## 自我介绍

一定要**自信**！**底气十足**！语调稍微**俏皮活泼**一点，不要像背书一样喔！保持**自信和大方**，保持一个良好的面试状态会给面试官留下很好的印象。

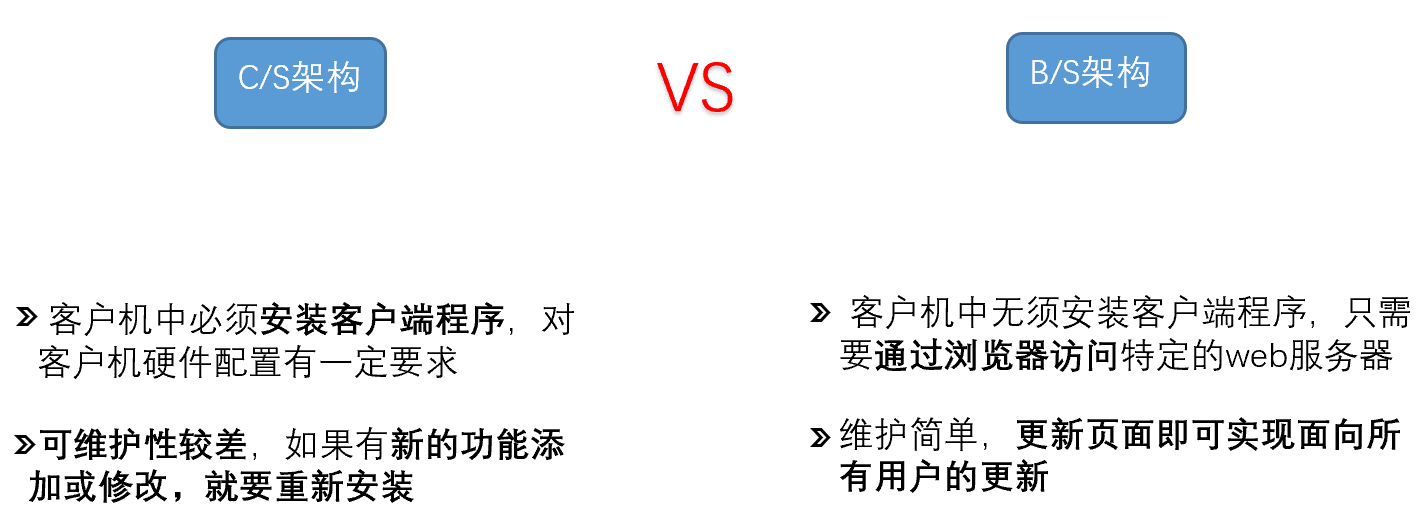
回答问题：www是什么、为什么、怎么样，即概念+优点+用法。其实最好是概念+优点+原理分析。还有不要局限为面试官问的东西，**多扩展**，**引导面试官提问**，还有切记听清楚了想好了再回答

### 1分钟版本

（面试之前调整好**积极乐观，自信，爱笑的状态**和情绪，**隐形加分！**）

面试官您好，我叫王慧兰，是来自浙江大学的**硕士研究生**，预计明年3月份毕业。本科就读于武汉的华中科技大学，2018年本科毕业后保送到浙江大学读研。

读研期间我做过两个项目，一个是和北京301解放军总医院合作的“高原病**患者管理及风险评估**系统”，**这个项目的背景**是设计一个系统来记录高原士兵的生命体征及进行患病风险评估。作为主要负责人，我设计了前端UI并用**React**框架独立进行开发，**使用**Redux管理系统的状态，引入Ant Design样式，(并扩展webpack配置，通过自定义配置antd按需加载，**将**开发环境下首次加载速度提升50%。)此外，我还用java语言编写部分后台代码。最终系统实现了**信息录入、患者管理及推理诊断**的功能。第二个项目是研究生刚入学时的一个软件技能培训项目，是要将一个基于C/S架构的winform版决策支持软件重构为基于B/S架构的web网页版，我的工作主要是先**分析原来桌面系统**的功能与应用逻辑，然后基于原始的**JavaScript**(Jquery)、HTML和CSS**进行Web页面开发**, **通过Ajax调用后端服务**，最终实现了页面动态效果与数据流动。



