Engineer

Progressive enhancement

att bygga sidor som fungerar i ren HTML och sedan bygga på

Vad det är

Progressive enhancement är ett sätt att bygga webbsidor i lager, där man börjar med det lager som alltid fungerar. Grunden är HTML som innehåller allt innehåll och klarar varje uppgift på egen hand: länkar leder någonstans, formulär skickas till en server, och sidan kan läsas och användas utan att något annat har laddats. Stilmallar läggs ovanpå för att lägga upp sidan och göra den behaglig att se på. Skript kommer sist och gör den snabbare eller smidigare att använda. Varje lager förbättrar det som ligger under, och sidan är inte beroende av något lager ovanför HTML-koden för att göra sitt jobb. GOV.UK:s Service Manual sammanfattar idén som att få sidan att fungera med HTML först. På svenska kallas det också progressiv förbättring.

Namnet kommer från ett föredrag av Steven Champeon och Nick Finck, *Inclusive Web Design For the Future with Progressive Enhancement*, som enligt föredragets skriftliga version hölls den 11 mars 2003 på SXSW i Austin i Texas. Texten skriver Progressive Enhancement med stora begynnelsebokstäver och förkortar det till PE. Den ställer idén mot graceful degradation, den äldre vanan att bygga för den mest kapabla webbläsaren och hoppas att de andra ändå kunde nå innehållet. PE börjar enligt texten med ett dokument som de minst kapabla enheterna kan använda och lägger sedan på presentationen som separat logik, så att de enkla enheterna inte tyngs av det som de moderna webbläsarna får.

Skillnaden mellan de två ligger i arbetets riktning. Graceful degradation utgår från hela upplevelsen och frågar vad som finns kvar när delar av den saknas. Progressive enhancement utgår från det som alltid finns och frågar vad som kan läggas till. I en nyare webbläsare med en bra uppkoppling kan båda sluta i samma sida. Skillnaden syns när något fallerar: en sida som har byggts nerifrån har fortfarande kvar sin grund, medan en sida som har byggts uppifrån bara har det som skaparna kom ihåg att ordna.

Det är ett arbetssätt, inte ett bibliotek eller en funktion. Inget i HTML, CSS eller JavaScript heter progressive enhancement, men standarderna ger utrymme för det. En webbläsare hoppar över en CSS-deklaration som den inte förstår och ignorerar ett skript vars typ den inte känner till, och regeln `@supports` låter en stilmall fråga om en funktion finns innan den räknar med den.

Använd begreppet i ett beslut

Använd ”Vad är progressive enhancement?” 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

Lagren fallerar på olika sätt, och GOV.UK:s Service Manual redogör för hur. CSS-lagret är ganska förlåtande: en webbläsare hoppar över en deklaration som den inte känner igen och fortsätter, men en selektor som den inte känner till får den att kasta hela regeln. JavaScript-lagret förlåter inte alls. När ett skript använder syntax eller ett gränssnitt som webbläsaren saknar körs inte resten av skriptet. En sida som behöver sitt skript för sin huvuduppgift är därför bara så pålitlig som varje steg mellan servern och läsaren.

Och stegen är många. Samma sida räknar upp skäl till att CSS och JavaScript kan låta bli att laddas eller köras: ett tillfälligt nätverksfel, en annonsblockerare, ett CDN som ligger nere, ett DNS-fel, en bugg i en webbläsaruppdatering, en brandvägg på jobbet eller hemma. Inget av dem har med en gammal webbläsare att göra, och läsaren kan inte se vilket det var. Det enda läsaren ser är en knapp som inte gör något. Manualen anger inga siffror för hur ofta vart och ett av dem händer; argumentet vilar på listan, inte på en procentsats.

Vissa läsare stänger också av saker med flit. Manualen skriver att vissa användare medvetet stänger av funktioner i sin webbläsare och att deras val ska respekteras. HTML-standarden skriver att många användare stänger av skriptkörning, och av det skälet, och för att deklarativa mekanismer ofta är lättare att underhålla, uppmanar den författare att använda dem i stället för skript där de kan. Dess exempel är elementet `details`, som visar och döljer en

del av en sida utan en enda rad JavaScript. Standarden ber också författare att se till att deras applikationer fortfarande fungerar på en enklare nivå när det inte finns stöd för skript.

