Hypertext Markup Language

Giuseppe Della Penna

Università degli Studi di L'Aquila
giuseppe.dellapenna@univaq.it
http://people.disim.univaq.it/dellapenna

> *Questo documento si basa sulle slide del corso di Web Engineering, riorganizzate per migliorare l'esperienza di lettura. Non è un libro di testo completo o un manuale tecnico, e deve essere utilizzato insieme a tutti gli altri materiali didattici del corso. Si prega di segnalare eventuali errori o omissioni all'autore.*

- 1. Introduzione a HTML
  - 1.1. Hypertext Markup Language
  - 1.2. Nozioni di base
  - 1.3. Compatibilità Cross-Browser
- 2. Struttura di base dei documenti HTML
  - 2.1. Apertura di un Documento
  - 2.2. Struttura di base di un documento HTML
  - 2.3. Classificazione degli elementi
  - 2.4. Attributi standard HTML
- 3. Elementi di flusso del testo
  - 3.1. Paragrafi
  - 3.2. Interruzioni di linea
  - 3.3. Sezioni
  - 3.4. Elementi strutturali HTML5
  - 3.5. Contenitori
- 4. Elementi di formattazione del testo
  - 4.1. Formattazione semantica
  - 4.2. Formattazione di base
- 5. Liste e tabelle
  - 5.1. Liste
  - 5.2. Menu in HTML5
  - 5.3. Tabelle
- 6. Immagini e oggetti incorporati

    - 6.1. Immagini
    - 6.2. Mappe immagine
    - 6.3. Oggetti
- 7. Meta elementi
    - 7.1. Collegamenti
    - 7.2. Fogli di Stile
- 8. Elementi interattivi
    - 8.1. Collegamenti
    - 8.2. Moduli
- 9. Riferimenti

1. Introduzione a HTML

1.1. Hypertext Markup Language

La parte strutturale del web

L'HyperText Markup Language (HTML) è l'elemento costitutivo più basilare del Web, e viene usato **per definire la struttura dei suoi contenuti**. L'HTML permette di definire **ipertesti**, cioè testi che sono connessi tra loro tramite collegamenti, o link.

Tecnicamente, HTML è un **linguaggio di markup**, in cui cioè il cui contenuto testuale è strutturato tramite particolari delimitatori detti **tag**, aventi la generica sintassi "`<tag>`". Esistono tag di *apertura* (`<tag>`) e *chiusura* (`<tag>`), esattamente come le parentesi aperte "(" e chiuse ")" nel linguaggio comune.

Il testo delimitato da un tag viene chiamato **elemento**. Il tag di ogni elemento fornisce la **semantica** del suo contenuto, indicando come questo debba essere interpretato (ma non rappresentato visivamente: a questo pensano i fogli di stile CSS). I tag quindi non vengono quindi visualizzati nei browser, ma li istruiscono su come mostrare il contenuto della pagina web.

HTML (come la maggior parte dei linguaggi di markup) è facilmente manipolabile dalle macchine ma anche leggibile dagli esseri umani.

HTML è inoltre totalmente **indipendente da architetture hardware, sistemi operativi e protocolli**, ed è addirittura **utilizzato al di fuori del Web** come formato alternativo per testo strutturato simile a quello generato dai word processor, ad esempio nelle email e persino per la definizione delle interfacce utente del software.

Evoluzione

Esistono numerose versioni di HTML, che si è evoluto seguendo le esigenze del web ma anche di tutte le altre applicazioni che ne fanno uso:

- 1991 Prima versione (HTML 1.0) definita da Tim Berners-Lee
- 1995 HTML 2.0 (definita dall'HTML Working Group)
- 1997 HTML 3.2 (W3C)
- 1999 HTML 4.01 (W3C)
- 2000 XHTML 1.0 (HTML 4.01 con sintassi XML)
- 2008 Draft HTML5 (Web Hypertext Application Technology Working Group)
- 2012 HTML5 Living Standard (WHATWG)
- **2014 HTML5 (W3C)**
- 2017 HTML5.2 (W3C)
- ...

La rivoluzione di HTML 5

HTML5 **è un linguaggio di markup indipendente**, non derivato da SGML come HTML4.

La sintassi di HTML5 è compatibile con quella di HTML4 e XHTML1, ma non supporta alcune caratteristiche della sintassi HTML4 che derivavano da SGML, come le *processing instructions*.

Come dice il W3C (http://www.w3.org/TR/html5-diff): "*The HTML[5] specification reflects an effort, started in 2004, to study contemporary HTML implementations and deployed content. The specification: Defines* ***a single language called HTML which can be written in HTML syntax and in XML syntax***, *defines detailed processing models to foster interoperable implementations*, ***improves markup for documents***, ***introduces markup and APIs for emerging idioms***, *such as Web applications [...]*"

HTML5 è diventato una **W3C Recommendation** (quindi una specifica finale) il **28 ottobre 2014**, *ma è uno standard ancora in evoluzione*. È quindi sempre necessario accertarsi che una caratteristica HTML5 sia adeguatamente supportata dai browser attuali prima di includerla nei propri progetti.

> In questo documento, nella descrizione del linguaggio, saranno evidenziati i cambiamenti più importanti introdotti da HTML5.

1.2. Nozioni di base

Codifica del testo

Le pagine HTML, come tutto il Web, sono costituite soprattutto da testo, per cui è importante capire come questo può essere rappresentato su una macchina.

Un **set di caratteri** (*character set*, *charset*) definisce l'insieme di caratteri (lettere, numeri, punteggiatura, simboli,...) necessari per un particolare scopo (non necessariamente legato alla trasmissione e conservazione digitale).

Un **set di caratteri codificato** è un set di caratteri a ciascuno dei quali è stato assegnato un numero univoco. Gli elementi di un set di caratteri codificato sono anche noti come **code points**. Un code

point, quindi, rappresenta la posizione di un carattere nel set di caratteri codificato: ad esempio, il code point per la lettera "á" nel set di caratteri codificati Unicode è 225.

La **codifica dei caratteri** (*character encoding*) definisce infine il modo in cui il set di caratteri codificato (o meglio i suoi code point numerici) verranno mappati su bytes per essere salvati su supporti digitali e trasmessi in rete.

Molti standard storici di codifica dei caratteri (ad esempio gli ISO 8859, ancora largamente diffusi) utilizzano un singolo byte per ogni code point che rappresenta semplicemente la posizione del carattere nel set. Ad esempio, la "A" nel set codificato ISO 8859-1 è il 65° carattere, ed è quindi codificata per la rappresentazione in un byte con il valore di 65.

Quando si comunicano informazioni testuali è quindi necessario specificare il set in uso e il corrispondente encoding, in modo che tali caratteri possano essere rappresentati in modo affidabile.

Unicode

Lo **Unicode Consortium** ha definito un ampio set di caratteri che include tutti quelli necessari a qualsiasi sistema di scrittura nel mondo.

Lo standard Unicode è fondamentale per l'architettura del Web e dei sistemi operativi, e sta gradualmente sostituendo i set di caratteri specifici creati in passato da diverse organizzazioni, quali l'ISO.

I primi 65536 code points nel set di caratteri Unicode costituiscono il **Basic Multilingual Plane** (BMP), che include la maggior parte dei caratteri più comunemente usati. Sono inoltre disponibili circa un milione di code points aggiuntivi per **caratteri supplementari**, comprese le emoji.

Sebbene il set di caratteri Unicode sia unico, le codifiche disponibili sono più di una: **UTF-8**, **UTF-16** e **UTF-32**.

- UTF-8 utilizza **un byte** per rappresentare i caratteri nel set **ASCII**, **due byte** per i caratteri più comuni in altri alfabeti, **tre byte** per il resto del BMP e **quattro byte** per i caratteri supplementari.
- UTF-16 utilizza 2 byte per qualsiasi carattere nel BMP e 4 byte per i caratteri supplementari.
- UTF-32 utilizza 4 byte per tutti i caratteri.

Quindi, sebbene il code point per la lettera "á" nel set di caratteri Unicode sia sempre 225, in UTF-8 è rappresentato da due byte.

In HTML, e in generale su Internet, attualmente si considera come **standard lo Unicode con codifica UTF-8**.

