1、vue组件传值

父传子prop

子传父通过this.$emit()触发父组件给的方法再通过方法传值（更像改值）

Vuex传值

中央事件总线 mitt

Provide和inject 跨级组件通信

$refs和$parent

2、vue-router

vue-router的原理是监听URL地址变化，从而渲染不同的组件

hash和history

**异：**

A、hash路径中的#，在刷新网页时，#号后面的信息不会提交给服务器

History路径为/ , /后面的信息会提交到服务器，刷新页面没有接口匹配会报404错误

B、hash通过hashChange监听路由地址变化，路由地址改变会根据当前路由地址找路由配置对应组件，再将组件显示在页面。Hash切换路由内部会创建hashHistory对象，再根据对象的push\replace方法切换页面，达到浏览器前进后退效果

history内部通过history.pushState()和history.replaceState()方法切换页面，实现浏览器历史效果

**同：**

都是前端路由实现方案，路由切换不刷新页面，不向服务器发送请求，页面的切换过程都是在浏览器本地通过js代码完成。

3、vue-cli工程用到哪些技术

1、vue.js

2、vue-router

3、vuex

4、axios

5、vux、 mint-UI、 AntDesign 、Element 等为vue所设计的UI组件库

6、webpack

7、AnimateCSS 动画库

4、vue-cli常用npm命令

1、npm run serve

2、npm run build

3、npm run lint

4、npm run build --report 用于查看 vue-cli 生产环境部署资源文件大小

5、vue-cli各文件作用

6、package.json配置

最重要的3个：

scripts：npm run xxx 命令调用node执行的 .js 文件

dependencies：生产环境依赖包的名称和版本号，这些依赖包会打包进生产环境的JS文件中

devDependencies：开发环境依赖包的名称和版本号，只在开发，我们敲代码的时候打包进JS

7、vue.js的核心

1、数据驱动

2、组件系统，vue组件的核心选项

1、模板(template)

2、初始数据(data)

3、接受的外部参数(props)

4、方法(methods)

5、生命周期钩子函数

6、私有资源：用户自定义指令和组件等

3、vuex 状态管理方案

8、vue是一套构建用户界面的渐进式框架

渐进式---不多做职责之外的事。最不可缺的是组件系统和双向数据绑定

Angular必须需要模块机制，依赖注入，特殊形式定义组件。排他性

React 必须理解 函数式编程，副作用，纯函数，隔离副作用，它是软性侵入

而Vue可以只用它的一部分功能。他只是一个轻量视图，只做最核心的部分就可以了

9、Vue常用指令

1、v-if

2、v-show

3、v-for

4、v-bind

5、v-on

6、v-model

7、v-pre 跳过编译过程

8、v-once 只编译一次

9、v-html 显示html

10、v-text 文本内容

11、v-cloak 解决页面初始化闪动

10、v-if和v-show的区别

同：动态显示DOM元素

异：

1. 编译过程：v-if真正的条件渲染，v-show切换display属性
2. 编译条件: v-if惰性，只有为真才会开始渲染条件块，v-show不论初始条件一定会渲染，显不显示则看条件
3. 性能消耗：v-if有更高的切换消耗，v-show有更高的初始渲染消耗
4. 应用场景：v-if适合条件很少改变时使用，v-show适合频繁切换

11、vue常见修饰符

<1>v-on常见修饰符

click keyup这些是事件名，不是修饰符，并且恰巧修饰符是用来修饰它们的

1、.stop 阻止冒泡

2、.prevent 阻止默认事件

3、.capture 事件侦听器，捕获

4、.self 绑定的元素本身

5、.{keycode | keyAlias} 特定键

6、.native 组件根元素原生事件

7、.once 只触发一次回调

8、.left 鼠标左

9、.right 右

10、.middle 中

当使用封装的组件或第三方UI库事件不起作用时，此时就应该使用native

@keyup.enter.native="searchFile(params)"

<2>v-bind指令修饰符

1、.prop 绑定DOM属性（property）

2、.camel kebab-case特性名转换为camelCase，短横线隔开改为驼峰式命名

3、.sync 语法糖，更新父组件绑定值

<3>v-model

1、.lazy 取代input监听change事件，只有当用户的input中失去焦点或者用户点击回车按钮时，才会将后台的数据进行修改。

2、.number 输入字符串转为数字

3、.trim 输入首尾空格过滤

12、v-on监听多个方法可以，一个事件也可以绑定多个方法

13、vue中key值的作用

Key用来管理可复用的元素，唯一标识符，更高效地渲染元素，在一个循环渲染过程中一般会复用已有元素而不是每次循环都从头开始渲染到指定部分。2.2.0+的版本，v-for必须要加key了

