！important>行内样式>id选择器>类选择器>标签>全局选择器

Display:none,

Opacity:0

Visibility:hidden

颜色文本阴影

元素水平垂直居中

定位+margin：auto

定位+Transform：translate

Flex布局 just item

Gird布局

Table布局：

.container {

display: table;

}

.content {

display: table-cell;

vertical-align: center;

}

Css三大特性 继承，层叠，样式

Css中那些属性可以继承：

1. 字体的一些属性
2. 文本的一些属性
3. 元素的可见性
4. 表格布局的属性
5. 列表的属性
6. 页面样式的属性
7. 声音的样式属性

Js有哪些内置对象：math num Boolean function array string map...

数组的常见方法：[JavaScript中数组的常用方法（含ES6） - 简书 (jianshu.com)](https://www.jianshu.com/p/18c9d6f0faa5)

基本数据类型和引用数据类型？

[JavaScript基本数据类型和引用数据类型详解 - 掘金 (juejin.cn)](https://juejin.cn/post/7196059552122273848)

堆和栈

事件代理

其中包含了事件冒泡与事件捕获原理

[浅谈事件冒泡和事件捕获 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/100831300" \l ":~:text=%E5%BE%AE%E8%BD%AF%E6%8F%90%E5%87%BA%E4%BA%86%E5%90%8D%E4%B8%BA%20%E4%BA%8B%E4%BB%B6%E5%86%92%E6%B3%A1%20%E7%9A%84%E4%BA%8B%E4%BB%B6%E6%B5%81%E3%80%82%20%E4%BA%8B%E4%BB%B6%E6%8C%89%E7%85%A7%E4%BB%8E%E6%9C%80%E7%89%B9%E5%AE%9A%E7%9A%84%E4%BA%8B%E4%BB%B6%E7%9B%AE%E6%A0%87%E5%88%B0%E6%9C%80%E4%B8%8D%E7%89%B9%E5%AE%9A%E7%9A%84%E4%BA%8B%E4%BB%B6%E7%9B%AE%E6%A0%87%20%28document%E5%AF%B9%E8%B1%A1%29%E7%9A%84%E9%A1%BA%E5%BA%8F%E8%A7%A6%E5%8F%91%E3%80%82,%E5%8F%AF%E4%BB%A5%E6%83%B3%E8%B1%A1%E6%8A%8A%E4%B8%80%E9%A2%97%E7%9F%B3%E5%A4%B4%E6%8A%95%E5%85%A5%E6%B0%B4%E4%B8%AD%EF%BC%8C%E6%B3%A1%E6%B3%A1%E4%BC%9A%E4%B8%80%E7%9B%B4%E4%BB%8E%E6%B0%B4%E5%BA%95%E5%86%92%E5%87%BA%E6%B0%B4%E9%9D%A2%E3%80%82%20%E4%B9%9F%E5%B0%B1%E6%98%AF%E8%AF%B4%EF%BC%8C%E4%BA%8B%E4%BB%B6%E4%BC%9A%E4%BB%8E%E6%9C%80%E5%86%85%E5%B1%82%E7%9A%84%E5%85%83%E7%B4%A0%E5%BC%80%E5%A7%8B%E5%8F%91%E7%94%9F%EF%BC%8C%E4%B8%80%E7%9B%B4%E5%90%91%E4%B8%8A%E4%BC%A0%E6%92%AD%EF%BC%8C%E7%9B%B4%E5%88%B0document%E5%AF%B9%E8%B1%A1%E3%80%82%20%E9%92%88%E5%AF%B9%E4%BA%8E%E5%88%9A%E6%89%8D%E7%9A%84%E4%BE%8B%E5%AD%90%EF%BC%8C%E4%B8%89%E4%B8%AA%E5%85%83%E7%B4%A0%E9%97%B4%E7%9A%84%E8%A7%A6%E5%8F%91%E9%A1%BA%E5%BA%8F%E5%B0%B1%E5%BA%94%E8%AF%A5%E6%98%AF%20dv3-%3Edv2-%3Edv1.%20%E8%BF%99%E5%B0%B1%E6%98%AF%E4%BA%8B%E4%BB%B6%E5%86%92%E6%B3%A1%E3%80%82)

