1. **HTML+CSS部分**
2. px，em，rem 的区别？

px 像素，是相对于显示器屏幕分辨率而言的，是一个虚拟长度单位，是计算机系统的数字化图像长度单位。

em 是相对长度单位，相对于当前对象内文本的字体尺寸。如当前对行内文本的字体尺寸未被人为设置，则相对于浏览器的默认字体尺寸。它会继承父级元素的字体大小，因此并不是一个固定的值。

rem 是 CSS3 新增的一个相对单位（root em，根 em），使用 rem 为元素设定字体大小时，仍然是相对大小，但相对的只是 HTML 根元素。

区别：IE 无法调整那些使用 px 作为单位的字体大小，而 em 和 rem 可以缩放，rem 相对的只是 HTML 根元素。这个单位可谓集相对大小和绝对大小的优点于一身，通过它既可以做到只修改根元素就成比例地调整所有字体大小，又可以避免字体大小逐层复合的连锁反应。目前，除了IE8 及更早版本外，所有浏览器均已支持 rem。

1. 什么的BFC?

2.1块级格式化上下文，它是一个独立的渲染区域，只有block-levelbox参与，它规定了内部block-levelbox如何布局，并且与这个区域外部毫不相干。

2.2布局方式：

1. 内部的box会在垂直方向，一个接一个地放位置。
2. box垂直方向的距离由margin决定，属于同一个BFC的两个相邻的box的margin会发生重叠。
3. BFC的区域不会和float box重叠。
4. 计算BFC的高度的时候，浮动元素也参与计算。

2.3 哪些元素会生成BFC？

a. 根元素

b. float元素不为none

c. position为absolute或者fixed

d.overflow不为visible

e.display为inline-block,table-cell,table-caption,flex,inline-fixed

1. css引入的方式有哪些？Link和@import的区别是什么？

Link:内联，内嵌，没有兼容，支持使用JavaScript改变样式，后者不可以。

@import：外联，导入，css2.1以下的浏览器不支持

1. 清楚浮动的几种方式：

在父级div加height(高度)或者overflow：hidden

结尾处加空div标签clear：both

1. 简述对web语义化的理解？

就是让浏览器更好的读懂你写的代码，在进行HTML结构、表现、行为设计时，尽量使用语义化的标签，使程序代码简介明了，易于进行Web操作和网站SEO，方便团队协作的一种标准，以图实现一种“无障碍”的Web开发。

1. display：none；与visibility：hidden的区别是什么？

display：none；使用该属性后，HTML元素（对象）的宽度、高度等各种属性值都将“丢失”；

visibility：hidden；使用该属性后，HTML元素（对象）仅仅是在视觉上看不见（完全透明），而它所占据的空间位置仍然存在，也即是说它仍具有高度、宽度等属性值。

1. [jQuery的选择器有哪几种类型？](https://www.zhihu.com/question/329845069/answer/719557834" \t "https://www.zhihu.com/_blank)

**基础选择器：**

ID选择器 #id；$("#test")

类选择器 .class；$(".test")

元素选择器 element；$("p") 选取所有的<p>元素

\*通配符选择器；$("\*") 选取所有的元素

**层次选择器：**

后代选择器；$("p span") 选取<p>元素里的所有的<span>元素

子选择器 $("parent>child")；$("p>span") 选择<p>元素下的所有<span>元素

同辈选择器 $("prev~siblings")；选取prev元素后的所有siblings元素；如：$("#two~p")选取id为two的元素后所有<p>同辈元素集合

同辈选择器 $("prev+next")；选取紧接在prev元素后的next元素，如；$(".one+p") 选取class为one的下一个<p>同辈元素集合

**过滤选择器：**

:first：选取第一个元素

 :last：选取最后一个元素

1. 响应式布局、弹性盒布局、rem布局、流式布局、网格布局分别是什么？

**响应式布局：**简单点说，就是做一个网站同时能兼容多个终端，由一个网站转变成多个网站,响应式布局它是通过CSS中Media Query（媒介查询）@media功能，来判断我们的终端设备宽度在多少像素内，然后就执行与之对应的CSS样式。用于解决不用设备之间不用分辨率之间的兼容问题(一般是指PC，平板，手机等设备之间较大的分辨率差异)

那么响应式布局的优点和缺点又有哪些呢？  
优点：  
a.面对不同分辨率设备灵活性强  
b.能够快捷解决多设备显示适应问题   
缺点：不能完全兼容所有浏览器，代码累赘，加载时间加长

**弹性盒布局：**是一种当页面需要适应不同的屏幕大小以及设备类型时确保元素拥有恰当的行为的布局方式。引入弹性盒布局模型对一个容器中的子元素进行排列、对齐和分配空白空间。

**区别：**

用响应式布局来进行布局是按比例分配全部空间；flex 主要是按比例分配多余空间，另外还有很多控制盒子排列、对齐方式的特性。

**rem布局：**rem是弹性布局的一种实现方式，rem布局的本质是等比缩放，一般是基于宽度

**流式布局：**主要是靠百分比进行排版，可以在不同分辨率下显示相同的版式。用于解决类似的设备不同分辨率之间的兼容(一般分辨率差异较少)

**网格布局：**把平面划分成多个（如果是基本网格布局，那么网格是等大的）网格。  
所有内容元素在其中摆放均由格子的位置编号以及所占格子数量来确定。

1. **HTML+CSS3部分**

1.CSS3新特性有哪些？

a.颜色：新增RGBA，HSLA模式

b.文字阴影（text-shadow、）

c.边框：圆角（border-radius）边框阴影：box-shadow

d.盒子模型：box-sizing

e.背景：background-size 设置背景图片的尺寸

g.自定义动画animate

h.媒体查询，多栏布局

i.2D,3D转换

j.弹性布局flex

2.CSS3的兼容问题怎么处理？

一般加私有前缀否则不做特殊处理，再有就是遵循2大原则，渐进增强和优雅降级

1. **JavaScript部分**
2. 闭包及其作用

1.1闭包：有权访问另外一个函数作用域中的变量的函数.可以理解为(能够读取其他函数内部变量的函数)

