ESERCITAZIONE 1

Corso di
CVeDI (8 CFU)

Elia Guarnieri
elia.guarnieri@unimib.it

RIPASSO HTML E CSS

1. Pagine web e codice di markup
2. Tag e attributi
3. Gerarchia dei tag
4. Introduzione ai CSS

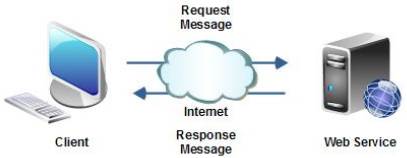
PAGINE WEB E CODICE DI MARKUP

L’ARCHITETTURA DEL WEB

- Internet è una rete di reti.
- I nodi delle reti di computer sono anche detti **host**, “ospiti”, perché ospitano le applicazioni.
- La comunicazione tra diverse entità della rete è regolata dai cosiddetti “**protocolli di comunicazione**”
- Possiamo affermare che il **World Wide Web** è nato dall’applicazione del concetto di **ipertesto** al contesto di Internet (Tim Barners-Lee, primi anni ‘90)

L’ARCHITETTURA DEL WEB

Il Web supporta l’interazione tra client e server via **HTTP**



- il client è realizzato da un “Browser”
- il server è realizzato da un “HTTP Server” o “Web Server”
- l’interazione “client/server” si configura come “domanda/risposta”

IL BROWSER

Il **browser** è l'applicazione per il Web sul lato del client. Dall'inglese "to browse" che vuol dire "sfogliare".

Un web browser (detto User-Agent) è un programma che consente la navigazione nel Web da parte di un utente.

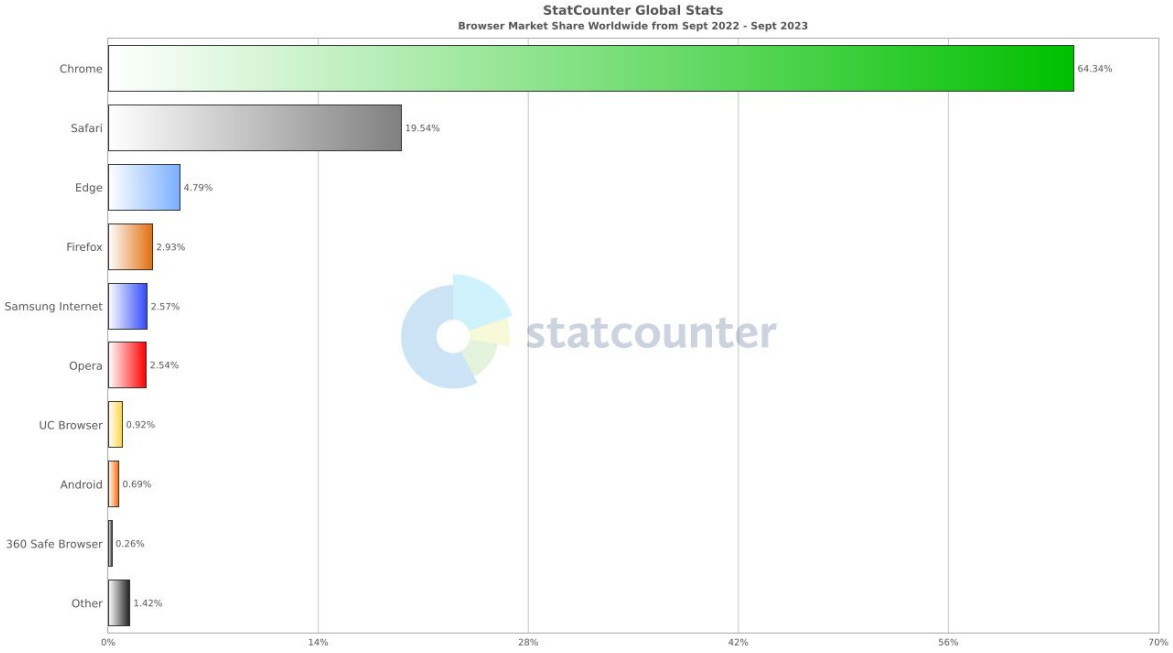
La funzione primaria di un browser è quella di **interpretare** il **codice** con cui sono espresse le informazioni (pagine web) e visualizzarlo (operazione di **rendering**) in forma di ipertesto.