Incredibilmente, persino l'uso malizioso dei *character encoding* Unicode può evidenziare **vulnerabilità** e permettere, ad esempio, il furto di informazioni.

- **UTF-7, originariamente definito per codificare il solo BMP, permetteva codifiche alternative di**

caratteri ASCII come "<" e ">". Le vecchie versioni di Internet Explorer potevano essere indotte a interpretare una pagina come UTF-7, e in questo caso le sequenze +ADw- e +AD4-, che le maggior parte dei validatori tratta come come testo semplice, venivano trasformate in "<" e ">" permettendo attacchi **XSS**.

- Poiché **molti caratteri di lingue diverse possono somigliarsi visivamente**, è possibile indurre un utente a navigare su un sito il cui indirizzo è visivamente simile a quello di un sito sicuro, anche se in realtà i caratteri in esso contenuti sono diversi, e quindi porteranno a un sito malevolo (attacchi di **omografia**), ad esempio usando il carattere cirillico "a" (Unicode 0430) invece di quello ASCII (Unicode 0041).

Identificatori di risorse, URI e URL

Per identificare univocamente le risorse, anche al di fuori della rete, vengono comunemente utilizzati gli "**Uniform Resource Identifier**" (URI):

```
<schema>://<authority>/<path>?<query>#<fragment>
```

- Lo *schema*, obbligatorio, fornisce informazioni sul formato e il significato del resto della URI
- L'*authority*, se presente, indica l'organizzazione a cui appartiene il nome definito dal resto della URI
- Il *path*, obbligatorio, fornisce la prima parte del nome univoco (eventualmente rispetto all'autorità) della risorsa, rappresentato in maniera gerarchica
- La *query*, se presente, fornisce un'ulteriore parte del nome, questa volta sotto forma di dati non gerarchici
- Il *fragment*, se presente, permette di puntare a una sotto-risorsa della risorsa principale

Un "**Uniform Resource Locator** " (URL) è un tipo specifico di URI che non solo identifica la risorsa, ma indica dove si trova e/o come accedervi.

```
<protocol>://<host>:<port>/<path>?<query>#<fragment>
```

- In una URL, lo *schema* identifica il **protocollo** di accesso alla risorsa (http, ftp, ...)
- L'*authority* è costituita dal FQDN dell '**host** che ospita la risorsa (o dal suo indirizzo IP) eventualmente seguito dal numero di *porta* che identifica il servizio logico a cui connettersi sull'host (se omessa, ha un default dipendente dal protocollo)
- Il **path** è una sequenza di stringhe separate da slash (/) che, come in un filesystem, identifica una risorsa attraverso il percorso logico necessario a raggiungerla (nel caso del filesystem, come raggiungere un file attraversando la gerarchia cartelle).

Rappresentazione delle risorse

Le risorse scambiate su una rete devono essere adeguatamente **rappresentate** per poter essere

comprensibili dai dispositivi che stanno dialogando. Ad esempio, quando si trasmette un'immagine non si sta trasmettendo veramente l'immagine, ma una rappresentazione di quest'ultima, codificata ad esempio tramite il formato JPEG.

Perché una comunicazione possa essere efficace, le due parti devono concordare, oltre che sul protocollo, anche sui formati da utilizzare per scambiare le risorse.

Su Internet sono presenti una gran quantità di **formati standard** per la rappresentazione di numerosi classi di risorse, come ad esempio l'HTML per gli ipertesti, JPEG, PNG, GIF, BMP, ecc. per le immagini, vari derivati dell'MPEG per audio e video, ecc.

Questi tipi sono indicati tramite i cosiddetti **media types** (noti anche come tipi **MIME**), che hanno la forma generica "*tipo/sottotipo*", ad esempio text/html, image/jpeg, audio/mpeg.

1.3. Compatibilità Cross-Browser

Modalità Standards e Quirks

I Browser supportano due modalità di rendering: *Standards* e *Quirks*.

- Lo Standards mode lavora seguendo il più possibile le specifiche del W3C, quindi in maniera (quasi) indipendente dal browser.
- Il Quirks mode segue le regole di formattazione dello specifico browser, con le sue limitazioni ed estensioni.

La modalità Quirks esiste per rendere i browser compatibili con i vecchi siti web, che erano sviluppati con codice molto *browser-dipendente*. Oggi, è *necessario* sviluppare i nuovi siti in modalità Standards.

Di default i browser usano la modalità Quirks. Per entrare in modalità Standards occorre inserire all'inizio del documento una dichiarazione specifica.

In particolare, per quel che riguarda la modalità standard HTML5:

- I documenti che usano HTML5 con **sintassi HTML** devono essere serviti col *media type* **text/html** e devono iniziare con la (pseudo) dichiarazione DOCTYPE `<!doctype html>`.
- I documenti che usano HTML5 con **sintassi XML** devono essere serviti con *media types* come **application/xml**, **application/xhtml+xml**, **text/xml**, devono contenere la dichiarazione XML e dichiarare il namespace **http://www.w3.org/1999/xhtml** nell'elemento `<html>`. Non sono invece necessarie dichiarazioni DOCTYPE se il documento è servito con il *media type* corretto.

2. Struttura di base dei documenti HTML

2.1. Apertura di un Documento

XHTML (HTML4)

```
<!--
XHTML Strict
(Content-type: application/xhtml+xml  )
-->

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DT

<html xmlns="http://www.w3.org/1999/xhtml  ">
 <head>
  <title>…</title>
 </head>
 <body> … </body>
</html>
```

```
<!--
XHTML Transitional
(Content-type: application/xhtml+xml  )
-->

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DT

<html xmlns="  http://www.w3.org/1999/xhtml  ">
 <head>
  <title>…</title>
 </head>
 <body> … </body>
</html>
```

HTML5

```
<!--
HTML5 con sintassi HTML
(Content-type: text/html)
-->

<!doctype html>
<html>
 <head>
  <meta charset="UTF-8">
  <title>…</title>
 </head>
 <body> … </body>
</html>

<!--
(è ammesso anche <!DOCTYPE html SYSTEM "about:legacy-compat">)
-->
```

```
<!--
HTML5 con sintassi XML
(Content-type: application/xhtml+xml  )
-->

<?xml version="1.0" encoding="UTF-8"?>
<html xmlns="http://www.w3.org/1999/xhtml  ">
 <head>
  <title>…</title>
 </head>
 <body> … </body>
</html>
```

2.2. Struttura di base di un documento HTML

`<html>`: elemento principale (radice)

Contenuto: `<head>`, `<body>` (entrambi obbligatori)
Attributi: lang, dir

Questo elemento deve aprire ogni documento HTML.

`<head>`: intestazione

Contenuto: `<title>` (obbligatorio), `<base>`, `<script>`, `<meta>`, `<link>`, `<style>`
Attributi: lang, dir

Contiene informazioni sul documento che solitamente non producono alcun output ma influiscono sulla logica e sulla presentazione del documento stesso.

`<body>`: corpo

Contenuto: *blocco*, `<script>`, `<ins>`, `<del>`
Attributi: standard HTML

Racchiude il vero contenuto del documento HTML.

2.3. Classificazione degli elementi

Il contenuto *inline* è rappresentato dal testo e dai seguenti elementi: `<tt>`, `<i>`, `<b>`, `<big>`, `<small>`, `<em>`, `<strong>`, `<dfn>`, `<code>`, `<samp>`, `<kbd>`, `<var>`, `<cite>`, `<abbr>`, `<acronym>`, `<a>`, `<img>`, `<object>`, `<br>`, `<script>`, `<map>`, `<q>`, `<sub>`, `<sup>`, `<span>`, `<bdo>`, `<input>`, `<select>`, `<textarea>`, `<label>`, `<button>`

Gli elementi che costituiscono il contenuto *blocco* sono: `<p>`, `<h1>`...`<h6>`, `<ol>`, `<ul>`, `<pre>`, `<dl>`, `<div>`, `<noscript>`, `<blockquote>`, `<form>`, `<hr>`, `<table>`, `<fieldset>`, `<address>`

Il contenuto di tipo *flusso* è dato dall'unione di *inline* e *blocco*

Questa classificazione è stata completamente cambiata HTML5, ma le categorie di elementi appena esposte sono compatibili con quelle nuove.