（面试官问起来再说选择前端的原因，见下）

### 3-5分钟版本

面试官您好，我叫王慧兰，是来自浙江大学的研二在读硕士生，研究方向是医学信息学和医疗软件工程，预计明年3月份毕业。本科就读于华中科技大学，本科期间先后获得过专业top5的国家励志奖学金和top1国家奖学金，2018年本科毕业后保送至浙江大学读研。

在读研期间，我做过两个项目。一个是和北京301解放军总医院合作的“高原生命刻画与决策支持系统”，这个项目的背景是设计一个系统来记录高原士兵的生命体征，比如在高原上表现出的各种症状，并根据这些信息对他们是否会患高原病进行风险评估以及给出干预建议。作为主要负责人，我先通过文献调研和请教医生确定系统需要录入的信息项，以及确定高原病诊断标准和治疗建议。然后明确系统的模块和功能，设计前端UI并用Angular框架独立进行开发，使用SpringBoot框架搭建后台服务并用java编写大部分后台代码。最终系统实现了信息录入、统计展示及推理诊断的功能，并交由301医院推广。

另一个项目是更早一些时候的一个研究生入学技能培训项目，要求将基于C/S架构的winform版决策支持软件重构为基于B/S架构的web应用。这个winfrom版应用是实验室师兄前几年开发的。我的工作主要包括先将原来基于C#的后台服务打包为web service工程，重写接口。然后设计新的web UI，并基于Jquery、HTML和CSS进行前端页面开发, 通过Ajax调用后端服务的Restful API，实现页面动态效果与数据流动。

在做这两个项目的过程中，我发现自己对前端开发有比较浓厚的兴趣和探索欲，看到自己写的代码变成网页显示在浏览器中时，会有较大的成就感。并且我会思考怎样让代码更加简洁规范和高效。使交互性更好，怎样更美观等等。这样的兴趣和热情让我决定以后想成为一名前端开发工程师。介绍完毕，请您指教。

## 非技术问题：

### 美团二面/三面

### 为什么选前端？（项目经历里既有前端又有后端）

×：喜欢思考怎样使交互性更好，使页面更美观。**这是UI设计师的工作**！

√：自己实现了一些视觉上可见的页面和交互，会比较有成就感，感觉很有意思。当然也具有挑战性，有时候遇到一些自己不太会实现的交互或者样式，我就会想办法去实现，比如在网上搜，看官方文档或者请教别人，在这个过程中，整个人的状态非常专注投入，感觉很充实，不觉得辛苦或疲惫，我很喜欢这样的状态。

**TODO：**还要再深入谈一下对前端的思考！

### 怎么学习前端的？（进一步）有没有系统地看过什么书？（推荐了高级程序设计3）

刚开始在慕课网上看了一些基础的关于js，html和css的教学视频，然后就开始动手做项目了，项目中遇到不会的问题或者发现了一些模棱两可的知识点，就会搜一些相关的博客来进一步学习。后面又看了《JavaScript dom编程艺术》这本书，在网易云课堂上跟了一个梳理ES6的教学视频，然后现在在看《HTML5与CSS3权威指南》，准备再看一下**javaScript高级程序设计3（面试官似乎都很认可这本书！）**。

### 你是怎么设计高原这个系统的？能不能具体说一下思路？

首先明确这个系统的大概需求。然后列举这个系统要实现什么功能，有几个模块，那调研后就确定下来主要有病人信息录入（包括增删改查），统计分析和推理诊断这几个大的功能模块，然后参考了一些网站的设计设计了系统的UI，弄清楚各个页面的内容和页面之间的跳转，然后就是写代码实现了。

