

# Unicode: not only a Charset

zhuyie

[zhuyie@gmail.com](mailto:zhuyie@gmail.com)

#### **Version 12.0.0**

The Unicode Consortium. The Unicode Standard, Version 12.0.0, (Mountain View, CA: The Unicode Consortium, 2019) ISBN 978-1-936213-22-1  
<http://www.unicode.org/versions/Unicode12.0.0/>

#### **Version 11.0.0**

The Unicode Consortium. The Unicode Standard, Version 11.0.0, (Mountain View, CA: The Unicode Consortium, 2018) ISBN 978-1-936213-19-1  
<http://www.unicode.org/versions/Unicode11.0.0/>

#### **Version 10.0.0**

The Unicode Consortium. The Unicode Standard, Version 10.0.0, (Mountain View, CA: The Unicode Consortium, 2017) ISBN 978-1-936213-16-0  
<http://www.unicode.org/versions/Unicode10.0.0/>

#### **Version 9.0.0**

The Unicode Consortium. The Unicode Standard, Version 9.0.0, (Mountain View, CA: The Unicode Consortium, 2016) ISBN 978-1-936213-13-9  
<http://www.unicode.org/versions/Unicode9.0.0/>

## **FAQ on FAQs**

Describes how and when new FAQs are created, how FAQs relate to specifications, and and what to do if you think there is an error in a Unicode specification.



# Agenda

- Legacy Charsets
- A Brief History of Unicode
- The Unicode Code Space & Encodings
- Combining Character Sequence & Normalization
- CJK Unified Ideographs
- BIDI

# Terminology

- A **character** is a minimal unit of text that has semantic value.
- A **character set** is a collection of characters that might be used by multiple languages. Example: The Latin character set is used by English and most European languages.
- A **coded character set** is a character set in which each character corresponds to a unique number.
- A **code point** of a coded character set is any allowed value in the character set or code space.
- A **code space** is a range of integers whose values are code points.

# Terminology (cont.)

- A **code unit** is the unit of storage of a part of an encoded code point. In UTF-8 this means 8-bits, in UTF-16 this means 16-bits. A single code unit may represent a full code point, or part of a code point.
- **character encoding form**: Mapping from a character set definition to the actual code units used to represent the data.
- **character encoding scheme**: A character encoding form plus byte serialization. There are seven character encoding schemes in Unicode: UTF-8, UTF-16, UTF-16BE, UTF-16LE, UTF-32, UTF-32BE, and UTF-32LE.

# Terminology (cont. 2)

- A **language** is a structured system of communication.
- A **script** is a collection of letters and other written signs used to represent textual information in one or more writing systems. For example, Russian is written with a subset of the Cyrillic script; Ukrainian is written with a different subset. The Japanese writing system uses several scripts.
- A **writing system** is a set of rules for using one or more scripts to write a particular language. Examples include the American English writing system, the British English writing system, the French writing system, and the Japanese writing system.

# Terminology (cont. 3)

- A **grapheme/grapheme cluster** is a sequence of one or more code points that are displayed as a single, graphical unit that a reader recognizes as a single element of the writing system. For example, both **a** and **ä** are graphemes, but they may consist of multiple code points.
- A **glyph** is an image, usually stored in a font (which is a collection of glyphs), used to represent graphemes or parts thereof.

# ASCII

- **A**merican **S**tandard **C**ode for **I**nformation **I**nterchange
- 初版发布于1963年，已经被标准化为[ISO/IEC 646](#)。

## ASCII

|    | 0 | 1 | 2 | 3 | 4  | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|----|---|---|---|---|----|---|---|---|---|---|---|---|---|---|---|---|
| 0x |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
| 1x |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
| 2x |   | ! | " | # | \$ | % | & | ' | ( | ) | * | + | , | - | . | / |
| 3x | 0 | 1 | 2 | 3 | 4  | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 4x | @ | A | B | C | D  | E | F | G | H | I | J | K | L | M | N | O |
| 5x | P | Q | R | S | T  | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 6x | ` | a | b | c | d  | e | f | g | h | i | j | k | l | m | n | o |
| 7x | p | q | r | s | t  | u | v | w | x | y | z | { |   | } | ~ |   |
| 8x |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
| 9x |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
| Ax |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
| Bx |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
| Cx |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
| Dx |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
| Ex |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
| Fx |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |



# ISO 8859-1

- [ISO 2022](#)在维持对ASCII兼容的基础上，设计出支持多字节字符集(MBCS)的架构。GB2312编码也符合此框架。
- [ISO 8859](#)兼容于ASCII，是基于ISO 2022的架构在G1区定义出16套扩展字符而形成的字符集。其中ISO 8859-1用于支持西欧语言。

## ISO 8859-1

|    | 0 | 1 | 2 | 3 | 4  | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|----|---|---|---|---|----|---|---|---|---|---|---|---|---|---|---|---|
| 0x |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
| 1x |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
| 2x |   | ! | " | # | \$ | % | & | ' | ( | ) | * | + | , | - | . | / |
| 3x | 0 | 1 | 2 | 3 | 4  | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 4x | @ | A | B | C | D  | E | F | G | H | I | J | K | L | M | N | O |
| 5x | P | Q | R | S | T  | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 6x | ` | a | b | c | d  | e | f | g | h | i | j | k | l | m | n | o |
| 7x | p | q | r | s | t  | u | v | w | x | y | z | { |   | } | ~ |   |
| 8x |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
| 9x |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
| Ax |   | ì | í | î | ï  | ñ | ò | ó | ô | õ | ö | ÷ | ø | ù | ú | û |
| Bx | ° | ± | ² | ³ | ´  | µ | ¶ | · | ¸ | ¹ | º | » | ¼ | ½ | ¾ | ¿ |
| Cx | À | Á | Â | Ã | Ä  | Å | Æ | Ç | È | É | Ê | Ë | Ì | Í | Î | Ï |
| Dx | Ð | Ñ | Ò | Ó | Ô  | Õ | Ö | × | Ø | Ù | Ú | Û | Ü | Ý | Þ | ß |
| Ex | à | á | â | ã | ä  | å | æ | ç | è | é | ê | ë | ì | í | î | ï |
| Fx | ð | ñ | ò | ó | ô  | õ | ö | ÷ | ø | ù | ú | û | ü | ý | þ | ÿ |

# Windows CP1252

- 操作系统厂商例如Microsoft/IBM都会设计自己的字符集。Microsoft在ISO 8859-1的基础上又增加了27个符号，定义出[Code Page 1252](#)。

| Windows-1252 |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
|--------------|---|---|---|---|----|---|---|---|---|---|---|---|---|---|---|---|
|              | 0 | 1 | 2 | 3 | 4  | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0x           |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
| 1x           |   |   |   |   |    |   |   |   |   |   |   |   |   |   |   |   |
| 2x           |   | ! | " | # | \$ | % | & | ' | ( | ) | * | + | , | - | . | / |
| 3x           | 0 | 1 | 2 | 3 | 4  | 5 | 6 | 7 | 8 | 9 | : | ; | < | = | > | ? |
| 4x           | @ | A | B | C | D  | E | F | G | H | I | J | K | L | M | N | O |
| 5x           | P | Q | R | S | T  | U | V | W | X | Y | Z | [ | \ | ] | ^ | _ |
| 6x           | ` | a | b | c | d  | e | f | g | h | i | j | k | l | m | n | o |
| 7x           | p | q | r | s | t  | u | v | w | x | y | z | { |   | } | ~ |   |
| 8x           | € | ‚ | ƒ | „ | •  | † | ‡ | ~ | ™ | § | < | œ | Ž | ž | ÿ |   |
| 9x           |   | ‚ | ƒ | „ | •  | † | ‡ | ~ | ™ | § | > | œ | Ž | ž | ÿ |   |
| Ax           | ° | ± | ² | ³ | ´  | µ | ¶ | · | ¸ | ¹ | º | » | ¼ | ½ | ¾ | ¿ |
| Bx           | À | Á | Â | Ã | Ä  | Å | Æ | Ç | È | É | Ê | Ë | Ì | Í | Î | Ï |
| Cx           | Ð | Ñ | Ò | Ó | Ô  | Õ | Ö | × | Ø | Ù | Ú | Û | Ü | Ý | Þ |   |
| Dx           | ð | ñ | ò | ó | ô  | õ | ö | ÷ | ø | ù | ú | û | ü | ý | þ |   |
| Ex           | à | á | â | ã | ä  | å | æ | ç | è | é | ê | ë | ì | í | î | ï |
| Fx           | ð | ñ | ò | ó | ô  | õ | ö | ÷ | ø | ù | ú | û | ü | ý | þ |   |

