2021秋季Web移动开发专业课

考卷（A）卷

考试时间（90分钟）

姓名_____ 班级_____学号_____得分_____

一、认知类

1. 关于前端开放框架，以下说法不正确的是（ ）

A) 前端著名的开发框架，常见的有ReactJS，AngularJS，VueJS，微信小程序等。

B) bootstarp 是一款优秀的前端CSS开发框架。

C) 学习和使用前端开发框架，除了掌握基础的HTML，CSS，JavaScript 需要掌握 nodejs，webpack等构建工具以及使用方法。

D) 前端开发需要深入了解数据库，中间件等技术细节。

2. 前端开发框架往往都是SPA架构。SPA（single-page application）仅在 Web 页面初始化时加载相应的 HTML、JavaScript 和 CSS。一旦页面加载完成，SPA 不会因为用户的操作而进行页面的重新加载或跳转；取而代之的是利用路由机制实现 HTML 内容的变换，UI 与用户的交互，避免页面的重新加载。关于SPA的优点，**以下说法不正确的是（ ）**

A) 用户体验好、快，内容的改变不需要重新加载整个页面，避免了不必要的跳转和重复渲染；

B) 基于上面一点，SPA 相对对服务器压力小；

C) 前后端职责分离，架构清晰，前端进行交互逻辑，后端负责数据处理。

D) 不需要后端参与就能完成各种完整的应用。

3. 关于SPA的缺点，以下说法**不正确**的是（ ）

A) 初次加载耗时多：为实现单页 Web 应用功能及显示效果，需要在加载页面的时候将 JavaScript、CSS 统一加载，部分页面按需加载；

B) 前进后退路由管理：由于单页应用在一个页面中显示所有的内容，所以不能使用浏览器的前进后退功能，所有的页面切换需要自己建立堆栈管理；

C) SEO 难度较大：由于所有的内容都在一个页面中动态替换显示，所以在 SEO 上其有着天然的弱势。

D) SPA开发难度较高。

4. 关于对Vue 生命周期的理解。Vue 实例有一个完整的生命周期，包含（ ）等一系列过程，我们称这是 Vue 的生命周期。

A) 初始化数据。

B) 渲染更新。

C) 渲染卸载。

D) 以上都有。

5. **关于MVVM以下说法错误的是（）**

Model—View—ViewModel （MVVM） 是一个软件架构设计模式，MVVM 源自于经典的 Model—View—Controller（MVC）模式 ，MVVM 的出现促进了前端开发与后端业务逻辑的分离，极大地提高了前端开发效率，MVVM 的核心是 ViewModel 层，它就像是一个中转站（value converter），负责转换 Model 中的数据对象来让数据变得更容易管理和使用，该层向上与视图层进行双向数据绑定，向下与 Model 层通过接口请求进行数据交互，起呈上启下作用。

A) View层:

View 是视图层，也就是用户界面。前端主要由 HTML 和 CSS 来构建 。

B) Model层:
Model 是指数据模型，泛指后端进行的各种业务逻辑处理和数据操控，对于前端来说就是后端提供的 api 接口。
C) ViewModel层:
ViewModel 是由前端开发人员组织生成和维护的视图数据层。在这一层，前端开发者对从后端获取的 Model 数据进行转换处理，做二次封装，以生成符合 View 层使用预期的视图数据模型。
D) MVVM 框架无法实现双向绑定。

二、程序题目

6. **前端开发光甲VUE中 v-show 与 v-if 有什么区别，以下说法错误的是（ ）**
A) v-if 是真正的条件渲染，因为它会确保在切换过程中条件块内的事件监听器和子组件适当地被销毁和重建；也是惰性的：如果在初始渲染时条件为假，则什么也不做——直到条件第一次变为真时，才会开始渲染条件块。
B) v-show 就简单得多——不管初始条件是什么，元素总是会被渲染，并且只是简单地基于 CSS 的“display”属性进行切换。
C) v-if 适用于在运行时很少改变条件，不需要频繁切换条件的场景；v-show 则适用于需要非常频繁切换条件的场景。
D) v-show 和 v-if 没什么差别。

7. **Class 与 Style 动态绑定。以下说法正确的是（AB）【多选】** 双向绑定

A) Class 可以通过对象语法和数组语法进行动态绑定：

对象语法：
```
<div v-bind:class="{ active: isActive, 'text-danger': hasError }"></div>
data: {
 isActive: true,
 hasError: false
}
```
数组语法：
```
<div v-bind:class="[isActive ? activeClass : '', errorClass]"></div>
data: {
 activeClass: 'active',
 errorClass: 'text-danger'
}
```
B) Style 可以通过对象语法和数组语法进行动态绑定：

