PWA

Progressive Web App

分享人：zhangshibiao

PWA OUTLINE

- PWA 是什么
- PWA 能做什么
- 怎么搭建PWA 应用
- PWA 技术细节 && 兼容性
- PWA 的缺点？PWA 的应用场景？

PWA 是什么

> Progressive Web App, 简称 PWA，是提升 Web App 的体验的一种新方法，能给用户原生应用的体验（渐进增强）。

关键点

- PWA 本质上是 Web App
- 渐进增强体验（安全、性能和体验）
- 不是一种新框架／新语言
- 是应用一些技术进行改进（具备了 Native App 的一些特性）

PWA 是什么

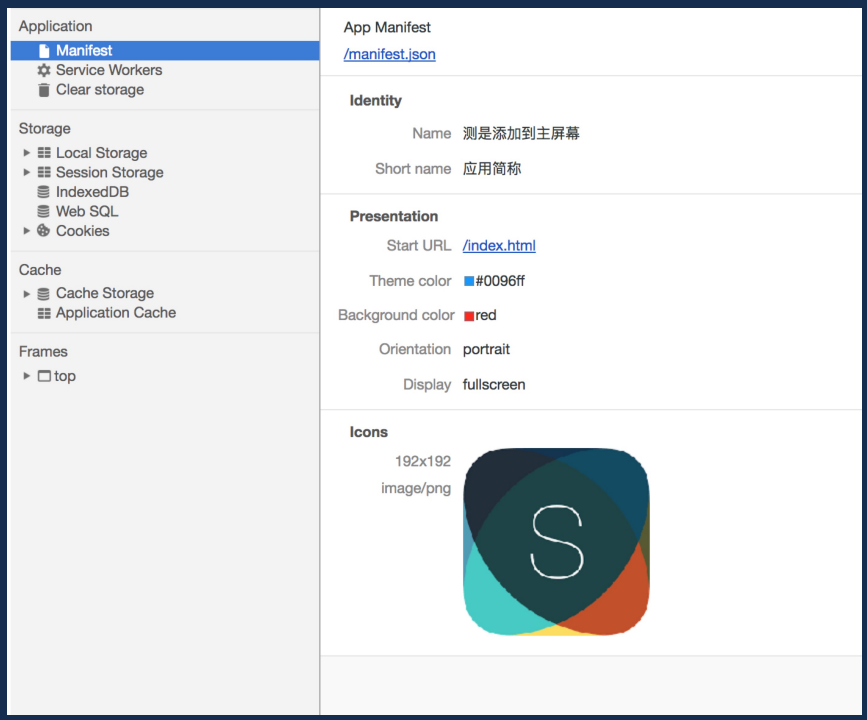
- 可以认为PWA 是以下几种技术的综合体
  - Service worker
  - Web App Manifest
  - Cache API
  - Fetch API
  - Push API
  - Web Push Protocol
  - Notification
  - ...

