

# Introduzione

## Materiale:

- Slides
- Esercizi
- Codici esempio
- Progetto
- Atom
- Xampp
- Chrome

## Svolgimento:

- 1°: Slide
- 2°: Esempi
- 3°: Q&A

# Programma

## 1° Lezione:

- HTML
- Introduzione CSS

## 2° Lezione:

- CSS
- Esercizi HTML e CSS

## 3° Lezione:

- Introduzione Javascript

## 4° Lezione:

- Javascript
- Introduzione DOM e BOM

## 5° Lezione:

- Introduzione JQuery
- Introduzione XML
- Analisi Progetto
- Esercizi
- Quiz Finale

# Programmazione Web

## Front End

- HTML
- CSS
- Javascript: Alto livello, scripting, interpretato

### Framework/Librerie:

- JQuery
- Bootstrap
- Angular JS
- Vue JS

## Back End

- PHP
- MySQL
- Python
- Ruby
- Perl
- JAVA
- Javascript

### Framework:

- Laravel
- Node JS

# Sommario

1. World Wide Web
2. Iper testi e HTML
3. HTML di base
4. Form
5. HTML5

# **1. World Wide Web**

# Il World Wide Web

- World Wide Web = sistema di accesso a Internet basato sul protocollo HTTP
  - insieme di protocolli e servizi (HTTP, FTP, ...)
  - insieme di tool per l'accesso (es. web browser)
- Basato sulla metafora dell'**ipertesto**  
linguaggio HTML
- Distinzione tra Internet e WWW
  - Internet = rete
  - WWW = sistema di accesso alla rete

# Architettura del web

Architettura client-server:

- Web client (es. web browser)
  - inoltra richieste di **risorse** (documenti, file, ecc.) ad una macchina (web server)
- Web server
  - risponde alle richieste dei web client
- Sia il web server che il web client sono programmi in esecuzione su macchine connesse a Internet
- Web server e web client comunicano in base al protocollo HTTP

# Architetture client-server

- Basate sul concetto di richiesta di servizio (client) e di fornitura di servizio (server)
- Enormemente diffuse in informatica, in particolare nelle applicazioni di rete
  - HTTP
  - FTP
  - DNS
  - PPP
  - Proxy
  - .....



# Protocolli

- Protocollo = insieme di convenzioni (o regole) per lo scambio di informazioni
- protocolli di “basso” livello per Internet:
  - determinano le modalità di comunicazione tra i nodi della rete
  - TCP/IP
- protocolli di “alto” livello:
  - determinano il formato dei messaggi e le modalità di scambio dei messaggi
  - HTTP, FTP, SMTP, TELNET...

# Il protocollo HTTP

HTTP = HyperText Transfer Protocol

- Client-server
  - ogni interazione è una richiesta (messaggio ASCII) seguita da una risposta (messaggio tipo MIME)
- 7 metodi nativi:
  - GET
  - HEAD
  - PUT
  - POST
  - DELETE
  - LINK
  - UNLINK

# Il protocollo HTTP

- Client-server
- HTTP è connectionless = **non** si instaura una connessione prima di richiedere un servizio

- FTP:

- richiesta connessione
- instaurazione connessione
- richiesta servizio 1
- fornitura servizio 1
- richiesta servizio 2 ...
- chiusura connessione

- HTTP:

- richiesta servizio
- fornitura servizio

# URL

- In HTTP ogni interazione è relativa ad una URL (Uniform Resource Locator)
- La URL è un nome che identifica univocamente ogni risorsa disponibile sul web
- Ogni URL specifica:
  - il protocollo da utilizzare per il trasferimento della URL
  - il dominio, cioè il nome (simbolico) del computer su cui si trova il server (web server o altro server) che gestisce la risorsa
  - il nome, all'interno del dominio, della risorsa che si vuole accedere

# Domain Name System (DNS)

- Sistema per introdurre nomi logici (o simbolici) ai computer su Internet
- IP (Internet Protocol):
  - l'indirizzo del computer è una sequenza di 4 numeri da 0 a 255
  - es. 151.100.16.20
- DNS:
  - l'indirizzo del computer è una stringa di caratteri
  - es. `www.google.com`
- Il DNS è basato sul concetto di **dominio**

# Domini

Domini radice o di primo livello:



- COM, ORG, NET, EDU, MIL, GOV, INT
- biletterale per ogni nazione (es. IT)

Per ogni dominio di primo livello:

- domini di secondo livello (es. IBM.COM, VIRGILIO.IT)
- ogni dominio di primo livello gestisce in modo autonomo i propri domini secondari
- domini di terzo livello, quarto, ecc.

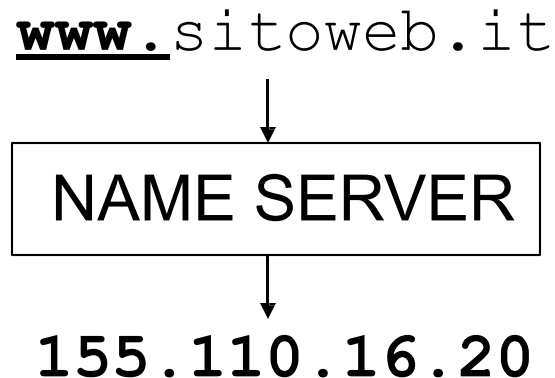
i nomi dei domini sono case-insensitive

# Name Server

- Occorre tradurre il nome simbolico in indirizzo IP
- NAME SERVER (o Domain Name Server)
- Si deve ricorrere ad un name server ogniqualvolta si fa uso di un nome simbolico
- Es. ogni web browser che richiede una URL, deve **prima** richiedere ad un name server l'indirizzo IP corrispondente al dominio nella URL:
- Dominio            Name Server            indirizzo IP

# URL: esempio

URL: http://www.sitoweb.it/index.html



- `/index.html` è il nome (completo di percorso) del file corrispondente alla URL



# Tipi di URL

Protocolli HTTP e FTP:

- Documenti ipertestuali (file HTML)
- Immagini
- Documenti multimediali (audio e video)
- Programmi
- File di altro genere

Altri protocolli: (es. mailto:)

- indirizzi di email
- .....

# Domini, siti web e pagine web

Distinzione tra domini, web server, siti web e pagine web:

- dominio = computer dove gira un web server (“identifica” il web server)
- **sito web** = insieme di URL gestite da un’unica entità e organizzate in modo da essere accedute secondo un ordine logico
  - più siti web possono essere gestiti dallo stesso web server
- **pagina web** = singolo documento HTML
- **home page** = pagina iniziale di un sito web

## **2. Iper testi e HTML**

# Iper testi

Iper testo = documento contenente:

- testo
- immagini
- audio
- video
- **collegamenti ipertestuali** = riferimenti ad altri documenti (o parti di documenti) ipertestuali

# Collegamenti ipertestuali

- Sul World Wide Web, un collegamento ipertestuale ad un documento può essere specificato tramite la URL che corrisponde al documento
- Si possono usare le tecnologie informatiche per **accedere direttamente** ai documenti corrispondenti ai link mentre si legge un ipertesto
- (esempio: web browser)
- superamento dell'accesso “sequenziale” al testo

# Linguaggio per ipertesti: HTML

- HTML = HyperText Markup Language
- Linguaggio standard per la specifica di documenti ipertestuali sul World Wide Web
- Linguaggio a marcatura, “figlio” di SGML (Standard Generalized Markup Language)
- Un documento HTML può contenere:
  - testo
  - link ipertestuali
  - immagini
  - link a risorse (URL) di ogni tipo

# Breve storia di HTML (1)

- definito (insieme ad HTTP) da Tim Berners-Lee del CERN di Ginevra nel 1989
- scopo: permettere lo scambio dei dati sperimentali tra i fisici
- 1993: diffusione di Mosaic (web browser sviluppato da NCSA)
- 1995: fondazione del World Wide Web Consortium (W3C)
- 1999: standardizzazione di HTML 4.0
- 2000: standardizzazione di XHTML 1.0 (ridefinizione di HTML basata su XML)

# Breve storia di HTML (2)

- 2004: viene fondato il WHATWG, Web Hypertext Application Technology Working Group
  - Apple, Mozilla Foundation, Opera Software, Google
- WHATWG nasce in contrapposizione con il futuro standard XHTML2 del W3C, che sarebbe stato un linguaggio non retrocompatibile con HTML 4
- ottobre 2014: standardizzazione di HTML5
- novembre 2016: standardizzazione di HTML 5.1
- dicembre 2017: standardizzazione di HTML 5.2
- maggio 2019: accordo tra W3C e WHATWG
  - Il «Living standard» di HTML di WHATWG diventa il riferimento comune dei due consorzi



# Sintassi di HTML

- Documento HTML = testo ASCII
- contiene:
  - blocchi di testo
  - tag (marcature o “comandi”)
- tag = testo delimitato dai simboli “<” e “>”
- esempio:

<nome-tag>

</nome-tag>

# Il concetto di tag

- TAG = “marcatura” (o marcatore)
- Un tag viene usato per **marcare** una parte di testo
- Tag:
  - di formattazione (per cambiare l’aspetto ad una parte di testo) (es. <font>)
  - “semantici” (per dare un “significato” ad una parte di testo) (es. <a>)
- 2 tipi di tag:
  - tag di apertura (marcatore iniziale)
  - tag di chiusura (marcatore finale)

<nome-tag>

</nome-tag>

# Attributi dei tag

- Ogni occorrenza di tag (di apertura) può contenere assegnazioni di **attributi**
- ogni tag ha un diverso insieme di possibili attributi
- assegnazione: nome-attributo = valore
  - es. `<font face="arial" size = "+1">`
- struttura del tag con attributi:

`<nome-tag attributo1 = valore1 attributo2 = valore2 ....>`

- alcuni attributi del tag sono **obbligatori**  
(vanno assegnati)

# Elementi e tag

- Elemento = insieme formato da tag di apertura, testo e tag di chiusura corrispondente
- Esempio di elemento (font):  
`<font face="arial">ciao </font>`
- In genere si usa il termine “tag” erroneamente, per indicare un elemento (composto da due tag)

Per HTML i tag sono case-insensitive (es.

`<font>` **e** `<FONT>` hanno lo stesso significato)

# Semantica di HTML

Il “significato” di un documento HTML è dato da due componenti:

- aspetto del documento
- “contenuto” (rispetto ai tag) del documento

La semantica “immediata” di un documento

HTML è la sua visualizzazione sul browser

- dipendente dal browser
- perdita (parziale) del significato legato al “contenuto”

# Struttura di un documento HTML

```
<html>      <-- inizio del documento
<head>
...
...          <-- intestazione del documento
</head>
<body>
...          <-- corpo del documento
...
</body>
</html>      <-- fine del documento
```

# Informazione e meta-informazione

- corpo del documento = contiene l'informazione (il documento stesso)
- intestazione del documento = contiene **meta-informazione** (cioè informazioni **sul** documento)
  - esempi:
    - autore del documento
    - parole chiave
    - “titolo” del documento
    - ....

