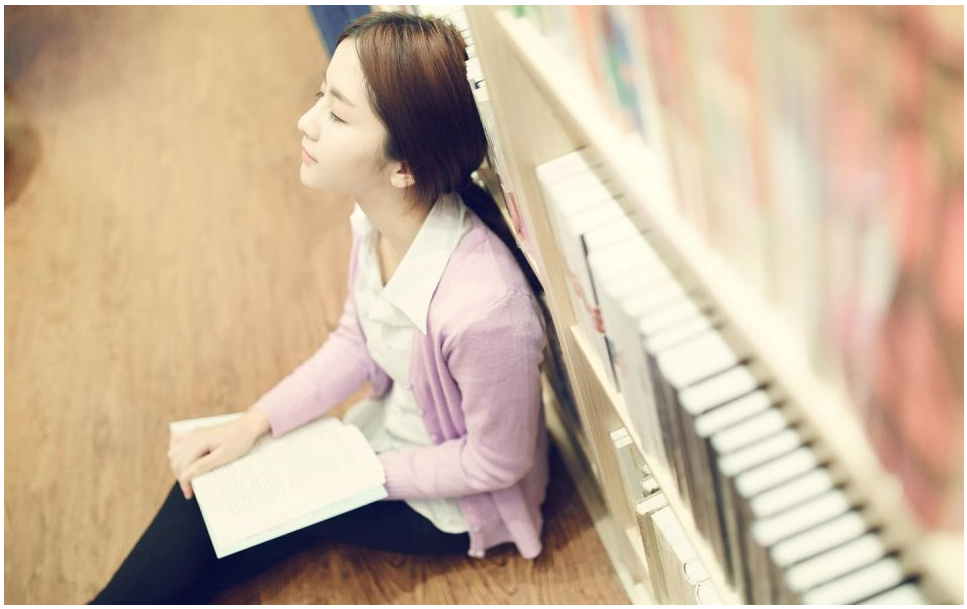
揭秘react生态体系

苏禾
全栈工程师



前言

欢迎来我的社区蹦迪

探知社区-知识探索社区
www.tanzhi.club

react 的生态体系比较庞大，它在web端，移动端，服务器端，VR领域都有涉及。

react可以说是目前为止最热门，生态最完善，应用范围最广的前端框架。react结合它的整个生态，它可以横跨web端，移动端，服务器端，乃至VR领域。

可以毫不夸张地说，react已不单纯是一个框架，而是一个行业解决方案。

下面就来说说 react庞大生态体系的构成。

一，react生态之一—web端

react本身是面向web端的，它很轻便灵活，只是MVC架构中的view(视图)层。由于只是view层，所以它需要配合生态体系中的其他框架或模块来使用。

react的web生态构成

- 1，路由
  react的路由解决方案有多个，这里只提用的最多，也最为推荐的*ef="https://github.com/ReactTrainin ..."*>react-router。现已更新到v4.1版本。另外，推荐使用*ef="https://github.com/reactjs/reac ..."*>react-router-redux和*ef="https://github.com/ReactTrainin..."*>react-router搭配使用，可保持路由器与应用程序状态同步。

- 2，状态管理器
  react只是UI层，对于如何管理应用的状态，facebook提出了flux架构，而基于这一架构，react生态陆续出现了*redux*、*refluxjs*、*mobx*、f="github.com/reactjs/reac..."">react-redux 等一系列状态管理框架。

  其中*redux*、*mobx*是无疑是最受欢迎的两个。但它们的应用场景则大不相同。

Mobx 适合做一些简单的应用，原型实验，适合小的团队使用。Mobx 的优点是响应状态的变化。

redux适合复杂的应用，大团队，需求变化多。它的优点是响应动作和事件。
redux不仅应用于react，也可以应用于angular，vue等框架，只是redux和react配合使用最为契合。

另外国内蚂蚁金服前端团队基于redux, react-router打造了另一个前端框架——*dva*。
*dva*简单来讲是对redux方案的集成与拓展，它突破框架的本身，形成一套略为完整的前端架构，处理了很多包括项目构建，异步处理、统一请求、统一错误处理等一系列诸多问题。

如果你选择redux方案，那么建议直接使用*dva*。

- 3，UI库
  和vue,angular相比，react的UI库无疑是最为丰富的，且十分优秀。目前react的UI库有将近二十多个，这里主要列举最为优秀的几个。

  首先国外的有*material-ui*、react-toolbox。它们都基于谷歌的material设计理念，因此界面非常精美，尤其适用于web开发。

  其次是国内蚂蚁金服开源的ant design，以及百分点公司开源的bfd-ui。这两个都是企业级的UI库，提供的组件极其丰富，此外逻辑交互也处理得非常好，基本不需要你操作过多的业务逻辑即可完成开发。

