生命周期函数面试题

1.什么是 vue 生命周期
是vue实例从创建到销毁的过程，开始创建，初始化数据，编译模版，挂载dom=》渲染、更新=》渲染、销毁等一系列过程，称之为生命周期.

2.简述每个周期具体适合哪些场景
beforecreat, 创建前，数据观测和初始化事件还未开始
created, 完成数据观测，初始化事件，属性和方法的运算，$el属性还没有显示出来
beforeMount, html挂载到页面之前之前，编译模版，把data里的数据和模版生成html
mounted, 编译好的html内容替换el属性指向的dom对象，html渲染到页面上，此过程进行ajax交互
beforeUpdate, 数据更新之前
updated,数据更新之后
beforeDestroy,实例销毁之前，实例仍然可用
destroyed，实例销毁之后，事件监听器被移除，所有的子实例会被销毁

3.vue生命周期的作用是什么
他的生命周期中有多个事件勾子，方便我们在组件各个时间段进行操作，形成更好的逻辑

4.第一次页面加载会触发哪几个钩子
beforeCreate, created, beforeMount, mounted

5.created和mounted的区别
dom实例是否挂载在页面上

6.vue获取数据在哪个周期函数
如果没有涉及到操作Dom, Created中，否则就是Mounted

vue路由面试题

1.mvvm 框架是什么?
model：数据模型，也可以在model中定义数据修改和操作的业务逻辑
view：ui组件，视图层
viewModel：监听模型数据的改变和控制视图行为、处理用户交互，
是一个同步view和model的对象，连接model和view
MVVM架构下，view和model并没有直接的联系，而是通过viewModel进行双向交互，数据改变出发视图更新，视图上的交互改变数据

好处：只需要关注业务逻辑，不需要手动操作dom,不需要关注数据状态的同步问题

2.vue-router 是什么?它有哪些组件
vue路由（路径）管理器，路由模块的本质 就是建立起url和页面之间的映射关系
router-link，router-view

3.active-class 是哪个组件的属性?

vue-router，router-link,当链接被点击时将会应用这个样式。用在首页时的active会一直被应用，搭配exact

4.怎么定义 vue-router 的动态路由? 怎么获取传过来的值

在router目录下的 index.js 文件中,对path加上/:id, 使用params.id获取

5.vue-router 有哪几种导航钩子?

全局: 前置router.beforeEach((to,from,next)=>{})，next()必须调用，否则函数无法resolved
后置router.afterEach((to,from)=>{})

单独路由独享： beforeEnter(to,from,next) {}

组件内: beforeRouterEnter(to,from,next) {}
beforeRouterUpdate(to,from,next) {}
beforeRouterLeave(to,from,next) {}

6.$route 和 $router 的区别

$route是路由信息对象，比如path,name等路由参数信息

$router是路由实例对象，包含了路由的跳转方法，钩子函数等

7.vue-router响应路由参数的变化

<router-view :key="$route.fullPath"></router-view>

复用组件时，想对路由参数的变化作出响应的话，你可以简单地 watch (监测变化) $route 对象：

```
const User = {
  template: '...',
  watch: {
    '$route' (to, from) {
      // 对路由变化作出响应...
    }
  }
}
```

8.vue-router传参

Params，query,接收参数的时候是route

params要用name引入，类似于post

query要用path引入，类似get

```
<!-- 命名的路由 -->
<router-link :to="{name:xxx, params:{key:value}}">valueString</router-link>
<!-- 通过$route.params.key来获取-->

<!-- 带查询参数，下面的结果为 /xxx?key=value -->
<router-link :to="{path:xxx, query:{key:value}}">valueString</router-link>
<!-- 通过$route.query.key来获取—>
```

9.vue-router的两种模式

Hash模式: 在浏览器中#和#后面的字符成为hash,用window.location.hash获取

特点：hash虽然在url中，但是不被包括在http请求中（浏览器不会请求服务器，不会重新加载页面，但是页面状态和url已经关联起来了，也叫前端路由）

History模式： 采用h5新特性，提供两个新的方法，pushState(),replaceState()可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。需要后台配置，如果 URL 匹配不到任何静态资源会404

10.vue-router实现路由懒加载（ 动态加载路由 ）

```
resolve=>require([‘@/path’, resolve])
{
  path: '/home',
  name: 'home',
  component: resolve => require([‘@/components/home'],resolve)

}
```

下面2行代码，没有指定webpackChunkName，每个组件打包成一个js文件。

```
/* const Home = () => import('@/components/home')
const Index = () => import('@/components/index')
const About = () => import('@/components/about') */
```

下面2行代码，指定了相同的webpackChunkName，会合并打包成一个js文件。 把组件按组分块

```
const Home =  () => import(/* webpackChunkName: 'ImportFuncDemo' */ '@/components/home')
const Index = () => import(/* webpackChunkName: 'ImportFuncDemo' */ '@/components/index')
const About = () => import(/* webpackChunkName: 'ImportFuncDemo' */ '@/components/about')
```

vue常见面试题

1.vue优点

mvvm模式，中文文档，轻量级

2.vue父组件向子组件传递数据?