Nuova classificazione in HTML5

La classificazione usata da HTML5 rispecchia il ruolo dell'elemento, piuttosto che il suo aspetto (blocco, inline).

- Contenuto *Metadata*: `<link>`, `<script>`, ecc.
- Contenuto *Phrasing*: `<span>`, `<img>`, ecc.
  Molto simile alla categoria "inline" di HTML4.
- Contenuto *Heading* (intestazione): `<h1>`, `<h2>`, ecc.
- Contenuto *Sectioning* (sezionamento): `<aside>`, `<section>`, ecc.
- Contenuto *Flow* (flusso): `<span>`, `<div>`, ecc.
  Corrisponde più o meno all'omonimo concetto di HTML4, e include i "Phrasing".
- Contenuto *Embedded* (incorporato): `<img>`, `<iframe>`, `<svg>`, ecc.
- Contenuto *Interactive* (interattivo): `<a>`, `<button>`, `<label>`, ecc.

2.4. Attributi standard HTML

- `id`: ID unico
  Utilizzato per riferirsi all'elemento negli script
- `class`: lista di classi
  Utilizzati per attribuire uno o più stili globali all'elemento. La lista è delimitata da spazi
- `style`: informazioni di stile

Utilizzato per fornire uno stile CSS specifico all'elemento

- `title`: titolo informativo
  Molti browser lo renderizzano come tooltip dell'elemento
- `lang`: codice lingua
  Codici linguistici come da standard I18N, ad es. "it" o "en-us"
- `dir`: direzione di scrittura
  rtl (destra-a-sinistra) o ltr (sinistra-a-destra)
- `onclick`, `ondblclick`, `onmousedown`, `onmouseup`, `onmouseover`, `onmousemove`, `onmouseout`, `onkeypress`, `onkeydown`, `onkeyup`: gestori eventi
  Utilizzati per associare degli script agli eventi corrispondenti

Nuovi attributi HTML5

- `contenteditable`: indica che il contenuto dell'elemento è modificabile.
- `spellcheck`: indica che il contenuto dell'elemento dovrebbe essere sottoposto a controllo ortografico.
- `translate`: indica che il contenuto dell'elemento dovrebbe essere tradotto (dai traduttori automatici).
- `hidden`: marca l'elemento come non rilevante
- `data-*`: attributi estesi definiti dall'utente.
  Qualsiasi nuovo attributo il cui nome è prefissato da "data-" può essere aggiunto ad ogni elemento. Questi attributi vengono usati per associare all'elemento dati utilizzabili dagli script nella pagina.
- `role`, `aria-*` : utilizzati per fornire informazioni specifiche di accessibilità alle *rich user interfaces* (ARIA= *Accessible Rich Internet Applications*).
  - `role` serve a indicare il ruolo dell'elemento nella pagina, che può prescindere dal nome del suo tag (ad es. "*checkbox*", " *grid*", "*link*", "*menu*", "*navigation*", "*form*" o " *img*": vedi www.w3.org/TR/wai-aria/roles).
  - I vari attributi `aria-*` sono usati per supportare questi ruoli con ulteriori informazioni semantiche. Ad esempio, aria-checked può essere usato con elementi il cui role è "checkbox" per identificare il loro stato (vedi www.w3.org/TR/wai-aria/states_and_properties).

3. Elementi di flusso del testo

3.1. Paragrafi

`<p>`: paragrafo

Contenuto: *inline*
Attributi: standard HTML

Il flusso di testo nei documenti HTML è suddiviso in paragrafi. L'elemento `<p>` delimita ogni singolo paragrafo.

Ogni `<p>` è solitamente rappresentato come un blocco di testo separato, solitamente con un piccolo spazio prima e dopo.

Gli elementi `<p>` vuoti non sono validi.

3.2. Interruzioni di linea

`<br>` : interruzione di linea

Contenuto: *vuoto*
Attributi: standard HTML, ~~clear~~

L'elemento `<br>` termina la riga corrente. Il testo viene mandato a capo, senza iniziare un nuovo paragrafo.

L'attributo `clear` è usato per indicare la distribuzione degli oggetti floating rispetto all'interruzione di riga. Se presente, i float indicati vengono disposti prima dell'interruzione. *È stato eliminato in HTML5*.

I browser possono inserire dei `<br>` impliciti al posto degli spazi per adattare il testo alla dimensione della finestra (word wrap). Utilizzare i nonbreakable spaces (` `) per evitare questo effetto.

3.3. Sezioni

I documenti HTML dovrebbero essere divisi in sezioni logiche.

Questo facilita il rendering (soprattutto per i browser non visuali), l'analisi dei motori di ricerca e la conversione in altri formati (es. PDF).

Anche quando si utilizzano pesantemente gli stili, bisognerebbe basare la struttura del documento sui tag di sezione, opportunamente modificati nell'aspetto, in maniera da rimanere compatibili all'indietro e dare una minima semantica al documento.

`<h1>` ... `<h6>`: intestazioni di sezione

Contenuto: *inline*
Attributi: standard HTML

Gli elementi `<hX>` creano un'intestazione di livello via via più basso.

Di solito sono renderizzati con caratteri in grassetto e di dimensione decrescente (`<h1>` ha di solito una dimensione doppia rispetto al testo normale)

È opportuno evitare di formattare il testo all'interno di un elemento `<hX>`: utilizzare invece gli stili per

modificarne globalmente l'aspetto.

3.4. Elementi strutturali HTML5

HTML5 introduce alcuni importanti nuovi elementi utili per definire la struttura del documento:

- `<main>` permette di *racchiudere* il contenuto principale di un documento.
- `<section>` permette di *racchiudere* una sezione logica del documento. Solitamente contiene un elemento `<hX>` come intestazione.
- `<article>` permette di *racchiudere* una parte indipendente del documento, come gli articoli di un giornale.
- `<header>` *racchiude* le parte introduttiva di una sezione (`<section>`, `<article>` ma anche `<body>` o `<td>`). Spesso, se presente, contiene l'intestazione (`<hX>`) della sezione.
- `<footer>` *racchiude* la parte di chiusura di una sezione.
- `<nav>` *racchiude* una parte di documento usata per la navigazione (menu e simili).
- `<aside>` permette di *racchiudere* una parte del documento che non ha un particolare legame con il resto del documento stesso.
- `<figure>` e `<figcaption>` rappresentano una parte auto-contenuta del documento, come una figura, e la rispettiva didascalia.
- `<template>` *racchiude* frammenti di HTML utilizzati come modello *dagli script* (solitamente, quindi, non è visualizzato nella pagina).

3.5. Contenitori

Esistono due elementi invisibili in HTML, che però hanno un ruolo fondamentale per l'utilizzo di caratteristiche avanzate come gli stili.

Questi due elementi sono `<div>` e `<span>`. La loro semantica di base è praticamente nulla: semplicemente, `<div>` rappresenta un blocco di testo (ma **non** è un paragrafo), mentre `<span>` è una parte del flusso testuale.

In pratica, il contenuto di un `<div>` è preceduto e seguito da un ritorno a capo, mentre uno `<span>` può trovarsi ovunque nel flusso del testo.

`<div>`, `<span>` : contenitori generici

Contenuto: `<span>` : *inline*, `<div>` : *blocco*
Attributi: standard HTML

Questi due elementi delimitano parti di documento a cui assegnare caratteristiche speciali, quali un formato, una lingua, ecc.

Sono utili alla realizzazione dei layout HTML più complessi, soprattutto in congiunzione con i CSS.

In generale, possono anche essere utilizzati per creare elementi HTML ad-hoc, con formattazione e comportamento specifici, da affiancare all'HTML standard.

Gli attributi `class` e `id` sono fondamentali per il loro funzionamento.

4. Elementi di formattazione del testo

4.1. Formattazione semantica

I *phrase elements* sono utilizzati per attribuire un significato (semantica) particolare ad alcune parti del testo.

La semantica di solito è resa esplicita da rendering particolari, ma può essere anche utilizzata per rendere il testo più facilmente analizzabile dai tool automatici.

