**1、active-class是哪个组件的属性？嵌套路由怎么定义？**

答; vue-router模块的router-link组件。 在某个路由配置它的children属性。

**2、怎么定义vue-router的动态路由？怎么获取传过来的动态参数？**

答：在router目录下的index.js文件中，对path属性加上/:id。 使用router对象的params.id。

**3、vue-router有哪几种导航钩子？**

答：三种。vue-router 提供的导航钩子主要用来拦截导航，让它完成跳转或取消。

一种是**全局导航钩子**:

beforEach、afterEach

const router = new VueRouter({ ... })

router.beforeEach((to, from, next) => {

// do something

next();

});

router.afterEach((to, from) => { //后置钩子并没有 next 函数，也不会改变导航本

// do something //身

});

每个钩子方法接收三个参数：

\*\*to: Route，即将要进入的目标 [路由对象]

\*\*from: Route，当前导航正要离开的路由[路由对象]

\*\*next: Function，一定要调用该方法来 resolve 这个钩子。执行效果依赖 next方法调用的参数。

①next()：进入管道中的下一个钩子，如果全部的钩子执行完了，则导航的状态就是 confirmed（确认的）

②next(false)：这代表中断掉当前的导航，即 to 代表的路由对象不会进入，被中断，此时该表 URL 地址会被重置到 from 路由对应的地址

③next(‘/’) 和 next({path: ‘/’})：在中断掉当前导航的同时，跳转到一个不同的地址

④next(error)：如果传入参数是一个 Error 实例，那么导航被终止的同时会将错误传递给 router.onError() 注册过的回调

next 方法必须要调用，否则钩子就不会被 resolved。

第二种：**单个路由独享的钩子**:

beforeEnter 是在路由配置上直接进行定义的

cont router = new VueRouter({

routes: [

{

path: '/file',

component: File,

beforeEnter: (to, from ,next) => {

// do someting

}

}

]

});

第三种：**组件内的钩子**:

beforeRouteEnter

beforeRouteUpdate (2.2 新增)

beforeRouteLeave

const File = {

template: `<div>This is file</div>`,

beforeRouteEnter(to, from, next) {

// do someting

// 在渲染该组件的对应路由被 confirm 前调用

},

beforeRouteUpdate(to, from, next) {

// do someting

// 在当前路由改变，但是依然渲染该组件是调用

},

beforeRouteLeave(to, from ,next) {

// do someting

// 导航离开该组件的对应路由时被调用

}

}

注意：beforeRouteEnter 不能获取组件实例 this，因为当守卫执行前，组件实例被没有被创建出来，剩下两个钩子则可以正常获取组件实例 this.

**4、stylus是什么？安装使用的步骤是？ 有哪些特征？**

答：Stylus 是一个CSS的预处理框架，2010年产生，来自Node.js社区，主要用来给Node项目进行CSS预处理支持，所以 Stylus 是一种新型语言，可以创建健壮的、动态的、富有表现力的CSS。

Stylus默认使用 .styl 的作为文件扩展名，支持多样性的CSS语法。

Stylus功能上更为强壮，和js联系更加紧密。

cnpm install stylus --save-dev

cnpm install stylus-loader --save-dev

stylus的特征：

冒号可选

分号可选

逗号可选

括号可选

变量

插值

混合书写

算术

强制类型转换

动态导入

条件

迭代

嵌套选择

父级参考

变量函数调用

词法作用域

内置函数(>25)

内部语言函数

压缩可选

图像内联可选

可执行Stylus

健壮的错误报告

单行和多行注释

CSS字面量

字符转义

TextMate捆绑

**5、mint-ui是什么？怎么使用？说出至少三个组件使用方法？**

答：基于vue的前端组件库。

npm安装，然后import样式和js，vue.use（mintUi）全局引入。在单个组件局部引入：import {Toast} from ‘mint-ui’。

组件一：Toast(‘登录成功’)；组件二：mint-header；组件三：mint-swiper

**6、axios是什么？怎么使用？描述使用它实现登录功能的流程？**

答：请求后台资源的模块。npm install axios -S装好，然后发送的是跨域，需在配置文件中config/index.js进行设置。后台如果是Tp5则定义一个资源路由。js中使用import进来，然后.get或.post。返回在.then函数中如果成功，失败则是在.catch函数中。

**7、vuex是什么？怎么使用？哪种功能场景使用它？**

答：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 。

**8、说出至少4种vue当中的指令和它的用法？**

答：v-if：判断是否隐藏；v-for：数据循环出来；v-bind:class：绑定一个属性；v-model：实现双向绑定

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。

**9、vue-router是什么？它有哪些组件？**

答：vue用来写路由一个插件。router-link、router-view。

**10、自定义指令（v-check、v-focus）的方法有哪些？它有哪些钩子函数？还有哪些钩子函数参数？**

答：全局定义指令：在vue对象的directive方法里面有两个参数，一个是指令名称，另外一个是函数。

组件内定义指令：directives

钩子函数：bind（绑定事件触发）、inserted(节点插入的时候触发)、update（组件内相关更新）

钩子函数参数：el、binding。

**11、Vue的双向数据绑定原理是什么？**

答：vue.js 是采用数据劫持结合发布者-订阅者模式的方式，通过Object.defineProperty()来劫持各个属性的setter，getter，在数据变动时发布消息给订阅者，触发相应的监听回调。