1.2闭包的作用: 正常函数执行完毕后,里面声明的变量被垃圾回收处理掉,但是闭包可以让作用域里的变量,在函数执行完之后依旧保持没有被垃圾回收处理掉

1. 不会污染全局环境，2.可以进行形参的记忆，减少形参的个数，延长形参的生命周期。3.方便进行模块化开发。

1.3闭包应用的场景：闭包随处可见，一个 Ajax 请求的成功回调，一个事件绑定的回调方法，一个 setTimeout 的延时回调，或者一个函数内部返回另一个匿名函数，这些都是闭包。简而言之，无论使用何种方式对函数类型的值进行传递，当函数在别处被调用时，都有闭包的身影

1. 垃圾回收机制

垃圾收集器会定期（周期性）找出那些不在继续使用的变量，然后释放其内存。但是这个过程不是实时的，因为其开销比较大，所以垃圾回收器会按照固定的时间间隔周期性的执行。垃圾回收器周期性运行，如果分配的内存非常多，那么回收工作也会很艰巨。

1. 讲一下js和native之间的交互问题？

**在IOS中：**

**native调用js:**就是app通过webview来获取window对象，从而可以调用window对象中的方法，而获取方式跨域使用其中stringByEvaluatingJavaScriptFromString:执行一段js脚步，并且返回执行结果这个方法。

例如：webview.stringByEvaluatingJavaScriptFromString(“Math.random()”)

**js调用native:**虽然native每个版本都会提供api，但是我们并不能直接拿来用，unwebview有一个特性：通过unwebview发起的网络请求可以在native层被拦截和通知。我们就可以自定义一个请求，当检测到这个请求是我们约定的交互请求时就阻止网络请求进而执行我们需要的native功能，这样就实现了js调用native。

**在Android 中 ：**

**js调用native：**有三种方式：

1.app安装后会在手机上注册一个schema，比如native会有一个进程监控webview发出的所有schema://请求，然后分发到“控制器”hybirdapi处理程序，native控制器处理时会需要param提供的参数（encode过），处理结束后将携带数据获取webview window对象中的callback然后调用。

2.native向window对象注入native对象，从而使js能使用这个对象，达到js交互效果。

3.在Android webview这一层是可以重写js原生中的prompt，console.log，alert这三个方法。

**naive调用js：**和ios一样的，获取window对象,不过使用的方法是loadUrl

1. json字符串和json对象怎么相互转化？

JSON.parse() 从一个字符串中解析出json对象

JSON.stringify()从一个对象中解析出字符串

1. http和https的区别是什么？

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，后者是443。

https是以安全为目标的http通道，简单讲是http的安全版。

由于https要还密钥和确认加密算法的过程，所以更安全

1. get和post的区别？

客户端对服务器的请求常用类型主要有四种

GET（从服务器获取）

POST（向服务器发送请求数据）

PUT（更新）

DELETE（删除）

post和get的区别

get是所有参数全部包装在URL中，明文显示，且服务器的访问日志会记录，非常不安全。

post的URL中只有资源路径，不包含参数，参数封装在二进制的数据体中，服务器也不会记录参数，相对安全。所有涉及用户隐私的数据都要用post传输。

GET：不同的浏览器和服务器不同，一般显示在2-8k之间，更加常见的是在1k以内。

POST：方法提交的数据比较大，大小靠服务器的设定值限制，php默认的是2M。

1. js中深拷贝和浅拷贝的区别？

**浅拷贝**只复制指向某个对象的指针，而不复制对象本身，新旧对象还是共享同一块内存。

**深拷贝**会另外创造一个一模一样的对象，新对象跟原对象不共享内存，修改新对象不会改到原对象。

深拷贝的方法：

a.递归拷贝

b.使用Object.create()方法

c.jquery 有提供一个$.extend也可以实现

d.函数库lodash，也有提供cloneDeep用来实现

1. Class（es6）继承和prototype继承的区别是什么？

以前JS传统做法是当生成一个对象实例，需要先定义构造函数，然后通过prototype的方式来添加方法,再生成实例。

其实class 说到底也是一种语法糖。 在ES5中原本的构造函数被constructor替代了,本来需要定义在prototype上面的方法直接定义在class里面即可。

1. map,filter,reduce,find,forEach

**forEach:**只是简单的遍历数组

**map:**遍历原数组，生成新数组，将数组的每一个元素拿出来处理完成之后再放回去

**filter：**遍历数组，生成新数组，判断每个值返回结果是否为true，是true的放进去

**reduce**主要是为了所有数组进行累加，最后返回一个值，不改变原数组

**find：**遍历数组找到第一个与之匹配的值。

1. New

调用new干了什么？

1. 创建一个新的对象
2. 将对象里面的this指向这个新的对象
3. 将新对象连接到构造函数的原型
4. 返回这个对象
5. addEventListener方法