### 做项目的过程中有没有遇到什么难点？如何解决的？（背的还不够熟）

印象比较深的就是系统的“新增患者”页面，就是**输入**一个新患者的信息并保存。

**首先是**表单渲染和提交的问题，刚开始的时候老师说加两个表就行了，一个表记录病人的基本信息，比如姓名，性别、年龄什么的。另一个表就用来记录他的症状，比如有没有有头晕，呼吸困难这些症状。因为这**些控件既有文本输入**框，又有Select选择和Switch开关，还有日期选择框，**有些是必填的有些又不是，**我刚开始就用了最笨的方法，把这些控件一个一个的单独添加到Form里。**而且由于要提交保存病人信息**，还给每个控件都绑定了一个onChange事件处理函数来记录每个字段的值。后来老师又说要再加一个表记录病人的一些**检查指标**，比如血红蛋白数目之类的，我本来打算还是按原来的方法加一个表，就把之前的表复制一下再改一改名字什么的，发现代码这时候看起来很多而且有点乱，而且我觉得老师的**要求可能还会再变**，**要记录的项会越来越多**，我这样的写法不太好，应该用更简洁高效的方式生成表单，提高**拓展性**，**万一要再加表**也比较方便。后来我就想到把这些要记录的表单项都放到一个数组里，并且同时记录它们的类型是input还是select，以及它们是否是必填项required，然后通过遍历的方式生成一个表的表单控件，在遍历的时候判断当前项的类型，如果是input，就添加input框，是select就添加select框，以此类推…所有的控件都绑定到同一个onChange处理函数就行了，传入各自的名字进行区分。**这样不仅整个代码简洁了很多，也提高了系统的拓展性，以后如果要再添加什么表单，也很方便，直接把表单项放到一个数组里就行了，用相同的模板来生成**。

还有一个点我纠结了更长时间。就是我这是个患者管理系统嘛，既可以新增患者，也可以在当前病人列表中选择一个已经存在的患者然后查看他的信息。新增患者和查看患者的信息其实是同一个页面，因为都是上面提到的基本信息，症状信息这几个表单。所以这个页面应该要实现新增的时候进入这个页面，各个表单应该是空白的，等着医生输入。查看信息的时候，我们查看哪个病人，表单就应该默认填充了这个病人的信息。听起来不难，可以设置一个currentPatient变量表示当前选中的病人，选中某个病人查看详情的时候，可以把currentPatient设置为这个病人，而准备新增病人的时候，这个变量就是空的，页面上的表单就根据curPatient来渲染。但是我这样做的时候并没有成功，当我选中某个病人并且成功赋值给了currentPatient之后，跳转到这个页面，表单还是空白的，并没有显示选中病人的信息。然后我就很困惑，看了半天也不知道到底哪里出问题了，我请教了实验室的师兄他也觉得很奇怪，然后就在网上找，又仔细看了ant design的文档说明，发现是因为我直接copy了官网的表单控件Form.item的写法，每个Form.item有个name属性，而**在ant design中被设置了name属性的Form.Item，表单控件会自动添加value和onChange，数据同步会被Form接管。**我又自己给表单控件添加了value和onChange，这种情况下，我自己设置的控件值value是不起作用的。因为我把name去掉就可以了，这样控件的值就由我自己设置的value来控制。当然我后面也实现了如果不去掉name属性，就按ant design的默认实现，要怎么解决我这个问题。

通过调这个bug，我发现平时在开发过程中，用这些框架也不能为了求快盲目地直接套用，这样肯定就会遇到一些莫名其妙的坑，还是要多看看文档相关的说明。

职业规划？

现在投了几家公司？有没有offer？

为啥没有投阿里、腾讯这些大厂？

### js和其他语言（如java）的区别

1. java是编译型语言，需要先编译再运行，javascript是脚本语言，在程序运行过程中被逐行地解释。
2. java是静态语言，定义变量时需要指明类型。javascript是动态语言，变量本身类型不固定。
3. java面向对象，javaScript则基于原型，在es6之前都没有class关键字，用函数来模拟类实现。
4. java主要运行在服务端，javaScript运行在客户端浏览器中。

6-9均为tp秋招提前批一面

### 成绩说一下？本科和研究生的（tp一面）

本科综合成绩是**第六**，其中大三的时候是专业第一，还拿了国奖。（不好意思地笑一下）大一大二就靠后一点，**第八第九**的样子。

研究生它现在还没有给出排名，不过我各科的分数在我同学里面算比较高的，应该也是**第五第六**的样子。

### 为什么选React？（其实是想问react的优点？基本都问了，然后问vue…）

（笑）最直接的原因是上一届的师兄师姐们用的是React，用React方便和他们交流。

当然我后面接触react的时候，发现React有些很明显的优点：

。。。

### 你这个加载时间优化具体是做了什么事情？

是这样的，刚开始引入ant design的时候引入了全部的antd样式，它的体积很大，就导致webpack打包的时候特别慢，我每次npm start启动应用要等很久才加载出页面。所以我就想改一下webpack的配置，但是React默认封装并隐藏了webpack的配置，后来是借助了react-app-rewired来对默认的webpack配置进行拓展，**使用babel-plugin-import插件实现了antd样式的按需加载**。

