Vue.js 前端开发实战

简答题

目录

课本

第一章

1. 请简述什么是 Vue。

Vue（读音/Vju:/，类似于 View）是一套用于构建用户界面的渐进式框架，与其他大型框架相比，Vue 被设计为可以自底向上逐层应用。其他大型框架往往一开始就对项目的技术方案进行强制性的要求，而 Vue 更加灵活，开发者既可以选择使用 Vue 来开发一个全新项目，也可以将 Vue 引入到一个现有的项目中。

2. 请简述 Vue 优势有那些。

轻量级：Vue 相对简单、直接，所以 Vue 使用起来更加友好。

数据绑定：Vue 是一个 MVVM 框架，即数据双向绑定。

指令：指令主要包括内置指令和自定义指令，以“v-”开头，作用于 HTML 元素。

插件： 插件用于对 Vue 框架功能进行扩展，通过 MyPlugin.install 完成插件的编写，简单配置后就可以全局使用。

第二章

3. 请简述什么是 Vue 实例对象。

在 Vue 项目中，每个 Vue 应用都是通过 Vue 构造器创建新的 Vue 实例开始的。

通过 new 关键字的方式创建 vm 实例对象。

创建方式：

```
<script>
var vm = new Vue({
  // 选项
})
<script>
```

其中，配置选项的主要内容及含义：

1）.data：Vue 实例数据对象

2）.methods：定义 Vue 实例中方法

3）.components：定义子组件

4）.computed：计算属性

5）.filters：过滤器

6）.el：唯一根元素

7）.watch：监听数据变化

4. 请简述什么是 Vue 组件化开发。

1）.在 Vue 中，组件是构成页面中独立结构单元，能够减少重复代码的编写

2）.提高开发效率，降低代码之间的耦合程度，使项目更易维护和管理

3）.组件主要以页面结构形式存在，不同组件也具有基本交互功能，根据业务逻辑实现复杂的项目功能

5. 请简单介绍 Vue 内置指令主要内容有哪些。

1）.v-model：双向数据绑定

2）.v-on：监听事件

3）.v-bind：单向数据绑定

4）.v-text：插入文本内容

5）.v-html：插入包含 HTML 的内容

6）.v-for：列表渲染

7）.v-if：条件渲染

8）.v-show：显示隐藏

第三章

1. 请简述什么是 Vue 插件。

Vue.use 主要用于在 Vue 中安装插件，通过插件可以为 Vue 添加全局功能。插件可以是一个对象或函数，如果是对象，必须提供 install()方法，用来安装插件；如果插件是一个函数，则该函数将被当成 install()方法。

2. 请简述 Vue 全局 API 接口主要内容。

1）.Vue.directive():Vue 中有很多内置指令，如 v-model、v-for 和 v-bind 等

2）.Vue.use():Vue.use 主要用于在 Vue 中安装插件，通过插件可以为 Vue 添加全局功能

3）.Vue.extend():Vue.extend 用于基于 Vue 构造器创建一个 Vue 子类，可以对 Vue 构造器进行扩展

4）.Vue.set():Vue 的核心具有一套响应式系统，简单来说就是通过监听器监听数据层的数据变化，当数据改变后，通知视图也自动更新

5）.Vue.mixin():Vue.mixin 用于全局注册一个混入，它将影响之后创建的每个 Vue 实例

3. 请简单介绍 Vue 实例对象属性和方法。

1）.vm.$props：使用 vm.$props 属性可以接收上级组件向下传递的数据

2）.vm.$options：Vue 实例初始化时，除了传入指定的选项外，还可以传入自定义选项

3）.vm.$el：vm.$el 用来访问 vm 实例使用的根 DOM 元素

4）.vm.$children：vm.$children 用来获取当前实例的直接子组件

5）.vm.$root：vm.$root 用来获取当前组件树的根 Vue 实例，如果当前实例没有父实例，则获取到的是该实例本身

6）.vm.$slots:插槽就是定义在组件内部的 template 模板，可以通过$slots 动态获取

7）.vm.$attrs：vm.$attrs 可以获取组件的属性，但其获取的属性中不包含 class、style 以及被声明为 props 的属性

第四章

4. 请简述 JavaScript 钩子函数包括哪些。

入场钩子分别是 beforeEnter（入场前）、enter（入场）、afterEnter（入场后）和 enterCancelled（取消入场）

出场钩子分别是 beforeLeave（出场前）、leave（出场）、afterLeave（出场后）和 leaveCancelled（取消出场）

```
<transition
  @before-enter="beforeEnter"
  @enter="enter"
  @after-enter="afterEnter"
  @enter-cancelled="enterCancelled"
  @before-leave="beforeLeave"
  @leave="leave"
  @after-leave="afterLeave"
  @leave-cancelled="leaveCancelled"
  v-bind:css="false">  // Vue 会跳过 CSS 的检测
</transition>
```

