

18 Webpack 性能优化

学习时长: 8分48秒

19 实现小型打包工具

学习时长: 13分43秒

20 React 和 Vue 两大框架之间的相互关系

学习时长: 11分20秒

21 Vue 常考基础知识

已学完 学习时长: 13分28秒

22 Vue 常考进阶知识点

已学完 学习时长: 26分37秒

23 React 常考基础知识

学习时长: 17分7秒

24 React 常考进阶知识点

学习时长: 5分25秒

25 监控

已学完 学习时长: 2分48秒

26 UDP

已学完 学习时长: 3分11秒

React 和 Vue 两大框架之间的相爱相杀

React 和 Vue 应该是国内当下最火热的前端框架，当然 Angular 也是一个不错的框架，但是这个产品面内使用的人很少再加上我对 Angular 也不怎么熟悉，所以框架的章节中不会涉及到 Angular 的内容。

这一章节，我们将会来学习以下几个内容

- MVVM 是什么
- Virtual DOM 是什么
- 前端路由是如何跳转的
- React 和 Vue 之间的区别

MVVM

涉及面试题：什么是 MVVM？比之 MVC 有什么区别？

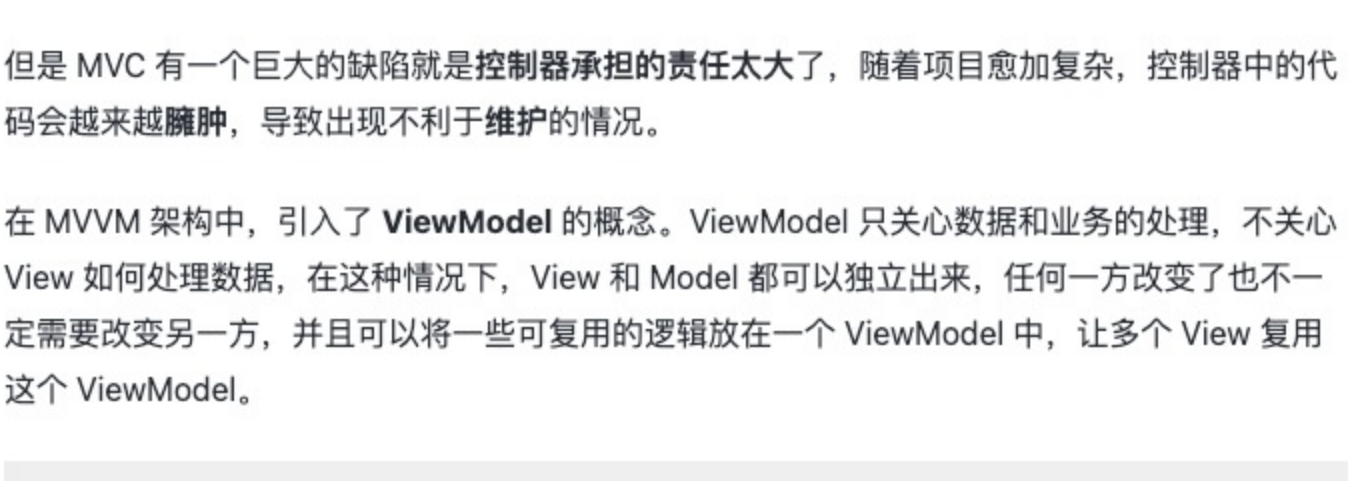
首先先申明一点，不管是 React 还是 Vue，它们都不是 MVVM 框架，只是有借鉴 MVVM 的思路，文中拿 Vue 举例也是为了更好地理解 MVVM 的概念。

