数组的方法会同步视图的特殊数组，它是由VM中的数组自动转换而来。方便与ms-repeat, ms-each配合使用，能批量同步一大堆DOM节点。

监控数组的方法与普通数组没什么不同，它只是被重写了某一部分方法，如 pop, shift, unshift, push, splice, sort, revert。其次添加了四个移除方法，remove, removeAt, removeAll, clear，及ensure, pushArray, set方法。

1. pushArray(el), 要求传入一数组，然后将它里面的元素全部添加到当前数组的末端。
2. remove(el), 要求传入一元素，通过全等于比较进行移除。
3. removeAt(index), 要求传入一数字，会移除对应位置的元素。
4. removeAll(arrayOrFunction), 有三种用法，如果是一个函数，则过滤比较后得到真值的元素，如果是一数组，则将此数组中与原数组相等于的元素全部移除；如果没有任何参数，则全部清空。
5. clear(), 相当于removeAll()的第三种方法，清空数组的所有元素。由于需要同步视图的缘故，不能通过vm.array.length = 0的方法来清空元素。
6. ensure(el), 只有当数组不存在此元素时，才添加此元素。
7. set(index, el), 用于更新某一索引位置中的元素，因为简单数组元素的数组，是不会转换它的元素。

注意，修改某个数组元素必须使用set方法。如果是修改 对象数组 的某个元素的属性可以用 `vm.array[1].prop = 'newValue'`

```
<body ms-controller="test">
    <div ms-for="el in @arr">
        {{el}}<button type="button" ms-click="@arr.remove(el)">点我
        删除该行</button>
    </div>
    <script>
        avalon.define({
            $id: 'test',
            arr: [1,2,3,4,5,6]
        })
    </script>
</body>
```

## 非监控属性

这包括框架添加的\$id, \$events, \$model属性, \$fire, \$watch, \$render方法, 及用户自己设置的以\$开头的属性, 放在\$skipArray数组中的属性, 值为函数、各种DOM节点的属性, 总之, 改变它们的值不会产生同步视图的效果。

## \$watch方法

在avalon早期是, 存在一个对象能Mixin进每个VM, 让VM具有\$watch, \$unwatch, \$fire, \$events等方法或属性. 这有点像jQuery的on, off, trigger方法, 只是为了更靠近angular等MVVM框架, 名字起成这样.

此方法是用于监听vm中的对象的属性变化.

换言之, 它不能监听函数, 不能监听简单数组的元素变化(如[1,2,3]变成[4,2,3])

它能监听子级对象的属性变化, 能监听对象数组的属性变化(如[{a:1,b:2}]变成[{a:'change',b:2}]), 还有数组的长度属性变化

此外从1.5起, 支持"~"通配符, 解决对数组元素, 子属性的监听. 注意, 号只能出现一次.

下面是\$watch方法的七种用法

```
var vm = avalon.define({
    $id: "test",
    array: [1, 2, 3],
    d: 888,
    arr: [
        {a: 1},
        {a: 2},
        {a: 3}
    ],
    obj: {
        a: 1,
        b: 2
    }
})
```

```
    },
    a: {
      b: {
        c: {
          d: 33
        }
      }
    }
  })
var expect = function (a) {
  return {
    to: {
      be: function (b) {
        console.log(a == b)
      }
    }
  }
}

vm.$watch("array.length", function (a, b, name) {
  console.log('第一组 数组长度', name)
})
vm.$watch("arr.*.a", function (a, b, name) {
  expect(a).to.be(99)
  expect(b).to.be(1)
  console.log('第二组 数组元素属性(模糊匹配, 不知道哪个元素变化)', name)
})
vm.$watch("obj.a", function (a, b, name) {
  expect(a).to.be(111)
  expect(b).to.be(1)
  console.log('第三组 属性的属性', name)
})

vm.$watch("obj.*", function (a, b, name) {
  expect(a).to.be(111)
  expect(b).to.be(1)
```

```

    console.log('第四组 属性的属性(模糊匹配)', name)
})

vm.$watch("a.b.c.d", function (a, b, name) {
    expect(a).to.be(88)
    expect(b).to.be(33)
    console.log('第五组 属性的属性的属性', name)
})
vm.$watch("a.*.c.d", function (a, b, name) {
    expect(a).to.be(88)
    expect(b).to.be(33)
    console.log('第六组 属性的属性的属性(模糊匹配)', name)
})
vm.$watch("*", function (a, b, name) {
    expect(a).to.be(999)
    expect(b).to.be(888)
    console.log('第七组 第一层对象的任意属性(模糊匹配)', name)
})
setTimeout(function () {
    vm.array.set(1, 6)
    vm.array.push(99)
    vm.arr[0].a = 99
    vm.obj.a = 111
    vm.a.b.c.d = 88
    vm.d = 999
}, 100)

```

\$watch会返回一个函数,用于解除监听:

```

var unwatch = vm.$watch("array.*", function (a, b) {
    expect(a).to.be(6)
    expect(b).to.be(2)
})
unwatch() //移除当前$watch回调
`
```

监听函数有三个参数， 第一个是新值， 第二个是旧值， 第三个是发生变动的属性的名字。

`$watch`方法供与其他操作DOM的库一起使用的,如富文本编辑器什么. 在`$watch`回调里更新VM自身的属性是非常危险的事,很容易引发死循环

## \$fire方法

`$fire`可以传多个参数， 第一个参数为事件名， 或者说是VM上已存在的属性名， 当VM中对应的属性发生变化时， 框架内部就调用`$fire`方法， 依次传入属性名， 当前属性值， 过去属性值。

## 数据模型

是指VM中的`$model`属性， 它是一个纯净的javascript对象， 去掉`$id`, `$watch`等方法或属性， 可以直接通过`$.ajax`提交给后端， 当然我们还可以通过`JSON.parse(JSON.stringify(vm.$model))`干掉里面的所有函数。

注意,不要修改`$model`,你只能通过VM来改动`$model`,否则在1.5中,`$model`是只读的,每次都是返回一个全新的对象给你 你改了也没有用!

## vm是如何作用视图

我们需要在页面上， 使用`ms-controller`或`ms-important`来圈定每个`vm`的作用范围。当页面`domReady`时， `vm`就将自动将其里面的数据替换到各种指令中去， 实现视图刷新效果。

注意一个`vm`只能在页面上使用一次。即页面上不能重复出现相同的值的`ms-controller`。

```
<div ms-controller="test">{{@aaa}}</div>
<div ms-controller="test">{{@aaa}}</div>
<div ms-controller="test">{{@aaa}}</div>
```

---

由于test这个vm拥有一个叫\$element的属性，它是保存其关联的元素节点，如果定义了多少个，那么它会保留最后的那个DIV。以后它的属性变化，只会作用最后的那个DIV。

## vm的运作原理

avalon之所以使用Proxy, Object.defineProperty或VBScript来构造vm，那是因为它们创建出来的对象有一种自省机制，能让我们得知vm正在操作或访问了我们的对象。

对于Object.defineProperty或VBScript，主要是靠将普通属性变成访问器属性。访问器属性内部是拥有两个方法，setter与getter。当用户读取对象的属性时，就将调用其getter方法，当用户为此属性赋值时，就会调用setter方法。因此，我们就不需要像angular那样，使用脏检测，就得知对象被修改了某些属性了。并且能准确得知那些属性，及时地同步视图的相应区域，实现最小化刷新视图。

对于Proxy(智能代理)，这最早发迹于firefox4，现在许多新浏览器都支持，它能监听外部用户对它的14种，比如说读写属性，调用方法，删除旧属性，添加新属性，被for in循环，被in关键字进行存在性检测，被new……因此之前所说的，不能监听没预先定义的属性，这个难题被Proxy搞定了。

当我们得知vm的属性发生了变化了，如何更新视图呢？在avalon2中，这个是由[虚拟DOM](#)来处理。

虚拟DOM比较复杂，大家看不懂可以略过。

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间：2016-09-02 20:21:24

# 虚拟DOM

avaon会在DOMReady对.ms-controller节点进行outerHTML操作 之前是直接用outerHTML,但后来发现outerHTML在各个浏览器下差异性太大了. IE6-7会对colgroup, dd, dt, li, options, p, td, tfoot, th, thead, tr元素自闭合 让我的htmlPaser跪掉 于是写了htmlfy,手动取每个元素nodeName, attrName, attrValue, nodeValue来构建outerHTML

第2阶段,将这个字符串进行parser,转换为虚拟DOM 这个阶段对input/textarea元素补上type属性, ms-\* 自定义元素补上ms-widget属性, 对table元素补上tbody, 在ms-for指令的元素两旁加上 `<!--ms-for-->`, `<!--ms-for-end-->` 占位符, 并将它们的之间的元素放到一个数组中(表明它们是循环区域) 并去掉所有只有空白的文本节点

第3个阶段,优化,对拥有 ms-\* 属性的虚拟DOM添加dynamic属性 表明它以后要保持其对应的真实节点,并对没有 ms-\* 属性的元素添加skipAttrs属性,表明以后不需要遍历其属性。 如果它的子孙没有 ms-\* 或插值表达式或ms-自定义元素,那么还加上skipContent, 表明以后不要遍历其孩子.

这三个属性,dynamic用于节点对齐算法,skipAttrs与skipContent用于diff算法

第4个阶段, 应用节点对齐算法, 将真实DOM中无用的空白节点移除,并插入占位符, 并将需要刷新的元素保持在以应的拥有dynamic属性的虚拟DOM中

第5个阶段,放进render方法中,render方法里面再调parseView,parseView会调每个指令的parse方法 将虚拟DOM树转换为一个\$render方法

第6个阶段,执行\$render方法,生成新的虚拟DOM,与最早的那个虚拟DOM树diff,一边diff一边更新真实DOM.

以后VM的属性发生变动,就直接执行第6个阶段.

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间： 2016-07-08 17:03:14



avalon的指令是一个非常重要的东西,它用来引入一些新的HTML语法,使元素拥有特定的行为。举例来说,静态的HTML不知道如何来创建和展现一个日期选择器控件。让HTML能识别这个语法,我们需要使用指令。指令通过某种方法来创建一个能够支持日期选择的元素。

指令一共拥有3种形式:

## 指令

avalon的指令是一个非常重要的东西,它用来引入一些新的HTML语法,使元素拥有特定的行为。举例来说,静态的HTML不知道如何来创建和展现一个日期选择器控件。让HTML能识别这个语法,我们需要使用指令。指令通过某种方法来创建一个能够支持日期选择的元素。

指令一共拥有3种形式

1. 插值表达式
2. 自定义标签
3. 绑定属性

其中 `绑定属性` 的种类是最多的,它们都位于元素节点中,以`ms-`开头或以`:`开头(avalon2.1.7新增)

绑定属性的属性名是以-分成几段 其中第二个就是指令的名字,如`ms-css`,`ms-attr`,`ms-html`,`ms-text`,`ms-on`都是来源于jQuery同名方法名,简单好记.

```
<p ms-on-click="@clickFn" ms-if="@toggle">{{@name}}</p>
```

# avalon 指令一览

## @作用域处理

ms-important  
ms-controller  
ms-skip

## @数据填充

ms-html  
ms-text  
{}}

## @样式

ms-css  
ms-visible

## @逻辑

<<!—ms-js:code—->>  
ms-for  
ms-if

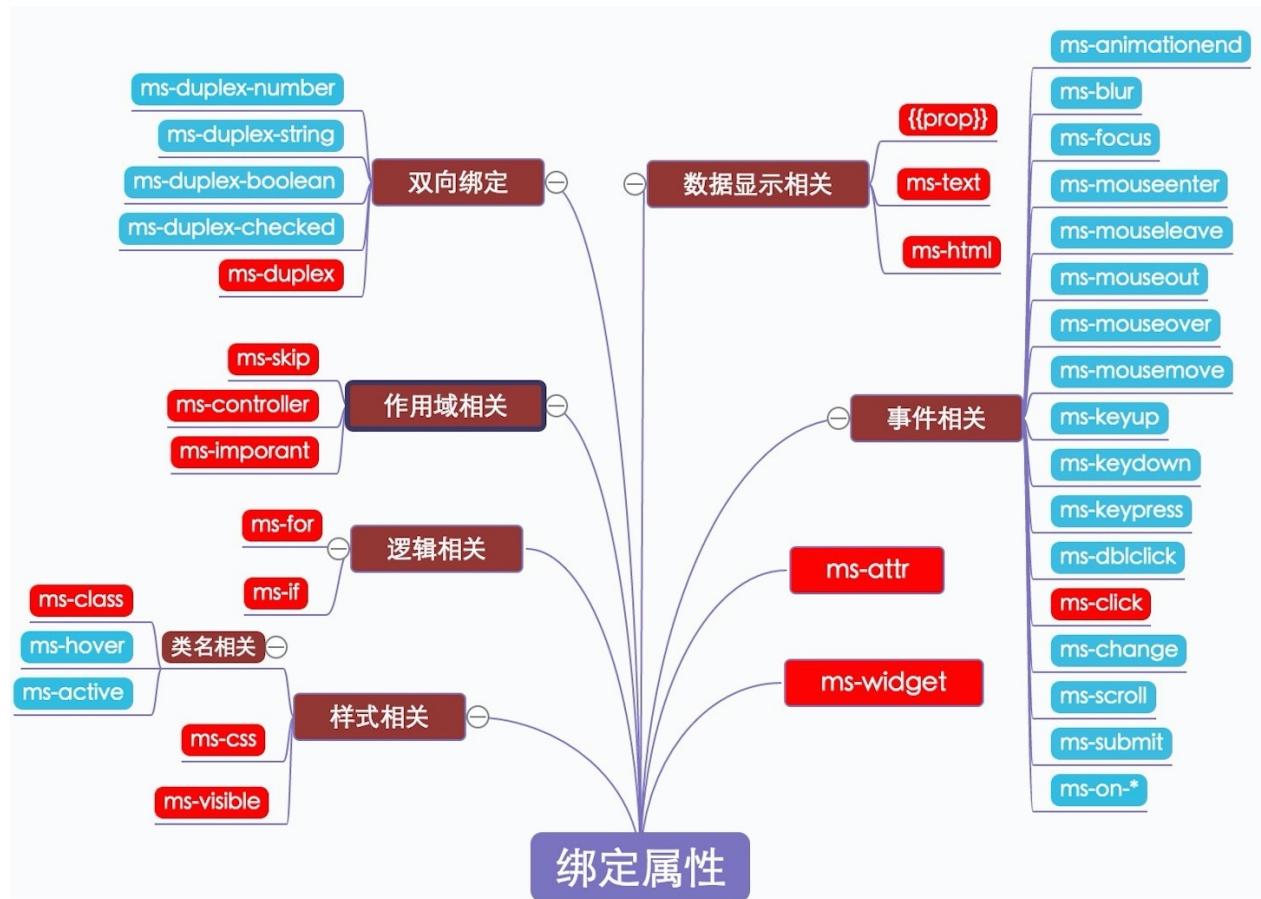
## @数据验证

ms-validate  
ms-rules  
@双工绑定  
ms-duplex

## @类名

ms-class  
ms-active  
ms-hover

ms-attr  
ms-on  
ms-effect  
ms-widget



与1.4,1.5相比, 2.0是移除了ms-repeat, ms-each, ms-with, ms-include, ms-include-src, ms-data, ms-scan, ms-if-loop指令.

Copyright © 司徒正美 2013 – 2016 all right reserved, powered by Gitbook  
该文件修订时间： 2016-07-14 16:55:21

# 插值表达式

位于文本节点中的双重花括号,当然这个可以配置.此指令其中文本ms-text指令的简单形式.

```
<body ms-controller="test">
  <script>
    avalon.define({
      $id: 'test',
      aaa: 'aaa',
      bbb: 'bbb'
    })

  </script>
  <p>{{@aaa}}{{@bbb}} 这个性能差些</p>
  <p>{{@aaa+@bbb}} 这个性能好些</p>
  <p>{{@aaa+@bbb | uppercase}} 选择器必须放在表达值的后端</p>
</body>
```

插值表达式一般与[带格式化功能的过滤器](#)一起使用.

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间: 2016-07-25 17:25:33

# skip绑定

让avalon的扫描引擎跳过某一部分区域, 方便能原样输出

合理使用ms-skip能大大提高性能

```
<body :controller="test">
<script>
var vm = avalon.define({
  $id: "test",
  aaa: "XXXX"
  toggle: false
})
</script>
<div ms-skip='true' >{{@aaa}}</div>
<div>{{@aaa}}</div>
</body>
```

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间： 2016-08-17 10:56:05

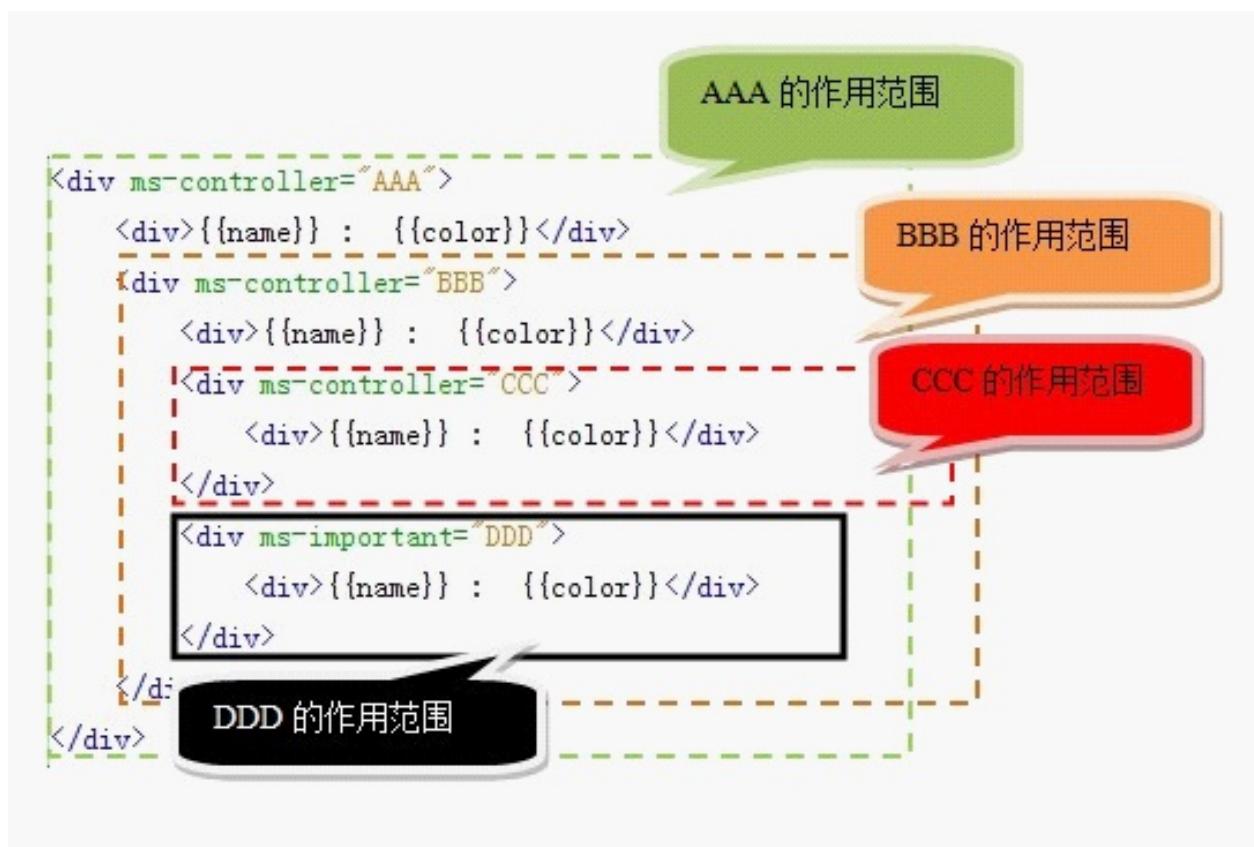
# controller绑定

这个指令是用于圈定某个VM的作用域范围(换言之,这个元素的outerHTML会被扫描编译,所有ms-\*及双花括号替换成vm中的内容),ms-controller的属性值只能是某个VM的\$id

ms-controller的元素节点下面的其他节点也可以使用ms-controller

每个VM的\$id可以在页面上出现一次,因此不要在ms-for内使用ms-controller.

