**js面试题**

1、js数据类型

**基本数据类型**

**Number、String、Boolean、Null、Undefined、Symbol、bigInt**

**引用数据类型**

**object、Array、Date、Function、RegExp**

2、js变量和函数声明的提升

在js中变量和函数的声明会提升到最顶部执行

函数的提升高于变量的提升

函数内部如果用 var 声明了相同名称的外部变量，函数将不再向上寻找。

匿名函数不会提升。

3、闭包

闭包就是能够读取其他函数内部变量的函数

闭包基本上就是一个函数内部返回一个函数

好处

可以读取函数内部的变量

将变量始终保持在内存中

可以封装对象的私有属性和私有方法

坏处

比较耗费内存、使用不当会造成内存溢出的问题

4、== 和 ===的区别

==是非严格意义上的相等

值相等就相等

===是严格意义上的相等，会比较两边的数据类型和值大小

值和引用地址都相等才相等

5、this

this总是指向函数的直接调用者

如果有new关键字，this指向new出来的对象

在事件中，this指向触发这个事件的对象

1. js数组和对象的遍历方式

for in

for

forEach

for-of

7、map与forEach的区别

forEach 方法，是最基本的方法，就是遍历与循环，默认有 3 个传参：分别是遍历的数组内

容 item、数组索引 index、和当前遍历数组 Array

map 方法，基本用法与 forEach 一致，但是不同的，它会返回一个新的数组，所以 callback

需要有 return 值，如果没有，会返回 undefined

8、箭头函数与普通函数的区别?

函数体内的 this 对象，就是定义时所在的对象，而不是使用时所在的对象

不可以当作构造函数，也就是说，不可以使用 new 命令，否则会抛出一个错误

不可以使用 arguments 对象，该对象在函数体内不存在。如果要用，可以用 Rest 参数代替

不可以使用 yield 命令，因此箭头函数不能用作 Generator 函数

9、同源策略

同源指的是域名、协议、端口号相同

10、如何解决跨域

jsonp跨域

document.domain + iframe 跨域

nodejs中间件代理跨域

后端在头部信息里面设置安全域名

11、严格模式的限制

变量必须声明后再使用

函数的参数不能有同名属性，否则报错

不能使用 with 语句

禁止 this 指向全局对象

12、es6新增

新增模板字符串

箭头函数

for-of（用来遍历数据—例如数组中的值。）

ES6 将 Promise 对象纳入规范，提供了原生的 Promise 对象。

增加了 let 和 const 命令，用来声明变量。

还有就是引入 module 模块的概念

13、attribute 和 property 的区别是什么?

attribute 是 dom 元素在文档中作为 html 标签拥有的属性

property 就是 dom 元素在 js 中作为对象拥有的属性。

对于 html 的标准属性来说，attribute 和 property 是同步的，是会自动更新的

但是对于自定义的属性来说，他们是不同步的

14、let和const 的区别是什么?

let 命令不存在变量提升，如果在 let 前使用，会导致报错

如果块区中存在 let 和 const 命令，就会形成封闭作用域

不允许重复声明

const定义的是常量，不能修改，但是如果定义的是对象，可以修改对象内部的数据

15、内存泄漏

定义：程序中己动态分配的堆内存由于某种原因程序未释放或无法释放引发的各种问题。

js中可能出现的内存泄漏情况：结果：变慢，崩溃，延迟大等

js中可能出现的内存泄漏原因

全局变量

dom 清空时，还存在引用

定时器未清除

子元素存在引起的内存泄露

16、script 引入方式?

html 静态 <script> 引入

js 动态插入 <script>

<script defer> : 异步加载，元素解析完成后执行

<script async> : 异步加载，但执行时会阻塞元素渲染

17、数组(array)方法

map : 遍历数组，返回回调返回值组成的新数组

forEach : 无法 break ，可以用 try/catch 中 throw new Error 来停止

filter : 过滤

some : 有一项返回 true ，则整体为 true

every : 有一项返回 false ，则整体为 false

join : 通过指定连接符生成字符串

push / pop : 末尾推入和弹出，改变原数组， 返回推入/弹出项

unshift / shift : 头部推入和弹出，改变原数组，返回操作项

sort(fn) / reverse : 排序与反转，改变原数组

concat : 连接数组，不影响原数组， 浅拷贝

slice(start, end) : 返回截断后的新数组，不改变原数组

splice(start,number,value…): 返回删除元素组成的数组，value 为插入项，改变原数组

indexOf / lastIndexOf(value, fromIndex) : 查找数组项，返回对应的下标

reduce / reduceRight(fn(prev, cur) ，defaultPrev) : 两两执行，prev 为上次化简函数的return 值，cur 为当前值(从第二项开始)

18、JavaScript 深浅拷贝?

浅拷贝

Object.assign

深拷贝

可以通过 JSON.parse(JSON.stringify(object)) 来解决

19、说说异步编程的实现方式?

回调函数

优点：简单、容易理解

缺点：不利于维护、代码耦合高

事件监听

优点：容易理解，可以绑定多个事件，每个事件可以指定多个回调函数

缺点：事件驱动型，流程不够清晰

发布/订阅(观察者模式)

类似于事件监听，但是可以通过‘消息中心’，了解现在有多少发布者，多少订阅者

Promise 对象

优点：可以利用 then 方法，进行链式写法；可以书写错误时的回调函数

缺点：编写和理解，相对比较难

Generator 函数

优点：函数体内外的数据交换、错误处理机制

缺点：流程管理不方便

async 函数

优点：内置执行器、更好的语义、更广的适用性、返回的是 Promise、结构清晰

缺点：错误处理机制

20、说说面向对象编程思想?

基本思想是使用对象，类，继承，封装等基本概念来进行程序设计

优点

易维护

易扩展

开发工作的重用性、继承性高，降低重复工作量。

缩短了开发周期

21、项目性能优化

减少 HTTP 请求数

减少 DNS 查询

使用 CDN

避免重定向

图片懒加载

减少 DOM 元素数量

减少 DOM 操作

使用外部 JavaScript 和 CSS

压缩 JavaScript、CSS、字体、图片等

优化 CSS Sprite

使用 iconfont

多域名分发划分内容到不同域名

尽量减少 iframe 使用

避免图片 src 为空

把样式表放在 link 中

把 JavaScript 放在页面底部

22、什么是单线程，和异步的关系？

单线程 ：只有一个线程，只能做一件事

原因 ： 避免 DOM 渲染的冲突

浏览器需要渲染 DOM

JS 可以修改 DOM 结构

JS 执行的时候，浏览器 DOM 渲染会暂停

两段 JS 也不能同时执行（都修改 DOM 就冲突了）

webworker 支持多线程，但是不能访问 DOM

解决方案 ：异步

23、说说负载均衡？

单台服务器共同协作，不让其中某一台或几台超额工作，发挥服务器的最大作用

http 重定向负载均衡：调度者根据策略选择服务器以 302 响应请求，缺点只有第一次有效果，后续操作维持在该服务器 dns 负载均衡：解析域名时，访问多个 ip 服务器中的一个（可监控性较弱）原因 - 避免 DOM 渲染的冲突