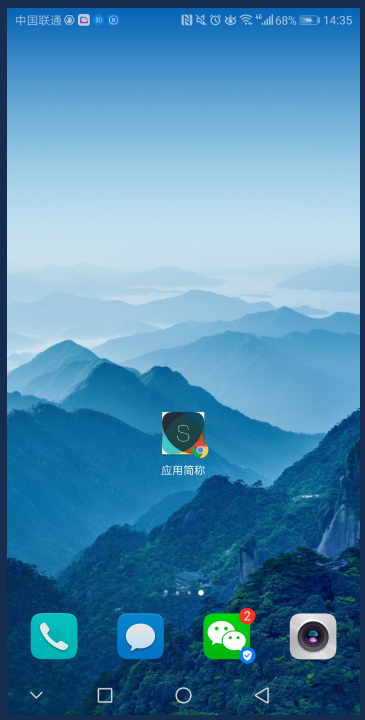
PWA 能做什么

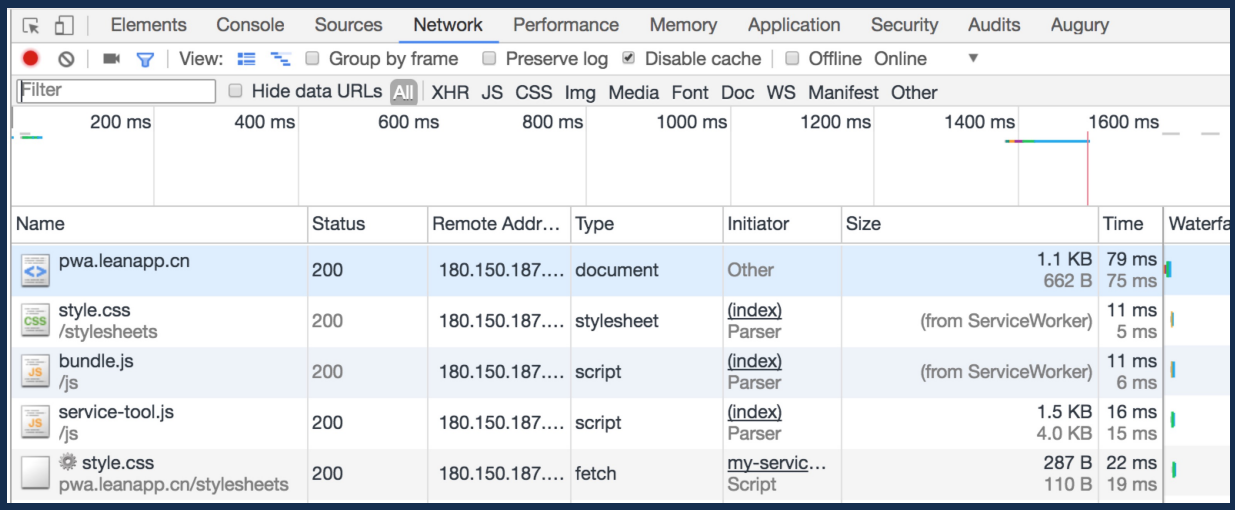
- PWA 支持能力
  - 离线工作能力
  - 推送通知
  - 几乎原生应用的展现和速度
  - 资源的本地和缓存
  - 添加主屏
- 特点
  - 可靠 > 即使在不稳定的网络环境下，也能瞬间加载并展现
  - 体验 (快速响应,资源缓存)
  - 粘性 入口可以添加到主屏幕， 推送

