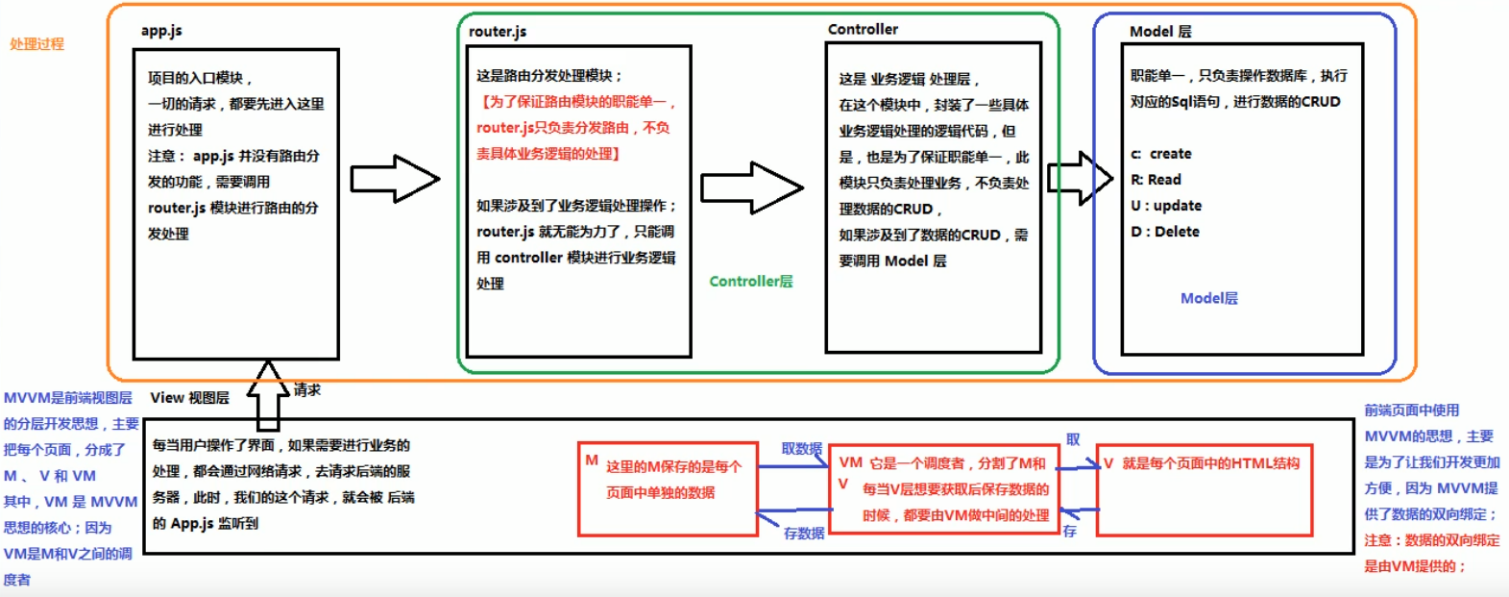
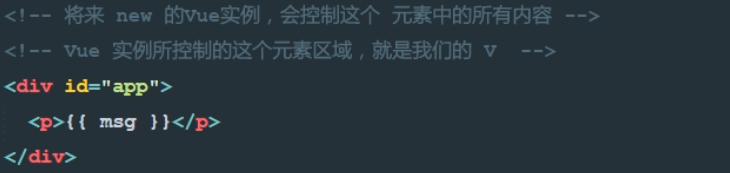
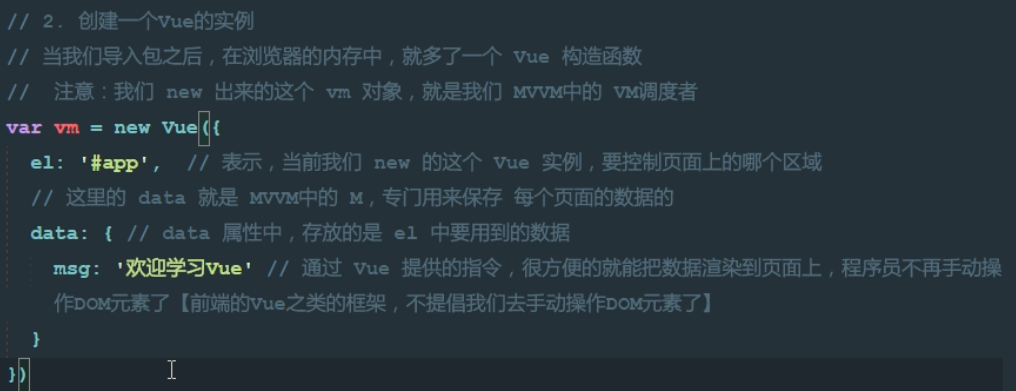
MVC和MVVM的区别



