Engineer

CSS

sproget, der siger, hvordan en side ser ud

Hvad det er

CSS, en forkortelse af Cascading Style Sheets og stadig ofte kaldt CSS3, er det regelsprog, en browser lægger ned over et dokument for at afgøre, hvordan hvert element tegnes: farver, størrelse, afstand, placering, skrift og bevægelse, uanset om dokumentet lander på en telefon, en bred skærm eller et ark papir.

Et stylesheet er en liste af regler. Hver regel parrer en selektor, der udpeger elementer, med deklarationer, som hver består af en egenskab og en værdi:

```
.note {
    color: #3a3a3a;
    margin-block: 1.5rem;
}
```

Det første ord i navnet er den del, der præger det daglige arbejde. Flere regler gør jævnligt krav på samme egenskab ved samme element, og kaskaden afgør, hvilken der vinder. Dens vigtigste trin er i rækkefølge oprindelse og vigtighed (browserens standardvalg, læserens egne indstillinger, sitets stylesheet, og om en deklaration er markeret `!important`), derefter kaskadelag, så specificitet og til sidst rækkefølgen i kilden, hvor en senere regel slår en tidligere med samme vægt. Specificitet tælles pr. selektor som tre tal: id'er, så klasser, attributter og pseudoklasser, så elementnavne. Den universelle selektor `*` lægger intet til nogen af dem. Arv virker ved siden af kaskaden: nogle værdier, som farve og skrift, går videre til et elements børn, mens andre, som margener og kanter, bliver, hvor de blev sat.

CSS har ikke ét versionsnummer. Efter CSS 2.1 gik arbejdsgruppen over til moduler: Level 3 bygger videre på 2.1 ét modul ad gangen, og derfra får hvert modul sine egne niveauer, så Selectors er i udkast på Level 4, mens moduler, der

er kommet til siden CSS 2.1, begynder på Level 1. »CSS3« er det folkelige navn for den modulære æra, og det årlige snapshot, der definerer nutidens CSS, siger ligeud, at der ikke findes noget CSS Level 4.

Den regel, der vejer tungest i praksis, er sprogets fejlhåndtering. En browser, der møder en egenskab, den ikke kender, eller en værdi, der er ugyldig for sin egenskab, skal ignorere netop den deklaration og anvende resten af reglen. Reglen findes, så nye egenskaber og værdier kan tilføjes uden at ødelægge ældre browsere, og det er den, der lader et site bruge en funktion, som kun nogle browsere understøtter. Prisen er stilhed. Siden bliver vist, en smule forkert, og det eneste spor er en overstreget linje i udviklerværktøjerne, som nogen skal gå på jagt efter.

Det meste af det, CSS sammenlignes med, viser sig at være en måde at skrive det på. Sass og Less er præprocessorer, der kompilerer til CSS. Utility-frameworks som Tailwind genererer det ud fra klassenavne i markuppen. CSS-in-JS-biblioteker udsender det fra JavaScript-komponenter, og en `style` -attribut sætter deklarationer direkte på ét element. Hver af dem ændrer, hvordan CSS skrives; hver af dem ender som deklarationer, kaskaden sorterer på præcis samme måde.

Brug begrebet i en beslutning

Brug »Hvad er CSS?« 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

Den beslutning, CSS tvinger et team til, er, hvor en regel må fejle i stilhed. Sproget smider det væk, det ikke kan anvende, og lader en stærkere regel vinde uden kommentar, så ethvert stylesheet af en vis alder rummer deklarationer, der intet gør. Det, et team har styr på, er, om nogen opdager det, og hvornår.

Tilgængelighed afhænger mere af det, end de fleste regner med. Kontrast, en synlig fokusring, respekt for et ønske om reduceret bevægelse, en brugbar tilstand med tvungne farver, tekst, der tåler at blive forstørret: hver af dem

er en håndfuld deklarationer, og hver kan slås fra af en regel et andet sted, der tilfældigvis er mere specifik. Siden ser fin ud for alle, der ikke havde brug for funktionen.

Vedligeholdelsen afhænger også af det. Et stylesheet, hvis opførsel ingen kan forudsige, vokser kun, for at slette en regel kan ødelægge en side, ingen har kontrolleret. Det sædvanlige slutpunkt er en fil, hvor det meste af vægten er overstyringer af tidligere overstyringer, og hver ændring laves ved at prøve værdier, til skærmen ser rigtig ud.

Verifikation afhænger mest direkte af det. En browser, der tager imod, hvad den kan, og ignorerer resten, kan ikke være testen. En linter, der læser kilden, fanger tastefejl og ugyldige værdier, før de sendes ud. Den kan ikke se, hvad en minificering eller et andet byggetrin gør ved filen bagefter, og den kan ikke se, hvilken af to gyldige regler der vinder på en given side. Kontroller mod den fil, der faktisk udleveres, og visuel sammenligning af renderede sider dækker det, linteren ikke kan.

At organisere kaskaden tidligt, med flad specificitet, navngivne lag og designtokens i custom properties, koster lidt disciplin pr. ændring. At omorganisere et stylesheet, der voksede uden den disciplin, koster som regel en omskrivning.