# Contenuto e presentazione

- Problema: distinguere tra
  - **contenuto** del documento
  - **presentazione** (o aspetto) del documento
- E' molto importante poter individuare il contenuto di un documento indipendentemente dalla formattazione del documento
- Nelle intenzioni, HTML doveva mantenere separati i due aspetti. Nella realtà, non è così:
  - i marcatori sono usati anche per dare attributi di formattazione al testo
  - i vari browser “interpretano” il codice HTML in modo diverso



# HTML e browser HTML

I principali web browser, specie in passato, hanno influito sull'evoluzione di HTML:

- imponendo nuovi elementi del linguaggio
- “rifiutando” (cioè non supportando) nuovi elementi del linguaggio
- Problemi principali:
  - differente interpretazione di HTML
  - presenza di vecchie versioni dei browser

# Creare documenti HTML

Documento HTML = testo ASCII

(analogo ad un programma sorgente JAVA)

Modalità di creazione di un documento HTML:

- con un editor per testo ASCII (es. blocco note o Wordpad sotto Windows)
- con un “editor WYSIWYG” o “editor HTML” (es. FrontPage, Dreamweaver)
- con un editor di testi “normale” che permette di esportare i documenti in HTML (es. Word)

# Editor HTML

- Permette di editare il documento vedendo direttamente come verrà visualizzato dal browser
- Facilità di utilizzo:
  - non è necessario conoscere il linguaggio HTML
- Limiti nella realizzazione:
  - non tutte le potenzialità di HTML possono essere utilizzate
- Editor HTML professionali possono essere utilizzati al massimo solo conoscendo il codice HTML

# **3. HTML di base**

# Sommario

- **intestazione**
- formattazione testo
- link
- oggetti, immagini e applet
- tabelle
- frame

# Parte intestazione

- contiene una serie di informazioni necessarie al browser per una corretta interpretazione del documento, ma non visualizzate all'interno dello stesso:
- tipo di HTML supportato
- titolo della pagina
- parole chiave (per motori di ricerca)
- link base di riferimento
- stili (comandi di formattazione)

# Parte intestazione

Elementi principali:

- DOCTYPE
- HTML
- HEAD
- TITLE
- META

# Parte intestazione

```
<!DOCTYPE HTML PUBLIC="-//W3C//DTD HTML 4.0//EN">
<HTML>
<HEAD>
<META name="keywords" Content= "HTML, parte
    intestazione, meta-informazione">
<META name= "author" Content = "Nome Cognome">
<meta name="GENERATOR" content="Blocco note di
    Windows 10">
<TITLE>Pagina web di prova </TITLE>
</HEAD>
.....
</HTML>
```



# Parte intestazione

## DOCTYPE:

```
<!DOCTYPE HTML PUBLIC="-//W3C//DTD HTML 4.0//EN">
```

- fornisce informazioni sul tipo di documento visualizzato
- deve essere il primo elemento ad aprire il documento
- non è obbligatorio

# Parte intestazione

## META:

```
<META name="keywords" Content= "HTML, parte  
intestazione, meta-informazione">
```

```
<META name= "author" Content = "Nome Cognome">
```

```
<meta name="GENERATOR" content="Blocco note di  
Windows 10">
```

- fornisce meta-informazioni sul contenuto del documento
- usate dai motori di ricerca per classificare il documento
- non è obbligatorio

# Parte intestazione

## TITLE:

```
<TITLE>Pagina web di prova </TITLE>
```

- Titolo della pagina
- Compare sulla barra del titolo della finestra del browser
- Usato dai motori di ricerca
- Usato nella visualizzazione di “bookmark” (siti preferiti)

# HTML di base

- intestazione
- **formattazione testo**
- link
- oggetti, immagini e applet
- tabelle
- frame

# Corpo del documento

- Memorizza il contenuto del documento (la parte visualizzata all'utente del browser)

```
<body ..... >
```

```
.....
```

```
</body>
```

- gli attributi di `<body>` configurano alcuni parametri di visualizzazione del documento (in realtà il loro utilizzo è «deprecato»)

# Attributi di <body>

- BACKGROUND: setta lo sfondo
- BGCOLOR: setta il colore di sfondo
- TEXT: setta il colore del testo
- LINK, VLINK, ALINK: settano il colore del testo corrispondente ai link:
  - LINK: link non visitato
  - VLINK: link già visitato
  - ALINK: link “attivo”

# Colori del corpo: formato RGB

Esempio:

```
<BODY BGCOLOR="red">
```

identico a

```
<BODY BGCOLOR="#ff0000">
```

- Colori in formato RGB = 3 numeri (componenti rossa, verde e blu)
- ogni colore va da 0 a 255 (FF in esadecimale)
  - es. #000000 = nero, #FFFFFF = bianco,  
#777777 = grigio, #00FF00 = verde, #0000FF = blu

# Esempio:

```
<body background="sfondo.jpg"  
  link="blue" vlink="red" alink="red">
```

- **BACKGROUND** fa riferimento ad una URL (nell'esempio un file locale) contenente una immagine
- l'immagine viene disegnata (eventualmente in modo ripetuto) per riempire lo sfondo del documento



# Nota bene

Per separare forma e contenuto:

- gli attributi di formattazione NON dovrebbero comparire nel corpo del documento
- gli attributi di `<body>` NON dovrebbero essere usati
- nella pratica vengono usati, ma il W3C “depreca” ufficialmente tale utilizzo (come anche, ad esempio, l’uso di `<font>`)
- alternativa: uso di XML = formattazione del tutto separata dal contenuto

# Header

- `<H1>,<H2>,...,<H6>`
- Permettono di inserire titoli (o intestazioni) all'interno del documento
- il testo tra `<Hn>...</Hn>` viene evidenziato dal browser
- 6 livelli di titoli
- 6 differenti livelli di enfaticizzazione del testo

# Enfatizzare il testo

<B>, <I>, <U>:

- <B> (bold):
  - permette di visualizzare testo in neretto
- <I> (italic):
  - permette di visualizzare testo in corsivo
- <U> (underlined):
  - permette di sottolineare il testo
- definiscono attributi **fisici** di formattazione

# Attributi logici e fisici

- tag fisico = ha il compito di formattare il documento
- tag logico = dà una struttura al documento, ed è indipendente dalla visualizzazione
- esempio: elemento `<address>`
  - permette di specificare che una parte di testo è un indirizzo
  - viene visualizzato come testo in corsivo
  - per il browser è come usare `<i>`
  - ma `<ADDRESS>` aggiunge “semantica” al documento

# Selezione del font

- Tag fisico
- deprecato da W3C
- definisce il modo in cui deve essere visualizzata una parte di testo:
  - tipo di carattere
  - colore
  - grandezza

# Attributi del tag <font>

- FACE
  - determina il tipo di carattere (font) usato
  - font disponibili: courier, times, arial, verdana, ...
- SIZE
  - determina la grandezza dei caratteri
  - si esprime con numeri assoluti (da 1 a 7) o relativi
- COLOR determina il colore

esempio:

```
<FONT FACE="verdana" SIZE="+1" COLOR="green">  
testo in verde</FONT>
```

# Apici e pedici

- `<SUB>` (subscript)
  - permette di scrivere testo come “pedici”
- `<SUP>` (superscript)
  - permette di scrivere “apici”
- **esempio:**

`I<SUB>5</SUB> = 2<SUP>4</SUP>`

viene visualizzato come

$$I_5 = 2^4$$

# Testo preformattato

- Tag `<XMP>` e `<PRE>`
- permettono di visualizzare il testo esattamente come è scritto (“preformattato”) nel file sorgente HTML
- il testo preformattato non viene “interpretato”
- `<XMP>` non interpreta neanche le occorrenze dei tag HTML dentro al testo preformattato
- con `<PRE>` invece le occorrenze di tag HTML vengono interpretate
  - `<PRE>` è il tag di preformattazione di riferimento

per HTML 4.0



# Testo preformattato con <XMP>

Codice HTML:

```
<XMP>
begin
  if
    then
end; I<SUB>5</SUB>      = 2<SUP>4</SUP>
</XMP>
```

Visualizzazione:

```
begin
  if
    then
end;      = 2<SUP>4</SUP>
```

Linguaggi e tecnologie per il Web

# Testo preformattato con <PRE>

Codice HTML:

```
<PRE>
begin
  if
    then
end; I<SUB>5</SUB>      = 2<SUP>4</SUP>
</PRE>
```

Visualizzazione:

```
begin
  if
    then
end;  $I_5 = 2^4$ 
```

# Stili logici

- **<ADDRESS>**
  - marcatura usata per indirizzi (mail, email, telefono,...)
- **<BLOCKQUOTE>**
  - usato per inserire nel testo citazioni da un altro testo o autore
- **<CITE>**
  - usato per la fonte della citazione
- **<EM> e <STRONG>**
  - “enfaticizzano” il testo all’interno del tag
- **<VAR> e <CODE>**
  - utilizzati per righe di codice di programmazione

# Separare e allineare il testo

- `<P>` (paragraph)
  - crea un paragrafo all'interno del testo
- `<BR>` (break)
  - interruzione di riga (“a capo”)
- `<DIV>`
  - usato per allineare il documento:
    - `<DIV align = left>` allinea a sinistra il testo
    - `<DIV align = center>` allinea al centro il testo
    - `<DIV align = right>` allinea a destra il testo
- `<CENTER>` tag non standard

# Righe orizzontali

- `<HR>` (horizontal row)
  - aggiunge una riga orizzontale al testo
  - usato per separare parti di testo
- attributi di `<HR>`:
  - `ALIGN` (left|center|right|) allineamento rispetto alla pagina
  - `WIDTH` lunghezza orizzontale (in pixel o in percentuale)
  - `SIZE` altezza della riga in pixel
  - `COLOR` colore della riga
  - `NOSHADE` elimina l'effetto 3D

# Liste puntate

- `<UL>` (unordered list)
  - produce un elenco (lista) di elementi (parti di testo)
  - ogni elemento è evidenziato all'inizio da un simbolo grafico (di solito cerchietto o quadratino)
- `<LI>` (list item)
  - per identificare un elemento della lista
- esempio:

```
<UL>
<LI>Primo elemento </LI>
<LI>Secondo elemento </LI>
<LI>Terzo elemento </LI>
</UL>
```

# Liste numerate

- `<OL>` (ordered list)
  - produce un elenco (lista) di elementi (parti di testo)
  - ogni elemento è evidenziato all'inizio dal numero d'ordine all'interno della lista
- `<LI>` (list item) (come per liste puntate)
- esempio:

```
<OL>  
<LI>Primo elemento </LI>  
<LI>Secondo elemento </LI>  
<LI>Terzo elemento </LI>  
</OL>
```

