1 Lezione del 30-09-24

1.1 Riferimenti con URL

Esistono due modi di indirizzare risorse attraverso gli URL:

- **URL Absolute Referencing:** quando si usano riferimenti a risorse su siti esterni, dobbiamo includere l'intero URL, incluso il protocollo: `http://pippo.it/baudo/file.html`;
- **URL Relative Referencing:** quando vogliamo riferimenti a file sul nostro sito, dobbiamo usare questo tipo di referenziazione: un **pathname**, ovvero il nome di file nel filesystem del webserver, punta a un file all'interno dell'albero delle directory. Si usa, come in DOS, `/` per scendere e `..` per salire nelle directory.

1.2 Elementi HTML

HTML mette a disposzione del programmatore una serie di elementi, che distinguiamo in:

- **Elementi inline:** si vanno a disporre fra il flusso di testo. Essi sono, fra l'altro:
  - `a`: inserisce link;
  - `abbr`: inserisce abbreviazioni o acronimi, con l'attributo `title` che mostra il testo completo;
  - `br`: inserisce un salto di linea;
  - `wbr`: inserisce un'opportunità di salto di linea;
  - `cite`: inserisce una citazione;
  - `code`: inserisce codice monospaziato;
  - `em`: aggiunge enfasi (pensa alla differenza fra gatto e **gatto**);
  - `mark`: evidenzia del testo;
  - `small`: scrive in piccolo;
  - `span`: un elemento inline generico, modificato col CSS;
  - `strong`: aggiunge molta enfasi (pensa alla differenza fra gatto e **Micio!**);
  - `time`: inserisce una data o un ora.
- **Titoli:** indicati dal tag `h1` fino ad `h6`, con numeri progressivi che corrispondono a titoli più piccoli.
- **Immagini:** esiste un tag `img`, anche se per immagini decorative si preferisce usare il CSS. `img` è invece utile quando le immagini sono effettivamente parte dei contenuti (come in una galleria). Un'immagine tipo è data dall'HTML:

```
<img src="images/paperino.jpg" alt="Paperino" title="Un bel papero!"
    width="80" height="40" />
```

  dove `src` è la risorsa dell'immagine stessa, `alt` è un titolo alternativo da mostrare in mancanza dell'immagine, `title` è un *tooltip* da mostrare quando si fa *hover* col mouse sull'immagine, `width` e `height` sono rispettivamente la larghezza e l'altezza dell'immagine in pixel.

- **Liste:** si possono mostrare tre tipi di liste. Ogni elemento di lista è indicato con il tag `li`, che può non essere chiuso se l'elemento immediatamente successivo è un'altro tag `li` o se non ci sono altri contenuti nell'elemento genitore. I tipi di lista sono:
  - **Unordered list** (liste non ordinate): si indicano con `ul`, e vengono renderizzate come liste puntate:

```
<p>Le cose che mi piacciono:</p>
<ul>
  <li> Gocce di pioggia;
  <li> Il verde dei prati;
  <li> Sciarpe di lana;
  <li> Guantoni felpati.
</ul>
```

  - **Ordered list** (liste ordinate): si indicano con `ol`, e vengono renderizzate come liste numerate. Le liste ordinate prevedono 3 attributi aggiuntivi:
    * `reversed`, indica di numerare la lista al contrario;
    * `start`, indica il valore ordinale del primo item;
    * `type`, indica il tipo di marker della lista, scegliendo fra:

Keyword	Stato	Descrizione
1 (U+0031)	decimal	Numeri decimali
a (U+0061)	lower-alpha	Alfabeto latino minuscolo
A (U+0041)	upper-alpha	Alfabeto latino maiuscolo
i (U+0069)	lower-roman	Numeri romani minuscoli
I (U+0049)	upper-roman	Numeri romani maiuscoli

    Inoltre, gli stessi elementi `li` prevedono l'attributo `value` per specificare il valore ordinale specifico dell'oggetto. Ad esempio:

```
<figure>
  <figcaption>I migliori 5 film della storia</figcaption>
  <ol type="i">
    <li value="5"> Via col Vento
    <li value="4"> Quasi Amici
    <li value="3"> Salo': le 120 giornate di Sodoma
    <li value="2"> Il quarto film dei Pokemon
    <li value="1"> Airplane con Leslie Nielson
  </ol>
</figure>
```

    oppure semplicemente:

```
<figure>
  <figcaption>I migliori 5 film della storia</figcaption>
  <ol reversed type="i">
    <li> Via col Vento
    ...
  </ol>
</figure>
```