Sådan ser det ud i praksis

Sitets egen kolofon beskriver ét CSS-bundt, sammenkædet og fingeraftrykt ved build, hvor de håndtegnede pile og marginnoter er tegnet i CSS alene, og hvor Stylelint holder kilden til sine regler i kvalitetsporten. Samme side linker til W3C's CSS-validator og advarer læseren om, hvad den vil sige. I august 2026 stopper dens profil før `system-ui`, `@property`, `clamp()` i en `font-size` og scroll-drevne animationer, så mod /terms/ meldte den 176 fejl, hvoraf 175 handlede om funktioner, browserne har, og sitet bruger. Den sidste var ægte. En hvilende `rotate: 0deg` mistede sin enhed til minificeringen, `rotate: 0` er ikke en vinkel, og browseren smed hele deklarationen væk, så en animation blev sendt ud uden den rotation, den var skrevet med. Værdispecifikationen tillader kun et nul uden enhed som vinkel nogle få steder, af hensyn til gammel kode. Stylelint så aldrig problemet, fordi den læser kilden, og minificeringen skriver den om bagefter; nu læser en kontrol, virksomheden selv har skrevet, det, der faktisk blev udleveret.

Specificitet var hele historien i et stykke tilgængelighedsarbejde. Sitet respekterede indstillingen om reduceret bevægelse med én universel regel, der slog hver overgang fra, og under en emuleret indstilling animerede 11 selektorer stadig. En universel selektor scorer nul, så enhver regel med en klasse slog den. Rettelsen var strukturel frem for en stærkere selektor: alt, der bevæger sig, ligger nu i én sen stylesheet-del, og en linter fejler en overgang på transform alle andre steder. Farveovergange består, fordi de ikke er bevægelse, og bevægelse skal skrives som en overgang frem for en keyframe-animation, fordi det er overgangen, der kan afbrydes rent. Læren, der er skrevet ned sammen med arbejdet, er kort: en regel, der taber på specificitet, fejler i stilhed.

Farver er det, stylesheets glider først på, og sitets palet blev lukket ved 28 farver, hver dokumenteret med det kontrastforhold, den måler på den flade, hvor den optræder, og 11 mærkeværdier erklæret én gang som tokens. To farver, der ligger tættere end en perceptuel afstand på 0,02, tæller som én farve stavet to gange, og linteren fandt to næsten ens stavemåder, så snart den kørte. Gennemsigtighed skrives med relativ farvesyntaks frem for blandingsfunktionen, af en grund, der opsummerer hele siden: vagten kan ikke se gennem en blanding.

Hvor det går galt

En deklaration forsvinder, og siden bliver stadig vist

Ét forkert tegn gør en deklaration ugyldig, og browseren springer den over uden at stoppe:

```
.card {
    padding: 16; /* en længde kræver en enhed: ignoreres */
    colour: #222; /* ikke et egenskabsnavn: ignoreres */
    margin-block: 1rem; /* anvendes */
}
```

Kortet vises med sin margen og uden sin padding og sin farve, og browseren melder ingen fejl. Lint kilden i buildet, og lad dens fejl stoppe den. Hvor en minificering, en præprocessor eller en bundler skriver CSS'en om, efter at linteren har kørt, så kontrollér også outputtet, for det er den fil, browseren læser.

En regel taber til en stærkere

En komponents regel møder en mere specifik selektor fra et andet sted, eller en lige så stærk, der indlæses senere, og taber. Den sædvanlige refleks er en længere selektor eller et `!important`, som vinder i dag og gør den næste ændring dyrere. Hold specificiteten flad, med klasser frem for id'er og lidt indlejring, og brug kaskadelag til at lægge hele grupper som reset, base, komponenter og utilities i en eksplicit rækkefølge, som specificitet ikke kan vælte. For `!important`-deklarationer går lagrækkefølgen den anden vej, så det tidligste lag vinder, og det er værd at vide, før man griber efter det.

Stylesheetet vokser kun

Nyt arbejde tilføjer regler, fordi det tager længere tid at bevise en gammel regel ubrugt end at skrive en ny. Værdier bliver tastet igen som literaler ved siden af de tokens, der definerer dem, og næsten ens farver formerer sig, til ingen kan se, hvilken der er ment. Hold fælles værdier i custom properties, mål, hvilke regler et sæt sider faktisk bruger, og slet som en del af arbejdet, med et størrelsesbudget i buildet, der gør væksten synlig.

Layoutet holder kun ved én størrelse

Et design, der er kontrolleret ved én bredde med kort engelsk tekst, går i stykker under længere oversættelser, forstørret tekst og smalle telefoner. Faste højder klipper deres indhold, og faste bredder tvinger vandret rulning. Retningslinjer for tilgængelighed beder om tekst, der kan forstørres til 200 %, og indhold, der flyder om ved en bredde på 320 CSS-pixels. Lad bokse få deres størrelse fra indholdet, brug `min()`, `max()` og `clamp()` til flydende værdier, og test med lange strenge og zoom, før et layout kaldes færdigt.

Læs mere

- CSS Snapshot 2026 — hvilke moduler der udgør CSS i dag, og hvorfor sproget har niveauer i stedet for versioner.
- CSS Cascading and Inheritance Level 5 — hele sorteringsrækkefølgen, kaskadelag og arv, trin for trin.
- Selectors Level 4 — hver selektor, og hvordan dens specificitet beregnes.
- Syntax and basic data types — CSS 2.1's regler for at ignorere ukendte egenskaber og ulovlige værdier, med eksempler.
- CSS Values and Units Module Level 4 — længder, vinkler, `clamp()`, og hvor et nul uden enhed stadig er tilladt.

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 CSS?« 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/css/