Engineer

Unicode

⌨ стандарт, що дає кожному символу власний номер

Що це таке

Unicode — це стандарт, що дає кожній літері, цифрі, розділовому знаку й символу писемностей світу власний номер. Українською його назву часто пишуть і кирилицею: Юнікод. Ті, хто супроводжує стандарт, називають його універсальним кодуванням символів: воно створене для того, щоб тексти мовами сучасного світу й тексти технічних галузей можна було передавати, обробляти й відображати будь-де, і поширене на класичні та історичні тексти багатьох писемних мов. Один репертуар вміщує скандинавську літеру å, українську Ґ, знак євро й літери писемностей, якими ніхто не пише вже багато століть, тож один документ або один стовпець бази даних може містити їх усі.

Формально версія стандарту Unicode складається з основної специфікації, кодових таблиць, додатків до стандарту, Unicode Standard Annexes, і Unicode Character Database, машинозчитуваних файлів, що описують властивості кожного символу. Технічний комітет Unicode постійно супроводжує стандарт, Консорціум Unicode його публікує, а версію 18.0 оголосили 16 вересня 2026 року.

Номер, який отримує символ, називають кодовою точкою. Кодовий простір охоплює значення від 0 до 10FFFF у шістнадцятковому записі, тобто 1 114 112 кодових точок, а кодову точку записують як U+ і щонайменше чотири шістнадцяткові цифри: å має кодову точку U+00E5, Ґ має U+0490, а знак євро U+20AC. Правила стабільності консорціуму обіцяють, що закодований символ ніколи не перемістять і не вилучать, а його назву не змінять, тож текст, закодований сьогодні, означатиме те саме в кожній наступній версії.

Більшість стандарту тримається на двох розрізненнях. Перше пролягає між символом і гліфом. Unicode кодує символ, абстрактну літеру, а форму залишає шрифту: основна специфікація прямо каже, що за вигляд гліфів відповідають виробники шрифтів і що він не є частиною стандарту. Та сама кодова точка може мати різний вигляд у двох гарнітурах, а гарнітура, у якій для неї немає форми, не зможе її намалювати взагалі. Друге розрізнення пролягає між кодовою точкою та байтами, які її зберігають. Unicode визначає три форми кодування, UTF-8, UTF-16 і UTF-32, що записують кожну кодову точку як одну або кілька одиниць завдовжки 8, 16 чи 32 біти, а специфікація називає пасткою спробу розрізняти «UTF-8 versus Unicode»: усі три форми однаково правильно реалізують ті самі символи.

Застосуйте поняття до рішення

Застосуйте «Що таке Unicode?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Програма, що природно читається на кожному ринку, де ви продаєте, — з правильними датами, валютами й сортуванням імен.

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. Створила англо-данську систему i18n — 12,027 повідомлень на локаль — англійською та данською

Створила англо-данську систему i18n Зв'язатися з нами →

Чому це важливо

До Unicode кожна писемність мала власні кодові сторінки, і байт нічого не означав, доки не стало відомо, яку сторінку вжито. У кириличній кодовій сторінці windows-1251 байт E5 означає літеру е; у західноєвропейській windows-1252 той самий байт означає å. Жодна з них не могла вмістити літер іншої: windows-1251 має Ґ, але не має ні å, ні æ, ні ø, а в windows-1252 навпаки. Продукт українською та данською потребував двох кодувань і ретельного обліку того, який текст яким кодуванням записано, а речення, що цитувало обидві мови, зберегти не вдавалося взагалі.

Unicode зробив цей облік зайвим. Усі мови, якими виходить продукт, і всі імена, які клієнт вводить у форму, вміщуються в один набір символів, а той самий номер означає той самий символ у кожній системі, що його читає. Саме це дає змогу базі даних, API та браузеру передавати текст одне одному, не домовляючись спершу про кодову сторінку.

Стандарт також вирішує питання, які інакше кожна програма вирішувала б по-своєму. Він об’єднує однакові символи різних мов: кома лишається тим самим символом незалежно від того, чи відокремлює вона тисячі, як в англійській, чи десяткові знаки, як в українській, тож значення визначає мова та її правила, а не кодова точка. Символів різних писемностей він не об’єднує: латинська о, кирилична о та грецька о в більшості шрифтів мають однаковий вигляд, але це три символи, бо вони належать трьом писемностям, а старіші кодування вже розрізняли їх. І він визначає, які послідовності еквівалентні. Å можна зберегти як готовий складений символ або як A, за яким іде комбінований знак кільця, і стандарт визначає ці дві форми як канонічно еквівалентні, тобто такі, що мають показуватися й тлумачитися однаково, а форми нормалізації зводять обидві до одного узгодженого написання.

