Engineer

Progressive web app

en webbplats som kan installeras och köras som en app

Vad det är

En progressive web app, även skriven Progressive Web App och oftast förkortad PWA, är en webbplats som är byggd så att en webbläsare kan installera den och köra den som en app: från en egen ikon, i ett eget fönster och ofta så att en del av den fortfarande fungerar när nätet är borta. Den förblir en vanlig webbplats hela tiden. Ingenting kommer från en appbutik, och sidorna är samma HTML, CSS och JavaScript som en webbläsare annars skulle läsa in i en flik.

Ingen standard definierar begreppet. Det standarderna definierar är de två delar som det byggs av. Den första är ett webbappsmanifest, en JSON-fil som beskrivs i W3C-specifikationen Web Application Manifest och som ger appen ett namn, ikoner, en start-URL och ett visningsläge. Specifikationen lägger ribban lågt för vem som räknas: ”Any website is an installable web application.” En webbläsare kan erbjuda ett sätt att installera en sådan webbplats, och den installerade appen öppnas sedan med inställningarna från manifestet. I visningsläget `standalone` ska den likna en nativ applikation, vilket kan innebära ett eget fönster och en ikon i programstartaren, och webbläsaren utelämnar sitt vanliga gränssnitt, till exempel adressfältet.

Den andra delen är en service worker från W3C-specifikationen Service Workers. Det är ett skript som webbläsaren kör skilt från sidan, som står mellan sidan och nätverket, ser de förfrågningar som sidan gör och kan besvara dem från cachar som skriptet själv styr. Det är det som gör att en webbapp kan fortsätta leverera sina sidor offline. De cacharna tillhör skriptet: specifikationen håller dem helt åtskilda från webbläsarens vanliga HTTP-cache, och de ändras bara när skriptet ändrar dem.

I Web Application Manifest förekommer inte ordet ”progressive” alls, och specifikationen Service Workers använder det en gång, i en anmärkning om hur Chromium känner igen en PWA. Guiden från Next.js om PWA:er rekommenderar progressiv förbättring (progressive enhancement), så att appen fortfarande fungerar bra i en webbläsare som saknar stöd för vissa av PWA-funktionerna. I den meningen är varje del valfri. En webbläsare som ignorerar manifestet, eller som saknar stöd för service workers, visar ändå webbplatsen.

En progressive web app är inte heller en egen sorts webbplats. En ensidesapp, en statisk webbplats och en webbplats med server-side rendering kan alla lägga till ett manifest och en service worker. Begreppet beskriver vad en webbläsare kan göra med en webbplats när den väl har lästs in, inte hur dess HTML tas fram.

Använd begreppet i ett beslut

Använd ”Vad är en progressive web app?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Gränssnitt som laddar snabbt, fungerar i mobilen och beter sig likadant i alla webbläsare dina kunder råkar använda.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Styrde full stack-utvecklingen av GIS-kartor — kartutveckling i full stack, från databas till webbläsare

Styrde full stack-utvecklingen av GIS-kartor Hör av er →

Varför det spelar roll

Guiden från Next.js lägger fram tre argument. Uppdateringar går ut direkt, utan att vänta på godkännande i en appbutik. En kodbas räcker för alla plattformar. Och appen får funktioner som folk förväntar sig av nativa appar, till exempel installation på hemskärmen och pushnotiser. Alla tre följer av att appen fortfarande är en webbplats: en ny version kan publiceras utan en separat utgåva i en appbutik; när varje enhet börjar använda den beror på cachning och uppdateringar av service workern.

Offline är skälet till att service workern finns. Specifikationens eget avsnitt om motivationen säger att webbapplikationer traditionellt har utgått från att nätverket går att nå, och att det försätter webbinnehåll i ett sämre läge än andra teknikstackar. Service workern är utformad för att rätta till den obalansen. När en sådan finns på plats behöver ett tåg i en tunnel eller ett fartyg långt från kusten inte längre sluta på webbläsarens felsida.

De två halvorna är oberoende av varandra. Guiden från Next.js påpekar att ett erbjudande om installation inte kräver offlinestöd, och manifestspecifikationen har tagit bort sin egen medlem `serviceworker`. En app kan vara installerbar utan service worker, och en service worker kan få en vanlig webbplats att fungera offline helt utan manifest.

Det båda halvorna har gemensamt är HTTPS. Next.js nämner en webbplats som levereras över HTTPS som en av två saker som en app behöver innan den kan installeras på hemskärmen i en mobil. Den andra är ett giltigt manifest. Specifikationen Service Workers säger att service workers måste köras i säkra kontexter, vilket i praktiken betyder HTTPS, och att localhost är tillåtet under utveckling.

