**Engineer**

# Progressive web app

et nettsted som kan installeres og kjøres som en app

## Hva det er

En progressive web app, også skrevet Progressive Web App eller progressiv nettapp og som regel forkortet til PWA, er et nettsted som er bygd slik at en nettleser kan installere det og kjøre det som en app: fra sitt eget ikon, i sitt eget vindu og ofte slik at en del av det fortsatt virker når nettet er borte. Det forblir et vanlig nettsted hele tiden. Ingenting kommer fra en appbutikk, og sidene er den samme HTML-en, CSS-en og JavaScript-koden som en nettleser ellers ville lastet inn i en fane.

Ingen standard definerer begrepet. Det standardene definerer, er de to delene det bygges av. Den første er et web app-manifest, en JSON-fil beskrevet i W3C-spesifikasjonen Web Application Manifest, som gir appen et navn, ikoner, en start-URL og en visningsmodus. Spesifikasjonen legger lista lavt for hvem som kvalifiserer: «Any website is an installable web application.» En nettleser kan tilby en måte å installere et slikt nettsted på, og den installerte appen åpnes deretter med innstillingene fra manifestet. I visningsmodusen `standalone` skal den ligne en native applikasjon, noe som kan bety et eget vindu og et ikon i programstarteren, og nettleseren utelater sitt vanlige grensesnitt, for eksempel adressefeltet.

Den andre delen er en service worker fra W3C-spesifikasjonen Service Workers. Det er et skript nettleseren kjører atskilt fra siden, som står mellom siden og nettet, ser forespørslene siden sender, og kan svare på dem fra cacher som skriptet selv styrer. Det er det som lar en webapp fortsette å levere sidene sine uten nett. Disse cachene tilhører skriptet: spesifikasjonen holder dem helt atskilt fra den vanlige HTTP-cachen i nettleseren, og de endres bare når skriptet endrer dem.

I Web Application Manifest forekommer ikke ordet «progressive» i det hele tatt, og spesifikasjonen Service Workers bruker det én gang, i en merknad om hvordan Chromium gjenkjenner en PWA. Veiledningen til Next.js om PWA-er anbefaler progressiv forbedring (progressive enhancement), slik at appen fortsatt fungerer godt i en nettleser som mangler støtte for enkelte av PWA-funksjonene. I den forstand er hver del valgfri. En nettleser som overser manifestet, eller som ikke støtter service workers, viser likevel nettstedet.

En progressive web app er heller ikke en egen type nettsted. En enkeltsideapp, et statisk nettsted og et nettsted med server-side rendering kan alle legge til et manifest og en service worker. Begrepet beskriver hva en nettleser kan gjøre med et nettsted etter at det er lastet inn, og ikke hvordan HTML-en blir laget.

Bruk begrepet i en beslutning

Bruk «Hva er en progressive web app?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Grensesnitt som laster raskt, fungerer på mobilen og oppfører seg likt i alle nettlesere kundene dine måtte bruke.

**Se tjenestene våre** Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Styrte fullstack-utviklingen av GIS-kart — fullstack-kartutvikling, fra database til nettleser

**Styrte fullstack-utviklingen av GIS-kart** Ta kontakt →

## Hvorfor det er viktig

Veiledningen fra Next.js begrunner det med tre punkter. Oppdateringer går ut med en gang, uten å vente på godkjenning i en appbutikk. Én kodebase betjener alle plattformer. Og appen får funksjoner folk forventer av native apper, som installasjon på hjemskjermen og push-varsler. Alle tre følger av at appen fortsatt er et nettsted: en ny versjon kan publiseres uten en egen utgivelse i en appbutikk; når hver enhet tar den i bruk, avhenger av caching og oppdateringer av service workeren.

Frakoblet bruk er grunnen til at service workeren finnes. Spesifikasjonens eget avsnitt om motivasjon sier at webapplikasjoner tradisjonelt har tatt for gitt at nettet er tilgjengelig, og at det stiller webinnhold svakere enn andre teknologistakker. Service workeren er laget for å rette opp den ubalansen. Når den er på plass, trenger ikke et tog i en tunnel eller et skip langt fra land lenger å ende på nettleserens feilside.

