i18n e web

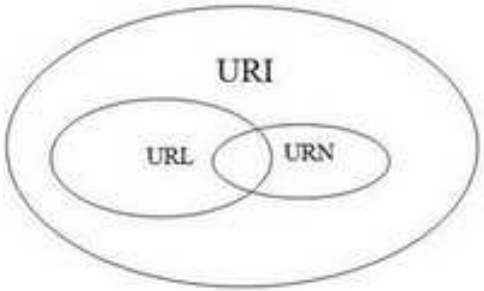
i18n: internationalization

l10n: localization

g11n: globalization

i18n nelle URL

URI, URL, URN



- RFC 1738, 2141, 2396, 2616, 2717, 2732, 3305, 3986
- URI – Uniform Resource Identifier
  - URL: Uniform Resource Locator
  - URN: Uniform Resource Name
- Uniform Resource Locator (URL)
  - Sottoinsieme di URI che identifica una risorsa attraverso la rappresentazione del meccanismo primario di accesso alla risorsa (network location)
- Uniform Resource Name (URN)
  - Sottoinsieme di URI che rimane globalmente unico e persistente anche quando la risorsa cessa di esistere o non è più disponibile
  - Gli URN non sono necessariamente “recuperabili”

Risorsa

- Qualunque cosa che abbia un'identità
- Mapping concettuale a un'entità (o insieme di entità):
  - Non necessariamente l'entità stessa
  - Non necessariamente rintracciabile via rete
  - La risorsa non deve esistere fisicamente (è concettuale)
  - Il contenuto può cambiare
- Esempi di risorse
  - File di qualunque tipo (.txt, .html, .pdf, .jpg, etc..)
  - Dispositivi (stampanti, ...)
  - Database, contenuti di database
  - Share di rete
  - Applicazioni, servizi
  - Persone, aziende
  - Libri, DVD, etc..
- Resource Identifier: un oggetto che può agire come riferimento per qualcosa che ha un identità:
  - Un nome
  - Un localizzatore
  - Entrambi

Esempi di URI

http://www.unicode.org/

https://www.google.com/search?q=unicode

mailto:manetti@aperion.it

ftp://ftp.unicode.org/Public/6.2.0/charts/CodeCharts.pdf

file:///M:/maurizio/Copyright.txt#toc

imaps://username%40gmail.com@imap.gmail.com/INBOX

urn:ietf:rfc:2648

urn:isbn:0201700522

tel:+39-055-5272864

URI: sintassi e componenti

<scheme>:<scheme-specific-part>
<scheme>://<authority><path>?<query>#<fragment>

URL in Ambito Web:

- **Scheme**: metodo per accedere alla risorsa
- **Authority**: nome host o IP addess
- **Path**: nome della risorsa definito come un percorso
- **Query**: informazioni interpretate dalla risorsa
- **Fragment**: identificazione indiretta di una risorsa secondaria tramite riferimento a una risorsa primaria

N.B.: ogni parte ha la sua sintassi

Lo schema

- Dichiara il tipo di risorsa e la modalità di accesso
- Definisce la sintassi e la semantica del resto dell'URI
- Le definizioni dei vari schemi sono negli RFC IETF
- Il registro ufficiale degli schemi:
  - http://www.iana.org/assignments/uri-schemes.html

Schemi (IANA)

aaa, aaas, about, acap, cap, cid, crid, data, dav, dict, dns, file, ftp, geo, go, gopher, h323, http, https, iax, icap, im, imap, info, ipp, iris, iris.beep, iris.xpc, iris.xpcs, iris.lwz, ldap, mailto, mid, msrp, msrps, mtqp, mupdate, news, nfs, ni, nih, nntp, opaquelocktoken, pop, pres, rtsp, service, session, shttp, sieve, sip, sips, sms, snmp, soap.beep, soap.beeps, tag, tel, telnet, tftp, thismessage, tn3270, tip, tv, urn, vemmi, ws, wss, xcon, xcon-userid, xmlrpc.beep, xmlrpc.beeps, xmpp, z39.50r, z39.50s
adiumxtra, afp, afs, aim, apt, attachment, aw, beshare, bitcoin, bolo, callto, chrome, chrome-extension, com-eventbrite-attendee, content, cvs, dlna-playsingle, dlna-playcontainer, dtn, dvb, ed2k, facetime, feed, finger, fish, gg, git, gizmoproject, gtalk, hcp, icon, ipn, irc, irc6, ircs, itms, jar, jms, keyparc, lastfm, ldaps, magnet, maps, market, message, mms, ms-help, msnim, mumble, mvn, notes, oid, palm, paparazzi, platform, proxy, psyc, query, res, resource, rmi, rsync, rtmp, secondlife, sftp, sgn, skype, smb, soldat, spotify, ssh, steam, svn, teamspeak, things, udp, unreal, ut2004, ventrilo, view-source, webcal, wtai, wyciwyg, xfire, xri, ymsgr,
fax, mailserver, modem, pack, prospero, snews, videotex, wais, z39.50

Authority (URL)

<scheme>://**<authority>**<path>?<query>#<fragment>

- authority = server | reg_name
- server = [[userinfo “@”] host [“:” port]
- host = hostname | IP address
- hostname = * (domainlabel “.”) toplabel [“.”]
- Le label:
  - consistono esclusivamente di (LDH)
    - lettere (letters)
    - cifre decimali (digits)
    - trattini (hyphens)
  - ogni label può essere al massimo 63 caratteri
  - l’hostname può essere al massimo 255 caratteri

Path, Query, Fragment

<scheme>://<authority>**<path>**?**<query>**#**<fragment>**

- path: specifico rispetto all'authority (o allo schema, in mancanza di authority) e identifica la risorsa nell'ambito dello schema e dell'authority
  - path = segment *("/" segment)
- query: una stringa di informazioni che devono essere intepretate dalla risorsa
  - query = *(pchar / "/" / "?")
- fragment: riferimento a una risorsa secondaria
  - può essere una porzione o una "vista" della risorsa primaria
  - la semantica dipende dalla risorsa primaria e dal suo media type ed è indipendente dallo schema
  - fragment = *(pchar / "/" / "?")

