

02

OPEN ORIENTED

凹凸实验室

# 高性能React替代方案——Nerv

[aotu.io](http://aotu.io)



02

OPEN ORIENTED  
凹凸实验室

京东  
商城用户体验设计部  
多终端研发部





JD.COM首页及核心频道

# 过往的技术架构

# 过往的技术架构



# 过往的技术架构



风格不一



维护困难



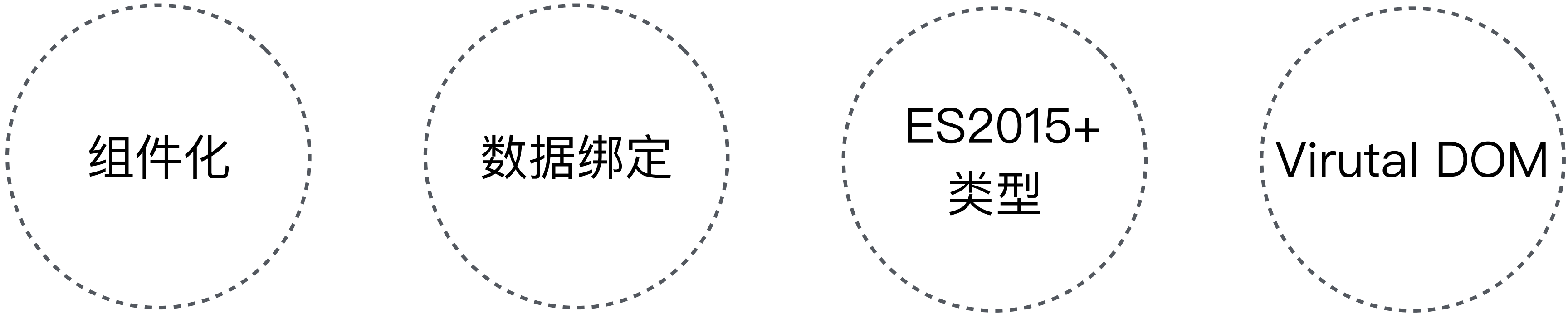
DOM操作



技术过时

**那么，大家都在玩什么？**

# 大家都在玩什么



组件化

数据绑定

ES2015+  
类型

Virutal DOM



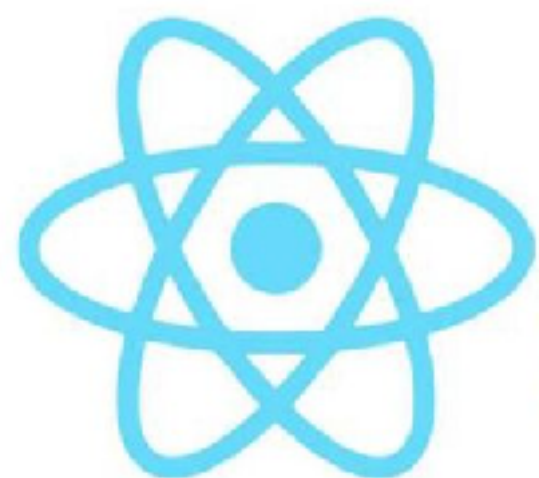
大家都下班了在愉快玩耍

我们却在

加班！

升级一下？

# 前端三大框架





框架对比

框架	版本	star	兼容性	性能 *	体积 *	开源协议	学习成本 *	迁移成本	源码类型
React	16.1.0	83931	IE 9+	1.52	31.9 Kb	MIT *	中	低	Flow
Angular	5.0.0	31250	IE 9+	1.46	61.7 Kb	MIT	高	高	TypeScript
Vue	2.5.0	77368	IE 9+	1.29	23.5 Kb	MIT	低	中	Flow

# 迁移成本

React Native 三端通用

开启更多可能性

# 学习成本

团队 React 经验丰富

熟悉相关技术栈



# 生态



## React 最佳

React 是最适合我们的

老板说

“既然如此，那用起来吧”



那就，评估一下吧...

# 风险

线上项目有 bug

框架的作者没空回答我们的问题怎么办？

万一开源协议又改了怎么办？

# 体积

研究表明

首页打开速度每增加 100 毫秒，  
网站销售量就会降低 1%。



# 兼容性

IE 8 全年贡献GMV

10,000,000,000+

老板又说

“技术要升级， 但……”

