

Lezione 1: (X)HTML

Matteo Vaccari

<http://matteo.vaccari.name/>
matteo.vaccari@uninsubria.it



(cc) Matteo Vaccari. Published in Italy.
Attribution – Non commercial – Share alike 2.5

HTML: HyperText Markup Language

Tim Berners-Lee e Robert Caillau al CERN nel 1991

HTML descrive la **struttura logica** di un documento

- ▶ il browser è libero di presentare i tag come preferisce
- ▶ il documento ha senso anche se si ignorano i tag

HTML combina idee preesistenti:

- ▶ ipertesti (Vannevar Bush, **As We May Think**, 1945)
- ▶ linguaggi di marcatura (SGML, 1970)

HTML è compatto, leggibile e portabile

Molti altri formati di documenti sono voluminosi:

- ▶ l'autore controlla il layout preciso
- ▶ tutti i dettagli di layout, compresi i font, sono salvati con il documento
- ▶ il documento può essere presentato solo su un medium specifico (es. foglio A4)

In confronto, HTML è magro:

- ▶ l'autore rinuncia al controllo per avere in cambio la portabilità
- ▶ HTML rappresenta solo il contenuto e la sua struttura logica

Marcatura: testo + marcatori

sorgente:

```
Questo è il <em>primo</em> esempio
```

risultato:

Questo è il *primo* esempio

em sta per **emphasis**

Elementi e tag

Questo è il $\overbrace{\langle \text{em} \rangle \text{primo} \langle / \text{em} \rangle}^{\text{elemento}}$ esempio

``: start tag

``: end tag

Elementi e tag (ii)

Gli elementi possono **racchiudere** del testo (ed altri elementi)

È il *primo* *giorno* di primavera** ...

stronger emphasis

emphasis

(È il *primo giorno di primavera*...)

Oppure possono essere *vuoti*

Con questo vado a capo.
 Ora sono su una nuova riga

(Con questo vado a capo.
Ora sono su una nuova riga)

Scheletro minimo di un documento html

```
<html>
  <head>
    <title>Titolo</title>
  </head>
  <body>
    Qui va il mio contenuto
  </body>
</html>
```

- ▶ tutto è racchiuso nell'elemento **html**
- ▶ il quale contiene l'elemento **head** e l'el. **body**
- ▶ ogni pagina deve avere un titolo

Provate! Copiate questo html in un file con estensione “.html” e visitate quel file in un browser

Intestazioni e paragrafi

Sei livelli di importanza, da **h1** a **h6**

```
<h1>Un'intestazione importante</h1>  
<h2>E una un po' meno importante</h2>
```

Ogni paragrafo dovrebbe iniziare con il tag `<p>`

```
<p>Bla bla, il primo paragrafo</p>  
<p>Bla bla, il secondo</p>
```

Html non rispetta lo spazio bianco

Link

`Tutto su Pippo`

target *label*

Posso usare url **relative**

Il `diario` delle lezioni

Lo start tag `<a>` contiene un **attributo**

- ▶ **nome**: "href"
- ▶ **valore**: la url (assoluta o relativa)

Sintassi di un tag

`< name attribute0 attribute1 ... >`

`<html lang='it'>`

``

Gli attributi specificano informazioni ausiliarie, oppure **meta**informazioni

Altri attributi che hanno senso in un link

```
<a accesskey='h' tabindex='1' title="Return to first page"  
  href='/'>Home page</a>
```

Accesskey: rende il link attivabile con *alt-h*

Tabindex: rende il link selezionabile con *tab*

Title: descrive cosa trovo se seguo il link

Tre tipi di liste (i)

lista “coi pallini” (**unordered list**)

```
<ul>  
  <li>the first list item</li>  
  <li>the second list item</li>  
  <li>the third list item</li>  
</ul>
```

- ▶ the first list item
- ▶ the second list item
- ▶ the third list item

Tre tipi di liste (ii)

lista “coi numeri” (**ordered list**)

```
<ol>  
  <li>the first list item</li>  
  <li>the second list item</li>  
  <li>the third list item</li>  
</ol>
```

