

# HTML, il linguaggio del web

Corso di Web Design

Fabio Pittarello, Università Ca' Foscari Venezia - DAIS pitt@unive.it

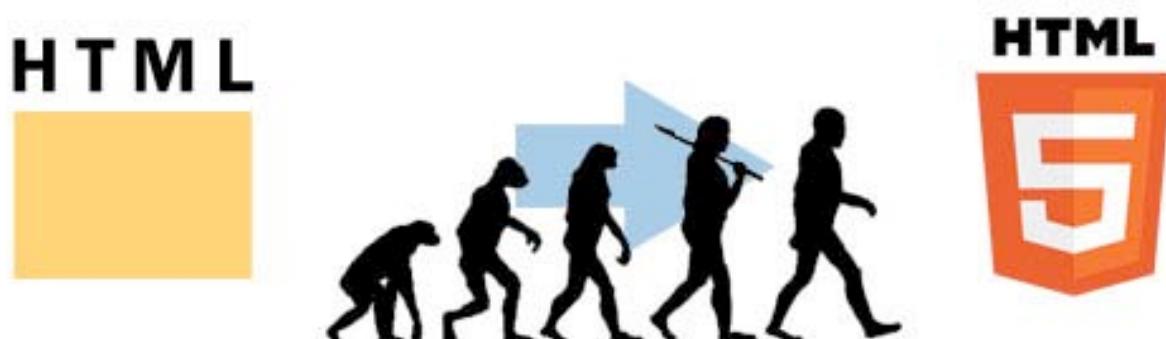
**Nota:** il materiale contenuto in questo documento è disponibile solo per uso interno nell'ambito del corso di Web Design.

## HTML

- ◆ **HTML (Hyper Text Markup Language)** è la lingua franca per pubblicare documenti ipertestuali sul World Wide Web. E' un formato non proprietario basato su SGML e può essere creato e manipolato da una vasta gamma di tool, da semplici editor di testo a sofisticati tool di authoring WYSIWYG (What You See Is What You Get).
- ◆ **SGML (Standard Generalized Markup Language)** è un vasto sistema di regole per definire linguaggi basati su marcatori. SGML stabilisce il sistema di descrivere documenti in termini di struttura, indipendentemente dall'apparenza. La modalità di visualizzazione viene stabilita successivamente, a seconda dell'uso.
- ◆ E' conveniente per una casa editrice conservare versioni SGML dei propri documenti in modo da poterli facilmente utilizzare per una varietà di usi (ad es. un libro o una pubblicazione multimediale)

# HTML

- ◆ HTML è un esempio di linguaggio basato su marcatori, che utilizza solo un sottoinsieme delle caratteristiche di SGML.  
Tutti gli elementi della pagina sono identificati da marcatori speciali (**tag**) che danno al browser istruzioni sul ruolo dell'elemento e sulla modalità di visualizzazione del contenuto.
- ◆ HTML, applicazione del sistema SGML, **conserva nella sua formulazione originaria il principio di mantenere l'informazione relativa allo stile separata dalla struttura del documento**. Nel corso degli anni questo principio è stato compromesso dalla creazione di tag proprietari che contengono esplicite definizioni di stile (es. <font>). La definizione dei fogli di stile (CSS) promette di riportare HTML alla chiarezza della formulazione iniziale.



## L'evoluzione di HTML

• **HTML**, sviluppato originariamente da Tim Berners-Lee al CERN, e reso popolare dal browser Mosaic browser sviluppato al NCSA.

• **HTML 2.0** (Novembre 1995), sviluppato dall'HTML Working Group di Internet Engineering Task Force (IETF). Stabilisce come standard caratteristiche usate nella pratica dell'anno 1994.

• **HTML 3.2** (Gennaio 1997). La prima Raccomandazione W3C per HTML. HTML 3.2 considera caratteristiche già adottate da molti browsers, come le tabelle, gli applet, il testo intorno alle immagini. Inoltre la Raccomandazione è compatibile con lo Standard 2.0.

• **HTML 4.0** (Dicembre 1997), oltre a considerare il testo, gli elementi multimediali e le caratteristiche ipertestuali delle versioni precedenti supporta più opzioni multimediali, linguaggi di scripting, fogli di stile, capacità di stampa e l'accessibilità per gli utenti disabili. HTML 4.0 inoltre è proteso verso l'internazionalizzazione dei documenti, allo scopo di rendere il web effettivamente universale.

• **HTML 4.01** è una revisione della Raccomandazione HTML 4.0. La revisione corregge alcuni errori minori.

**SGML**



**HTML**



**HTML 2.0**



**HTML 3.2**



**HTML4.01**

• **XHTML 1.0** (e non HTML 5) rappresenta la sucessiva Raccomandazione di W3C per redarre documenti documenti ipertestuali per il web.

• La specifica di XHTML 1.0 è basata su HTML 4.01 per quanto riguarda il significato degli elementi XHTML e dei loro attributi.