# Liste numerate

Oltre al numero progressivo, si possono usare altre indicizzazioni per le liste puntate

Uso dell'attributo TYPE di <OL>:

- <OL TYPE=A> indicizza con lettere alfabetiche maiuscole
- <OL TYPE=a> indicizza con lettere alfabetiche minuscole
- <OL TYPE=I> indicizza con numeri romani maiuscoli
- <OL TYPE=i> indicizza con numeri romani minuscoli



# Esempio

## Esempio di liste annidate:

```
<ol>  
<li>gruppo di nomi:  
<ul>  
<li>nome a </li>  
<li>nome b </li>  
</ul></li>  
<li>gruppo di nomi:  
<ul>  
<li>nome c </li>  
<li>nome d </li>  
<li>nome e </li>  
</ul></li>  
</ol>
```

# HTML di base

- intestazione
- formattazione testo
- **link**
- oggetti, immagini e applet
- tabelle
- frame

# Link ipertestuali

Tag <A> (anchor)

- permette l'inserimento di link ipertestuali all'interno del documento
- attributo principale: HREF (Hypertext reference)
  - specifica la URL che viene associata al link
- il testo tra <A> e </A> viene associato a tale URL
  - cliccando su tale testo il browser accede alla URL
- Esempio:

```
<A HREF="http://www.virgilio.it">Visita  
virgilio.it</A>
```

# Attributi del tag <a>

- 2 tipi di tag <a>:
- con attributo HREF:
  - aggiungono un link ipertestuale (esterno o interno al documento)
- con attributo NAME:
  - definiscono uno specifico punto del documento come un link interno
  - tale punto può essere direttamente acceduto attraverso un tag <A HREF...>
- altri attributi: TARGET, TITLE

# Esempio

...

```
<a name="zona1"> zona 1 del documento  
raggiungibile direttamente </a>
```

...

```
<a href="#zona1">torna alla zona 1 del  
documento</a>
```

...

**Con l'anchor zona1 si può anche accedere direttamente dall'esterno del documento:**

```
<a href="www.esempio.com#zona1">  
vai alla zona 1 del documento index.html del sito  
esempio.com </a>
```

# Attributo target di <a>

Attributo TARGET di <A>:

- permette di specificare dove deve essere visualizzata la URL associata al link

valori principali di TARGET:

- `_NEW`: visualizza la URL collegata in una nuova finestra (o tab) del browser
- `_PARENT`: visualizza nella stessa finestra, eliminando tutti i frame presenti (vedere sezione sui frame)

# Attributo title di <a>

Attributo TITLE di <A>:

- permette di specificare una informazione associata al link
- es. commento riguardante il link
- i browser visualizzano tale informazione (pop-up) quando il puntatore del mouse passa sopra al link

# HTML di base

- intestazione
- formattazione testo
- link
- oggetti, immagini e applet
- tabelle
- frame



# Immagini

- HTML permette di inserire immagini nel documento
- Tag <IMG> (singolo)
- permette di inserire nel documento una immagine, memorizzata in un file (o URL) a parte
- i browser riconoscono i principali formati immagine (es. JPG, GIF, BMP)

# Attributi del tag <img>

- SRC
  - specifica il nome della URL contenente l'immagine
  - es. `<IMG SRC="foto1.jpg">`
- WIDTH larghezza (in pixel o percentuale)
- HEIGHT altezza (in pixel o percentuale)
  - se WIDTH e HEIGHT mancano, l'immagine viene visualizzata nelle sue dimensioni originali
- ALT
  - permette di aggiungere un commento testuale associato all'immagine

# Attributi del tag <img>

- **BORDER**
  - spessore cornice dell'immagine (in pixel)
- **HSPACE e VSPACE**
  - distanze minime orizzontali e verticali (in pixel) dell'immagine dagli oggetti più vicini
- **ALIGN**
  - determina l'allineamento del testo rispetto all'immagine

# L'attributo ALT

- Permette di aggiungere una informazione testuale all'immagine
- Il testo viene visualizzato dai browser (pop-up)
- Necessario per i browser solo testuali
- Oppure per la navigazione con immagini disabilite

**es.** `<IMG SRC="topolino.gif"  
ALT="disegno che raffigura Topolino  
e Pippo">`

# L'attributo **ALIGN**

- determina l'allineamento del testo rispetto all'immagine
  - **ALIGN=top**: allinea la prima riga di testo sulla sinistra al top dell'immagine
  - **ALIGN=middle**: allinea la prima riga di testo sulla sinistra al centro dell'immagine
  - **ALIGN=bottom**: allinea la prima riga di testo sulla sinistra nella parte più bassa dell'immagine
  - **ALIGN=left**: allinea il testo sulla destra dell'immagine partendo dal top
  - **ALIGN=right**: allinea il testo sulla sinistra dell'immagine partendo dal top

# Mappe cliccabili

- Una immagine può essere associata ad un link
- **es.** `<a href="pippo.htm"></a>`
- spesso si vogliono associare due o più link alla stessa immagine
  - associare zone diverse dell'immagine a diversi link
- **mappe cliccabili**
- 2 tipi:
  - lato server (poco diffuse)
  - lato client (USEMAP)

# Mappe cliccabili (lato client)

```
<IMG SRC="img1.gif" WIDTH=400 HEIGHT=100  
  BORDER=0 usemap="#immagine1">
```

```
<map name="immagine1">
```

```
<area shape="rect" alt="parte 1  
  immagine" coords="0,0,200,100"  
  href="doc1.htm" title="parte 1  
  immagine">
```

```
<area shape="rect" alt="parte 2  
  immagine" coords="201,0,400,100"  
  href="doc2.htm" title="parte 2  
  immagine">
```

```
<area shape="default" nohref>
```

# Generare mappe cliccabili

- Tipi di aree definibili con usemap:
  - rect
  - circle
  - poly
- difficili da definire a mano
- uso di programmi (editor di mappe)
  - es. MAPEDIT



# Programmi e documenti HTML

Tipi più diffusi di programmi associati a pagine HTML:

- applet JAVA
- script (es. scritti in JavaScript)

**applet** = file (estensione .class) esterni al documento HTML

**script** = righe di codice scritte all'interno del documento HTML

# Applet e script

- Applet = codice compilato (bytecode JAVA)
- script = codice sorgente

Il browser deve essere in grado di interpretare sia bytecode JAVA che sorgente JavaScript:

- JAVA virtual machine
- interprete JavaScript

Differenze:

- Applet non modificabile (bytecode)
- script facilmente modificabile (sorgente)

# Applet

## Esempio:

```
<APPLET CODE= "applet1.class" WIDTH=500  
  HEIGHT = 300>
```

- L'uso di `<applet>` è deprecato in HTML 4.0
- proposta: uso di `<object>`
  - inclusione di oggetti generici (tra cui applet) nel documento HTML
  - prepara HTML per future applicazioni

# Simboli speciali

- Come rappresentare simboli non-ASCII e simboli utilizzati da HTML?
- insieme di definizioni di simboli (ogni simbolo è rappresentato da un nome)
- l'invocazione di un simbolo predefinito inizia con “&” e termina con “;”
- esempio: `&copy;` per rappresentare ©

# Simboli speciali

- esempi: lettere accentate:
  - `&agrave;` à
  - `&egrave;` è
  - `&eacute;` é
  - `&igrave;` ì
  - `&ograve;` ò
  - `&ugrave;` ù
- il simbolo “&” si rappresenta con `&amp;`

# Il simbolo “<”

- < è un simbolo particolarmente speciale in HTML (apertura dei tag)
- < si rappresenta con `&lt;`;
- > si rappresenta con `&gt;`;

# HTML di base

- intestazione
- formattazione testo
- link
- oggetti, immagini e applet
- **tabelle**
- frame

# Tabelle

- Rappresentano informazione in forma tabellare (righe e colonne) nei documenti HTML
- Molto utilizzate anche come strumento di formattazione di testi e/o immagini
- HTML permette una gestione piuttosto potente delle tabelle



# Elementi relativi alle tabelle

- TABLE
  - definisce la tabella
- TD
  - definisce un campo “dati” all’interno della tabella
- TR
  - suddivide i campi in righe all’interno della tabella
- TH
  - definisce campi intestazione all’interno della tabella
- THEAD, TFOOT

# L'elemento <table>

- Racchiude tutta l'informazione relativa ad una tabella

- Esempio:

```
<TABLE WIDTH=300 HEIGHT=200>
```

```
.....
```

```
</TABLE>
```

- gli attributi di <table> settano proprietà globali della tabella

# Dimensioni della tabella

Esprese in:

- pixel (punti)
  - es. `<table width=300 height=200>`
  - indipendente dalle dimensioni della finestra di visualizzazione
- percentuale della dimensione della pagina
  - es. `<table width="60%">`
  - dipendente dalle dimensioni della finestra di visualizzazione

la scelta tra i due tipi di dimensioni dipende  
dalla applicazione

# Attributi di <table>

- WIDTH larghezza
- HEIGHT altezza (non dovrebbe essere usato)
- BORDER spessore del bordo (in pixel)
- BGCOLOR colore sfondo tabella
- SUMMARY testo che spiega il contenuto della tabella (per media non visuali)
- CELSPACING distanza tra i campi (celle) della tabella
- CELLPADDING distanza in pixel tra il contenuto del campo e i margini del campo

# L'elemento <TD>

<TD> permette la definizione di un singolo campo (cella)

- va usato per campi di tipo “dati”
- non va usato per campi di tipo intestazione di colonne
- esempio:

```
<TD width=100>prova</TD>
```

definisce una cella con contenuto `prova`

# Attributi di <TD>

- WIDTH, HEIGHT non dovrebbero essere usati
- VALIGN (top|bottom|middle) allineamento verticale
- ALIGN (left|center|right) allineamento orizzontale
- BGCOLOR colore di sfondo della cella
- BACKGROUND motivo di sfondo della cella
- ROWSPAN, COLSPAN

# L'elemento <TR>

- Divide le celle in righe
- Esempio:

```
<TABLE border=1 cellpadding=2>
```

```
<TR>
```

```
<TD>cella 1</TD>
```

```
<TD>cella 2</TD>
```

```
<TD>cella 3</TD>
```

```
<TR>
```

```
<TD>cella 1 riga 2</TD>
```

```
<TD>cella 2</TD>
```

```
<TD>cella 3</TD>
```

```
</TABLE>
```

cella 1	cella 2	cella 3
cella 1 riga 2	cella 2	cella 3

# Attributi di <TR>

- ALIGN (left|center|right) allineamento orizzontale delle celle che seguono <TR>
- VALIGN (top|middle|bottom) allineamento verticale
- BGCOLOR



# Esempio

```
<TABLE WIDTH=300 HEIGHT=200>  
<TD width=100 VALIGN=TOP>  
Prova1</TD>  
<TD WIDTH=100 VALIGN=BOTTOM>  
Prova2</TD>  
<TD WIDTH=100 VALIGN=MIDDLE>  
Prova3</TD>  
</TABLE>
```

