一、Vue生命周期

1.1 Vue生命周期都有哪些?

解答：系统自带了八个生命周期，分别是：

```
        beforeCreate
        created
  beforeMount
  mounted
  beforeUpdate
  updated
  beforeDestroy
  destroyed
```

1.3 一旦进入组件或者一旦进入页面，会执行哪些生命周期?

解答：会之前前面4个

```
        beforeCreate
        created
  beforeMount
        mounted
```

1.4 如果使用了keep-alive会多出来俩个生命周期

```
activated
deactivated
```

1.5 如果使用了keep-alive第一次进入组件会执行5个生命周期

```
beforeCreate
created
beforeMount
mounted
activated
```

1.6 如果使用了keep-alive第二次或者第N次，每次都会执行一个生命周期

`activated`

二、v-show和v-if是干什么？有什么区别?

2.1 v-show

显示和隐藏 ： `display:none`进行隐藏 、`display:block`进行显示

2.2 v-if

创建和删除：`remove`、`append`

2.3 区别：

显示和隐藏用：`v-show`
创建和删除用：`v-if`

频繁切换用：`v-show`
不频繁切换用：`v-if`

首次加载：用`v-if`，不用`v-show`
为什么：

如果用`v-if`可以没有这个盒子，然后再通过`v-if`进行创建（但是第一次进入页面是没有这
如果用`v-show`这个盒子不管是显示还是隐藏，该盒子都是在的（节点都是存在）

2.4 使用场景：

`v-show` ： 加入购物车、分享、蒙层这种都基本上用`v-show`
`v-if` ： 首页栏目切换的时候`v-if`

三、v-if和v-for 优先级

v-for的优先级要比v-if的优先级高

证明这个事情，是在vue.js源码种10997行

```
if (el.staticRoot && !el.staticProcessed) {
  return genStatic(el, state)
} else if (el.once && !el.onceProcessed) {
  return genOnce(el, state)
} else if (el.for && !el.forProcessed) {
  return genFor(el, state)
} else if (el.if && !el.ifProcessed) {
  return genIf(el, state)
} else if (el.tag === 'template' && !el.slotTarget && !state.pre) {
  return genChildren(el, state) || 'void 0'
} else if (el.tag === 'slot') {
  return genSlot(el, state)
} else {
```

注：v-if和v-for不要写在同一个节点上，这个性能很差。（v-if要写在父节点上）

四、ref

4.1 是什么?

获取dom

4.2 场景?

如果项目中使用插件，并且插件是要获取dom的，那么就可以使用ref了。

五、keep-alive

5.1 是什么?

缓存组件

5.2 一旦使用keep-alive会多俩个生命周期

activated
deactivated

5.3 功能

提升性能的

六、nextTick

6.1 是什么?

当dom更新完毕执行内部代码

6.2 场景

使用插件的时候会用到。例如new Swiper这个插件可能会获取当前元素的宽度或者高度，等dom都加载完毕再去获取宽度和高度就不会有任何问题了。

七、computed、methods、watch区别

computed：计算属性

可以监听某些数据的变化，并且有缓存。

如果一进入页面调用，就会触发

methods： 可以放入函数

没有缓存

如果一进入页面调用，就会触发

watch：监听（路由和数据）

当数据发生改变时，才会触发

可以得到现在的值和过去的值

八、Vue组件的通信（组件的传值）

8.1 父传子

```
父:
<HelloWorld :msg="str" />
<HelloWorld :msg="str" ></HelloWorld>

子:
props:['msg']
props: {
        msg: String,
},
```

8.2 子传父

```
子:
<button @click="changeParentName">改变父组件的name</button>
export default {
    methods: {
        //子组件的事件
        changeParentName: function() {
            this.$emit('handleChange', 'Jack') // 触发父组件中handleChange事件并传参Jack
            // 注：此处事件名称与父组件中绑定的事件名称要一致
        }
    }
}

父:
<child @handleChange="changeName"></child>
methods: {
    changeName(name) {  // name形参是子组件中传入的值Jack
        this.name = name
    }
}
```

8.3 兄弟组件传值

创建bus作为中转

```
import Vue from "vue";
export default new Vue;
```

A组件：

```
<button @click='btn'>HelloWorld按钮</button>
data () {
    return {
      hlStr:"这是helloWorld组件的数据"
    }
},
methods:{
    btn(){
      bus.$emit('selectItem',this.hlStr);
    }
}
```

B组件：

```
created(){
                bus.$on('selectItem',(val)=>{
                        console.log( val , 1111);
                })
}
```

九、slot插槽

使用场景：组件中有些地方的布局可能大多一致，但是细微有些小小变化

十、Vue路由的高频面试题

10.1 SPA单页面应用和传统页面跳转有什么区别?

SPA跳转是一个页面进行切换

传统页面跳转就是跳转不同的html了

SPA对于seo部分不是特别好，只能收录一个

传统的页面对于seo比较好，多个html文件收录

10.2 路径传值

显示：

```
传:
this.$router.push({
        path:'/about',
        query:{
          key:'你好'
        }
})
接:
this.$route.query
```

