Engineer

Unicode

⌨ Estándar que asigna un número propio a cada carácter

Qué es

Unicode es el estándar que asigna un número propio a cada letra, dígito, signo de puntuación y símbolo de los sistemas de escritura del mundo. Sus responsables lo denominan la codificación universal de caracteres: permite intercambiar, procesar y mostrar en cualquier lugar texto de las lenguas y los ámbitos técnicos del mundo moderno, y también representar escrituras antiguas e históricas. Un mismo repertorio contiene la å danesa, la ґ ucraniana, el signo del euro y letras de escrituras que nadie ha utilizado durante siglos, de modo que un solo documento o una sola columna de base de datos puede contenerlos todos.

En sentido estricto, una versión de Unicode incluye varias partes coordinadas: su especificación principal, las tablas de caracteres, los anexos y una base de datos de caracteres formada por archivos legibles por máquinas que registran las propiedades de cada carácter. El Comité Técnico de Unicode lo mantiene de forma continua, el Consorcio Unicode lo publica y la versión 18.0 se anunció el 16 de septiembre de 2026.

El número asignado a un carácter es su punto de código. El espacio de códigos va de 0 a 10FFFF en hexadecimal, con 1.114.112 puntos de código en total. Un punto de código se escribe con U+ seguido de al menos cuatro dígitos hexadecimales: å es U+00E5, ґ es U+0490 y el signo del euro es U+20AC. La política de estabilidad del Consorcio promete que un carácter codificado conserva su lugar para siempre: nunca se mueve ni se retira y su nombre nunca cambia, de modo que un texto codificado hoy mantiene su significado en todas las versiones posteriores.

Dos distinciones son fundamentales en el estándar. La primera separa el carácter del glifo. Unicode codifica el carácter, la letra abstracta, y deja su forma a la fuente: la especificación principal establece que las formas de los glifos son

responsabilidad de los proveedores de fuentes y no forman parte del estándar. El mismo punto de código puede verse distinto en dos tipografías, y una tipografía que no tenga una forma para él no puede dibujarlo. La segunda distingue el punto de código de los bytes que lo almacenan. Las tres formas de codificación de Unicode son UTF-8, UTF-16 y UTF-32, que escriben cada punto de código mediante una o más unidades de 8, 16 o 32 bits. La especificación advierte de que contraponer «UTF-8 frente a Unicode» es una trampa: las tres son formas igualmente válidas de implementar los mismos caracteres.

Utilice el concepto en una decisión

Aplique «¿Qué es Unicode?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Software que se lee con naturalidad en cada mercado donde vende, incluidas las fechas, las monedas y la forma de ordenar los nombres.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. i18n en inglés y danés — 12,027 mensajes por configuración regional en inglés y danés

i18n en inglés y danés Póngase en contacto →

Por qué importa

Antes de Unicode, cada escritura tenía sus propias páginas de códigos y un byte no significaba nada hasta saber qué página se usaba. En la página de Europa occidental windows-1252, el byte E5 es å; en la cirílica windows-1251, el mismo byte es la letra е. Ninguna podía contener las letras de la otra: windows-1251 tiene ґ y no tiene å, æ ni ø, y windows-1252 hace lo contrario. Un producto en danés y ucraniano necesitaba dos codificaciones y un registro fiable de cuál correspondía a cada texto; una oración que citara ambos idiomas no cabía en ninguna.

Unicode eliminó esa contabilidad. Todos los idiomas en los que se publica un producto y todos los nombres que un cliente escribe en un formulario caben en un único conjunto de caracteres, y el mismo número significa el mismo carácter en cualquier sistema que lo lea. Eso permite que una base de datos, una API y un navegador se pasen texto sin acordar primero una página de códigos.

El estándar también resuelve cuestiones que cada programa resolvería a su manera. Unifica caracteres entre idiomas: la coma es un mismo carácter tanto si separa millares, como en inglés, como si separa decimales, como en francés; el significado procede de la configuración regional, no del punto de código. No unifica entre escrituras: la o latina, la o cirílica y la o griega se parecen en la

mayoría de las tipografías, pero son tres caracteres porque pertenecen a tres escrituras y las codificaciones anteriores ya los distinguían. También registra qué secuencias son equivalentes. Å puede almacenarse como un carácter precompuesto o como A seguida de un anillo combinante. El estándar las define como canónicamente equivalentes, destinadas a mostrarse e interpretarse igual, y proporciona formas de normalización que convierten ambas en una representación acordada.

