09 - 说说你对虚拟 DOM 的理解?

分析

现有框架几乎都引入了虚拟 DOM 来对真实 DOM 进行抽象，也就是现在大家所熟知的 VNode 和 VDOM，那么为什么需要引入虚拟 DOM 呢？围绕这个疑问来解答即可！

思路

1. vdom是什么
2. 引入vdom的好处
3. vdom如何生成，又如何成为dom
4. 在后续的diff中的作用

回答范例

1. 虚拟dom顾名思义就是虚拟的dom对象，它本身就是一个 `JavaScript` 对象，只不过它是通过不同的属性去描述一个视图结构。
2. 通过引入vdom我们可以获得如下好处：

   将真实元素节点抽象成 VNode，有效减少直接操作 dom 次数，从而提高程序性能

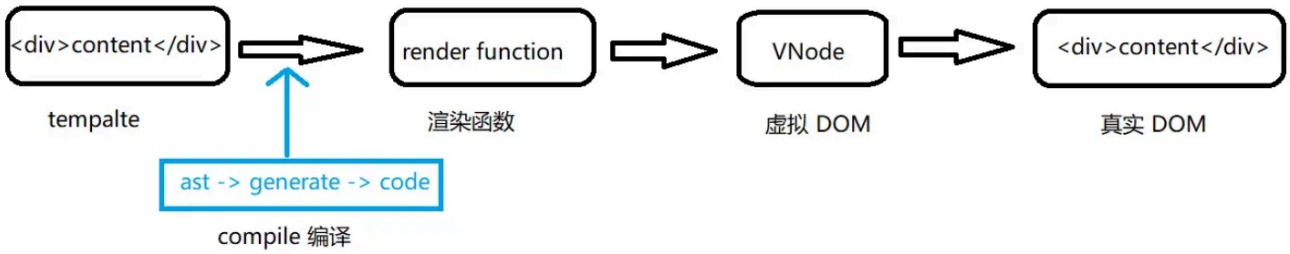
   - 直接操作 dom 是有限制的，比如：diff、clone 等操作，一个真实元素上有许多的内容，如果直接对其进行 diff 操作，会去额外 diff 一些没有必要的内容；同样的，如果需要进行 clone 那么需要将其全部内容进行复制，这也是没必要的。但是，如果将这些操作转移到 JavaScript 对象上，那么就会变得简单了。
   - 操作 dom 是比较昂贵的操作，频繁的dom操作容易引起页面的重绘和回流，但是通过抽象 VNode 进行中间处理，可以有效减少直接操作dom的次数，从而减少页面重绘和回流。

   方便实现跨平台

   - 同一 VNode 节点可以渲染成不同平台上的对应的内容，比如：渲染在浏览器是 dom 元素节点，渲染在 Native(iOS、Android) 变为对应的控件、可以实现 SSR 、渲染到 WebGL 中等等
   - Vue3 中允许开发者基于 VNode 实现自定义渲染器（renderer），以便于针对不同平台进行渲染。

---

3. vdom如何生成？在vue中我们常常会为组件编写模板 - template， 这个模板会被编译器 - compiler编译为渲染函数，在接下来的挂载（mount）过程中会调用render函数，返回的对象就是虚拟dom。但它们还不是真正的dom，所以会在后续的patch过程中进一步转化为dom。



4. 挂载过程结束后，vue程序进入更新流程。如果某些响应式数据发生变化，将会引起组件重新render，此时就会生成新的vdom，和上一次的渲染结果diff就能得到变化的地方，从而转换为最小量的dom操作，高效更新视图。

知其所以然

vnode定义：

https://github1s.com/vuejs/core/blob/HEAD/packages/runtime-core/src/vnode.ts#L127-L128

观察渲染函数：21-vdom/test-render-v3.html

创建vnode：

- createElementBlock:

https://github1s.com/vuejs/core/blob/HEAD/packages/runtime-core/src/vnode.ts#L291-L292

- createVnode:

https://github1s.com/vuejs/core/blob/HEAD/packages/runtime-core/src/vnode.ts#L486-L487

- 首次调用时刻：

https://github1s.com/vuejs/core/blob/HEAD/packages/runtime-core/src/apiCreateApp.ts#L283-L284

---

mount:

https://github1s.com/vuejs/core/blob/HEAD/packages/runtime-core/src/renderer.ts#L1171-L1172

