面试题

H5 移动 web 开发

1.H5 新特征有哪些? C3 新特征有哪些?

H5 新特性:

- 拖拽释放是一种常见的特性,即抓取对象以后拖放到另一个位置在 HTML5 中,拖放是一种标准的一部分,任何元素都可以拖放
- 自定义属性 data-id
- 一些语义化更好的内容标签(<header,nav,footer,aside 等)
- 音频,视频如果浏览器不支持自动播放,在属性中添加 autoplay(谷歌浏览器不支持音频自动播放,但是视频支持静音播放)
- 表单的各种控件 data,time,email,url,file,number

CSS3 新特性:

- 颜色:新增 RGBA,HSLA 模式
- 文字阴影(text-shadow(
- 边框:圆角(border-radius)边框阴影:box-shadow
- 盒子模型:box-sizing
- 背景:background-size,background-origin,background-clip
- 渐变:linnear-gradient,radial-gradient
    - 自定义动画:animate@keyfrom2\
- 过渡 :transition 可实现属性的渐变
- 字体图标,弹性布局,2D/3D 转换,图片边框

2.em 和 rem 区别

共同点：都是相对单位，都是相对元素字体大小

不同点：参考元素不同

em：参考元素自身字体大小

rem：参考根元素（html）字体大小

rem 的工作原理：监听视口变化，动态修改 html 根元素字体大小（动态修改 rem 基准值）

常用的单位

- px 绝对单位 网页布局最常用的单位， 一般用于pc端布局。 px 像素的意思。 版心是 1200px
- pt 点 常用于 印刷。 或者 ios 常用的单位。 我们前端基本不用。
- em 相对单位。 1em 就是当**前的一个文字**大小。 场景： 段落首行缩进 2个字。 text-indent: 2em;
- rem 相对单位。 相对于 **html标签** 的文字大小，跟其余标签没有任何关系。 场景： 做适配。
- vw 可以看做是适配的终极版本。 vw也是相对单位。 vw 把屏幕划分了 100等份。 场景： 做适配。

3.less 常用功能

- ○ 变量
- ○ 混合
- ○ 嵌套
- ○ 导入
- ○ 函数

4.清除浮动几种方式

出现浮动的原因及引起的问题?

浮动将元素排除在普通流之外,即元素脱离文档流,不占据空间,浮动元素碰到包含它的边界或者浮动元素的边界停留

清除浮动的原因:

(1)子元素浮动后,不占位置,父元素的高度无法被撑开,影响父元素的同级元素

(2)与浮动元素同级的非浮动元素会跟随其后

(3)如果不是第一个元素浮动,该元素 之前的元素也需要浮动,否则会影响页面显示的结果

- ○ 手动给父元素设置高度
- ○ 设置父元素 overflow：hidden
- ○ 双伪元素清除
- ○ 单伪元素清楚法
- ○ 额外标签法

5.元素居中几种方式

- ○ flex 布局：水平垂直居中
- ○ 定位：left 50% top50% 然后使用 translate 偏移自己宽高的一半
- ○ 子元素同时设置 margin-top 和 margin-left
- ▢ 设置子元素为行内块,父元素的 text-align 为 center,line-height 等于父元素高度
- ▢ 设置父元素 display:table,并且 vertical-align 为 middle

6.如何实现双飞翼(圣杯)布局

(1)利用定位实现两侧固定中间自适应

(2)利用 flex 布局实现两侧固定,中间自适应

(3)利用 BFC 块级格式化上下文.实现两侧固定中间自适应

7.解决塌陷的几种方案

1.设置父元素 overflow:hidden(原理:开启 BFC)

2.给父元素添加 border

3.使用 padding 代替 margin

8.css 的盒子模型

一个盒子由四部分组成:content,padding,border,margin

盒子模型分为两种:

(1)W3C 标准盒子模型(标准盒模型)

(2)IE 的标准盒子模型(怪异盒模型)

区别:

(1) width 不同:标准盒子模型 width 指的是内容区域,怪异盒子模型 width 指的是内容,边框,内边距总的宽度

(2)height 不同:标准盒子 content 的宽度 heright 指的是内容区域,怪异盒子模型 height 指的是内容,边框,内边距总的高度

(3) 盒子大小不同:

标准盒子大小=content+border+padding+margin

怪异盒子模型大小=content+border+paddin

9.CSS 选择器的权重

Important > 行内样式 >id 选择器 > 类选择器 > 标签 >通配符 > 继承

css 解析顺序是 从右往左

10.透明度 opacity 和 rgba 的区别

透明度:对整个盒子所有的内容生效(颜色 + 文本)

rgba:只对颜色生效

11.CSS 中哪些属性可以继承,哪些不可以?

能继承的属性:字体系列属性,文本类型属性(内联元素,块级元素,元素可见性,表格布局属性,列表布局属性)

不能继承的属性:

display:规定元素应该生成的框的类型

文本类型:vertical-align,text-decoration

盒子模型的属性:width,height,margin,padding,border

背景属性:background,background-color,background-image

定位属性:float,clear,position.....,.....

12.BFC 是什么?

BFC 是一种 css 渲染模式(类似于 flex 伸缩盒子一样,也是一种盒子模型,都是 css 自带的

使用 BFC 的方法(四种方式开启 BFC 盒子)

(1)float 不是 none(有 float 属性且不为 none,此时盒子自动变成 BFC 盒子)

(2)overflow 属性不是 visible(有 overflow 属性且不为 visble,此时盒子自动变成 BFC 盒子)

(3)position 属性不是 static 和 relative

(4)display 属性为一下值:table-cell,inline-block,table-caption

BFC 解决的问题

- ▢ 清除元素内部浮动
- ▢ 解决盒子 margin 合并问题(塌陷也可以解决)
- ▢ 实现元素宽度自适应多列布局
- ▢ 制作右侧盒子自适应宽度的问题(左侧盒子宽度固定,右侧宽度不固定)

13.有哪几种定位方式,他们的区别是什么?

静态定位(static) :静态定位就是标准流

相对定位:(relative) 1.需要配合方位属性实现移动 2.相对于自己原来的位置进行移动 3.在页面中占位置(没有脱标)

绝对定位:(absolute): 1.需要配合方位属性实现移动 2.默认相对于浏览器可视区进行移动 3.在页面中不占位置(已经脱标)

固定点位(fixed): 1.相对于浏览器窗口进行移动 2.在页面中不占位置(已经脱标)

14.Css 如何画一个三角形?

- ▢ 设置 div 有一定的宽高,给 div 设置边框
- ▢ 设置 div 宽高为 0,四边设置边框宽度
- ▢ 把其他三个边框透明度设置为 0,就得到一个三角形

15.display:noen 与 visibility:hidden 的区别

相同点:都是来设置元素的隐藏

不同点:(1)display:none 设置元素隐藏,不占据页面位置 (2)visibility:hidden 设置元素隐藏:会占据页面位置

(2)display:none 会引起回流(重排),visibility:hidden 会引起重绘

16.flexbox

flexbox 就是弹性布局,可以简便,响应式的实现各种页面布局

flex 容器属性有:

- [ ] flex-direction
- [ ] flex-wrap
- [ ] flex-flow
- [ ] justify-content
- [ ] align-items
- [ ] align-content

17.css 与 less sass 有哪些区别?