# GB2312/GBK/GB18030

- [GB2312](#)由中国国家标准总局于1980年发布，共收录6763个汉字。
- GB2312对所收录汉字进行了“分区”处理，每区含有94个汉字 / 符号，共计94个区，因此也称为“区位码”。
- GB2312字符串通常使用符合ISO 2022架构的[EUC-CN](#)方式存储。每个汉字采用2个字节表示，将码点的“区”值加上160得到高字节，“位”值加上160得到低字节。
- GBK和GB18030是对GB2312字符集的扩展标准。
- [Windows CP936](#)能支持绝大多数的GBK字符。

# Charset detection

- 由于存在着数十上百种字符集，也不存在统一的标识机制，很多时候需要使用字符集检测技术。
- 并不是一个非常简单的流程。Mozilla有一篇这个领域里面比较有名的[论文](#)，组合使用了三类方法：
  - Coding scheme method
  - Character Distribution
  - 2-Char Sequence Distribution
- 参考[ucharset](#)。

# What is Unicode?

[Unicode](#) is an information technology (IT) **standard for the consistent encoding, representation, and handling of text expressed** in most of the world's writing systems. The standard is maintained by the **Unicode Consortium**, and as of March 2020, there is a total of 143,859 characters, with Unicode 13.0 covering 154 modern and historic scripts, as well as multiple symbol sets and emoji. The character repertoire of the Unicode Standard is **synchronized with ISO/IEC 10646**, and both are code-for-code identical.

# A Brief History of Unicode

- 1984年，一个工作组开始准备ISO/IEC 10646，试图解决传统字符集的各类问题。这个工作组的正式名称是：ISO/IEC JTC1/SC2/WG2 (that's "ISO/IEC Joint Technical Committee #1 [Information Technology], Subcommittee #2 [Coded Character Sets], Working Group #2 [Multioctet codes]"), or just "**WG2**" for short.
- 1988年，另一个由Xerox, Apple等公司的技术人员组成的小组也开始做类似的事情，他们的工作基于Xerox早期的XCCS编码标准。其中来自Xerox的Joe Becker的论文中首次提到了"**Unicode**"这个词。这个小组也就是今天"Unicode Consortium"的前身。

# A Brief History of Unicode (cont.)

- 尽管有着类似的目标，两个团队的技术方案却有着较大的差异。
- ISO 10646的初始版本采用4字节的code point，但对每个字节的部分取值范围做了禁用，其实际code space为：192(groups) x 192(planes) x 192(rows) x 192(cells)。
- 由于采用了4字节的code point，为了节省存储空间而引入了多种复杂的编码方式。
- 引入了"**B**asic **M**ultilingual **P**lane"的概念。
- 将简中、繁中、日文、韩文分配到不同的plane。

# A Brief History of Unicode (cont. 2)

- 另一方面，初始版本的Unicode基于2字节的code point设计，大致等价于ISO 10646中的1个plane。
- 对字节的取值范围没有额外限制，因此最大可编码65536个字符。
- 也没有定义其它编码方式，就是每个字符占用2字节。UTF-8之类的编码方式是后面才引入的。
- 对于汉字，Unicode尝试建立一个CJKV中的公共汉字字符子集，并进行统一编码。其思路类似于只分配一个'A'的code point，虽然它同时存在于English, Spanish, French, Italian, German...



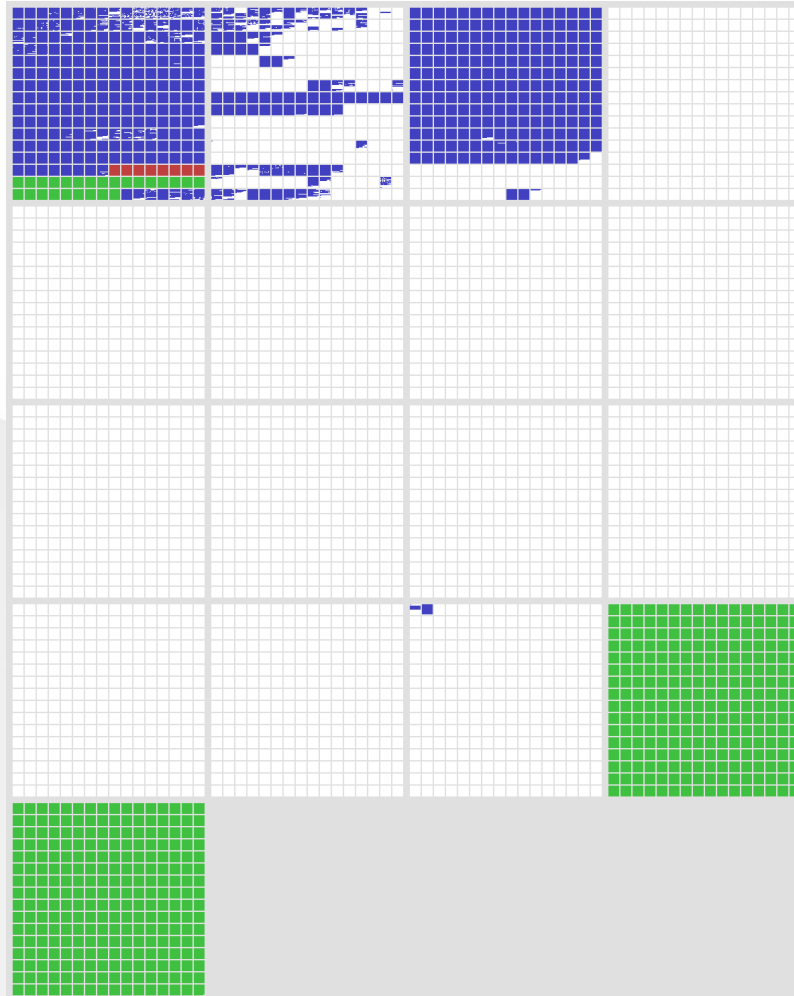
# A Brief History of Unicode (cont. 3)

- ISO 10646的初始投票没有获得通过~~~
- 两个团队开始讨论技术融合方案：
  - 保留了ISO 10646的32-bit code space，但去掉了单个字节的取值范围限制。
  - 编码方式进行了简化，只保留UCS-4和UCS-2(仅支持BMP)。
  - 采用了Unicode的统一汉字编码方案，并将其放入到BMP中。
- 从1991年开始，两个团队开始做技术方案和码点的统一，使得"Universal Character Set"和"The Unicode Standard"这两个标准在常规使用层面上是等价的，并在后继的演进中维持了同步。

# The Unicode Codespace

- 当前最新的Unicode版本是13.0 (March 2020)。
- 共定义了17个平面(planes), 每个平面包含 $256 \times 256$ 个码点(code points), 理论上最大支持的码点(code points)个数为:  $17 \times 65536 = 1,114,112$ 。
- 当前一共给143,859个字符(characters)分配了码点, 分属于154种文字(scripts)。

# The Unicode Codespace (cont.)



White: unassigned space.  
Blue: assigned code points.  
Green: private-use areas.  
Red: surrogates.

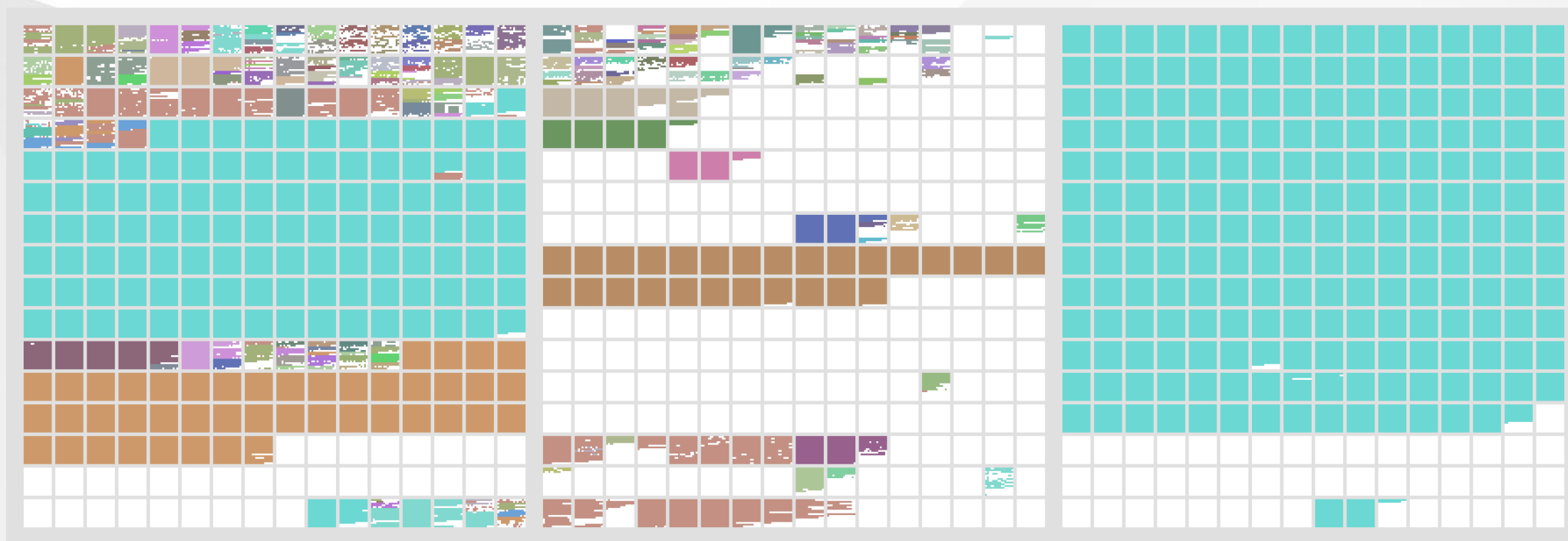
# The Unicode Codespace (cont. 2)

Plane0: basic multilingual plane.