  而在这众多的UI库中，表现最为出色的莫过于蚂蚁金服开源的ant design。
- 4, 一些工具
  Immutable
  immutable-js是facebook推出的完全独立的一个js库，侧重函数式编程中不可变数据结构，使用Immutable可使你的react应用性能上会有很大的提升。

  draft-js——基于react的编辑器语言
  draft-js 是由facebook开源的编辑器语言。它提供了众多API可用于定制化开发你想要的react编辑器。

  css-modules ——css模块化解决方案
  css-modules不是为react而生的，它是css模块化的一种解决方案，但它和react配合使用非常的契合。
  css的发展，已从最原始的css,到后来的less/sass，再到postcss，以及css in js，再到css-modules。无一不是向着模块化进发，甚至于有没有可能发展到组件化style，还有待实践和考察。

  React Devtools——react调试工具
  ef="https://github.com/facebook/rea...">react-devtools是facebook推出的一款调试工具。可有助于提高你的react应用开发效率。

  Babel——es6/7
  开发react应用，推荐使用babel搭建es6/7开发环境。这样你就可以尽情地使用高逼格兼高效率、高体验的es6/7语法了。
  无论是react，还是redux，还是Immutable等等，甚至是不相关的rx.js，都强调函数式编程。
  说句题外话，整个react体系，都在强调js，甚至连css(css-modules)、html(jsx)都融入了js处理，所以提高你的js驾驭能力可使你在前端路上不会迷失。

  TypeScript
  TypeScript是微软开源的JavaScript 的一个超集，主要提供了类型系统和对 ES6 的支持。而TypeScript也因此成为了angular的一个制胜关键点。
  那react能否使用呢？当然是可以的。
  蚂蚁金服所开源的React UI库ant design就是使用TypeScript来开发的。由此可知其强大。
  顺便提供几个TypeScript的学习教程：官方文档、简易中文版、中文版教程

- 5，react项目构建
  前端构建工具有很多种，比如最为流行的webpack、百度开源的fis3、以及 *gulp*。而开发react应用，推荐使用强大的webpack做项目构建。这也是官方的推荐。

react的web技术栈的选择

通过上面可以了解到，react的web技术栈非常的丰富，搭配不同的路由、状态管理器、UI库，构建工具等不同的组合，可整理出二十几种技术栈方案。

下面来说说根据不同应用场景，最为推荐的几种技术栈方案。

1，开发后台应用

- react+react-router+mobx+webpack+bfd-ui/ant design （三星半）
- react+react-router+redux+webpack+bfd-ui （三星）

- react+react-router+redux+webpack+ant design （四星）
- react+dva+bfd-ui （四星半）
- react+dva+ant design （五星）

2，开发前台web应用

- react+dva+bfd-ui/ant design （四星）
- react+dva+material-ui/react-toolbox （四星半）

二，react生态之——移动端

react不仅在web端占据主流的位置，同时在移动端表现尤为突出。这里不得不提到RN，也就是 *ef="https://github.com/facebook/rea..."*>*react-native*。

ef="https://github.com/facebook/rea...">*react-native是目前最优秀的非原生开发移动框架，一处开发，多端使用。同时具有出色的性能，支持热更新等超强的优势，使得ef="https://github.com/facebook/rea..."*>*react-native顿时站在风口浪尖*。

vue和angular在移动端同样有建树，分别是weex和ionic+cordova。然其综合性价比仍不如 *ef="https://github.com/facebook/rea..."*>*react-native*。

而最近facebook推出React Fiber 架构,使用Fiber对react核心算法进行重写，届时RN的性能将会再次直线式的上升，向原生步步紧逼。

开发RN应用所用的技术栈与web端大致相同，同样需要结合redux,react-router, dva, mobx等周边生态来使用。

另外也有一些适合RN的移动端UI库。比如ant design的mobile版——*ant-design-mobile*，以及响应式的 *material-ui*。

三，react生态之——服务器端

react不仅涉足web端，移动端，同样在服务器端也有非常好的涉猎。
react在服务器端的实践有两部分，一个是服务器端渲染，另一个就是强大的*graphql*。

react服务器端渲染

react提供了两个API来实现服务器端渲染，分别是——renderToString 和 renderToStaticMarkup。

而对于react服务器端渲染实践最为出色的莫过于*next.js*。这是一个基于react可实现服务器和浏览器都能渲染的框架。

除了*next.js*，还有一个比较出色的案例 ——*react-server*。它同样实现 了React 框架在服务器和浏览器中进行快速渲染和无缝切换。

除了以上这两个服务器端渲染的框架外，还有一些比较好用的实现react服务器端渲染模块，比如如"https://github.com/reactjs/express-react-views">express-react-views，*react-view*等等。

GraphQL——超前，另类的API

*graphql*是facebook开源的应用层查询语言。它很超前，时髦，可适用于包括nodejs，java,php等绝大都数后台语言。

它超前到什么程度？举个例子，我们只需要使用一个 GraphQL 的接口，即可满足一个简单博客网站的所有需求。有没有觉得很神奇？准确地说graphql是为接替RESTful而生的。