1.声明变量 less(@) sass($)

2.嵌套

3.混入(声明函数与函数调用

4.导入其它文件 @import 最后要加分号

18.静态页面访问慢解决方式

浏览器的并发连接数受到限制, 并不是无限发起异步请求的。 每款浏览器都有自己的默认并发连接数，而且浏览器默认对同一域下的资源，只保持一定的连接数，会阻塞过多的连接，这都会影响到浏览器对网页的加载速度

19.前后端交互

通过 js 发送 ajax 请求,就可以和后端进行交互,然后和后端处理 Json 格式数据

Javascript

1.javaSciprt 的基本数据类型和引用数据类型的区别?👌

基本数据类型有 Number,string,boolean,null,undefined

引用数据类型有 function(函数),Object(对象),Array(数组)

区别:存储方式不同:基本数据类型存储在栈内存,而引用数据内存存储在堆内存中

2.如何判断 JavaScript 的数据类型(typeof 和 instanceof)👌

使用 **typeof** 可以检测数据类型是基本数据类型还是引用数据类型,typeof null 返回的一个 object,是 js 中的 bug,他不是一个引用数据类型.现在 null 被认为是对象的占位符 技术上来说他仍然是个原始值 被 ECMAScript 沿用

instanceof 是检测构造函数的 `prototype` 属性是否出现在某个实例对象的原型链上

区别:相同点:typeof 与 instanceof 都是判断数据类型的方法

不同点:1.typeof 会返回一个变量的基本类型,instanceof 返回的是一个布尔值

2.instanceof 可以准确判断复杂引用类型数据类型,但是不能准确判断基本数据类型,typeof 可以准确判断基本数据类型(null 除外),但是引用数据类型中,除了函数类型外,其他的无法判断

> 📌 ' ' ,0 ,NaN, -0, undefind ,null, false 只有这 7 钟会转成 boolean 的 false 其他的都是 true

3.===和==的区别👌

===:比较等号两边的值和数据类型是否相等

==:比较两边的值是否相等

4.null 和 undefined 的区别

null 表示一个对象被定义了,值为空,undefined 表示没有被定义

相同点:【1】值相等 【2】转布尔类型都是 false

不同点: 【1】数据类型不同 【2】转 number 类型值不同

用法:

1.变量被声明了,但没有赋值时,就等于 undefinde

2.调用函数的时候,需要提供参数但是没有提供参数的时候,参数就等于 undefind

3.函数如果没有返回值,默认是 undefined

5.JavaScript 中声明情况下会返回 undefined 值👌

1.访问变量,但是没有初始化

2.访问不存在的属性

3.访问的函数的参数没有传参

4.访问变量的值是 undefined 的时候

5.在函数内 return 没有显示返回的任何内容

6.如何判断两个对象相等👌

1.Es6 中有一个判断两个对象的引用地址是否一致(不能判断值是否相等)

2.遍历对象的所有的属性名和属性值判断是否一致

7.什么是类数组(伪数组),如何转化为真数组👌

伪数组就是具有 length 属性,数组的下标索引,但是不具有数组的 push.pop 等方法,可以对真正地数据遍历方法遍历

使用 Array.from()来转化为真数组

8.如何遍历对象的属性👌

1.Object.keys()方法,会返回一个属性名数组

2.Object.getOwnPropertyNames()方法,返回一个由指定对象的所有自身属性组成的数组

3.for in 遍历对象的属性

4.遍历所有的自身属性和继承属性

9.JavaScript 中的作用域,预解析与变量声明提升👌

作用域分为全局作用域和局部作用域,块级作用域,全局作用域中声明的变量,对于函数内部和块级作用域都是可以访问到,局部作用域在函数内声明的变量,只能在函数内部使用,块级作用域,在 {} 内声明的变量只能在块级作用与内访问使用

预解析器:

javasrciot 代码的执行是由浏览中的解析器来执行的,执行过程:1.把变量的声明提升到当前作用域的最前面,只提升声明,不会提升赋值,2.把函数的声明提升到当前作用域的最前面,只会提升声明,也不会提升调用

变量提升

变量提升:定义变量的时候,变量的声明会被提升到作用域的最上面,变量的赋值不会提升

函数提升:会把当前作用域的函数声明提前到整个作用域的最前面

10.变量提升和函数提升的区别👌

相同点:变量声明和函数声明都会提升至对用作用域的最顶端

不同点:函数内声明的变量只会提升到函数作用域的最顶端,函数声明智慧提升函数声明的写法,函数表达式写法不存在函数提升,函数提升的优先级大于变量提升的优先级,函数提升在变量提升之上

11.什么是作用域链👌

作用域就是起作用的一个区域

作用域链:遵守一个就近原则，当使用一个变量的时候,会在当前作用域下寻找这个变量,如果没有找到,就去它上层作用域查找,直到找到这个变量或查找到全局作用域,如果全局找不到这个变量的话,会报错.

11.DOM 事件模型（事件捕获和事件冒泡）👌

事件捕获就是事件从外往内触发 事件在触发时 先触发 祖元素身上的同名事件 再触发 父元素身上的同名事件，最后才触发自己身上的事件

事件冒泡就是先触发自己身上的事件，然后触发祖先元素身上的同名事件

- ▢ 阻止事件冒泡的方法：
- ○ w3c 方法：e.stopPropagation()
- ○ IE 使用：e.cancelBubble=true
- ○ JQ：returnfalse

12.事件三要素👌

- ○ 事件源：就是被触发的 dom 对象
- ○ 事件类型：用什么方式触发，比如鼠标点击，经过
- ○ 事件处理函数：要做的事情

13.绑定事件和接触事件的区别👌

相同点：都可以为元素绑定事件

不同点：【1】.方法名不一样 【2】.参数不一样，addEventListener 接受三个参数，addchEvent 接受两个参数 【3】兼容性不同：attachEvent 只兼容 ie6-8 ，addEventListener 不支持 IE8 【4】this 不同，addEventListener 中 this 是当前绑定事件的对象，addachEvent 中的 this 是 window 【5】addEventListener 中事件的类型（事件名字）没有 on attachEvent 有 on

14.谈谈对事件委托的理解👌

事件就是给父元素注册事件,利用事件冒泡的原理委托子元素处理,冒泡到父元素身上触发父元素事件

- ○ 优点:减少注册事件,提高程序的性能

15.document.write 和 innerHTML 的区别?👌

- ○ 相同点:document.write 和 innerHTML 都是把内容添加到页面上的
- ○ 不同点:【1】document.write 是直接将内容写入页面的内容流，会导致页面全部重绘 ,innerHTML 将内容写入某个 DOM 节点，不会导致页面全部重绘【2】document.write 写入内容是字符串的 html
  innerHTML 是 HTMLElement 的属性，是一个元素的内部 html 内容

16.描述浏览器的渲染过程：DOM 树和渲染树的区别?👌

浏览器的渲染过程:

- 先解析 HTML 文件,构建 DOM 树
- 再解析 CSS,开始构建 CSS 树
- 合并 DOM 树和 CSS 树,构建渲染树
- 生成布局,浏览器在屏幕上'画出'渲染树中所有的节点
- 将布局绘制在屏幕上,显示整个页面

DOM 树和渲染树的区别:

- DOM 树与 HTML 标签一一对应，包括 head 和隐藏元素
- 渲染树不包括 head 和隐藏元素，大段文本的每一个行都是独立节点，每一个节点都有对应的 css 属性

17.如何最小化重绘和回流?

重绘:当元素的一部分属性发生了颜色,外观,背景的改变,就会形成重绘

回流:当元素的一部分或者全部因为几何的大小或者边距发生改变,就会形成回流

重绘不一定引起回流,回流一定会引起重绘

方法:

- 需要对元素进行复杂的操作时,可以先隐藏该元素,操作完成后再显示
- 需要创建多个 DOM 节点时,使用 DocumentFragment 创建完后一次性的加入 document
- 缓存 layout 属性值
- 批量修改元素样式

18.什么是闭包? 👌

- 闭包就是内层函数访问外层函数的变量就会形成闭包
- 作用:扩展变量的作用范围,封闭数据,防止变量污染,让外部的函数也能访问内部函数的变量
- 缺点:容易造成内存泄漏,占据内存
- 解决方案:在用完这个变量之后,给这个变量赋值为 null,利用 js 垃圾回收机制自动回收

19.什么是内存泄漏?哪些操作会造成内存泄漏?👌

内存泄漏就是该内存空间使用完毕后没有回收,一直占据该内存单元.