Plane1: historical scripts, emojis.

Plane2: less-common and historical Han characters.

■ : Han  
■ : Korean



# UTF-32

- "UTF" stands for "Unicode Transformation Format".
- Unicode的码点取值范围：U+0000到U+10FFFF。
- UTF-32用4字节整数表示1个code point。
- 需要考虑字节序：UTF-32BE/UTF-32LE。
- 逻辑简单，例如求字符串长度和取字符串中第n个字符都是 $O(1)$ 的复杂度。
- 由于常用字符都集中在BMP中，单个字符占用4字节是巨大的浪费，实际使用不广泛。

# UTF-16

- 初始版本采用2字节整数表示1个code point，因此其表示范围为U+0000到U+FFFF，也即仅能表示BMP中的字符。
- 同样需考虑字节序：UTF-16BE/UTF-16LE。
- 存储效率适中，逻辑简单。微软选择UTF-16作为Windows中Unicode文本的标准编码方式。

# UTF-16 (cont.)

- 然而，不能表示BMP之外的字符是个巨大的缺陷，例如生僻汉字、emoji等等。
- 在Unicode 2.0中引入了**surrogates**，也称为surrogate pairs，使得在UTF-16中可以完整表示所有的Unicode code point。
- surrogates通过两个连续的code unit（也即两个连续的uint16），来表示一个值大于U+FFFF的码点。
- 在增强表示能力的同时，它也增加了UTF-16的复杂度。计算字符串长度等操作不再是 $O(1)$ 复杂度了。

# UTF-16 (cont. 2)

- 一个surrogate pair分为high和low两个部分，其二进制形式为  
110110xxxxxxxxxx 110111yyyyyyyyyy
- 已知Unicode码点的范围是[0, 0x10FFFF]。给定一个值大于U+FFFF的码点，将其值减去0x10000，结果的范围是[0, 0xFFFFF]，也即可以用**20个bit**来表示。将这20个bit分为高低两个10bit的部分，分别代入上面的xxxxxxxxxx和yyyyyyyyyyyy中，就得到了对应的surrogate。



# UTF-8

- 由 Ken Thompson 和 Rob Pike 在贝尔实验室的 Plan9 操作系统中首次实现。
- 是一种基于单字节(8 bit)编码单元、可变长度的Unicode字符编码方式。
- 与ASCII直接兼容，在储存英文字符串时空间效率高，是当今Internet和Linux世界的事实字符编码标准。

# UTF-8 (cont.)

- 首字节高位为0时，表明此code point用1个字节表示。
- 首字节高位为1时，有几个连续的1就说明此code point用几个连续字节来表示。
- 后继字节高2位为10，并带有6-bit的有效数据。

| UTF-8 (binary)                      | Code point (binary)    | Range            |
|-------------------------------------|------------------------|------------------|
| 0xxxxxxx                            | xxxxxxx                | U+0000–U+007F    |
| 110xxxxx 10yyyyyy                   | xxxxxyyyyyyy           | U+0080–U+07FF    |
| 1110xxxx 10yyyyyy 10zzzzzz          | xxxxyyyyyyyzzzzzz      | U+0800–U+FFFF    |
| 11110xxx 10yyyyyy 10zzzzzz 10wwwwww | xxxyyyyyyyzzzzzzwwwwww | U+10000–U+10FFFF |

# UTF-7

[UTF-7](#) (7-bit Unicode Transformation Format) is an **obsolete** variable-length character encoding for representing Unicode text using a stream of ASCII characters. It was originally intended to provide a means of encoding Unicode text **for use in Internet E-mail messages** that was more efficient than the combination of UTF-8 with quoted-printable.

# BOM

- Byte Order Mark 是一串特定的字节序列，通常放置于字符串的开始处，用来标识后继字符串的字节序。
- UTF-8这种单字节的编码理论上不存在字节序的问题，但为了更方便的进行字符串编码方式的识别，也设计了对应的BOM。
- BOM是可选的，不一定存在。

| Encoding             | Representation (hexadecimal) |
|----------------------|------------------------------|
| UTF-8 <sup>[a]</sup> | EF BB BF                     |
| UTF-16 (BE)          | FE FF                        |
| UTF-16 (LE)          | FF FE                        |
| UTF-32 (BE)          | 00 00 FE FF                  |
| UTF-32 (LE)          | FF FE 00 00                  |

# Combining character sequence

- 某些文字中会用到变音符号，例如：Café, Jalapeño, TÜV

|   |   |   |   |
|---|---|---|---|
| á | à | ä | â |
| é | è | ë | ê |
| í | ì | ï | î |
| ó | ò | ö | ô |
| ú | ù | ü | û |

→

|   |   |   |   |   |
|---|---|---|---|---|
| a | e | i | o | u |
|   |   | + |   |   |
| ´ | ` | ¨ | ^ |   |

- Unicode包含一类被称为"combining marks"的字符，它们可以与基字符(base character)进行组合。例如得到一个带变音符号的拉丁字母。

# CCS (cont.)

- "Combining marks"字符总是与它前面的字符进行组合。

| Code point sequences   | Result text |
|--|-------------|
| U+006F LATIN SMALL LETTER O<br>U+0308 COMBINING DIAERESIS<br>U+006F LATIN SMALL LETTER O | öö          |

- 当基字符有多个attachable slots时，mark字符的顺序不影响结果。

| Code point sequences  | Result text |
|---|-------------|
| U+006F LATIN SMALL LETTER O<br>U+0302 COMBINING CIRCUMFLEX ACCENT<br>U+0323 COMBINING DOT BELOW | ộ          |
| U+006F LATIN SMALL LETTER O<br>U+0323 COMBINING DOT BELOW<br>U+0302 COMBINING CIRCUMFLEX ACCENT | ộ          |

# CCS (cont. 2)

- 当多个mark字符attach到同一位置时，其先后顺序对结果有影响。

| Code point sequences   | Result text |
|--|-------------|
| U+0075 LATIN SMALL LETTER U<br>U+0308 COMBINING DIAERESIS<br>U+0304 COMBINING MACRON | Ü           |
| U+0075 LATIN SMALL LETTER U<br>U+0304 COMBINING MACRON<br>U+0308 COMBINING DIAERESIS | Ü           |

# Precomposed characters

- 但是为了向后兼容，Unicode也包括了一堆已经预先组合好的字符，直接分配了码点。

|        |        |        |        |        |        |        |        |
|--------|--------|--------|--------|--------|--------|--------|--------|
| À      | Á      | Â      | Ã      | Ä      | Å      | Æ      | Ç      |
| U+00C0 | U+00C1 | U+00C2 | U+00C3 | U+00C4 | U+00C5 | U+00C6 | U+00C7 |
| È      | É      | Ê      | Ë      | Ì      | Í      | Î      | Ï      |
| U+00C8 | U+00C9 | U+00CA | U+00CB | U+00CC | U+00CD | U+00CE | U+00CF |
| Ð      | Ñ      | Ò      | Ó      | Ô      | Õ      | Ö      |        |
| U+00D0 | U+00D1 | U+00D2 | U+00D3 | U+00D4 | U+00D5 | U+00D6 |        |

- 一脸懵，感觉刚才这一堆聪明事都白做了...