具体步骤：

第一步：需要observe的数据对象进行递归遍历，包括子属性对象的属性，都加上 setter和getter

这样的话，给这个对象的某个值赋值，就会触发setter，那么就能监听到了数据变化

第二步：compile解析模板指令，将模板中的变量替换成数据，然后初始化渲染页面视图，并将每个指令对应的节点绑定更新函数，添加监听数据的订阅者，一旦数据有变动，收到通知，更新视图

第三步：Watcher订阅者是Observer和Compile之间通信的桥梁，主要做的事情是:

1、在自身实例化时往属性订阅器(dep)里面添加自己

2、自身必须有一个update()方法

3、待属性变动dep.notice()通知时，能调用自身的update()方法，并触发Compile中绑定的回调，则功成身退。

第四步：MVVM作为数据绑定的入口，整合Observer、Compile和Watcher三者，通过Observer来监听自己的model数据变化，通过Compile来解析编译模板指令，最终利用Watcher搭起Observer和Compile之间的通信桥梁，达到数据变化 -> 视图更新；视图交互变化(input) -> 数据model变更的双向绑定效果。

**12、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beforeCreated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。

**挂载前/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，创建了el节点 可以操作DOM了。

**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。

**13、vue-loader是什么？使用它的用途有哪些？**

答：解析.vue文件的一个加载器，跟template/js/style转换成js模块。

用途：js可以写es6、style样式可以scss或less、template可以加jade等。

**13、请说出vue.cli项目中src目录每个文件夹和文件的用法？**

答：assets文件夹是放静态资源；

components是放组件；

router是定义路由相关的配置;

App.vue是一个应用主组件；

main.js是入口文件

**14、对Vue.js的template编译的理解？**

答：简而言之，就是先转化成AST树，再得到的render函数返回VNode（Vue的虚拟DOM节点）

详情步骤：

首先，通过compile编译器把template编译成AST语法树（abstract syntax tree 即 源代码的抽象语法结构的树状表现形式），compile是createCompiler的返回值，createCompiler是用以创建编译器的。另外compile还负责合并option；

然后，AST会经过generate（将AST语法树转化成render funtion字符串的过程）得到render函数，render的返回值是VNode，VNode是Vue的虚拟DOM节点，里面有（标签名、子节点、文本等等）。

**15、Vue底层实现原理**

答：Vue是一个典型的MVVM框架，模型（Model）只是普通的JavaScript对象，修改它则视图（View）会自动更新。

Vue.js 最核心的功能有两个，一是响应式的数据绑定系统，二是组件系统。

vue.js 则是采用数据劫持结合发布者-订阅者模式的方式，通过Object.defineProperty()来劫持各个属性的setter，getter，在数据变动时发布消息给订阅者，触发相应的监听回调。

Vue实现数据双向绑定的效果，需要三大模块：

（1）Observer：能够对数据对象的所有属性进行监听，如有变动可拿到最新值并通知订阅者；

（2）Compile：对每个元素节点的指令进行扫描和解析，根据指令模板替换数据，以及绑定相应的更新函数；

（3）Watcher：作为连接Observer和Compile的桥梁，能够订阅并收到每个属性变动的通知，执行指令绑定的相应回调函数，从而更新视图。

Observer

Observer的核心是通过Obeject.defineProperty()来监听数据的变动，这个函数内部可以定义setter和getter，每当数据发生变化，就会触发setter。这时候Observer就要通知订阅者，订阅者就是Watcher。

Compile

Compile主要做的事情是解析模板指令，将模板中的变量替换成数据，然后初始化渲染页面视图，并将每个指令对应的节点绑定更新函数，添加监听数据的订阅者，一旦数据有变动，收到通知，更新视图。