5. 请简述 6 个内置的过渡类名有哪些。

进入（enter）：

v-enter：在元素被插入之前生效，在元素被插入之后的下一帧移除

v-enter-active： 在整个进入过渡的阶段中应用，在元素被插入之前生效，在过渡动画完成之后移除

v-enter-to： 在元素被插入之后下一帧生效（与此同时 v-enter 被移除），在过渡动画完成之后移除

离开（leave）：

v-leave：在离开过渡被触发时立刻生效，下一帧被移除

v-leave-active：在整个离开过渡的阶段中应用，在离开过渡被触发时立刻生效，在过渡完成之后移除

v-leave-to：在离开过渡被触发之后下一帧生效（与此同时 v-leave 被移除），在过渡动画完成之后移除

6. 请简述自定义过渡类名的属性有哪些。

```
enter-class
enter-active-class
enter-to-class
leave-class
leave-active-class
leave-to-class
```

注意：自定义类名的优先级高于普通类名

第五章

7. 请简述 npm 方式安装 vue-router 的步骤。

```
1.首先通过 cd 命令进入创建好的项目目录里面
cd 文件名
2.使用以下 npm 命令来安装路由
方式一：npm install vue-router --save（不加版本号）
// --save 会在 package.json 包配置文件中添加对应的配置
```

方式二：npm install vue-router@3.1.x（指定版本号）

安装完成之后可以在 package.json 文件中查看到 vue-router 的相关信息

3.在 main.js 文件中引入路由、安装路由功能等，示例代码如下

```
import Vue from 'vue'
import VueRouter from 'vue-router' // 引入插件
import App from './App'
Vue.use(VueRouter)                    // 注册组件
const router = new VueRouter({        // 创建实例
    routes:[]                         // 配置路由规则
})
const app = new Vue({
    el: '#app',
    router:router,                    // 挂载路由
    render:h=>h(App)
})
```

8. 请简述 vue-router 路由的作用。

根据不同的 url 哈希值，在路由视图中显示不同的页面，实现非跳转式的页面切换

在单页面应用中更新视图可以不用重新请求页面

用户体验好，不需要每次都从服务器全部获取，快速展现给用户

9. 请简单列举并说明路由对象包括哪些属性。

路由对象表示当前激活的路由的状态信息，包含了当前 URL 解析得到的信息，还有 URL 匹配到的路由记录，this.$router 表示全局路由器对象，this.$route 表示当前正在用于跳转的路由器对象，$route 的常用属性信息如下：

$route.path：对应当前路由地址

$route.query：一个{key:value}对象，表示 URL 查询参数

$route.params：一个{key:value}对象，路由转跳携带参数

$route.hash：在 history 模式下获取当前路由的 hash 值（带#），如果没有 hash 值，则为空字符串

$route.fullPath：完成解析后的 URL，包含查询参数和 hash 的完整路径

$route.name：当前路由的名称

$route.matched：路由记录，当前路由下路由声明的所有信息，从父路由（如果有）到当前路由为止

$route.redirectedFrom：如果存在重定向，即为重定向来源的路由

第六章

10. 请简要概述 Vuex 的设计思想。

Vuex 是 Vue 团队提供的一套组件状态管理维护的解决方案。Vuex 作为 Vue 插件来使用，进一步完善了 Vue 基础代码功能，使 Vue 组件状态更加容易维护，为大型项目开发提供了强大的技术支持。

11. 简述 Vuex 配置对象中的主要内容有哪些。

1）.actions：用来定义事件处理方法，用于处理 state 数据

2）.mutations：选项中的事件处理方法接收 state 对象作为参数，即初始数据

3）.getters:store 实例允许在 store 中定义 getters 计算属性,类似于 Vue 实例的 computed

4）.modules：modules 用来在 store 实例中定义模块对象

5）.plugins：Vuex 中的插件配置选项为 plugins，插件本身为函数

6）.devtools：store 实例配置中的 devtools 选项用来设置是否在 devtools 调试工具中启用 Vuex，默认值为 true，表示在启用，设为 false 表示停止使用

12. 简述 Vuex 中的 actions 的含义。

actions 选项用来定义事件处理方法，用于处理 state 数据。actions 类似于 mutations，不同之处在于 actions 是异步执行的，事件处理函数可以接收{commit}对象，完成 mutation 提交，从而方便 devtools 调试工具跟踪状态的 state 变化。

在使用时，需要在 store 仓库中注册 actions 选项，在里面定义事件处理方法。事件处理方法接收 context 作为第 1 个参数，payload 作为第 2 个参数（根据需要进行选择）。

第八章

13. 请简述什么是服务器端渲染。

服务器端渲染（简称 SSR），是将组件或页面通过服务器生成 html 字符串，再发送到浏览器，最后将静态标记"混合"为客户端上完全交互的应用程序，简单理解就是将页面在服务器中完成渲染，然后在客户端直接展示。

14. 请简述服务器端渲染的代码逻辑和处理步骤。

Vue 进行服务器端渲染时，需要利用 Node.js 搭建一个服务器，并添加服务器端渲染的代码逻辑。

使用 webpack-dev-middleware 中间件对更改的文件进行监控，使用 webpack-hot-middleware 中间件进行页面的热更新，使用 vue-server-renderer 插件来渲染服务器端打包的 bundle 文件到客户端。

