PWA的核心目标就是提升 Web App 的性能,改善 Web App 的用户体验。媲美native的流畅 PWA的中文名叫做渐进式网页应用 体验,将网络之长与应用之长相结合。 ● 由于其天生封闭的基因,内容无法被索引(无法SEO) ● 用户 80% 的时间被 Top3 的超级 App 占据,对于站点来说,应用分发的性价比也越 Native APP 用起来很流畅,但是也有其天然的基因缺陷: 来越不划算 ● 要使用它,首先还需要下载几十兆上百着兆的安装包 为什么W3C和谷歌在推广这项技术 ● 离线时用户无法使用 虽然社区之前也做过很多努力,例如virtual dom、spa、混合编程、用canvas将整个页面画 WEB前端虽然天生具有开放的基因,但是很多时候页面会卡顿,用户体验不佳。 ● 无法接收消息推送 出来,用户体验也有了很大的改善,但是仍然无法解决几个重要的问题: ● 移动端没有一级入口 • 渐进式 - 适用于选用任何浏览器的所有用户,因为它是以渐进式增强作为核心宗旨来开发的 • 自适应 - 适合任何机型:桌面设备、移动设备、平板电脑或任何未来设备。 • 连接无关性 - 能够借助于服务工作线程在离线或低质量网络状况下工作。 • 类似应用 - 由于是在 App Shell 模型基础上开发,因此具有应用风格的交互和导航,给用户以应用般的熟悉感 • 安全 - 通过 HTTPS 提供,以防止窥探和确保内容不被篡改。 • 可发现 - W3C 清单和服务工作线程注册作用域能够让搜索引擎找到它们,从而将其识别为"应用"。 • 可再互动 - 通过推送通知之类的功能简化了再互动。 • 可安装-用户可免去使用应用商店的麻烦,直接将对其最有用的应用"保留"在主屏幕上。 • 可链接 - 可通过网址轻松分享,无需复杂的安装。 跨平台 -可以把<mark>Service worker</mark>当做是一种客户端代理 ______ 借助服务工作线程(<mark>Service worker</mark>)实现 ______ 离线工作 但可以像原生APP在主屏幕上留有图标。

—— 无需安装 持续更新 比如支付宝就有把某些应用添加到主屏幕的功能 目前的应用 https://juejin.cn/post/6844903556470816781 各个前端应用还可以独立运行、独立开发、独立部署。微前端不是单纯的前端框架或者工具, 微前端(Micro-Frontends)是一种类似于微服务的架构,它将微服务的理念应用于浏览器 而是一套架构体系 端,即将 Web 应用由单一的单体应用转变为多个小型前端应用聚合为一的应用 https://juejin.cn/post/6844904162509979662 将这些庞大应用进行拆分,并随之解耦,每个部分可以单独进行维护和部署,提升效率。 整合历史系统 通过Nginx配置反向代理来实现不同路径映射到不同应用,例如www.abc.com/app1对应 app1,www.abc.com/app2对应app2,这种方案本身并不属于前端层面的改造,更多的是 运维的配置 Nginx路由转发 在切换应用时会触发浏览器刷新,影响体验 —— 缺点 父应用单独是一个页面,每个子应用嵌套一个iframe,父子通信可采用postMessage或者 contentWindow方式 iframe的样式显示、兼容性等都具有局限性;太过简单而显得low 缺点 每个子应用需要采用纯Web Components技术编写组件,是一套全新的开发模式 每个子应用拥有独立的script和css,也可单独部署 ______ 优点 _____ Web Components] 每个子应用独立构建和部署,运行时由父应用来进行路由管理,应用加载,启动,卸载,以及 通信机制 纯前端改造,体验良好,可无感知切换,子应用相互隔离 _____ 优点 ____ 组合式应用路由分发(当前流行方案) 需要设计和开发,由于父子应用处于同一页面运行,需要解决子应用的样式冲突,变量对象污 缺点 染,通信机制等技术点 基座应用大多数是一个前端SPA项目,主要负责应用注册,路由映射,消息下发等 该方案的核心是"主从"思想,即包括一个基座(MainApp)应用和若干个微(MicroApp)应 微应用是独立前端项目,这些项目不限于采用React,Vue,Angular或者JQuery开发,每个 微应用注册到基座应用中,由基座进行管理,但是如果脱离基座也是可以单独访问 当整个微前端框架运行之后,给用户的体验就是类似下图所示: 简单描述下就是基座应用中有一些菜单项,点击每个菜单项可以展示对应的微应用,这些应用 的切换是纯前端无感知的, 遗留系统(框架 jQuery)— 远程拉取机制通常会采用fetch API来首先获取到微应用的HTML内容,然后通过解析将微应用 技术广度 的JavaScript和CSS进行抽离,采用eval方法来运行JavaScript,并将CSS和HTML内容append 组合式应用路由分发方案 当然这个流程里会涉及到CSS样式污染以及JavaScript对全局对象污染 — 到基座应用中留给微应用的展示区域,当微应用切换走时,同步卸载这些内容,这就构成的当 前应用的展示流程。 