当我们在某个指令上用@aaa时,它会先从其最近的ms-controller元素上找,找不到再往其更上方的ms-controller元素



```
<script>
avalon.define({
  $id: "AAA",
  name: "liger",
  color: "green"
});
avalon.define({
  $id: "BBB",
  name: "sphinx",
  color: "red"
});
avalon.define({
  $id: "CCC",
  name: "dragon" //不存在color
});

avalon.define({
  $id: "DDD",
  name: "sirenia" //不存在color
});

</script>
<div ms-controller="AAA">
  <div>{@name} : {@color}</div>
  <div ms-controller="BBB">
    <div>{@name} : {@color}</div>
    <div ms-controller="CCC">
      <div>{@name} : {@color}</div>
    </div>
    <div ms-important="DDD">
      <div>{@name} : {@color}</div>
    </div>
  </div>
</div>
```

当avalon的扫描引擎扫描到ms-controller/ms-important所在元素时, 会尝试移除ms-controller类名. 因此基于此特性, 我们可以在首页渲染页面时, 想挡住双花括号乱码问题, 可以尝试这样干(与avalon1有点不一样):

```
[ms-controller]{  
    visibility: hidden;  
}
```

Copyright © 司徒正美 2013 – 2016 all right reserved, powered by Gitbook  
该文件修订时间： 2016-09-05 01:52:52

## important绑定

这个指令是用于圈定某个VM的作用域范围(换言之,这个元素的outerHTML会被扫描编译,所有 `ms-*` 及双花括号替换成vm中的内容),`ms-important`的属性值只能是某个VM的\$id

`ms-important`的元素节点下面的其他节点也可以使用[ms-controller](#)或`ms-important`

与`ms-controller`不一同的是,当某个属性在`ms-important`的VM找不到时,就不会所上寻找

不要在`ms-for`内使用`ms-important`.

`ms-important`这特性有利协作开发,每个人的VM都不会影响其他人,并能大大提高性能

**ms-important只能用于ms-controller的元素里面**

```
<div ms-important='aaa'>
  <div ms-controller='ccc'>
    <div ms-important='ddd'>
      </div>
    </div>
    <div ms-controller='bbb'>
      </div>
    </div>
  </div>
</div>
```



## 属性绑定

属性绑定用于为元素节点添加一组属性，因此要求属性值为对象或数组形式。数组最后也会合并成一个对象。然后取此对象的键名为属性名，键值为属性值为元素添加属性。

如果键名如果为for, char这样的关键字，请务必在两边加上引号。

如果键名如果带横杠，请务必转换为驼峰风格或两边加上引号。

注意，**不能在ms-attr中设置style属性**

```
<p ms-attr="{style:'width:20px'}">这样写是错的，需要用ms-css指令!!</p>
```

示例：

```
<body ms-controller="test">
  <script>
    avalon.define({
      $id: 'test',
      obj: {title: '普通 ', align: 'left'},
      active: {title: '激活'},
      width: 111,
      height: 222,
      arr: [{img: 'aaa'}, {img: 'bbb'}, {img: 'ccc'}]
      path: '../aaa/image.jpg'
      toggle: false,
      array: [{width: 1}, {height: 2}]
    })
  </script>
  <span ms-attr="@obj">直接引用对象</span>
  <img ms-attr="{src: @path}" />
  <ul>
    <li ms-for="el in @arr"><a ms-attr="{href: 'http://www.ccc.xxx/
ddd/' + el.img}">下载</li>
  </ul>
  <span :attr="{width: @width, height: @height}">使用对象字面量</span>
  <span :attr="@array">直接引用数组</span>
  <span :attr="[@obj1, @toggle && @active ]" :click="@toggle = !@to
tgle">选择性添加多余属性或重写已有属性</span>
</body>
```

# 样式绑定

CSS绑定用于为元素节点添加一组样式, 因此要求属性值为对象或数组形式. 数组最后也会合并成一个对象. 然后取此对象的键名为样式名, 键值为样式值为元素添加样式

如果键名为表示长宽,字体大小这样的样式, 那么键值不需要加单位,会自动加上px

如果键名如果为float,请务必在两边加上引号

如果键名如果为font-size,请务必转换为驼峰风格或两边加上引号

```

<body ms-controller="test">
  <script>
    avalon.define({
      $id: 'test',
      obj: {backgroundColor: '#3bb0d0', width:300, height:50,
'text-align': 'center'},//属性名带-,必须用引号括起
      active: {color: 'red'},
      width: 300,
      height: 60,
      toggle: true,
      array: [{width:100},{height:50},{border: '1px solid #5c
b85c'}]
    })
  </script>
  <div ms-css="@obj">直接引用对象</div>
  <div :css="{width: @width, height: @height,background: 'pink'}">
使用对象字面量</div>
  <div :css="@array">直接引用数组</div>
  <div :css="[@obj, @toggle && @active ]" :click="@toggle = !@togg
le">选择性添加多余属性或重写已有属性</div>
</body>

```

需要注意的是 设置背景图片是比较复杂

```
<span :css="{background: 'url('+@imageUrl + ') no-repeat center cen  
ter;'}">图片</span>  
<span :css="{backgroundImage: 'url('+@imageUrl + ')'}">图片</span>
```

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook

该文件修订时间： 2016-08-21 03:57:38

# 文本绑定

文本绑定是最简单的绑定,它其实是双花括号插值表达式的一种形式

它要求VM对应的属性的类型为字符串, 数值及布尔, 如果是null, undefined将  
会被转换为空字符串

```
<span ms-text="@aaa">不使用过滤器</span>  
<span ms-text="@aaa | uppercase">使用过滤器</span>
```

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook

该文件修订时间： 2016-07-07 19:38:03

# HTML绑定

HTML绑定类似于文本绑定,能将一个元素清空,填上你需要的内容

它要求VM对应的属性的类型为字符串

```
<span ms-html="@aaa">不使用过滤器</span>
<span ms-html="@aaa | uppercase">使用过滤器</span>
```

我们可以通过ms-html异步加载大片内容。

```
<body :controller="test">
<script>
var vm = avalon.define({
  $id: "test",
  aaa: "loading..."
})
jQuery.ajax({
  url:'action.do',
  success: function(data){
    vm.aaa = data.html
  }
})
</script>
<div ms-html="@aaa"></div>
</body>
```

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook

该文件修订时间： 2016-07-25 17:51:25

# 类名绑定

属性绑定用于为元素节点添加几个类名, 因此要求属性值为字符串或字符串数组.

字符串形式下, 可以使用空格隔开多个类名

字符串数组形下, 可以在里面使用三元运算符或与或号

```
<body ms-controller="test">
  <script>
    avalon.define({
      $id: 'test',
      aaa: "aaa bbb ccc",
      bbb: 'ddd',
      ccc: [ 'xxx', 'yyy', 'zzz' ],
      ddd: 'eee',
      toggle: true,
      toggle2: false
    })

    </script>

    <span :class="@aaa">直接引用字符串</span>
    <span :class="@ccc">直接引用数组</span>
    <span :class="[@aaa, @bbb]">使用数组字面量</span>
    <span :class="['aaa', 'bbb', (@toggle? 'ccc': 'ddd')]">选择性添加
    类名</span>
    <span :class="[@toggle && 'aaa']">选择性添加类名</span>
    <span :class="[@ddd + 4]">动态生成类名</span>
  </body>
```



## active绑定

为元素添加:active效果,当元素被点击时添加几个类名, 鼠标弹起后则立即移除

```
<body ms-controller="test">
  <script>
    avalon.define({
      $id: 'test',
      aaa: "aaa bbb ccc",
      bbb: 'ddd',
      ccc: ['xxx', 'yyy', 'zzz'],
      ddd: 'eee',
      toggle: true,
      toggle2: false
    })
  </script>

  <span :active="@aaa">直接引用字符串</span>
  <span :active="@ccc">直接引用数组</span>
  <span :active="[@aaa, @bbb]">使用数组字面量</span>
  <span :active="['aaa', 'bbb', (@toggle? 'ccc': 'ddd')]">选择性添加类名</span>
  <span :active="[@toggle && 'aaa']">选择性添加类名</span>
  <span :active="[@ddd + 4]">动态生成类名</span>
</body>
```

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间: 2016-08-10 11:27:48

# hover绑定

用于类实现: hover伪类的效果, 当用户鼠标移动元素上方时添加几个类名, 鼠标移走时将刚才的类名移除

用法类似于[类名绑定](#)

```
<body ms-controller="test">
  <script>
    avalon.define({
      $id: 'test',
      aaa: "aaa bbb ccc",
      bbb: 'ddd',
      ccc: [ 'xxx', 'yyy', 'zzz' ],
      ddd: 'eee',
      toggle: true,
      toggle2: false
    })

    </script>

    <span :hover="@aaa">直接引用字符串</span>
    <span :hover="@ccc">直接引用数组</span>
    <span :hover="[@aaa, @bbb, 'kkk']">使用对象字面量</span>
    <span :hover="['aaa', 'bbb', (@toggle? 'ccc': 'ddd')]">选择性添加
      类名</span>
    <span :hover="[@toggle && 'aaa']">选择性添加类名</span>
    <span :hover="[@ddd + 4]">动态生成类名</span>
  </body>
```

## if绑定

通过属性值决定是否渲染目标元素, 为否时原位置上变成一个注释节点

avalon.\* 中ms-if-loop指令已经被废掉,请使用limitBy, selectBy, filterBy过滤器代替相应功能

```
<body :controller="test">
<script>
var vm = avalon.define({
  $id: "test",
  aaa: "这是被隐藏的内容"
  toggle: false
})
</script>
<p><button type="button" :click="@toggle = !@toggle">点我</span></p>

<div :if="@toggle">{@aaa}</div>
</body>
```

注意,有许多人喜欢用ms-if做非空处理,这是不对的,因此ms-if只是决定它是否插入DOM树与否,ms-if里面的 ms- 指令还是会执行.

```
avalon.define({
  $id: 'test',
  aaa: {}
})
```

```
<div ms-controller="test">
  <div ms-if="@aaa.bbb">
    {{@aaa.bbb.ccc}}这里肯定会出现错
  </div>
</div>
```

正确的做法是,当你知道这里面有非空判定,需要用方法包起来

```
avalon.define({
  $id: 'test',
  aaa: {},
  show: function(aaa, bbb, ccc){
    var obj = aaa[bbb]
    if(obj){
      return obj[ccc]
    }
    return ''
  }
})
```

```
<div ms-controller="test">
  <div ms-if="@aaa.bbb">
    {{@show(@aaa, 'bbb', 'ccc')}}
  </div>
</div>
```

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间: 2016-08-08 18:07:32

# 可见性绑定

这是通过修改元素的style.display值改变元素的可见性, 要求属性值对应一个布尔, 如果不是布尔, avalon会自动转换值为布尔。

```
<body :controller="test">
<script>
var vm = avalon.define({
  $id: "test",
  aaa: "这是被隐藏的内容"
  toggle: false
})
</script>
<p><button type="button" :click="@toggle = !@toggle">点我</span></p>

<div :visible="@toggle">{@aaa}</div>
</body>
```

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间: 2016-07-25 17:59:21

# 循环绑定

avalon2.0的ms-for绑定集齐了ms-repeat, ms-each, ms-with的所有功能, 并且更好用, 性能提升七八倍

ms-for可以同时循环对象与数组

```
<ul>
    <li ms-for="el in @aaa">{{el}}</li>
</ul>
```

现在采用类似angular的语法, in前面如果只有一个变量,那么它就是数组元素或对象的属性名

```
vm.aaa = ['aaa', 'bbb', 'ccc']
vm.bbb = {a: 1, b: 2, c: 3}
```

```
<ul>
    <li ms-for="(aaa, el) in @aaa">{{aaa}}-{{el}}</li>
</ul>
<ul>
    <li ms-for="(k, v) in @bbb">{{k}}-{{v}}</li>
</ul>
```

依次输出的LI元素内容为0-aaa,1-bbb,2-ccc, a-1,b-2,c-3

in 前面有两个变量, 它们需要放在小括号里,以逗号隔开, 那么分别代表数组有索引值与元素, 或对象的键名与键值, 这个与jQuery或avalon的each方法的回调参数一致。

小括号里面的变量是随便起的,主要能符合JS变量命名规范就行,当然,也不要与window, this这样变量冲突.

```
<li ms-for="($index, el) in @arr">{$index}-{$el}</li>
<li ms-for="($key, $val) in @obj">{$key}-{$val}</li>
```

写成这样,就与avalon1.\*很相像了

如果你想截取数组的一部分出来单独循环,可以用limitBy过滤器, 使用as来引用新数组

```
<ul>
  <li ms-for="el in @aaa | limitBy(10) as items">{$el}</li>
</ul>
```

上例是显示数组的前10个元素, 并且将这10个元素存放在items数组中, 以保存过滤或排序结果

使用注释节点实现循环,解决同时循环多个元素的问题

```

<!DOCTYPE html>
<html>
  <head>
    <title>TODO supply a title</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <script src="../dist/avalon.js"></script>
    <script>
      vm = avalon.define({
        $id: 'for4',
        arr: [1, 2, 3, 4]
      })
    </script>
  </head>
  <body>
    <div ms-controller='for4'>
      <!--ms-for: el in @arr-->
      <p>{{el}}</p>
      <p>{{el}}</p>
      <!--ms-for-end:-->
    </div>
  </body>
</html>

```

avalon 不需要像angular那样要求用户指定trace by或像react 那样使用key属性来提高性能,内部帮你搞定一切

ms-for还可以配套data-for-rendered回调,当列表渲染好时执行此方法

如果你只想循环输出数组的其中一部分,请使用filterBy,只想循环输出对象某些键值并设置默认值,则用selectBy. 不要在同一个元素上使用ms-for与ms-if,因为这样做会在页面上生成大量的注释节点,影响页面性能

可用于ms-for中的过滤器有limitBy, filterBy, selectby, orderBy

ms-for支持下面的元素节点继续使用ms-for,形成双重循环与多级循环,但要求双重循环对应的二维数组.几维循环对应几维数组

```
vm.array = [{arr: [111,222, 333]}, {arr: [111,222, 333]}, {arr: [111, 222, 333]}]  
<p>array的元素里面有子数组,形成2维数组</p>  
<ul>  
  <li ms-for="el in @array"><div ms-for='elem in el.arr'>{{elem}}</div></li>  
</ul>
```

### 如何双向绑定ms-for中生成的变量?

由于 循环生成的变量前面不带@, 因此就找不到其对应的属性,需要特别处理一下

```

<div ms-controller="test">
  <div ms-for="(key,el) in @styles">
    <label>{{ key }}::{{ el }}</label>
    <input type="text" ms-duplex="@styles[key]" >
    <!--不能ms-duplex="el"-->
  </div>
</div>

<script type="text/javascript">
  var root = avalon.define({
    $id: "test",
    styles: {
      width: 200,
      height: 200,
      borderWidth: 1,
      borderColor: "red",
      borderStyle: "solid",
      backgroundColor: "gray"
    }
  })
</script>

```

## 過濾器

### **limitBy(limit[,begin])**

limit

Type:Number

指定截取数组的数量，由数组首位开始截取至指定limit值结束，如limit值大于数组数量则返回整个数组

begin(可选值， 默认值为0)

Type:Number

指定截取开始位置，若为负值则由末尾向前取值。注意，**begin与limit的关系不是start与length的关系，而是start与end的关系。相当于slice(begin,limit)的截取**

如果需要在循环中访问截取数组的属性时，可通过el值直接访问，或通过as生成新引用数组。

```
<!DOCTYPE html>
<html>
    <head>
        <title>Test</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="../dist/avalon.js"></script>
        <script>
            vm = avalon.define({
                $id: 'test',
                arr:[{name:'a',age:11},{name:'b',age:3},{name:'c',age:22},{name:'d',age:33}]
            })
        </script>
    </head>
    <body>
        <div ms-controller='test'>
            <div ms-for="(index,el) in @arr | limitBy(4,2) as items">
                <label>{{items[index].name}}::{{el.age}}</label>
            </div>
        </div>
    </body>
</html>
```

## filterBy(search)

search

Type:Function,Number,String

当search值为数字或字符串时，返回循环中值包含search的选项。如search为函数，则通过传递的函数过滤。

需要注意，filterBy生成新引用数组的方式与limitBy不同

```
<div ms-for="(index,el) in @arrayOrObject as newArray | filterBy('name')">{{el}}::{{newArray.length}}</div>
```

### **selectby(array[,defaults])**

array

Type:Array

array为属性名集合，将循环中包含array属性名的属性筛选出来。

defaults(可选值)

Type:Object

当循环值中无array中的属性名，则可通过defaults为属性设置初始值。默认初始值为"

```

<!DOCTYPE html>
<html>
    <head>
        <title>Test</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0">
        <script src="../dist/avalon.js"></script>
        <script>
            vm = avalon.define({
                $id: 'test',
                arr:[{name: 'a', age:11}, {name: 'b', age:3}, {name: 'c', a
ge:22}, {name: 'd', age:33}]
            })
        </script>
    </head>
    <body>
        <div ms-controller='test' >
            <table border="1">
                <tr ms-for="(index,el) in @arr">
                    <td ms-for="value in el | selectBy(['name', 'age
', 'grade'], {grade: 'male'})">{{value}}</td>
                </tr>
            </table>
        </div>
    </body>
</html>

```

## orderBy(criteria[,reverse])

criteria

Type:String,Function

当criteria为字符串时，将根据指定的属性名排序。当criteria为方法时，则按照传入的方法排序。

reverse(可选值， 默认值1)

Type:Number

reverse为1时， 正序排列。反之为-1时， 倒序排列。

```
<!DOCTYPE HTML>
<html>
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset
=UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <script src="./dist/avalon.js" ></script>
        <script>
            if (!Date.now) {//fix 旧式IE
                Date.now = function() {
                    return new Date - 0;
                }
            }
            var model = avalon.define({
                $id: "test",
                selected: "name",
                options: ["name", "size", "date"],
                trend: 1,
                data: [
                    {name: "aaa", size: 213, date: Date.now() + 20},
                    ,
                    {name: "bbb", size: 4576, date: Date.now() - 4},
                    {name: "ccc", size: 563, date: Date.now() - 7},
                    {name: "eee", size: 3713, date: Date.now() + 9}
                    ,
                    {name: "555", size: 389, date: Date.now() - 20}
                ]
            })
        </script>
    </head>
    <body ms-controller="test">
        <div style="color:red">
```

```
<p>本例子用于显示如何做一个简单的表格排序</p>
</div>
<p>
    <select ms-duplex="@selected">
        <option ms-for="el in @options">{{el}}</option>
    </select>
    <select ms-duplex-number="@trend">
        <option value="1">up</option>
        <option value="-1">down</option>
    </select>
</p>
<table width="500px" border="1">
    <tbody >
        <tr ms-for="el in @data | orderBy(@selected, @trend
)">
            <td>{{el.name}}</td> <td>{{el.size}}</td> <td>{{el.date}}</td>
        </tr>
    </tbody>
</table>
</body>
</html>
```

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间： 2016-08-22 11:23:24

# 事件绑定

此绑定为元素添加交互功能，对用户行为作出响应。`ms-on-*="xxx"` 是其使用形式，`*` 代表`click`, `mouseover`, `touchstart`等事件名，只能与小写形式定义，`xxx`是事件回调本身，可以是方法名，或表达式。默认,事件回调的第一个参数是事件对象,并进行标准化处理.如果你是用 `ms-on-click="@fn(el, 1)"` 这样的传参方式，第一个传参被你占用，而你又想用事件对象，可以使用`$event`标识符，即 `ms-on-click="@fn(el, 1, $event)"` 那么第三个参数就是事件对象。

如果你想绑定多个点击事件,可以用 `ms-on-click-1="@fn(el)", ms-on-click-2="@fn2(el)", ms-on-click-3="@fn3(el)"` 来添加。

并且,avalon对常用的事件,还做了快捷处理,你可以省掉中间的on。

avalon默认对以下事件做快捷处理:

```
animationend、 blur、 change、 input、 click、 dblclick、 focus、 keydown、 keypress、 keyup、 mousedown、 mouseenter、 mouseleave、 mousemove、 mouseout、 mouseover、 mouseup、 scroll、 submit
```

此外,avalon2相对avalon1, 还做了以下强化:

以前 `ms-on-*` 的值只能是vm中的一个函数名 `ms-on-click="fnName"` , 现在其值可以是表达式,如 `ms-on-click="el.open = !el.open"` , 与原生的`onclick`定义方式更相近. 以前 `ms-on-*` 的函数,`this`是指向绑定事件的元素本身,现在`this`是指向`vm`, 元素本身可以直接从`e.target`中取得.

