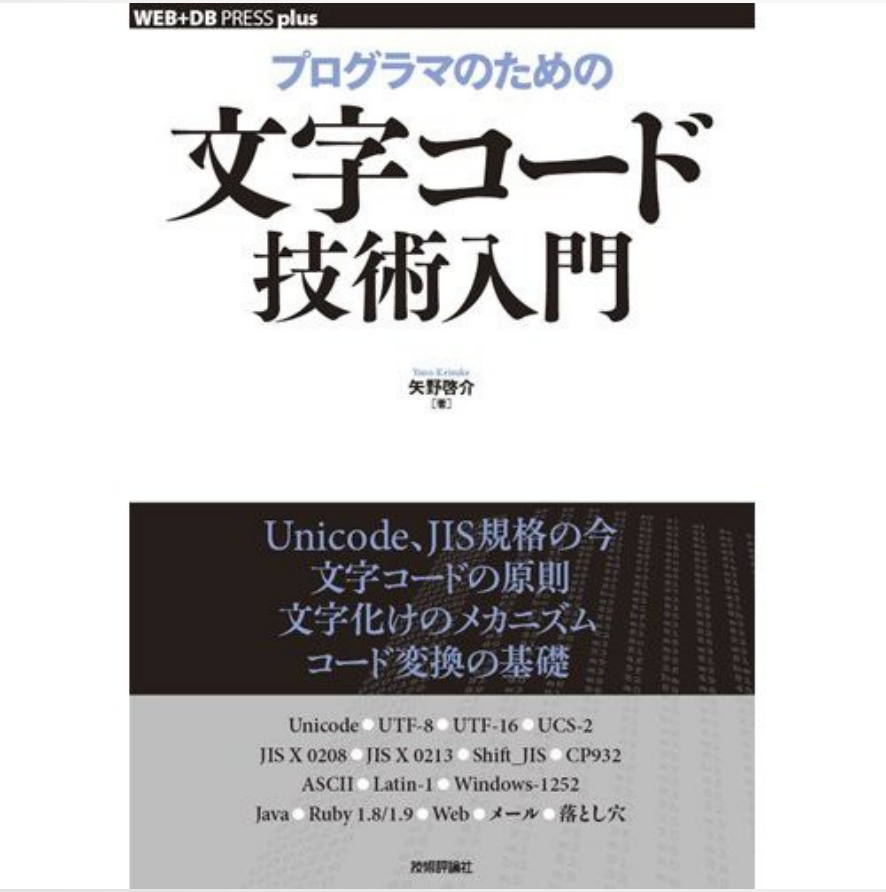
# 文字コード(3)

林部祐太 (NAIST)

@国立国会図書館 関西館 電子図書館課
2013/12/6

# 参考書

『プログラマのための文字コード技術入門』（技術評論社、2010年）




※特に注記がない場合，本スライドの図表は本書からの引用



2

# 予定

- 第1回　文字コードとは (2013-7)
  - 文字とコンピュータ
  - 文字コードの編成 (文字集合･符号化文字集合･文字符号化方式)
- 第2回　代表的な符号化文字集合･文字符号化方式 (2013-9)
  - 代表的な文字符号化方式(EUC-JP, ISO-2022-JP, Shift_JIS, UTF-16, UTF-32, UTF-8)
- 第3回　インターネットと文字コード (2013-12)
  - 文字コードの自動判別の仕組み
  - インターネットと文字コードの関係

# 今回のアウトライン

- インターネットと文字コード (参考書:5章6章相当)
  - 文字コードの変換
  - 文字コードの自動判別の仕組み
  - インターネットと文字コードの関係
  - 文字コードに関するトラブル