15. 请简述 Nuxt.js 中，声明式路由和编程式路由的区别

声明式路由：在页面中使用<nuxt-link>完成路由跳转。

编程式路由：在 JavaScript 代码中实现路由的跳转。

传智播客

第一章

1、请介绍什么是 MVVM

MVVM 主要包含 3 个部分，分别是 Model、View 和 ViewModel。
Model 指的是数据部分，主要负责业务数据；
View 指的是视图部分，即 DOM 元素，负责视图的处理；
ViewModel 是连接视图与数据的数据模型，负责监听 Model 或者 View 的修改；

2、请简单描述 Vue 的优势有哪些

①轻量级：Vue 简单、直接，简单易学
②数据绑定：数据驱动视图，视图驱动数据
③指令：指令绑定在元素上，给绑定元素添加行为
④插件：用于对 Vue 框架功能进行扩展

指令：指令主要包括内置指令和自定义指令，以“v-”开头，作用于 HTML 元素；
轻量级：AngularJS 的学习成本高，使用起来比较复杂，而 Vue 相对简单、直接，所以 Vue 使用起来更加友好；
数据绑定：Vue 是一个 MVVM 框架，即数据双向绑定，即当数据发生变化的时候，视图也就发生变化，当视图发生变化的时候，数据也会跟着同步变化，这也算是 Vue 的精髓之处，尤其是在进行表单处理时，Vue 的双向数据绑定非常方便；
插件：插件用于对 Vue 框架功能进行扩展，通过 MyPlugin.install 完成插件的编写，简单配置后就可以全局使用。常用的扩展插件有 vue-router、Vuex 等；

3、请简单描述什么是 npm

npm 提供了快速操作包的命令，只需要简单命令就可以很方便地对第三方包进行管理。

4、请介绍什么是 webpack

webpack 是一个模块打包工具，可以把前端项目中的 js、cs、scss/less、图片等文件都打包在一起，实现自动化构建。

5、Web 前端开发使用的三大语言是什么

1.HTML 页面结构;
2.CSS 页面样式;
3.JavaScript 行为;

6、请简单描述 vue.js 下载和引入的基本步骤

①下载 vue.js：
通过 vue.js 官网网址来下载 vue.js。进入网页（https://vuejs.org/v2/guide/installation.html）后，点击 Development Version 进行下载。
②引入 vue.js：
<script src="vue.js"></script>

1.从 Vue 官方网站可以获取下载地址;
2.当在 HTML 网页中使用 Vue 时，使用<script>标签引入 vue.js 即可;

第二章

1、请简单描述 computed 计算属性的基本用法。

Vue 提供了一种更通用的方式来观察和响应 Vue 实例上的数据变动，当有一些数据需要随着其他数据变动而变动时，就需要使用 computed 计算属性。
在事件处理方法中，this 指向的 Vue 实例的计算属性结果会被缓存起来，只有依赖的响应式属性变化时，才会重新计算，返回最终结果。

2、请简述 Vue 钩子函数有哪些以含义

beforeCreate：创建实例对象之前执行；
created：创建实例对象之后执行；
beforeMount：页面挂载成功之前执行；
mounted：页面挂载成功之后执行；
beforeUpdate：组件更新之前执行；
updated：组件更新之后执行；
beforeDestroy：实例销毁之前执行；
destroyed：实例销毁之后执行；

3、请简述什么是组件

在 Vue 中，组件是构成页面中独立结构单元，能够减少重复代码的编写，提高开发效率，降低代码之间的耦合程度，使项目更易维护和管理。
组件主要以页面结构形式存在，不同组件也具有基本交互功能，根据业务逻辑实现复杂的项目功能。

4、请简述 Vue 的内置指令有哪些构成

v-model：双向数据绑定
v-on：监听事件
v-bind：单向数据绑定
v-text：插入文本内容
v-html：插入包含 HTML 的内容
v-for：列表渲染
v-if：条件渲染
v-show：显示隐藏

5、请简单描述 filters 过滤器的功能

在前端页面开发中，通过数据绑定可以将 data 数据绑定到页面中，页面中的数据经过逻辑层处理后展示最终的结果。
数据的变化除了在 Vue 逻辑层进行操作外，还可以通过过滤器来实现。

第三章

1、请简述 Vue.extend()的含义并创建一个 Vue 的子类

Vue.extend 用于基于 Vue 构造器创建一个 Vue 子类，可以对 Vue 构造器进行扩展。它有一个 options 参数，表示包含组件选项的对象。
示例代码：
<script>

```
var Vue2 = Vue.extend({
   data () {
     return { title: 'hello' }
   }
 })
 var vm2 = new Vue2()
console.log(vm2.title) // 输出结果：hello
</script>
```

其中，vm2 是通过实例化 Vue2 类创建的实例对象。

2、请简述 Vue 全局配置对象中的 productionTip 的含义

当在网页中加载了 vue.js（开发版本）文件时，浏览器的控制台会出现英文的提示信息，提醒用户“您正在开发模式下运行 Vue，为生产部署时，请确保打开生产模式”。

3、请简述$options 含义并设置自定义选项 myName 的值为“我是自定义选项”