（一个对 create-react-app 进行自定义配置的社区解决方案）

### 你觉得打开一个页面需要十几秒算快还是算慢？开发环境是localhost，实际部署应该是还要往后延时的，因为生产环境下是通过网络访问，加载时间只会更长？（后面就换问题了，是想问生产环境下怎么优化首屏时间吗？）

### 占位

## React相关

<https://juejin.im/post/5cf0733de51d4510803ce34e#comment>

## 什么是React?

**声明式的**、**基于组件的**、**用于构建用户界面**的高效javascript库。

* 在react中通常采用JSX语法，是一个 JavaScript 的语法扩展。**结合了javascript和HTML，将标记和逻辑耦合在一起，生成可以在DOM中呈现的react元素**。元素是构成React应用的最小单元，它描述了你在屏幕上想看到的内容
* 组件：通常将应用程序的整个逻辑分解为**小的单个部分**。 我们将每个单独的部分称为组件，组件允许将 **UI 拆分为独立可复用的代码片段**。 通常，**组件是一个javascript函数，它接受输入，处理它并返回在UI中呈现的React元素**。

## React的特点/优缺点

1. jsx更加语义化且易懂
2. 单向数据流（好处是更容易追踪数据变化排查问题）
3. 一切都是component：**代码更加模块化，重用代码更容易**，可维护性高

## 什么是虚拟DOM及其工作原理？

详见：

<https://www.zhihu.com/question/29504639>（已收藏）

<https://www.jianshu.com/p/f2908d30ac78>（写在onenote中，可以打开）

**概念**：在React中，使用render方法来返回需要在网页中展示的内容，也就是用JSX语法描述的React元素，实际上在编译的时候Babel会把JSX转换成一个名叫React.createElement()的函数调用，这个函数会创建一个描述渲染内容的轻量级的JavaScript对象，这个javaScript对象就是虚拟DOM,也就是说虚拟DOM是对真实DOM的一个抽象。React DOM通过diff算法更新实际的DOM 来与虚拟DOM保持一致。

**工作原理**可以概括为三个步骤：

1. **用JS对象模拟DOM树**：一开始用 JavaScript 对象**表示** DOM 树的结构，也就是创建虚拟DOM, 然后根据虚拟DOM树构建一个真正的 DOM 树，插到文档当中。
2. **比较两棵虚拟DOM树的差异**：当状态变更的时候，重新构造一棵虚拟DOM**树**，然后将新的树和旧的树进行比较，记录两棵树的差异。算法实现：会对新旧两棵树进行一个深度优先的遍历，~~（这样每个节点都会有一个唯一的标记，）~~每遍历到一个节点就把该节点和新的树进行对比。如果有差异的话就记录到一个对象里面。
3. 最后把差异应用到真正的DOM树上，视图就更新了。

