三大前端框架 React Angular Vue

|  |  |
| --- | --- |
| React |  |
| Angular |  |
| Vue | 是一款构建用户界面的框架，MVC的视图层 |
| 共同点 | 都是前端的主流框架 |

MVC 模型

|  |  |  |
| --- | --- | --- |
| MVC是程序的一种设计模式，把业务逻辑，数据，界面（视图）显示分离，通过控制器连接业务逻辑与视图。 | | |
|  |  |  |
| m=>model | 模型 | 程序处理应用数据逻辑的部分，通常模型对象赋值在数据库中存取数据。 |
| v=>view | 视图 | 程序中处理数据显示的部分，通常依据模型数据创建的 |
| c=>controller | 控制器 | 业务逻辑与视图交互的部分，通常从视图读取数据，控制用户输入，并向模型发送数据。 |

他人的案例

|  |  |  |
| --- | --- | --- |
| 个人博客 | Vue+Node+Mysql | <https://github.com/Moon-Future/Vue-Node-Blog> |

vue完全卸载

|  |
| --- |
| 删除node\_modules等文件 |
| npm cache clean --force |
| npm uninstall -g @vue/cli |
| npm uninstall -g vue |
| 清除缓存，cmd卸载，重装 |

vue安装

|  |  |  |
| --- | --- | --- |
| 安装之前 | npm install vue |  |
| 全局安装 | npm install -g @vue/cli | 公司使用 npm install -g vue-cli |
| 删除缓存 | npm cache clean --force |  |

vue-cli 脚手架

|  |
| --- |
| npm install -g vue-cli |
| vue init webpack ku |
| npm install （安装依赖） |
| npm run dev |
|  |

注册组件

|  |
| --- |
| export default {  name: 'app',  components:{//1、组件注册  asd:qwe  },  }; |
| <asd>2、使用组件（通过标签使用）这里的内容不会显示</asd> |
| import qwe from "./components/test1.vue" 导入组件 |

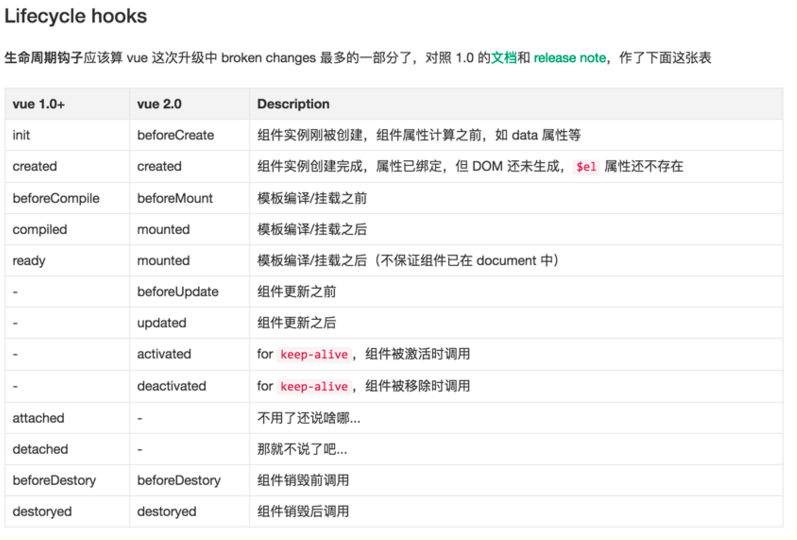
基础语法

|  |  |  |
| --- | --- | --- |
| data(){ return{}} | 静态数据 | {{ obj.name }}  data() {  return {  tit: '标题',  num: 66,  bool: true,  bool2: true,  arr: ['a', 'b', 'c'],  obj: { name: '张三', age: 18 },  arr\_obj: [{ name: '赵六', age: 18 }, { name: '李四', age: 18 }, { name: '王五', age: 18 }],  cla: 'on'  };  }, |
| methods:{} | 方法数据 | v-on:click="asd"  methods: {  asd: function() {  this.bool = !this.bool; //修改data数据  }  } |
| scoped | 当前页面生效 | <style scoped> |
| script中的this | 指data静态数据 |  |
| 保存后会自动更新页面 |  |  |

