Engineer

Unicode

⌨ 一つ一つの文字に固有の番号を割り当てる標準

どのようなものか

Unicodeは、世界の文字体系にある文字、数字、句読点、記号の一つ一つに、固有の番号を与える標準です。保守する側は、これを普遍的な文字エンコーディングと呼びます。現代の言語や技術分野のテキストを、どこでも交換、処理、表示できるようにし、古い文字体系や歴史上の表記も表せる設計です。一つの文字集合に、デンマーク語のå、ウクライナ語のґ、ユーロ記号、何世紀も使われていない文字が入るため、一つの文書や一つのデータベース列に、すべてを含められます。

厳密には、Unicodeの版は、互いに対応する複数の部分から成ります。中核の仕様、文字表、附属書、そして各文字の属性を記録する機械可読ファイル群である文字データベースです。Unicode技術委員会が継続的に保守し、Unicodeコンソーシアムが公開します。18.0は2026年9月16日に発表されました。

文字に割り当てる番号を、コードポイントと呼びます。コード空間は十六進の0から10FFFFまで、全部で1,114,112コードポイントあります。表記はU+に少なくとも四桁の十六進数を続けます。åはU+00E5、ґはU+0490、ユーロ記号はU+20ACです。コンソーシアムの安定性方針は、一度符号化した文字の位置を永久に維持すると約束しています。移動も削除もせず、名前も変えません。そのため、今日符号化したテキストは、後のどの版でも同じ意味を保ちます。

標準を理解する鍵になる区別は二つあります。一つ目は、文字とグリフです。Unicodeは抽象的な文字を符号化し、形はフォントに任せます。中核仕様は、グリフの形はフォント提供元の責任であり、標準には含まれないと明記しています。同じコードポイントでも書体によって違って見え、対応する形を持たない書体では描けません。二つ目は、コードポイントと、それを保存するバイトの区別です。Unicodeの三つの符号化形式UTF-8、UTF-16、UTF-32は、各コードポイン

トを8、16、32ビットの一つ以上の単位で書きます。仕様は「UTF-8かUnicodeか」と区別しようとすることを落とし穴と呼びます。どれも、同じ文字を実装する等しく有効な方法だからです。

概念を判断に生かす

「Unicodeとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 日付、通貨、名前の並び順まで含め、販売先のそれぞれの市場で自然に読めるソフトウェアにします。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 英語・デンマーク語のi18nシステムを構築 — 12,027 英語・デンマーク語の言語ごとのメッセージ

英語・デンマーク語のi18nシステムを構築 お問い合わせ →

なぜ重要なのか

Unicode以前は、文字体系ごとにコードページがあり、どのページかが分からなければ、一つのバイトにも意味が定まりませんでした。西欧のwindows-1252では、バイトE5はåです。キリル文字のwindows-1251では、同じバイトがeです。一方のページには、もう一方の文字を入れられません。windows-1251にはЃがありますがå、æ、øはなく、windows-1252はその逆です。デンマーク語とウクライナ語の製品には、二つの符号化方式と、どのテキストがどちらかを示す確かな記録が必要でした。両言語を引用する一文は、どちらの方式にも収まりませんでした。

Unicodeは、その管理を不要にしました。製品が公開するどの言語も、顧客がフォームに入力するどの名前も、一つの文字集合で表せます。同じ番号は、読むシステムが違っても同じ文字を意味します。そのため、データベース、API、ブラウザーは、先にコードページを取り決めずにテキストを受け渡せます。

各プログラムが独自に決めるはずだったことも、標準が定めています。言語をまたいで文字を統一します。コンマは、英語のように千の区切りを示しても、フランス語のように小数を区切っても、同じ文字です。意味を決めるのはコードポイントではなくロケールです。一方、文字体系をまたいでは統一しません。ラテン文字のo、キリル文字のo、ギリシャ文字のoは、多くの書体で同じに見えても三つの文字です。三つの文字体系に属し、以前の符号化方式でも区別されていたからです。また、どの並びが等価かも記録します。Åは、合成済みの一文字でも、Aに上付きの丸を表す結合文字を続けた並びでも保存できます。標準は両者を正準等価と定め、同じように表示・解釈されるものとし、いずれも共通の一つの表現へ変える正規化形式を提供します。

