Long long ago…

• People wanted to use natural language words.

• But computers, at their most basic level, only has 0 and 1.

• Therefore there should be a way to represent words with 0 and 1.

• What’s more, this way need to be consistent across users so they can communicate with each other.
Encodings

• FIELDATA, by US Army
• ASCII, by ASA
• EBCDIC, by IBM

• But they are quite different.

• There would be lots of trouble switching back and forth between different encodings.
ISO

- ISO 2202
- ISO 8859
- ISO 8859-1, -2, -3, ⋯, -16
- They covered pretty much every single letter in European language alphabets.

- But what about other languages?
East Asian Languages

• Chinese has \(~23k\) different characters.
• Japanese has >50k different characters.
• There is no way to encode them within 1 byte like European alphabets.
• Then how?
Multibytes

• ISO 10646, extension of ISO646 (ASCII)
• Unicode, by Apple and Xerox

• ISO 10646 finally aligned with Unicode after 1993
But what is Unicode trying to solve?

• Character vs Glyph

• pool vs хорошо
Character

• Name, Code point, Representative Glyph, General Category, Combining Class, Bidirectional Category, Bidirectional Mirrored, Bidirectional Mirroring Glyph, Decimal Digit Value, Digit Value, Numeric Value, Ideographic, Default Ignorable, Deprecated, and more
Goals and Principles

• universal (addressing the needs of world languages),
• uniform (fixed-width codes for efficient access),
• unique (bit sequence has only one interpretation into character codes)

• Universality, efficiency, separate characters and glyphs, the well-defined semantics of characters, plain text, logical order, unification, dynamic composition, equivalent sequences, convertibility, (permanent stability)
Architecture

• 5 Levels:
  • abstract character repertoire
  • coded character set
  • character encoding form
  • character encoding scheme
  • transfer encoding syntax

Character map
abstract character repertoire

• 137,439 characters from 146 scripts, 65 control characters, 2789 emojis.
### Coded Character Set

- 7 planes:

<table>
<thead>
<tr>
<th>Plane 0</th>
<th>Plane 1</th>
<th>Plane 2</th>
<th>Planes 3–13</th>
<th>Plane 14</th>
<th>Planes 15–16</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000–FFFF</td>
<td>10000–1FFFF</td>
<td>20000–2FFFF</td>
<td>30000–DFFFF</td>
<td>E0000–EFFFF</td>
<td>F0000–10FFFF</td>
</tr>
</tbody>
</table>

#### Unicode Planes and Used Code Point Ranges

<table>
<thead>
<tr>
<th>Basic Multilingual Plane</th>
<th>Supplementary Multilingual Plane</th>
<th>Supplementary Ideographic Plane</th>
<th>Planes 3–13</th>
<th>Plane 14</th>
<th>Planes 15–16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic BMP</td>
<td>SMP</td>
<td>SIP</td>
<td>Planes 3–13</td>
<td>Plane 14</td>
<td>Planes 15–16</td>
</tr>
<tr>
<td>0000–0FFF</td>
<td>8000–8FFF</td>
<td>10000–10FFF</td>
<td>20000–20FFF</td>
<td>E0000–E0FFF</td>
<td>F0000–FFFF</td>
</tr>
<tr>
<td>1000–1FFF</td>
<td>9000–9FFF</td>
<td>11000–11FFF</td>
<td>21000–21FFF</td>
<td>F0000–FFFF</td>
<td>F0000–FFFF</td>
</tr>
<tr>
<td>2000–2FFF</td>
<td>A000–AFFF</td>
<td>12000–12FFF</td>
<td>22000–22FFF</td>
<td>F0000–FFFF</td>
<td>F0000–FFFF</td>
</tr>
<tr>
<td>3000–3FFF</td>
<td>B000–BFFF</td>
<td>13000–13FFF</td>
<td>23000–23FFF</td>
<td>F0000–FFFF</td>
<td>F0000–FFFF</td>
</tr>
<tr>
<td>4000–4FFF</td>
<td>C000–CFFF</td>
<td>14000–14FFF</td>
<td>24000–24FFF</td>
<td>F0000–FFFF</td>
<td>F0000–FFFF</td>
</tr>
<tr>
<td>5000–5FFF</td>
<td>D000–DFFF</td>
<td>1D000–1DFFF</td>
<td>25000–25FFF</td>
<td>F0000–FFFF</td>
<td>F0000–FFFF</td>
</tr>
<tr>
<td>6000–6FFF</td>
<td>E000–EFFF</td>
<td>1E000–1EFFF</td>
<td>26000–26FFF</td>
<td>F0000–FFFF</td>
<td>F0000–FFFF</td>
</tr>
<tr>
<td>7000–7FFF</td>
<td>F000–FFFF</td>
<td>1F000–1FFFF</td>
<td>27000–27FFF</td>
<td>F0000–FFFF</td>
<td>F0000–FFFF</td>
</tr>
</tbody>
</table>

