【Vue指令】

【常用指令】

* v-text、v-html、v-once、{{}}、v-clack
* v-on、v-bind、v-slot
* v-if/else-if/else、v-show
* v-for
* v-model
* Vue.directive(‘’, fn)、directives

【v-model 详解】

* v-model = :value + @input，语法基础：父子组件通信（props、$emit()）
* v-model 在组件化应用
* .lazy / .trim / .number 实现原理
* vue3中的v-model：

同时使用多个v-model:a

props接收

emits选项接收事件 emits: [‘update:a’]

支持自定义修饰符...

【扩展思考】

* 动态样式怎么写？
* v-once有什么特点？（.once）
* {{}} 一闪而过效果，v-clack
* v-for为什么key，为什么key不能用索引，v-for能循环哪些类型的类型？
* 自定义指令，你封装过哪些？指令背后的原理？

【Vue选项】

【常用选项】

* data, props, watch, computed
* methods, 11个生命周期
* mixins, filter, directives, components
* name,
* 局部路由守卫 beforeRouteEach

【watch, computed 详解】

【扩展思考】

* data为什么工厂模式？
* 谈一谈mixin理解？（）
* filter过滤器，只能{{}}和v-bind使用，可以用computed代替。

【Vuex状态管理】

【背景与作用】

* 背景：Flux，响应式状态管理
* 作用：数据在组件之间共享，数据缓存。

【五大概念】

* state：\_\_ob\_\_
* getters：计算属性
* mutations：能调接口，最好只放同步方法
* actions：可以直接修改state，但不建议
* modules：是协同开发的一种代码管理方式

【使用说明】

* 新建store文件，分模块管理
* 在根组件上挂载
* 在组合使用this.$store访问，使用四个map进行使用

【扩展思考】

* vuex是状态管理工具，用得规范容易维护，用得不好可以负面作用。
* vuex内存中，配合devtools方便调试数据，数据流管理相当重要。

【Webpack】

【手动搭环境】

入口 entry：三种写法，尤其是对象写法可以抽离vender

出口output：添加 filename: [name].[chunkhash].js 强缓存与协商缓存（HTTP缓存相关字段）

Loaders: moduls.rules

常用的loaders：file-loader、style-loader、css-loader、babel-loader、vue-loader, eslint-loader

Babel-loader: @babel/core、@babel/preset-env, @babel/preset-react, @babel/preset-typescript，在项目根添加babel.config.js（presets, plugins）

Eslint：在v4和v5两种写法，eslint, eslint-plugin-react, eslint-webpack-plugin, ……，ESlint有五六种配置方式，常用.eslintrc.js 配置文件（parser，parseOptions, plugins, rules）, devServer.overlay，如何解决eslint报错？

自定义loader：node代码，module.exports = function(source) { ..., return `module.exports = ${JSON.stringify{source}`, 一个loader本质上对一种字符串模板进行另一种格式化处理, 返回字符串.

常用的plugin: HtmlWebpackPlugin, EslintPlugin, ...., 自定义plugins: class MyPlugin { apply(compiler) { compiler.hooks.emit.tap(‘my-plugin’, fn)}}

devServer: 区分开发环境和生产环境, webpack-dev-server, 热更新, 端口, 代理... v4: cross-env, v5: module.exports = function(env) webpack --env development ..... 使用webpack-merge配置文件的分离.

【打包优化】

盘点10条

【运行优化】

盘点10点

【拓展思考】

Webpack，gulp, vite 有什么区别？你如何评价这些构建工具？

Wepack-chain 好不好用？有一个项目中用到了？

webpack搭建vue环境？

vite搭建react环境？

【React基础】

【JSX基础】

为什么使用jsx ？是什么？哪家公司？

使用 @babel/preset-react 进行编译，编译成 React.createElement() 语法

Jsx 是不可变对象，每次render时是如何生成虚拟DOM、diff运算

JSX 支持 {} 这种语法，表达式、基本数据类型、Array<jsx元素>，undefined、null, 函数调用。在这个基本语法，条件渲染？动态样式？列表循环？

事件绑定：bind，箭头函数。事件对象，阻止冒泡，键盘事件，

表单绑定：受控表单，文件上传

组件化：props.children， render props，设计思想是“组合”

【React知识】

This.setState({}, callback), setState((state,props)=>({}), callback)，异步性（合成事件，自动合并多个setState()）,在宏任务中是同步的，在合成事件的微任务中它是异步的。

生命周期（3-2-1）每个生命周期的特点一定要知道。

forceUpdate()

上下文：有什么用？在哪些第三方插件用到过？createContext() 怎么使用？

高阶组件：是代码复用的一种逻辑，不是什么新语法，相当于是一种设计模式。它本质上一个纯函数，接受一个UI组件，返回一个新的UI组件，所以高阶组件也被称呼容器组件。在mobx、redux、react-router-dom都有用到，在类组件开发中，使用装饰器语法可以方便快捷地使用高阶组件，在函数式组件开发只能调用高阶组件。在类组件中，高阶组件的能力比较突出，通过高阶组件可以向UI组件注入props（状态数据、jsx、可复用的函数）。

Hooks：在react16.8之后，useState, useEffect, useMemo, useCallback, useRef, useContext, useReducer, .... 路由Hooks，Redux中Hooks， React-Use, 自定义封装Hooks（代码复用）。自定义封装useLang() / React-Intel。

【React状态管理】