Questi elementi hanno nomi descrittivi: `<em>`, `<strong>`, `<dfn>`, `<code>`, `<samp>`, `<kbd>`, `<var>`, `<cite>`, `<abbr>`, ~~`<acronym>`~~

Contenuto: *inline*
Attributi: standard HTML

L'elemento `<acronym>` è *deprecated* in HTML5.

elementi di base

- `<em>`: Enfasi
  di solito equivalente a corsivo
- `<strong>`: Enfasi forte
  di solito equivalente a grassetto
- `<cite>`: Riferimento esterno o citazione
- `<dfn>`: Testo di una definizione
  da non confondere con le liste di definizioni
- `<code>`: Codice sorgente
- `<samp>`: Esempio di output
- `<kbd>`: Testo scritto da tastiera (digitato dall'utente)
- `<var>`: Nome di variabile
- `<abbr>`: Abbreviazione
  l'attributo *title* può essere usato per contenere la forma completa
- `<acronym>`: Acronimo

citazioni

`<blockquote>`, `<q>`: citazioni

Contenuto: `<q>` : *inline*, `<blockquote>` : *blocco*
Attributi: standard HTML, cite

L'elemento `<blockquote>` è utilizzato per citare blocchi di testo, mentre `<q>` serve ad inserire brevi citazioni nel flusso del testo

Entrambi hanno un attributo `cite` che può essere usato per fornire la URL del testo citato

I browser dovrebbero inserire opportune virgolette prima e dopo la citazione

La citazione inserita in un `<blockquote>` è indentata rispetto al resto del testo

L'uso di `<blockquote>` per indentare il testo è vivamente sconsigliato!

Testo preformattato

`<pre>` : testo *preformattato*

Contenuto: *inline* con esclusioni
Attributi: standard HTML

Il rendering di HTML ignora gli spazi bianchi e i ritorni a capo nel testo. Il flusso del testo segue le sole regole dettate dagli elementi corrispondenti (`<p>` , `<br>` ,...) e dalle dimensioni della finestra.

Con l'elemento `<pre>` si richiede al browser di rispettare la forma data al testo nel sorgente della pagina: il testo è visualizzato con un font a spaziatura fissa, gli spazi bianchi sono mantenuti e l'a capo automatico è disabilitato.

Nel testo preformattato si possono comunque usare i tag HTML inline tranne `<img>` , `<object>` , `<big>` , `<small>` , `<sub>` e `<sup>`

Revisioni

`<ins>` , `<del>` : testo inserito e cancellato

Contenuto: *inline* o *blocco*
Attributi: standard HTML, cite, datetime

Questi elementi sono usati per indicare revisioni del testo.

L'attributo `cite` può essere usato per indicare una URL in cui si trovano dettagli sulla corrispondente revisione. Un'indicazione sintetica del motivo della revisione si può inserire anche nell'attributo title.

L'attributo `datetime` può essere usato per contenere la data/ora della revisione

Sono gli unici due elementi HTML che possono essere usati sia come inline che come blocco.

Indirizzi

`<address>` : informazioni per contattare l'autore

Contenuto: *inline*
Attributi: standard HTML

Questo elemento può esser usato per marcare il testo in esso contenuto come "contatto informativo" per il blocco in cui è inserito.

Di solito lo si usa a livello di corpo del documento (`<body>`) o all'interno di un modulo (`<form>`).

I browser potrebbero renderizzare le informazioni di contatto in maniera speciale, cambiandone posizione e formattazione (ad es. sempre all'inizio del blocco, o come popup attivato da un piccolo bottone specifico, ecc.).

4.2. Formattazione di base

Gli elementi che seguono agiscono sulla formattazione di base dei caratteri. Il loro uso è ancora tollerato, ma si incoraggia sempre a sostituirli con elementi di formattazione semantica (ad es. `<em>`) o con stili (utilizzando ad es. degli `<span>` opportuni)

La combinazione (nidificazione) di questi elementi può essere sfruttata per ottenere formattazioni complesse (es. grassetto + corsivo)

Gli elementi sono: ~~`<tt>`~~ , `<i>` , `<b>` , ~~`<big>`~~ , `<small>`

Contenuto: *inline*
Attributi: standard HTML

Gli elementi `<big>` e `<tt>` sono *deprecated in* HTML5.

Brassetto, corsivo...

- `<tt>` : Testo a spaziatura fissa
  "Teletype", solitamente corrisponde al carattere courier. Eliminato in HTML5.
- `<i>` : Corsivo
  In HTML5 gli elementi `<i>` hanno un nuovo significato e sono utilizzati per rappresentare del testo con *qualità differenti* rispetto a quello che lo circonda. Per questo motivo, il testo non è più corsivo di default (ma è possibile usare i CSS per ripristinare questo effetto).
- `<b>` : Grassetto
  In HTML5 gli elementi `<b>` sono utilizzati genericamente per rappresentare del testo con *maggior importanza* rispetto a quello che lo circonda.
- `<big>` : Testo più grande
  Eliminato in HTML5.
- `<small>` : Testo più piccolo
  In HTML5 gli elementi `<small>` sono utilizzati genericamente per rappresentare *commenti a*

margine, tipicamente testo scritto in caratteri più piccoli.

Apice e pedice

`<sub>`, `<sup>` : apice e pedice

Contenuto: *inline*
Attributi: standard HTML

Questi elementi trasformano il testo contenuto in apice o pedice. In altre parole, diminuiscono leggermente la dimensione del carattere e spostano la sua linea di base più in alto o più in basso rispetto al testo normale

5. Liste e tabelle

5.1. Liste

HTML permette di definire tre tipi di liste: ordinate (numerate), non ordinate (puntate) e liste di definizioni (termini o altro)

Il rendering standard delle liste è molto semplice. Tuttavia, è possibile usare i fogli di stile per modificarne tutti gli aspetti

Le liste, anche di tipi diversi, possono venire nidificate a piacimento

Gli elementi usati per costruire liste sono `<ul>`, `<ol>`, `<li>`, `<dl>`, `<dt>`, `<dd>`

Non ordinate (puntate)

`<ul>`, `<li>`: liste non ordinate

Contenuto: `<ul>` : uno o più `<li>`, `<li>` : *flusso*
Attributi: standard HTML

Utilizzate di solito per generare liste puntate.

Ogni elemento `<li>` rappresenta un elemento della lista e può contenere qualsiasi tipo di markup, comprese altre liste.

Una lista dovrebbe contenere almeno un elemento `<li>`. Tuttavia in HTML5, per ovviare a un errore molto comune, questo vincolo è stato rimosso.

L'elemento `<li>` non può essere usato al di fuori delle liste.

Ordinate (numerate)

`<ol>`, `<li>`: liste ordinate

Contenuto: `<ol>` : uno o più `<li>` , `<li>` : *flusso*
Attributi: standard HTML

Utilizzate di solito per generare liste numerate

Ogni elemento `<li>` rappresenta un elemento della lista e può contenere qualsiasi tipo di markup, comprese altre liste. Gli elementi sono numerati progressivamente nell'ordine in cui appaiono nel documento sorgente.

Una lista dovrebbe contenere almeno un elemento `<li>`. Tuttavia in HTML5, per ovviare a un errore molto comune, questo vincolo è stato rimosso.

L'elemento `<li>` non può essere usato al di fuori delle liste

Di definizioni

`<dl>` , `<dt>` , `<dd>` : liste di definizioni

Contenuto: `<dl>` : uno o più `<dt>` e `<dd>` , `<dt>` : *inline*, `<dd>` : *flusso*
Attributi: standard HTML

Utilizzate di solito per definire termini o elementi similari, hanno grande versatilità

Gli elementi `<dt>` rappresentano termini a cui è associato un blocco di testo `<dd>` che li definisce

Si può associare un `<dd>` a più `<dt>` consecutivi

La definizione `<dd>` può contenere qualsiasi elemento di flusso, comprese altre liste

Di solito i `<dt>` sono enfatizzati, mentre i `<dd>` sono indentati

5.2. Menu in HTML5

Storia di un elemento in continua evoluzione

Inizialmente, i draft di HTML5 contenenvano specifici elementi come `<menu>` e `<menuitem>` che, uniti ad attributi come *contextmenu* (per ogni elemento HTML) o *menu* (nei `<button>`), permettevano di definire menu (e toolbar) usando un markup semantico specifico.

Le voci dei menu potevano essere composte usando anche i comuni elementi `<li>` e `<hr>`, nonché attributi specifici quali *type*, *label*, *icon*, *disabled*, *checked*, *radiogroup*, *default*, *command*.