Prova1		
	Prova2	Prova3

# Esempio

```
<TABLE WIDTH=300 HEIGHT=200 border=1>  
<TD width=100 ALIGN=RIGHT>  
prova1</TD>  
<TD WIDTH=100 ALIGN=CENTER>  
Prova2</TD>  
<TD WIDTH=100 ALIGN=LEFT>  
Prova3</TD>  
</TABLE>
```

Prova1	Prova2	Prova3
--------	--------	--------

# L'elemento <TH>

- Come <TD> ma va usato per specificare campi **intestazione**
- permette di dare più “semantica” alla tabella
- da un punto di vista di presentazione, per i browser non c'è differenza
- stessi attributi di <TD>

# Esempio

```
<TABLE border=1 cellpadding=2>
<TR>
<TH>nome</TH>
<TH>cognome</TH>
<TH>età</TH>
<TR>
<TD>Mario</TD>
<TD>Rossi</TD>
<TD>36</TD>
<TR><TD>Paola</TD>
<TD>Bianchi</TD>
<TD>32</TD>
</TABLE>
```

nome	cognome	età
Mario	Rossi	36
Paola	Bianchi	32

# L'elemento <CAPTION>

- Permette di associare una didascalia alla tabella
- esempio:

```
<TABLE border=1 cellpadding=2>  
<CAPTION>Elenco degli  
impiegati:</CAPTION>  
<TR>  
<TH>nome</TH>  
<TH>età</TH>  
<TR>  
<TD>Mario</TD>  
<TD>Rossi</TD>  
<TD>36</TD>  
<TD>Paola</TD>  
<TD>Bianchi</TD>  
<TD>32</TD>  
</TABLE>
```

Elenco degli impiegati:

nome	cognome	età
Mario	Rossi	36
Paola	Bianchi	32

# Elementi di raggruppamento righe

- `<THEAD>`, `<TBODY>`, `<TFOOT>`
- Permettono di suddividere l'informazione contenuta in una tabella per righe
- Permettono la gestione separata di intestazione e contenuto
- Permettono di raggruppare il contenuto della tabella in più gruppi (più occorrenze di `<TBODY>...</TBODY>`)
- Struttura non visualizzata dai browser (occorrono fogli di stile)

# Esempio

```
<TABLE border=1
cellpadding=2>
<THEAD>
<TR><TH>nome</TH>
<TH>cognome</TH>
<TH>et&agrave;</TH>
</THEAD>
<TBODY>
<TR>
<TD>Mario</TD>
<TD>Rossi</TD>
<TD>36</TD>
<TR>
<TD>Paola</TD>
<TD>Bianchi</TD>
<TD>32</TD>
</TBODY>
```

# Raggruppamento delle colonne

- `<COLGROUP>`, `<COL>`
- Permettono di suddividere l'informazione contenuta in una tabella per colonne
- Struttura non visualizzata dai browser (occorrono fogli di stile)



# Celle variabili

- Una cella può occupare più righe o più colonne di una tabella
- Attributi ROWSPAN, COLSPAN di <TD>
- ROWSPAN = numero righe occupate dalla cella
- COLSPAN = numero colonne occupate dalla cella

# Esempio

```
<TABLE border=1 cellpadding=2>
<TR>
<TH>nome</TH>
<TH>cognome</TH>
<TH>età</TH>
<TR>
<TD colspan=2>Rossi</TD>
<TD>36</TD>
<TR><TD>Paola</TD>
<TD rowspan=2>Bianchi</TD>
<TD>32</TD>
<TR><TD>Maria</TD>
<TD>36</TD>
</TABLE>
```

Nome	Cognome	Età
Rossi		36
Paola	Bianchi	32
Maria		36

# HTML di base

- intestazione
- formattazione testo
- link
- oggetti, immagini e applet
- tabelle
- **frame**

# Frame

- Frame = riquadro (zona rettangolare) della finestra di visualizzazione del browser
- ogni frame è gestito in modo indipendente dal browser (come una finestra a sé stante)
- In HTML è possibile “organizzare” più documenti in un insieme di frame
  - i documenti sono presentati in un’unica schermata (finestra) divisa in frame
  - ogni documento è visualizzato in un diverso frame
  - la presentazione sfrutta la divisione in

# Ha senso utilizzare i frame?

- I frame aiutano a migliorare la fruizione di un sito
- ma: molti navigatori su web “odiano” i frame
- E’ possibile avere “annidamenti” di frame, che rendono difficile la fruizione delle pagine
- I frame rendono impossibile la navigazione per alcuni media (per esempio per i non vedenti)

Nel caso si usino i frame, è buona norma prevedere **sempre** una versione senza frame dei documenti

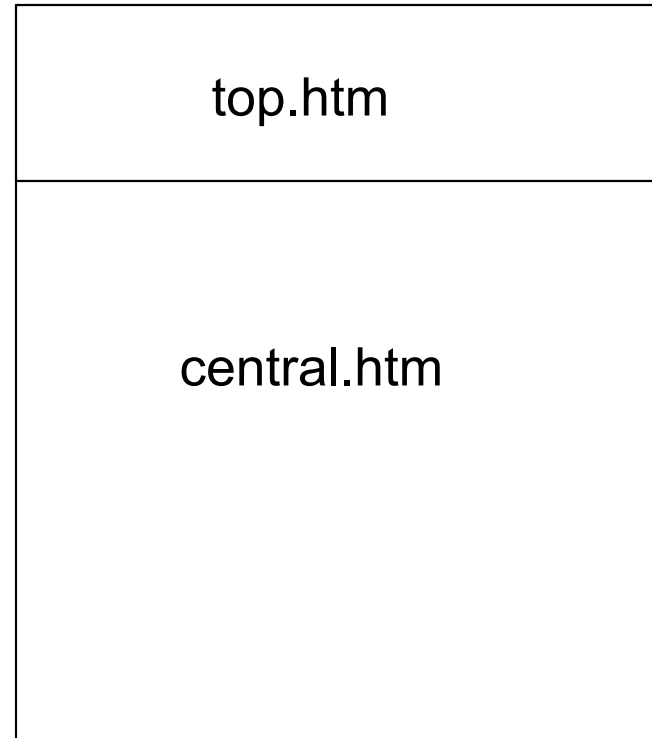
# Frame principale

- Per creare una pagina divisa in frame è necessario creare più files HTML richiamati da un file principale
- Il documento principale contiene l'elemento `<frameset>`

```
<FRAMESET rows="80, *">  
    <frame name="alto" src="top.htm">  
    <frame name="centrale"  
        src="central.htm">  
</FRAMESET>
```

# Frame

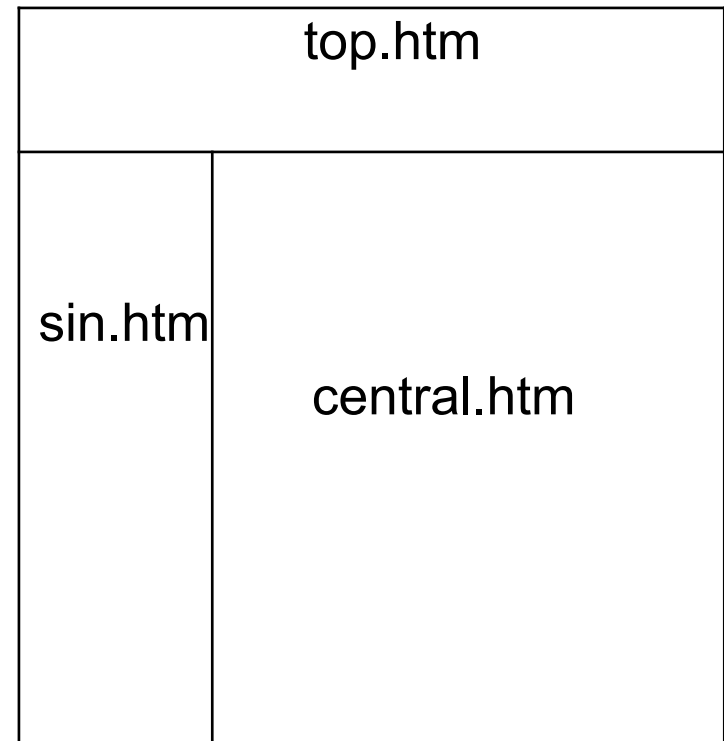
```
<FRAMESET rows="80, *">  
<frame name="alto"  
      src="top.htm">  
<frame name="centrale"  
      src="central.htm">  
</FRAMESET>
```



# Dimensioni dei frame

- righe (rows)
- colonne (cols)

```
<frameset rows="100, *">  
<frame name="alto"  
    src="top.htm">  
<frameset cols="150, *">  
<frame name="sx"  
    src="sin.htm">  
<frame name="centrale"  
    src="central.htm">  
</frameset>  
</frameset>
```



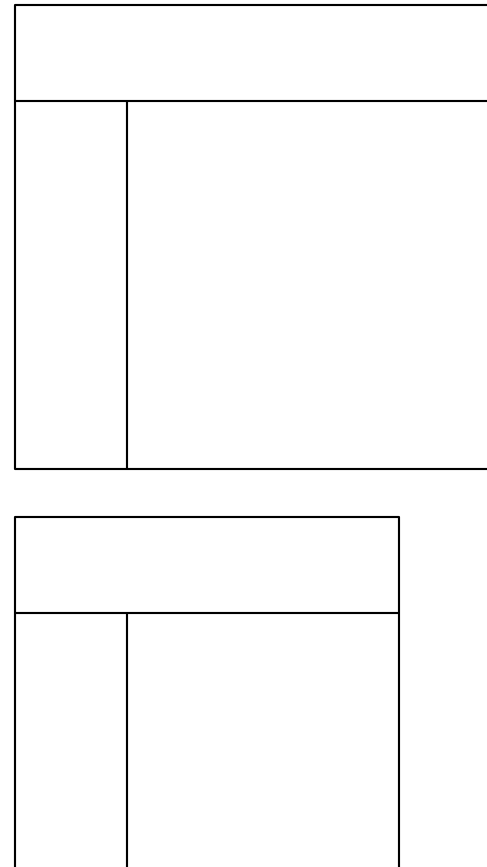


# Dimensioni dei frame

- Dimensioni assolute:
  - espresse come numero di punti (pixel)
  - non cambiano se cambiano le dimensioni della finestra del browser
  - es. `<FRAMESET rows="80, *">`
- Dimensioni relative:
  - espresse come percentuale della dimensione corrente della finestra
  - cambiano al variare della dimensione della finestra
  - es: `<FRAMESET rows="20%, *">`

