Sapienza Università di Roma
Corso di laurea in Ingegneria Informatica e Automatica

Linguaggi e tecnologie per il Web

a.a. 2023/2024

Parte 1
HTML

Lorenzo Marconi
Dipartimento di Ingegneria informatica, automatica e gestionale
Sapienza Università di Roma

Sommario

1. World Wide Web
2. Ipertesti 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. 151.100.59.104
- DNS:
  - l'indirizzo del computer è una stringa di caratteri
  - es. `www.diag.uniroma1.it`
- 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

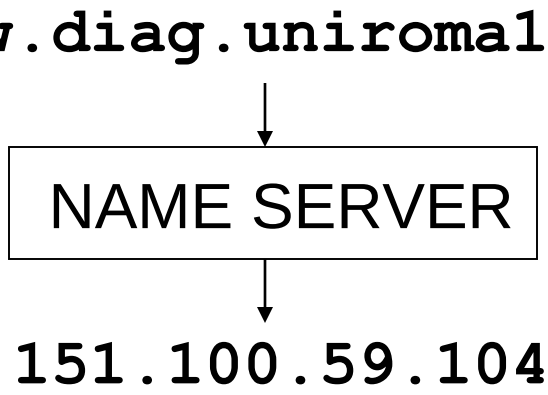
Name Server

- 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.diag.uniroma1.it/index.php`**



- **`index.php`** 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. Ipertesti e HTML

Ipertesti

Ipertesto = 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>
```

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) `<nome-tag>`
  - tag di chiusura (marcatore finale) `</nome-tag>`

Attributi dei tag (1)

- ogni occorrenza di tag (di apertura) può contenere assegnazioni di **attributi**
- assegnazione: `nome-attributo="valore"`
  - es. `<img src="pic.jpg" alt="testo">`
- gli attributi assegnabili per ogni tag sono finiti (ad eccezione di `data-*`)
- alcuni attributi sono assegnabili con un insieme finito di valori
  - caso particolare: attributi **booleani**

Attributi dei tag (2)

- struttura del tag con attributi:

```
<nome-tag attributo1="valore1"
          attributo2="valore2"
          attributoBooleano …>
```

- esistono attributi **globali**
- alcuni attributi sono **obbligatori** (vanno assegnati)

Attributi globali

Tra i più usati / importanti:

- ID (non confondere con NAME)
- CLASS
- STYLE
- TITLE
  - permette di specificare una informazione associata all'elemento
  - i browser visualizzano tale informazione (pop-up) quando il puntatore del mouse passa sopra al link

Elementi e tag

- Elemento = insieme formato da tag di apertura, testo e tag di chiusura corrispondente
- Esempio di elemento (font):

```
<img src="pic.jpg" align="left">
```

- In genere si usa il termine “tag” erroneamente, per indicare un elemento (composto da due tag)

Per HTML i tag sono case-insensitive (es. `<img>` e `<IMG>` 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
- È 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 e immagini
- tabelle
- frame

Parte intestazione (1)

- 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 (2)

Elementi principali:

- DOCTYPE
- HTML
- HEAD
- TITLE
- META

Parte intestazione (3)

```
<!DOCTYPE html>

<html>

<head>

<meta name="keywords" content="HTML, parte
  intestazione, meta-informazione">

<meta name="author" content="Lorenzo Marconi">

<meta name="generator" content="Blocco note di
  Windows 11">

<title>Pagina web di prova</title>

</head>

…

</html>
```

Parte intestazione (4)

DOCTYPE:

```
<!DOCTYPE html>
```

- fornisce l'informazione sul tipo di documento
- deve essere il primo elemento ad aprire il documento
- è obbligatorio (in HTML5)
- (nelle versioni precedenti a HTML5 doveva indicare una DTD, dichiarazione di tipo di documento)

Parte intestazione (5)

META:

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

<meta name="author" content="Lorenzo Marconi">

<meta name="generator" content="Blocco note di
  Windows 11">
```

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

Parte intestazione (6)

META:

- Attributi supportati:
  - `charset`: per specificare la codifica del documento (es. `charset="UTF-8"`)
  - `name` : per specificare la codifica del documento (es. `name="viewport"`)
  - `http-equiv` : per simulare l'HTTP response header (es. `http-equiv="content-security-policy"` o `http-equiv="content-type"`)
  - `content` : per specificare un valore associato a `name` o `http-equiv`

Parte intestazione (7)

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)

