#### 1.vue

vue我们需要vuex，需要router，需要服务器，需要服务缓存，需要代理跨域....

#### **vuex**

相当于某种意义上设置了读写权限的全局变量，将数据保存保存到该“全局变量”下，并通过一定的方法去读写数据。

Vuex 并不限制你的代码结构。但是，它规定了一些需要遵守的规则：

应用层级的状态应该集中到单个 store 对象中。

提交 mutation 是更改状态的唯一方法，并且这个过程是同步的。

异步逻辑都应该封装到 action 里面。

 Vuex 相关代码分割到模块中。下面是项目结构示例：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15 | ├── index.html  ├── main.js  ├── api  │   └── ... # 抽取出API请求  ├── components  │   ├── App.vue  │   └── ...  └── store      ├── index.js          # 我们组装模块并导出 store 的地方      └── moving            # 电影模块          ├── index.js      # 模块内组装，并导出模块的地方          ├── actions.js    # 模块基本 action          ├── getters.js    # 模块级别 getters          ├── mutations.js  # 模块级别 mutations          └── types.js      # 模块级别 types |  |  |  |

所以我们开始在我们的src目录下新建一个名为store 的文件夹 为了后期考虑 我们新建了moving 文件夹，用来组织电影，考虑到所有的action,getters,mutations,都写在一起，

1. stroe文件夹建好，我们要开始在main.js里面引用vuex实例：

import store from './store'

new Vue({

el: '#app',

router,

store,

template: '<App/>',

components: { App }

})

#### webpack proxyTable 代理跨域

proxyTable: {

'/api': {

target: 'http://api.douban.com/v2',

changeOrigin: true,

pathRewrite: {

}

Xxx.js

这样当我们访问localhost:8080/api/movie

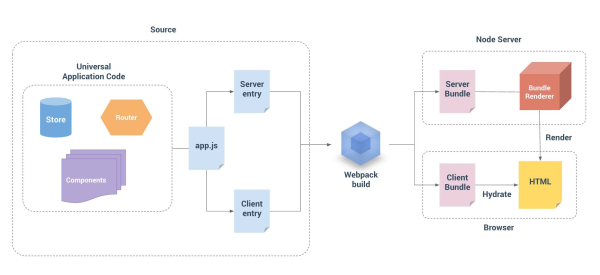
的时候 其实我们访问的是http://api.douban.com/v2/movie

'^/api': ''

}

}

#### 4.服务器端渲染



#### ****5 .Object.defineProperty****

定义对象的新属性或修改原有的属性Object.defineProperty(obj, prop, descriptor)

(对象是由多个名/值对组成的无序的集合。对象中每个属性对应任意类型的值。  
定义对象可以使用构造函数或字面量的形式：

var obj = new Object; //obj = {}

obj.name = "张三"; //添加描述

obj.say = function(){}; //添加行为)

**格式：**

Object.defineProperty(obj,"test",{

configurable:true | false,

enumerable:true | false,//是否可以被枚举（使用for...in或Object.keys()）

value:任意类型的值,

writable:true | false

});

\*\*\*兼容性

在ie8下只能在DOM对象上使用，尝试在原生的对象使用 Object.defineProperty()会报错。

#### 前端调试 console network source



#### mvvm：model view viewmodel

-1-双向数据绑定

数据劫持 set 触发

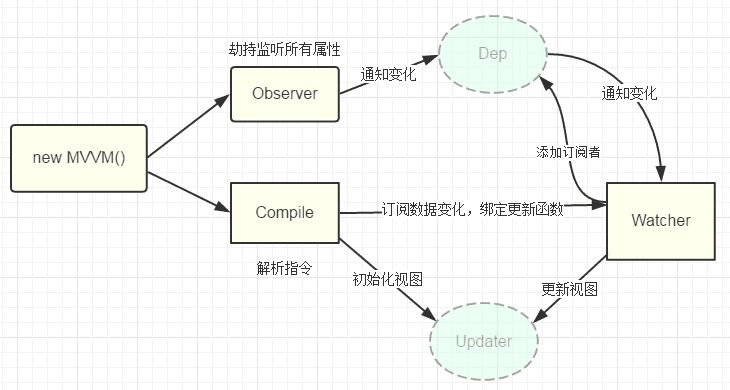
数据劫持体现：***observe***的Object.defineProperty设置set，get方法 同步vm对象属性到dep。

-2-虚拟dom树，使用js描述dom节点，相当于实现了一个ＨＴＭＬ文本解析器，浏览器直接使用减少浏览器渲染时间。更改之后真实dom的js描述与改动的js对dom描述对比使用深度有限算法，实现渲染最优

”差异化类型“和”差异化位置“。   
现在就很明朗了，剩下的操作就是将差异化内容经过ＤＯＭ操作渲染到页面上即可完成

(dom操作，目前的render和repaint都是通过jq的html功能实现，由于页面数据改变次数较多，会导致页面重构重绘压力很大)

-3- 发布订阅 run notify watch update



#### 7.1.Mvvm

observe(**data**, **this**);

**this**.$compile = new Compile(options.el || document.body, **this**)

$watch: function(key, cb, options) {

new Watcher(this, key, cb);//????

#### **7.2.**Observer

Obeject.defineProperty()来监听属性变动，将需要observe的数据对象进行递归遍历，包括子属性对象的属性，都加上 setter和getter

#### 7.3.Watcher

设置Watcher订阅者作为Observer和Compile之间通信的桥梁，主要做的事情是:  
1、在自身实例化时往属性订阅器(dep)里面添加自己  
2、自身必须有一个update()方法  
3、待属性变动dep.notice()通知时，能调用自身的update()方法，并触发Compile中绑定的回调，则功成身退。

#### **7.4. Compile**对象

主要做的事情是解析模板指令，将模板中的变量替换成数据，然后初始化渲染页面视图，并将每个指令对应的节点绑定更新函数，添加监听数据的订阅者，一旦数据有变动，收到通知，更新视图

(\*\*\*算法实现4.1 步骤一：用JS对象模拟DOM树2 步骤二：比较两棵虚拟DOM树的差异3 步骤三：把差异应用到真正的DOM树上

Observe walk函数触发dep对象，dep发布通知notify

同时在watcher对象中调用addDep() 添加监听对象，watcher的update指向vm的update即触发compile的update方法

--MVVM是Model-View-ViewModel的简写。它本质上就是MVC 的改进版。MVVM 就是将其中的View 的状态和行为抽象化，让我们将视图 UI 和业务逻辑分开。当然这些事 ViewModel 已经帮我们做了

1. 准备 -1-vue+webpack搭建todo应用

视频源码 <https://www.imooc.com/learn/935>

-2-Webpack4 升级指北 慕课

1. 目标： vue组件，方法方式调用 多页面

Tab切换组件 多页调用

Ssr