Prop

3.子组件像父组件传递事件

$emit

4.v-show和v-if指令的共同点和不同点

V-show控制视图显示和隐藏

v-if控制dom是否渲染

5.如何让CSS只在当前组件中起作用
scope

6.<keep-alive></keep-alive>的作用是什么?
用来缓存组件，避免多次加载相同的组件，减少性能消耗，如果需要频繁切换路由，这个时候就可以考虑用keep-alive了，来达到避免数据的重复请求的目的

7.如何获取dom
$refs

8.说出几种vue当中的指令和它的用法?
V-on绑定事件
V-if
V-show
v-for 遍历

9. vue-loader是什么？使用它的用途有哪些?
基于webpack的loader，解析和转换.vue文件，提取其中的javascript逻辑代码，样式代码style，template模版，再分别把它们交给对应的loader去处理，还有一些打包、压缩的功能等

10.为什么使用key
组件在进行v-for更新渲染过的元素列表的时候，默认用就地复用的策略，当数据的顺序发生改变，vue将不会移动dom元素来匹配数据的顺序，而是简单的复用此处的每个元素，并且确保它在特定的索引下显示已经被渲染过的每个元素，key就是为了高校的更新虚拟dom

11.axios及安装
Nom install axios —save
12.axios解决跨域
在vue.config.js里面设置proxy代理，在接口前加上代理的api

13.v-modal的使用
v-model用于表单数据的双向绑定，其实它就是一个语法糖，这个背后就做了两个操作：
1. v-bind绑定一个value属性
2. v-on指令给当前元素绑定input事件

14.scss的安装以及使用

15. 请说出vue.cli项目中src目录每个文件夹和文件的用法?
Components组件
Assets静态资源
router定义路由相关配置
Main.js入口文件
app.vue是应用主组件

16.分别简述computed和watch的使用场景

computed计算属性是群体监听，相关依赖发生改变就会触发，会有缓存
watch是单个监听

17.v-on可以监听多个方法吗
可以<button @click="a(),b()">点我ab</button>

```
<button @mouseenter="onEnter" @mouseleave="onLeave">鼠标进来2</button>
```

18.$nextTick的使用
dom加载完全以后再触发，等待dom生成以后再来获取dom对象或值

19.vue组件中data为什么必须是一个函数
data作为一个函数，每次复用组件会生成对应的data，相当于每个组件实例创建一个私有的数据空间，让各个组件实例维护各自的数据，而单纯的写成对象形式，就使得组件实例共享了一份data，就会造成一个变了全都会变

20.vue事件对象的使用
.stop 阻止冒泡
.once只执行一次
.self只有触发自己的时候才会执行
.capture事件行为在捕获阶段执行

21 组件间的通信
路由，
eventBus，

```
bus.$emit('globalEvent',val)
bus.$on('globalEvent',(val)=>{
            this.brothermessage=val;
        })
```

Vuex

22.渐进式框架的理解
搭积木，我们可以根据需求，利用社区良好的生态，借助已有的工具和库搭建我们的项目，用最小、最快的成本一步步搭建

23.Vue中双向数据绑定是如何实现的
结合发布者订阅者模式，通过object.defineProperty()来劫持各个属性的geter()和seter(),在数据变动时发布消息给订阅者触发相应回调

24.单页面应用和多页面应用区别及优缺点
单页应用:只有一个主页面的应用，浏览器在一开始就加载所有的资源，相当于所有的内容都包含在主页面上，单页面的页面跳转由路由程序动态载入，只是做局部刷新
用户体验好，内容改变时不需要刷新整个页面所以服务器压力小，容易实现转场动画
不利于seo,初次加载耗时多，页面复杂度高
多页应用：应用有多个页面，页面之间的跳转是整个页面刷新

25.vue中过滤器有什么作用及详解

filter
{msg | filterMethods}
Data:{},
Filters:{
  filterMethods(){}
}

26.v-if和v-for的优先级

v-for的优先级高于v-if,v-if 将分别重复运行于每个 v-for 循环中，v-if最好写在外面

27.assets和static的区别

assets的资源会进行打包（压缩体积格式化代码）

static不会走打包等流程，直接进入打包好的目录，适合第三方已经处理过的文件

29.vue常用的修饰符

事件修饰符。 .stop .once .prevent .self

<div class="" @click.self="outer">

键盘修饰符 .enter .up .space .left .delete

<!-- 只有在 `keyCode` 是 13 时调用 `vm.submit()` -->
<input v-on:keyup.13="submit">
<!-- 同上 -->
<input v-on:keyup.enter="submit">
<!-- 缩写语法 -->
<input @keyup.enter="submit">

30.数组更新检测

.pop() .push() .shift() .splice() .sort() .reverse()

31.Vue.set视图更新

$set(). $forceUpdate()

33.vue的两个核心点

数据驱动，组件系统

34.vue和jQuery的区别

jq是通过选择器操作dom对象

Vue，数据和视图是分离的，对数据进行操作不需要操作dom

35 引进组件的步骤

Important 然后在components里面写好组件名

36.Vue-cli打包命令是什么？打包后导致路径问题，应该在哪里修改

npm run build,config/index.js,找到build.assetsPublicPath: './'

39.delete和Vue.delete删除数组的区别

delete只是把数组的值变成了empty/undefined，但是还是保留下标,其他元素键值不变

vue.delete是从数组中移除，改变了数组的键值

40.SPA首屏加载慢如何解决

1、按需加载

2、首页加载loading动图
3、将公用的JS库通过script标签外部引入，减小\app.bundel的大小，让浏览器并行下载资源文件，提高下载速度

41.Vue-router跳转和location.href有什么区别
Location.href是整个页面刷新，重新加载页面
vue-router是局部加载

42. vue slot
内容分发

43.你们vue项目是打包了一个js文件，一个css文件，还是有多个文件?
一个js，一个css

44.vue遇到的坑，如何解决的?