Vue 实例初始化时，除了传入指定的选项外，还可以传入自定义选项。自定义选项的值可以是数组、对象、函数等，通过 vm.$options 来获取。

示例代码：

```
<script src = "vue.js"></script>
<div id="app"></div>
<script>
var vm = new Vue({
  el:'#app',
  customOption: '我是自定义选项',
})
console.log(vm.$options.customOption); //输出结果为：'我是自定义选项'
</script>
```

4、请简述 vm.$slots 的含义并实现在 my-component 组件中引入一个名为“demo”的插槽

Vue 中的组件中使用 template 模板定义 HTML 结构，为了方便使用 template 公共模板结构，Vue 提出了插槽的概念，插槽就是定义在组件内部的 template 模板，可以通过$slots 动态获取。

```
  <div id="app">
    <my-component>
      <template v-slot:demo>
        <div>
          我是 demo
        </div>
      </template>
    </my-component>
  </div>
  <template id="first">
    <div>
      <slot name="demo"></slot>
    </div>
  </template>
   <script>
```

```
    Vue.component('my-component', { template: '#first' })
    //创建 Vue 实例,得到 ViewModel
    var vm = new Vue({
      el: '#app',
      data: {},
      methods: {}
    })
     </script>
```
执行上述代码后，在页面中展示“我是 demo”。

5、请简述 vm.$attrs 的含义并实现动态的获取组件的 name 属性

vm.$attrs 可以获取组件的属性,但其获取的属性中不包含 class、style 以及被声明为 props 的属性。

示例代码:

```
<div id="app">
    <my-component name="test"></my-component>
  </div>
<script>
Vue.component('my-component', {
  template: '<div></div>',
  created () {        console.log(this.$attrs.name)
 }
  })
 var vm = new Vue({ el: '#app' })
 </script>
```
执行上述代码后，在控制台打印‘test’。

第四章

1、请使用@keyframes 实现元素显示隐藏的动画效果且动画开始是红色，50%变为绿色、100%变为橙色的效果。

```
<style>
  div.circular {
    width: 100px; height: 100px; background: red;
    border-radius: 50%; margin-top: 20px; text-align: center;
    line-height: 100px; color: #fff;
 }
 .bounce-enter-active {
   animation: Ami .5s;
 }
 .bounce-leave-active {
   animation: Ami .5s;
 }
 @keyframes Ami {
```

```
    0% {background-color: red;}
   50% {background-color: orange;}
   100% {background-color:green;}
 }
</style>                                                    <script
src="https://unpkg.com/vue/dist/vue.js"></script>
<body>
  <div id="app">
    <button @click='flag=!flag'>显示/隐藏</button>
    <transition name="bounce">
      <div class="circular" v-if="flag"></div>
    </transition>
  </div>
  <script>
    var vm = new Vue({
      el: '#app',
      data: {
        flag:"true"
      },
    });
  </script>                                    </body>
```

2、请列举 transition 组件的类名有哪些及含义。

v-enter 进入过渡的开始状态，作用于开始的一帧
v-enter-active 进入过渡生效时的状态，作用于整个过程
v-enter-to 进入过渡的结束状态，作用于结束的一帧
v-leave 离开过渡的开始状态，作用于开始的一帧
v-leave-active 离开过渡生效时的状态，作用于整个过程
v-leave-to 离开过渡的结束状态，作用于结束的一帧

3、请简单描述,<transition>组件过渡模式有哪些及含义分别是什么。

在 transition 中加入 mode 属性，它有两个值，分别是 in-out 和 out-in，out-in 表示当前元素先进行过渡，完成之后新元素过渡进入，in-out 表示新元素先进行过渡，完成之后当前元素过渡离开。

4、请简述列表交错过渡的实现过程。

在 Vue 中还可以实现列表的交错过渡效果，它是通过 data 属性与 JavaScript 通信来实现的。

1.引入 js 库

```
<script src="https://unpkg.com/vue/dist/vue.js"></script>
    <script
src="https://cdnjs.cloudflare.com/ajax/libs/velocity/1.5.0/velocity
.min.js"></script>
```

2.编写 HTML 结构；

```
 <div id="app">
   <input placeholder="请输入要查找的内容" v-model="query">
    <transition-group      name="item"      tag="ul"      @before-
```

```
enter="beforeEnter"
    @enter="enter" @leave="leave" v-bind:css="false">
      <li v-for="(item, index) in ComputedList" :key="item.msg"
       :data-index="index">
        {{ item.msg }}
      </li>
    </transition-group>
  </div>
```

3.编写 JavaScript 代码；

```
  var vm = new Vue({
    el: '#app',
  data () {
      return {
        query: '',     // v-model 绑定的值
        items: [
         { msg: '张三' }, { msg: '李四' }, { msg: '张芳芳' },
          { msg: '王琳琳' }, { msg: '冯圆' }
        ]
      }
   },
  computed: {  // 计算属性
     ComputedList () {
       var vm = this.query    // 获取到 input 输入框中的内容
        var nameList = this.items  // 数组
        return nameList.filter(function (item) {
         return
item.msg.toLowerCase().indexOf(vm.toLowerCase()) !== -1
        })
      }
  },
   methods: {
     beforeEnter (el) {
       el.style.opacity = 0
      el.style.height = 0
    },
     enter (el, done) {
      var delay = el.dataset.index * 150
        setTimeout(function () {
         Velocity(el, {opacity: 1, height: '1.6em'}, {complete:
done})
       }, delay)
      },
     leave (el, done) {
       var delay = el.dataset.index * 150
```

```
        setTimeout(function () {
          Velocity(el, {opacity: 0, height: 0}, {complete: done})
        }, delay)
    }
  }
})
```