`ms-on-*` 会优先考虑使用事件代理方式绑定事件,将事件绑在根节点上!这会带来极大的性能优化! `ms-on-*` 的值转换为函数后,如果发现其内部不存在`ms-for`动态生成的变量,框架会将它们缓存起来! 添加了一系列针对事件的过

滤器 对按键进行限制的过滤器esc, tab, enter, space, del, up, left, right, down 对事件方法stopPropagation, preventDefault进行简化的过滤器stop, prevent

```
var vm = avalon.define({
    $id: "test",
    firstName: "司徒",
    array: ["aaa", "bbb", "ccc"],
    argsClick: function(e, a, b) {
        alert([] .slice .call(arguments) .join(" "))
    },
    loopClick: function(a, e) {
        alert(a + " " + e.type)
    },
    status: "",
    callback: function(e) {
        vm.status = e.type
    },
    field: "",
    check: function(e) {
        vm.field = e.target.value + " " + e.type
    },
    submit: function() {
        var data = vm.$model
        if (window.JSON) {
            setTimeout(function() {
                alert(JSON.stringify(data))
            })
        }
    }
})
```

```

<fieldset ms-controller="test">
    <legend>有关事件回调传参</legend>
    <div ms-mouseenter="@callback" ms-mouseleave="@callback">{{@sta
tus}}<br/>
        <input ms-on-input="@check"/>{{@field}}
    </div>
    <div ms-click="@argsClick($event, 100, @firstName)">点我</div>
    <div ms-for="el in @array" >
        <p ms-click="@loopClick(el, $event)">{{el}}</p>
    </div>
    <button ms-click="@submit" type="button">点我</button>
</fieldset>

```

绑定多个同种事件的例子：

```

var count = 0
var model = avalon.define({
    $id: "multi-click",
    str1: "1",
    str2: "2",
    str3: "3",
    click0: function() {
        model.str1 = "xxxxxxxx" + (count++)
    },
    click1: function() {
        model.str2 = "xxxxxxxx" + (count++)
    },
    click2: function() {
        model.str3 = "xxxxxxxx" + (count++)
    }
})

```

```

<fieldset>
    <legend>一个元素绑定多个同种事件的回调</legend>
    <div ms-controller="multi-click">
        <div ms-click="@click0" ms-click-1="@click1" ms-click-2="@c
lick2" >请点我</div>
        <div>{@str1}</div>
        <div>{@str2}</div>
        <div>{@str3}</div>
    </div>
</fieldset>

```

回调执行顺序的例子：

```

avalon.define({
    $id: "xxx",
    fn: function() {
        console.log("11111111")
    },
    fn1: function() {
        console.log("2222222")
    },
    fn2: function() {
        console.log("3333333")
    }
})

```

```

<div ms-controller="xxx"
    ms-on-mouseenter-3="@fn"
    ms-on-mouseenter-2="@fn1"
    ms-on-mouseenter-1="@fn2"
    style="width:100px;height:100px;background: red;">
</div>

```

avalon已经对ms-mouseenter, ms-mouseleave进行修复，可以在这里与这里了解这两个事件。到chrome30时，所有浏览器都原生支持这两个事件。

```
avalon.define({
    $id: "test",
    text: "",
    fn1: function (e) {
        this.text = e.target.className + " " + e.type
    },
    fn2: function (e) {
        this.text = e.target.className + " " + e.type
    }
})
```

```
.bbb{
    background: #1ba9ba;
    width:200px;
    height: 200px;
    padding:20px;
    box-sizing:content-box;
}
.ccc{
    background: #168795;
    width:160px;
    text-align: center;
    line-height: 160px;
    height: 160px;
    margin:20px;
    box-sizing:content-box;
}
```

```

<div class="aaa" ms-controller="test">
  <div class="bbb" ms-mouseenter="@fn1" ms-mouseleave="@fn2">
    <div class="ccc" >
      {{@text}}
    </div>
  </div>
</div>

```

最后是mousewheel事件的修改，主要问题是出现firefox上，它死活也不愿意支持mousewheel，在avalon里是用DOMMouseScroll或wheel实现模拟的。我们在事件对象通过wheelDelta属性是否为正数判定它在向上滚动。

```

avalon.define({
  $id: "event4",
  text: "",
  callback: function(e) {
    this.text = e.wheelDelta + " " + e.type
  }
})

```

```

<div ms-controller="event4">
  <div ms-on-mousewheel="@callback" id="aaa" style="background: #1ba9ba; width:200px; height: 200px;">
    {{@text}}
  </div>
</div>

```

此外avalon还对input, animationend事件进行修复，大家也可以直接用avalon.bind, avalon.fn.bind来绑定这些事件。但建议都用ms-on绑定来处理。

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间：2016-07-08 19:45:32



# 双工绑定

双工绑定是MVVM框架中最强大的指令.react推崇单向数据流,没有双工绑定,那么需要redux等额外的库来实现相同的功能.

双工绑定只要用于表单元素上.或当一个div设置了contenteditable为true,也可以用ms-duplex指令.

## 各个表单元素的用法

```
<body ms-controller="test">
  <script>
    avalon.define({
      $id: 'test',
      aaa: 'aaa',
      bbb: 'bbb',
      ccc: 'ccc'
    })
  </script>

  <input ms-duplex="@aaa"/>{@aaa}
  <input ms-duplex="@bbb" type="password"/>{@bbb}
  <textarea ms-duplex="@ccc" /></textarea>{@ccc}
</body>
```

上面有三个控件, text, password, textarea它们都是属于输入型控件, 只要每为控件敲入一个字符, 后面的文本都会立即变化.那是因为它们默认是绑定oninput事件, 如果想控件全部输入好, 失去焦点时 才同步, 那么可以使  
用 change 过滤器

```
<input ms-duplex="@aaa | change"/>{@aaa}
```

如果你是做智能提示, 控件是绑定了一个AJAX请求与后端不断交互, 使用oninput事件会太频繁, 使用onchange事件会太迟钝,那么我们可以使用 debounce 过滤器

```
<input ms-duplex="@aaa | debounce(300)"/>{{@aaa}}
```

300ms同步一次.

另外,可编辑元素的用法与过滤器与上面三种控件一样.

```
<div contenteditable="true" ms-duplex="@aaa | debounce(300)"/></div>  
<p>{{@aaa}}</p>
```

这两个过滤器只能适用于上面的情况.

此外, 控件还有许多种, 像checkbox, radio,它们的同步机制也不一样.

```
<body ms-controller="test">
  <script>
    avalon.define({
      $id: 'test',
      aaa: '33',
      bbb: ['22']
    })
  </script>

  <input type="radio" value="11" ms-duplex="@aaa"/>
  <input type="radio" value="22" ms-duplex="@aaa"/>
  <input type="radio" value="33" ms-duplex="@aaa"/>
  <input type="checkbox" value="11" ms-duplex="@bbb"/>
  <input type="checkbox" value="22" ms-duplex="@bbb"/>
  <input type="checkbox" value="33" ms-duplex="@bbb"/>
  <p>radio: {{@aaa}}; checkbox:{{@bbb}}</p>
</body>
```

checkbox与radio是一点击就会更新.radio要求在vm中为一个简单数据类型数据,字符串,数字或布尔. 而checkbox则要求是一个数组. 并且在最开始时,ms-duplex会令radio钩上其value值等vm属性的控件, checkbox则可以勾选多个. 如此一来,vm中的属性些总是等于radio与checkbox的属性值. 但我们也可以让vm的属性值等于此控件的勾选状态,这时需要用上 `ms-duplex-checked` 转换器.

```

<body ms-controller="test">
  <script>
    avalon.define({
      $id: 'test',
      aaa: false,
      bbb: false
    })
  </script>
  <input type="radio" ms-duplex-checked="@aaa"/>
  <input type="checkbox" ms-duplex-checked="@bbb"/>
  <p>radio: {{@aaa}}; checkbox:{{@bbb}}</p>
</body>

```

最后表单元素还有select控件,它根据其multiple属性分为单选下拉框与复选下拉框, 其在vm中的值与radio,checkbox一样.即单选时,必须是一个简单数据类型, 复选时为一个数组. 在最开始时, 当option元素的value值或innerText(不在value值)与数据相同,它们就会被选上.

```

<body ms-controller="test">
  <script>
    avalon.define({
      $id: 'test',
      aaa: 'bbb'
      bbb: ['bbb', 'ccc'],
    })
  </script>
  <select :duplex="@aaa"><option>aaa</option><option>bbb</option><option>ccc</option></select>
  <select multiple="true" :duplex="@bbb"><option>aaa</option><option>bbb</option><option>ccc</option></select>
</body>

```

控件	触发时机	数据
text,password,textarea及可编辑元表	oninput, onchange, debounce	简单数据
radio,checkbox	onclick	简单数据或数组
select	onchange	简单数据或数组

## 数据转换

上面我们已经提到一个数据转换器ms-duplex-checked了.那只能用于checkbox与radio.

为什么会有这种东西呢?因为无论我们原来的数据类型是什么,跑到表单中都会变成字符串,然后我们通过事件取出来 它们也是字符串,不会主动变回 原来的类型 .我们需要一种机制保持数据原来的类型,这就是数据转换器.

avalon内置了4种过滤器

1. ms-duplex-string="@aaa"
2. ms-duplex-number="@aaa"
3. ms-duplex-boolean="@aaa"
4. ms-duplex-checked="@aaa"

前三个是将元素的value值转换成string, number, boolean (只有为'false'时转换为false)

最后是根据当前元素 (它只能是radio或checkbox) 的checked属性值转换为vm对应属性的值。

它们都是放在属性名上。当数据从元素节点往vmodel同步时, 转换成预期的数据。

```
<input value="11" ms-duplex-number="@aaa" />
```

## 数据格式化

一般来说,数据格式化是由过滤器实现的,如

```
<input value="11" ms-duplex="@aaa | uppercase"/>
```

但这里有一个隐患,可能导致死循环,因此建议放在事件回调中实现.

```
<body ms-controller="test">
  <script>
    var vm = avalon.define({
      $id: 'test',
      aaa: '111',
      bbb: '222',
      format1: function(e){//只能输入数字
        vm.aaa = e.target.value.replace(/\D/g, '')
      },
      format2: function(e){//只能输入数字
        vm.bbb = avalon.filter.date(e.target.value, 'yyyy-MM-dd')
      }
    })

    </script>

    <input :duplex="@aaa" :on-input="@format1"/>{{@aaa}}
    <input :duplex="@bbb" :on-change="@format2"/>{{@bbb}}
  </body>
```

数据格式化是放在属性值时,以过滤器形式存在,如

```
ms-duplex='@aaa | uppercase'
ms-duplex='@aaa | date('yyyy:MM:dd')'
```

## 数据验证

这必须在所有表单元素的上方form元素加上ms-validate指令, 当前元素加上ms-rules才会生效

```
<form ms-validate="@validation">
<input ms-duplex='@aaa'
       ms-rules='require,email,maxlength'
       data-maxlength='4'
       data-maxlength-message='太长了' >
</form>
```

详见[ms-rules](#)指令

## 同步后的回调

ms-duplex还有一个回调, data-duplex-changed, 用于与事件绑定一样, 默认第一个参数为事件对象。如果传入多个参数, 那么使用\$event为事件对象占位。

```
<input value="11"  ms-duplex-number="@aaa" data-duplex-changed="@fn"
/>
```

## 示例

现在我们来一些实际的例子!

## 全选与非全选

```

var vm = avalon.define({
  $id: "duplex1",
  data: [{checked: false}, {checked: false}, {checked: false}],
  allchecked: false,
  checkAll: function (e) {
    var checked = e.target.checked
    vm.data.forEach(function (el) {
      el.checked = checked
    })
  },
  checkOne: function (e) {
    var checked = e.target.checked
    if (checked === false) {
      vm.allchecked = false
    } else {//avalon已经为数组添加了ecma262v5的一些新方法
      vm.allchecked = vm.data.every(function (el) {
        return el.checked
      })
    }
  }
})

<table ms-controller=" duplex1" border="1">
  <tr>
    <td><input type="checkbox"
      ms-duplex-checked="@allchecked"
      data-duplex-changed="@checkAll"/>全选</td>
  </tr>
  <tr ms-for="($index, el) in @data">
    <td><input type="checkbox" ms-duplex-checked="el.checked" d
ata-duplex-changed="@checkOne" />{$index}::{{el.checked}}</td>
  </tr>
</table>

```

我们仔细分析其源码，`allchecked`是用来控制最上面的复选框的打勾情况，数组中的`checked`是用来控制下面每个复选框的下勾情况。由于是使用`ms-duplex`，因此会监听用户行为，当复选框的状态发生改变时，就会触发`data-duplex-changed`回调，将当前值传给回调。但这里我们不需要用它的`value`值，只用它的`checked`值。

最上面的复选框对应的回调是`checkAll`，它是用来更新数组的每个元素的`checked`属性，因此一个`forEach`循环赋值就是。

下面的复选框对应的`checkOne`，它们是用来同步最上面的复选框，只要它们有一个为`false`上面的复选框就不能打勾，当它们被打勾了，它们就得循环整个数组，检查是否所有元素都为`true`，是才给上面的`checkall`属性置为`true`。

现在我们学了循环指令，结合它来做一个表格看看。现在有了强大无比的`orderBy`, `limitBy`, `filterBy`, `selectBy`。我们做高性能的大表格是得心应手的！

```

if (!Date.now) {//fix 旧式IE
    Date.now = function() {
        return new Date - 0;
    }
}
avalon.define({
    $id: "duplex2",
    selected: "name",
    options: ["name", "size", "date"],
    trend: 1,
    data: [
        {name: "aaa", size: 213, date: Date.now() + 20},
        {name: "bbb", size: 4576, date: Date.now() - 4},
        {name: "ccc", size: 563, date: Date.now() - 7},
        {name: "eee", size: 3713, date: Date.now() + 9},
        {name: "555", size: 389, date: Date.now() - 20}
    ]
})

```

```

<div ms-controller=" duplex2">
<div style="color:red">
    <p>本例子用于显示如何做一个简单的表格排序</p>
</div>
<p>
    <select ms-duplex="@selected">
        <option ms-for="el in @options">{{el}}</option>
    </select>
    <select ms-duplex-number="@trend">
        <option value="1">up</option>
        <option value="-1">down</option>
    </select>
</p>
<table width="500px" border="1">
    <tbody >
        <tr ms-for="el in @data | orderBy(@selected, @trend)">
            <td>{{el.name}}</td> <td>{{el.size}}</td> <td>{{el.date
}}</td>
        </tr>
    </tbody>
</table>
</div>

```

我们再来一个文本域与下拉框的联动例子，它只用到ms-duplex，不过两个控件都是绑定同一个属性。

```

avalon.define({
    $id: "fruit",
    options: ["苹果", "香蕉", "桃子", "雪梨", "葡萄",
              "哈蜜瓜", "橙子", "火龙果", "荔枝", "黄皮"],
    selected: "桃子"
})

```

```
<div ms-controller=" fruit">
  <h3>文本域与下拉框的联动</h3>
  <input ms-duplex="@selected" />
  <select ms-duplex="@selected" >
    <option ms-for="el in @options" ms-attr="{value: el}" >
      {{el}}
    </option>
  </select>
</div>
```

## 下拉框三级联动

```

var map = {
    "中国": ["江南四大才子", "初唐四杰", "战国四君子"],
    "日本": ["日本武将", "日本城堡", "幕府时代"],
    "欧美": ["三大骑士团", "三大魔幻小说", "七大奇迹"],
    "江南四大才子": ["祝枝山", "文征明", "唐伯虎", "周文宾"],
    "初唐四杰": ["王勃", "杨炯", "卢照邻", "骆宾王"],
    "战国四君子": ["楚国春申君黄歇", "齐国孟尝君田文", "赵国平原君赵胜",
    "魏国信陵君魏无忌"],
    "日本武将": ["织田信长", "德川家康", "丰臣秀吉"],
    "日本城堡": ["安土城", "熊本城", "大坂城", "姬路城"],
    "幕府时代": ["镰仓", "室町", "丰臣", "江户"],
    "三大骑士团": ["圣殿骑士团", "医院骑士团", "条顿骑士团"],
    "三大魔幻小说": ["冰与火之歌", "时光之轮", "荆刺与白骨之王国"],
    "七大奇迹": ["埃及胡夫金字塔", "奥林匹亚宙斯巨像", "阿尔忒弥斯月神殿"
    , "摩索拉斯陵墓", "亚历山大港灯塔", "巴比伦空中花园", "罗德岛太阳神巨像"]
}
}

var vm = avalon.define({
    $id: 'linkage',
    first: ["中国", "日本", "欧美"],
    second: map['日本'].concat(),
    third: map['日本武将'].concat(),
    firstSelected: "日本",
    secondSelected: "日本武将",
    thirdSelected: "织田信长"
})

vm.$watch("firstSelected", function (a) {
    vm.second = map[a].concat()
    vm.secondSelected = vm.second[0]
})
vm.$watch("secondSelected", function (a) {
    vm.third = map[a].concat()
    vm.thirdSelected = vm.third[0]
})

```

```
<div ms-controller=" linkage">
<h3>下拉框三级联动</h3>
<select ms-duplex="@firstSelected" >
    <option ms-for="el in @first" ms-attr="{value:el}" >{{el}}</option>
</select>
<select ms-duplex="@secondSelected" >
    <option ms-for="el in @second" ms-attr="{value:el}" >{{el}}</option>
</select>
<select ms-duplex="@thirdSelected" >
    <option ms-for="el in @third" ms-attr="{value:el}" >{{el}}</option>
</select>
</div>
```

这里的技巧在于使用\$watch回调来同步下一级的数组与选中项。注意，使用concat方法来复制数组。

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间： 2016-07-25 20:27:42

# 验证规则绑定

avalon2砍掉了不少功能（如 `ms-include`, `ms-data`），腾出空间加了其他更有用的功能。数据验证就是其中之一。现在avalon2内置的验证指令是参考之前的oniui验证框架与jquery validation。

此指令只能用于添加`ms-duplex`指令的表单元素上。

avalon内置验证规则有

规则	描述
<code>required(true)</code>	必须输入的字段
<code>norequired(true)</code>	不是必填的字段
<code>email(true)</code>	必须输入正确格式的电子邮件
<code>url(true)</code>	必须输入正确格式的网址
<code>date(true或正则)</code>	必须输入正确格式的日期。默认是要求YYYY-MM-dd这样的格式
<code>number(true)</code>	必须输入合法的数字（负数，小数）
<code>digits(true)</code>	必须输入整数
<code>pattern(正则或true)</code>	让输入数据匹配给定的正则，如果没有指定，那么会到元素上找pattern属性转换成正则再匹配
<code>equalto(ID名)</code>	输入值必须和 #id 元素的value 相同
<code>maxlength: 5</code>	输入长度最多是 5 的字符串（汉字算一个字符）
<code>minlength: 10</code>	输入长度最小是 10 的字符串（汉字算一个字符）
<code>chs (true)</code>	要求输入全部是中文
<code>max:5</code>	输入值不能大于 5
<code>min:10</code>	输入值不能小于 10