不能有损于业务

**考虑来造个轮子吧!**

**我们的目标...**

我们的目标

# 前端工程化



要接入现代前端开发流程



我们的目标

**快速定位 Bug**

当线上项目突然有问题  
不再惊慌失措 Google 百度  
直接撸起袖子干源码！

我们的目标

# 增强技术信心

如果能造一个业内领先的框架

那构建一个业内领先的业务

自然也不在话下了

我们的目标

**提升业务性能**

控制底层实现

搞定精细的性能需求

解放做业务的同学

我们的目标

# 提升团队影响力

加强团队的归属感

吸引更多牛人加入

~~加薪!~~

我们的目标

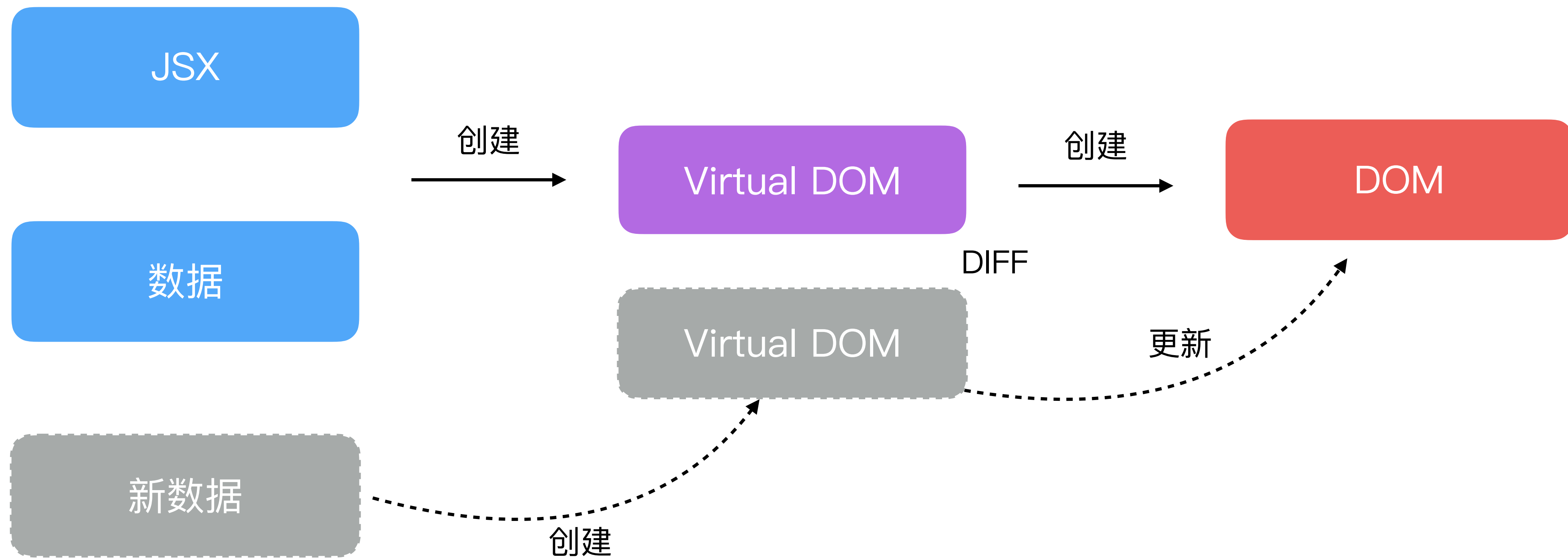
提升开发效率

~~早点下班!~~

我们热爱工作

开始造!





## JSX

```
<div id="one">  
  Hello, World!  
</div>;
```

转换



## Hyperscript

```
h('div', { id: 'one' },  
  'Hello, World!'  
);
```

```
1 /** @jsx h */
2
3 <foo />;
4
5 <foo bar="baz" one={1} />;
6
7 <ul>
8   <li>One</li>
9   <li>Two</li>
10 </ul>;
11
12 let Something = 'foo';
13 <Something a="b" />;
14
```

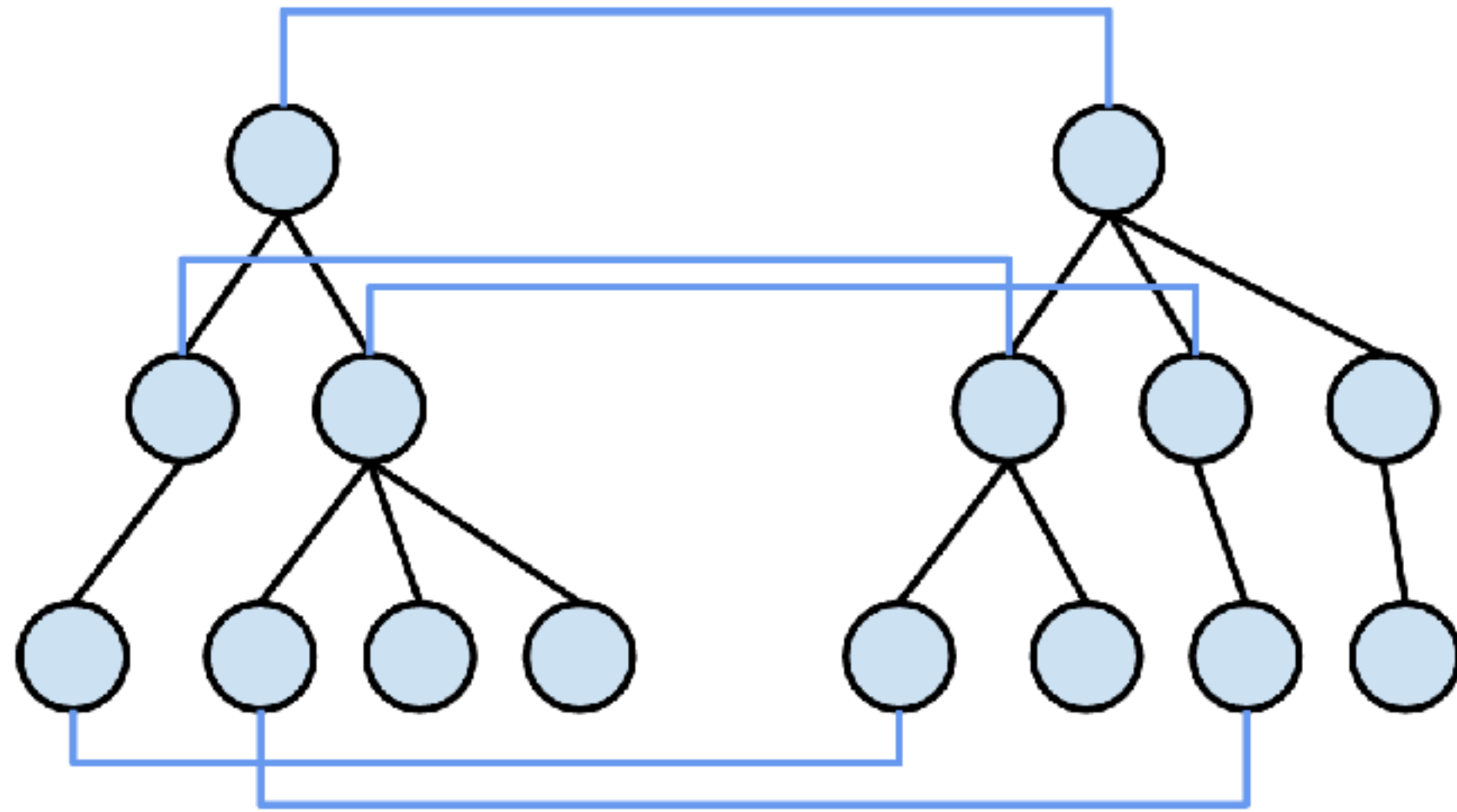
```
1 "use strict";
2
3 /** @jsx h */
4
5 h("foo", null);
6
7 h("foo", { bar: "baz", one: 1 });
8
9 h(
10   "ul",
11   null,
12   h(
13     "li",
14     null,
15     "One"
16   ),
17   h(
18     "li",
19     null,
20     "Two"
21   )
22 );
23
24 var Something = 'foo';
25 h(Something, { a: "b" });
```

```
function h (  
  nodeName, // String | Function  
  attributes, // Object | null,  
  ...children // Rest | parameters  
) {  
  return { nodeName, attributes, children }  
}
```

