## 1，现代前端与传统前端开发模式上有什么不同？

传统前端：HTML + CSS + JS + JQuery 使用 JSP or 其他模板引擎填入数据，渲染依赖于后端；

缺点：HTML/CSS 占主导；

无模块化，无法产出符合工程化要求代码，组件隔离性差，

无法复用，异步控制流混乱，

多页面应用，用户体验差，

开发需要一遍遍刷新浏览器，查看效果，

现代前端：一般采用前后分离的架构，ES6 + FP + Node.js + Webpack + React/Vue/Angular + 状态管理 + babel + ......;

优势：以 JS 为核心；

代码模块化；

开发热更新；

良好的工程化解决方案；

组件隔离性好，无门槛复用；

一般是单页面应用，用户体验好

## 2，命令式编程 与 声明式编程 有何不同？

命令式编程：命令“机器”如何去做事情(how)，这样不管你想要的是什么(what)，它都会按照你的命令实现。比如：C, Java

声明式编程：告诉“机器”你想要的是什么(what)，让机器想出如何去做(how)。比如：sql,HTML, CSS

## 3，什么是 React Compoent? 它一般继承于什么？一般至少实现哪个方法？

组件（Component）是 React 最强大的功能之一

组件是自定义元素，是 React 的最小重用单元；

组件应该高内聚，低耦合；

继承于React.Component，React.Component 是一个抽象的基类，至少实现render()方法；

## 4，JSX语法对比普通 JavaScript 有何不同？它是如何解析的？（建议结合网上资料回答）

JavaScript：一种直译式脚本语言

JSX:即JavaScript XML——一种在React组建内部构建标签的类XML语法。(增强React程序组件的可读性)

JSX是一种JavaScript语法扩展，在React中可以方便地用来描述UI。JSX本身也是一个表达式，在编译后，JSX表达式会变成普通的JavaScript对象。

JSX就是Javascript和XML结合的一种格式。利用HTML语法来创建虚拟DOM。当遇到<，JSX就当HTML解析，遇到{就当JavaScript解析。

区别：

1、浏览器只能识别普通的js，普通的css，并不能识别scss，或者jsx（scss是css的拓展，jsx可以看做是js的拓展），所以webpack的作用是把scss转换为css，把jsx转换为浏览器可以识别的js，然后浏览器才能正常使用；

2、js就是本身并不支持react里面的jsx（也就是在js文件里面直接写html那种），现在他们可以直接写是因为编辑器可以选择语言的解析模式，编辑器正确显示是因为 虽然是.js文件，编辑器用了.jsx的解析模式，所以显示正确

3..jsx文件会自动触发编辑器以jsx的模式解析当前的文件，所以可以更不会出错

JSX语法，像是在Javascript代码里直接写XML的语法，实质上这只是一个语法糖，每一个XML标签都会被JSX转换工具转换成纯Javascript代码，只是使用JSX，组件的结构和组件之间的关系看上去更加清晰。

## 5，阻塞调用与非阻塞调用有何不同？

简单点说:

阻塞就是干不完不准回来，     
非阻塞就是你先干，我现看看有其他事没有，完了告诉我一声；

对于面向连接的协议，在连接建立阶段，阻塞与非阻塞也表现不一。在阻塞模式下，如果没有连接请求到达，则等待连接调用将阻塞直到有连接请求到达；但在非阻塞模式下，如果没有连接请求到达，等待连接调用将直接返回。