这些验证规则要求使用`ms-rules`指令表示，要求为一个普通的JS对象。

此外要求验证框架能动起来，还必须在所有表单元素外包一个form元素，在form元素上加ms-validate指令。

```
var vm = avalon.define({
    $id: "validate1",
    aaa: "",
    bbb: '',
    ccc: '',
    validate: {
        onError: function (reasons) {
            reasons.forEach(function (reason) {
                console.log(reason.getMessage())
            })
        },
        onValidateAll: function (reasons) {
            if (reasons.length) {
                console.log('有表单没有通过')
            } else {
                console.log('全部通过')
            }
        }
    }
})
```

```
<div ms-controller="validate1">
  <form ms-validate="@validate">
    <p><input ms-duplex="@aaa" placeholder="username"
      ms-rules='{required:true,chs:true}'>{@aaa}</p>
    <p><input type="password" id="pw" placeholder="password"
      ms-rules='{required:true}'
      ms-duplex="@bbb" /></p>
    <p><input type="password"
      ms-rules='{required:true,equalto:'pw'}' placeholder="再填一次"
      ms-duplex="@ccc | change" /></p>
    <p><input type="submit" value="submit"/></p>
  </form>
</div>
```

因此，要运行起avalon2的内置验证框架，必须同时使用三个指令。`ms-validate`用于定义各种回调与全局的配置项（如什么时候进行验证）。`ms-duplex`用于将单个表单元素及相关信息组成一个Field对象，放到`ms-validator`指令的`fields`数组中。`ms-rules`用于定义验证规则。如果验证规则不满足你，你可以自行在**avalon.validators**对象上添加。

现在我们可以一下`ms-validate`的用法。其对应一个对象。

配置项	描述
fields	框架自行添加，用户不用写。为一个数组，放置ms-duplex生成的Field对象。
onSuccess	空函数，单个验证成功时触发，this指向被验证元素this指向被验证元素，传参为一个对象数组外加一个可能存在的事件对象。
onError	空函数，单个验证失败时触发，this与传参情况同上
onComplete	空函数，单个验证无论成功与否都触发，this与传参情况同上。
onValidateAll	空函数，整体验证后或调用了validateAll方法后触发；有了这东西你就不需要在form元素上ms-on-submit="submitForm"，直接将提交逻辑写在onValidateAll回调上
onReset	空函数，表单元素获取焦点时触发，this指向被验证元素，大家可以在里清理className、value
validateInBlur	true，在blur事件中进行验证,触发onSuccess, onError, onComplete回调
validateInKeyup	true, 在keyup事件中进行验证,触发onSuccess, onError, onComplete回调。当用户在ms-duplex中使用change debounce过滤器时会失效
validateAllInSubmit	true, 在submit事件中执行onValidateAll回调
resetInFocus	true, 在focus事件中执行onReset回调
deduplicateInValidateAll	false, 在validateAll回调中对reason数组根据元素节点进行去重

我们看一下如何自定义验证规则。

比如说我们有一个变态的需求，一个字段可以不填，但如果要填的话一定要是合法的数字，并且大于零。这就需要自定义规则了。

```
<!DOCTYPE html>
<html>
    <head>
        <title>ms-validate</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <script src="../dist/avalon.js"></script>
        <script>
            avalon.validators.aaa = {
                message: '必须数字并大于0',
                get: function (value, field, next) {
                    //想知道它们三个参数是什么,可以console.log(value,
                    field,next)
                    var ok = (value === '' || (Number(value) > 0))
                    next(ok)
                    return value
                }
            }
            var vm = avalon.define({
                $id: "test",
                aaa: '',
                validate: {
                    onError: function (reasons) {
                        reasons.forEach(function (reason) {
                            console.log(reason.getMessage())
                        })
                    },
                    onValidateAll: function (reasons) {
                        if (reasons.length) {
                            console.log('有表单没有通过')
                        } else {
                            console.log('全部通过')
                        }
                    }
                }
            })
        
    

```

```

        </script>
    </head>

    <body ms-controller="test">
        <form class="cmxform" ms-validate="@validate" >
            <fieldset>
                <legend>自定义规则</legend>
                <p>
                    <input
                        ms-duplex="@aaa"
                        ms-rules="{aaa: true}"
                    >
                </p>
            </fieldset>
            <p>
                <input class="submit" type="submit" value="提交" >
            </p>
        </fieldset>
    </form>
</body>
</html>

```

在上表还有一个没有提到的东西是如何显示错误信息，这个avalon不帮你处理。但提示信息会帮你拼好，如果你没有写，直接用验证规则的message，否则在元素上找data-message或data-required-message这样的属性。

```

<!DOCTYPE html>
<html>
    <head>
        <title>ms-validate</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <script src="../dist/avalon.js"></script>
        <script>
            var vm = avalon.define({
                $id: "test",

```

```

        rules:{required:true,email:true},
        email:'',
        validate: {
            onError: function (reasons) {
                reasons.forEach(function (reason) {
                    console.log(reason.getMessage())
                })
            },
            onValidateAll: function (reasons) {
                if (reasons.length) {
                    console.log('有表单没有通过')
                } else {
                    console.log('全部通过')
                }
            }
        }
    )
</script>
</head>

<body ms-controller="test">
<form class="cmxform" ms-validate="@validate" >
<fieldset>
    <legend>验证完整的表单</legend>
    <p>
        <label for="email">Email</label>
        <input id="email"
               name="email"
               type="email"
               ms-duplex="@email"
               ms-rules="@rules"
               data-required-message="请输入"
               data-email-message="请输入一个正确的邮箱"
               >
    </p>
    </fieldset>
    <p>
        <input class="submit" type="submit" value="提交"
    
```

```
>
    </p>
    </fieldset>
</form>
</body>
</html>
```

最后给一个复杂的例子：

```
<script>
    var vm = avalon.define({
        $id: "validate2",
        firstname: '司徒正美',
        lastname: '',
        username: '',
        password: '',
        confirm_password: '',
        email: '',
        agree: false,
        topic: [],
        toggle: false,
        validate: {
            onError: function (reasons) {
                reasons.forEach(function (reason) {
                    console.log(reason.getMessage())
                })
            },
            onValidateAll: function (reasons) {
                if (reasons.length) {
                    console.log('有表单没有通过')
                } else {
                    console.log('全部通过')
                }
            }
        }
    })

```

```

avalon.validators.checked = {
    message: '必须扣上',
    get: function (value, field, next) {
        next(value)
        return value
    }
}
avalon.validators.selecttwo = {
    message: '至少选择两个',
    get: function (value, field, next) {
        next(!vm.toggle || value.length >= 2)
        return value
    }
}
</script>

<div ms-controller="validate2">
    <form class="cmxform" ms-validate="@validate" >
        <fieldset>
            <legend>验证完整的表单</legend>
            <p>
                <label for="firstname">名字</label>
                <input id="firstname"
                    name="firstname"
                    ms-duplex="@firstname"
                    ms-rules="{required:true, pattern: /[\\u4e00-\\u9fa5a-zA-Z]{2-8}/i }"
                    data-required-message="必须是中文或字母(3-8个字符)" >
                </p>
                <p>
                    <label for="lastname">姓氏</label>
                    <input id="lastname"
                        name="lastname"
                        ms-duplex="@lastname"
                        ms-rules="{required:true}"
                        data-required-message="请输入您的姓氏"
                        >
                </p>
            </fieldset>
        </form>
    </div>

```

```
</p>
<p>
    <label for="username">用户名</label>
    <input id="username"
        name="username"
        ms-duplex="@username | change"
        ms-rules="{required:true, minlength:2}"
    >
</p>
<p>
    <label for="password">密码</label>
    <input id="password"
        name="password"
        type="password"
        ms-duplex="@password"
        ms-rules="{required:true,minlength:5}"
        data-required-message="请输入密码"
        data-required-message="密码长度不能小于 5
个字母"
    >
</p>
<p>
    <label for="confirm_password">验证密码</label>
    <input id="confirm_password"
        name="confirm_password"
        type="password"
        ms-duplex="@confirm_password | change"
        ms-rules="{required:true,equalto:'passwo
rd'}"
        data-equalto-message="两次密码输入不一致"
    >
</p>
<p>
    <label for="email">Email</label>
    <input id="email"
        name="email"
        type="email"
    >

```

```
        ms-duplex="@email"
        ms-rules="{email:true}"
        data-email-message="请输入一个正确的邮箱"
    >
</p>
<p>
    <label for="agree">请同意我们的声明</label>
    <input type="checkbox" class="checkbox" id="agree"
ee" name="agree"
        ms-duplex-checked="@agree"
        ms-rules="{checked:true}"
    >
</p>
<p>
    <label for="newsletter">我乐意接收新信息</label>
    <input type="checkbox" class="checkbox"
        id="newsletter"
        name="newsletter"
        ms-duplex-checked="@toggle"
    >
</p>
<fieldset id="newsletter_topics" ms-visible="@toggle">
    <legend>主题 (至少选择两个) </legend>
    <label for="topic_marketflash">
        <input type="checkbox"
            id="topic_marketflash"
            value="marketflash"
            name="topic[]"
            ms-duplex="@topic"
            ms-rules="{selecttwo:true}"
        >Marketflash
    </label>
    <label for="topic_fuzz">
        <input type="checkbox"
            id="topic_fuzz"
            value="fuzz"
            name="topic[]"
        >Fuzz
    </label>
</fieldset>
```

```
        ms-duplex="@topic"
        ms-rules="{selecttwo:true}"
        >Latest fuzz

    </label>
    <label for="topic_digester">
        <input type="checkbox"
            id="topic_digester"
            value="digester"
            name="topic[]"
            ms-duplex="@topic"
            ms-rules="{selecttwo:true}"
            >Mailing list digester
    </label>
    <label for="topic" class="error" style="display
:none">至少选择两个</label>
    </fieldset>
    <p>
        <input class="submit" type="submit" value="提交"
>
    </p>
</fieldset>
</form>
</div>
```

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间： 2016-09-02 14:29:32

# 验证绑定

avalon2新引入的指令，只能用于form元素上，用于为表单添加验证功能。它需要与ms-duplex, ms-rules指令一起配合使用。

此组件要依赖于Promise，显然Promise支持情况不太好，因此建议大家配合 es6 Promise库一起使用。

IE	Edge	Firefox	Chrome	Safari	Opera	iOS Safari
			29			
8	IE 8 Not supported		45			8.4
9	Browser usage Global: 0.77%	45	48	9	36	9.2
11	13	46	50	9.1	37	9.3
	14	47	51	TP	38	
		48	52		39	
		49	53			

ms-validate的值应该对应一个对象，由于对象比较大，建议写在vm，像下面那样：

```

vm.validate = {
    onValidateAll: function(reasons){
        //返回一个数组，如果长度为零说明没有错
    },
    onError: avalon.noop,//针对单个表单元素（使用了ms-duplex的input,
    select)
    onSuccess: avalon.noop,//针对单个表单元素
    onComplete: avalon.noop,//针对单个表单元素
    onReset: avalon.noop,//针对单个表单元素
    validateInBlur: true, // {Boolean} true, 在blur事件中进行验证,触发onSuccess, onError, onComplete回调
    validateInKeyup: true, // {Boolean} true, 在keyup事件中进行验证,触发onSuccess, onError, onComplete回调
    validateAllInSubmit: true, // {Boolean} true, 在submit事件中执行onValidateAll回调
    resetInFocus: true, // {Boolean} true, 在focus事件中执行onReset回调,
    deduplicateInValidateAll: false // {Boolean} false, 在validateAll回调中对reason数组根据元素节点进行去重
}

```

onError, onSuccess, onComplete, onValidateAll的第一个参数都是reasons对象, this指向被验证的元素, reason里面有你需要的各种东西.

```

var reason = {
    element: elem,
    data: field.data,
    message: elem.getAttribute("data-" + ruleName + "-message") ||
    elem.getAttribute("data-message") || hook.message,
    validateRule: ruleName,
    getMessage: getMessage
}

```

```
<body>
```

```
<script>
    var vm = avalon.define({
        $id: "test",
        action: '',
        firstName: '',
        add: function() {
            this.action = "add.php";
            this.validate.onManual();
        },
        update: function(){
            this.action = "update.php";
            this.validate.onManual();
        },
        validate: {
            validateAllInSubmit: false,
            onSuccess: function(reasons, event) {
                console.info('OK')
            },
            onError: function(reasons, event) {
                console.info('error')
            },
            onValidateAll: function(reasons, form) {
                if(reasons.length) {
                    // 表单有错误
                    console.info("error");
                    return false;
                } else {
                    // 验证成功
                    form.submit()
                }
            }
        }
    })
</script>

<div ms-controller="test">
    <form :validate="@validate" id="f1" :attr="{ action: @action }"
```

```
>
    <input type="text" placeholder="Insert your First Name" :du
plex="@firstName" :rules="{ required: true }" />
    <input type="submit" value="新建用户 action: 127.0.0.1/add"
:click="@add"/>
    <input type="submit" value="修改用户 action: 127.0.0.1/updat
e" :click="@update"/>
</form>

</div>
</body>
```

有关它的详细用法建议看[ms-rules](#)指令

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间： 2016-09-01 18:02:53

# 动画绑定

ms-effect拥有这三种绑定形式:

```
<p ms-effect="fade">fade为特效名</p>
<p ms-effect="@configObj, {is: 'fade'}">属性值为字面量,其中一个对象必须
包括is属性,这用于指定特效名</p>
<p ms-effect="{is:fade, stagger:300}">属性值为对象字面量, 里面拥有is
属性</p>
<p ms-effect="@fadeConfig">属性值为vm的对象,里面拥有is属性</p>
```

avalon2实际上没有实现完整的动画模块, 它只是对现有的CSS3动画或jquery animate再包装一层。

我们先说如何用CSS3为avalon实现动画效果。首先要使用avalon.effect注册一个特效。

```
avalon.effect(name, definition)
```

所有注册了的特效, 都可以在avalon.effects对象中找到。

css3动画要求我们至少添加4个类名。这个是从angular那里学过来的。因此如何你以前的项目是基于angular, 它那些CSS动画类可以原封不动地搬过来用。

```
avalon.effect('animate', {
    enterClass: 'animate-enter',
    enterActiveClass: 'animate-enter-active',
    leaveClass: 'animate-leave',
    leaveActiveClass: 'animate-leave-active',
})
```

当然，这些类名会默认帮你添加，因为它内部是这样实现的。

```
avalon.effect = function (name, definition) {
    avalon.effects[name] = definition || {}
    if (support.css) {
        if (!definition.enterClass) {
            definition.enterClass = name + '-enter'
        }
        if (!definition.enterActiveClass) {
            definition.enterActiveClass = definition.enterClass + '-active'
        }
        if (!definition.leaveClass) {
            definition.leaveClass = name + '-leave'
        }
        if (!definition.leaveActiveClass) {
            definition.leaveActiveClass = definition.leaveClass + '-active'
        }
    }
    if (!definition.action) {
        definition.action = 'enter'
    }
}
```

因此你可以简化成这样：

```
avalon.effect('animate', {})
avalon.effect('animate')
```

注册完，我们就需要在样式表中添加真正的CSS类。

```
.animate-enter, .animate-leave{
    width:100px;
    height:100px;
    background: #29b6f6;
    transition: width 2s;
    -moz-transition: width 2s; /* Firefox 4 */
    -webkit-transition: width 2s; /* Safari 和 Chrome */
    -o-transition: width 2s; /* Opera */
}
.animate-enter-active, .animate-leave{
    width:300px;
}
.animate-leave-active{
    width:100px;
}
```

我们还得定义一个vm，里面指明动画的动作（默认有三种方式，enter，leave，move）及动画结束时的回调（这是可选的）

```
var vm = avalon.define({
    $id: 'effect',
    aaa: "test",
    action: 'enter',
    enterCb: function(){
        avalon.log('动画完成')
    },
    leaveCb: function(){
        avalon.log('动画回到原点')
    }
})
```

然后页面上这样使用：

```

<div ms-controller='effect' >
  <div ms-effect="{is:'animate', action:@action, onEnterDone: @enterCb, onLeaveDone: @leaveCb}">
    {{@aaa}}
  </div>
  <button ms-click='@action = @action !== "leave" ? "leave": "enter"' type="button">click</button>
</div>

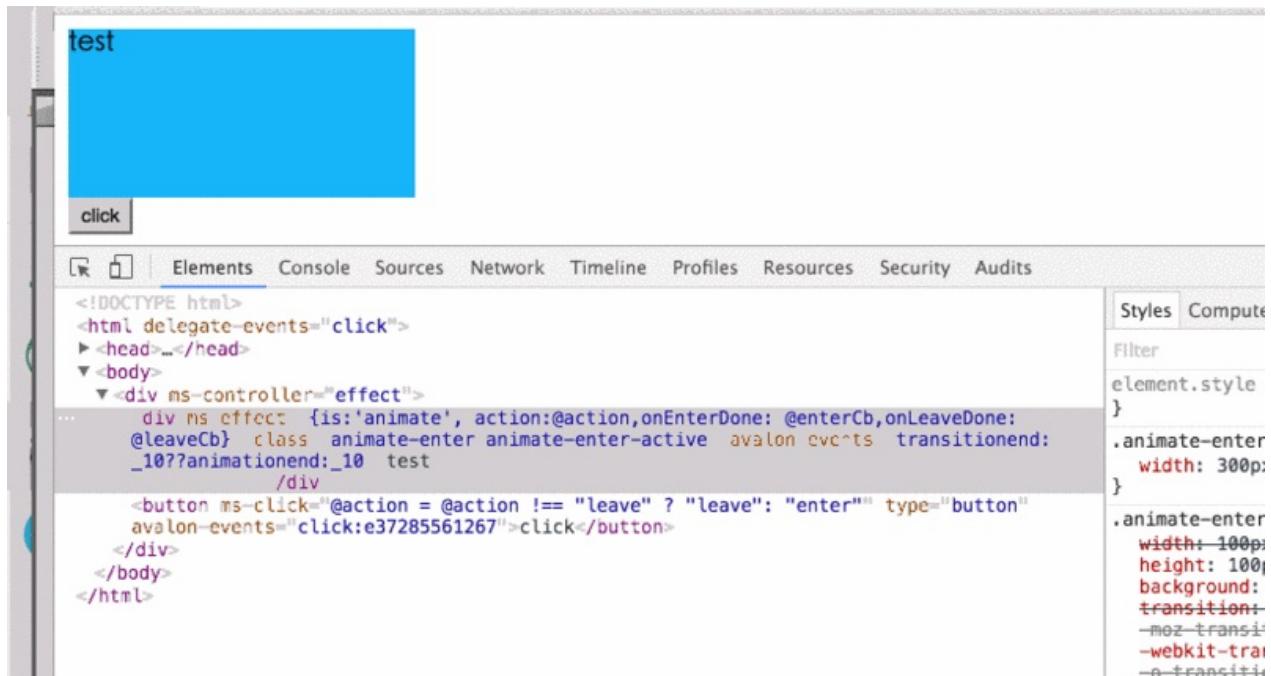
```

ms-effect的值为一个对象，其中is是必选。除了action, 还支持这么多种回调：

```

onEnterDone, onLeaveDone, onEnterAbort, onLeaveAbort, onBeforeEnter
, onBeforeLeave

```



如果使用JS实现，则是这样的：

```

<!DOCTYPE html>
<html>
  <head>
    <title>TODO supply a title</title>

