**Engineer**

# CSS

die Sprache für das Aussehen einer Seite

## Was es ist

CSS, kurz für Cascading Style Sheets und noch immer oft CSS3 genannt, ist die Regelsprache, mit der ein Browser festlegt, wie jedes Element eines Dokuments gezeichnet wird: seine Farben, Größe, Abstände, Position, Schrift und Bewegung, ob das Dokument auf einem Smartphone, einem breiten Monitor oder einem Blatt Papier erscheint.

Ein Stylesheet ist eine Liste von Regeln. Jede Regel verbindet einen Selektor, der Elemente auswählt, mit Deklarationen, die jeweils aus einer Eigenschaft und einem Wert bestehen:

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

Das erste Wort im Namen prägt die tägliche Arbeit. Regelmäßig beanspruchen mehrere Regeln dieselbe Eigenschaft desselben Elements, und die Kaskade entscheidet, welche gewinnt. Ihre wichtigsten Schritte sind der Reihe nach Herkunft und Wichtigkeit (die Vorgaben des Browsers, die eigenen Einstellungen des Lesers, das Stylesheet der Website und die Frage, ob eine Deklaration mit `!important` markiert ist), dann die Kaskadenebene, dann die Spezifität und schließlich die Reihenfolge des Auftretens. Dabei schlägt eine spätere Regel eine frühere mit gleichem Gewicht. Die Spezifität wird pro Selektor mit drei Zahlen gezählt: zuerst IDs, dann Klassen, Attribute und Pseudoklassen, dann Elementnamen. Der Universalselektor `*` trägt zu keiner davon etwas bei. Neben der Kaskade wirkt die Vererbung: Manche Werte, etwa Farbe und Schrift, gehen auf die Kindelemente über, andere, etwa Außenabstände und Rahmen, bleiben dort, wo sie festgelegt wurden.

CSS hat keine einheitliche Versionsnummer. Nach CSS 2.1 wechselte die Arbeitsgruppe zu Modulen: Level 3 baut Modul für Modul auf 2.1 auf. Danach erhält jedes Modul seine eigene Stufe, sodass Selectors als Entwurf auf Level 4 steht, während Module ohne älteren Vorgänger bei Level 1 beginnen. „CSS3“ ist die verbreitete Bezeichnung für diese modulare Ära. Der jährliche Snapshot, der das aktuelle CSS definiert, stellt ausdrücklich fest, dass es kein CSS Level 4 gibt.

Die Regel mit dem größten praktischen Gewicht ist der Umgang der Sprache mit Fehlern. Trifft ein Browser auf eine unbekannte Eigenschaft oder einen für diese Eigenschaft ungültigen Wert, muss er diese eine Deklaration ignorieren und den Rest der Regel anwenden. So lassen sich neue Eigenschaften und Werte ergänzen, ohne die Funktion älterer Browser zu beeinträchtigen, und eine Website kann Funktionen nutzen, die nur manche Browser unterstützen. Der Preis ist das Schweigen. Die Seite wird etwas falsch dargestellt. Die einzige Spur ist eine durchgestrichene Zeile in den Entwicklerwerkzeugen, nach der jemand gezielt suchen muss.

Das meiste, was CSS gegenübergestellt wird, erweist sich als eine Art, CSS zu schreiben. Sass und Less sind Präprozessoren, die zu CSS kompilieren. Utility-Frameworks wie Tailwind erzeugen es aus Klassennamen im Markup. CSS-in-JS-Bibliotheken geben es aus JavaScript-Komponenten aus, und ein `style`-Attribut schreibt Deklarationen direkt an ein Element. Alle verändern, wie CSS verfasst wird; alle enden bei Deklarationen, die dieselbe Kaskade auf genau dieselbe Weise ordnet.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist CSS?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Oberflächen, die schnell laden, auf dem Smartphone funktionieren und sich in allen Browsern Ihrer Kunden gleich verhalten.

**Unsere Leistungen entdecken** Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. Full-Stack-GIS-Kartenentwicklung geleitet — Full-Stack-Kartenentwicklung von der Datenbank bis zum Browser

**Full-Stack-GIS-Kartenentwicklung geleitet** Kontakt aufnehmen →

## Warum es wichtig ist

CSS zwingt ein Team zu der Entscheidung, wo eine Regel unbemerkt wirkungslos bleiben darf. Die Sprache verwirft, was sie nicht anwenden kann, und lässt eine stärkere Regel kommentarlos gewinnen. Deshalb enthält jedes etwas ältere

Stylesheet Deklarationen, die nichts bewirken. Das Team kann bestimmen, ob jemand davon erfährt und wann.

