目录

[1. 地理定位 4](#_Toc529127014)

[2. React生命周期shouldComponentUpdate 4](#_Toc529127015)

[3. Vue中，Object.defineProperty方式挂载全局方法优点 4](#_Toc529127016)

[4. WebAPP、native、混合开发、小程序、H5页面 4](#_Toc529127017)

[5. H5怎么跟native进行通讯 4](#_Toc529127018)

[6. 发起一次url的过程 4](#_Toc529127019)

[7. JWT（json-web-token）技术来做前后端鉴权。 4](#_Toc529127020)

[8. JWT技术：( json-web-token)登录验证 5](#_Toc529127021)

[10. vue、react的路由原理? 5](#_Toc529127022)

[11. 单页面应用点击前进后退按钮不刷新页面渲染怎么实现？纯js实现不刷新路由跳转效果 6](#_Toc529127023)

[12. vue、react的生命周期? 6](#_Toc529127024)

[13. redux、vuex的实现流程? 6](#_Toc529127025)

[14. diff算法的作用与实现 6](#_Toc529127026)

[15. Vue/react循环遍历时的key相关 6](#_Toc529127027)

[16. 高阶组件和高阶函数的区别？ 6](#_Toc529127028)

[17. 跳转路由的方法 react、vue 6](#_Toc529127029)

[18. function和箭头函数的区别 7](#_Toc529127030)

[19. react的一些优化 7](#_Toc529127031)

[20. React中连续多次调用setState的后果？ 7](#_Toc529127032)

[21. 如何在固定时间刷新页面？ 7](#_Toc529127033)

[22. Less和sass的区别 7](#_Toc529127034)

[23. Git，svn的特性 7](#_Toc529127035)

[24. 语法糖 8](#_Toc529127036)

[25. 数据持久化 8](#_Toc529127037)

[26. H5新增 8](#_Toc529127038)

[27. 项目开发中使用的规范 8](#_Toc529127039)

[28. 项目开发遇到的问题 8](#_Toc529127040)

[29. 项目上线遇到的问题？ 8](#_Toc529127041)

[30. ES6比ES5有那些改进？ES6新增了那些js的语法糖 9](#_Toc529127042)

[31. Axios和fetch对比？为什么项目使用axios和fetch 9](#_Toc529127043)

[32. PWA（progressive web app） 9](#_Toc529127044)

[33. 离线存储web service 9](#_Toc529127045)

[34. MVVM框架和jQuery这种大型js框架的区别 9](#_Toc529127046)

[35. 预加载和懒加载 9](#_Toc529127047)

[36. Get和set 9](#_Toc529127048)

[37. 项目优化 9](#_Toc529127049)

[38. Vue中使用 bootcdn 对打包的vendor进行优化CND加速 10](#_Toc529127050)

[39. get和post请求特点 10](#_Toc529127051)

[40. webpack的使用 10](#_Toc529127052)

[41. 一般电商首页逻辑 10](#_Toc529127053)

[42. 编辑个人资料逻辑 10](#_Toc529127054)

[43. seo的实现? 10](#_Toc529127055)

[44. TypeScript的具体阐述 10](#_Toc529127056)

[45. var 和 let 和 cont的区别 10](#_Toc529127057)

[46. 单向数据流的解释 11](#_Toc529127058)

[47. 实现深拷贝和浅拷贝的方法 11](#_Toc529127059)

[48. v-model的原理 11](#_Toc529127060)

[49. object的方法 11](#_Toc529127061)

[50. 小程序开发时只能用https的url 11](#_Toc529127062)

[51. 闭包？ 11](#_Toc529127063)

[52. Gitflow工作流？ 11](#_Toc529127064)

[53. 什么是generator函数？ 12](#_Toc529127065)

[54. 数据类型 12](#_Toc529127066)

[55. 冰桶游戏：canvas+js实现游戏 12](#_Toc529127067)

[56. 面试话术 12](#_Toc529127068)

[57. Vue中渲染方式render和template 12](#_Toc529127069)

[58. 判断对象是否为数组 12](#_Toc529127070)

[59. 跨域 12](#_Toc529127071)

[60. 移动端适配 13](#_Toc529127072)

[61. 前后端交互 13](#_Toc529127073)

[62. Git分支冲突 13](#_Toc529127074)

[63. Android和ios兼容 13](#_Toc529127075)

[64. 原型链 13](#_Toc529127076)

[65. 继承 14](#_Toc529127077)

