Engineer

Progressive web app

et website, der kan installeres og køre som en app

Hvad det er

En progressive web app, også skrevet Progressive Web App eller progressiv webapp og som regel forkortet PWA, er et website, der er bygget, så en browser kan installere det og køre det som en app: fra sit eget ikon, i sit eget vindue og ofte sådan, at en del af det stadig virker, når netværket er væk. Det er hele tiden et almindeligt website. Intet kommer fra en appbutik, og siderne er den samme HTML, CSS og JavaScript, som browseren ellers ville indlæse i en fane.

Ingen standard definerer begrebet. Det, standarderne definerer, er de to dele, det er bygget af. Den første er web app-manifestet, en JSON-fil, som W3C beskriver i Web Application Manifest, og som giver appen et navn, ikoner, en start-URL og en visningstilstand. Specifikationen lægger barren lavt for, hvem der kan være med: »Any website is an installable web application.« En browser kan tilbyde en måde at installere det på, og den installerede app åbner derefter med manifestets indstillinger. I visningstilstanden `standalone` skal den ligne en native applikation, hvilket kan betyde et vindue for sig og et ikon i systemets programstarter, og browseren udelader sin almindelige brugerflade, for eksempel adresselinjen.

Den anden del er en service worker, som W3C beskriver i specifikationen Service Workers. Det er et script, som browseren kører adskilt fra siden. Det sidder mellem siden og netværket, ser de forespørgsler, siden sender, og kan besvare dem fra caches, som scriptet selv styrer. Det er det, der lader en webapp blive ved med at levere sine sider uden forbindelse. De caches hører til scriptet: specifikationen holder dem helt adskilt fra browserens almindelige HTTP-cache, og de ændrer sig kun, når scriptet ændrer dem.

Web Application Manifest bruger aldrig ordet »progressive«, og Service Workers-specifikationen bruger det én gang, i en note om, hvordan Chromium genkender en PWA. Next.js-guiden til PWA'er anbefaler progressive enhancement, så appen stadig fungerer godt i en browser, der mangler nogle af dens PWA-funktioner. I den forstand er hver del valgfri. En browser, der ignorerer manifestet eller slet ikke understøtter service workers, viser stadig websitet.

En progressive web app er heller ikke en særlig slags website. En single-page application, et statisk site og et site, der renderes på serveren, kan hver især få et manifest og en service worker. Begrebet beskriver, hvad en browser kan gøre med et site, når det er indlæst, og ikke hvordan dets HTML bliver til.

Brug begrebet i en beslutning

Brug »Hvad er en progressive web app?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Brugerflader, der indlæses hurtigt, virker på en telefon og opfører sig ens i alle de browsere, jeres kunder nu bruger.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har styret full-stack-udviklingen af GIS-kort — full stack-kortløsning, fra database til browser

Har styret full-stack-udviklingen af GIS-kort Kontakt os →

Hvorfor det betyder noget

Next.js-guiden fremfører tre argumenter. Opdateringer går ud med det samme uden at vente på godkendelse i en appbutik. Én kodebase dækker alle platforme. Og appen får funktioner, som folk kender fra native apps, for eksempel installation på hjemmeskærmen og pushnotifikationer. Alle tre følger af, at appen stadig er et website: en ny version kan offentliggøres uden en særskilt udgivelse i en appbutik; hvornår hver enhed tager den i brug, afhænger af caching og opdateringer af service workeren.

Offline er grunden til, at service workeren findes. Specifikationens eget afsnit om motivationen siger, at webapplikationer traditionelt har gået ud fra, at netværket kan nås, og at det stiller webindhold dårligere end andre teknologistakke. Service workeren er designet til at rette op på den skævhed. Med en sådan på plads behøver et tog i en tunnel eller et skib uden for rækkevidde af kysten ikke længere at ende ved browserens fejlside.

De to halvdele er uafhængige. Next.js-guiden bemærker, at en app godt kan tilbydes til installation uden at virke offline, og manifestspecifikationen har fjernet sit eget medlem `serviceworker` . En app kan være installerbar uden en service worker, og en service worker kan få et helt almindeligt site til at virke offline uden noget manifest.