Parte intestazione (8)

Altri tag:

- `<style>`: per applicare uno stile definito localmente
- `<link>`: per specificare un collegamento con un'altra risorsa (es. favicon, foglio di stile o versione alternativa di un documento)
- `<script>`: per specificare script da eseguire lato client
- `<base>`: per specificare l'URL di base per i link nella pagina e/o le azioni di default per aprirli

HTML di base

- intestazione
- **formattazione testo**
- link
- oggetti e immagini
- 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»)

Tag contenitori

- `<div>…</div>` contenitore di tipo **block**
- `<span>…</span>` contenitore di tipo **inline**
- entrambi possono essere usati per applicare delle proprietà (es. classi CSS) ad un certo contenuto
- non supportano attributi che non siano globali

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 enfatizzazione del testo
- Non confondere con <head> e <header>!

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

- <FONT> è un tag fisico
- **deprecato** da W3C: non dovrebbe mai essere usato (al suo posto vanno usate le proprietà di CSS)
- 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 <PRE>
- permette di visualizzare il testo esattamente come è scritto (“preformattato”) nel file sorgente HTML
  - viene usato un font a larghezza fissa ("fixed-width")
  - vengono preservati tutti gli spazi e le andate a capo

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;
    I5    = 24
```

Tag logici (1)

- <ADDRESS>
  - marcatura usata per indirizzi (mail, email, telefono,…)
- <Q> e <BLOCKQUOTE>
  - usati per inserire nel testo citazioni da un altro testo o autore (rispettivamente brevi o lunghe)
- <CITE>
  - usato per la fonte della citazione
- <EM> e <STRONG>
  - “enfatizzano” il testo all’interno del tag
- <VAR>, <KBD>, <CODE> e <SAMP>
  - utilizzati per variabili, testo inserito da tastiera, righe di codice di programmazione e output di un programma

Tag logici (2)

- <ABBR> e <ACRONYM>
  - marcature usate per abbreviazioni e acronimi
- <DEL> e <INS>
  - usati per indicare parti di testo cancellate o inserite nel documento (verranno indicate sbarrate e sottolineate, risp.)
- <DFN>
  - usato per dare una definizione
- <EM> e <STRONG>
  - “enfatizzano” 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
-
 (break)
  - interruzione di riga ("a capo")
- <HR> (horizontal row)
  - aggiunge una riga orizzontale al testo
  - usato per separare parti di testo

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

Liste annidate

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>
```

Reso come:

1. gruppo di nomi
   - nome a
   - nome b
2. gruppo di nomi:
   - nome c
   - nome d
   - nome e

HTML di base

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

Link ipertestuali

Tag <A> (anchor)

- permette l'inserimento di link ipertestuali all'interno del documento
- non confondere con <link>!
- 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>
```

Anchor

```
...
<p id="par1">un certo paragrafo del documento</p>
...
<a href="#par1">vai all'anchor "par1" nel
documento</a>
...
```

Con l'anchor par1 si può anche accedere direttamente dall'esterno del documento:

```
<a href="www.diag.uniroma1.it/index.html#par1">
vai al paragrafo "par1" del documento index.html
   del sito diag.uniroma1.it </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:

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

HTML di base

- intestazione
- formattazione testo
- link
- **oggetti e immagini**
- 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, PNG)

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

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 disabilitate

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

Mappe cliccabili

- Una immagine può essere associata ad un link
- es. `<a href="pippo.htm"><img src="immagine.gif"></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 (es. ISMAP)
  - lato client (USEMAP)

Mappe cliccabili (lato client)

```
<IMG SRC="img1.gif" alt="Una bella mappa"
  usemap="#immagine1">

<map name="immagine1">
<area shape="rect" coords="0,0,250,150"
  href="page1.html" alt="Page 1">
<area shape="circle" coords="300,150,50"
  href="page2.html" alt="Page 2">
<area shape="poly"
  coords="350,50,450,150,400,200,300,100"
  href="page3.html" alt="Page 3">
<area shape="default" href="#">
</map>
```

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

Oggetti

- Tag `<object>` e `<embed>`
  - Permettono l'inclusione di oggetti (risorse) generici nel documento HTML
  - Esempio:

```
<embed type="text/html"
  src="page1.html" width="600"
  height="400">
```

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 ©
- le entità si possono anche rappresentare mediante i loro codici Unicode (in decimale o esadecimale (es:. `&#123;` `&#xDE9F;`)

