



What is UTF-8 Encoding? A Guide for Non-Programmers



Jamie Juviler

Updated: November 02, 2021

Published: August 10, 2020



Text: its importance on the internet goes without saying. It's the first "T" in "HTTP", the only "T" in "HTML", and virtually every website uses it somehow, be it a URL, a piece of marketing copy, a product review, a viral Tweet, or a blog post. (Hi there!)



Free Workbook: How to Plan a Successful Website Redesign

But, web text might not actually be as simple as you think. Consider the thousands of languages spoken today, or all the punctuation and symbols we can add to enhance them, or the fact that new emojis are being created to capture every human emotion. How do websites store and process all of this?

The truth is, even something as basic as text requires a well-coordinated, clearly-defined system to appear in web browsers. In this post, I'll explain the basics of one technology central to text on the web, **UTF-8**. We'll learn the basics of text storage and encoding, and discuss how it helps put engaging words across your site.

Before we begin, you should be familiar with the **basics of HTML** and ready to dive into some light computer science.

What Is UTF-8?

UTF-8 stands for "Unicode Transformation Format - 8 bits." That's not helpful to us yet, so let's rewind to the basics.

Binary: How Computers Store Information

In order to store information, computers use a binary system. In binary, all data is represented in sequences of 1s and 0s. The most basic unit of binary is a *bit*, which is just a single 1 or 0. The next largest unit of binary, a byte, consists of 8 bits. An example of a byte is "01101011".

FREE WEBSITE REDESIGN WORKBOOK

Learn how to redesign your website with this free guide.

DOWNLOAD THE FREE WORKBOOK



The American Standard Code for Information Interchange (ASCII) was an early standardized encoding system for text. Encoding is the process of converting characters in human languages into binary sequences that computers can process.

ASCII’s library includes every upper-case and lower-case letter in the Latin alphabet (A, B, C...), every digit from 0 to 9, and some common symbols (like /, !, and ?). It assigns each of these characters a unique three-digit code and a unique byte.

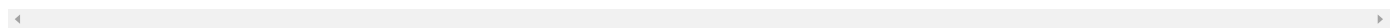
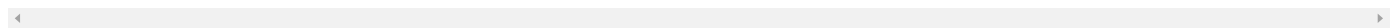
The table below shows examples of ASCII characters with their associated codes and bytes.

CHARACTER	ASCII CODE	BYTE
A	065	01000001
a	097	01100001
B	066	01000010
b	098	01100010
Z	090	01011010
z	122	01111010
0	048	00110000
9	057	00111001
!	033	00100001
?	063	00111111

Learn how to redesign your website with this free guide.



```
01010100 01101000 01100101 00100000 01110001  
  
01110101 01101001 01100011 01101011 00100000  
  
01100010 01110010 01101111 01110111 01101110  
  
00100000 01100110 01101111 01111000 00100000  
  
01101010 01110101 01101101 01110000 01110011  
  
00100000 01101111 01110110 01100101 01110010  
  
00100000 01110100 01101000 01100101 00100000  
  
01101100 01100001 01111010 01111001 00100000  
  
01100100 01101111 01100111 00101110
```



That doesn't mean much to us humans, but it's a computer's bread and butter.

The number of characters that ASCII can represent is limited to the number of unique bytes available, since each character gets one byte. If you do the math, you'll find that there are 256 different ways of groups eight 1s and 0s together. This gives us 256 different bytes, or 256 ways to represent a character in ASCII. When ASCII was introduced in 1960, this was okay, since developers needed only 128 bytes to represent all the English characters and symbols they needed.

But, as computing expanded globally, computer systems began to store text in languages besides English, many of which used non-ASCII characters. New systems were created to map other languages to the same set of 256 unique bytes, but having multiple encoding systems was inefficient and confusing. Developers needed a better way to encode all possible characters with one system.

Unicode: A Way to Store Every Symbol, Ever

Enter Unicode, an encoding system that solves the space issue of ASCII. Like ASCII, Unicode assigns a unique code, called a *code point*, to each character. However, Unicode's more sophisticated system can produce over a million code points, more than enough to account for every character in any language.

[Learn how to redesign your website with this free guide.](#)



CHARACTER	CODE POINT
A	U+0041
a	U+0061
0	U+0030
9	U+0039
!	U+0021
Ø	U+00D8
€	U+0683
๓	U+0C9A
制	U+2070E
😬	U+1F601

If you want to learn how code points are generated and what they mean in Unicode, check out this [in-depth explanation](#).

So, we now have a standardized way of representing every character used by every human language in a single library. This solves the issue of multiple labeling systems for different languages — any computer on Earth can use Unicode.

But, Unicode alone doesn't store words in binary. Computers need a way to translate Unicode into binary so that its characters can be stored in text files. Here's where UTF-8 comes in.

UTF-8: The Final Piece of the Puzzle

UTF-8 is an encoding system for Unicode. It can translate any Unicode character to a matching unique binary string, and can also translate the binary string back to a Unicode character. This is the meaning of “UTF”, or “Unicode Transformation Format.”

There are other encoding systems for Unicode besides UTF-8, but UTF-8 is unique because it represents characters in one-byte units. Remember that one byte consists of eight bits, hence the “-8” in its name.