RESTful是当下非常流行的，主流的一套前后端API交互设计规范。几乎绝大都数互联网公司都在使用。那么为什么还需要*graphql*来接替RESTful呢？

其实，现在谈*graphql*替代RESTful还为时尚早，因为*graphql*还有很多问题需要解决，而即使真到了那一天，RESTful也仍有一定的市场空间。

RESTful本身存在的一些缺点和不足，比如，当需求或数据发生变化时，需要建立新的接口来适应变化，而不断添加的接口，会造成服务器代码的不断增长，即使通过增加接口版本，也并不能够完全限制服务器代码的增长。另外不断地增加接口，意味着将带来更多的开发工作，而且每个接口基本都无法复用等一系列问题。

*graphql*就是为解决这些问题而生的。
下面来看看一个简单的GraphQL请求:

```
{
  latestPost {
    _id,
    title,
    content,
    author {
      name
    },
    comments {
      content,
      author {
        name
      }
    }
  }
}
```

而请求的结果是这样：

```
{
  "data": {
    "latestPost": {
      "_id": "03390abb5570ce03ae524397d215713b",
      "title": "New Feature: Tracking Error Status with Kadira",
      "content": "Here is a common feedback we received from our users ...",
      "author": {
        "name": "Pahan Sarathchandra"
      },
      "comments": [
        {
          "content": "This is a very good blog post",
          "author": {
            "name": "Arunoda Susiripala"
          }
        },
        {
          "content": "Keep up the good work",
          "author": {
            "name": "Kasun Indi"
          }
        }
      ]
    }
  }
}
```

很明显，GraphQL是从客户端业务维度出发的，当客户端需要某些字段的数据时，只需要发出这些字段的GraphQL请求即可。按需索取，可复用，可定制化，灵活性很强。

这会不会就是未来前后端交互的一种趋势呢？我们拭目以待。

Relay 和 Apollo Client

上面已经有讲到， *relay*是facebook出品的一个前端数据框架。

我们使用*graphql*，为什么需要用到 *relay*呢？

首先，graphql是在后端实现的，准确地说是在后端搭起一个graphql服务器，它不会主动推送数据给客户端，也无法像RESTful那样由客户端发起一个ajax请求来请求后端的RESTful API。graphql需要在客户端有一样东西能与它进行交互，但不是ajax或者fetch，而是经过封装的如*relay*或apollo等能够与graphql服务器进行交互的前端数据框架。

*relay*就是这样应运而生。但实践中会发现使用relay来配合graphql使用通常会遇到很多坑，relay操作起来并没有想象中的简单，除了比较复杂以外，它同时存在很多局限性。

于是，不得不提到apollo。它是一个全功能的 GraphQL 客户端，用于 React 、Angular 等的交互，允许你轻松通过 GraphQL 获取数据并构建 UI 组件。

使用apollo+*graphql*，它并没有relay那样的繁琐，也没有relay那样的局限性。无疑和graphql是最配的。

关于*graphql*的体验，很多用过的人表示，爱不释手。

下面是一些关于*graphql*的比较好的文章和资料：

- graphql官方文档
- Apollo Client官方文档
- relay官方文档
- graphql学习资料收集
- Node.js 服务端实践之 GraphQL 初探
- 深入理解 GraphQL

四，react生态之——VR领域

目前，VR(虚拟现实)是科技界的新生事物，也是一大革命，在未来，VR将很大程度地改变人类的生活方式。

试想一下，VR将为你呈现一个虚拟现实世界，你将可以使用VR身临其境般地玩游戏，看新闻，购物，社交娱乐，看世界名景等等，这将是一个全新的世界。

那么，react难道有涉足VR领域吗?

是的，没错。

react不仅走在web端、移动端以及服务器端的前沿，还很超前地在VR领域布了个局——react-vr。

react-vr——webvr框架

不得不说facebook是一家技术含量很高，着眼长远的科技巨头。

下面主要讲讲webVR。

webVR是VR技术在浏览器的实现。

目前VR的技术实现方案仍在起草阶段，但有些技术方案几乎是铁定的。比如作为如底层的OpenGL、WebGL、three.js。

而目前就有一些基于底层技术开发出来的webVR框架，比如——*react-vr*和*aframe*。

*aframe*是由mozilla于15年开源的，目前已更新到0.5.0版本。

react-vr 于2017年4月18日举行的facebook F8开发者大会上正式发布，并在github放出了第一个开源版本—— react-vr。并号称——使用react即可创造出惊人的360°景象和VR应用。

react始终是走在web巅峰的路上。

以下是参考资料：

- aframe官方文档
- react-vr官方文档
- three.js官方文档

五，react生态之——全平台的reactxp

reactxp是微软Skype团队开源的一个基于react和react native开发的全平台框架，它不仅支持Android和iOS，还支持web和windows，目前还尚不支持mac。是眼下跨平台最广的一个框架，准确来说是一个js库。

reactxp的出现算是微软在wp跨平台计划失败后的又一次跨平台实践。是否也预示着前端终将会实现大一统呢?拭目以待。

reactxp目前还很嫩，还有很多问题需要解决，比如它的某些API只有ios能用，并不能做到全兼容，像浏览器一样存在兼容性问题，这只是其一。

reactxp真正要火起来还有很长的路要走，但不妨碍我们可以继续关注它。也许它能超越RN，成为最前沿的跨平台应用框架也说不定。

总结