Engineer

CSS

мова, яка описує вигляд сторінки

Що це таке

CSS, скорочення від Cascading Style Sheets, яке досі часто називають CSS3, є мовою правил, яку браузер накладає на документ, щоб вирішити, як намалювати кожен елемент: його кольори, розмір, відступи, положення, шрифт і рух, хоч би де документ опинився, на телефоні, широкому моніторі чи аркуші паперу.

Таблиця стилів є списком правил. Кожне правило поєднує селектор, який вибирає елементи, з оголошеннями, і кожне оголошення складається з властивості та значення:

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

Перше слово назви описує ту частину, що визначає щоденну роботу. Кілька правил раз у раз претендують на ту саму властивість того самого елемента, і каскад вирішує, яке з них перемагає. Його основні кроки по черзі такі: походження й важливість (типові стилі браузера, власні налаштування читача, таблиця стилів сайту і те, чи позначено оголошення як `!important`), далі каскадний шар, далі специфічність і наостанок порядок появи, де пізніше правило перемагає раніше з такою самою вагою. Специфічність рахують для кожного селектора як три числа: ідентифікатори, потім класи, атрибути й псевдокласи, потім імена елементів. Універсальний селектор `*` нічого не додає до жодного з них. Поруч із каскадом діє успадкування: деякі значення, як-от колір і шрифт, переходять до дочірніх елементів, а інші, як-от зовнішні відступи й рамки, лишаються там, де їх задали.

CSS не має єдиного номера версії. Після CSS 2.1 робоча група перейшла на модулі: Level 3 надбудовується над 2.1 модуль за модулем, а далі кожен модуль отримує власні рівні, тож Selectors уже має чернетку Level 4, а модулі, яких у CSS 2.1 не було, починаються з Level 1. «CSS3» є популярною назвою цієї модульної доби, а щорічний знімок, що визначає сучасний CSS, прямо каже, що CSS Level 4 не існує.

Правило з найбільшою практичною вагою стосується того, як мова поводиться з помилками. Браузер, який натрапляє на невідому властивість або на значення, недійсне для своєї властивості, мусить проігнорувати саме це оголошення й застосувати решту правила. Це правило існує, щоб нові властивості й значення можна було додавати, не ламаючи старіших браузерів, і саме воно дає сайту змогу використовувати можливість, яку підтримують лише деякі браузери. Ціна цього — тиша. Сторінка відображається, трохи неправильно, і єдиний слід — закреслений рядок в інструментах розробника, який хтось має піти й пошукати.

Більшість того, з чим порівнюють CSS, виявляється способом його писати. Sass і Less є препроцесорами, що компілюються в CSS. Утилітарні фреймворки на кшталт Tailwind генерують його з імен класів у розмітці. Бібліотеки CSS-in-JS видають його з компонентів JavaScript, а атрибут `style` ставить оголошення прямо на один елемент. Кожен із цих способів змінює те, як CSS пишуть; кожен закінчується оголошеннями, які каскад сортує точнісінько так само.

Застосуйте поняття до рішення

Застосуйте «Що таке CSS?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Інтерфейси, що швидко завантажуються, працюють на телефоні й поводяться однаково в усіх браузерах ваших клієнтів.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Керувала full-stack-розробкою GIS-карт — full-stack-картографія — від бази даних до браузера

Керувала full-stack-розробкою GIS-карт Зв'язатися з нами →

Чому це важливо

Рішення, яке CSS нав’язує команді, полягає в тому, де правилу дозволено відмовляти мовчки. Мова відкидає те, чого не може застосувати, і без коментарів дає сильнішому правилу перемогти, тож кожна таблиця стилів

певного віку містить оголошення, які нічого не роблять. Команда керує лише тим, чи хтось про це дізнається, і коли.

Доступність залежить від цього більше, ніж очікує більшість людей. Контраст, видиме кільце фокусу, повага до запиту на зменшений рух, придатний режим примусових кольорів, текст, що витримує збільшення: кожне з цього — жменька оголошень, і кожне може вимкнути правило деінде, яке випадково виявилося специфічнішим. Сторінка добре виглядає для всіх, кому ця можливість не була потрібна.

Вартість підтримки теж залежить від цього. Таблиця стилів, поведінку якої ніхто не може передбачити, лише росте, бо видалення правила може зламати сторінку, яку ніхто не перевірив. Звичний кінцевий стан — файл, де більша частина ваги припадає на перевизначення попередніх перевизначень, а кожну зміну роблять, перебираючи значення, доки екран не виглядатиме правильно.

Перевірка залежить від цього найпряміше. Браузер, який приймає те, що може, і ігнорує решту, не може бути тестом. Лінтер, що читає джерело, ловить описки й недійсні значення ще до випуску. Він не бачить, що мініфікатор чи інший крок збирання робить із файлом потім, і не бачить, яке з двох дійсних правил перемагає на конкретній сторінці. Перевірки файла, який справді віддається браузеру, і візуальне порівняння відрендерених сторінок покривають те, чого лінтер не може.

Організувати каскад рано, з пласкою специфічністю, іменованими шарами й токенами дизайну в користувацьких властивостях (custom properties), коштує трохи дисципліни на кожну зміну. Реорганізація таблиці стилів, що виросла без такої дисципліни, зазвичай коштує переписування.