Det, de to halvdele har til fælles, er HTTPS. Next.js nævner et site, der leveres over HTTPS, som det ene af to krav, før en app kan installeres på en mobils hjemmeskærm. Det andet er et gyldigt manifest. Service Workers-specifikationen siger, at service workers skal køre i sikre kontekster, hvilket i praksis betyder HTTPS, mens localhost er tilladt under udvikling.

Sådan ser det ud i praksis

Tre små filer er nok til en fungerende PWA, og alle tre er skrevet til denne artikel. Manifestet giver appen et navn, vælger visningstilstanden og angiver ikonerne:

```
{
  "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" }
  ]
}
```

Hver side linker til manifestet og registrerer service workeren, men kun i en browser, der understøtter det:

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

Og service workeren gemmer et lille sæt filer til brug uden forbindelse:

```
// 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 bruger `.webmanifest`, den filendelse, der er registreret til formatet, og leveres med medietypen `application/manifest+json`. Service workeren ligger i roden, fordi dens scope som standard er den mappe, scriptet ligger i, så en worker på `/js/sw.js` kun ville kunne styre sider under `/js/`, medmindre serveren sendte en `Service-Worker-Allowed`-header. Når den installeres, åbner den en cache ved navn `shell-v2` og gemmer offlinesiden, stylesheetet og scriptet. `addAll()` afviser, hvis bare én af filerne kommer tilbage med en fejlstatus, og fordi `waitUntil()` fik det promise, fejler hele installationen med det i stedet for

at efterlade en halvt fyldt cache. Ved aktivering sletter den alle caches med et andet navn. Sidenavigationer går først til netværket og falder tilbage til den gemte offlineside, når netværket ikke svarer. Alt andet leveres fra cachen, når det ligger der, og hentes, når det ikke gør. Andre forespørgsler end `GET` blander den sig ikke i.

Virksomheden beskriver en fuld udgave af det her på en rigtig platform. Portfolio-siden om PWA-arbejdet forklarer, hvorfor der var brug for det: mange af platformens brugere er søfolk og feltpersonale med telefonen i hånden, ofte offline eller på en svag forbindelse, så appen skulle kunne installeres som en native app og blive ved med at fungere, når signalet forsvinder. Siden dækker begge halvdele. Appen lægger sig på hjemmeskærmen med ikoner i flere størrelser, en splash-skærm og temafarver. Service workeren er sat op med pluginet next-pwa, og Workbox' runtime-caching leverer skrifttyper, billeder, lyd, video og filer fra CDN'et med cache først, fordi intet af det ændrer sig fra den ene forespørgsel til den næste, suppleret af Cache-Control-headere i HTTP-svarene. Da service workers kræver HTTPS, blev installation og offlineadfærd testet over HTTPS på rigtige enheder. Resultatet, siden nævner, er en app, folk installerer og bliver ved med at bruge offline, mens gentagne besøg hentes fra cachen.

Installationshalvdelen ser også anderledes ud fra læserens side. Sitets side om at installere det som en app gennemgår, hvor hver browser gemmer kommandoen: Del og Føj til hjemmeskærm i Safari på en iPhone, et lille ikon i højre side af adresselinjen i Chrome, Edge, Brave, Opera og Vivaldi på en computer og Føj til Dock i Safari på macOS Sonoma og nyere. Siden gør det også klart, at iOS aldrig selv foreslår installation. Firefox tilbyder særskilte webappvinduer på Windows fra version 143, eller version 150 hvis Firefox er installeret fra Microsoft Store, men ikke på macOS eller Linux. Siden er også tydelig om, hvad installation af dette site giver: et ikon og et vindue, men ingen kopi til brug offline, for det skal der en service worker til.

Hvor det går galt