原型链

原型就是一个普通对象，为构造函数的实例共享属性和方法

Prototype节约内存的问题

New操作符具体做了什么？

1. 创建一个新对象
2. 把空对象和构造函数通过原型链进行链接
3. 把构造函数的this绑定到新的空对象身上
4. 根据构建函数返回的类型判断，如果是值类型，则返回对象，，如果是引用类型，就要返回这个引用类型

Js如何继承

Es6继承、原型链继承、构造函数继承

Js中this指向问题

[(20条消息) JS中的this指向问题(详细版)\_js中this指向问题\_LoveyL0201的博客-CSDN博客](https://blog.csdn.net/chen_junfeng/article/details/109235442)

延迟加载：async、defer

例如：<script defer type="text/javascript" src='script.js'></script>

defer : 等html全部解析完成，才会执行js代码，顺次执行js脚本。

async : async是和html解析同步的（一起的），不是顺次执行js脚本（谁先加载完谁先执行）。

Settimeout 最小执行时间4ms与setinterval区别 最小执行时间10ms

[settimeout和setinterval之间有什么区别？-js教程-PHP中文网](https://www.php.cn/js-tutorial-463472.html" \l ":~:text=settimeout%E5%92%8Csetinterval%E7%9A%84%E5%8C%BA%E5%88%AB%201%E3%80%81setTimeout%E5%92%8CsetInterval%E9%83%BD%E5%B1%9E%E4%BA%8EJS%E4%B8%AD%E7%9A%84%E5%AE%9A%E6%97%B6%E5%99%A8%EF%BC%8C%E5%8F%AF%E4%BB%A5%E8%A7%84%E5%AE%9A%E5%BB%B6%E8%BF%9F%E6%97%B6%E9%97%B4%E5%86%8D%E6%89%A7%E8%A1%8C%E6%9F%90%E4%B8%AA%E6%93%8D%E4%BD%9C%EF%BC%8C%E4%B8%8D%E5%90%8C%E7%9A%84%E6%98%AFsetTimeout%E5%9C%A8%E8%A7%84%E5%AE%9A%E6%97%B6%E9%97%B4%E5%90%8E%E6%89%A7%E8%A1%8C%E5%AE%8C%E6%9F%90%E4%B8%AA%E6%93%8D%E4%BD%9C%E5%B0%B1%E5%81%9C%E6%AD%A2%E4%BA%86%EF%BC%9B%E8%80%8CsetInterval%E5%88%99%E5%8F%AF%E4%BB%A5%E4%B8%80%E7%9B%B4%E5%BE%AA%E7%8E%AF%E4%B8%8B%E5%8E%BB%EF%BC%8C%E8%A6%81%E6%83%B3%E5%81%9C%E6%AD%A2%EF%BC%8C%E5%8F%AF%E7%94%A8window.clearInterval,%28%29%3B%201)

es6新增特性

[ES6 有哪些新特性 ? 全网最详细 - 掘金 (juejin.cn)](https://juejin.cn/post/7199962962630082618)

Let const

解构赋值

箭头函数

函数默认值

Promise

Async

Generator

类的继承

call、apply、bind区别

call传的是一个参数列表

apply传的是一个数组

深拷贝

利用递归 扩展运算符 assign

[(20条消息) JavaScript深拷贝看这篇就行了！（实现完美的ES6+版本）\_js深拷贝\_码飞\_CC的博客-CSDN博客](https://blog.csdn.net/cc18868876837/article/details/114918262)

Js中宏任务和微任务

1. js是单线程的语言。

2. js代码执行流程：同步执行完==》事件循环

同步的任务都执行完了，才会执行事件循环的内容

进入事件循环：请求、定时器、事件....

3. 事件循环中包含：【微任务、宏任务】

微任务：promise.then

宏任务：setTimeout..

要执行宏任务的前提是清空了所有的微任务

流程：同步==》事件循环【微任务和宏任务】==》微任务==》宏任务=》微任务...