Як це виглядає на практиці

Сторінка подяк самого сайту описує один пакет CSS, з’єднаний і позначений відбитком під час збирання, у якому намальовані від руки стрілки й нотатки на полях зроблено самим лише CSS, а Stylelint у шлюзі якості тримає джерело в межах своїх правил. Та сама сторінка посилається на CSS-валідатор W3C і попереджає читача, що він скаже. Станом на серпень 2026 року його профіль не доходить до `system-ui`, `@property`, `clamp()` у `font-size` та анімацій, керованих прокруткою, тож на /terms/ він видав 176 зауважень, і 175 з них стосувалися можливостей, які браузери підтримують,

а сайт використовує. Одне, що лишилося, було справжнім. `rotate: 0deg` у стані спокою втратив одиницю під час мініфікації, `rotate: 0` не є кутом, і браузер відкинув усе оголошення, тож анімація дійшла до відвідувачів без обертання, з яким її написали. Специфікація значень дозволяє нуль без одиниці як кут лише в кількох місцях, і то заради сумісності зі старим кодом. Stylelint цієї проблеми не побачив, бо читає джерело, а мініфікатор переписує його пізніше; тепер власна перевірка компанії читає те, що справді віддано браузеру.

Специфічність становила всю історію в одній роботі з доступності. Сайт шанував уподобу зменшеного руху одним універсальним правилом, що вимикало кожен перехід, і за емульованої вподоби 11 селекторів усе одно анімувалися. Універсальний селектор набирає нуль, тож будь-яке правило з класом його перемагало. Виправлення було структурним, а не сильнішим селектором: усе, що рухається, тепер живе в одній пізній частині таблиці стилів, а лінтер валить перехід на transform будь-де інде. Згасання кольору проходять, бо це не рух, а рух треба записувати як перехід, а не як анімацію за ключовими кадрами, бо саме перехід скасовується чисто. Урок, записаний разом із роботою, короткий: правило, що програє за специфічністю, відмовляє мовчки.

Колір — це те, у чому таблиці стилів розповзаються найперше, і палітру сайту закрито на 28 кольорах. Для кожного задокументовано коефіцієнт контрасту, виміряний на поверхні, де він з’являється, а 11 брендових значень оголошено один раз як токени. Два кольори, ближчі за перцептивну відстань 0,02, вважаються одним кольором, записаним двічі, і лінтер знайшов два майже однакові написання, щойно його запустили. Прозорість записують відносним колірним синтаксисом, а не функцією змішування, з причини, що підсумовує всю сторінку: вартовий не бачить крізь суміш.

Де виникають проблеми

Оголошення зникає, а сторінка однаково відображається

Один хибний символ робить оголошення недійсним, і браузер пропускає його, не зупиняючись:

```
.card {
    padding: 16; /* довжині потрібна одиниця: ігнорується */
    colour: #222; /* це не назва властивості: ігнорується */
```

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

Картка відображається із зовнішнім відступом, але без внутрішнього відступу й кольору, а браузер про це ніяк не попереджає. Перевіряйте джерело лінтером під час збирання й зупиняйте збирання на його помилках. Якщо мініфікатор, препроцесор чи бандлер переписує CSS після того, як лінтер відпрацював, перевіряйте й результат, бо саме цей файл читає браузер.

Правило програє сильнішому

Правило компонента зустрічає специфічніший селектор звідкись іще або такий самий, завантажений пізніше, і програє. Звичний рефлекс — довший селектор або `!important`, що перемагає сьогодні й підвищує ціну наступної зміни. Тримайте специфічність пласкою, з класами замість ідентифікаторів і мінімальною вкладеністю, і використовуйте каскадні шари, щоб розставити цілі групи, як-от reset, base, components і utilities, у явному порядку, який специфічність не може перевернути. Для оголошень `!important` порядок шарів працює навпаки, і перемагає найраніший шар, що варто знати, перш ніж за ним тягнутися.

Таблиця стилів лише росте

Нова робота додає правила, бо довести, що старе правило не використовується, довше, ніж написати ще одне. Значення переписують як літерали поруч із токенами, що їх визначають, а майже однакові кольори множаться, доки вже неможливо сказати, який із них мався на увазі. Тримайте спільні значення в користувацьких властивостях, вимірюйте, які правила справді використовує набір сторінок, і видаляйте як частину роботи, з бюджетом розміру у збиранні, щоб зростання було видно.

Макет тримається лише за одного розміру

Дизайн, перевірений за однієї ширини з коротким англійським текстом, ламається під довшими перекладами, збільшеним текстом і вузькими телефонами. Фіксовані висоти обрізають свій вміст, а фіксовані ширини змушують прокручувати вбік. Настанови з доступності просять тексту, який можна збільшити до 200%, і вмісту, що перетікає за ширини 320 CSS-пікселів. Задавайте розміри блоків від їхнього вмісту, використовуйте `min()`, `max()` і `clamp()` для плавних значень і тестуйте з довгими рядками та масштабуванням, перш ніж вважати макет готовим.

Що почитати далі

- CSS Snapshot 2026 — які модулі складають CSS сьогодні і чому мова має рівні замість версій.
- CSS Cascading and Inheritance Level 5 — повний порядок сортування, каскадні шари й успадкування, крок за кроком.
- Selectors Level 4 — кожен селектор і те, як обчислюється його специфічність.
- Syntax and basic data types — правила CSS 2.1 щодо ігнорування невідомих властивостей і неприпустимих значень, із прикладами.
- CSS Values and Units Module Level 4 — довжини, кути, `clamp()` і місця, де ще дозволено нуль без одиниці.

Словник

Обирайте за фактичними умовами

Опишіть робочий процес, наявні інтерфейси та користувачів, яким потрібна зміна. Ми можемо визначити перевірюваний результат і спосіб передачі, який дасть вашій команді змогу підтримувати його.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке CSS?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

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

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/css/