この汎用性には代償があり、プログラマーが払います。文字はバイトではありません。また、文字列上の一単位や、読者が一文字と数える単位と常に一致するわけでもありません。この三つを同じだと扱うコードは、英語のデータではテストを通り、そのコードを使う人の名前、住所、メッセージで失敗します。

実際の活用例

JavaScriptでは、層の違いが見えます。文字列はUTF-16コード単位の列で、配列へ展開するとコードポイントに分かれます。

```
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);
}
```

Node.js 25で動かすと、各文字、そのコードポイント、UTF-16単位での長さ、コードポイント数が出ます。

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

最後の行は古イタリア文字です。コードポイントがU+FFFFを超えるため、UTF-16ではサロゲートペアという二つの単位で保存し、`length`は一文字に対して2を返します。

等価性も同じように確認できます。次の三つの文字列は、いずれもÅと表示されます。

```
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
```

一つ目は、合成済み文字U+00C5です。二つ目は、通常のAに、上付きの丸を表す結合文字U+030Aを続けたものです。三つ目はオングストローム記号U+212Bで、Unicode文字データベースではU+00C5に対応付けられています。そのまま比較すると別々の三つの文字列ですが、NFCで正規化すると同じ一つになります。画面上では見分けられないので、元の文字列を比較する検索、一意インデックス、ログイン処理には、三つの別名として映ります。

このサイトは、もっと目立たない場所でもコードポイントに依存しています。すべての文字を横切った下線についてのノートは、本文用Ubuntuのラテン文字とキリル文字のフォントを、ラテン三ウェイト、キリル三ウェイトの計六つの書体ファイルとして説明しています。ウクライナ語のページも英語と同じ書体で組むためです。各ファイルは、対応する文字の範囲を宣言します。その範囲が、ページの文字に対するフォント選択を導きます。明示的なプリロードは、通常のフォント検出より先にファイルを要求することもあります。キリル文字の宣言は、範囲をコードポイントで示します。

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

最初の範囲はÈからμまでで、基本キリル文字のほか、ウクライナ語のЄ、І、Їも含みます。二つ目はґとrだけで、キリル文字ブロック内の、さらに後ろの位置にあります。範囲は、その文字にどのフェイスを選択できるかを宣言しますが、フォント自体にもグリフが必要です。一つの文字体系のファイルしか取得しないという保証ではありません。英語とデンマーク語の本文はラテン文字のフェイスを使いますが、ウクライナ語の言語ラベルにはキリル文字が必要になることがあります。さらにサイトは、非キリル文字のページでも、画面表示用に細いウェイトのキリル文字本文フォントを明示的にプリロードします。

問題が起きるところ

長さをコード単位で数える。 JavaScriptの `length` はUTF-16単位を数えるので、𐐷は二つになり、Aに上付きの丸を表す結合文字を付けたものも、人には一文字に見えますが二単位と数えられます。単位数の制限では、収まるはずのテキストを拒否し、固定位置での切断は文字を分断することがあります。 `"A\u030A".slice(0, 1)` はAを残して上付きの丸を捨てます。𐐷を半分で切ると、サロゲートが片方だ

け残り、UTF-8エンコーダーは置換文字U+FFFDに変えます。各上限が何を数えるかを決めます。保存容量の制限ならバイト、人が決める制限なら読者が見た文字です。`Intl.Segmenter`は後者を数え、上付きの丸の結合文字を使ったÅrhusでは、`length`が6を返すところで5を返します。テキストは、それが示す境界でだけ切ります。