• **XHTML Basic** è la seconda Raccomandazione nell'ambito delle specifiche XHTML. E' stata disegnata per browser web per palmari e cellulari che non supportano il set completo di caratteristiche di XHTML

• **Modularizzazione di XHTML** è la terza Raccomandazione nell'ambito delle specifiche XHTML. Elementi e attributi vengono distinti in gruppi per facilitare l'uso di documenti che combinano XHTML con altri linguaggi basati su marcatori

**SGML**



**HTML**



**HTML 2.0**



**HTML 3.2**



**HTML 4.01**

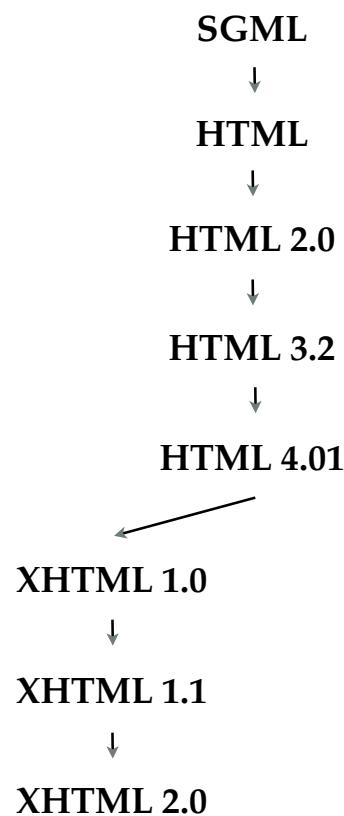
**XHTML 1.0**



# Un linguaggio al bivio

- **XHTML 1.1** è definito da una Raccomandazione basata sul framework di modularizzazione citato precedentemente. E' essenzialmente una riformulazione di XHTML 1.0 Strict attraverso l'uso di moduli XHTML; in questa riformulazione non hanno spazio i tag sconsigliati nella versione precedente di XHTML.

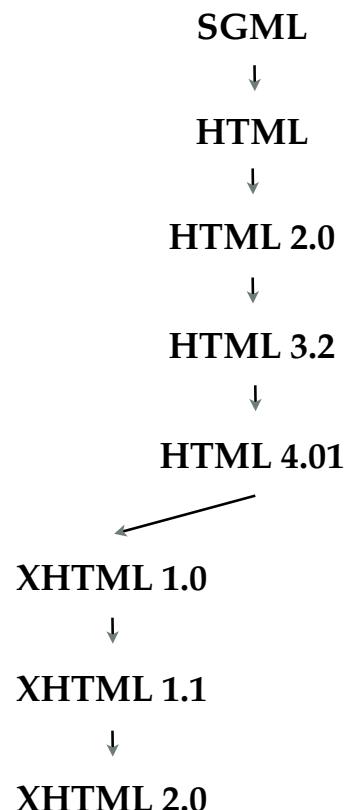
- **XHTML Mobile Profile 1.0** è basato sulla versione Basic di XHTML, sviluppata da Nokia per aggiungere all'XHTML Basic funzionalità dei terminali telefonici. E' il linguaggio di markup ufficiale di **WAP 2.0**, che ha sostituito WML di WAP 1.0. La specifica viene mantenuta dalla Open Mobile Alliance.



- Una differenza importante con XHTML 1.1 è che, mentre XHTML 1.0 era HTML riformulato come XML, **XHTML 1.1 è vero XML** e non può essere servito dal server con un mime-type text/html.

- Questo significò che **i browser più popolari al momento dell'introduzione di XHTML 1.1 non riuscirono a renderizzare i documenti scritti con questo standard.**

- **XHTML 2 proseguì su questa strada**, perdendo la compatibilità con le precedenti versioni di HTML.



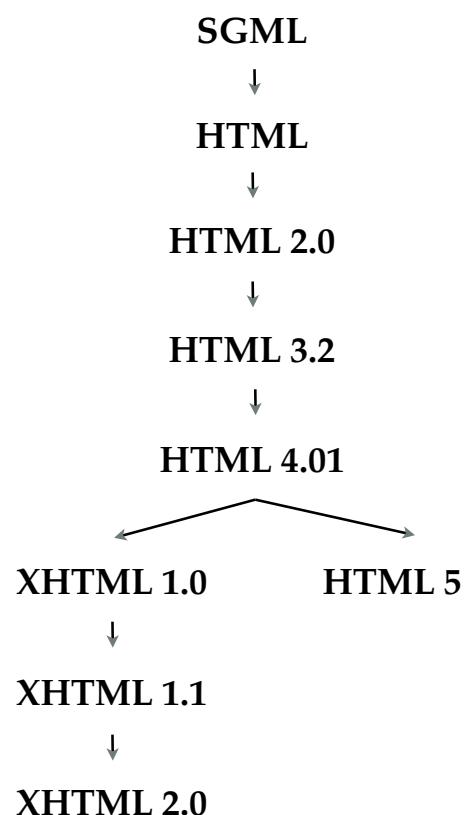
- Nell'Ottobre 2006 Tim Berners Lee ammette che il tentativo di spostarsi da HTML a XML non funziona e pochi mesi più tardi il W3C istituisce un gruppo di lavoro per HTML. Il lavoro svolto da WHATWG viene utilizzato come base per lo sviluppo di HTML.

- Nel 2009 il W3C annuncia **l'interruzione dei lavori per XHTML 2**

- Attualmente ci sono due gruppi che lavorano su HTML 5:

- WHATWG crea una specifica utilizzando il processo di proposta e di revisione/discussione successiva (**commit then review**)

- W3C prende la specifica risultante e la inserisce in un processo di revisione/discussione per arrivare alla proposta finale (**review then commit**)



# Riassumendo ...

## ◆ XHTML 2.0

- ◆ Evoluzione di XHTML 1.1
- ◆ XML puro
- ◆ Non provvisto di funzioni importanti per lo sviluppo di applicazioni
- ◆ Estensibile
- ◆ W3C non ha rinnovato il mandato al gruppo di lavoro

## ◆ HTML 5

- ◆ Compatibile con standard precedenti
- ◆ Estensioni per lo sviluppo di applicazioni
- ◆ Non ha meccanismi di estensione
- ◆ Sviluppato *contemporaneamente* da WHATWG e W3C

# I prossimi passi di HTML 5

- ◆ Candidate Recommendation prevista nel 2012
- ◆ Specifica definitiva nel 2014
- ◆ Ma questo non significa che non si possa già utilizzare ...