Caratteri nelle URI (RFC 3986)

- Sottoinsieme di ASCII
- Caratteri riservati:
  : / ? # [ ] @ ! $ & ' ( ) * + , ; =
- Caratteri non riservati:
  A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
  a b c d e f g h i j k l m n o p q r s t u v w x y z
  0 1 2 3 4 5 6 7 8 9 - _ . ~
- Uso ambiguo (sostanzialmente vietati):
  { } | \ ^ `
- Caratteri sicuramente VIETATI:
  Control codes, spazio, DEL, < > % "
- Percent encoding (%HH)
  - Case insensitive
  - Da non usare per i caratteri non riservati
  - Da usare con:
    - Caratteri riservati (con dipendenza dal contesto)
    - Caratteri vietati
    - Tutto il resto (ottetti sopra il 7F esadecimale)
  - Ottetti > 7F... Quale encoding?
  - Implementazioni non standard (%uXXXX, funzione js escape)
  - application/x-www-form-urlencoded (+ ⇄ %20)

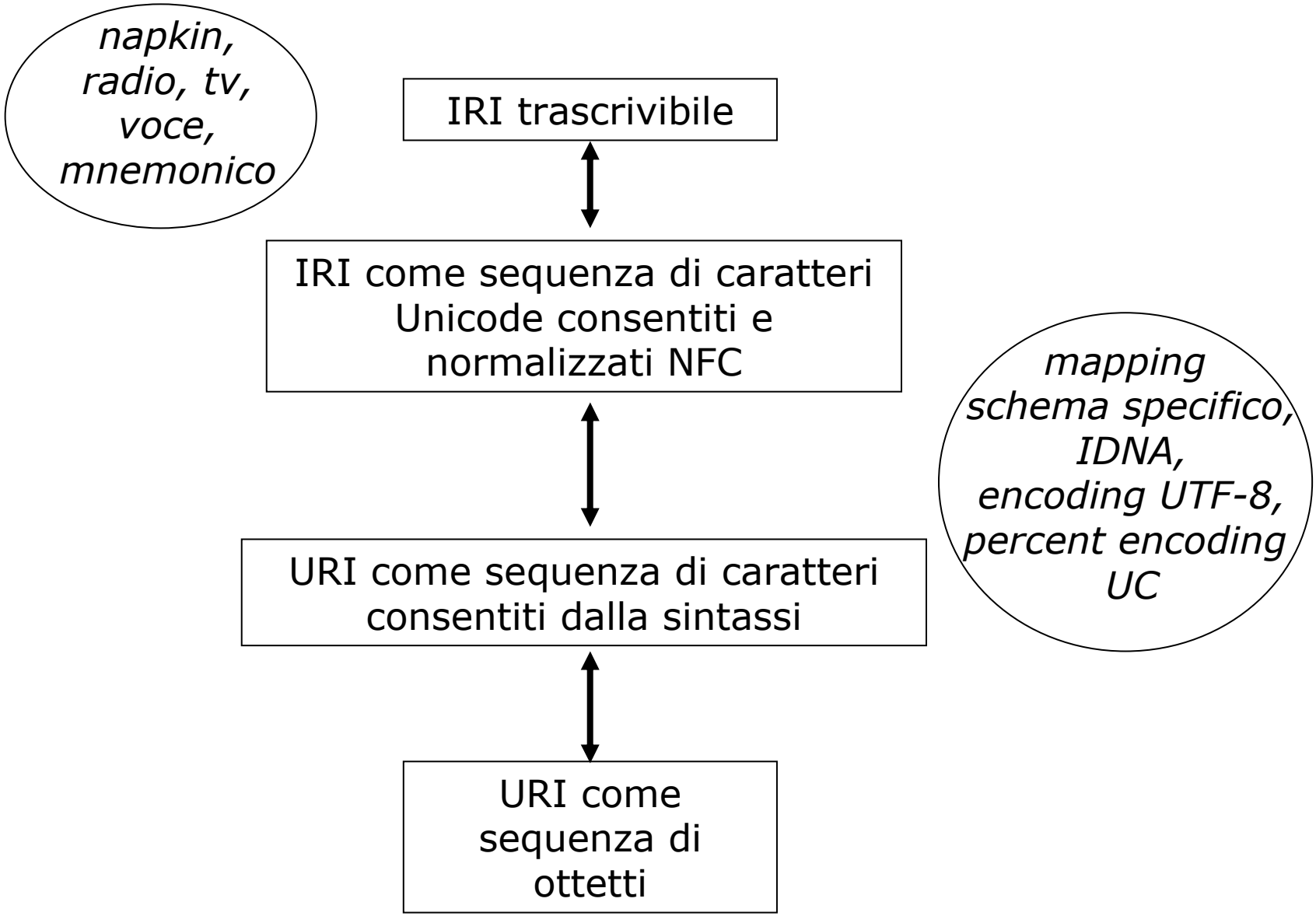
Mancanza di encoding

- http://www.google.com/search?q=caffè
  - http://www.google.com/search?q=caff%C3%A8
  - http://www.google.com/search?q=caff%E8
- http://www.aperion.it/caffè.html
  - http://www.aperion.it/caff%C3%A8.html
  - http://www.aperion.it/caff%E8.html
- http://www.caffè.it/
  - ????

