# -----Vue生命周期的理解

vue从创建到销毁的这一系列过程.就叫做生命周期

然而生命周期的话分为八个阶段

创建前,创建后,挂载前,挂载后,更新前,更新后,销毁前,销毁后

分别是boreCreate.created,boremonut,monuted,boreupdate,updated.boredestroy,destroyed

# ----vue双向绑定的原理

vue双向绑定原理的话每个人理解都不一样 我个人的理解就是

简单来说就是采用数据劫持结合发布者订阅者模式

通过一个函数object.defineproperty(第烦破破儿提)

来监听各个属性的setter和getter方法,在数据变动时发布消息给订阅者,触发相应的监听回调

# ----父子组件传值

组件传值的话分为三种 分别是 子传父 父传子 还有兄弟之间的传值 除此之外路由也可以进行传值 vuex也可以

父传子的话子组件就是用props(破破死)方法来接收数据

子传父的话就是用$emit(a门特)来接收数据的

兄弟之间传值的话就是创建一个vue的实例.用这个vue实例来进行传递和接收数据的

# -----封装一个组件

第一 首先所有的组件都是基于挂载到vue.entend这个方法之上的

第二 在vue.component方法注册组件

第三 如果子组件需要数据 可以调用props中接收定义

如果需要子组件把数据传递给父组件 可以使用$emit方法

# ----v-if和v-show的区别 使用场景

v-if的话就是通过销毁或者重建dom元素来达到显示或者隐藏的

然后的话v-show的话就是通过控制c3的属性display none block来达到显示或者隐藏的

如果一个图片在页面上渲染需要多次频繁切换显示或者隐藏的话就用v-show

相反的话的话就用v-if

# ----v-for和v-if哪个优先级更高 为什么避免俩个在一起用

v-for的优先级比v-if的高

当vue处理指令的时候,v-for比v-if更具有优先级嘛,然后v-if移动到容器元素

不会重复遍历列表里每个值 .我们只要检查一次就行了

# ----v-for里面的key 这个key代表什么

vue里面的key的话代表标识的一个作用

当vue组件里有相同标签名元素切换的时候,就是需要通过这个key来让vue区分他们

要不然vue就会替换相同标签的内容

# ----vue-router有几种导航钩子有什么作用

比如再项目中 路由页面跳转时进行登录验证,路由跳转判断

三种 分别是

组件内的钩子(befoEerouteenter, beforeRouteUpdate，beforeRouteLeave)

然后的话还有路由独享钩子(在路由上直接定义beforeEnter)

全局导航钩子(router.beforeEach全局前置守卫), (router.beforeresolve 全局解析守卫) (router.afterEach全局后置守卫)

都有三个参数 to进入的路由对象 from 离开的路由对象 next 是个方法 相当于保安 需要验证 也可以进行拦截

# ----vue-router的原理

本质上来讲的话就是监听url的变化,然后匹配路由规则,显示相应的页面,并且不需要刷新页面

# ----vue-router怎么配置

第一步首先npm安装路由插件 npm i vue-router

第二步引入vue.use文件里面,然后导入路由 import vue-router form ‘vue-reter’

第三部 配置路由 首先创建组件然后引入组件, 定义路由 实例化路由

第四步 挂载到vue实例化对象上,

第五步 放在app.vue组件里面

# ----vue参数传值 params和query有什么区别

路由传值分为三种方式

携带路由的跳转

Params(po瑞安死) name 传参(显示参数) 又分为编程式和声明式, 声明式(router-link通过to属性实现),编程式(this.$router.push)

params(不显示参数) 也分为编程式和声明式 不过不同的是是通过name属性进行传参

query传参 path 也分为编程式和声明式 也是通过声明式组件router-link to属性实现,

不过需要在子路由提前配置好name属性 this.$router.query

区别:

第一 用query传参的时候跳转时要用path来配置 params跳转的时候用name

第二 params传参刷新路由时候参数会丢失 如果不想丢失那么就要在路由上配置好参数

Query传参是用路径传参的所以刷新也不会清空,参数会显示到路径上

# ----keep-alive作用是什么

主要就是保留组件的状态或者组件的重新渲染

假如一个组件切换的时候,保存一些组件的状态防止多次渲染,就可以使用keep-alive包裹需要保存的组件,

对于 keep-alive 组件来说，它拥有两个独有的生命周期钩子函数，分别为 activated 和 deactivated 。用 keep-alive 包裹的组件在切换时不会进行销毁，而是缓存到内存中并执行 deactivated 钩子函数，命中缓存渲染后会执行 actived 钩子函数。