Tuttavia, questo elemento è stato alla fine considerato poco utile e fonte di ambiguità, ed **è stato inizialmente rimosso nella specifica finale di HTML5**, mentre ora è accettato come mero sinonimo di `<ul>`, con una semantica differente.

Al momento attuale, sebbene alcuni browser contengano ancora una prototipale implementazione di questi elementi, il modo più corretto per strutturare un menu è usare le "comuni" liste non ordinate.

5.3. Tabelle

Le tabelle HTML offrono un sistema estremamente potente e versatile per disporre informazioni in righe e colonne

Le tabelle non sono progettate per creare layout di pagina, ma solo per strutturare informazioni in forma tabulare. Utilizzare le tabelle per creare layout rende questi ultimi poco portabili ed è fortemente sconsigliato.

Le tabelle fanno parte degli elementi di tipo blocco, quindi possono apparire direttamente nel `<body>` di un documento o in un contenitore `<div>`, e non devono mai essere nidificate in elementi come `<p>`.

L'elemento base della definizione delle tabelle è `<table>`

Specifica dell'ampiezza

Le larghezze delle colonne e della tabella stessa (attributo *witdh*) si possono specificare:

- In pixel (width="10")
- In percentuale, rispetto allo spazio disponibile per la tabella (width="10%")

Solo per le colonne, si possono usare anche i seguenti sistemi:

- Proporzionalmente, rispetto alla dimensione richiesta dalla tabella (width="10*")
- Per richiedere il minimo spazio necessario, si può usare la forma width="0*"

Se non si specifica una larghezza:

- Per una tabella, lo spazio è quello necessario a dare larghezza minima a tutte le colonne
- Per una colonna, la larghezza è ottenuta distribuendo proporzionalmente lo spazio disponibile e fornendo sempre almeno la larghezza minima necessaria al contenuto

Elementi di base

`<table>` : definizione di una tabella

Contenuto: caption (opzionale), sequenza di col o colgroup (opzionale), thead (opzionale), tfoot (opzionale), tbody (implicito se non specificato)
Attributi: standard HTML, cellspacing, cellpadding, ~~width~~, border, ~~rules~~, ~~summary~~, ~~frame~~

Una tabella è definita dagli elementi che sono nidificati in `<table>`, che devono essere nell'ordine:

- Una didascalia opzionale (`<caption>`)
- Una definizione opzionale delle colonne/gruppi di colonne
- Una intestazione di tabella opzionale

- Un piè di tabella opzionale
- Il corpo della tabella, che contiene i dati veri e propri.

L'elemento `<caption>`, se presente, ha contenuto *inline* e rappresenta la didascalia della tabella, che potrà essere renderizzata un maniera opportuna dal browser

In HTML5 l'attributo *summary* è *deprecated* e l'attributo *border* può valere solo "1" o essere omesso.

Attributi di base

`<table>` : definizione di una tabella

L'attributo `cellspacing` determina lo spazio (in pixel) tra le celle e tra le celle più esterne e il bordo della tabella

L'attributo `cellpadding` determina lo spazio (in pixel) tra il bordo di ciascuna cella e il suo contenuto

L'attributo ~~`width`~~ specifica la larghezza della tabella, in pixel o in percentuale. Si consiglia sempre di specificarlo per velocizzare il rendering (il valore di default è "il minimo necessario").

L'attributo ~~`border`~~ imposta lo spessore del bordo esterno della tabella. Un valore pari a zero elimina il bordo (utile per utilizzare la formattazione avanzata dei bordi tramite CSS)

L'attributo ~~`frame`~~ (*void, above, below, hsides, vsides, lhs, rhs, box, border*) determina quali dei bordi esterni della tabella saranno disegnati (con lo spessore dato da border). Il default è *box*.

L'attributo ~~`rules`~~ (*none, groups, rows, cols, all*) determina quali dei bordi interni alla tabella (tra le celle) saranno disegnati (con lo spessore dato da border). Il default è *all*.

Tutti questi attributi sono *deprecated in* HTML5 e i CSS devono essere usati al loro posto.

Righe

`<tr>` : righe di una tabella

Contenuto: uno o più `<td>` e `<th>`
Attributi: standard HTML, ~~align~~, ~~valign~~

Le tabelle sono composte da una serie di righe (`<tr>`), ognuna della quali contiene una o più celle.

Il numero di massimo celle presenti su una singola riga determina il numero di colonne della tabella. Il rendering mostrerà celle vuote a destra di ogni riga le cui celle sono minori del numero di colonne della tabella.

L'attributo ~~`align`~~ (*left*, *right*, *center*, *justify*, *char*) definisce l'allineamento orizzontale per tutte le celle della riga, mentre ~~`valign`~~ (*top*, *bottom*, *middle*, *baseline*) definisce quello verticale.

Gli attributi *align* e *valign* sono *deprecated in* HTML5 e i CSS devono essere usati al loro posto.

Celle

`<td>`, `<th>` : celle e celle di intestazione di una tabella

Contenuto: *flusso*
Attributi: standard HTML, ~~align~~, ~~valign~~, rowspan, colspan, ~~abbr~~, ~~axis~~, headers, ~~scope~~, ~~width~~, ~~height~~

Ogni riga di una tabella contiene delle celle. Le celle possono contenere contenuto arbitrario HTML, comprese altre tabelle, immagini, ecc.

Le **celle di intestazione** sono identiche alle celle standard, ma il browser dovrebbe renderizzarle in maniera più evidente.

Tipicamente una cella rappresenta l'intersezione di una riga con una colonna, tuttavia gli attributi `rowspan` e `colspan` permettono di specificare l'estensione della cella, rispettivamente in righe e colonne.

Gli attributi `abbr`, `axis`, `headers` e `scope` sono utilizzati per fornire dati avanzati di accessibilità alla tabella.

Gli attributi `width` e `height` servono a fornire informazioni sulle dimensioni della cella. Sono però **sconsigliati**: al loro posto andrebbero usati gli omonimi attributi degli elementi `<col>`.

Gli attributi *height*, *width*, *align*, *valign*, *abbr*, *axis* e *scope* sono *deprecated* in HTML5.

Gruppi di righe

`<thead>`, `<tbody>`, `<tfoot>` : raggruppamento di righe

Contenuto: uno o più `<tr>`
Attributi: standard HTML, align, valign

Le righe di una tabella possono essere suddivise in tre gruppi: **intestazione** (`<thead>`), **corpo** (`<tbody>`) e **piè di tabella** (`<tfoot>`).

Tipicamente, le righe in `<thead>` e `<tfoot>` vengono poste rispettivamente all'inizio e alla fine della tabella. Se la tabella è spezzata in più pagine, ogni segmento conterrà la stessa intestazione e piè di tabella.

Se si omettono i raggruppamenti, tutte le righe sono poste in un `<tbody>` esplicito. Non è possibile avere tabelle con soli `<thead>` e/o `<tfoot>`. Se specificati, questi due gruppi dovrebbero trovarsi entrambi all'inizio della definizione della tabella, prima del `<tbody>`.

In HTML5 `<tfoot>` può apparire anche alla fine della definizione della tabella.

Colonne

`<col>` : definizione di una colonna

Contenuto: vuoto
Attributi: standard HTML, ~~align~~, ~~valign~~, span, ~~width~~

Tramite uno o più elementi `<col>` **posti all'inizio della tabella** (prima delle righe) è possibile predefinire il numero e le caratteristiche delle colonne che la comporranno. Questo facilita il rendering della tabella e diminuisce il codice necessario a crearla.

Ogni `<col>` rappresenta un numero di colonne pari al suo attributo `span`. Ciascuna colonna avrà la dimensione specificata dall'attributo `width` e le celle corrispondenti avranno l'allineamento definito da `align` e `valign`.

In HTML5, questi elementi **possono comparire solo all'interno di un `<colgroup>`**. Inoltre, gli attributi *align*, *valign* e *width* sono *deprecated* in HTML5 e i CSS devono essere usati al loro posto.

gruppi di colonne

`<colgroup>` : definizione di gruppi di colonne

Contenuto: vuoto oppure uno o più `<col>`
Attributi: standard HTML, align, valign, span, width

Gli elementi `<colgroup>` rappresentano raggruppamenti logici di (definizioni di) colonne, che i browser possono renderizzare in vario modo.