IRI (RFC 3987)

- Internationalized Resource Identifiers
- Uso di caratteri non ASCII nelle URI
- Backward compatibility
  IRI ⇄ URI
- schema: solo ASCII
- authority: IDN (internazionalized domain names)
- path: Unicode, NFC, codifica UTF-8 e percent encoding
- query e fragment?? (dipende)

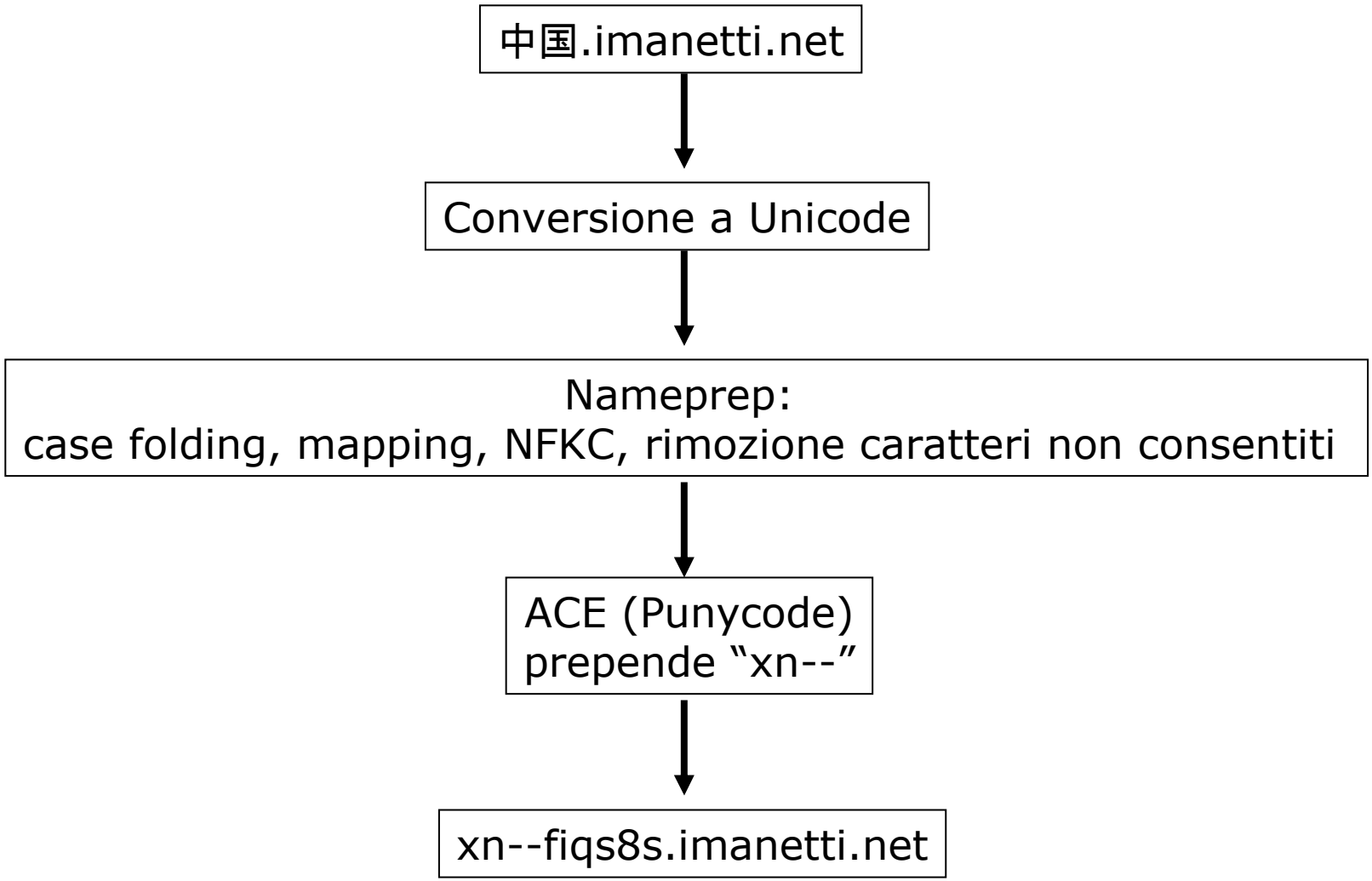
Da IRI a URI


napkin,
radio, tv,
voce,
mnemonico
IRI trascrivibile
IRI come sequenza di caratteri
Unicode consentiti e
normalizzati NFC
mapping
schema specifico,
IDNA,
encoding UTF-8,
percent encoding
UC
URI come sequenza di caratteri
consentiti dalla sintassi
URI come
sequenza di
ottetti

IDN: internationalized domain names

- I record DNS possono contenere qualsiasi ottetto, tuttavia...
- Le label utilizzate per gli hostname:
  - possono essere solo LDH (letter, digit, hyphen)
  - non possono cominciare con hyphen
  - le risposte del DNS devono trattarle case insensitive
  - “_” consentito in certe implementazioni
- IDNA introduce un livello di astrazione sopra il DNS
- Precedenti: ThayURL

IDNA


中国.imanetti.net
Conversione a Unicode
Nameprep:
case folding, mapping, NFKC, rimozione caratteri non consentiti
ACE (Punycode)
prepende “xn--”
xn--fiqs8s.imanetti.net

IDNA: problemi

- 2 versioni:
  - IDNA2003 (2003) RFC 3490, 3491, 3492, 3454
  - IDNA2008 (2010) RFC 5890, 5891, 5892, 5893, 5894
  - http://unicode.org/reports/tr46/
  - http://unicode.org/cldr/utility/idna.jsp
  - http://punycode.phlymail.de/
- IDN homograph attack
  - http://www.paypal.com/
  - Esiste anche in ASCII: G00GLE.com, PayPal.com