将监听器注册到指定的标签上，当该标签触发指定事件的时候，指定的回调函数就会被执行，addEventListenner中可以传递三个参数target.addEventListener(type,listener[,options]);

type表示监听事件类型的字符串。

listener当监听的事件类型触发时，回调的一个函数。

options传入一个对象或者布尔

a.布尔capture指定事件是否在捕获或者冒泡阶段执行

b.布尔once表示当前事件最多只调用一次

c.布尔 passive表示listener永远不会调用preventDefault()。如果listener仍然调用了这个函数，客户端将会忽略它并抛出一个控制台警告

8.Cookie localStorage sessionStorage  的异同？

**a.存储大小限制也不同**cookie数据不能超过4k，同时因为每次http请求都会携带cookie，所以cookie只适合保存很小的数据，如会话标识。sessionStorage和localStorage 虽然也有存储大小的限制，但比cookie大得多，sessionStorage约5M、localStorage约20M

**b.数据有效期不同：**sessionStorage：仅在当前浏览器窗口关闭前有效，自然也就不可能持久保持；localStorage：始终有效，窗口或浏览器关闭也一直保存，因此用作持久数据；cookie只在设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭。

**c.作用域不同：**sessionStorage在相同的浏览器窗口中共享，即使是同一个页面localStorage、cookie 在所有同源窗口中都是共享的。

9.箭头函数

箭头函数表达式的语法比函数表达式更短，并且没有自己的this，argument，super或new.target。这些函数表达式更适用于那些本来需要匿名函数的地方，并且它们不能用作构造函数。

10.module、export、import是什么，有什么作用？

module、export、import是ES6用来统一前端模块化方案的设计思路和实现方案.

export、import的出现统一了前端模块化的实现方案，整合规范了浏览器/服务端的模块化方法，用来取代传统的AMD/CMD、requireJS、seaJS、commondJS等等一系列前端模块不同的实现方案，使前端模块化更加统一规范，JS也能更加能实现大型的应用程序开发。

import引入的模块是静态加载（编译阶段加载）而不是动态加载（运行时加载）。

import引入export导出的接口值是动态绑定关系，即通过该接口，可以取到模块内部实时的值。import用于在一个模块中加载另一个含有export接口的模块。

export用于对外输出本模块（一个文件可以理解为一个模块）变量的接口。

import和export命令只能在模块的顶部，不能在代码块之中。

11.AJAX的原理?

Ajax的原理简单来说通过XMLHttpRequest对象来向服务器发出异步请求，从服务器获取数据，然后用JavaScript来操作DOM而更新页面，这其中最关键的一步就是从服务器获取请求数据。XMLHttpRequest是ajax的核心机制，它是IE5中首先引入的，是一种支持异步请求的技术，简单来说，也就是JavaScript可以及时向服务器提出请求和处理响应，而不是阻塞用户达到无刷新的效果。

12.AXjx请求过程

(1)创建XMLHttpRequest对象,也就是创建一个异步调用对象.

(2)创建一个新的HTTP请求,并指定该HTTP请求的方法、URL及验证信息.

(3)设置响应HTTP请求状态变化的函数.

(4)发送HTTP请求.

(5)获取异步调用返回的数据.

(6)使用JavaScript和DOM实现局部刷新.

13.同步异步？

同步：同步任务指的是：在主线程上排队执行的任务，只有前一个任务执行完毕，才能执行后一个任务。

异步：不进入主线程、而进入“任务队列”（task queue）的任务，只有等主线程任务执行完毕，“任务队列”才开始通知主线程，请求执行任务，该任务才会进入主线程执行。

14.jsonp的原理，有什么特点？

ajax请求受同源策略影响，不允许进行跨域请求，而script标签src属性中的链接却可以访问跨域的js脚本，利用这个特性，服务端不再返回JSON格式的数据，而是返回一段调用某个函数的js代码，在src中进行了调用，这样就实现了跨域。

**优点：**完美解决了在测试或者开发中获取不同域下的数据，用户传递一个callback参数给服务端，然后服务端返回数据时会将这个callback参数作为函数名来包裹json数据，这样客户端就可以随意定制自己的函数来自动处理返回数据了

**缺点：**jsonp只支持get请求不支持post请求，也就是说如果想传给后台一个json格式的数据，此时问题就来了，浏览器会报一个http状态码415错误，请求格式不正确。

15.跨域的几种解决方案？

1. 使用 JSONP的get 请求进行跨域：因为浏览器对script不存在同源策略，所以script可以跨域请求外部资源，返回的数据是json格式。

缺点：1、只能发送get请求，无法发送post请求2、无法判断请求成功还是失败

1. porxy代理：让代理服务器请求目标地址，因为请求是在服务端进行的，在服务端不存在跨域，从而解决跨域问题
2. 跨域资源共享（CORS）：CORS 需要浏览器和后端同时支持。浏览器会自动进行 CORS 通信，实现 CORS 通信的关键是后端。只要后端实现了 CORS，就实现了跨域。

16.JavaScript里面的防抖和节流？

**一、防抖：**在用户高频触发一个事件的时候，我们不是让函数立即执行，而是看一下在下一个我们规定的时间用户还会不会去执行和触发这个事件。如果同时很多用户进行相同的操作，无异于是给服务端造成极大的压力。

是？等待：调用时间

***使用场景：***

1.resize事件或者scroll事件等，可以通过防抖减少触发次数

2.用户在输入框中频繁输入

**二、节流：**就是在规定的时间内只执行一次，设置一个时间范围，在一定的时间范围内，仅允许执行一次执行一次事件的回调函数，也就是说只要在规定时间范围内，无论事件触发几次，都只能执行一次回调函数。

***使用场景：***

1.鼠标不断点击的情况

2.页面无限加载，每隔一段时间发起请求而不是用户停止滚动发起请求

17.sass/scss与less区别？

**Sass** (Syntactically Awesome Stylesheets)是一种动态样式语言，Sass语法属于缩排语法，比css比多出好些功能(如变量、嵌套、运算,混入(Mixin)、继承、颜色处理，函数等)，更容易阅读。Sass文件中有2个变量，保存的是颜色值，然后我们使用这些变量来设置主体元素的前景颜色和背景颜色。当代码被编译成CSS时，变量被编译为具体值。

Sass与Scss是什么关系?