# Unicode equivalence

- **Canonical equivalence** is a fundamental equivalency between characters or sequences of characters which represent the same abstract character, and which when correctly displayed should always have the same visual appearance and behavior.

| Subtype                     | Examples          |
|-----------------------------|-------------------|
| Combining sequence          | Ç ↔ C+̈́          |
| Ordering of combining marks | q+̈́+̈́ ↔ q+̈́+̈́ |
| Hangul & conjoining jamo    | 가 ↔ ㄱ + ㅏ         |
| Singleton equivalence       | Ω ↔ Ω             |

# Unicode equivalence (cont.)

- **Compatibility equivalence** is a weaker type of equivalence between characters which represent the same abstract character, but which may have distinct visual appearances or behaviors.

| Subtype                 | Examples      |   |     |
|-------------------------|---------------|---|-----|
| Circled variants        | ①             | → | 1   |
| Width variants          | ㇏             | → | 力   |
| Rotated variants        | ⸏             | → | {   |
|                         | ⸚             | → | }   |
| Superscripts/subscripts | $i^9$         | → | i9  |
|                         | $i_9$         | → | i9  |
| Fractions               | $\frac{1}{4}$ | → | 1/4 |

# Normalization Forms

- [Unicode Normalization Forms](#) are formally defined normalizations of Unicode strings which make it possible to determine whether any two Unicode strings are **equivalent** to each other.

| Form                         | Description   |
|------------------------------|---|
| Normalization Form D (NFD)   | Canonical Decomposition   |
| Normalization Form C (NFC)   | Canonical Decomposition,<br>followed by Canonical Composition     |
| Normalization Form KD (NFKD) | Compatibility Decomposition                                       |
| Normalization Form KC (NFKC) | Compatibility Decomposition,<br>followed by Canonical Composition |

# Normalization Forms (cont.)

Figure 4. Canonical Composites

| Source    |   | NFD       |            | NFC       |
|-----------|---|-----------|------------|-----------|
| Å<br>00C5 | : | Å<br>0041 | ◌̊<br>030A | Å<br>00C5 |
| Ô<br>00F4 | : | Ô<br>006F | ◌̂<br>0302 | Ô<br>00F4 |

Figure 5. Multiple Combining Marks

| Source     |   | NFD       |                       | NFC                             |
|------------|---|-----------|-----------------------|---------------------------------|
| Š<br>1E69  | : | Š<br>0073 | ◌̇<br>0323 ◌̇<br>0307 | Š<br>1E69                       |
| đ<br>1E0B  | : | đ<br>0064 | ◌̇<br>0323 ◌̇<br>0307 | đ<br>1E0D ◌̇<br>0307            |
| q̇<br>0071 | : | q<br>0071 | ◌̇<br>0323 ◌̇<br>0307 | q<br>0071 ◌̇<br>0323 ◌̇<br>0307 |

# Normalization Forms (cont. 2)

Figure 6. Compatibility Composites

| Source                      |   | NFD                     |  | NFC              |  | NFKD                    |  | NFKC             |
|-----------------------------|---|-------------------------|--|------------------|--|-------------------------|--|------------------|
| fi<br>FB01                  | : | fi<br>FB01              |  | fi<br>FB01       |  | f i<br>0066 0069        |  | f i<br>0066 0069 |
| 2 <sup>5</sup><br>0032 2075 | : | 2 5<br>0032 2075        |  | 2 5<br>0032 2075 |  | 2 5<br>0032 0035        |  | 2 5<br>0032 0035 |
| ḟ<br>1E9B 0323              | : | f ̣ ̣<br>017F 0323 0307 |  | ḟ ̣<br>1E9B 0323 |  | s ̣ ̣<br>0073 0323 0307 |  | ṡ<br>1E69        |

# Normalization Forms (cont. 3)

- The [Unicode Normalization Algorithm](#) is fairly complex.
- Use library, e.g. [ICU](#)

```
→ ~ python3
Python 3.8.6 (default, Oct 8 2020, 14:06:32)
[Clang 12.0.0 (clang-1200.0.32.2)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import unicodedata
>>> s1 = '\u1e0b\u0323'
>>> s1
'ḋ'
>>> t1 = unicodedata.normalize('NFD', s1)
>>> t2 = unicodedata.normalize('NFC', s1)
>>> print(ascii(t1))
'd\u0323\u0307'
>>> print(ascii(t2))
'\u1e0d\u0307'
>>> 
```

# Grapheme Clusters

- 如前所述，在Unicode中一个用户所感知的字符可能有多种底层表示方式。我们将这样的"字符"称为"Grapheme Cluster"，其具体定义见[UAX #29](#)。
- 显而易见，在文本编辑领域需要细致的处理，以确保光标的位置以及选中区域的边界，能正确的落在grapheme cluster boundary上。
- 另一种情况是字符串超过长度限制需要进行截断处理时（例如数据库字段限制最多xx字节）。首先需要在code point边界上进行截断（例如不能在UTF-8的多个字节序列中，否则会导致非法字符串），然后需要考虑grapheme cluster边界以免改变字符逻辑含义。

# CJK Unified Ideographs

- 东亚文字多为表意文字，通常字符个数众多，且历史悠久，存在各种文化变迁与融合。
- 以汉字为例，就存在于简体中文、繁体中文、日文、韩文和越南文中。相互之间存在交集，但并不相同。
- Unicode一开始基于2字节Code space来设计，最大支持65536个码点，其中的20940(~32%)被保留给CJK文字。这些空间明显不能支持全部的CJK字符，因此通过[Han unification](#)将CJK中的交集部分尽可能统一化，以减少总的字符个数。

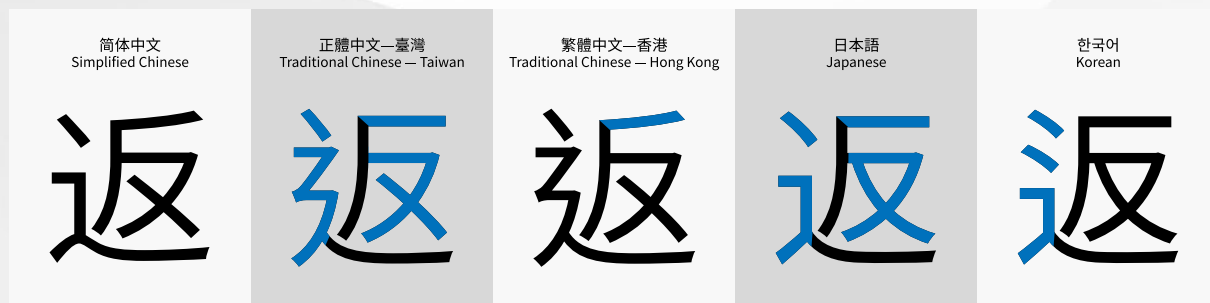


# CJK Unified Ideographs (cont.)

- Unicode为CJK文字分配码点的[3轴哲学](#):
  - X-variants: 语义上不同的字符，例如：U+6C49 汉 和 U+5B57 字。
  - Y-variants: 语义上相同但外观差异明显的字符，例如：U+732B 猫 和 U+8C93 貓。
  - Z-variants: 语义上相同且外观差异细微的字符，例如：U+8358 莊 和 U+838A 莊，U+8AAC 說 和 U+8AAA 說。
- Z-variants理论上应该被统一，但基于兼容性等因素而独立分配了码点（从某旧字符编码字符串转换为Unicode再转回来，尽量无损）。

# CJK Unified Ideographs (cont. 2)

- 虽然存在基础原则，在实践中某些字符是否要统一，仍然受到各种复杂因素的影响。
- 某些被统一了码点的字符，在不同语言的书写习惯上仍可能不同。例如U+8FD4:



- 使得无法简单的基于code point来选择字体，必须再附加上locale上下文。增加了text stack的实现复杂度（例如font fallback时）。

# CJK Unified Ideographs (cont. 3)

- Unicode 13.0中定义了92,856个CJK统一表意字符。
- Block **CJK Unified Ideographs** (4E00–9FFF) contains 20,989 basic Chinese characters.
- Block **CJK Unified Ideographs Extension A** (3400–4DBF) contains 6,592 additional characters.
- Block **CJK Unified Ideographs Extension B** (20000–2A6DF) contains 42,718 characters.
- ...
- Block **CJK Unified Ideographs Extension G** (30000-3134F) contains 4,939 characters.