1. 地理定位
   1. 先百度地图申请成为开发者，获取key。然后在Index.html页面引入百度/腾讯地图的一个js文件。
   2. 在需要定位的页面里面new qq.map.geoLocation(key, myID) ，可以拿到当前所在位置的经纬度
   3. 把经纬度传递给后台，后台向地图发起请求，根据经纬度返回附近建筑信息，最后传递给前端进行渲染
   4. Iframe的src属性可以输出整个百度/腾讯地图
2. React生命周期shouldComponentUpdate
   1. 它可以优化页面渲染，提高性能优化。State状态发生改变的时候触发，可以判断如果新的state跟上一次的状态是相同的我们就可以return一个false来阻止这一次的页面渲染。
   2. 举例：input的初始值是整数1，我们输入的也是1，但是输入之后获取到的是字符串1,这两次的值是一样的，只是类型不一样，我们完全可以不让页面进行重新渲染。
   3. 如果涉及到引用类型的更新，主要就是怎么判断两次的数据值是相同的。我们先判断新值和旧值的length是否相同。然后我们可以封装一个函数，里面可以用递归循环遍历两次的数据并判断是否一致。
3. Vue中，Object.defineProperty方式挂载全局方法优点
   1. defineProperty可以自定义，设置了不可写，防止了误操作，更安全。
   2. 如果用vue.prototype方式挂载，会造成全局污染，一个地方改了值，全部调用都被改动了，不安全。
4. WebAPP、native、混合开发、小程序、H5页面
   1. webApp：所有用H5开发的可运行于浏览器端的应用都称为webAPP
   2. H5页面：是HTML5的简称，是一种高级网页技术。比H4有更多的交互和功能。最大的优点之一是在移动设备上支持多媒体。平时看到的幻灯片、小游戏等都是H5网页。跟我们上网看到的网页本质上没有任何区别。H5一般用来品牌传播、活动推广、产品展示。
   3. 混合开发：ybrid APP。H5+native开发。重点是H5与native的交互。Android中通过webview加载html页面。例如支付宝-口碑里面，打开其中一个网页下拉，发现是‘网页有XXX提供’  
      混合开发还有一种：套壳APP。就是把H5网页打包成APP，每个界面都是网页。每次访问都需要从云端调取内容，导致反应慢。  
      优点：便于调试、开发周期短、费用低、内容方便更新。  
      缺点：功能/界面无法自定，加载慢，网络要求高，安全性低
   4. NativeAPP开发：  
      优点：app数据保存在本地，app能及时调取，响应速度快，运行流畅。兼容性好，安全性高，功能可以自定。  
      缺点：开发周期长，费用高，发版慢等。Native访问手机原生功能强。但是发版慢、开发周期长
   5. 小程序：运行与微信，其优点：更新迭代快，功能权限高，综合H5和native的优点，体积小，宣传快。
5. H5怎么跟native进行通讯
   1. native内嵌H5需要webview容器（iframe就是）
   2. 通讯需要一个JSBridge桥梁，JSBridge作用就是连接H5和native（native细分为Android和IOS）
   3. Android下可以捕捉window对象，IOS下可以捕捉资源请求（ajax请求或src属性）
   4. Android下在H5页面使用window[“call\_share\_fun”]({需要传递的参数})，其中call\_share\_fun是跟native约定好的字段，在H5页面调用执行。注意：window下没有这个函数，所以，需要放trycache里面防止报错
   5. IOS下在H5里面使用ajax发起一次请求，url就是协议好的url。或者src属性，src=“callFun({传递的参数})”
   6. 通讯有一个指定好的协议。公司名:// + 公司内部自定义函数名 ，拼接起来就是完整的url。举例：mjlx://call\_share\_fun
6. 发起一次url的过程
   1. 先走一遍本地缓存资源，找本地的DNS缓存
   2. 如果本地没有，去读取电脑的PC操作系统下的hosts（是整个电脑的访问路径配置）配置文件，如果文件里面有对应的url，会访问对应url的对应网址。
   3. 如果hosts没有，才去internet访问真实url
   4. 浏览器发起一个DNS系统调用，获取到域名对应的IP地址之后，发起HTTP三次握手
   5. TCP/IP链接建立之后，浏览器向服务器发送http请求。
   6. 服务端接收到请求，根据路径参数，把后端处理后的结果返回给浏览器，可以是一个完整的HTML页面
   7. 浏览器拿到HTML页面代码，开始解析和渲染，html中的js/css/图片等静态资源也是http请求，都需要走上面相同步骤
   8. 浏览器根据拿到的资源最终完整呈现页面
7. JWT（json-web-token）技术来做前后端鉴权。
   1. 用户使用用户名密码来请求服务器 🡺服务器进行验证用户的信息 🡺 服务器通过验证发送给用户一个token
   2. 客户端存储token，并在每次请求时附送上这个token值 🡺 服务端验证token值，并返回数据
   3. JWT的构成
      1. 第一部分我们称它为头部（header),第二部分我们称其为载荷（payload, 类似于飞机上承载的物品)，第三部分是签证（signature)
8. JWT技术：( json-web-token)登录验证
   1. 每次请求都需要携带token字段，可以进行路由等权限的验证，做前后端验证健全。如果没有携带token去后台，后台会认为没有权限，直接抛出错误
   2. 发起一次登录请求，如果登录验证通过，后台返回数据会包括token字段，本地存储token字段，然后拿着token字段去后台请求对应的用户权限
   3. 客户端收到服务器返回的JWT字段，可以存储在cookie里面，也可以存储在localStorage里面，但是cookie无法跨域，所以，最好的做法是放到http的头信息Authorization字段里面。另一种做法：跨域时，JWT放到post请求的数据体里面
9. CMS场景用户权限及localStorage不更新问题，接口权限、登录权限  
   在用户第一次使用普通管理员账号登录成功之后返回的数据里面有token字段，本地存储token字段

\* 如果他返回登录页，使用另外一个超级管理员的账号再次登录，还是会返回token字段，重新设置localStorage存储的token

\* 可以看到，浏览器里面的token字段已经更新，但是因为是单页面应用，跳转路由进入首页之后，fetch.js里面封装的http方法获取到的localStorage还是上次的token字段

\*\* 原因：单页面应用只有reload，js才会刷新， 页面不刷新js不会重新获取localStorage的值，这个字段就不会更新，所以用户看到的仍然是上次登录的权限界面。为了解决这个问题，有以下2中解决办法：

* 1. 1. location.href = '/home' 使用location.href跳转页面，可以刷新页面
  2. 2. window全局监听localStorage的setItem保存token字段时的事件  
     重构setItem，在setItem里面修改localStorage值时，派发一个事件，window全局监听，值发生改变就立即触发监听函数，改变对象的值  
     tools.js里面封装dispatchEventStroage函数并抛出

function dispatchEventStroage () {

const signSetItem = localStorage.setItem

// 重构setItem

localStorage.setItem = function (key, newValue) {

// 实例化自定义事件

let setEvent = new Event('setItemEvent')

