**响应式**

**一 思路流程:**

分析如何响应式的原理是如何实现的, 结合简化版的虚拟dom

**1.** **最简核心原理**

**1 目标**

监听数据改变, 可以操作dom

**0 知识补充**

**1 Object.defineProperty**

不想什么 依赖收集, 派发更新, dep, watcher ...之类的名词, 单纯想怎么实现数据改变, 视图更新

总得有个东西监听把

Object.defineProperty 可以监听, 打印输出

这里核心主要是object.defineProperty

这个方法可以监听对象, 数组中已有属性值, 索引值的变化

但是不能监听对象, 数组中属性值的增添

以数组为例:

var arr = [1, 2, 3, 4]

arr.forEach((item, index) => {

Object.defineProperty(arr, index, {

set: function (val) {

console.log('set')

item = val

},

get: function (val) {

console.log('get')

return item

}

})

})

arr[1]; // get 2

arr[1] = 1; // set 1

arr[10] = 10 // 没有输出!

**2 原理**

-> object.defineProperty 实现监听

手动给初始值 -> 监听改变 -> 改变更新数据

**3 存在问题**

data中只有单个属性

**2. 可使用多个数据**

**1 目标**

支持多个数据, dom元素

**2 原理**

将修改dom的方式封装一下

将响应式封装一下

**3 存在问题**

修改页面没有用到的数据, 也会操作dom

**3. 页面使用的数据改变才修改dom**

**1 目标**

只有页面使用的数据改变才修改dom

**2 原理**

-> 识别哪些数据被使用

在首次setDom的时候, 因为对数据取值了, 所以可以识别出那些数据使用了

-> 将setDom方法挂载到上述数据上

给每个数据分配一个 dep 实例, dep 实例自身可以存储方法, 执行方法

在这个数据使用(读取)时, 触发get钩子, 把setDom添加进其dep中

等这个数据改变了, 触发set钩子, 执行其dep中添加过的方法(setDom)

-> 总结

只有首次 setDom 读取过的数据, 才会将 setDom 加入其dep, 其他数据则不会

**3 存在问题**

1 只有让app指向一个页面元素, 数据和页面才确立关联关系, 才有必要进行后续步骤

否则不应该首次执行 setDom, 也不应该在数据读取的时候添加setDom

**4** **.确保页面挂载才执行后续逻辑**

**1 目标**

只有绑定了app,

才执行一次 setDom,

才在读取数据的时候添加setDom

否则数据改变和视图无关

**2 原理**

**乞丐版:**

1 将绑定app 的操作和 首次setDom放在一起执行, 保证setDom不会单独执行

2 通过一个标志位变量 isMount, 绑定了app, isMount = true, 才会给数据加setDom

存在问题: 添加了辅助变量太乱了, 每个vue实例就需要一个变量

**watcher版:**

1 将绑定app 的操作和 首次setDom放在一起执行, 保证setDom不会单独执行

2 Watcher 类,

每个watcher可以看做 setDom 的一层封装

因为setDom最后是要加入到dep上的, 索性将标志位变成Dep的一个静态属性target,

绑定app后 new watcher(setDom), 内部Dep.target = watcher,

数据改变时, 如果Dep.target 有值, 说明绑定过app了

同时Dep.target(watcher)中也包含了setDom, 可以达到将setDom添加进数据dep的效果

为什么不直接 Dep.target = setDom ?

watcher实例可以保存更多数据(vm, cb) 等, 有利于后续拓展

**3 存在问题**

格式太不 vue 了

**5. 基础版封装成vue的写法**

**1 目标**

写成vue的样子

**2 原理**

1 将数据, app, setDom(render), mount 都变为vue实例的属性或方法

2 watcher实例可以保存vue实例 (这个以后会用到)

3 响应式监听单独封装了一下 observe

**3 存在问题**

1 只能监听 data 中一级属性的改变, 无法监听深层的

**6. 数组 和 对象的响应式处理**

**目标**

1 实现vue中对 数组, 对象的监听方式

2 $set, $delete 两个api

**知识补充**

**1 vue中的数组, 对象表现**

1 vue中 obj 可以监听属性值改变, 但是数据不能监听元素值的改变

**2 重新数组原型**

通过Object.create创建一个全新的数组原型对象arrayProto, 在这上面对执行数组变异方法添加额外处理, 然后将arrayProto作为想监听数组的原型

tip: 不能直接重写Array.prototype, 会污染全局所有数组

// 获得原型上的方法

const arrayProto = Array.prototype;

// 创建一个隔离的新对象，使用现有的对象来提供新创建的对象的\_\_proto\_\_

const arrayMethods = Object.create(arrayProto);

// 做一些拦截的操作, 此处只处理push方法一个

Object.defineProperty(arrayMethods, 'push', {

value(...args) {

console.log('用户传进来的参数', args);