## I tag HTML

## Tag X/HTML

- ◆ Nota: la sintassi di un documento XHTML è sostanzialmente la stessa di un documento HTML, ma viene richiesto un maggior rigore, come una maggiore attenzione nell'identificazione dei tag e dei loro contenuti, l'utilizzo di caratteri minuscoli per l'identificazione di tag a attributi, l'uso di doppi apici per denotare i valori degli attributi.

# Tag X/HTML

- ◆ Un tag X/HTML è racchiuso tra i segni < e >. Il tag è formato dal **nome dell'elemento** seguito da una **lista opzionale di attributi**. Tutto quello che viene scritto all'interno di < > non viene mostrato dal browser.
- ◆ Distinguiamo due tipologie di tag
  - ◆ **Elementi non vuoti:** <elemento>texto</elemento>
  - ◆ **Elementi vuoti:** <elemento />
- ◆ La maggior parte degli elementi HTML sono contenitori formati da una coppia di tag (tag iniziale e tag finale) che racchiudono una stringa di testo. Il tag finale contiene lo stesso nome dell'elemento, preceduto da una barra /).  
Es. Il tempo è <em> meraviglioso </em> oggi

# Tag X/HTML

- ◆ Lo standard XHTML richiede SEMPRE il tag finale, opzionale in HTML (in HTML veniva lasciato al browser il compito di stabilire dove finisce il tag, generando in alcuni casi problemi di interpretazione).
- ◆ Nel caso in cui il contenuto non sia presente, per gli elementi non vuoti, per compatibilità con i browser HTML, va utilizzata la forma con tag iniziale e finale e non la forma minimizzata generalmente permessa da XML (dunque in XHTML va usata la forma <p></p> e non <p />).

# Tag X/HTML

- ◆ **Elementi vuoti:** <elemento />
- ◆ Alcuni (pochi) tag non hanno l'elemento di conclusione; vengono utilizzati per inserire elementi singoli nella pagina (es. immagini). Va inserito (obbligatoriamente per XHTML) il carattere / prima della fine del tag.
- ◆ Per compatibilità con i vecchi browser HTML è opportuno inserire una spaziatura prima del carattere / (per esempio, <br />, <hr /> e ).
- ◆ Questa sintassi minimizzata per gli elementi vuoti è da preferire alla sintassi alternativa <br></br> permessa da XML, che dà risultati imprevisti con molti browser esistenti.

# Nidificazione di Tag

- ◆ Un elemento HTML può essere contenuto all'interno di un altro elemento HTML. Questa tecnica viene chiamata **nidificazione**.
- ◆ Es. il tempo è <em> <strong> meraviglioso </ strong> </ em> oggi  
Non è necessario che i tag nidificati siano adiacenti
- ◆ Vanno evitati i tag sovrapposti, che possono condurre a situazioni di errore
- ◆ Es. il tempo è <strong> <em> meraviglioso </ strong> </ em> oggi
- ◆ In termini più formali, si definiscono i **documenti XHTML** come **ben formati** (ben formato è un concetto introdotto da XML) **quando tutti gli elementi hanno il tag di chiusura** [o devono essere scritti in una forma speciale] e **sono annidati correttamente**.

# Attributi di Tag X/HTML

- ◆ Gli attributi vengono aggiunti per estendere o modificare le azioni di un tag.
- ◆ L'attributo va inserito sempre nel tag iniziale.
- ◆ E' possibile inserire molteplici attributi nell'ambito di un singolo tag, separati da uno spazio; l'ordine degli attributi non è importante.
- ◆ XHTML richiede che tutti i tag e attributi siano scritti utilizzando lettere minuscole e che i valori degli attributi siano racchiusi tra doppie virgole (In HTML 4.01 in alcuni casi si potevano omettere, ad esempio per indicare valori numerici)
- ◆ I valori degli attributi possono invece essere case-sensitive (vedi slide successive).
- ◆ Sintassi per un tag contenitore con attributi  
`<elemento attributo="valore">testo</elemento>`
- ◆ Sintassi per un tag singolo con attributi  
`<elemento attributo="valore" />`

`<, > e ""`

- ◆ In alcuni casi è necessario utilizzare i caratteri `<, >` e `"` non come parte della sintassi di un tag, ma come parte del valore di un attributo.
- ◆ In questo caso è necessario utilizzare stringhe alternative, come `&quot;` per i doppi apici e `&lt;` o `&gt;` per il segno di minore o di maggiore.
- ◆ Ad es.:
  - ◆ ``
  - ◆ `<p>x &lt; y e z &gt; y</p>`

# Maiuscole e minuscole

- ◆ Come già ricordato in precedenza, in XHTML (ma non in HTML) i **nomi dei tag e degli attributi** devono essere obbligatoriamente scritti utilizzando lettere minuscole:
- ◆ Es. <IMG SRC="pinco.gif"> o 
- ◆ E' necessario prestare attenzione all'output automatico di tool che generano codice (es. versioni dattate di Dreamweaver, ImageReady, ecc.) per evitare inserimenti errati di lettere maiuscole.
- ◆ I **valori di particolari attributi** possono invece contenere anche in XHTML caratteri maiuscoli (ad es. gli URL e i nomi dei files). E' bene tuttavia ricordare che:
  - ◆ l'uso del minuscolo e del maiuscolo non è equivalente per i sistemi operativi case-sensitive come Linux (es. < img src ="pinco.gif">, < img src ="PINCO.gif"> e < img src ="pinco.GIF"> fanno riferimento a tre files diversi). Per questo motivo è necessaria una coerenza, ad esempio in questo caso tra l'indicazione del nome di una risorsa ed il riferimento ad essa;
  - ◆ a causa di questa diversità è buona abitudine uniformarsi ad uno standard comune per la descrizione dei valori degli attributi e delle risorse (ad esempio usando sempre le lettere minuscole sia per i riferimenti ai files che per i nomi delle risorse).