MVVM中的M(模型)、V(视图)、VM(调度)所对应的关系





**指令的使用**

|  |
| --- |
| **v-cloak**    插值表达式和**v-text**的区别    **v-html**的使用  注意：v-html可以将字符串类型的HTML解析成HTML语言，但是标签中有文字的话，文字将会被覆盖      **v-bind**      **v-on**      缩写形式 |

**事件修饰符**

|  |
| --- |
| **.stop**      **.prevent**    **.capture**        **.self**  只有点击自己的时候才可以触发事件  **.once**  只能触发一次 |

**v-model双向绑定**

|  |
| --- |
| **v-model计算的应用实例** |

**Vue中使用样式**

|  |
| --- |
|  |

**Vue中使用内联样式**

|  |
| --- |
|  |

**v-for的四种用法**

|  |
| --- |
|  |

**v-if和v-show**

|  |
| --- |
|  |

创建组件

|  |
| --- |
| 方式一：    使用组件    方式一简化版 |

方式二

|  |
| --- |
|  |

方式三

|  |
| --- |
|  |

定义私有组件

|  |
| --- |
| 报错 |

组件中的数据及访问

|  |
| --- |
| 使用组件    自定义组件 |

使用v-if和v-else切换组件

|  |
| --- |
| <div id="app">  <!-- @click.prevent阻止点击超链接时，跳转到href -->  <a href="" @click.prevent="flag=true">登陆</a>  <a href="" @click.prevent="flag=false">注册</a>  <login v-if="flag"></login>  <register v-else></register>  </div>  <script type="text/javascript">  Vue.component("login",{  template:"<h3>登陆</h3>"  })  Vue.component("register",{  template:"<h3>注册</h3>"  })  </script> |

使用component标签来实现组件的切换

|  |
| --- |
|  |

父组件向子组件传递数据

|  |
| --- |
|  |

**在webpack中使用Vue组件**

|  |
| --- |
|  |

**keyup事件**

|  |
| --- |
| 监听文本框的变化 |

**watch事件**

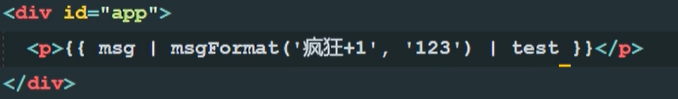
|  |
| --- |
| watch是个对象，监听多个值的时候在里面写多个值  watch:{  text1:function(){},  text2:function(){}  }  watch监听路由变化    **计算属性computed** |