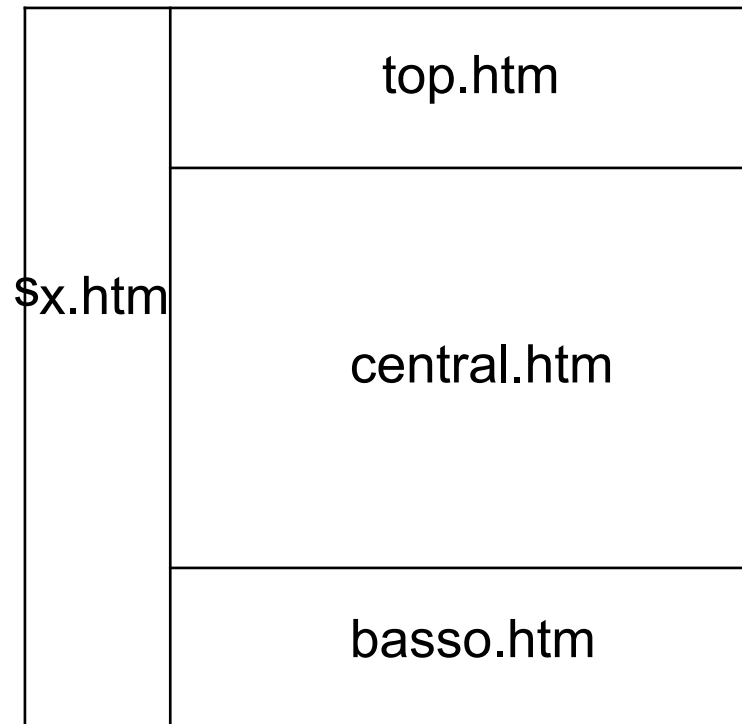
# Esempio

```
<frameset rows="100,*">  
<frame name="alto"  
  src="top.htm">  
<frameset cols="150,*">  
<frame name="sx"  
  src="sin.htm">  
<frame name="centrale"  
  src="central.htm">  
</frameset>  
</frameset>
```



# Esempio

```
<frameset cols="120, *" >
<frame name="sx"
      src="sx.htm">
<frameset
      rows="20%, 60%, 20%, *" >
<frame name="alto"
      src="top.htm">
<frame name="centrale"
      src="central.htm">
<frame name="basso"
      src="basso.htm">
</frameset>
</frameset>
```



# Elementi relativi ai frame

- <FRAMESET>
  - sostituisce l'elemento <BODY> nel frame principale
- <FRAME>
  - serve ad importare i frame secondari dal frame principale
- <NOFRAMES>
  - serve a specificare un documento **alternativo** al frame

# Attributi di <frameset>

- **ROWS** altezza
- **COLS** larghezza
- attributi della cornice:
  - **FRAMEBORDER** (= yes | no) presenza della cornice
  - **BORDER** spessore della cornice
    - BORDER = 0 elimina la cornice
  - **BORDERCOLOR** colore della cornice
- esempio:

```
<frameset cols="120,*"  
  rows="120,*"  
  bordercolor="#FF0000"  
  border="5">
```

# Attributi di <frame>

- **SRC** (search) URL da caricare nel riquadro
- **NAME** denominazione del frame
- **SCROLLING** (yes|no|auto) tipo di barra di scorrimento nel riquadro
- **MARGINWIDTH, MARGINHEIGHT**  
larghezza e altezza dei margini (risp. distanza dal margine alto e dal margine sinistro)
- **NORESIZE** dimensione non modificabile
- **BORDER, BORDERCOLOR** spessore e colore del margine

# Nomi dei frame

- L'attributo NAME dà un nome al riquadro
- Tale nome è usato per **indirizzare** il caricamento di una URL in un particolare riquadro
- Per fare questo si assegna il nome del frame all'attributo TARGET dell'elemento <A>
- es:

```
<A href="top2.htm" target = "centrale">
```

carica il documento top2.htm nel frame di  
nome

centrale (se esiste)

# Esempio

- Contenuto di top.htm:

```
<html>
```

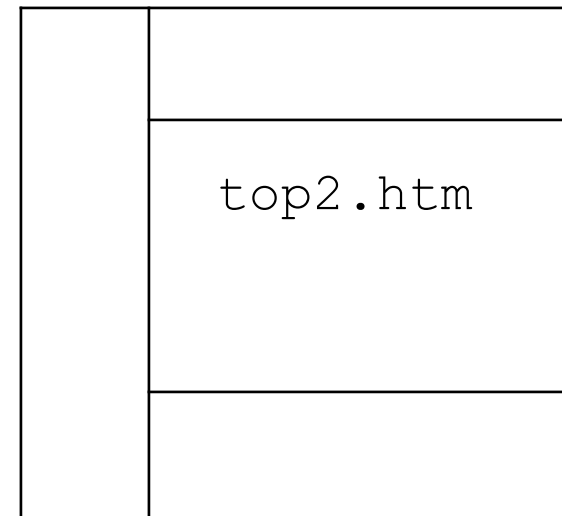
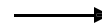
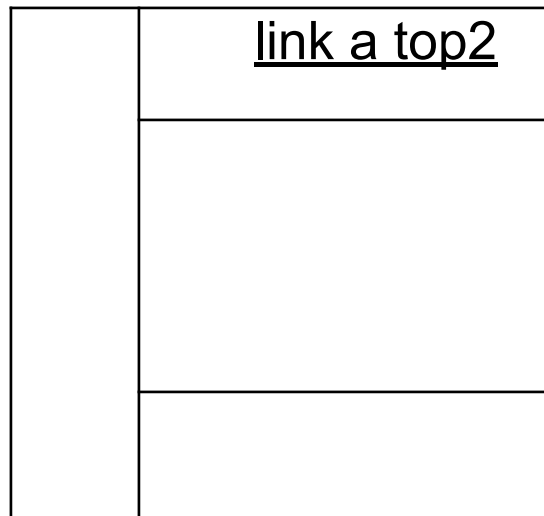
```
<head> </head>
```

```
<body>
```

```
<a href="top2.htm" target="centrale">link a  
  top2</a>
```

```
</body>
```

```
</html>
```



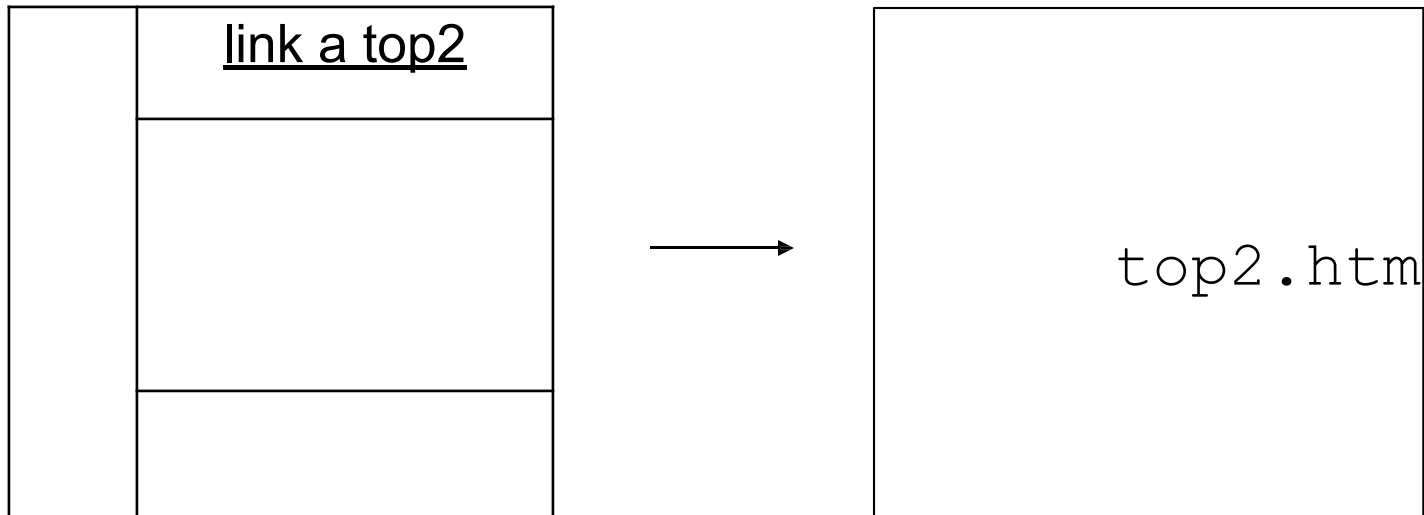


# Esempio (cont.)

- Uso del valore `_parent`:

```
<a href="top2.htm" target="_parent">link a  
top2</a>
```

l'attivazione di questo link elimina tutti i frame  
dalla finestra



# Caricare più frame

- E' possibile con una sola operazione effettuare il caricamento simultaneo di più pagine in due o più frame
- Per tale operazione è necessario uno **script** (per esempio in JavaScript)

# Esempio

```
<HEAD>
<script language="JavaScript">
<!--  Hiding
function loadtwo(page1, page2) {
parent.alto.location.href=page1;
parent.centrale.location.href=page2;}
// -->
</script>
</HEAD>

<BODY>
<FORM NAME="buttons">
<INPUT TYPE="button" VALUE="Clicca"
onClick="loadtwo('nuovo1.htm','nuovo2.htm') ">
</FORM>
</BODY>
```

# L'elemento `<noframes>`

- Necessario per permettere l'accesso al documento a tutti i media per cui **non** ha senso la nozione di frame
  - vecchie versioni dei browser
  - sistemi di navigazione web per non vedenti
  - wap browser, ecc.
- struttura:  
`<noframes>`  
codice HTML alternativo ai frame  
`</noframes>`

# Esempio

```
<frameset rows="100,*">  
<frame name="alto" src="top.htm">  
<frameset cols="150,*">  
<frame name="sx" src="sin.htm">  
<frame name="centrale" src="central.htm">  
</frameset>  
<noframes>  
<html><body>  
Attenzione! il tuo browser non supporta l'opzione  
frame. Per visualizzare queste pagine &grave;  
necessario un browser recente  
</body></html>  
</noframes>  
</frameset>
```

# L'elemento <iframe>

- Questo elemento («inline frame») permette di definire un frame in qualsiasi punto di un documento HTML
- struttura:

```
<iframe src="http://www.esempio.com"  
width="500" height="300">
```

codice HTML alternativo (per i browser  
che non supportano iframe

```
</iframe>
```

# 4. Form

# Form (modulo)

- usati per inviare informazioni via WWW
- il modulo viene compilato dall'utente (sul browser)
- quindi viene inviato al server
- un programma apposito sul server elabora il modulo
- in genere tale programma invia una “risposta” all'utente (pagina web, email, ecc.)