Sass的缩排语法，对于写惯css前端的web开发者来说很不直观，也不能将css代码加入到Sass里面，因此sass语法进行了改良，Sass 3就变成了Scss(sassy css)。与原来的语法兼容，只是用{}取代了原来的缩进。

**Less**也是一种动态样式语言. 对CSS赋予了动态语言的特性，如变量，继承，运算， 函数.Less 既可以在客户端上运行 (支持IE 6+, Webkit, Firefox)，也可在服务端运行 (借助 Node.js)。

Sass与Scss区别是什么?

1. **编译环境不一样**

Sass的安装需要Ruby环境，是服务端处理的，而less是需要引入less.js来处理less代码输出css到浏览器

1.变量符不一样，less是@，而Scss是$，而且变量的作用域也不一样。

2.输出设置，less没有输出设置，Sass提供4种输出选项：nested,compact,compressed和expanded。

3.sass支持条件语句，可以使用if{}else{},for{}循环等等，而less不支持。

总结：不管是Sass，还是Less，都可以视为一种基于CSS之上的高级语言，其目的是使得CSS开发更灵活和更强大，Sass的功能比Less强大,基本可以说是一种真正的编程语言了，Less则相对清晰明了,易于上手,对编译环境要求比较宽松。考虑到编译Sass要安装Ruby,而Ruby官网在国内访问不了,个人在实际开发中更倾向于选择Less。

18.什么是模块化

类似于java中package，将相关的代码组织到同一个包内，形成一个独立的空间，外部则通过import调用包

19.使用模块封装代码的目的

a.解决命名冲突（防止全局变量的污染）

b.更好地复用代码，提高效率（我的移动端开发体验）

c.提高代码可读性

20.如何使用模块

1、对象:通过对象.属性的方法调用解决命名冲突，但是无法解决变量、函数暴露的问题

2、立即执行函数:通过立即执行函数，来达到隐藏细节的目的

21.规范类型CommonJS 、AMD、CMD分别是什么？

**CommonJS规范（用于服务器端）**

a.一个单独的文件就是一个模块。每一个模块都是一个单独的作用域（在该模块内部定义的变量，无法被其他模块读取，除非定义为global对象的属性）

b.模块导出： 模块由module.exports对象输出。我们需要把模块希望输出的内容放入该对象

c.加载模块： 加载模块使用require方法，该方法读取一个文件并执行，返回文件内部的module.exports对象。

**AMD规范——异步模块定义（用于浏览器端）**

AMD是异步模块定义的意思，由于不受原生JS支持，使用AMD规范进行页面开发时需要引入对应库函数——RequireJS；

RequireJS主要解决两个问题

a.多个js文件可能有依赖关系，被依赖的文件需要早于依赖它的文件加载到浏览器。

b.js加载的时候浏览器会停止页面渲染，加载文件越多，页面失去响应时间越长。

**CMD规范——通用的模块定义**

CMD有个浏览器的实现SeaJS。SeaJS要解决的问题和requireJS一样，只不过在模块定义方式和模块加载（可以说运行、解析）时机上有所不同。Sea.js 推崇一个模块一个文件，遵循统一的写法。

**AMD & CMD区别是什么？**

a.AMD推崇依赖前置，在定义模块的时候就要声明其依赖的模块

b.CMD推崇就近依赖，只有在用到某个模块的时候再去require

最大区别：对依赖模块的执行时机处理不同

AMD用户体验好，因为没有延迟，依赖模块提前执行

CMD性能好，因为只有用户需要的时候才执行的原因

1. **ES6部分**
2. 关于原型的理解？

**原型**：每个函数创建都会有一个自带的属性，这个属性指向一个对象，这个对象就是原型。

每个构造函数所创建出来的实例都可以调用这个原型里面的属性和方法。

原型本身也是一个对象，这个对象就是object的实例，所以原型也可以调用到object的原型。这样就组成了一个链式结构，这个就是**原型链**

1. proxys是什么？

proxy是ES6新增的一个构造函数，可以理解为JS语言的一个代理，用来改变JS默认的一些语言行为。

proxy用于修改某些操作的默认行为

proxy可以理解为对目标对象上面设置的一层拦截

1. map是什么？

map是ES6引入的一种类似Object的新的数据结构，map可以理解为是Object的超集，打破了以传统键值对形式定义对象，对象的key不再局限于字符串，也可以是Object。可以更加全面的描述对象的属性。

1. var let const的区别

var会变量提升 let和const不会

var在全局命名的变量会挂载到window上，let const不会

let和const有块级作用（暂时性死区），var没有

let const不允许重复命名

1. Class

clss语法相对原型、构造函数、继承更接近传统语法，它的写法能够让对象原型的写法更加清晰、面向对象编程的语法更加通俗

1. Promise

promise是异步编程的一种解决方案，将异步操作以同步操作的流程表达出来，避免了层层嵌套的回调函数。

promise 构造函数包含一个参数和一个带有 resolve（解析）和 reject（拒绝）两个参数的回调。在回调中执行一些操作（例如异步），如果一切都正常，则调用 resolve，否则调用 reject。对于已经实例化过的 promise 对象可以调用 promise.then() 方法，传递resolve和reject方法作为回调，**.then()方法**接收两个参数：onResolve和onReject，分别代表当前 promise 对象在成功或失败时。