对象语法：
```
<div v-bind:style="{ color: activeColor, fontSize: fontSize + 'px' }"></div>
data: {
 activeColor: 'red',
 fontSize: 30
}
```
数组语法：
```
<div v-bind:style="[styleColor, styleSize]"></div>
data: {
  styleColor: {
  color: 'red'
},
   styleSize:{
   fontSize:'23px'
 }
```

}

C) 仅只能对Style 进行动态绑定。

D) Class 与 Style 无法动态绑定。

8. 关于BootStarp 响应式布局，以下说法正确的是（ABCD） 【多选】

A) 同一套页面可以兼容不同分辨率的设备。

B) 依赖于栅格系统：将一行平均分成12个格子，可以指定元素占几个格子

C) 定义元素。指定该元素在不同的设备上，所占的格子数目。

样式：col-设备代号-格子数目

设备代号：

1. xs：超小屏幕 手机 (<768px)：col-xs-12
2. sm：小屏幕 平板 (≥768px)
3. md：中等屏幕 桌面显示器 (≥992px)
4. lg：大屏幕 大桌面显示器 (≥1200px)

D) 定义行。相当于tr 样式：row

9. 直接给一个数组项赋值，Vue 能检测到变化吗，以下说法正确的（AB） 【多选】

直接设置一个数组项时，例如：vm.items[indexOfItem] = newValue

当修改数组的长度时，例如：vm.items.length = newLength

A) 由于 JavaScript 的限制，Vue 不能直接检测到数组的变动。

B) Vue 提供了以下操作方法来获取数组中的值

```
// Vue.set
Vue.set(vm.items, indexOfItem, newValue)
// vm.$set，Vue.set的一个别名
vm.$set(vm.items, indexOfItem, newValue)
// Array.prototype.splice
vm.items.splice(indexOfItem, 1, newValue)
```

Vue 提供了以下操作来获取数组长度

```
vm.items.splice(newLength)
```

C) VUE无法检测数组变动后的值。

10. 关于Vue各个生命周期的作用，以下描述正确的是（ ）

生命周期	描述
A) beforeCreate	组件实例被创建之初，组件的属性生效之前
B) created	组件实例已经完全创建，属性也绑定， **dom 已经有生成，$el 可以调用**
C) beforeMount	在挂载开始之前被调用：相关的 render 函数首次被调用
D) mounted	el 被新创建的 vm.$el 替换，并挂载到实例上去之后调用该钩子
E) beforeUpdate	组件数据更新之前调用，发生在虚拟 DOM 打补丁之前
F) update	组件数据更新之后
G) activited	keep-alive 专属，组件被激活时调用
H) deactivated	keep-alive 专属，组件被销毁时调用
I) beforeDestory	组件销毁前调用
J) destoryed	组件销毁后调用

11. Vue 的父组件和子组件生命周期钩子函数执行顺序说法错误的是(C)

Vue 的父组件和子组件生命周期钩子函数执行顺序可以归类为以下几部分：

A) 父 beforeCreate -> 父 created -> 父 beforeMount -> 子 beforeCreate -> 子 created -> 子 beforeMount -> 子 mounted -> 父 mounted

B) 子组件更新过程

父 beforeUpdate -> 子 beforeUpdate -> 子 updated -> 父 updated

C) 父组件更新过程

12. 关于 V-Model 以下说法正确的是 ABCD 【多选】

以input 表单元素为例：

```
<input v-model='something'>
```

相当于

```
<input v-bind:value="something" v-on:input="something = $event.target.value">
```

复制代码

在自定义组件中，v-model 默认会利用名为 value 的 prop 和名为 input 的事件，如下所示：

父组件：

```
<ModelChild v-model="message"></ModelChild>
```

子组件：

```
<div>{{value}}</div>
props:{
   value: String
},
methods: {
 test1(){
    this.$emit('input', '小红')
 },
```

A) v-model 指令在表单 input、textarea、select 等元素上创建双向数据绑定。

B) text 和 textarea 元素使用 value 属性和 input 事件；

C) checkbox 和 radio 使用 checked 属性和 change 事件；

D) select 字段将 value 作为 prop 并将 change 作为事件。

13. Vue 组件间通信Vue 组件间通信是熟练使用框架的重要知识点。Vue 组件间通信只要指以下 3 类通信：父子组件通信、隔代组件通信、兄弟组件通信，关于下面介绍每种通信方式信与适用场景正确的是 （ABCD）【多选题】

A) props / $emit 适用 父子组件通信

这种方法是 Vue 组件的基础，相信大部分同学耳闻能详，所以此处就不举例展开介绍。

B) ref 与 $parent / $children 适用 父子组件通信

- ref：如果在普通的 DOM 元素上使用，引用指向的就是 DOM 元素；如果用在子组件上，引用就指向组件实例
- $parent / $children：访问父 / 子实例

