Vue 异步更新

dep.notify()

dep 通知更新

遍历 dep 收集的所有 watcher ， 让这些 watcher 依次去执行自己的 update 方法

// watcher.update()
如果是懒执行，比如 computed
将 dirty 置为 true ，组件更新后 ，当响应数据再次更新时，执行 computed getter 重新执行 computed 回调函数 ， 计算新值将新值缓存到 watcher.value

// this.watcher() 或者 watcher 选项时 ， 传递一个 sync 配置 比如 {.sync : true }
如果是同步执行的直接执行 run 方法

//watcher.run()

执行 watcher.get() 方法获取值
如果新值不等于旧值

将新值赋值给 watcher.value

如果是用户 watcher 执行用户 watcher 的回调

不是就执行本身的回调

// watcher.get()
评估getter，并重新收集依赖项。
触发 updateComponent 的执行，进行 组件更新，进入 patch 阶段
更新组件时 先执行 render 生成 vnode ，期间触发 读取操作，进行依赖收集

pushTarget(this)
对新值做依赖收集 ， observe setter 只是进行了响应式处理并没有进行依赖收集

value = this.getter.call(vm, vm)
执行实例化 watcher 时 传递进来的第二个参数，有可能是一个函数，实例化渲染 watcher 时传递的 updateComponent 函数
用户 watcher 可能传递是一个 key 也可能是读取 this.key 的函数
触发读取操作，被 setter 拦截，进行依赖收集

弹出当前 watcher
清除 deps

//如果是异步的
将 watcher 放入 watcher 队列 ， 一般都是走这个分支 queueWatcher(this)

获取当前 watcher id
判重保证同一个 watcher 不会重复入队
判断 flushing 是否为 false，即当前 watcher 队列是否在被刷新，如果为 false，即当前队列不在被刷新，直接将 watcher 入队
如果 flushing 为 true，即当前 watcher 队列正在被刷新，需要按 watcher 队列倒序遍历找到刚刚好大于当前 watcher id 的位置，将当前 watcher 插入到后一位
保证即使把当前 watcher 插入队列，该 watcher 队列依旧有序

//flushSchduleQueue
将 flushing 置为 true ，表示当前 watcher 队列正在被刷新
对 watcher 队列进行从小到大排列
【
1.保证组件是从父组件到子组件。 父组件总是在子组件之前创建
2.组件的观察 watcher 在其渲染 watcher 之前运行
3.如果某个组件在父组件观察程序运行期间销毁，它的 watcher 可以被跳过
】
遍历 watcher 队列 【不要缓存长度，因为 watcher 可能会不断添加】
首先执行 watcher.before()
清空缓存，表示当前 watcher 已经被执行，当 watcher 再次入队时就可以入队
执行 watcher.run 方法

当 waiting 为 false，即表示浏览器的异步队列中没有 flushSchduleQueue 函数
将 waiting 赋值 为 true

如果是同步，直接执行 flushSchduleQueue()
【同步执行，直接去刷新队列，性能差】

如果是异步的，执行 nextTick(flushSchedulerQueue)

//nextTicker
将包装后的函数放入 callbacks 数组中， callbacks 是一个全局的数组

将 nextTicker 的回调函数用 try catch 包裹，方便捕获错误

如果 pending 是 false，即当前浏览器任务队列没有任务
将 pending 置为 true
利用浏览器的异步任务队列，将回调函数放进异步任务队列中
【优先使用微任务队列即 promise 等，若当前浏览器不支持使用微任务，则使用宏任务，即 setimeout 等】

//flushCallbacks
将 pending 置为 false
在同一时刻， 浏览器的任务队列中只有一个 flushCallbacks
1.将 pending 再次置为 false 表示下一个 flushCallbacks 函数可以进入浏览器的异步任务队列中
2. 清空 callbacks 数组
3. 执行 callbacks 数组的所有函数
callbacks 中有什么
flushSchdulerQueue
用户调用 this.$nextTick 传递的回调函数