More specifically, UTF-8 converts a code point (which represents a single character in Unicode) into a set of one to four bytes. The first 128 characters in the Unicode library — the characters we saw in ASCII — are represented as one byte. Characters that appear later in the Unicode library are encoded as two-byte, three-byte, and eventually four-byte binary units.

Below is the same character table from above, with the UTF-8 output for each character added. Notice how some characters are represented as just one byte, while others use more.



A	U+0041	01000001
a	U+0061	01100001
0	U+0030	00110000
9	U+0039	00111001
!	U+0021	00100001
Ø	U+00D8	11000011 10011000
€	U+0683	11011010 10000011
ভ	U+0C9A	11100000 10110010 10011010
判	U+2070E	11110000 10100000 10011100 10001110
😄	U+1F601	11110000 10011111 10011000 10000001

Learn how to redesign your website with this free guide.



common characters. These less-common characters are encoded into two or more bytes, but this is okay if they're stored sparingly.

Spatial efficiency is a key advantage of UTF-8 encoding. If instead every Unicode character was represented by four bytes, a text file written in English would be four times the size of the same file encoded with UTF-8.

Another benefit of UTF-8 encoding is its backward compatibility with ASCII. The first 128 characters in the Unicode library match those in the ASCII library, and UTF-8 translates these 128 Unicode characters into the same binary strings as ASCII. As a result, UTF-8 can take a text file formatted by ASCII and convert it to human-readable text without issue.

UTF-8 Characters in Web Development

UTF-8 is the most common character encoding method used on the internet today, and is the default character set for HTML5. **Over 95% of all websites**, likely including your own, store characters this way. Additionally, common data transfer methods over the web, like **XML** and **JSON**, are encoded with UTF-8 standards.

Since it's now the standard method for encoding text on the web, all your site pages and databases should use UTF-8. A content management system or website builder will save your files in UTF-8 format by default, but it's still a good idea to make sure you're sticking to this best practice.

Text files encoded with UTF-8 must indicate this to the software processing it. Otherwise, the software won't properly translate the binary back into characters. In HTML files, you might see a string of code like the following near the top:

```
<meta charset="UTF-8">
```

This tells the browser that the HTML file is encoded by UTF-8, so that the browser can translate it back to legible text.

UTF-8 vs. UTF-16

As I mentioned, UTF-8 is not the only encoding method for Unicode characters — there's also UTF-16. These methods differ in the number of bytes they need to store a character. UTF-8 encodes a character into a binary string of one, two, three, or four bytes. UTF-16 encodes a Unicode character into a string of either two or four bytes.

This distinction is evident from their names. In UTF-8, the smallest binary representation of a character is one byte, or eight bits. In UTF-16, the smallest binary representation of a character is two bytes, or sixteen bits.

Both UTF-8 and UTF-16 can translate Unicode characters into computer-friendly binary and back again. However, they are not compatible with each other. These systems use different algorithms to map code points to binary strings, so the binary output for any given character will look different from both methods:

CHARACTER	UTF-8 BINARY ENCODING	UTF-16 BINARY ENCODING
A	01000001	01000001 11011000 00001110 11011111
判	11110000 10100000 10011100 10001110	01000001 11011000 00001110 11011111

Learn how to redesign your website with this free guide.



bytes. Still, if your pages are filled with ABCs and 123s, stick with UTF-8.

Decoding the World of UTF-8 Encoding

That was a lot of words about words, so let's summarize what we've covered:

1. Computers store data, including text characters, as binary (1s and 0s).
2. ASCII was an early way of encoding, or mapping characters to binary code so that computers could store them. However, ASCII did not provide enough room for non-Latin characters and numbers to be represented in binary.
3. Unicode was the solution to this problem. Unicode assigns a unique "code point" to every character in every human language.
4. UTF-8 is a Unicode character encoding method. This means that UTF-8 takes the code point for a given Unicode character and translates it into a string of binary. It also does the reverse, reading in binary digits and converting them back to characters.
5. UTF-8 is currently the most popular encoding method on the internet because it can efficiently store text containing any character.
6. UTF-16 is another encoding method, but is less efficient for storing text files (except for those written in certain non-English languages).

Unicode translation isn't something most of us need to think about when browsing or **designing websites**, and that's exactly the point — to create a seamless text processing system that functions for all languages and web browsers. If it's working well, you won't notice it.

But, if you find your website's pages are using up an inordinate amount of space, or if your text is littered with □s and ♦s, it's time to put your new knowledge of UTF-8 to work.

FREE RESOURCE

The Ultimate Workbook for Redesigning Your Website


[Download Now](#)




Topics: **Website Design**

Related Articles


Learn how to redesign your website with this free guide.






The Best Resume Website Templates: 27 of Our Favorites

Mar 13, 2023




7 Ways to Optimize Your Homepage for Lead Generation (and How to Keep Interest)

Mar 13, 2023



20 Simple Website Templates You Can Try Today

Mar 07, 2023



Creative Webs of our Favorite

Mar 06, 2023

<

Popular Features

>

Free Tools

>

Company

>

Customers

>

Partners

>