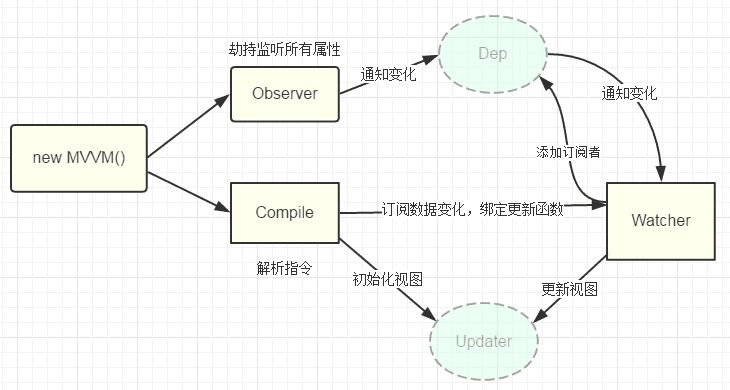
Watcher

Watcher订阅者作为Observer和Compile之间通信的桥梁，主要做的事情是：

（1）在自身实例化时往属性订阅器(dep)里面添加自己

（2）自身必须有一个update()方法

（3）待属性变动dep.notice()通知时，能调用自身的update()方法，并触发Compile中绑定的回调。



var obj = { };

// 为obj定义一个名为 hello 的访问器属性

Object.defineProperty(obj, "hello",{

set:function(){},

get:function(){}

})



get 和 set 方法内部的 this 都指向 obj，这意味着 get 和 set 函数可以操作对象内部的值。另外，访问器属性的会"覆盖"同名的普通属性，因为访问器属性会被优先访问，与其同名的普通属性则会被忽略。

**16、vue2组件懒加载的两种方法**

（1）resolve => require([URL], resolve), 支持性好

（2）() => import(URL), webpack2官网推荐使用, 属于es7范畴, 需要配合babel的syntax-dynamic-import插件使用。

export default new Router({

routes: [

{

path: '/login',

name: 'login',

component: resolve=>require(['@/components/ login.vue'], resolve)

},

{

path: '/register',

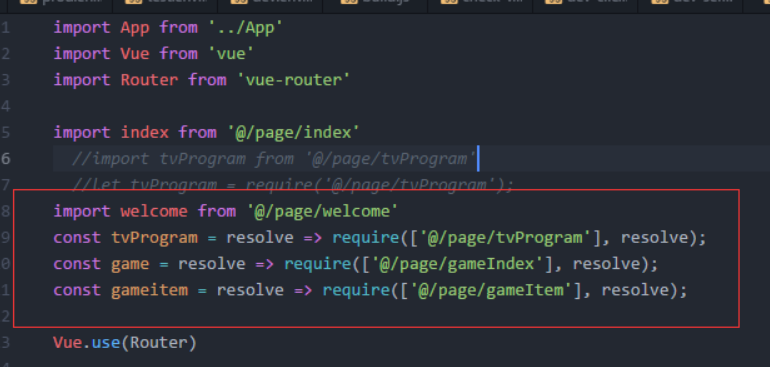
name: ' register',

component: () => import('@/components/ register.vue') //懒加载

},

]

})；





**17、vue父子组件传值**

(1)父组件向子组件传值：通过props属性来实现；

v-bind:dataOfChild="dataOfParent"(父组件)====>props:['dataOfChild'](子组件)

(2) 子组件向父组件传值：通过this.$emit()方法来实现；

vue是禁止子组件直接向父组件传值的，所以只能通过触发事件来达到目的。

this.$emit('parentFunction','helloworld')(子组件)====>

v-on:parentFunction="customParentFunction"(父组件)====>

customParentFunction:function (data) {}(父组件)

**18、谈谈你对MVVM开发模式的理解**

MVVM分为Model、View、ViewModel三者。

Model：代表数据模型，数据和业务逻辑都在Model层中定义；

View：代表UI视图，负责数据的展示；

ViewModel：负责监听Model中数据的改变并且控制视图的更新，处理用户交互操作；

Model和View并无直接关联，而是通过ViewModel来进行联系的，Model和ViewModel之间有着双向数据绑定的联系。因此当Model中的数据改变时会触发View层的刷新，View中由于用户交互操作而改变的数据也会在Model中同步。

这种模式实现了Model和View的数据自动同步，因此开发者只需要专注对数据的维护操作即可，而不需要自己操作dom。

**19、v-show和v-if 有什么区别？**

v-show 仅仅控制元素的显示方式，将 display 属性在 block 和 none 来回切换；而v-if会控制这个 DOM 节点的存在与否。当我们需要经常切换某个元素的显示/隐藏时，使用v-show会更加节省性能上的开销；当只需要一次显示或隐藏时，使用v-if更加合理。

20、Vue中如何监控某个属性值的变化？

计算属性和监听器

<div id="demo">{{ fullName }}</div>

var vm = new Vue({

el: '#demo',

data: {

firstName: 'Foo',

lastName: 'Bar',

fullName: 'Foo Bar'

},

watch: {

firstName: function (val) {

this.fullName = val + ' ' + this.lastName

},

lastName: function (val) {

this.fullName = this.firstName + ' ' + val

}

}

})

上面代码是命令式且重复的。将它与计算属性的版本进行比较：

var vm = new Vue({

el: '#demo',

data: {

firstName: 'Foo',

lastName: 'Bar'

},

computed: {

fullName: function () {

return this.firstName + ' ' + this.lastName

}

}

})