# Form e CGI

- CGI (Common Gateway Interface): metodo inizialmente più usato per elaborare form
- si possono usare programmi lato server alternativi al CGI usando i linguaggi di scripting lato server (PHP, JSP, ASP, Node.js/JavaScript, ecc.)
- in teoria è possibile evitare l'uso di CGI o di qualunque programma lato server (es. invio diretto di email dal browser)
- in pratica, ciò è possibile solo per form estremamente semplici

# L'elemento **<form>**

Apre e chiude il modulo e raccoglie il contenuto dello stesso

Esempio:

```
<FORM method="post"  
  action="http://www.esempio.it/cgi-  
  bin/nome_script.cgi">
```

- **attributo ACTION:** specifica la URL della risorsa (programma) che elabora la form

# L'attributo method

`method=get`

- i dati inseriti nella form vengono spediti al server e separati in due variabili
- le coppie nome-campo-form=valore-inserito compaiono alla fine della URL usata per l'invio del messaggio (i dati sono quindi visibili già nella URL)

- Esempio (Google):

`https://www.google.com/search?q=linguaggi+e+tecnologie+_per+il+web&ie=utf-8&oe=utf-8&client=firefox-b`

- per questo metodo il numero massimo di caratteri contenuti nella form è circa 3000

# L'attributo method

`method=post`

- i dati vengono ricevuti direttamente dal programma (lato server) senza un preventivo processo di decodifica
- questa caratteristica fa sì che lo script possa leggere una quantità illimitata di caratteri

# Campi editabili del modulo

Vengono definiti tramite gli elementi:

- INPUT (campi editabili, checkbox, radio, ecc.)
- TEXTAREA (area di testo)
- SELECT (creazione di menu di opzioni)

# L'elemento INPUT

- Permette l'inserimento di campi editabili e/o modificabili nel modulo
- Attributi principali:
  - TYPE: determina il tipo di campo
  - NAME
  - VALUE
  - MAXLENGTH

# Attributo TYPE di <INPUT>

- Attributo TYPE: determina il tipo di campo
- Principali possibili valori di tale attributo:
  - HIDDEN
  - TEXT
  - PASSWORD
  - CHECKBOX
  - RADIO
  - SUBMIT
  - RESET
  - IMAGE

# TYPE=“HIDDEN”

- Usato per dare informazioni “nascoste” al CGI (cioè al server)
- Il suo uso dipende dal tipo di CGI associato al modulo



# TYPE="HIDDEN"

## Esempi:

```
<INPUT type="HIDDEN" name=MAILFORM_SUBJECT  
value="titolo del form">
```

- Questo codice determina il titolo (subject) del messaggio che verrà ricevuto via e-mail, e che conterrà il contenuto del modulo

```
<INPUT TYPE="HIDDEN" NAME=MAILFORM_URL  
VALUE="http://www.tuosito.it">
```

- dopo aver compilato e spedito correttamente il form, dà in risposta una pagina HTML successiva (es. pagina di conferma di invio modulo avvenuta)

# TYPE="TEXT"

```
<INPUT type="TEXT"
  name="nome"   maxlength="40"
  size="33"     value="inserisci
  nome">
```

- crea i tipici campi di testo
- usato soprattutto per informazioni non predefinite che variano di volta in volta
- attributi di <INPUT> nel caso TYPE=TEXT:
  - MAXLENGTH (num. max caratteri inseribili)
  - SIZE (larghezza campo all'interno della pagina)
  - VALUE (valore di default che compare nel campo)

# TYPE="PASSWORD"

```
<INPUT type="PASSWORD" name="nome"  
    maxlength="40" size="33">
```

- crea i campi di tipo password
- vengono visualizzati asterischi al posto dei caratteri
- i dati NON vengono codificati (problema per la sicurezza)

# TYPE="CHECKBOX"

```
<INPUT type="CHECKBOX"  
  name="eta"   size="3"  
  VALUE="YES" CHECKED>
```

- crea campi booleani (si/no)
- crea delle piccole caselle quadrate da spuntare o da lasciare in bianco
- `VALUE` impostato su `YES` significa che di default la casella è spuntata
- `CHECKED` controlla lo stato iniziale della casella, all'atto del caricamento della pagina

# TYPE="RADIO"

```
<INPUT type="RADIO"  
  name="giudizio"  
  value="sufficiente">
```

```
<INPUT type="RADIO"  
  name="giudizio" value="buono">
```

```
<INPUT type="RADIO" name="giudizio"  
  value="ottimo">
```

- usato per selezionare una tra alcune scelte
- tutte le scelte con lo stesso “name” (altrimenti una scelta non esclude le altre)

# TYPE="SUBMIT"

```
<INPUT type="SUBMIT" value="spedisci">
```

- crea un bottone che invia il modulo al server
- la lunghezza del bottone dipende dalla lunghezza del valore di `VALUE`

# TYPE="RESET"

```
<INPUT type="RESET" value="reimposta">
```

- crea un bottone che resetta i campi del modulo
- i dati inseriti vengono eliminati
- la lunghezza del bottone dipende dalla lunghezza del valore di `VALUE`

# TYPE="IMAGE"

```
<INPUT type="IMAGE" SRC="bottone.gif">
```

- come SUBMIT, ma crea un bottone tramite l'immagine (URL) specificata in SRC



# L'elemento TEXTAREA

```
<TEXTAREA cols=40 rows=5  
  WRAP="physical" name="commento">  
</textarea>
```

- utilizzato per commenti o campi che prevedono l'inserimento di molto testo
- `WRAP="physical"` stabilisce che qualora il testo inserito superi la larghezza della finestra, venga automaticamente riportato a capo

# L'elemento SELECT

```
<SELECT size=1 cols=4 NAME="giudizio">  
  <OPTION selected Value=nessuna>  
  <OPTION value=buono> Buono  
  <OPTION value=sufficiente> Sufficiente  
  <OPTION Value=ottimo> Ottimo  
</select>
```

- Permette la creazione di menu a tendina con scelte multiple

# L'elemento FIELDSET

```
<fieldset name="giudizio">  
  <legend>Dati dello studente:</legend>  
  <input type="text" name="matricola">  
  <input type="text" name="anno-iscrizione">  
  <input type="text" name="numero-esami">  
</fieldset>
```

- permette la creazione di un gruppo di campi all'interno della form
- l'elemento <legend> permette di inserire una descrizione (o titolo) per il gruppo di campi

# Esempio di form

cognome e nome:	<input type="text"/>
data di nascita:	<input type="text"/>
CAP:	<input type="text"/>
codice fiscale:	<input type="text"/>
studente lavoratore:	<input type="checkbox"/>
foto:	<input type="button" value="Sfoggia..."/> Nessun file selezionato.
giudizio:	<input type="radio"/> sufficiente <input type="radio"/> buono <input checked="" type="radio"/> ottimo
descrizione lavoro:	<div><div></div></div>
<input type="button" value="Invia form"/>	<input type="button" value="Reset form"/>

# Codice HTML della form

```
<form action="" method="post" name="eseform"
  enctype="multipart/form-data">
<table border="1">
<tr>
<td align="right">cognome e nome:
<td><input type="text" name="nome" size="50" maxlength="50"
  required>
<tr>
<td align="right">data di nascita:
<td><input type="date" name="ddn">
<tr>
<td align="right">CAP:
<td><input type="number" name="cap" size="5" maxlength="5">
<tr>
<td align="right">codice fiscale:
<td><input type="text" name="cf" size="16" maxlength="16">
<tr>
<td align="right">studente lavoratore:
<td><input type="checkbox" name="lavoratore">
```

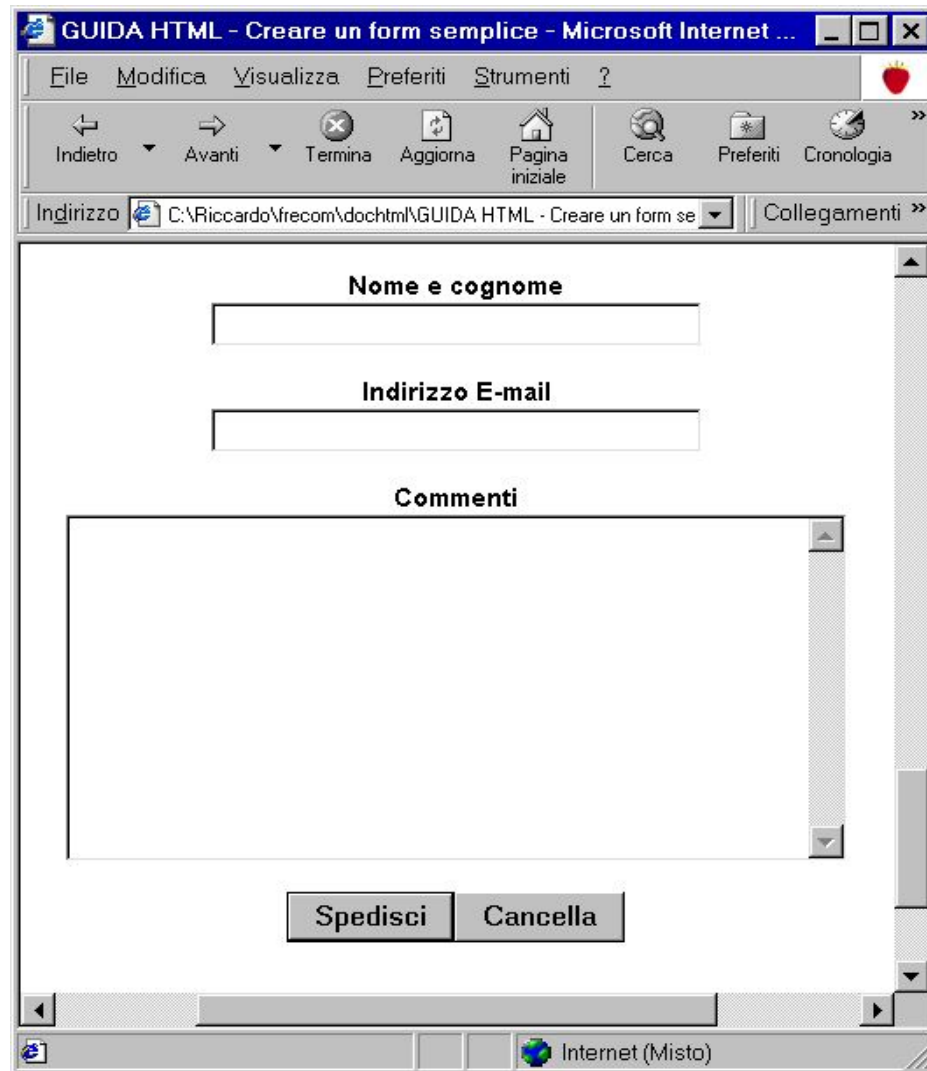
# Codice HTML della form (continua)

```
<tr>
<td align="right">foto:
<td><input type="file" name="foto">
<tr>
<td align="right">giudizio:
<td>
<input type="radio" name="giudizio"
  value="sufficiente">sufficiente
<input type="radio" name="giudizio" value="buono">buono
<input type="radio" name="giudizio" value="ottimo">ottimo
<tr>
<td align="right">descrizione lavoro:
<td><textarea name="descr" cols="60" rows="10"></textarea>
<tr>
<td align="center"><input type="submit" value="Invia form">
<td align="center"><input type="reset" value="Reset form">
</table>
</form>
```