# Caratteri ignorati ...

- ◆ Non tutti i caratteri che si possono inserire nel codice di una pagina HTML sono significativi dal punto di vista della presentazione e vengono ignorati. Ad es. vengono ignorate/i:
  - ◆ **Interruzioni di linea**
    - ◆ Il testo e gli elementi scorrono fino a quando trovano un tag <p> o <br>
    - ◆ Le interruzioni di linea vengono comunque mostrate quando il testo è marcato <pre>
  - ◆ **Tabulatori e spazi multipli**
    - ◆ Vengono ignorati; in caso di necessità va usato il carattere &nbsp;
    - ◆ Gli spazi multipli vengono comunque mostrati quando il testo è marcato <pre>
  - ◆ **Tag <p> nidificati**
  - ◆ **Tag sconosciuti**
    - ◆ Il browser può non mostrare nulla oppure mostrare il tag come testo normale

# Struttura di un documento XHTML 1

- ◆ Un documento XHTML 1.0 è composto da 4 parti:
  - ◆ Prologo XML opzionale
  - ◆ Dichiarazione del tipo di documento <!DOCTYPE>
  - ◆ Sezione di testa <head>
  - ◆ Corpo del documento <body>
- ◆ Lo standard richiede che l'intero documento (eccettuata la dichiarazione e il prologo) compaia all'interno del tag contenitore <html>. L'eccezione a questa regola è quando il documento contiene un frameset