隐示:

```
传:
this.$router.push({
        name:'About',
        params:{
          key:'你好'
        }
})
接:
this.$route.params
```

10.3 路由的模式

```
mode: "history"   http://localhost:8080/about

mode:"hash"       http://localhost:8080/#/about
```

10.4 路由导航守卫（拦截、路由钩子函数）

全局

```
beforeEach
beforeResolve
afterEach
```

路由独享

```
beforeEnter
```

组件内

```
beforeRouteEnter
beforeRouteUpdate (2.2 新增)
beforeRouteLeave
```

场景：要去拦截，判断用户是否是登录状态。功能：进入地址管理，用户如果没有登录是进入不了地址管理（在进入之前判断拦截），需要先登录。

10.5 子路由、动态路由

子路由：children
动态路由：path: '/user/:id'

11、Vuex

11.1 Vuex有哪些部分构成

state、getters、mutations、actions、modules

11.2 什么场景用Vuex

共享、方便管理、方便维护、组件传值.....

项目：购物车数据，订单数据，用户的登录信息....

11.3 mutations和actions的区别

本质区别：

mutations	必须是同步函数
actions	“可以包含”任意异步操作

使用区别：mutations中可以放入函数，actions也可以放入函数，但是一般我们在mutations中放入函数而actions是提交mutations

12、双向绑定原理

通过Object.defineProperty劫持数据发生的改变，如果数据发生改变了（在set中进行赋值的），触发update方法进行更新节点内容（{{ str }}），从而实现了数据双向绑定的原理。

13、diff算法

功能：提升性能

虚拟dom ===》其实就是数据（ 把dom数据化 ）

主流：snabbdom、virtual-dom

snabbdom：https://www.npmjs.com/package/snabbdom

13.1 搭建环境

npm init -y

cnpm install webpack@5 webpack-cli@3 webpack-dev-server@3 -S

cnpm install snabbdom -S

新建webpack.config.js

配置webpack.config.js

13.2 虚拟节点 和 真实节点

虚拟节点：

```
{
  children: undefined
  data: {}
  elm: h1
  key: undefined
  sel: "h1"
  text: "你好h1"
}
```

真实节点：

```
<h2>你好</h2>
```

13.3 新老节点替换的规则

1、如果新老节点不是同一个节点名称，那么就暴力删除旧的节点，创建插入新的节点。

2、只能同级比较，不能跨层比较。如果跨层那么就暴力删除旧的节点，创建插入新的节点。

3、如果是相同节点，又分为很多情况

3.1 新节点有没有children

如果新的节点没有children，那就证明新节点是文本，那直接把旧的替换成新的文本

3.2 新节点有children

新的有children，旧的也有children ===》就是diff算法的核心了【3.3】

新的有children，旧的没有 ===》创建元素添加（把旧的内容删除清空掉，增加新的）

3.3 diff算法的核心（最复杂的情况）

1、 旧前 和 新前

匹配：旧前的指针++ 、 新前的指针++

2、 旧后 和 新后

匹配：旧后的指针-- 、 新后的指针--

3、 旧前 和 新后

匹配：旧前的指针++ 、 新后的指针--

4、 旧后 和 新前

匹配：旧后的指针-- 、 新前的指针++

5、 以上都不满足条件 ===》查找

新的指针++，新的添加到页面上并且新在旧的种有，要给旧的复制成undefined

6、 创建或者删除

***注意：如果要提升性能，一定要加入key，key是唯一标示，在更改前后，确认是不是同一个节点。

14、谈一下MVVM框架

web1.0时代

文件全在一起，也就是前端和后端的代码全在一起

问题：

1、前端和后端都是一个人开发。（技术没有侧重点或者责任不够细分）

2、项目不好维护。

3、html、css、js页面的静态内容没有，后端是没办法工作的（没办法套数据）。

mvc....都是后端先出的

web2.0时代

ajax出现了，就可以：前端和后端数据分离了。

解决问题：后端不用等前端页面弄完没，后端做后端的事情（写接口）、前端布局、特效、发送请求。

问题：

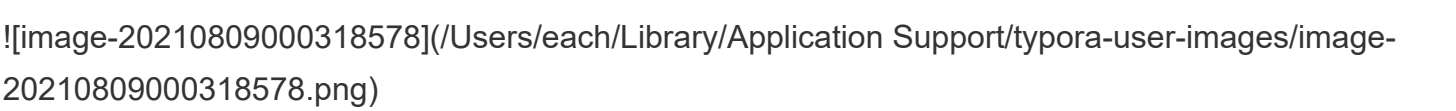
1、html、css、js都在一个页面中，单个页面可能内容也是比较多的（也会出现不好维护的情况）。

出现前端的框架了MVC、MVVM

解决问题：可以把一个“特别大”页面，进行拆分（组件化），单个组件进行维护

什么是MVVM

Model-View-的简写



view : 视图【dom==》在页面中展示的内容】

model：模型【数据层：vue中的data数据】

viewModel：视图模型层【就是vue源码】