Virtual DOM



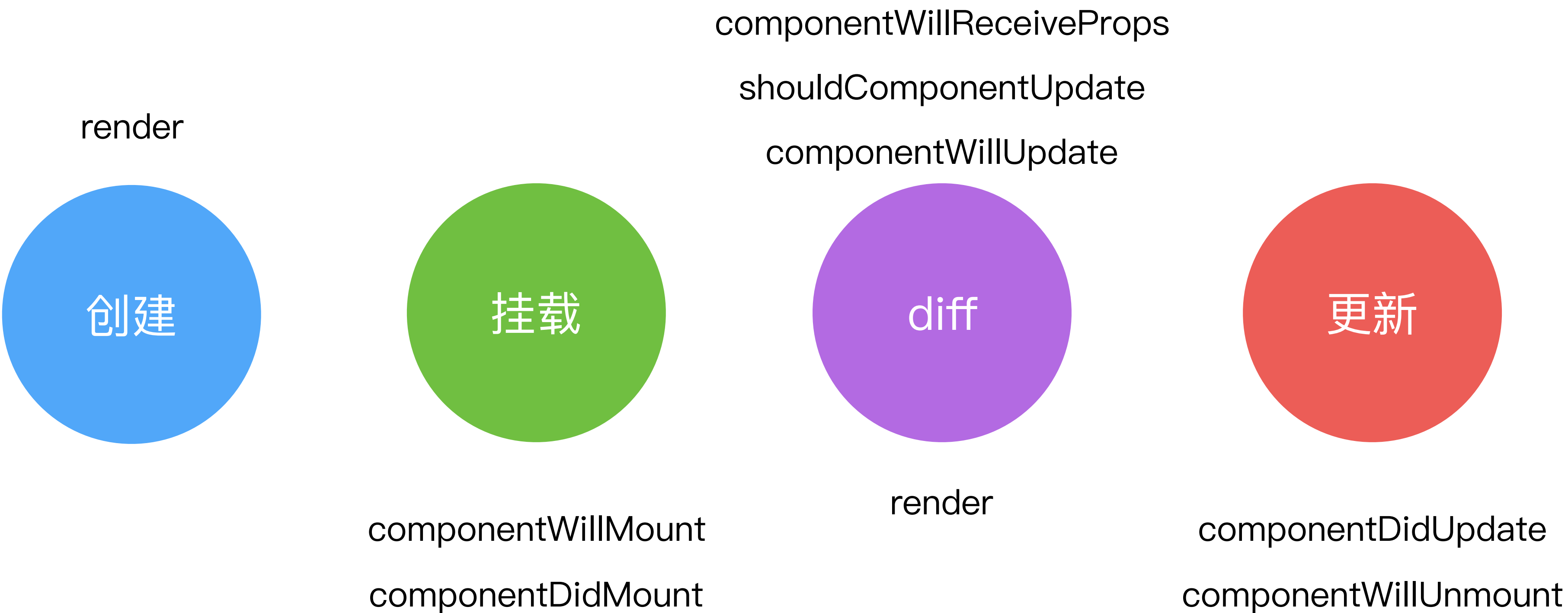
比较 virtual dom tree



Virtual DOM tree diff

时间复杂度  $O(n)$

# 组件系统运行机制



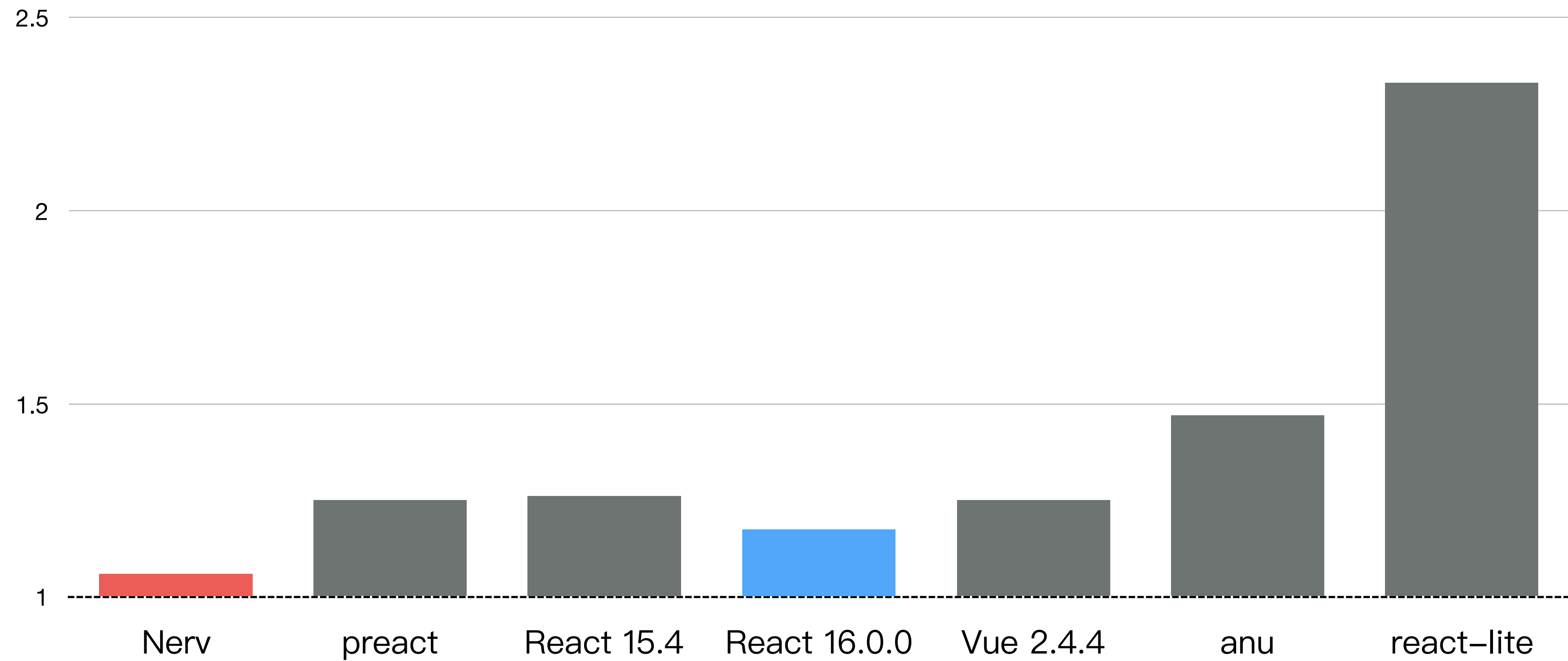
Nerv.*Alpha* release