# 文字コードの変換

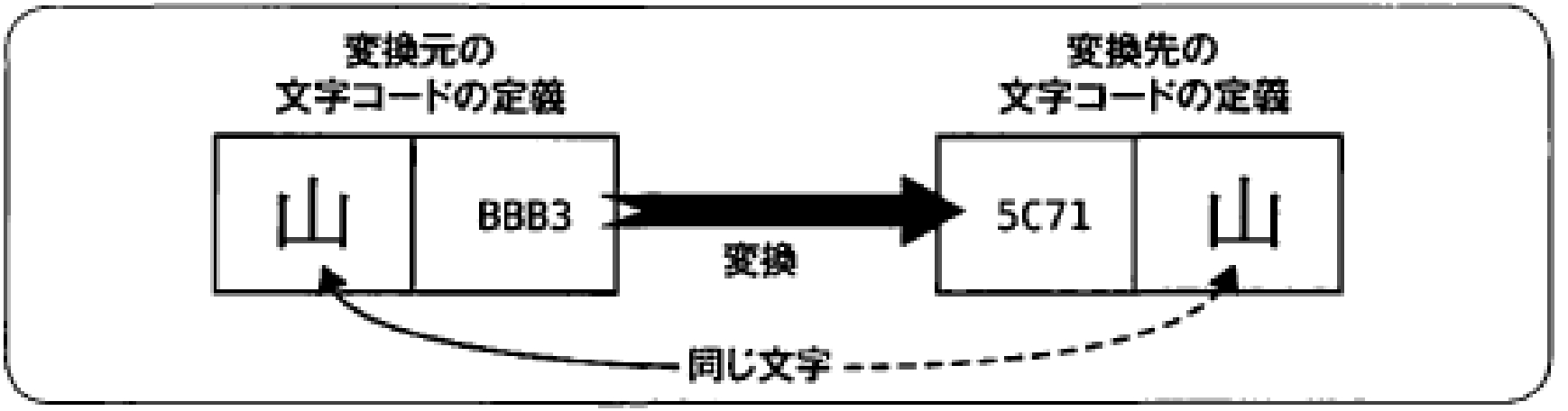
# 文字コード変換の例

- ISO-2022-JPで符号化された受信メール
  - 内部処理用の文字コードに変換
- 内部的にUTF-16 で表現されている文字列データ
  - ファイルにShift-JISやUTF-8で保存