För brittiska offentliga tjänster gör manualen det till en regel: de bör kunna användas med enbart HTML, och JavaScript bör, där det används, bara förbättra HTML och CSS, så att folk fortfarande kan använda tjänsten när skriptet fallerar. Regeln är skriven för en regerings tjänster, inte för hela webben, men resonemanget håller för varje webbplats vars läsare skaparna inte själva kan välja.

Teamet bakom sidan vinner också något. När en förbättring går sönder förlorar läsaren förbättringen, inte uppgiften, så ett fel i ett skript blir ett irritationsmoment och inte ett avbrott. En ny funktion kan skickas ut till de webbläsare som stöder den utan en plan för dem som inte gör det, eftersom de behåller den sida de redan hade.

Så ser det ut i praktiken

HTML-standardens eget exempel på något som fungerar utan skript är en tabell vars kolumnrubriker sorterar den, och det är ett bra prov på metoden. Grunden är en tabell med vanliga länkar i rubrikerna. Ett klick ber servern om samma tabell sorterad efter den kolumnen, och servern skickar tillbaka den i den ordningen. Det fungerar i alla webbläsare, med eller utan skript.

```
<table id="orders">
  <thead>
    <tr>
      <th><a href="?sort=date">Datum</a></th>
      <th><a href="?sort=total">Belopp</a></th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>2026-10-20</td>
      <td>120</td>
    </tr>
    <tr>
      <td>2026-10-24</td>
      <td>45</td>
    </tr>
  </tbody>
</table>
<script type="module" src="sort.mjs"></script>
```

Skriptet är förbättringen. Det hittar samma länkar, avbryter navigeringen och sorterar om raderna på själva sidan, vilket sparar en tur fram och tillbaka till servern.

```
const table = document.querySelector("#orders");
const body = table.tBodies[0];
const collator = new Intl.Collator("sv", { numeric: true });

function sortBy(column) {
  const text = (row) => row.cells[column].textContent;
  const rows = [...body.rows];
  rows.sort((a, b) => collator.compare(text(a), text(b)));
  body.append(...rows);
}

for (const link of table.tHead.querySelectorAll("a")) {
  link.addEventListener("click", (event) => {
    event.preventDefault();
    sortBy(link.closest("th").cellIndex);
  });
}
```

Om ett fel uppstår innan en lyssnare har lagts till fungerar den aktuella länken fortfarande. Om `sort.mjs` aldrig kommer fram gör länkarna samma sak som förut. Om skriptet kastar ett fel halvvägs använder de rubriker som skriptet inte hann lägga till lyssnare för fortfarande servern. En webbläsare som inte stöder modulskript uppfattar `type="module"` som en skripttyp den inte känner till och ignorerar det, som HTML-standarden beskriver. Ett modulskript utan `async` körs när sidan har parsats klart, så tabellen finns när skriptet letar efter den. Det här exemplet återhämtar sig inte om en tillagd klickhanterare kastar ett fel efter att `preventDefault()` har avbrutit navigeringen.

Priset betalas på servern: grunden kräver en endpoint som kan sortera, och de två sätten att sortera måste ge samma resultat. Det är metodens vanliga kostnad, betald en gång. Den avsedda vinsten är en sida vars huvuduppgift fortfarande kan utföras via servern som reserv; en tillagd händelsehanterare måste bevara den möjligheten när dess eget arbete misslyckas.

Den här webbplatsen gör grunden tydlig. Tillgänglighetsredogörelsen beskriver kompletta sidor i HTML, CSS och SVG utan körbar JavaScript. Sökkoden finns kvar men är avstängd, så sidorna inte är beroende av ett filter. Redogörelsen nämner också en kvarstående gräns: de små etiketter som visas när en ikon får fokus eller

pekaren hålls över den kan inte stängas med en tangent. Det skulle kräva webbläsarkod som webbplatsen inte levererar. Därför uppfylls 1.4.13 Content on Hover or Focus bara delvis.