1.意外的全局变量(另一种意外的全局变量可能由 this 创建)

2.定时器,setTimeout 的第一个参数使用字符串,不是函数的话,会引发内存泄漏

3 没有清理对 DOM 元素的引用同样造成内存泄漏

4.使用事件(addEventlistener)监听的时候,在不监听的时候使用 removeEventlister 取消事件监听

20.说说你对原型(prototype)的理解?👌

在 javascript 中,原型是一个对象,通过原型可以实现对象的属性继承,javaScript 的函数对象中都包含了一个 prototype 属性,这个属性指向了构造函数的原型对象,但是这个属性是不能直接被访问的,为了方便查看这个对象的原型,火狐和谷歌的内核 javascript 引擎提供了一个__proto__这个非标准的访问器

原型的主要作用就是为了实现继承与扩展对象

21.常见的 js 中的继承有哪些?👌

- 原型链继承
- 借用构造函数继承
- 组合继承
- 型式继承
- 寄生式继承
- 寄生组合式继承

22.this 的指向👌

- 谁调用,this 就指向谁
- 在普通函数内,this 指向 window
- 在构造函数内,this 指向实例化对象
- 箭头函数内,没有 this 指向,它会去他的上一层寻找 this 指向

23.forin 和 forof 区别👌

- 功能不同:forin 是遍历数组下标,forof 是遍历数组元素
- 原型属性不同:forin 会遍历原型的元素 ,forof 不会遍历原型的元素
- 数据类型不同:forin 可以遍历 object 类型,forof 不可以遍历 object 类型

24.call 和 apply,bind 的区别👌

他们的共同点是都可以修改函数 this 指向
他们两个区别
第一个是传参方式不同： call 和 bind 是列表传参，apply 是数组或伪数组传参
第二个是执行机制不同：call 和 apply 是立即执行，bind 不会立即执行而是生成一个修改 this 之后的新函数

23.介绍下原型链👌

当访问一个对象的属性和方法时,会在当前的属性上查找,如果没有,会去__proto__原型上查找,如果还没有,就会去 object 原型对象上查找,一层一层往上查找形成原型链

原型链的作用是面向对象继承

对象访问原型链规则： 就近原则

24.js 防抖和节流👌

- 函数防抖就是单位时间内,频繁触发事件,以最后一次为准
- 函数节流就是单位时间内,频繁触发事件,只执行一次

函数防抖和函数节流的异同点:

- 相同点:都是减少事件触发的次数,从而提高页面性能

- 不同点:使用场景不同,防抖适用于用户主动触发事件(输入框输入)鼠标移入

  节流适用于事件本身高频率触发(滚动事件,鼠标移动)

25.new 操作符的执行流程 👌

- 在内存中创建一个空对象
- 构造函数的 this 指向这个对象
- 执行构造函数内代码,给这个对象添加属性
- 返回一个新对象

26.什么是深拷贝和浅拷贝? 👌

- 浅拷贝它只能拷贝对象的第一层,如果第一层有引用类型,拷贝的是内存地址,如果是简单类型,拷贝的是值
- 深拷贝它可以拷贝多层,不管是引用类型好好说简单类型,都会拷贝他们的值
- 实现深拷贝的方法:1.使用递归 2.使用 json 字符串 3.引用 lodash 的 js

27.localstorage 和 seesionstorage 区别 👌

相同点:都是用来存储字符串类型的数据

不同点:存储方式不同,localstorage 存储在硬盘中,页面关闭了还在,seesionstorage 存储在内存中,页面关闭或者刷新就消失了

28.什么是事件循环?事件循环的规则? 👌

事件循环是:js 编辑器 '解析与执行' 代码的一种规则

js 代码分为两大类:同步和异步,异步又分为宏任务,微任务

- [ ] 微任务:promise.then()　　await/async
- [ ] 宏任务:script 标签 ,事件处理函数, 定时器回调 ,ajax 回调

事件循环规则

- 先'解析'默认 script 标签,进入第一个宏任务(默认宏)
- 判断代码是同步还是异步
- 如果是同步;立即执行
- 如果是异步:微任务放入微队列,宏任务放入宏队列
- 当前同步执行完毕之后,开始执行异步队列
- 先清空微任务队列,之后再'解析'宏任务队列,此时完成一次事件循环
- 按照以上步骤反复解析执行每一个宏任务(事件循环就是按照相同的规则反复循环解析执行代码)

- [ ] 待办列表

任务（代码）	宏/微 任务	环境
<script>标签	宏任务	浏览器
事件	宏任务	浏览器
网络请求（Ajax）	宏任务	浏览器
setTimeout() 定时器	宏任务	浏览器 / Node
fs.readFile() 读取文件	宏任务	Node
Promise.then()	**微任务**	浏览器 / Node
async/await	**微任务**	浏览器 / Node

29.谈谈 Promise 的工作流程及原理? 👌(🔔)

原理:Promise 是用来解决开发中的回调地狱的问题

如何解决回调地狱:我们 Promise 是通过在上 一个 Promise 实例的 then 方法里面返回下一个 Promise 实例,形成 then 的链式调用来解决这个异步回调地狱

工作流程:当我们创建 Promise 的时候,它就会立即执行里面的异步代码,此时它的状态是进行中,剩下两种状态是由异步本身决定的,分别对应的是异步操作的成功和失败,当他成功或者失败之后,Promise 的状态会发生改变,从进行中到成功,与从进行中到失败,这个时候就会分别调用它的 resolve 和 reject 方法来处理这个异步的结果,reslove 会执行 Promise 的 then,reject 会执行 Promise 的 cacth

- [ ] promise 本身并没有改变异步的顺序, 而是通过操作异步结果的顺序从而‘间接’控制异步的顺序

30.async 异步函数的使用 👌

async 函数相当于 Promise 的高级写法,我们先要用 async 关键字来修饰 function 函数,修饰的作用是 能够在函数的内部使用 await 关键字,用 await 关键字来调用一个函数,awiat 得到的结果就是这个函数 then 里面的 reslove

31.谈谈你对 Js 垃圾回收机制的理解👌

Js 中堆内存数据在栈中被引用的时候,将会产生引用计数,当一个堆的引用计数为 0 的时候,js 会自动回收堆内存,这种处理堆内存的机制就被叫做垃圾回收机制

32.同步和异步的区别？ js 编辑器工作原理?👌

同步与异步区别:

- 顺序不同:同步有序,异步无序,同步优先执行
- 性能不同:异步不会阻塞线程,同步会阻塞线程
- 语法不同:异步有回调(异步一般有回调,但不是所有的回调都是异步)

js 编辑器工作原理:

- 从上往下依次'解析' 代码
- 判断代码是同步还是异步

  如果是同步,立即执行;如果是异步,不会立即执行,放入事件队列中
- 当前所有同步执行完,开始执行异步

33.数组常用的几种方法👌

- map 遍历:会映射数组,返回个新的数组,要 return
- forEach 遍历:只遍历数组,没有返回值
- filter 遍历:筛选数组,返回一个新的数组
- som 遍历:判断数组是否有一个条件符合(逻辑或)
- every 遍历:判断数组中是否所有条件都符合(逻辑非)
- findindex 遍历:查找引用类型数据符合条件的元素,如果有返回该元素下标,没有返回-1
- indexof 遍历:查找数值类符合条件的元素,如果有返回该元素的下标,没有返回-1