C) EventBus （$emit / $on） 适用于 父子、隔代、兄弟组件通信

这种方法通过一个空的 Vue 实例作为中央事件总线（事件中心），用它来触发事件和监听事件，从而实现任何组件间的通信，包括父子、隔代、兄弟组件。

D) $attrs/$listeners 适用于 隔代组件通信

- $attrs：包含了父作用域中不被 prop 所识别 (且获取) 的特性绑定 (class 和 style 除外)。当一个组件没有声明任何 prop 时，这里会包含所有父作用域的绑定 (class 和 style 除外)，并且可以通过 v-bind="$attrs" 传入内部组件。通常配合 inheritAttrs 选项一起使用。
- $listeners：包含了父作用域中的 (不含 .native 修饰器的) v-on 事件监听器。它可以通过 v-on="$listeners" 传入内部组件

14. 关于vue-router的路由模式

vue-router 有 3 种路由模式：hash、history、abstract，对应的

关于这三种种路由模式，以下说法正确的是 (ABC) 【多选题】

代码：

```
switch (mode) {
 case 'history':
    this.history = new HTML5History(this, options.base)
    break
 case 'hash':
    this.history = new HashHistory(this, options.base, this.fallback)
    break
 case 'abstract':
    this.history = new AbstractHistory(this, options.base)
    break
 default:
    if (process.env.NODE_ENV !== 'production') {
      assert(false, `invalid mode: ${mode}`)
    }
}
```

A) hash: 使用 URL hash 值来作路由。支持所有浏览器，包括不支持 HTML5 History Api 的浏览器；

B) history：依赖 HTML5 History API 和服务器配置。具体可以查看 HTML5 History 模式；

C) abstract：支持所有 JavaScript 运行环境，如 Node.js 服务器端。如果发现没有浏览器的 API，路由会自动强制进入这个模式。

D) 默认情况下route无法指定路由。

15. Vue Router是vuejs官方的路由管理器，是学习vue的必备技能，利用vue-router，可以轻松构建单页应用。

以下界面展示正确的是 (ABC) 【多选择】 path模式

代码部分

(1)安装vue-router包

```
npm install --save vue-router
```

(2)定义路由

创建route.js

```
import Vue from 'vue'
import Router from 'vue-router'
import Home from './Home'
import Hello from './Hello'
Vue.use(Router)

export default new Router({
 routes: [{
        path: '/',
        name: 'Home',
        component: Home,
```

```
        children: [{
            path: '/about',
            name: 'About',
            component: About,
            }]
    }, {
        path: '/about/a',
        name: 'a',
        component: a
}]
})
```

创建 App.vue

```
<template>
 <div id="app">
   <router-view/>
 </div>
</template>
```

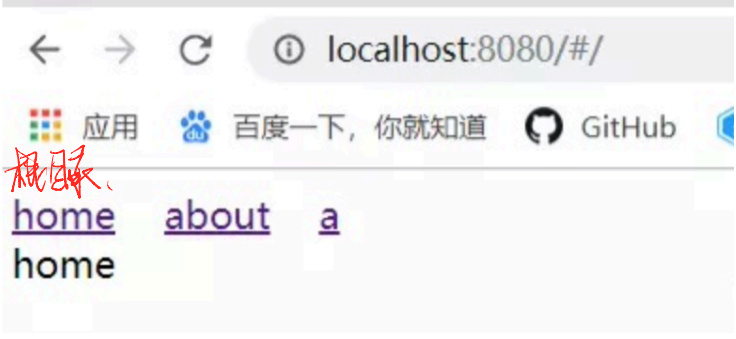
创建 Home.vue

```
<template>
   <div class="">
      <div>home</div>
      <router-view></router-view>
   </div>
</template>
```

启动项目访问根路径，既能打开Home组件内容。

展示部分：

A) 路径 /



B) 路径 /about

C) 路径 /about/a

3、 **操作类**
1、使用脚手架，创建一个VUE 工程。

2、使用github 发布add,commit ,push。
README.md 描述自己的项目内容，关键实现。

3、填写你的github项目仓库地址：