Ajax是什么？怎么实现的

创建交互式网页应用的网页开发技术

在不加载整个网页的前提下，与服务器交换数据并更新部分内容

通过XHR对象向服务器发送异步请求，然后从服务器拿到数据，最后通过js操作Dom更新页面

1. 创建xmlhttprequest对象
2. 通过xhr对象里的open()方法和服务器建立连接
3. 构建请求所需的数据，并通过xhr对象的send()方法发送给服务器
4. 通过xhr对象的onreadystate change事件监听服务器和你的通信状态
5. 接受并处理服务器响应的数据结果
6. 把处理的数据更新到html页面上

Get和post请求有什么区别

1. get一般是获取数据，post一般是提交数据
2. get参数会放在url上，所以安全性比较差，post放在body中
3. get请求刷新服务器或退回是没有影响的，post请求退回时会重新提交数据
4. get请求会被缓存，而post请求不会被缓存
5. get请求会被保存在浏览器历史记录中，post不会
6. get请求只能进行url编码，post请求支持很多种类

promise内部原理 以及优缺点

Promise对象，封装了一个异步操作并且还可以获取成功或失败的结果

Promise主要就是解决回调地狱的问题，之前如果异步任务比较多，同时他们之间有相互依赖的关系，就只能使用回调函数处理，这样就容易形成回调地狱，代码的可读性差，可维护性也很差

有三种状态: pending初始状态fulfilled成功状态rejected失败状态

状态改变只会有两种情况，

pending -> fulfilled; pending -> rejected一旦发生，状态就会凝固，不会再变首先就是我们无法取消promise，一旦创建它就会立即执行，不能中途取消

如果不设置回调，promise内部抛出的错误就无法反馈到外面

若当前处于pending状态时，无法得知目前在哪个阶段。

原理:;

构造一个Promise实例，实例需要传递函数的参数，这个函数有两个形参，分别都是函数类型，一个是resolve一个是reject

promise上还有then方法，这个方法就是来指定状态改变时的确定操作，resolve是执行第一个函数，reject是执行第二个函数

promise和async await的区别是什么?

1.都是处理异步请求的方式

2.promise是ES6，async await是ES7的语法

3.async await是基于promise实现的，他和promise都是非阻塞性的

优缺点:

1.promise是返回对象我们要用then，catch方法去处理和捕获异常，并且书写方式是链式，容易造成代码重叠，不好维护，async await是通过try catch进行捕获异常

2.async await最大的优点就是能让代码看起来像同步一样，只要遇到await就会立刻返回结果，然后再执行后面的操作promise.then()的方式返回，会出现请求还没返回，就执行了后面的操作

浏览器的存储方式有哪些?

1.cookies

H5标准前的本地存储方式兼容性好，请求头自带cookie

存储量小，资源浪费，使用麻烦（封装)

2.localstorage

H5加入的以键值对为标准的方式操作方便，永久存储，兼容性较好

保存值的类型被限定，浏览器在隐私模式下不可读取，不能被爬虫

3.sessionstorage

当前页面关闭后就会立刻清理，会话级别的存储方式

公共点：在客户端存放数据

区别：

1. 数据存放有效期

sessionStorage : 仅在当前浏览器窗口关闭之前有效。【关闭浏览器就没了】

localStorage : 始终有效，窗口或者浏览器关闭也一直保存，所以叫持久化存储。

cookie : 只在设置的cookie过期时间之前有效，即使窗口或者浏览器关闭也有效。

2. localStorage、sessionStorage不可以设置过期时间

cookie 有过期时间，可以设置过期（把时间调整到之前的时间，就过期了）

3. 存储大小的限制

cookie存储量不能超过4k

localStorage、sessionStorage不能超过5M

\*\*\*\*根据不同的浏览器存储的大小是不同的。

token存在sessionstorage还是loaclstorage?

token:验证身份的令牌，一般就是用户通过账号密码登录后，服务端把这些凭证通过加密等一系列操作后得到的字符串

1.存loaclstorage里，后期每次请求接口都需要把它当作一个字段传给后台

2.存cookie中，会自动发送，缺点就是不能跨域

如果存在localstorage中，容易被xss攻击，但是如果做好了对应的措施，那么是利大于弊

Token登录流程

1. 客户端用账号密码请求登录
2. 服务器收到请求后，需要验证账号密码
3. 验证成功后，服务器会签发一个token，把这个token发给客户端
4. 客户端收到token之后保存起来，可以放在cookie也可以放在localstorage
5. 客户端每次想服务器发送请求资源的时候，都需要携带这个token
6. 服务端收到请求，接着去验证客户端里的token，验证成功才会返回数据

页面渲染的过程是怎样的?

