Engineer

Single-page application

en webbapp som laddar en sida och ändrar den med JavaScript

Vad det är

En single-page application, även kallad ensidesapp och oftast förkortad SPA, är en webbapplikation som läser in ett HTML-dokument och sedan ändrar det dokumentet med JavaScript i stället för att be servern om en ny sida. Ett klick på en länk inuti den får inte webbläsaren att lämna sidan. Ett skript fångar klicket, hämtar de data som nästa vy behöver, skriver om en del av DOM:en och uppdaterar adressfältet, så att URL:en fortfarande talar om var läsaren befinner sig.

Ingen standard äger begreppet, och de som bygger verktygen säger det själva. Next.js inleder sin guide i ämnet med ”The definition of a SPA varies.” och slår sedan fast en definition för eget bruk, som Next.js kallar en strict SPA. Den har två kännetecken. Appen levereras av en enda HTML-fil, och varje route, varje sidbyte och varje datahämtning hanteras av JavaScript i webbläsaren. Och det sker inga fullständiga omladdningar av sidan: i stället för att begära ett nytt dokument för varje route ändrar JavaScript på klienten den aktuella sidans DOM och hämtar data vid behov. Den här artikeln använder den definitionen eftersom den är den mest direkta som finns och eftersom den pekar ut egenskapen som allt annat hänger på, nämligen ett dokument som aldrig byts ut.

Webbläsardelarna är standardiserade. HTML-standardens History-gränssnitt ger en sida `history.pushState()`, som lägger till en ny post i sessionshistoriken och sätter URL:en utan att läsa in något, och `history.replaceState()`, som i stället skriver om den aktuella posten. När läsaren går bakåt eller framåt mellan posterna skickar webbläsaren en `popstate`-händelse till fönstret och överlåter åt sidan att rita om innehållet. Det nyare Navigation API, som standarden presenterar som efterföljaren till de klassiska location- och history-API:erna, arbetar från andra hållet: en `navigate`-händelse utlöses när en navigering börjar, och dess metod `intercept()` gör navigeringen till en navigering inom samma dokument, som sidan hanterar själv.

Motsatsen är den vanliga webbplatsen med flera sidor, där varje länk ber servern om ett nytt dokument och webbläsaren kastar det gamla. Om en sidas HTML-kod skapas en gång av en statisk webbplatsgenerator eller vid varje förfrågan med server-side rendering är en annan fråga. Den handlar om hur HTML-koden kommer till, inte om vad som händer när den väl har kommit fram. Next.js suddar ut gränsen med flit. En statisk export från Next.js skriver en HTML-fil för varje route, och bytet mellan dem sker fortfarande på klienten.

Använd begreppet i ett beslut

Använd ”Vad är en single-page application?” 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

Eftersom dokumentet aldrig byts ut överlever allt som finns i det ett klick: ett halvt ifyllt formulär, en video som spelas, scrollpositionen i en sidopanel, data som redan har hämtats. Bara de data som nästa vy behöver måste över nätverket, och inte en hel ny sida där sidhuvud, stilmallar och skript följer med på nytt. Ett byte av vy kan vara så litet som att skriva om ett område på sidan.

Next.js nämner kostnaderna i samma andetag som definitionen. En strict SPA måste ofta läsa in en stor mängd JavaScript innan sidan kan bli interaktiv, och kaskader av förfrågningar på klienten, där en förfrågan kan börja först när en annan har kommit tillbaka, är svåra att hantera. Allt som läser sidan utan att köra skript får också betala. Med en enda HTML-fil bakom varje route får en sökrobot som inte kör JavaScript samma nästan tomma dokument, oavsett vilken URL den ber om. Next.js säger samma sak om sina egna komponenter som bara körs i webbläsaren: sökrobotar som inte kör JavaScript ser fallbacken i stället för komponenten.