单双向数据，组件，侦听器，计算，过滤器

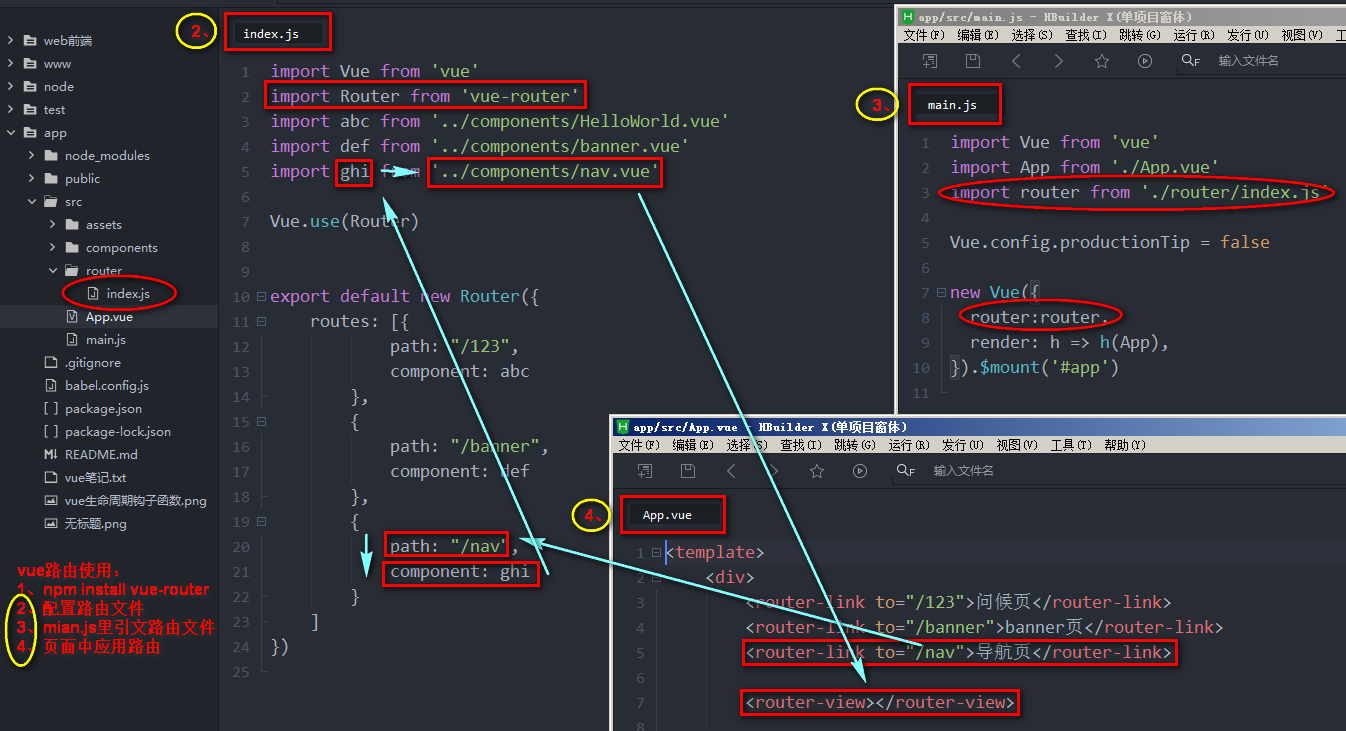
|  |  |  |
| --- | --- | --- |
| v-bind | 单向数据 |  |
| v-model | 双向数据，改变静态数据里的值也改变 |  |
| v-html |  |  |
| maxlength | 限定字符串长度 |  |
| computed:{} | 页面多次执行里面的函数时，只会执行一次，避免了重复运行 |  |
| watch:{} | 侦听器 |  |
| filters:{} | 过滤器 | {{meg|len|qwe}}  len:function(val){  var le=val.length  return "您已输入"+le+"个字符"  } |
| template | 在if 判断及for循环中不会自动转化为一个div |  |