接下来先说下 View 和 Model：

- View 很简单，就是用户看到的视图
- Model 同样很简单，一般都是本地数据和数据库中的数据

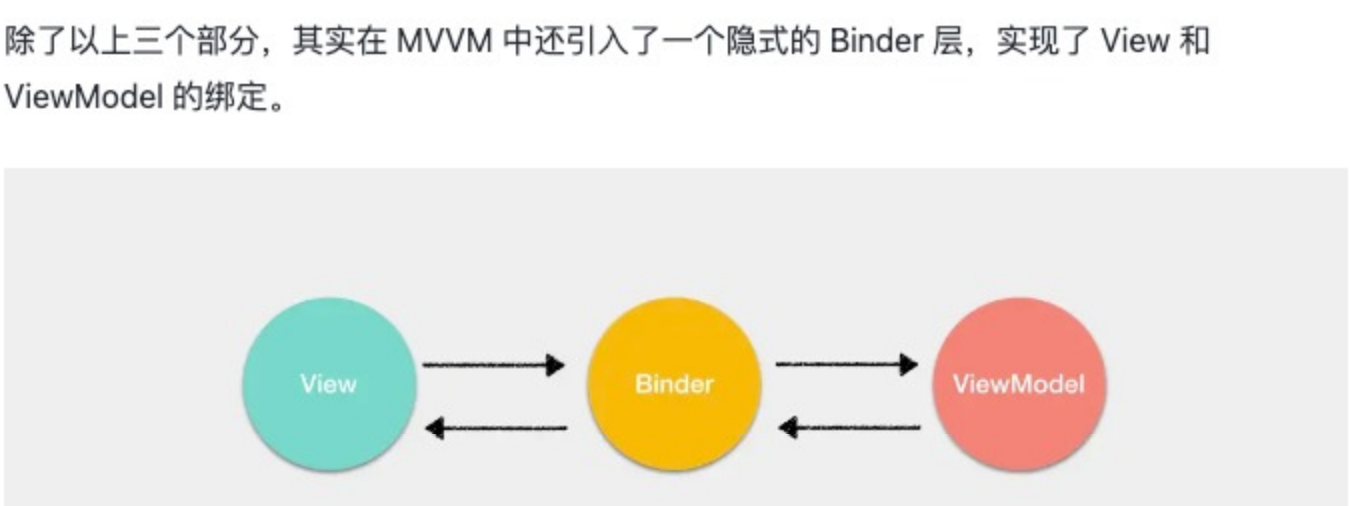
基本上，我们写的产品都是通过接口从数据库中读取数据，然后将数据经过处理展现到用户看到的视图上，当然我们还可以从视图上读取用户的输入，然后又将用户的输入通过接口写入到数据库中，但是，如何将数据展示到视图上，然后又如何将用户的输入写入到数据库中，不同的人就产生了不同的看法，从此出现了很多种架构设计。

传统的 MVC 架构通常使用控制器更新模型，视图从模型中获取数据去渲染，当用户有输入时，会通过控制器去更新模型，并且通知视图进行更新。



但是 MVC 有一个巨大的缺陷就是控制器承担的责任太大了，随着项目愈加复杂，控制器中的代码会越来越臃肿，导致出现不利于维护的情况。

在 MVVM 架构中，引入了 **ViewModel** 的概念，**ViewModel** 只关心数据和业务的处理，不关心 View 如何处理数据，在这种情况下，View 和 Model 都可以独立出来，任何一方改变了也不一定需要改变另一方，并且可以将一些可复用的逻辑放在一个 **ViewModel** 中，让多个 View 复用这个 **ViewModel**。



以 Vue 框架来举例，**ViewModel** 就是组件的实例，**View** 就是模板，**Model** 的话在 **Vuex** 的情况下是完全可以和组件分离的。

除了以上三个部分，其实在 MVVM 中还引入了一个隐式的 **Binder** 层，实现了 View 和 **ViewModel** 的绑定。



同样以 Vue 框架来举例，这个隐式的 **Binder** 层就是 Vue 通过解析模板中的指令和指令从而实现对 View 与 **ViewModel** 的绑定。

对于 MVC 来说，其实最重要的并不是通过双向绑定或者其他方式将 View 与 **ViewModel** 绑定起来，而是通过 **ViewModel** 将视图中的状态和用户的行为分离出一个抽象，这才是 **MVVM** 的精髓。