IDNA: ulteriori limitazioni sui caratteri

- ccTLD e TLD in generale:
  - Decide l'ICANN
  - Esempi: .中国 .香港 .台灣 .pф مصر.
- Domini di 2° livello:
  - Decide il gestore del TLD
  - Ad esempio in Italia il Nic consente i caratteri:
    à, â, ä, è, é, ê, ë, ì, î, ï, ò, ô, ö, ù, û, ü, æ, œ, ç, ÿ, ß
- Domini oltre il 2° livello:
  - Decide il detentore del dominio
    (ed eventualmente chi gestisce il DNS per quel dominio)

Visualizzazione nell'address bar

- In generale i browser:
  - mostrano caratteri ASCII xn--... al posto dell'IDN nella barra dell'indirizzo (convertono a Punycode anche in visualizzazione la parte dell'autorithy nell'IRI)
- Internet Explorer (7+) mostra il Punycode se:
  - Il dominio contiene caratteri di uno script che non è utilizzato nelle lingue incluse nelle preferenze di lingua dell'utente
  - Un label del dominio contiene un mix di caratteri di script diversi (con un'eccezione per alcuni script misti ad ASCII)
  - Il dominio contiene caratteri che non fanno parte di nessun script, es. www.I♥NY.museum (obsoleto con IDNA2008)
- Firefox
  - mostra sempre il Punycode tranne per certi whitelisted TLD http://www.mozilla.org/projects/security/tld-idn-policy-list.html
- Opera
  - non chiaro: dovrebbe funzionare come Firefox, ma poi mostra quasi sempre il testo Unicode
- Safari
  - preferenze dell'utente sugli script ammessi
- Chrome
  - preferenze utente + meccanismi di blacklisting

i18n nel path

- Da parte del browser:
  - Conversione a Unicode
  - NFC
  - Encoding UTF-8
  - Percent encoding
- Web server:
  - Il web server può riconvertire ad un altro encoding per servire la risorsa richiesta
  - IIS e Windows: nomi dei file in UTF-16, espone in UTF-8
  - Apache e Linux: i nomi dei file contengono semplicemente byte (alcuni vietati)
  - Apache e mod_rewrite, posso giocare con le richieste
- Attenzione:
  - un link con un IRI potrebbe essere scritto correttamente in un documento con codifica diversa da UTF-8 (ad es. Latin-1) benché ciò sia esplicitamente sconsigliato dal W3C
  - cliccandoci il browser deve convertire a UTF-8, effettuare il percent encoding e quindi effettuare la chiamata
  - http://www.aperion.it/caffè.html è un IRI, il cui URI è
  - http://www.aperion.it/caff%C3%A8.html
  - http://www.aperion.it/caff%E8.html è diverso ed è già un URI

Query e fragment?

- In teoria potrebbero seguire lo stesso meccanismo del path
- In realtà è meno chiaro cosa debba essere fatto: la risorsa stessa dovrebbe dettare le regole, infatti...
  - la query potrebbe essere processata da uno script che esegue ricerche su un DB Latin-1
  - il fragment è un riferimento interno alla risorsa, può dipendere dalla sua codifica
- Se non è la risorsa stessa a creare URI con query e fragment, i browser manifestano comportamenti diversi nel passaggio da IRI a URI
- N.B.:
  - la query viene inviata al server
  - il fragment è processato dal browser quando la richiesta è servita (può essere rielaborato da Javascript)

Esempio

- IRI:
  http://中国.imanetti.net/中国.php?s=पूर्ति#caffè
- URI corrispondente:
  http://xn--fiqs8s.imanetti.net/%E4%B8%AD%E5%9B%BD.php?s=%E0%A4%AA%E0%A5%82%E0%A4%B0%E0%A5%8D%E0%A4%A4%E0%A4%BF#caff%C3%A8
- Come costruisco questo link nella mia pagina?
  <a href="...???">
- URI sempre ammesso, HTML5: UTF-8 o UTF-16 per scriverlo come IRI, altrimenti *dovrei* inserire l'URI corrispondente
- Se creo il link all'IRI...
  - Quando faccio "copia e incolla" del testo dovrei ottenere l'IRI
  - Quando faccio "copia collegamento" dovrei ottenere l'URI

Copia e incolla collegamento...

- Firefox
  http://xn--fiqs8s.imanetti.net/%E4%B8%AD%E5%9B%BD.php?s=%E0%A4%AA%E0%A5%82%E0%A4%B0%E0%A5%8D%E0%A4%A4%E0%A4%BF#caff%C3%A8
- Chrome
  http://xn--fiqs8s.imanetti.net/%E4%B8%AD%E5%9B%BD.php?s=%E0%A4%AA%E0%A5%82%E0%A4%B0%E0%A5%8D%E0%A4%A4%E0%A4%BF#caffè
- Internet Explorer
  http://中国.imanetti.net/中国.php?s=पूर्ति#caffè
- Opera
  http://中国.imanetti.net/中国.php?s=%E0%A4%AA%E0%A5%82%E0%A4%B0%E0%A5%8D%E0%A4%A4%E0%A4%BF#caffè

Charset in HTTP, HTML, XML, CSS

HTTP

- Mime Content-type nelle intestazioni

```
200 OK HTTP/1.1
Content-Type: text/html;charset=UTF-8
---Blank line
Corpo della risposta
```

- Apache / IIS (configurazione base), .htaccess
- Apache mod_mime: AddDefaultCharset, AddCharset, AddType
- PHP: header("Content-type: text/html; charset=UTF-8")
- Altri linguaggi (CGI, Asp, moduli Apache) possono riscrivere le intestazioni: il web server fa il merge con quelle di default e privilegia quelle impostate dal programma