反向代理负载均衡：访问统一的服务器，由服务器进行调度访问实际的某个服务器，对统一的服务器要求大，性能受到 服务器群的数量

24、作用域链？

作用域链可以理解为一组对象列表，包含 父级和自身的变量对象，因此我们便能通过作用域链访问到父级里声明的变量或者函数

25、什么是原型、原型链、继承？

所有的函数都有prototype属性（原型）

所有的对象都有\_\_proto\_\_属性

在Javascript中，每个函数都有一个原型属性prototype指向自身的原型，而由这个函数创建的对象也有一个proto属性指向这个原型，而函数的原型是一个对象，所以这个对象也会有一个proto指向自己的原型，这样逐层深入直到Object对象的原型，这样就形成了原型链。

26、JS垃圾回收机制是怎样的？

1.概述

js的垃圾回收机制是为了防止内存泄漏（已经不需要的某一块内存还一直存在着），垃圾回收机制就是不停歇的寻找这些不再使用的变量，并且释放掉它所指向的内存。

在JS中，JS的执行环境会负责管理代码执行过程中使用的内存。

2.变量的生命周期

当一个变量的生命周期结束之后，它所指向的内存就会被释放。js有两种变量，局部变量和全局变量，局部变量是在他当前的函数中产生作用，当该函数结束之后，该变量内存会被释放，全局变量的话会一直存在，直到浏览器关闭为止。

3.js垃圾回收方式

有两种方式： 标记清除、引用计数

标记清除：大部分浏览器使用这种垃圾回收，当变量进入执行环境(声明变量）的时候，垃圾回收器将该变量进行了标记，当该变量离开环境的时候，将其再度标记，随之进行删除。

引用计数：这种方式常常会引起内存的泄露，主要存在于低版本的浏览器。它的机制就是跟踪某一个值得引用次数，当声明一个变量并且将一个引用类型赋值给变量得时候引用次数加1，当这个变量指向其他一个时引用次数减1，当为0时出发回收机制进行回收。

27、逐进增强和优雅降级

逐进增强

针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高版本浏览器进行效果、交互等改进和追加功能达到更好的用户体验。

优雅降级

一开始就构建完整的功能，然后再针对低版本浏览器进行兼容

vue面试题

1、vue优点

轻量级

速度快

简单易学

低耦合

可重用性

独立开发

文档齐全，且文档为中文文档

2、vue父组件向子组件传递数据

props

3、子组件向父组件传递事件

$emit

4、v-show和v-if指令的共同点和不同点

相同点:都可以控制dom元素的显示和隐藏

不同点:v-show只是改变display属性，dom元素并未消失，切换时不需要重新渲染页面

v-if直接将dom元素从页面删除，再次切换需要重新渲染页面

5、如何让CSS只在当前组件中起作用

scoped

6、的作用是什么

主要是用于需要频繁切换的组件时进行缓存，不需要重新渲染页面

7、如何获取dom

给dom元素加ref=‘refname’,然后通过this.$refs.refname进行获取dom元素

8、说出几种vue当中的指令和它的用法

v-model

v-on

v-html

v-text

v-once

v-if

v-show

9、vue-loader是什么？它的用途是什么？

vue文件的一个加载器，将template/js/style转换为js模块

用途:js可以写es6、style样式

10、为什么用key

给每个dom元素加上key作为唯一标识 ，diff算法可以正确的识别这个节点，使页面渲染更加迅速。

11、axios及安装？

vue项目中使用ajax时需要axios插件

下载方式cnpm install axios --save

12、v-model的使用

v-model用于表单的双向绑定，可以实时修改数据

13、请说出vue.cli项目中src目录每个文件夹和文件的用法

components存放组件

app.vue主页面入口

index.js主文件入口

ass存放静态资源文件

14、分别简述computed和watch的使用场景

用官网的一句话来说，所有需要用到计算的都应该使用计算属性。多条数据影响一条数据时使用计算属性，使用场景购物车。

如果是一条数据更改，影响多条数据时，使用watch，使用场景搜索框。

15、v-on可以监听多个方法吗？

可以，比如 v-on=“onclick,onbure”

16、$nextTick的使用

在data()中的修改后，页面中无法获取data修改后的数据，使用$nextTick时，当data中的数据修改后，可以实时的渲染页面

17、vue组件中data为什么必须是一个函数？

因为javaScript的特性所导致，在component中，data必须以函数的形式存在，不可以是对象。

组件中的data写成一个函数，数据以函数返回值的形式定义，这样每次复用组件的时候，都会返回一份新的data，相当于每个组件实例都有自己私有的数据空间，他们值负责各自维护数据，不会造成混乱。而单纯的写成对象形式，就是所有组件实例共用了一个data，这样改一个全部都会修改。

18、渐进式框架的理解

主张最少

可以根据不同的需求选择不同的层级

19、vue在双向数据绑定是如何实现的？

vue双向数据绑定是通过数据劫持、组合、发布订阅模式的方式来实现的，也就是说数据和视图同步，数据发生变化，视图跟着变化，视图变化，数据也随之发生改变

核心：关于vue双向数据绑定，其核心是Object.defineProperty()方法

20、单页面应用和多页面应用区别及缺点

单页面应用（SPA），通俗的说就是指只有一个主页面的应用，浏览器一开始就加载所有的js、html、css。所有的页面内容都包含在这个主页面中。但在写的时候，还是分开写，然后再加护的时候有路由程序动态载入，单页面的页面跳转，仅刷新局部资源。多用于pc端。

多页面（MPA），就是一个应用中有多个页面，页面跳转时是整页刷新

单页面的优点：用户体验好，快，内容的改变不需要重新加载整个页面，基于这一点spa对服务器压力较小；前后端分离，页面效果会比较酷炫

单页面缺点：不利于seo；导航不可用，如果一定要导航需要自行实现前进、后退。初次加载时耗时多；页面复杂度提高很多。

21、Vue 项目中为什么要在列表组件中写 key，其作用是什么？

key是给每一个vnode的唯一id,可以依靠key,更准确, 更快的拿到oldVnode中对应的vnode节点。

更准确

因为带key就不是就地复用了，在sameNode函数 a.key === b.key对比中可以避免就地复用的情况。所以会更加准确。

更快

利用key的唯一性生成map对象来获取对应节点，比遍历方式更快。

22、父组件和子组件生命周期钩子执行顺序是什么？

加载渲染过程

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

子组件更新过程

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

父组件更新过程

父 beforeUpdate -> 父 updated

销毁过程

父 beforeDestroy -> 子 beforeDestroy -> 子 destroyed -> 父 destroyed

23、谈一谈你对 nextTick 的理解？

当你修改了data的值然后马上获取这个dom元素的值，是不能获取到更新后的值，你需要使用$nextTick这个回调，让修改后的data值渲染更新到dom元素之后在获取，才能成功。

24、vue组件中data为什么必须是一个函数？

因为JavaScript的特性所导致，在component中，data必须以函数的形式存在，不可以是对象。

组建中的data写成一个函数，数据以函数返回值的形式定义，这样每次复用组件的时候，都会返回一份新的data，相当于每个组件实例都有自己私有的数据空间，它们只负责各自维护的数据，不会造成混乱。而单纯的写成对象形式，就是所有的组件实例共用了一个data，这样改一个全都改了。