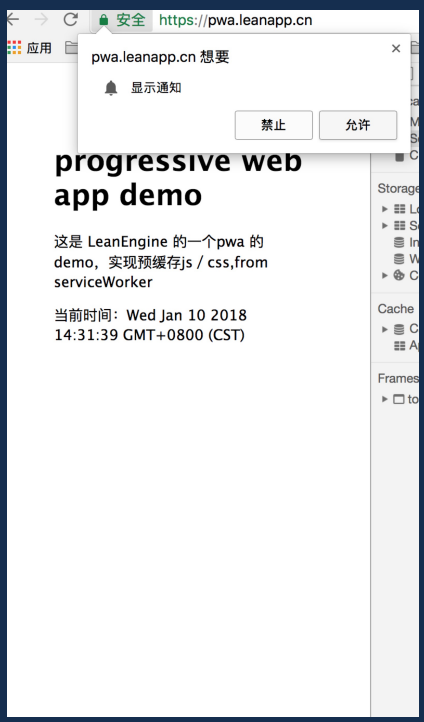
搭建PWA 应用

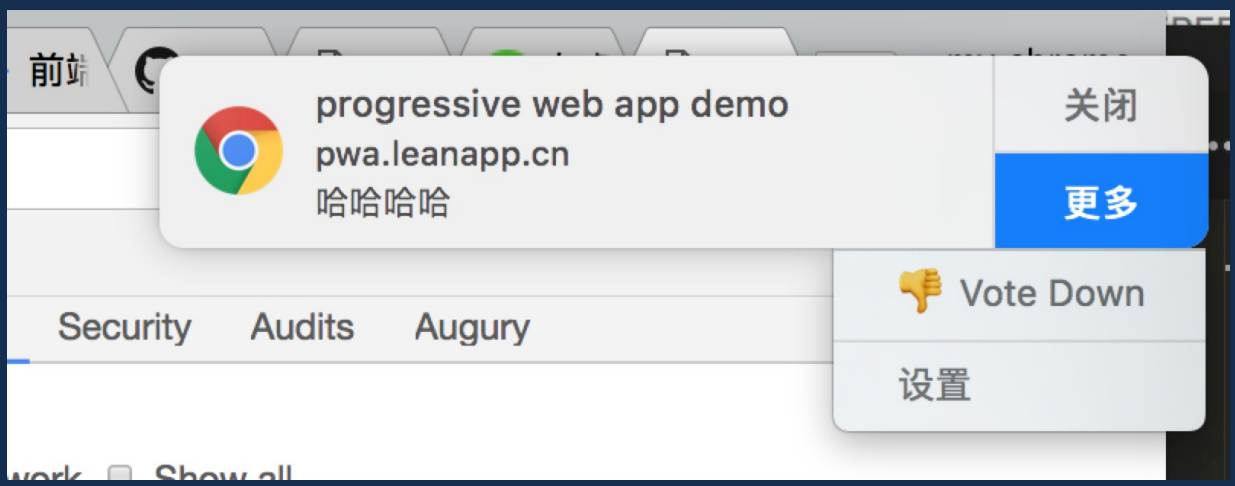
- 搭建 PWA 应用步骤
  - Https
  - manifest.json
  - ServiceWork.js
  - > Https 基于安全性考虑，pwa 必须要要求是https 站点,manifest.json 用于添加主屏幕的一些设置 ServiceWork.js 作为核心来处理一系列的缓存逻辑，推送通知，通信等
  - 让我们以一个demo 来大概预览下上面的几个技术点 demo


Application
Manifest
Service Workers
Clear storage
Storage
Local Storage
Session Storage
IndexedDB
Web SQL
Cookies
Cache
Cache Storage
Application Cache
Frames
top
App Manifest
/manifest.json
Identity
Name 测是添加到主屏幕
Short name 应用简称
Presentation
Start URL /index.html
Theme color #0096ff
Background color red
Orientation portrait
Display fullscreen
Icons
192x192
image/png


中国联通
68%
14:35
应用简称


Elements Console Sources Network Performance Memory Application Security Audits Augury
View: Group by frame Preserve log Disable cache Offline Online
Filter Hide data URLs All XHR JS CSS Img Media Font Doc WS Manifest Other
200 ms 400 ms 600 ms 800 ms 1000 ms 1200 ms 1400 ms 1600 ms
Name Status Remote Addr... Type Initiator Size Time Waterfa
pwa.leanapp.cn 200 180.150.187.... document Other 1.1 KB 662 B 79 ms 75 ms
style.css /stylesheets 200 180.150.187.... stylesheet (index) Parser (from ServiceWorker) 11 ms 5 ms
bundle.js /js 200 180.150.187.... script (index) Parser (from ServiceWorker) 11 ms 6 ms
service-tool.js /js 200 180.150.187.... script (index) Parser 1.5 KB 4.0 KB 16 ms 15 ms
style.css pwa.leanapp.cn/stylesheets 200 180.150.187.... fetch my-servic... Script 287 B 110 B 22 ms 19 ms


安全 https://pwa.leanapp.cn
应用
pwa.leanapp.cn 想要
显示通知
禁止
允许
progressive web app demo
这是 LeanEngine 的一个pwa 的 demo，实现预缓存js / css,from serviceWorker
当前时间：Wed Jan 10 2018 14:31:39 GMT+0800 (CST)
Storage
Cache
Frames


progressive web app demo
pwa.leanapp.cn
哈哈哈哈
关闭
更多
Vote Down
设置
Security
Audits
Augury