# Esempio 2

```
<FORM
  ACTION=http://www.coder.com/code/mailform/mailform.pl.
  cgi METHOD=POST>
<INPUT TYPE=HIDDEN NAME=MAILFORM_ID VALUE="Val_7743">
<INPUT TYPE=HIDDEN NAME=MAILFORM_SUBJECT VALUE="Il mio
  primo FORM">
<INPUT TYPE=HIDDEN NAME=MAILFORM_URL
  VALUE="http://www.html.it/risposta.htm">
<B>Nome e cognome</B><BR>
<input type=text NAME=MAILFORM_NAME size=33><BR><BR>
<B>Indirizzo E-mail</B><BR>
<input type=text NAME=MAILFORM_FROM size=33><BR><BR>
<B>Commenti</B><BR>
<TEXTAREA NAME=MAILFORM_TEXT ROWS=10 COLS=42
  WRAP></TEXTAREA><BR><BR>
<INPUT TYPE=SUBMIT VALUE="Spedisci"><INPUT TYPE=RESET
  VALUE="Annulla">
</FORM>
```

# Esempio 2 (cont.)



The screenshot shows a Microsoft Internet Explorer window with the title bar 'GUIDA HTML - Creare un form semplice - Microsoft Internet ...'. The address bar displays the local file path 'C:\Riccardo\frecom\dohtml\GUIDA HTML - Creare un form se'. The form itself contains three input fields: 'Nome e cognome', 'Indirizzo E-mail', and a large 'Commenti' text area. At the bottom of the form are two buttons: 'Spedisci' and 'Cancella'. The browser's status bar at the bottom indicates 'Internet (Misto)'.

GUIDA HTML - Creare un form semplice - Microsoft Internet ...

File Modifica Visualizza Preferiti Strumenti ?

Indietro Avanti Termina Aggiorna Pagina iniziale Cerca Preferiti Cronologia

Indirizzo C:\Riccardo\frecom\dohtml\GUIDA HTML - Creare un form se Collegamenti >>

**Nome e cognome**

**Indirizzo E-mail**

**Commenti**

Spedisci Cancella

Internet (Misto)



# **5. HTML5**

# HTML5: breve storia

- **HTML5** è il successore di HTML 4 e XHTML 1.0, standardizzati dal W3C nel 1999 e 2000
- Nel 2002 il W3C decide che la futura versione 2.0 di XHTML debba rimpiazzare HTML (questo implica la non-retrocompatibilità del nuovo linguaggio con HTML 4), ed essere *document-oriented* e non *application-oriented*
- Nel 2004 si forma il consorzio WHATWG (Web Hypertext Application Technology Working Group) in opposizione al W3C e alla sua decisione di abbandonare HTML per XHTML
- Slogan di WHATWG: “Don’t break the Web”

# HTML5: breve storia (segue)

- WHATWG rilascia nel 2008 il primo draft del suo HTML5
- Nel frattempo il W3C torna sui suoi passi, e finisce per sposare la visione WHATWG. Il progetto XHTML 2.0 viene chiuso, e parte il lavoro per la standardizzazione di HTML5
- Nel 2012 WHATWG rilascia l'“HTML5 living standard”, ovvero una specifica di HTML5 che verrà lasciata evolvere continuamente (senza rilascio di versioni specifiche)

# HTML5: breve storia (segue)

- Nel 2014 il W3C rilascia il suo standard HTML5
- Nel 2016 il W3C rilascia HTML 5.1
- Nel 2017 il W3C rilascia HTML 5.2
- Nel 2019 W3C e WHATWG concordano che in future l'unico riferimento comune per i due consorzi sarà il “living standard” di WHATWG

# HTML5 vs. HTML 4

- Nuove marcature strutturali e «semantiche»
- Altri nuovi tag
- Elementi HTML editabili
- Supporto ai microdati
- Nuovi tag e attributi per le form
- Nuove API

# HTML5 vs. HTML4 (segue)

Nuove API create per supportare:

- Multimedialità
- Geolocalizzazione
- Esecuzione asincrona e parallela di script
- Comunicazioni bidirezionali tra web client e web server
- Drag and drop
- Applicazioni web offline
- Grafica
- Cronologia della navigazione
- ...

# Tag strutturali e semantici

- `<header>`
- `<footer>`
- `<section>`
- `<article>`
- `<nav>`
- `<aside>`
- `<hgroup>`
- `<mark>`
- `<time>`
- `<meter>` e `<progress>`
- `<picture>`

# Altri tag

- `<figure>`
- `<figcaption>`
- `<ruby>`
- `<wbr>`
- `<command>`
- `<menu>`
- `<details>`
- `<summary>`
- `<keygen>`
- `<output>`



# DOCTYPE

In HTML5 il DOCTYPE non fa più riferimento a una DTD, e viene semplificato nel modo seguente:

```
<!DOCTYPE html>
```

# Modificare il contenuto di una pagina

I seguenti nuovi attributi globali permettono di dichiarare elementi del documento HTML modificabili da utente:

- contenteditable
- contextmenu
- data-\* (attributi definibili da utente)
- draggable
- hidden
- spellcheck

# Nuove API

- HTML5 aumenta significativamente le API messe a disposizione degli script
- Gli obiettivi sono:
  - In generale, accrescere le possibilità offerte alla programmazione lato client
  - Standardizzare alcuni tipi più comuni di operazioni effettuate lato client
  - Aggiornare le API alle necessità dei media agent più recenti (smartphone, tablet)

# Nuove API

- Gestione di risorse e flussi audio e video
- Accesso off-line alle applicazioni
- Estensione delle capacità di comunicazione, sia verso il web server che verso altre applicazioni
- Esecuzione di azioni in background
- Estensione del concetto di cookie (salvataggio di informazioni sul dispositivo dell'utente)
- Gestione della cronologia della navigazione

# Nuove API (segue)

- Text editing
- Gestione del «drag and drop»
- Generazione di oggetti grafici 2D/3D
- Gestione di informazioni multimediali generate dall'utente (ad esempio mediante webcam e microfono)

# Offline API

- permettono di salvare copie locali di un insieme di risorse, allo scopo di permettere al browser di eseguire applicazioni anche in modalità offline
- l'elenco delle risorse da salvare è contenuto in un file chiamato **manifest** (MIME type 'text/cache-manifest')
- L'attributo manifest del tag html permette di dichiarare il file manifest associato al documento HTML
- La cache costituita dalle risorse elencate nel file manifest è gestita da apposite API (associate all'oggetto corrispondente alla proprietà applicationCache dell'oggetto window)

# WebStorage API

- Estendono le capacità di memorizzazione dei cookies
- Oggetti **localStorage** e **sessionStorage** (accessibili come array associativi)
- Possono memorizzare solo stringhe (testo): per memorizzare oggetti arbitrari occorre serializzarli (ad esempio tramite JSON)

# WebSocket API

- Permettono di instaurare una connessione dati bidirezionale tra web client e web server
- La creazione di un nuovo oggetto WebSocket crea una connessione con un server (la cui url va specificata nel costruttore)
- La funzione associata all'event handler onmessage viene eseguita quando dal server arriva un messaggio
- Il metodo send(x) invia il testo x al server



# Canvas

- elemento `<canvas>` per dichiarare una zona della pagina su cui è possibile disegnare, tramite nuove API
- si può disegnare una canvas in un contesto 2D o in un contesto 3D
- le API per disegnare (in contesto 2D) si dividono in
  - metodi path (linee, archi,...)
  - metodi modificatori (rotazioni, traslazioni,...)
  - metodo `drawImage` (disegna un'immagine)
  - metodi per scrivere testo
  - metodi per scrivere singoli pixel

# Geolocation API

Servono a gestire dati geospaziali, tipicamente la posizione (anche se questa è effettivamente disponibile solo su alcuni user agent)

La proprietà geolocation dell'oggetto navigator ha due metodi per ottenere la posizione corrente:

- **getCurrentPosition**
- **watchPosition**

(il secondo metodo differisce dal primo perché restituisce una nuova posizione ogni volta che questa cambia)

# Audio/Video e nuove API

- HTML5 permette la gestione nativa (ovvero senza ricorrere a plug-in del browser) di contenuti multimediali
- tag **<video>**: permette di inserire un contenuto video
- tag **<audio>**: permette di inserire un contenuto audio
- il formato dei video e degli audio supportati dipende dai browser, tuttavia esistono dei formati di riferimento (mp4, webm, ogg per i video)
- esistono delle nuove API per video e audio che permettono la gestione di questo tipo di risorse da script

# Form

- Nuovi attributi ed input type per le form sono stati introdotti in HTML5
- L'obiettivo principale è quello di permettere di definire le più comuni forme di validazione di form lato client direttamente nel documento HTML (cioè senza bisogno di aggiungere codice JavaScript)
- Sono stati inoltre aggiunti tipi di elementi utili soprattutto nei nuovi media agent (smartphone e tablet)

# Autofocus, placeholder, form

Nuovi attributi:

- **autofocus** (booleano): all'apertura della form, il focus va sull'elemento che ha dichiarato autofocus
- **placeholder** (per elementi input o textarea): valore che all'apertura della form compare sul campo editabile (N.B.: non corrisponde ad un valore (value) iniziale del campo, viene solo visualizzato all'inizio)
- **form**: attributo che permette di associare un elemento ad una form. E' così possibile, ad esempio, dichiarare un campo che appartiene a più form, o dichiarare un campo all'esterno di un elemento form

# Required, autocomplete

- **required** (booleano): rende obbligatoria la compilazione dell'elemento al momento del submit della form a cui l'elemento appartiene
- **autocomplete**: attributo che può assumere due valori:
  - on = il browser può effettuare l'**autocompletion** di questo campo, cioè completare il campo in maniera automatica usando i valori precedentemente inseriti in questo campo
  - off = il browser non può fare l'autocompletion
  - se autocomplete non viene assegnato, viene usato il default del browser (di solito è on)

# Multiple, pattern

- **multiple** (booleano): permette al campo di accettare valori multipli
- **esempio:**

```
<form action="demo_form.asp">  
  Select images: <input type="file"  
name="img"  multiple>  
  <input type="submit">  
</form>
```

- **pattern:** viene assegnato ad una espressione regolare; i valori del campo devono appartenere al linguaggio denotato dall'espressione regolare

# Min, max, step

- **min** = valore minimo ammesso
- **max** = valore massimo ammesso
- **step** = intervallo tra un valore ammesso e il successivo
- **novalidate** (booleano): se dichiarato sull'elemento form, indica che **non** verrà effettuata la validazione degli elementi di quella form



# Nuovi input type

I seguenti input types permettono di gestire informazioni testuali di tipo specifico:

