**### vue 的生命周期**

1.vue 的实例 和组件 组件和实例都有生命周期 而且是一样的

生命周期

组件的生命阶段 创建（猥琐发育别浪） 挂载（出生） 更新（换装备） 销毁（88 弥补遗憾）

组件到达某一个阶段会自动触发某一些函数 这函数就叫做生命周期的钩子函数 1（错误处理） 2（页面缓存） 8

创建

组件创建的时候触发

beforeCreate

created 组件刚创建的数据请求

挂载

创建完场挂载前后触发

beforeMount

mounted dom的初始化操作

更新

数据发生改变的时候触发

beforeUpdate

updated 数据的变化监听 尽量不要做数据修改 死循环

销毁

1.尸骨无存 控制组件渲染 v-if

2.有尸体 通过 vm.$destroy()

beforeDestroy

destroyed 擦屁股的操作 计时器 事件监听

1.将swiper 变成网络请求的

2.a.一个组件 组件有一个div 随着滚轮的滚动div随机变颜色 b、创建一个按钮控制该组件的销毁

**### 异步请求渲染问题**

1.原因：异步请求事件 远远大于生命周期的执行时间 mounted 初始化dom 是数据未返回 渲染数据是空数组

导致轮播图的容器层宽度为0 渲染轮博图不能滚动

2.数据返回之后再做初始化操作

3.延时初始化 nextTick(()=>{}) 上一次数据更新 （赋值）-》页面更新（dom更新）结束后执行

**### ref 获取dom元素**

ref绑定一个元素

this.$refs

**### filter 过滤数据**

按照需求对数据进行处理

全局 局部

Vue.filter('name',(data,params1,params2...)=>{ 数据处理 return 处理完的数据})

{{要过滤的数据|过滤器名(...)}}

{

data:

methods:

components:

filters:{

name(){

数据处理

return

}

}

}

**### computed 计算属性**

1.和数据进行关联 简化书写操作

2.缓存性 关联的属性发生变化之后会重新计算

**### watch 监听**

监听数据的变化

{

data(){

},

methods:{},

watch:{

name(newValue,oldValue){ }

}

}

**#### 插槽**

你需要在组件标签里写东西

slot 在组件内部开辟一片空间 来存放标签内部的内容

命名插槽 <p slot='hehe'></p> <slot name='hehe'>

**#### 响应式原理**

vue中数据发生变化 视图不更新 ？

data Object.definePropty get/set watcher dom

vue 中data 数据在生命周期被初始化的时候 ，会Object.definePropty进行处理

data 里的数据会被赋予 get 和set 正常情况下 数据获取会触发get 修改数据会触发set

修改数据的时候 优先触发set 触发watcher监听 在通知更新界面

vue中如果动态添加属性 ，该属性没有经过处理 就没有get。set 所以数据变页面不变

1.不动态添加属性

2.this.$set(目标对象,'动态添加的属性名','属性值')

**#### vue-cli（vue的脚手架工具）**

前端自动化构建 webpack（前端打包工具）

2018 vue-cli 2.9.6

2019 vue/cli 3.7.0 //操作简单

npm install -g @vue/cli 安装vue-cli脚手架工具

vue --version (vue -V) 查看脚手架工具的版本

vue create hehe 创建一个项目

预处理语言 less sass stylus

eslint

let name = 123

文件的目录结构

node\_modules 安装的插件

public 公有目录

index.html 网页的入口文件

src 源码目录

assets 资源目录

components 组件

xxx.vue 单文件组件 一个 xx.vue 文件就是一个组件

main.js js的入口文件

常用指令

npm install 按照 pageage.json

npm run serve 本地环境下运行

npm run build 线上的编译打包

脚手架项目的使用

index.html 入口的html文件 提供实例的挂载点

main.js 入口的js文件 创建一个实例 渲染一个根组件 将根组件挂载到渲染元素上

xxxx.vue 单文件组件 一个个组件

template

js

style

使用组件 1.创建组件 创建xx.vue 文件 里面写3部分

2.抛出组件 export default { }

3.哪里用哪里引入 import 组件名 from ‘路径’

4.将引入的组件注册到components

components:{组件名}

5.将组件名当成标签名使用

项目： http://ustbhuangyi.com/music/#/rank

spa 单页面应用

页面跳转时组件的切换

优点 ：跳转流畅 组件化开发 组件可以复用 开发便捷

缺点 ：1.首屏加载过慢 路由懒加载

2.seo 优化不好 服务渲染

mpa 多页面应用

跳转新的页面

缺点 ：跳转不流畅

优点 ：seo 优化好

项目

100页面 100k

单页面 100组件 10m

多页面 100页面

<meta name="keywords" content="lol ，王者，守望先锋，。。。"/>

http://www.xiaochi.com 5 500

**#### 移动端项目**

移动 spa

pc mpa 门户网站 官网

1.运行之后less 预处理语言报错 less插件版本不匹配 将less版本变成2.7.3

less

变量 @color:red;

嵌套 根据页面的嵌套关系进行嵌套

mixin 封装一个函数

.hehe(@params,...){

withd:100px;

background:@params;

}

.hehe(red)

计算

数量之间的计算

**### 移动端适配**

移动端:rem+flex

pc:百分比布局栅格媒体查询

主流的设计分辨率 ：375*\*667（物理分辨率） （750\**1134）（逻辑分辨率）

1.产品原型图 ui设计图

适配方式： 将屏幕的宽度变成10rem

**### 真机测试**

1.将项目在服务器端打开

2.将项目的url生成二维码

3.手机扫码测试

**### vue项目配置**

**#### alias 起别名**

给一个绝对路径起一个名字

/apple/desktop/vue/1911music/node\_modules/swiper/dist/css/swiper.min.css

/apple/desktop/vue/1911music/node\_modules = @

@/swiper/dist/css/swiper.min.css

在根目录下放一个文件 vue.config.js 功能配置目录

注意：style标签里使用别名需要 在别名前方加一个~ @import ‘~style/index.less’