如何添加到主屏幕

- 添加到主屏幕的条件
  - 需要 manifest.json 文件
  - 清单文件需要启动 URL
  - 需要 144x144 的 PNG 图标
  - 网站正在使用通过 HTTPS 运行的 Service Worker
  - 用户需要至少浏览网站两次，并且两次访问间隔在五分钟之上
- 添加到主屏幕是系统行为，不需要额外处理，不过我们可以控制取消或者延迟提示及获取用户的行为

manifest.json

```
{
    "name": "测是添加到主屏幕",
    "short_name": "应用简称",
    "description": "这是应用的描述",
    "lang": "cn",
    "icons": [
        {
            "src": "logo-192x192.png",
            "type": "image/png",
            "sizes": "192x192"
        }
    ],
    "start_url": "/index.html",
    "display": "fullscreen",
    "orientation": "portrait",
    "theme_color": "#0096ff",
    "background_color": "red"
}
```

display显示模式

- display主要控制浏览器的展现形式
  - Fullscreen 占用整个可用的显示区域
  - Standalone 相对于Fullscreen 去掉状态栏
  - Minimal-ui 此模式类似于 fullscreen，但为终端用户提供了可访问的最小 UI 元素集合，例如，后退按钮、前进按钮、重载按钮以及查看网页地址的一些方式。
  - Browser 使用操作系统内置的标准浏览器来打开 Web 应用

ServieWorker（服务工作线程）

> serviceWorker 是浏览器在后台独立于网页运行的脚本。sw 是javascript的一种工作线程（worker,webworker),是独立的线程，并不需要关注网页内容及用户交互行为，因为在sw的运行的脚本是无法访问dom 或者window的。SW是离线、定期的后台同步、推送通知等功能的技术基础。

- 可以简单理解为浏览器与服务器之间的代理
  - 可以拦截fetch 请求
  - 可以调用cache storeage 对资源进行缓存，删除
  - 可以接受推送的通知进行中转

ServiceWork 技术细节

注册

```
if ('serviceWorker' in navigator) {
    navigator.serviceWorker.register('/my-service.js').then(function (registration) {
            registration.addEventListener("updatefound", () => {
                var newWorker = registration.installing;
                newWorker.addEventListener("statechange", () => {
                    // newWorker 状态发生变化
                    var state = newWorker.state; //
                    console.log('statechange', newWorker.state);
                    if (state === 'activated') {
                        // 如果sw 更新来，是否要刷新页面
                        window.location.reload();
                    }
                });
            });

            console.log('注册成功', registration.scope);
        }).catch(function (err) {
            // 注册失败
            console.log('注册失败 registration failed: ', err);
        });
}
```

ServiceWork 技术细节

解析（parsed）

- 如果 浏览器支持sw，则进行下载，下载完成之后进行解析，解析成功之后就尝试进行安装install
- install的监听做缓存处理

```
self.addEventListener('install', event => {
    console.log('安装service worker...-----');
    var _that = self;
    _that.skipWaiting();
    event.waitUntil(
        caches.open(cacheName)
        .then(function (cache) {
            return cache.addAll(preCacheUrls).then(function () {
                console.log('安装complete')
            }, function (error) {
                console.log(error);
            })
        })
    );
});
});
```

ServiceWork 技术细节

激活

> 如果install处理成功，并且文件缓存成功的话，此时install 结束，变为installed的状态，开始进行激活，注意：激活未必立刻生效，而是等待下次刷新浏览器才真正的生效，生效的状态为activated

```
self.addEventListener('activate', function (event) {
    console.log('Activated');
    self.clients.claim();
    event.waitUntil(
        caches.keys().then((keys) => {
            return Promise.all(
                keys.map(key => {
                    if (!expectedCaches.includes(key)) {
                        return caches.delete(key);
                    }
                })
            )
        }).then(() => {
            console.log("skipWaiting handle ! ");
        })
    );

});
```

sw生命周期总结

- sw lifecycle
  - sw 更新（下载），浏览器开始解析 => parsed
  - 尝试进行安装(installing)，此时触发install 事件，waitUntil 返回resole，标志installed
  - 激活，此时触发activate 事件，waitUntil resove，则activated ，但是下次才会生效
  - slef.skipWaiting() 和 self.clients.claim(); 来保证本次激活之后并且控制client