25、vue和jQuery的区别

jQuery是使用选择器（$）选取DOM对象，对其进行赋值、取值、事件绑定等操作，其实和原生的HTML的区别只在于可以更方便的选取和操作DOM对象，而数据和界面是在一起的。比如需要获取label标签的内容：$("lable").val();,它还是依赖DOM元素的值。

Vue则是通过Vue对象将数据和View完全分离开来了。对数据进行操作不再需要引用相应的DOM对象，可以说数据和View是分离的，他们通过Vue对象这个vm实现相互的绑定。这就是传说中的MVVM。

26、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。

27、SPA首屏加载慢如何解决

安装动态懒加载所需插件；使用CDN资源。

28、vue项目是打包了一个js文件，一个css文件，还是有多个文件？

根据vue-cli脚手架规范，一个js文件，一个CSS文件。

29、vue更新数组时触发视图更新的方法

push()；

pop()；

shift()；

unshift()；

splice()；

sort()；

reverse()

30、什么是 vue 生命周期？有什么作用？

每个 Vue 实例在被创建时都要经过一系列的初始化过程——例如，需要设置数据监听、编译模板、将实例挂载到 DOM 并在数据变化时更新 DOM 等。同时在这个过程中也会运行一些叫做 生命周期钩子 的函数，这给了用户在不同阶段添加自己的代码的机会。

31、第一次页面加载会触发哪几个钩子？

beforeCreate， created， beforeMount， mounted

32、vue获取数据在一般在哪个周期函数

created

beforeMount

mounted

33、created和mounted的区别

created:在模板渲染成html前调用，即通常初始化某些属性值，然后再渲染成视图。

mounted:在模板渲染成html后调用，通常是初始化页面完成后，再对html的dom节点进行一些需要的操作。

34、vue生命周期的理解

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。

载入前/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。

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。

35、vuex是什么？

vue框架中状态管理。

36、vuex有哪几种属性？

有五种，State、 Getter、Mutation 、Action、 Module

state： 基本数据(数据源存放地)

getters： 从基本数据派生出来的数据

mutations ： 提交更改数据的方法，同步！

actions ： 像一个装饰器，包裹mutations，使之可以异步。

modules ： 模块化Vuex

37、vue全家桶

vue-cli、vuex、vueRouter、Axios

38、vue-cli 工程常用的 npm 命令有哪些?

npm install 下载 node\_modules 资源包的命令

npm run dev 启动 vue-cli 开发环境的 npm 命令

npm run build vue-cli 生成 生产环境部署资源 的 npm 命令

npm run build–report 用于查看 vue-cli 生产环境部署资源文件大小的 npm 命令

39、请说出 vue-cli 工程中每个文件夹和文件的用处?

build 文件夹是保存一些 webpack 的初始化配置。

config 文件夹保存一些项目初始化的配置

node\_modules 是 npm 加载的项目依赖的模块

src 目录是我们要开发的目录:

assets 用来放置图片

components 用来放组件文件

app.vue 是项目入口文件

main.js 项目的核心文件

40、v-if 和 v-show 有什么区别

共同点：都是动态显示 DOM 元素

区别点：

v-if 是动态的向 DOM 树内添加或者删除 DOM 元素

v-show 是通过设置 DOM 元素的 display 样式属性控制显隐

v-if 切换有一个局部编译/卸载的过程，切换过程中合适地销毁和重建内部的事件监听和子组件

v-show 只是简单的基于 css 切换

性能消耗

v-if 有更高的切换消耗

v-show 有更高的初始渲染消耗

使用场景

v-if 适合运营条件不大可能改变

v-show 适合频繁切换

41、v-for 与 v-if 的优先级?

v-for 和 v-if 同时使用，有一个先后运行的优先级，v-for 比 v-if 优先级更高，这就说明在

v-for 每次的循环赋值中每一次调用 v-if 的判断，所以不推荐 v-if 和 v-for 在同一个标签中同时使用。

42、 vue 常用的修饰符?

事件修饰符

.stop 阻止事件继续传播

.prevent 阻止标签默认行为

.capture 使用事件捕获模式，即元素自身触发的事件先在此处处理，然后才交由内部元素进行处理

.self 只当在 event.target 是当前元素自身时触发处理函数

.once 事件只会触发一次

.passive 告诉浏览器你不想阻止事件的默认行为

v-model 的修饰符

.lazy 通过这个修饰符，转变为在 change 事件再同步

.number 自动将用户输入值转化为数值类型

.trim 自动过滤用户输入的收尾空格

键盘事件修饰符

.enter

.tab

.delete (捕获“删除”和“退格”键)

.esc

.space

.up

.down

.left

.right

系统修饰符

.ctrl

.alt

.shift

.meta

鼠标按钮修饰符

.left

.right

.middle

43、vue 事件中如何使用 event 对象?

获取事件对象，方法参数传递 $event 。注意在事件中要使用 $ 符号

<button @click="Event($event)">事件对象</button>

1

44、组件传值方式有哪些

父传子：子组件通过props[‘xx’] 来接收父组件传递的属性 xx 的值

子传父：子组件通过 this.$emit(‘fnName’,value) 来传递,父组件通过接收 fnName 事件方法来接收回调

其他方式：通过创建一个bus，进行传值

使用Vuex

45、vue 中子组件调用父组件的方法?

直接在子组件中通过 this.$parent.event 来调用父组件的方法。

在子组件里用$emit()向父组件触发一个事件，父组件监听这个事件就行了。

父组件把方法传入子组件中，在子组件里直接调用这个方法。

46、 如何让 CSS 只在当前组件中起作用？

在组件中的 style 前面加上 scoped

47、如何获取 dom?

ref="domName" 用法：this.$refs.domName

48、vue路由跳转

(一)声明式导航router-link

不带参数：

// 注意：router-link中链接如果是'/'开始就是从根路由开始，如果开始不带'/'，则从当前路由开始。

<router-link :to="{name:'home'}">

<router-link :to="{path:'/home'}"> //name,path都行, 建议用name

1

2

3

带参数：

<router-link :to="{name:'home', params: {id:1}}">

<router-link :to="{name:'home', query: {id:1}}">

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

//传递对象

<router-link :to="{name:'detail', query: {item:JSON.stringify(obj)}}"></router-link>

1

2

3

4

5

(二)this.$router.push()

不带参数：

this.$router.push('/home')

this.$router.push({name:'home'})

this.$router.push({path:'/home'})

1

2

3

query传参

1.路由配置：

name: 'home',

path: '/home'

2.跳转：

this.$router.push({name:'home',query: {id:'1'}})

this.$router.push({path:'/home',query: {id:'1'}})

3.获取参数

html取参: $route.query.id

script取参: this.$route.query.id

1

2

3

4

5

6

7

8

9

params传参

1.路由配置：

name: 'home',

path: '/home/:id'(或者path: '/home:id')

2.跳转：

this.$router.push({name:'home',params: {id:'1'}})