- tel
- search
- url
- email

# Nuovi input type (segue)

- color: permette la selezione di un colore
- number: permette l'inserimento di un numero
- range: permette l'inserimento di un numero attraverso uno slider (cursore orizzontale)

# Nuovi input type (segue)

Per gestire le date sono stati introdotti i seguenti input types:

- datetime
- datetime-local
- date
- month
- week
- time

# Il tag <datalist>

- Permette di definire un campo testuale sul quale il browser può effettuare **autocompletion** usando un insieme predefinito di valori (l'utente però è libero di scrivere un valore al di fuori dell'elenco predefinito)
- esempio:

```
<input type="text" name="giudiz" list="listagiudizi">  
  <datalist id="listagiudizi">  
    <option value="insufficiente">  
    <option value="sufficiente">  
    <option value="discreto">  
    <option value="buono">  
    <option value="ottimo">  
  </datalist>
```

# Supporto ai microdati

- i microdati servono a creare un tagging «semantico» di porzioni del documento
- sono basati vocabolari che definiscono identificatori di proprietà relative ad un (micro-)dominio di interesse
- si utilizzano gli attributi HTML `itemscope`, `itemtype` e `itemprop`

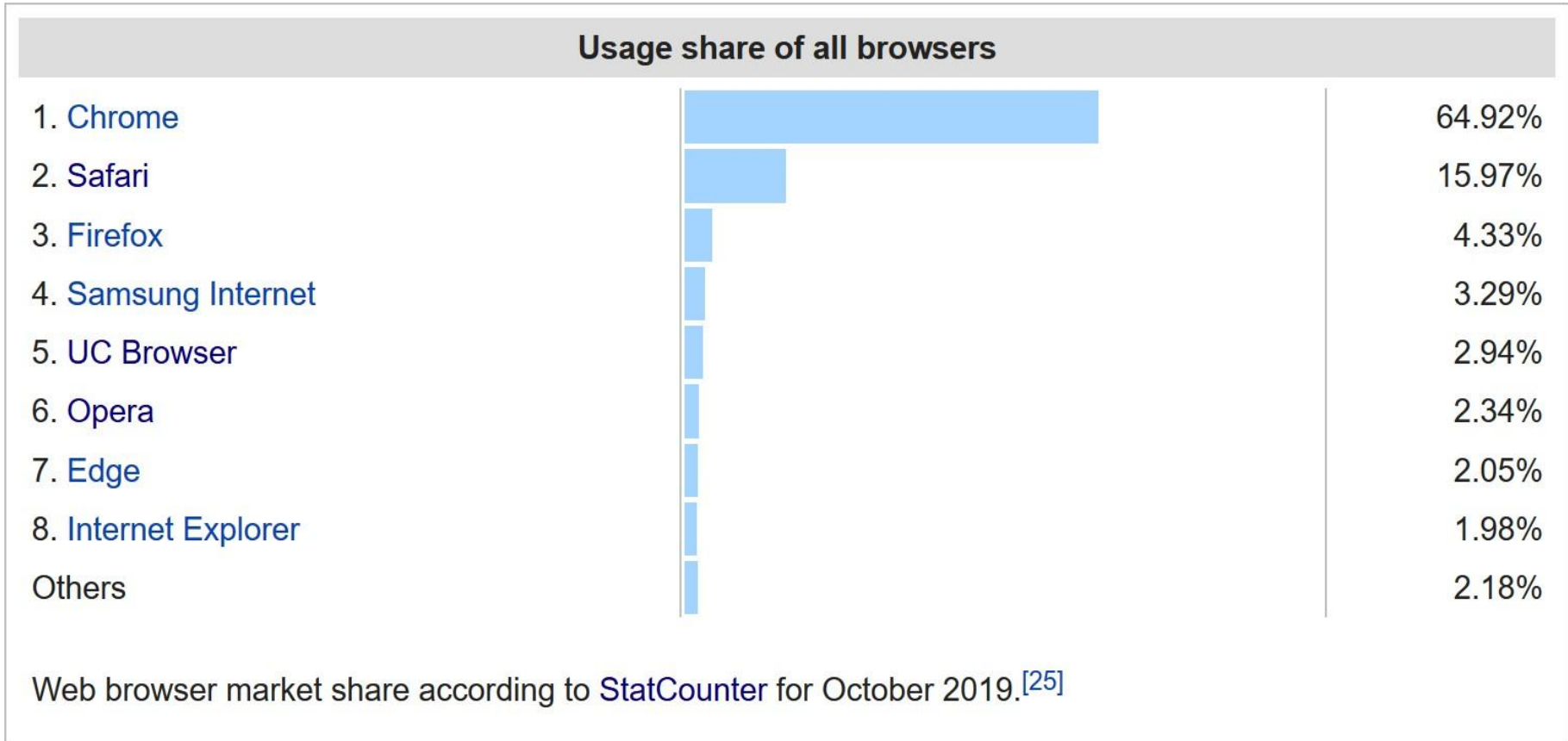
# Esempio: il vocabolario Person

Property	Description
name (fn)	Name.
nickname	Nickname.
photo	An image link.
title	The person's title (for example, Financial Manager).
role	The person's role (for example, Accountant).
url	Link to a web page, such as the person's home page.
affiliation (org)	The name of an organization with which the person is associated (for example, an employer). If fn and org have the exact same value, Google will interpret the information as referring to a business or organization, not a person.
friend	Identifies a social relationship between the person described and another person.
contact	Identifies a social relationship between the person described and another person.
acquaintance	Identifies a social relationship between the person described and another person.
address (adr)	The location of the person. Can have the subproperties street-address, locality, region, postal-code, and country-name.

# Microdati in HTML5: esempio

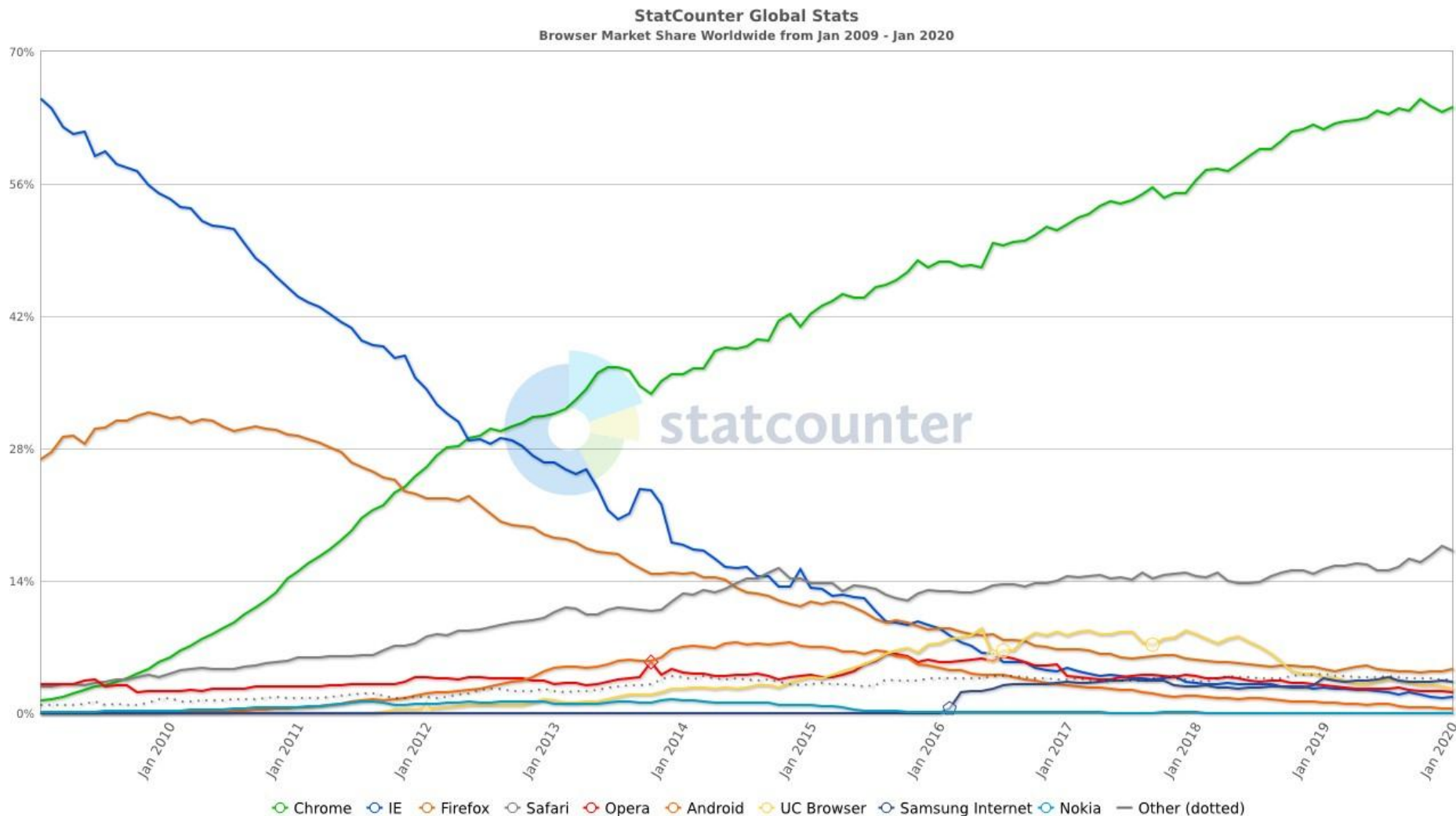
```
...  
<div itemscope  
  itemtype="http://datavocabulary.org/Person">  Benvenuti  
  nella home page di  
  <span itemprop="name">Mario Rossi</span>,  
  <span itemprop="title">professore associato</span> alla  
  <span itemprop="affiliation">Sapienza Università di  
  Roma</span>.  
</div>  
...
```

# I web browser più diffusi





# Diffusione dei web browser, 2009-2020



# Riferimenti

- Raccomandazioni ufficiali W3C su HTML:
  - [www.w3.org](http://www.w3.org)
  - <http://www.w3.org/TR/html5/>
  - <https://www.w3.org/TR/html51/>
  - <https://www.w3.org/TR/html52/>
- HTML living standard (WHATWG):
  - <https://html.spec.whatwg.org/multipage/>
- Guide (libri) su HTML/HTML5:
  - es.: Jennifer Niederst Robbins: Learning Web Design: A Beginner's Guide to Html, Css, Javascript, and Web Graphics. 5th edition. O'Reilly editore, 2018. ISBN 978-1491960202 (in inglese)

# Riferimenti

- Guida online per HTML e altre tecnologie Web:
  - [www.w3schools.com](http://www.w3schools.com)
- Sito italiano per HTML e altre tecnologie Web:
  - [www.html.it](http://www.html.it)
- Microdati, data vocabulary:
  - <http://www.schema.org>
- Statistiche sull'utilizzo dei web browser:
  - <http://gsa.statcounter.com/global-stats/browser-market-share>