I browser moderni hanno anche funzioni più avanzate per trattare altri tipi di dati (i.e. Multimedialità, RSS, ...)

Il rendering dipende dai dispositivi utilizzati, anche "non visuali", ad esempio per supportare utenti non-vedenti (es., sintesi vocale, alfabeto Braille)

DIFFUSIONE DEI BROWSER

Fonte

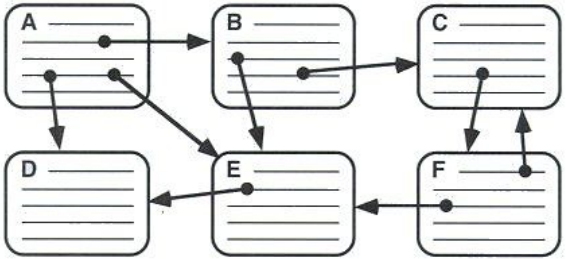


WEB PAGE

- Una **pagina web** (web page, o anche documento) è costituita da diversi oggetti (risorse nella terminologia del web).
- Una **risorsa** è un file, cioè una sequenza di dati (in formato digitale), residente in un computer. Questo file è identificato da un **URL** (cioè un *indirizzo univoco* per la risorsa).
- La maggior parte delle pagine web sono costituite da un file **HTML** che definisce *la struttura e i contenuti* della pagina.
- L'HTML è il linguaggio con cui si scrivono gli ipertesti. Permette di definire collegamenti, inserire immagini, elenchi...
- Il **Web Server** si occupa di gestire le risorse (file) su un computer e di renderle disponibili ai client.

GLI IPERTESTI

Un **ipertesto** (hypertext) è un insieme di testi o pagine leggibili in maniera *non sequenziale*, tramite particolari collegamenti che si chiamano hyperlink (o più semplicemente link), che costituiscono un rete raggiata o variamente incrociata di informazioni organizzate secondo criteri paritetici o gerarchici (i.e. menu)



I LINGUAGGI DI MARKUP

HTML (HyperText Markup Language) è un linguaggio di markup

In generale, un linguaggio di markup è un linguaggio che consente di descrivere dati tramite dei **marcatori** (tag)

Il termine markup (o marcatura) deriva dall'**ambiente tipografico** (le parti del testo che andavano evidenziate durante la stampa venivano segnalate con dei marcatori)

Così come il compositore delle tipografie interpretava i marcatori, oggi i client web interpretano i tag e trattano il loro contenuto seguendo determinate convenzioni

NON È UN LINGUAGGIO DI PROGRAMMAZIONE

I linguaggi di programmazione sono dei linguaggi formali che permettono l'elaborazione dei dati in input per generare un output

Per fare questo i linguaggi di programmazione mettono a disposizione degli strumenti di base come le istruzioni e le variabili

HTML (così come CSS) non è un linguaggio di programmazione. I tag possono venire annidati l'uno dentro l'altro e definire la struttura gerarchica del documento, ma si occupano principalmente della **presentazione** dei dati, non della loro elaborazione o generazione

PAGINA HTML

Una **pagina HTML** è rappresentata da un file di testo, ovvero un file che possiamo modificare con programmi come notepad.

Basta salvare il documento con l'estensione ".html"

Da quel momento in poi, sarà possibile aprirlo in un browser per visualizzare il rendering

PAGINA HTML BASE - ESEMPIO DI ANNIDAMENTO DEI TAG

```
<!doctype html>
<html>

<head>
  <title>Pagina HTML base</title>
</head>

<body>
  <!-- solo il codice in questa sezione sarà visualizzato -->
  Costruiamo una pagina web…
</body>

</html>
```

TAG E ATTRIBUTI

APERTURA

CHIUSURA

GLI ELEMENTI IN HTML

Molti elementi in HTML servono per descrivere porzioni di pagina, aree, o contenuti. Ad esempio `<body>` descrive il contenuto di tutta la pagina, `<h1>` racchiude un titolo e `<p>` denota un paragrafo nel testo

Pertanto **un elemento HTML è quasi sempre un contenitore** e il suo contenuto è delimitato da

- tag di apertura (es. `<div>`);
- tag di chiusura (es. `</div>`).

```
<div>
  Lorem ipsum, quia dolor sit, amet, consectetur, adipisci
  velit, sed quia non numquam eius modi tempora incidunt.
</div>
```