注意：

// 只能用 name匹配路由不能用path

// params传参数(类似post) 路由配置 path: "/home/:id" 或者 path: "/home:id"否则刷新参数消失

3.获取参数

html取参:$route.params.id

script取参:this.$route.params.id

1

2

3

4

5

6

7

8

9

10

11

直接通过path传参

1.路由配置：

name: 'home',

path: '/home/:id'

2.跳转：

this.$router.push({path:'/home/123'})

或者：

this.$router.push('/home/123')

3.获取参数：

this.$route.params.id

1

2

3

4

5

6

7

8

9

params和query的区别

query类似 get，跳转之后页面 url后面会拼接参数，类似?id=1。

非重要性的可以这样传，密码之类还是用params，刷新页面id还在。

params类似 post，跳转之后页面 url后面不会拼接参数。

(三)this.$router.replace()

用法同上

(四)this.$router.go(n)

向前或者向后跳转n个页面，n可为正整数或负整数

区别:

this.$router.push

跳转到指定url路径，并在history栈中添加一个记录，点击后退会返回到上一个页面

this.$router.replace

跳转到指定url路径，但是history栈中不会有记录，点击返回会跳转到上上个页面 (就是直接替换了当前页面)

this.$router.go(n)

向前或者向后跳转n个页面，n可为正整数或负整数

49、Vue.js 双向绑定的原理

Vue.js 2.0 采用数据劫持（Proxy 模式）结合发布者-订阅者模式（PubSub 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。

每个组件实例都有相应的watcher程序实例，它会在组件渲染的过程中把属性记录为依赖，之后当依赖项的setter被调用时，会通知watcher重新计算，从而致使它关联的组件得以更新。

Vue.js 3.0, 放弃了Object.defineProperty ，使用更快的ES6原生 Proxy (访问对象拦截器, 也称代理器)

50、Computed和Watch的区别

computed 计算属性 : 依赖其它属性值,并且 computed 的值有缓存,只有它依赖的 属性值发生改变,下一次获取 computed 的值时才会重新计算 computed 的值。

watch 侦听器 : 更多的是观察的作用,无缓存性,类似于某些数据的监听回调,每 当监听的数据变化时都会执行回调进行后续操作。

运用场景：

当我们需要进行数值计算,并且依赖于其它数据时,应该使用 computed,因为可以利用 computed的缓存特性,避免每次获取值时,都要重新计算。

当我们需要在数据变化时执行异步或开销较大的操作时,应该使用 watch,使用 watch 选项允许我们执行异步操作 ( 访问一个 API ),限制我们执行该操作的频率, 并在我们得到最终结果前,设置中间状态。这些都是计算属性无法做到的。

多个因素影响一个显示，用Computed；一个因素的变化影响多个其他因素、显示，用Watch;

Computed 和 Methods 的区别

computed: 计算属性是基于它们的依赖进行缓存的,只有在它的相关依赖发生改变时才会重新求值对于 method ，只要发生重新渲染，

method 调用总会执行该函数

51、过滤器 (Filter)

在Vue中使用filters来过滤(格式化)数据，filters不会修改数据，而是过滤(格式化)数据，改变用户看到的输出（计算属性 computed ，方法 methods 都是通过修改数据来处理数据格式的输出显示。

使用场景： 比如需要处理时间、数字等的的显示格式；

52、axios是什么

易用、简洁且高效的http库， 支持node端和浏览器端，支持Promise，支持拦截器等高级配置。

53、sass是什么？如何在vue中安装和使用？

sass是一种CSS预编译语言安装和使用步骤如下。

用npm安装加载程序（ sass-loader、 css-loader等加载程序)。

在 webpack.config.js中配置sass加载程序。

54、Vue.js页面闪烁

Vue. js提供了一个v-cloak指令，该指令一直保持在元素上，直到关联实例结束编译。当和CSS一起使用时，这个指令可以隐藏未编译的标签，直到实例编译结束。用法如下。

[v-cloak]{

display:none;

}

<div v-cloak>{{ title }}</div>

1

2

3

4

55、如何解决数据层级结构太深的问题

在开发业务时，经常会岀现异步获取数据的情况，有时数据层次比较深，如以下代码: span 'v-text="a.b.c.d">, 可以使用vm.$set手动定义一层数据: vm.$set("demo"，a.b.c.d)

56、vue常用指令

v-model 多用于表单元素实现双向数据绑定（同angular中的ng-model）

v-bind 动态绑定 作用： 及时对页面的数据进行更改

v-on:click 给标签绑定函数，可以缩写为@，例如绑定一个点击函数 函数必须写在methods里面

v-for 格式： v-for=“字段名 in(of) 数组json” 循环数组或json(同angular中的ng-repeat)

v-show 显示内容 （同angular中的ng-show）

v-hide 隐藏内容（同angular中的ng-hide）

v-if 显示与隐藏 （dom元素的删除添加 同angular中的ng-if 默认值为false）

v-else-if 必须和v-if连用

v-else 必须和v-if连用 不能单独使用 否则报错 模板编译错误

v-text 解析文本

v-html 解析html标签

v-bind:class 三种绑定方法

对象型 ‘{red:isred}’

三元型 ‘isred?“red”:“blue”’

数组型 ‘[{red:“isred”},{blue:“isblue”}]’

v-once 进入页面时 只渲染一次 不在进行渲染

v-cloak 防止闪烁

v-pre 把标签内部的元素原位输出

57、$route和$router的区别

$route是“路由信息对象”，包括path，params，hash，query，fullPath，matched，name等路由信息参数。

$router是“路由实例”对象包括了路由的跳转方法，钩子函数等

58、怎样理解 Vue 的单项数据流

数据总是从父组件传到子组件，子组件没有权利修改父组件传过来的数据，只能请求父组件对原始数据进行修改。这样会防止从子组件意外改变父组件的状态，从而导致你的应用的数据流向难以理解。

注意：在子组件直接用 v-model 绑定父组件传过来的 props 这样是不规范的写法，开发环境会报警告。

如果实在要改变父组件的 props 值可以再data里面定义一个变量，并用 prop 的值初始化它，之后用$emit 通知父组件去修改。

59、虚拟DOM是什么？有什么优缺点？

由于在浏览器中操作DOM是很昂贵的。频繁操作DOM，会产生一定性能问题。这就是虚拟Dom的产生原因。Vue2的Virtual DOM 借鉴了开源库 snabbdom 的实现。Virtual DOM本质就是用一个原生的JS对象去描述一个DOM节点，是对真实DOM的一层抽象。

优点：

1、保证性能下限：框架的虚拟DOM需要适配任何上层API可能产生的操作，他的一些DOM操作的实现必须是普适的，所以它的性能并不是最优的；但是比起粗暴的DOM操作性能要好很多，因此框架的虚拟DOM至少可以保证在你不需要手动优化的情况下，依然可以提供还不错的性能，既保证性能的下限。

2、无需手动操作DOM：我们不需手动去操作DOM，只需要写好 View-Model的 代码逻辑，框架会根据虚拟DOM和数据双向绑定，帮我们以可预期的方式更新视图，极大提高我们的开发效率。