5、请简单描述 filters 过滤器的功能

在前端页面开发中，通过数据绑定可以将 data 数据绑定到页面中，页面中的数据经过逻辑层处理后展示最终的结果。
数据的变化除了在 Vue 逻辑层进行操作外，还可以通过过滤器来实现。

6、请简述怎么使用 Velocity.js 库结合钩子函数实现元素的显示隐藏动画效果。

```
引入 js 库
<script src="https://unpkg.com/vue/dist/vue.js"></script>
<script
src="https://cdnjs.cloudflare.com/ajax/libs/velocity/1.5.0/velocity.min.js">
</script>
编写 css 样式
<style>
      .circular{
        width: 100px;
        height: 100px;
        border-radius: 50px 50px;
        background-color: red;
      }
</style>
<body>
  <!-- 示例代码 -->
  <div id="app">
    <!-- Velocity 动画使用 -->
    <button @click="show=!show">单击按钮显示隐藏</button>
    <transition     @before-enter="beforeEnter"     @enter="enter"
@leave="leave" v-bind:css="false">
      <div class="circular" v-if="show"></div>
    </transition>
  </div>
  <script>
    //创建 Vue 实例,得到 ViewModel
    var vm = new Vue({
      el: '#app',
      data: {
        show: "true"
      },
      methods: {
        beforeEnter(el) {
```

```
            el.style.opacity = 0     // 透明度为0
            el.style.color = 'red'   // 颜色为红色
          },
          enter(el, done) {          // duration动画执行时间
            Velocity(el, {opacity: 1}, {duration: 300})
          },
          leave(el, done) {
            Velocity(el, {opacity: 0}, {duration: 300})
          }
        }
      });
</script>
</body>
```

7、请简述什么是过渡和动画。

过渡就是从一个状态向另外一个状态插入值，新的状态替换了旧的状态；动画相比过渡来说，可以在一个声明中设置多个状态，当然动画也可以实现过渡的效果，只需要从开始到结束插入状态即可。

第五章

1、简述如何在项目中安装Less预处理器。

第一步： npm install less less-loader -D
第二步：在webpack.config.js文件中添加rules规则 { test: /\.less$/, use: ['style-loader', 'css-loader', 'less-loader'] },
第三步：在页面中使用 less 的地方给<style>添加 lang 属性即可 <style lang="less"></style>

2、简述如何在项目中使用history模式

第一步：修改router.js文件，添加history模式

```
var router = new VueRouter({
  mode: 'history',
}
```

第二步：在module.exports对象中添加devServer的配置

```
devServer: {
historyApiFallback: true // 开启服务器对history模式支持
},
```

3、讲述<template id="contact-tmp"></template>的含义

定义id为contact-tmp的模板组件，渲染contact组件模板的内容

4、简述什么是命名视图

在开发中，有时候想同时或同级展示多个视图，而不是嵌套展示，则可以在页面中定义多个单独命名的视图。例如，创建一个布局，有 header（头部区域）、sidebar（侧导航区域）和 mainBox（主体区域）3个视图，这时候就可以使用命名视图来实现。

5、简述声明式导航和编程式导航的区别

通过<router-link>来完成页面的切换，这种方式属于声明式导航。为了更方便地在项目中开发导航功能，Vue 提供了编程式导航，也就是利用 JavaScript 代码来实现地址的跳转，通过 router 实例方法来实现。

6、简述如何添加自定义 class，实现路由导航的高亮效果。

第一步：在 router.js 路由文件中，找到创建路由实例代码，添加自定义 class

```
var router = new VueRouter({
  linkActiveClass: 'my-active',   linkExactActiveClass: 'my-exact-active',
})
```

第二步：在 App.vue 中添加高亮效果的样式

```
<style lang="scss" scoped>
  .my-active, .my-exact-active {
    background: #007aff;
    font-weight: 800;
    color: #fff;
  }
</script>
```

7、简述如何在项目中安装 Sass/SCSS 预处理器

第一步： npm install sass-loader node-sass -D

第二步：在 webpack.config.js 文件中添加 rules 规则 { test: /\.less$/, use: ['style-loader', 'css-loader', 'sass-loader'] },

第三步：在页面中使用 sass 的地方给<style>添加 lang 属性即可 <style lang="scss"></style>

8、简述如何在项目中安装 Stylus 预处理器

第一步：npm install stylus stylus-loader -D

第二步：在页面中使用 Stylus 的地方给<style>添加 lang 属性即可

```
<style lang="stylus"></style>
```

9、简述什么是命名路由

vue-router 提供了一种隐式的引用路径，可以在创建 Router 实例的时候，在 routes 中给某个路由设置名称 name 值，执行一些跳转的时候，可以通过路由的名称取代路径地址。