Esa generalidad tiene un precio, que paga quien programa. Un carácter no es un byte, no siempre es una unidad de una cadena y no siempre coincide con lo que una persona cuenta como una letra. El código que trata esas tres cosas como si fueran iguales supera sus pruebas con datos en inglés y falla con los nombres, las direcciones y los mensajes de las personas para las que se escribió.

Cómo se aplica en la práctica

JavaScript hace visibles las distintas capas. Una cadena es una secuencia de unidades de código UTF-16 y expandirla en una matriz la divide en puntos de código:

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

Al ejecutarlo en Node.js 25, imprime cada carácter, su punto de código, su longitud en unidades UTF-16 y su longitud en puntos de código:

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

La última línea es una letra de la antigua escritura itálica. Su punto de código está más allá de U+FFFF, así que UTF-16 la almacena en dos unidades, un par de sustitutos, y `length` devuelve 2 para un solo carácter.

La equivalencia se observa del mismo modo. Estas tres cadenas se muestran todas como 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
```

La primera es la letra precompuesta, U+00C5. La segunda es una A simple seguida de U+030A, el anillo combinante superior. La tercera es U+212B, el signo del ángstrom, que la base de datos de caracteres de Unicode relaciona con U+00C5. Comparadas tal como están, son tres cadenas distintas; después de normalizarlas a NFC, las tres son la misma. Nada en pantalla las distingue, así que una búsqueda, un índice único o un inicio de sesión que compare las cadenas sin normalizar ve tres nombres diferentes.

Este sitio depende de los puntos de código en un lugar menos visible. La nota sobre el subrayado que atravesaba todas las letras describe las fuentes Ubuntu del cuerpo del sitio, latinas y cirílicas, como seis variantes: tres grosores latinos y tres cirílicos, para componer una página ucraniana con la misma tipografía que una inglesa. Cada variante declara el rango de caracteres que cubre. Esos rangos orientan la selección de fuentes para los caracteres de la página; una precarga explícita también puede solicitar un archivo antes del descubrimiento normal de fuentes. La declaración cirílica expresa su rango en puntos de código:

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

El primer rango va de È a џ y contiene las letras ucranianas Є, І y Ї junto con el resto del alfabeto cirílico básico. El segundo solo contiene Ґ y ґ, situadas más adelante en el bloque cirílico. Los rangos declaran qué variante de fuente puede elegirse para un carácter, pero la fuente también debe contener su glifo. No garantizan que una página descargue archivos de una sola escritura. El texto

inglés y danés utiliza las variantes latinas, pero la etiqueta del idioma ucraniano puede necesitar una cirílica; además, el sitio precarga explícitamente la variante cirílica de grosor ligero para el cuerpo en las páginas no cirílicas destinadas a pantalla.

Dónde surgen los problemas

La longitud se cuenta en unidades de código. El `length` de JavaScript cuenta unidades UTF-16, por lo que ᳇ cuenta como dos y una A con un anillo combinante cuenta como dos donde quien lee ve una letra. Un límite expresado en unidades rechaza texto que cabría, y cortar en una posición fija puede partir un carácter: `"A\u030A".slice(0, 1)` conserva la A y elimina el anillo; cortar ᳇ por la mitad deja un sustituto aislado que el codificador UTF-8 reemplaza por U+FFFD, el carácter de sustitución. Decida qué cuenta cada límite: bytes cuando lo impone el almacenamiento, caracteres tal como se ven al leer cuando lo impone una persona. `Intl.Segmenter` cuenta estos últimos y, para Århus escrita con un anillo combinante, devuelve 5 donde `length` devuelve 6. Corte el texto solo en los límites que indique.

El texto equivalente se compara unidad por unidad. Las tres formas de Å anteriores son tres cadenas distintas para `===`, para un hash y para un índice único. Por eso, un cliente puede registrar dos veces el mismo nombre o no encontrar un documento por el título que escribió. Normalice el texto a NFC al entrar en el sistema, almacénelo normalizado y compare las formas normalizadas. La política de estabilidad garantiza que una cadena normalizada con una versión sigue normalizada en todas las posteriores, siempre que solo contenga caracteres conocidos por esa versión; por tanto, el código de normalización debe estar tan actualizado como el texto.