34.说说你对递归的理解👌

递归就是:在函数内部调用自己

递归的作用:是可以实现浅拷贝和深拷贝,遍历 dom 树

35.箭头函数与普通函数的区别👌

- 语法不同:箭头函数使用箭头定义,普通函数中没有
- 箭头函数不能用于构造函数
- 箭头函数中没有 this 指向,它的 this 指向父级普通函数的 this
- 箭头函数不具有 arguments 对象
- 箭头函数也不有 prototype 原型对象

36.var,let,const 之间的区别👌

var 声明变量可以重复声明,而 let 不可以重复声明

var 是不限于块级,而 let 受限于块级

const 定义不可变的量,改变了就会报错

const 和 let 一样不会与 window 相映射,致辞块级作用域,在声明上面访问变量会报错

37.ES6 新特性👌

(1)新增声明命令 let 和 const

(2)模板字符串

(3)对函数的扩展,一些默认参数

(4)箭头函数

(5)对象的扩展,解构对象,解构数组 ,object.keys()方法,object.assign()

(6)promise 对象

(7)improt 和 export 导出导入

38.介绍一下 Set 和 Map 的区别👌

set 用于数组去重,Map 用于数据存储

set:只有键值没有键名,类似于数组,可以遍历,方法有 add,delete

Map 本质上是键值对的集合,类似于集合,可以遍历,可以根各种数据格式转换

39.柯里化函数

函数柯里化就是能够接受多个参数的函数转化为接收单一参数的函数,并且返回接受余下参数且返回结构的新函数的技术

作用:实现参数复用,提前返回,延迟执行

40.图片懒加载

原理就是在图片没有进入可视区域时,不给 img 标签的 src 赋值,这样浏览器就不会请求了,等图片进入可视区域再给 src 赋值

可视区域距离顶部的高度,然后判断 scolltop 滚动值是否等于可视区域的高度,就赋值

懒加载思路

1.加载 loading 图片

2.判断哪些图片需要加载

3.隐形加载图片

4.替换成真图片

Ajax

1..POST 发送两次请求的作用 👌

- 检测服务器是否支持修改请求头,如果浏览器支持,就在第二次中发送真正地请求
- 检测服务器是否为同源请求,是否支持跨域

2.get 和 post 区别： 👌

- 1.传参方式不同：get 参数在请求行中发送，URL 后面拼接，post 参数是在请求体中发送
- 2.传参速度不同：get 传参快，post 传参慢
- 3.安全性不同：get 的安全性低，post 的安全性高
- 4.数据大小不同：get 有大小限制，一般 2~5MB，post 参数没有大小限制

3.常见的 HTTP 状态码 👌

- 200:请求成功
- 302 服务器重定向(服务器主动修改)
- 400:参数错误　401 未认证　403:没有权限　404:URL 错误　405:请求方法错误　413:文件大小超出限制
- 5xx:服务器问题

4.Ajax 工作流程👌

- ○ 创建 xhr 对象(XMLHttpReques)
- ○ 设置请求方法和地址
- ○ 发送请求
- ○ 注册响应事件

5.输入 URL 到呈现网页过程👌

- ○ DNS 解析:把网址域名解析成 ip 地址
- ○ TCP 三次握手:建立安全可靠的网络传输
- ○ HTTP 请求:网络传输协议(保证传输是安全可靠的)
- ○ 服务器响应后,渲染引擎开始渲染
  - ▢ 1.(解析 html 生成 dom 树)
  - ▢ 2.(解析 css 生成样式树
  - ▢ 3.(dom 树与样式树合并得到渲染树)
  - ▢ 4.(呈现页面)
- ○ 断开链接

6.HTTP 协议原理👌

http 协议就是网络传输协议,协议就是规定网络传输格式

- ○ 浏览器请求必须是请求报文,服务器请求必须是响应报文

请求报文有三个部分组成:

(1)请求行:请求方法和地址

(2)请求头:浏览器告诉服务器数据是什么格式

(3)请求体:请求参数

响应报文有三部分组成:

(1)响应行:服务器状态码,地址

(2)响应头:服务器告诉浏览器,响应的数据格式

(3)响应体:服务器响应数据

7.Ajax 技术组成部分👌

ajax 就是异步的 js 和 xml 对象

a:synchropnons(异步):异步不会立即执行,而是等同步执行完后在执行

J:javscript

A:and

x:xmlHttprequest:负责发送 aax 请求

xml 作用与 Json 一致,json 没有出来之前的跨平台数据格式都是 xml 完成的

8.XMLHTTPRequest 状态码👌

XMLHttpRequest 对象的状态码 （xhr.readyState）

0: 请求未建立（创建了 xhr 对象，但是还没调用 open)

1: 服务器连接已建立

1. 2:请求已接收（send 之后,服务器已经接收了请求
2. 3:请求处理中
3. :4:请求已完成，且响应已就绪 （ 4 状态码等同于 onload 事件 ）

9.HTTP 与 HTTPS 的区别 👌

1.Http 是超文本协议传输,信息是明文传输,Https 是具有安全性的 SSL 加密传输协议

2.HTTP 和 HTTPS 使用的是完全不同的连接方式,用的端口也不一样,前者是 80 端口,后者是 443 端口

3.HTTP 链接简单,是无状态的,HTTPS 是由 SSL+HTTP 协议构建的可进行加密传输,身份认证的网络协议,比 https 协议安全

10.介绍一下 websocket(高薪常问)

websocket 是一种网络通信协议,是 html5 开始提供的一种单个 TCP 链接上进行全双共工通,webscoket 中浏览器和服务器只要完成一次握手,两者之间就直接可以持久性链接,并进行双向数据通信

HTTP 通信请求只能由客户端向服务端发起连接,一个请求,一个回答,一次建立链接,一次断开链接,websocket 连接允许客户端和服务器之间进行全双工通信,客户端可以主动向服务器发送消息,而服务器端可以主动向客户端发送消息,websocket 只需要建立一次链接,就可以一直保持连接状态.

11.HTTP 缓存机制(高薪常问)

hppt 缓存有很多,cookie,localostorage 等等,浏览器缓存分为强缓存和协商缓存

(1)强缓存是利用 http 的返回头中的 Expires 或者 Cache-Control 两个字段来控制,用来表示资源的缓存时间,Expires:缓存过期时间,用来指定资源到期的时间,是服务器端的具体的时间点,cache-Control 是一个相对时间.

(2)协商缓存 304 在第一次请求的时候,返回的是 200 状态码,和响应头中返回的是 cache-Control,控制缓存使用时间/方式,在第二次发送请求的时候先看 max-age 如果过期了,在请求头设置 if-none-match 等于刚刚 Etag 值,去跟后台比较,如果相等就说明后台没有更新,前端提取本地的缓存继续使用

12.get 请求存在缓存的问题原因?怎么解决?