**我们还能做些什么？**



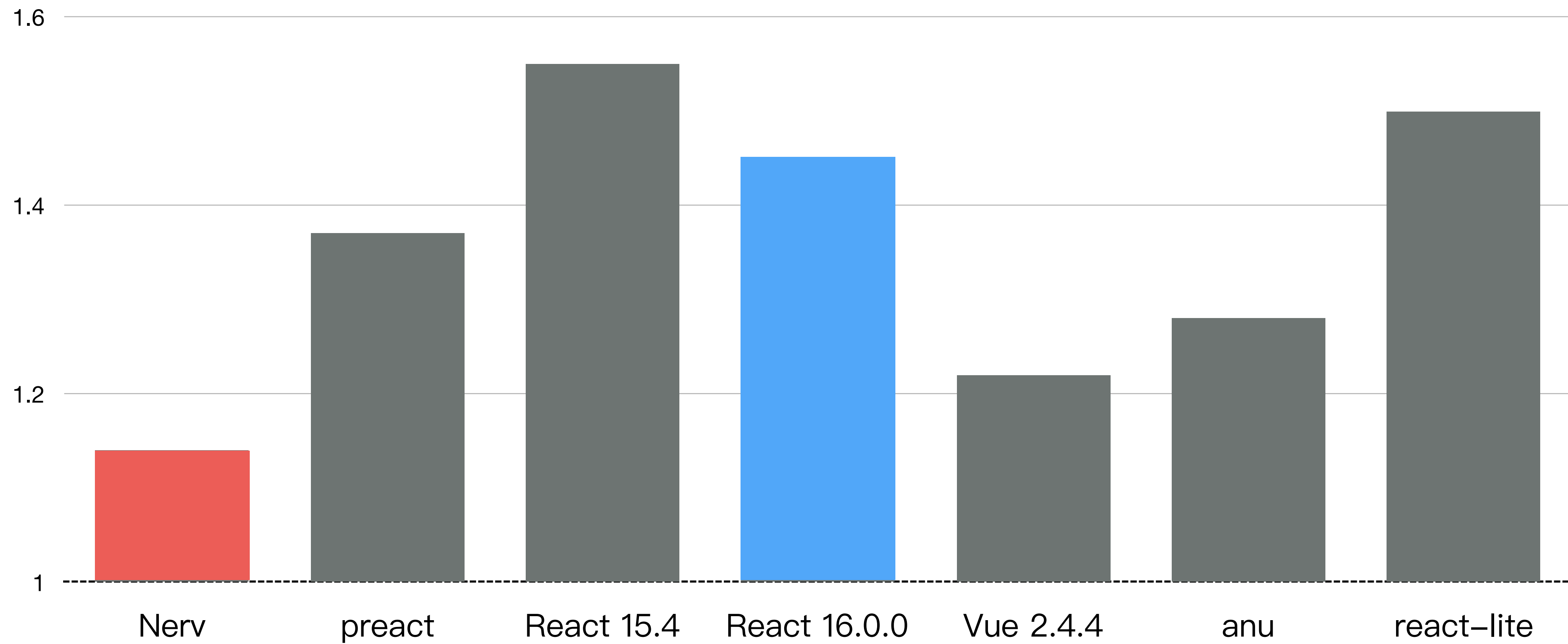
高性能？

## JS Framework Bench



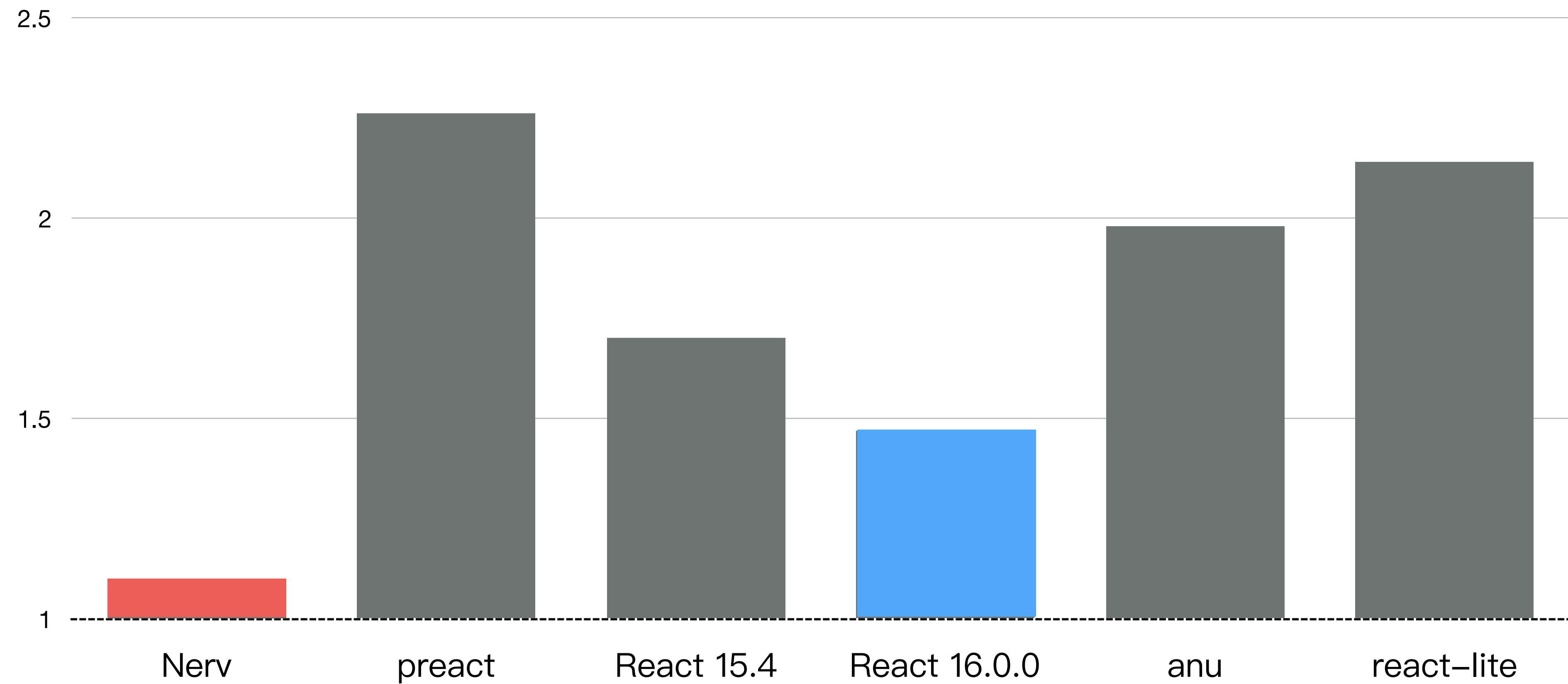
<https://github.com/krausest/js-framework-benchmark>