// 设置key值和value

setEvent.key = key

setEvent.newValue = newValue

// 当setItem加小括号调用执行时，dispatchEvent触发window全局的setItemEvent事件执行

window.dispatchEvent(setEvent)

signSetItem.apply(this, arguments)

}

}

// main.js引入toos.js文件，调用执行

tool.dispatchEventStroage()

// fetch.js里面封装的http请求数据方法

// 为了安全和权限验证，使用JWT技术，每次请求都需要携带token字段

const reqHeaders = {

Accept: 'text/json',

token: localStorage.getItem('token') || ''

}

window.addEventListener('setItemEvent', function(e) {

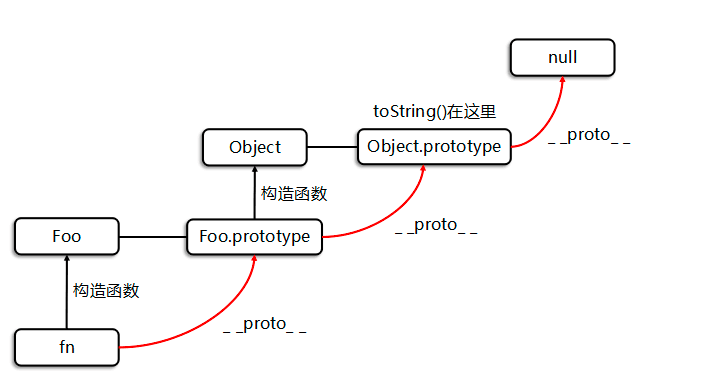
reqHeaders.token = e.newValue

})