Ця широта має ціну, і платить її програміст. Символ не є байтом, не завжди є однією одиницею в рядку й не завжди збігається з тим, що читач вважає літерою. Код, який плутає ці три речі, проходить тести на англійських даних і ламається на іменах, адресах і повідомленнях саме тих людей, для яких його написали.

Як це виглядає на практиці

JavaScript робить ці рівні видимими. Рядок у ньому складається з кодових одиниць UTF-16, а якщо розгорнути його в масив, він поділиться на кодові точки:

```
const hex = (n) => n.toString(16).toUpperCase().padStart(4, "0");
for (const text of ["a", "å", "Ґ", "€", "𐌂"]) {
  const points = [...text].map((c) => "U+" + hex(c.codePointAt(0)));
  console.log(text, points.join(" "), text.length, [...text].length);
}
```

Запущений у Node.js 25, код виводить кожен символ, його кодову точку, довжину в одиницях UTF-16 і довжину в кодових точках:

```
a U+0061 1 1
å U+00E5 1 1
Ґ U+0490 1 1
€ U+20AC 1 1
𐌂 U+10302 2 1
```

Останній рядок показує літеру староіталійського письма. Її кодова точка лежить вище за U+FFFF, тож UTF-16 зберігає її як дві одиниці, сурогатну пару, і `length` повідомляє 2 для того, що є одним символом.

Еквівалентність видно так само. Ось три рядки, які всі показуються як Å:

```
const forms = ["\u00C5", "A\u030A", "\u212B"];
console.log("form length equal NFC");
for (const [i, s] of forms.entries()) {
  console.log(i + 1, s.length, s === forms[0], s.normalize("NFC") === forms[0]);
}
```

```
form length equal NFC
1 1 true true
2 2 false true
3 1 false true
```

Перший рядок містить готову складену літеру U+00C5. Другий містить звичайне A, за яким іде U+030A, комбіноване кільце над літерою. Третій містить U+212B, знак ангстрема, який Unicode Character Database зводить до U+00C5. Якщо порівнювати їх як є, це три різні рядки; після нормалізації до NFC усі три стають одним рядком. На екрані їх ніщо не відрізняє, тож пошук, унікальний індекс або вхід до облікового запису, що порівнює сирі рядки, побачить три різні імена.

Навіть цей сайт залежить від кодових точок, лише в менш помітному місці. Нотатка про підкреслення, яке перекреслювало кожну літеру описує латинські та кириличні шрифти Ubuntu для основного тексту сайту як шість накреслень: три ступені насиченості для латиниці й три для кирилиці, щоб українська сторінка була набрана тим самим шрифтом, що й англійська. Кожне накреслення оголошує діапазон символів, які воно покриває. Ці діапазони спрямовують вибір шрифту для символів на сторінці; явно задане попереднє завантаження також може запросити файл ще до звичайного виявлення шрифту браузером. Кириличне оголошення задає свій діапазон у кодових точках:

```
unicode-range: U+0400-045F, U+0490-0491;
```

Перший діапазон охоплює символи від È до ц, зокрема українські Є, І та Ї разом з рештою основної кирилиці. Другий містить лише Ґ і ґ, які лежать далі в кириличному блоці. Діапазони вказують, яке накреслення можна вибрати для символу, але шрифт також має містити його гліф. Вони не гарантують, що сторінка завантажує файли лише для однієї системи письма. Англійський і данський текст використовує латинські накреслення, але назва української мови може потребувати кириличного, і сайт явно задає попереднє завантаження легкого кириличного накреслення основного тексту на некириличних сторінках для показу на екрані.

Де виникають проблеми

Довжину рахують у кодових одиницях. `length` у JavaScript рахує одиниці UTF-16, тож 𝐂 рахується за дві, а A з комбінованим кільцем теж за дві, хоча читач бачить одну літеру. Обмеження, яке перевіряють в одиницях, відхиляє текст, що насправді вміщується, а обрізання на фіксованій позиції може розрізати символ: `"A\u030A".slice(0, 1)` лишає A й губить кільце, а якщо розрізати 𝐂 навпіл, лишиться одинокий сурогат, який кодувальник UTF-8 замінить на U+FFFD, символ заміни. Для кожного обмеження вирішіть, що воно рахує: байти там, де межу задає зберігання, і символи такими, якими їх бачить читач, там, де межу задає людина. `Intl.Segmenter` рахує у другий спосіб: для слова «Київ», де ї записано як і з комбінованим знаком U+0308, він дає 4, а `length` дає 5. Обрізайте текст лише на межах, які він повідомляє.

