React是前端最受欢迎的框架之一，解读其源码的文章非常多，但是我想从另一个角度去解读React：从零开始实现一个React，从API层面实现React的大部分功能，其包含：

1.火热的0配置的打包工具parcel

全局安装parcel

项目依赖配置

启动配置

2.封装JSX和理解虚拟DOM

ReactDom.render原理剖析

理解虚拟DOM

3.封装组件和生命周期

组件基类React.Component的实现

render方法实现

组件渲染实现

生命周期方法实现

4.diff算法对比

什么是diff算法

对比策略

对比文本、组件、非文本DOM、属性实现

对比子节点实现

5.异步的setState实现

真正的React中setState如何实现

合并setState实现

setState队列机制

清空队列

从0实现React

实现功能:

```
1.下载nodejs
2.下载脚手架:npm i create-react-app -g
3.创建项目:create-react-app react-test
```

0. 火热的0配置的打包工具parcel

安装babel插件,将jsx语法转换成js对象(虚拟DOM)

```
cnpm i babel-core babel-preset-env babel-plugin-
transform-react-jsx --save-dev
```

1. 封装JSX和理解虚拟DOM
2. 组件和生命周期
3. diff算法

diff算法

diff算法?what?什么玩意

如何减少DOM更新:我们需要找出渲染前后真正变化的部分,只更新这一部分.而对比变化,找出需要更新部分的算法称之为**diff算法**

对比策略

在前面我们实现了`_render`方法,它能将虚拟DOM转换成真正的DOM

但是我们需要改进它,不要让它傻乎乎地重新渲染整个DOM树,而是找出真正变化的部分进行替换

这部分很多类似React框架实现方式都不太一样，有的框架会选择保存上次渲染的虚拟DOM，然后对比虚拟DOM前后的变化，得到一系列更新的数据，然后再将这些更新应用到真正的DOM上。

我们会选择直接对比虚拟DOM和真实DOM，这样就不需要额外保存上一次渲染的虚拟DOM，并且能够一边对比一边更新，这也是我们选择的方式。

不管是DOM还是虚拟DOM，它们的结构都是一棵树，完全对比两棵树变化的算法时间复杂度是O(n^3)，但是考虑到我们很少会跨层级移动DOM，所以我们只需要对比同一层级的变化。

总而言之,我们的diff算法有两个原则

- 对比当前真实的DOM和虚拟DOM,在对比过程中直接更新真实DOM
- 只对比同一层级的变化

4. 异步的setState