1. vue、react的路由原理?
   1. 在地址栏中加入#以欺骗浏览器：地址的改变是由于正在进行页内导航。
   2. 使用h5的window.history功能，使用URL的hash来模拟一个完整的URL。这种的优势是，在地址栏中不存在#，可以避免后台需要验证地址栏的时候，出现不识别的错误。
   3. Vue和react约定使用push方式来改变路由。Push做了2件事，1是改变了地址栏，2是更新了页面内容
      1. 可以用history新增的pushState/replaceState方法，不刷新页面更改路由
      2. Vue提供了<component is=”listComponent”> 组件，is属性是啥页面就会渲染对应的组件。
   4. 实例化路由的时候，是把所有的组件放到一个数组里面，当我们切换路由的时候，就是取数组对应组件赋值给is
2. 单页面应用点击前进后退按钮不刷新页面渲染怎么实现？纯js实现不刷新路由跳转效果
   1. H5扩展了history的API。有存储当前历史记录点pushState，替换当前历史记录的replaceState，监听历史记录的popstate
   2. popstate可以捕捉到前进后退按钮的点击，就可以获取到需要切换的url，更改url
   3. 然后更换页面内容。方法：读取路由对应的文件内容，通过innerHTML或者appendChild添加到页面box
3. vue、react的生命周期?
   1. Vue生命周期：分为实例期、存在期、销毁期。beforeCreate，created；beforeMount，mounted；beforeUpdate，updated；beforeDestroy，destoryed；activited，deactivited；
   2. Vue中，走到created时，data/methods/computed/watch已经初始化完成。beforeMount的时候看页面有没有template/$mount/el，然后开始编译虚拟dom。销毁组件使用$destory()方法。
   3. React生命周期：componentWillMount；componentDidMount；render；componentWillReceiveProps，shouldComponentUpdate；componentWillUpdate；componentDidUpdate；componentWillUnmount；getDefaultProps；getInitialState；
   4. React销毁组件可以用 flag ? <myCom/> : ‘’，组件就会自动销毁
4. redux、vuex的实现流程?
   1. Vuex：是一个专为vue.js开发的状态管理模式。采用集中式存储 管理应用的所有组件的状态，以相应规则保证状态可预测。  
      具体实现：VueComponents 用dispatch触发一个actions，actions里面commit一个mutations，只能在mutations里面改变state状态，state的改变更新到视图。同时，vuex提供了mapActions/mapState/mapGetters方法快捷在页面。
   2. Redux：component 🡺dispatch(action) 🡺 reducer 🡺 subscribe 🡺 getState 🡺 component
   3. React-redux：component 🡺 actionCreator(data) 🡺reducer 🡺component。同时提供了Provider和Connect。
      1. provider是一个封装好的组件，接受store作为props，这样react中任何组件都可以获取store。
      2. connect(mapStateToProps, mapDispatchToProps, mergeProps, options)(component)
         1. mapStateToProps有state和自定义的props两个参数，return一个对象，这个对象将作为props的一部分传入组件。自定义props变化也会触发它
         2. mapDispatchToProps（dispatch，【ownProps】）
         3. mergeProps（stateProps，dispatchProps，ownProps）将mapStateToProps() 与 mapDispatchToProps()返回的对象和组件自身的props合并成新的props并传入组件。默认返回 Object.assign({}, ownProps, stateProps, dispatchProps) 的结果。
         4. options={pure=true}表示connect容器组件将在shouldComponentUpdate中对store中的state和ownProps进行浅对比，优化性能。Pure为false时不对比
5. diff算法的作用与实现
   1. diff算法执行时有三个维度：Tree-diff/Component-diff/Element-diff，按顺序依次执行。
   2. Tree-diff是对树的每一层进行遍历，如果某个组件不存在就直接销毁。
   3. Component和element-diff是同级对比。Vue/react的虚拟dom利用了diff算法
6. Vue/react循环遍历时的key相关
   1. V-for遍历时key值是唯一的，如果不加key就会报错。这里的key是vue识别结点的一个通用机制，
7. 高阶组件和高阶函数的区别？
   1. 高阶函数：以函数作为参数的函数，return一个函数。如：map，reduce，filter，sort都是高阶函数，forEach是for循环的高阶函数
   2. 高阶组件：以组件作为参数的组件，return一个组件。它是一个纯函数，接受一个组件参数。高阶组件最好不要有自己的state，传值用props方法。不去改变原始组件，通过组合的方式。HOC（table）。HOC是父组件，table是子组件，子父组件的传参
8. 跳转路由的方法 react、vue
   1. React跳转路由的方法：
      1. 方法一：引入Redirect标签，用redirect标签的to属性跳转 import { Redirect } from 'react-router'; <Redirect to="/chat" />
      2. 方法二：引入prop-types包 import PropTypes from 'prop-types';
      3. 方法三：使用withRouter解决。可百度查看
   2. Vue跳转路由的方法
      1. This.$router.push({name:’mater’, params: {id: 1}})刷新页面参数会丢失
      2. This.$router.push({path:’/mater’, query: {id: 1}})
      3. Replace方式替换，不会留下history记录  
         this.$router.push({path: ‘home’, replace: true})  
         <router-link :to="..." replace></router-link>