问题原因:如果 get 请求相同的 url,浏览器会默认使用之前的缓存数据,而不是重新请求接口,导致页面数据没有更新

解决:在请求前增加时间戳,使每个操作都是独立惟一的,这就保证了每一次请求时,url 都不同,就成功解决了浏览器的缓存问题

13.简单请求和复杂请求

这两种请求的区别主要在于是否会触发 cors 预检请求

简单请求:get,post

复杂请求: put、delete 方法的 ajax 请求,发送 json 格式的 ajax 请求,带自定义头的 ajax 请求

复杂请求会多发一次请求，例：我们向服务器发送 get 请求，浏览器会额外发送一个 options 请求，这个请求我们称为预请求，服务器也会做出“预响应”，预请求实际上是一种权限请求，只有预请求成功后，实际的请求才会执行，预请求也存在跨域问题

对于 **跨域** 的复杂请求会进行预检请求，预检请求是不会携带请求体和自定义的请求头的， 因此对于处理复杂请求的在自定义中间件，遇到预检请求，我们需要直接放行，否则会出现非预期的结果。如果不跨域，是没有预检请求的。

NodeJs

1.跨域是什么?什么叫做同源?👌

跨越就是 ajax 请求地址 和 网页地址 不同源

同源就是它们的协议名和端口号，主机号（ip 地址）完全一致

同源策略的好处：保护浏览器安全，防止被恶意攻击

- ▢ 当你的 ajax 地址和页面地址不一样时，浏览器会认为你的页面用 ajax 技术偷偷向别人服务器发送请求。为了保证你的安全，所以会拒绝这个请求。

2..解决跨域的方法👌

- ○ CORS 技术（主流）：后台实现，与前端无关
- ○ JSONP(原理)：1.使用 script 标签 src 属性发送请求【只有 ajax 才有跨域，其他方式没有跨域】

  2.浏览器会执行 srcipt 标签 src 属性响应的 js 代码
- ○ JSONP 实现流程：前端：【1】声明一个函数，服务器响应 js 之后，浏览器会执行这个函数 【2】使用 script 标签 src 属性发送请求 【3】添加一个额外参数：callback=函数名 后端：【4】接受 callback 参数，响应函数调用

3.前端 Web 优化方案👌

- ○ 减少 http 请求: 把精灵图,一些不经常使用的图标之类的,多个文件合成一个
- ○ 资源压缩: 把 js,css 等,一些第三方包压缩文件体积,使用 min 版本
- ○ 合理利用浏览器缓存: 某一些接口数据是固定的,服务器就没有必要每一次都响应数据,可以让浏览器进行缓存

4.对渐进式和非渐进式的理解👌

- ○ 渐进式就是我们使用框架文件的时候，需要用到其中的某一个部分，就导入那一个部分（不浪费资源）
- ○ 非渐进式就是需要用到其中的一个部分，就要导入全部文件（浪费资源）

5.POST 发两次请求作用👌

- ○ 询问服务器是否致辞修改请求头,如果浏览器支持,则在第二次中发送真正地请求
- ○ 检测服务器是否为同源请求,是否致辞跨域

6.nodejs 三种模块及加载原理

(1)nodejs 三种模式

- 内置模块:node 环境自带的,不需要下载
- 第三方模块:别人写的模块,需要下载
- 自己写的模块:需要使用路径导入

(2)require('路径')方法原理

- 如果 require()参数是'路径',直接通过路径寻找 js 文件并导入
- 如果 require()参数是'模块名'

a.先找核心模块,有就加载,没有就去找 node_module

b.找当前文件中 node_moudules 中的同名'文件夹'

c.如果可以找到'文件夹',则自动加载根目录下的 index.js

d.如果找不到,就会遵循就近原则,自己没有就往上级找,一直没有找到磁盘根目录就会去内置找,内置找不到就会报错

7.对 node.js 的了解,它的特点,适合做的业务? 👌

什么是 node:它既是开发平台,也是运行环境,本身是基于 google 的,他是 javscriptV8 引擎开发的,所以编写基于 javascript 语言,它的服务端功能必须在服务器运行

特点就是:他一个 javascript 运行环境,依赖于 ChormeV8 引擎进行代码解释,特征:单线程(一个应用对应一个进程,一个进程下面会有多个线程,每个线程用于处理任务)

适用业务:node.js 是单线程,非阻塞线路,事件驱动,它的特点决定了他适合做一些大量的 I/O 的东西,不需要大量计算的功能

webpack

1.npm run build 命令原理🔔

- 找到 webpack 真正命令:检查 package.json,找到 build 对应的真实的 webpack 命令
- 找 config 配置文件:检查有没有配置文件,如果有配置文件就加载配置文件配置,没有就加载默认配置
- 找入口文件,根据上一个步骤的配置文件信息,找到入口文件
- 编译模块,构建依赖关系图
- 开始编译,根据配置文件输出出口文件

2.什么是 webpack?它的优点是什么?

webpack 是一个静态模块打包工具,它是用来分析,压缩,打包代码的,webpack 的好处是减少了文件的体积,减少了文件的数量,提高网页加载的速度

3.什么是 loader?什么是 Plugin

loader 就是一个加载器,说白了就是用来解析文件的,webpack 原生只能解析 js 文件,如果想要其他文件也打包的话,就需要用到 loader,作用就是让 webpack 拥有加载和解析非 javascipt 文件的能力

Plugin 是一个插件,可以扩展 webpack 的功能,,让我们的 wevpack 变得更灵活,在 webpack 运行的生命周期中会有很多时间,plugin 可以监听这些事件,在核实的时机通过 webpack 提供的 API 改变输出结果

4.dependcies 和 devDependencies 两者的区别

dependcies 用于生产环境(生产依赖)

devDependencies 用于开发环境(开发依赖)

5.webpack 的构建流程?从读取配置到输出文件过程

(1)初始化参数:从配置文件和 Shell 语句中读取与合并参数,得到最终的参数

(2)开始编译:用得到的参数初始化 Compiler 对象,加载所有配置的插件,执行对象的 run 方法开始执行编译

(3)确定入口:根据配置中的 entry 找到所有的入口文件

(4)编译模块:从入口文件出发,调用所有配置 Loader 对模块进行翻译.再找出该模块依赖的模块,再递归本步直到所有入口依赖的文件都经过了本步骤的处理

(5)完成模块编译:使用 LOader 编译完所有模块后,得到每一个模块被翻译后的最终内容以及他们之间的依赖关系

(6)输出资源:根据入口和模块之间的依赖关系,组装成一个个包含多个模块的 chunk 再把每一个 Chunk 转换成一个单独的文件加入输出列表,这步是可以修改输出内容的最后机会

(7)输出完成:在确认输出内容后,根据配置确定输出的路径和文件名,把文件内容写到文件系统.

6.说一下 webpack 的热更新原理

热更新的作用:可以做到不用刷新浏览器二将新变更的模块替换旧的模块

热更新的核心定义:

(1)客户端从服务端拉去更新后的文件,准确的是就是 chunkdiff(chunk 需要更新的部分),实际上 WDS 与浏览器之间维护了一个 websocket,当本地资源发送变化时,WDS 会向浏览器推送更新,并带上构建时的 hash,让客户端与上一次资源进行对比

(2)客户端对比出差异后会向 WDS 发去 AJax 请求来获取更改内容(文件,列表,hash),这样客户端就可以再借助这些信息继续向 WDS 发起 jsonp 请求获取该 chunk 的增量更新

(3)后续的部分由 HotModulePlugin 来完成,提供了相关 Api 提供开发者针对自身场景处理

7.如何利用 webpack 来优化前端性能

1.压缩代码.ugfligJsPlugin 压缩 js 代码,mini-css-extrct-plugin 压缩 css 代码

2.利用 CDN 加速,将引用的静态资源修改为 Cdn 上对应的路径,可以利用 webpack 杜宇 output 参数和 loader 的 publicpath 参数来修改资源路径

3.删除死代码,css 需要使用 purify-css

4.提取公共代码.用 optimization,splitChunks 和 optimization,runtimeChunk 来代替

8.使用 webpack 开发时,你用过哪些可以提高效率的插件

1.webpack-dashboard:可以更友好的展示相关打包的信息

2.webpack-merge:提取公共配置,减少重复配置代码

4.size-plugin:监控资源体积变化,尽早发现问题

5.模块热替换

9.什么是长缓存?在 wevpack 中如何做到长缓存优化?

1.长缓存是浏览器在用户访问页面的时候,为了加快加载速度,会对用户访问的静态资源进行存储,但是每一次代码升级或者更新,都需要浏览器去下载新的代码,最方便和最简单的更新方式

2.具体的实现:在 webpack 中,可以在 output 给出输出的文件制定 chunkhash,并且分离经常更新的代码和框架代码,通过 NameModulesPlugin 或者 HashedModulesPlugin 使再次打包文件名不变

10.如何提高 webpack 的构建速度?