HTML, XML, XHTML, CSS

- HTML 4: non esiste un default
  `<META HTTP-EQUIV="Content-type" CONTENT="text/html; charset=UTF-8">`
- XML: default UTF-8
  `<?xml version="1.0" encoding="UTF-8"?>`
  - Dichiarazione alternativa con il BOM (UTF-8 e UTF-16)
  - Se UTF-16 DEVE cominciare col BOM
- XHTML ??
  - Dipende
- HTML 5: **`<meta charset="UTF-8">`, BOM**
- CSS: **`@charset "UTF-8";`**
  BOM ammesso dalla versione 2.1
- http://www.w3.org/International/questions/qa-html-encoding-declarations.en

Dichiarazione del charset nel link

- HTML
  ```
  <LINK title="Arabic text" type="text/html"
  charset="ISO-8859-6" rel="alternate"
  href="arabic.html">
  <A href="http://www.unicode.org"
  charset="UTF-8"> Unicode</A>
  ```
- XML
  ```
  <?xml–stylesheet href="..." type="..."
  charset="UTF-16"?>
  ```
- Non supportato dai browser principali (tranne che per i CSS)
- Deprecato in HTML 5

Priorità di codifica HTML4

- Dall’alto verso il basso, con priorità decrescente, i browser utilizzano le seguenti regole:

- User override
- HTTP “Content-Type” charset
- <META HTTP-EQUIV=“...” ...
- Link o altra sintassi di documento esterno
- Metodi euristici

Priorità di codifica HTML5

- Dall’alto verso il basso, con priorità decrescente, i browser utilizzano le seguenti regole:

1. User override
2. HTTP “Content-Type” charset
3. BOM
4. Uno dei meta tag (DEVE essere uno solo):
   <META HTTP-EQUIV=“Content-Type” CONTENT=“...”>
   <meta charset=“UTF-8”>
5. Metodi euristici
   http://www.whatwg.org/specs/web-apps/current-work/multipage/parsing.html#determining-the-character-encoding

Priorità di codifica CSS 2.1

- Dall’alto verso il basso, con priorità decrescente, i browser utilizzano le seguenti regole:

1. HTTP “Content-Type” charset
2. BOM / @charset rule (non possono essere in conflitto)
3. Tag <LINK ...> nel documento referente
4. Charset del documento referente
5. Assumere UTF-8

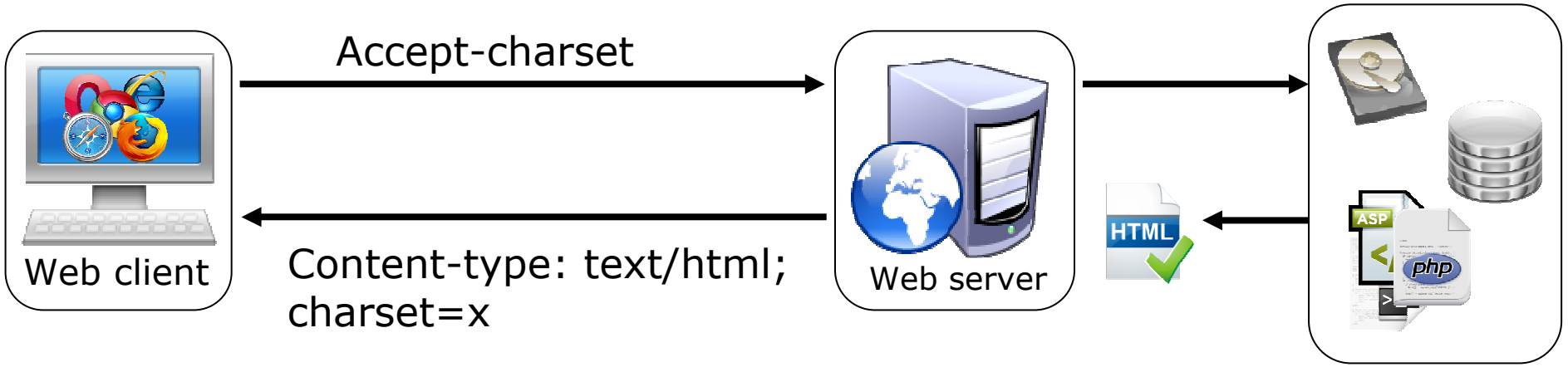
In-document declarations

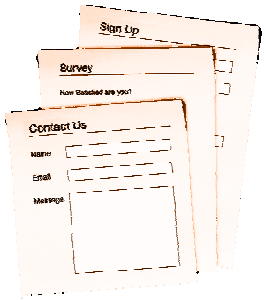
Formato	**Cosa fare**
HTML5	Usare il nuovo meta tag charset nella sezione head del documento. Deve essere entro i primi 1024 byte. <meta charset="UTF-8">
HTML5 in UTF-16	Usare il BOM. Il W3C sta ancora discutendo se è il caso di metterci anche il meta tag o no. Per il momento no.
HTML4	Usare il meta tag http-equiv come prima riga della sezione head: <meta http-equiv="Content-type" content="text/html;charset=UTF-8">
XHTML 1.x servito con MIME type text/html	Come sopra, ma se non usate UTF-8 o UTF-16 dovreste anche effettuare la dichiarazione di encoding XML
XHTML 1.x servito con MIME type application/xhtml+xml	Dichiarazione di encoding XML (BOM non necessario se UTF-8) <?xml version="1.0" encoding="UTF-8"?>