DNS解析

建立TCP连接

发送HTTP请求

服务器处理请求

渲染页面

浏览器会获取HTML和cSs的资源，然后把HTML解析成DOM树

再把css解析成cSSOM

把DOM和cSSOM合并为渲染树

布局

把渲染树的每个节点渲染到屏幕上(绘制)

断开Tcp连接

精灵图:把多张小图整合到一张大图上，利用定位的一些属性把小图显示在页面上，当访问页面可以减少请求，提高加载速度

base64:传输8Bit字节代码的编码方式，把原本二进制形式转为64个字符的单位，最后组成字符串

base64是会和html css一起下载到浏览器中，减少请求，减少跨域问题，但是一些低版本不支持，若base64体积比原图片大，不利于css的加载。

svg格式了解多少?

基于XML语法格式的图像格式，可缩放矢量图，其他图像是基于像素的，SVG是属于对图像形状的描述，本质是文本文件，体积小，并且不管放大多少倍都不会失真

1.sVG可直接插入页面中，成为DOM一部分，然后用jS或cSS进行操作

<svg></svg>

2.svG可作为文件被引入

<img src="pic.svg”/>

3.svG可以转为base64引入页面

了解过jwT吗?

JSON web Token通过JSON形式作为在web应用中的令牌，可以在各方之间安全的把信息作为JSON对象传输信息传输、授权

JwT的认证流程

1.前端把账号密码发送给后端的接口

⒉.后端核对账号密码成功后，把用户id等其他信息作为jWT 负载，把它和头部分别进行base64编码拼接后签名，形成jwT (token） 。

3.前端每日请求时都会把jwT放在HTTP请求头的Authorization字段内

4.后端检查是否存在，如果存在就验证jwT的有效性（签名是否正确，token是否过期)

5.验证通过后后端使用jwT中包含的用户信息进行其他的操作，并返回对应结果

简洁、包含性、因为Token是jsON加密的形式保存在客户端，所以jwT是跨语言的，原则上是任何web形式都支持。

Npm的底层环境是什么

Npm 是node的包管理和分发工具

Npm的组成：网站、注册表、命令行工具

HTTP协议规定的响应头和请求头有什么?

1.请求头信息:

Accept:浏览器告诉服务器所支持的数据类型

Host:浏览器告诉服务器我想访问服务器的哪台主机

Referer:浏览器告诉服务器我是从哪里来的（防盗链)

User-Agent:浏览器类型、版本信息

Date:浏览器告诉服务器我是什么时候访问的

connection:连接方式

cookie

X-Request-with:请求方式

2.响应头信息:

Location:这个就是告诉浏览器你要去找谁

Server:告诉浏览器服务器的类型

content-Type:告诉浏览器返回的数据类型

Refresh:控制了的定时刷新

说一下浏览器的缓存策略。

强缓存（本地缓存）、协商缓存（弱缓存)I

强缓:不发起请求，直接使用缓存里的内容，浏览器把jS，css，image等存到内存中，下次用户访问直接从内存中取，提高性能

协缓:需要像后台发请求，通过判断来决定是否使用协商缓存，如果请求内容没有变化，则返回304，浏览器就用缓存里的内容

强缓存的触发:

HTTP1.e:时间戳响应标头

HTTP1.1:cache-control响应标头

协商缓存触发:

HTTP1.0:请求头: if-modified-since响应头: last-modified

HTTP1.1:请求头:if-none-match 响应头:Etag

说一下什么是“同源策略”?

http:/l www.aaa.com :8080/index/vue.js

协议 子域名 主域名 端口号 资源

同源策略是浏览器的核心，如果没有这个策略就会遭受网络攻击

主要指的就是协议+域名+端口号三者一致，若其中一个不一样则不是同源，会产生跨域三个允许跨域加载资源的标签: img link script

跨域是可以发送请求，后端也会正常返回结果，只不过这个结果被浏览器拦截了

JSONP

CORS

Websocket

反向代理

节流和防抖

防抖是回城，避免重复触发

节流是技能，把频繁触发的事件减少，每隔一段时间内执行

什么是Json

Json是纯字符串形式的数据，他不提供任何方法，适合在网络中传输

Json数据存储在.json文件中，也可以把Json数据以字符串的形式保存在数据库、cookise中提供了

Json.parse() 接收服务器数据时一般是字符串。可以使用 Json.parse() 方法将数据转换为 [JavaScript](https://so.csdn.net/so/search?q=JavaScript&spm=1001.2101.3001.7020" \t "_blank) 对象。

Json.stringify()将某个对象转换成 Json[字符串](https://so.csdn.net/so/search?q=%E5%AD%97%E7%AC%A6%E4%B8%B2&spm=1001.2101.3001.7020" \t "_blank)形式

什么时候使用json:定义接口;序列化;生成token;配置文件package.json

1.语义化的理解。I

在写HTML页面结构时所用的标签有意义

头部用head主 体用main底部用foot...