1. the first list item
2. the second list item
3. the third list item

Tre tipi di liste (iii)

definition list

```
<dl>
  <dt>the first term</dt>
  <dd>its definition</dd>

  <dt>the second term</dt>
  <dd>its definition</dd>
</dl>
```

the first term

its definition

the second term

its definition

A capo e non a capo

Come forzare un “a capo”

Si usa l'elemento `
`

```
Nocturnis ego somniis<br/>  
iam captum teneo! Iam volucrem sequor<br/>  
te per gramina Martii<br/>  
campi, te per aquas, dure, volubiles!<br/>
```

Come **non** andare a capo

Il browser può andare a capo dopo ogni spazio; per impedirlo si usa il Non-Breaking Space

...la Coca Cola è una bevanda gassata...

HTML Entities

Le **entità** html sono simboli introdotti per nome, come ` `;

less than	<code>&lt;</code>	<
greater than	<code>&gt;</code>	>
ampersand	<code>&amp;</code>	&
em dash	<code>&mdash;</code>	—
euro	<code>&euro;</code>	€
a grave	<code>&agrave;</code>	à
a acute	<code>&aacute;</code>	á
a umlaut	<code>&auml;</code>	ä
:	:	:

Vedi <http://www.w3.org/MarkUp/Guide/Advanced.html> per una lista più completa

Ancora su html

I commenti non vengono mostrati dal browser

```
<!-- questo e' un commento -->
```

L'elemento `pre` è per testo `pre`formattato

```
<pre>
#include <stdio.h>
int main() {
    printf("Hello, world!\n");
    return 0;
}
</pre>
```

Inserire le immagini

```
<img src='pippo.png' alt='Effige di Pippo'  
width='120' height='100' />
```

src: url o pathname del file grafico; obbligatorio

alt: testo **alternativo**: obbligatorio

width, height: facoltativi ma fortemente consigliati

Un “bottono” è semplicemente un elemento **a** con dentro un elemento **img**

```
<a href='http://www.disney.it/pippo.html'><img  
src='pippo.png' alt='' /></a>
```

Tabella in html

Sono usate sia per presentare dati, che per realizzare layout grafici

Year	Sales
2000	€18M
2001	€25M
2002	€36M

in html:

```
<table border='1'>
<tr> <th>Year</th> <th>Sales</th> </tr>
<tr> <td>2000</td> <td>&euro; 18M</td> </tr>
<tr> <td>2001</td> <td>&euro; 25M</td> </tr>
<tr> <td>2002</td> <td>&euro; 36M</td> </tr>
</table>
```

tr table row

td table data

th table heading

Tabelle in html, cont.

Spesso sono usate non per rappresentare dati in forma tabellare, ma per **impaginare**


- ▶ testo su più colonne
- ▶ “box” di testo incorniciati
- ▶ layout della pagina
- ▶ effetti grafici di ogni tipo

Dalla struttura logica alla struttura fisica

In origine, gli elementi HTML descrivevano solo la struttura

- ▶ h2: “questa è una **intestazione** di livello 2”
- ▶ em: “questo testo deve avere **enfasi**”
- ▶ ul: “questa è una **lista**”

Ben presto, gli utenti desideravano un maggiore controllo:

- ▶ “questa intestazione è **centrata** e in **Times-Roman** dimensione **28pt**”
- ▶ “questo testo è in **corsivo**”
- ▶ “questi elementi di lista sono indentati **7mm** e usano  come pallini”

Elementi orientati al layout

Gli inventori di HTML risposero con **nuovi elementi** orientati al layout fisico

```
<center>
  <font size='2' family='Times-Roman' color='#112233'>
    Ma <i>che <blink>cosa</blink> abbiamo</i> combinato?!?
  </font>
</center>
```

Male!

Markup: semantic or presentation?