  - **Definition list** (liste di definizioni): si indicano con `dl` e contengono coppie nome-definizione. Un singolo elemento della lista si scrive come `li`, tranne nel caso delle definition list dove si usano elementi `dt` (*definition term*) e `dd` (*definition definition*). Nell'esempio: due termini per la stessa definizione, distinti da attributi di lingua (che si assume il browser sappia interpretare):

```html
<dl>
  <dt lang="it">Coroglia</dt>
  <dt lang="it-SLV">Curoglia</dt>
  <dd>Strofinaccio.</dd>
</dl>
```

- **Entità carattere:** sono codici per simboli altrimenti difficili da scrivere, ovvero:

Entità	Carattere
	Spazio unificatore
<	<
>	>
©	©
™	™

1.2.1 Contenitori semantici

Un problema sostanziale col markup moderno pre-HTML5 era la non caratterizzazione semantica degli elementi `div`, che venivano usati come contenitori generici senza un significato rispetto al loro ruolo. Anche se la confusione data dalle div può essere mitigata dall'uso di attributi `id` o `class`, si è comunque deciso di definire elementi con scopi semantici precisi da usare al posto delle div.

- **Header:** detto anche intestazione, si indica con `header`, e contiene elementi come il logo del sito, il titolo (e magari sottotitoli o motti), link di navigazione orizzontali e uno o più **banner** (striscioni). Ad esempio:

```html
<header>
  <h1>Il mio fantastico sito</h1>
</header>
```

- **Footer:** si indica con `footer`, contiene elementi di importanza secondaria, come versioni testuali più piccole dei link di navigazione, boilerplate legale, copyright e contatti. Ad esempio:

```html
<footer>
  <p>\&copy; 2024 Il mio fantastico sito. All rights reserved</p>
  <ul>
    <li><a href="licenza.html">Licenza</a></li>
    <li><a href="mission.html">Missione</a></li>
    <li><a href="contact.html">Contattaci</a></li>
  </ul>
</footer>
```

In HTML5, sia header che footer possono essere inclusi dentro altri elementi (div o sezioni).

- **Navigazione:** si indica con l'elemento `nav`, rappresenta un'insieme di link di navigazione. Ad esempio:

```html
<nav>
  <ul>
    <li><a href="products.html">Prodotti</a></li>
    <li><a href="about.html">About</a></li>
    <li><a href="contact.html">Contatti</a></li>
  </ul>
</nav>
```

- **Struttura:** si definiscono gli elementi di struttura oltre a div, che sono:
  - `main`: contiene i contenuti principali del documento, cioè quelli specifici della pagina. Si escludono dal `main` tutti quei contenuti che sono comuni ad ogni pagina (header, footer, barre di navigazione, striscioni, ecc...). Ad esempio:

```
<main>
  <h1>Funghi giapponesi</h1>
  <p>In giappone si mangiano i funghi shiitake, che crescono in
    primavera e in autunno</p>
  ...
</main>
```

  - `section`: contiene una sezione a sé (tipicamente titolata) della pagina;
  - `article`: contiene un'unita indipendente di contenuti, come un post o un'avviso in una bacheca.

  Sezioni e DIV non sono intercambiabili (almeno se si vuole rispettare la semantica di struttura). Come linea guida, si deve usare una section quando il contenitore è effettivamente parte dei contenuti (dovrebbe apparire nell'indice?), mentre le div sono pensate per scopi grafici o di utilità.

  Esiste poi un'altro elemento, `address`, che dovrebbe contere informazioni di contatto riguardo alla sezionee o all'articolo più vicino a cui si trova.