3、跨平台：虚拟DOM本质上是JavaScript对象，而DOM与平台强相关，相比之下虚拟DOM可以进行更方便地跨平台操作，例如服务器端渲染、weex开发等等。

缺点：

1、无法进行极致优化：虽然虚拟DOM + 合理的优化，足以应对大部分应用的性能需要，但在一些性能要求极高的应用中虚拟DOM无法进行针对性的极致优化。

2、首次渲染大量DOM时，由于多了一层DOM计算，会比innerHTML插入慢。

60、Vuex 页面刷新数据丢失怎么解决？

需要做 vuex 数据持久化，一般使用本地储存的方案来保存数据，可以自己设计存储方案，也可以使用第三方插件。

推荐使用 vuex-persist (脯肉赛斯特)插件，它是为 Vuex 持久化储存而生的一个插件。不需要你手动存取 storage，而是直接将状态保存至 cookie 或者 localStorage中。

61、Vuex 为什么要分模块并且加命名空间？

模块： 由于使用单一状态树，应用的所有状态会集中到一个比较大的对象。当应用变得非常复杂时，store 对象就有可能会变得相当臃肿。为了解决以上问题，Vuex 允许我们将 store 分割成模块（module）。每个模块拥有自己的 state、mutation、action、getter、甚至是嵌套子模块。

命名空间： 默认情况下，模块内部的 action、mutation、getter是注册在全局命名空间的 — 这样使得多个模块能够对同一 mutation 或 action 做出响应。如果希望你的模块具有更高的封装度和复用性，你可以通过添加 namespaced:true 的方式使其成为带命名的模块。当模块被注册后，他所有 getter、action、及 mutation 都会自动根据模块注册的路径调整命名。

62、vue 中使用了哪些设计模式？

1、工厂模式 - 传入参数即可创建实例

虚拟 DOM 根据参数的不同返回基础标签的 Vnode 和组件 Vnode。

2、单例模式 - 整个程序有且仅有一个实例

vuex 和 vue-router 的插件注册方法 install 判断如果系统存在实例就直接返回掉。

3、发布-订阅模式。（vue 事件机制）

4、观察者模式。（响应式数据原理）

5、装饰器模式（@装饰器的用法）

6、策略模式，策略模式指对象有某个行为，但是在不同的场景中，该行为有不同的实现方案 - 比如选项的合并策略。

63、你都做过哪些 Vue 的性能优化？

这里只列举针对 Vue 的性能优化，整个项目的性能优化是一个大工程。

对象层级不要过深，否则性能就会差。

不需要响应式的数据不要放在 data 中（可以使用 Object.freeze() 冻结数据）

v-if 和 v-show 区分使用场景

computed 和 watch 区分场景使用

v-for 遍历必须加 key，key最好是id值，且避免同时使用 v-if

大数据列表和表格性能优化 - 虚拟列表 / 虚拟表格

防止内部泄露，组件销毁后把全局变量和时间销毁

图片懒加载

路由懒加载

异步路由

第三方插件的按需加载

适当采用 keep-alive 缓存组件

防抖、节流的运用

服务端渲染 SSR or 预渲染

64、Vue.set 方法原理

在两种情况下修改 Vue 是不会触发视图更新的。

1、在实例创建之后添加新的属性到实例上（给响应式对象新增属性）

2、直接更改数组下标来修改数组的值。

Vue.set 或者说是 $set 原理如下

因为响应式数据 我们给对象和数组本身新增了\_\_ob\_\_属性，代表的是 Observer 实例。当给对象新增不存在的属性，首先会把新的属性进行响应式跟踪 然后会触发对象 ob 的dep收集到的 watcher 去更新，当修改数组索引时我们调用数组本身的 splice 方法去更新数组。

65、函数式组件使用场景和原理

函数式组件与普通组件的区别

1、函数式组件需要在声明组件时指定 functional:true

2、不需要实例化，所以没有this，this通过render函数的第二个参数context代替

3、没有生命周期钩子函数，不能使用计算属性，watch

4、不能通过$emit对外暴露事件，调用事件只能通过context.listeners.click的方式调用外部传入的事件

5、因为函数组件时没有实例化的，所以在外部通过ref去引用组件时，实际引用的是HTMLElement

6、函数式组件的props可以不用显示声明，所以没有在props里面声明的属性都会被自动隐式解析为prop，而普通的组件所有未声明的属性都解析到$attrs里面，并自动挂载到组件根元素上（可以通过inheritAttrs属性禁止）

优点：1.由于函数组件不需要实例化，无状态，没有生命周期，所以渲染性要好于普通组件2.函数组件结构比较简单，代码结构更清晰

使用场景：

一个简单的展示组件，作为容器组件使用 比如 router-view 就是一个函数式组件。 “高阶组件”—用于接受一个组件为参数，返回一个被包装过的组件。

相关代码如下：

if (isTrue(Ctor.options.functional)) { // 带有functional的属性的就是函数式组件 return createFunctionalComponent(Ctor, propsData, data, context, children); } const listeners = data.on; data.on = data.nativeOn; installComponentHooks(data); // 安装组件相关钩子 （函数式组件没有调用此方法，从而性能高于普通组件）

66、子组件为何不可以修改父组件传递的 Prop？

所有的 prop 都使得其父子 prop 之间形成了一个单向下行绑定：父级 prop 的更新会向下流动到子组件中，但是反过来则不行。这样会防止从子组件意外改变父级组件的状态，从而导致你的应用的数据流向难以理解。

67、vue项目创建。路由配置、环境配置以及组件传值等

# css、html面试题

HTML和HTML5有什么区别？

主要有三个区别：

1、文档声明区别

HTML：超文本标记语言，一种纯文本类型的语言。

HTML5.0：文档声明HTML5方便书写，精简，有利于程序员快速的阅读和开发。

2、结构语义区别

html：没有体现结构语义化的标签，如：<div id="nav"></div>

html5：添加了许多具有语义化的标签，如：<article>、<aside>、<audio>、<bdi>...

3、绘图区别

HTML：指可伸缩矢量图形，用于定义网络的基于矢量的图形。

HTML5：HTML5的canvas元素使用脚本（通常使用JavaScript）在网页上绘制图像，可以控制画布每一个像素。

什么是盒子模型？

一个盒子我们会分成几个部分：内容区(content)、内边距(padding)、边框(border)、外边距(margin)，也就是盒模型的组成由margin，padding，boder，content组成

盒子模型分为标准盒子模型和IE盒子模型

如何理解HTML5语义化？

HTML语义化标签

header – 标题

nav – 导航

article – 文章

section – 节或段

aside – 侧边栏

footer – 页脚

语义化的好处？

在没有css代码的情况下，也能很好的呈现内容结构、代码结构（让非技术员也能看懂代码）

提高用户体验，比如：title，alt用于解释名词和图片信息

利于SEO。语义化能和搜索引擎建立更好的联系，优化搜索

便于团队开发与维护，语义化更具有可读性

cookies、sessionStorage、localStorage的区别是什么？(浏览器)