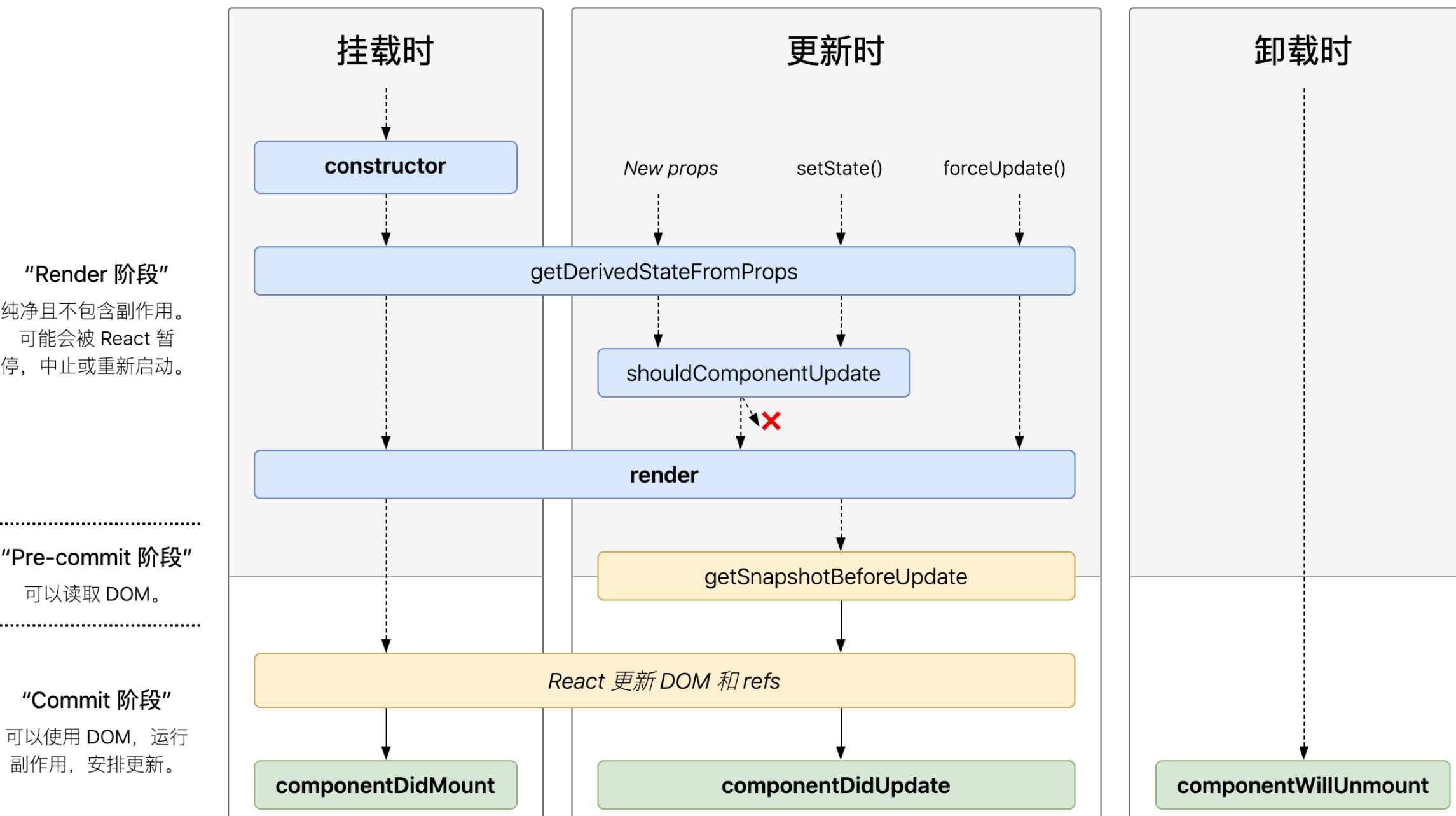
**优点**：

频繁地**直接**操作**实际**DOM来更新页面会很慢，导致性能变差，**而创建和操作普通js对象开销是很小的**，只需要在js对象模拟的虚拟DOM进行比较，再将更改的部分应用到真实的DOM树上就可以进行更新了。

## React生命周期

可以参考通俗易懂理解生命周期from微信：

图示：



生命周期分为**挂载**、**更新**和**卸载**三个阶段：

**挂载**阶段的生命周期方法按顺序依次为：

1. **constructor**: 构造函数，这是第一个被调用的方法，我们通常在构造函数里**初始化**state或者给自定义方法绑定this（在组件被挂载到 DOM 之前）
2. **getDerivedStateFromProps**: static getDerivedStateFromProps(nextProps, prevState),这是个静态方法, 以 props 和 state 作为参数，返回一个用于**更新**组件状态的对象，可以在组件被渲染之前更新它的状态。（这个方法允许组件基于 props 的变更来更新其内部状态，在组件被挂载到 DOM 之前调用，很少使用）
3. **render**: render是纯函数，只返回需要渲染的内容，不应该包含其它的业务逻辑。（可以返回原生的DOM、React组件、Fragment、Portals、字符串和数字、Boolean和null等内容）
4. **componentDidMount**: **组件被挂载到 DOM 之后会立即调用这个方法**，我们可以在**这里获取到DOM节点并进行操作**~~（比如对canvas，svg的操作）~~，**还可以向服务器发起请求及设置订阅**（例如计时器），但是记得在componentWillUnmount中取消订阅