第六章

1、请简单描述,Vuex 的下载和安装方式以及步骤。

直接通过<script>标签引入 vuex.js 文件：

1.从 Vue 官方网站可以获取 vuex.js 文件并下载到本地；

2.通过<script>标签引入；

通过 npm 导入 vuex 包；

1.去官网下载 Node 安装包，解压后安装到本地；

2.令行工具执行 npm install vue-cli -g 安装脚手架工具；

3.执行 vue init webpack demo02 命令创建 demo02 项目；

4.打开 demo02 项目，执行 npm install vuex@3.1.1 --save 命令安装 vuex 依赖包；

2、请描述购物车案例的实现过程

主要实现思路是：

案例分析：
购物车案例是在线商城中的基本功能之一，可以实现将顾客想要购买的商品添加到购物车，计算购物车中商品的总价格。
主要由两个页面组成，分别是“商品列表页面”和“购物车”页面。
代码实现：
1.初始化项目；
2.实现底部 Tab 栏切换；
3.获取商品数据 ；
4.实现商品列表页面；
5.实现购物车页面；

3、请使用 actions 实现添加列表功能

```
<!-- 页面结构 -->
  <div id="app">
    <button @click = "addList">添加列表</button>
    <ul>
      <li v-for = "item,key in list" :id = 'key'>{{item}}</li>
    </ul>
  </div>
  <script>
    // 实例化 store
    const store = new Vuex.Store({
      state: {
        list: ['列表 1','列表 1','列表 1']
      },
      mutations:{
        addList(state){
          state.list.push('列表 1')
        }
      },
      actions: {
        addList({commit}){
          commit('addList')
        }
      }
    })
    // 实例化 vm 实例
    var vm = new Vue({
      el: '#app',
      data: {
        id:'',
        list: []
      },
      created(){
        this.list = this.$store.state.list
```

```
    },
    methods: {
      addList() {
        this.$store.dispatch('addList')
      }
    },
    store
  })
</script>
```

4、请使用 getters 对数组中每个元素进行求和计算并返回

```
<!-- 页面结构 -->
<div id="app">
  <button @click="sum">求数组的和</button>
  <div>{{numSum}}</div>
</div>
<script>
  // 实例化 store
  const store = new Vuex.Store({
    state: {
      arr: [1, 2, 3, 4, 5, 6, 3]
    },
    getters: {
      sum: (state) => {
        return state.arr.reduce(function getSum(total, num) {
          return total + num;
        })
      }
    }
  })
  // 实例化 vm 实例
  var vm = new Vue({
    el: '#app',
    data: {
      numSum: 0
    },
    methods: {
      sum() {
        this.numSum = this.$store.getters.sum
      }
    },
    store
  })
</script>
```

5、请介绍 Vuex 单向数据流的主要构成部分以及含义

Vue 的单向数据流增强了组件之间的独立性，但是存在多个组件共享状态的时候，单向数据流状态就会被破坏。
主要构成部分：
State：驱动应用的数据源；
View：以声明方式将 state 映射到视图；
Actions：响应在 View 上的用户输入导致的状态变化；

6、请使用 mutations 实现单击页面的“更新”按钮，实现页面更新

```
<!--页面结构-->
<div id="app">
    <button @click = "update">页面更新</button>
    <div>{{newName}}</div>
  </div>
  <script>
    const store = new Vuex.Store({
      state: {
        name: '初始状态'
      },
      mutations:{
        update(state,payload){
          state.name = payload
        }
      }
    })
    var vm = new Vue({
      el: '#app',
      data: {
        name: 'vue'
      },
      computed:{
        newName(){
          return this.name = this.$store.state.name
        }
      },
      methods: {
        update(){
          this.$store.commit('update','页面已更新')
        }
      },
      store
    })
  </script>
```

7、请简述什么是 Vuex

Vuex 是 Vue 团队提供的一套组件状态管理维护的解决方案。
Vuex 作为 Vue 插件来使用，进一步完善了 Vue 基础代码功能，使 Vue 组件状态更加容易

维护，为大型项目开发提供了强大的技术支持。
通过 new Vuex.Store({})实例化创建 store 实例对象。

第七—九章

1、简述如何实现动态设置页面头部标题效果

在 router.js 文件中，为每个路由添加 meta 属性的页面 title，示例代码如下：

```
    routes: [
     { path: '/', redirect: '/home', name: 'home', meta: { title: '首页' } },
     { path: '/home', component: HomeCon, meta: { title: '首页' } },
     ...
],
```

在 App.vue 组件中使用如下代码获取 meta 数据，示例代码如下

```
<mt-header fixed :title="$route.meta.title"></mt-header>
```

在 main.js 文件中的路由钩子函数里获取 meta 数据，示例代码如下

```
router.beforeEach((to, from, next) => {
  // 路由发生改变修改页面 title
  if (to.meta.title) {
    document.title = to.meta.title
  }
  next() // 确保要调用 next 方法
})
```