Uno o più elementi `<colgroup>` possono essere **posti all'inizio della tabella** (prima delle righe) **in alternativa** agli elementi `<col>`.

Un `<colgroup>` vuoto rappresenta un numero di colonne pari al suo attributo `span`. Ciascuna colonna avrà la dimensione specificata dall'attributo `width` e le celle corrispondenti avranno l'allineamento definito da `align` e `valign`.

Per definire separatamente le caratteristiche di ciascuna colonna in un gruppo, si possono nidificare elementi `<col>` all'interno di un `<colgroup>`. Le caratteristiche delle `<col>` nidificate (compreso il loro numero totale) hanno la precedenza su quelle specificate globalmente sul `<colgorup>`.

6. Immagini e oggetti incorporati

6.1. Immagini

`<img>` : inclusione di immagine

Contenuto: vuoto
Attributi: standard HTML, src, alt, ~~longdesc~~, width, height, ismap, usemap

Inserisce nel documento l'immagine esterna referenziata dall'URL nell'attributo `src`

Il testo alternativo per l'immagine (`alt`) è una caratteristica essenziale per un documento HTML ad alta accessibilità.

L'attributo `longdesc` può essere usato per puntare alla URL di un documento che descrive nel dettaglio l'immagine. È *deprecated* in HTML5.

Gli attributi `width` e `height` dovrebbero essere sempre usati per fornire al browser un suggerimento sulle dimensioni da riservare per l'immagine sulla pagina. Se non corrispondono alle dimensioni reali dell'immagine, questa verrà ridimensionata di conseguenza (in maniera proporzionale se si specifica solo uno degli attributi). In HTML5 questi attributi non possono più contenere percentuali e non possono essere usati per ottenere ridimensionamenti non proporzionali.

6.2. Mappe immagine

Mappe immagine server-side

`<img>` : inclusione di immagine

Contenuto: vuoto
Attributi: standard HTML, src, alt, longdesc, width, height, ismap, usemap

L'attributo booleano `ismap`, se presente, trasforma l'immagine in una **server-side image map** quando questa è parte di un link creato da un elemento `<a>`.

Quando l'immagine viene cliccata, attivando il link, le coordinate del click vengono aggiunte alla URL indicata dall'attributo `href` del tag `<a>` sotto forma di *parametro get*.

Ad esempio, se l'URL è http://test.org/test e le coordinate x=1,y=7 il browser richiederà la risorsa con la URL http://test.org/test?1,7

Mappe immagine client-side

`<img>` : inclusione di immagine

Contenuto: vuoto
Attributi: standard HTML, src, alt, longdesc, width, height, ismap, usemap

L'attributo `usemap`, se presente, permette di trasformare l'immagine un una **client-side image map**.

L'attributo `usemap` deve contenere il nome di una image map definita nello stesso documento tramite l'elemento `<map>`

Le aree dell'immagine definite dalla mappa diverranno cliccabili.

Le image map client-side sono sempre preferibili per motivi di accessibilità.

`<map>` : client side image map

Contenuto: *blocco*, `<area>`
Attributi: standard HTML, name

L'elemento `<map>` dichiara una client-side image map con il nome specificato dall'attributo `name`.

Le aree della mappa possono essere specificate tramite una serie di elementi `<area>` o `<a>`, entrambi nidificati nell'elemento `<map>`

L'uso di elementi `<a>` è utile per creare mappe ad alta accessibilità, con un ricco contenuto testuale alternativo. In questo caso, il tag `<a>` può essere arricchito con attributi quali `shape` e `coords`, propri del tag `<area>`. Questo tipo di costrutto però è *deprecated* in HTML5.

`<area>` : client side image map area

Contenuto: *vuoto*
Attributi: standard HTML, shape, coords, href, alt

Gli elementi `<area>` nidificati in una `<map>` definiscono le aree cliccabili di un'immagine e le relative destinazioni

Ogni area ha una forma, determinata dall'attributo `shape`, che può valere *rect*, *circle* o *poly*.

L'attributo `coords` contiene le coordinate, separate da virgole, dei punti che definiscono la forma specificata:

- Per i rettangoli, le coordinate x e y degli angoli superiore sinistro e inferiore destro,
- Per i cerchi, le coordinate x e y del centro e il raggio,
- Per i poligoni, le coordinare x e y di tutti i vertici.

La destinazione del link è specificata dall'attributo `href`.

Una descrizione testuale dell'area, specificata con l'attributo `alt`, è obbligatoria per una mappa ad alta accessibilità.

6.3. Oggetti

`<object>` : inclusione di oggetto generico

Contenuto: *flusso*, `<param>`
Attributi: standard HTML, ~~classid~~, ~~codebase~~, ~~codetype~~, data, type, ~~standby~~, width, height

Gli attributi `classid` e `data` possono essere usati per specificare (in maniera mutuamente esclusiva):

- L'**implementazione** dell'oggetto: `classid` è una URI che punta all'implementazione dell'oggetto da includere (es. applet o altri piccoli programmi)

- I **dati** che costituiscono l'oggetto: `data` è una URI che punta alla sorgente dati (es. immagini, video, audio)

Per passare **parametri** all'oggetto caricato, si possono nidificare elementi `<param>`. Gli attributi `name` e `value` di questi ultimi determinano le coppie (nome, valore) passate all'oggetto in fase di inizializzazione.

Tutto l'eventuale altro contenuto dell'elemento `<object>` è gestito dall'oggetto caricato.

Origine e tipo di oggetto

`<object>` : inclusione di oggetto generico

Contenuto: *flusso*, `<param>`
Attributi: standard HTML, ~~classid~~, ~~codebase~~, ~~codetype~~, data, type, ~~standby~~, width, height

L'attributo `codebase` può essere usato per risolvere le URI relative presenti in `classid` e `data`

Gli attributi `codetype` e `type` specificano il tipo MIME rispettivamente per le risorse puntate da `classid` e `data`

L'attributo `standby` può essere utilizzato per specificare un testo da mostrare durante il caricamento dell'oggetto.

Gli attributi `width` e `height` hanno lo stesso scopo descritto per l'elemento `<img>`.

Gli attributi *classid*, *codetype*, *codebase* e *standby* sono *deprecated* in HTML5.

Elementi specifici HTML5

I tag `<object>` erano spesso impiegati in HTML4 per riprodurre audio e video o importare *applet*.

HTML5 fornisce elementi specifici per molti degli usi comuni dell'elemento `<object>`:

- `<audio>` e `<video>` sono utilizzati per importare (e riprodurre) flussi audio e/o video, *con un'interfaccia utente comune fornita dal browser e una ricca API per lo scripting*.
- `<embed>` è usata per importare plugin.
- `<canvas>` è usata per delimitare un'area in cui è possibile generare grafica dinamicamente attraverso le API fornite dall'elemento stesso.

7. Meta elementi

7.1. Collegamenti

relazioni tra documenti

`<link>`: relazioni tra documenti

Contenuto: vuoto
Attributi: standard HTML, href,hreflang, type, rel, ~~rev~~, ~~charset~~, media

L'elemento `<link>` è utilizzabile più volte e solo nella `<head>` del documento

Un `<link>` di default non genera link visibili all'utente, ma dichiara delle relazioni tra il documento corrente ed altre risorse. I browser possono sfruttare queste informazioni in vari modi.

L'attributo `rel` (o `rev`, se il collegamento è logicamente all'indietro) è fondamentale per gli elementi `<link>`, perché definisce il tipo di relazione con la risorsa identificati dall'URL contenuto nell'attributo `href`.

Gli attributi *rev* e *charset* sono *deprecated* in HTML5.

