

# 13.1 ASCII and UTF

Peter Rutschmann

05.11.2025

## ASCII und UTF (Text <-> binär)

**ASCII** (American Standard Code for Information Interchange) ist eine ältere Zeichencodierung, die 7 Bit nutzt und 128 Zeichen definiert (vorwiegend englische Buchstaben, Ziffern und Steuerzeichen).

**Erweitertes ASCII (8 Bit):** Später wurde ASCII auf 8 Bit erweitert (256 Zeichen), um zusätzliche Zeichen wie Umlaute, Währungssymbole oder grafische Elemente darzustellen. Es entstanden verschiedene Codepages (z. B. ISO-8859-1 fuer westeuropäische Sprachen), die jedoch nicht untereinander kompatibel waren.

**UTF-8** ist eine moderne, variable-length Codierung fuer Unicode: sie kann alle weltweit verwendeten Zeichen darstellen und ist abwärtskompatibel zu ASCII (die ersten 128 Zeichen sind identisch). UTF-8 verwendet 1 bis 4 Bytes pro Zeichen; einfache ASCII-Zeichen bleiben dabei ein Byte (z. B. 'A' = U+0041 = 0x41). Beim Arbeiten mit Textdateien, Notebooks und beim Rendern von Webseiten oder PDFs sollte immer UTF-8 verwendet werden.

[Liste der UTF8 Codierung](#)

Es gibt mehrere UTF-Kodierungen (UTF-8, UTF-16, UTF-32), weil sie verschiedene Kompromisse zwischen Speicherplatz, Kompatibilität und Geschwindigkeit eingehen. Alle drei gehören zur Unicode-Familie – sie kodieren dieselben Zeichen, aber auf unterschiedliche Weise.

Kodierung	Wie sie funktioniert	Hauptvorteil	Nachteil	Typischer Einsatz
<b>UTF-8</b>	Variable Länge (1–4 Byte pro Zeichen)	Kompatibel mit ASCII, sehr platzsparend für westliche Texte	Asiatische oder seltene Zeichen brauchen mehr Platz	Web, Linux, Internet (Standard in HTML, JSON etc.)

Kodierung	Wie sie funktioniert	Hauptvorteil	Nachteil	Typischer Einsatz
<b>UTF-16</b>	Meist 2 Byte pro Zeichen, Sonderzeichen 4 Byte	Kompakter für viele nicht-lateinische Schriften	Nicht kompatibel mit ASCII, Byte-Order-Fragen (Little/Big Endian)	Windows, Java, .NET, XML (teilweise)
<b>UTF-32</b>	Immer 4 Byte pro Zeichen	Sehr einfach zu verarbeiten (jedes Zeichen = 1 Codepunkt)	Hoher Speicherbedarf (vierfach gegenüber UTF-8)	Spezialfälle, interne Verarbeitung

### Beispiel

Text: A□

Kodierung	Bytes (hexadezimal)
<b>UTF-8</b>	41 F0 9F 98 8A (1 Byte für „A“, 4 Byte für □)
<b>UTF-16</b>	00 41 D8 3D DE 0A (2 Byte für „A“, 4 Byte für □)
<b>UTF-32</b>	00 00 00 41 00 01 F6 0A (je 4 Byte pro Zeichen)

### Zusammenfassung

- Alle UTF-Formate **repräsentieren dieselben Unicode-Zeichen**.
- Sie unterscheiden sich **nur in der Art, wie sie Bytes speichern**.
- UTF-8 hat sich weltweit durchgesetzt, weil es **am besten mit älterer ASCII-Software funktioniert**.