14、vue事件中使用event对象

注意：event在html绑定事件中要加$符号。$event入参。在js部分使用一个event可以不带$符号参数承接

15、$nextTick

$nextTick在它写的位置发生的下一次dom更新完毕之后再执行，因为vue是异步更新，可以用它知道DOM什么时候更新完成

16、Vue组件中data为什么要是函数

因为组件是可以共享的，同一个组件类可以创建很多个组件对象。但每一个组件对象的data应该是其私有的。所以每个组件return一个唯一的data对象，不和其他组件对象共享。

17、v-for和v-if的优先级

Vue2 中v-for高于v-if

但是但是在vue3中v-if优先级高于v-for，建议不要放在同一个元素上，在v-if外部套一层标签使用v-for

18、vue子组件调用父组件的方法

1、父组件声明并传递（写在组件名上）一个可供修改状态的自定义事件给子组件

2、子组件使用$emit触发当前实例上的自定义事件，同时用emits声明父组件传来的方法

19、vue中keep-alive的作用

保留组件状态避免组件重新渲染

标签上的属性有：include：字符串或正则表达式，只有匹配的才会被缓存

exclude：匹配的不被缓存

20、vue中如何编写可复用的组件

组件由状态props、事件events和嵌套的片段slots（插槽）组成。组件的可复用需要清晰的公开接口

props:组件当前的数据和属性，允许外部环境传递数据给组件

events:组件在特定时机触发操作的行为，允许组件触发外部环境的副作用

slots：嵌套在组件标签的内容，根据条件展现。允许外部环境将额外内容组合在组件中

21、vue生命周期

Vue实例从创建到销毁的一系列过程。统共六个过程

1、开始创建 创建

2、初始化数据 数据

3、编译模板 编译

4、挂载DOM->渲染 渲染

5、更新->渲染 更新

6、卸载 卸载

过程中会自行运行生命周期钩子函数

22、vue生命周期钩子函数

beforeCreate

created

beforeMount

mounted

beforeUpdate

updated

activated keep-alive激活

deactiveated keep-alive停用

beforeUnmount

unmounted

errorCaptured 捕获子孙组件的错误时调用

23、监听键盘事件的按键，即按键别名

1、.enter

2、.tab

3、.delete

4、.esc

5、.space

6、.up

7、.down

8、.left

9、.right

系统修饰键：

#1、.ctrl

#2、.alt

#3、.shift

#4、.meta meta键（我键盘没得）

#5、.exact 精确触发

鼠标按钮修饰符：

.left

.right

.middle

24、vue更新数组触发视图更新的方法

即会改变原始数组的数组方法

- push()

- pop()

- shift()

- unshift()

- splice()

- sort()

- reverse()

而这些就是返回新数组不修改原数组的方法,也就不会触发视图更新

filter(), concat()和 slice()

25、解决网速慢初始化页面闪动问题：v-cloak，

v-clock不需要表达式，它会在`Vue`实例结束编译时从绑定的HTML元素上移除，经常和CSS的`display:none`配合使用。

26、v-for产生的列表，如何实现active样式的切换

根据索引设置当前currentIndex，并绑定一个动态class属性，根据currentIndex和Index进行判断，二者相等时显示active样式

27、vue-cli工程中自定义一个过滤器

Vue3移除了过滤器，用计算属性或方法替换。但保留了全局过滤器

在main.js定义

//app.config.globalProperties 其实就是 Vue.prototype

app.config.globalProperties.$filters = {

currencyUSD(value) {

return '$' + value

}}

在vue实例调用，用过滤器属性包裹（状态数据）

<p>{{ $filters.currencyUSD(accountBalance) }}</p>

28、vue单页面应用的优缺点

单页面应用只有一个页面，SPA加载单个HTML

**优点：**

1、用户体验更好，具有应用程序的即时性，又有网站的可移植性和可访问性

2、页面内容改变不需要加载整个页面

3、页面切换不会出现白屏，假死或闪烁

4、对服务器的压力小，服务器只需要提供数据

5、良好的前后端分离

**缺点：**

1. 首屏加载慢
2. 不容易被搜索引擎搜到
3. 容易造成css命名冲突，因为是单页面，在一个html里
4. 前进、后退、地址栏、书签等需要程序管理，页面复杂度高

