Engineer

CSS

langage qui décrit la présentation d'une page

Ce que c’est

CSS, abréviation de Cascading Style Sheets et encore souvent appelé CSS3, est le langage de règles qu’un navigateur applique à un document pour déterminer comment dessiner chaque élément : couleurs, dimensions, espacements, position, police et mouvement, que le document soit affiché sur un téléphone, un grand écran ou une feuille de papier.

Une feuille de style est une liste de règles. Chaque règle associe un sélecteur, qui désigne les éléments, à des déclarations, chacune composée d’une propriété et d’une valeur :

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

Le premier mot du nom est celui qui façonne le travail quotidien. Plusieurs règles revendiquent régulièrement la même propriété du même élément, et la cascade décide laquelle l’emporte. Ses grandes étapes sont, dans l’ordre, l’origine et l’importance (les styles natifs du navigateur, les réglages du lecteur, la feuille de style du site et la présence éventuelle de `!important`), puis la couche de cascade, la spécificité et enfin l’ordre d’apparition, où une règle plus tardive l’emporte sur une règle antérieure de même poids. La spécificité de chaque sélecteur se compte en trois nombres : les identifiants, puis les classes, attributs et pseudo-classes, puis les noms d’éléments. Le sélecteur universel `*` n’ajoute rien à aucun de ces nombres. L’héritage fonctionne à côté de la cascade : certaines valeurs, comme la couleur et la police, se transmettent aux enfants d’un élément, tandis que d’autres, comme les marges et les bordures, restent là où elles ont été définies.

CSS n’a pas de numéro de version unique. Après CSS 2.1, le groupe de travail est passé aux modules : le niveau 3 prolonge 2.1 module par module, puis chaque module avance à son propre niveau. Le module Selectors est ainsi à l’état de projet au niveau 4, tandis que les modules sans équivalent antérieur commencent au niveau 1. « CSS3 » est le nom courant de cette époque modulaire, et l’état des lieux annuel qui définit le CSS actuel dit explicitement qu’il n’existe pas de CSS de niveau 4.

La règle qui pèse le plus en pratique concerne le traitement des erreurs. Un navigateur qui rencontre une propriété inconnue, ou une valeur invalide pour sa propriété, doit ignorer cette seule déclaration et appliquer le reste de la règle. Cela permet d’ajouter des propriétés et des valeurs sans casser les anciens navigateurs, et d’utiliser sur un site une fonctionnalité que seuls certains navigateurs prennent en charge. Le prix en est le silence. La page s’affiche, légèrement de travers, et la seule trace est une ligne barrée dans les outils de développement que quelqu’un doit aller chercher.

La plupart des solutions comparées à CSS sont en réalité des façons de l’écrire. Sass et Less sont des préprocesseurs qui compilent vers CSS. Les frameworks utilitaires comme Tailwind le génèrent à partir des noms de classes dans le balisage. Les bibliothèques CSS-in-JS le produisent depuis des composants JavaScript, et un attribut `style` place directement des déclarations sur un élément. Chacune de ces solutions change l’écriture de CSS ; toutes aboutissent à des déclarations que la cascade départage exactement de la même façon.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que CSS ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Des interfaces rapides, utilisables sur téléphone et cohérentes dans tous les navigateurs de vos clients.

Découvrir nos prestations Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. Développement cartographique GIS full-stack — cartographie full-stack, de la base au navigateur

Développement cartographique GIS full-stack Contactez-nous →

Pourquoi c’est important

La décision que CSS impose à une équipe est de savoir où une règle peut échouer sans bruit. Le langage abandonne ce qu’il ne peut appliquer et laisse une règle plus forte gagner sans commentaire ; toute feuille de style, quel que soit son âge,

contient donc des déclarations sans effet. L'équipe peut décider si quelqu'un le découvrira, et quand.

