## 一、前端开发中会有哪些问题需要考虑

讨论这个问题我觉得应该回到问题的本质- 【前端开发会考虑些什么问题】，这些问题即是前端开发过程中的痛点也是难点，了解了这些问题才能知道为什么会有 react 出现，以及 react 如何解决这些问题的。

首先，对于一个前端团队来说，在进行前端技术规划的时候都应该考虑的事情：

1. 组件库、模块化
2. 开发效率
3. 运行效率
4. 可维护性
5. 体验优化

### 组件库、模块化

首先是组件库，任何一个前端业务团队都会做的事情就是沉淀组件，公共基础组件，业务组件，函数工具库，这对于业界的前端来说是共识。 组件库也就是轮子库，是提高团队开发效率的最好方式，同时也是团队的基础沉淀（拿 kpi 的绝佳帮手）

然后是模块化，在几年前，经常会看到一个 js 几千行的情况，但是基于可维护性和重用性的考虑，会把js 拆分成模块，模块化的需求已经很普遍，出现了很多如 `amd` `cmd` `commonjs` `umd` 这些规范，以及 `require.js` `seajs` `browserify` `webpack` 这些工具和库来解决这些问题。

### 开发效率

开发效率是前端团队对业务响应速度的反馈，如果一个业务交给前端团队过后几个月都没有结果那必然会引起上下游的不满， 不管技术做的多棒，选什么框架，最终的目的都是完成业务。 那哪些因素会影响开发效率呢？

1. 业务代码架构设计
2. 可重用模块和组件

第一点是业务代码的架构设计，好的设计能够极大的减少代码量和出 bug 的可能。 第二是拥有大量可重用的模块和组件，能够快速的实现交互

### 运行效率

运行效率是用户体验的关键，对于对效率要求极高的业务场景来说，这可能是选择框架的第一标准

### 可维护性

前端开发中大多数在做的事情是：

1. 新业务加功能
2. 改版
3. 解决 bug特别是在大公司的前端更是体会深刻，可能重来没有做过新业务，都是在维护旧的代码，填坑加埋坑。 如果业务代码设计差，可阅读性差，很难定位 bug。 特别是千奇百怪的 mvc 设计，大控制器，复杂的 model ，想要定位出哪里出了问题真是一件 eggache 的事情。

### 体验优化

体验已经成了现代化前端开发的必谈之物，所以出现了当页面应用(spa)，instant loading，application shell]，progress webapp 这些名词。

## 二、目前解决这些问题的技术方案

组件化：webcomponent、polymer、x-tag、react、jquery-plugin、angular-directive

模块化：webpack、browserify、require.js、sea.js

开发效率：mvc(backbone) < flux(react) < mvvm(angular.js、vue、ember.js)

运行效率：backbone、react

可维护性：flux、redux

现代化的一些框架几乎都包含组件化的考虑，不过在其他方面各有其优势，关键点是在开发效率和运行效率之间的平衡

## 三、react 技术栈对上述问题的解决

注意我这里提的是 react 技术栈，并非题主说的 react，个人认为在描述 react 的时候应该是在讲 react 生态体系，那对于上面说的难点痛点在 react 中一一对应的解决方案。

组件化：react 天生组件化，这是 react 的核心，除了能够在团队内部积累业务组件以外，也能找到众多开源组件的实现

模块化：基于 webpack 可以使用 es6 或 commonjs 的写法实现模块化代码

开发效率：react 的代码基本就是组件的组合，分而治之的方式让代码的可阅读性很高，容易理解。 而且相比于 mvc 几乎是去除了 controller 的角色，只用关心一个 render 函数，不用关系视图局部的修改。

运行效率：react 实现了 virtual dom ，相比于 mvvm 框架具有更优的效率 可维护性：react 基于 flux 或 redux 的架构设计，确定性的 store 很容易定位问题，无论是新增业务代码还是查找业务 bug 都不再是难题体验：基于 react 可以很容易的实现 spa （react-router）;

组件化是目前所有框架类库中做得最好的。react能将所有数据都对应页面的一个节点（当然false, undefined, null, true会被忽略掉，或变成一个注释节点做占位符）。编写组件非常简单，这依赖于其强大的jsx dsl。 生命周期非常强大，并且足够用。

性能，对于react、angular、jquery，我们在pc和mobile端都做过对比测试，react是唯一的update时能优于create的框架，vdom的作用毋庸置疑。甚至将性能最差的angular结合react，性能也能获得提升。

虚拟dom页面改变特别快，减少页面的重绘和回流

1、复杂的状态更新；  
2、组件化开发：适可而止，不可过度组件化；  
3、提升了开发效率；  
4、虚拟DOM解决了以往大量操作DOM带来的性能问题；  
5、diff算法：树形结构不同直接移除再新增节点，不去徒劳的进行比较，简单粗暴高效。