在多入口的情况下,使用 CommonsChunkPlugin 来提取公共代码

1.通过 externals 配置来提取常见库

2.利用 Dllplugin 和 DllReferencePlugin 预编译资源模块通过 DllPlugin 来对那些我们引用但绝对不会修改的 npm 包来进行预编译,再通过 DllReferncePlugin 将编译的模块加载进来

3.使用 Happypack 实现多线程加速编译

4.使用 Webpack-uglify-parallel 来提升 uglifyPlugin 的压缩速度,原理采用了多核并行压缩来提升压缩速度

5.使用 Tree-shaking 和 ScopeHoisting 来剔除多余代码

11.怎么实现 webpack 按需加载?什么是神奇注释?

按需加载:在 improt 中,improt 不仅仅 Es6 模块导入方式,还是一个类似于 require 的函数,我们可以通过 improt('module")的方式导入一个模块,improt()返回是一个 promise 对象:使用 improt()方式就可以实现

神奇注释:在 improt()里添加一些注释,如定义该 chunk 的名称,要过滤的文件,指定引入的文件等等,这类带有特殊功能的注释被称为神奇注释

vue

1.v-if 和 v-show 的区别:👌

- 原理不同:v-if 本质是元素新增与移除,v-show 本质是修改元素的 display 属性
- 场景不同:v-if 用于不需要频繁切换的元素,v-show 用于需要频繁切换的元素
- 性能不同:v-if 初始渲染性能高,切换性能低.v-show 初始渲染性能低,切换性能高

2.谈谈你对虚拟 DOM 的理解👌

虚拟 DOM 本质是一个 js 对象,它与真实 DOM 最大的区别是虚拟 DOM 只存储少部分核心属性,(作用)从而提高渲染效率

虚拟 DOM 工作流程:

- 把页面上的元素转换成虚拟 DOM
- 当 vue 数据变化的时候,使用 diff 算法对比新旧元素的虚拟 DOM 节点
- 只更新变化的部分

3.vue 中 key 值作用 👌

作用:给元素添加唯一标识符,可以让 vue 更准确的渲染元素

vue 会把页面 DOM 转换成虚拟 DOM,当页面数据变化的时候,vue 会使用 diff 算法对比新旧 DOM,当两个盒子元素是一致的时候,vue 不会更新元素,而是直接复用,如果给元素添加唯一标识符 key 值,vue 就会强制更新 key 值不同的元素

4.vue 中 key 值为什么不能是下标? 👌

当数组元素发生改变的时候,其他的元素下标也会变化,如果使用下标作为 key 值,那么当数组元素发生变化的时候,可能会导致其他元素下标变化从而引起 vue 的就地更新,如果使用 id 作为 key 值,那么当数组元素变化的时候,下标变化不会引起 vue 就地更新,vue 会复用相同的 id 元素,只会更新不同的 id 元素

5.vue 有没有遇到数据更新页面不变的情况?vue 修改 data 数据一定会更新吗?vue 有没有遇到哪些 bug? 👌

- 数组有些方法不会改变数组自身,而是得到一个新数组,这种情况数组变化不会引起页面更新
- 一般遇到这种情况,可以使用 vue.$set()方法强制更新数组 或者 把之前的数组给覆盖掉

6. 侦听器与计算机属性区别

- 作用不同:计算属性解决模板渲染冗余问题,侦听器侦听某个数据变化
- 属性不同:计算属性是新增一个属性;侦听器只能侦听 data 中的属性
- 🔔缓存机制不同:计算属性有缓存,侦听器没有缓存
- 🔔代码不同:计算属性不支持异步操作,侦听器支持
- 监听数量不同:计算属性可以监听多个数据变化,侦听器只能监听一个数据变化
- 初始执行时机不同:计算属性默认会执行一次,而侦听器只有数据在第一次变化之后才会执行

7 为什么组件的 data 是一个函数 👌

因为组件是需要在很多地方复用的

- 如果组件 data 是一个对象，每个地方都会引用相同的地址,。 一旦某一个地方修改，就会全部修改,影响其他的页面
- data 是一个函数，每一次复用组件的时候就会从调用这个函数返回一个新的对象。 这样组件在复用的时候就可以做到数据互不干扰。

8.scoped 的作用与原理? 👌

scoped 作用:设置组件的 css 作用域

scoped 工作原理:给组件添加一个自定义属性 data-v-xxx,通过属性选择器增加权重

9.四个阶段和 8 个钩子 👌

（1）初始化阶段：beforeCreate(创建 vue 实例,但是没有创建 data)，

created(创建了 data,但是没有挂载)

（2）挂载阶段：beforeMount(创建挂载点,没有渲染) ，

mounted(完成初始渲染,把 data 挂载到挂载点)

（3）更新阶段：beforeUpdate(检测 data 中数据变化,并没有完成页面更新)，

updated(把 vue 变化的数据更新到页面)

（4）销毁阶段 :beforeDestroy(检测到 vue 销毁)，

destroyed(完成销毁)

10.vue 初始渲染会执行哪些钩子 （4 个） 👌

- ○ beforeCreate
- ○ created
- ○ beforeMount
- ○ mounted

11.两个常用的钩子👌

created :一般用于发送 ajax（页面一加载需要发送 ajax 在这个钩子里面）

mounted：一般用于操作 DOM（页面一加载需要操作 DOM 再这个钩子里面）

有两个钩子会执行多次，其他钩子都是执行一次 beforeUpdate，updated

12.在 Vue 中遇见的 Bug👌

(1)在操作 data 中的数据的时候,发现数据没有变,数组有些方法会改变数组本身,有写方法不会改变数组本身,所以对数据进行在 vue 中有个内置方法,$set 可以强制更新数组

(2)在 created 和事件里面操作 DOM 的时候,ref 挂载了,拿不到,因为 vue 更新数据是异步微任务,代码默认同步,等所有同步走完才会走异步代码,vue 才更新,使用 $nextTick 等待当前页面 DOM 更新完后执行完后执行,

13.路由工作模式👌

路由模式分为 hash 路由,history 路由

默认 hash 模式:原理是 hash 值监听,改变页面 hash 值

history 模式:原理是改变路径,需要服务器配置

区别:

(1)路径不同:hash 模式有#号,history 模式没有#号 9

(2)history 模式会给服务器发送请求,需要服务器单独配置

hash 模式,上线之后,刷新页面,正常的

history 模式,上线之后,刷新页面,白屏

14.vuex 中 action 工作流程👌

(1)组件 dispatch 提交 actions 更新

(2)actions 执行异步请求数据

(3)actions 将请求到的数据提交给 mutations 更新

(4)mutations 同步更新 state 中的数据

15.Vuex 作用级五大组成部分👌

- vuex 作用： 全局数据管理 解决复杂的父子组件传值
- state 作用：存储数据

- getter 作用：派生数据。相当于 state 计算属性
- mutations 作用：修改 state 中的数据
- actions 作用： 异步更新数据
- module 作用：模块化处理 vuex 数据

16.$route 和 $router 的区别?👌

- routes : 数组。 路由匹配规则
- router ： 对象。 路由对象
- $router ： 对象。 用于跳转路由 和 传递参数
- $route ： 对象。 用于接收路由跳转参数

17.说出至少 4 个 Vue 指令及作用👌

- v-on 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-bind 动态绑定 作用： 及时对页面的数据进行更改, 可以简写成:分号
- v-slot: 缩写为#, 组件插槽
- v-for 根据数组的个数, 循环数组元素的同时还生成所在的标签
- v-show 显示内容
- v-if 显示与隐藏
- v-else 必须和 v-if 连用 不能单独使用 否则报错
- v-text 解析文本
- v-html 解析 html 标签