```

```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<script src="../dist/avalon.js"></script>
<script src="//cdn.bootcss.com/jquery/3.0.0-beta1/jquery.js"
></script>
<style>
    .ani{
        width:100px;
        height:100px;
        background: #29b6f6;
    }
</style>
<script>
    avalon.effect('animate', {
        enter: function(el, done){
            $(el).animate({width: 300},1000,done)
        },
        leave: function(el, done){
            $(el).animate({width: 100},1000,done)
        }
    })
    var vm = avalon.define({
        $id: 'effect',
        aaa: "test",
        action: 'enter',
        enterCb: function(){
            avalon.log('动画完成')
        },
        leaveCb: function(){
            avalon.log('动画回到原点')
        }
    })
</script>
</head>
<body>
```

```

<div ms-controller='effect' >
    <div class='ani' ms-effect="{is:'animate', action:@action, onEnterDone: @enterCb, onLeaveDone: @leaveCb}">
        {{@aaa}}
    </div>
    <button ms-click='@action = @action !== "leave" ? "leave": "enter"' type="button">click</button>
</div>
</body>
</html>

```

## 一个CSS3位置效果

```

<!DOCTYPE html>
<html>
    <head>
        <title>TODO supply a title</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="../dist/avalon.js"></script>
        <script src="//cdn.bootcss.com/jquery/3.0.0-beta1/jquery.js"></script>
    <style>
        .ani{
            width:100px;
            height:100px;
            background: #ff6e6e;
        }
        .wave-enter, .wave-leave {
            -webkit-transition:all cubic-bezier(0.250, 0.460, 0.450, 0.940) 0.5s;
            -moz-transition:all cubic-bezier(0.250, 0.460, 0.450, 0.940) 0.5s;
            -o-transition:all cubic-bezier(0.250, 0.460, 0.450,

```

```
0.940) 0.5s;
        transition:all cubic-bezier(0.250, 0.460, 0.450, 0.
940) 0.5s;
    }

.wave-enter {
    position:absolute;
    left:45%;
}

.wave-enter-active {
    left:0;
}

.wave-leave {
    position:absolute;
    left:0;
}

.wave-leave-active {
    left:45%;
}

</style>
<script>
    avalon.effect('wave', {})
    var vm = avalon.define({
        $id: 'effect',
        action: 'enter',
        enterCb: function () {
            avalon.log('动画完成')
        },
        leaveCb: function () {
            avalon.log('动画回到原点')
        }
    })

```

```

        </script>
    </head>
    <body>
        <div ms-controller='effect' >
            <div class='ani' ms-effect="{is:'wave', action:@action,
onEnterDone: @enterCb, onLeaveDone: @leaveCb}">
                <button ms-click='@action = @action !== "leave" ? "leave": "enter"' type="button">click</button>
            </div>
        </div>
    </body>
</html>

```

```

<!DOCTYPE html>
<html lang="zh-CN">
    <head>
        <meta charset="utf-8">
        <meta name="format-detection" content="telephone=no">
        <meta name="viewport" content="width=device-width, initial-
scale=1.0, minimum-scale=1.0, maximum-scale=1.0, user-scalable=no">
        <style>
            .animate-enter, .animate-leave{
                width:100px;
                height:100px;
                background: #29b6f6;
                transition: width 2s;
                -moz-transition: width 2s; /* Firefox 4 */
                -webkit-transition: width 2s; /* Safari 和 Chrome */
            /
                -o-transition: width 2s; /* Opera */
            }
            .animate-enter-active, .animate-leave{
                width:300px;
            }
        </style>
    </head>
    <body>
        <div ms-controller='effect' >
            <div class='ani' ms-effect="{is:'wave', action:@action,
onEnterDone: @enterCb, onLeaveDone: @leaveCb}">
                <button ms-click='@action = @action !== "leave" ? "leave": "enter"' type="button">click</button>
            </div>
        </div>
    </body>
</html>

```

```
.animate-leave-active{
    width:100px;
}
</style>
<title></title>
</head>
<body ms-controller="body">
    <button ms-click="@show">show</button>
    <template :widget="{is:'ms-test',$id:'effxx'}"></template>
    <script src="../dist/avalon.js"></script>
    <script>
        avalon.effect("animate",{});
        avalon.component("ms-test",{
            template : "<div><div :for='el in @data' :effect='{
is : \"animate\\\",action : el.action}'></div></div>",
            defaults : {
                //这里不会报错
                data : [{action : 'enter'}],
                add : function(){
                    //push的时候报错
                    this.data.push({
                        action : "enter"
                    });
                }
            }
        });
        avalon.define({
            $id : "body",
            show : function(){
                avalon.vmodels.effxx.add();
            }
        });
    </script>
</body>
</html>
```

## ms-widget+ms-for+ms-if+ms-effect的动画效果

```
<!DOCTYPE html>
<html>
    <head>
        <title>ms-if</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width">
        <script src="../dist/avalon.js"></script>
        <script src="//cdn.bootcss.com/jquery/3.0.0-beta1/jquery.js"
></script>
        <style>
            .ani{
                width:100px;
                height:100px;
                background: #ff6e6e;
            }
        </style>
        <script >
            avalon.component('ms-button', {
                template: '<button type="button"><span><slot name="buttonText"></slot></span></button>',
                defaults: {
                    buttonText: "button"
                },
                soleSlot: 'buttonText'
            })
            avalon.effect('zoom', {
                enter: function (el, done) {

                    $(el).css({width: 0, height: 0}).animate({
                        width: 100, height: 100
                    }, 1000, done)
                },
                leave: function (el, done) {
                    $(el).animate({
                        width: 0, height: 0
                    }, 1000, done)
                }
            })
        </script>
    </head>
    <body>
        <div>
            <ms-if>
                <ms-button></ms-button>
            </ms-if>
        </div>
    </body>
</html>
```

```

        var vm = avalon.define({
            $id: "effect1",
            arr: [1,2,3],
            aaa: 222,
            toggle: true
        })

    </script>

</head>
<body ms-controller="test" >
    <div ms-for="el in @arr">
        <div class='ani'
            ms-attr="{eee:el}"
            ms-if="@toggle"
            ms-widget='{is:"ms-button"}'
            ms-effect='{is:'zoom'}'>{{@aaa}}::{{el}}</div>
    </div>
    <button ms-click="@toggle = !@toggle " >点我 </button >
</body>
</html>

```

## ms-for与stagger的动画效果

这次为了与angular一致，stagger改为一个小数，它会让当前元素延迟stagger秒执行。

```

<!DOCTYPE html>
<html>

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <script src="../dist/avalon.js"></script>
    <style>

```

```
.my-repeat-animation {
    width: 400px;
    height: 30px;
    -webkit-animation-duration: 1s;
    animation-duration: 1s;
}

.ng-enter {
    -webkit-animation-name: enter_animation;
    animation-name: enter_animation;
}
.ng-enter-stagger {
    animation-delay: 300ms;
    -webkit-animation-delay: 300ms;
}
.ng-leave {
    -webkit-animation-name: leave_animation;
    animation-name: leave_animation;
}

@keyframes enter_animation {
    0% {
        opacity: 0;
    }
    100% {
        opacity: 1;
    }
}

@keyframes leave_animation {
    from {
        opacity: 1;
    }
    to {
        opacity: 0;
    }
}
```

```

@-webkit-keyframes enter_animation {
    from {
        opacity: 0;
    }
    to {
        opacity: 1;
    }
}

@-webkit-keyframes leave_animation {
    from {
        opacity: 1;
    }
    to {
        opacity: 0;
    }
}
</style>
<script>
avalon.effect("my-repeat-animation", {
    enterClass: "ng-enter",
    leaveClass: "ng-leave"
})
var vm = avalon.define({
    $id: "test",
    array: [1, 2, 3, 4],
    getBg: function() {
        return '#' + Math.floor(Math.random() * 16777215).t
oString(16);
    },
    add: function() {
        vm.array.push(vm.array.length + 1)
        vm.array.push(vm.array.length + 1)
    }
})

```

```

        vm.array.push(vm.array.length + 1)
        vm.array.push(vm.array.length + 1)
    },
    value: ""
})
vm.$watch("value", function(a) {
    if (a) {
        vm.array.removeAll(function(el) {
            return el !== a
        })
    } else {
        if(vm.array.length < 12)
            vm.add()
    }
})
</script>
</head>

<body ms-controller="test">
    <button ms-click="@add">Add</button>
    <input placeholder="只保留" ms-duplex-number="@value" />
    <div class="my-repeat-animation" ms-for="item in @array"
        ms-css="{background:@getBg()}"
        ms-effect="{is:'my-repeat-animation',stagger:0.3}">
        {{item}}
    </div>
</body>

</html>

```

目前，avalon的ms-effect可以与ms-visible,ms-if,ms-repeat连用。ms-effect也可以单独或与其他指令使用，这时需要你指定action。

```
<div ms-effect="{is:'effectName', action: @action}">
```

该文件修订时间： 2016-08-25 11:29:22

## 组件绑定

此绑定只能应用于wbr, xmp, template, 及ms-开头的自定义标签。它将在原位置上转换成对应的组件的template的外观，加加上对应的数据与事件。

如果此组件没有注册（使用avalon.component进行定义），或其存在子组件，而某个子组件没有注册，都会导致初始化失败。在对应位置上变成一个注释节点。

有关组件的使用详看[组件](#)一章。

Copyright © 司徒正美 2013 – 2016 all right reserved, powered by Gitbook

该文件修订时间： 2016-07-08 11:01:34

# 自定义标签

以ms-开头的自定义标签,我们需要用avalon.component方法定义它,然后在里面使用ms-widget指令 为它添加更多行为.

avalon.component方法有两个参数,第一个标签名,必须以ms-开头;第二个是配置对象.

配置对象里也有4个配置项

1. template,自定义标签的outerHTML,它必须是用一个普通的HTML元素节点包起来,里面可以使用 ms-\* 等指令
2. defaults,用来定义这个组件的VM有什么属性与方法
3. soleSlot,表示自定义标签的innerHTML为一个默认的插入点 (或可理解为定义标签的innerHTML为当前组件某个属性的属性值),可选
4. getTemplate, 用来修改template, 依次传入vm与template, 返回新的模板. 可选

```
avalon.component('ms-pager', {
    template: '<div><input type="text" ms-duplex-number="@num"/><br/>
    <button type="button" ms-on-click="@onPlus">++++</button></div>',
    defaults: {
        num: 1,
        onPlus: function () {
            this.num++;
        }
    },
    getTemplate: function(vm, template){
        return template.replace('ms-on-click', 'ms-on-mousenter')
    }
});
```

注意,在avalon2中是严禁在绑定属性中混入插值表达式

自定义标签

---

该文件修订时间： 2016-07-08 00:55:28

# 组件

avalon拥有两大利器，强大的组件化功能以应对**复杂墙**问题，顶级的虚拟DOM机制来解决**性能墙**问题。

组件可谓是指令的集合，但  $1+1 > 2$  !

## 组件容器

组件容器是一个占位用的元素节点。当avalon扫描到此位置上时将它替换成组件。

在avalon2中有4类标签可以用作组件容器,分别是wbr, xmp, template, 及ms-\*开头的自定义标签.

其兼容性如下

元素	类型	说明
wbr	所有浏览器, 自闭合标签	需要使用ms-widget来指定组件类型
xmp	所有浏览器, 闭合标签	需要使用ms-widget来指定组件类型, 里面可以使用slot属性元素
template	IE9+及W3C浏览器, 闭合标签	需要使用ms-widget来指定组件类型, 里面可以使用slot属性元素
ms-*	IE9+及W3C浏览器, 闭合标签	可以省略ms-widget, 里面可以使用slot属性元素

闭合标签，比如 `<div></div><a></a>` 自闭合标签，比如 `<input><img><br><link>`

根据上表，如果要兼容IE6-8，那么只能使用wbr, xmp来做组件容器

如果不打算支持IE, 那么使用template元素性能最好

如果追求语义化, 只支持IE9+及其他现代浏览器,则使用ms-\*自定义标签.

xmp元素里面不能放xmp, template元素里面不能放template,这是html规范,就像script元素里面不能放script, textarea元素里面不能放textarea.但我们可以把这些元素里面直接放ms-\*自定义标签.

```
<xmp ms-widget="{is:'ms-dialog'}">
<ms-title slot="title">{{@title}}</ms-title>
</xmp>
```

## 声明组件

如果我们想在页面上使用组件,需要用组件容器与ms-widget指令声明一下.

```
<wbr ms-widget="{is:'ms-button'}" />
```

这就是声明使用一个按钮组件.

当然还有其他三种方式

```
<xmp ms-widget="{is:'ms-button'}" /></xmp>
<template ms-widget="{is:'ms-button'}" /></template>
<ms-button></ms-button>
```

自定义标签都是闭合标签,不能写成下面这样

```
<ms-button />
```

但是如果你的ms-button是放在xmp或template下面,则允许这样写.

```
<xmp ms-widget="{is:'ms-dialog'}">
<ms-title slot="title" />
<div slot="content">这是弹出层的内容</div>
<div slot="footer">
<ms-button :widget="@ok" /> <ms-button :widget="@ng" />
</div>
</xmp>
```

## 组件命名

由于组件名在高级浏览器中,可以作用自定义标签的标签名.而HTML标签在HTML5中有严格的规定,只能出现 \$,-,数字与英文单词,并且只能以字母开头,中间必须有'-'.

此外,为了方便avalon辨识这个标签名是否为一个数组,avalon强制规定以ms-开头,即

ms-button, ms-date-picker, ms-router-link

但是如果你不用自定义标签声明组件,使用ms-widget配置对象 来声明组件呢,你就可以突破部分限制,可以不以 ms- 开头

logger, date-picker, router-link

```
<wbr ms-widget="{is:'texer'}" />
```

## 配置对象

ms-widget 可以省略成 :widget ,它应该对应一个对象,即配置对象.

avalon2 的默认配置项比avalon1.5 少许多。所有组件通用的配置项

- is, 字符串, 指定组件的类型。如果你使用了自定义标签,从标签名就得知组件类型,则可以省略。

- `id`, 字符串, 指定组件vm的\$id, 这是可选项。如果该组件是位于SPA的子页里面, 也请务必指定此配置项, 能大大提高性能。
- `define`, 函数, 自己决定如何创建vm, 这是可选项。
- `onInit, onReady, onViewChange, onDispose`四大生命周期钩子。

注意,如果你在配置对象没有写`id`(或\$id, 兼并2.1.4之前的版本), 会在控制下看一个警告

其他组件需要传入的属性与方法, 也可以写配置对象中。为了方便传数据, `ms-widget`也像`ms-class`那样能对应一个数组。

```
<wbr ms-widget="[@allConfig, {id: 'xxx_+'+$index}]"/>
```

此外, 如果你的组件是位于SPA的子页面中, 或是由`ms-html`动态生成。

但组件对应的真实节点被移出DOM树时, 该组件会被销毁。为了进一步提高性能, 你可以在组件容器中定义一个`cached`属性, 其值为`true`, 它就能常驻内存。

```
<wbr cached="true" ms-widget="[@allConfig, {id: 'xxx_+'+$index}]"/>
```

用了`cached`时, 必须指定`id`配置项(2.1.5之前是`$id`)。

## 插槽元素与插卡元素

为了方便传入很长的HTML格式的参数, web components规范发明了`slot`机制。

avalon使用了一些黑魔法也让旧式IE浏览器支持它。

通俗来说, 我们用组件容器为组件占位, 我们也用**插槽容器**为**插卡元素**占位。

我们看一下 `ms-view` 组件的定义与声明:

## 组件

```
avalon.component('ms-view',{
  template:'<div class="view"><slot name="content" /></div>',
  defaults: {
    content: ""
  }
})
```

```
<div ms-controller='test'>
<ms-view>
<div slot="content">这是子视图的内容</div>
</ms-view>
</div>
```

<slot name="content" /> 叫做**插槽元素**,用来占位的,实际上它在内部会转换两个注释节点

```
<div class="view">
<!--slot:content-->
<!--slot-end:-->
</div>
```

组件容器中的带slot属性的元素, <div slot="content">这是子视图的内容</div>,就是**插卡元素**. 插卡元素最终会移动到组件对应的注释节点中去.

```
<div class="view">
<!--slot:content-->
<div slot="content">这是子视图的内容</div>
<!--slot-end:-->
</div>
```

我们可以对插卡元素使用除ms-if外的各种指令,如ms-for

```
<xmp :widget="{is:'ms-tabs',buttons: @buttons,tabs:@tabs}">

<div ms-for='(index,tab) in @tabs'
      ms-visible='index === @activeIndex '
      slot='tabs'
      >{{tab}}</div>
</xmp>
```

## soleSlot机制

中文叫单插槽或匿名插槽. 这是插槽机制的一个特例.

比如我们做一个按钮组件:

```
avalon.component('ms-button', {
    template: '<button type="button"><span><slot name="buttonText"
/></span></button>',
    defaults: {
        buttonText: "button"
    }
})
```

那么外面要这么使用

```
<ms-button><b slot="buttonText">xxx</b></ms-button>
```

事实上我们只想传入一个文本,不想传入b元素.这样定义太冗余了.

就像button标签,可以直接

```
<button>按钮</button>
```

于是就有单插槽机制. 它要求 组件 内部只有一个地方可以插东西, 并且将 组件容器 的所有孩子或文本都作为一个插卡.

## 组件

我们看一下新的定义与声明方式:

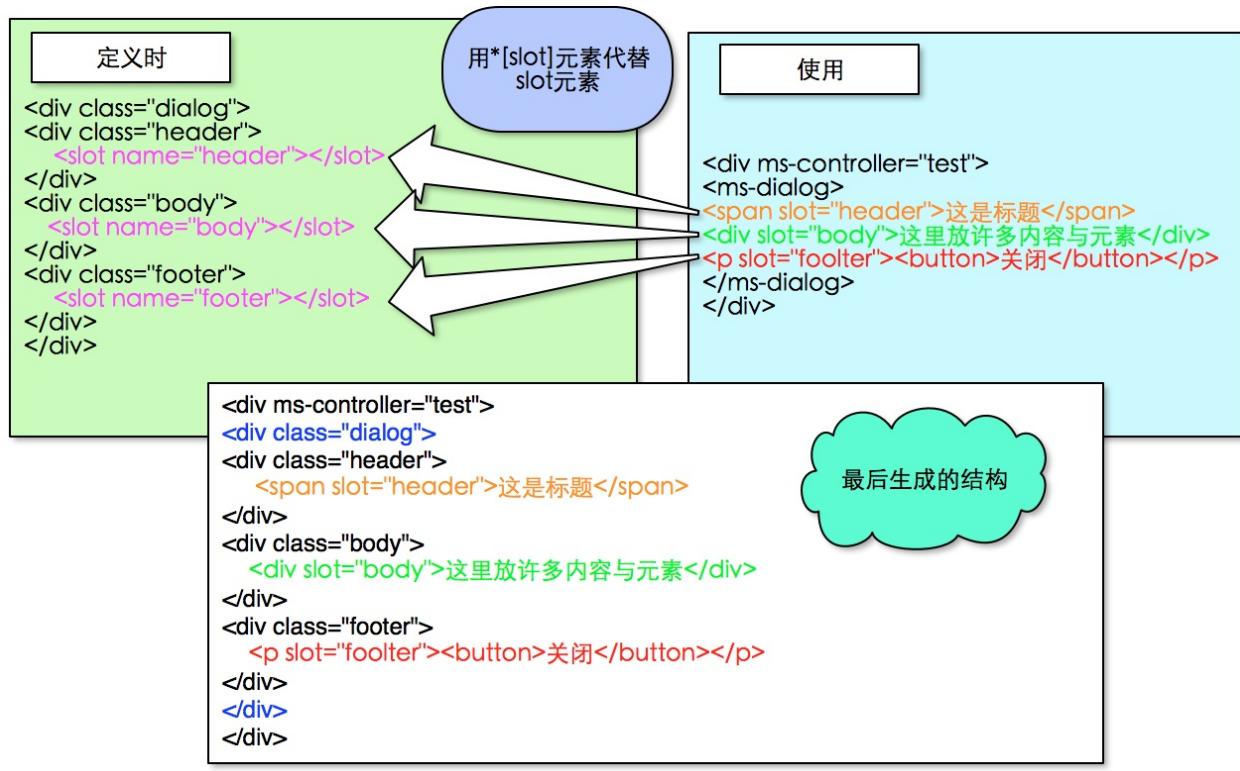
```
avalon.component('ms-button', {
    template: '<button type="button"><span><slot /></span></button>',
    defaults: {
        buttonText: "button"
    },
    soleSlot: 'buttonText'
})
```

```
<ms-button>xxx</ms-button>
```

avalon扫描后, 生成的组件是这个样子:

```
<button type="button">
<span>
<!--slot:buttonText-->
xxx
<!--slot-end:-->
</span>
</button>
```

插槽机制可以解决我们传入大片内容的难题, 多个slot元素拥有同一个name值。



## 组件定义

avalon定义组件时是使用**avalon.component**方法。

avalon.component方法有两个参数,第一个标签名,必须以ms-开头;第二个是配置对象.