ESEMPIO DI RENDERING

Una frase contenuta tra i tags <u></ u>, quando viene richiamata da un browser, viene visualizzata come sottolineata (u=underline, sottolinea)

```
<u>Esercitazioni CVeDI</u>
```

→ Esercitazioni CVeDI

```
<tag
  attributo1="valore1"
  attributo2="valore2"
>
  <!-- contenuto -->
</tag>
```

GLI ATTRIBUTI

In sostanza gli **attributi**:

- sono **coppie chiave-valore** separate dal carattere “=” (uguale)
- i valori sono tipicamente racchiusi tra virgolette "", ma è possibile anche utilizzare gli apici ' '
- si scrivono lasciando almeno uno spazio dopo il nome dell’elemento nel tag di apertura (o nell’unico tag nel caso di elementi non contenitori)

DIVERSI TIPI DI TAG

Con tag di chiusura vs empty tag

- **Con necessità di chiusura**: quasi tutti i tag html. Necessitano di un tag di apertura, seguito dal contenuto, e di un tag di chiusura

```
<p>Lorem ipsum</p>
```

- **Empty tag**: non hanno contenuto. Spesso si affidano agli attributi

```
<br/> <!— tag per andare a capo —>

<img/> <!— tag per inserire immagini (attributi comuni:
src, alt, title) —>
```

TAG PER IL LAYOUT

Divisore

```
<div>
  Lorem ipsum, quia dolor sit, amet, consectetur, adipisci
  velit, sed quia non numquam eius modi tempora incidunt.
</div>
```

- Uno dei tag più utilizzati per strutturare le pagine web
- Elemento blocco
- Necessita chiusura

TAG PER I TESTI

<p>

<h1>
<h ... >

Paragrafo

```
<p>Lorem ipsum</p>
```

- I browser aggiungono automaticamente del margine prima e dopo questo tag
- Elemento blocco
- Necessita chiusura

A capo

```
Lorem ipsum<br/>
```

- Non necessita di chiusura

Headings/Titolo

```
<h1>Titolo 1</h1>
<h2>Titolo 2</h2>
<h3>Titolo 3</h3>
<h4>Titolo 4</h4>
<h5>Titolo 5</h5>
<h6>Titolo 6</h6>
```

- h1 definisce il titolo più importante, h6 il sottotitolo meno importante
- Elemento blocco
- Necessita chiusura

TAG PER I TESTI

<u>

Sottolineato

```
<u>Lorem ipsum</u>
```


<u>Lorem ipsum</u>

Corsivo

```
<em>Lorem ipsum</em>
```


Lorem ipsum

Grassetto

```
<strong>Lorem ipsum</strong>
```


Lorem ipsum

- Elementi inline
- Necessita chiusura

TAG PER I COMMENTI

Commento

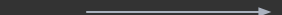
```
<!-- questo è un commento -->
```

- Scrivere annotazioni generiche al markup
- Rendere inattive porzioni di codice in fase di debug o per effettuare dei test
- Segnalare la chiusura di blocchi di codice per evitare confusione

ELENCHI PUNTATI E NUMERATI

Elenco puntato

```
<ul>
  <li>Primo elemento</li>
  <li>Secondo elemento</li>
</ul>
```



- Primo elemento
- Secondo elemento

- Bisogna fare attenzione alla gerarchia degli elementi
- L'apertura del tag `<ul>` deve venire all'inizio della lista
- La chiusura del tag `</ul>` deve venire alla fine
- In mezzo, ogni elemento della lista deve essere aperto "`<li>`" e poi chiuso "`</li>`" prima che se ne apra un altro

```
<ul>
  <li></li>
</ul>
```

ELENCHI PUNTATI E NUMERATI

Elenco numerato

```
<ol>
  <li>Primo elemento</li>
  <li>Secondo elemento</li>
</ol>
```

→

1. Primo elemento
2. Secondo elemento

- Bisogna fare attenzione alla gerarchia degli elementi
- L'apertura del tag `<ol>` deve venire all'inizio della lista
- La chiusura del tag `</ol>` deve venire alla fine
- In mezzo, ogni elemento della lista deve essere aperto "`<li>`" e poi chiuso "`</li>`" prima che se ne apra un altro

```
<ol>
  <li></li>
</ol>
```

TABELLE

```
<table>
  <thead>
      <tr>
          <th>Colonna 1</th>
          <th>Colonna 2</th>
      </tr>
  </thead>
  <tbody>
      <tr>
          <td>Cella 1</td>
          <td>Cella 2</td>
      </tr>
  </tbody>
</table>
```