# ----watch和computed的区别是什么

watch

watch的话是观察动作就是监听器

应用的话监听porps,#eimt或本组件的值然后执行异步操作

没有缓存性,页面重新渲染时值不变化也会执行,

compute

computed的话就是一个计算值

应用的话就收简化tempalte里面的插值表达式的计算和prpos和eimt的传值

具有缓存性,页面重新渲染值不会变化,然后计算属性会立即返回之前的计算结果,就没必要再次执行函数了

# ----什么是webpack

webpack的话就是一个打包工具 就是可以解析我们实际开发中用到的html.然后js啊

然后css.还有图片文件字体啥的,打包成多个浏览器可以识别的一个js文件

就是让开发过程中更高效嘛,

# ----vuex是什么

vuex的话就是一个状态管理仓库嘛 就是一个数据仓库,

就是我们实际开发中 我们用传统的组件的话通信会比较复杂 然后维护起来也不方便

这个时候就用vuex来管理这些数据 更方便 更高效

State: 存放数据的状态

Mutations 修改数据的状态

Getters 类似vue过滤属性 主要用来过滤一些数据,获取状态

Actions 异步更改数据状态

Modules 将state分成多个modules,便于管理

# ----什么是mvvm

mvvm的话就是一种设计思想,就是model-view-viewmodel的简写嘛

model的话代表数据模型 也可以在model中修改数据

view的话代表ui组件 它负责将数据模型转化成ui展现出来

viewmodel的话就是监听模型数据的改变和控制试图行为,简单来说就是一个同步view和model的一个对象连接model和view

# ----什么是mvc

mvc就是比较直观的架构模式,就是一种模式

mvc就是model view controller(肯戳了)的缩写

就是创建web引用程序的一个模式

然后的话model就是代表模型 表示应用程序核心(比如数据库记录列表之类的)

view的话就是视图嘛 显示数据的(比如数据库记录这些)

controller的话就是控制器 处理输入的(比如的写入数据库记录)

# ----mvvm和mvc的区别

mvvm实现了view和model的自动同步,也就是当model的属性改变时我们不用在手动操作dom元素来改变view的显示而是改变属性后对应view层显示会自动改变

然后的话mvc是单向通信然后的话mvvm是双向数据绑定

# ----数组的方法

push尾部添加

unshift头部添加 然后会返回数组的长度

pop(跑普)将数组最后一项移出并返回然后的话会改变原数组

shift将数组头部移出并返回

concat(康凯特)数组拼接 两个数组合并成一个

splice(死不来死)修改数组 然后想数组中添加内容 删除数组成员

join将数组函数成员倒序

# ----ES6promise有什么用?对他的理解

promise是异步编程的一种解决方案,就是解决异步回调的,就是我们通常说的回调地域

# ----ajxa和axios有什么区别

Jquery项目太大.单纯使用ajxa的话要引入整个jquey非常不合理 体积比较大

Axios的话比较轻,可以解决回调地域的问题,然后就是自动转换json数据

# ---- let const区别

共同点

let声明的是变量 const声明的是常量

let和const是块级作用域

let和const不可重复声明

let和const存在暂时性死区 只能先声明再使用,要不然会报错

不同点

let声明的变量值和类型都可以改变

const声明的常量不能改变 但是声明对象是可以改变的

# ----localstorage和sessionstorage有什么区别

存储时间长短不一样

localstorage浏览器关闭存储的数据还有

sessionstorage浏览器关闭的话储存的数据就没有了

储存的大小也不一样

localstorage存储的大小比sessionstorage要大 localstorage约20msessionstorage约4m

# ----js基本数据类型

undefind,null,Boolean,number,string

# ----js内置对象

Object 是 JavaScript 中所有对象的父对象

数据封装类对象：Object、Array、Boolean、Number 和 String

其他对象：Function、Arguments、Math、Date、RegExp、Error

# ----常见的兼容性问题

Png24位图片在ie6浏览器出现背景,解决方案做成png8

浏览器默认margin和padding不同 解决方案就是加一个全局的初始化(margin0,padding0)来同一

# ----什么是同步和异步

同步, 线程被阻塞,等待任务返回结果,

异步. 异步就是线程不会被阻塞,任务完成时通知js引擎

# ----变量提升

Js代码运行前都会进行AST(就是作用域)解析,函数默认会提到当前作用域最前面,变量声明也会进行提升,但赋值不会提升

在某一作用域中，声明变量的语句会默认解析为在该作用域的最开始就已经声明了

# -----浅拷贝和深拷贝

浅拷贝的话就是只复制指向某个对象的指针 而不是复制对象本身

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

# ----什么是面向过程,什么是面向对象

面向过程就是分析出决绝问题所需要的步骤,然后用函数一步步的实现,使用的时候一个个调用这就是面向过程

面向对象是把问题分解成各个对象,建立对象的目的不是为了完成一个步骤,而是为了描述某个事物在整个解决问题

# ----Call,apply,bind有什么区别

首先他们都是为了改变this指向的,作用是一样的,只是传参的方式不同

假如我们生命一个变量 然后给个值, 然后通过一个函数分别传两个参数,然后在函数内部分别打印,然后我们用apply和call

这里的call可以接收一个参数的列表,而apply可以接收一个参数的数组

通过apply和call改变函数的this指向，他们两个函数的第一个参数都是一样的表示要改变指向的那个对象，第二个参数，apply是数组，而call则是arg1,arg2...这种形式。