## Memory usage



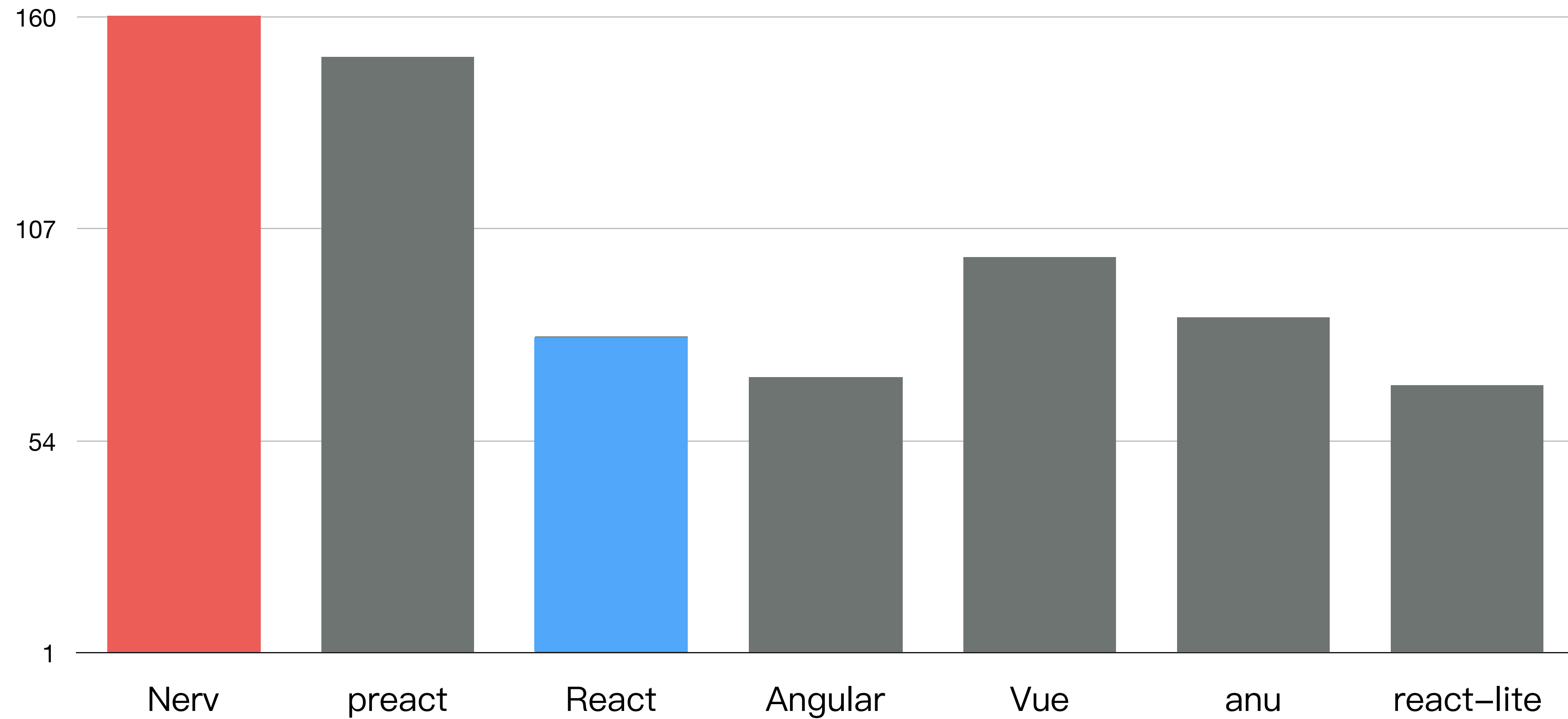
<https://github.com/krausest/js-framework-benchmark>

## UI Bench



<https://github.com/localvoid/uibench>

DB Monster (fps)



<https://github.com/mathieuancelin/js-repaint-perfs>

**怎么做到的？**

ECMAScript 标准

操作次数

`speed = compiler(ecma(fn), JIT?) * operation(s)`

编译器

是否优化, ~~如何优化~~



ECMAScript 标准

操作次数

`speed = compiler(ecma(fn), JIT?) * operation(s)`

编译器

是否优化



# ECMAScript 标准



## ECMAScript 标准

```
array.forEach( () => {  
    doSomething()  
})
```



When the **forEach** method is called with one or two arguments, the following steps are taken:

1. Let *O* be ? **ToObject**(**this** value).
2. Let *len* be ? **ToLength**(? **Get**(*O*, "length")).
3. If **IsCallable**(*callbackfn*) is **false**, throw a **TypeError** exception.
4. If *thisArg* was supplied, let *T* be *thisArg*; else let *T* be **undefined**.
5. Let *k* be 0.
6. Repeat, while *k* < *len*
  - a. Let *Pk* be ! **ToString**(*k*).
  - b. Let *kPresent* be ? **HasProperty**(*O*, *Pk*).
  - c. If *kPresent* is **true**, then
    - i. Let *kValue* be ? **Get**(*O*, *Pk*).
    - ii. Perform ? **Call**(*callbackfn*, *T*, « *kValue*, *k*, *O* »).
  - d. Increase *k* by 1.
7. Return **undefined**.