Virtual DOM

涉及面试题：什么是 Virtual DOM？为什么 Virtual DOM 比原生 DOM 快？

大家都知道操作 DOM 是很慢的，为什么慢的原因已经在「浏览器渲染原理」章节中说过，这里就不再赘述了。

那么相较于 DOM 来说，操作 JS 对象会快很多，并且我们也可以通过 JS 来模拟 DOM

```
const ul = {
  tag: 'ul',
  props: {
    class: 'list'
  },
  children: {
    tag: 'li',
    children: '1'
  }
}
```

上述代码对应的 DOM 就是

```
<ul class="list">
  <li-1/li>
</ul>
```

那么既然 DOM 可以通过 JS 对象来模拟，反之也可以通过 JS 对象来模拟对应的 DOM，当然了，通过 JS 来模拟 DOM 并且渲染对应的 DOM 只是第一步，难点在于如何判断新旧两个 JS 对象的最小差异并实现局部更新 DOM。

首先 DOM 是一个多叉树的结构，如果需要完整的对比两颗树的差异，那么需要的时间复杂度会是 $O(n^3)$ ，这个复杂度肯定是不能接受的，于是 React 团队优化了算法，实现了 $O(n)$ 的复杂度来对比差异。实现 $O(n)$ 复杂度的关键就是只对比同层的节点，而不是跨层对比，这也是考虑到在实际业务中很少会去跨层的移动 DOM 元素。所以判断差异的算法就分为了两步

- 首先从上至下，从左往右遍历对象，也就是树的深度遍历，这一步中会给每个节点添加索引，便于最后渲染差异
- 一旦节点有子元素，就去判断子元素是否有不同

在第一步算法中我们需要判断新旧节点的 **tagName** 是否相同，如果不相同的话就代表节点被替换了，如果没有更改 **tagName** 的话，就需要判断是否有子元素，有的话就进行第二步算法。

在第二步算法中，我们需要判断原本的列表中是否有节点被移除，在新的列表中需要判断是否有新的节点加入，还需要判断节点是否有移动。

举个例子来说，假设页面中只有一个列表，我们对列表中的元素进行了变更

```
// 假设这里新增了一个 ul，其中包含了 3 个 li
[1, 2, 3, 4, 5]
// 这里删除上面的 li
[1, 2, 5, 4]
```

从上述例子中，我们一眼就可以看出先前的 **ul** 中的第三个 **li** 被移除了，四五替换了位置。

那么在实际的算法中，我们如何去识别改动的是哪个节点呢？这就引入了 **key** 这个属性，想必大家在 **Vue** 或者 **React** 的列表中都用过这个属性。这个属性是用来给每一个节点打标签的，用于判断是否是同一个节点。

当然在判断以上差异的过程中，我们还需要判断节点的属性是否有变化等等。

当判断出以上差异后，就可以把这些差异记录下来，当对比完两棵树后，就可以通过差异去局部更新 DOM，差异性能的最优化。

另外再来回答「为什么 Virtual DOM 比原生 DOM 快」这个问题。首先这个问题得分场景来说，如果无脑替换所有的 DOM 这种场景来说，Virtual DOM 的局部更新肯定是要来的快，但是如果你可以人工也何样去局部替换 DOM，那么 Virtual DOM 必然没有你直接操作 DOM 来的快，毕竟还有一层 diff 算法的损耗。

当然了 Virtual DOM 提高性能是其中一个优势，其实最大的优势还是在：

1. 将 Virtual DOM 作为一个兼容层，让我们还能对接非 Web 端的系统，实现跨端开发。
2. 同样的，通过 Virtual DOM 我们可以渲染到其他平台，比如实现 SSR、同构渲染等等。
3. 实现组件的高度抽象化

路由原理

涉及面试题：前端路由原理？两种实现方式有什么区别？

前端路由要实现起来其实很简单，本质就是监听 **URL** 的变化，然后匹配路由规则，显示相应的页面，并且无须刷新页面。目前前端使用的路由就只有两种实现方式

- Hash 模式
- History 模式

Hash 模式

www.test.com/#/ 就是 Hash URL，当 # 后面的哈希值发生变化时，可以通过 **hashchange** 事件来监听到 URL 的变化，从而进行跳转页面，并且无论哈希值如何变化，服务端接收到的 URL 请求永远是 **www.test.com**。

```
window.addEventListener('hashchange', () => {
  // ... 业务逻辑
})
```