Så ser det ut i praktiken

Tre små filer räcker till en fungerande PWA, och alla är skrivna för den här artikeln. Manifestet ger appen ett namn, väljer visningsläge och listar ikonerna:

```
{
  "name": "Field Notes",
  "short_name": "Notes",
  "start_url": "/",
  "display": "standalone",
  "icons": [
    { "src": "/icon-192.png", "sizes": "192x192", "type": "image/png" },
    { "src": "/icon-512.png", "sizes": "512x512", "type": "image/png" }
  ]
}
```

Varje sida länkar till manifestet och registrerar service workern, men bara i en webbläsare som stöder det:

```
<link rel="manifest" href="/manifest.webmanifest" />
<script>
  if ("serviceWorker" in navigator) {
    navigator.serviceWorker.register("/sw.js");
  }
</script>
```

Och själva service workern sparar en liten uppsättning filer för offlinebruk:

```
// sw.js
const CACHE = "shell-v2";
const SHELL = ["/offline.html", "/app.css", "/app.js"];

async function precache() {
  const cache = await caches.open(CACHE);
  await cache.addAll(SHELL);
}

async function dropOldCaches() {
  const names = await caches.keys();
  const old = names.filter((name) => name !== CACHE);
  await Promise.all(old.map((name) => caches.delete(name)));
}

async function respond(request) {
  if (request.mode === "navigate") {
    try {
      return await fetch(request);
    } catch {
      return caches.match("/offline.html");
    }
  }
  const hit = await caches.match(request);
  return hit ?? fetch(request);
}

self.addEventListener("install", (event) => {
  event.waitUntil(precache());
});

self.addEventListener("activate", (event) => {
  event.waitUntil(dropOldCaches());
});

self.addEventListener("fetch", (event) => {
  if (event.request.method !== "GET") return;
  event.respondWith(respond(event.request));
});
```

Manifestet använder `.webmanifest`, den filändelse som är registrerad för det, och levereras med medietypen `application/manifest+json`. Service workern ligger i roten eftersom dess omfång (scope) som standard är katalogen som skriptet ligger i, så en worker på `/js/sw.js` skulle bara kunna styra sidor under `/js/` om inte servern skickade ett `Service-Worker-Allowed`-huvud. När den installeras öppnar den en cache som heter `shell-v2` och sparar reservsidan för offlinebruk, stilmallen och skriptet där. `addAll()` avvisar om någon av filerna kommer tillbaka med en felstatus, och eftersom `waitUntil()` fick det promiset misslyckas hela

installationen tillsammans med det, i stället för att lämna en halvfylld cache efter sig. Vid aktiveringen raderar den varje cache med ett annat namn. Sidnavigeringar går först till nätverket och faller tillbaka på den sparade reservsidan när nätverket inte svarar. Allt annat levereras från cachen när det finns där och hämtas när det inte gör det. Förfrågningar som inte är `GET` lämnar den i fred.

Företaget beskriver ett fullständigt bygge av det här på en riktig plattform. Portfoliosidan om PWA-arbetet förklarar varför det behövdes: många av plattformens användare är sjöfolk och fältpersonal som använder mobilen, ofta offline eller på en svag uppkoppling, så appen måste kunna installeras som en nativ app och fortsätta fungera när signalen tappas. Den täcker båda halvorna. Appen installeras på hemskärmen med ikoner i flera storlekar, en startskärm och temafärger. Service workern kommer på plats med hjälp av next-pwa, och cachningen i Workbox under körning levererar typsnitt, bilder, ljud, video och CDN-resurser enligt CacheFirst, eftersom inget av det ändras mellan förfrågningar, tillsammans med HTTP-huvuden för cache-control. Eftersom service workers kräver HTTPS testades installationen och offlinebeteendet över HTTPS på riktiga enheter. Resultatet som sidan redovisar är en app som folk installerar och fortsätter att använda offline, där upprepade besök hämtas ur cachen.

Installationshalvan ser också annorlunda ut från läsarens sida. Webbplatsens sida om att installera den som en app visar var varje webbläsare har kommandot: knappen Dela och Lägg till på hemskärmen i Safari på en iPhone, en liten ikon ytterst till höger i adressfältet i Chrome, Edge, Brave, Opera och Vivaldi på en dator, och Lägg till i Dock i Safari på macOS Sonoma eller senare. Den säger också att iOS aldrig själv föreslår installation. Firefox erbjuder separata webbappfönster på Windows från version 143, eller version 150 om Firefox har installerats från Microsoft Store, men inte på macOS eller Linux. Sidan är också tydlig med vad installation av den här webbplatsen ger: en ikon och ett fönster, men ingen kopia för offlinebruk, eftersom en webbplats bara fungerar offline när den har en service worker.