# 文字コード変換の原則

- 符号化されたテキスト却の文字を変えずにコードだけを変換する

図5.2　変換の原則



# 包摂基準が異なる場合

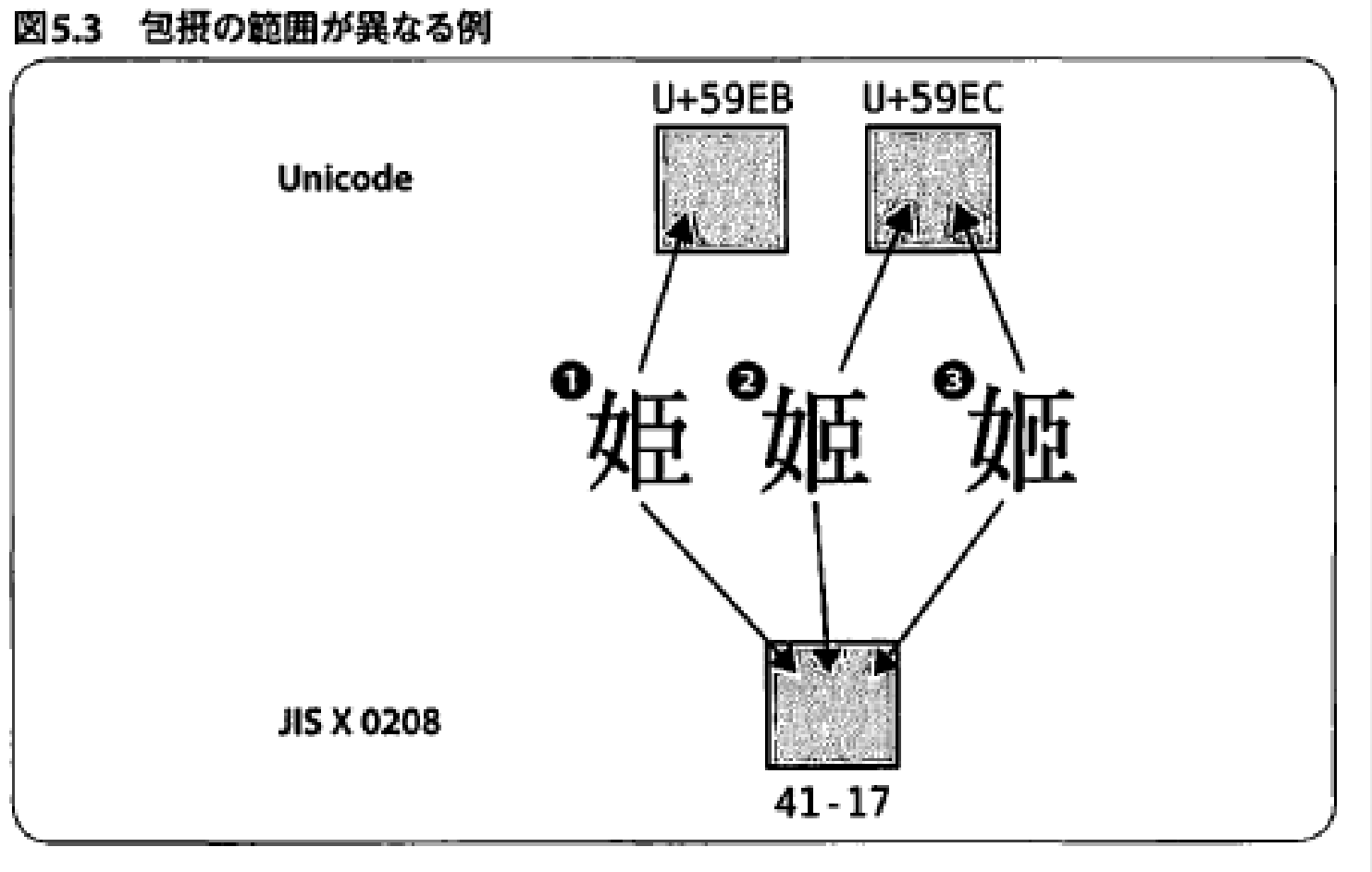
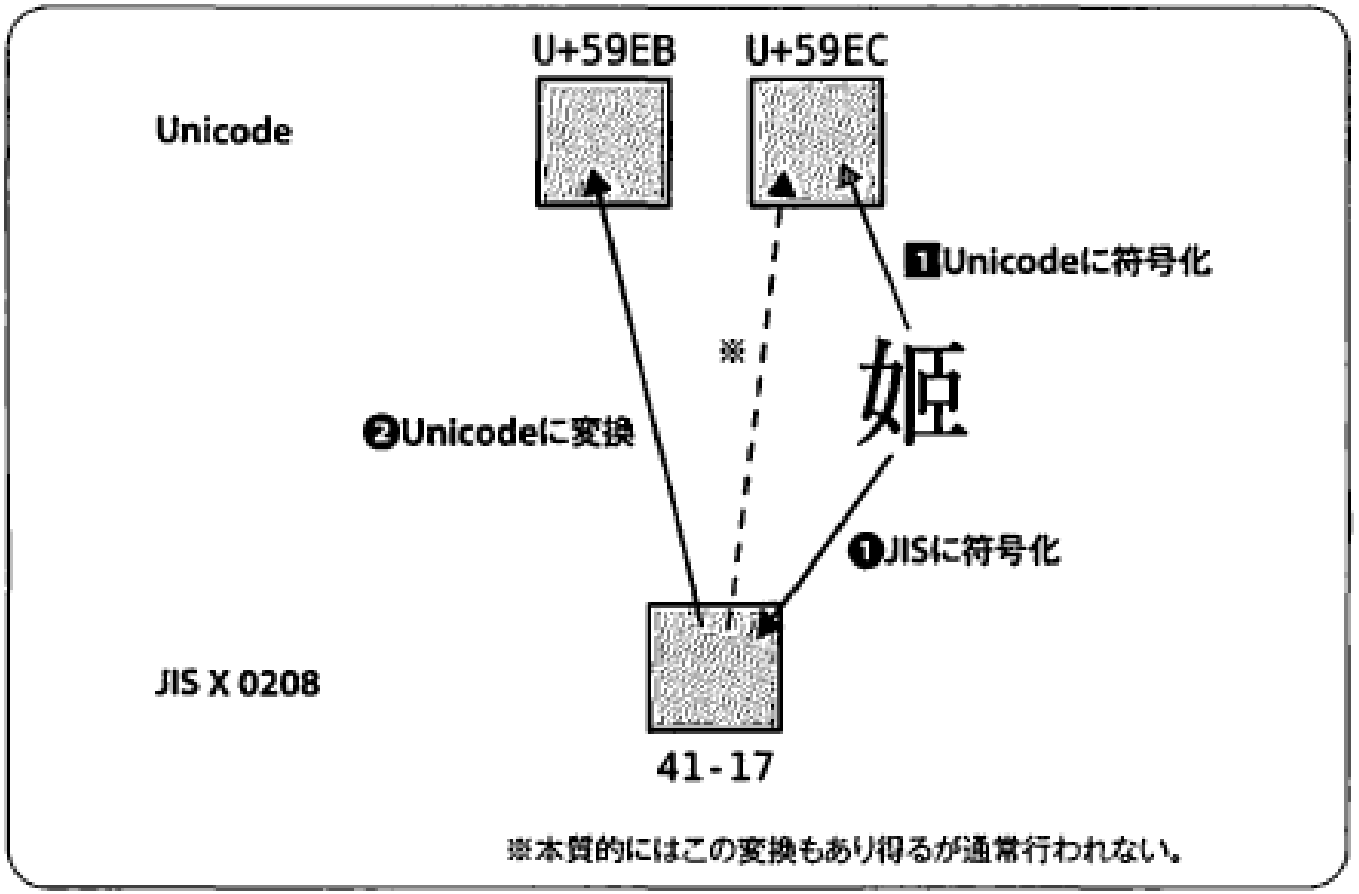


