Vue 原理

基于 mvvm模型，通过数据劫持及发布-订阅模式,触发watcher，修改虚拟dom，然后模版解析最后绘制成页面；diff 虚拟dom

---

什么是MVVM

Model-View-ViewModel,Model代表数据模型，view代表ui组件，viewmodel，同步view和model，在框架之下，view与model无直接关系，通过viewmodel交互，通过双向绑定数据将view和model连接起来，只需要关注业务，不需要手动操作dom

---

Vue优点

低耦合，可服用，专注逻辑与数据开发，渐进式

组件之间传值

props $emit bus new Vue一个新的实例 $emit $on vuex 单向数据流 $attrs/$listeners ==> 仅仅是传递数据，而不做中间处理 location 本地缓存 provide/inject ==> 提供与注入 $parent/$children 与 ref

总结

1. 父子通信： props/$emit; 父链/子链($parent/$children); ref访问组件实例; provide/inject; $attrs/$listeners (api)
2. 兄弟通信： bus; vuex
3. 跨级通信： bus; vuex; provide/inject; $attrs/$listeners (api)

---

vue-router

几种导航钩子(导航守卫)

1. 全局路由 beforeEach
2. 全局解析守卫 beforeResolve
3. 全局后置钩子 afterEach
4. 路由独享守卫 路由配置上直接定义beforeEnter守卫
5. 组件内守卫 beforeRouteEnter，beforeRouteUpdate，beforeRouteLeave

> 完整的导航解析流程
>
> 1. 导航被触发；
> 2. 在失活的组件里调用beforeRouteLeave
> 3. 调用全局守卫
> 4. 在重用的组件里调用 beforeRouteUpdate 守卫
> 5. 在路由配置里调用 beforeEnter
> 6. 解析异步路由组件
> 7. 在被激活的组件里调用 beforeRouteEnter

> 8. 调用全局的 beforeResolve
> 9. 导航被确认
> 10. 调用全局的 afterEach 钩子
> 11. 触发 DOM 更新
> 12. 调用 beforeRouteEnter 守卫中传给 next 的回调函数，创建好的组件实例会作为回调函数的参数传入

传参

1. 动态路由(页面刷新数据不丢失，存在url中)
2. 路由name匹配，通过params this.$route.params
3. url传参 this.$route.query.id

响应路由参数的变化

1. watch $route

跳转

1. push
2. router-link标签

hash模式和history模式

1. hash url#部分，监听并匹配，进行解析，加载不同的组件
2. history 利用了HTML5 History Interface 中新增的pushState()和replaceState()方法。需要后台支持

路由懒加载

1. 优化页面，按需加载；

组件封装

日常项目中，一般都会用的组件封装，采用组件化的思维去开发，复杂的页面抽成相对独立的模块； 可重用代码，组件复用，

过程

Vue.extend 方法也创建组件，用Vue.component注册组件； 在脚手架工具工具中，.vue文件就是一个组件，import导入就行；

插件

Vue.use 也可以使用插件封装，可以在任何地方使用，比较方便

---

vue中想要监听data中某个属性的方法

```
watch: {
 "obj.a":() {}
}
```

---

computed，watch，method执行的先后顺序

默认加载的时候先computed再watch，不执行methods；等触发某一事件后，则是：先methods再watch。

---

v-model实现原理

创建双向数据绑定，

1. 指令解析；
2. 绑定数据 并 订阅；
3. 监听 input；
4. 更新

观察者模式 和 发布/订阅模式

观察者模式中观察者和目标直接进行交互，而发布订阅模式中统一由调度中心进行处理，订阅者与发布者互不干扰，实现了结耦，还可以实现更细微的控制；权限控制等等

Vue优化

编码阶段

1. 尽量减少data中的数据，object.freeze()冻结对象
2. v-if v-show
3. v-for绑定事件使用事件代理
4. key保证唯一，diff效率
5. 使用懒加载，异步组件
6. 防抖，节流
7. 第三方模块按需导入
8. 长列表动态加载
9. 图片懒加载
10. 公用组件抽象