Var det går fel

Användarna fastnar i den gamla versionen. Enligt specifikationen Service Workers uppdateras cacheobjekt bara när författaren uttryckligen ber om det, och de förfaller aldrig av sig själva utan finns kvar tills författaren raderar dem. En worker som går till cachen först levererar därför gårdagens stilmall tills skriptet säger något annat. Sätt en version i cachens namn, ändra den varje gång

de cachade filerna ändras och radera de gamla cacharna vid aktiveringen, som exemplet gör. Det är en ändring i skriptet som startar en uppdatering, eftersom webbläsaren jämför den nyss hämtade `sw.js` byte för byte med den installerade. Även då väntar den nya workern tills ingen flik eller inget fönster längre använder den gamla, om den inte anropar `skipWaiting()` , så en användare som har en flik öppen kör den tidigare versionen så länge fliken är öppen. Guiden från Next.js levererar `sw.js` med `Cache-Control: no-cache, no-store, must-revalidate` , så att användarna alltid får den senaste versionen av själva workern.

Det fungerar på den bärbara datorn men inte på testtelefonen. `navigator.serviceWorker` finns bara i säkra kontexter. På en sida som levereras över ren HTTP blir den vanliga funktionskontrollen helt enkelt false, workern registreras aldrig, och sidan säger ingenting om det. Specifikationen låter webbläsare godta localhost, 127.0.0.0/8 och ::1/128 under utveckling, och därför kan en worker som körs på utvecklarens egen dator saknas när en telefon öppnar samma app över HTTP från datorns adress i det lokala nätverket. Next.js erbjuder `next dev --experimental-https` för lokal testning över HTTPS. Testa installation och offlinebeteende på riktiga enheter, över HTTPS, innan du kallar något av dem klart.

En fetch-lyssnare som inte gör något. Specifikationen Service Workers noterar att vissa webbplatser registrerar en `fetch` -lyssnare med tom kropp bara för att Chromium ska känna igen dem som en progressiv webbapplikation. Specifikationen upptäcker nu lyssnare som `() => {}` och uppmuntrar webbläsare att varna för att sådana lyssnare är onödiga och kan försämra prestandan. Skriv en hanterare som faktiskt besvarar förfrågningar, eller låt bli att lägga till lyssnaren.

Att räkna med att webbläsaren föreslår installation. Om installation ska erbjudas, och hur, avgör webbläsaren. Next.js skriver att moderna webbläsare automatiskt erbjuder installation när en webbplats har ett giltigt manifest och HTTPS, och avråder från att bygga en egen installationsknapp på `beforeinstallprompt` , eftersom den händelsen inte fungerar i alla webbläsare och på alla plattformar, bland annat inte i Safari på iOS. Manifestspecifikationen har sedan dess tagit bort händelsen helt. Visningsläget och start-URL:en är också bara rekommendationer: en webbläsare som inte stöder `standalone` faller tillbaka på `minimal-ui` och sedan på `browser` , en vanlig flik. Förklara för användarna hur de installerar, som instruktionerna för iOS i guiden från Next.js gör, och se till att appen fungerar även när den öppnas i en flik.

Läs mer

- Web Application Manifest — manifestformatet, dess medlemmar, visningslägena och vad det innebär att installera och avinstallera.
- Service Workers — en workers livscykel, omfång, regler för uppdatering, kravet på säker kontext och Cache API.
- Web Application Manifest: Installable web applications — hur en webbläsare installerar en webbapplikation, startar den och låter användaren ta bort den igen.
- Service Workers: Understanding cache lifetimes — varför cacheobjekt aldrig uppdateras eller förfaller om inte skriptet ber om det.
- How to build a Progressive Web Application (PWA) with Next.js — en manifestfil, pushnotiser, installationsinstruktioner för iOS, HTTPS under utveckling och säkerhetshuvuden för service workern.

Ordlista

Välj utifrån era faktiska förutsättningar

Beskriv arbetsflödet, befintliga gränssnitt och användarna som behöver förändringen. Vi kan definiera en testbar leverans och en överlämning som gör att teamet kan underhålla lösningen.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är en progressive web app?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

welcome@engineer.company +45 50 10 61 84 Köpenhamn, Danmark

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/progressive-web-app/