## init

### this. \_initData

this.\_proxy //数据代理

observe //设置数据观察

ob = new Observer(value); //开始创建ob对象

this.value .\_\_ob\_\_ = this

this.dep = new Dep;

开始针对value做数据监控

defineReactive(this.value, key, val);  
判断数据data是不是有子结构

observe 递归调用，观察数据

dom v-for对应生成一个Directive

Directive中开始构建Watcher

将当年的watcher对象绑定到指令对应的数据data上

this.data.item 上的deps 订阅对应指令的watcher

this.data.itme[0,1,…] 订阅子依赖

### \_initComputed

扩充 vn.b 方法

Object.defineProperty(vm, b,{

enumerable: true,

configurable: true

set:

get: new Watcher();

});

\_directives.\_bind

生成watcher 对象

收集 deps依赖

## Data是数组

data: {

items: [

{ message: 'Foo' },

{ message: 'Bar' }

],

Aaa

Bbb,

…..

}

observe(data, this);

data.dep

data.value

data.value.\_\_ob\_\_ = data

function defineReactive (data,item , val){

var dep = new Dep()

var childOb = observe(val);

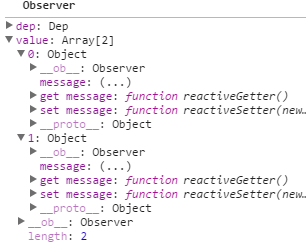
items: [

{ message: 'Foo' },

{ message: 'Bar' }

],

递归给每一个key创建observe



Object.defineProperty(data, item, set,get)

}