# Struttura di un documento XHTML 1

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
    <title>An XHTML 1.0 Strict standard template</title>
    <meta http-equiv="content-type"
      content="text/html; charset=utf-8" />
  </head>
  <body>
    <p>... Your HTML content here ...</p>
  </body>
</html>
```

# Il prologo XML

- ◆ Il prologo XML è opzionale e serve a specificare
  - ◆ la versione XML
  - ◆ Il tipo di codifica dei caratteri
- ◆ Es. <?xml version="1.0" encoding="ISO-8859-1"?>
- ◆ W3C raccomanda di utilizzare il prologo, ma alcuni browser non lo gestiscono correttamente, rendendo invisibile la pagina, visualizzandola in modo sbagliato o causando un crash del browser.
- ◆ Fortunatamente al posto del prologo è possibile inserire la codifica di carattere inserendo un elemento content-type nella sezione <head> del documento.

## Il prologo e la specifica dei caratteri

Computer codes	Encoding table	Characters
83 F9 7F 72 0B	47 slash	parenleft (
AC AA 3C 34 00	48 zero	parenright )
0E 80 00 C3 8B	49 one	asterisk *
AE CD 2F 3C 00	50 two	plus +
2A 21 5F 51 73	51 three	comma ,
C3 50 56 33 C9	52 four	minus -
15 F6 C7 20 75	53 five	period .
E6 80 F7 20 EB	54 six	slash /
03 F1 2B C6 8B	55 seven	zero 0
00 F3 A4 B0 0D	56 eight	one 1
F8 8B CF 81 E9	57 nine	two 2
83 F9 7F 72 0B	58 colon	three 3
AC AA 3C 34 00	59 semicolon	four 4
0E 80 00 C3 8B	60 less	five 5
AE CD 2F 3C 00	61 equal	six 6

- ◆ In generale in tutte le applicazioni che manipolano testi è necessario stabilire una relazione tra i singoli caratteri utilizzati e il codice numerico che li rappresenta.

# La specifica dei caratteri

- ◆ Esistono diversi standard che definiscono questa mappatura:
- ◆ La famiglia di standard **ISO-8859** raccoglie una serie di set di caratteri grafici multilingua a 8 bit per scrivere in lingue alfabetiche (**Latin-1** per mappare caratteri dell'Europa dell'Ovest, **Latin-2** per l'Europa dell'Est, **Turkish**, **Greek**, **Hebrew**, **Nordic**, ecc.). Questo standard fu creato all'inizio degli anni '80 da ECMA e approvato da ISO.
- ◆ Il più recente standard **Unicode** è una mappatura che permette di codificare tutti i caratteri utilizzati nei linguaggi scritti utilizzati nel mondo.

# La specifica dei caratteri

- ◆ Unicode è uno standard universale per mappare i caratteri utilizzati per la rappresentazione del testo. Le versioni di Unicode sono sincronizzate e pienamente compatibili con le versioni dello standard ISO/IEC 10646.
- ◆ Unicode fornisce una modalità consistente per la scrittura di testo multilingua, semplificando il lavoro di tutti coloro che usano e scambiano questi testi.
- ◆ Il design di Unicode è basato sulla semplicità e sulla consistenza tipiche dell'ASCII; ma va oltre la capacità di ASCII di codificare solo l'alfabeto Latino. **Unicode permette di codificare i caratteri utilizzati nei linguaggi scritti utilizzati nel mondo.** Ad ogni carattere vengono assegnati un nome e un numero univoci.

# La specifica dei caratteri

- ◆ Il limite più evidente dello standard **ISO-8859** era la limitata portabilità del testo scritto al di fuori di ambiti regionali specifici
- ◆ Ad esempio un testo scritto con un **sistema in uso nell'Occidente**, composto utilizzando un set di caratteri Latin 1, non era leggibile utilizzando un **sistema in uso nell'Estremo Oriente**, sul quale tipicamente erano installati font riferiti a set di caratteri diversi dal Latin 1 (e viceversa).
- ◆ A questo si aggiunse una mancanza di portabilità interregionale tra piattaforme operative diverse, dovuta alle scelte dei progettisti di sistemi operativi Microsoft ed Apple.

# La specifica dei caratteri

- ◆ Nei primi anni del web la piattaforma Windows utilizzava per la codifica dei caratteri una mappatura leggermente diversa (**Windows-1252**) rispetto a quanto previsto dallo standard ISO-8859
- ◆ Anche la piattaforma Macintosh utilizzava per il mondo occidentale una mappatura diversa (**Mac OS Roman**)
- ◆ Il risultato fu che all'interno dello stesso ambito regionale solo una parte dei caratteri scritti utilizzando una delle due piattaforme poteva essere visualizzato correttamente con l'altra piattaforma (es. le lettere alfabetiche non accentate e i numeri), mentre una parte consistente di caratteri (es. le lettere accentate) non lo era.

# CER

- ◆ Per questo motivo si prevede nella specifica dei linguaggi per il web la possibilità di utilizzare stringhe alfanumeriche, denominate **CER** o character entity references, che identificassero in maniera univoca il carattere.
- ◆ Un CER si riferisce al carattere utilizzando il formato  
`&#nnnn;` oppure `&xhhh;`  
dove nnnn e hhhh sono i riferimenti numerici (punti di codifica) ad un elemento del set di caratteri espressi in forma decimale ed esadecimale
- ◆ Ci si può riferire ad un elemento di un set di caratteri utilizzando anche la forma `&name;` dove name è il nome dell'entità

# CER

- ◆ La **specific XML** definisce **cinque entità predefinite** che rappresentano caratteri speciali:  
`'` `"` `&` `<` e `>`
- ◆ La **specific XHTML** prevede **253 entità** (incluse le cinque entità predefinite di XML); queste entità (con l'eccezione di `'`) hanno lo stesso nome e rappresentano gli stessi caratteri delle **252 entità** previste dalla specifica **HTML**
- ◆ Importante: **l'utilizzo delle CER porta allo stesso risultato, sia che il documento faccia riferimento allo standard ISO ISO-8859-1 che allo standard Unicode**, dal momento che i riferimenti numerici al set di caratteri ISO coincidono con quelli al set di caratteri Unicode (per i primi 256 caratteri)

# La specifica dei caratteri

- ◆ Le recenti versioni dei sistemi operativi (OSX per la piattaforma Macintosh, Windows NT e successivi) supportano lo standard Unicode, rendendo inutile la codifica delle lettere accentate e di altri caratteri attraverso le CER.