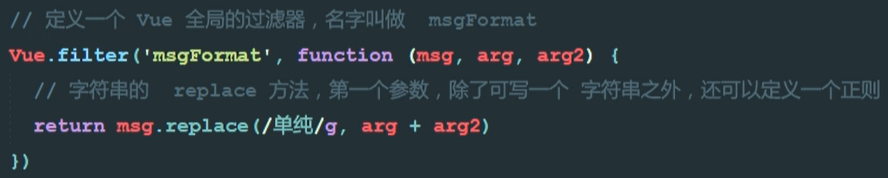
**过滤器**

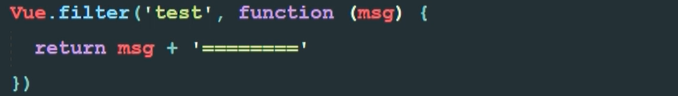


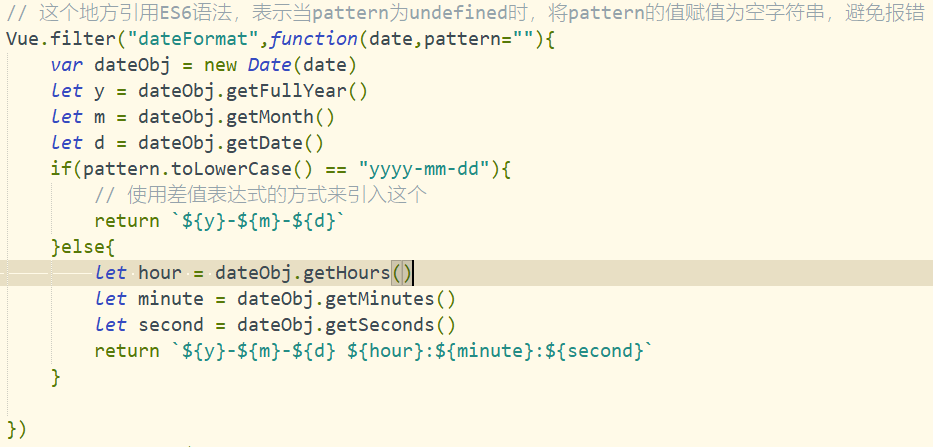


**全局过滤器**









**私有过滤器**



**字符串填充方法**

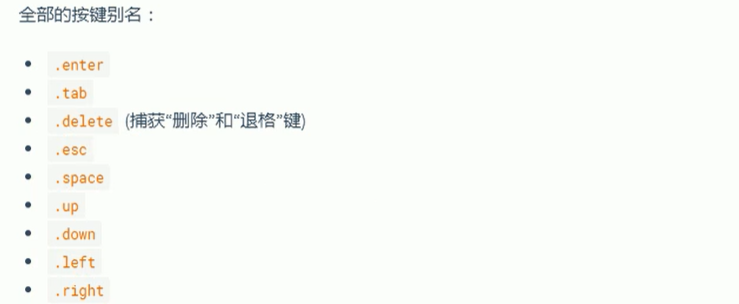


padStrat:在字符串的前面，将字符串添加长度到maxLength，填充字符串为fillString

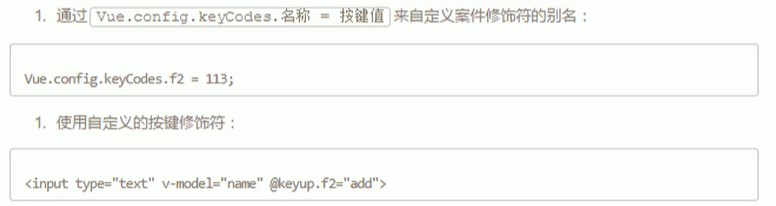
padEnd：在字符串的后面，将字符串的长度填充到maxLength，填充字符串为fillString