钩子函数 生命周期



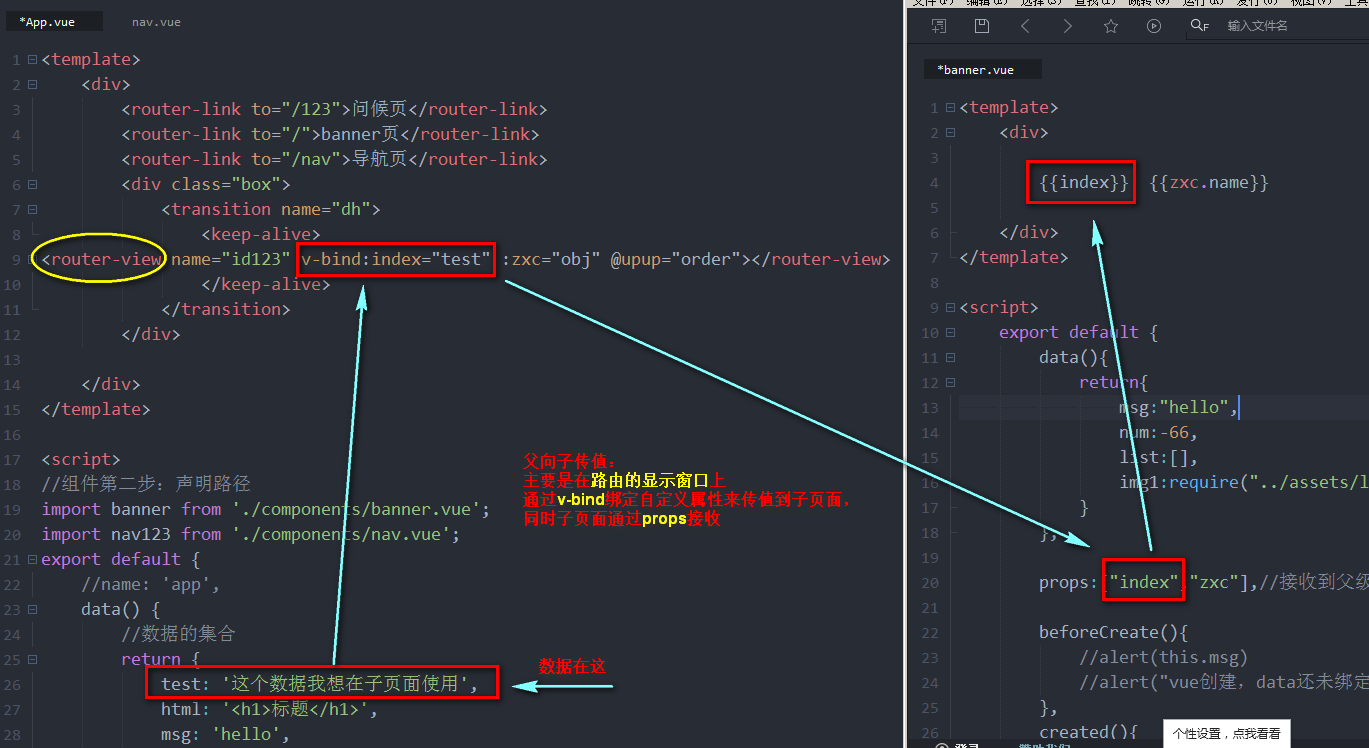
|  |  |
| --- | --- |
| beforeCreate | data之前 |
| created | data之后 |
| beforeMount | 编译之前 |
| mounted | 编译之后 |
| mounted | 编译之后（不保证组件已在documnet中） |
| beforeUpdate | 组件更新之前 |
| updated | 组件更新之后 ，数据更改 |
| activated | 组件被激活调用 |
| deactivated | 组件被移除时调用 ， |
| beforeDestory | 组件销毁前调用 a连接切换 |
| destoryed | 组件销毁后调用 |