De to halvdelene er uavhengige av hverandre. Veiledningen fra Next.js påpeker at et tilbud om installasjon ikke krever støtte for frakoblet bruk, og manifestspesifikasjonen har fjernet sitt eget medlem `serviceworker` . En app kan være installerbar uten service worker, og en service worker kan få et vanlig nettsted til å virke frakoblet uten noe manifest i det hele tatt.

Det de to halvdelene har felles, er HTTPS. Next.js nevner et nettsted som leveres over HTTPS, som én av to ting en app trenger før den kan installeres på hjemskjermen på en mobil. Den andre er et gyldig manifest. Spesifikasjonen Service Workers sier at service workers må kjøre i sikre kontekster, som i praksis betyr HTTPS, og at localhost er tillatt under utvikling.

## Slik ser det ut i praksis

Tre små filer utgjør en fungerende PWA, og alle er skrevet for denne artikkelen. Manifestet gir appen navn, velger visningsmodus og lister opp ikonene:

```
{
  "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 lenker til manifestet og registrerer service workeren, men bare i en nettleser som støtter det:

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

Og selve service workeren tar vare på et lite sett filer til frakoblet bruk:

```
// 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 bruker `.webmanifest`, filendelsen som er registrert for det, og leveres med medietypen `application/manifest+json`. Service workeren ligger i roten fordi virkeområdet (scope) som standard er mappen skriptet ligger i, så en worker på `/js/sw.js` ville bare kunne styre sider under `/js/` med mindre serveren sendte et `Service-Worker-Allowed`-hode. Når den installeres, åpner den en cache som heter `shell-v2`, og lagrer reservesiden for frakoblet bruk, stilarket og skriptet der. `addAll()` avviser hvis noen av filene kommer tilbake med en feilstatus, og fordi `waitUntil()` fikk dette promiset, mislykkes hele installasjonen sammen

med det i stedet for å etterlate en halvfull cache. Ved aktivering sletter den alle cacher med et annet navn. Sidenavigasjoner går til nettet først og faller tilbake til den lagrede reservesiden når nettet ikke svarer. Alt annet leveres fra cachen når det ligger der, og hentes når det ikke gjør det. Forespørsler som ikke er `GET`, lar den være.

Selskapet beskriver en full utbygging av dette på en ekte plattform. Porteføljesiden om PWA-arbeidet forklarer hvorfor det trengtes: mange av plattformens brukere er sjøfolk og feltpersonell som er på mobilen, ofte uten nett eller med svak forbindelse, så appen måtte la seg installere som en native app og fortsette å fungere når dekningen blir borte. Den dekker begge halvdelene. Appen installeres på hjemskjermen med ikoner i flere størrelser, en oppstartsskjerm og temafarger. Service workeren kommer på plass via next-pwa, og Workbox sin caching under kjøring leverer fonter, bilder, lyd, video og CDN-ressurser etter CacheFirst-prinsippet, siden ingen av dem endrer seg mellom forespørsler, sammen med HTTP-hoder for cache-control. Fordi service workers krever HTTPS, ble installasjon og frakoblet bruk testet over HTTPS på ekte enheter. Resultatet siden oppgir, er en app folk installerer og fortsetter å bruke uten nett, der gjentatte besøk hentes fra cachen.

Installasjonshalvdelen ser også annerledes ut fra leserens side. Nettstedets side om å installere det som en app viser hvor hver nettleser har kommandoen: Del-knappen og Legg til på Hjem-skjerm i Safari på iPhone, et lite ikon ytterst til høyre i adressefeltet i Chrome, Edge, Brave, Opera og Vivaldi på en datamaskin, og Legg til i Dock i Safari på macOS Sonoma eller nyere. Den sier også at iOS aldri foreslår installasjon av seg selv. Firefox tilbyr egne nettappvinduer på Windows fra versjon 143, eller versjon 150 hvis Firefox er installert fra Microsoft Store, men ikke på macOS eller Linux. Siden er også tydelig på hva installasjon av dette nettstedet gir: et ikon og et vindu, men ingen frakoblet kopi, fordi et nettsted bare virker uten nett når det har en service worker.