45.Vue里面router-link在电脑上有用，在安卓上没反应怎么解决?
@click.native

46.Vue2中注册在router-link上事件无效解决方法
@click.native

47.RouterLink在IE和Firefox中不起作用（路由不跳转）的问题
router.navigate

48.axios的特点有哪些
在浏览器中发送XMLHttpRequest请求
在node.js中发送http请求
支持promise API
拦截，转换请求和响应
axios拦截器
创建axios实例

```
Const service = axios.creat({
baseUrl: ‘’,
header:{}
})
Service.interceptors.request.use(
config=>{
if(config.method==‘post’){
  //处理
}
Return config
},
Err =>{
Return promise.reject(err)
}
)
service.response.use(
Response=>{
}
)
```

49.请说下封装 vue 组件的过程?

50.vue 各种组件通信方法（父子 子父 兄弟 爷孙 毫无关系的组件）

51.params和query的区别

52. vue mock数据

53 vue封装通用组件

54.vue初始化页面闪动问题

55.vue禁止弹窗后的屏幕滚动

```
methods : {
  //禁止滚动
  stop(){
      var mo=function(e){e.preventDefault();};
      document.body.style.overflow='hidden';
      document.addEventListener("touchmove",mo,false);//禁止页面滑动
   },
   /***取消滑动限制***/
   move(){
      var mo=function(e){e.preventDefault();};
      document.body.style.overflow='';//出现滚动条
      document.removeEventListener("touchmove",mo,false);
   }
}
```

56.vue更新数组时触发视图更新的方法

57.vue常用的UI组件库

elementUi,Iview

58. vue如何引进本地背景图片

Import … from ‘’

59. vue如何引进sass

60.vue修改打包后静态资源路径的修改

61.vue中封装公共方法，全局使用

1.util下面新建方法文件uni.js

```
Export default{
   functionName: function(){}
}
```

2.在main.js中引入

```
Import uni.js from ‘@/unit/uni.js’
Vue.prototype.uni = uni
```

3.使用

```
this.change = this.uni.functionName()
```

vuex常见面试题

1.vuex是什么？怎么使用？哪种功能场景使用它?
状态管理，在main.js引入store，注入,新建了一个目录store.js，….. export
场景有：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

2.vuex有哪几种属性
State,getter,mutation,action,module
state=>基本数据存放
getter=>基本数据派生出来的数据
mutation=>提交更改数据的方法，同步
action=>装饰器，包裹mutations使之可以异步
module=>模块化vuex

3.不使用Vuex会带来什么问题

4.Vue.js中ajax请求代码应该写在组件的methods中还是vuex的actions中?
如果数据不需要公共就直接卸载组件内部，需要复用就action

5.vuex一个例子方法
6.Vuex中如何异步修改状态
在调用vuex中的方法action的时候，用promise实现异步修改

```
const actions = {
    asyncInCrement({ commit }, n){
        return new Promise(resolve => {
            setTimeout(() => {
                commit(types.TEST_INCREMENT, n);
                resolve();
            },3000)
        })
    }
}
```

7.Vuex中actions和mutations的区别
事实上在 vuex 里面 actions 只是一个架构性的概念，并不是必须的，说到底只是一个函数，你在里面想干嘛都可以，只要最后触发 mutation 就行。异步竞态怎么处理那是用户自己的事情Action 提交的是 mutation，而不是直接变更状态。
Action 可以包含任意异步操作。 个人觉得这个action的 产生就是 因为mutation 不能进行异步操作，如果有异步操作那么就用action 来提交mutation

```
actions: {
  incrementAsync ({ commit }) {
    setTimeout(() => {
      commit('increment')
    }, 1000)
  }
}
```

vue项目实战

1.顶部悬停效果

2.电话本列表效果（ 右边字母分类 上下滑动 旁边字母显示高亮）

3.vue做代理

4.Vue路由切换时的左滑和右滑效果示例

html

```
<transition :name="transitionName">
  <keep-alive>
     <router-view class="transitionBody"></router-view>
  </keep-alive>
</transition>
.transitionBody{
 transition: all 0.15s ease; /*定义动画的时间和过渡效果*/
}
```

Css

```
.transitionLeft-enter,
.transitionRight-leave-active {
  -webkit-transform: translate(100%, 0);
  transform: translate(100%, 0);
   /*当左滑进入右滑进入过渡动画*/
}

.transitionLeft-leave-active,
.transitionRight-enter {
  -webkit-transform: translate(-100%, 0);
  transform: translate(-100%, 0);
}
```