Los caracteres parecidos pasan por ser iguales. La o latina, la о cirílica y la ο griega son puntos de código distintos que la mayoría de las tipografías dibujan de forma parecida. Dos nombres de cuenta o códigos de producto pueden parecer idénticos y diferir en un carácter. Unicode los mantiene separados deliberadamente y su estándar de mecanismos de seguridad, UTS #39, explica cómo reducir esta clase de suplantación. Cuando un nombre identifica a una persona o a un destinatario de un pago, permita una sola escritura por nombre o compruébelo con los datos de seguridad del estándar antes de aceptarlo.

Los bytes se leen con la codificación incorrecta. El texto almacenado como UTF-8 y leído como windows-1252 convierte blåbær en blÃ¥bÃ¦r, porque cada letra de dos bytes se lee como dos de un byte. La palabra Unicode no dice cómo se escribieron los bytes; solo lo dice la forma de codificación. Declárela donde el texto cruce un límite: en la cabecera HTTP, en la página, en la conexión a la base de datos y en el formato del archivo. Trate un archivo que llega sin ella como una pregunta para una persona, no como una adivinanza para el código.

Se culpa al texto de la ausencia de un glifo. Si una letra aparece como una caja vacía, la causa habitual es una fuente sin forma para ese punto de código; el texto sigue intacto. Como Unicode deja las formas a las fuentes, cada tipografía de un producto debe cubrir todos los caracteres que necesitan sus idiomas, tanto Є, І, Ї y Ґ como los que van de A a я; de lo contrario, el navegador los dibuja con la fuente alternativa que encuentre. La nota anterior explica que esa es la razón de servir cirílico en este sitio: sin él, una página ucraniana recurriría a lo que ofreciera el sistema. Pruebe cada idioma con todo su alfabeto, en lugar de con una oración de muestra.

Los nombres se comprueban de la A a la Z. Un patrón como `^[A-Za-z' -]+$` acepta O'Neill y rechaza Søren Ågård, Zoë y Ґалаґан: precisamente los nombres a los que debe servir un producto danés o ucraniano. Compruebe las letras mediante sus propiedades Unicode: en JavaScript, `/^[\p{L}\p{M}' -]+$/u` acepta los cuatro. Esas propiedades proceden de la copia de la base de datos de caracteres de Unicode incluida en el entorno de ejecución. Por ejemplo, el entorno Node.js 25.2.1 utilizado aquí indica Unicode 16.0; se trata de una compilación concreta, no de una garantía para todas las versiones 25.x. Las letras añadidas en versiones posteriores de Unicode no superan la comprobación hasta que se actualizan los datos de caracteres del entorno. Valide los nombres con criterios flexibles y deje que una persona, no un patrón, decida los casos difíciles.

Lecturas adicionales

- Acerca del estándar Unicode — la descripción del propio Consorcio: de qué consta una versión y dónde se publica la más reciente.
- El estándar Unicode, capítulo 2: estructura general — los diez principios de diseño, el espacio de códigos, las tres formas de codificación y las reglas de equivalencia de secuencias.
- Políticas de estabilidad de la codificación de caracteres Unicode — lo que nunca puede cambiar tras codificar un carácter, desde su punto de código y su nombre hasta su normalización.

- Unicode 18.0.0 — las novedades de la versión actual y los estándares técnicos que se versionan con ella, entre ellos los de ordenación y seguridad.
- Node.js: process.versions — el informe de versiones de los componentes del entorno de ejecución, utilizado aquí para vincular el ejemplo de datos Unicode a una compilación de Node.js, en vez de a toda una serie de versiones.
- CSS Fonts: rangos de caracteres — cómo los rangos declarados y la cobertura real de glifos de la fuente determinan conjuntamente qué caracteres puede representar una variante.

Glosario

Elija según sus restricciones reales

Describa el flujo de trabajo, las interfaces existentes y los usuarios que necesitan el cambio. Podemos definir una entrega comprobable y un traspaso que permita a su equipo mantener lo entregado.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es Unicode?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

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

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

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

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/unicode/