      4. 跳转this.$router.go(n)  n可为正可为负
9. function和箭头函数的区别
   1. 箭头函数是匿名函数，不能作为构造函数，不能使用new运算符。
   2. 箭头函数不绑定arguments，用rest参数即…解决
   3. 箭头函数不绑定this，会捕获其所在的上下文（即定义时）的this值作为自己的this，this指向调用它的那个对象
   4. 箭头函数通过call和apply调用一个函数时，只传入一个参数对this指向没有影响
   5. 箭头函数没有原型属性prototype
   6. 箭头函数不能当做Generator函数，不能使用yield关键字
   7. 箭头函数的this永远指向其上下文的this，任何方法都改变不了其指向。如call，bind和apply
10. react的一些优化
    1. react组件优化：
       1. 属性传递优化
          1. 事件优化。事件的声明方式有三种,其中3>2>1
             1. <button onClick={this.clickFun.bind(this)}> 声明时bind改变this
             2. <button onClick={() => this.clickFn()}> 使用函数式声明事件
             3. <button onClick={this.clickFun}>.然后在constructor中写this.clickFun=this.clickFun.bind(this)
          2. 组件嵌套时父组件往子组件传参，应该将对象的key和value在render内**先定义再使用**render() {const age = {age:18} return ( <Demo name = {age} />); }。在传递props/state时，**只传递需要的参数**（如<Demo name={this.state.name}/>，而不是整个state传递）。否则每一次使用子组件都会生成新的对象进行传递。
       2. 多组件优化
          1. shouldComponentUpdate(nextProps, nextState)判断是否有必要更新当前组件
          2. React.PureComponent 替换 React.Component。只适用于浅层比较
          3. Facebook提供了immutable-js库，它提供了不可变的数据，即要让数据改变只能通过创建新数据的方式，而不能直接修改。
          4. immutable-js库可能比较大，有一个简易版叫做seamless-immutable，只支持Map,Set,List三种数据类型
       3. key优化。数组形式的数据遍历时要求key，key尽量不用索引，用唯一的key
    2. redux性能优化
       1. 在使用Redux进行数据的传递时，特别是经常有重复性的数据传递操作时，可以使用**reselect库**在内部对数据进行缓存处理，在重复调用时便可使用缓存快速加载，加强性能
    3. 类库按需引入等
11. React中连续多次调用setState的后果？
    1. 调用setState时，不会立即渲染，而是放到一个队列里面，等待事件结束，看setState最后渲染的那个结果，最后在进行渲染
    2. 在React中setState不是每次调用就立刻渲染的。而是放到一个队列里面，队列的顺序是在一次事件之内进行结算（比如在click事件过程中可能有很多setState在等待，等Click事件完成之后，setState这个队列里面的内容就开始进行结算了），所以setState多次调用并不会导致渲染多次。但是事务的次数可能会导致渲染。
12. 如何在固定时间刷新页面？
    1. setTimeout(()=>{window.location.reload()},1000\*60\*60)
13. Less和sass的区别
    1. Less是基于js，是在客户端处理的。定义变量用@
    2. Sass是基于ruby的，是在服务端处理的。定义变量用$
14. Git，svn的特性
    1. Git与svn核心区别就是：Git是分布式的版本控制系统。Svn是集中式管理的版本控制系统。
    2. Git与svn一样有自己的集中式版本库或服务器，但git更倾向于分布式模式。  
       开发人员从中心版本库/服务器上克隆代码到本地，本地就有了自己的版本库。确保开发人员离线仍然可以操作版本库，提交文件，查看历史版本记录，创建分支等。
    3. Git把内容按元数据方式存储，svn是按文件存储
    4. Git分支和svn不同。svn操作分支经常会发生分支被遗漏的情况，git却可以让你在同一个工作目录下快速在几个分支之间切花
    5. Svn有全局的版本号，git没有。Git的内容完整性比svn强。
15. 语法糖
    1. 语法糖又叫糖衣语法。在不影响程序运行的前提下，对语法进行包装，方便程序运行，可读性高，更方便程序员使用。
    2. Class是构造函数的语法糖。Async是generator语法糖。Promise可以说是callback的语法糖。箭头函数也是语法糖。
16. 数据持久化
    1. 单页面应用开发，vue和react状态管理时，点击增加按钮，number由1增加number到100，当刷新时number又变回原来的1？
    2. 用vuex刷新页面数据丢失。本地存储解决。
    3. 一进入页面判断本地存储是否有内容，如果有合并到vuex里面，如果没有，直接用vuex的数据
17. H5新增
    1. 语义化标签 ：header footer nav aside article section hgroup。语义化标签：名字跟它本身的概念相似
    2. 本地存储 ：sessionStorage、localStorage和indexedDB加强本地存储
    3. 离线web应用：window.applicationCache
    4. CSS3 ：before、after、first-child、nth-child，C3动画
    5. 地理定位 ：window.navigator.geolocation
    6. Canvas，Audio和video音视频
18. 项目开发中使用的规范
    1. 前端规范：模块化规范、前端工程化规范、自动化构建规范、组件化规范
    2. 模块化规范：组件建立在模块化之上。Js模块化。组件内部拆分模块
    3. 工程化规范：接口文档规范、js/css语法命名规范、命名规范  
       前端工程目录规范：项目文件的划分、模块的划分、命名规范等  
       代码规范：建立在ESlint规范之上，我们公司内部自定了一些规范，包括：等号/函数有空格、单一换行符规范等
    4. 自动化构建：打包压缩、抽离css、动态注入js  
       构建层目录规范：package.json必须有，.babelrc文件配置babel，构建本地server，代码的编译和打包，config配置等。除了src其他的都可称为基础设施层/构建层
    5. 组件化规范：拆分组件、拆分功能模块