Js

```
export default {
 data() {
   return {
    transitionName: 'transitionLeft'
   };
 },
 watch: {
  '$route' (to, from) {
   const arr = ['/goods','/ratings','/seller'];
   const compare = arr.indexOf(to.path)>arr.indexOf(from.path);
   this.transitionName = compare ? 'transitionLeft' : 'transitionRight';
  }
 }
}
```

ES6面试题

ES6新增方法面试题

1.let const var比较

Var ，变量提升，可以重复声明，全局作用域

let ，不可变量提升，不可重复声明，块级作用域

eg:

```
function(let a=1;a<5;a++){
    Let a = 3
    Console.log(a) //3
}
Console.log(a) //not defined 没有申明
```

const定义一个常量，定义的时候必须赋值，不可提升不可重复声明，块级作用域

2.反引号（`）标识

字符串模版

3.函数默认参数

4.箭头函数

5.属性简写

6.方法简写

7.Object.keys()方法，获取对象的所有属性名或方法名

8.Object.assign ()原对象的属性和方法都合并到了目标对象

9.for...of 循环

10.import和export

11.Promise对象

```
Function myPromise(constructor){
  Let self = this
  Self.status = ‘pedding’
  Self.value = undefined
  Self.reason = undefined
  Function resolve(value){
    if(self.state == ‘pedding){
       This.state = ‘resolved’
        Self.value = value
     }
  }
Function reject(reason){
     if(self.state===‘pedding){
        Self.state = ‘rejected’
        Self.reason = reason
      }
    }
Try{
    constructor(resolve,reject)
}catch(e){reject(e)}
}
myPromise.prototype.then = function(onFullFilled,onRejected){
   Let self = this
   Switch (self.state){

       Case ‘resolved’:
```

```
      onFullFilled(self.value);
      Break;

      Case ‘rejected’:
      onRejected(self.reason);
      Break;

      default;
    }
}
```

12.解构赋值

13.set数据结构（可用于快速去重）

```
Const s = new Set([…Arr])
```

14.Spread Operator 展开运算符(…)

数组是一维数组时，扩展运算符可以深拷贝一个数组

当数组为多维时，数组中的数组变成浅拷贝（对象同理）

扩展：

浅拷贝是一个传址,也就是把a的值赋给b的时候同时也把a的址赋给了b，当b（a）的值改变的时候，a（b）的值同时也会改变

15.字符串新增方法

include是否包含

startWith开始位置是否包含

repeat（n）字符串就重复多少次

ES6数组面试题

1.forEach()

2.map() //所有分别进行一个操作

3.filter() //所有分别进行一个某个条件的筛选操作

4.reduce() //所有一起进行某个操作

5.some()

6.every()

7.all()方法

8.from() //把类数组转换成真的数组

9.of() //创建一个数组

ES6编程题

1.使用解构，实现两个变量的值的交换
```
[a,b] = [b,a];
```

2.利用数组推导，计算出数组 [1,2,3,4] 每一个元素的平方并组成新的数组。
```
Var arr = []
Arr.map(item => {arr.push(item*item)})
```

3.使用ES6改下面的模板

4.把以下代码使用两种方法，来依次输出0到9?

react面试题

react生命周期面试题

1.react 生命周期函数
2.react生命周期中，最适合与服务端进行数据交互的是哪个函数
3.运行阶段生命周期调用顺序
4.shouldComponentUpdate 是做什么的，（react 性能优化是哪个周期函数？）
5.指出(组件)生命周期方法的不同

react 基础面试题

1.React 中 keys 的作用是什么?
2.React 中 refs 的作用是什么?
3.React 中有三种构建组件的方式
4.调用 setState 之后发生了什么?
5.react diff 原理（常考，大厂必考）
6.为什么建议传递给 setState 的参数是一个 callback 而不是一个对象
7.除了在构造函数中绑定 this，还有其它方式吗
8.setState第二个参数的作用
9.(在构造函数中)调用 super(props) 的目的是什么
10.简述 flux 思想
11.在 React 当中 Element 和 Component 有何区别?
12.描述事件在 React 中的处理方式。
13.createElement 和 cloneElement 有什么区别?
14.如何告诉 React 它应该编译生产环境版本?
15.Controlled Component 与 Uncontrolled Component 之间的区别是什么?

react组件面试题

1.展示组件(Presentational component)和容器组件(Container component)之间有何不同
2.类组件(Class component)和函数式组件(Functional component)之间有何不同
3.(组件的)状态(state)和属性(props)之间有何不同
4.何为受控组件(controlled component)
5.何为高阶组件(higher order component)
6.应该在 React 组件的何处发起 Ajax 请求
7.react中组件传值
8.什么时候在功能组件(Class Component)上使用类组件(Functional Component)?
9.受控组件(controlled component)与不受控制的组件(uncontrolled component)有什么区别?
10.react 组件的划分业务组件技术组件?