Simboli speciali

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

Il simbolo “<”

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

HTML di base

- intestazione
- formattazione testo
- link
- oggetti e immagini
- **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, TBODY, TFOOT

L’elemento <table>

Racchiude tutta l’informazione relativa ad una **tabella**

Tag da usare dentro `<table>`:

- `<tr>` per le righe
- `<td>` e `<th>` per le celle (usare dentro `<tr>`)
- `<caption>` per la didascalia
- `<thead>`, `<tbody>` e `<tfoot>` per suddividere il contenuto della tabella
- `<colgroup>` per raggruppare delle colonne

L’elemento <TD>

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

Attributi (opzionali):

- `colspan` (num. colonne occupate dalla cella)
- `rowspan` (num. righe occupate dalla cella)
- `headers` (header a cui si riferisce la cella)

L’elemento <TR>

- Divide le celle in righe
- Esempio:

```
<TABLE>
<TR>
   <TD>cella 1</TD>
   <TD>cella 2</TD>
   <TD>cella 3</TD></TR>
<TR>
   <TD>cella 4</TD>
   <TD>cella 5</TD>
   <TD>cella 6</TD></TR>
</TABLE>
```

cella 1	cella 2	cella 3
cella 4	cella 5	cella 6

Esempio

```
<TABLE>
<TR>
   <TD colspan=2>A</TD>
   <TD>B</TD>
</TR>
<TR>
   <TD>C</TD>
   <TD rowspan=2>D</TD>
   <TD>E</TD>
</TR>
<TR>
   <TD>F</TD>
   <TD>G</TD>
</TR></TABLE>
```

<table>
<tr><td colspan="2">A</td><td>B</td></tr>
<tr><td>C</td><td rowspan="2">D</td><td>E</td></tr>
<tr><td>F</td><td>G</td></tr>
</table>

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>
<TR>
  <TH>nome</TH>
  <TH>cognome</TH>
  <TH>età</TH></TR>
<TR>
  <TD>Mario</TD>
  <TD>Rossi</TD>
  <TD>36</TD></TR>
<TR>
  <TD>Paola</TD>
  <TD>Bianchi</TD>
  <TD>32</TD></TR>
</TABLE>
```

nome	**cognome**	**età**
Mario	Rossi	36
Paola	Bianchi	32

L’elemento <CAPTION>

- Permette di associare una didascalia alla tabella
- esempio:

```
<TABLE>
<CAPTION>Elenco degli impiegati:</CAPTION>
<TR>
   <TH>nome</TH>
   <TH>cognome</TH>
   <TH>età</TH></TR>
<TR>
   <TD>Mario</TD>
   <TD>Rossi</TD>
   <TD>36</TD></TR>
<TR>
   <TD>Paola</TD>
   <TD>Bianchi</TD>
   <TD>32</TD></TR>
</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>
<THEAD>
   <TR><TH>nome</TH>
       <TH>cognome</TH>
       <TH>et&agrave;</TH></TR>
</THEAD>
<TBODY>
   <TR>
       <TD>Mario</TD>
       <TD>Rossi</TD>
       <TD>36</TD></TR>
   <TR>
       <TD>Paola</TD>
       <TD>Bianchi</TD>
       <TD>32</TD></TR>
</TBODY>
</TABLE>
```

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)

HTML di base

- intestazione
- formattazione testo
- link
- oggetti e immagini
- 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)
- Nelle versioni precedenti di HTML erano presenti diversi elementi per organizzare la finestra in frame
- Nel seguito vedremo l'unico elemento tuttora supportato, l'elemento iframe

L’elemento <iframe>

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

```
<iframe src="http://www.uniroma1.it"
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.diag.uniroma1.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

Tag da usare in un modulo

Vengono definiti tramite gli elementi:

- `INPUT` (campi editabili, checkbox, radio, ecc.)
- `TEXTAREA` (area di testo)
- `SELECT` (creazione di menu di opzioni)
  - da usare in combinazione con `OPTION` (e eventualmente `OPTGROUP`)
- `BUTTON` (simula un tasto da premere)
- `LABEL` (etichetta associata a un altro elemento)
- `OUTPUT` (risultato di una certa computazione)
- `FIELDSET` (insieme di campi)
  - da usare in combinazione con `LEGEND`