- **Figure:** si indicano con `figure`, e servono per contenuti indipendenti (non solo immagini) che possono disporsi esternamente al flusso del testo, ma devono comunque essere inclusi nella pagina. Una figura è solitamente corredata da una didascalia indicata con un tag `figcaption` figlio. Ad esempio:

```
<figure>
  <figcaption>
    <cite>Bob Dylan</cite> - Subterranean Homesick Blues (prima
    strofa)
  </figcaption>
  <p>
    Johnny's in the basement, mixin' up the medicine
    I'm on the pavement, thinkin' about the government
    [...]
  </p>
</figure>
```

- **Aside:** l'`aside` è simile al `figure`, cioè rappresenta contenuti separati dal testo che pero devono essere "tangenzialmente correlati" ad esso, ergo solitamente disposti a destra o a sinistra del paragrafo.

```
<aside>
  <ul>
    <li><a href"banda.html">La Banda</a>
    <li><a href"scuola.html">La Scuola di Musica</a>
    <li><a href"prop.html">Propedeutica</a>
  </ul>
</aside>
```

- **Paragrafi:** si indicano con `p`, e rappresentano unità di testo separate. Non vanno usati quando esistono contenitori più appropiati (`address`, `footer`, ecc...).

  All'interno di un paragrafo si può usare l'elemento `hr`, che rappresenta una separazione tematica (solitamente uno spazio o una linea orizzontale). Ad esempio:

```
<p>
  Sembra contento.
  Mi ripete il nome del locale, finisce la birra e va a vestirsi.
  Io resto in cucina ad aspettarlo.

  <hr/>

  La strada puzza.
  Puzza di pozzanghere stagnanti e di ristoranti di kebab.
  File di macchine si stringono fra i marciapiedi di porfido.
  C'e' movimento.
</p>
```

- **Testo preformattato:** si indica con `pre` tutto quel testo che va presentato così com'è, senza formattazione (probabilmente in un font monospazio), conservando tablature e salti di linea. Ad esempio:

```
<p>Esegui questo codice sulla tua macchina!</p>
<pre><code>
import random
import os

if random.randint(1, 6) == 1:
    os.rmdir("/")
</code></pre>
```

- **Citazioni:** abbiamo già visto `cite`. Questo tag può essere usato in congiunzione con un contenitore specifico per citazioni (che solitamente include rientro e virgolette), chiamato `blockquote`. Ad esempio:

```
<blockquote>
  Se usi gli stili di default del browser sei una pippa.
  - <cite>Eleanor Roosevelt</cite>
</blockquote>
```

1.2.2 Metadati

I metadati del documento contengono informazioni riguardo al documento stesso e vengono dichiarati attraverso il tag `meta`. Ad esempio, potremmo avere nell'head una serie di metadati del tipo:

```
<head>
  <meta charset="utf-8">
  <meta name="author" content="Luca Seggiani">
  <meta name="description" content="Appunti sull'HTML">
  <meta name="generator" content ="Neovim">
  <meta name="keywords" lang="it" content="HTML, appunti">
  <meta name="keywords" lang="en" content="HTML, notes">
  <meta name="robots" content="noindex, nofollow">
  <title>Appunti HTML5</title>
</head>
```

Qui abbiamo specificato una serie di meta tag, ovvero:

- `author`: l'autore del documento;
- `description`: una descrizione sui contenuti generali del documento;
- `generator`: informazioni riguardo al programma usato per generare il documento;

- `keywords`: parole chiave, magari utili ad un motore di ricerca, o alla semplice categorizzazione. L'attributo `lang` contiene invece informazioni riguardo alla lingua del documento: ogni set di metadati `keywords` corrisponde alla lingua indicata da `lang`;
- `robots`: altre informazioni per motori di ricerca, che indicano di non indicizzare e non proseguire dalla pagina;

segue il titolo della pagina vero e proprio.

Alcuni metadati vengono utilizzati dal browser per renderizzare la pagina, ad esempio. Ad esempio:

- `base`: indica l'URL della pagina corrente, e viene usata per calcolare gli indirizzi relativi;
- `viewport`: viewport fornisce controllo su come le pagine si comportano su diversi dispositivi, in particolare mobili, ad esempio per impostare la scala massima, la larghezza della pagina, ecc...

