VUE面试题

1. 怎么做一个论坛的评论（不用组件）？

难点在于数据如何保存，结合vue评论组件

2.vue基础组件和业务组件分别包括什么

3.vue组件通信

父组件向子组件传值：

props

<template id="aaa">

<h2>我是父组件->{{msg}}</h2>

<bbb :m="msg"></bbb>

</template>

<script>

window.onload=function(){

var vm=new Vue({

el:"#box",

components:{

'aaa':{

data(){

return {

msg:'子组件你好'

}

},

template:"#aaa",

components:{

'bbb':{

props:{

'm':String

},

template:"<h3>我是子组件->{{m}}</h3>"

}

}

}

}

})

}

</script>

子组件向父组件传值：

过程：子组件执行add --> 触发$emit --> 触发父组件increment --> 执行 incrementTotal 方法

{{total}}

<mime @increment="incrementTotal"></mime>

<template id="myInput">

<button @click="add">{{counter}}</button>

</template>

...

<script>

....

data:{

tatal: 0

},

methods:{

incrementTotal:function(){

this.total +=1;

}

},

components:{

data : function(){

return:{

counter : 0

}

},

methods : {

add : function(){

this.counter +=1;

this.$emit('increment'); //子组件通过 $emit触发父组件的方法 increment 还可以传参 this.$emit('increment' ,this.counter);

}

}

}

</script>

非父子组件间的传值：

简单场景用bus（中间件），复杂场景用vuex

<div id="app4">

<display></display>

<increment></increment>

</div>

<script>

var bus = new Vue();

Vue.component('increment', {

template: `<button @click="add">+</button>`,

data: function () {

return {count: 0}

},

methods: {

add: function () {

bus.$emit('inc', this.count+=1)

}

}

});

Vue.component('display', {

template: `<span>Clicked: <mark>{{c}}</mark> times</span>`,

data: function () {

return {c: 0}

},

created: function () {

var self=this;

// bus.$on('inc', function (num) {

// self.c = num

// });

bus.$on('inc', (num) =>

this.c = num

);

}

});

vm = new Vue({

el: "#app4",

})

</script>

4.vue路由嵌套

<!DOCTYPE html>

<html>

<head>

<title></title>

<meta charset="utf-8">

<script src="http://unpkg.com/vue/dist/vue.js"></script>

<script src="https://unpkg.com/vue-router/dist/vue-router.js"></script>

</head>

<body>

<div id="box">

<p>

<!--一级路由-->

<router-link to="/home">home</router-link>

<router-link to="/news">news</router-link>

</p>

<router-view></router-view>

</div>

<!-- 模板抽离出来 -->

<template id="home">

<!-- 注意：组件只能有一个根元素，所以我们包装到这个div中 -->

<div>

<h2>首页</h2>

<!--二级路由-->

<router-link to="/home/login">登录</router-link>

<router-link to="/home/reg">注册</router-link>

<!-- 路由匹配到的组件将渲染在这里 -->

<router-view></router-view>

</div>

</template>

<template id="news">

<div>新闻</div>

</template>

<template id="login">

<div>登录界面</div>

</template>

<template id="reg">

<div>注册界面</div>

</template>

<script type="text/javascript">

// 1. 定义（路由）组件。

const Home = { template: '#home' };

const News = { template: '#news' };

const Login = { template: '#login' };

const Reg = { template: '#reg' };

// 2. 定义路由

const routes = [

{ path: '/', redirect: '/home' },

{

path: '/home',

component: Home,

children:[

{ path: '/home/login', component: Login},

{ path: '/home/reg', component: Reg}

]

},

{ path: '/news', component: News}

]

// 3. 创建 router 实例，然后传 `routes` 配置

const router = new VueRouter({

routes // （缩写）相当于 routes: routes

})

// 4. 创建和挂载根实例。

// 记得要通过 router 配置参数注入路由，

// 从而让整个应用都有路由功能

const app = new Vue({

router

}).$mount('#box')

// 现在，应用已经启动了！

</script>

</body>

</html>

5.vue-router有哪几种导航钩子

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。有多种方式可以在路由导航发生时执行钩子：全局的, 单个路由独享的, 或者组件级的。

**全局钩子：**

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

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

// do something

next();

});

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

console.log(to.path);

});