L'accessibilité en dépend davantage qu'on ne le pense. Un contraste, un indicateur de focus visible, le respect d'une demande de mouvement réduit, un mode de couleurs forcées utilisable, un texte qui supporte l'agrandissement : chacun tient en quelques déclarations, et chacun peut être désactivé par une règle ailleurs qui se trouve être plus spécifique. La page paraît correcte à tous ceux qui n'avaient pas besoin de la fonctionnalité.

Le coût d'entretien en dépend aussi. Une feuille de style dont personne ne peut prévoir le comportement ne fait que grossir, car supprimer une règle pourrait casser une page que personne n'a vérifiée. L'issue habituelle est un fichier composé surtout de surcharges de surcharges précédentes, où chaque changement se fait en essayant des valeurs jusqu'à ce que l'écran paraisse juste.

La vérification en dépend le plus directement. Un navigateur qui accepte ce qu'il peut et ignore le reste ne peut pas faire office de test. Un linter qui lit le code source repère les fautes de frappe et les valeurs invalides avant livraison. Il ne peut voir ce qu'un minificateur ou une autre étape de construction fait ensuite au fichier, ni laquelle de deux règles valides l'emporte sur une page donnée. Les contrôles du fichier servi et la comparaison visuelle des pages rendues couvrent ce qu'il ne voit pas.

Organiser tôt la cascade, avec une spécificité uniforme, des couches nommées et des jetons de conception conservés dans des propriétés personnalisées, demande un peu de discipline à chaque changement. Réorganiser une feuille de style qui a grandi sans cette discipline coûte généralement une réécriture.

À quoi cela ressemble en pratique

La page des crédits du site décrit un bundle CSS unique, concaténé et pourvu d'une empreinte à la construction, avec les flèches dessinées à la main et les notes en marge réalisées uniquement en CSS. Stylelint impose ses règles au code source lors du contrôle qualité obligatoire. La même page renvoie au validateur CSS du W3C et prévient le lecteur de son verdict. En août 2026, son profil ne couvrait pas `system-ui`, `@property`, `clamp()` dans un `font-size` ni les animations pilotées par le défilement ; sur /terms/, il a donc signalé 176 problèmes, dont 175 concernaient des fonctionnalités livrées dans les navigateurs et utilisées par le site. Le dernier était réel. Un `rotate: 0deg` au repos

avait perdu son unité dans le minificateur ; `rotate: 0` n'est pas un angle, et le navigateur avait ignoré toute la déclaration. Une animation avait donc été livrée sans la rotation prévue. La spécification des valeurs n'autorise un zéro sans unité pour un angle que dans quelques cas hérités. Stylelint n'avait jamais vu le problème, car il lit le code source avant la réécriture par le minificateur ; un contrôle propre à l'entreprise lit désormais ce qui a réellement été servi.

La spécificité expliquait tout dans un travail d'accessibilité. Le site respectait la préférence de mouvement réduit avec une règle universelle désactivant toutes les transitions ; pourtant, avec cette préférence émulée, 11 sélecteurs produisaient encore des animations. Un sélecteur universel vaut zéro ; toute règle avec une classe le battait donc. La correction était structurelle, plutôt qu'un sélecteur plus fort : tout ce qui bouge se trouve désormais dans une partie tardive de la feuille de style, et un linter fait échouer le contrôle s'il trouve ailleurs une transformation avec transition. Les fondus de couleur passent, car ce ne sont pas des mouvements. Le mouvement doit être écrit comme une transition, et non comme une animation par images-clés, car seule la transition s'annule proprement. La leçon consignée est courte : une règle qui perd sur la spécificité échoue sans bruit.

La couleur est le premier point où les feuilles de style divergent, et la palette du site a été limitée à 28 couleurs, chacune documentée avec le rapport de contraste mesuré sur la surface où elle apparaît, et 11 valeurs de marque déclarées une seule fois comme jetons. Deux couleurs dont la distance perceptuelle est inférieure à 0,02 comptent comme une même couleur écrite deux fois, et le linter a trouvé deux quasi-doublons dès sa première exécution. La transparence utilise la syntaxe de couleur relative plutôt que la fonction de mélange, pour une raison qui résume toute la page : le contrôle ne peut pas voir à travers un mélange.