L’elemento INPUT

- Permette l’inserimento di campi editabili e/o modificabili nel modulo
- Attributi principali:
  - **`TYPE`**: determina il tipo di campo
  - **`NAME`**: chiave da passare al server
  - **`VALUE`**: valore da passare al server
  - `MINLENGTH` / `MAXLENGTH`
  - `SIZE`: larghezza specificata in numero di caratteri

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 consegnare al server delle informazioni non visibili dall'utente (se non nel sorgente della pagina)
- Il suo uso dipende dal tipo di CGI associato al modulo

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

TYPE=“FILE”

- Usato per caricare un file

Attributi associati:

- `ACCEPT`
  - serve filtrare i file per estensione o media type
- `MULTIPLE` (da HTML5)
  - attributo booleano per abilitare caricamenti multipli

L’elemento TEXTAREA

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

Attributi associati:

- MAXLENGTH
- COLS
- ROWS

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:	
data di nascita:	
CAP:	
codice fiscale:	
studente lavoratore:	☐
foto:	Sfoglia... Nessun file selezionato.
giudizio:	○ sufficiente ○ buono ◉ ottimo
descrizione lavoro:	
Invia form	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">
<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">
```

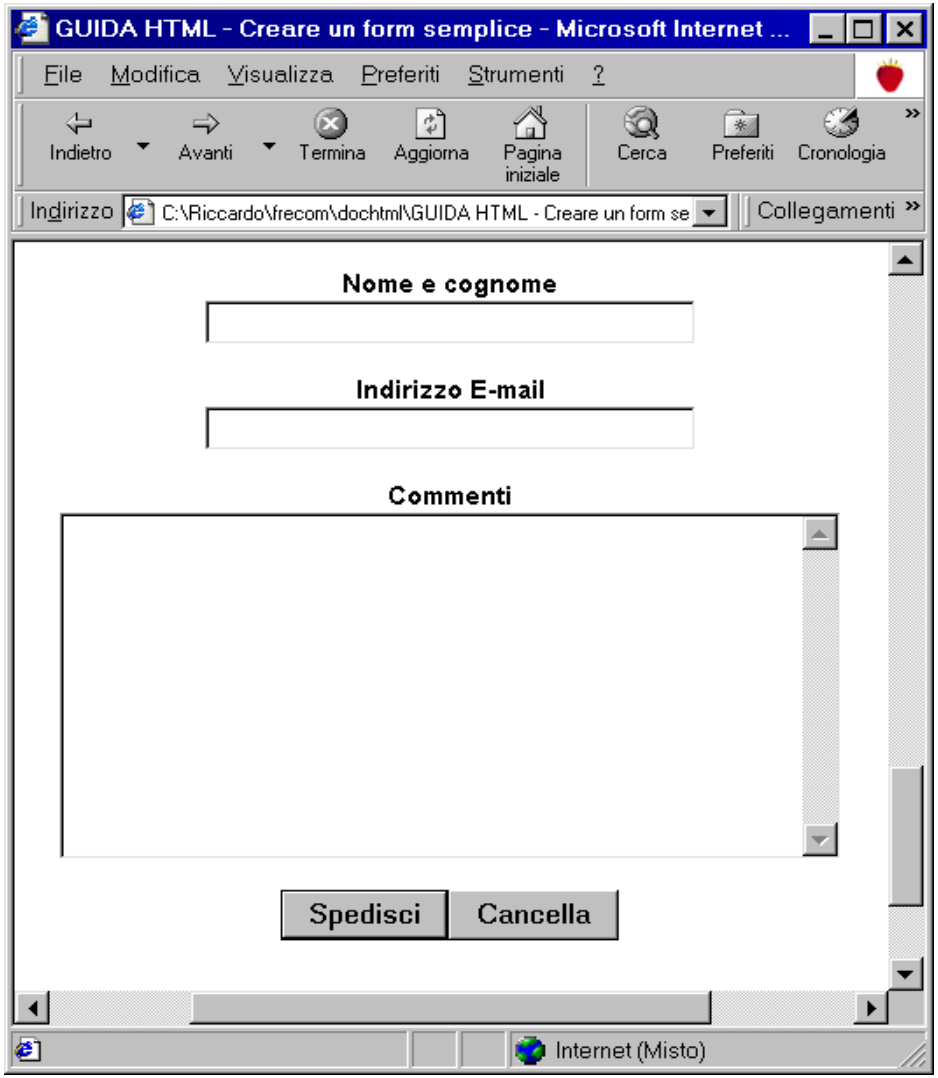
Codice HTML della form (continua)