怎么判断页面是否语义化了?

把CSS去掉，如果能够清晰的看出来页面结构，显示内容较为正常

为什么要选择语义化?

1.让HTML结构更加清晰明了

2.方便团队协作，利于开发

3.有利于爬虫和SEO

4.能够让浏览器更好的去解析代码

5.给用户带来良好的体验

H5C3有哪些新特性?

H5的新特性:

1.语义化的标签

2.新增音频视频

3.画布canvas

4.数据存储 localstorage sessionstorage

5.增加了表单控件email url search...

6.拖拽释放API

CSS3的新特性:

1.新增选择器:属性选择器、伪类选择器、伪元素选择器

2.增加了媒体查询

3.文字阴影

4.边框

5.盒子模型box-sizing

6.渐变

7.过度

8.自定义动画

9.背景的属性

10.2D和3D

v-if和v-show的区别?

都可以控制元素的显示和隐藏

1. v- show时控制元素的display值来让元素显示和隐藏; v-if 显示隐藏时把DOM元素整个添加和删除

2.v-if有一个局部编译/卸载的过程，切换这个过程中会适当的销毁和重建内部的事件监听和子组件;v-show只是简单的css切换

3.v-if才是真正的条件渲染; v- show从false变成true的时候不会触发组件的生命周期，v-if 会触发生命周期

4.v- if的切换效率比较低，v- show的效率比较高

如何理解MVVM

M:模型，对应的就是data的数据

V:视图，用户界面，DOM

VM:视图模型: Vue的实例对象，连接View和Mode1的桥梁

核心是提供对View和ViewModel的双向数据绑定，当数据改变的时候，ViewModel能监听到数据的变化，自动更新视图，当用户操作视图的时候，ViewModel 也可以监听到视图的变化，然后通知数据进行改动，这就实现了双向数据绑定

ViewModel通过双向绑定把View和Model连接起来，他们之间的同步是自动的，不需要认为干涉，所以我们只需要关注业务逻辑即可，不需要操作DOM，同时也不需要关注数据的状态问题，因为她是由MVVM统一管理.

v-for中的key值的作用是什么

key属性是DOM元素的唯一标识

作用：

1. 提高虚拟Dom的更新
2. 若不设置key，可能会触发一些bug
3. 为了触发过度效果

对vue生命周期的理解

组件从创建到销毁的过程就是生命周期

1. 创建

beforeCreate 在这个阶段属性和方法都不能用

Create 这里时实例创建完成之后，在这里完成了数据监测，可以使用数据，修改数据，不会触发update，也不会更新视图

1. 挂载

Beforemount 完成了模板的编译，虚拟Dom也完成了创建，即将渲染，修改数据，不会触发updated

Mount 把编译好的模板挂载到页面，这里可以发送异步请求也可以访问dom节点

1. 更新

beforeUpdate组件数据更新之前使用，数据是新的，页面上的数据时旧的，组件即将更新，准备渲染，可以改数据

updated render重新做了渲染，这时数据和页面都是新的，避免在此更新数据

1. 销毁

beforeDestroy实例销毁前，在这里实例还可以用，可以清楚定时器等等

destroyed组件已经被销毁了，全部都销毁

1. 使用了keep-alive时多出两个周期:

Activated 组件激活时

Deactivated 组件被销毁时

在created和mounted去请求数据，有什么区别?