図5.4　包摂の範囲が異なる文字集合間の変換の例



# 文字コードの自動判別の仕組み

# 文字コードの自動判別(1)

- SHIFT_JIS, EUC-JP, ISO-2022-JP
  - ISO-2022-JP　エスケープシーケンスがある
  - SHIFT_JIS　0xA1～0xDFはあまり使われない
- UTF-16, UTF-8, UTF-32
  - 文書先頭のバイト順マーク(BOM)
    - UTF-16　BE　　0xFE 0xFF
    - UTF-16　LE　　0xFF 0xFE
    - UTF-32　BE　　0x00 0x00 0xFE 0xFF
    - UTF-32　LE　　0xFF 0xFE 0x00 0x00

# 文字コードの自動判別(2)

- 使用頻度の偏りを利用する
  - 0xA1～0xDF　EUC-JPの可能性が高い
  - 0x80～0x9F　Shift_JISの可能性が高い
- 例
  - 「入口」はEUC-JPでは”C6 FE 88 FD”
  - 第2バイトのFE,FDはいずれもShift_JISでは現れない値

- ただし「大方うまく判定できる」という程度
  - 正しく文字コードを文書内で指定するべき(後述)

# インターネットと文字コード

# インターネットと文字コード

- 文字コードの扱いについての概要
  - 電子メール
  - HTML
  - XML
  - URL
  - HTTP

# 電子メール

- もともとASCII(7bit)のみ対応
  - 日本語の7bit文字コード　ISO-2022-JP
- MIME (Multipurpose Internet Mail Extensions)
  - 電子メールの拡張
  - メールの本文やヘッダにさまざまな文字コード
  - 画像、音声などテキスト以外のデータも扱えるように
  - 添付ファイルのように、複数の部分での構成も可
  - 現在，殆ど全てのメーラーが対応している

# MIMEの例

- MIMEの例

```
Content-Type: text/plain; charset=iso-2022-jp
Content-Transfer-Encoding: 7bit
Subject: =?iso-2022-jp?B?GyRCJDQwJzsiGyhC?=
Frorn: a@exarnple.com
To: b@exarnple.com
Oate: Sat , 11 Jul 2999 21:46:35 +99ge (JST)

こんにちは.これは本文です.
```

本文の文字コードを指定

ヘッダ中の非ASCII文字は
“Q符号化”(または“B符号化”)を行う

- 8bit符号(Latin-1, UTF-8等)
  - 符号化が必要 (quoted-printable, base64)
  - 詳細は割愛

# 電子メールの添付ファイル名のエンコード

- MIMEエンコード（RFC2047）
  - Outlook Express, Windows Mailer, Gmail

```
Content-Type: application/octet-stream;
 name="=?ISO-2022-JP?B?GyRCJF4kJCRhGyhCLnR4dA==?="
Content-Disposition: attachment;
 filename="=?ISO-2022-JP?B?GyRCJF4kJCRhGyhCLnR4dA==?="
Content-Transfer-Encoding: base64
```