```
<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="Cancella">
</FORM>
```

Esempio 2 (cont.)



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 futuro 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>
</input>
```

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

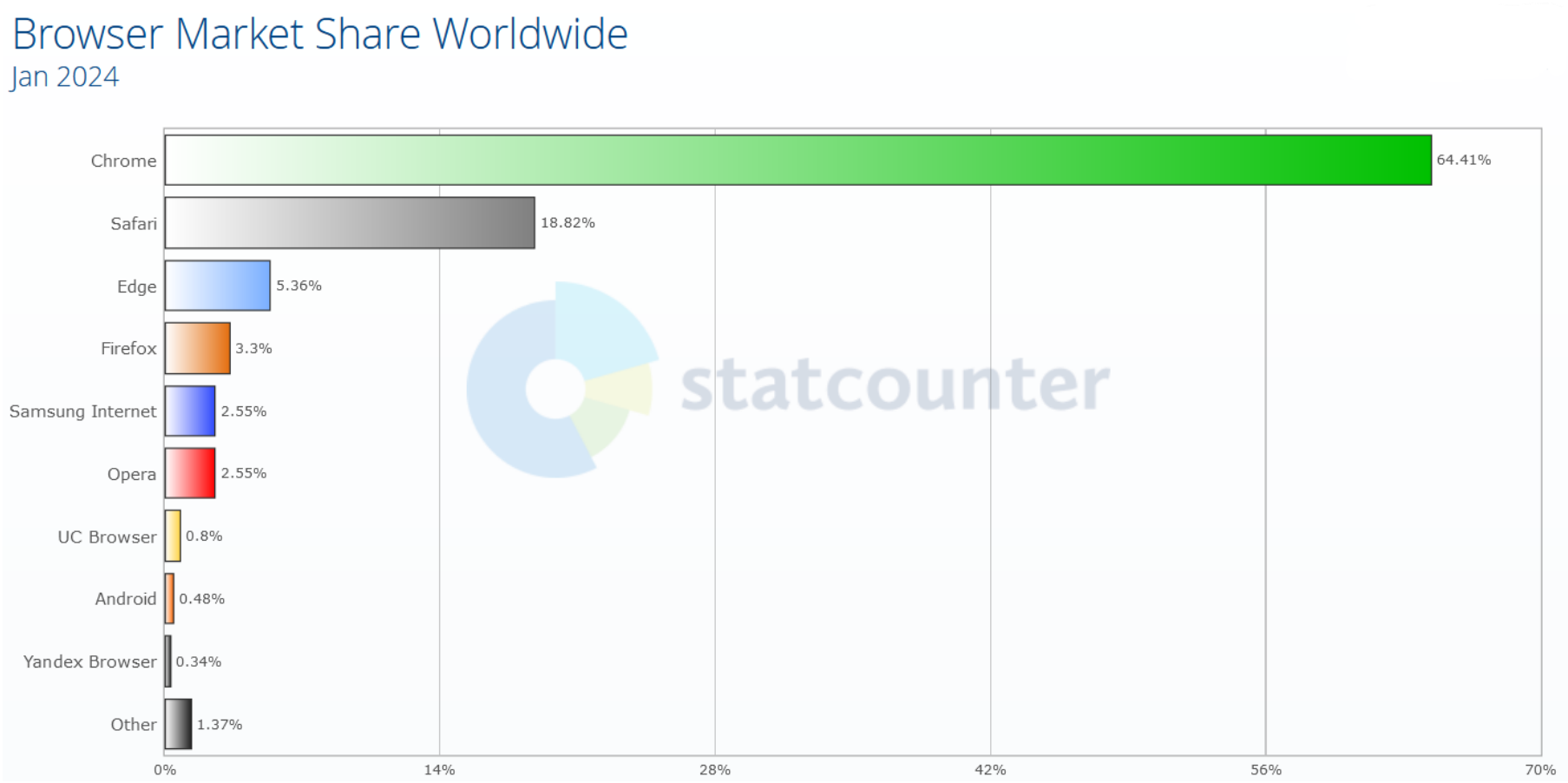
Esempio: il vocabolario Person

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

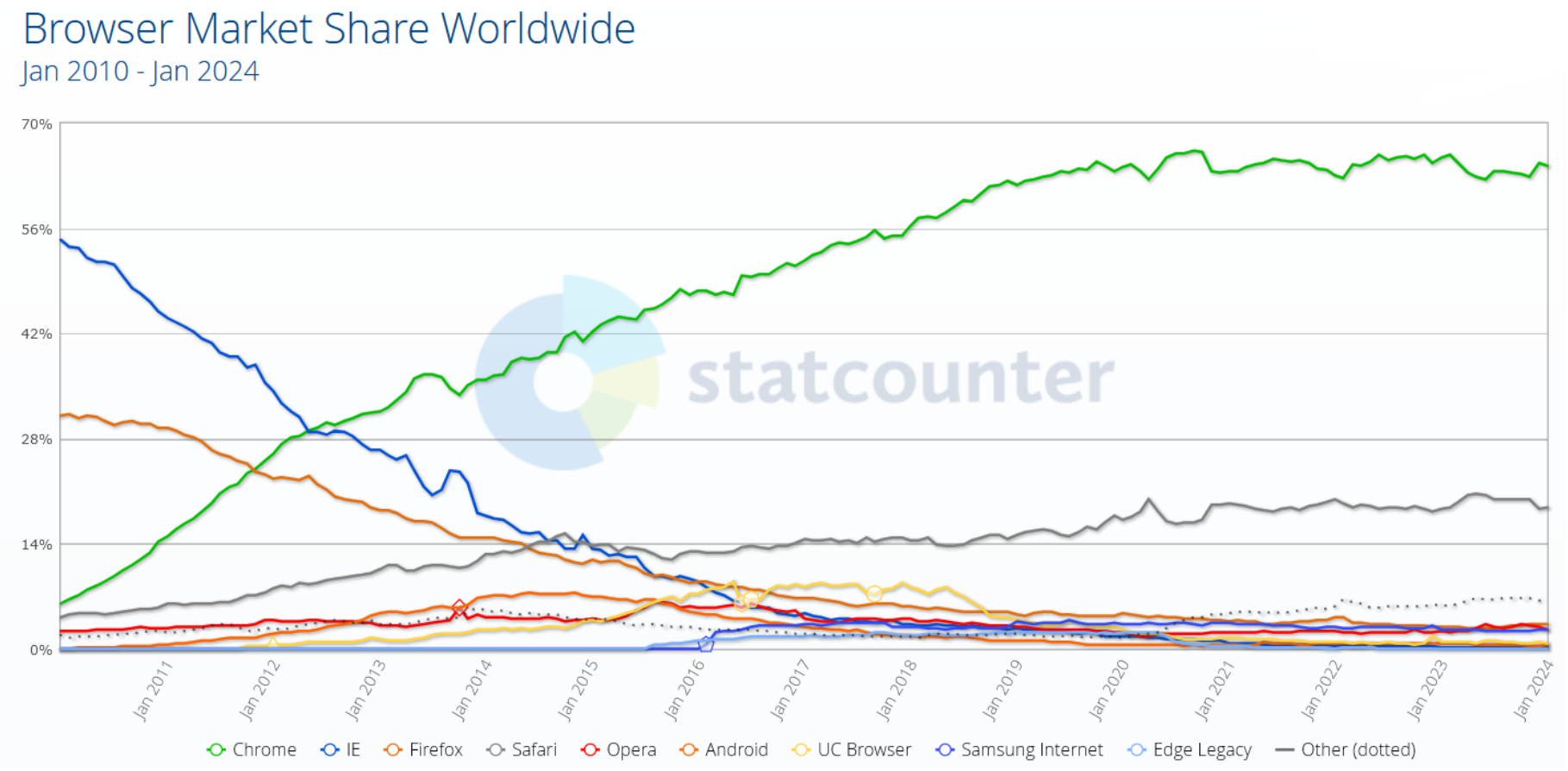
```
…
<div itemscope itemtype="http://datavocabulary.org/Person">
Benvenuti nella home page di
<span itemprop="name">Lorenzo Marconi</span>,
<span itemprop="title">docente a contratto</span> alla <span
itemprop="affiliation">Sapienza Università di Roma</span>.
</div>
…
```

I web browser più diffusi (gennaio 2024)



(fonte: gs.statcounter.com)

Diffusione dei web browser, 2009-2024



(fonte: gs.statcounter.com)

Riferimenti

- Raccomandazione ufficiale W3C su HTML:
  - http://www.w3.org/TR/html5/
    (attualmente è equivalente al link successivo)
- 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
- Sito italiano per HTML e altre tecnologie Web:
  - www.html.it
- Microdati, data vocabulary:
  - www.schema.org