created:在道染前调用，通常先初始化属性，然后做渲染

mounted:在模板渲染完成后，一般都是初始化页面后，在对元素节点进行操作

在这里请求数据可能会出现闪屏的问题，created里不会，一般用created比较多

请求的数据对DOM有影响，那么使用created

如果请求的数据对DOM无关，可以放在mounted

Vue中的修饰符有哪些？

1. 事件修饰符

. stop 组织冒泡

. prevent组织默认行为

. capture 内部元素触发的事件先在次处理

.self只有在event.target是当前元素时触发

.once事件只会触发- -次

.passive立即触发默认行为

. native 把当前元素作为原生标签看待

1. 按键修饰符

. keyup 键盘抬起

. keydown键盘按下

1. 系统修饰符

.ctrl

.alt

. meta

1. 鼠标修饰符

.1eft 鼠标左键

. right 鼠标右键

. middle 鼠标中键

1. 表单修改符

.1azy 等输入完之后再显示

.trim 删除内容前后的空格

. number 输入是数字或转为数字

Vue如何进行组件通信

[(20条消息) vue组件通信6种方式](https://blog.csdn.net/bb_xiaxia1998/article/details/127681992)

[总结（常问知识点）-CSDN博客](https://blog.csdn.net/bb_xiaxia1998/article/details/127681992)

Keep-alive的理解

[vue中keep-alive的理解 - 简书 (jianshu.com)](https://www.jianshu.com/p/4ded1696f61d)

axios怎么封装

下载，创建实例 接着封装请求响应拦截器，抛出，最后暴露

vue路由时怎么传参的?

params传参

this . $router . push({name: ' index' ,params:{id: item. id}})

this . $route. params. id

路由属性传参

this . $router . push( {name: ' /index/${item.id}'})

路由配置{ path: '/index:id' }

query传参(可以解决页面刷新参数丢失的问题)

this . $router . push({

name: ' index '

query:{id:item.id}

})

vue路由的hash模式和history模式有什么区别?

1. hash的路由地址上有#号，history模式没有

2.在做回车刷新的时候，hash模式会加载对应页面，history 会报错404

3. hash模式支持低版本浏览器，history不支持， 因为是H5新增的API

4.hash不会重新加载页面，单页面应用必备【】

5. history有历史记录，H5新增了pushstate和replaceState()去修改历史记录，并不会立刻发送请求

6.history需要后台配置

路由拦截是怎么实现的?

路由拦截axios拦截

需要在路由配置中添加一个字段，它是用于判断路由是否需要拦截

{

name: ' index',

path:' / index' ,

component: Index,

meta:{

requirtAuth:true

}

}

router . beforeEach( (to, from,next) => {

if(to.meta.requirtAuth){

if( store.satte.token ){

next( )

}else{

: })

Vue的动态路由？？

vuex刷新数据会丢失吗?怎么解决?

vuex肯定会重新获取数据，页面也会丢失数据

1.把数据直接保存在浏览器缓存里(cookie localstorage sessionstorage )

2.页面刷新的时候，再次请求数据，达到可以动态更新的方法

监听浏览器的刷新事件，在刷新前把数据保存到sessionstorage里，刷新后请求数据，请求到了用vuex，如果没有，那就用sess ionstorage里的数据

computed和watch的区别?

1. computed是计算属性，watch是监听，监听的是data中数据的变化

2. computed是支持缓存，依赖的属性值发生变化，计算属性才会重新计算，否则用缓存; watch不支持缓存

3. computed不支持异步，watch是可以异步操作

4. computed是第一次加载就监听，watch是不监听

5. computed函数中必须有return， watch不用

vuex在什么场景会去使用?属性有哪些?

state存储变量

Getters state的计算属性

mutations提交更新数据的方法

actions和mutations差不多，他是提交mutations来修改数据，可以包括异步操作

modules模块化vuex

使用场景:

用户的个人信息、购物车模块、订单模块

vue的双向数据绑定原理是什么?

通过数据劫持和发布订阅者模式来实现，同时利用object . defineProperty( )劫持各个属性的setter和getter,

在数据发生改变的时候发布消息给订阅者，触发对应的监听回调渲染视图，也就是说数据和视图时同步的，数据发生改变，视图跟着发生改变，视图改变，数据也会发生改变。

第一步:需要observer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模板解析指令，把模板中的变量替换成数据，然后初始化渲染视图，同时把每个指令对应的节点绑定上更

新函数，添加订阅者，如果数据变化，收到通知，更新视图

第三步: Watcher订阅者是Observer和Compile之间的通信桥梁，作用: .