- URLエンコード（RFC2231）
  - Thunderbird

```
Content-Disposition: attachment; filename*0*=utf-8''%E3%81%82%E3%81%84%E3%81%86%E3%81;
      filename*1*=%88%E3%81%8A.doc
Content-Type: application/msword; name*0*=utf-8''%E3%81%82%E3%81%84%E3%81%86%E3%81;
      name*1*=%88%E3%81%8A.doc
Content-Transfer-Encoding: Base64
```

# 電子メールの文字コードの実際

- 日本語を送るなら，ISO-2022-JPが無難
- 問題点
  - 中国語や韓国語と日本語を混在できない
  - 日本語でも表現できない文字がある（丸囲み文字など）
- UTF-8
  - 様々な文字が混在できる
  - ただし一部のメールソフトは対応していない

# HTML

- ウェブ上の文書を記述するためのマークアップ言語
- metaタグ
  - 文字コードを指定できる

リスト6.1　meta要素

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html lang="ja">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=EUC-JP">
  <title>なになに株式会社</title>
 </head>
 <body>
  <p>なになに株式会社のホームページです。</p>
 </body>
</html>
```

- lang属性
  - ページで使われている言語名を明示できる
- ブラウザによっては統合漢字を描画し分ける
  - 骨 (U+9AA8)

リスト6.2　lang属性（hone.html）

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
 <html>
  <head>
   <meta http-equiv="Content-Type" content="text/html; charset=
   <title>骨</title>
  </head>
  <body>
   <ul>
    <li lang="ja">骨</li>      <!-- jaは日本語 -->
    <li lang="zh-CN">骨</li>   <!-- zh-CNは中国語（中国） -->
    <li lang="zh-TW">骨</li>   <!-- zh-TWは中国語（台湾） -->
    <li lang="ko">骨</li>      <!-- koは韓国語 -->
   </ul>
  </body>
 </html>
```

図6.3　「骨」が各国・地域の字体で表示される例

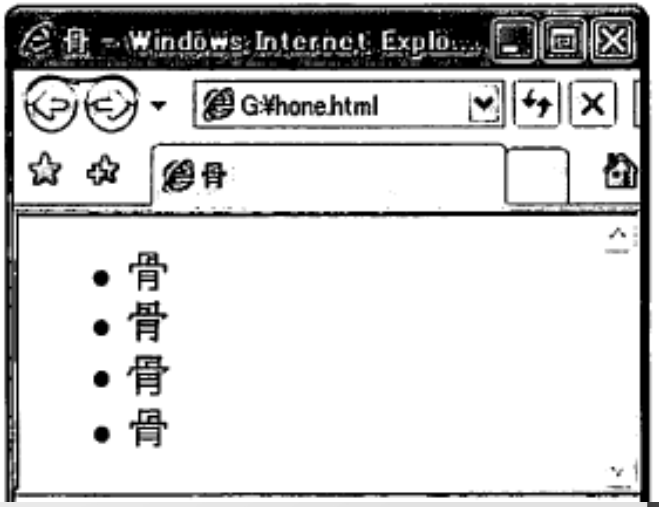



図6.2　ISO/IEC 10646における「骨」の各国・地域字形例※

| | | | | | |
|---|---|---|---|---|---|
| 154/168 | 骨 | 骨 | 骨 | 骨 | 骨 |
| 9AA8 | 0-3947<br>0-2539 | 1-586C<br>1-5676 | 0-397C<br>0-2592 | 0-4D69<br>0-4573 | 1-6C51<br>1-7649 |

※ 出典：ISO/IEC 10646:2003(E) より。

# XML

- 符号化宣言できる
  - <?xml version=”1.1” encoding="UTF-8"?>

# URL

- URL符号化 (パーセント符号化)
  - 非ASCIIをASCIIで表現する
  - 記号%の後に2桁の16進数を書く記法
  - どの文字コードとして解釈するかは決まっていない

- 例
  - http://example.com/page/%B6%E2%C2%F4

# HTTP

- Web サーバとクライアントのやり取りに使われるプロトコル

- HTML文書内部の文字コード指定の問題点
  - 復号するファイルの中に復号に必要な文字コードが宣言してある
  - ファイルの文字コードを変換すると，ファイル内部の文字コード宣言が嘘になる
- HTTPもMIME同様宣言できる