29、vue的计算属性

当前属性值根据其他属性通过一定计算得来的，

computed：{

花括号内部写计算属性函数，本质是一个函数，但在使用时当成普通属性调用不要加（）调用，要有return返回值。并且计算结果会被缓存，调用普通方法改变状态返回新值不会被缓存，每次使用都需要调用，而计算属性这里不需要

}

30、vue父组件向子组件传值

1、父组件用v-bind向子组件传变量值

2、子组件声明props接收父组件传来的数据

31、组件中使用全局变量

可以将全局变量配置到 app.config.globalProperties 中。

32、vue禁止弹窗后面的滚动条出现。关闭再现身

1.设置document的overflow为hidden

2.给document绑定touchmove事件，阻止默认事件

Stop：document.addEventListener("touchmove",mo,false);//禁止页面滑动

Move：document.removeEventListener("touchmove",mo,false); //取消滑动限制

33、计算属性的缓存和方法调用的区别

1、计算属性必须return返回结果

2、计算属性基于依赖缓存，所依赖的数据发生改变，它也会重新取值

3、使用计算属性还是methods取决于是否需要缓存，大计算用计算属性建议优先计算属性

4、就算属性根据依赖自动执行，methods需要调用，然后return返还数据

34、vue.js的自定义指令

内置指令有如v-if等。但还需要一些特殊功能时，对DOM进行底层操作时用到自定义指令。

app.directive('focus', {

在自定义指令内部写自定义钩子函数，实现js方法效果

beforeMount: function (el) {

// 每当指令绑定到元素上的时候，会立即执行这个beforeMount函数，只执行一次

35、自定义指令的钩子函数

Vue3中钩子函数的生命周期和组件的生命周期类似

- created - 元素创建后，但是属性和事件还没有生效时调用。

- beforeMount- 仅调用一次，当指令第一次绑定元素的时候。

- mounted- 仅调用一次，元素被插入父元素时调用.

- beforeUpdate: 在更新包含组件的 VNode 之前调用。

- updated - 在包含组件的 VNode 及其子组件的 VNode 更新后调用。

- beforeUnmount: 元素卸载前调用，仅调用一次

- unmounted -当指令卸载后调用，仅调用一次

36、自定义指令钩子函数参数

1、el：指令绑定的元素。除el其余参数应该是readyOnly

2、binding：数据对象，对象有如下属性：{

#1、instance 当前组件实例

#2、value 指令值

#3、oldValue 指令的前一个值

#4、arg 传给指令的参数，

#5、modifiers

3、vnode: 生成的虚拟节点

4、prevVNode 更新时的上一个节点

37、vue-router响应路由参数的变化，组件复用路由不响应

从`/content?id=1` 到 `content?id=2`路由参数发生变化，但组件实例默认复用，组件的生命周期钩子不会再被调用，

1、 :key阻止复用，加一个时间戳，使每一次路由跳转重新构建组件

<router-view :key="key"></router-view>

computed: {

key() {

return this.$route.name !== undefined? this.$route.name +new Date(): this.$route +new Date()

2、复用组件时，对路由参数变化作出响应，watch(监测变化) $route对象

watch: {

'$route' (to, from) {

// 对路由变化作出响应...

}

3、使用vue-router的钩子函数。三个参数

beforeRouteEnter (to, from, next) {

beforeRouteUpdate (to, from, next) {路由改变但组件复用即动态参数路径的页面间跳转

beforeRouteLeave (to, from, next) {

38、完整的vue-router导航解析流程

localhost:8000/#a ---> localhost:8000/#b 从A路由到B路由，路由导航八步

1、在A组件里调用离开守卫，即A组件的beforeRouteLeave

2、调用全局的beforeEach守卫，router.beforeEach

3、再执行写在router文件夹下的B路由配置中调用beforeEnter

4、再执行B组件的进入守卫 ，即B组件的beforeRouteEnter

5、调用全局的beforeResolve守卫 ，router.BeforeResolve

6、导航被确认

7、调用全局的afterEach钩子 router.afterEach

8、触发DOM更新

39、vue-router有哪几种路由钩子（守卫）

**1、全局守卫**：{

#1、router.beforeEach

#2、router.beforeResolve

#3、router.afterEach

2、**路由独享的守卫**

beforeEnter

3、**组件内的守卫**

beforeRouteEnter、beforeRouteUpdate、beforeRouteLeave

40、vue-router的几种实例方法及参数传递

**实例方法**：onComplete?,onAbort?分别是导航完成（(在所有的异步钩⼦被解析之后)）和中止（导航到相同的路由、或在当前导航完成之前导航到另⼀个不同的路由)的回调函数，

location可以解构为path，query，params等

this.$router.push(location,onComplete?,onAbort?)

