**函数式课后**

1. 什么是纯函数？

答：纯函数是指给定相同输入时总是返回相同输出的函数，并且不产生外部副作用。

2. 什么是函数柯里化？

答：函数柯里化是一种将多参数函数转换为一系列单参数函数的技术。

通过柯里化，我们可以将一个接受多个参数的函数转换为一系列使用一个参数的函数。

3. 什么是高阶函数？

答：高阶函数是指接受一个或多个函数作为参数，并返回一个新函数的函数。

接受函数作为参数：高阶函数可以将一个或多个函数作为输入参数。这使得高阶函数具有很高的灵活性，可以根据传入的函数参数来实现不同的功能。

返回函数：高阶函数可以返回一个新的函数。这使得高阶函数具有很强的组合能力，可以将多个简单函数组合成具有复杂功能的函数。

代码复用和抽象：高阶函数可以用于封装通用的逻辑，从而实现代码复用和抽象。通过将变化的部分作为函数参数传入，高阶函数可以保持稳定的核心逻辑，提高代码的可维护性。

函数式编程支持：高阶函数是函数式编程的基本构建块，它们可以用于实现诸如函数柯里化、函数组合、惰性计算等函数式编程技术。

提高代码可读性：使用高阶函数可以将复杂的逻辑分解为多个简单的函数，从而提高代码的可读性。同时，高阶函数的名称通常可以清楚地表达其功能，有助于理解代码的意图。

无副作用：高阶函数通常遵循函数式编程的原则，尽量避免副作用。这有助于保持代码的纯粹性，降低代码之间的耦合度，提高代码的可测试性和可维护性。

4. 什么是函数组合？

答：函数组合是一种将多个函数组合在一起以创建一个新函数的技术。新函数的输出是由最内层函数到最外层函数依次执行的结果。

5. 什么是函数式编程中的副作用？什么情况下建议使用纯函数？

答：副作用是指函数在执行过程中对外部环境产生的影响，例如修改全局变量、操作 DOM、发送 HTTP 请求等。

函数式编程鼓励使用纯函数，避免副作用。

6. 什么是函数式编程的惰性计算？

答：惰性计算是指只在需要时才计算结果的一种策略。

在函数式编程中，惰性计算可以用于优化性能，例如避免不必要的计算或延迟计算直到需要结果时。

**性能优化课后**

1. 什么是浏览器的重绘和重排？如何减少重绘和重排？

答：重绘是指元素的外观发生改变，但不影响布局的过程。

重排是指元素的尺寸、位置或者其他布局属性发生改变，导致浏览器重新计算布局的过程。

减少重绘和回流的方法：

减少可能对位置尺寸布局属性产生变化的样式属性

2. Core Web Vitals的核心三指标以及举例自己的项目中每个指标分别如何做可能性优化

Largest Contentful Paint (LCP) 是一个性能指标，用于衡量页面主要内容渲染所需的时间。

优化 LCP 的方法包括：减小资源体积，使用 CDN，设置 HTTP 缓存策略，优先加载关键资源，使用骨架屏等。

First Input Delay (FID) 是一个性能指标，用于衡量用户与页面交互时，浏览器响应用户输入所需的时间。

优化 FID 的方法包括：减小 JavaScript 代码体积，避免长时间的主线程任务，使用 Web Workers 进行异步处理等。

Cumulative Layout Shift (CLS) 是一个性能指标，用于衡量页面在加载过程中的视觉稳定性。

优化 CLS 的方法包括：为图片和其他媒体元素设置尺寸属性，避免插入内容导致页面布局变化，优化字体加载策略等。

3. 如何在前端项目中监控 Core Web Vitals？

答：在前端项目中监控 Core Web Vitals 可以使用 PerformanceObserver API 或者第三方性能监控工具（如 Google Analytics、Lighthouse 等）。

监控体系包含：上报、存储、数据清洗、数据展示、分析告警、检查回归等

4. 如何优化首屏加载速度？哪些方向可以提升前端渲染整体性能？

优化 JavaScript 代码执行性能？

优化 JavaScript 代码执行性能的方法包括：避免使用昂贵的操作（如深拷贝、大数组操作等），使用 Web Workers 进行异步处理，避免阻塞 UI 线程，对计算密集型任务使用 WebAssembly 等。

减小前端资源的体积

减小前端资源体积的方法包括：压缩和优化图片、压缩和合并 JavaScript 和 CSS 文件、使用代码分割和按需加载、移除不必要的代码和库等。

设置HTTP缓存策略

HTTP缓存是一种通过在客户端存储资源副本，避免重复请求的技术。设置HTTP缓存策略可以通过设置响应头的 Cache-Control 和 ETag 等字段来实现

使用CDN进行优化

CDN（Content Delivery Network）是内容分发网络，通过将资源分布到全球的服务器上，使用户可以从就近的服务器获取资源。

使用CDN可以减少资源加载时间，提高访问速度。

使用懒加载、骨架屏、代码分割按需加载

5. 什么是Web Workers？如何使用Web Workers进行性能优化？

简述web worker、 service worker、 worklet三者的使用场景和联系

答：Web Workers 是一种在浏览器后台运行 JavaScript 代码的技术，不会阻塞 UI 线程。使用 Web Workers 进行性能优化可以将耗时的计算任务放在后台运行，避免阻塞 UI 线程。

Web Worker、Service Worker 和 Worklet 是三种在 Web 平台上实现并行处理的技术，它们各自有不同的用途和特点。

Web Worker： Web Worker 是一种在浏览器后台运行 JavaScript 代码的技术，它在一个独立的线程中执行，不会阻塞 UI 线程。Web Worker 主要用于处理耗时的计算任务，以避免阻塞 UI 线程导致页面卡顿。Web Worker 无法访问 DOM 和主线程的全局变量，它们之间通过 postMessage 进行通信。

Service Worker： Service Worker 是一种特殊类型的 Web Worker，它在浏览器后台独立于网页运行，主要用于实现离线缓存、推送通知和网络代理等功能。Service Worker 可以拦截和处理网络请求，使得开发者可以实现自定义的缓存策略和请求处理逻辑。与 Web Worker 类似，Service Worker 也无法访问 DOM，它们之间通过 postMessage 进行通信。

Worklet： Worklet 是一种轻量级的 Web Worker，用于在独立的线程中执行特定类型的任务，如样式计算、布局和绘制等。Worklet 具有更低的启动和终止开销，适用于频繁执行的短时任务。Worklet 通常与特定的 Web API（如 CSS Paint API、Audio Worklet API 等）一起使用，用于实现自定义的渲染和处理逻辑。

总结： Web Worker 用于处理耗时的计算任务，Service Worker 用于实现离线缓存和网络代理等功能，而 Worklet 用于执行特定类型的短时任务。这三者都在独立的线程中运行，不会阻塞 UI 线程。不过，它们之间有不同的使用场景和限制。