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  
**next: Function**: 一定要调用该方法来 **resolve** 这个钩子。执行效果依赖 next  
方法的调用参数。  
next(): 进行管道中的下一个钩子。如果全部钩子执行完了，则导航的状态就是**confirmed** （确认的）。  
next(false): 中断当前的导航。如果浏览器的 URL 改变了（可能是用户手动或者浏览器后退按钮），那么 URL 地址会重置到 from  
路由对应的地址。  
next('/') 或者 next({ path: '/' }): 跳转到一个不同的地址。当前的导航被中断，然后进行一个新的导航。

**确保要调用 next方法，否则钩子就不会被 resolved。**

**单独路由独享的钩子：**

你可以在路由配置上直接定义 beforeEnter 钩子：

const router = new VueRouter({

routes: [

{

path: '/foo',

component: Foo,

beforeEnter: (to, from, next) => { // ... }, beforeEnter: (route) => {

// ...

}

} ]

});

这些钩子与全局 before 钩子的方法参数是一样的。

**组件内的钩子：**

最后，你可以在路由组件内直接定义以下路由导航钩子：

beforeRouteEnter   
beforeRouteUpdate (2.2 新增)   
beforeRouteLeave

const Foo = {

template: `...`,

beforeRouteEnter (to, from, next) {

// 在渲染该组件的对应路由被 confirm 前调用

// 不！能！获取组件实例 `this`

// 因为当钩子执行前，组件实例还没被创建

},

beforeRouteUpdate (to, from, next) {

// 在当前路由改变，但是该组件被复用时调用

// 举例来说，对于一个带有动态参数的路径 /foo/:id，在 /foo/1 和 /foo/2 之间跳转的时候，

// 由于会渲染同样的 Foo 组件，因此组件实例会被复用。而这个钩子就会在这个情况下被调用。

// 可以访问组件实例 `this`

},

beforeRouteLeave (to, from, next) {

// 导航离开该组件的对应路由时调用

// 可以访问组件实例 `this`

}

}

beforeRouteEnter 钩子不能访问 this，因为钩子在导航确认前被调用,因此即将登场的新组件还没被创建。

不过，你可以通过传一个回调给 next来访问组件实例。在导航被确认的时候执行回调，并且把组件实例作为回调方法的参数。

beforeRouteEnter (to, from, next) {

next(vm => {

// 通过 `vm` 访问组件实例

})

}

你可以 在 beforeRouteLeave 中直接访问 this。这个 leave 钩子通常用来禁止用户在还未保存修改前突然离开。可以通过 next(false) 来取消导航。

6.vue组件动态连接

7.ployfill代码冗余

Polyfill你可以理解为“腻子”，就是装修的时候，可以把缺损的地方填充抹平。

对于一些代码，不同浏览器，不同版本，有些支持，有些不支持。

有些人就写对应的Polyfill（Polyfill有很多），帮你把这些差异化抹平，不支持的变得支持了（简单来讲，写些代码判断当前浏览器有没有这个功能，没有的话，就写一些支持的补丁代码）。

你只需要把需要的Polyfill引入到你的程序里，就可以了。

在项目进行时，vue-cli 为我们自动添加了 babel-plugin-transform-runtime 这个插件，**异步加载组件时，会产生 polyfill 代码冗余。**这是因为babel-plugin-transform-runtime 采用了**沙箱机制**来编译我们的代码（即：不修改宿主环境的内置对象）。

由于异步组件最终会被编译为一个单独的文件，所以即使多个组件中使用了同一个新特性（例如：Object.keys()），那么**在每个编译后的文件中都会有一份该新特性的 polyfill 拷贝。**如果项目较小可以考虑不使用异步加载，但是首屏的压力会比较大。

8.你的项目为什么选用vue而不是React

<http://www.cnblogs.com/inshua/p/6794862.html>

9.把vue低版本的项目迁移到高版本，有没有用过迁移工具

使用vue的版本更新了，里面有指令语法发生了变化，在1.0的习惯上用2.0，有些容易出错，并且不容易察觉，那么迁移工具就显得特别有用，让你事半功倍，帮助你查找哪里不符合。

步骤如下：1.安装  npm install --global vue-migration-helper

      2.在文件所在路径下，在cmd中运行vue-migration-helper

结果如图，方便快捷搜索版本错误：