1. Symblo

ES6引入了一种新的原始数据类型symblo，表示独一无二的值。它是JavaScript语言的第七数据类型，前六种是：undefined、null、布尔值、字符串、数值、对象。symblo值通过symblo函数生成。这就是说，对象的属性名现在可以有两种类型，一种是原来就有的字符串，另外一种就是新增的symblo类型。凡是属性名属于symblo类型，就都是独一无二的，可以保证不会与其他属性产生冲突。

1. Map是什么，有什么作用？

Map是ES6引入的一种类似Object的新的数据结构，Map可以理解为是Object的超集，打破了以传统键值对形式定义对象，对象的key不再局限于字符串，也可以是Object。可以更加全面的描述对象的属性。

1. async和await

async 会将其后的函数（函数表达式或 Lambda）的返回值封装成一个 promise 对象，

而await 会等待这个 promise 完成，并将其 resolve 的结果返回出来。

1. map, filter, reduce 各自有什么作用？

1.map 作用是生成一个新数组，遍历原数组，将每个元素拿出来做一些变换然后放入到新的数组中。另外 map 的回调函数接受三个参数，分别是当前索引元素，索引，原数组

2.filter 的作用也是生成一个新数组，在遍历数组的时候将返回值为 true 的元素放入新数组，我们可以利用这个函数删除一些不需要的元素

3.对于 reduce 来说，它接受两个参数，分别是回调函数和初始值

1. **Vue框架部分**
2. Vue双向绑定的底层原理是什么？

vue.js 是采用数据劫持结合发布者-订阅者模式的方式，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。

具体步骤：

第一步：需要 observe 的数据对象进行递归遍历，包括子属性对象的属性，都加上 setter 和 getter

第二步：compile 解析模板指令

第三步：Watcher主要做的事情是:1、在自身实例化时往属性订阅器(dep)里面添加自己 2、自身必须有一个 update()方法 3、待属性变动 dep.notice()通知时，能调用自身的update()方法，并触发 Compile 中绑定的回调，则功成身退。

第四步：MVVM 作为数据绑定的入口，整合 Observer、Compile 和 Watcher 三者，通过 Observer 来监听自己的 model 数据变化，通过 Compile 来解析编译模板指令，最终利用 Watcher 搭起 Observer 和 Compile 之间的通信桥梁，达到数据变化。

1. 请详细说下你对 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：

1、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。

2、在 created 阶段，vue 实例的数据对象 data 有了，$el 还没有。

3、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。

4、在 mounted 阶段，vue 实例挂载完成，data.message 成功渲染。

5、更新前/后：当 data 变化时，会触发 beforeUpdate 和 updated 方法。

6、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。

***vue-router和vuex都是在beforeCreate阶段挂载的***

1. 请说下关于token的身份验证的过程？

用户通过用户名和密码发送请求。

程序验证。

程序返回一个签名的token 给客户端。

客户端储存token,并且每次用于每次发送请求。

服务端验证token并返回数据。

1. 关于你理解的token？

Token 是在服务端产生的。如果前端使用用户名/密码向服务端请求认证，服务端认证成功，那么在服务端会返回 Token 给前端。前端可以在每次请求的时候带上 Token 证明自己的合法地位。如果这个 Token 在服务端持久化（比如存入数据库），那它就是一个永久的身份令牌

可以解决哪些问题呢？

a.Token 完全由应用管理，所以它可以避开同源策略

b.Token 可以避免 CSRF 攻击(http://dwz.cn/7joLzx)

C.Token 可以是无状态的，可以在多个服务间共享

1. MongoDB和Mysol？

MySQL与MongoDB都是开源的常用数据库，但是MySQL是传统的关系型数据库，MongoDB则是非关系型数据库，也叫文档型数据库；MongoDB储存的方式是以类JSON的文档的格式存储；十分适合文档格式的存储与查；Mysols是不同引擎有不同的存储方式。

1. ajax和axios、fetch的区别

**Ajax**是基于原生的XHR开发，XHR本身的架构不清晰，已经有了fetch的替代方案，JQuery整个项目太大，单纯使用ajax却要引入整个JQuery非常的不合理（采取个性化打包的方案又不能享受CDN服务）

**axios**从 node.js 创建 http 请求，支持 Promise AP，客户端支持防止CSRF，提供了一些并发请求的接口（重要，方便了很多的操作）

**fetch**更加底层，提供的API丰富（request, response），脱离了XHR，是ES规范里新的实现方式，fetchtch只对网络请求报错，对400，500都当做成功的请求，需要封装去处理，fetch默认不会带cookie，需要添加配置项，fetch不支持abort，不支持超时控制，使用setTimeout及Promise.reject的实现的超时控制并不能阻止请求过程继续在后台运行，造成了量的浪费，fetch没有办法原生监测请求的进度，而XHR可以。

1. 为什么要用axios?

axios 是一个基于Promise 用于浏览器和 nodejs 的 HTTP 客户端，它本身具有以下特征：

从浏览器中创建 XMLHttpRequest

从 node.js 发出 http 请求

支持 Promise API

拦截请求和响应

转换请求和响应数据

取消请求

自动转换JSON数据

客户端支持防止CSRF/XSRF

1. vue全家桶

Vue-cli3+vue-router+vuex+axios+element-ui+mongoDB

Vu-cli3是官方提供的快速搭建项目的工具，支持热更新，相当于启动一个请求器，vue-router是vue生态里面的路由工具，采用路由能轻松实现单页面跳转程序，vuex是vue提供的状态管理工具，简单解释就是vue各个组件直接的变量是不能直接共享的，组件直接的参数传递才多层的时候变得异常复杂，所以就诞生了vuex的状态管理工具，保证了状态的统一和可追踪，axios是一个基于promise用于浏览器和node.js的http客户端，可以通过向axios传递相关的配置创建请求。Element-ui是一个ui库，它不依赖于vue。但是却是当前和vue配合做项目开发的一个比较好的ui框架。mongoDB是一个分布式文档存储库。

