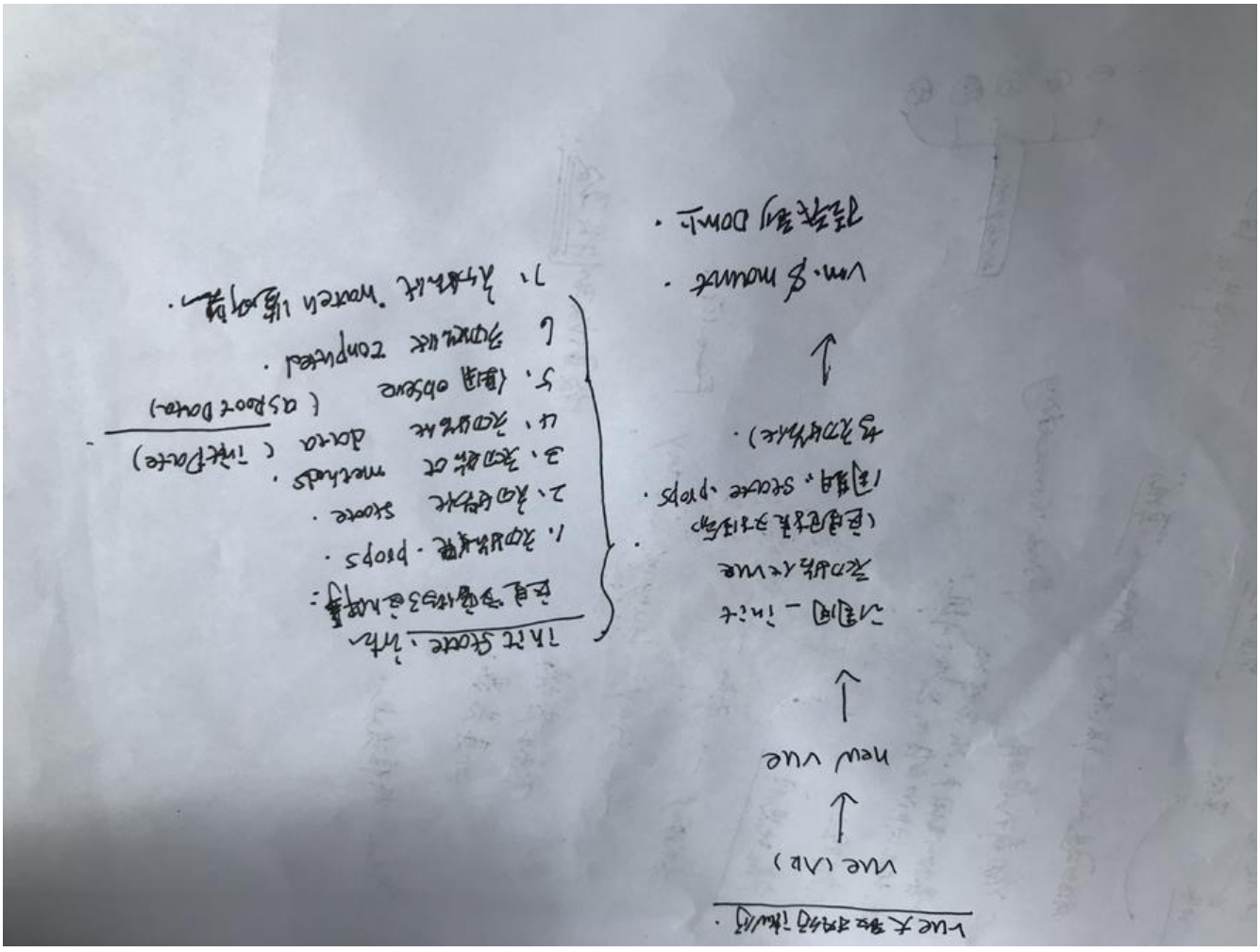
从源码的角度，解读 vue 数据双向绑定

作者：gmw-zjw
原文链接：https://ld246.com/article/1571233115639
来源网站：链滴

vue数据双向绑定，首先谈谈 v-model这个指令， 指令是vue中一个重要的概念，这里不详展开。

这里收手绘了一张图，大概描述了vue的执行过程



原文链接：从源码的角度，解读 vue 数据双向绑定

在initData中，我们可以看到

```
// 得到data
function initData (vm) {
  var data = vm.$options.data;
  data = vm._data = typeof data === 'function'
    ? getData(data, vm)
    : data || {};
  if (!isPlainObject(data)) {
    data = {};
    process.env.NODE_ENV !== 'production' && warn(
      'data functions should return an object:\n' +
      'https://vuejs.org/v2/guide/components.html#data-Must-Be-a-Function',
      vm
    );
  }
  // proxy data on instance
  var keys = Object.keys(data);
  var props = vm.$options.props;
  var methods = vm.$options.methods;
  var i = keys.length;
// 循环遍历
  while (i--) {
    var key = keys[i];
    if (process.env.NODE_ENV !== 'production') {
      if (methods && hasOwn(methods, key)) {
        warn(
          ("Method \"" + key + "\" has already been defined as a data property."),
          vm
        );
      }
    }
    if (props && hasOwn(props, key)) {
      process.env.NODE_ENV !== 'production' && warn(
        "The data property \"" + key + "\" is already declared as a prop. " +
        "Use prop default value instead.",
        vm
      );
    } else if (!isReserved(key)) {
// 把data上面的属性代理带vm上
      proxy(vm, "_data", key);
    }
  }
  // observe data
// observe 中对进行依赖收集，这里是真正的要点，通过observe中的defineReactive对数据进行双绑定。
  observe(data, true /* asRootData */);
}
```

observe

```
function observe (value, asRootData) {
  if (!isObject(value) || value instanceof VNode) {
    return
```

```
  }
  var ob;
  if (hasOwn(value, '__ob__') && value.__ob__ instanceof Observer) {
    ob = value.__ob__;
  } else if (
    shouldObserve &&
    !isServerRendering() &&
    (Array.isArray(value) || isPlainObject(value)) &&
    Object.isExtensible(value) &&
    !value._isVue
  ) {
// ob 实例
    ob = new Observer(value);
  }
  if (asRootData && ob) {
    ob.vmCount++;
  }
  return ob
}
```

observe是使用defineReactive对数据进行响应式双向绑定的，最终通过Object.defindProtery进行赖收集。

Observe为数据添加上相应式数据绑定，如果是数组，将修改后可以截获响应的数组方法替换掉该数的原型中的原生方法，达到监听数组数据变化响应的效果。

```
export class Observer {
  value: any;
  dep: Dep;
  vmCount: number; // number of vms that have this object as root $data

  constructor (value: any) {