redux面试题

1.redux中间件
2.redux有什么缺点
3.了解 redux 么，说一下 redux 把

react性能比较面试题

1.vue和react的区别
2.react性能优化的方案
3.React 项目用过什么脚手架
4.介绍一下webpack webpack
5.如果你创建了类似于下面的 Twitter 元素，那么它相关的类定义是啥样子的?
6.为什么我们需要使用 React 提供的 Children API 而不是 JavaScript 的 map?

js面试题

1.简述同步和异步的区别
同步是

2.怎么添加、移除、复制、创建、和查找节点
creatDocumentFragment() //dom片段

creatElement //具体元素
creatTextNode() //文本节点

removeChild()
appendChild()
replaceChild()
insertBefore()

getElementById()
getElementsByClass()
getElementsByTagName()

3.实现一个函数clone 可以对Javascript中的五种主要数据类型（Number、string、Object、Array、Boolean）进行复制

```
Function copy(obj){
  if(typeOf(obj)==‘object’){
     Var result = obj.constructor == Array ? [] : {}
    for(let i in obj) {
      result[i] = typeOf obj[i] == ‘object’ ? copy(obj[i]) : obj[i]

  }else{
    Var result = obj
}
  Return result
}
```

4.如何消除一个数组里面重复的元素

5.写一个返回闭包的函数

```
Function aa (){
  Var I = 0
  Return ha(){
    console.log( i+ 1)
 }
}
Var hg = aa
hg()
```

6.使用递归完成1到100的累加

```
function add(num){
      if(num == 1){
          return 1
      }
      return add(num - 1) + num
   }
   var s = add(5)
   console.log(s)
```

7.Javascript有哪几种数据类型
Number,string,boolean,undefined,null

8.如何判断数据类型
typeOf

9.console.log(1+’2’)和console.log(1-'2')的打印结果
12,-1

10.Js的事件委托是什么，原理是什么
利用冒泡原理，将事件加到父级上出发执行效果

11.如何改变函数内部的this指针的指向
apply(arr)，call(1,2,3)

12.列举几种解决跨域问题的方式，且说明原理
Proxy
Jsonp 动态生成script标签执行js脚本

13.谈谈垃圾回收机制的方式及内存管理
定期找到不再继续使用的变量释放内存，遍历所有可访问的对象回收不可访问的对象

14.写一个function ，清除字符串前后的空格
str.trim()
全局。。str.replace(/ \s */g, ‘’)

15.js实现继承的方法有哪些

原型链继承

存在的问题：通过原型链实现继承时，原型实际上会变成另一个类型实例，而原先的实例属性也会变成原型属性，如果该属性为引用类型时，所有的实例都会共享该属性，一个实例修改了该属性，其它实例也会发生变化，同时，在创建子类型时，我们也不能向超类型的构造函数中传递参数

借用构造函数

优点：解决了原型链继承中引用类型的共享问题，同时可以在子类型构造函数中向超类型构造函数传递参数

缺点：定义方法时，将会在每个实例上都会重新定义，不能实现函数的复用

寄生式继承

```
function Parent（name）{this,name = name}
Parent.prototype.say = function(){console.log(this.name)}
Function Child(name,parentName){Parent.call(this,parentName)  this, name = name}
Function Creat (proto){function F(){}  F.prototype = proto}. Return new F()}

Child.proptype = creat(Parent.prototype)
Child.prototype.say = function(){console.log(this.name)}
Child.prototype.construction = Child
```

16.判断一个变量是否是数组，有哪些办法

isArray

```
function isArray (obj) {
  return Array.isArray(obj);
}
```

instanceOf

```
function isArray (obj) {
  return obj instanceof Array;
}
```

18.箭头函数与普通函数有什么区别
箭头函数全都是匿名函数
外观不一样
this指向不一样，普通函数是调用的作用域，箭头函数是定义时候的作用域
箭头函数不能作为构造函数
尖头函数不具有arguments对象

19.随机取1-10之间的整数
Math.radom*10 + 1

20.new操作符具体干了什么
1.创建一个对象
2.将对象的原型指向prototype
3.指向正确的原型
4.返回这个对象

21.Ajax原理
XMLHttpRequest（）

22.模块化开发怎么做

23.异步加载Js的方式有哪些

24.xml和 json的区别
25.webpack如何实现打包的
26.常见web安全及防护原理

27.用过哪些设计模式
状态模式
根据状态改变，切换行为模式。例如在权限管理中，刚注册时，只有查看自己资料的权限；开通产品后，创建相应的权限。
观察者模式
或者说是消息订阅模式。监听器管理订阅事件，将监听器注册到事件总线。生产者用事件总线发布消息，事件总线触发监听器。（eventBUs）

28.为什么要同源限制
29.offsetWidth/offsetHeight,clientWidth/clientHeight与scrollWidth/scrollHeight的区别
30.javascript有哪些方法定义对象
31.说说你对promise的了解
32.谈谈你对AMD、CMD的理解
33.web开发中会话跟踪的方法有哪些
34.介绍js有哪些内置对象?
35.说几条写JavaScript的基本规范?
36.javascript创建对象的几种方式?
37.eval是做什么的?
38.null，undefined 的区别?
39.[“1”, “2”, “3”].map(parseInt) 答案是多少?
1，NaN,NaN
parseInt(string, 要解析的数字基数)
parseInt（’1’，0）
parseInt（’2’，1）
parseInt（’3’，2）

40.javascript 代码中的”use strict”;是什么意思 ? 使用它区别是什么?