1. vue.cli项目中src目录每个文件夹和文件的用法？

assets文件夹是放静态资源

components是放组件

router是定义路由相关的配置

view试图

app.vue是一个应用主组件

main.js的入口文件

1. 怎么定义vue-router的动态路由？怎么获取传过来的动态参数？

在router目录下的index.js文件中，对path属性加上/:id

动态路由的定义：即某种模式匹配到的所有路由，全部映射到同一组件。使用动态路由参数来实现。例如，我们有一个user组件，对于所有ID各不相同的用户，都要使用这个组件来渲染。那么，我们可以在vue-router的路由路径中使用“动态路径参数”来达到这个效果。

1. 路由的实现原理

hash模式：在浏览器中符号“#”，用window.location.hash读取

特点：hash虽然在URL中，但不被包括在HTTP请求中；用来指导浏览器动作，对服务端安全无用，hash不会重加载页面。仅 hash 符号之前的内容会被包含在请求中，

如：http://www.xxx.com，因此对于后端来说，即使没有做到对路由的全覆盖，也不会返回404 错误。

history模式：history采用HTML5的新特性；且提供了两个新方法：pushState（），replaceState（）可以对浏览器历史记录栈进行修改，以及popState事件的监听到状态变更。

特点：前端的 URL 必须和实际向后端发起请求的 URL 一致，

如：http://www.xxx.com/items/id。后端如果缺少对 /items/id 的路由处理，将返回 404 错误。你要在服务端增加一个覆盖所有情况的候选资源：如果 URL 匹配不到任何静态资源，则应该返回同一个 index.html 页面，这个页面就是你 app 依赖的页面。

1. vue-router有哪几种导航钩子？

1.全局导航钩子：rouer.berforeEach(to,from,next),作用：跳转前进行判断拦截

2.组件内的钩子

3.单独路由独享组件

1. 请说下封装组件的过程？

使用vue-extend方法创建一个组件，然后使用vue.component方法注册组件。子组件需要数据，可以在pros中接受定义。而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。

1. vue组件的scoped属性的作用

当style标签有scoped属性时，它的css只会作用于当前组件中的元素，你可以在一个组件中同时使用有scoped和非scoped样式。但是有时候在vue-cli中引入UI库后，修改UI库一些样式，可能不会生效

**状态管理**

1. redux的理解

是什么：是一个应用数据流框架，集中式管理

怎么用：view 调用 store 的 dispatch 接收 action 传入 store，reducer 进行 state 操作，view 通过 store 提供的 getState 获取最新的数据

缺点： state必须由父组件传过来，子组件相关数据更新，父组件会强渲

1. flux 思想

单向数据流

用户访问 View

View 发出用户的 Action

Dispatcher 收到 Action，要求 Store 进行相应的更新

Store 更新后，发出一个"change"事件

View 收到"change"事件后，更新页面

1. vuex的理解

原理：vuex 仅仅是作为 vue 的一个插件而存在，不像 Redux,MobX 等库可以应用于所有框架，vuex 只能使用在 vue 上，很大的程度是因为其高度依赖于 vue 的 computed 依赖检测系统以及其插件系统，

vuex 整体思想诞生于 flux,可其的实现方式完完全全的使用了 vue 自身的响应式设计，依赖监听、依赖收集都属于 vue 对对象 Property set get 方法的代理劫持。最后一句话结束 vuex 工作原理，vuex 中的 store 本质就是没有 template 的隐藏着的 vue 组件；

使用：在 main.js 引入 store，注入。新建了一个目录 store，….. export

场景：单页应用中，组件之间的状态。音乐播放、登录状态、加入购物车

1. vuex有哪几种状态，会在哪些功能场景使用它？

1.state： Vuex store实例的根实例状态对象，用于定义共享的状态变量。

2.action-动作：向store发出调用通知，执行本地或者远端的某一个操作（可以理解为store的methods）

3.Mutations-修改器：它只用于修改state中定义的状态变量。

4.getter-读取器:外部程序通过它获取变量的具体值，或者在取值前做一些计算（可以认为是store的计算属性）

应用场景：

1.数据需要在多个组件里面来回调用的

2.全局都要使用的方法，并且方法是需要保持不被浏览器清理掉的

3.组件需要保存的数据

1. vuex中使用异步修改，为什么要异步修改？

怎么使用？：在vuex里面创建一个actions方法

为什么需要actions管理异步？

在vue早期版本中mutation中必须同步执行，虽然说现在的版本中在某些情况下在里面写异步代码不仅不会报错，有时候还会正确的执行，但是还是不推荐。

在实际开发中会遇到这样的问题，第二个异步操作的条件依赖第一个异步操作的结果。当遇到这种情况的时候我们肯定希望被依赖的数据在拿到结果之后再处理。所以这时候我们就需要使用actions来处理。