**### 获取网络数据**

jsonp

使用jsonp 插件

import jsonp from 'jsonp'

jsonp(url,{param:'传给后端的回调key值'},(err,data)=>{})

ajax axios

跨域问题 cors jsonp 代理

代理的原理 服务器请求没有跨域问题

1.ajax请求自己服务器

2.自己服务器转发地方服务器

3.获取数据结果进行返回

/hehe/music/api/getDiscList?g\_tk=1928093487&inCharset=utf-8&outCharset=utf-8&notice=0&format=json&platform=yqq&hostUin=0&sin=0&ein=29&sortId=5&needNewCode=0&categoryId=10000000&rnd=0.2297472346880829'

http://ustbhuangyi.com/hehe/music/api/getDiscList?g\_tk=1928093487&inC

作业

1.首页实现

2.点击实现切换到其他页面 动态组件 路由

3.better-scroll

**## 浏览器渲染流程**

1. 浏览器会将HTML解析成一个DOM树，DOM

树的构建过程是一个深度遍历过程：当前节

点的所有子节点都构建好后才会去构建当前

节点的下一个兄弟节点。

2. 将CSS解析成 CSS Rule Tree。

3. 根据DOM树和CSSOM来构造 Rendering Tree。注意：Rendering Tree 渲染树并不等同于 DOM 树，因为一些像Header或display:none的东西就没必要放在渲染树中了。

4. 有了RenderTree，浏览器已经能知道网页

中有哪些节点、各个节点的CSS定义以及他们

的从属关系。下一步操作称之为layout，顾

名思义就是计算出每个节点在屏幕中的位置。

5. 再下一步就是绘制，即遍历render树，并使用UI后端层绘制每个节点。

　注意：上述这个过程是逐步完成的，为了更好的用户体验，渲染引擎将会尽可能早的将内容呈现到屏幕上，并不会等到所有的html都解析完成之后再去构建和布局render树。它是解析完一部分内容就显示一部分内容，同时，可能还在通过网络下载其余内容

**#### 端口号问题**

protfinder 插件内部bug

http://note.youdao.com/noteshare?id=3ce54c8a404aa8575c3829b931445c7c

**#### 真机调试**

1.ip 局域网ip 192.168.1.102 localhost肯定不能用

2.用电脑测试 自己测 同位测试（防火墙禁止）

3.用手机测试 同一局域网 wx浏览器测试 尝试开启手机的开发者模式

**#### 组件切换**

**##### 动态组件**

任意标签名 添加 is ='组件名' 在当前位置渲染指定的组件

<div is='singer'></div>

推荐使用<component>

**##### 路由**

1.监听地址栏的变化

2.根据变化切换不同的组件

> 基本使用

1.安装路由

2.创建路由表

```

import Vue form 'vue'

import VueRouter from 'vue-router'

Vue.use(VueRouter)

const router =new VueRouter({

routes:[

{path:'ur路径',component:'关联组件'},

....

]

})

export default router

```

3. 路由注册到实例

import router from './router.js'

new Vue({

router:router

})

router-link 控制地址栏改变 当场a

to='要变的地址栏'

activeClass='激活状态的类名'

tag='div' 控制渲染元素 默认为a

router-view 开辟一片空间 根据地址栏的变化显示不同的组件

动态导航

1. 声明路由表 {path:'/user/:id/:hehe',component:'singer'}

2. 目标组件 通过this.$route.params 保存的动态导航数据

3. 跳转动态导航 动态导航有几级 路径添加 /user/hehe/

4. 跳转组件并且传递参数 可以考虑动态导航

编程式导航

window.location.href='http://www.,baidu.com'

window.histroy.go(1) go(-1) back() push() replace()

push 可以实现页面的跳转 a->b->c go back 返回或者前进 页面栈里的数据

replace 以实现页面的跳转 a->b->c 没有返回

声明式导航

<a href='http://www.,baidu.com'>

<router-linkt to='/singer'></router-linkt>

this.$router 负责路由的跳转 push replace go back forward

this.$route 接受数据 params query

this.$router.push({path:'/singer',query:{us:123,ps:123}})

this.$router.push({name:'hehe',params:{us:123,ps:123}})

命名视图

给router-view 起一个名字 和组件形成对应关系

在同时渲染不同的组件的时候用到命名视图

1.渲染几个组件 用几个router-view 通过name 属性 起名

2.路由表中 componet 变成components

{path:"/xxx',components:{

name1:component1,

name2:component2

.....

}}

嵌套路由

**#### 路由使用场景**

1.编程式导航

2.声明式导航 tab

3.路由传参 params query

4.动态路由

5.路由嵌套

6.命名路由

7.命名视图

8.重定向

9.模式

1.多组件共享状态(有一个变量 在所有组件都可用)

2.一个组件发生改变 其他组件自动变（必须按照规定的套路修改）

**#### 核心**

state 全局状态（全局变量 state里的数据在所有的组件都可以使用）

mutations 一个对象 存了一堆方法

vuex 中规定 想修改state 的值必须通过mutation 里的方法

commit 触发mutation里想触发的方法

commit 只有2个参数 commit('要触发的事件',传递的数据{name:'wangyi',age:16})

actions 处理异步

dispath 触发actions 在actions 做异步请求 在通过commit 触发mutation

getters 派生属性 （vuex里的计算属性）