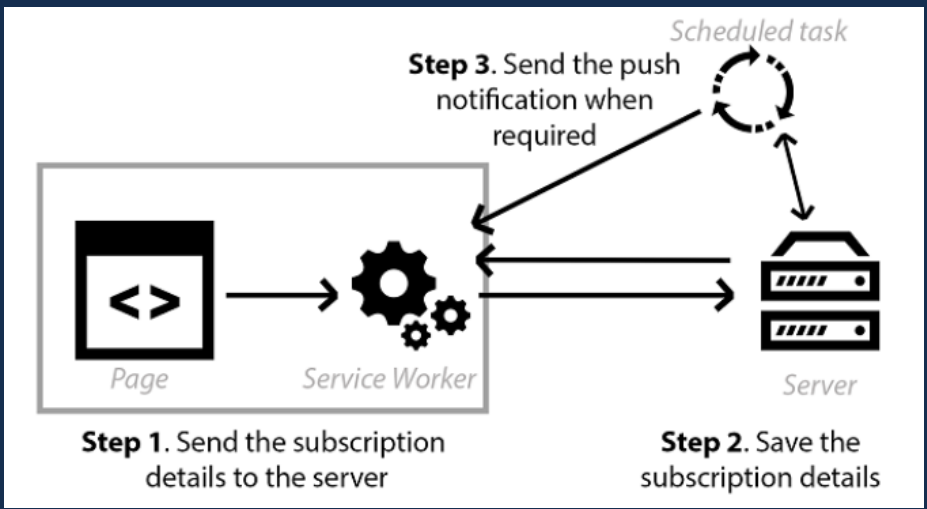
作用域

- sw 引入路径决定来sw的作用域

> sw 注册、安卓、激活之后会对其作用域的clients 生效，sw 只会对路径下的同级别或者子文件拥有控制权，eg: /js/sw.js => js/{dir}，如果我们在 /example/sw.js 处注册服务工作线程文件，则sw将只能看到网址以 /example/ 开头（即 /example/page1/、/example/page2/）的页面的 fetch 事件 。一般情况下会把sw 放在根目录下。

ServiceWork 技术细节

- push
  - 客户端询问是否允许推送
  - 用户允许推送，生成唯一标示(VAPID 协议)
  - 应用服务调用推送服务向客户端推送
  - sw 接收到推送消息给用户提示，并且监听响应用户的动作
  - 要不来段代码看一看



ServiceWork push技术细节

```
self.addEventListener('push', function (event) {

    var payload = event.data ? JSON.parse(event.data.text()) : 'no payload';
    // some logics here
    event.waitUntil(
        self.registration.showNotification(title, {
            body: payload.msg,
            url: payload.url,
            icon: payload.icon,
            actions: [
                // { action: 'voteup', title: '👍 Vote Up' },
                {
                    action: 'votedown',
                    title: '👎 Vote Down'
                }
            ]
        })
    );
});
```

ServiceWork 技术细节

sw更新策略

- diff 不同更改
- 24 小时自动更新
- sw.js?version={{versionNumber}} 也会触发更新

缓存文件的更新策略

- networkfirst
- cache first(如何保证缓存文件的及时更新)
    - sw 及时更新
    - 缓存文件的更新
    - 监听更新完成，引导用户手动更新 或者强制刷新
- .....

sw 的浏览器支持现状 && 未来

可用的浏览器日益增多。服务工作线程受 Firefox 和 Opera 支持。Microsoft Edge 现在表示公开支持。甚至 Safari 也暗示未来会进行相关开发。 您可以在 Jake Archibald 的 is Serviceworker ready 网站上查看所有浏览器的支持情况。 lavas pwa 兼容性

应用场景

- 离线应用
- 静态资源的缓存
- 推送场景

在借贷宝端外应用场景

- 缓存静态的css/js/img等
- 提升用户加载速率

怎么做

- 1.配置预缓存文件
- 2.自动化更新sw文件，实现即时更新（1，2 在自动化打包上线做对开发无感）
- 3.支持添加到主屏

google 官方对PWA提供的工具

- sw-precache
- sw-tool
- workbox

todo

- 在借贷宝的端外h5项目PWA工程化落地
  - 编译支持
  - 针对众多浏览器的api的兼容性处理
  - 降级开关方案

Q & A

THANKS