seo优化

1. 预渲染 ==> 构建阶段生成匹配预渲染路径的 html 文件
2. ssr

打包优化 ==> webpack

1. 压缩代码

2. 使用cdn加载第三方模块
3. sourceMap优化

用户体验

1. 使用骨架屏，组件懒加载
2. loading增加
3. 缓存优化，服务端开启 gzip压缩等；

计算属性和 watch 的区别

1. 计算属性，基于依赖的缓存值，当所依赖的值改变时，计算属性才会跟着改变，重新计算，一般多对一；
2. watch 一般 一对一或者一对多，当该变量变化时，会触发 watch 中的方法；

单向数据流

使用一个上传数据流和一个下传数据流进行双向数据通信，两个数据流之间相互独立。数据的安全性，可控

---

keep-alive

组件之间切换的时候，保持这些组件的状态，或避免重新渲染

---

vuex

应用的状态管理工具，采用集中式存储管理所有状态，提供统一的数据操作

原理

resetStoreVM new Vue; 通过全局注入store对象，来实现组件间的状态共享

核心

1. state 数据源；
2. getter
3. mutation(变化) 改版状态的唯一方法就是提交mutation; dispatch actions
4. action ==> Action 提交的是 mutation，而不是直接变更状态；可以包含任意异步操作；
5. module 复杂项目，将store分割成模块，每个模块独立

---

vue3

区别

1. 绑定原理改变，2.0 Object.defineProperty 3.0 Proxy;
    1. defineProperty 只能监听某个属性，需要递归；
    2. proxy可以监听数组，不用单独处理；

> 只要不是重新赋值一个新的数组对象，任何对数组内部的修改都不会触发setter方法

2. 默认进行懒观察
3. ts
4. 重写 vDom

为什么js操作dom的性能低

操作DOM会 导致 重排reflow 和 重绘repaint，重排会占用、消耗CPU; 重绘会占用、消耗GPU;

---

http 状态码

- 100 继续 客户打继续请求
- 101 协议切换，服务器应客户端升级协议的请求（Upgrade请求头）正在进行协议切换
- 200 请求成功。一般用于GET与POST请求
- 201 已创建。成功请求并创建了新的资源
- 202 已接受。已经接受请求，但未处理完成，对于请求的处理确实无保证的
- 203 非授权信息。请求成功。但返回的meta信息不在原始的服务器，而是一个副本
- 204 无内容。服务器成功处理，但未返回内容。在未更新网页的情况下，可确保浏览器继续显示当前文档
- 205 重置内容。服务器处理成功，用户终端（例如：浏览器）应重置文档视图。可通过此返回码清除浏览器的表单域
- 206 部分内容。服务器成功处理了部分GET请求

- 300 是一个用来表示重定向的响应状态码，表示该请求拥有多种可能的响应。用户代理或者用户自身应该从中选择一个。由于没有如何进行选择的标准方法，这个状态码极少使用。
- 301 永久重定向 最好是在应对GET或HEAD方法时使用301，其他情况使用308 来替代301
- 302 临时移动 最好是在应对GET或HEAD方法时使用301，其他情况使用307 来替代302
- 303 查看其它地址。与301类似。使用GET和POST请求查看
- 304 未修改。所请求的资源未修改，服务器返回此状态码时，不会返回任何资源。
- 305 使用代理。所请求的资源必须通过代理访问
- 307 当发送重定向请求的时候，307 状态码可以确保请求方法和消息主体不会发生变化 302
- 308 在重定向过程中，请求方法和消息主体不会发生改变 301

> 304 GET 或HEAD 或在请求中附带了头部信息： If-None-Match 或If-Modified-Since 200 响应会带有头部 Cache-Control, Content-Location, Date, ETag, Expires，和 Vary