Barrierefreiheit hängt stärker davon ab, als die meisten erwarten. Kontrast, ein sichtbarer Fokusring, die Berücksichtigung reduzierter Bewegung, ein nutzbarer Modus mit erzwungenen Farben, Text, der sich vergrößern lässt: All das besteht jeweils aus wenigen Deklarationen und kann durch eine zufällig spezifischere Regel an anderer Stelle abgeschaltet werden. Für alle, die die Funktion nicht brauchen, sieht die Seite gut aus.

Auch die Wartungskosten hängen davon ab. Ein Stylesheet, dessen Verhalten niemand vorhersagen kann, wächst nur noch, weil das Löschen einer Regel eine ungeprüfte Seite beschädigen könnte. Am Ende besteht der größte Teil der Datei meist aus Überschreibungen früherer Überschreibungen. Jede Änderung entsteht durch das Ausprobieren von Werten, bis der Bildschirm richtig aussieht.

Am unmittelbarsten betrifft es die Prüfung. Ein Browser, der Akzeptiertes anwendet und den Rest ignoriert, kann nicht selbst als Test dienen. Ein Linter erkennt im Quelltext Tippfehler und ungültige Werte vor der Auslieferung. Er sieht weder, was ein Minifier oder ein anderer Buildschritt danach mit der Datei macht, noch, welche von zwei gültigen Regeln auf einer bestimmten Seite gewinnt. Prüfungen der ausgelieferten Datei und visuelle Vergleiche gerenderter Seiten decken ab, was der Linter nicht sehen kann.

Die Kaskade früh zu ordnen, mit flacher Spezifität, benannten Ebenen und in Custom Properties gehaltenen Designtokens, verlangt bei jeder Änderung etwas Disziplin. Ein ohne diese Disziplin gewachsenes Stylesheet neu zu ordnen, kostet meist eine vollständige Neufassung.

## Wie es in der Praxis aussieht

Die Credits-Seite dieser Website beschreibt ein einziges CSS-Bundle, das beim Build zusammengefügt und mit einem Fingerprint versehen wird. Die handgezeichneten Pfeile und Randnotizen entstehen allein in CSS, und Stylelint setzt die Regeln für den Quelltext in der verbindlichen Qualitätsprüfung durch. Dieselbe Seite verlinkt den W3C CSS Validator und kündigt an, was er melden wird. Im August 2026 deckte dessen Profil `system-ui`, `@property`, `clamp()` in einem `font-size` und scrollgesteuerte Animationen noch nicht ab. Für /terms/ meldete er deshalb 176 Beanstandungen, davon 175 zu Funktionen, die Browser bereits ausliefern und die Website verwendet. Die verbleibende war echt. Bei

`rotate: 0deg` für den Ruhezustand entfernte der Minifier die Einheit. `rotate: 0` ist kein Winkel, und der Browser verwarf die ganze Deklaration. So wurde eine Animation ohne die Drehung ausgeliefert, mit der sie geschrieben worden war. Die Wertespezifikation erlaubt eine einheitenlose Null für Winkel nur an wenigen historisch bedingten Stellen. Stylelint sah den Fehler nie, weil es den Quelltext liest und der Minifier ihn später umschreibt. Eine unternehmenseigene Prüfung liest jetzt, was tatsächlich ausgeliefert wurde.

Bei einer Arbeit zur Barrierefreiheit lag das Problem allein an der Spezifität. Die Website berücksichtigte den Wunsch nach reduzierter Bewegung mit einer universellen Regel, die alle Übergänge abschaltete. Unter emulierter Einstellung bewegten sich trotzdem noch 11 Selektoren. Ein Universalselektor hat die Spezifität null, also schlug ihn jede Regel mit einer Klasse. Die Korrektur war strukturell, nicht ein stärkerer Selektor: Alles, was sich bewegt, steht jetzt in einem einzigen Abschnitt gegen Ende des Stylesheets. Ein Linter meldet einen Fehler, wenn irgendwo sonst transform per Transition animiert wird. Farbübergänge sind erlaubt, weil sie keine Bewegung sind. Bewegung muss als Übergang statt als Keyframe-Animation geschrieben werden, weil nur ein Übergang sauber abbricht. Die dazu festgehaltene Lehre ist kurz: Eine Regel, die bei der Spezifität verliert, versagt lautlos.

