1. **请结合自身所了解ES6的新特性，写出var、let和const的使用要求？**

var:

作用域: var 声明的变量具有函数作用域或全局作用域，这意味着它在整个函数或全局范围内都有效。

变量提升: var 声明的变量会被提升到其作用域的顶部，即使变量在声明之前使用，也不会报错（只会有 undefined）。

重复声明: 允许在同一作用域内重复声明同一个变量，可能会导致意外的覆盖。

let:

作用域: let 声明的变量具有块级作用域，即只在 {} 内部有效。

变量提升: 同样会被提升，但在声明之前使用会导致 ReferenceError，称为“暂时性死区”。

重复声明: 不允许在同一作用域内重复声明同一个变量，否则会报错。

const:

作用域: const 声明的变量同样具有块级作用域。

不可变性: 声明的变量是常量，一旦赋值就不能再改变。注意，对于对象或数组，其内容可以修改，但引用不可变。

变量提升: 同样会提升，但具有“暂时性死区”。

1. **请写出在企业级项目开发过程中，前后端通信方式xml与JSON的区别与联系？**

**结构和复杂性:**

XML: 使用标签结构来表示数据，支持复杂的嵌套结构，适合用于传输复杂的文档类型数据。

JSON: 使用键值对的形式表示数据，结构更简洁，适合用于传输结构化数据，特别是与 JavaScript 交互时表现更佳。

易用性和解析:

XML: 解析相对复杂，需要专门的解析器（如 DOMParser），学习和使用成本较高。

JSON: 解析非常简单，JavaScript 中可以直接使用 JSON.parse() 和 JSON.stringify() 进行解析和序列化。

可读性:

XML: 结构清晰，标签名和内容结合，可读性较好，但代码量较大。

JSON: 结构简洁，键值对形式，易于阅读和理解。

应用场景:

XML: 常用于企业级系统的复杂数据交换，特别是需要传输结构化文档的场景。

JSON: 广泛应用于 Web 应用中的前后端数据交互，特别是在 RESTful API 中。

1. **请写出在企业级项目开发过程中，HTTP Method请求方式中Get与Post的提交方式的区别与联系？**

Get:

数据传输方式: 通过 URL 参数传递数据，数据可见，长度受限于 URL 长度限制（通常为 2048 字符）。

安全性: 数据暴露在 URL 中，安全性较低，不适合传输敏感信息。

使用场景: 适合用于获取数据，不修改服务器状态的请求（如查询、搜索等）。

幂等性: Get 请求是幂等的，多次请求的结果相同。

Post:

数据传输方式: 通过请求体（body）传递数据，数据不可见，长度不受限制。

安全性: 数据不暴露在 URL 中，安全性较高，适合传输敏感信息。

使用场景: 适合用于提交数据，修改服务器状态的请求（如表单提交、创建资源等）。

幂等性: Post 请求是非幂等的，多次请求可能会产生不同的结果。

1. **请写出在企业级项目开发过程中，C/S架构与B/S架构的优势与不足，及两者的实际应用场景？**

C/S 架构（Client/Server Architecture）:

优势:

性能高: 客户端和服务器之间的通信效率高，适合需要大量计算和处理的场景。

功能丰富: 客户端可以提供丰富的用户界面和功能，用户体验较好。

安全性强: 数据可以集中在服务器端管理，客户端不易被攻击。

不足:

部署和维护复杂: 每个客户端都需要安装和更新软件，维护成本高。

平台依赖性强: 客户端通常是特定平台的，跨平台支持较难。

应用场景:

桌面应用程序: 如图形处理软件、专业领域软件（如 CAD、ERP 系统）。

网络游戏: 需要高性能和实时通信的场景。

B/S 架构（Browser/Server Architecture）:

优势:

跨平台: 只需要浏览器即可访问，跨平台能力强。

易于部署和维护: 只需更新服务器端代码，客户端无需安装和更新。

成本低: 开发和维护成本相对较低，适合快速迭代和部署。

不足:

性能相对较低: 受限于网络带宽和服务器性能，不适合需要大量计算的场景。

功能受限: 浏览器端的功能和用户体验受限于浏览器本身。

应用场景:

Web 应用: 如内容管理系统（CMS）、电子商务平台、企业内部管理系统。

移动 Web 应用: 适合需要跨平台的应用，特别是在移动设备上。