同じテキストを、単位ごとに比較する。 上の三種類のÅは、`===`、ハッシュ、一意インデックスには異なる三つの文字列です。顧客が同じ名前を二度登録できたり、入力した題名で文書が見つからなかったりします。システムの入口でNFCに正規化し、正規化した状態で保存し、正規化後の形を比較します。安定性方針は、ある版で正規化した文字列が、後の版でも正規化済みであり続けると保証します。ただし、その文字列には、元の版が知る文字だけを含む必要があります。そのため、正規化コードも扱うテキストに見合う新しさが必要です。

似た文字を同じものとして通す。 ラテン文字のo、キリル文字のo、ギリシャ文字のoは別のコードポイントですが、多くの書体は似た形で描きます。アカウント名や製品コードが、見た目は同じでも一文字だけ違うことがあります。Unicodeは意図して区別しており、セキュリティー機構の標準UTS #39が、この種のなりすましを減らす方法を示しています。人や受取人を識別する名前では、一つの名前に一つの文字体系だけを認めるか、受け付ける前に標準のセキュリティーデータと照合します。

バイトを別の符号化方式で読む。 UTF-8で保存したテキストをwindows-1252で読むと、blåbærはblÃ¥bÃ¦rになります。二バイトの文字を、それぞれ一バイトの二文字として読むためです。Unicodeという言葉だけでは、バイトをどう書いたかは分かりません。それを示すのは符号化形式です。テキストが境界を通る場所では、HTTPヘッダー、ページ、データベース接続、ファイル形式に明記します。符号化方式が示されていないファイルは、コードに推測させず、人に確認する問題として扱います。

グリフがないのに、テキストを疑う。 文字が空の四角で表示される場合、通常はそのコードポイントの形をフォントが持たないのであり、テキスト自体は壊れていません。形をフォントに任せるUnicodeでは、製品が使う各書体が、言語に必要な文字をすべて含む必要があります。Aからяだけでなく、Є、І、Ї、Ґも必要で、なければブラウザーは見つけた代替フォントで描きます。上のノートは、キリル文字を配信する理由をここに求めています。なければウクライナ語のページは、システムが提供する書体に頼ります。例文だけでなく、言語の全アルファベットでテストします。

名前をAからZだけで検査する。 `^[A-Za-z' -]+$` のようなパターンはO'Neillを受け付け、Søren Ågård、Zoë、Ґалаґанを拒否します。デンマーク語やウクライナ語の製品が対象にしている、まさにその人たちの名前です。代わりにUnicode属性で文字を調べます。JavaScriptの `/^[\p{L}\p{M}' -]+$/u` なら四つとも通ります。この属性は、実行環境に含まれるUnicode文字データベースのコピーに由来します。たとえば、この例に使ったNode.js 25.2.1はUnicode 16.0と報告します。一つのビルドについての値であり、25.x全体の保証ではありません。後のUnicodeで加わった文字は、実行環境の文字データが更新されるまで、この検査に通りません。名前の検証は緩やかにし、難しい例はパターンではなく人が判断できるようにします。

参考資料

- Unicode標準について — 標準の内容、一つの版を構成するもの、最新版の公開場所について、コンソーシアム自身が説明しています。
- Unicode標準、第2章：全体構造 — 十の設計原則、コード空間、三つの符号化形式、等価な並びの規則を扱います。
- Unicode文字エンコーディングの安定性方針 — 文字の符号化後に変えてはならないものを、コードポイント、名前、正規化まで説明しています。
- Unicode 18.0.0 — 現行版の追加内容と、照合やセキュリティーなど、同時に版管理される技術標準です。
- Node.js：process.versions — 実行環境の各構成要素の版を示す報告です。ここではUnicodeデータの例を、リリース系列全体ではなく、一つのNode.jsビルドに固定するために使います。
- CSS Fonts：文字の範囲 — 宣言した文字範囲と、フォントが実際に持つグリフの両方が、そのフェイスで描ける文字を決める仕組みを説明しています。

用語集

実際の制約に合わせて選ぶ

業務の流れ、既存のインターフェース、変更を必要とする利用者をお知らせください。検証可能な納品内容と、お客様のチームが保守を続けられる引き継ぎ内容を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「Unicodeとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/unicode/