1、cookie

（1）本身用于客户端和服务端通信

（2）但是它有本地存储的功能，于是就被“借用”

（3）document.cookie = …获取和修改即可

（4）cookie用于存储的缺点

①存储量太小，只有4kb

②所有http请求都带着，会影响获取资源的效率

③api简单，需要封装才能用document.cookie

2、localStorage,sesseionStorage

（1）html5专门为存储而设计，最大容量5M

（2）api简单易用

（3）lcoalStorage.setItem(key, value);localStorage.getItem(key);

（4）ios safari隐藏模式下:localStorage.getItem会报错，建议统一使用try-catch封装

3、sessionStorage用于本地存储一个会话（session）中的数据，这些数据只有在同一个会话中的页面才能访问并且当会话结束后数据也随之销毁。因此sessionStorage不是一种持久化的本地存储，仅仅是会话级别的存储。而localStorage用于持久化的本地存储，除非主动删除数据，否则数据是永远不会过期的。

常见的浏览器内核有哪些？

\*\*Trident内核：\*\*代表浏览器是ie浏览器，因此Trident内核又称E内核，此内核只能用于Windows平台，并且不是开源的。

\*\*Gecko内核：\*\*代表浏览器是Firefox浏览器。Gecko内核是开源的，最大优势是可以跨平台。

webkit ：Webkit内核:代表浏览器是Safari(苹果的浏览器)以及低版本的谷歌浏览器，是开源的项目。

\*\*Presto内核：\*\*代表浏览器是Opera浏览器(中文译为“欧朋浏览器”)，Presto内核是世界公认最快的渲染速度的引擎，但是在2013年之后，Open宣布加入谷歌阵营，弃用了该内核。

\*\*Blink内核：\*\*由谷歌和Opera开发，2013年4月发布，现在Chrome内核是Blink。

谈谈你对web标准以及W3C的理解？

web标准：

web标准主要分为结构、表现、行为3部分

结构：指我们平时在body里面写的标签，主要是由HTML标签组成

表现：指更加丰富HTML标签样式，主要由CSS样式组成

行为：指页面和用户的交互，主要由JS部分组成

W3C：

W3C对web标准提出了规范化的要求，即代码规范

对结构的要求

1、标签字母要小写

2、标签要闭合

3、标签不允许随意嵌套

对表现和行为的要求

1、建议使用外链CSS和js脚本，实现结构与表现分离、结构与行为分离，能提高页面的渲染效率，更快地显示网页内容

如何实现浏览器响应式布局？

使用媒体查询（@media）

使用flex弹性布局

使用百分比单位

使用rem单位

使用VH、HW单位

CSS选择器以及优先级的理解？

常用的CSS选择器

ID选择器、类选择器、标签选择器、属性选择器、伪类选择器、后代选择器

权重划分

在同一层级下：

!important > 内联样式 > ID选择器 > 类选择器 > （标签选择器、伪类选择器、属性选择器）

不同层级下：

正常来说权重值越高的优先级越高，但是一直以来没有具体的权重值划分，所以目前大多数开发中层级越深的优先级越高

谈谈你对回流和重绘的理解？

什么是回流？

当一个元素自身的宽高，布局，显示或隐藏，或元素内部的文字结构发生变化，导致需要重新构建页面的时候，就产生了回流

什么是重绘？

当一个元素自身的宽高，布局，及显示或隐藏没有改变，而只是改变了元素的外观风格的时候，就产生了重绘

什么时候会进行回流？

添加或者删除可见的 DOM 元素的时候

元素的位置发生改变

元素的尺寸发生改变

内容改变

页面第一次渲染的时候

什么时候会进行重绘？

列举一些相关的 CSS 样式：color、background、background-size、visibility、box-shadow

opacity: 0、visibility: hidden、display: none有什么区别？

opacity=0，该元素隐藏起来了，但不会改变页面布局，并且，如果该元素已经绑定一些事件，如click事件，那么点击该区域，也能触发点击事件的

visibility=hidden，该元素隐藏起来了，但不会改变页面布局，但是不会触发该元素已经绑定的事件

display=none，把元素隐藏起来，并且会改变页面布局，可以理解成在页面中把该元素删除掉一样

css 预处理器

less

sass

<img>的 title 和 alt 有什么区别

通常当鼠标滑动到元素上的时候显示

alt 是<img> 的特有属性，是图片内容的等价描述，用于图片无法加载显示、读屏器阅读图片。可提图片高可访问性，除了纯装饰图片外都必须设置有意义的值，搜索引擎会重点分析。

行内元素和块级元素有哪些？img属于什么元素

块元素

address – 地址

blockquote – 块引用

center – 举中对齐块

dir – 目录列表

div – 常用块级容易，也是CSS layout的主要标签

dl – 定义列表

fieldset – form控制组

form – 交互表单

h1 – 大标题

h2 – 副标题

h3 – 3级标题

h4 – 4级标题

h5 – 5级标题

h6 – 6级标题

hr – 水平分隔线

isindex – input prompt

menu – 菜单列表

noframes – frames可选内容，（对于不支持frame的浏览器显示此区块内容

noscript – 可选脚本内容（对于不支持script的浏览器显示此内容）

ol – 有序表单

p – 段落

pre – 格式化文本

table – 表格

ul – 无序列表

内联元素

a – 锚点

abbr – 缩写

acronym – 首字

b – 粗体(不推荐)

bdo – bidi override

big – 大字体

br – 换行

cite – 引用

code – 计算机代码(在引用源码的时候需要)

dfn – 定义字段

em – 强调

font – 字体设定(不推荐)

i – 斜体

img – 图片

input – 输入框

kbd – 定义键盘文本

label – 表格标签

q – 短引用

s – 中划线(不推荐)