通过bind改变this作用域会返回一个新的函数，这个函数不会马上执行。

# ----跨域有几种方式,什么是josnp

Jsonp cors

Jsonp的原理很简单,就是利用sceipt标签没有跨域限制的漏洞,通过标签指向一个需要访问的地址并且提供一个回调函数来接受数据

# ----什么是递归

通俗讲的话函数直接或者间接自己调用自己 这就是递归

# ----什么是闭包

闭包就是一个概念 每个人理解的方式不同

简单来说闭包就是一个作用域可以访问另一个函数的局部变量 这就是闭包

作用的话就是 延伸了变量作用范围

优点:防止全局变量污染,缺点,使用不当的话会产生内存泄漏

# ----防抖和节流

防抖和节流的作用都是防止函数多次调用

他俩的区别就是

假如一个用户一直触发这个函数,且每次触发函数的间隔小于wait

防抖只会调用一次,而节流的话会每隔一段时间调用

# ----什么是内存泄漏 哪些操作会造成内存泄漏

内存泄漏就是指任何对象不在使用或者需要他之后仍然存在 这就是内存泄漏

闭包

定时器(settimeout)的第一个参数使用字符串而不是函数的话就会引起内存泄漏

Dom清空时,还存在引用

# ----什么是原型链

通俗易懂的话来讲原型链就是一个链条

实例对象和原型之间连接

当我们创建一个实例化对象的时候内部都会有一个属性 就是这个prototype

当我们访问一个对象属性的时候 如果内部不存在这个属性那么他就会去prototype里面去寻找

这个属性.这个protorype又会有自己的protorype于是就这样找下去.也就是我们常说的原型链

# ----什么是作用域链and作用域的理解

作用域就是变量与函数的可访问范围

作用域链就是保证作用域链的作用是保证执行环境里有权访问的变量和函数是有序的，作用域链的变量只能向上访问，变量访问到window对象即被终止，作用域链向下访问变量是不被允许的

# ----Js如何实现继承

构造继承,原型继承,实例继承,拷贝继承

# ----Js异步加载有哪些

Async(额森可)

Defer(第fao)

# ----冒泡捕获

冒泡就是子元素出发了事件会影响父元素的事件

捕获的话就是父元素会影响子元素的事件

# ----一个页面从输入url到页面加载显示完成，这个过程都发生了什么？

三次握手 四次挥手

浏览器根据请求的URL交给DNS域名解析，找到真实IP，向服务器发起请求；

服务器交给后台处理完成后返回数据，浏览器接收文件（HTML、JS、CSS、图象等）；

浏览器对加载到的资源进行解析，建立相应的内部数据结构

载入解析到的资源文件，渲染页面，完成

----------------------------------------

* 首先浏览器主进程接管，开了一个下载线程。
* 然后进行HTTP请求（DNS查询、IP寻址等等），中间会有三次捂手，等待响应，开始下载响应报文。
* 将下载完的内容转交给Renderer进程管理。
* Renderer进程开始解析css rule tree和dom tree，这两个过程是并行的，所以一般我会把link标签放在页面顶部。
* 解析绘制过程中，当浏览器遇到link标签或者script、img等标签，浏览器会去下载这些内容，遇到时候缓存的使用缓存，不适用缓存的重新下载资源。
* css rule tree和dom tree生成完了之后，开始合成render tree，这个时候浏览器会进行layout，开始计算每一个节点的位置，然后进行绘制。
* 绘制结束后，关闭TCP连接，过程有四次挥手

。

# ----js性能优化

减少HTTP请求

减少nds查询

减少Dom节点数量

减少cookie 大小的传输 将静态文件 放到同一域名

将图片(字体,图标)优化 尽可能的使用png格式的图片 因为png格式的图片比gif图片体积小

优化精灵图

图片懒加载

将css样式表放在顶部

将javascript脚本放在底部

避免使用css表达式

使用外部javascript和css

# ----vue性能优化

减少es6转为es5的沉余代码

提取组件的css

浏览器缓存

v-for遍历的必须为item添加key 然后避免同时使用v-if

路由懒加载

# -----js兼容性问题

获取body方式

标准的获取的话 document.body就行了

火狐的话 document.document.elemnt

获取节点的兼容性问题

Document.getelementsByClassName()在ie8以下兼容性问题

建议使用 document.querySelector()

获取页面滚动的距离

比如top 和left

一个函数里面 return

阻止浏览器默认时间的兼容

阻止时间冒泡的兼容

Ajax的兼容

# ----Http请求有哪几种方式

Post 提交数据

Get 发送数据

Put 跟post很类似,也是提交数据,他跟post不同的是put指定了资源在服务器上的位置,get不会

Head 请求页面首部

Delete(迪利特)删除服务器的资源

# ----Px和em的区别

Px和em都是长度单位

区别是

px的值是固定的,指定多少就是多少,计算比较容易

em的值是不固定的,并且em会继承父级元素的字体大小

浏览器的字体都是16px 没调整之前1em=16px