Var det går fel

Ett `noscript` -block får ersätta en grund. Elementet `noscript` visar bara sitt innehåll när skript är avstängt, och HTML-standarden säger det rakt ut: ”The noscript element is a blunt instrument.” Skript kan vara påslaget och sidans skript ändå fallera, och då får läsaren varken den skriptade sidan eller reservlösningen. Börja med en sida som fungerar utan skriptet och låt skriptet ändra den, som standardens eget exempel gör när dess skript lägger till ett levande resultat i ett formulär och tar bort formulärets skicka-knapp.

En funktion utan stöd stoppar hela skriptet. Eftersom JavaScript-lagret inte tål fel kan en enda bit nyare syntax i ett skriptpaket dra med sig alla förbättringar i samma paket i fallet, också de som annars hade fungerat. Håll grunden oberoende av skriptet, så att ett fel bara kostar förbättringen. Ladda modern kod med `type="module"` , som äldre webbläsare hoppar över, och om äldre webbläsare alls ska få en förbättrad version, ge dem ett separat klassiskt paket märkt med `nomodule` , som webbläsare med stöd för moduler inte kör.

En ny selektor drar med sig bra regler. En webbläsare som möter en selektor som den inte känner till kastar hela regeln, deklarationer och allt. Står en nyare pseudoklass i en lista tillsammans med selektorer som alla webbläsare förstår kan den därför ta bort stilar från webbläsare som annars hade använt dem. Ge en selektor som vissa webbläsare kanske inte stöder en egen regel.

Reservstilar läcker in i den förbättrade layouten. Bredder som är skrivna för webbläsare utan ett nyare layoutsystem kan motverka det layoutsystemet i de webbläsare som har det. CSS Conditional Rules Level 3 visar lösningen med en flexlayout och ett block med `@supports not (display: flex)` för alternativet och påpekar att breddangivelserna kan skada flexlayouten. Därför hör de bara hemma i stilarna för webbläsare utan den.

Bilder väljs av ett skript. HTML-standarden påpekar att om en sida inte anger någon bild i sin HTML-kod och lämnar valet åt ett skript laddas ingen bild alls ned för läsare som har stängt av skript, och om ett skript byter ut en bild som HTML-

koden redan har angett kan webbläsaren starta två nedladdningar. Ange kandidaterna i HTML med `srcset` och `sizes` eller med `picture`, och låt webbläsaren välja.

Beteende utan något under. Champeon och Fincks lista över strategier ber om en reserv för allt beteende: ett riktigt `href` bakom varje klickhanterare och en validering på servern bakom varje validering i webbläsaren. En kontroll som bara finns som klickhanterare, eller ett formulär som bara valideras i webbläsaren, har ingen grund alls. Använd riktiga länkar och riktiga formulärinskick först, validera alltid på servern, och se validering i webbläsaren som en bekvämlighet som har lagts ovanpå.

Man tror att det handlar om gamla webbläsare. Ett team som tror att grunden är till för gamla webbläsare tar bort den så snart de webbläsarna är borta. GOV.UK:s lista över orsaker handlar om nätverk, blockerare, brandväggar och felaktiga uppdateringar, och dem möter dagens webbläsare också. Håll grunden vid liv som en väg som används och testas: ladda då och då sidan med skript blockerade, och genomför dess huvuduppgift från början till slut.

Läs mer

- Using progressive enhancement — vägledningen i GOV.UK:s Service Manual, senast uppdaterad den 27 september 2024: de tre lagren, hur vart och ett av dem fallerar, och varför offentliga tjänster bör fungera med enbart HTML.
- HTML Standard: Scripting — elementet `script` med `type="module"` och `nomodule`, och elementet `noscript` med rådet att inte förlita sig på det.
- CSS Conditional Rules Module Level 3 — W3C:s Candidate Recommendation Draft från den 15 augusti 2024, som definierar `@supports` med `not`, `and` och `or`.
- HTML Standard: Images — `srcset`, `sizes` och `picture`, och varför det sviker läsare som har stängt av skript när ett skript väljer bilden.

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 progressive enhancement?” 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-enhancement/