18.vue 组件传值👌

- 父传子
  - 1. 子组件 props 定义变量
  - 2.父组件在使用子组件时通过行内属性给 props 变量传值
  - 特点：单向数据流
- 子传父
  - 1.子组件：$emit 触发父的事件
  - 2.父在使用组件用 @ 自定义事件名=父的方法 (子把值带出来)
  - 特点：事件监听
- 非父子组件
  - vuex

19.Vue 的 nextTick 的原理是什么?

vue 更新 DOM 操作是异步微任务

nextTick 原理是 vue 通过异步队列控制 DOM 更新,

nextTick 底层是 promise,等待当前队列 DOM 完成更新后执行

20.组件使用 v-model(v-model 底层原理)

- ▢ v-model 本质就是 props:['value]和 $emit(input)的简写
- ▢ 流程: v-model 会做两件事(父传子 + 子传父)

  (1)把 v-model 绑定的值(v-model="值"),通过 props 传递给组件的 value

  (2)监听组件 input 事件,修改 v-model 的值为 input 事件的参数 $emit("input" ,事件参数)

21【高频】路由之间是怎么跳转的？有哪些方式👌

- 1、<router-link to="需要跳转到页面的路径">
- 2、this.$router.push()跳转到指定的 url，并在 history 中添加记录，点击回退返回到上一个页面
- 3、this.$router.replace()跳转到指定的 url，但是 history 中不会添加记录，点击回退到上上个页面
- 4、this.$touter.go(n)向前或者后跳转 n 个页面，n 可以是正数也可以是负数

22.Vue 最大的优势是什么?👌

1.vue 是一款轻量级的框架,vue 可以进行组件化开发,数据和视图相分离,使代码量减少,提高开发效率

2.vue 它最突出的优势在于对数据进行双向绑定,使用虚拟 DOM

1.双向数据绑定

2.数据驱动视图

3.组件化开发

4.数据和视图分离

5.单页面应用可以实现数据局部刷新,加快访问速度,提高用户体验

23.Vue 常用的修饰符有哪些?👌

.prevent:提交事件不在重复加载页面(阻止默认跳转)

.stop:阻止单击事件冒泡

.once:只执行一次这个事件

.enter:监听键盘 enter 键

24.为什么避免 v-for 和 v-if 在一起使用👌

Vue 处理指令时,v-for 比 v-if 具有更高的优先级,虽然用起来也没有报错,但是性能不高,如果有 5 个元素被 v-for 循环,v-if 也会分别执行 5 次

25.方法和计算机属性和侦听器的区别?👌

方法:

需要主动调用触发

不会缓存

计算属性:

监听多个属性:只要计算属性内部数据改变就会触发

有缓存机制

侦听器:

监听一个属性

不会缓存

26.请说下封装 vue 组件的过程👌

1.我会先分析需求, 确定业务的需求,把页面中可以复用的结构,样式以及功能

找出业务需求中存在复用的地方

2.具体步骤:Vue.component 或者在 new Vue 配置项 components 中,定义组件名,可以在 props 中接受给组件传的参数和值,子组件修改好数据后,想把数据传递给父组件.可以采用 $emit 方法

26.v-slot 插槽与作用域插槽👌

1.插槽的作用:父组件 传递 html 结构给 子组件

2.具名插槽作用:父组件 传递多个 html 结构 给子组件

3.作用域插槽作用:父组件给子组件传递插槽时,可以使用子组件内部的数据👌

插槽使用 2 个步骤

- ▢ 第一步:在子组件中定义一个插槽 <slot>默认值:如果父组件没有传递则默认显示</slot>
- ▢ 第二步:在父组件中传递结构:<子组件>父组件需要传递的结构</子组件>

具名插槽语法:

1.给子组件的<slot>添加 name 属性:name="插槽名"

2.父组件使用 v-slot:插槽名:给指定的插槽传递结构

注意:这个 v-slot 指令必须要写在<template>标签中,<template>是 html5 新增的一个语义化标签,模板的意思,这个标签本身不会被渲染,因此在页面上恐怖到的,这个标签类似于 div,就是一个空盒子容器,与 div 唯一的区别就是它不会渲染

27.请讲一下组件的命名规范👌

(1)给组件命名有两种方式,一种是使用链式命名"my-component",一种是使用大驼峰命名"Mycomponent"

(2)因为要遵循 W3c 规范中的自定义组件名(字母全小写且必须包含一个连字符),避免和当前以及未来的 Html 元素相冲突

28.请说出路由配置项常用的属性及作用👌

路由配置参数

- ▢ path:跳转路径
- ▢ component:路径相对于的组件

- [ ] name::命名路由
- [ ] children:子路由的配置参数(路由嵌套)
- [ ] props:路由解耦
- [ ] redirect:重定向路由

29.vue 路由作用与原理👌

路由作用:实现单页面应用

原理:监听 location 的 hash 值

30.自定义指令的方法有哪些?它有哪些钩子函数? 还有哪些钩子函数参数?

- [ ] 全局定义指令:在 vue 对象的 directive 方法里面有两个参数,一个是指令名称,另一个是函数.组件内定义指令:directives
- [ ] 钩子函数:bind(绑定事件触发),inserted(节点插入的时候触发),update(组件内相关更新)
- [ ] 钩子函数参数:el,binding

31.is 这个特性你有用过吗?主要是用在哪些方面?

is 作用:用于动态组件

32.跟 kccp-alive 有关的生命周期有哪些?

1.前言：在开发 Vue 项目的时候，大部分组件是没必要多次渲染的，所以 Vue 提供了一个内置组件 keep-alive 来缓存组件内部状态，避免重新渲染，在开发 Vue 项目的时候，大部分组件是没必要多次渲染的，所以 Vue 提供了一个内置组件 keep-alive 来缓存组件内部状态，避免重新渲染

2.生命周期函数：在被 keep-alive 包含的组件/路由中，会多出两个生命周期的钩子:activated 与 deactivated。

activated 钩子：`在在组件第一次渲染时会被调用`，之后在每次缓存组件被激活时调用。

Activated 钩子调用时机：`第一次进入缓存路由/组件`，在 mounted 后面，beforeRouteEnter 守卫传给 next 的回调函数之前调用，并且给因为组件被缓存了，再次进入缓存路由、组件时，不会触发这些钩子函数，beforeCreate created beforeMount mounted 都不会触发

deactivated 钩子：`组件被停用（离开路由）时调用`

deactivated 钩子调用时机：使用 keep-alive 就不会调用 beforeDestroy(组件销毁前钩子)和 destroyed(组件销毁)，因为组件没被销毁，被缓存起来了，这个钩子可以看作 beforeDestroy 的替代，如果你缓存了组件，要在组件销毁的的时候做一些事情，可以放在这个钩子里，组件内的离开当前路由钩子 beforeRouteLeave => 路由前置守卫 beforeEach =>全局后置钩子 afterEach => deactivated 离开缓存组件 => activated 进入缓存组件(如果你进入的也是缓存路由)

33..sync 与 v-model 区别

相同点:都是语法糖,都可以实现父子组件数据的双向通信

不同点:

(1)格式不同. v-model="num" ;"num.sync="num"

(2)v-model:@input +value

(3):num.sync:@update:num

(4)v-model 只能用一次 .sync 可以有多个

34.MVVM 和 MVC 区别是什么👌

MVVM 设计模式主要有三个部分组成,第一个是 view 层也就是我们页面的样式,第二个是 M 层页面需要的数据,第三个是 VM 层就是我们 vue 的实例,MVVM 最大的优势 VM 可以同时和 M 跟 V 建立双向绑定,当页面变化了数据会自动变化,不需要更新 Dom

MVC 是传统设计模式,M 是 model 模型的意思,处理应用程序数据逻辑的部分,V 是 view 视图,处理数据显示的部分,C 是 controler 控制器的意思,处理用户交互的部分 MVC 设计模式最大思想就是在于分层,业务逻辑全部放 model,控制器只负责处理后端的交互

35.什么是 diff 算法👌

Diff 算法是一种对比算法。对比两者是 旧虚拟 DOM 和新虚拟 DOM，对比出是哪个 虚拟节点更改了，找出这个 虚拟节点并只更新这个虚拟节点所对应的 真实节点而不用更新其他数据没发生改变的节点，实现 精准地更新真实 DOM，进而 提高效率

36.vue2 中过滤器 filter👌

filter 过滤器就是一种方法,不改变原始数据,只对数据进行加工

1.一个表达式可以使用多个过滤器,过滤器之间用 '| ' 隔开

2.执行顺序,从左往右

3.使用场景:单位转换,文本格式化,时间格式化等

小程序

1.小程序有哪些优化性能的方式?

