1:什么是mvvm?

> MVVM是Model-View-ViewModel的缩写。mvvm是一种设计思想。Model 层代表数据模型，也可以在Model中定义数据修改和操作的业务逻辑；View 代表UI 组件，它负责将数据模型转化成UI 展现出来，ViewModel 是一个同步View 和 Model的对象。

在MVVM架构下，View 和 Model 之间并没有直接的联系，而是通过ViewModel进行交互，Model 和 ViewModel 之间的交互是双向的， 因此View 数据的变化会同步到Model中，而Model 数据的变化也会立即反应到View 上。

ViewModel 通过双向数据绑定把 View 层和 Model 层连接了起来，而View 和 Model 之间的同步工作完全是自动的，无需人为干涉，因此开发者只需关注业务逻辑，不需要手动操作DOM, 不需要关注数据状态的同步问题，复杂的数据状态维护完全由 MVVM 来统一管理。

2:mvvm和mvc区别?

> mvc和mvvm其实区别并不大。都是一种设计思想。主要就是mvc中Controller演变成mvvm中的viewModel。mvvm主要解决了mvc中大量的DOM 操作使页面渲染性能降低，加载速度变慢，影响用户体验。和当 Model 频繁发生变化，开发者需要主动更新到View 。

2:vue的优点是什么?

1. **低耦合。视图（View）可以独立于Model变化和修改，一个ViewModel可以绑定到不同的"View"上，当View变化的时候Model可以不变，当Model变化的时候View也可以不变。**

2. **可重用性。你可以把一些视图逻辑放在一个ViewModel里面，让很多view重用这段视图逻辑。**

3. **独立开发。开发人员可以专注于业务逻辑和数据的开发（ViewModel），设计人员可以专注于页面设计，使用Expression Blend可以很容易设计界面并生成xml代码。**

4. 可测试。界面素来是比较难于测试的，而现在测试可以针对ViewModel来写。

3:请详细说下你对vue生命周期的理解?

> 答：总共分为8个阶段创建前/后，载入前/后，更新前/后，销毁前/后。

- 创建前/后： 在beforeCreate阶段，vue实例的挂载元素$el和数据对象data都为undefined，还未初始化。在created阶段，vue实例的数据对象data有了，$el还没有。
- 载入前/后：在beforeMount阶段，vue实例的$el和data都初始化了，但还是挂载之前为虚拟的dom节点，data.message还未替换。在mounted阶段，vue实例挂载完成，data.message成功渲染。
- 更新前/后：当data变化时，会触发beforeUpdate和updated方法。
- 销毁前/后：在执行destroy方法后，对data的改变不会再触发周期函数，说明此时vue实例已经解除了事件监听以及和dom的绑定，但是dom结构依然存在

3:组件之间的传值?

1:父组件与子组件传值

```
//父组件通过标签上面定义传值
<template>
    <Main :obj="data"></Main>
</template>
<script>
    //引入子组件
    import Main form "./main"

    exprot default{
        name:"parent",
        data(){
            return {
                data:"我要向子组件传递数据"
            }
        },
        //初始化组件
        components:{
```

```
            Main
        }
    }
</script>


//子组件通过props方法接受数据

<template>
    <div>{{data}}</div>
</template>
<script>
    exprot default{
        name:"son",
        //接受父组件传值
        props:["data"]
    }
</script>
```

2:子组件向父组件传递数据

```
//子组件通过$emit方法传递参数
<template>
   <div v-on:click="events"></div>
</template>
<script>
    //引入子组件
    import Main form "./main"

    exprot default{
        methods:{
            events:function(){

            }
        }
    }
</script>


//

<template>
    <div>{{data}}</div>
```

```
</template>
<script>
    exprot default{
        name:"son",
        //接受父组件传值
        props:["data"]
    }
</script>
```

4:路由之间跳转?

声明式（标签跳转）	编程式（ js跳转）
<router-link :to="index">	router.push('index')

5:组件的使用和自己创建公用组件?

第一步：在components目录新建你的组件文件（indexPage.vue），script一定要export default {}

第二步：在需要用的页面（组件）中导入：import indexPage from '@/components/indexPage.vue'

第三步：注入到vue的子组件的components属性上面,components:{indexPage}

第四步：在template视图view中使用，<index-page> </index-page>

问题有indexPage命名，使用的时候则index-page。

6:vue如何实现按需加载配合webpack设置?

webpack中提供了require.ensure()来实现按需加载。以前引入路由是通过import 这样的方式引入，改为const定义的方式进行引入。
不进行页面按需加载引入方式：`import home from '../../common/home.vue'`
进行页面按需加载的引入方式：`const home = r => require.ensure( [], () => r`

```
(require('../../common/home.vue')))
```

7:vuex是什么？怎么使用？哪种功能场景使用它?

> vue框架中状态管理。在main.js引入store，注入。新建了一个目录store，..... export 。场景有：单页应用中，组件之间的状态。音乐播放、登录状态、加入购物车