    6. React规范：ES6方式定义组件即react.component。static方式定义props。对props做非空验证propsType。展示组件和容器组件的划分，展示组件不做任何逻辑。避免多次render。
    7. Vue规范：定义在vue实例上的方法统一挂载到vue原型上，使用了defineproperty，而不是vue.prototype，避免全局污染和误操作。项目共用组件参照elementUI，把所有组件放到数组里面，通过遍历统一注册到vue.component上。尽可能的使用computed。vuex尽可能使用map辅助函数。vuex尽可能拆分module，可维护性更强
19. 项目开发遇到的问题
    1. 移动端有300毫秒的点击延迟，可以用fastClick第三方包解决
20. 项目上线遇到的问题？
    1. 路由拦截中，我需要访问实例的一些方法，但是上线之后this访问不到。定义router，把this改为router就行。
    2. 路由拦截中，我需要触发一个action，但是拿不到store，因为异步。解决方案：settimeout、$nextTick
    3. 上传图片的时候，pm2 --watch监听文件变动。每次上传都会发生文件变动，文件一边动watch就会触发，pm2就会一直重启，服务就一直报错。
    4. 生产/开发的时候，线上/线下接口的判断。process\_ENV\_ENVNODE判断是开发还是生产
    5. 项目上线时，有使用pm2做进程守护。
       1. pm2 start不支持npm run build 的方式，可以自定义配置文件。新建build.json文件，里面是{apps:{name:’build’,script:’npm run build’,watch: true // 开启监听}}。然后启动的时候pm2 start build.json即可。Script里面可以写sheel命令
       2. 可以在全局新建文件，mysql\_bd.sh文件，里面定义好MySQL的启动命令，下次就不需要那么麻烦了,里面的内容是#!/bin/sh mysql -uroot -p1234
    6. 单页面应用使用hash方式跳转路由，带#，不好看，解决办法
       1. 更改路由：mode改为history，实现去掉url中的 /#
       2. 用history，刷新的时候会找不到页面，可以用到connect-history-api-fallback中间件。当用户在浏览器刷新的时候发送的是get请求，中间件用的就是这个原理，把原本的req的url改变
       3. nginx服务做反向代理，实现去掉端口号。nginx配置upstream实现负载均衡，如果nginx能代理多台服务器的话，可以配置多个服务器，多个服务器一起负担多人的并发请求，实现了负载均衡。
       4. 全局nginx安装完成之后再etc里面，编辑nginx.conf文件，在里面插入 upstream，更改location，nginx -s stop 停止nginx
21. ES6比ES5有那些改进？ES6新增了那些js的语法糖
    1. 改进了ES5的变量提升机制。增加了块级作用域。增加了常量的定义，解决了变量提升。
    2. 静态类的改进。字符串模板，结构赋值，对象，class类的定义，扩展的方法，箭头函数
    3. class，async。generator是ES6的。箭头函数。promise在callback的基础上做了优化
22. Axios和fetch对比？为什么项目使用axios和fetch
    1. fetch更友好，直接就是promise和then调用，更符合现在的编程思想
    2. axios是基于promise的http库，是对ajax的进一步封装，支持拦截请求和响应
23. PWA（progressive web app）
    1. 渐进式网页应用。使用多种技术增强web app的功能。核心是离线缓存web service。开发PWA是为了开发接近native的应用。为了提升用户体验。解决web app访问更友好的问题，更逼近原生应用。
24. 离线存储web service
    1. 离线缓存web service因为安全问题，限制在https和localhost本地使用
    2. create-react-app创建的src目录下有registerServiceWorker.js文件，就是web service原理
    3. 解决web app（浏览器端）端访问更友好的问题。利用的是浏览器端的缓存机制。浏览器做好缓存之后，浏览器之后所有的请求都是从web service中请求的。更逼近与原生应用
25. MVVM框架和jQuery这种大型js框架的区别
    1. mvvm数据驱动试图，数据优先。jQuery操作dom
    2. vue里面操作dom，只有2种方法：自定义指令/ref
26. 预加载和懒加载
    1. 懒加载：举例：瀑布流  
        图片懒加载：当访问一个页面的时候，先把img元素背景图片路径替换成一张大小为1\*1px图片的路径（这样就只需请求一次，俗称占位图），只有当图片出现在浏览器的可视区域内时，才设置图片正真的路径，让图片显示出来。这就是图片懒加载。  
        懒加载的原理就是先在页面中把所有的图片统一使用一张占位图进行占位，把正真的路径存在元素的“data-url”属性里，页面加载完成后，根据scrollTop判断图片是否在用户的视野内，如果在，则将data- url属性中的值取出存放到src属性中。在滚动事件中重复判断图片是否进入视野，如果进入，则将data-url属性中的值取出存放到src属性中
    2. 预加载：loading加载过程中就是在预加载、requireJS也是预加载（提前把所有依赖文件加载完成）  
       提前加载图片，当用户需要查看时可直接从本地缓存中渲染。  
       实现方式：方法一：用CSS和JavaScript实现预加载 方法二：仅使用JavaScript实现预加载 方法三：使用Ajax实现预加载
    3. ajax预加载css/js/img  
       举例：setTimeout(function () {  
        let head = document.getElementsByTagName(‘head’)[0];  
        let css = document.createElement(‘link’) css.type=’text/css’;css.rel=’stylesheet’;css.href=css-url;  
        let js = document.createElement(‘script’) js.type =’text/javascript’; js.src = js-url;  
        head.appendChild(css); head.appendChild(js);  
        new Image().src = imgSrc;  
        }, 1000)
27. Get和set
    1. 在Object.defineProperty中的数据劫持。react中可以使用。
    2. react中，get是访问器。set是修改的时候触发。返回true/false的时候使用get方式定义
28. 项目优化
    1. 减少http请求，减少操作dom数量