Raccomandazioni W3C in breve

- Salvare le pagine in UTF-8, quando è possibile
- Dichiarare sempre l'encoding del documento:
  - Utilizzare le intestazioni HTTP se possibile
  - Includere ANCHE la dichiarazione in-document come da tabella precedente
  - Utilizzare il "preferred MIME name" dell'encoding IANA registry
- Usare la @charset rule per fogli CSS esterni al documento nel caso in cui ci sia contenuto non-ASCII (font names, ids o class names, etc.)
- Evitare se possibile il BOM in UTF-8, e assicurasi che il codice sia in NFC (n.b.: la notazione NCR è ammessa purché faccia riferimento a caratteri NFC [Fully normalized text]).
- Evitare i codici di escape, ad eccezione dei caratteri invisibili o ambigui.
- Non utilizzare "*caratteri di controllo*" Unicode dove si può usare il markup HTML.
- http://www.w3.org/International/tutorials/tutorial-char-enc/Overview.en

Character encoding negotiation





Forms

- Il browser accetta contenuto immesso dall'utente nei form, in quale charset li invia ??
- Ci sono fondamentalmente 3 metodi di submission dei form:
  - GET
  - POST in "application/x-www-form-urlencoded"
  - POST in "multipart/form-data"

GET o POST application/x-www-form-urlencoded

- Nella GET nella parte query dell'URL...
- ...nella POST nel corpo della richiesta avremo:
- Name=Value&Name2=Value2&...
- Coppie nome=valore
- Nomi separati dai valori dal carattere =
- Coppie separate dal carattere &
- Spazi sostituiti dal carattere +
- Line breaks come CR LF percent-encoded: %0D%0A
- Caratteri non consentiti nelle URL e riservati in questo contesto (+,&,=) sono anche percent-encoded
- Ma quale codifica di carattere?

Esempio di FORM

Name: **François René Strauß** SEND

- Nel corpo della richiesta POST application/x-www-form-urlencoded o nella query della richiesta GET potremmo trovare:
  - **Name=Fran%E7ois+Ren%E9+Strau%DF**
    (Charset=ISO-8859-1)

  - **Name=Fran%C3%A7ois+Ren%C3%A9+Strau%C3%9FWeb**
    (Charset=UTF-8)

- Nella pratica i browser inviano al server i dati x-www-form-urlencoded nel charset in cui il form è stato interpretato, qualunque sia stata la scelta di questa interpretazione (HTTP header, meta tag, default, user override)

- Questa scelta non è necessariamente una soluzione valida in generale

multipart/form-data

- Content-type più efficiente del application/x-www-form-urlencoded per dati non-ASCII, file e dati binari in generale
- Non ha limiti (teorici) di lunghezza a livello client come accade invece per le URL
- Ogni coppia nome/valore è una parte mime separata, teoricamente potrebbe avere un charset diverso
- Il browser invia direttamente gli ottetti codificati nel corpo della richiesta
- Di fatto anche con questa codifica i browser NON inviano alcuna informazione sul charset utilizzato
- Esempio con il form precedente:

```
Content-Length: 399
Content-Type: multipart/form-data; boundary=----
WebKitFormBoundaryl3cH5oGYSXtKMawr

------WebKitFormBoundaryl3cH5oGYSXtKMawr
Content-Disposition: form-data; name="name"

François René Strauß
```

Risolvere le ambiguità di charset nei form

- L'attributo ACCEPT-CHARSET:
  `<form accept-charset="ISO-8859-1">`
  - Deve essere valorizzato con il preferred MIME name di un IANA charset
  - Forza il browser ad utilizzare la codifica richiesta
  - Ora standard in HTML5: supportato da tutti i browser ad eccezione di Internet Explorer (almeno fino alla versione 8 inclusa) dove però funziona se il valore è UTF-8

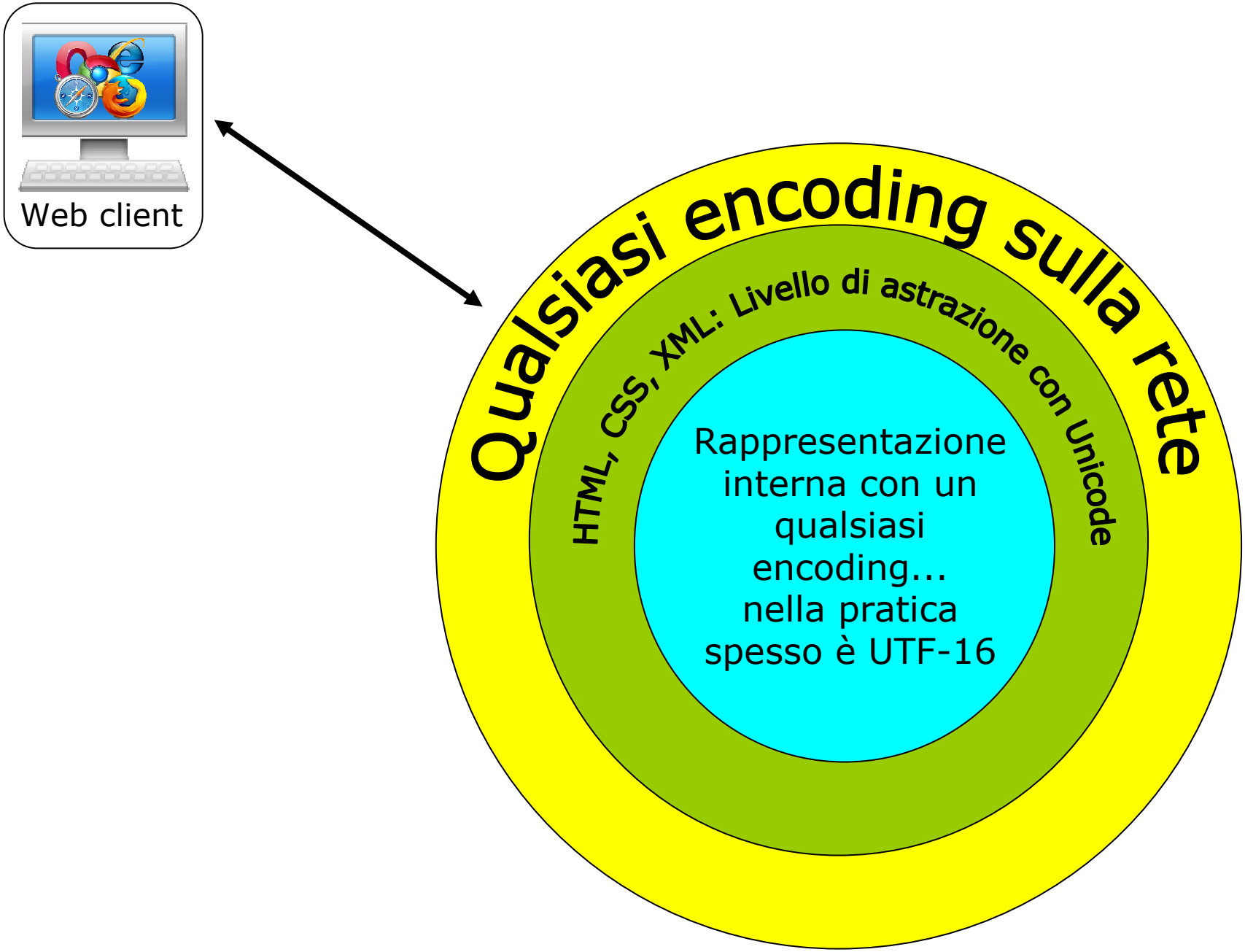
- Il campo hidden **`_charset_`**
  - Inventato dalla Microsoft
  - Viene popolato con il nome del charset utilizzato per la submission
  - Supportato da tutti i browser moderni, adesso standard HTML5