Hash 模式相对来说更简单，并且兼容性也更好。

History 模式

History 模式是 HTML5 新推出的功能，主要使用 **history.pushState** 和 **history.replaceState** 改变 URL。

通过 History 模式改变 URL 同样不会引起页面的刷新，只会更新浏览器的历史记录。

```
// 新增历史记录
history.pushState(stateObject, title, URL)
// 替换当前历史记录
history.replaceState(stateObject, title, URL)
```

当用户做出浏览器动作时，比如点击后退按钮时会触发 **popState** 事件

```
window.addEventListener('popstate', e => {
  // e.state 是 pushState(stateObject) 中传 stateObject
  console.log(e.state)
})
```

两种模式对比

- Hash 模式只可以更改 # 后面的内容，History 模式可以通过 API 设置任意的同源 URL
- History 模式可以通过 API 添加任意类型的数据到历史记录中，Hash 模式只能更改哈希值，也就是字符串
- Hash 模式无需后端配置，并且兼容性好。History 模式在用户手动输入地址或者刷新页面的时候会发起 URL 请求，后端需要配置 **index.html** 页面用于匹配不到静态资源的时候

Vue 和 React 之间的区别

Vue 的表单可以使用 **v-model** 支持双向绑定，相比于 React 来说开发上更加方便，当然了 **v-model** 其实就是个语法糖，本质上和 React 写表单的方式没什么区别。

改变数据方式不同，Vue 修改状态相比来说要简单许多，React 需要使用 **setState** 来改变状态，并且使用这个 API 也有一些坑点，并且 Vue 的底层使用了依赖追踪，页面更新渲染已经是最优的了，但是 React 还是需要用户手动去优化这方面的问题。

React 16以后，有些钩子函数会执行多次，这是因为引入 Fiber 的原因，这在后续的章节中讲到。

React 需要使用 JSX，有一定的上手成本，并且需要一整套的工具链支持，但是完全可以通过 JS 来控制页面，更加的灵活，Vue 使用了模板语法，相比于 JSX 来说没有那么灵活，但是完全可以脱离工具链，通过直接编写 **render** 函数就能在浏览器中运行。

在生态上来说，两者其实没多大的差距，当然 React 的用户是远远高于 Vue 的。

在上手成本上来说，Vue 一开始的定位就是尽可能的降低前端的门槛，然而 React 更多的是去改变用户去接受它的概念和思想，相较于 Vue 来说上手成本更高。

小结

这一章节我们学习了几大框架中的相似点，也对比了 React 和 Vue 之间的区别，其实我们可以发现，React 和 Vue 虽然是两个不同的框架，但是他们的底层原理都是很相似的，无非在上层掩饰了自己的概念上去，所以我们无需去对比到底哪个框架牛逼，引用尤大的一句话

说到底，就算你证明了 A 比 B 牛逼，也不意味着你或者你的项目就牛逼了... 比起争论这个，不如多想想怎么让自己变得更牛逼吧。

留言

输入评论 (Enter 换行, 写 + Enter 发送)

发表评论

全部评论 (49)

OlLong

前端开发

1月前

谁时候了还在说Virtual比原生Dom快，这个小弟完全是存疑。

点踩 回复

不会做蛋糕的程序员

前端开发

1月前

「为什么 Virtual DOM 比原生 DOM 快」这个问题是不是等于没说

点踩 回复

白宇

前端开发工程师

2年前

感觉是在黑React?