## Hvor det går galt

**Brukerne blir hengende på den gamle versjonen.** Ifølge spesifikasjonen Service Workers oppdateres ikke cache-objekter med mindre forfatteren uttrykkelig ber om det, og de utløper ikke før forfatteren sletter dem. En worker som går til cachen først, leverer derfor gårsdagens stilark helt til skriptet sier noe annet. Sett et versjonsnummer i cachenavnet, endre det hver gang de cachede filene endres, og slett de gamle cachene ved aktivering, slik eksemplet gjør. Det er en endring i skriptet som starter en oppdatering, fordi nettleseren sammenligner den nylig hentede `sw.js` byte for byte med den installerte. Selv da venter den

nye workeren til ingen faner eller vinduer bruker den gamle lenger, med mindre den kaller `skipWaiting()`, så en bruker som holder en fane åpen, kjører den forrige versjonen så lenge fanen står åpen. Veiledningen fra Next.js leverer `sw.js` med `Cache-Control: no-cache, no-store, must-revalidate`, slik at brukerne alltid får den nyeste versjonen av selve workeren.

**Det virker på den bærbare, men ikke på testtelefonen.** `navigator.serviceWorker` finnes bare i sikre kontekster. På en side som leveres over ren HTTP, gir den vanlige funksjonssjekken bare false, workeren blir aldri registrert, og siden sier ingenting om det. Spesifikasjonen lar nettlesere godta localhost, 127.0.0.0/8 og ::1/128 under utvikling, og derfor kan en worker som kjører på utviklerens egen maskin, mangle når en telefon åpner den samme appen over HTTP fra maskinens adresse på det lokale nettverket. Next.js tilbyr `next dev --experimental-https` for lokal testing over HTTPS. Test installasjon og frakoblet bruk på ekte enheter, over HTTPS, før du kaller noen av dem ferdig.

**En fetch-lytter som ikke gjør noe.** Spesifikasjonen Service Workers beskriver at noen nettsteder registrerer en `fetch`-lytter med tom kropp bare for at Chromium skal gjenkjenne dem som en progressiv webapplikasjon. Spesifikasjonen oppdager nå lyttere som `() => {}`, og den oppfordrer nettlesere til å advare om at slike lyttere er unødvendige og kan gå ut over ytelsen. Skriv en handler som faktisk svarer på forespørsler, eller utelat lytteren helt.

**Å regne med at nettleseren foreslår installasjon.** Om installasjon skal tilbys, og hvordan, bestemmer nettleseren. Next.js skriver at moderne nettlesere viser et tilbud om installasjon automatisk når et nettsted har et gyldig manifest og HTTPS, og fraråder å bygge en egen installasjonsknapp på `beforeinstallprompt`, fordi den hendelsen ikke fungerer i alle nettlesere og på alle plattformer, blant annet ikke i Safari på iOS. Manifestspesifikasjonen har siden fjernet hendelsen helt. Visningsmodusen og start-URL-en er også bare veiledende: en nettleser som ikke støtter `standalone`, faller tilbake til `minimal-ui` og deretter til `browser`, en vanlig fane. Forklar brukerne hvordan de installerer, slik instruksjonene for iOS i veiledningen fra Next.js gjør, og sørg for at appen også fungerer når den åpnes i en fane.

## Les mer

- Web Application Manifest — manifestformatet, medlemmene, visningsmodusene og hva det betyr å installere og avinstallere.

- Service Workers — livssyklusen til en worker, virkeområdet, reglene for oppdatering, kravet om sikker kontekst og Cache API.
- Web Application Manifest: Installable web applications — hvordan en nettleser installerer en webapplikasjon, starter den og lar brukeren fjerne den igjen.
- Service Workers: Understanding cache lifetimes — hvorfor cache-objekter aldri oppdateres eller utløper med mindre skriptet ber om det.
- How to build a Progressive Web Application (PWA) with Next.js — en manifestfil, push-varsler, installasjonsinstruksjoner for iOS, HTTPS under utvikling og sikkerhetshoder for service workeren.

Ordliste

Velg ut fra deres faktiske rammer

Beskriv arbeidsflyten, eksisterende grensesnitt og brukerne som trenger endringen. Vi kan definere en testbar leveranse og en overlevering som gjør teamet i stand til å vedlikeholde løsningen.

**Se hva det koster** Ta kontakt →

## Gjør begrepene praktiske

Spør hvordan «Hva er en progressive web app?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

**Se tjenestene våre**

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/progressive-web-app/