vue面试题解析第二弹

内容概要

- 面试官意图分析
- 回答策略
- 经典题目实战

面试官意图分析

- 考查熟练度
- 考查深度
- 考查知识面

回答策略

- 结构化套路：3w1h，总分总
- 专业性：文档中的答案最权威，多结合实践应用
- 多做扩展：优缺点、应用场景、横向对比等
- 深入到源码中：升华你的回答

new Vue()都做了什么?

此问题考查大家对Vue初始化的了解程度，多数同学可能没有想过这个问题，是区分度很高的题目。

答题思路：总分总

1. 先给结论
2. 分述细节
3. 总结一波

回答范例：

1. new Vue()是创建Vue实例，它内部执行了根实例的初始化过程。
2. 具体包括以下操作：
    - 选项合并 用户选项、默认选项
    - $children, $refs, $slots, $createElement等实例属性和方法初始化
    - 自定义事件处理
    - 数据响应式处理
    - 生命周期钩子调用
    - 可能的挂载
3. 总结：new Vue()创建了根实例并准备好数据和方法，未来执行挂载时，此过程还会递归的应用于它的子组件上，最终形成一个有紧密关系的组件实例树。

知其所以然

测试代码：05-new Vue.html

可能的追问

new Vue得到的实例和组件实例有什么区别?

说一说你对vue响应式理解?

最可能被问的题，但却不是每个人都能回答到位。如果只是看看别人写的网文，通常没什么底气，也经不住面试官推敲，但像我们这样即看过源码还造过轮子的，回答这个问题就会比较有底气。

答题思路：3w1h

1. 啥是响应式?
2. 为什么vue需要响应式?
3. 它能给我们带来什么好处?
4. vue的响应式是怎么实现的？有哪些优缺点?
5. vue3中的响应式的新变化

回答范例：

1. 所谓数据响应式就是用户对数据层做的更改能够触发视图层做出更新响应的机制。
2. mvvm框架中要解决的一个核心问题是连接数据层和视图层，通过数据驱动应用，数据变化，视图更新，要做到这点的就需要实现一套响应式机制，这样一旦数据发生变化就可以立即做出更新处理。
3. 以vue为例说明，通过数据响应式加上虚拟DOM和patch算法，可以使我们只需要操作数据，完全不用接触繁琐的dom操作，从而大大提升开发效率，降低开发难度。
4. vue2中的数据响应式会根据数据类型来做不同处理，如果是对象则采用Object.defineProperty()的方式定义数据拦截，当数据被访问或发生变化时，我们感知并作出响应；如果是数组则通过覆盖该数组原型的方法，扩展它的7个变更方法，使这些方法可以额外的做更新通知，从而作出响应。这种机制很好的解决了数据响应化的问题，但在实际使用中也存在一些缺点：比如初始化时的递归遍历会造成性能损失；新增或删除属性时需要用户使用Vue.set/delete这样特殊的api才能生效；对于es6中新产生的Map、Set这些数据结构不支持等问题。
5. 为了解决这些问题，vue3重新编写了这一部分的实现：利用ES6的Proxy机制代理要响应化的数据，它有很多好处，编程体验是一致的，不需要使用特殊api，初始化性能和内存消耗都得到了大幅改善；另外由于响应化的实现代码抽取为独立的reactivity包，使得我们可以更灵活的使用它，我们甚至不需要引入vue都可以体验。

知其所以然

06-reactive.html

你知道nextTick吗?

这道题考查大家对vue异步更新队列的理解，有一定深度，如果能够很好回答此题，对面试效果有极大帮助。

答题思路：3w1h

1. nextTick是啥？下一个定义
2. 为什么需要它呢？用异步更新队列实现原理解释
3. 我再什么地方用它呢？抓抓头，想想你在平时开发中使用它的地方
4. 下面介绍一下如何使用nextTick
5. 最后能说出源码实现就会显得你格外优秀

先看看官方定义

> Vue.nextTick([callback, context])
>
> 在下次 DOM 更新循环结束之后执行延迟回调。在修改数据之后立即使用这个方法，获取更新后的 DOM。

```
// 修改数据
vm.msg = 'Hello'
// DOM 还没有更新
Vue.nextTick(function () {
// DOM 更新了
})
```

回答范例：

1. nextTick是Vue提供的一个全局API，由于vue的异步更新策略导致我们对数据的修改不会立刻体现在dom变化上，此时如果想要立即获取更新后的dom状态，就需要使用这个方法
2. Vue 在更新 DOM 时是**异步**执行的。只要侦听到数据变化，Vue 将开启一个队列，并缓冲在同一事件循环中发生的所有数据变更。如果同一个 watcher 被多次触发，只会被推入到队列中一次。这种在缓冲时去除重复数据对于避免不必要的计算和 DOM 操作是非常重要的。nextTick方法会在队列中加入一个回调函数，确保该函数在前面的dom操作完成后才调用。
3. 所以当我们想在修改数据后立即看到dom执行结果就需要用到nextTick方法。
4. 比如，我在干什么的时候就会使用nextTick，传一个回调函数进去，在里面执行dom操作即可。
5. 我也有简单了解nextTick实现，它会在callbacks里面加入我们传入的函数，然后用timerFunc异步方式调用它们，首选的异步方式会是Promise。这让我明白了为什么可以在nextTick中看到dom操作结果。

知其所以然

07-timerFunc.html

你知道key的作用吗?