Svaret från Next.js är att behålla navigeringen på klienten och flytta resten. Next.js delar automatiskt upp JavaScript-bundlarna och genererar flera HTML-ingångar till olika routes, så att en besökare inte läser in kod som en route inte behöver. Komponenten `next/link` förhämtar routes, vilket ger de snabba sidbytena från en strict SPA och samtidigt håller applikationens routingtillstånd i

URL:en, där det går att länka till och dela. En statisk export går ett steg längre och skriver en färdigrenderad sida för varje route, så att besökare får innehåll utan att vänta på klientens bundle, medan byten mellan sidor på klienten förblir lika snabba som i en strict SPA.

Valet görs inte heller en gång för alla. Next.js beskriver ett projekt som börjar som en statisk webbplats, eller till och med som en strict SPA som renderas helt på klienten, och som får serverfunktioner allteftersom det växer, och enligt Next.js kan befintliga SPA:er migreras utan att skrivas om helt.

Så ser det ut i praktiken

Routern nedan är skriven för den här artikeln och är den minsta fungerande versionen av idén. Den förutsätter att sidan har ett `<main>` -element och att servern svarar på varje sökväg i appen med samma HTML-dokument. De två vyerna står för det som appen egentligen visar.

```
// router.js
const views = {
  "/": { title: "Orders", text: "Your recent orders." },
  "/settings": { title: "Settings", text: "Delivery preferences." },
};
const main = document.querySelector("main");

function show(path, moveFocus) {
  const view = views[path] ?? views["/"];
  const heading = document.createElement("h1");
  const body = document.createElement("p");
  heading.textContent = view.title;
  heading.tabIndex = -1;
  body.textContent = view.text;
  main.replaceChildren(heading, body);
  document.title = view.title;
  if (moveFocus) heading.focus();
}

document.addEventListener("click", (event) => {
  const link = event.target.closest("a[href]");
  const modified = event.metaKey || event.ctrlKey || event.shiftKey;
  if (!link || link.origin !== location.origin) return;
  if (modified || event.button !== 0) return;
  event.preventDefault();
  history.pushState(null, "", link.href);
  show(location.pathname, true);
});
```

```
window.addEventListener("popstate", () => show(location.pathname, true));
show(location.pathname, false);
```

Klickhanteraren tar över vanliga vänsterklick på länkar till samma origin och släpper igenom klick med modifieringstangenter, så att det fortfarande går att öppna en länk i en ny flik. `preventDefault()` stoppar webbläsarens egen navigering, och `pushState()` registrerar en ny post i sessionshistoriken med länkens URL. Det mellersta argumentet finns bara av historiska skäl, och därför är det en tom sträng. `show()` byter ut vyn, sätter dokumentets titel och flyttar fokus till den nya rubriken, som en `tabIndex` på `-1` gör fokuserbar från skript, medan den hålls utanför tabbordningen. Lyssnaren på `popstate` ritar om den matchande vyn när läsaren går bakåt eller framåt, och den sista raden ritar den URL som sidan öppnades på, vilket bara fungerar om servern skickar samma dokument för `/settings` som för `/` . En riktig app skulle också visa en egen sida för adresser den inte känner till, i stället för att i tysthet falla tillbaka till startsidan, och den skulle låta länkar till filer och till andra delar av webbplatsen vara.

I grundarens nuvarande arbete för NextMariner visar frontenden i Next.js 16 kostnadssidan av det här på en riktig produkt. Portfoliosidan om frontenden beskriver en dashboard för inloggade användare som skulle kännas snabb och förbli genuint interaktiv, och berättar vad som händer när allt hämtas på klienten: den första laddningen blir seg, och N+1-mönstret dyker upp, där varje komponent skickar sin egen förfrågan när den vaknar, tills en enda sida har blivit en kaskad av rundresor. Det är just den kaskad av förfrågningar på klienten som Next.js-guiden varnar för, här i praktiken. Lösningen behöll interaktiviteten på klientsidan där den spelade roll och flyttade den första datahämtningen till servern. På dashboarden och de stora katalogsidorna lämnar en fabrik som heter `createPrefetchedServerPage` över varje sida till klienten med data redan på plats. Sidan sticker inte under stol med resten: de övriga sidorna för inloggade användare renderas fortfarande på klienten och väntar på sin tur, så den beskriver en migrering som fungerar och har en given nästa kandidat, inte en som är avslutad.