基座要能拉取到微应用的页面内容,需要远程拉取机制 目前针对远程拉取机制这套流程,已有现成的库来实现,可以参考<u>import–html–entry</u> 路由切换的分发问题。 最先接收到这个变化的是基座的router,通过<mark>查询注册信息</mark>可以获取到转发到那个微应用,经 vue-router开发的基座SPA应用来举例, 过一些逻辑处理后,采用修改hash方法或者pushState方法来将路由信息推送给微应用的路 1. 当浏览器的路径变化后,vue-router会监听hashchange或者popstate事件, —— 路由切换 由,微应用可以是手动监听hashchange或者popstate事件接收,或者采用React-router, 从而获取到路由切换的时机。 vue-router接管路由,后面的逻辑就由微应用自己控制。 可以采用webpack的postcss插件,在打包时添加特定的前缀。 css隔离 主微应用的隔离问题。 在window上挂载特定对象 _____ JavaScript隔离 需要解决的问题 在基座应用中会定义事件中心Event,每个微应用分别来注册事件,当被触发事件时再由事件 通信问题 中心统一分发,这就构成了基本的通信机制 如果基座和微应用采用的是React或者是Vue,是可以结合Redux和Vuex来一起使用, 实现应用之间的通信。 Qiankun: 基于Single-Spa,阿里系开源微前端框架。 <u>lcestark</u>:阿里飞冰微前端框架,兼容多种前端技术栈 1. 微前端最佳的使用场景是一些B端的管理系统,既能兼容集成历史系统,也可以将新的 系统集成进来,并且不影响原先的交互体验。 首次绘制, 页面第一次绘制元素 首次内容绘制,首次绘制文本、 务结束时间就是 TTI 记录的时间 首次输入延迟,记录在 FCP 和 TTI 之间用户与页面交互时响应 用户体验指标 LCP 素绘制的时间 累计位移偏移,记录了页面上非 预期的位移波动 @揭金技术社区 由于window.preformance.timing是一个在不同阶段,被不停修正的一个参数 对象,所以,建议在window.onload中进行性能数据读取和上报。 navigationStart, 开始导航 unloadEventStart,开始卸载上一个页面 vindow.preformance.timin unloadEventEnd,完成卸载上一个页面 redirectStart, 开始重定向 redirectEnd,结束重定向 fetchStart, 请求开始 domainLookupStart,域名解析开始 domainLookupEnd,域名解析结束 connectStart,开始建立连接 connectEnd,完成建立连接 secureConnectionStart,开始建立安全连接 requestStart,请求发出 responseStart,接收到响应 responseEnd,接收全部响应 domLoading,当前网页DOM结构开始解析时,Document.readyState属性变 为"loading" 前端性能监控 domInteractive, HTML加载和解析完成, DOM树构建完成(可交互), 开始加载内嵌资 源,Document.readyState属性变为"interactive" domContentLoadedEventStart, 注册的(DOMContentLoaded)事件处理器开始执行, 此时DOM树和CSSOM树构建完成(渲染树构建完成) domContentLoadedEventEnd,所有事件处理器执行完毕 domComplete, 当前文档解析完成并加载完所有的子资源,即Document.readyState 变 为 'complete' loadEventStart, 页面资源加载完毕, 触发onload事件 loadEventEnd, onload事件执行完毕 (可分析网页内容并生成建议,还可以分别对移动端和PC端进行分析 ogle PageSpeed Insights https://pagespeed.web.dev/?utm_source=psi&utm_medium=redirect <u>Lighthouse</u> 是一个开源的自动化工具,用于改进网络应用的质量。 您可以将其 作为一个 Chrome 扩展程序运行,或从命令行运行。 您为 Lighthouse 提供一 个您要审查的网址,它将针对此页面运行一连串的测试,然后生成一个有关页面 性能的报告。 LightHouse

https://developers.google.com/web/tools/lighthouse