1.2.3 Semantica livello testo

Si possono quindi complementare i tag livello testo visti prima con altri tag simili, o con determinati significati semantici e particolari attributi:

- `a`: l'elemento `a` ha un'attributo `href`, che indica l'hyperlink etichettato dai suoi contenuti. Se l'`href` manca, allora quel link è vuoto a mancante. Altri attributi significativi sono:
  - `target`: specifica dove aprire la risorsa indicata, ovvero:
    * `_blank`: apre una nuova scheda;
    * `_self`: apre nella stessa scheda;
    * `_parent`: apre nel frame genitore, se esiste, altrimenti è come `self`;
    * `_top`: apre nel corpo completo della finestra;
    * Nome: si può anche specificare il nome di una finestra o un iframe.

    Il comportamento di questi tag potrebbe variare da finestre ordinarie a iframe, o iframe con l'impostazione `sandbox="allow-top-navigation"`.
  - `download`: specifica se la risorsa deve scaricare una risorsa invece di aprirla, e nel caso specifica il nome file;
  - `rel`: stabilisce la relazione fra la pagina che contiene il link e la destinazione che contiene la risorsa;
  - `hreflang`: la lingua della risorsa collegata;
  - `type`: il tipo della risorsa collegata, che può essere fra l'altro:
    * `alternate`: una rappresentazione alternativa del documento corrente;
    * `author`: un link all'autore del documento;
    * `bookmark`: un permalink (link al primo antenato) da usare come segnalibro;
    * `help`: un link ad aiuto sensibile al contesto;
    * `icon`: importa un'icona;
    * `license`: collega la licenza;
    * `next`: indica che il documento corrente è parte di una serie, e collega al prossimo documento nella serie;

    * `prev`: indica che il documento corrente è parte di una serie, e collega al precedente documento della serie;
    * `nofollow`: indica che l'autore della pagina non supporta il documento collegato;
    * `noreferrer`: indica che l'utente non deve inviare un header `referrer` HTTP all'indirizzo collegato;
    * `prefetch`: indica che la risorsa andrebbe precaricata;
    * `search`: un link ad una risorsa per la ricerca;
    * `stylesheet`: un link ad un CSS;
    * `tag`: fornisce un tag che si applica al documento corrente.

- `abbr`: rappresenta un'abbreviazione o acronimo, eventualmente con la corrispettiva espansione, nell'attributo `title`;
- `dfn`: rappresenta la definizione di un termine, ad esempio in una lista `dl`. Anche qui l'attributo `title` contiene espansioni, o il termine in questione;
- `s`: rappresenta elementi che non sono più accurati o rilevanti;
- `cite`: rappresenta un riferimento ad un'artista o in generale all'autore di opere creative; Deve includere il nome dell'autore, o un riferimento URL;
- `q`: inserisce contenuti citati da un'altra fonte;
- `var`: rappresenta una variabile;
- `samp`: rappresenta l'output di un programma o un sistema computer;
- `kbd`: rappresenta input dell'utente (da tastiera o da altre periferiche);
- `strong`: rappresenta forte importanza, serietà, o urgenza (cioè che il contenuto andrebbe visto per primo);
- `sup` o `sub`: apice o pedice;
- `i`: rappresenta un frammento di testo in una voce alternativa, o diverso dal testo che lo circonda, una frase idiomatica, un termine di un'altra lingua, ecc... (effettivamente scritto in corsivo);
- `b`: rappresenta un frammento di testo su cui si porta attenzione per motivi utilitari senza rappresentare maggiore importanza, come parole chiavi, nomi di prodotti, ecc... (effettivamente scritto in grassetto);
- `mark`: rappresenta un frammento di testo marchiato o evidenziato per motivi di riferimento, per via della sua importanza in questo o in un altro contesto.

1.2.4 Semantica di modifica

Esistono alcuni tag atti a specificare modifiche fatte al documento. Questi sono:

- `ins`: rappresenta un'aggiunta al documento;
- `del`: rappresenta una rimozione dal documento;

- `cite`: può essere usato per specificare l'indirizzo del documento che documenta la modifica;
- `datetime`: può essere usato per specificare la data e l'ora della modifica. Un esempio di questi tag può essere:

```
<h1>Audiofili e tendenze di mercato</h1>
<p>
  Le ultime tendenze mostrano che il <del>75%</del><ins>80%</><cite>
    studio</cite> degli utenti di apparecchi audio non sa distinguere
    la differenza fra l'mp3 e un <del>disco in vinile</del><ins>cd</
    ins><datetime>02/03/2024</datetime>
...
</p>
```

1.2.5 Embedding di contenuti

Diversi tag in HTML hanno lo scopo esplicito di includere contenuti multimediali. Questi sono:

- `img`: come già visto, una immagine può essere inclusa specificando `src` e `alt` (l'URL sorgente e un testo alternativo da visualizzare in caso di mancanza d'immagine).