// 真正的push 保证数据如用户期望

arrayProto.push.apply(this, args);

},

enumerable: true,

writable: true,

configurable: true,

});

let list = [1];

list.\_\_proto\_\_ = arrayMethods; // 重置原型

list.push(2, 3);

console.log('用户得到的list:', list);

**1 递归观测数组,对象 的value/元素**

在defineReactive里面递归调用 observe, 观察属性值value, 直到value不是对象

**问题:**

消耗性能, vue中并不观测数组的元素变化

**2 递归观测对象的value, 数组的元素的value(如果元素是对象)**

对象递归观测时依然观测每个value

数组观测时不观测数组的每个元素的改变, 而是观测数组元素内部的value(如果数组元素是对象)

**问题:**

数组必须重新赋值才能更新, 对象属性的增加或删除也不会更新

**3 追加对对象, 数组本身的观测**

**官方做法**

-> vue中对于上面问题的解决方案

数组的api执行时更新

数组, 对象属性的增删: 强制用户使用自定义$set, $delete

**思路**

-> 依赖收集

观测父对象时, 会给每个value生成一个observer实例, 作为\_\_ob\_\_属性, observer里面同样有一个 dep实例

当value被读取的时候, 将 watcher 添加进value的observer的dep中(注意不是key的dep)

tip: 因为是从data开始观测, 所以不会有遗漏

-> 数据改变更新视图

既然不能依靠get, set钩子自动监听更新, 那只能手动监听更新

1 数组的api执行: 重新数组原型的api方法, 执行api的时候手动 让数组自身的observer的dep更新

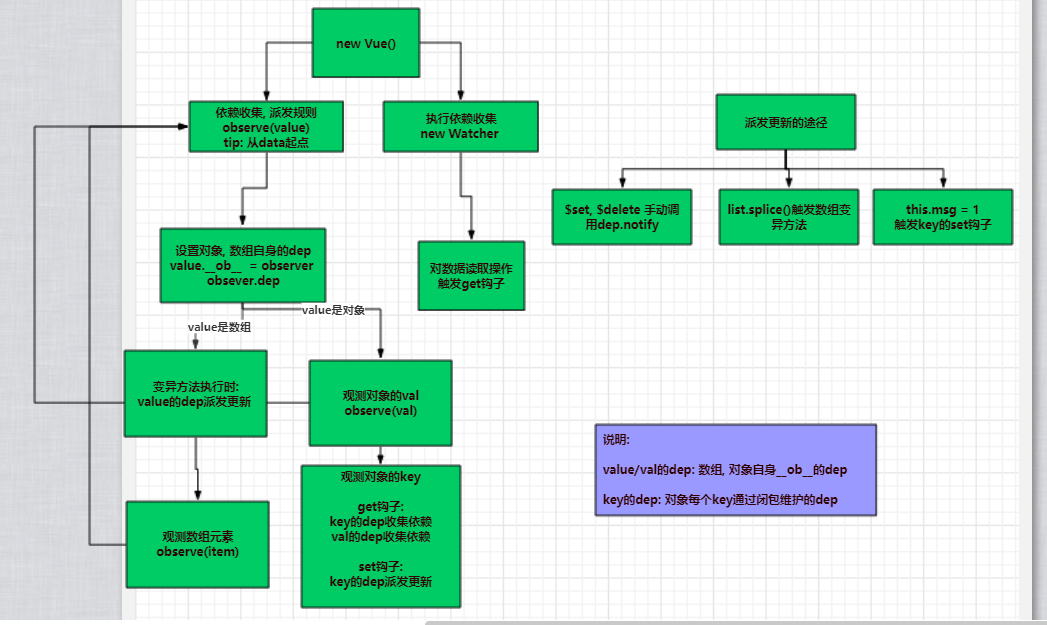
2 数组, 对象属性的增删: 强制用户使用自定义$set, $delete 来执行原操作同时, 手动让数组, 对象自身的observer的dep更新

**存在问题**

1 补充: 数据代理,

**二 原理总结:**

**1. 流程图总结**



**2. 表格总结**

|  |  |  |
| --- | --- | --- |
|  | 数组 | 对象 |
| 收集依赖 | get钩子的:  \_\_ob\_\_.dep | get钩子的:  key.dep  \_\_ob\_\_.dep |
| 触发数组变异方法 | 对变异方法劫持中手动更新 | 无 |
| 改变key/index | $set调数组变异方法, 自动更新 | set钩子: key.dep.notify |
| 增加key/index | $set调数组变异方法, 自动更新 | $set增加新key响应式  手动更新 |
| 删除key/index | $set调数组变异方法, 自动更新 | $set删除key  手动更新 |