- ◆ Tuttavia è bene ricordare che l'utilizzo delle CER permette comunque una corretta mappatura delle lettere accentate rispetto allo standard Unicode, consentendo al tempo stesso una compatibilità verso i sistemi operativi meno recenti (la codifica delle lettere non accentate e dei numeri non rappresenta poi un problema, dal momento che la loro codifica Unicode è compatibile con lo standard ISO-8859 e il precedente standard ASCII).

## DTD - Document Type Definition

- ◆ Un documento X/HTML valido (cioè che si conforma precisamente allo standard X/HTML) deve iniziare con una dichiarazione nella quale si dice che versione di X/HTML viene utilizzata nel documento stesso.  
Ogni versione di X/HTML è definita da una distinta DTD (document type definition) che descrive in un linguaggio preciso e leggibile da computer la sintassi e la grammatica permesse nel documento.
- ◆ I documenti DTD sono disponibili on-line, ad un indirizzo stabile; ad essi ci si può riferire con due modalità: nominando il documento pubblicamente riconosciuto oppure specificando l'URL (nel caso il browser non conosca l'identificatore pubblico del documento).
- ◆ Quando il documento XHTML viene esaminato da un validatore (es. <http://validator.w3.org>) o anche da alcuni tipi di browser (es. Amaya) il contenuto viene validato rispetto al DTD indicato, generando messaggi di errore nel caso in cui vengano utilizzati tag non previsti.

# DTD - Document Type Definition

- ◆ I ricercatori di W3C, consci che ci sarebbe voluto un pò prima che i vecchi browser e le vecchie pratiche di codifica fossero dismessi e che gli autori incominciassero a marcare i documenti secondo le nuove specifiche HTML 4 e XHTML 1.
- ◆ Per questo motivo le specifiche HTML 4 e XHTML 1.0 fanno riferimento a tre diverse DTD (document type definition) che includono tre diversi sottoinsiemi di tag consentiti nel documento.

## Le DTD di XHTML

### **STRICT**

La preferita, da utilizzare con codice XHTML privo di tag o attributi di presentazione (sconsigliati), in congiunzione con CSS

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

### **TRANSITIONAL**

Da utilizzare se il documento include tags sconsigliati (es. tag di presentazione come font)

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

### **FRAMESET**

Da utilizzare se il documento utilizza, oltre ai tag previsti dalla DTD transitional, anche le frames

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

# Il tag <html>

- ◆ Il tag <html> contiene una dichiarazione di namespace.
- ◆ I cosiddetti XML namespaces forniscono un metodo per qualificare elementi ed attributi utilizzati da un documento XML, associandoli con i namespaces identificati da un riferimento URI (una locazione online stabile dove è possibile reperire un certo documento)
- ◆ In generale ci possono essere problemi di riconoscimento e di collisione per documenti che fanno riferimenti a vocabolari multipli. Per evitare questi problemi viene descritto un meccanismo che assegna nomi espansi a nomi ed attributi in modo da ottenere un'identificazione univoca.
- ◆ Es.: <html xmlns= "<http://www.w3.org/1999/xhtml>"  
xml:lang= "it" lang = "it">
- ◆ I due attributi opzionali xml:lang e lang specificano che la versione xml che si sta usando è italiana e che il documento è scritto in italiano.

# Conformità

- ◆ Un documento XHTML strettamente **conforme** deve rispettare tutti i seguenti punti:
  - ◆ L'elemento radice del documento deve essere <html>.
  - ◆ L'elemento radice del documento deve indicare lo spazio dei nomi di XHTML usando l'attributo xmlns. Lo spazio dei nomi per XHTML è definito in <http://www.w3.org/1999/xhtml>.
  - ◆ All'interno del documento ci deve essere una dichiarazione di DOCTYPE prima dell'elemento radice. L'identificatore pubblico incluso nella dichiarazione di DOCTYPE si deve riferire ad una delle tre DTD strict, transitional e frameset.
  - ◆ Deve essere validato rispetto ad una delle tre DTD.

# <head> Sezione di testa di un documento HTML

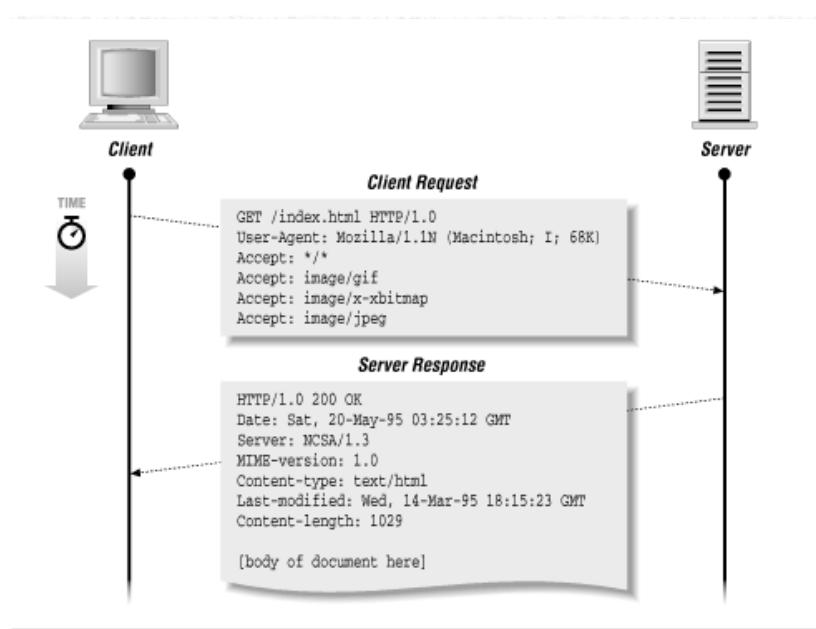
- ◆ Il tag <head> non ha attributi, serve come contenitore per altri tag utilizzati per definire i contenuti del documento:
  - ◆ <title>
    - ◆ Fornisce una descrizione dei contenuti della pagina. Tipicamente viene mostrato nella barra superiore del browser e come item nell'elenco dei bookmarks personali. I motori di ricerca lo utilizzano come elemento importante per indicizzare la pagina. Per tutte queste ragioni è bene utilizzare titoli significativi.
    - ◆ Il titolo dovrebbe contenere solo caratteri ASCII (lettere, numeri e punteggiatura di base).
    - ◆ In XHTML 1.0 è un elemento obbligatorio.
  - ◆ <base>
    - ◆ Identifica la localizzazione base sul server web, da utilizzare come riferimento per i links presenti nel documento
  - ◆ <link>
    - ◆ Definisce la relazione con un altro documento. Utilizzato molto spesso per collegare il documento ad un foglio di stile
  - ◆ <script>

## Sezione di testa: il tag <meta>

- ◆ Il tag <meta> può essere utilizzato in una grande varietà di situazioni
- ◆ I dati inclusi nel tag sono utili per server, browser e motori di ricerca, ma sono invisibili al navigatore.
- ◆ Il tag deve essere sempre inserito nella sezione <head>
- ◆ Un documento può avere un numero a piacere di tags meta.
- ◆ Ci sono due categorie di tag meta, distinte in base all'attributo associato:
  - ◆ **name**
  - ◆ **http-equiv**

# Sezione di testa: il tag <meta>

- ◆ In entrambi i casi è necessario specificare l'attributo content, per fornire un valore all'informazione
- ◆ Sintassi di base
  - ◆ <meta http-equiv="name" content="contenuto">
  - ◆ <meta name="name" content="contenuto">