- Inventate il vostro controllo:
  - Utilizzate un campo hidden con una stringa precompilata non ASCII

- Possibili problemi:
  - Inserimento di caratteri teoricamente non consentiti dall'accept-charset o dall'encoding della pagina
  - Programmi che ricevono richieste da diversi form su siti diversi il cui charset è ignoto e/o di cui è difficile controllare la sorgente (lead.aperion.it)

Reference processing model

- Differenti codifiche richiedono metodi diversi di decodifica/parsing/processazione:
  - Encoding single byte e multibyte
  - Encoding con switching (ISO 2022)
  - Forward combining (` + e = è)
  - Backward combining (e + ` = è)
  - Logical order / Visual order (scritture RTL)
- Del resto adottare un encoding universale (ad.es. UTF-8) renderebbe obsoleti dati già esistenti
- Si usa un modello di astrazione:
  - Tutti i caratteri sono caratteri Unicode
  - Le specifiche si danno in termini di caratteri Unicode
  - Le implementazioni non devono necessariamente utilizzare caratteri Unicode, ma si comportano come se i caratteri lo fossero
- Benefici
  - Rimozione delle ambiguità, semplificazione delle specifiche
  - Consente flessibilità per l‘uso di encoding locali comunemente utilizzati
  - Backward compatibility per browser obsoleti
  - Supporto all‘internazionalizzazione
  - Indipendenza dall‘endianness


Web client
Qualsiasi encoding sulla rete
HTML, CSS, XML: Livello di astrazione con Unicode
Rappresentazione interna con un qualsiasi encoding... nella pratica spesso è UTF-16

HTML, CSS, XML: tutti i caratteri sono caratteri Unicode

- Internamente le implementazioni possono usare qualunque encoding (di fatto un qualche encoding Unicode)
- Questo significa in sostanza che *“in RAM“* tutti i caratteri che vedo nel browser sono Unicode, anche se il documento è in un encoding non Unicode (ISO-8859-1, GB2312, etc..)
- Quindi in un documento HTML posso inserire qualunque carattere Unicode (ammesso che abbia i font per vederlo) indipendentemente dalla codifica utilizzata per quel documento
- Ma non posso utilizzare una sequenza di byte se l‘encoding non prevede quel carattere
- Per esempio in un documento HTML ISO-8859-1 non posso scrivere direttamente 中 (in teoria neanche €)
- Come faccio?

Character escaping

- NCR (numeric character references)
  - HTML e XML
    - Decimale: &#dddd; **è** = è
    - Esadecimale: &#xhhhh; **è** = è
    - La "x“ deve essere minuscola nella notazione esadecimale in XML
  - CSS
    - \hh spazio finale, oppure \hhhhhh (6 cifre esadecimali) \0000E8
- Named character reference (character entities): HTML
  - Definizione esplicita (case sensitive) nel DTD
  - **è** = è
  - HTML 4 definisce 252 named entities
  - HTML 5 definisce 2231 named entities (working draft)
- 5 Predefined entities (XML):
  - **"** = "
  - **&** = &
  - **'** = '
  - **<** = <
  - **>** = >

Utilità dell’escaping

- Aggirare le limitazioni di charset
- Utilizzare caratteri propri della sintassi del markup: < > & "
- Eliminare ambiguità visive:
- Usa ASCII per fare riferimento ai code point e non agli encoding (stesso valore sempre)
- Semplifica la transcodifica (per i browser)
- Aggirare i problemi di charset:
  - p.e.: HTML in ISO-8859-1, CSS in UTF-8, nomi di classe con lettere accentate

```
<p class="caff&#xe8;">
p.caff\e8 { color: maroon;}
```

- N.b.: non si può usare negli script e negli style embedded

Altri aspetti i18n e Web

Language information: perché?