配置对象里也有4个配置项

- **template**,自定义标签的outerHTML,它必须是用一个普通的HTML元素节点包起来,里面可以使用ms-\*等指令
- **defaults**,用来定义这个组件的VM有什么属性与方法
- **soleSlot**,表示组件只有一个插槽,会将组件容器的所有孩子都移到这里来 ,可选。
- **getTemplate**, 用来修改template, 依次传入vm与template, 返回新的模板, 可选。

```
avalon.component('ms-pager', {
    template: '<div><input type="text" ms-duplex-number="@num"/><
button type="button" ms-on-click="@onPlus">+++</button></div>',
    defaults: {
        num: 1,
        onPlus: function () {
            this.num++;
        }
    },
    getTemplate: function(vm, template){
        return template.replace('ms-on-click','ms-on-mousenter')
    }
});
```

## 渲染真相

```
var widgetVTree = widgetName(widgetOptions, slots, getTemplate(temp
late))
/*
widgetName: ms-widget中的is配置项或自定义标签的标签名
widgetOptions: ms-widget配置项
slots: 所有插卡元素组成的对象
getTemplate: 组件定义中getTemplate配置项
template: 组件定义中template配置项
widgetVTree: 组件的虚拟DOM,
*/

```

## 生命周期

avalon2组件拥有完善的生命周期钩子，方便大家做各种操作。

	<b>avalon2</b>	<b>web component</b>	<b>react</b>
初始化	onInit	createdCallback	getDefaultProj
插入DOM树	onReady	attachedCallback	componentDidC
视图变化	onViewChange	attributeChangedCallback	componentDic
移出DOM树	onDispose	detachedCallback	componentWill

onInit，这是组件的vm创建完毕就立即调用时，这时它对应的元素节点或虚拟DOM都不存在。只有当这个组件里面不存在子组件或子组件的构造器都加载回来，那么它才开始创建其虚拟DOM。否则原位置上被一个注释节点占着。

onReady，当其虚拟DOM构建完毕，它就生成其真实DOM，并用它插入到DOM树，替换掉那个注释节点。相当于其他框架的attachedCallback, inserted, componentDidMount.

onViewChange，当这个组件或其子孙节点的某些属性值或文本内容发生变化，就会触发它。它是比Web Component的attributeChangedCallback更加给力。

onDispose，当这个组件的元素被移出DOM树，就会执行此回调，它会移除相应的事件，数据与vmodel。

## 工作原理

avalon会先将组件容器转换为一个渲染函数,传入组件VM,成一个虚拟DOM(shellRoot)

再将组件定义中的template转换为第二个渲染函数,传入组件VM,成一个虚拟DOM(component)

然后将shellRoot的最外层元素的所有属性合并到component的最外层的元素上.

再将shellRoot中的插卡元素, 插入到component中的插槽元素的位置上.

将component变成真实DOM, 替换组件容器.

## 具体例子

请移步到[Github](#)

## 官方组件或推荐组件

### Promise

[mmPromise](#)

```
npm install avalon-promise
```

[bluebird](#)

[ypromise](#)

### ajax组件

重点推荐 [fetch-polyfill](#)

```
npm install fetch-polyfill2
```

[mmRequest](#)

[qwest.js](#)

[reqwest.js](#)

[ForbesLindesay/ajax](#)

## redux事件派发组件

[mmDux](#)

```
npm install mmDux
```

## 路由组件

[mmRouter](#)

[page.js](#)

## 动画组件

[tween.js](#)

```
npm install tween.js
```

## 弹窗组件

[ms-modal](#)

```
npm install ms-modal
```

## 分页组件

[ms-pager](#)

```
npm install ms-pager
```

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间： 2016-09-05 01:52:52

# 過濾器

## 格式化過濾器

用于处理数字或字符串，多用 `于{{}}` 或`ms-attr`或`ms-class`

注意: avalon的過濾器与ng的過濾器在传参上有点不一样,需要用()括起来

### uppercase

将字符串全部大写

```
vm.aaa = "aaa"  
  
<div>{@aaa | uppercase}</div>
```

### lowercase

将字符串全部小写

```
vm.aaa = "AAA"  
  
<div>{@aaa | lowercase}</div>
```

### truncate

对长字符串进行截短，有两个可选参数

`number`, 最后返回的字符串的长度, 已经将truncation的长度包含在内, 默认为30。 `truncation`, 告知用户它已经被截短的一个结尾标识, 默认为"..."

```
vm.aaa = "121323234324324"  
  
<div>{{@aaa | truncate(10, '...')}}</div>
```

## camelize

驼峰化处理， 如"aaa-bbb"变成"aaaBBB"

## escape

对类似于HTML格式的字符串进行转义， 如将<、>转换为<、>

## sanitize

对用户输入的字符串进行反XSS处理， 去掉onclick,

javascript:alert , <script> 等危险属性与标签。

## number

对需要处理的数字的整数部分插入千分号（每三个数字插入一个逗号）， 有一个参数fractionSize， 用于保留小数点的后几位。

fractionSize:小数部分的精度， 默认为3。

```
<!DOCTYPE html>
<html>
<head>
    <title>TODO supply a title</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width">
    <script src="avalon.js"></script>
    <script>
        avalon.define({
            $id: "number",
            aaa: 1234.56789
        })
    </script>
</head>

<body>
    <div ms-controller="number">
        <p>输入数字:<br/>
            <input ms-duplex="@aaa" type="text">
        </p>
        <p>不处理: {{@aaa}}</p>
        <p>不传参: {{@aaa|number}}</p>
        <p>不留小数: {{@aaa|number(0)}}</p>
        <p>负数:{{-@aaa|number(4)}}</p>
    </div>
</body>

</html>
```

## currency

用于格式化货币，类似于number过滤器（即插入千分号），但前面加了一个货币符号，默认使用人民币符号 \uFFE5

symbol, 货币符号，默认是 \uFFE5 fractionSize, 小数点后保留多少数，默认是2

## date

对日期进行格式化，`date(formats)`，目标可能是符合一定格式的字符串，数值，或Date对象。

```
<!DOCTYPE html>
<html>
<head>
    <title>TODO supply a title</title>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width">
    <script src="avalon.js"></script>
    <script>
        avalon.define({
            $id: 'testtest',
            name: "大跃进右",
            d1: new Date,
            d2: "2011/07/08",
            d3: "2011-07-08",
            d4: "01-01-2000",
            d5: "03 04,2000",
            d6: "3 4,2000",
            d7: 1373021259229,
            d8: "1373021259229",
            d9: "2014-12-07T22:50:58+08:00",
            d10: "\/Date(1373021259229)\/"
        })
    </script>
</head>
<body>
    <div ms-controller="testtest">
        <p>生成于{{ @d1 | date("yyyy MM dd:HH:mm:ss") }}</p>
        <p>生成于{{ @d2 | date("yyyy MM dd:HH:mm:ss") }}</p>
        <p>生成于{{ @d3 | date("yyyy MM dd:HH:mm:ss") }}</p>
        <p>生成于{{ @d4 | date("yyyy MM dd:HH:mm:ss") }}</p>
        <p>生成于{{ @d5 | date("yyyy MM dd:HH:mm:ss") }}</p>
```

```

<p>生成于{{ @d6 | date("yyyy MM dd") }}</p>
<p>生成于{{ @d7 | date("yyyy MM dd:HH:mm:ss") }}</p>
<p>生成于{{ @d8 | date("yyyy MM dd:HH:mm:ss") }}</p>
<p>生成于{{ @d9 | date("yyyy MM dd:HH:mm:ss") }} //这是ISO860
1的日期格式</p>
<p>生成于{{ @d10| date("yyyy MM dd:HH:mm:ss") }} //这是ASP.NE
T输出的JSON数据的日期格式</p>
</div>
</body>

</html>

```

标记	说明
yyyy	将当前的年份以4位数输出，如果那一年为300，则补足为0300
yy	将当前的年份截取最后两位数输出，如2014变成14，1999变成99，2001变成01
y	将当前的年份原样输出，如2014变成2014，399变成399，1变成1
MMMM	在中文中，MMMM与MMM是没有区别，都是"1月", "2月"……英语则为该月份的单词全拼
MMM	在中文中，MMMM与MMM是没有区别，都是"1月", "2月"……英语则为该月份的单词缩写(前三个字母)
MM	将月份以01-12的形式输出(即不到两位数，前面补0)
M	将月份以1-12的形式输出
dd	以日期以01-31的形式输出(即不到两位数，前面补0)
d	以日期以1-31的形式输出
EEEE	将当前天的星期几以“星期一”，“星期二”，“星期日”的形式输出，英语则Sunday-Saturday
EEE	将当前天的星期几以“周一”，“周二”，“周日”的形式输出，英语则Sun-Sat

HH	将当前小时数以00-23的形式输出
H	将当前小时数以0-23的形式输出
hh	将当前小时数以01-12的形式输出
h	将当前小时数以0-12的形式输出
mm	将当前分钟数以00-59的形式输出
m	将当前分钟数以0-59的形式输出
ss	将当前秒数以00-59的形式输出
s	将当前秒数以0-59的形式输出
a	将当前时间是以“上午”，“下午”的形式输出
Z	将当前时间的时区以-1200-+1200的形式输出
fullDate	相当于y年M月d日EEEE 2014年12月31日星期三
longDate	相当于y年M月d日EEEE 2014年12月31日
medium	相当于yyyy-M-d H:mm:ss 2014-12-31 19:02:44
mediumDate	相当于yyyy-M-d 2014-12-31
mediumTime	相当于H:mm:ss 19:02:44
short	相当于yy-M-d ah:mm 14-12-31 下午7:02
shortDate	相当于yy-M-d 14-12-31
shortTime	相当于ah:mm 下午7:02

## 循环过滤器

用于ms-for指令中

### limitBy

只能用于ms-for循环,对数组与对象都有效,限制输出到页面的个数,有两个参数

1. limit: 最大个数,必须是数字或字符,当个数超出数组长或键值对总数时,等于后面
2. begin: 开始循环的个数,可选,默认0

```
<script>
    avalon.define({
        $id: "limitBy",
        array: [1, 2, 3, 4, 5, 6],
        object: {a: 1, b: 2, c: 3, d: 4, e: 5},
        num: 3
    })
</script>
<div ms-controller='limitBy'>
    <select ms-duplex-number='@num'>
        <option>2</option>
        <option>3</option>
        <option>4</option>
        <option>5</option>
    </select>
    <ul>
        <li ms-for='el in @array | limitBy(@num)'>{{el}}</li>
    </ul>
    <ul>
        <li ms-for='el in @object | limitBy(@num)'>{{el}}</li>
    </ul>
</div>
```

## orderBy

只能用于ms-for循环,对数组与对象都有效,用于排序,有两个参数

1. key: 要排序的属性名
2. dir: -1或1,顺序或倒序,可选,默认1

```

<script>
avalon.define({
    $id: "orderBy",
    array: [{a: 1, b: 33}, {a: 2, b: 22}, {a: 3, b: 11}],
    order: 'a',
    dir: -1
})
</script>
<div ms-controller='orderBy'>
    <select ms-duplex='@order'>
        <option>a</option>
        <option>b</option>
    </select>
    <select ms-duplex-number='@dir'>
        <option>1</option>
        <option>-1</option>
    </select>
    <table border='1' width='200'>
        <tr ms-for="el in @array | orderBy(@order, @dir)">
            <td ms-for='elem in el'>{{elem}}</td>
        </tr>
    </table>
</div>

```

## filterBy

只能用于ms-for循环,对数组与对象都有效, 用于获取它们的某一子集, 有至少一个参数

search, 如果为函数时, 通过返回true决定成为子集的一部分; 如果是字符串或数字, 将转换成正则, 如果数组元素或对象键值匹配它, 则成为子集的一部分, 但如果是空字符串则返回原对象; 其他情况也返回原对象。 其他参数, 只有当search为函数时有效, 这时其参数依次是组元素或对象键值, 索引值, 多余的参数 此过滤多用于自动完成的模糊匹配!

```
<script>
```

## 過濾器

```
avalon.define({
    $id: "filterBy",
    array: ['aaaa', 'aab', 'acb', 'ccc', 'ddd'],
    object: {a: 'aaaa', b: 'aab', c: 'acb', d: 'ccc', e: 'ddd'}
},
    search: "a",
    searchFn: function (el, i) {
        return i > 2
    },
    searchFn2: function (el, i) {
        return el.length === 4
    },
    searchFn3: function (el, i) {
        return this.key === 'b' || this.key === '1'
    }
})
</script>
<div ms-controller='filterBy'>
    <select ms-duplex='@search'>
        <option>a</option>
        <option>b</option>
        <option>c</option>
    </select>
    <p><button ms-click="@search = @searchFn | prevent">变成过滤函数1</button></p>
    <p><button ms-click="@search = @searchFn2 | prevent">变成过滤函数2</button></p>
    <p><button ms-click="@search = @searchFn3 | prevent">变成过滤函数3</button></p>
    <ul>
        <li ms-for='el in @array | filterBy(@search)'>{{el}}</li>
    </ul>
    <ul>
        <li ms-for='el in @object | filterBy(@search)'>{{el}}</li>
    </ul>
</div>
```

,

```
<script>
    var vm = avalon.define({
        $id: 'test',
        arr: [{name: 'wanglin', age: 11},
               {name: 'lin', age: 3},
               {name: 'Hunt', age: 22},
               {name: 'Joe', age: 33}],
        fn: function(a){//过滤数组中 name属性值带lin中的元素
            return /lin/.test(a.name)
        }
    })
</script>
<div ms-controller='test' >
    <div ms-for="(index,el) in @arr as items | filterBy(@fn)" >
        {{el.name}} -- {{items.length}}
    </div>
</div>
`
```

## selectBy

只能用于ms-for循环,只对对象有效, 用于抽取目标对象的几个值,构成新数组返回.

1. array, 要抽取的属性名
2. defaults, 如果目标对象不存在这个属性,那么从这个默认对象中得到默认值,否则为空字符串, 可选 这个多用于表格, 每一列的对象可能存在属性顺序不一致或缺少的情况

```

<script>
    avalon.define({
        $id: "selectBy",
        obj: {a: 'aaa', b: 'bbb', c: 'ccc', d: 'ddd', e: 'eee'},
        grid: [{a: 1, b: 2, c: 3}, {c: 11, b: 22, a: 33}, {b: 23, a: 44}],
        defaults: {
            a: '@@@',
            b: '$$$',
            c: '###'
        }
    })
</script>
<div ms-controller='selectBy'>
    <ul>
        <li ms-for='el in @obj | selectBy(["c","a","b"])'>{{el}}</li>
    </ul>
    <table border='1' width='200'>
        <tr ms-for="tr in @grid">
            <td ms-for="td in tr | selectBy(['a','b','c'],@defaults)">{{td}}</td>
        </tr>
    </table>
</div>

```

## 事件過濾器

事件過濾器只要是对一些常用操作进行简化处理

对按键事件(keyup,keydown,keypress)到底按下了哪些功能键 或方向键进行友好的处理.许多人都记不清回车退格的keyCode是多少. 对阻止默认行为与防止冒泡进行封装

## **esc**

当用户按下esc键时,执行你的回调

## **tab**

当用户按下tab键时,执行你的回调

## **enter**

当用户按下enter键时,执行你的回调

## **space**

当用户按下space键时,执行你的回调

## **del**

当用户按下del键时,执行你的回调

## **up**

当用户按下up键时,执行你的回调

## **down**

当用户按下down键时,执行你的回调

## **left**

当用户按下left键时,执行你的回调

## **right**

当用户按下right键时,执行你的回调

## prevent

阻止默为行为,多用于form的submit事件防止页面跳转,相当于调用了event.preventDefault

```
<a href='./api.html' ms-click='@fn | prevent'>阻止跳转</a>
```

## stop

阻止事件冒泡,相当于调用了event.stopPropagation

页面的过滤器只能用于事件绑定

## 同步频率过滤器

这两个过滤器只用于ms-duplex

## change

在文本域或文本区使用ms-duplex时,默认是每输入一个字符就同步一次. 当我们想在失去焦点时才进行同步, 那么可以使用此过滤器

```
<input ms-duplex='@aaa | change'>{{@aaa}}
```

## debounce

当我们实现搜索框的自动完成时, 每输入一个字符可能就会向后台请求一次(请求关键字列表), 这样太频繁, 后端撑不住, 但使用change过滤器, 则又太慢了. 改为每隔几十毫秒请求一次就最好. 基于此常用需要开发出此过滤器. 拥有一个参数.

1. debounceTime: 数字, 不写默认是300,不能少于4,否则做无效处理

```
<input ms-duplex='@aaa | debounce(200)'>
```

## 编写过滤器

编写一个过滤器是非常简单的. 目前用户可编写的过滤器有两种, 不带参数的及带参数.