- **Code Point Ranges:**
  - **BMP (Basic Multilingual Plane):** 0000–FFFF
  - **SMP (Supplementary Multilingual Plane):** 10000–FFFF
  - **SIP (Supplementary Ideographic Plane):** 20000–FFFF
  - **Planes 3–13:** 30000–DFFFF
  - **Plane 14:** E0000–EFFFF
  - **Planes 15–16:** F0000–10FFFF

- **Special-Purpose Planes:**
  - **Plan 14:** E0000–EFFFF
  - **Planes 15–16:** F0000–10FFFF
coded character set

• Each code point is described with a string starting with U+, followed by a bunch of hex digits.

• U+xxxx for BMP, U+xxxxxx or U+xxxxxxxx for rest.
Basic multilingual plane

• The most important plane
• Contains pretty much everything.
Supplementary multilingual plane

- Historic scripts.
Supplementary Ideographic Plane

• Mainly CJK (Chinese, Japanese and Korean), especially historical CJK Characters.
Supplementary Special-purpose Plane

- Tags and Variation Selectors
  Supplement
character encoding form

• Differ in different standards.
• UTF-8, UTF-16BE, UTF-32BE, UTF-16LE, UTF-32LE,
• Ofc there are more:
• Big5, Big5hkscs, Cp037, Cp424, Cp43, Cp50, Cp737, Cp775, Cp850, cp852, cp855, cp856
UTF-8

- Variable length.
- Predominant encoding, 77.3% of all websites.

<table>
<thead>
<tr>
<th>Number of bytes</th>
<th>Bits for code point</th>
<th>First code point</th>
<th>Last code point</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
<th>Byte 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7</td>
<td>U+0000</td>
<td>U+007F</td>
<td>0xxxxxx</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>U+0080</td>
<td>U+07FF</td>
<td>110xxxx</td>
<td>10xxxxx</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>16</td>
<td>U+0800</td>
<td>U+FFFF</td>
<td>1110xxxx</td>
<td>10xxxxx</td>
<td>10xxxxx</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>21</td>
<td>U+10000</td>
<td>U+10FFFF</td>
<td>11110xxx</td>
<td>10xxxxx</td>
<td>10xxxxx</td>
<td>10xxxxx</td>
</tr>
</tbody>
</table>
UCS-2 -> UTF-16

- Still Variable Length
- 2 bytes or 2* 2 bytes.
- The use of surrogates.
- E.g.
  - U+4BEEF
  - 0x4BEEF-0x010000=0x3BEEF
- Separate into two parts, top 10 and bottom 10.
- Top 10 is added to 0xD800, =0xD8EF
- Bot 10 is added to 0xDC00, = 0xDEEF
- Overall encoding is 0xD8EFDEEF
UTF-32

- Fixed length.
- Put everything into 32 bits.
Byte Order Mask
When you try to use Unicode without knowing the encoding…

```python
>>> a='每天过的都一样'.encode(encoding='UTF-8',errors='ignore')
>>> a
b'\xe6\xa4\xa9\x8f\xe5\xa4\xa9\xe5\xe8\xb0\xe6\xa0\xb7'
>>> a.decode(encoding='UTF-16',errors='ignore')
'蛟忓ぉ杩囩殑閮戒竴鏍'
>>> a.decode(encoding='gbk',errors='ignore')
'蛟忏㏙杩囥闆闂敲鍙'
>>> a.decode(encoding='gb2312',errors='ignore')
'蛟澶十戒'
>>> a.decode(encoding='cp950',errors='ignore')
'瘥燎抵賭'

Etc.
```
transfer encoding syntax

• transcription that can occur at the very end to adapt data to certain transmission environments.
• Something beyond the concern of Unicode.