# BIDI

- BIDI是Unicode Bidirectional Algorithm的简称。
- 不同的文字有不同的书写方向，通常为LTR和RTL。



- 当具有不同书写方向的文字混合在一行时，就需要引入bidi算法。



- 与此同时，底层字符串中的存储顺序并不理解文字方向。

# BIDI (cont.)

- 在细化到具体文字的方向之前，首先需要确定基础书写方向(**Base Direction**)，或者说上下文方向。
- Base Direction = LTR

bahrain مصر kuwait

1 2 3

- Base Direction = RTL

kuwait مصر bahrain

3 2 1

# BIDI (cont. 2)

- 每一个Unicode字符都被赋予了一个方向性属性。

| 方向性类别                            | 说明                              | 示例文字        |
|----------------------------------|---------------------------------|-------------|
| Strong Left-to-Right             | 强字符从左至右，不受上下文影响，并可能影响其前后字符的方向性。 | 英文字母、汉字     |
| Strong Right-to-Left             | 强字符从右至左，不受上下文影响，并可能影响其前后字符的方向性。 | 阿拉伯文字、希伯来文字 |
| Neutral                          | 中性，方向性不确定，由上下文决定。               | 大部分标点符号和空格  |
| Weak Left-to-Right/Right-to-Left | 方向性确定但不影响前后字符。                  | 数字和数字相关的符号  |

# BIDI (cont. 3)

- 位于两个强类型字符之间的中性字符(例如空格), 将跟随强类型字符的方向性。

The title is مفتاح معايير الويب in Arabic.



- 若某个中性字符, 位于两个方向相反的强字符之间呢? 这时候受基础方向 (上下文) 控制。

The title is مفتاح معايير الويب, in Arabic.



- 有些时候的效果不是我们想要的:

The title is !مفتاح معايير الويب in Arabic.



# BIDI (cont. 4)

- 数字通常是弱类型的字符，其方向性是确定的。

one two ثلاثة ١٢٣٤ خمسة  
one two אחד 1234 חמש

- 某些字符根据当前的文字方向具有**镜像**的显示效果，例如下图中的尖括号（在两行中使用的都是完全相同的字符）。

a > b > c

א < ב < ג



# BIDI (cont. 5)

- 某些场景需要显式的进行方向控制，因此Unicode设计了Explicit Markers字符：

|   |         |                                  |
|---|---------|----------------------------------|
| 1 | U+202A: | LEFT-TO-RIGHT EMBEDDING (LRE)    |
| 2 | U+202B: | RIGHT-TO-LEFT EMBEDDING (RLE)    |
| 3 | U+202D: | LEFT-TO-RIGHT OVERRIDE (LRO)     |
| 4 | U+202E: | RIGHT-TO-LEFT OVERRIDE (RLO)     |
| 5 | U+202C: | POP DIRECTIONAL FORMATTING (PDF) |

- OVERRIDE类的Marker可以强制改变文字方向：

How was your day? 还行。

。行还 ?yad ruoy saw woh

<- 行首插入 U+202E

# BIDI (cont. 6)

- BIDI在光标移动、文字块选时也有非常多的逻辑要处理。
- 参考[UAX #9](#)
- Use library, e.g. [ICU](#), [GNU FriBidi](#).