- Charset: è sostanzialmente “ortogonale” alla lingua
- Classificazione, ricerca, ordinamento
- Sillabazione, quoting, spazi e legature
- Sintesi vocale
- Ambiguità di glifo (*undo* della unificazione Han)
- SEO (??)

I tag di lingua

- HTML5: TUTTI i tag adesso supportano gli attributi:
  - **`dir`** (direzione)
  - **`lang`** (lingua)
- **`lang`** adesso può avere un valore vuoto ad indicare che la lingua *principale* è ignota
- **`xml:lang`** è supportato, a patto che abbia lo stesso valore di **`lang`**
- **`hreflang`** applicabile ai tag a, link ed area

Language identification

- Identificativi di lingua definiti dall’RFC 3066
- Codice ISO-639 di 2 lettere o 3 lettere, seguito opzionalmente dal country code di 2 lettere ISO-3166 separati dal carattere – (e non _)
- Case insensitive
- L’attributo di lingua è ereditato dagli elementi figli

Language identification

- RFC 3066 non copre tutte le necessità
- p.es. Spagnolo dell'America Latina, distinzione di Script–Addressed
- Non c'è una chiara distinzione tra l'identificazione di una "lingua" e di un "locale"
- BCP47: RFC 4646, 4647, 5646
  - language-country diventa language-script-country
  - considera molti altri aspetti

Language identification

- HTTP: intestazione **`Content-Language`**
- HTML4: attributo **`lang`** (p.es. in **`<html>`**)
- XML: attributo **`xml:lang`**
- XHTML 1.0 e HTML5: sia **`lang`** che **`xml:lang`**

```
<p xml:lang="la" lang="la">Verba volant, scripta manent.
</p>
```

- XHTML 1.1: **`xml:lang`**

Utilizzo di lang / xml:lang

- Funzione lang() di XPath
- Pseudo-classe in CSS
  `*:lang(zh) {font-family: SimSun}`
- Selettore di attributo CSS
  `*[lang|=fr] {font-weight: bold}`
- Quoting specifico per lingua:

```
q:before { content: open-quote; }
q:after { content: close-quote; }
blockquote:before { content: open-quote; }
blockquote:after { content: close-quote; }
[lang|='en'] > * { /* English */quotes: "\201C" "\201D" }
[lang|='fr'] > * { /*guillemets*/quotes: "\AB\A0" "\A0\BB" }
[lang|='fi'] > *{/*same direction*/ quotes: "\201D" "\201D" }
[lang|='de'] > * { /* German */quotes: "\201E" "\201C" }
[lang|='ja'] > * { /* Japanese */quotes: "\300C" "\300D" }
[lang|='nl'] > * { /* Dutch */quotes: "\2018" "\2019" }
[lang|='pl'] > * { /* Polish */quotes: "\201E" "\201D" }
```

```
<p lang="en">English text with <q>English quoted text</q>.</p>
 <p lang="fr">Text en Français avec <q>English quoted text</q>.</p>
 <p lang="fr">Text en Français avec <q lang="en">English quoted text
  containing a <q>quote</q> itself</q>.</p>
 <p lang="fi"><q>Quotes</q> in Finnish.</p>
 <p lang="pl"><q>Quotes</q> in Polish.</p>
 <p lang="ja"><q>Quotes</q> in Japanese.</p>
 <p lang="de"><q>Quotes</q> in German.</p>
 <p lang="nl"><q>Quotes</q> in Dutch.</p>
 <blockquote lang="fr">A paragraph using blockquote.</blockquote>
```

English text with “English quoted text”.

Text en Français avec « English quoted text ».

Text en Français avec « English quoted text containing a “quote” itself ».

”Quotes” in Finnish.

„Quotes” in Polish.

「Quotes」in Japanese.

„Quotes“ in German.

‘Quotes’ in Dutch.

> “A paragraph using blockquote.”

Altre dipendenze dalla lingua

- CSS text-transform (uppercase, lowercase, capitalize, none, inherit)
- CSS 2 consente ai browser di ignorare la regola per i caratteri non Latin-1 e per altri casi particolari
- CSS 3 forza il rispetto delle regole Unicode di casing (lingua specifiche)
  Casi speciali:
  - ß tedesca : SS
  - i turca: İ

Bi-di in HTML

- Sarebbe possibile utilizzare direttamente il testo Unicode e le regole inerenti i caratteri
- Tuttavia a causa delle ambiguità e per il fatto che HTML è un linguaggio strutturato è preferibile utilizzare
  - l'attributo **`dir`**
  - l'elemento **`<BDO>`**
- L'attributo dir può valere ltr (default) o rtl
- Ha effetto sul valore di default dell'allineamento
- Elementi figli lo ereditano
- L'elemento BDO effettua l'override delle proprietà direzionali implicite del contenuto
- Richiede l'attributo dir

Bi-di in CSS

- Esistono le proprietà (da usare insieme):
  - **`direction: { ltr | rtl } ;`**
  - **`unicode-bidi: { bidi-override | embed | normal | inherit } ;`**
- direction ha lo stesso significato dell’attributo dir di HTML
- unicode-bidi specifica il comportamento degli elementi in linea (max 15 livelli di annidamento)
- Mal supportato in Internet Explorer fino alla versione 7 inclusa
- N.B.: il markup HTML è sufficiente e raccomandato
- http://www.w3.org/International/tutorials/bidi-xhtml/

Altre aspetti di i18n e web (da approfondire)

- descrittore unicode-range nella dichiarazione @font-face
- Numbered list in stile (ebraico, georgiano, armeno, cinese, katakana, etc...)
- Testo verticale ltr, rtl (mongolo, cinese)
- Testo bustrofedico
- Annotazioni Ruby (giapponese)
- Sorting XSL
- What else??