2、简述<template>标签的作用

Vue 提供了<template>标签来定义结构的模板，可以在该标签中书写 HTML 代码，然后通过 id 值绑定到组件内的 template 属性上。

3、简述服务器端渲染的不足

第一点：服务器端压力增加　　第二点：涉及构建设置和部署的要求

4、简述什么是 vue-server-renderer 模块及其作用

vue-server-renderer 是 Vue 中处理服务器加载的一个模块，给 Vue 提供在 Node.js 服务器端渲染的功能。vue-server-renderer 依赖一些 Node.js 原生模块，所以目前只能在 Node.js 中使用。

5、简述 npm 和 cnpm 的区别。

npm 即 node.js 包管理工具的全称为 node.js package manager，cnpm 为淘宝镜像，一般同步频率为 10 分钟一次。cnpm 与 npm 使用语法相同，区别在于服务器不同。

6、简述一个项目或者产品的开发流程

1、产品创意
2、产品原型
3、美工设计
4、前端实现
5、后端实现
6、测试、试运行、上线
前端工程师主要专注第 4 步的前端代码实现，其他步骤了解即可。

7、简述如何使用 GUI 创建 vue 项目

第一步：使用 mkdir vue-ui（项目名）命令，创建一个名称为 vue-ui 的项目目录
第二步：执行 cd vue-ui 进入项目中，
第三步：执行 vue ui 命令，执行完毕后，默认会启动一个本地服务，在浏览器打开 localhost:8000 网址，会出现一个 Vue 项目管理器（中文），说明搭建成功，在该界面中需要用户根据项目需求去手动创建并选择配置。

8、简述什么是生产 / 开发依赖。

生产依赖：项目运行时需要的依赖包
开发依赖：项目构建打包时需要的依赖包

9、简述什么是 Nuxt.js 框架。

Nuxt.js 是一个基于 Vue.js 的轻量级应用框架，可用来创建服务端渲染应用，也可充当静态站点引擎生成静态站点应用，具有优雅的代码结构分层和热加载等特性。

10、简述使用 Vue CLI 3 创建项目的方法步骤。

步骤如下：
打开命令行工具，切换到项目根目录，执行以下指令来创建项目：
vue create hello-vue（项目名）
在交互界面中，选择手动配置项，进行配置
项目创建完成后，执行以下命令进去项目目录：
cd hello-vue
执行命令，启动项目
npm run server

11、简述什么是服务器端渲染。

服务器端渲染，顾名思义就是将页面或者组件通过服务器生成 HTML 字符串，将它们直接发送到浏览器，最后将静态标记“混合”为客户端上完全交互的应用程序。

12、讲述 delete 和 Vue.delete 删除数组的区别。

delete 只是被删除的元素变成了 empty/undefined 其他的元素的键值还是不变。
Vue.delete 直接删除了数组，改变了数组的键值。

13、简述如何实现新闻资讯详情页面的数据获取和展示。

请求接口，获取页面数据。
1、在 data 函数中定义空数组，用来存放接口返回数据

```
import { Toast } from 'mint-ui'
export default {
  data () {
    return { newslist: [] }  // 存放列表数据
  },
}
```

2、在 methods 函数中定义获取列表数据的方法 getNewsList()

```
methods: {
  getNewsList () {
    this.$http.get('接口 API').then(result => {
      // 请求成功处理的代码块
    })
    .catch(function (error) {
      // 请求失败处理的代码块
```

```
    })
  }
}
```

3、在 created()钩子函数中调用 getNewsList()方法

```
created () {
  this.getNewsList()
},
```

4、在页面中使用 v-for 进行数据的循环遍历，注意在跳转到详情页时，需要把 id 传递过去，用来区分页面内容

```
<li v-for="item in newslist" :key="item.id">
  <router-link :to="'/home/newsinfo/'+item.id"></router-link>
</li>
```

14、简述使用代码演示父组件向子组件传值。

```
子组件:
<template>
 <h2>{{msg}}</h2> // msg 必须是父组件传递的
</template>
<script>
  export default (){
     props:["msg"]   // 可以是数组，也可以是对象
  }
</script>
 父组件:  // 动态传值,titleVar 是变量
<template>
  <child   :msg = "titleVar"></child>
</template>
<script>
 import Child from '../components/child.vue' // 引入子组件
 export default (){
    components: {Child}, // 注册子组件
    data(){ titleVar :'你好' }
  }
</script>
```

答案说明:

父组件向子组件传值，使用 props 属性

15、简单描述 Vue CLI 3 安装的过程。

步骤如下：以 npm 包管理器为例

推荐使用 Node 8.11.0+和 NPM 3+

安装版本要求:

Node.js 8.11.0+

NPM 3+

如果之前已经全局安装了旧版的 vue-cli（1.x 或 2.x），需要先进行卸载，指令如下:

npm uninstall vue-cli -g

如果之前没有全局安装旧版，则直接全局安装@vue/cli 脚手架，指令如下:

npm install @vue/cli –g
vue –V 查看版本号

16、简述如何安装 vuetify 第三方 UI 插件。

第一步：在项目中，执行 vue add vuetify 命令进行安装。 执行上述命令之后，程序会提示安装选项，使用默认值即可 。
第二步：安装完成后，会在 src 目录里创建一个 plugins 目录，里面会自动生成关于插件的配置文件。