压缩js/css文件。在加载js/css文件时，我们需要将公共的js/css文件加载在header头中，其他的加载在相应的body中，如果我们不这样做，就会重复的加载js/css文件，js标签不重要的可以加异步async和defer

* 1. 应该把公共的js文件放在header头中，加载时一定要注意加载顺序，因为js文件会阻塞加载，可能会改变表的dom结构
  2. 精灵图，合并页面小图标，一次性加载进来，然后进行图片定位，找出需要用到的图片

1. Vue中使用 [bootcdn](http://www.bootcdn.cn/" \t "_blank) 对打包的vendor进行优化CND加速
   1. 打包 vender 时不打包 vue、vuex、vue-router、axios 等，换用国内的 [bootcdn](http://www.bootcdn.cn/" \t "_blank) 直接引入到根目录的 index.html 中
      1. 使用方法：index.html中，<script src = “//cdn.bootcss.com/vue/2.2.5/vue.min.js”></script>
      2. 在webpack中有externals属性，可以忽略不需要打包的库  
         externals：{‘vue’：‘Vue’}
   2. 还可以拆分其他的库等。
2. get和post请求特点
   1. Get 请求能缓存，Post 不能
   2. Post 相对 比Get 安全，因为Get 请求都包含在 URL 里，且会被浏览器保存历史纪录，Post 不会，但是在抓包的情况下都是一样的。
   3. Post 可以通过 request body来传输比 Get 更多的数据，Get 没有这个技术
   4. URL有长度限制，会影响 Get 请求，但是这个长度限制是浏览器规定的，不是 RFC 规定的
   5. Post 支持更多的编码类型且不对数据类型限制
3. webpack的使用
   1. webpack使用有2种情况。process\_ENV\_ENVNODE判断是开发还是生产
   2. dev开发环境：用webpack-dev-server构建开发环境，起本地服务，引入依赖，定义入口出口文件，定义modules等。
   3. production生产环境：打包静态资源、js和css做抽离、代码压缩
   4. 例如vue-cli生成的项目build下有webpack.base/dev/prod.js，base里面配置基础的出入口、依赖等，dev里面配置devServer起服务，prod就是使用各种plugin对项目的代码进行抽离和压缩。
   5. run build打包出来的dist目录下有index.html和一个static文件夹。static下的js文件夹里面有vendor.js和app.js，vendor里面是项目引入的所有类库。app.js里面是项目的component组件。当项目上线时，把vendor文件部署到CDN上，使用CDN加速，剩下的app.js很小，所以，访问速度可以达到很快。
4. 一般电商首页逻辑
   1. 首页一进入需要获取当前地理位置，也就是用户所在的城市。获取到城市ID
   2. 把城市ID传递给后台，后台 返回对应的轮播位、导航列表和内容部分
5. 编辑个人资料逻辑
   1. 点击我的，获取用户信息，渲染到页面，点击修改用户名时，input输入之后，点击确定修改，清除之前的本地存储，重新把新的用户名存储进去，同时把用户名发送到后台，后台更新用户名。当用户再次进入就是最新的用户资料了。
   2. 如果是单页面应用，返回个人信息页面时，获取本地存储好的用户名
6. seo的实现?
   1. Div+css布局，符合W3C规范的语义化标签。Meta标签优化（title，description，keywords）。
   2. 服务端渲染SSR：服务端渲染优势：更好的SEO，更快的内容到达时间，在服务端渲染一部分代码，提高页面渲染速度
   3. 判断是不是真的需要服务端渲染，预渲染也可以解决部分SEO优化，如果我们做的是营销方面的网页，可以用预渲染。如果我们做的是追求内容更快到达，可以考虑服务端渲染
   4. 前端单页面应用预渲染方案：  
      [prerender-spa-plugin@2.X版本插件定义多个html](mailto:prerender-spa-plugin@2.X版本插件定义多个html)文件，就相当于有了多个页面，很多个title和meta，对SEO友好  
      vue-meta-info，动态更改meta标签，vue的meta标签来做SEO优化，可跟prerender配合使用更好
7. TypeScript的具体阐述
   1. 把javascript开发集成为真正的面向对象的、强类型的框架，使前台的开发可以像后台开发一样，模块化，便利化，大大的提高了开发的效率
8. var 和 let 和 cont的区别
   1. 使用 let 语句声明一个变量，该变量的范围限于声明它的块中。  可以在声明变量时为变量赋值，也可以稍后在脚本中给变量赋值。
   2. 使用 let 声明的变量，在声明前无法使用，否则将会导致错误。
   3. 如果未在 let 语句中初始化您的变量，则将自动为其分配 JavaScript 值 undefined
   4. 重复声明一个变量时。var会覆盖上一个变量的赋值，而let则会报错你已经定义过了
   5. Var的作用域是最近的函数作用域；let的作用域是最近的块作用域，会比函数作用域小
   6. 举例：一个for循环放在了函数中，let只在这个for里面有效，而var是在整个函数里面有效
9. 单向数据流的解释
   1. 单向数据流指只能从一个方向来修改状态。View里面触发action，action更改state状态，state的改变反应到view。
   2. 子组件内部不能直接修改从父组件传递过来的数据。如果用了vuex/redux，可以间接修改
10. 实现深拷贝和浅拷贝的方法
    1. 深拷贝和浅拷贝的区别：内存地址的复制
    2. Object.assign（{}，obj1，obj2.。。。）只能做浅拷贝
    3. JSON.parse(JSON.stringify(obj1)) 深拷贝，缺点：如果obj1是{ fun: function(){ console.log(123) } }这种function，就没办法了
    4. jQuery提供的$.extend(true, {}, obj1，obj2.。。);值为true就是深拷贝，为false就是浅拷贝
    5. lodash函数库，提供\_.cloneDeep来做深拷贝
    6. 递归遍历深拷贝
11. v-model的原理
    1. <input v-model="sth">相当于<input v-bind:value="sth" v-on:input="sth=$event.target.value">自从html5开始input每次输入都会触发oninput事件，所以输入时input的内容会绑定到sth中，于是sth的值就被改变
12. object的方法
    1. Object.assign方法实行的是浅拷贝，而不是深拷贝。也就是说，如果源对象某个属性的值是对象，那么目标对象拷贝得到的是这个对象的引用。同名属性会替换
    2. Object.create(parent, {z : { writable:true, configurable:true, value: "newAdd"} }) 使用指定的原型对象及其属性去创建一个新的对象
    3. Object.defineProperty(Object , ‘is’,{value:’a’, configurable:true, enumerable: false, writable: true}) 在一个对象上定义一个新属性，或者修改一个对象的现有属性， 并返回这个对象
    4. Object.keys(obj) ，Object.values(obj)，Object.entries(obj)返回一个由一个给定对象的自身可枚举属性/属性值/键值对组成的数组，排列顺序和使用 for...in 循环遍历该对象时返回的顺序一致。与for in的区别是for in循环会枚举原型链上的属性。
    5. obj.hasOwnProperty('name') 判断对象自身属性中是否具有指定的属性
    6. Object.getPrototypeOf()返回指定对象的原型（内部[[Prototype]]属性的值，即\_\_proto\_\_，而非对象的prototype）
    7. Object. isPrototypeOf()判断一个对象是否存在于另一个对象的原型链上
    8. Object.setPrototypeOf(obj,prototype) 设置对象的原型对象
    9. Object.is()判断两个值是否相同。如果两个值都是undefined/null/true/false/NaN/指向同一个对象，则两个值相同
13. 小程序开发时只能用https的url
    1. 不支持http，可以在微信开发者工具中勾选“不校验合法域名”，即可正常请求。
    2. 可以在微信公众平台小程序部分在开发设置里面配置服务器域名。完成真实的接口调试。
14. 闭包？
    1. js只有全局和局部、函数作用域，没有块级作用域，闭包可实现块级作用域效果
    2. 我们外部要用到函数内部的一些变量和方法是不行的，这就用到闭包了，闭包是指有权访问另一个函数作用域中的变量的函数。 就是函数内部在创建另一个函数。因为闭包的关系可以模拟出私有的方法，来定义一些公共的函数，并且可以访问私有函数的变量，就比如说react中一个模块来extends来继承react.Component中的一些方法，就是利用了闭包，把一些私有的方法抛出，所以this上就有这些方法