router 路由

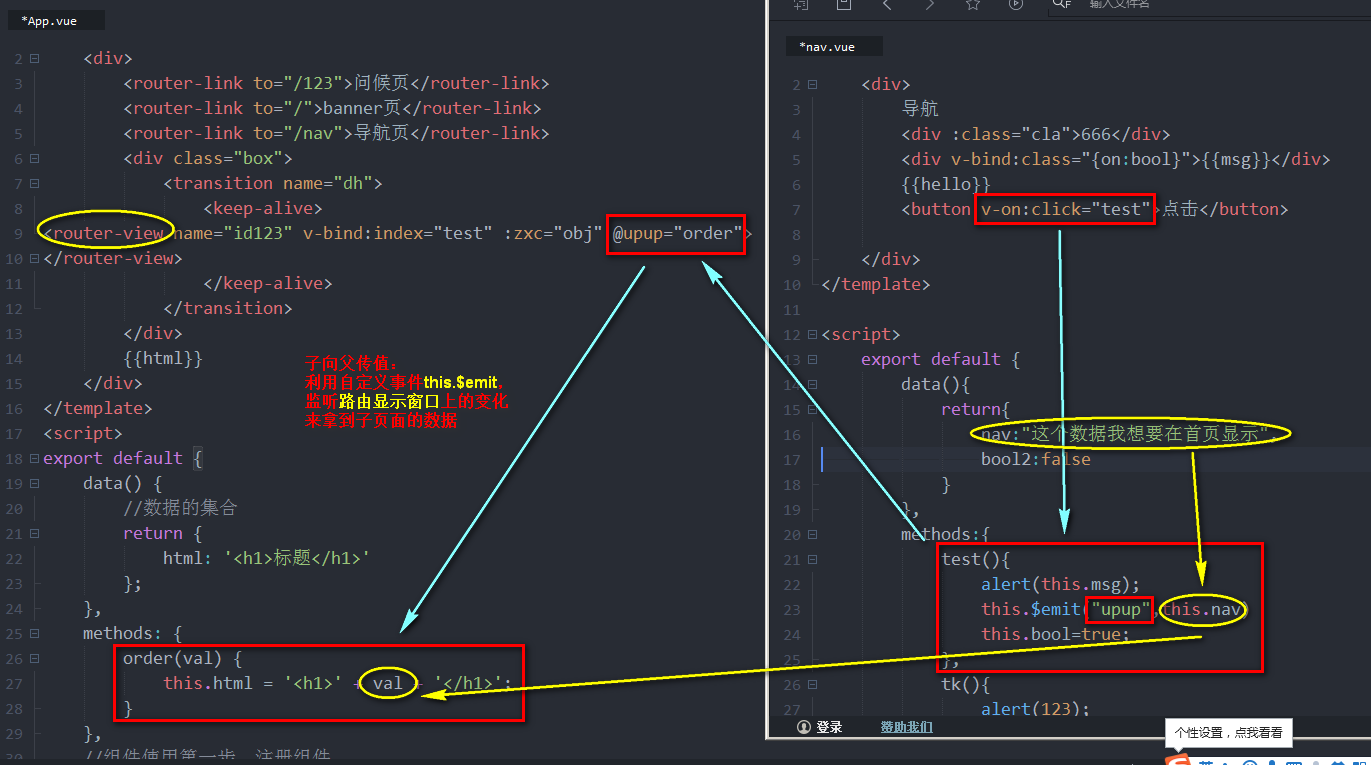


|  |  |  |
| --- | --- | --- |
|  |  |  |
| mode:"history" | 解决地址栏的#号问题 |  |

父向子传值



子向父传值



传值

|  |  |
| --- | --- |
| 父向子传值 | 子向父传值 |
| :ku="fu" | @click="beijin" |
| props:["ku","obj"], | beijin(){this.$emit("jieshou",this.num);} |
| {{ku}} | @jieshou="jie" |
|  | jie(val){ this.ku=val; } |
|  | ku:null ==val |

computed 计算函数

|  |  |
| --- | --- |
| computed |  |
| 计算属性的结果会被缓存，在发生改变的时候才重新进行求和。  意味着这个属性没有发生改变，多次访问这个计算属性会立刻返回之前的计算结果，而不必再次执行函数。 | |

v-show VS v-if

|  |
| --- |
| v-if与v-show 都可以动态控制dom元素显示隐藏 |
| v-if显示隐藏是将整个dom元素添加或删除，  而v-show隐藏则是为此元素添加display:none属性，dom元素还在 |