ECMAScript 标准

操作次数

`speed = compiler(ecma(fn), JIT?) * operation(s)`

编译器

是否优化

```
// 判定一个 DOM 节点是否为文本节点(TextNode)
```

```
// slow
```

```
if (dom.nodeType === 3) { ... }
```

---

```
> Object.getOwnPropertyDescriptor(Text.prototype, 'nodeType')
```

```
< undefined
```

---

```
> Object.getOwnPropertyDescriptor(Node.prototype, 'nodeType')
```

```
< ▶ {get: f, set: undefined, enumerable: true, configurable: true}
```

---

```
> |
```

```
// 判定一个 DOM 节点是否为文本节点(TextNode)
```

```
// slow
```

```
if (dom.nodeType === 3) { ... }
```

```
// better, but not perfect
```

```
if (dom.splitText) { ... }
```

## 隐式类型转换

a + b

类型不确定  
————→

```
mov eax, a  
mov ebx, b  
call RuntimeAdd
```

Slow

都是 int  
————→

```
mov eax, a  
mov ebx, b  
add eax, ebx
```

Fast



隐式类型转换

TypeScript

```
// 判定一个 DOM 节点是否为文本节点(TextNode)
```

```
// slow
```

```
if (dom.nodeType === 3) { ... }
```

```
// better, but not perfect
```

```
if (dom.splitText) { ... }
```

```
// good
```

```
if (dom.splitText !== void 0) { ... }
```

ECMAScript 标准

操作次数

`speed = compiler(ecma(fn), JIT?) * operation(s)`

编译器

是否优化

## 编译器是否优化

```
// 包含需要审查的用法的函数（这里是 with 语句）
function containsHeight() {
    return 3;
    // with({}) { } // with 注释掉之后，又可以优化了
}
// 以下为检验工具
function printStatus(fn) {
    switch(%GetOptimizationStatus(fn)) {
        case 1: console.log("is optimized"); break;
        case 2: console.log("is not optimized"); break;
        case 3: console.log("is always optimized"); break;
        case 4: console.log("is never optimized"); break;
        case 6: console.log("is maybe deoptimized"); break;
    }
}
// 告诉编译器类型信息
containsHeight();
// 为了使状态从 uninitialized 变为 pre-monomorphic，再变为
// monomorphic，两次调用是必要的
containsHeight();
%OptimizeFunctionOnNextCall(containsHeight);
// 下一次调用
containsHeight();
// 检查
printStatus(containsHeight);
```