比方说uppercase,就是不带参数

```
vm.aaa = "aaa"  
  
<div>{{@aaa | uppercase}}</div>
```

输出:

```
<div>AAA</div>
```

那它是怎么实现的呢? 源码是这样的

```
avalon.filters.uppercase = function (str) {  
    return String(str).toUpperCase()  
}
```

过滤器总是把它前方的表达式生成的东西作为过滤器的第一个参数,然后返回一个值

同理lowercase的源码也很简单. 之所以用String,因为我们总想返回一个字符串

```
avalon.filters.lowercase = function (str) {  
    return String(str).toLowerCase()  
}
```

## 过滤器

那么我们自定义一个过滤器,就首先要看一下文档,注意不要与现有的过滤器同名. 比如我们定义一个haha的过滤器

```
<script>
  avalon.filters.haha = function(a){
    return a + 'haha'
  }
  avalon.define({
    $id: 'test',
    aaa: '111'
  })
</script>
<div ms-controller='tesst'>{@aaa | haha}</div>// 111haha
```

我们再看带参数的,带参数的必须写的括号,把第二个,第三个,放到里面

```
<div>{@aaa | truncate(10, '...')}</div>
```

truncate要传两个参数,那么看一下其源码是这样的:

```
avalon.filters.truncate = function (str, length, truncation) {
  //length, 新字符串长度, truncation, 新字符串的结尾的字段, 返回新字符串

  length = length || 30
  truncation = typeof truncation === "string" ? truncation : "..."

  return str.length > length ?
    str.slice(0, length - truncation.length) + truncation :
    String(str)
}
```

好了,我们看一下如何写一个带参数的过滤器,里面重复利用已有的过滤器

众所周知,ms-attr是返回一个对象. 我们只想对其中的一个字段进行格式化. 比如我们要处理title. 那么就起名为title.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <script type="text/javascript" src="../../dist/avalon.js"></script>
  <script>
    avalon.filters.title = function (obj, a, b) {
      var title = obj.title
      var newTitle = avalon.filters.truncate(title, a, b)
      obj.title = newTitle
      return obj
    }
    var vm = avalon.define({
      $id: 'test',
      el: '123456789qwert'
    })
  </script>
  </head>
  <body ms-controller="test">
    <div ms-attr="{title:@el} | title(10, '...')">333</div>
  </body>
</html>
```

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook

该文件修订时间： 2016-08-17 15:45:41

# 类型转换器

由于我们从表单元素拿到的所有东西都是字符串或字符串数组, 因此从 `avalon0.*` 起就提供了几个ms-duplex的辅助指令来实现数据类型转换功能

格式为 `ms-duplex-xxxx`

默认提供了4个数据转换器

- `ms-duplex-string`
- `ms-duplex-number`
- `ms-duplex-boolean`
- `ms-duplex-checked`

具体用法详见双工绑定

```
<input ms-duplex-number='@aaa'>{{@aaa}}
```

我们也可以自定义类型转换器, 直接在`avalon.parser`上添加

```
parsers: {
    number: function (a) {
        return a === '' ? '' : /\d\.$/.test(a) ? a : parseFloat(a)
    } || 0
},
string: function (a) {
    return a === null || a === void 0 ? '' : a + ''
},
boolean: function (a) {
    if(a === '')
        return a
    return a === 'true' || a == '1'
}
},
```

上面number, string, boolean就是ms-duplex-xxx的实际转换方法, a为元素的value值. ms-duplex-checked比较特殊它是使用checked属性,因此不在其列.

上面number, string, boolean就是ms-duplex-xxx的实际转换方法, a为元素的value值. ms-duplex-checked比较特殊它是使用checked属性,因此不在其列.

我们看一下转换器的用法。

```
<div ms-controller="test">
    <input type="checkbox" value="1" ms-duplex="@aaa">
    <input type="checkbox" value="2" ms-duplex="@aaa">
    <input type="checkbox" value="3" ms-duplex="@aaa">
    <p>{@aaa}</p>
</div>
<script>
var vm = avalon.define({
    $id: 'test',
    aaa: [1, 2]
})
</script>
```

如果不使用ms-duplex-number转换器，最开始时，第一个，第二个checkbox是能正确选中，但当我们点击第二个时，发现下面的文本没有变化。因此我们是尝试从[1,2]数组中移除"2"这个字符串，当然移除不了，没有效果。

当我们改成这样

```
<input type="checkbox" value="1" ms-duplex="@aaa">
<input type="checkbox" value="2" ms-duplex="@aaa">
<input type="checkbox" value="3" ms-duplex="@aaa">
```

点击第二个checkbox时，它会转换出input.value的“2”，然后再用avalon.parsers.number转换成数字，就能成功移除，于是文本就会变化。

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间：2016-07-11 20:38:09

# 表单验证

avalon内置了强大的表单验证功能，它需要结合[ms-duplex](#), [ms-validate](#), [ms-rules](#)这个三个指令一起使用。

- `ms-duplex`负责监控每个表单元素的输入。
- `ms-rules`负责对表单元素的值进行各种检测，包括非空验证，长度验证，格式匹配等等。
- `ms-validate`负责控制验证的时机，及针对每个表单元素的验证结果触发各种回调。

验证规则定义在`avalon.validators`对象中，为一个个带有`message`与`get`属性的对象。

具体用法详见[验证规则绑定](#)

```
avalon.shadowCopy(avalon.validators, {
    pattern: {
        message: '必须匹配{{pattern}}这样的格式',
        get: function (value, field, next) {
            var elem = field.dom
            var data = field.data
            if (!isRegExp(data.pattern)) {
                var h5pattern = elem.getAttribute("pattern")
                data.pattern = new RegExp('^(:' + h5pattern + ')$')
            }
            next(data.pattern.test(value))
            return value
        }
    },
    digits: {
        message: '必须整数',
        get: function (value, field, next) {//整数
            next(/^\-?\d+$/ . test(value))
            return value
        }
    }
})
```

### 手动调用验证并根据点击不同按钮提交不同网址的例子

```
<!DOCTYPE html>
<html>
    <head>
        <title>ms-validate</title>
        <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <script src="../dist/avalon.js"></script>
        <script>
            avalon.validators.gtOne = {
                message: '必须数字并大于1',
```

```

        get: function (value, field, next) {
            //想知道它们三个参数是什么,可以console.log(value,
            field,next)
            var ok = Number(value) > 1
            next(ok)
            return value
        }
    }
    var greasons = []
    var vm = avalon.define({
        $id: "test",
        aaa: '',
        url: 'javascript:void(0)',
        message: '',
        submit: function (url) { //submit是真正的验证方法,通
过点击时手动验证
            vm.validate.onManual()
            setTimeout(function () {
                if (greasons.length) {
                    var a = greasons.map(function (el) {
                        return '<p>' + el.getMessage() + '<
/p>'
                })
                vm.message = a.join('') //打印所有错误
                vm.url = 'javascript:void(0)'
            } else {
                greasons = []
                vm.message = ''
                vm.url = url
            }
        })
    },
    validate: {
        //禁止提交时自动验证
        validateAllInSubmit: false,
        //这个是用来占位的
        onManual: avalon.noop,
    }
}

```

## 表单验证

```
//这个转移到submit方法
onValidateAll: function (reasons) {
    greasons = reasons.concat()
}
})
</script>
</head>

<body ms-controller="test">
    <form class="cmxform" ms-validate="@validate" ms-attr='{action: @url}'>
        <fieldset>
            <legend>自定义规则</legend>
            <p>
                <input
                    ms-duplex="@aaa"
                    ms-rules="{required: true, number:true, gt0
ne: true}"
                />
            </p>
            <p>
                <input :click="@submit('/add.php')" type="submi
t" value="add"/>
                <input :click="@submit('/update.php')" type="su
bmit" value="update"/>
            </p>
            <p ms-html="@message" style="color: red"></p>
        </fieldset>
    </form>
</body>
</html>
```

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间： 2016-09-01 16:59:11



# 配置

avalon2遵循coc原则，配置项比较少。只有两个配置项。

双花括号也默认是python一些著名模板的界定符，为了防止冲突，我们有更换界定符的需求。这时我们可以这样做

```
avalon.config({
    interpolate: ['{%', '%}']
})
//或
avalon.config({
    interpolate: ['{?', '?}']
})
//或
avalon.config({
    interpolate: ['{&', '&}']
})
```

注意,左右界定符的长度应该为2,并且不能出现 < , >,因为出现源码括号在旧式IE下会变成注释节点 我们再看下一个有用的配置项, debug。avalon默认是在控制台下打印没有调试消息的, 上线时我们不愿用户看到它们, 可以这样关掉它们。

```
avalon.config({
    debug: false
})
```

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook

该文件修订时间： 2016-07-11 20:39:55

## 移动端支持

avalon在[这里](#)提供了几种手势模块, 来满足你的移动开发.

PC端与移动端的事件是完全不一样, 移动端的大多数事件都是基于ontouchxxx事件合成出来. 这些JS, 我们在使用时, 可以直接这样

```
var avalon = require('../avalon')
var tap = require('../tap')
```

使用时

```
<div ms-on-tap="@tapfn">点我</div>
```

当你使用tap模块, 请把tap与recognizer.js模块放在同一文件夹下.

drag, pinch, press, rotate, swipe, tap都依赖于recongizer模块.

具体教程可以看[这里](#)

**drag模块**: 在指定的dom区域内, 一个手指放下并移动事件, 即触屏中的拖动事件。这个事件在屏触开发中比较常用, 如: 左拖动、右拖动等. 如手要上使用QQ时向右滑动出现功能菜单的效果。具体事件有:

1. dragstart: 拖动开始
2. dragmove: 拖动过程
3. dragend: 拖动结束

**pinch模块**: 在指定的dom区域内, 两个手指相对（越来越近）移动或相向（越来越远）移动时事件。具体事件有:

1. pinchstart: 多点触控开始
2. pinch: 多点触控过程
3. pinchend: 多点触控结束
4. pinchin: 多点触控时两手指距离越来越近

5. pinchout: 多点触控时两手指距离越来越远

**press**模块: 在指定的dom区域内触屏版本的点击事件(longtap), 这个事件相当于PC端的click事件, 该不能包含任何的移动, 最小按压时间为500毫秒, 常用于我们在手机上用的“复制、粘贴”等功能。具体事件有:

1. longtap: 长按
2. doubletap: 双击

**rotate**模块: 在指定的dom区域内, 当单个手指围绕某个点转动时触发事件. 具体事件有:

1. rotatestart: 旋转开始
2. rotatemove: 旋转过程
3. rotateend: 旋转结束

**swipe**模块: 在指定的dom区域内, 一个手指快速的在触屏上滑动。即我们平时用到最多的滑动事件。具体事件有:

1. swipeleft: 向左滑动
2. swiperight: 向右滑动
3. swipeup: 向上滑动
4. swipedown: 向下滑动
5. swipe: 滑动(可以通过事件对象的direction属性知道当前滑动方向)

**tap**模块: 在指定的dom区域内, 一个手指轻拍或点击时触发该事件(类似PC端的click)。该事件最大点击时间为250毫秒, 如果超过250毫秒则按longtap事件进行处理。具体事件有:

1. tap: 轻拍

另外针对众多手机浏览器的奇怪设定,我们需要做一些hack

```
<!--开启对web app的支持-->
<meta name="apple-mobile-web-app-capable" content="yes" />
<!--主要是正对苹果手机将数字自动识别为号码-->
<meta name="format-detection" content="telephone=no" />
<!-- 忽略识别邮箱，主要是针对安卓手机会自动将符合邮箱格式的字符串识别为邮箱地址-->
<meta content="email=no" name="format-detection" />
<meta name="apple-itunes-app" content="app-id=myAppStoreID, affiliate-data=myAffiliateData, app-argument=myURL" />
<!-- 添加智能 App 广告条 Smart App Banner: 告诉浏览器这个网站对应的app， 并在页面上显示下载banner:https://developer.apple.com/library/ios/documentation/AppleApplications/Reference/SafariWebContent/PromotingAppswithAppBanners/PromotingAppswithAppBanners.html -->
<!-- 针对手持设备优化，主要是针对一些老的不识别viewport的浏览器，比如黑莓 -->
<meta name="HandheldFriendly" content="true">
<!-- 微软的老式浏览器 -->
<meta name="MobileOptimized" content="320">
<!-- uc强制竖屏 -->
<meta name="screen-orientation" content="portrait">
<!-- QQ强制竖屏 -->
<meta name="x5-orientation" content="portrait" />
<!-- UC强制全屏 -->
<meta name="full-screen" content="yes" />
<!-- QQ强制全屏 -->
<meta name="x5-fullscreen" content="true" />
<!-- UC应用模式 -->
<meta name="browsermode" content="application" />
<!-- QQ应用模式 -->
<meta name="x5-page-mode" content="app" />
<!-- windows phone 点击无高光 -->
<meta name="msapplication-tap-highlight" content="no" />
```

移动端支持

---

## 常见问题

### 如何隐藏首屏加载页面时出现的花括号

答 :在页面上添加一个样式

```
.ms-controller{  
    visibility: hidden  
}
```

使用在ms-controller, ms-important的元素上加上这个ms-controller类名

```
<div ms-controller="test" class="ms-controller">{{@aaa}}</div>
```

### IE6-8下为vm的数组重新赋给一个新数组失败?

#### 具体案例

```
vm.arr2 = vm.arr1 //报错
```

记住,任何时候,不能将vm中的数组或子对象取出来,再用它们赋给vm的某个数组或子对象,因为放在vm中的数组与子对象已经变成VM了,而VM重写VM不被允许的.

并且你要保证原数据不被污染,需要使用深拷贝.

```
vm.arr2 = avalon.mix(true, [], arr1)  
vm.obj2 = avalon.mix(true, {}, obj1)
```

你也可以这样,将原数据转换为纯数据就行了

```
vm.arr2 = vm.arr1.$model //正常
```

## 为什么我的指令没有效果？

```
<p title="att-{{ddd}}>例子!</p>
```

答：因为avalon只会对 `ms-*` 属性敏感，另外，花括号里的ddd要加上 `@`，即

```
<p ms-attr="{title: 'att-' + @ddd}">例子!</p>
```

## 如何在页面扫描后执行一个回调

答：avalon2支持`onReady`方法

```
var vm = avalon.define({
  $id: 'test',
  ddd: false
})
vm.$watch('onReady', function(){
  //当test这个区域第一次扫描后会被执行
})
```

[详见API文档页](#)

## 对表单元素的值输入进行限制

答：avalon提供了4个转换器，那是将`value`值上传到`vm`时用，也提供了许多格式过滤器，但在`ms-duplex`格式化很容易死循环，因此建议在另加`input`事件做处理。

比如说我们限制只能输入数字

```
<script>
    avalon.define({
        $id: 'test',
        aaa: 111,
        fix: function(e){
            e.target.value = e.target.value.replace(/\D+/, '')
        }
    })

</script>

<body :controller="test">
    <input :duplex-number="@aaa" :input="@fix"/>{{@aaa}}
</body>
```

## 如果手动执行验证

答：ms-validate提供了各种全自动的验证，但可能大家需要手动执行验证表单。在ms-validate的配置对象上添加一个onManual，页面被扫描后，你就可能拿这个方法来自己执行验证。

```
<!doctype html>
<html>
    <head>
        <meta charset="UTF-8">
        <title>Drag-Drop</title>
        <script src="../dist/avalon.js"></script>
    </head>
    <body>
        <div ms-controller="validate1">
            <form ms-validate="@validate">
                <p><input ms-duplex="@aaa" placeholder="username"
                    ms-rules='{"required":true,"chs":true}'>{{@aa}}
                </p>
                <p><input type="password" id="pw" placeholder="pass">
                </p>
            </form>
        </div>
    </body>
</html>
```

```
word"
    ms-rules='{required:true}'
    ms-duplex="@bbb" /></p>
<p><input type="password"
    ms-rules='{required:true,equalto:'pw'}' p
laceholder="再填一次"
    ms-duplex="@ccc | change" /></p>
<p><input type="submit" value="submit"/></p>
</form>
</div>
<script>

var vm = avalon.define({
    $id: "validate1",
    aaa: "",
    bbb: '',
    ccc: '',
    validate: {
        onManual:avalon.noop,//IE6-8必须指定,avalon一会儿会重写这方法
        onError: function (reasons) {
            reasons.forEach(function (reason) {
                console.log(reason.getMessage())
            })
        },
        onValidateAll: function (reasons) {//它会被onManual调用
            if (reasons.length) {
                console.log('有表单没有通过')
            } else {
                console.log('全部通过')
            }
        }
    }
}

setTimeout(function(){
    vm.validate.onManual()
})
</script>
</body>
```

```
</html>
```

## 页面用了avalon后, 元素间没有距离了

答: 因为avalon在页面加载好后,会清掉所有空白文本,减少页面的节点数,从而减少以后diff的节点个数. 详见[这里](#).

## 组件的注意事项

答 : 最好指定全局不重复的\$id, 特别在ms-for循环中,必须指定\$id

```
<!DOCTYPE html>
<html>
<head>
    <title></title>
    <meta charset="UTF-8">
    <script src="../dist/avalon.js"></script>
    <script>
        var vm = avalon.define({
            $id: 'test',
            tests: [0,1]
        })
        avalon.component('ms-button', {
            template: '<button type="button"><span><slot name="buttonText"></slot></span></button>',
            defaults: {
                buttonText: "默认内容"
            },
            soleSlot: 'buttonText'
        })
    </script>
</head>
<body>
<ul ms-controller="test">
    <li ms-for="(index,test) in @tests">
        <span ms-text="test"></span>
        <wbr ms-widget='{is:"ms-button",$id:"btn_"+index}'/>
    </li>
</ul>
</body>
</html>
```

## 为什么我的日期不能同步

```
var vm = avalon.define({
  $id: 'aaa',
  date: new Date
})

setTimeout(function(){
  vm.date = new Date
}, 1000)
```

答：因为avalon只会对number, string, boolean, 纯对象, 纯数组这几个类型同步, 其他类型需要转换. 将上面的 new Date 改成 new Date - 0 即可

## 如何将页面模块化?

答：<https://github.com/RubyLouvre/avalon/issues/1655>

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间： 2016-08-15 10:35:56

## 与jQuery共存

jQuery是世界上最流行的DOM库,它拥有各式各样的插件,在日常开发中我们可能还是离不开它.因此与它一起使用是常态,下面是一些注意

### domReady后如何扫描

```
$(function(){
    var vm = avalon.define({/* */})
    //如果你将vm定义在jQuery的ready方法内部,那么avalon的扫描就会失效,需要手动扫描
    avalon.scan(document.body) //现在只要传入扫描范围的根节点就行
})
```

## 如何AJAX提交数据

提交整个VM

```
jQuery.ajax({
    method: "POST",
    url: 'url-adress',
    //这里是取vm的数据模型 ,通过JSON.stringify会去掉其所有方法, 变成JSON字符串
    //再用JSON.parse变回纯JS对象
    data: JSON.parse(JSON.stringify(vm.$model))
})
```

提交VM中的某个 对象 属性

```
data: JSON.parse(JSON.stringify(vm.data.$model))  
``
```

提交VM中的某个`数组`属性

```
```javascript  
data: JSON.parse(JSON.stringify(vm.data.$model))
```

## 如何让后台回来的数据更新VM

后台的数据更新VM,只能是更新VM的某些已经定义属性. 如果后台数据很大,那么我们可以定义一个空对象(假如后台数据是对象类型)或一个空数组(假如后台数据是数组类型)来占位

```
var vm = avalon.define({  
    $id: 'aaa',  
    array: []  
})  
  
jQuery.ajax({  
    method: "POST",  
    url: 'url-adress',  
    data: {/**/},  
    success: function(data){  
        vm.array = data.array  
    }  
})
```

## 如何同步表单的数据

假如我的某个表单是用于jQuery的日历插件,那么它数据如何同步到vm

```
$(datepick_input_css_selector).input(function(){
    vm.aaa = this.value
})
```

如何同步复选框 在avalon中,checkbox要对应一个数组 首先是取得所有同名的checkbox,并要求它们在选中状态,然后用map方法收集它们的value值

```
$('checkbox').change(function(){
    var array = $('checkbox[name="'+this.name+'"] :checked').map(function(){
        return $(this).val();
    })
    vm.checkboxProps = array
})
```

如何同步下拉框

```
$('select').change(function(){
    vm.selectProps = $(this).val()
})
```

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook  
该文件修订时间： 2016-07-20 11:11:41

# API

## 静态成员

### vm&component

#### scan

用于描述HTML,将包括ms-controller/ms-imporant的元素的outerHTML取出来, 变成对应的vm的render方法, 最终将里面的ms-\*或双花括号变成vm中的属性与方法

注意: 如果你是将vm定义放在jQuery.ready或avalon.ready中必须手动调用这个方法.