**更新**阶段的生命周期方法按顺序依次为：

1. **getDerivedStateFromProps**: 在刚刚的挂载阶段说过了，所有这个方法在挂载和更新阶段都~~可能~~会被调用。
2. **shouldComponentUpdate**: ~~shouldComponentUpdate(nextProps, nextState),有两个参数nextProps和nextState，表示新的属性和变化之后的state，~~返回一个布尔值，true表示**会触发重新渲染**，false表示不会触发重新渲染，默认返回true（~~这个生命周期方法主要用于优化性能~~）
3. **render**: 返回更新后的内容（更新阶段也会触发此生命周期）
4. **getSnapshotBeforeUpdate**: 这个方法**在最新的渲染数据提交给DOM前会立即调用**，**可以在DOM 更新之前的获取它的值**~~，即使之前已经调用了 render 方法。getSnapshotBeforeUpdate(prevProps, prevState),这个方法在render之后，componentDidUpdate之前调用，有两个参数prevProps和prevState，表示之前的属性和之前的state，~~**它有一个返回值，会作为第三个参数传给接下来的componentDidUpdate方法，也就是这个生命周期方法必须与componentDidUpdate搭配使用**。
5. **componentDidUpdate**: **组件完成更新之后立即调用**。componentDidUpdate(prevProps, prevState, snapshot),该方法在getSnapshotBeforeUpdate方法之后被调用，有三个参数prevProps，prevState，snapshot，表示之前的props，之前的state，和snapshot。第三个参数是getSnapshotBeforeUpdate返回的,如果触发某些回调函数时需要用到 DOM 元素的状态，则将对比或计算的过程迁移至 getSnapshotBeforeUpdate，然后在 componentDidUpdate 中统一触发回调或更新状态。

**卸载阶段的生命周期方法是：**

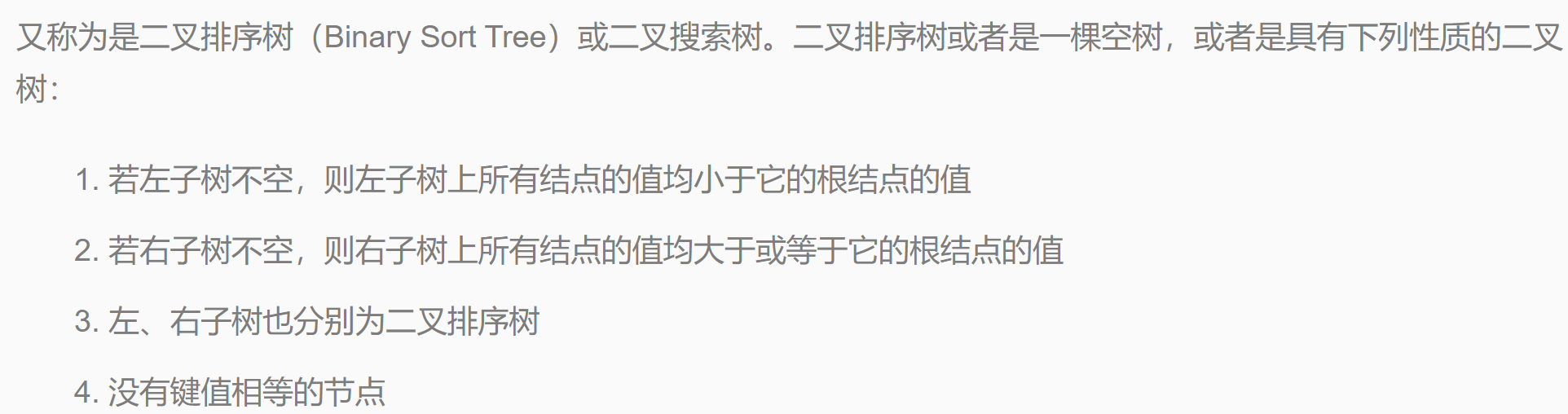
**componentWillUnmount**：在卸载和销毁组件之前会调用这个方法。可以在这个方法里清理资源，例如清除计时器、取消网络请求或清理在 componentDidMount() 中**创建**的订阅，