41.js延迟加载的方式有哪些?

42.defer和async
Defer 先下载script文件，等文档解析完成以后再执行js
Async，下载script立即执行，同时进行文档解析

43.说说严格模式的限制
44.attribute和property的区别是什么?

45.ECMAScript6 怎么写class么，为什么会出现class这种东西?

```
Class Pap (name,age){
  constructor(name,age){
    This.name = name
    This.age = age
    Return this
   }
   Prin (){}
}
Class chid extends Pap(){
constructor(name,age){}
```

```
   super(name,age)
This.h = Pap.prin()
Return this
}
```

46.常见兼容性问题

47.函数防抖节流的原理

防抖是创建一个定时器，在规定时间内触发则会更新这个定时器，规定时间过后再触发

节流是在某个时间段内不执行

```
Function debt (fn,awit){
   Let timmer
   Return function(){
     if(timmer) clearTimeOut(timmer)
     Timmer = setTimeOut(()=>{
         fn.apply(this,arguments)
   },awit)
  }
}
Function the (fn,awit){
   Let pre = new Date()
   Return function(){
     Const now = new Date()
     if(now - pre > awit){
       Fn.apply(this,arguments)
       Pre = new Date()
}
}
}
```

48.原始类型有哪几种？null是对象吗?

49.为什么console.log(0.2+0.1==0.3) //false

50.说一下JS中类型转换的规则?

51.深拷贝和浅拷贝的区别？如何实现

52.如何判断this？箭头函数的this是什么

53.== 和 ===的区别

54.什么是闭包

55.JavaScript原型，原型链？有什么特点?

56.typeof()和instanceof()的用法区别

57.什么是变量提升

58.all、apply以及bind函数内部实现是怎么样的

59.为什么会出现setTimeout倒计时误差？如何减少

60.谈谈你对JS执行上下文栈和作用域链的理解

61.new的原理是什么？通过new的方式创建对象和通过字面量创建有什么区别?

62.prototype 和 proto 区别是什么?

63.使用ES5实现一个继承?

64.取数组的最大值（ES5、ES6）

65.ES6新的特性有哪些?

66.promise 有几种状态, Promise 有什么优缺点 ？

67.Promise构造函数是同步还是异步执行，then呢 ?promise如何实现then处理 ？

68.Promise和setTimeout的区别 ？

69.如何实现 Promise.all ？
70.如何实现 Promise.finally ？
71.如何判断img加载完成
72.如何阻止冒泡?
73.如何阻止默认事件?
74.ajax请求时，如何解释json数据
75.json和jsonp的区别?
76.如何用原生js给一个按钮绑定两个onclick事件?
77.拖拽会用到哪些事件
78.document.write和innerHTML的区别
79.jQuery的事件委托方法bind 、live、delegate、on之间有什么区别?
80.浏览器是如何渲染页面的?
81.$(document).ready()方法和window.onload有什么区别?
82. jquery中$.get()提交和$.post()提交有区别吗?
83.对前端路由的理解？前后端路由的区别?
84.手写一个类的继承
85.XMLHttpRequest：XMLHttpRequest.readyState;状态码的意思
86.正则表达式常见面试题

1.给一个连字符串例如：get-element-by-id转化成驼峰形式。
2.匹配二进制数字
3.非零的十进制数字 (有至少一位数字, 但是不能以0开头)
4.匹配一年中的12个月
5.匹配qq号最长为13为
6.匹配常见的固定电话号码
7.匹配ip地址
8.匹配用尖括号括起来的以a开头的字符串
9.分割数字每三个以一个逗号划分
10.判断字符串是否包含数字
11.判断电话号码
12.判断是否符合指定格式
13.判断是否符合USD格式
14.JS实现千位分隔符
15.获取 url 参数
16.验证邮箱
17.验证身份证号码
18.匹配汉字
19.去除首尾的'/'
20.判断日期格式是否符合 '2017-05-11'的形式，简单判断，只判断格式
21.判断日期格式是否符合 '2017-05-11'的形式，严格判断（比较复杂）
22.IPv4地址正则
23.十六进制颜色正则
24.车牌号正则
25.过滤HTML标签