    3. 使用闭包产生内存泄漏问题，目前闭包只会存在于IE，chrome和firefox已经可以做到垃圾回收
15. Gitflow工作流？
    1. 首先项目开发人员由dev创建功能分支进行开发，开发完成后向leader发起pull request，和code review（代码审核，先让领导看，如果领导一下就看出了问题，就没话了），完毕之后，合并到dev分支。由测试人员从dev分支拉取release测试分支，如果没问题，不做操作。如果有问题，由dev分支拉取代码到fixbug分支进行修复，完毕之后，合并到dev和release分支
16. 什么是generator函数？
    1. Generator不同于普通函数，所以要加\*来作为标识，它是一种状态机，封装了多个内部状态，执行的函数会返回一个遍历器对象，可以依次依赖generator函数内部的每一个状态
    2. 。在generator函数内部使用yield来定义不同的内部状态，可以暂停函数执行，可以通过next的方式调用，generator函数会从上一次暂停的地方继续执行，找见yield暂停。
    3. 在遍历器对象中有一个done的可以key值，是false表示还没有执行完，true表示函数内部没有课执行的东西了。yield就是暂停的标志，也就是异步的分界线。函数内部有return的话会当做值来输出，没有的话会返回undefined。ES6新增了async语法糖。
17. 数据类型
    1. 基本数据类型：number、string、boolean、null、undefined，引用数据类型：Array、object、function，ES6：symbol
    2. 基本数据类型，名值存储在栈（zhan）内存中。等号赋值时，栈内存会开辟一个新内存。
    3. 引用数据类型：名存在栈内存中，值存在堆内存中，但是栈内存会提供一个引用的地址指向堆内存中的值。
18. 冰桶游戏：canvas+js实现游戏
    1. 在canvas画布上覆盖一层div，div按照九宫格划分，绑定点击事件，当点击某一个格的时候，触发冰桶的出现。
19. 面试话术
    1. 自己负责的模块开发完成之后，不是接手别人已经开发一部分的模块熟悉代码，而是接手新的功能模块进行开发。
    2. 刚进入公司需要多久适应时间就可以上手开发？ 半天/1天
    3. 官网首页从开发到测试，3,4天时间足够。详情页更快
    4. CMS系统，一般没有原型图，只有业务需求。为了快速开发搭建项目，我们使用了ElementUI库。
       1. 登录页面使用了基于canvas的particle粒子图。
       2. 在数据筛选时，每次都传递一个条件对象，把筛选条件跟vuex里面定义的edit对象合并之后发送到后台，后台返回符合条件的数据
    5. 项目怎么分配？
       1. 首先，我手上可能是有项目的，我把项目做完之后，在参与到新项目中
       2. 如果类似购物车之类的，我会先向后台要UID，如果没有合适UID，就本地mock数据
       3. leader分发任务的时候,一般为了快速开发，我会跟后台要一个uid提前异步开发。如果同步开发耽误进程。
    6. 做播放器：用H5的video标签做视频的请求截流。，后台把视频流分段，分为很多个Http请求，前台只负责分段发送请求
    7. 编辑个人资料
       1. 点击我的，获取用户信息，渲染到页面，点击修改用户名时，input输入之后，点击确定修改，清除之前的本地存储，重新把新的用户名存储进去，同时把用户名发送到后台，后台更新用户名。当用户再次进入就是最新的用户资料了。
       2. 如果是单页面应用，返回个人信息页面时，获取本地存储好的用户名，就可以正常显示更改过后的用户名
20. Vue中渲染方式render和template
    1. 用vue-cli脚手架下载下来之后的vue项目目录里面的node\_modules下的vue文件夹里面，你将会找到很多不同的 Vue.js 构建版本
    2. CommonJS和ESModule负责项目使用ES5还是ES6语法，完整版和runtime版负责项目是用template还是render渲染
    3. Runtime运行时版本（vue.runtime.common/esm.js），只能用render渲染，如果用components+template方式就会报错：“你正在使用runtime方式构建vue项目，template模板编译器不被允许，你可以把模板预编译成渲染函数render，或者使用包含template模板编译器的完整编译方式”
    4. 而完整版(vue.common/esm.js)，render和components+template都可以。
21. 判断对象是否为数组
    1. objectName instanceof Array
    2. objectName.constructor == Array
    3. 特性判断：有length属性和splice方法，并且length不可枚举才是数组object.propertyIsEnumerable('length')
    4. ES6方法Array.isArray(objectName);
    5. Object.prototype.toString.call(objectName) === ‘[object Array]‘;
22. 跨域
    1. server2server数据中转
       1. nodeFetch第三方包在服务端请求数据，做数据中转，实现跨域。
       2. 原理：本地向线上自己的服务器发起请求，然后由服务器向真正的接口请求数据
    2. Cors：设置响应头res.Header
    3. Jsonp，script标签的src属性，本质上是一次script外链资源的访问
    4. Proxy代理
    5. Window.name
    6. H5新增的window.postMessage
    7. Web socket，跟postMessage类似，是服务端向客户端推送数据
    8. 修改document.domain
23. 移动端适配
    1. 媒体查询@media screen + 响应式布局，bootstrap
    2. 样式缩放，直接用px开发，然后计算屏幕与网页的宽高比，用transform：scale进行网页全局缩放，不推荐
    3. 网易方案：除了font-size，其他全部用rem布局，通过js计算分辨率的改变，html的font-size随之改变。设置了临界点，分辨率大于1080时，font-size不会变化了
    4. 淘宝的flexbile.js，原理：动态通过js改变meta的viewport的缩放比，同时改变HTML的font-size。也设置了临界点
24. 前后端交互
    1. 前端请求数据URL和接口文档主要由后台来写，前端发现问题可协商更改
    2. 前后端交互的数据格式
    3. 前端应该告知后台那些有效信息，后台才能返回前端想要的数据
    4. 前端应该如何回拒一些本不属于自己做的一些功能需求或任务
25. Git分支冲突
    1. 首先，我们的git分支划分很规范，一般不会遇到问题。如果遇到问题可以进行代码检索，这个有相关的工具可以帮助我们，比如sourcetree
    2. 如果冲突文件比较多，我们也可以采用版本回退的方式，当然，在回退之前需要把自己本地的代码进行备份，然后用reset HEAD^版本号，回滚到上一个冲突之前的版本，本地修改冲突。
26. Android和ios兼容
    1. IOS中，Margin失效，Android无问题
    2. IOS中，定位问题，position带来的浮动问题（Android的问题）
    3. IOS中， input的keyup事件不会立即响应，只有通过删除才可以响应
    4. IOS中，设置input按钮样式会被默认样式覆盖
    5. 移动端 HTML5 audio autoplay 失效问题
    6. 移动端 video在部分android机播放之后浮在最上层，设置z-index无效
    7. 移动端点击300ms延迟，fastclick解决
    8. iOS 系统中，中文输入法输入英文时，字母之间可能会出现一个六分之一空格
    9. iOS端字体的优化(横竖屏会出现字体加粗不一致等)
27. 原型链
    1. \_\_proto\_\_是只有对象有的，prototype是函数有的
    2. 构造函数实例返回的是对象，对象就有隐式原型
    3. 定义在prototype上的都会最后显示在实例对象的\_\_proto\_\_隐式原型上
    4. 实例的隐式原型指向构造的显式原型
    5. instance.\_\_proto\_\_.\_\_proto\_\_ 是Object对象
    6. instance.\_\_proto\_\_.constructor 是构造函数本身
    7. 原型链是隐式原型形成的
    8. 所有引用类型都有proto属性，隐式原型
    9. 所有函数拥有prototype显式原型
28. 继承
    1. 构造函数继承，call和apply方法
    2. 原型继承，A.prototype=new B()
    3. 直接继承，A.prototype = B.prototype
    4. 拷贝继承，遍历A所有属性，赋值给B
    5. 组合继承 = 构造继承 + 原型继承