调试mount过程：mountComponent

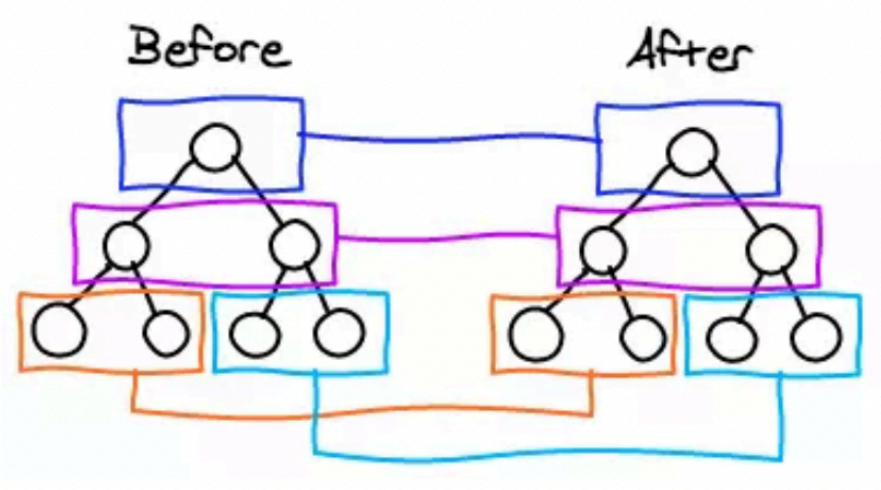
21-vdom/test-render-v3.html

---

10 - 你了解diff算法吗?

分析

必问题目，涉及vue更新原理，比较考查理解深度。



思路

1. diff算法是干什么的
2. 它的必要性
3. 它何时执行
4. 具体执行方式
5. 拔高：说一下vue3中的优化

回答范例

1.Vue中的diff算法称为patching算法，它由Snabbdom修改而来，虚拟DOM要想转化为真实DOM就需要通过patch方法转换。

2.最初Vue1.x视图中每个依赖均有更新函数对应，可以做到精准更新，因此并不需要虚拟DOM和patching算法支持，但是这样粒度过细导致Vue1.x无法承载较大应用；Vue 2.x中为了降低Watcher粒度，每个组件只有一个Watcher与之对应，此时就需要引入patching算法才能精确找到发生变化的地方并高效更新。

3.vue中diff执行的时刻是组件内响应式数据变更触发实例执行其更新函数时，更新函数会再次执行render函数获得最新的虚拟DOM，然后执行patch函数，并传入新旧两次虚拟DOM，通过比对两者找到变化的地方，最后将其转化为对应的DOM操作。

4.patch过程是一个递归过程，遵循深度优先、同层比较的策略；以vue3的patch为例：

- 首先判断两个节点是否为相同同类节点，不同则删除重新创建
- 如果双方都是文本则更新文本内容
- 如果双方都是元素节点则递归更新子元素，同时更新元素属性
- 更新子节点时又分了几种情况：
    - 新的子节点是文本，老的子节点是数组则清空，并设置文本；
    - 新的子节点是文本，老的子节点是文本则直接更新文本；
    - 新的子节点是数组，老的子节点是文本则清空文本，并创建新子节点数组中的子元素；
    - 新的子节点是数组，老的子节点也是数组，那么比较两组子节点，更新细节blabla

5. vue3中引入的更新策略：编译期优化patchFlags、block等

知其所以然

patch关键代码

https://github1s.com/vuejs/core/blob/HEAD/packages/runtime-core/src/renderer.ts#L354-L355

调试 test-v3.html

11 - 你知道哪些vue3新特性

分析

官网列举的最值得注意的新特性：https://v3-migration.vuejs.org/

- Composition API
- SFC Composition API Syntax Sugar (`<script setup>`)
- Teleport
- Fragments
- Emits Component Option
- `createRenderer` API from `@vue/runtime-core` to create custom renderers
- SFC State-driven CSS Variables (`v-bind` in `<style>`)
- SFC `<style scoped>` can now include global rules or rules that target only slotted content
- Suspense experimental

也就是下面这些：

- Composition API
- SFC Composition API语法糖
- Teleport传送门
- Fragments片段
- Emits选项
- 自定义渲染器
- SFC CSS变量
- Suspense

以上这些是api相关，另外还有很多框架特性也不能落掉。

回答范例

1. api层面Vue3新特性主要包括：Composition API、SFC Composition API语法糖、Teleport传送门、Fragments 片段、Emits选项、自定义渲染器、SFC CSS变量、Suspense
2. 另外，Vue3.0在框架层面也有很多亮眼的改进：

- 更快
  - 虚拟DOM重写
  - 编译器优化：静态提升、patchFlags、block等
  - 基于Proxy的响应式系统
- 更小：更好的摇树优化
- 更容易维护：TypeScript + 模块化
- 更容易扩展
  - 独立的响应化模块
  - 自定义渲染器

知其所以然

体验编译器优化

https://sfc.vuejs.org/

reactive实现

https://github1s.com/vuejs/core/blob/HEAD/packages/reactivity/src/reactive.ts#L90-L91