Однаковий текст порівнюють одиниця за одиницею. Для `===`, для геш-функції та для унікального індексу три форми Å вище є трьома різними рядками, тож клієнт може зареєструвати те саме ім’я двічі або не знайти документ за назвою, яку сам набрав. Нормалізуйте текст до NFC там, де він потрапляє в систему, зберігайте його нормалізованим і порівнюйте нормалізовані форми. Правила стабільності гарантують, що рядок, нормалізований за однієї версії, лишається нормалізованим за всіх наступних, якщо він містить лише символи, відомі тій версії, тож код, який нормалізує, мусить бути не старішим за текст.

Схожі символи сприймають як однакові. Латинська o, кирилична о та грецька ο мають різні кодові точки, але більшість шрифтів малює їх однаково, тож два імена облікових записів або два артикули можуть мати однаковий вигляд і відрізнятися одним символом. Unicode свідомо тримає їх окремо, а стандарт консорціуму про механізми безпеки, UTS #39, описує, як

обмежити такі підробки. Там, де ім'я ідентифікує людину чи одержувача платежу, дозволяйте лише одну писемність в імені або перевіряйте ім'я за даними безпеки зі стандарту, перш ніж його прийняти.

Байти читають у неправильному кодуванні. Якщо текст, збережений в UTF-8, прочитати як windows-1251, `Привіт` перетвориться на `РџСЂРёРІС–С‚`, бо кожну кириличну літеру завдовжки два байти буде прочитано як дві однобайтові. Українською такий безлад здавна звуть кракозябрами. Слово Unicode не каже, як текст записано байтами; це каже лише форма кодування. Зазначайте її всюди, де текст перетинає межу: у заголовку HTTP, на сторінці, у з'єднанні з базою даних і у форматі файлу, а файл без неї вважайте питанням до людини, а не здогадом для коду.

У відсутньому гліфі звинувачують текст. Коли літера показується порожнім прямокутником, причина здебільшого в шрифті, у якому немає форми для цієї кодової точки, а сам текст цілий. Оскільки Unicode залишає форми шрифтам, кожен шрифт, яким користується продукт, мусить покривати всі символи його мов, українські Є, І, Ї та Ґ так само, як усю абетку від А до я, інакше браузер намалює їх резервним шрифтом, який знайде. Нотатка вище називає саме це причиною, з якої сайт узагалі віддає кириличні частини: без них українська сторінка була б набрана тим шрифтом, який запропонує система. Перевіряйте кожну мову з повною абеткою, а не одним зразковим реченням.

Імена перевіряють за абеткою від A до Z. Шаблон на кшталт `^[A-Za-z' -]+$` пропускає O'Neill і відхиляє Ольгу Ковальчук, Юлію Мар'янчук і Zoë, тобто саме ті імена, заради яких існує український продукт. Натомість визначайте, чи символ є літерою, за його властивостями Unicode: у JavaScript шаблон `/^[\p{L}\p{M}' -]+$/u` пропускає всі чотири. Апостроф в українському прізвищі трапляється ще у двох написаннях: як U+02BC, який Unicode вважає літерою і шаблон пропускає, і як типографський U+2019, який є розділовим знаком і який шаблон відхиляє. Тож або додайте U+2019 до шаблону, або зводьте всі три написання до одного ще до перевірки. Властивості беруться з копії Unicode Character Database у середовищі виконання. Наприклад, середовище Node.js 25.2.1, використане для наведених тут прикладів, повідомляє про Unicode 16.0; це конкретна збірка, а не гарантія для кожного випуску 25.x. Літери, додані в пізніших версіях Unicode, відхилятимуться, доки дані про символи в середовищі виконання не оновлять. Перевіряйте імена поблажливо, а складні випадки залишайте людині, не шаблону.

Що почитати далі

- About the Unicode Standard — опис стандарту від самого консорціуму: з чого складається версія і де публікують найновішу.
- The Unicode Standard, Chapter 2: General Structure — десять принципів проєктування, кодовий простір, три форми кодування й правила для еквівалентних послідовностей.
- Unicode Character Encoding Stability Policies — що вже ніколи не зміниться після того, як символ закодовано, від кодової точки й назви до нормалізації.
- Unicode 18.0.0 — що додає поточна версія і які технічні стандарти отримують нові версії разом із нею, зокрема сортування та безпека.
- Node.js: process.versions — звіт середовища виконання про версії компонентів; тут він прив’язує приклад із даними Unicode до конкретної збірки Node.js, а не до всієї гілки випусків.
- CSS Fonts: character ranges — як оголошені діапазони символів і фактичний набір гліфів шрифту разом визначають, які символи може відтворити накреслення.

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 Unicode?» 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/unicode/