1. 表单样式设计 数组方法 还有添加按钮 过滤
2. 过滤器

全局定义过滤器-🡪



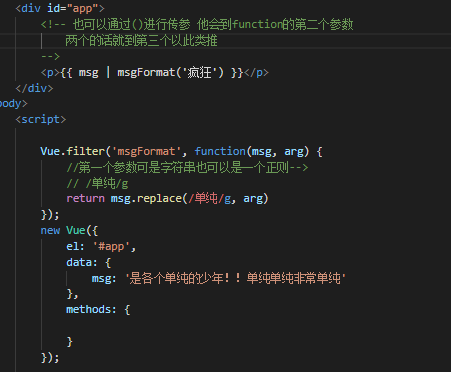
注意：过滤器只用用在mustache插件表达式( {{}} )和v-bind中。

参数可以填data也可以填data里面的message等等(填需要过滤的字符)。

在html中表现形式为-🡪



过滤器传参的使用🡪



可以多个过滤器处理 他会从左到右依次使用过滤器-🡪



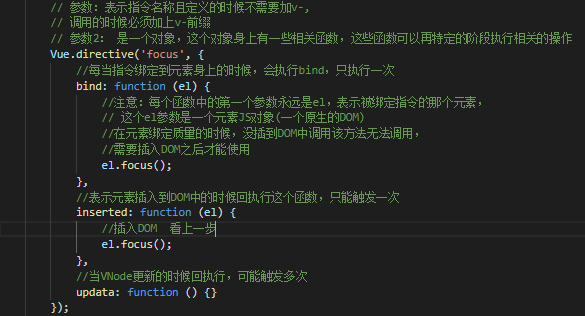
先msgFormat过滤在test过滤

三．自定义按钮修饰符.

Vue.config.keyCodes.f2 = ASC码

四．padStart(填充几位, “前面填充的东西”) padEnd(填充几位,”后面填充的东西”)需转化为字符串填充🡪时间.toString().padStart(2, “0”);时间一般这样填充

五．自定义指令



注意：设置样式指令的时候不需要inserted。因为不插入DOM就能生效。(只需要绑定指令到元素身上就能生效)

总结：样式相关的在bind中设置。 行为有关的在inserted设置

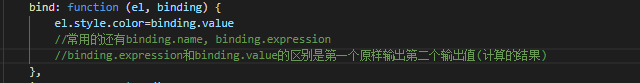
六．自定义指令传参(设置第二个参数)

Html中



注意：”yellow”表示一个变量会到data里面去找， “’yellow’”表示一个字符，传参

Script中



七．形参不是固定写死的只要下面用和上面同意就行如上面的binding可以随便写什么下面同样用写的来接受就行🡪

Bind:function (element, second) {

Element.style.color = second.value

}

八．Vue里面的methods, filters, directives都是对象。

注意：局部定义才用这个加s 全局直接Vue.filter没s

九．简写自定义指令

只需要bind和updata不需要inserted可简写成🡪

Vue.directive(‘自定义指令名称’, function () {})---》



如果自定义有单位的可以paseInt(binding.value) + ‘单位’ 来容错。

十．生命周期

生命周期钩子 = 生命周期函数 = 生命周期事件

1. beforeCreate在beforeCreate生命周期函数执行的时候，

data和methods中的数据没有初始化。

1. created在created生命函数执行的时候，data和methods都已经初始化好了。表示如果要调用methods中的方法或者要操作data中的数据最早只能在created中操作

data和methods初始化完成

1. 接下Vue开始编译模板，把vue中的指令执行，最终在内存中生成一个编译好的最终模板字符串然后把模板字符串渲染到内存中的DOM，此时只是在内存中渲染好了模板，并没有挂载到真正的页面上去。
2. beforeMount在beforeMount生命周期函数，表示模板已经在内存中编译完成了，但尚未把模板渲染到页面上去，页面中的元素没有被真正替换过来，只是之前的模板字符串(页面还是旧的页面)
3. 将内存中的模板真正的替换到页面上去。
4. mount在mount生命周期执时，表示内存中的模板已经真实挂载到了页面中，用户可以看到渲染好的页面了。

执行这个函数的时候就表示实例已经完整的创建完成了。

如果需要通过插件操作页面上的DOM节点，最早在mounted中进行

上面是创建阶段的函数

之后进入了运行阶段的生命周期函数

1. 执行beforeUpdate时候页面中的数据是旧的，data数据是最新的，页面中的数据和data中的数据并未同步。
2. 执行updated执行这个函数的时候数据已经和页面同步了，都是最新的。
3. BeforeDstroy钩子函数执行的时候vue实例从运行越短进入了销毁阶段，实例身上所有的data和所有的methods，过滤器等都还可以执行 还没真正的执行到销毁。
4. dstroy组件已经被完全销毁了此时所有的指令，数据，函数，过滤器都不可用了。

十一．vue的数据请求

可以使用除了vue-resource之外，还可以使用’axios’第三方包来实现数据请求

注意：引入vue-resource包需要放在vue包后面

1>GET请求

this.$http.get(‘地址’, [options]).then(successCallback, errorCallback)--->

this.$http.get(‘https://vue.studyit.io/api/getlunbo’).then(function (result)) {

//通过result.body拿到后代数据 })

2>POST请求

this.$http.post(‘地址’, [body],[options]).then(successCallback, erroeCallback)--->

[body]是表单对象注意是一个对象 发送给服务器的数据

post的第三个参数是提交的表单格式(手动发起的post请求默认没有表单格式需设置emulateJSON)

This.$http.post(‘https://vue.studyit.io/api/post’, {}, {emulateJSON: true}).then(result => {

//通过result.body拿到后台数据})

3>JSONP请求

this.$http.jsonp(‘地址’,[options]).then(successCallback, errorCallback)--->

和GET请求差不多

注意：服务器只接受表单格式的提交所以POST请求需设置第三个参数来设置提交数据的格式，第三个参数配置对象在GET和JSONP请求可以省略，errorCallback可在所有请求省略，

通过参数来拿到服务器的数据，虽然可以用参数.data拿到数据但更多推荐使用参数.body拿到后台数据，因为data在控制台是黑的而body是亮的 data像系统自定义的

十二．JSONP请求数据详情

由于浏览器安全不允许跨域请求所以可以通过script标签的src来请求数据达到跨域的目的