1. MVVM由View层 视图层，view-model层 连接view层和model层以及Model层 模型层组成
2. 解耦，使模块之间减少依赖，方便维护
3. 组件化：各自作用的简单组件，构成更加复杂的组件，是从UI界面划分的；模块化：是从代码逻辑角度划分；实现组件化和模块化可以解耦，减少各个组件之间的耦合度，减少单个文件的代码量
   1. beforeCreate 创建之前 获取不到data，可以进行重定向；
   2. created 创建之后 最早可以设置data值，数据初始化；
   3. beforeMount 界面挂载之前 比created多了需要渲染的dom对象
   4. mounted 渲染之后 可以对dom进行初始化
   5. beforeUpdate 更新之前 data已经修改，但是还没有更新，不能异步操作；
   6. updated 更新之后 data和界面已经同步；
   7. beforedestroy 销毁之前；
   8. destroyed 销毁之后
4. 父组件通过props向子组件传值；标签的属性用：绑定

<body>

    <div id='app'>

        <son :msg="msg"></son>

    </div>

    <template id="sontemp">

        <div>

            {{msg}}

        </div>

    </template>

    <script>

         const vm = new Vue({

            el: '#app',

            data: {

                msg: "我是一组信息"

            },

            components: {

                son: {

                    template: "#sontemp",

                    props: ["msg"]

                }

            }

        })

    </script>

</body>

1. 子组件向父组件传递数据通过$emit传递

<body>

    <div id='app'>

        <son :msg="fatherMsg" @change-msg="fatherChange"></son>

    </div>

    <template id="sonTemp">

        <div>

            {{sonMsg}}

        </div>

    </template>

    <script>

        const vm = new Vue({

            el: '#app',

            data: {

                fatherMsg: ""

            },

            methods: {

                fatherChange() {

                    this.fatherMsg = sonMsg;

                }

            },

            components: {

                son: {

                    template: "#sonTemp",

                    data() {

*return* {

                            sonMsg: "我是子组件的数据"

                        }

                    },

                    methods: {

                        change() {

                            this.$emit("change-msg", this.sonMsg);

                        }

                    },

                }

            }

        })

    </script>

</body>

1. 需要进行转化后才能得到的属性；依赖内容变化的时候才能重新进行转化
2. 通过条件选择是否显示；v-if 是删除dom；v-show是设置display：none；
3. 需要定义过滤器 ；全局定义 Vue.filter(过滤器的名字, function (data, ...){return "要显示的值"})；私有定义 filters:{过滤器的名字(data,..其他参数){return "要显示的值"}}
4. 为了提高列表的重排率
5. .stop 阻止事件冒泡，从里到外；.prevent 阻止默认事件；.capture 使用事件捕获模式，从父容器到目标容器；.self 只有事件作用于该元素本身的时候会触发；.once 事件只触发一次
6. 导入 import 导出export 默认导出只可以导出一个变量，普通导出可以导出多个变量

*for* (let i = 0; i < list.length; i++) {

*for* (let j = 0; j < list.length - 1; j++) {

*if* (type == 'asce') {

*if* (list[j].id > list[j + 1].id) {

                                    let element = list[j];

                                    list[j] = list[j + 1];

                                    list[j + 1] = element;

                                }

                            } *else* {

*if* (list[j].id < list[j + 1].id) {

                                    let element = list[j];

                                    list[j] = list[j + 1];

                                    list[j + 1] = element;

                                }

                            }

                        }

                    }

1. $route可以获取到元数据的信息；$router 是router构造方法的实例
2. Axios 进一步封装了ajax
3. 将页面上的布局先进行大块的分类；然后再就大块的组件进行细分
4. watch 可以监听$router
5. 函数跳转、声明跳转、对象跳转
6. query和params query在地址后面通过?拼接参数；params在路由中定义传入的参数
7. Webpack是一个前段工具，实现合并、打包、压缩、混淆等诸多功能
8. entry，output，devServer，plugins，module
9. css处理规则：css-loader style-loader；less: less、less-loader；sass: sass-loader、node-sass、sass；使用url-loader file-loader处理图片数据等