## Redux

Redux 是一个状态管理库，用于整个应用的状态管理，简化了React中**多级传递**的**单向数据流**。

## 算法相关

### 二叉搜索树及其应用场景



应用场景：

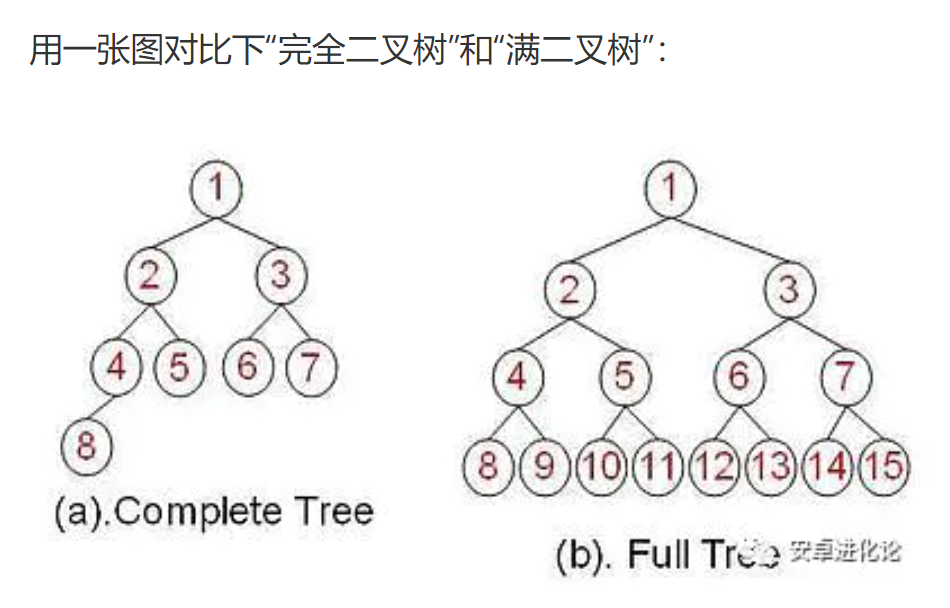
**海量数据的查找和插入**，二叉搜索树的查找和插入都是对数级别的，在处理大批量的动态的数据是比较有用。

### 完全二叉树是什么？满二叉树呢？

**搞错啦！**这是二叉堆！二叉堆是堆有序的完全二叉树，不是说完全二叉树就是堆！**×××**

~~完全二叉树就是用数组来表示堆有序的二叉树，~~





完全二叉树使用场景：堆（排序）用到就是堆有序的完全二叉树！

堆有序就是指二叉树的**每个节点都大于等于它的两个子节点**。**具体表示方法**就是将二叉树的节点按照**层级顺序**放入数组中。那么**位置为k的节点**的父节点的位置为k/2，而它的两个子节点的位置分别为2k和2k+1，这样在不使用指针的情况下，我们可以通过计算数组的索引在树中上下移动。

插入和删除元素的时间复杂度都是logN，即**对数级别**。特别适合处理海量数据的插入操作和删除操作混合的动态场景。

### 快排的原理？时间复杂度？最差情况下呢？空间复杂度？

快排是一种**分治**的排序算法，通过切分来排定一个元素的位置，然后再将这个元素**左边的子数组和右边的子数组**分别排序。其实就是**递归地调用切分函数**，**不断地确定元素在数组中的位置**，直到整个数组有序。

时间复杂度：平均：NlogN **线性对数**级别；

最差：**N^2**/2，**平方级别**（**数组本来就是从小到大或从大大小排序的，及顺序及倒序数组**，**这样每次都只能一个元素，导致一个大的数组需要切分很多次**）

空间复杂度：**logN**, **对数级别**（递归调用栈消耗的）

快排相比归并（时间复杂度也是NlogN）更快的原因是：**快排在内循环中移动数据的次数更少**