samp – 定义范例计算机代码

select – 项目选择

small – 小字体文本

span – 常用内联容器，定义文本内区块

strike – 中划线

strong – 粗体强调

sub – 下标

sup – 上标

textarea – 多行文本输入框

tt – 电传文本

u – 下划线

img属于行内替换元素，效果与块元素一致

表单中readonly和disabled的区别

共同点：能够使用户不能改变表单中的内容

不同点：

1、readonly只对input和textarea有效，但是disabled对所有的表单元素都是有效的，包括radio、checkbox

2、readonly可以获取到焦点，只是不能修改。disabled设置的文本框无法获取焦点

3、如果表单的字段是disabled，则该字段不会发送（表单传值）和序列化

浏览器

1、浏览器中输入url到网页显示，整个过程发生了什么

域名解析

发起tcp三次握手

建立tcp连接之后发起htttp请求

服务器端响应http请求，浏览器得到html代码

浏览器器解析html代码，并请求html代码中的资源

浏览器对页面进行渲染呈现给用户

2、cookie的弊端

每个特定的域名下最多生成的cookie的个数有限制

IE和Opera会清理近期最少使用的cookie，firefox会随机清理cookie

cookie最大为4096字节，为了兼容一般不超过4095字节

安全性问题，如果cookie被人劫持，就可以获得所有的session信息

3、主流浏览器及内核

Google chrome：webkit/blink

safari：webkit

IE：trident

firefox：gecko

Opera：presto/webkit/blink

4、sessionStorage和localStorage的区别

sessionStorage用于本地存储一个会话session中的数据，这些数据只有在同一个会话中的页面才能访问并且当会话结束后数据会被销毁。

localStorage用于持久化的本地存储，除非主动删除数据，否则数据永远不会过期的。

5、谈谈对bfc规范的理解

bfc是block formatting context即格式化上下文

bfc是页面css视觉渲染的一部分，用于决定块盒子的布局及浮动相互影响范围的一个区域

bfc最重要的一个效果是，让处于bfc内部与外部的元素相互隔离，使内外的元素的定位不会相互影响

6、请说出至少三种减少页面加载时间的方法

尽量减少页面中重复的http请求

css样式放置在文件头部、js脚本放置在文件末尾

压缩合并js、css代码

服务器开启gzip压缩

7、 如何进行网站性能优化？

content 方面

减少 HTTP 请求：合并文件、CSS 精灵、inline Image

减少 DNS 查询：DNS 缓存、将资源分布到恰当数量的主机名

减少 DOM 元素数量

Server 方面

使用 CDN

配置 Etag

对组件使用 Gzip 压缩

Cookie 方面

减少cookie大小

css 方面

将样式表放到页面顶部

不使用 CSS 表达式

使用 <link> 不使用 @import

Javascript 方面

将脚本放到页面底部

将 javascript 和 css 从外部引入

压缩 javascript 和 css

删除不需要的脚本

减少 DOM 访问

图片方面

优化图片：根据实际颜色需要选择色深、压缩

优化 css 精灵

不要在 HTML 中拉伸图片

8、浏览器存储？

短暂性的时候，我们只需要将数据存在内存中，只在运行时可用

持久性存储，可以分为 浏览器端 与 服务器端

浏览器:

cookie : 通常用于存储用户身份，登录状态等，http 中自动携带， 体积上限为 4K ， 可自行设置过期时间

localStorage / sessionStorage : 长久储存/窗口关闭删除， 体积限制为 4~5M

indexDB

服务器

分布式缓存 redis

数据库

9、get / post？

get : 缓存、请求长度受限、会被历史保存记录

无副作用(不修改资源)，幂等(请求次数与资源无关)的场景

post : 安全、大数据、更多编码类型

10、安全性问题？

XSS 攻击: 注入恶意代码

cookie 设置 httpOnly

转义页面上的输入内容和输出内容

CSRF : 跨站请求伪造，防护:

get 不修改数据

不被第三方网站访问到用户的 cookie

设置白名单，不被第三方网站请求

请求校验

性能优化

1、性能优化的几个方面?

资源压缩合并,减少HTTP请求

非核心代码异步加载

利用浏览器缓存

使用CDN

预解析DNS

2、异步加载?

动态脚本加载

defer

async

3、加载方式区别？

defer是在html解析完毕才执行,如果有多个则按加载顺序执行

async是加载完毕后立即执行,如果是多个,执行顺序与加载顺序无关

4、预加载?

在开发中，可能会遇到这样的情况。有些资源不需要马上用到，但是希望尽早获取，这时候就可以使用预加载。

预加载其实是声明式的 fetch ，强制浏览器请求资源，并且不会阻塞 onload 事件，可以使用以下代码开启预加载

<link rel="preload" href="http://example.com">

预加载可以一定程度上降低首屏的加载时间，因为可以将一些不影响首屏但重要的文件延后加载，唯一缺点就是兼容性不好。

5、DNS 预解析?

DNS 解析也是需要时间的，可以通过预解析的方式来预先获得域名所对应的 IP。

<meta http-equiv='x-dns-prefetch-control' content='on'>

<link rel="dns-prefetch" href="//yuchengkai.cn">

在https协议中默认a标签不会开启预解析,因此需要手动设置meta

6、懒执行?

懒执行就是将某些逻辑延迟到使用时再计算。该技术可以用于首屏优化，对于某些耗时逻辑并不需要在首屏就使用的，就可以使用懒执行。懒执行需要唤醒，一般可以通过定时器或者事件的调用来唤醒。

7、懒加载?

懒加载就是将不关键的资源延后加载。

懒加载的原理就是只加载自定义区域（通常是可视区域，但也可以是即将进入可视区域）内需要加载的东西。对于图片来说，先设置图片标签的 src 属性为一张占位图，将真实的图片资源放入一个自定义属性中，当进入自定义区域时，就将自定义属性替换为 src 属性，这样图片就会去下载资源，实现了图片懒加载。

懒加载不仅可以用于图片，也可以使用在别的资源上。比如进入可视区域才开始播放视频等等。

react面试题

1、什么时候使用状态管理器?

从项目的整体看

目用户的使用方式复杂

不同身份的用户有不同的使用方式（比如普通用户和管理员）

多个用户之间可以协作

与服务器大量交互，或者使用了 WebSocket

View 要从多个来源获取数据

从组件角度看

某个组件的状态，需要共享

某个状态需要在任何地方都可以拿到

一个组件需要改变全局状态

一个组件需要改变另一个组件的状态

组件有相当大量的，随时间变化的数据

state 需要有一个单一可靠数据源

所有 state 放在顶层组件已经无法满足需求了

2、说说 React 有什么特点?

它使用\*\*虚拟 DOM \*\*而不是真正的 DOM。

它可以用服务器端渲染。

它遵循单向数据流或数据绑定

3、列出 React 的一些主要优点?

它提高了应用的性能

可以方便地在客户端和服务器端使用

由于 JSX，代码的可读性很好

React 很容易与 Meteor，Angular 等其他框架集成

使用 React，编写 UI 测试用例变得非常容易

4、什么是 JSX?

它 JSX 是 J avaScript XML 的简写。是 React 使用的一种文件，它利用 JavaScript 的表现力和类似 HTML 的模板语法。这使得 HTML 文件非常容易理解。此文件能使应用非常可靠，并能够提高其性能

例子

render() {

return(

<div>

<h1> Hello World </h1>

</div>

)

}

1

2

3

4

5

6

7

5、说说为什么浏览器无法读取 JSX?

浏览器只能处理 JavaScript 对象，而不能读取常规 JavaScript 对象中的 JSX。所以为了使

浏览器能够读取 JSX，首先，需要用像 Babel 这样的 JSX 转换器将 JSX 文件转换为

JavaScript 对象，然后再将其传给浏览器

6、你理解“在 React 中，一切都是组件”这句话?

组件是 React 应用 UI 的构建块。这些组件将整个 UI 分成小的独立并可重用的部分。每个

组件彼此独立，而不会影响 UI 的其余部分

7、 React 中 render()的目的?

每个 React 组件强制要求必须有一个 render()。它返回一个 React 元素，是原生 DOM 组件的表示。如果需要渲染多个 HTML 元素，则必须将它们组合在一个封闭标记内，例如<form>、<group>、<div> 等。此函数必须保持纯净，即必须每次调用时都返回相同的结果

8、什么是 Props?