注意: avalon2不会像avalon1那样将ms-\*属性去掉了

注意: avalon不能扫描iframe的内容

有两个参数

1. 元素节点

```
avalon.ready(function(){
    avalon.define({
        $id: 'test',
        aaa: 111
    })
    vm.$watch('onReady', function(){
        //页面上每个ms-controller, ms-important元素
        //在其区域内的所有ms-*指令被扫描后会执行
    })
    avalon.scan(document.body)
})
```

onReady回调,在2.1.0新加入,只会调用一次!

```
<!doctype html>
<html>
  <head>
    <meta charset="UTF-8">
    <script src="../dist/avalon.js"></script>
    <script>
      function heredoc(fn) {
        return fn.toString().replace(/^[\^\/]+\/\*!\?\s?/, '')
).
        replace(/\*\//[^\/]+$/, '').trim().replace(/>\s*/g, '><')
      }
      var v123 = heredoc(function () {
        /*
          <div ms-controller="test2">
          <p ms-click="@alert">123</p>
            <wbr ms-widget="{is:'ms-span'}"/>
          </div>
        */
      })
      var v456 = heredoc(function () {
        /*
          <div ms-controller="test3">
          <p ms-click="@alert">456</p>
            <wbr ms-widget="{is:'ms-span'}"/>
          </div>
        */
      })
    </script>
    <script>

      var vm = avalon.define({
        $id: 'test',
        tpl: '',
        switch1: function () {
          setTimeout(function () {

```

```
        vm.tpl = v123
    })

},
switch2: function () {
    setTimeout(function () {
        vm.test.tpl = v456
    })
}

})
};

vm.$watch('onReady', function(){
    avalon.log('vm1 onReady')
})

var vm2 = avalon.define({
    $id: 'test2',
    ddd: 'aaaa',
    alert: function(){
        avalon.log('????')
    }
});
vm2.$watch('onReady',function(){
    avalon.log('vm2 onReady')
})

var vm3 = avalon.define({
    $id: 'test3',
    ddd: 'bbbb',
    alert: function(){
        avalon.log('!!!!')
    }
});
vm3.$watch('onReady',function(){
    avalon.log('vm3 onReady')
})

var vm4 = avalon.define({
    $id: 'test4',
    fff: 'rrrr',
    alert: function(){


```

```
        avalon.log('!!!!')
    }
});

vm4.$watch('onReady', function(){
    avalon.log('vm4 onReady')
})

avalon.component('ms-span', {
    template: '<span ms-click="@click">{@ddd}</span>'

    ,
    defaults: {
        ddd:'3333',
        click: function(){
            avalon.log('inner...')

        }
    }
});
```

```
</script>
</head>
<body ms-controller="test">
    <div ms-html="@tpl"></div>
    <button ms-click="@switch1">aaaa</button>
    <button ms-click="@switch2">bbbb</button>
    <div ms-important="test4">
        {@fff}
    </div>

    </body>
</html>
```

## define

创建一个vm对象,必须指定\$id,详见[这里](#)

```
avalon.define({
  $id: 'aaa',
  bbb: 1
})
```

## ready

当**domReady**发生时,框架会自动调用的方法,会传入avalon作为参数

该方法与jQuery.ready相仿.

```
avalon.ready(fn1)
avalon.ready(fn2)
avalon.ready(fn3)
avalon.ready(fn4)
```

当**domReady**发生时,fn1, fn2, fn3, fn4会依次执行!

熟悉jQuery的人, 都知道**domReady**事件. **window.onload**事件是在页面所有的资源都加载完毕后触发的. 如果页面上有大图片等资源响应缓慢, 会导致**window.onload**事件迟迟无法触发. 所以出现了**DOM Ready**事件. 此事件在**DOM**文档结构准备完毕后触发, 即在资源加载前触发. 另外我们需要在**DOM**准备完毕后, 再修改**DOM**结构, 比如添加**DOM**元素等. 否则有可能出现“Internet Explorer无法打开站点”的问题. 要模拟此错误, 可以在页面上添加下面的代码, 并用IE6打开:

```
<div>
  <script type="text/javascript">
    var div = document.createElement('div');
    div.innerHTML = "test";
    document.body.appendChild(div);
  </script>
</div>
```

## vmodels

放置所有用户定义的vm及组件指令产生的组件vm

```
avalon.define({$id:'aaa'})
console.log(avalon.vmodels) // {aaa: vm}
```

## component

用于定义一个组件,详见[这里](#)

## components

放置所有用avalon.component方法添加的组件配置对象

## effect

用于定义一个动画效果,详见[这里](#)

## validators

放置所有验证规则,详见[这里](#)

## parsers

放置所有数据格式转换器

## filters

放置所有过滤器,也可以在上面添加你的自定义过滤器

```
avalon.filters.haha = function(str){
  return str + 'haha'
}
```

## directive

定义一个指令,请翻看源码,看css, attr, html是怎么玩的

1. 指令名
2. 配置对象

```

avalon.directive('html', {
  parse: function (copy, src, binding) {
    /*
    copy: 每次VM的属性时,avalon就会调用vm.$render方法重新生成一个虚拟DOM树
    ,这个copy就是新虚拟DOM树的一个子孙节点

    src: 第一次调用vm.$render方法生成的虚拟DOM树的某个节点,它将永驻于内存中,
    除非其对应的真实节点被移除.以后不断用src与新生成的copy进行比较,
    然后应用 update方法,最后用copy的属性更新src与真实DOM.

    binding 配置对象,包括name,expr,type,param等配置项

    return 用于生成vtree的字符串 */
  },
  diff: function (copy, src, name) {
    /*
    copy 刚刚生成的虚拟DOM树的某个子孙节点
    src 最初的虚拟DOM树的节点
    name 要比较的指令名
    */
  },
  update: function (node, vnode, parent) {
    /*
    node 当前的真实节点
    vnode 此真实节点在虚拟树的相应位置对应的虚拟节点
    parent node.parentNode
    */
  }
})

```

## noop

## 空函数

# 控制台输出

## log

类似于console.log,但更安全,因为IE6没有console.log,而IE7下必须打开调试界面才有console.log

可以传入多个参数

```
avalon.log('aaa','bbb')
```

## warn

```
avalon.warn('aaa','bbb')
```

类似于console.warn,不存在时内部调用avalon.log

## error

抛出一个异常对象

1. 字符串,错误消息
2. 可选, Error对象的构造器(如果是纯字符串,在某些控制台下会乱码,因此必须包成一个对象)

```
avalon.error('aaa')
```

# 类型检验

## type

取得目标的类型

```
avalon.type('str') // 'string'  
avalon.type(123) // 'number'  
avalon.type(/\w+/) // 'regexp'  
avalon.type(avalon.noop) // 'function'
```

## isWindow

判定是否为一个window对象

```
avalon.isWindow('ddd') // false
```

## isFunction

判定是否为一个函数

```
avalon.isFunction(window.alert) // true
```

## isObject

是否为一个对象, 返回布尔

```
avalonisObject({a:1,b:2}) // true  
avalonisObject(window.alert) // true  
avalonisObject('aaa') // false
```

## isPlainObject

判定是否为一个纯净的JS对象, 不能为window, 任何类(包括自定义类)的实例, 元素节点, 文本节点

用于内部的深克隆, VM的赋值, each方法

```
avalon.isPlainObject({}) //true  
avalon.isPlainObject(new Object) //true  
avalon.isPlainObject(Object.create(null)) //true  
var A = function(){}  
avalon.isPlainObject(new A) //false
```

## DOM

### bind

添加事件

1. 元素节点,window, document
2. 事件名
3. 回调
4. 是否捕获,可选

```
avalon.bind(window, 'load', loadFn)
```

### unbind

移除事件 参数与bind方法相同

### parseHTML

转换一段HTML字符串为一个文档对象

```
avalon.parseHTML('<b>222</b><b>333</b>')
```

### innerHTML

类似于 `element.innerHTML = newHTML` ,但兼容性更好

1. node 元素节点
2. 要替换的HTML字符串

```
var elem = document.getElementById('aaa')
avalon.innerHTML(elem, '<b>222</b><b>333</b>')
```

## clearHTML

用于清空元素的内部

```
avalon.clearHTML(elem)
```

## contains

判定A节点是否包含B节点

1. A 元素节点
2. B 元素节点

```
avalon.contains(document.body, document.querySelector('a'))
```

## String

### hyphen

转换为连字符线风格

```
avalon.hyphen('aaaAaa') //aaa-aaa
```

### camelize

转换为驼峰风格

```
avalon.hyphen('aaa-Bbb') //aaaBBB
```

## rword

切割字符串为一个个小块，以空格或逗号分开它们，结合replace实现字符串的forEach

```
"aaa,bbb,ccc".replace(avalon.rword, function(a){
  console.log(a)//依次打出 aaa, bbb, ccc
  return a
})
```

## Array&Object

### range

用于生成整数数组

1. start 开始值,
2. end 结束值, 可以为负数
3. step 每隔多少个整数

```
avalon.range(10)
=> [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
avalon.range(1, 11)
=> [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
avalon.range(0, 30, 5)
=> [0, 5, 10, 15, 20, 25]
avalon.range(0, -10, -1)
=> [0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
avalon.range(0)
```

### each

用于遍历对象或类数组,数组

```
avalon.each(arr, function(index, el){  
})
```

## mix

用于合并多个对象或深克隆,类似于jQuery.extend

注意,不要加VM批量赋值时用它

```
//合并多个对象,返回第一个参数  
target = avalon.mix(target, obj1, obj2, obj3)  
//深拷贝模式, 要求第一个参数为true, 返回第二个参数  
target = avalon.mix(true, target, obj1, obj2)
```

## slice

用于转换类数组对象为纯数组对象

1. 目示对象
2. 开始索引, 默认为0
3. 结束索引, 默认为总长

```
avalon.slice(document.body.childNodes)
```

## oneObject

将一个以空格或逗号隔开的字符串或数组,转换成一个键值都为1的对象

1. 以空格或逗号隔开的字符串或数组, "aaa,bbb,ccc",'[aaa','bbb','ccc']
2. 生成的对象的键值都是什么值,默认1

```
avalon.oneObject("aaa,bbb,ccc")//{aaa:1,bbb:1,ccc:1}
```

## Array.merge

合并两个数组

```
avalon.Array.merge(arr1,arr2)
```

## Array.ensure

只有当前数组不存在此元素时只添加它

```
avalon.Array.ensure([1,2,3],3)//[1,2,3]
```

```
avalon.Array.ensure([1,2,3],8)//[1,2,3,8]
```

## Array.removeAt

移除数组中指定位置的元素，返回布尔表示成功与否

```
avalon.Array.removeAt([1,2,3],1)//[1,3]
```

## Array.remove

移除数组中第一个匹配传参的那个元素，返回布尔表示成功与否

```
avalon.Array.remove(['a','b','c'],'a')//['b','c']
```

# 实例成员

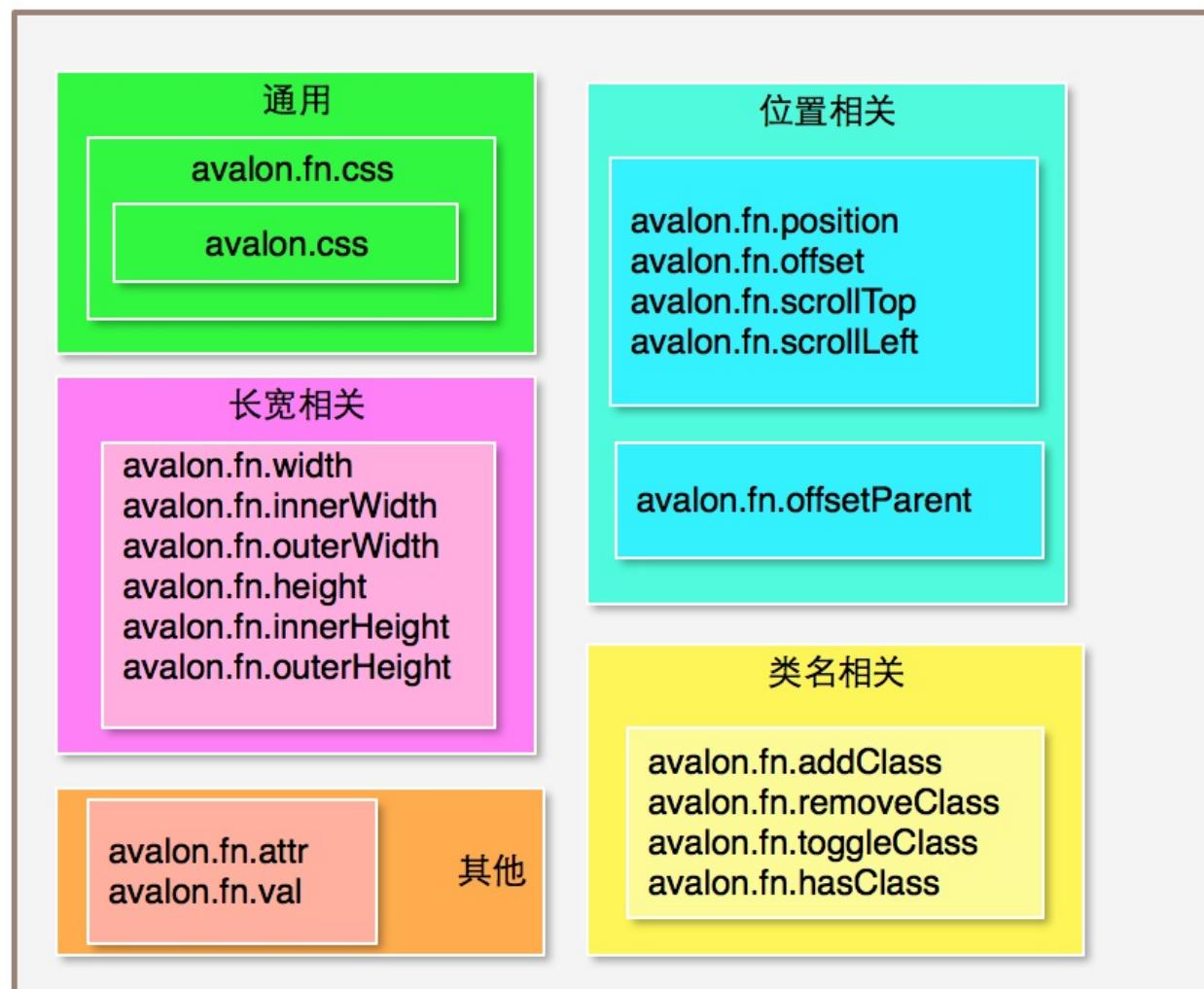
## avalon

1. 可以传入元素节点,文档对象>window对象构成一个avalon实例

```
var a = avalon(document.body)
console.log(a[0]) //document.body
console.log(a.element) // document.body
```

与jQuery对象不同的是,它只能传入一个元素或对象,而jQuery是可以传入CSS选字符串获得一大堆元素节点,组成一个类数组对象 avalon作为一个MVVM框架,目的是实现最小化刷新,通常没有操作一大堆节点的需求

avalon的实例方法主要供框架内部使用,除了自己写组件,所有操作的DOM的需求请使用ms- 如果要使用第三方的jQuery插件,请务必把它们封装成 \*avalon 的组件



## CSS

用于获取或修改样式,自动修正厂商前缀及加px,与jQuery的css方法一样智能

```
avalon(elem).css('float','left')
```

## width

取得目标的宽,不带单位,如果目标为window,则取得窗口的宽,为document取得页面的宽

```
avalon(elem).width()
```

## height

取得目标的高,不带单位,如果目标为window,则取得窗口的高,为document取得页面的高

## innerWidth

类似于jQuery的innerWidth

## innerHeight

类似于jQuery的innerHeight

## outerWidth

类似于jQuery的outerWidth

## outerHeight

类似于jQuery的outerHeight

## offset

取得元素的位置, 如 {top:111, left: 222}

## attr

用于获取或修改属性

```
avalon(elem).attr('title','aaa')
```

注意,这个方法内部只使用setAttribute及getAttribute方法,非常弱 建议使用ms-attr指令实现相同的功能

## addClass

添加多个类名, 以空格隔开

```
avalon(elem).addClass('red aaa bbb')
```

## removeClass

移除多个类名, 以空格隔开

```
avalon(elem).removeClass('red aaa bbb')
```

## hasClass

判定目标元素是否包含某个类名

## toggleClass

切换多个类名

1. 类名,以空格隔开
2. 可选, 为布尔时强制添加或删除这些类名

```
avalon(elem).toggleClass('aaa bbb ccc')
```

## bind

类似于avalon.bind

```
avalon(elem).bind('click', clickFn)
```

## unbind

类似于avalon.unbind

# 被修复了的ecma262方法

1. String.prototype.trim
2. Function.prototype.bind
3. Array.isArray  
avalon内部使用它判定是否数组
4. Object.keys
5. Array.prototype.slice  
IE6-8下有BUG,这里做了修复
6. Array.prototype.indexOf
7. Array.prototype.lastIndexOf
8. Array.prototype.forEach
9. Array.prototype.map
10. Array.prototype.filter
11. Array.prototype.some
12. Array.prototype.every

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook

该文件修订时间： 2016-09-05 01:54:00



# 更新日志

## 2.1.14

修正 ms-important的BUG

重构 escapeHTML与unescapeHTML方法

改用id来定义组件VM的\$id

修正pattern验证规则

添加大量测试, 覆盖率达到90%

## 2.1.13

修正 ms-controller, ms-important的移除类名的实现

实现后端渲染

fix safari, 微信不支持使用Object.defineProperty重写元素属性的BUG

分离DOM API

fix ms-on BUG

## 2.1.12

修正 ms-click 在 ms-if 下失效的问题 #1652

修正 limitBy BUG

修正 节点对齐算法 BUG

优化 mediatorFactory

修正 data-for-rendered BUG

## 2.1.11

修正 limitBy BUG

修正 节点对齐算法 BUG

优化 mediatorFactory

修正 data-for-rendered BUG

## 2.1.10

修正 ms-for两个BUG

修正 ms-controller BUG

## 2.1.9

component/initjs中的protected变量更名为immunity,方便在严格模式下运行

为伪事件对象过滤掉原生事件对象中的常量属性

修复class,hover,active指令互相干扰的BUG

修复事件绑定中表达式太复杂,不会补上(\$event)的BUG

当组件被移出DOM树并且没有被cached时,其虚拟DOM应该清空上面的事件

重写for, widget指令,

## 2.1.8

component/initjs中的protected变量更名为immunity,方便在严格模式下运行

为伪事件对象过滤掉原生事件对象中的常量属性

修复class,hover,active指令互相干扰的BUG

修复事件绑定中表达式太复杂,不会补上(\$event)的BUG

当组件被移出DOM树并且没有被cached时,其虚拟DOM应该清空上面的事件

## 2.1.7

修正注释节点包括HTML结构(里面有引号),节点对齐算法崩溃的BUG

修正tap事件误触发BUG

升级ms-widget的slot机制,让它们的值也放到组件VM中

添加:xxx短指令的支持

## 2.1.6

### 全新的lexer与插值表达式抽取方法

添加unescapeHTML与escapeHTML方法

修正xmp元素的内容生成BUG

修正input.value = newValue的同步BUG

修正双击事件BUG

修正ms-widget遇上ms-if找到原先占位DOM的BUG

## 2.1.5

always **添加htmlify模块**

用于解决IE6-7对colgroup,dd,dt,li,options,p,td,tfoot,th,thead,tr元素  
自闭合,html parser解析出错的问题

重构ms-controller, ms-important指令

虚拟DOM移除template属性

修正ms-for的排序问题

fix 在chrome与firefox下删掉select中的空白节点，会影响到selectedIndex  
BUG

## 2.1.4

修复IE光标问题

修复输入法问题

修复双层注释节点ms-for循环问题(markRepeatRange BUG)

ms-html中script, style标签不生效的问题

## 2.1.3

修正isSkip方法,阻止regexp, window, date被转换成子VM

checkbox改用click事件来同步VM #1532

ms-duplex-string在radio 的更新失效问题

## 2.1.2

ms-for+expr在option元素不显示的问题 (实质是节点对齐问题)

模板中的@×没有被htmlDecode的问题

绑定在组件模板中最外层元素上的事件不生效

ie7,8下 ms-duplex 因为onpropertychange环调用，导致辞爆栈的问题

修正节点对齐算法中 对空白节点与script等容器处理的处理

## 2.1.1

简化VElement转换DOM的逻辑

将改 order的连接符为 ‘， ’， 这样就可以重用更简单的avalon.rword

修正e.which BUG

修正 **ms-duplex-checked**在低版本浏览器不断闪烁的问题

## 2.1.0

**重构lexer方法**

添加新的对齐节点算法

修复IE6-8下复制闭包中的对象返回相同对象,导致ms-for出BUG的问题

所有vm都支持**onReady**,在它第一次刷新作用区载时触发

重构ms-for, ms-html,ms-if,ms-text,ms-html,ms-on指令

参考react 的classNames插件， 重构ms-class/active-hover,

上线全新的parseHTML, 内部基于avalon.lexer, 能完美生成script, xml,svg 元素

重构isInCache, saveInCache

Copyright © 司徒正美 2013–2016 all right reserved, powered by Gitbook

该文件修订时间： 2016-08-30 17:59:12