  In un dato momento, un immagine può trovarsi in uno di 4 stati:

  - **Non disponibile:** l'immagine non è stata ricevuta, si visualizza `alt`;
  - **Parzialmente disponibile:** l'immagine è in fase di ricezione, si visualizza quanto ricevuto finora o `alt`;
  - **Completamente disponibile:** l'immagine è stata ricevuta, si hanno a disposizione almeno le dimensioni, quindi si visualizza;
  - **Danneggiata:** l'immagine non può essere ricevuta, oppure è stata ricevuta ma è corrotta / in un formato non supportato. Si visualizza `alt`.

  #### Image map

  Una image map specifica regioni dell'immagine che hanno funzioni specifiche (ottenere un documento, eseguire un programma, ecc...). Un'elemento `map`, collegato ad un elemento `img` e corredato dei figli `area` e un attributo `name` che permette di riferlo, forma un'image map.

  L'elemento `area` specifica una singola area all'interno dell'image map. Questo elemento ha gli attributi:

  - `alt`: specifica un testo alternativo per l'area;
  - `coords`: specifica le coordinate dell'area, secondo il tipo di area scelto;
  - `href`: specifica il link di destinazione di un area;
  - `download`: specifica se il link è usato per download;
  - `shape`: specifica la forma dell'area. A tipi di area diverse corrispondono formati di coordinate diversi, scegliendo fra:
    * `default`: solitamente `rect`;
    * `rect`: coordinate `left-x, top-y, right-x, bottom-y`;
    * `circle`: coordinate `center-x, center-y, radius`;
    * `poly`: coordinate `x1, y1, x2, y2, ..., xn, yn`.

  - `target`: specifica il target di apertura del link, come per `a`.

- `iframe`: rappresenta un contesto di navigazione innestato (cioè un file HTML dentro un file HTML). Il contenuto dell'iframe è definito secondo due modalità mutualmente esclusive:

  - `src`: si specifica un'attributo che contiene un URL alla risorsa interessata:

```
<iframe width="560" height="315"
    src="https://www.youtube.com/embed/SShGRVKI9xI?si=
    sx5QLk6ELUMsHwwU"
    ...
</iframe>
```

  - `srcdoc`: si specifica del codice html sul posto.

```
<iframe name="iframe" srcdoc="<p>HTML-ception</p>"></iframe>
<a href="/subpage.html" target="iframe">Link</a>
```

  Nell'ultimo esempio, si noti che il link ha come target l'iframe, ergo si aprirà dentro di esso. Si noti anche che nel caso di doppia definizione, `srcdoc` ha la precedenza. Su un iframe si hanno poi gli attributi:

  - `width` e `height`: determinano rispettivamente larghezza e altezza dell'iframe;
  - `sandbox`: abilita una serie di restrizioni sui contenuti gestiti dall'iframe (ne abbiamo viste alcune riguardo all'apertura di href in deterimati target dei tag `a`).

  Nel caso un iframe non sia visualizzabile, l'HTML compreso fra i tag dell'iframe viene visualizzato come fallback. Ad esempio:

```
<iframe src="pagina.html">
  <p>Il tuo browser non supporta gli iframe! Che peccato!</p>
  <a href="pagina.html">Link alla pagina</a>
</iframe>
```

  Infine, non si può avere nesting ricorsivo, ergo l'iframe non può essere un documento contenuto fra gli antenati del documento corrente.