# Ideographic Description Sequence

- 表意文字由部首和笔划组成。Unicode定义了一类用来描述部首组合形式的字符(Ideographic Description Char), 例如: 𠂇, 𠂈, 𠂉。
- 基于IDS数据库可以得到某个表意文字的表意组字序列。

```
➔ ids ./ids
Database: IDS.txt
Records : 92856







Char: 国
Code: U+56FD
IDS : 囙 口 玉

Char: 港
Code: U+6E2F
IDS : 𠂇 𠂈 巷

Char: 丽
Code: U+4E3D
IDS : 𠂇 一 𠂈 𠂉 𠂊 𠂋 𠂌 𠂍 𠂎 𠂏 𠂐 𠂑 𠂒 𠂓 𠂔 𠂕 𠂖 𠂗 𠂘 𠂙 𠂚 𠂛 𠂜 𠂝 𠂞 𠂟 𠂠 𠂡 𠂢 𠂣 𠂤 𠂥 𠂦 𠂧 𠂨 𠂩 𠂪 𠂫 𠂬 𠂭 𠂮 𠂯 𠂰 𠂱 𠂲 𠂳 𠂴 𠂵 𠂶 𠂷 𠂸 𠂹 𠂺 𠂻 𠂼 𠂽 𠂾 𠂿 𠃀 𠃁 𠃂 𠃃 𠃄 𠃅 𠃆 𠃇 𠃈 𠃉 𠃊 𠃋 𠃌 𠃍 𠃎 𠃏 𠃐 𠃑 𠃒 𠃓 𠃔 𠃕 𠃖 𠃗 𠃘 𠃙 𠃚 𠃛 𠃜 𠃝 𠃞 𠃟 𠃠 𠃡 𠃢 𠃣 𠃤 𠃥 𠃦 𠃧 𠃨 𠃩 𠃪 𠃫 𠃬 𠃭 𠃮 𠃯 𠃰 𠃱 𠃲 𠃳 𠃴 𠃵 𠃶 𠃷 𠃸 𠃹 𠃺 𠃻 𠃼 𠃽 𠃾 𠃿 𠄀 𠄁 𠄂 𠄃 𠄄 𠄅 𠄆 𠄇 𠄈 𠄉 𠄊 𠄋 𠄌 𠄍 𠄎 𠄏 𠄐 𠄑 𠄒 𠄓 𠄔 𠄕 𠄖 𠄗 𠄘 𠄙 𠄚 𠄛 𠄜 𠄝 𠄞 𠄟 𠄠 𠄡 𠄢 𠄣 𠄤 𠄥 𠄦 𠄧 𠄨 𠄩 𠄪 𠄫 𠄬 𠄭 𠄮 𠄯 𠄰 𠄱 𠄲 𠄳 𠄴 𠄵 𠄶 𠄷 𠄸 𠄹 𠄺 𠄻 𠄼 𠄽 𠄾 𠄿 𠅀 𠅁 𠅂 𠅃 𠅄 𠅅 𠅆 𠅇 𠅈 𠅉 𠅊 𠅋 𠅌 𠅍 𠅎 𠅏 𠅐 𠅑 𠅒 𠅓 𠅔 𠅕 𠅖 𠅗 𠅘 𠅙 𠅚 𠅛 𠅜 𠅝 𠅞 𠅟 𠅠 𠅡 𠅢 𠅣 𠅤 𠅥 𠅦 𠅧 𠅨 𠅩 𠅪 𠅫 𠅬 𠅭 𠅮 𠅯 𠅰 𠅱 𠅲 𠅳 𠅴 𠅵 𠅶 𠅷 𠅸 𠅹 𠅺 𠅻 𠅼 𠅽 𠅾 𠅿 𠆀 𠆁 𠆂 𠆃 𠆄 𠆅 𠆆 𠆇 𠆈 𠆉 𠆊 𠆋 𠆌 𠆍 𠆎 𠆏 𠆐 𠆑 𠆒 𠆓 𠆔 𠆕 𠆖 𠆗 𠆘 𠆙 𠆚 𠆛 𠆜 𠆝 𠆞 𠆟 𠆠 𠆡 𠆢 𠆣 𠆤 𠆥 𠆦 𠆧 𠆨 𠆩 𠆪 𠆫 𠆬 𠆭 𠆮 𠆯 𠆰 𠆱 𠆲 𠆳 𠆴 𠆵 𠆶 𠆷 𠆸 𠆹 𠆺 𠆻 𠆼 𠆽 𠆾 𠆿 𠇀 𠇁 𠇂 𠇃 𠇄 𠇅 𠇆 𠇇 𠇈 𠇉 𠇊 𠇋 𠇌 𠇍 𠇎 𠇏 𠇐 𠇑 𠇒 𠇓 𠇔 𠇕 𠇖 𠇗 𠇘 𠇙 𠇚 𠇛 𠇜 𠇝 𠇞 𠇟 𠇠 𠇡 𠇢 𠇣 𠇤 𠇥 𠇦 𠇧 𠇨 𠇩 𠇪 𠇫 𠇬 𠇭 𠇮 𠇯 𠇰 𠇱 𠇲 𠇳 𠇴 𠇵 𠇶 𠇷 𠇸 𠇹 𠇺 𠇻 𠇼 𠇽 𠇾 𠇿 𠈀 𠈁 𠈂 𠈃 𠈄 𠈅 𠈆 𠈇 𠈈 𠈉 𠈊 𠈋 𠈌 𠈍 𠈎 𠈏 𠈐 𠈑 𠈒 𠈓 𠈔 𠈕 𠈖 𠈗 𠈘 𠈙 𠈚 𠈛 𠈜 𠈝 𠈞 𠈟 𠈠 𠈡 𠈢 𠈣 𠈤 𠈥 𠈦 𠈧 𠈨 𠈩 𠈪 𠈫 𠈬 𠈭 𠈮 𠈯 𠈰 𠈱 𠈲 𠈳 𠈴 𠈵 𠈶 𠈷 𠈸 𠈹 𠈺 𠈻 𠈼 𠈽 𠈾 𠈿 𠉀 𠉁 𠉂 𠉃 𠉄 𠉅 𠉆 𠉇 𠉈 𠉉 𠉊 𠉋 𠉌 𠉍 𠉎 𠉏 𠉐 𠉑 𠉒 𠉓 𠉔 𠉕 𠉖 𠉗 𠉘 𠉙 𠉚 𠉛 𠉜 𠉝 𠉞 𠉟 𠉠 𠉡 𠉢 𠉣 𠉤 𠉥 𠉦 𠉧 𠉨 𠉩 𠉪 𠉫 𠉬 𠉭 𠉮 𠉯 𠉰 𠉱 𠉲 𠉳 𠉴 𠉵 𠉶 𠉷 𠉸 𠉹 𠉺 𠉻 𠉼 𠉽 𠉾 𠉿 𠊀 𠊁 𠊂 𠊃 𠊄 𠊅 𠊆 𠊇 𠊈 𠊉 𠊊 𠊋 𠊌 𠊍 𠊎 𠊏 𠊐 𠊑 𠊒 𠊓 𠊔 𠊕 𠊖 𠊗 𠊘 𠊙 𠊚 𠊛 𠊜 𠊝 𠊞 𠊟 𠊠 𠊡 𠊢 𠊣 𠊤 𠊥 𠊦 𠊧 𠊨 𠊩 𠊪 𠊫 𠊬 𠊭 𠊮 𠊯 𠊰 𠊱 𠊲 𠊳 𠊴 𠊵 𠊶 𠊷 𠊸 𠊹 𠊺 𠊻 𠊼 𠊽 𠊾 𠊿 𠋀 𠋁 𠋂 𠋃 𠋄 𠋅 𠋆 𠋇 𠋈 𠋉 𠋊 𠋋 𠋌 𠋍 𠋎 𠋏 𠋐 𠋑 𠋒 𠋓 𠋔 𠋕 𠋖 𠋗 𠋘 𠋙 𠋚 𠋛 𠋜 𠋝 𠋞 𠋟 𠋠 𠋡 𠋢 𠋣 𠋤 𠋥 𠋦 𠋧 𠋨 𠋩 𠋪 𠋫 𠋬 𠋭 𠋮 𠋯 𠋰 𠋱 𠋲 𠋳 𠋴 𠋵 𠋶 𠋷 𠋸 𠋹 𠋺 𠋻 𠋼 𠋽 𠋾 𠋿 𠌀 𠌁 𠌂 𠌃 𠌄 𠌅 𠌆 𠌇 𠌈 𠌉 𠌊 𠌋 𠌌 𠌍 𠌎 𠌏 𠌐 𠌑 𠌒 𠌓 𠌔 𠌕 𠌖 𠌗 𠌘 𠌙 𠌚 𠌛 𠌜 𠌝 𠌞 𠌟 𠌠 𠌡 𠌢 𠌣 𠌤 𠌥 𠌦 𠌧 𠌨 𠌩 𠌪 𠌫 𠌬 𠌭 𠌮 𠌯 𠌰 𠌱 𠌲 𠌳 𠌴 𠌵 𠌶 𠌷 𠌸 𠌹 𠌺 𠌻 𠌼 𠌽 𠌾 𠌿 𠍀 𠍁 𠍂 𠍃 𠍄 𠍅 𠍆 𠍇 𠍈 𠍉 𠍊 𠍋 𠍌 𠍍 𠍎 𠍏 𠍐 𠍑 𠍒 𠍓 𠍔 𠍕 𠍖 𠍗 𠍘 𠍙 𠍚 𠍛 𠍜 𠍝 𠍞 𠍟 𠍠 𠍡 𠍢 𠍣 𠍤 𠍥 𠍦 𠍧 𠍨 𠍩 𠍪 𠍫 𠍬 𠍭 𠍮 𠍯 𠍰 𠍱 𠍲 𠍳 𠍴 𠍵 𠍶 𠍷 𠍸 𠍹 𠍺 𠍻 𠍼 𠍽 𠍾 𠍿 𠎀 𠎁 𠎂 𠎃 𠎄 𠎅 𠎆 𠎇 𠎈 𠎉 𠎊 𠎋 𠎌 𠎍 𠎎 𠎏 𠎐 𠎑 𠎒 𠎓 𠎔 𠎕 𠎖 𠎗 𠎘 𠎙 𠎚 𠎛 𠎜 𠎝 𠎞 𠎟 𠎠 𠎡 𠎢 𠎣 𠎤 𠎥 𠎦 𠎧 𠎨 𠎩 𠎪 𠎫 𠎬 𠎭 𠎮 𠎯 𠎰 𠎱 𠎲 𠎳 𠎴 𠎵 𠎶 𠎷 𠎸 𠎹 𠎺 𠎻 𠎼 𠎽 𠎾 𠎿 𠏀 𠏁 𠏂 𠏃 𠏄 