## Il tag <meta> attributo **http-equiv**

L'informazione fornita da un attributo http-equiv viene processata come se venisse da un header di risposta http proveniente dal server (gli header http contengono informazioni che il server passa al browser subito prima di inviare il documento; contengono informazioni sui tipi MIME e altri valori che condizionano l'azione del browser)

Di conseguenza, l'attributo http-equiv fornisce informazioni che condizionano le modalità con le quali il browser manipola il documento

Esiste un grande numero di http-equiv predefiniti. Una lista completa è disponibile in <http://vancouver-webpages.com/META/>

# Il tag <meta> attributo http-equiv

- ◆ Alcuni esempi
- ◆ Il valore refresh può essere utilizzato per richiedere automaticamente una nuova versione del documento dopo un periodo di tempo.
  - ◆ `<meta http-equiv="refresh" content="15">`
- ◆ La tecnica può anche essere utilizzata per redirezionare i navigatori verso un nuovo URL (tuttavia la tecnica è scoraggiata da W3C in favore di redirezione server-side)
  - ◆ `<meta http-equiv="refresh" content="1; URL=mydoc.htm">`
- ◆ Il valore expires indica il tempo dopo il quale il documento deve essere considerato non più valido. Questo valore può essere utilizzato dai motori di ricerca per cancellare il documento dal proprio indice o per stabilire quando chiedere una nuova versione
  - ◆ `<meta http-equiv="expires" content="0">`

# Il tag <meta> attributo http-equiv

- ◆ L'attributo http-equiv con valore content-type, seguito dall'attributo content viene usato per specificare il set di caratteri utilizzati dal documento. La sua introduzione è parte delle misure dirette all'internazionalizzazione del web; **la disponibilità di questo tag permette di superare le difficoltà di specificare il set di caratteri nel prologo XML**
  - ◆ Es. `<meta http-equiv="content-type" content="text/html; charset=UTF-8">`
- ◆ Il valore content-language può essere utilizzato per identificare il linguaggio nel quale il documento è stato scritto.
  - ◆ Es. `<meta http-equiv="content-language" content="fr">`

# Il tag <meta> attributo name

- ❖ Questo attributo viene utilizzato per inserire informazioni riguardante il documento
- ❖ E' possibile creare propri valori per l'attributo name oppure utilizzare valori suggeriti da motori di ricerca e altre organizzazioni
- ❖ description: utilizzato per dare una breve descrizione dei contenuti della pagina web, particolarmente utile quando la pagina contiene poco testo. Alcuni motori di ricerca utilizzano solo le prime 20 parole di descrizione.
  - ❖ Es. <meta name="description" content="Il sommario del corso di web design e alcune indicazioni bibliografiche">
- ❖ keywords: parole chiave separate da una virgola, per dare info supplementari sul documento
  - ❖ Es. <meta name="keywords" content="web design, accessibilità, usabilità, CSS">
- ❖ author: identifica l'autore della pagina web
  - ❖ Es. <meta name="author" content="Fabio Pittarello">

# Il tag <meta> attributo name

- ❖ copyright: utilizzato per dare informazione di copyright sul documento
  - ❖ Es. <meta name="copyright" content="2002, Mondadori">
- ❖ robots: utilizzato per prevenire l'indicizzazione di una pagina. Sono accettabili i valori index (default), noindex (non indicizzare),nofollow (non seguire i links della pagina), none (noindex, nofollow).
  - ❖ Es. <meta name="robots" content="noindex, nofollow">
- ❖ rating: fornisce un metodo di classificazione dl contenuto. Sono disponibili 4 ratings: general, mature, restricted, 14 years
  - ❖ Es. <meta name="rating" content="general">
- ❖ generator: alcuni produttori di authoring tools aggiungono un'indicazione relativa al nome e alla versione del prodotto per verificare la penetrazione nel mercato
  - ❖ Es. <meta name="generator" content="Adobe GoLive">

# Attributi del tag <body>

- ◆ Il tag <body>, che definisce il corpo del documento, contiene le informazioni che verranno presentate all'utente sulla pagina (o perlomeno le informazioni statiche)
- ◆ Nel passato sono stati definiti un insieme di tag di presentazione, normati nella DTD Transitional, ma sconsigliati (*deprecated*) dalla specifica, dal momento che mescolano indicazioni di presentazione in un documento che dovrebbe descrivere solo contenuti.
- ◆ Oltre a questo questi attributi di presentazione agiscono a livello dell'intero documento, rappresentando i colori del testo, dei link e le caratteristiche dello sfondo, mentre utilizzando i fogli di stile è possibile definire in maniera più sofisticata queste caratteristiche.
- ◆ Nelle due slides successive vengono riportati, come memoria storica, alcuni esempi di attributi associati al tag <body>

# Attributi del tag <body>

- ◆ Attributi *deprecated* per definire i colori del testo e dei link
  - ◆ <body text="color"> colore del testo – default nero
  - ◆ <body link="color"> colore dei links – default blu
  - ◆ <body vlink="color"> colore dei links visitati – default porpora
  - ◆ <body alink="color"> colore dei links quando cliccati – default rosso
- ◆ color può essere specificato come tripletta RGB (#rrggbb) o con il nome del colore

# Attributi del tag <body>

- ◆ Attributi *deprecated* per definire lo sfondo della pagina
- ◆ <body bgcolor="color"> colore di sfondo della pagina
- ◆ <body background="nome\_immagine"> immagine di sfondo
- ◆ L'attributo background del tag body consente di aggiungere un'immagine di sfondo, che viene aggiunta al documento a partire dall'angolo superiore sinistro dell'area visibile della finestra e obbligatoriamente ripetuta sia orizzontalmente che verticalmente, nel caso in cui le dimensioni della finestra siano superiori a quelle dell'immagine
- ◆ Possono essere utilizzate immagini JPEG, GIF e PNG.



**HTML 5**

*Il linguaggio del web*

# Struttura di un documento HTML 5

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>Page Title</title>
  </head>
  <body>
  </body>
</html>
```