Presentation markup (Avoid!)

```
<font size='7'>Introduzione</font><br/>
Testo introduttivo<br/><br/>
<font size='6'>Secondariamente</font><br/>
Bla bla<br/>
```

Semantic markup (Good!)

```
<h1>Introduzione</h1>
<p>Testo introduttivo</p>
<h2>Secondariamente</h2>
<p>Bla bla</p>
```

Markup: semantic or presentation? (ii)

```
un punto<br />
un altro<br />
e un altro ancora<br />
```

```
<ul>
  <li>un punto</li>
  <li>un altro</li>
  <li>e un altro ancora</li>
</ul>
```

La guerra dei browser

La metà degli anni '90 vede Microsoft e Netscape combattere per il predominio dei rispettivi browser

Una parte della strategia: estensioni proprietarie a HTML: nuovi elementi, nuovi attributi

La **portabilità** venne distrutta

Ancora oggi molti siti sono realizzati in due versioni: IE e Firefox

Problemi:

- ▶ il “browser sniffing” è quasi sempre realizzato male
- ▶ il costo aumenta enormemente
- ▶ ... e tutti gli altri browser?
- ▶ il markup fisico aumenta il “peso” delle pagine

Regola d'oro

Una pagina web ben scritta funziona bene su tutti i browser
ragionevolmente recenti

“Questa pagina è ottimizzata per XYZ”

= “Questa pagina l’ho testata solo con XYZ”

“Si consiglia una risoluzione di 1024x768”

= “Il mio schermo ha una risoluzione di 1024x768”

La reazione del World-Wide Web Consortium (W3C)

Separazione struttura logica (HTML) da presentazione (Cascading Style Sheets, CSS)

Introduzione di standard: HTML 4.0, XHTML 1.0,...

Ma gli standard sono inutili se sono “lettera morta”

Verso il 2000, appaiono i primi **browser che implementano gli standard:**

- ▶ Internet Explorer 5 per Macintosh
- ▶ Internet Explorer 6 per Windows
- ▶ Opera 7
- ▶ Mozilla, Netscape 7
- ▶ Safari

Diverse versioni di HTML

Gli standard **correnti** di HTML sono:

- ▶ HTML 4.01 strict, transitional, frameset
- ▶ XHTML 1.0 strict, transitional, frameset
- ▶ XHTML 1.1

transitional: contiene elementi di layout

strict: separa nettamente contenuto da presentazione

frameset: per pagine con frame (da evitare!)


XHTML: una riformulazione di HTML in XML

Quale versione di HTML scegliere?

- ▶ HTML 4.01: consiglio di evitare: non è XML
- ▶ XHTML 1.0 strict, transitional: OK
- ▶ XHTML 1.0 frameset: evitare i frameset
- ▶ XHTML 1.1: mal supportato: evitare

Sintassi e validazione

La sintassi di (X)HTML è definita in maniera precisa e formale.

- ▶ ogni documento (X)HTML deve soddisfare questa definizione
- ▶ la qual cosa si può validare automaticamente
- ▶ i documenti validi possono fregiarsi di una patacca: 
- ▶ i documenti non validi producono una lista di messaggi di errore

Validare una singola pagina è facile - essere certi che l'output di un app web sia sempre valido è più difficile

I browser sono lassisti

La maggior parte dei documenti in realtà **non** sono validi:

- ▶ gli autori sono negligenti
- ▶ i documenti sono validati guardandoli in un browser
- ▶ HTML generato in automatico spesso non è valido

Ciononostante, molte pagine funzionano

- ▶ i browser fanno tutto il possibile
- ▶ non danno **mai** un messaggio di errore

Pessimo HTML

```
<h2>Pessimo HTML</h1>
<li><a>No, non va tanto</b> bene.
<li><i>Anzi, va abbastanza
      <g>male</g></em>
</ul>
Ma il browser si <a naem=lkjh>arrangia
```

Pessimo HTML

- ▶ No, non va tanto bene.
- ▶ *Anzi, va abbastanza male* Ma il browser si *arrangia*

Qual'è il problema?