26.密码强度正则，最少6位，包括至少1个大写字母，1个小写字母，1个数字，1个特殊字符
27.URL正则
28.匹配浮点数

浏览器/html/css面试题

1.什么是盒模型
2.行内元素有哪些？块级元素有哪些？ 空(void)元素有那些？行内元素和块级元素有什么区别?
3.简述src和href的区别
4.什么是css Hack
5.什么叫优雅降级和渐进增强
6.px和em的区别
7.HTML5 为什么只写
8.Http的状态码有哪些
9.一次完整的HTTP事务是怎么一个过程
10.HTTPS是如何实现加密
11.浏览器是如何渲染页面的
12.浏览器的内核有哪些？分别有什么代表的浏览器
13.页面导入时，使用link和@import有什么区别
14.如何优化图像，图像格式的区别
15.列举你了解Html5. Css3 新特性
16.可以通过哪些方法优化css3 animation渲染
17.列举几个前端性能方面的优化
18.如何实现同一个浏览器多个标签页之间的通信
19.浏览器的存储技术有哪些
20.css定位方式
21.尽可能多的写出浏览器兼容性问题
22.垂直上下居中的方法
23.响应式布局原理
25.清除浮动的方法
26.http协议和tcp协议
27.刷新页面，js请求一般会有哪些地方有缓存处理
28.如何对网站的文件和资源进行优化
29.你对网页标准和W3C重要性的理解
30.Http和https的区别
31.data-属性的作用
32.如何让Chrome浏览器显示小于12px的文字
33.哪些操作会引起页面回流（Reflow）
34.CSS预处理器的比较less sass
35.如何实现页面每次打开时清除本页缓存
36.什么是Virtual DOM,为何要用Virtual DOM
37.伪元素和伪类的区别
38.http的几种请求方法和区别

39.前端需要注意哪些SEO
40.的title和alt有什么区别
41.从浏览器地址栏输入url到显示页面的步骤
42.如何进行网站性能优化
43.语义化的理解
44.HTML5的离线储存怎么使用，工作原理能不能解释一下?
45.浏览器是怎么对HTML5的离线储存资源进行管理和加载的呢
46.iframe有那些缺点?
47.WEB标准以及W3C标准是什么?
48.Doctype作用? 严格模式与混杂模式如何区分？它们有何意义?
49.HTML全局属性(global attribute)有哪些
50.Canvas和SVG有什么区别?
51.如何在页面上实现一个圆形的可点击区域?
52.网页验证码是干嘛的，是为了解决什么安全问题
53.请描述一下 cookies，sessionStorage 和 localStorage 的区别?
54. CSS选择器有哪些？哪些属性可以继承?
55.CSS优先级算法如何计算?
56.CSS3有哪些新特性?
57.请解释一下CSS3的flexbox（弹性盒布局模型）,以及适用场景?
58.用纯CSS创建一个三角形的原理是什么?
59.常见的兼容性问题?
60.为什么要初始化CSS样式
61.absolute的containing block计算方式跟正常流有什么不同?
62.CSS里的visibility属性有个collapse属性值？在不同浏览器下以后什么区别?
63.display:none与visibility：hidden的区别?
64.position跟display、overflow、float这些特性相互叠加后会怎么样?
65.对BFC规范(块级格式化上下文：block formatting context)的理解?
66.为什么会出现浮动和什么时候需要清除浮动？清除浮动的方式?
67.上下margin重合的问题
68. 设置元素浮动后，该元素的display值是多少?
69.移动端的布局用过媒体查询吗?
70.CSS优化、提高性能的方法有哪些?
71.浏览器是怎样解析CSS选择器的?
72.在网页中的应该使用奇数还是偶数的字体？为什么呢?
73.margin和padding分别适合什么场景使用?
74.元素竖向的百分比设定是相对于容器的高度吗?
75.全屏滚动的原理是什么？用到了CSS的哪些属性?
76.什么是响应式设计？响应式设计的基本原理是什么？如何兼容低版本的IE?
77. 视差滚动效果?
78.::before 和 :after中双冒号和单冒号有什么区别？解释一下这2个伪元素的作用
79.让页面里的字体变清晰，变细用CSS怎么做?
80. position:fixed;在android下无效怎么处理?
81.如果需要手动写动画，你认为最小时间间隔是多久，为什么?
82.li与li之间有看不见的空白间隔是什么原因引起的？有什么解决办法?

83.display:inline-block 什么时候会显示间隙?
84. 有一个高度自适应的div，里面有两个div，一个高度100px，希望另一个填满剩下的高度
85.png、jpg、gif 这些图片格式解释一下，分别什么时候用。有没有了解过webp?
86.style标签写在body后与body前有什么区别?
87.CSS属性overflow属性定义溢出元素内容区的内容会如何处理?
88.阐述一下CSS Sprites
89. 一行或多行文本超出隐藏

微信小程序开发（持续更新）

初识小程序

1.注册小程序
2.微信开发者工具
3.小程序与普通网页开发的区别
4.小程序尺寸单位rpx
5.样式导入（WeUI for）
6.选择器
7.小程序image高度自适应及裁剪问题
8.微信小程序长按识别二维码
9.给页面加背景色
10.微信小程序获取用户信息
11.代码审核和发布
12.小程序微信认证
13.小程序申请微信支付
14.小程序的目录解构及四种文件类型
15.小程序文件的作用域
16.小程序常用组件
    1.view
    2.scroll-view
    3.swiper组件
    4.movable-view
    5.cover-view
    6.cover-image

小程序基础

17.授权得到用户信息
18.数据绑定
19.列表渲染
20.条件渲染
21.公共模板建立

22.事件及事件绑定
23.引用
24.页面跳转
    1.wx.switchTab
    2.wx.reLaunch
    3.wx.redirectTo
    4.wx.navigateTo
    5.wx.navigateBack
25.设置tabBar
26.页面生命周期
27.转发分享

小程序高级

28.request请求后台接口
29.http-promise 封装
30.webview
31.获取用户收货地址
32.获取地里位置
33.自定义组件
34.微信小程序支付问题