Var det går fel

Bakåtknappen lämnar appen. Om ett skript byter vy utan att anropa `pushState()` har sessionshistoriken ingen post för vyerna. Bakåt hoppar över varje vy som läsaren har sett och lämnar appen helt, och en omladdning hamnar på den URL som appen först öppnades på. Lägg till en post för varje vy som en

läsare skulle kalla en sida, använd `replaceState()` för ändringar som inte ska bli egna steg, till exempel ett filter som håller på att skrivas in, och hantera `popstate`. Next.js argumenterar för att hålla routingtillståndet i URL:en av samma skäl: det är det som gör det möjligt att länka till en vy och dela den.

En direktlänk till en undersida möter en server som bara känner till en fil. I en strict SPA betjänar en enda HTML-fil varje route, men adressfältet visar `/settings`, och det är den adressen webbläsaren ber om vid en omladdning eller när någon öppnar en delad länk. En server som bara har `index.html` svarar med en 404. Konfigurera antingen servern så att den svarar på varje sökväg i appen med dokumentet, eller generera en fil per route. Den statiska exporten i Next.js gör det senare: dess `out`-mapp innehåller filer som `index.html`, `404.html` och `blog/post-1.html`, och guiden visar en nginx-regel, `try_files $uri $uri.html $uri/ =404;`, som kopplar en snygg URL till den matchande filen.

Fokus stannar där den gamla vyn var. `pushState()` ändrar sessionshistoriken och URL:en och ingenting annat. När ett skript har bytt vy stannar tangentbordsfokus på länken som klickades, eller går förlorat om elementet som hade det togs bort, och ingenting talar om för hjälpmedel att en ny vy har kommit, om inte sidan gör det. Sätt titeln och flytta fokus till den nya vyns rubrik, som exemplet gör. Navigation API tar själv på sig mer av det arbetet. En navigering som hanteras med `intercept()` återställer som standard fokus till det första elementet med `autofocus` eller annars till body, och när hanteraren är klar signalerar webbläsaren till användaren att navigeringen är slutförd, vilket enligt standarden också omfattar hjälpmedel. En app som själv placerar fokus väljer bort det med `focusReset: "manual"`.

Allt betalas i förväg. En strict SPA måste ofta läsa in en stor mängd JavaScript innan något är interaktivt, och dess första dataförfrågningar startar först när den koden körs. Dela upp bundlen per route, så att en besökare bara laddar ner det som den aktuella vyn behöver, och starta de första förfrågningarna på servern. Mönstret i Next.js hämtar data i en Server Component utan att invänta dem och skickar promiset vidare till en Client Component, som packar upp det med `use()`. När förfrågan startar innan resten av appen renderas kan svaret enligt guiden strömmas direkt, och kaskader av förfrågningar på klienten undviks.

Läs mer

- How to build single-page applications with Next.js — definitionen av en strict SPA, vad den kostar och de mönster som Next.js använder för att behålla navigeringen på klienten och flytta arbete till servern.
- HTML Standard: The History interface — `pushState()`, `replaceState()` och hur poster i sessionshistoriken läggs till och byts ut.
- HTML Standard: Navigation API — `navigate`-händelsen, `intercept()` och den hantering av fokus och scroll som följer med.
- How to create a static export of your Next.js application — en HTML-fil per route, vad en statisk export inte kan, och snygga URL:er med nginx.

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 single-page application?” 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/single-page-application/