Bei Farben driften Stylesheets zuerst auseinander. Die Palette dieser Website wurde auf 28 Farben begrenzt. Jede ist mit dem auf ihrer jeweiligen Fläche gemessenen Kontrastverhältnis dokumentiert, und 11 Markenwerte sind einmalig als Tokens deklariert. Zwei Farben mit einem wahrnehmungsbezogenen Abstand unter 0,02 zählen als eine Farbe in zwei Schreibweisen. Der Linter fand beim ersten Lauf zwei nahezu doppelte Schreibweisen. Transparenz wird mit relativer Farbsyntax statt mit der Mischfunktion geschrieben. Der Grund fasst die ganze Seite zusammen: Die Prüfung kann eine Mischung nicht durchschauen.

## Wo es schiefgeht

### Eine Deklaration verschwindet, und die Seite wird trotzdem dargestellt

Ein falsches Zeichen macht eine Deklaration ungültig, und der Browser überspringt sie, ohne anzuhalten:

```
.card {
    padding: 16; /* a length needs a unit: ignored */
    colour: #222; /* not a property name: ignored */
```

```
    margin-block: 1rem; /* applied */
}
```

Die Karte wird mit Außenabstand, aber ohne ihren Innenabstand oder ihre Farbe dargestellt. Der Browser schlägt keinen Alarm. Prüfen Sie den Quelltext beim Build mit einem Linter und lassen Sie dessen Fehler den Build stoppen. Wenn ein Minifier, Präprozessor oder Bundler CSS nach dem Linter umschreibt, prüfen Sie auch die Ausgabe. Diese Datei liest der Browser.

## Eine Regel verliert gegen eine stärkere

Die Regel einer Komponente trifft auf einen spezifischeren Selektor von anderswo oder einen gleich starken, der später geladen wird, und verliert. Der übliche Reflex ist ein längerer Selektor oder ein `!important`. Das gewinnt heute und verteuert die nächste Änderung. Halten Sie die Spezifität flach, mit Klassen statt IDs und wenig Verschachtelung. Verwenden Sie Kaskadenebenen, um ganze Gruppen wie Reset, Basis, Komponenten und Utilities in eine ausdrückliche Reihenfolge zu bringen, die Spezifität nicht umkehren kann. Bei `!important`-Deklarationen gilt die umgekehrte Ebenenreihenfolge: Die früheste Ebene gewinnt. Das sollte man wissen, bevor man dazu greift.

## Das Stylesheet wächst immer weiter

Bei neuen Arbeiten werden weitere Regeln angehängt, weil der Nachweis, dass eine alte ungenutzt ist, länger dauert als eine weitere Regel zu schreiben. Werte werden als Literale neben den Tokens erneut eingetragen, die sie definieren. Fast identische Farben vermehren sich, bis niemand mehr weiß, welche gemeint ist. Halten Sie gemeinsame Werte in Custom Properties, messen Sie, welche Regeln eine Gruppe von Seiten tatsächlich verwendet, und löschen Sie im Rahmen der Arbeit. Ein Größenbudget im Build macht das Wachstum sichtbar.

## Das Layout funktioniert nur bei einer Größe

Ein bei einer einzigen Breite mit kurzem englischem Text geprüftes Design bricht bei längeren Übersetzungen, vergrößertem Text und schmalen Smartphones auseinander. Feste Höhen schneiden Inhalte ab, feste Breiten erzwingen horizontales Scrollen. Die Richtlinien zur Barrierefreiheit verlangen auf 200 % vergrößerbaren Text und Inhalte, die sich bei einer Breite von 320 CSS-Pixeln neu anordnen. Lassen Sie die Inhalte die Größe ihrer Boxen bestimmen, verwenden Sie `min()`, `max()` und `clamp()` für fließende Werte und prüfen Sie lange Zeichenfolgen sowie Zoom, bevor Sie ein Layout für fertig erklären.

# Weiterführende Lektüre

- CSS Snapshot 2026 — welche Module CSS heute ausmachen und warum die Sprache Stufen statt Versionen hat.
- CSS Cascading and Inheritance Level 5 — die vollständige Sortierreihenfolge, Kaskadenebenen und Vererbung, Schritt für Schritt.
- Selectors Level 4 — sämtliche Selektoren und die Berechnung ihrer Spezifität.
- Syntax und grundlegende Datentypen — die CSS-2.1-Regeln zum Ignorieren unbekannter Eigenschaften und unzulässiger Werte, mit Beispielen.
- CSS Values and Units Module Level 4 — Längen, Winkel, `clamp()` und die Stellen, an denen eine einheitenlose Null noch zulässig ist.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Arbeitsablauf, die vorhandenen Schnittstellen und die Nutzer, die die Veränderung benötigen. Wir können eine prüfbare Lieferung definieren und die Übergabe so gestalten, dass Ihr Team die gelieferten Bestandteile warten kann.

**Preise ansehen** Kontakt aufnehmen →

## Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist CSS?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

**Unsere Leistungen entdecken**

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/css/