- ▶ si promuove cattivo HTML
- ▶ browser diversi sono “astuti” in maniera diversa
- ▶ è difficile usare documenti non validi per processarli automaticamente con altri strumenti

Un appropriato DOCTYPE

Per poter validare un documento occorre **dichiarare** per quale standard è scritto

La dichiarazione **doctype** serve a questo

Ad esempio, per un documento XHTML 1.0 Strict:

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

Altri doctype: vedi <http://www.alistapart.com/articles/doctype/>

La presenza di un doctype corretto fa eseguire i browser in **standards mode** (più affidabile); altrimenti usano il **quirks mode**

Scheletro di documento XHTML

```
<!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' xml:lang='en' lang='en'>  
<head>  
  <meta http-equiv='Content-Type' content='text/html; charset=utf-8' />  
  <title>untitled</title>  
</head>  
<body>  
  Qui ci va il mio documento  
</body>  
</html>
```

Perché aderire agli standard W3C?

- ▶ Ben supportati oggi su Firefox, IE 6+, Safari, Opera
- ▶ A prova di futuro
- ▶ Markup leggero → più veloce
- ▶ Compatibili con i client più strani
- ▶ Accessibile

Leggi “99.9% of Websites Are Obsolete” di Zeldman

Un sito di ricette

<h2>Patate fritte</h2>

<p>Serve: 3 persone</p>

<p>Difficoltà: facile</p>

<h3>Ingredienti</h3>

1Kg patate

1l olio extra-vergine di oliva

<h3>Procedimento</h3>

<p>Scaldare l'olio in una padella...</p>

Un po' più di struttura

```
<h2 id='title'>Patate fritte</h2>
<p class='servings'>Serve: 3 persone</p>
<p class='difficulty'>Difficoltà: facile</p>
<h3>Ingredienti</h3>
<ul id='ingredients-list'>
  <li>1Kg patate</li>
  <li>1l olio extra-vergine di oliva</li>
</ul>
<h3 id='process'>Procedimento</h3>
<p>Scaldare l'olio in una padella...</p>
```

Ancora più struttura

```
<div class='recipe'>
  <h2 id='title'>Patate fritte</h2>
  <p id='servings'>Serve: 3 persone</p>
  <p id='difficulty'>Difficoltà: facile</p>
  <div class='ingredients'>
    <h3>Ingredienti</h3>
    <ul>
      <li>1Kg patate</li>
      <li>1l olio extra-vergine di oliva</li>
    </ul>
  </div>
  <div id='process'>
    <h3>Procedimento</h3>
    <p>Scaldare l'olio in una padella...</p>
  </div>
</div>
```

XML: marcatura ad hoc

```
<?xml version='1.0' encoding='utf-8' ?>
<recipe>
  <title>Patate fritte</title>
  <servings>Serve: 3 persone</servings>
  <difficulty>Difficoltà: facile</difficulty>
  <ingredients>
    <ingredient>1Kg patate</ingredient>
    <ingredient>1l olio extra-vergine di oliva</ingredient>
  </ingredients>
  <process>
    <p>Scaldare l'olio in una padella...</p>
  </process>
</recipe>
```

Relazione fra HTML e XML

XML ha regole più strette

XHTML è un dialetto XML che i browser riconoscono come HTML

Regole:

- ▶ Tutti gli elementi devono essere chiusi
- ▶ Tutti gli attributi devono avere un valore
- ▶ Il valore degli attributi va racchiuso fra virgolette

HTML valido ma *non XHTML*

```
<p>Primo paragrafo <p>Secondo  
paragrafo  
<option value=bar  
selected>x</option>
```

XHTML valido

```
<p>Primo paragrafo</p> <p>Secondo  
paragrafo</p>  
<option value="bar"  
selected="selected">x</option>
```

Esempio: citazione (markup non semantico)

```
<p>Eric Meyer wrote:</p>
```

```
<p>
```

```
What's so interesting to me is that the guys who decided  
to focus on the positive went out and did something;  
those who want to mix in the negative seem to have  
nothing to offer except complaints.
```

```
</p>
```

```
<p>An excellent contrast between those who want to  
build new things and those who want to tear them down.
```

```
</p>
```