1.减少 this.setData 的调用次数,可以将多次 setData 合并成一次

2.减少每次 setData 的数据传输量,每次尽量精确传输只发生更改的数据(因为小程序是一个双线程模型,每次通信它很消耗性能,减少通信次数以及传输量),上拉加载更多页,考虑使用防抖节流

3.提高页面加载速度,控制包的大小,压缩代码,清理无用的代码,使用外部图片,采用分包策略,分包预加载,独立分包

2.小程序登录流程

调用 uni.getUserinfo Api 获取加密的用户信息,调用 uni.login API 获取临时登录凭证 code 码,将 code 给后端,后端通过 code,appid,appseceret 调用接口,返回 openid 和 session-key,后端通过 openid 和 seeion-key 生成 token 后返回前端,前端将后端返回的 token,发送登录请求,使用 vuex 持久化把用户的信息及头像存储在 loclstorage 实现登录

3.小程序的兼容性问题

1.ios 做 new Date()时,时间格式不兼容, ios 日期只支持 2020/08/08,解决办法:运用正则转为需要的 ios 兼容格式

2.API wx.request()返回的状态码, 安卓和 ios 不一致

返回状态码 res.statusCode 的值在 ios 下是 int 型数据,而在 Android6.0.1 上却是 String 型数据.正确的做法是使用== 而不是使用===来判断

4.小程序支付流程

5.小程序生命周期

页面生命周期:onLoad(发送请求获取数据)

onShow(页面显示,请求数据)

onReady(监听页面初次渲染完成 获取页面元素)

onHide(监听页面隐藏,终止任务,如定时器)

onUnload(监听页面卸载)

组件的生命周期: created:监听页面加载

attached:监听页面显示

ready:监听页面初次渲染完成

moved:监听页面隐藏

detached:监听页面写在

error:每当组件方法抛出错误时执行

6.小程序的优缺点:

优点:

1.随搜随用,用完既走,是的小程序代替许多 App

2.流量大,易接受,小程序借助自身平台更多容易引入更多流量

3.安全

4.开发门槛低

5.降低兼容性限制

缺点:

1.体积受限:微信小程序只有 2M 大小,无法开发大型一些的小程序

2.受控微信,比起 App,尤其是安卓版的高自由度,小程序要面对来自微信的限制

3.支付流程

使用 uni.login 获取 code => 将获取的 code 吗发送后台,使用微信提供的登录凭证校验接口进行请求,获取用户的 openid,生成业务订单,创建微信支付请求对象向微信发起支付请求,返回请求成功的微信订单数据,传到前端,在小程序端发起微信支付,支付成功后,微信向业务后台发送回调通知,业务操作修改支付状态,修改库存数量等

Vue3

1.vue3 和 vue2 的区别👌

(1)原理不同:vue2 中数据的双向绑定原理是 object.definePropety,递归数据劫持,object.defineProperty . 无法监控数组方法,导致通过数组添加元素,不能实时响应,它只能劫持对象的属性,从而需要对每. . 个对象,每个属性进行遍历,如果属性值是对象的话,还需要深度遍历,从而需要对每个对象,每个属 . 性进行遍历

vue3 中数据的双向绑定原理是 new Proxy,proxy 可以直接对整个对象劫持,还可以代理数组,和 . 代理动态增加的属性,大大优化了响应式监听的性能

(2)api 不同: vue2 使用的是选项类型 APi(Options API)

vue3 使用的是组合式 APi(Composition AP) 康坡嘴 xien

(3)源码是 ts 重写的,有更好的类型推导(类型检测更为严格稳定)

(4)vu2 只有一个根元素,而 vue3 模板可以有多个根元素

2.proxy 代理

proxy 就是一层拦截器,外界对该对象进行访问,必须经过这个拦截器,如果访问的目标对象属性不存在就会希望抛出一个错误,如果没有代理/拦截操作,都会返回 undefined

proxy 可以代理对象里面的所有属性,他不仅控制获取值,还可以控制删除属性,判断属性还有构造器等,new 一个 proxy,接受两个参数,第一个 target(我要代理谁,target 就是谁),第二个是 hander,他是一个对象,定义了我们拦截的行为,行为有很多,常用的就是 get,set,

get 是拦截对象属性的读取,而 set 是拦截对象属性的设置,get 对应有三个参数,常用的有两个参数,第一个是 target,第二个是一个 key 值,而 set 有四个参数,常用的有三个参数,target,属性名称,值,

get:用来拦截对目标对象属性的访问

3.动态给 vue 的 data 添加一个新的属性时会发生什么？怎样解决?

- ue2 中，劫持数据的操作在实例创建完成就已经进行完毕了，所以对对象后续新增的属性是劫持不到的，也就意味着后续新增的属性是不具备响应式能力的，可以通过 this.$set 或 Vue.set 来解决，语法是 Vue.set(对象, 属性, 值)。
- Vue3 响应式的原理换成了 proxy 就不存在这个问题了，因为它劫持的是整个对象，而后续添加的属性也是属于这个对象的。

TS

1.type 如何实现继承

```Vue
type Point2D = { x: number; y: number }

// 继承 Point2D

type Point3D = {

  z: number

} & Point2D
```

2.说说你对 TypeScript 的理解？与 JavaScript 的区别?

1.TS 是 js 的超级,是 js 的语法糖,可以在 ts 文件里写任意的 js 代码

2.ts 在浏览器和 nodejs 环境下不能直接运行,需要经过编译转换成 js 才能在浏览器和 nodejs 里面执行

3.现在的 vue3 源码底层都是 ts 书写,vue2 和 react 源码底层是 Flow 系统类型

3.typescript 的数据类型有哪些?

1.ts 和 js 共有类型有:boolean,number,string,array,object

2.Ts 独有的数据类型 any(任意类型),unknown(),void 类型,never,tuple(元组类型),enum(枚举类型)

4.说说你对 TypeScript 中枚举类型的理解？应用场景?

枚举是组织收集有关联变量的一种方式 ,每个枚举成员都有一个 name 和一个 value。数字枚举成员值的默认类型是 number 类型

5.说说你对 TypeScript 中接口的理解？应用场景

计算机基础

1.七层网络模型

应用层:用户的应用程序(微信,qq 等)

表达层:发送图片,文字,语言信息进行解码和编码

会话层:建立会话.本地主机和远程主机

传输层:网络通信(upd 比 tcp 传输快,upd 会丢包,tcp 不会丢包)

网络层: 网络地址

数据链路层: 中间交换机

物理层:定义怎么用物理信号来表示数据

TCP 属于传输层,HTTP 属于应用层,IP 属于网络层