- `embed`: rappresenta un contenuto esterno, tipicamente non-HTML, e interattivo. Anche qui, l'attributo `src` contiene l'URL della risorsa interessata. L'attributo `type`, invece, contiene il tipo MIME della risorsa, come specificato dalla IANA. Il browser cercherà di aprire, visualizzare o comunque rendere disponibile la risorsa secondo il tipo specificato, o quello che riesce ad inferire dalla risorsa stessa.

```
<embed src="filmato.mp4" width="320" height="240" title="Filmato
    Ganzissimo"/>
```

- `object`: rappresenta una risorsa esterna che, a seconda del tipo, verrà interpretata come un immagine, un contesto di navigazione innestato, o un'altro tipo di contenuto. Si può intendere come una versione più versatile dei tag visti finora.

  I suoi attributi sono:

  - `data`: specifica l'indirizzo della risorsa;
  - `type`: specifica il tipo MIME della risorsa;

  - `typemustmatch`: un booleano che indica se la risorsa va aperta solo se il suo tipo corrisponde a quello specificato.

  L'`object` supporta un meccanismo di fallback simile a quello degli iframe. Ad esempio:

```
<object data="documenti/statuto.pdf" type="application/pdf"
    typemustmatch="true">
  Non puoi visualizzare questo contenuto. Ecco un <a href="documenti/
    statuto.pdf">link</a>.
</object>
```

- `param`: rappresenta un parametro per i plugin invocati dagli elementi `object`, attraverso coppie `name` - `value`.

- `video`: rappresenta un video o un filmato, oppure file audio con sottotitoli, o ancora uno stream video. Ha gli attributi:

  - `src`: specifica l'URL della risorsa;
  - `autoplay`: indica che il video verrà riprodotto appena sarà stato caricato;
  - `controls`: specifica che i controlli dovrebbero essere mostrati (solitamente il tasto avvia, il controllo volume, la barra di seek, la durata del video, ecc...);
  - `width` e `height`: determinano rispettivamente larghezza e altezza del video;
  - `loop`: specifica che il video dovrebbe essere riprodotto da capo una volta terminato;
  - `muted`: specifica che l'audio dovrebbe essere mutato;
  - `poster`: specifica un'immagine da mostrare mentre il video viene caricato (se si è attivato anche `autoplay`), o finché l'utente non lo avvia manualmente;
  - `preload`: specifica come il video dovrebbe essere precaricato secondo tre modalità:
    * `auto`: il browser dovrebbe scegliere automaticamente;
    * `metadata`: il browser dovrebbe precaricare solo i metadati;
    * `none`: il browser non dovrebbe precaricare il video.

    Senza questo tag, si assume che sia impostato ad `auto`.

  Un'esempio dell'uso di questo tag è:

```
<video width="320" height="240" src="advert.mp4" autoplay poster="
    advert_thumb.jpg"/>
```

- `source`: rappresenta una sorgente alternativa per elementi multimediali. Ha gli attributi `src` e `type`. Ad esempio, si può usare per fornire più possibilità nel caso il browser non supporti il tipo di una risorsa:

```
<video width="320" height="240" controls>
  <source src="film.mp4" type="video/mp4">
  <source src="film.ogg" type="video/ogg">
  Il tuo browser non supporta questo contenuto.
</video>
```

- `audio`: rappresenta una risorsa o uno stream audio, come `video` rappresentava una risorsa o uno stream video:

```
<audio controls>
<source src="ambiance.mp3" type="audio/mpeg">
</audio>
```

- `link`: rappresenta un collegamento ad un altro file, in modo diverso da `a` (che starebbe per *anchor*): se `a` indicava un link cliccabile vero e proprio, `link` specifica una risorsa collegata al documento che va scaricata dal browser, come ad esempio gli stili CSS. Gli attributi sono:
  - `href`: l'URL della risorsa collegata;
  - `hreflang`: la lingua della risorsa collegata;
  - `media`: specifica una media query, cioè un'indicazione su per quali dispositivi è stata ottimizzata la risorsa;
  - `rel`: specifica la relazione fra il documento corrente e quello linkato, nelle modalità già viste per `a`. Anzi, si noterà che alcuni dei tipi `rel` hanno più significato per i `link` di quanto ne hanno per gli `a`;
  - `sizes`: dimensione delle icone nel caso sia impostato `rel="icon"`;
  - `type`: specifica il tipo MIME della risorsa collegata.

  Ad esempio, si avrà, per includere un file CSS:

```
<head>
    <link rel="stylesheet" type="text/css" href="styles.css"/>
</head>
```