Props 是 React 中属性的简写。它们是只读组件，必须保持纯，即不可变。它们总是在整个应用中从父组件传递到子组件。子组件永远不能将 prop 送回父组件。这有助于维护单向数据流，通常用于呈现动态生成的数据

9、React 中的状态是什么？

状态是 React 组件的核心，是数据的来源，必须尽可能简单。基本上状态是确定组件呈现和行为的对象。与 Props 不同，它们是可变的，并创建动态和交互式组件。可以通过this.state() 访问它们。

10、区分状态和 Props?

条件 State Props

从父组件中接受初始值 Yes Yes

父组件可以改变值 No Yes

在组件中设置默认值 No Yes

在组件的内部变化 Yes No

设置子组件的初始值 Yes Yes

在子组件的内部改变 No Yes

11、如何更新组件的状态?

使用this.setState()更新组件的状态

12、React 组件生命周期的阶段是什么?

React 组件的生命周期有三个不同的阶段：

初始渲染阶段：这是组件即将开始其生命之旅并进入 DOM 的阶段。

更新阶段：一旦组件被添加到 DOM，它只有在 prop 或状态发生变化时才可能更新和重新渲染。这些只发生在这个阶段

卸载阶段：这是组件生命周期的最后阶段，组件被销毁并从 DOM 中删除

13、你对 React 的 refs 有什么了解?

Refs 是 React 中引用的简写。它是一个有助于存储对特定的 React 元素或组件的引用的属性，它将由组件渲染配置函数返回。用于对 render()返回的特定元素或组件的引用。当需要进行 DOM 测量或向组件添加方法时，它们会派上用场

列出一些应该使用 refs 的情况?

需要管理焦点、选择文本或媒体播放时

触发式动画

与第三方 DOM 库集成

14、如何模块化 React 中的代码?

可以使用 export 和 import 属性来模块化代码。它们有助于在不同的文件中单独编写组件

15、什么是高阶组件 HOC?

高阶组件是重用组件逻辑的高级方法，是一种源于 React 的组件模式。 HOC 是自定义组件，在它之内包含另一个组件。它们可以接受子组件提供的任何动态，但不会修改或复制其输入组件中的任何行为。你可以认为 HOC 是“纯（Pure）”组件

16、你能用 HOC 做什么?

HOC 可用于许多任务:

用代码重用，逻辑和引导抽象

渲染劫持

状态抽象和控制

Props 控制

17、 React 中 key 的重要性是什么?

key 用于识别唯一的 Virtual DOM 元素及其驱动 UI 的相应数据。它们通过回收 DOM 中当前所有的元素来帮助 React 优化渲染。这些 key 必须是唯一的数字或字符串，React 只是重新排序元素而不是重新渲染它们。这可以提高应用程序的性能

18、MVC 框架的主要问题是什么?

key 用对 DOM 操作的代价非常高

程序运行缓慢且效率低下

内存浪费严重

由于循环依赖性，组件模型需要围绕 models 和 views 进行创建

19、请你解释一下 Flux?

Flux 是一种强制单向数据流的架构模式。它控制派生数据，并使用具有所有数据权限的中心store 实现多个组件之间的通信。整个应用中的数据更新必须只能在此处进行。 Flux 为应用提供稳定性并减少运行时的错误。

20、你对“单一事实来源”有什么理解

Redux 使用 “store” 将程序的整个状态存储在同一个地方。因此所有组件的状态都存储在store 中，并且它们从 store 本身接收更新。单一状态树可以更容易地跟踪随时间的变化，并调试或检查程序。

21、列出 Redux 的组件?

Redux 由以下组件组成:

Action 这是一个用来描述发生了什么事情的对象

Reducer 这是一个确定状态将如何变化的地方

Store 整个程序的状态/对象树保存在 Store 中

View 查只显示 Store 提供的数据

22、 Store 在 Redux 中的意义是什么?

store 是一个 JavaScript 对象，它可以保存程序的状态，并提供一些方法来访问状态、调度操作和注册侦听器。应用程序的整个状态/对象树保存在单一存储中。因此，Redux 非常简单且是可预测的。我们可以将中间件传递到 store 来处理数据，并记录改变存储状态的各种操作。所有操作都通过 Reducer 返回一个新状态

23、 Redux 有哪些优点?

结果的可预测性

可维护性

服务器端渲染

易于测试 -

24、 什么是 React 路由?

React 路由是一个构建在 React 之上的强大的路由库，它有助于向应用程序添加新的屏幕和流。这使 URL 与网页上显示的数据保持同步。它负责维护标准化的结构和行为，并用于开发单页 Web 应用。 React 路由有一个简单的 API。

25、说说你对 React 的渲染原理的理解?

单向数据流。React 是一个 MVVM 框架，简单来说是在 MVC 的模式下在前端部分拆分出数据层和视图层。单向数据流指的是只能由数据层的变化去影响视图层的变化，而不能反过来（除非双向绑定）

数据驱动视图。我们无需关注页面的 DOM，只需要关注数据即可

渲染过程，生命周期……

setState()大部分时候是异步执行的，提升性能。

26、React 中三种构建组件的方式?

React.createClass()、ES6 class 和无状态函数。

JQuery

说出jQuery中常见的几种函数以及他们的含义是什么?

（1）get()取得所有匹配的DOM元素集合；

（2）get(index)取得其中一个匹配的元素.index表示取得第几个匹配的元素；

（3）append(content)向每个匹配的元素内部追加内容；

（4）after(content)在每个匹配的元素之后插入内容；

（5）html()/html(var)取得或设置匹配元素的html内容；

（6）find(expr)搜索所有与指定表达式匹配的元素；

（7）bind(type,[data],fn)为每个匹配元素的特定事件绑定事件处理函数；

（8）empty()删除匹配的元素集合中所有的子节点；

（9）hover(over,out)一个模仿悬停事件(鼠标移动到一个对象上面及移出这个对象)的方法；

（10）attr(name)取得第一个匹配元素的属性值。

jQuery 能做什么？

获取页面的元素；修改页面的外观；改变页面大的内容；响应用户的页面操作；为页面添加动

态效果；无需刷新页面，即可以从服务器获取信息；简化常见的javascript任务。

如何将一个HTML元素添加到DOM树中的？

可以通过appendTo()方法在指定的DOM元素末尾添加一个现存的元素或者一个新的HTML元素。

什么是jQuery? jQuer能做什么？

jQuery是一套JavaScript的库，它简化了使用Javascript进行网页特效开发的一些复杂性，提供了对常见任务的自动化和复杂任务的简化

jQuery的作用

快速获取文档元素

提供漂亮的页面动态效果

创建Ajax无刷新网页

提供对JavaScript语言的增强

增强的事件处理

更改网页内容

JQuery可以修改网页中的内容,比如更改网页的文本、插入或转网页图像, jQuery简化了原本使用JavaScript代码需要处理的方式。

JQuery 的优点

1.利用css的选择器提供高速的元素查找行为。

2.提供了一个抽象层来标准化各种常见的任务，可以解决各种浏览器的兼容问题。

3.将复杂的代码简化，提供连缀编程模式，大大简化了代码的操作。

————————————————

版权声明：本文为CSDN博主「WSZFSNFK」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/WSZFSNFK/article/details/122376695