**按键修饰符**





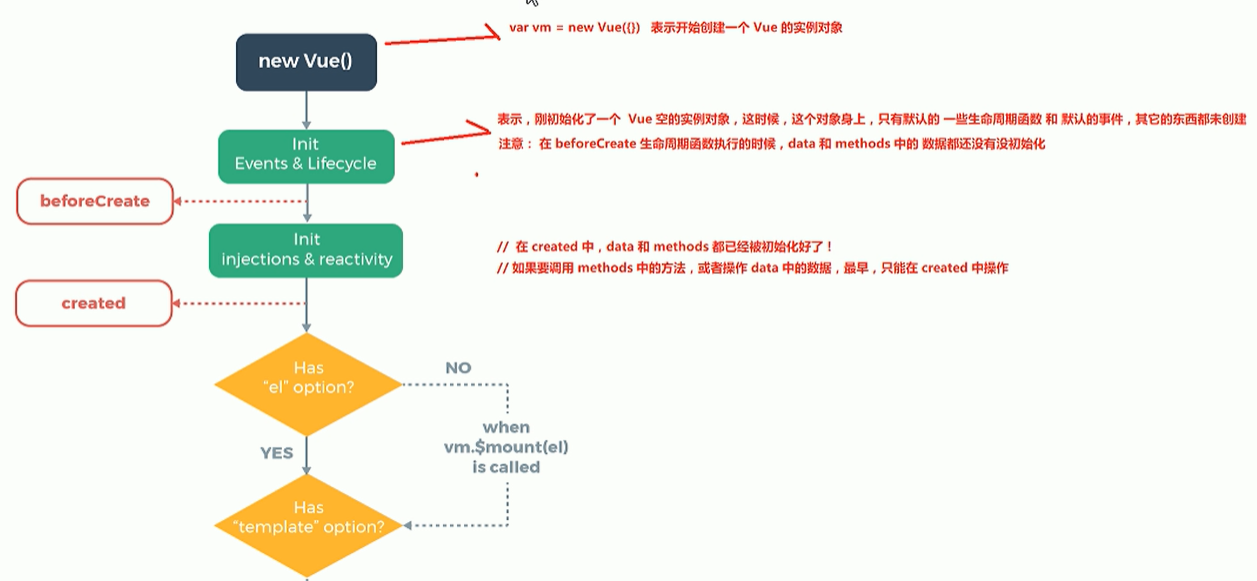
**自定义按键修饰符**

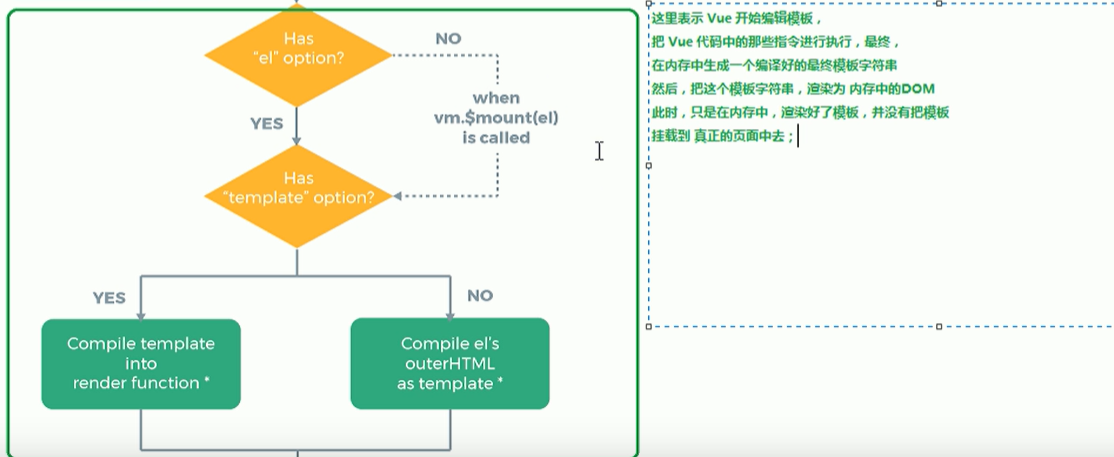


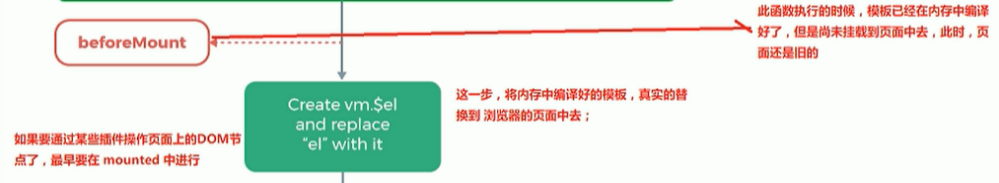
**自定义指令**

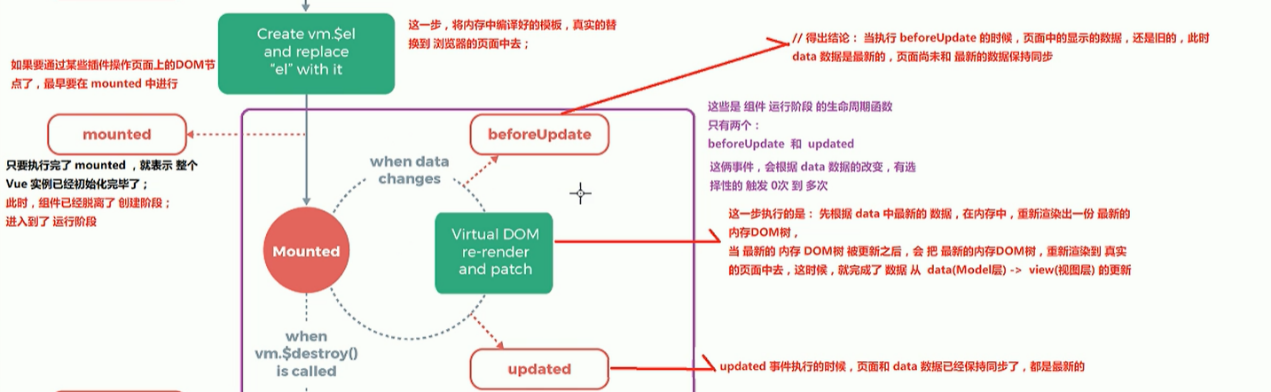
|  |
| --- |
| **定义全局指令**      **钩子函数参数**    **获得指令后面传递过来的参数值**      **自定义私有指令**    **指令简写** |