小程序项目实战

35.微信小程序本地数据缓存
36.下拉刷新和下拉加载
37.列表页向详情页跳转（动态修改title）
38.客服电话
39.星级评分组件
40.小程序插槽的使用slot
41.模糊查询
42.wxs过滤
43.小程序动画
44.列表根据索引值渲染
45.小程序动态修改class
46.小程序常用框架
47.参数传值的方法
48.提高小程序的应用速度
49.微信小程序的优劣势
50.小程序的双向绑定和vue的区别
51.微信小程序给按钮添加动画
52.微信小程序的tab按钮的转换
53.微信小程序引进echarts
54.APP打开小程序流程

55.小程序解析富文本编辑器

小程序常见bug

1.域名必须是HTTPS
2. input组件placeholder字体颜色
3. wx.navigateTo无法跳转到带tabbar的页面
4. tabbar在切换时页面数据无法刷新
5.如何去掉自定义button灰色的圆角边框
6.input textarea是APP的原生组件，z-index层级最高
7.一段文字如何换行
8.设置最外层标签的margin-bottom在IOS下不生效
9.小程序中canvas的图片不支持base64格式
10.回到页面顶部
11.wx.setStorageSync和wx.getStorageSync报错问题
12.如何获取微信群名称?
13.new Date跨平台兼容性问题
14.wx.getSystemInfoSync获取windowHeight不准确
15.图片本地资源名称，尽量使用小写命名

移动端热点问题

1. 1px border问题
2.2X图 3X图适配
3.图片在安卓上，有些设备模糊问题
4.固定定位布局 键盘挡住输入框内容
5.click的300ms延迟问题和点击穿透问题
6.phone及ipad下输入框默认内阴影
7.防止手机中页面放大和缩小
scale=1
8.flex布局
9.px、em、rem、%、vw、vh、vm这些单位的区别
10. 移动端适配- dpr浅析
11.移动端扩展点击区域
12 上下拉动滚动条时卡顿、慢
13 长时间按住页面出现闪退
14. ios和android下触摸元素时出现半透明灰色遮罩
15. active兼容处理 即 伪类：active失效
16.webkit mask兼容处理
17. pc端与移动端字体大小的问题
18. transiton闪屏
19.圆角bug
20.如何解决禁用表单后移动端样式不统一问题?

js常用插件

轮播图插件
二级城市插件
三级城市插件
文字滑动效果
手风琴效果
视频播放插件
弹层插件
百度编辑器
ACE编辑器（轻巧）
上传图片（裁剪）
页面加载效果
全选反选各种效果
京东楼层效果
懒加载

vue项目首页加载速度优化

凡是做SPA的项目，特别是移动端的SAP项目，首屏加载速度必定是一个绕不过去的话题。接下来我就我们项目里的一些实践来做一下总结。希望抛砖引玉，如果各位有更好的方案，不吝赐教。

1: 针对第三方js库的优化

我们项目里用到的第三方js库主要有：vue, vue-router, vuex, axio, 我们还用到了qiniu。大家知道这些依赖库的js文件都会被一起打包到vender那个js文件里面，如果这些你的第三方依赖库很多，很大的话，那就会导致vender这个文件很大，那首屏加载的速度肯定会被拖慢。

针对这个问题我们的解决方案是，用文档的cdn文件代替，而不用打包到vender里面去。具体的做法是：

1: 在index.html里面引入依赖库js文件

```
 // index.html
<script src="https://cdn.bootcss.com/vue/2.3.3/
vue.min.js"></script>
<script src="https://cdn.bootcss.com/axios/0.16.2/
axios.min.js"></script>
```

2: 去掉第三方js的import,因为在第一步已经通过script标签引用进来了。
3: 把第三方库的js文件从打包文件里去掉
这一步的做法就是利用webpack的externals。具体做法就是在`build/webpack.base.conf.js`文件的module里面与rules同层加入externals：

```
module.exports = {
    entry: {...},
    output: {"path": config.build.assetsRoot...},
    resolve: {...},
    module: {
        rules: [...]
    },
    externals: {
        'vue': 'Vue',
        'vue-router': 'VueRouter',
        'vuex': 'Vuex',
        'axios': 'axios',
        'qiniu': 'Qiniu'
    }
};
```

2: 利用vue-router进行页面的懒加载（lazy load）
这里的页面的懒加载是指，假如我现在访问A页面，只会去请求A页面里的东西，其他页面的东西不会去请求。
具体怎么做，vue-router的官网都写得很清楚了,有需要的去看一下就懂了：
通过vue-router实现页面的懒加载
3: 图片资源的压缩
严格说来这一步不算在编码技术范围内，但是却对页面的加载速度影响很大，特别是对于移动端的项目来说。
对于非logo的图片文件，让UI设计师提供jpg格式的，不要用png.
对于所有的图片文件，都可以在一个叫tinypng的网站上去压缩一下。网址：
https://tinypng.com/
成果总结：
通过以上的那些方案，再加上我们的代码本来就是部署在CDN上面的，我们在移动端（微信，QQ，浏览器等）基本都能达到秒开的效果。