1. vue中组件直接的通信是如何实现的？

组件关系可以分为父子组件通信、兄弟组件通信。

1.父组件向子组件：

通过props属性来实现

2.子组件向父组件：

子组件用$emit()来触发事件，父组件用$on()来监听子组件的事件。

父组件可以直接在子组件的自定义标签上使用v-on来监听子组件触发的自定义事件。

3.兄弟之间的通信：

通过实例一个vue实例bus作为媒介，要相互通信的兄弟组件之中都引入bus，之后通过分别调用bus事件触发和监听来实现组件之间的通信和参数传递。

1. 为什么要使用vue构造器？

常规的渲染组件的方式就是，在规定的地方渲染组件，如果组件的模板是从后台发过来的，需要动态的渲染组件。需要实现一些提示框之类的页面，这种页面通常他的父级是在body里面而不是在#app里面的

1. 什么是高阶组件？

高阶组件是一个以组件为参数并返回一个新组件的函数。HOC 运行你重用代码、逻辑和引导抽象。最常见的可能是 Redux 的 connect 函数。除了简单分享工具库和简单的组合，HOC 最好的方式是共享 React 组件之间的行为。如果你发现你在不同的地方写了大量代码来做同一件事时，就应该考虑将代码重构为可重用的 HOC。

1. render函数

我们都知道vue的核心是虚拟DOM。我们通常将我们的页面的构造逻辑都写在template里面的，然后vue将我们的代码转换成虚拟DOM。相比起DOM，虚拟DOM是通过js来处理的，所以消的性能相对于来说很小。vue推荐在绝大多数情况下使用template来创建你的HTML。然而在一些场景中，你真的需要JavaScript的完全编程的能力，这时你可以用render函数，它比template更接近编程器。render函数的作用就是讲template里面的结构转换成Vnode

1. render函数渲染和template模板的区别是什么？

Template更适合逻辑简单，render适合复杂逻辑。Template属于声明是渲染，render属于自定render函数，声明式渲染，使用者理解起来相对容易，但是灵活性不足，自定义render函数灵活性高，但是对使用者要求较高。

1. template是如何编译成真正的HTML并做到双向绑定等特殊功能的？

1.首先第一步实例化一个vue项目

2.模板编译是在vue生命周期的mount阶段进行的

3.在mount阶段的时候执行了compile方法将template里面的内容转化成真正的HTML代码

4parse阶段是将html转化成ast,用来表示代码的数据结构。在Vue中我把它理解为嵌套的、携带标签名、属性和父子关系的js对象，以树来表现DOM结构。

1. 虚拟DOM

虚拟 dom 相当于在 js 和真实 dom 中间加了一个缓存，利用 dom diff 算法避免了没有必要的 dom 操作，从而提高性能。

用 JavaScript 对象结构表示 DOM 树的结构；然后用这个树构建一个真正的 DOM 树，插到文档当中当状态变更的时候，重新构造一棵新的对象树。然后用新的树和旧的树进行比较，记录两棵树差异把 2 所记录的差异应用到步骤 1 所构建的真正的 DOM 树上，视图就更新了。

1. diff算法（patch）

patch主要是对虚拟的DOM所产生的数据进行差异比较的，是vue中的一个核心功能，其中差异比较的核心就是diff算法。

我们先对新数据和老数据精选划分。划分成四个部分对每一个部分进行比价，只对差异部分进行修改。这样做就会减少性能消耗。

1. 组件、插件、控件分别是什么？

1）组件：首先最广的应该是组件。凡是在软件开发中用到了软件的复用，被复用的部分都可以称为组件。

2）插件：插件是组件中中的一种，凡是在应用程序中已经预留接口的组件就是插件。可以拿IE插件作为例子，IE中之所以可以嵌入很多应用程序，那是因为IE允许他们插入，说的明白一点，那就是在IE的源程序中已经为这些应用程序预留了接口，只要把通知浏览器已经加载了什么插件，浏览器就会调用预留的接口调用这些所谓的插件。

3）控件：控件也是组件中的一种。控件就是具有用户界面的组件（可视化组件），比如：按钮、列表框、编辑框或者某个对话框中的静态文本等。

1. MVP MVC M的区别？

三者的目的都是分离关注，使得UI更容易变换，使得UI更容易测试。

MVC是View接受用户的交互请求，view将请求转交给controller，controller操作model进行数据更新，数据更新后，model通知view数据变化，view显示更新之后的数据。

MVP是View接受用户的交互请求，view将请求转交给presenter，presenter操作model进行数据更新，数据更新后,model通知presenter数据变化,presenter更新view的数据。和MVC不同的是，Presenter会反作用于View，不像Controller只会被动的接受View的指挥。

MVVM就是 vm 框架视图、m 模型用来定义驱动的数据、v 经过数据改变后的 html、vm 就是用来实现双向绑定。

1. axios是什么？

axios是一个基于promise的http库，可以用在浏览器和nodeJS中，因为axios中使用了很多方法，所以axios需要转化一下才可以使用，安装babel-polyfill和es6-promise插件就可以解决

1. axios怎么解决跨域问题？

axios是不支持jsonp的所以jsonp方法被否决了；

反向代理 就是搭建一个服务器将我的请求通过我的服务发送给对方服务器。因为服务器之间发送请求不会出现跨域的问题，针对后台不支持options，那么让后台使用cors插件，那么自己写一个后台，进行反向代理。