## HTML 5: DOCTYPE

- ◆ La dichiarazione di tipo del documento viene semplificata sostanzialmente:
  - ◆ si passa dalla verbosità della dichiarazione di HTML 4
    - ◆ `<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"`
    - ◆ `"http://www.w3.org/TR/html4/strict.dtd">`
  - ◆ alla sintesi di HTML 5
    - ◆ `<!DOCTYPE html>`

# HTML 5: Principi di design

- ◆ Approccio evolutivo:
  - ◆ viene supportato il contenuto esistente, redatto utilizzando i tag di HTML 4.01
  - ◆ non si reinventa la ruota, ma **si pavimentano i sentieri**, cioè se c'è una modalità largamente adottata dai web designers per soddisfare una certa esigenza, questa viene codificata in HTML 5
- ◆ La specifica definisce in modo esplicito come i browser devono comportarsi quando hanno a che fare con documenti non corretti (badly formed).

## HTML 5: DOCTYPE

- ◆ Gli estensori di HTML 5 motivano la semplificazione affermando che:
  - ◆ HTML 5 è pensato per supportare il contenuto esistente di HTML 4.01 o di XHTML 1.0
  - ◆ Le versioni successive di HTML dovranno supportare il contenuto di HTML 5
  - ◆ Pertanto non è necessario specificare la versione di HTML

# HTML 5: Principi di design

- ◆ I browser supportano features, non doctypes
  - ◆ il rendering del documento avverrà secondo le capacità del browser, non secondo il doctype specificato nel documento
- ◆ Le dichiarazioni di tipo sono state create per i validatori, per renderizzare il contenuto è sufficiente la dichiarazione minimale di HTML 5

# HTML 5: Semplificare

- ◆ Lo stesso spirito di semplificazione si ritrova nella dichiarazione del set di caratteri:
  - ◆ `<meta charset="UTF-8">`  
anziché  
`<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">`
- ◆ nella dichiarazione di un collegamento ad un file javascript:
  - ◆ `<script src="file.js"></script>`  
anziché  
`<script type="text/javascript" src="file.js"></script>`

# HTML 5: Semplificare

- ◆ oppure nella specifica di un link ad un file CSS
  - ◆ `<link rel="stylesheet" href="file.css">`  
anziché  
`<link rel="stylesheet" type="text/css" href="file.css">`
- ◆ Le dichiarazioni precedenti sono ridondanti perché rimarcano situazioni che sono già descritte nella versione semplificata oppure che sono evidenti della pratica (ad es. CSS in linea teorica è uno dei linguaggi per specificare il rendering della pagina, ma di fatto è l'unico)

## HTML 5: Codifica alla tua maniera ...

- ◆ Alcuni linguaggi, come **Python**, richiedono un modo preciso di scrivere il codice, in cui conta anche esempio l'uso degli spazi per indentare il codice.
- ◆ Altri linguaggi, come **Javascript**, non richiedono particolare attenzione alla formattazione
- ◆ I linguaggi di markup, come già ricordato, non considerano come significativo l'uso degli spazi; tuttavia alcuni linguaggi di markup (es. XHTML 1.0) rendono obbligatorio uno stile specifico di scrittura.

# HTML 5: Codifica alla tua maniera ...

- ◆ XHTML 1.0 rende obbligatoria la sintassi di XML:
  - ◆ tutti i tag devono essere scritti in minuscolo
  - ◆ tutti i valori di attributi devono essere tra virgolette
  - ◆ tutti gli elementi devono avere il tag di chiusura (nel caso di elementi standalone, come br, ci deve essere comunque una barra / di chiusura)
- ◆ In HTML 5 la scelta dipende da chi scrive il codice!

## Biasimato od obsoleto?

- ◆ Non ci sono elementi o attributi **biasimati** (deprecated) in HTML 5, ma ci sono molti elementi o attributi **obsoleti**
- ◆ Non è una questione di essere politicamente corretti; si vuole piuttosto continuare a mantenere compatibilità con il passato (informando anche sugli elementi sconsigliati) pur dando un giudizio di obsolescenza.
- ◆ Un documento che contiene elementi o attributi obsoleti verrà considerato non conforme.

# Biasimato od obsoleto?

- ◆ Ad esempio:
  - ◆ **frame**, **frameset** e **noframes** sono elementi obsoleti
  - ◆ **acronym** è obsoleto e dovrebbe essere sostituito con abbr
  - ◆ Elementi di presentazione come **font**, **big**, **center** e **strike** oppure attributi come **bgcolor**, **cellspacing**, **cellpadding** e **valign** sono obsoleti; devono essere sostituiti utilizzando le regole CSS

# Programma di rieducazione

- ◆ Alcuni elementi di presentazione sono stati tuttavia ridefiniti per superare i limiti passati. Ad es:
  - ◆ **small** è stato ridefinito per indicare le clausole piuttosto che per indicare il rendering a piccola dimensione
  - ◆ **b** non indica più il grassetto, ma un testo che spicca stilisticamente sul resto senza avere un'importanza addizionale (nel qual caso servirebbe il tag **strong**)
  - ◆ **i** non indica più il corsivo, ma un testo alternativo oppure uno stato d'animo senza che questo implichia un'importanza o un'enfasi (nel qual caso servirebbe il tag **em**)
  - ◆ **cite** non indica più un generico riferimento ad altre fonti informative, ma indica il titolo di un'opera

# L'elemento a potenziato

- ◆ Altri elementi sono stati potenziati, consentendo ad esempio di includere elementi multipli all'interno di un unico contenitore. Anche in questo caso si è trattato di *pavimentare i sentieri* esistenti, dal momento che i browser esistenti consentivano già di farlo, anche se non era consentito dalla specifica precedente
  - ◆ <a href="/" about">  
    <h2>About me</h2>  
    <p>Find out what makes me tick.</p>  
  </a>