𠏅 𠏆 𠏇 𠏈 𠏉 𠏊 𠏋 𠏌 𠏍 𠏎 𠏏 𠏐 𠏑 𠏒 𠏓 𠏔 𠏕 𠏖 𠏗 𠏘 𠏙 𠏚 𠏛 𠏜 𠏝 𠏞 𠏟 𠏠 𠏡 𠏢 𠏣 𠏤 𠏥 𠏦 𠏧 𠏨 𠏩 𠏪 𠏫 𠏬 𠏭 𠏮 𠏯 𠏰 𠏱 𠏲 𠏳 𠏴 𠏵 𠏶 𠏷 𠏸 𠏹 𠏺 𠏻 𠏼 𠏽 𠏾 𠏿 𠐀 𠐁 𠐂 𠐃 𠐄 𠐅 𠐆 𠐇 𠐈 𠐉 𠐊 𠐋 𠐌 𠐍 𠐎 𠐏 𠐐 𠐑 𠐒 𠐓 𠐔 𠐕 𠐖 𠐗 𠐘 𠐙 𠐚 𠐛 𠐜 𠐝 𠐞 𠐟 𠐠 𠐡 𠐢 𠐣 𠐤 𠐥 𠐦 𠐧 𠐨 𠐩 𠐪 𠐫 𠐬 𠐭 𠐮 𠐯 𠐰 𠐱 𠐲 𠐳 𠐴 𠐵 𠐶 𠐷 𠐸 𠐹 𠐺 𠐻 𠐼 𠐽 𠐾 𠐿 𠑀 𠑁 𠑂 𠑃 𠑄 𠑅 𠑆 𠑇 𠑈 𠑉 𠑊 𠑋 𠑌 𠑍 𠑎 𠑏 𠑐 𠑑 𠑒 𠑓 𠑔 𠑕 𠑖 𠑗 𠑘 𠑙 𠑚 𠑛 𠑜 𠑝 𠑞 𠑟 𠑠 𠑡 𠑢 𠑣 𠑤 𠑥 𠑦 𠑧 𠑨 𠑩 𠑪 𠑫 𠑬 𠑭 𠑮 𠑯 𠑰 𠑱 𠑲 𠑳 𠑴 𠑵 𠑶 𠑷 𠑸 𠑹 𠑺 𠑻 𠑼 𠑽 𠑾 𠑿 𠒀 𠒁 𠒂 𠒃 𠒄 𠒅 𠒆 𠒇 𠒈 𠒉 𠒊 𠒋 𠒌 𠒍 𠒎 𠒏 𠒐 𠒑 𠒒 𠒓 𠒔 𠒕 𠒖 𠒗 𠒘 𠒙 𠒚 𠒛 𠒜 𠒝 𠒞 𠒟 𠒠 𠒡 𠒢 𠒣 𠒤 𠒥 𠒦 𠒧 𠒨 𠒩 𠒪 𠒫 𠒬 𠒭 𠒮 𠒯 𠒰 𠒱 𠒲 𠒳 𠒴 𠒵 𠒶 𠒷 𠒸 𠒹 𠒺 𠒻 𠒼 𠒽 𠒾 𠒿 𠓀 𠓁 𠓂 𠓃 𠓄 𠓅 𠓆 𠓇 𠓈 𠓉 𠓊 𠓋 𠓌 𠓍 𠓎 𠓏 𠓐 𠓑 𠓒 𠓓 𠓔 𠓕 𠓖 𠓗 𠓘 𠓙 𠓚 𠓛 𠓜 𠓝 𠓞 𠓟 𠓠 𠓡 𠓢 𠓣 𠓤 𠓥 𠓦 𠓧 𠓨 𠓩 𠓪 𠓫 𠓬 𠓭 𠓮 𠓯 𠓰 𠓱 𠓲 𠓳 𠓴 𠓵 𠓶 𠓷 𠓸 𠓹 𠓺 𠓻 𠓼 𠓽 𠓾 𠓿 𠔀 𠔁 𠔂 𠔃 𠔄 𠔅 𠔆 𠔇 𠔈 𠔉 𠔊 𠔋 𠔌 𠔍 𠔎 𠔏 𠔐 𠔑 𠔒 𠔓 𠔔 𠔕 𠔖 𠔗 𠔘 𠔙 𠔚 𠔛 𠔜 𠔝 𠔞 𠔟 𠔠 𠔡 𠔢 𠔣 𠔤 𠔥 𠔦 𠔧 𠔨 𠔩 𠔪 𠔫 𠔬 𠔭 𠔮 𠔯 𠔰 𠔱 𠔲 𠔳 𠔴 𠔵 𠔶 𠔷 𠔸 𠔹 𠔺 𠔻 𠔼 𠔽 𠔾 𠔿 𠕀 𠕁 𠕂 𠕃 𠕄 𠕅 𠕆 𠕇 𠕈 𠕉 𠕊 𠕋 𠕌 𠕍 𠕎 𠕏 𠕐 𠕑 𠕒 𠕓 𠕔 𠕕 𠕖 𠕗 𠕘 𠕙 𠕚 𠕛 𠕜 𠕝 𠕞 𠕟 𠕠 𠕡 𠕢 𠕣 𠕤 𠕥 𠕦 𠕧 𠕨 𠕩 𠕪 𠕫 𠕬 𠕭 𠕮 𠕯 𠕰 𠕱 𠕲 𠕳 𠕴 𠕵 𠕶 𠕷 𠕸 𠕹 𠕺 𠕻 𠕼 𠕽 𠕾 𠕿 𠖀 𠖁 𠖂 𠖃 𠖄 𠖅 𠖆 𠖇 𠖈 𠖉 𠖊 𠖋 𠖌 𠖍 𠖎 𠖏 𠖐 𠖑 𠖒 𠖓 𠖔 𠖕 𠖖 𠖗 𠖘 𠖙 𠖚 𠖛 𠖜 𠖝 𠖞 𠖟 𠖠 𠖡 𠖢 𠖣 𠖤 𠖥 𠖦 𠖧 𠖨 𠖩 𠖪 𠖫 𠖬 𠖭 𠖮 𠖯 𠖰 𠖱 𠖲 𠖳 𠖴 𠖵 𠖶 𠖷 𠖸 𠖹 𠖺 𠖻 𠖼 𠖽 𠖾 𠖿 𠗀 𠗁 𠗂 𠗃 𠗄 𠗅 𠗆 𠗇 𠗈 𠗉 𠗊 𠗋 𠗌 𠗍 𠗎 𠗏 𠗐 𠗑 𠗒 𠗓 𠗔 𠗕 𠗖 𠗗 𠗘 𠗙 𠗚 𠗛 𠗜 𠗝 𠗞 𠗟 𠗠 𠗡 𠗢 𠗣 𠗤 𠗥 𠗦 𠗧 𠗨 𠗩 𠗪 𠗫 𠗬 𠗭 𠗮 𠗯 𠗰 𠗱 𠗲 𠗳 𠗴 𠗵 𠗶 𠗷 𠗸 𠗹 𠗺 𠗻 𠗼 𠗽 𠗾 𠗿 𠘀 𠘁 𠘂 𠘃 𠘄 𠘅 𠘆 𠘇 𠘈 𠘉 𠘊 𠘋 𠘌 𠘍 𠘎 𠘏 𠘐 𠘑 𠘒 𠘓 𠘔 𠘕 𠘖 𠘗 𠘘 𠘙 𠘚 𠘛 𠘜 𠘝 𠘞 𠘟 𠘠 𠘡 𠘢 𠘣 𠘤 𠘥 𠘦 𠘧 𠘨 𠘩 𠘪 𠘫 𠘬 𠘭 𠘮 𠘯 𠘰 𠘱 𠘲 𠘳 𠘴 𠘵 𠘶 𠘷 𠘸 𠘹 𠘺 𠘻 𠘼 𠘽 𠘾 𠘿 𠙀 𠙁 𠙂 𠙃 𠙄 𠙅 𠙆 𠙇 𠙈 𠙉 𠙊 𠙋 𠙌 𠙍 𠙎 𠙏 𠙐 𠙑 𠙒 𠙓 𠙔 𠙕 𠙖 𠙗 𠙘 𠙙 𠙚 𠙛 𠙜 𠙝 𠙞 𠙟 𠙠 𠙡 𠙢 𠙣 𠙤 𠙥 𠙦 𠙧 𠙨 𠙩 𠙪 𠙫 𠙬 𠙭 𠙮 𠙯 𠙰 𠙱 𠙲 𠙳 𠙴 𠙵 𠙶 𠙷 𠙸 𠙹 𠙺 𠙻 𠙼 𠙽 𠙾 𠙿 𠚀 𠚁 𠚂 𠚃 𠚄 𠚅 𠚆 𠚇 𠚈 𠚉 𠚊 𠚋 𠚌 𠚍 𠚎 𠚏 𠚐 𠚑 𠚒 𠚓 𠚔 𠚕 𠚖 𠚗 𠚘 𠚙 𠚚 𠚛 𠚜 𠚝 𠚞 𠚟 𠚠 𠚡 𠚢 𠚣 𠚤 𠚥 𠚦 𠚧 𠚨 𠚩 𠚪 𠚫 𠚬 𠚭 𠚮 𠚯 𠚰 𠚱 𠚲 𠚳 𠚴 𠚵 𠚶 𠚷 𠚸 𠚹 𠚺 𠚻 𠚼 𠚽 𠚾 𠚿 𠛀 𠛁 𠛂 𠛃 𠛄 𠛅 𠛆 𠛇 𠛈 𠛉 𠛊 𠛋 𠛌 𠛍 𠛎 𠛏 𠛐 𠛑 𠛒 𠛓 𠛔 𠛕 𠛖 𠛗 𠛘 𠛙 𠛚 𠛛 𠛜 𠛝 𠛞 𠛟 𠛠 𠛡 𠛢 𠛣 𠛤 𠛥 𠛦 𠛧 𠛨 𠛩 𠛪 𠛫 𠛬 𠛭 𠛮 𠛯 𠛰 𠛱 𠛲 𠛳 𠛴 𠛵 𠛶 𠛷 𠛸 𠛹 𠛺 𠛻 𠛼 𠛽 𠛾 𠛿 𠜀 𠜁 𠜂 𠜃 𠜄 𠜅 𠜆 𠜇 𠜈 𠜉 𠜊 𠜋 𠜌 𠜍 𠜎 𠜏 𠜐 𠜑 𠜒 𠜓 𠜔 𠜕 𠜖 𠜗 𠜘 𠜙 𠜚 𠜛 𠜜 𠜝 𠜞 𠜟 𠜠 𠜡 𠜢 𠜣 𠜤 𠜥 𠜦 𠜧 𠜨 𠜩 𠜪 𠜫 𠜬 𠜭 𠜮 𠜯 𠜰 𠜱 𠜲 𠜳 𠜴 𠜵 𠜶 𠜷 𠜸 𠜹 𠜺 𠜻 𠜼 𠜽 𠜾 𠜿 𠝀 𠝁 𠝂 𠝃 𠝄 𠝅 𠝆 𠝇 𠝈 𠝉 𠝊 𠝋 𠝌 𠝍 𠝎 𠝏 𠝐 𠝑 𠝒 𠝓 𠝔 𠝕 𠝖 𠝗 𠝘 𠝙 𠝚 𠝛 𠝜 𠝝 𠝞 𠝟 𠝠 𠝡 𠝢 𠝣 𠝤 𠝥 𠝦 𠝧 𠝨 𠝩 𠝪 𠝫 𠝬 𠝭 𠝮 𠝯 𠝰 𠝱 𠝲 𠝳 𠝴 𠝵 𠝶 𠝷 𠝸 𠝹 𠝺 𠝻 𠝼 𠝽 𠝾 𠝿 𠞀 𠞁 𠞂 𠞃 𠞄 𠞅 𠞆 𠞇 𠞈 𠞉 𠞊 𠞋 𠞌 𠞍 𠞎 𠞏 𠞐 𠞑 𠞒 𠞓 𠞔 𠞕 𠞖 𠞗 𠞘 𠞙 𠞚 𠞛 