分析：这是一道特别常见的问题，主要考查大家对虚拟DOM和patch细节的掌握程度，能够反映面试者理解层次。

思路分析：总分总模式

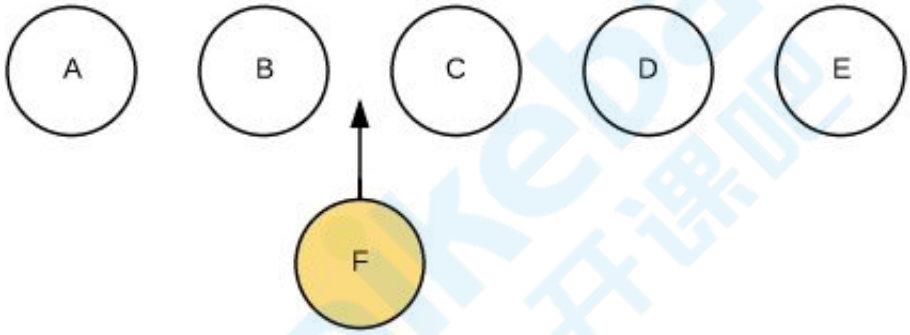
1. 给出结论，key的作用是用于优化patch性能
2. key的必要性
3. 实际使用方式
4. 总结：可从源码层面描述一下vue如何判断两个节点是否相同

回答范例：

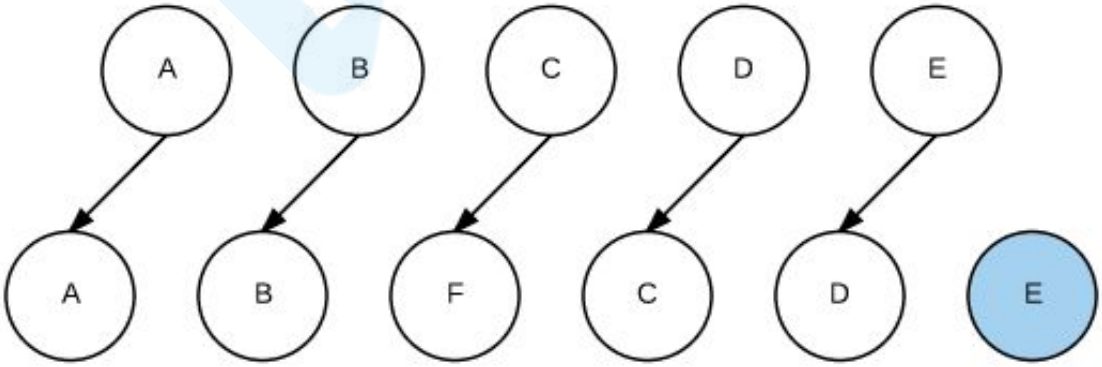
1. key的作用主要是为了更高效的对比虚拟DOM中的某个节点是否是相同节点。
2. vue在patch过程中**判断两个节点是否是相同节点是key是一个必要条件**，渲染一组列表时，key往往是唯一标识，所以**如果不定义key的话，vue只能认为比较的两个节点是同一个**，哪怕它们实际上不是，这导致了频繁更新元素，使得整个patch过程比较低效，影响性能。
3. 实际使用中在渲染一组列表时key必须设置，而且必须是唯一标识，应该避免使用数组索引作为key，这可能导致一些隐蔽的bug；vue中在使用相同标签元素过渡切换时，也会使用key属性，其目的也是为了让vue可以区分它们，否则vue只会替换其内部属性而不会触发过渡效果。
4. 从源码中可以知道，vue判断两个节点是否相同时主要判断两者的key和元素类型等，因此如果不设置key，它的值就是undefined，则可能永远认为这是两个相同节点，只能去做更新操作，这造成了大量的dom更新操作，明显是不可取的。

测试代码，02.html

上面案例重现的是以下过程



不使用key



如果使用key

```
// 首次循环patch A
A B C D E
A B F C D E

// 第2次循环patch B
```

```
B C D E
B F C D E

// 第3次循环patch E
C D E
F C D E

// 第4次循环patch D
C D
F C D

// 第5次循环patch C
C
F C

// oldCh全部处理结束，newCh中剩下的F，创建F并插入到C前面
```

源码中找答案：src\core\vdom\patch.js - sameVnode()

更多题目和解答

https://github.com/57code/vue-interview

欢迎来提问，有代表性的我会更新上去，欢迎star。