I `<link>` sono usati, ad esempio, per collegare un documento ai suoi fogli di stile, per specificare documenti alternativi in altre lingue, per definire una sequenza logica di lettura in un insieme di documenti, ecc.

tipi di relazioni tra documenti

- **Start, Next, Prev** : documento iniziale, successivo, precedente
  Definiscono il documento iniziale, successivo e precedente nella sequenza lineare cui appartiene il documento corrente
- **Contents**: sommario
  Indica il documento da usare come sommario
- **Index**: indice del documento
  Indica il documento da usare come indice per il documento corrente
- **Glossary**: glossario del documento
  Indica il documento da usare come glossario per il documento corrente
- **Copyright**: copyright statement del documento
- **Chapter**, **Section**, **Subsection, Appendix** : inizio capitolo, sezione, sottosezione, appendice correnti
- **Help**: aiuto sul documento
- **Bookmark**: documento per segnalibro
  Indica il documento "chiave" nell'insieme cui appartiene il documento corrente, da usare per impostare un segnalibro.

- **Alternate**: documento alternativo
  Usando gli attributi hreflang, media e type, si possono definire alternative al documento corrente in base alla lingua e/o al dispositivo di lettura.
  Un esempio molto comune è il tipo *application/rss+xml*, che collega a una pagina html il suo *feed RSS*.

- **Stylesheet**: foglio di stile
  Usato per collegare un foglii di stile al documento. Gli attributi type e media devono identificare rispettivamente il tipo MIME del foglio di stile (di solito *text/css*) e il dispositivo per il quale è pensato. Se si vogliono fornire fogli di stile alternativi, usare l'attributo title per dare un nome allo stile ed utilizzare il tipo composto "alternate stylesheet".
- **Shortcut icon** : icona per il sito (standard di fatto
  L'immagine così collegata a una pagina web viene utilizzata dai browser come icona nella barra del titolo e tra i preferiti. Il tipo e la dimensione dell'immagine collegata sono soggetti a forti restrizioni.
- **Preload**: precaricamento ad alta priorità di risorse esterne
  Utilizzato per *precaricare* risorse, suggerendo cioè al browser quali risorse presenti nella pagina corrente sono ad alta priorità e quindi devono essere caricate al più presto nella cache locale. E' usato principalmente con fogli di stile, script, font e immagini. L'attributo as si usa in questo caso per specificare il tipo della risorsa, ad esempio `as="style"` , `as= "image"` oppure `as="script"` .
  Attenzione: il preloading non applica la risorsa, ma la carica soltanto: ad esempio, se precarichiamo un foglio di stile, dovremo comunque poi applicarlo facendovi riferimento con un normale elemento `<link>` di tipo *stylesheet*, oppure se precarichiamo uno script dovremo poi attivarlo facendovi riferimento con un elemento `<script>` .
- **Prefetch**: precaricamento di risorse utili a pagine correlate
  Il *prefetching* viene utilizzato per precaricare nella cache del browser risorse non immediatamente necessarie alla pagina corrente, ma utili per le pagine ad essa collegate. Queste risorse verranno quindi scaricate, ma con bassa priorità rispetto al resto della pagina.

Risoluzione delle URL relative

`<base>` : base per le URL relative

Contenuto: vuoto
Attributi: standard HTML, href, target

Questo elemento, utilizzabile una sola volta nella `<head>` del documento, definisce (attributo `href`) la URL di base utilizzata per risolvere tutte le URL relative presenti nel documento.

L'attributo `target` può essere usato per definire il target di default in un documento con frames.

Se `<base>` non è specificato, la base della URL del documento corrente viene usata per risolvere tutte le URL relative.

7.2. Fogli di Stile

`<style>` : fogli di stile incorporato

Contenuto: testo
Attributi: type, media

Uno o più elementi `<style>`, posti nella `<head>` del documento, permettono di incorporare nello stesso (frammenti di) fogli di stile.

L'attributo `type` specifica il tipo MIME dello stile. In HTML5 l'attributo ha come default *text/css* e può essere omesso.

L'attributo `media` permette di specificare i dispositivi per cui lo stile è stato progettato.

I fogli di stile possono essere anche importati dall'esterno con il tag `<link>`. Inoltre, è possibile specificare uno stile specifico per ogni elemento HTML tramite l'attributo standard *style*.

L'attributo standard *class* permette infine di raggruppare diversi elementi HTML in classi, utili per poter attribuire loro uno stile uniforme.

8. Elementi interattivi

8.1. Collegamenti

Collegamenti attivi a risorse esterne

`<a>`: link attivi verso altre risorse

Contenuto: *inline* (ma senza altri link)
Attributi: standard HTML, href, name, hreflang, type, rel, ~~rev~~, ~~charset~~, accesskey

L'elemento `<a>` viene usato sia per definire l'origine di un link che per definirne particolari destinazioni.

- Se `<a>` è origine di un link, il suo attributo `href` individua l'URL della destinazione
- Se `<a>` è destinazione di un link, il suo attributo `name` è un nome univoco che potrà essere indirizzato dal *fragment identifier* di una URL. Questo permette di definire link ad un punto particolare del documento destinazione.

L'attributo `name` condivide lo stesso spazio di identificatori dell'attributo `id`. In effetti, è possibile usare l'**id** di qualsiasi elemento come *fragment identifier*. In HTML5, questa è la modalità preferita.

Il contenuto di `<a>` può essere qualsiasi HTML inline, ma non si possono avere link nidificati. L'aspetto predefinito di un link (caratteri sottolineati) può essere variato a piacere usando i fogli di stile.

Titolo e relazione della risorsa

`<a>`: link attivi verso altre risorse

Contenuto: *inline* (ma senza altri link)
Attributi: standard HTML, href, name, hreflang, type, rel, ~~rev~~, ~~charset~~, accesskey

Gli attributi `hreflang`, `charset` e `type` possono essere usati per dare al browser delle informazioni sulla destinazione del link: lingua del documento, set di caratteri usato e tipo (MIME) del contenuto.

L'attributo `accesskey` permette di specificare un carattere che l'utente potrà usare come *shortcut* per attivare il link. È utile per creare menu ad accesso rapido o anche per navigare in un documento molto lungo senza usare il mouse.

Gli attributi `rel` e `rev` permettono di dichiarare la relazione tra il documento corrente e quello collegato dal link (si veda l'elemento `<link>`), considerato rispettivamente link in avanti o all'indietro.

L'attributo `title` può esserer usato per descrivere meglio la destinazione del link.

Gli attributi *rev* e *charset* sono *deprecated* in HTML5.

8.2. Moduli

I moduli (o form) sono (parti di) documenti HTML contenenti, oltre al normale markup, anche particolari elementi detti **controlli**, con i quali l'utente può interagire.

I moduli sono inseriti nello speciale elemento HTML `<form>`. Solitamente, i moduli prevedono un sistema per inviare il valore dei loro controlli al server (*submit*) per ulteriori elaborazioni.

Tuttavia, non sono rari dei form che lavorano completamente *client side*, assistiti da script e oggetti incorporati.

Controlli

Gli elementi di controllo sono `<input>`, `<textarea>`, `<select>`, `<optgroup>`, `<label>`, `<fieldset>` e `<button>`.

Ogni controllo in un modulo deve necessariamente essere identificato da un nome, specificato tramite l'attributo `name`.

I controlli possono avere un valore (`value`) iniziale, che viene impostato alla creazione del modulo o quando si effettua un *reset* dello stesso.

Quando il modulo viene inviato, il server riceve le coppie (nome, valore) di ciascun controllo.

In HTML5 i controlli possono anche apparire al di fuori dell'elemento `<form>`, a patto che abbiano un attributo `form` impostato all'*id* di una `<form>` posta all'interno del documento.

Elementi di base

`<form>` : definizione di un modulo

Contenuto: *blocco* (esclusi altri `<form>`), `<script>`
Attributi: standard HTML, method, action, enctype, name, accept-charset

La definizione di un modulo richiede almeno la specifica, tramite l'attributo `action`, della URL della risorsa che ne elaborerà i dati (ad es. uno script server side)

L'attributo `method` (*get* o *post*) permette di specificare il metodo di invio dei dati alla risorsa indicata.

Se si usa il metodo *post*, può essere necessario specificare anche il metodo di codifica dei valori tramite l'attributo `enctype`:

- La codifica *application/x-www-form-urlencoded* è quella di default
- La codifica *multipart/form-data* è necessaria se si inviano files come parte del form.
- L'attributo `name` fornisce un nome al modulo, da utilizzare per lo scripting.
- L'attributo `accept-charset` è spesso utilizzato per indicare l'*encoding* dei caratteri strasmessi con la form, *consentendone una corretta decodifica da parte del server*.