Brugerne bliver hængende i den gamle version. Ifølge Service Workers-specifikationen bliver cacheobjekter ikke opdateret, medmindre udvikleren udtrykkeligt beder om det, og de udløber ikke, medmindre udvikleren sletter dem. En worker, der svarer fra cachen først, leverer derfor gårsdagens stylesheet, indtil scriptet siger andet. Sæt et versionsnummer i cachens navn, skift det, hver gang de gemte filer ændrer sig, og slet de gamle caches ved aktivering, som eksemplet gør. Det er en ændring af scriptet, der starter en

opdatering, fordi browseren sammenligner den nyhentede `sw.js` byte for byte med den installerede. Selv da venter den nye worker, til ingen faner eller vinduer længere bruger den gamle, medmindre den kalder `skipWaiting()` , så en bruger, der lader en fane stå åben, kører den forrige version, lige så længe fanen er åben. Next.js-guiden leverer `sw.js` med `Cache-Control: no-cache, no-store, must-revalidate` , så brugerne altid får den nyeste version af selve workeren.

Det virker på den bærbare og ikke på testtelefonen. `navigator.serviceWorker` findes kun i sikre kontekster. På en side over almindelig HTTP falder det sædvanlige featuretjek stille negativt ud, workeren bliver aldrig registreret, og intet på siden afslører det. Specifikationen lader browsere tillade localhost, 127.0.0.0/8 og ::1/128 under udvikling, og derfor kan en worker, der kører fint på udviklerens egen maskine, mangle, når en telefon åbner den samme app over HTTP via maskinens adresse på det lokale netværk. Next.js tilbyder `next dev --experimental-https` til lokal test over HTTPS. Test installation og offlineadfærd på rigtige enheder over HTTPS, før nogen af delene kaldes færdig.

En fetch-lytter, der ingenting gør. Service Workers-specifikationen noterer, at nogle sites registrerer en `fetch` -lytter med en tom krop alene for at få Chromium til at genkende dem som en PWA. Specifikationen opdager nu lyttere som `() => {}` , og den opfordrer browsere til at advare om, at sådanne lyttere er unødvendige og kan gå ud over ydeevnen. Skriv en handler, der rent faktisk besvarer forespørgsler, eller udelad lytteren.

At regne med, at browseren selv foreslår installation. Om installation skal tilbydes, og hvordan, bestemmer browseren. Next.js siger, at moderne browsere automatisk opfordrer til at installere, når et site har et gyldigt manifest og HTTPS, og fraråder at bygge en egen installationsknap på `beforeinstallprompt` , fordi den hændelse ikke virker på tværs af browsere og platforme, Safari på iOS medregnet. Manifestspecifikationen har siden fjernet hændelsen helt. Visningstilstanden og start-URL'en er også kun vejledende: en browser, der ikke understøtter `standalone` , falder tilbage til `minimal-ui` og derefter til `browser` , altså en helt almindelig fane. Fortæl brugerne, hvordan de installerer, som iOS-vejledningen i Next.js-guiden gør, og design efter, at appen også bliver åbnet i en fane.

Læs mere

- Web Application Manifest — manifestets format, dets medlemmer og visningstilstande, og hvad det vil sige at installere og afinstallere.

- Service Workers — workerens livscyklus, scope, regler for opdatering, kravet om sikker kontekst og Cache API'et.
- Web Application Manifest: Installable web applications — hvordan en browser installerer en webapplikation, starter den og lader brugeren fjerne den igen.
- Service Workers: Understanding cache lifetimes — hvorfor cacheobjekter aldrig opdateres eller udløber, medmindre scriptet beder om det.
- How to build a Progressive Web Application (PWA) with Next.js — en manifestfil, pushnotifikationer, installationsvejledning til iOS, HTTPS under udvikling og sikkerhedsheadere til service workeren.

Ordliste

Vælg ud fra jeres faktiske vilkår

Beskriv arbejdsgangen, de eksisterende grænseflader og de brugere, der har brug for ændringen. Vi kan definere en testbar leverance og en overdragelse, der gør teamet i stand til at vedligeholde løsningen.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er en progressive web app?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

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

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/progressive-web-app/