this.$router.replace(location,onComplete?,onAbort?)

this.$router.go(n)

**参数传递方式：**

动态路由参数，params，query，meta

params必须和name搭配使用，和path一起使用不起效果。写成对象格式

query 查询参数，写出对象格式

meta：路由元信息，写在路由配置里面的，也是对象格式

**组件获取路由参数：**

this.$route.params

this.$route.query

this.$route.meta

41、$route和$router的区别

$route是路由信息对象，包括path，query等一系列路由参数信息。

而$router是路由实例对象，包括了跳转方法，钩子函数等。

42、vue-router的动态路由匹配和使用

{ path: '/user/:id', component: User }，路径后跟冒号

/可以在路由配置的时候写上props:true把动态路由参数注入到组件的props中，但冒号：依然不能少，只是可以不用$route.params.id也能获取动态路由参数了

43、vue-router定义嵌套路由

1、父组件留一个<router-view />给子组件，

2、在路由配置中要使用children配置。即在父路由的大括号下写一个children数组对象。

然后在数组对象里接着写子路由对象

44、<router-link></router-link>组件及其属性

用于用户在具有路由功能的应用中点击跳转导航，类似于a标签的样式

常用属性：

to

replace

append

tag

active-class

exact

event

exact-active-class

45、vue-router实现动态加载路由组件（懒加载）

在打包构建应用时，js包会变得很大，影响页面加载，此时把不同路由对应的组件分割成不同的代码块，当路由被访问时才加载对应组件。这就是路由懒加载

这里可以结合Vue异步组件和Webpack的代码分割功能来实现路由懒加载。共有三步：

1. 定义一个能够被Webpack自动代码分割的异步组件。import的右侧用箭头函数的写法把组件赋值给左侧。这也就是动态引入路由

const index = () => import('../page/list/index.vue');

2、路由配置照旧

3、在build/webpack.base.conf.js下的output属性，新增chunkFilename:’[name].js’

46、vuex

Vuex是专为vue.js开发的状态管理器，采用集中式存储管理所有组件状态

Vuex生成一个store实例，并把这个实例挂载到所有组件，所有组件引用同一个store实例，一个组件改变store其他组件收到的状态也会发生改变

47、vuex核心概念

Vuex核心是store，相当于一个仓库存储应用的各种状态

1. state 定义共享状态变量
2. action 向store发出调用通知执行本地或远端的操作，store的methods
3. mutations 修改器修改state，action调用mutations，但可以忽略action这一步直接使用mutations，但触发动作还是有的。
4. getter 读取器，store的计算属性，不改变原有的state

48、vuex异步修改

vuex异步action

49、Promise对象

Promise本身不异步，但Promise中可以有异步任务

Promise接收一个函数作参数，这个函数是立刻执行的，该函数有两个参数，resolve和reject这是两个异步执行的函数了

Promise对象实例方法：

.then 指定调用成功后的回调函数 .then返回一个新的Promise实例。后面还可以再跟另一个then方法，实现链式操作

.catch 用于指定发生错误的回调

50、axios、fetch与ajax

**1、Ajax**

核心是使用XMLHttpRequest，最早的后端发送请求的技术，如果多个XHR有先后关系就会出现回调地狱的问题；

缺点：

1、针对MVC编程，不符合现在的MVVM软件架构

2、基于原生XHR开发，XHR本身架构不清晰

3、为了AJAX单独引入JQuery不合理

4、回调地狱

5、不符合关注分离的原则

**#2、axios**

基于Promise的请求库，用于浏览器和node.js中，本质上浏览器的axios是对原生XHR的封装，不过它是Promise的实现版本

特征：

//1、从浏览器中创建XHR

//2、支持PromiseAPI

//3、客户端支持防止CSRF，（跨站请求伪造）

//4、提供了并发接口

//5、从node创建http请求

//6、拦截请求和响应

//7、转换请求和响应

//8、取消请求

//9、自动转换JSON数据

**#3、fetch**

ES6出现的，基于Promise设计的，fetch没有使用XHR对象

优点：

1、符合关注分离，没有将输入、输出和用事件来跟踪的状态混杂在一个对象里

2、更好更方便的写法

3、基于Promise实现，支持async和await

4、更加底层，提供的API丰富（request，reponse）

5、脱离了XHR