Controlli <input>

`<input>`: controllo modulo

Contenuto: vuoto
Attributi: standard HTML, type, name, value, size, maxlength, checked, disabled, readonly, src, usemap, ismap, alt

L'elemento `<input>` viene usato per generare gran parte dei controlli all'interno dei moduli. La chiave della sua versatilità è l'attributo `type`, che può assumere i seguenti valori:

- **text**: crea una riga di input testuale
- **password**: come *text*, ma maschera i caratteri digitati
- **checkbox**: crea una casella di controllo
- **radio**: crea un pulsante di opzione
- **submit**: crea un bottone per l'invio del modulo
- **reset**: crea un bottone per la reinizializzazione del modulo
- **file**: crea un controllo per l'upload di un file
- **hidden**: crea un campo nascosto nel modulo
- **image**: crea un controllo per l'invio del modulo usando un'immagine
- **button**: crea un bottone

Controlli <input> in HTML5

HTML5 introduce molti altri tipi di controlli `<input>`, sempre distinti dal valore del loro attributo `type`:

- **tel**: controllo di input per numero telefonico
- **search**: controllo di ricerca
- **url**: controllo di input per URL

- **email**: controllo di input per indirizzi email
- **time, date**: controlli specifici di input per data/ora
- **number**: controllo di input numerico
- **range**: controllo di input per intervalli
- **color**: controllo di selezione per colori

Lo *user agent* dovrebbe mostrare i controlli più appropriati utilizzando questi raffinamenti nella specifica del tipo di input, in modo da fornire all'utente un'interfaccia più ricca.

Semantica dei controlli <input>

`<input>`: controllo modulo

Contenuto: vuoto
Attributi: standard HTML, type, name, value, size, maxlength, checked, disabled, readonly, src, usemap, ismap, alt

L'attributo `value` fornisce:

- la stringa di **inizializzazione** per i tipi *text*, *password*, *hidden*, *file*
- la **label** per i controlli di tipo *submit*, *reset* e *button*

L'attributo `size` fornisce la larghezza del controllo, espressa in pixel o in caratteri per i tipi *text* e *password*

L'attributo `maxlength` fornisce il massimo numero di caratteri digitabili nei campi di tipo *text* e *password*

L'attributo booleano `checked` determina se i controlli di tipo *checkbox* e *radio* saranno inizialmente selezionati

L'attributo `src` viene usato per i controlli di tipo *image*, come pure `ismap`, `usemap` (eliminato in HTML5) e `alt`. I bottoni grafici di tipo *image* passano in ogni caso le coordinate del click come campi di controllo addizionali (*nome.x, nome.y*) del modulo.

Gli attributi booleani `disabled` e `readonly` possono essere utilizzati per disabilitare e/o rendere di sola lettura i controlli.

Semantica dei controlli <input> in HTML5

HTML5 introduce anche nuovi attributi per gli `<input>`:

- L'attributo `required` marca il campo come necessario. Il browser non dovrebbe inviare la form se questi campi non sono compilati. Si applica anche agli elementi `<select>` e `<textarea>`.
- Gli attributi `min`, `max` sono usati per definire l'intervallo di valori consentiti (per numeri, date, ecc.).

- L'attributo `autocomplete`, che può valere *on* oppure *off*, controlla le funzioni di auto completamento dell'input fornite dal browser.
- L'attributo `multiple` richiede al browser di accettare più di un valore nel campo. Il modo in cui questo comportamento viene reso visivamente dipende dallo specifico tipo di controllo.
- L'attributo `pattern` specifica un'espressione regolare che deve fare match col valore del controllo.
- L'attributo `step` definisce la granularità dei valori ammessi dal controllo.
- Gli attributi `formaction`, `formethod`, `formenctype` permettono di sovrascrivere i corrispondenti attributi dell'elemento `<form>` nel caso in cui l'input sia usato come bottone di submit.
- L'attributo `placeholder` permette di fornire un testo di aiuto da mostrare nel campo quando questo è non compilato, e si applica anche alle `<textarea>`.

Controlli <textarea>

`<textarea>` : area di testo per i moduli

Contenuto: testo
Attributi: standard HTML, name, rows, cols, disabled, readonly

L'elemento `<textarea>` crea un'area di testo ampia in cui l'utente può digitare più righe

L'ampiezza dell'area *visibile* è determinata dagli attributi `rows` (righe) e `cols` (colonne). Il numero massimo di caratteri digitabili non può essere però limitato a priori.

Il testo contenuto nell'elemento viene usato come valore iniziale

I tag HTML contenuti nel testo di non vengono interpretati.

Controlli <select>

`<select>` : liste a scelta multipla

Contenuto: uno o più elementi `<option>` e `<optgroup>`
Attributi: standard HTML, name, size, multiple

L'elemento `<select>` crea un controllo lista contenente più opzioni, ciascuna rappresentata da un elemento `<option>`.

L'attributo booleano `multiple` determina de l'utente può selezionare un o più elementi della lista

L'attributo `size` indica quante delle opzioni debbano essere visibili contemporaneamente sul controllo

Il valore iniziale e il valore assegnato a questo controllo è specificato tramite le `<option>` e le `<optgroup>` nidificate.

Opzioni dei controlli <select>

`<option>`, `<optgroup>` : opzioni per i controlli `<select>`

Contenuto: `<optgroup>` : uno o più `<option>`, `<option>` : testo
Attributi: standard HTML, label, `<option>` : selected, disabled, value

Gli elementi `<option>` definiscono le opzioni selezionabili nei controlli `<select>`. Gli elementi `<optgroup>` servono a raggruppare `<option>` in modo da creare strutture logiche come i menu.

L'attributo `label` determina il testo visualizzato per le `<option>` e le `<optgroup>`. Nel caso di `<option>`, è possibile anche omettere la **label** e specificare il testo da visualizzare all'interno dell'elemento.

L'attributo `value` determina il valore dell'opzione, che sarà assegnato al nome del corrispondente campo `<select>` in fase di invio del modulo. Se non è specificato, verrà usato al suo posto il contenuto dell'elemento

L'attributo booleano `selected` determina se l'opzione sarà inizialmente selezionata.

Controlli <button>

`<button>` : bottoni per i moduli

Contenuto: *flusso*, eccetto elementi `<a>` e tutti gli elementi usati nei moduli
Attributi: standard HTML, name, value, type, disabled

Gli elementi `<button>` creano bottoni esattamente come gli elementi `<input>` con il corrispondente `type` (che può essere *submit*, *reset* o *button*)

La differenza è che il contenuto del bottone non è definito dall'attributo `value`, che qui rappresenta solo il valore dato al corrispondente nome quando il bottone viene premuto.

Il contenuto dell'elemento, che può essere HTML di qualsiasi tipo e lunghezza, verrà utilizzato per creare la "faccia" del bottone.

Associazione di testo ai controlli

`<label>` : testo associato a un controllo

Contenuto: *inline*
Attributi: standard HTML, for

L'elemento `<label>` permette di associare una parte di testo *inline* a un controllo del modulo.

Il controllo associato è identificato dal valore dell'attributo for, che deve corrispondere all'`id` (non al `name`!) di uno dei controlli del modulo corrente.

Il browser potrà, ad esempio, cambiare la renderizzazione del testo se il corrispondente controllo viene disabilitato.

Si possono associare più `<label>` allo stesso controllo

Raggruppamento dei controlli

`<fieldset>`, `<legend>` : raggruppamento di controlli

Contenuto: `<fieldset>`: *flusso*, un `<legend>` opzionale, `<legend>`: *inline*
Attributi: standard HTML

Gli elementi `<fieldset>` permettono di raggruppare logicamente parti di un modulo.

L'elemento `<legend>`, se specificato, fornisce una descrizione testuale del `<fieldset>`.

Questi elementi sono utili per garantire un'alta accessibilità ai moduli e facilitarne la compilazione.

9. Riferimenti

Web Hypertext Application Technology Working Group (WHATWG)
https://whatwg.org

Specifica HTML 5
http://www.w3.org/TR/html5/

Specifica HTML 4
http://www.w3.org/TR/html401/

Specifica XHTML 1
http://www.w3.org/TR/xhtml1/

Differenze tra HTML5 e HTML4
http://www.w3.org/TR/html5-diff/