    this.value = value
    this.dep = new Dep()
    this.vmCount = 0
    def(value, '__ob__', this)
    if (Array.isArray(value)) {
      /*
        如果是数组，将修改后可以截获响应的数组方法替换掉该数组的原型中的原生方法，达到监听数数据变化响应的效果。
        这里如果当前浏览器支持__proto__属性，则直接覆盖当前数组对象原型上的原生数组方法，如不支持该属性，则直接覆盖数组对象的原型。
      */
      if (hasProto) {
        protoAugment(value, arrayMethods)
      } else {
        copyAugment(value, arrayMethods, arrayKeys)
      }
      // 如果是数组则需要遍历对数组的每一个元素进行observe
      this.observeArray(value)
    } else {
      // 如果是对象则用walk进行绑定
      this.walk(value)
    }
```

原文链接：从源码的角度，解读 vue 数据双向绑定

```
  }

  /**
   * Walk through all properties and convert them into
   * getter/setters. This method should only be called when
   * value type is Object.
   */
  walk (obj: Object) {
    const keys = Object.keys(obj)
    for (let i = 0; i < keys.length; i++) {
      defineReactive(obj, keys[i])
    }
  }

  /**
   * Observe a list of Array items.
   */
  observeArray (items: Array<any>) {
    for (let i = 0, l = items.length; i < l; i++) {
      observe(items[i])
    }
  }
}
```

是数组的时候就会出现一个问题，原生的数组方法 无法使用 pop push，这时 vue 对原生数组方法行二次封装。

```
// observe/arr.js
/*
 * not type checking this file because flow doesn't play well with
 * dynamically accessing methods on Array prototype
 */
// 这个类会重写数组的方法 push、pop、shift、unshift、sort、splice、reverse
import { def } from '../util/index'

/*取得原生数组的原型*/
const arrayProto = Array.prototype
/*创建一个新的数组对象，修改该对象上的数组的七个方法，防止污染原生数组方法*/
export const arrayMethods = Object.create(arrayProto)

const methodsToPatch = [
  'push',
  'pop',
  'shift',
  'unshift',
  'splice',
  'sort',
  'reverse'
]

/**
 * Intercept mutating methods and emit events
 */
methodsToPatch.forEach(function (method) {
```

```
  // cache original method
  // 将数组的原生方法缓存起来，稍后会使用
  const original = arrayProto[method]
  def(arrayMethods, method, function mutator (...args) {
    // 调用原生数组方法
    const result = original.apply(this, args)
    // 将数组新插入的元素需要observe才能响应式
    const ob = this.__ob__
    let inserted
    switch (method) {
      case 'push':
      case 'unshift':
        inserted = args
        break
      case 'splice':
        inserted = args.slice(2)
        break
    }
    if (inserted) ob.observeArray(inserted)
    // notify change
    /*dep通知所有注册的观察者进行响应式处理*/
    ob.dep.notify()
    return result
  })
})
```

原文链接：从源码的角度，解读 vue 数据双向绑定