  HTTP/1.1 2S9 OK

  Date: Tue, 97 Jul 2889 14:42:85 GMT

  last-Modified: Tue, 87 Jul 2889 14:36:85 GMT

  Accept-Ranges: bytes

  Content-Length: 58137

  Content-Type: text/html; charset=utf-8

# 国際化ドメイン名(IDN)

- Internationalized Domain Name
  - 2003年頃標準化. 現行ブラウザは全て対応
  - Punycode (RFC 3492)でUnicode文字列を符号化する
  - 「http://例え.テスト」→「http://xn—r8jz45g.xn—zckzah」
  - 「http://кц.рф」→「http://xn--j1ay.xn--p1ai」
- 偽キリル文字
  - URLを偽装によるフィッシング詐欺が可能
  - 一部文字を含む場合はナマのPunycodeで表示するブラウザも
    - 正yahoo.com, 偽 **у**ahoo.com

# 文字コードに関するトラブル

# 文字コードに関するトラブル

- いくつかの話題をピックアップ
  - 文字化け
  - 改行コード
  - 全角･半角
  - 円記号
  - 波ダッシュ

# 文字化けの原因

- 文字コード指定が誤っている
  - "Shift_JIS"とすべきところを"SJIS"
  - "UTF-8"とすべきところを"utf8"
  - （場合によってはうまく表示できるので厄介）
- 機種依存文字を使っている
  - ISO-2022-JPのメールに丸囲み数字
  - （これも, 場合によってはうまく表示できるので厄介）

# 改行コード

- 改行コードが混在していると問題がおきることも
- 改行コード
  - LF(0x0A)
  - CRLF(0D 0A)

# 全角･半角

- 本来の意味
  - 印字上の長さの単位を示す印刷用語
  - 日本の仮名漢字混じり文の印刷に使われる活字は、1 文字が正方形の枠に収まる
  - この枠の長さが全角. 半分の長さが半角.
- 現在では
  - 表示幅はフォントによって異なる
  - 規格上も表示幅は定義していない
- 全角･半角文字を同一視するかどうかは, プログラム次第

# 円記号

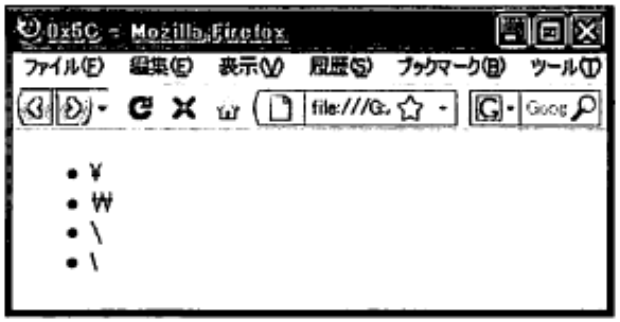
- 「通貨円記号」と「バックスラッシュ」が入れ替わる問題
- JIS X 0201
  - ASCIIの日本版
  - 0x5C の「バックスラッシュ」が「円記号」に入れ替え
- UTF-8はASCIIの上位互換
  - 0x5Cは「バックスラッシュ」