点踩 回复

痛得睡不着

web前端

2年前

React和vue的domdiff算法是不一样的

点踩 回复

前端古力士

前端工程师 @ 字节跳动

3年前

老师，我觉得小弟里的某些知识点可以展开讲解一下，只讲了概念的话，就不能很理解，硬记的话更难记住且很快会忘记，当然你可能觉得为什么自己不去查资料扩展学习一下，但是花点钱买个小的目的就是为了让看中的资料，节省的时间和精力。

点踩 2 回复

Gong42503

加一

3年前

点踩 回复

饼干NICK

2年前

感觉这本小弟很多内容读了一个知识点都得再点两页外搜索，不然就感觉没读大区别，是我太菜了么

点踩 回复

前端古力士

前端工程师 @ 字节跳动

3年前

两种路由模式的对比,如果能举出例子就好了,单纯的看这些对比很难记住他们的区别。

点踩 6 回复

待我长发及腰就是我

4年前

那传统MVC框架是什么，jquery+vue1.0算传统的MVC框架吗，这个概念不是很好理解啊，还是说清楚什么是MVVM框架，vue2.0不是MVVM框架，那什么是

点踩 13 回复

蒙面大虾

软件开发 @ 京东

4年前

MVC的惯例，箭头指向是不是有点问题？ C -> M?

点踩 1 2 回复

明辨是非

3年前

确实是有问题的

点踩 回复

成都麻麻辣

3年前

确实感觉不对

点踩 回复

法令男

4年前

请问下方老师讲的MVC有没有参考价值:

点踩 回复

努男

4年前

到底什么是mvvm 看不懂

点踩 14 回复

zy_ch

外委员 @ 美团

4年前

通过 ViewModel 将视图中的状态和用户的 行为分离出一个抽象，这才是 MVVM 的精髓

点踩 1 回复

helloworldkk

前端工程师

4年前

为什么说不管是 React 还是 Vue，它们都不是 MVVM 框架？

点踩 11 回复

不躺平中

前端

4年前

传统的 MVC 架构通常使用控制器更新模型，视图从模型中获取数据去渲染，这句话感觉有问题，视图不是经过控制器才能获取数据吗？

点踩 1 回复

rocky191

前端小菜鸟 @ 创业公司

4年前

与其争论哪个框架厉害，不如让自己变得厉害

点踩 4 回复

于先森

前端工程师 @ 某的游戏...

4年前

在React里监听输入是通过onChange事件之后不断更新state(这里要用setState方法否则不引起重新渲染)之后把input.value设为state中的值，vue-v-model就是把这工作帮你做了

点踩 3 回复

桃瑞

前端打杂 @ 蚂蚁集团

4年前

当用了 v-model 其实就是个语法糖，本质上和 React 写表单的方式没什么区别。这句话是怎么理解呢？ v-model 我没想到和 React 写表单有什么关系

点踩 4 回复

佛系的阿海

4年前

就是v-model在表单中一般是value和change事件的包装，对React来说就是帮你省去了这两个事件

点踩 回复

佛系的阿海

4年前

错了，是value和input事件，不好意思

点踩 1 回复

查看更多回复

琪巴

前端工程师

4年前

能否介绍下React中的mobx和riot

点踩 2 回复

rocky191

4年前

react跟原生本已经没有minix的概念了

点踩 回复

dcbryant18...

4年前

react也没有riot啊

点踩 回复

BrownHu

front end @ xx

4年前

diff 算法应该还是优先广度遍历吧

点踩 2 4 回复

Swiftty10583

4年前

树的遍历，大多是深度优先遍历

点踩 回复

yybbb

4年前

上面写的是「从左往右遍历对象」，这就是广度遍历，深度遍历是优先从根节点一路向下去到达最深叶子节点。

点踩 1 回复

查看更多回复

H046602

web前端

4年前

MVC 的图画画错了吧。

点踩 2 回复

yck (作者)

4年前

这个并没有绝对对错，在不同的地方MVC的指向都有略微的区别

点踩 回复

琪巴

4年前

表示MVC的图画和我之前看的不一樣

点踩 回复

newbeekk

前端开发

4年前

说到底，就算你证明了 A 比 B 牛逼，也不意味着你或者你的项目就牛逼了... 比起争论这个，不如多想想怎么让自己变得更牛逼吧。

点踩 4 1 回复

nn在掘金

3年前

书这句话的人数挺牛逼

点踩 回复

查看全部 49 条回复