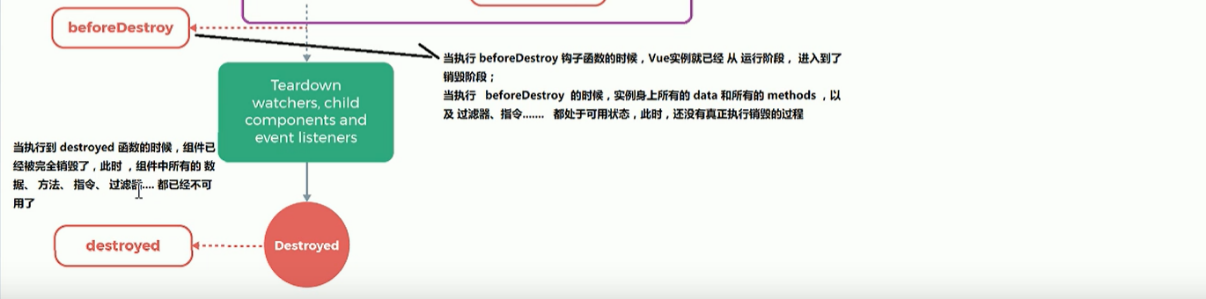
**Vue的生命周期**











**Vue中的template**

|  |
| --- |
| **template的使用** |

**slot的使用**

|  |
| --- |
| **匿名插槽**          **具名插槽**          <https://blog.csdn.net/weixin_41646716/article/details/80450873>  **作用域插槽**  父组件中使用子组件中的数据  子组件      父组件    页面显示 |

**ref的使用**

|  |
| --- |
| <https://www.jianshu.com/p/623c8b009a85>  **1、获得当前页面的DOM元素**        **2、获取子组件中的数据**  父组件    子组件        3、调用子组件中的方法  父组件    子组件        4、通过ref调用父组件的方法 |

**Vue中的nextTick**

|  |
| --- |
| 点击按钮前页面数据    点击按钮后页面数据 |

**Vue中的keep-alive标签**

**Vue的地址**

[**https://liqing.blog.csdn.net/**](https://liqing.blog.csdn.net/)

|  |
| --- |
| 作用：当进行组件切换时，包裹在<keep-alive><keep-alive>中的组件在切换回来时，组件中的数据还保持的原来的数据，不会回到初始化的状态，没有包裹的组件中的数据，数据会显示为组件中初始化的数据  **注意**：使用v-if切换组件也可以达到同样的效果  **组建一**    **组件二**    **初始化数据**    **页面代码**    **页面内容**    **点击第二个按钮，修改页面数据**    **切换组件**  切换为组件二    切回组建一 |

**Vue的项目搭建**

|  |
| --- |
| 1、打开项目所在的文件路径，cmd进入  2、创建项目名称  vue init webpack 项目名称  3、项目创建过程中的选择    4、项目初始化  npm install  5、启动项目  npm run dev |

**Vue项目的文件结构**

|  |
| --- |
| **一、main.js**  main.js文件是我们的入口文件，主要用户初始化vue实例并使用相应的插件    1.首先我们来看第一句代码：  import Vue from ‘vue’  其实最完整的写法是：  import Vue from "../node\_modules/vue/dist/vue.js";  2.我们再看看第二、三行代码  import App from ‘./App‘  import router from ‘./router‘  其实在这句代码中省略了后缀名，完整的代码应该是：  import App from ‘./App.vue‘  import router from ‘./router.js‘  在文件引入时，我们通常可以省略后缀名，而Vue会自动帮我们添加上‘.js‘\‘.vue‘\‘.less‘等后缀名，去帮我们匹配相应的文件。  **二、App.vue文件**  是我们的主组件，所有的组件都在APP.vue下面进行切换  **三、index.html文件**  开发的时候看起来没什么关联，其实webpack打包时，生成的index.html里面引入了各个js |

**路由的引入及使用**

|  |
| --- |
| 1、路由的安装  npm install vue-router --save-dev  注意：有可能因为路由版本问题造成路由无法访问的情况，确保路由配置没有问题时，执行命令  npm i vue-router@3.0 -S  2、 |