- 実際，円記号がどのように表示されるかは，フォントやプログラムに依存

リスト8.1　ブラウザのlang属性

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN">
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>0x5C</title>
  </head>
 <body>
  <ul>
   <li lang="ja">\</li> --jaは日本語
   <li lang="ko">\</li> --koは韓国語
   <li lang="zh">\</li> --zhは中国語
   <li lang="en">\</li> --enは英語
  </ul>
 </body>
</html>
```

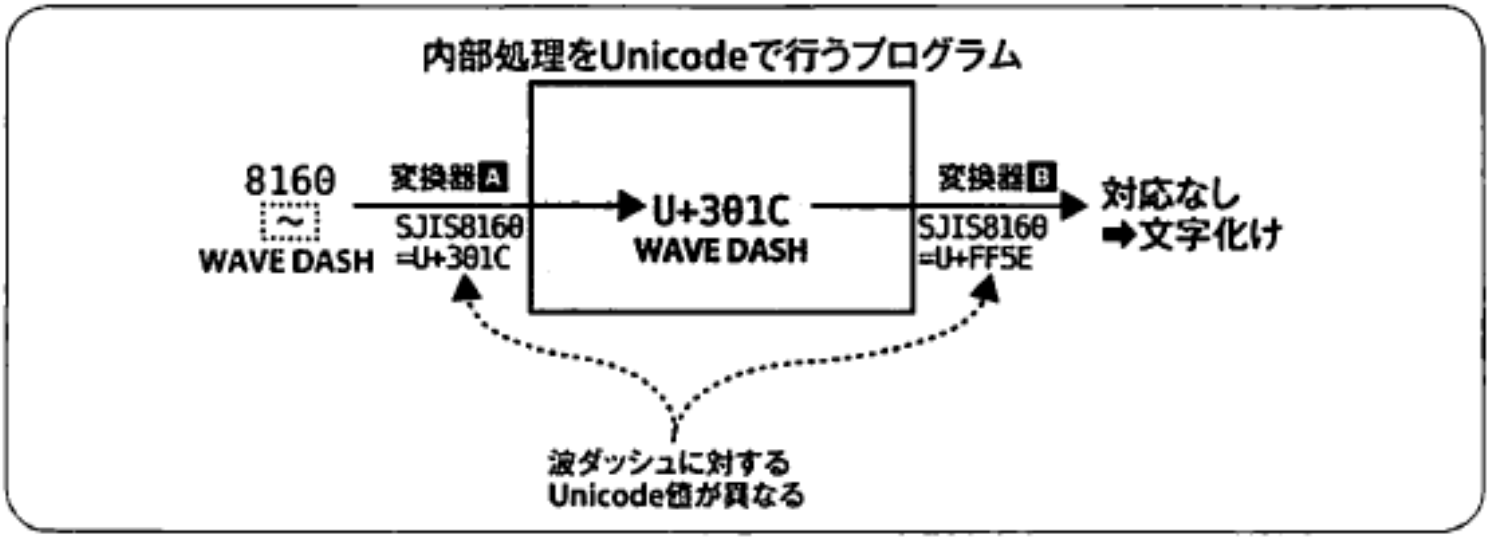
図8.11　FirefoxでUTF-8の0x5Cを表示した画面



# 波ダッシュ

- Shift_JISやEUC-JPで符号化した波ダッシュ(1区33点)
  - Unicodeに変換し、また元の文字コードに戻そうとしても，戻らない
- 2種類の変換の実装があるのが原因
  - U+301C(WAVEDASH)
  - U+FF5E (FULLWIDTH TILDE)

図8.14　変換の違いによって波ダッシュが文字化けする様子



- 本来的には U+301C(WAVEDASH) への変換が妥当
- 対策
  - Unicodeに変換しない
  - Unicodeへの変換時にどちらかに統一する
  - Unicodeの変換後で，U+301Cに統一する

# その他の文字コード変換時に問題が起きそうな文字

表8.1　ベンダ依存の変換の問題の発生する代表的な文字

| 名称 | 文字 | 区点 | 標準のUnicode文字<br>一部の実装が変換するUnicode文字 |
|---|---|---|---|
| 波ダッシュ | ～ | 1-33 | WAVE DASH(U+301C)<br>FULLWIDTH TILDE(U+FF5E) |
| 双柱 | ‖ | 1-34 | DOUBLE VERTICAL LINE(U+2016)<br>PARALLEL TO(U+2225) |
| 負符号 | − | 1-61 | MINUS SIGN(U+2212)<br>FULLWIDTH HYPHEN-MINUS(U+FF0D) |
| セント記号 | ¢ | 1-81 | CENT SIGN(U+00A2)<br>FULLWIDTH CENT SIGN(U+FFE0) |
| ポンド記号 | £ | 1-82 | POUND SIGN(U+00A3)<br>FULLWIDTH POUND SIGN(U+FFE1) |
| 否定記号 | ¬ | 2-44 | NOT SIGN(U+00AC)<br>FULLWIDTH NOT SIGN(U+FFE2) |

# 参考文献


- メールの添付ファイル名とMIME文字コードと色々メモ
  - http://adiary.blog.abk.nu/0253
- 国際化ccTLD
  - http://en.wikipedia.org/wiki/Internationalized_country_code_top-level_domain
- First New gTLDs Get the Green Light for Delegation, 2013/10/22
  - http://blog.icann.org/2013/10/first-new-gtlds-get-the-green-light-for-delegation/