1. 双向数据绑定原理（数据劫持和发布订阅）

Object.defineProperty()

let object1 = {};

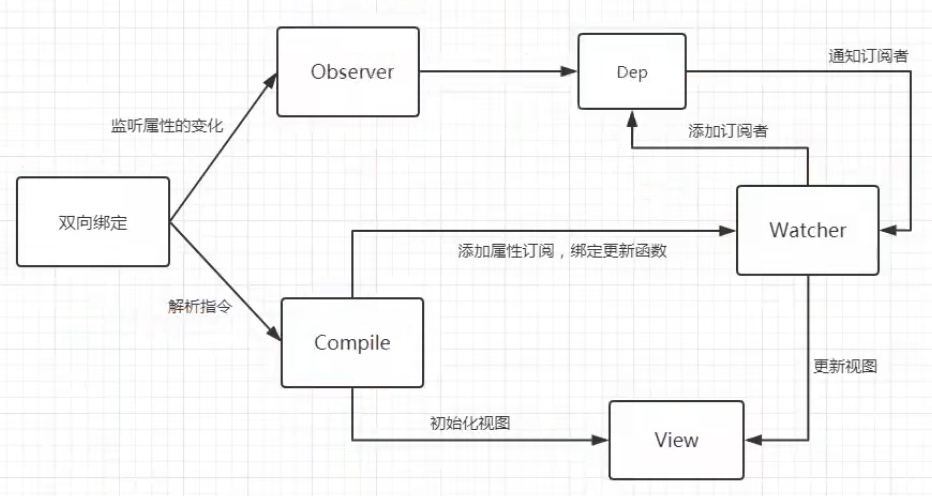
Object.defineProperty( object1, “property1”, {value: 33, writable: false} )

详细流程：

Observer 监听器，用来监听属性的变化通知订阅者

Watcher 订阅者，收到属性的变化更新视图

Compile 解析器，解析指令，初始化模板，绑定订阅者



Vue的双向数据绑定的代码实现（有助于双向数据绑定的理解）

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Title</title>

</head>

<body>

<div id="app">

<input type="text" v-model="msg">

<h3>{{msg}}</h3>

</div>

<script>

// XxVue构造方法

function XxVue(options = {}) {

this.$options = options;

// 获取el元素

this.$el = document.querySelector(options.el);

this.\_data = options.data;

// 设置订阅池保存订阅器

this.\_watcherTpl = {};

// 设置observer函数，对data数据进行重写，实现数据变化的监听

this.\_observer(this.\_data);

// 编译模板和指令

this.\_compile(this.$el);

}

// XxVue里的构造中的方法 \_observer ( 通过原型找到\_observer方法 )

XxVue.prototype.\_observer = function(data) {

Object.keys(data).forEach((key)=>{

// 添加每一个属性的订阅器的存放位置

this.\_watcherTpl[key] = {

\_directives: []

}

let watcherTpl = this.\_watcherTpl[key];

// 获取值

var value = data[key];

Object.defineProperty(this.\_data, key, {

configurable: true,

enumerable: true,

get() {

console.log("获取值");

return value;

},

set(newValue) {

console.log("设置值");

if(newValue != value) {

value = newValue;

// 将订阅池（watcher）对应的属性进行触发更新

console.log("将订阅池（watcher）对应的属性进行触发更新")

watcherTpl.\_directives.forEach(item=>{

item.update();

})

}

}

})

})

}

// watcher 构造函数

function Watcher(el, vm, val, attr) {

this.el = el;

this.vm = vm;

this.val = val;

this.attr = attr;

this.update();

}

Watcher.prototype.update = function () {

this.el[this.attr] = this.vm.\_data[this.val];

}

// 编译器

XxVue.prototype.\_compile = function(el) {

var nodes = el.children;

for (let i = 0; i<nodes.length; i++) {

var node = nodes[i];

if(node.children.length) {

// 进行递归便利

this.\_compile(node);

}

// 判断是否有v-model指令， 并且是输入框

if(node.hasAttributes("v-model") && node.tagName == "INPUT") {

node.addEventListener("input", ((key)=>{

var attrVal = node.getAttribute("v-model");

// 创建watcher对象， 并且将订阅器根据属性放入到对应的订阅器集合中

var watcher = new Watcher(node, this, attrVal, "value");

this.\_watcherTpl[attrVal].\_directives.push(watcher);

return ()=>{

this.\_data[attrVal] = nodes[key].value;

}

})(i))

}

// 匹配模板变量{{}}

var reg = /\{\{\s\* (.\*?) \s\* \}\}/igs;

var txt = node.textContent;

if(reg.test(txt)) {

node.textContent = txt.replace(reg, (matched, placeholder)=>{

var watcher = new Watcher(node, this, placeholder, "innerHTML");

this.\_watcherTpl[placeholder].\_directives.push(watcher);

return placeholder.split(".").reduce((val, key)=>{

return val[key];

}, this.\_data)

});

}

}

}

// 使用XxVue构造

var app = new XxVue({

el: '#app',

data: {

msg: "helloWord"

}

})

</script>

</body>

</html>

二、