1. axios和ajax的区别？

允许浏览器与服务器通信而无须刷新当前页面技术都被叫做ajax。

依赖的传输对象：XMLHttpRequest

Axios本质上也是对原生XHR的封装，只不过它是Promise的实现版本，符合最新的ES规定，它有以下几条特性：

1. 从node.js创建http请求；
2. 支持promise API
3. 客户端支持防止CSRF
4. 提供了一些并发请求的接口
5. Axios既提供了并发的封装，也没有fecth的各种问题，而且体积也比较小。
6. **webpack部分**
7. webpack中的entry，output是做什么的？

entry：用来写入口文件，它将是整个依赖关系的根。当我们需要多个入口文件的时候，可以把entry写成一个对象。

output：即使入口文件有多个，但是只有一个输出配置，如果你定义的入口文件有多个，那么我们需要使用占位符来确保输出的文件的唯一性。

1. gulp和webpack的区别

gulp可以进行js，html，css，img的压缩打包，是自动化构建工具，可以将多个js文件或是css压缩成一个文件，并且可以压缩为一行，以此来减少文件体积，加快请求速度和减少请求次数；并且gulp有task定义处理事务，从而构建整体流程，它是基于流的自动化构建工具。

Webpack是前端构建工具，实现了模块化开发和文件处理。他的思想就是“万物皆为模块”，它能够将各个模块进行按需加载，不会导致加载了无用或冗余的代码。所以他还有个名字叫前端模块化打包工具。

1. get和post区别

GET和POST本质上没有区别

GET/POST都是TCP链接。GET和POST能做的事情是一样一样的。你要给GET加上request body，给POST带上url参数，技术上是完全行的通的。

对于GET方式的请求，浏览器会把http header和data一并发送出去，服务器响应200（返回数据）；

而对于POST，浏览器先发送header，服务器响应100 continue，浏览器再发送data，服务器响应200 ok（返回数据）

1. webpack中的loader和plugins的作用是什么?

loader:1.实现对不同格式的文件处理，比如讲scss转换成css，或者typescript转为js

2.转换这些文件，从而使其能够被添加到依赖图中，

我们平时常用的loader：

bable-loader:让一代的js文件转换成现代浏览器能够支持的js文件。

css-loader,style-loader：用来解析css文件

flie-loader:生成的文件名就是文件内容的MD5哈希值会保留所引用资源的原始扩展名

url-loader:功能类似file-loader,但是文件大小低于指定的限制时，会返回一个DataUrl

loader负责的是处理源文件的如css、jsx，一次处理一个文件。而plugins并不是直接操作单个文件，它直接对整个构建过程起作用。

1. **Node部分**
2. 为什么会使用node？

node有以下几个特点：简单强大，轻量可扩展，简单体现在node使用的是JavaScript,json来进行编码；强大体现在非阻塞IO，可以适应分块传输数据，较慢的网络环境，尤其擅长高并发访问；轻量体现在node本身即是代码，又是服务器，前后端使用统一语言；可扩展体现在可以轻松应对实例，多服务器架构，同时有海量的第三方应用组件。

1. node有那些全局对象？

Process, console，buffer和exports

1. node.js适用于哪些场景？

a.实时应用：如在线聊天，实时通知推送等等

b.分布式应用：通过高效的并行IO使用已有的数据

c.工具类应用：海量的工具，小到前端压缩部署，大到桌面图形界面应用程序

d.利用稳定接口提升web渲染能力

1. **怎么与后端沟通配合**

a.前后端都要充分了解项目的需求

b.后端只提供能力（如数据库能力、消息能力、应用协同能力等等）

c.前端组织业务（如组织Auth、组织数据POST/GET/DELETE）

d.接口设计（出接口文档），前后端沟通设计接口，前端需要后台返回什么样的数据（格式），后台需要前端传递什么参数（哪些参数是必须的，哪些参数是可选的，采用get还是post，哪些数据需要前端先进行校验，哪些需要双方都校验）。共同制定出整个程序所有的接口说明，形成文档。前后台按照约定好的接口进行开发

1. **Git**

1.Git和SVN的区别

Git是分布式版本控制系统，SVN是集中式版本控制系统

2.Git工作流程

1、在工作目录中修改某些文件

2、对修改后的文件进行快照，然后保存到暂存区域

3、提交更新，将保存在暂存区域的文件快照永久转储到Git目录中

3.常用命令

git show # 显示某次提交的内容 git show $id

git add <file> # 将工作文件修改提交到本地暂存区

git rm <file> # 从版本库中删除文件

git reset <file> # 从暂存区恢复到工作文件

git reset HEAD^ # 恢复最近一次提交过的状态，即放弃上次提交后的所有本次修改

git diff <file> # 比较当前文件和暂存区文件差异 git diff

git log -p <file> # 查看每次详细修改内容的diff

git branch -r # 查看远程分支

git merge <branch> # 将branch分支合并到当前分支

git stash # 暂存

git stash pop #恢复最近一次的暂存

git pull # 抓取远程仓库所有分支更新并合并到本地

git push origin master # 将本地主分支推到远程主分支

4.Git恢复先前的提交

git reset –hard HEAD~1

5.提交时发生冲突，你能解释冲突是如何产生的吗？你是如何解决的？

如公共类的公共方法，我和别人同时修改同一个文件，他提交后我再提交就会报冲突的错误。

发生冲突，一般都是对比本地文件和远程分支的文件，然后把远程分支上文件的内容手工修改到本地文件，然后再提交冲突的文件使其保证与远程分支的文件一致，这样才会消除冲突，然后再提交自己修改的部分。特别要注意下，修改本地冲突文件使其与远程仓库的文件保持一致后，需要提交后才能消除冲突，否则无法继续提交。必要时可与同事交流，消除冲突。

发生冲突，也可以使用命令。

通过git stash命令，把工作区的修改提交到栈区，目的是保存工作区的修改；

通过git pull命令，拉取远程分支上的代码并合并到本地分支，目的是消除冲突；

通过git stash pop命令，把保存在栈区的修改部分合并到最新的工作空间中；

6.如何查看分支提交的历史记录？

命令git log –number：表示查看当前分支前number个详细的提交历史记录；