%GetOptimizationStatus

%OptimizeFunctionOnNextCall

编译器是否优化

```
$ node --trace_opt --trace_deopt --allow-natives-syntax test.js
```

## 常见拒绝优化函数

对象字面量包含\_\_proto\_\_, get, set 声明的函数

直接给 arguments 赋值的函数

包含 eval, with, debugger 语句的函数

for..in 语句中 key 不在本地作用域中的函数

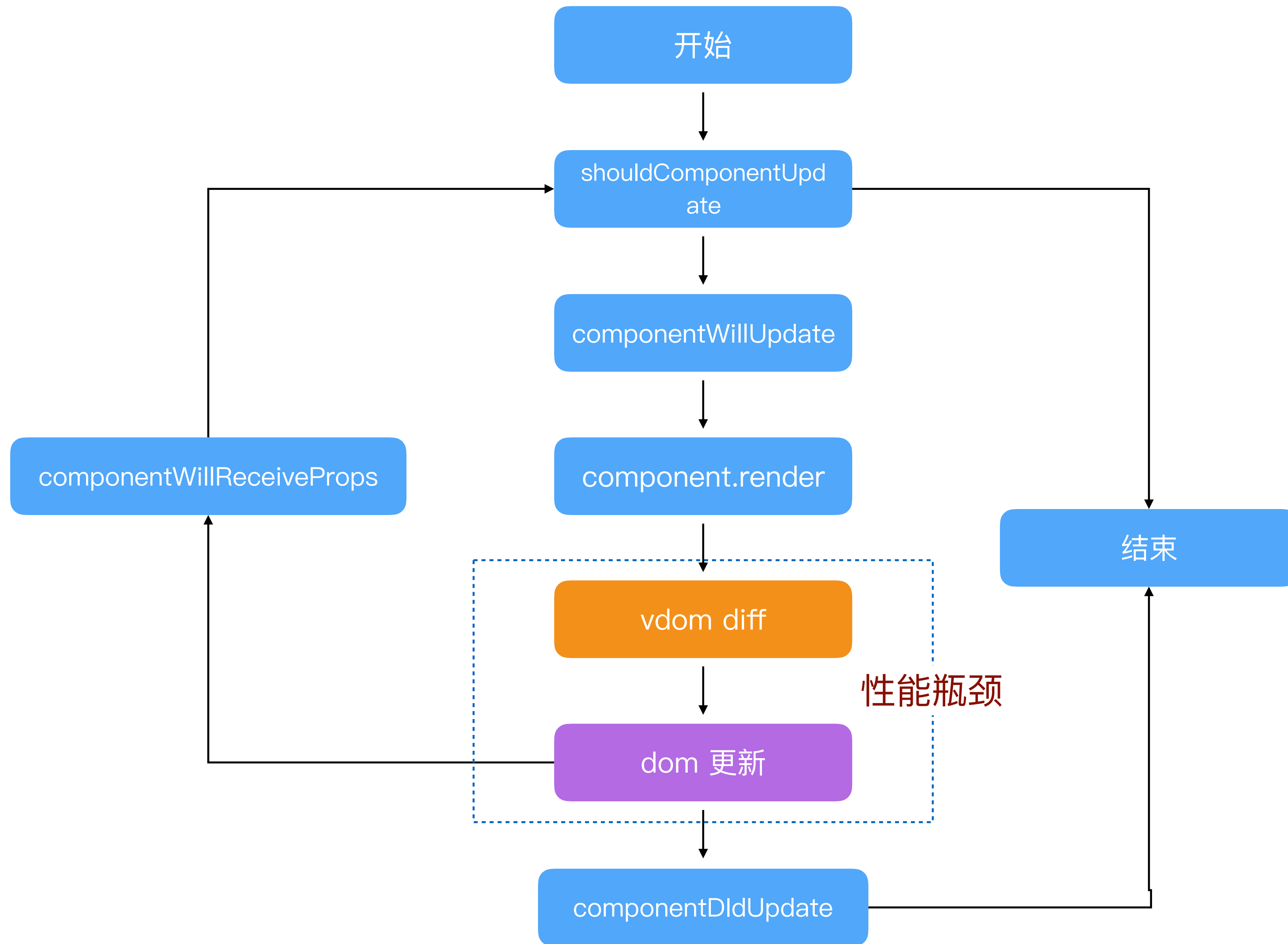
ECMAScript 标准

操作次数

$$\text{speed} = \text{compiler}(\text{ecma}(\text{fn}), \text{JIT?}) * \text{operation}(\text{s})$$