𠞜 𠞝 𠞞 𠞟 𠞠 𠞡 𠞢 𠞣 𠞤 𠞥 𠞦 𠞧 𠞨 𠞩 𠞪 𠞫 𠞬 𠞭 𠞮 𠞯 𠞰 𠞱 𠞲 𠞳 𠞴 𠞵 𠞶 𠞷 𠞸 𠞹 𠞺 𠞻 𠞼 𠞽 𠞾 𠞿 𠟀 𠟁 𠟂 𠟃 𠟄 𠟅 𠟆 𠟇 𠟈 𠟉 𠟊 𠟋 𠟌 𠟍 𠟎 𠟏 𠟐 𠟑 𠟒 𠟓 𠟔 𠟕 𠟖 𠟗 𠟘 𠟙 𠟚 𠟛 𠟜 𠟝 𠟞 𠟟 𠟠 𠟡 𠟢 𠟣 𠟤 𠟥 𠟦 𠟧 𠟨 𠟩 𠟪 𠟫 𠟬 𠟭 𠟮 𠟯 𠟰 𠟱 𠟲 𠟳 𠟴 𠟵 𠟶 𠟷 𠟸 𠟹 𠟺 𠟻 𠟼 𠟽 𠟾 𠟿 𠠀 𠠁 𠠂 𠠃 𠠄 𠠅 𠠆 𠠇 𠠈 𠠉 𠠊 𠠋 𠠌 𠠍 𠠎 𠠏 𠠐 𠠑 𠠒 𠠓 𠠔 𠠕 𠠖 𠠗 𠠘 𠠙 𠠚 𠠛 𠠜 𠠝 𠠞 𠠟 𠠠 𠠡 𠠢 𠠣 𠠤 𠠥 𠠦 𠠧 𠠨 𠠩 𠠪 𠠫 𠠬 𠠭 𠠮 𠠯 𠠰 𠠱 𠠲 𠠳 𠠴 𠠵 𠠶 𠠷 𠠸 𠠹 𠠺 𠠻 𠠼 𠠽 𠠾 𠠿 𠡀 𠡁 𠡂 𠡃 𠡄 𠡅 𠡆 𠡇 𠡈 𠡉 𠡊 𠡋 𠡌 𠡍 𠡎 𠡏 𠡐 𠡑 𠡒 𠡓 𠡔 𠡕 𠡖 𠡗 𠡘 𠡙 𠡚 𠡛 𠡜 𠡝 𠡞 𠡟 𠡠 𠡡 𠡢 𠡣 𠡤 𠡥 𠡦 𠡧 𠡨 𠡩 𠡪 𠡫 𠡬 𠡭 𠡮 𠡯 𠡰 𠡱 𠡲 𠡳 𠡴 𠡵 𠡶 𠡷 𠡸 𠡹 𠡺 𠡻 𠡼 𠡽 𠡾 𠡿 𠢀 𠢁 𠢂 𠢃 𠢄 𠢅 𠢆 𠢇 𠢈 𠢉 𠢊 𠢋 𠢌 𠢍 𠢎 𠢏 𠢐 𠢑 𠢒 𠢓 𠢔 𠢕 𠢖 𠢗 𠢘 𠢙 𠢚 𠢛 𠢜 𠢝 𠢞 𠢟 𠢠 𠢡 𠢢 𠢣 𠢤 𠢥 𠢦 𠢧 𠢨 𠢩 𠢪 𠢫 𠢬 𠢭 𠢮 𠢯 𠢰 𠢱 𠢲 𠢳 𠢴 𠢵 𠢶 𠢷 𠢸 𠢹 𠢺 𠢻 𠢼 𠢽 𠢾 𠢿 𠣀 𠣁 𠣂 𠣃 𠣄 𠣅 𠣆 𠣇 𠣈 𠣉 𠣊 𠣋 𠣌 𠣍 𠣎 𠣏 𠣐 𠣑 𠣒 𠣓 𠣔 𠣕 𠣖 𠣗 𠣘 𠣙 𠣚 𠣛 𠣜 𠣝 𠣞 𠣟 𠣠 𠣡 𠣢 𠣣 𠣤 𠣥 𠣦 𠣧 𠣨 𠣩 𠣪 𠣫 𠣬 𠣭 𠣮 𠣯 𠣰 𠣱 𠣲 𠣳 𠣴 𠣵 𠣶 𠣷 𠣸 𠣹 𠣺 𠣻 𠣼 𠣽 𠣾 𠣿 𠤀 𠤁 𠤂 𠤃 𠤄 𠤅 𠤆 𠤇 𠤈 𠤉 𠤊 𠤋 𠤌 𠤍 𠤎 𠤏 𠤐 𠤑 𠤒 𠤓 𠤔 𠤕 𠤖 𠤗 𠤘 𠤙 𠤚 𠤛 𠤜 𠤝 𠤞 𠤟 𠤠 𠤡 𠤢 𠤣 𠤤 𠤥 𠤦 𠤧 𠤨 𠤩 𠤪 𠤫 𠤬 𠤭 𠤮 𠤯 𠤰 𠤱 𠤲 𠤳 𠤴 𠤵 𠤶 𠤷 𠤸 𠤹 𠤺 𠤻 𠤼 𠤽 𠤾 𠤿 𠥀 𠥁 𠥂 𠥃 𠥄 𠥅 𠥆 𠥇 𠥈 𠥉 𠥊 𠥋 𠥌 𠥍 𠥎 𠥏 𠥐 𠥑 𠥒 𠥓 𠥔 𠥕 𠥖 𠥗 𠥘 𠥙 𠥚 𠥛 𠥜 𠥝 𠥞 𠥟 𠥠 𠥡 𠥢 𠥣 𠥤 𠥥 𠥦 𠥧 𠥨 𠥩 𠥪 𠥫 𠥬 𠥭 𠥮 𠥯 𠥰 𠥱 𠥲 𠥳 𠥴 𠥵 𠥶 𠥷 𠥸 𠥹 𠥺 𠥻 𠥼 𠥽 𠥾 𠥿 𠦀 𠦁 𠦂 𠦃 𠦄 𠦅 𠦆 𠦇 𠦈 𠦉 𠦊 𠦋 𠦌 𠦍 𠦎 𠦏 𠦐 𠦑 𠦒 𠦓 𠦔 𠦕 𠦖 𠦗 𠦘 𠦙 𠦚 𠦛 𠦜 𠦝 𠦞 𠦟 𠦠 𠦡 𠦢 𠦣 𠦤 𠦥 𠦦 𠦧 𠦨 𠦩 𠦪 𠦫 𠦬 𠦭 𠦮 𠦯 𠦰 𠦱 𠦲 𠦳 𠦴 𠦵 𠦶 𠦷 𠦸 𠦹 𠦺 𠦻 𠦼 𠦽 𠦾 𠦿 𠧀 𠧁 𠧂 𠧃 𠧄 𠧅 𠧆 𠧇 𠧈 𠧉 𠧊 𠧋 𠧌 𠧍 𠧎 𠧏 𠧐 𠧑 𠧒 𠧓 𠧔 𠧕 𠧖 𠧗 𠧘 𠧙 𠧚 𠧛 𠧜 𠧝 𠧞 𠧟 𠧠 𠧡 𠧢 𠧣 𠧤 𠧥 𠧦 𠧧 𠧨 𠧩 𠧪 𠧫 𠧬 𠧭 𠧮 𠧯 𠧰 𠧱 𠧲 𠧳 𠧴 𠧵 𠧶 𠧷 𠧸 𠧹 𠧺 𠧻 𠧼 𠧽 𠧾 
```

# Emoji diversity

- Unicode Emoji的技术方案相当复杂，具体参考[UTS #51](#)。
- 举个例子，Emoji中包括很多"头像"，需要在技术方案上考虑肤色多样性。

| Code point sequences                          | Result text   |
|---|---|
| U+1F466 Boy                                   |    |
| U+1F466 Boy<br>U+1F3FB Light skin tone        |    |
| U+1F466 Boy<br>U+1F3FC Medium-light skin tone |   |
| U+1F466 Boy<br>U+1F3FD Medium skin tone       |  |
| U+1F466 Boy<br>U+1F3FE Medium-dark skin tone  |  |
| U+1F466 Boy<br>U+1F3FF Dark skin tone         |  |

# Some interesting unicode characters

- chess pieces: ♔ ♚ ♜ ♞ ♠ ♡ ♛ ♝ ♞ ♟ ♠ ♡
- playing card suits: ♥ ♦ ♠ ♣ ♥ ♦ ♠ ♣
- mahjong tiles: 東 南 西 北 發 冬
- dice: 🎲 🎲 🎲 🎲 🎲 🎲
- weather symbols: ☀ ☁ ☔ ☃ 🌨
- musical symbols: ♪ ♫ ♬ ♪ ♫ ♬



# Thanks