Où les problèmes surviennent

Une déclaration disparaît et la page s'affiche quand même

Un caractère erroné rend une déclaration invalide, et le navigateur la saute sans s'arrêter :

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

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

La carte s'affiche avec sa marge, sans son espacement intérieur ni sa couleur, et le navigateur ne donne aucune alerte. Analysez le code source avec un linter pendant la construction et faites échouer celle-ci en cas d'erreur. Si un minificateur, un préprocesseur ou un outil de regroupement réécrit le CSS après le linter, vérifiez aussi la sortie : c'est ce fichier que lit le navigateur.

Une règle perd face à une règle plus forte

La règle d'un composant rencontre un sélecteur plus spécifique venu d'ailleurs, ou un sélecteur équivalent chargé plus tard, et perd. Le réflexe habituel est d'allonger le sélecteur ou d'ajouter un `!important`, ce qui gagne aujourd'hui et renchérit le changement suivant. Gardez une spécificité uniforme, avec des classes plutôt que des identifiants et peu d'imbrication. Les couches de cascade permettent d'ordonner explicitement des groupes entiers, comme la réinitialisation, la base, les composants et les utilitaires, sans que la spécificité puisse renverser cet ordre. Pour les déclarations `!important`, l'ordre des couches s'inverse et la première gagne ; mieux vaut le savoir avant d'y recourir.

La feuille de style ne fait que grossir

Les nouveaux travaux ajoutent des règles, car prouver qu'une ancienne est inutilisée prend plus de temps qu'en écrire une autre. Les valeurs sont ressaisies en littéraux à côté des jetons qui les définissent, et des couleurs presque identiques se multiplient jusqu'à ce que personne ne sache laquelle est voulue. Conservez les valeurs partagées dans des propriétés personnalisées, mesurez les règles réellement utilisées par un ensemble de pages et supprimez-en dans le cadre du travail. Un budget de taille dans la construction rend la croissance visible.

La mise en page ne tient qu'à une taille

Une mise en page vérifiée à une seule largeur avec un court texte anglais se dégrade avec des traductions plus longues, du texte agrandi et des téléphones étroits. Les hauteurs fixes coupent le contenu, et les largeurs fixes imposent un défilement horizontal. Les recommandations d'accessibilité demandent un texte agrandissable à 200 % et un contenu qui se redistribue à une largeur de 320 pixels CSS. Dimensionnez les boîtes d'après leur contenu, utilisez `min()`, `max()` et `clamp()` pour des valeurs fluides, puis testez les textes longs et le zoom avant de considérer la mise en page comme terminée.

Pour aller plus loin

- CSS Snapshot 2026 — les modules qui composent le CSS actuel et les raisons pour lesquelles le langage a des niveaux plutôt que des versions.
- CSS Cascading and Inheritance Level 5 — l'ordre de départage complet, les couches de cascade et l'héritage, étape par étape.
- Selectors Level 4 — tous les sélecteurs et le calcul de leur spécificité.
- Syntaxe et types de données de base — les règles de CSS 2.1 pour ignorer les propriétés inconnues et les valeurs illégales, avec des exemples.
- CSS Values and Units Module Level 4 — longueurs, angles, `clamp()` et cas où le zéro sans unité reste autorisé.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le processus de travail, les interfaces existantes et les utilisateurs qui ont besoin du changement. Nous pouvons définir une livraison testable et une transmission permettant à votre équipe d'assurer la maintenance des éléments livrés.

Consulter les tarifs Contactez-nous →

Rendez la terminologie concrète

Demandez comment « Qu'est-ce que CSS ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

Découvrir nos prestations

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

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

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/css/