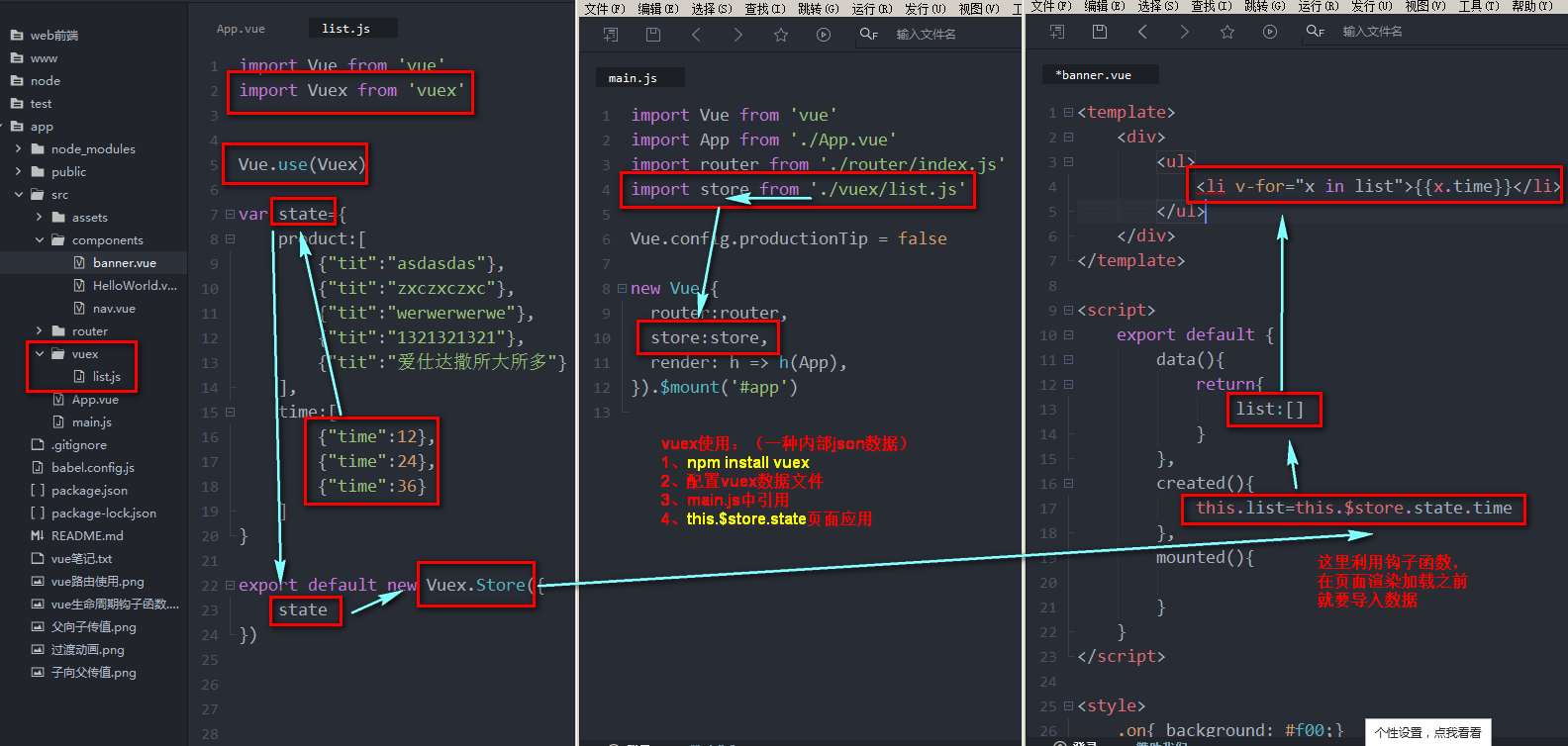
v-if 与 v-for一起使用

|  |
| --- |
| v-for具有比v-if更高的优先级 |

动画



vuex 路由懒加载



路由懒加载的好处

|  |  |
| --- | --- |
| 如果没有用用懒加载，项目打包后的文件会异常的大，造成进入首页时，需要加载的内容过多，时间过长，会出现一定时间的白屏，不利于用户体验，而运用懒加载则可以将页面进行划分，需要的时候加载页面，可以有效的分担页面所承接的加载压力，减少首页加载用时 | |
| 使用方式 | <router-view name="luyou"></router-view>  {path:"/123",components:{ //懒加载路由  "luyou":resolve=>require(['../components/test/test2.vue'],resolve)  },}, |

图片与背景图片注意事项 背景路径

|  |  |
| --- | --- |
| data中的图片路劲 | url:require("./assets/logo.png") |
| html中的图片路劲 | <img src="./assets/logo.png" alt=""> |
| 背景图片的路劲 | background: url(assets/logo.png); |

事件修饰符

|  |  |  |
| --- | --- | --- |
| .stop | 阻止冒泡事件或者说阻止单击事件继续传播 | <av-on:click.stop="doThis"></a> |
| .prevent | 阻止提交事件重新加载页面 | <form v-on:submit.prevent="onSubmit"></form> |
| .capture | 先在此处理，然后才交由内部元素进行处理 |  |
| .self | 只当在 event.target 是当前元素自身时触发处理函数  即事件不是从内部元素触发的 | <divv-on:click.self="doThat">...</div> |
| .once |  |  |
| 修饰符串联 | <a v-on:click.stop.prevent="doThat"></a> | |