编译器

是否优化





优化 diff 算法，降低运行次数

## diff算法升级

ListA

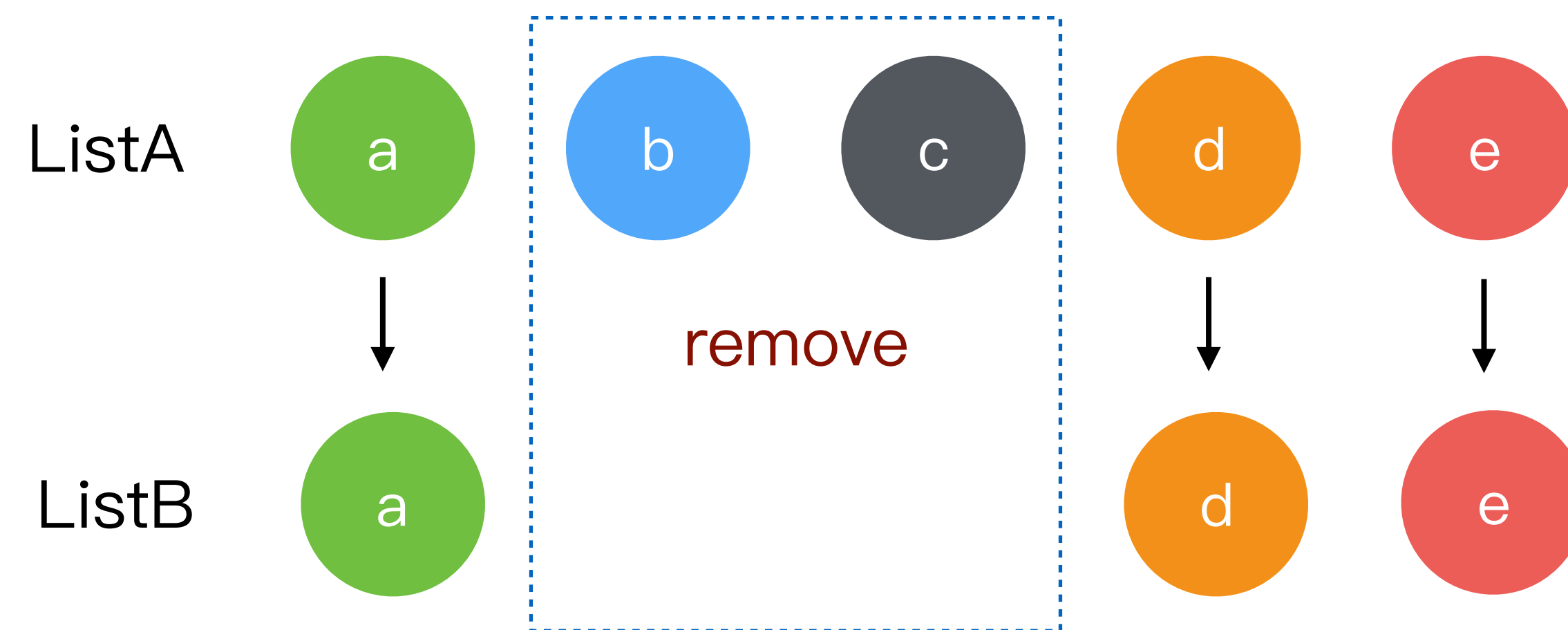


ListB



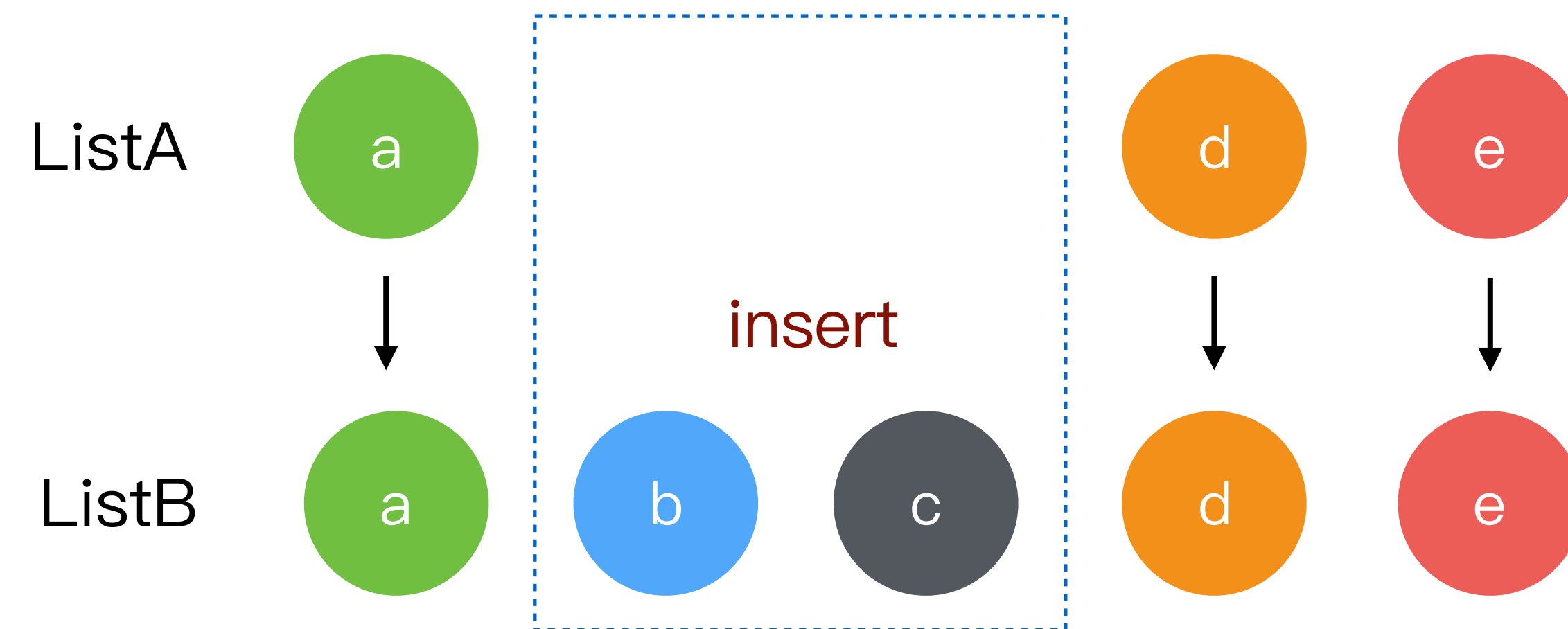
找到最少的DOM操作序列

## diff算法升级



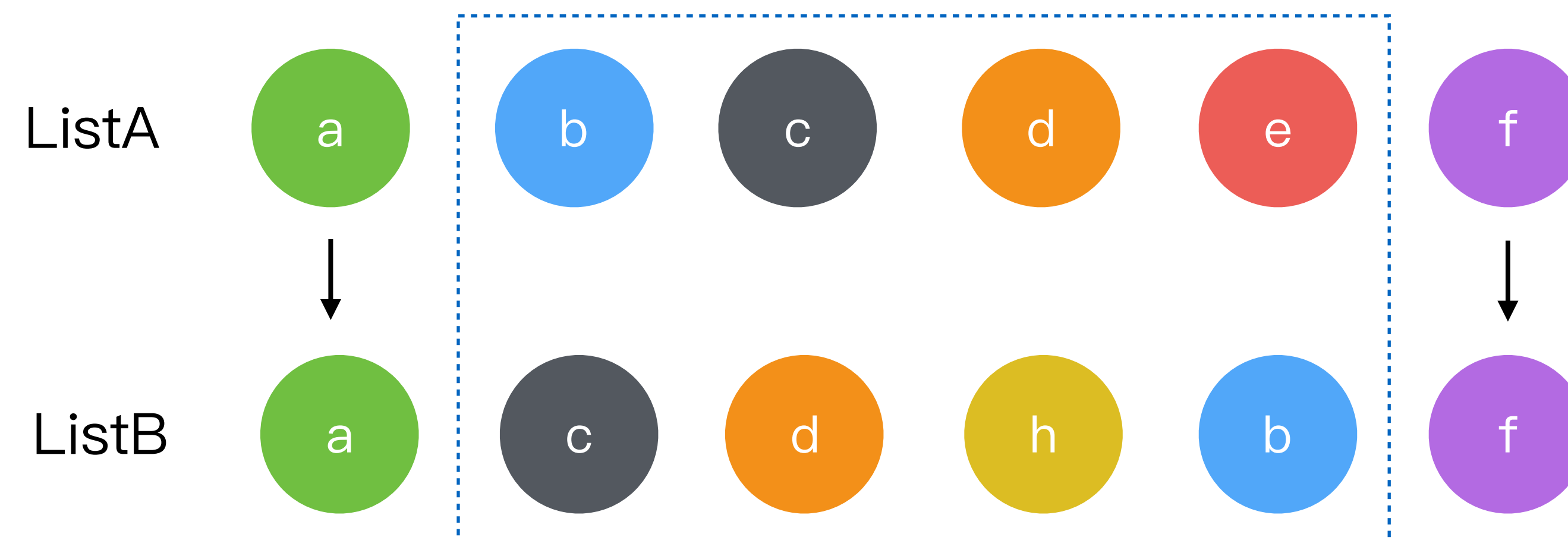
比较头尾，找到相同元素

## diff算法升级



比较头尾，找到相同元素

## diff算法升级



比较头尾，找到相同元素

## diff算法升级

ListA



ListB



找到被删除、插入以及是否有被移动过的元素

git.io/nerv

Nerv.RC release

似乎还差点什么



# 生态建设

生态

nerv-redux

nerv-devtools

nerv-server

nerv-test-utils

生态

react-router

Mobx

无缝对接React生态

react-redux

各种react组件库

.....

## 工具



- 自动生成项目、模块、组件
- 零配置开发
- 久经考验的编译预览功能
- 基于webpack的高度扩展功能

# Nerv 1.0 release

**这次就完整了**



**Nerv 1.0**

React16功能支持

轻量、可嵌入

测试覆盖率95%+

SSR

HMR

TypeScript

合成事件

高效 Diff 算法

PWA

**于是， 我们愉快地用起来了**



JD.COM首页及核心频道



TOPLIFE

TOPLIFE奢侈品PC全站





OPEN ORIENTED

凹凸实验室



[aotu.io](http://aotu.io)