Esempio: citazione (markup semantico)

```
<p><cite>Eric Meyer</cite>  
<a href="http://meyerweb/...social-protocols/">wrote</a>:</p>  
<blockquote>  
  <p>  
    What's so interesting to me is that the guys who decided  
    to focus on the positive went out and did something;  
    those who want to mix in the negative seem to have  
    nothing to offer except complaints.  
  </p>  
</blockquote>  
<p>An excellent contrast between those who want to  
build new things and those who want to tear them down.  
</p>
```

id e class

id: identifica **un** elemento

class: etichetta una **classe** di elementi

div e span

div: una **divisione** del documento

span: una **sequenza** di caratteri

non hanno particolare semantica

servono per attaccarvi semantica con *id* e *class*

Character sets: ASCII

ASCII: 128 simboli

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL	SOH	STX	ETX	EDT	ENG	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
2	SPC	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL

Non comprende le accentate....

Character sets: IBM OEM

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	☺	☹	♥	♦	♣	♠	↑	↓	→	←	♀	♂	↔	♂	♀
1	▶	◀	‡	!!	¶	§	±	↑	↓	→	←	↳	↔	▲	▼
2		!	"	#	\$	%	&	'	<	>	*	;	+	-	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N
5	P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^
6	'	a	b	c	d	e	f	g	h	i	j	k	l	m	n
7	p	q	r	s	t	u	v	w	x	y	z	{		}	~
8	ç	ü	é	â	ä	à	â	ç	ê	ë	è	ï	î	ÿ	À
9	É	æ	æ	ô	ö	ò	û	ü	ÿ	ö	ü	ç	½	¼	¾
A	á	í	ó	ú	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ	ñ
B	▨	▩	▪												
C	▬	▮	▯	▰	▱	▲	△	▴	▵	▶	▷	▸	▹	►	▻
D	μ	τ	π	ρ	σ	μ	τ	π	ρ	σ	μ	τ	π	ρ	σ
E	α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
F	≡	±	≥	≤	ρ	σ	τ	υ	φ	χ	ψ	ω	ζ	η	θ

Estende ASCII con un arbitrario mix di caratteri occidentali e grafici

Character sets: UNICODE

Tentativo di costruire **singolo char set per tutte le lingue**

Associa ad ogni carattere un **code point** (un numero)

“A” corrisponde a U+0065 (decimale)

I primi 127 code points coincidono con ASCII

Encodings

Un **character set encoding** è un codice per mappare una sequenza di **byte** su una sequenza di **code points**

UTF-16: assegnamo 16 bit per ogni carattere (2 versioni: little e big endian)

UTF-8: usa 1 byte per i caratteri ascii

ISO-8859-1 (latin-1): usa 1 byte, codifica ASCII e caratteri occidentali

ISO-8859-15: come latin-1, con l'aggiunta di "€"

Se non conosco l'encoding...

... non so come interpretare un file!

Specificare l'encoding

Attraverso HTTP:

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

Nel documento HTML:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd" >
<html>
  <head>
    <meta http-equiv="Content-type" content="text/html; charset=utf-8" >
    <title>Page Title</title>
  </head>
  <body>
    Ora posso scrivere direttamente "è" senza bisogno di "&egrave;"
  </body>
</html>
```

Inserire caratteri Unicode arbitrari

A equivale ad “A”

Alcuni caratteri utili:

’ ’ apice singolo sinistro (o apostrofo)

“ “ apici doppi sinistri

” ” apici doppi destri

Esercizi

- ▶ sperimentare con tutto quanto visto finora
- ▶ visitate il vostro sito preferito e esaminate il codice HTML
- ▶ idem, esaminando il codice CSS
- ▶ scaricate HTML Tidy e provate ad usarlo

Altri esercizi: trova il tuo posto nel mondo

Procurati accesso su un computer dotato di un web server (es. Apache)

Trova la **root** del tuo web server

Metti un file **index.html** nella root con il tuo nome

Connettiti con un browser

Verifica il tuo HTML con un validatore (es. validator.w3.org)

Trova nell'**access log** la traccia dei tuoi accessi