按键修饰符 系统修饰符

|  |  |
| --- | --- |
| .enter | .ctrl |
| .tab | .alt |
| .esc | .shift |
| .space | .meta |
| .up |  |
| .down |  |
| .left |  |
| .right |  |

<keep-alive>

|  |
| --- |
| 用来页面切换时，input里面的数据不会更改 |
| <keep-alive>  <router-view :opm="meg" v-on:suibian="message" :zou="song" @jieshou="jie">  </router-view>  </keep-alive> |
|  |
|  |

vuex 静态资源 其它资源的调用

|  |  |
| --- | --- |
| npm install vuex |  |
| 单个 | //主要模块  import Vue from 'vue'  import Vuex from 'vuex'  Vue.use(Vuex)  var state={  product:[  {"tit":"gettreserd"},  {"tit":"breewererw"},  {"tit":"wettrerhse"},  {"tit":"afewgertge"}  ],  time:[  {"time":45},  {"time":32},  {"time":41},  {"time":26}  ]  }  export default new Vuex.Store({  state  })  //main.js  import store from './vuex/index.js'  store:store, |
| 多个 | // 主要模块  import new from ‘./new/new.js’  modules:{  new  }  var ji={  // 新闻模块  pro:[{name:"jieke"},{name:"mingjie"},{name:"xialuoke"},{name:"youjie"}]  }  export default{  ji  } |
| 使用数据 | //数据加载之前  created(){  this.news=this.$store.state.news  } |

axios 使用 免费接口调用

|  |  |
| --- | --- |
| npm install axios |  |
| main.js中 | import axios from 'axios'  Vue.prototype.ajax=axios |
| 原始方式  let xhr=new XMLHttpRequest();  xhr.open("get","https://www.apiopen.top/novelInfoApi?name=盗墓笔记",false);  xhr.onload=function(){  that.list=JSON.parse(this.responseText).data.data;  // console.log(JSON.parse(this.responseText).data.data);  }  xhr.send(); | |
| axios方式  beforeCreate() {  //组件刚刚被创建 data数据计算之前  this.ajax.get("https://www.apiopen.top/novelInfoApi?name=盗墓笔记").then(function(val){  console.log(val.data.data.data)  })  }, | |
|  | |
|  | |

node 端口 数据 http https 跨域

|  |
| --- |
| //设置允许跨域的域名，\*代表允许任意域名跨域  res.header("Access-Control-Allow-Origin","\*");  //允许的header类型  res.header("Access-Control-Allow-Headers","content-type");  //跨域允许的请求方式  res.header("Access-Control-Allow-Methods","DELETE,PUT,POST,GET,OPTIONS"); |

axios使用注意

|  |  |
| --- | --- |
| 获得数据this的指向 | 在外面进行赋值 that=this 这个this指静态数据 |

jquery引入方式

|  |
| --- |
| //方式1 public index.html中引入  <link type="text/jscript" rel="stylesheet" href="<%= BASE\_URL %>jquery-1.7.2.js"> |
|  |
|  |

router路由配置

|  |  |
| --- | --- |
| npm install vue-router |  |
| //router 中 index.js  import Vue from 'vue'  import Router from 'vue-router'  export default new Router({  /\* 路由路径配置 \*/  routes:[  {path:"/ming1",component:resolve=>require(['../components/ming/ming1.vue'],resolve)},  {path:"/ming2",component:resolve=>require(['../components/ming/ming2.vue'],resolve)},  {path:"/ming3",component:resolve=>require(['../components/ming/ming3.vue'],resolve)}  ],  /\* 去除#号 \*/  mode:"history"  }) | |
| //main.js  import router from './router/index.js'  router:router, | |

viewpor兼容

|  |  |
| --- | --- |
| 由于viewport单位得到众多浏览器的兼容，lib-flexible这个过渡方案已经可以放弃使用，不管是现在的版本还是以前的版本，都存有一定的问题。建议大家开始使用viewport来替代此方案。vw的兼容方案可以参阅《[如何在Vue项目中使用vw现移动端适配](https://www.w3cplus.com/mobile/vw-layout-in-vue.html)》一文。 | |
| 做响应式页面很好用的一个插件 | https://github.com/amfe/lib-flexible |
|  |  |