1.在自身实例化的时候忘订阅器内添加自己

2.自身要有一个update()方法

3.等待属性变动时，调用自身的update方法，触发compile这种的回调

第四步: MVVM作为数据绑定的入口，整合了observer. compile 和watcher三者，通过observer来监听自己的数据变化，

通过compile解析模板指令，最后利用watcher把observer和compile联系起来，最终达到数据更新视图更新，视图更新数据更新的效果

了解diff算法和虚拟DOM吗?

虚拟DOM，描述元素和元素之间的关系，创建一个JS对象

如果组件内有响应的数据，数据发生改变的时候，render函数会生成一个新的虚拟DOM，这个新的虚拟DOM会和旧的虚拟DOM进行比对，找到需要修改的虚拟DOM内容，然后去对应的真实DOM中修改

diff算法就是虛拟DOM的比对时用的，返回一个patch对象，这个对象的作用就是存储两个节点不同的地方，最后用patch里记录的信息进行更新真实DOM

步骤:

1.JS对象表示真实的DOM结构，要生成一个虚拟DOM，再用虚拟DOM构建一个真实DOM树，渲染到页面

2.状态改变生成新的虚拟DOM跟旧的虚拟DOM进行比对，这个比对的过程就是diff算法，利用patch记录差异

3.把记录的差异用在第一个虚拟DOM生成的真实DOM上，视图就更新了。

[从了解到深入虚拟DOM和实现diff算法 - 掘金 (juejin.cn)](https://juejin.cn/post/6990582632270528525)

vue和jquery的区别是什么?

1.原理不同

vue就是数据绑定; jq是先获取dom再处理

2.着重点不同.

vue是数据驱动，jq是着重于页面

3.操作不同

4.未来发展不同

vuex的响应式处理。

vuex是vue的状态管理工具

vue中可以直接触发methods中的方法，vuex是不可以的。未来处理异步，当触发事件的时候，会通过dispatch来访问

actions中的方法，actions 中的commit会触发mutations中的方法从而修改state里的值，通过getter把 数据更新到视图

Vue . use(vuex)，调用install方法，通过applymixin(vue)在任意组件内执行this . $store就可以访问到store对象。

vuex的state是响应式的，借助的就是vue的data，把state 存到vue实例组件的data中

vue中遍历全局的方法有哪些?

1.普通遍历，对象. forEach( )

arr. forEach( function(item, index, arr){

console. log(item, index)

})

2.对元素统- -操作对象.map()

var newarr = arr . map( function(item){

return item+1

})

3.查找符合条件的元素对象. filter()

arr. filter(function( item){

if(item > 2){

return false

}else{

return true

})

4.查询符合条件的元素，返回索引对象.findindex()

arr. finindex(function(item){

if(item>1){

return true

}else{

return false

}

})

对象. every()遇到不符合的对象会停止

对象.some()找到符合条件的元素就享出

如何封装一个组件?

1.使用Vue . extend()创建一个组件

2.使用Vue . components ()方法注册组件

3.如果子组件需要数据，可以在props中接收定义

4.子组件修改好数据，要把数据传递给父组件，可以用emit()方法

原则:

把功能拆开

尽量让组件原子化，一个组件做一件事情

容器组件管数据，展示组件管视图

封装一个可复用的组件，

需要满足什么条件?

1.低耦合，组件之间的依赖越小越好

2.最好从父级传入信息，不要在公共组件中请求数据

3.传入的数据要进行校验

4.处理事件的方法写在父组件中

vue的过滤器怎么使用?

vue的特性，用来对文本进行格式化处理

使用它的两个地方，一个是插值表达式，一个是v-bind

分类:

1.全局过滤器

Vue . filter(' add' , function(v){

returnV<10?'0'+V:v

})

<div>{{33| add}}</div>

2.本地过滤器

和methods同级

filter:{

add: function(v){

returnV<10?'0'+v:V

}

git如何合并、拉取代码?

拉取代码git pull ' 仓库地址‘

查看状态git status

提交到本地缓存区git add .

提交本地仓库git commit -m '修改描述'

提交到远程仓库git push ' 仓库地址’ master

创建分支git branch -b xxx

合并分支git merge '合并分支的名字'

git如何解决冲突问题?

1.两个分支中修改了同一个文件

2.两个分支中修改了同一个文件的名字

1.解决:当前分支上，直接修改代码add commit

2.解决:在本地当前分支上，修改冲突代码add commit push|