17、简述如何使用 Nuxt.js 脚手架创建项目。

方式一：npx　　npx create-nuxt-app（项目名）
方式二：yarn　　yarn create-nuxt-app（项目名）

18、简述什么是生产 / 开发环境。

生产环境：项目运行时需要的环境
开发环境：项目构建打包时需要的环境

19、简述什么是 Koa 框架。

Koa 是一个基于 Node.js 平台的 Web 开发框架，致力于成为 Web 应用和 API 开发领域更富有表现力的技术框架。

20、简述什么是客户端渲染。

客户端渲染，即传统的单页面应用（SPA）模式，Vue.js 构建的应用程序默认情况下是一个 HTML 模板页面，只有一个 id 为 app 的 根容器，然后通过 webpack 打包生成 css、js 等资源文件，浏览器加载、解析来渲染 HTML

21、简述什么是 Mint UI 框架。

Mint UI 是基于 Vue.js 的移动端组件库，使用 Vue 技术封装出来了成套的组件，可以无缝的和 Vue 项目进行集成开发。

22、简述常用的实现服务器端渲染的方式有哪些。

第 1 种：手动进行项目的简单搭建，
第 2 种：使用 Vue CLI 3 脚手架进行搭建，
第 3 种：是利用一些成熟框架来搭建（如 Nuxt.js）。

23、简述如何实现 SPA 类型的项目。

SPA 就是单页面应用程序，主要依靠路由来实现，路由根据不同的值来展示不同的组件。

24、简述服务器端渲染对 Vue 相关插件版本要求有哪些。

需要的最低 Vue 相关插件版本如下:
vue & vue-server-renderer 2.3.0+
vue-router 2.5.0+
vue-loader 12.0.0+ & vue-style-loader 3.0.0+

25、简述 webpack 服务器端渲染的基本流程。

webpack 将这 entry-server.js 和 entry-client.js 两个入口文件分别打包成给服务器端用的 Server Bundle 和给客户端用的 Client Bundle。当服务器接收到了来自客户端的请求之后，会创建一个 Bundle Renderer 渲染器，这个渲染器会读取 Server Bundle 文件，并且执行它的代码，然后发送一个生成好的 HTML 到浏览器。

26、简述单独路由的配置。

1、使用 npm 方式为项目安装 vue-router
　npm install vue-router --save
2、在 src 目录下，创建单独的路由文件 router.js。

```
import Vue from 'vue'
import VueRouter from 'vue-router'
Vue.use(VueRouter)
var router = new VueRouter({ })  // 创建路由实例对象 router
export default router      // 暴露路由对象属性
3、在 src/main.js 入口文件中引入 router.js 文件。
import router from './router.js'
new Vue({
  router,
  render: h => h(App)
}).$mount('#app')
```

27、简述图片预览插件 vue-preview 的安装与导入。

```
1、安装 vue-preview 插件
npm install vue-preview --save
2、在 main.js 文件导入插件
import VuePreview from 'vue-preview'
Vue.use(VuePreview)
```

28、简述使用代码演示子组件向父组件传值。

```
父组件示例代码:
<template>
 <div>
  <h1>{{title}}</h1>
  <child @getMessage="showMsg"></child>
 </div>
</template>
<script>
 import Child from '../components/child.vue'
 export default {
  components: {Child},
  data(){
   return{
    title:''
   }
  },
  methods:{
   showMsg(title){
    this.title=title;
   }
 }
 }
</script>
 子组件示例代码:
<template>
 <h3>我是子组件</h3>
```

```
</template>
<script>
 export default {
   mounted: function () {
     this.$emit('getMessage','我是父组件！')// 触发当前实例上的事件 getMessage，并把“我是父组件”传递给父组件中。
   }
 }
</script>
```

答案说明:

子组件向父组件传值，使用$emit 触发父组件的自定义事件

29、简述 Nuxt.js 中，声明式路由和编程式路由的区别。

声明式路由：在页面中使用<nuxt-link>完成路由跳转。
编程式路由：在 JavaScript 代码中实现路由的跳转。

30、代码实现在服务器脚本文件 test.js 中将 Vue 实例的渲染结果输出到控制台。

```
// ① 创建一个 Vue 实例
const Vue = require('vue')
const app = new Vue({
  template: '<div>SSR 的简单使用</div>'
})
// ② 创建一个 renderer 实例
const renderer = require('vue-server-renderer').createRenderer()
// ③ 将 Vue 实例渲染为 HTML
renderer.renderToString(app, (err, html) => {
  if (err) {
    throw err
  }
  console.log(html)
})
```

31、简述如何解决在 Windows 上通过 MinTTY 使用 git-bash，交互提示符不起作用的问题。

方式一：使用 winpty 来执行 vue 命令，如 winpty vue.cmd create hello（项目名）
方式二：在 git-bash 安装目录下找到 etc\bash.bashrc 文件，添加“alias vue='winpty vue.cmd'”行为为命令添加别名，重新启动 Git Bash 终端会话，这样更新后的 bashrc 文件才会生效。

32、简述什么是 GUI。

Vue CLI 引入了图形用户界面（GUI）来创建和管理项目，功能十分强大，给初学者提供了便利，可以快速搭建一个 Vue 项目。