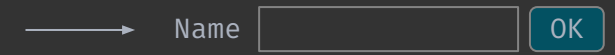
Colonna 1	Colonna 2
Cella 1	Cella 2

Form

Uno dei fattori che ha decretato il successo del Web è senz'altro la **possibilità di interagire**: la possibilità cioè di iscriversi a servizi di vario tipo (ad esempio mailing list), ma soprattutto di partecipare a vere e proprie comunità virtuali

Per organizzare questo genere di servizi è necessario **raccogliere i dati** dell'utente: per farlo si utilizzano, in maniera molto semplice, i moduli (cioè i form)

```
<form action="#">
  <label for="name">Name</label>
  <input id="name" type="text">
  <input type="submit" value="OK">
</form>
```



Immagini

Il tag <img> rappresenta il principale elemento per inserire un'immagine in una pagina HTML

```
<img src="image.png" alt="Immagine" />
```

- Elementi inline-block
- Non necessitano chiusura

LINK

I link sono “il ponte” che consente di passare da un testo all’altro

Link con riferimento ad altro sito web

```
<a href="https://it.wikipedia.org/">Wikipedia</a>
```

Link ad una pagina all’interno del progetto

```
<a href="pagina1.html">Pagina 1</a>
```

Link ad una pagina all’interno di un cartella

```
<a href="pages/pagina2.html">Pagina 2</a>
```

- Elementi inline
- Necessita chiusura

<a>

File paths

Il “File path”, o percorso del file, descrive la posizione di una risorsa all’interno della struttura di cartelle del sito web.

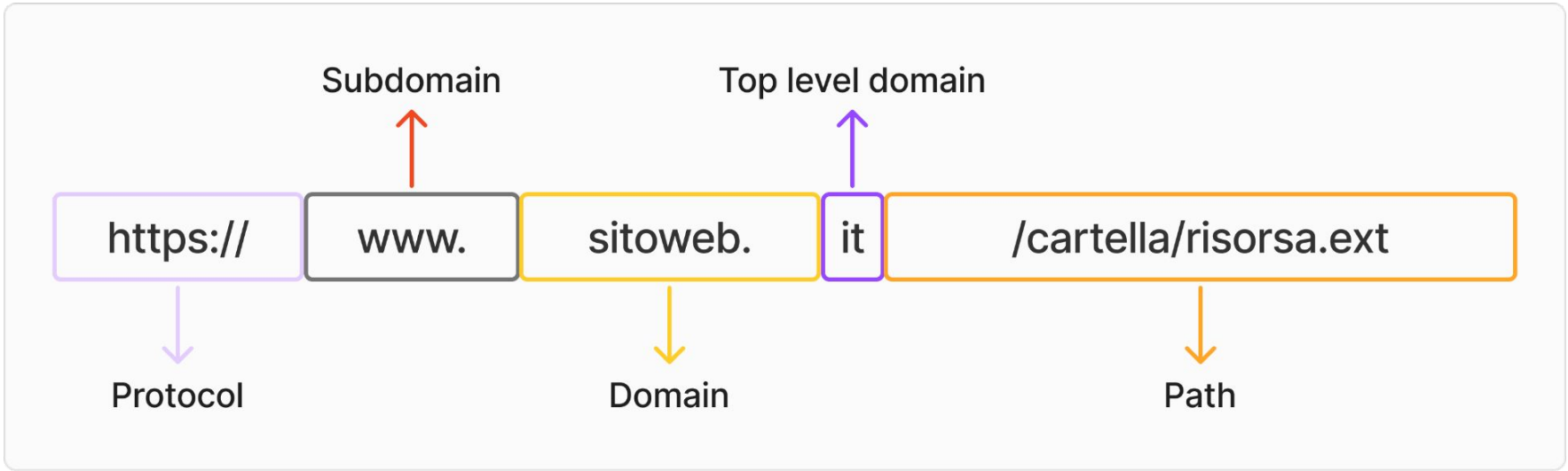
Esistono due tipologie di file path

- **Absolute path:** Il percorso assoluto è l’URL completo alla risorsa
- **Relative path:** Il percorso relativo punta alla risorsa relativamente alla pagina corrente

Le *best practice* consigliano utilizzare i ***relative path* salvo non sia possibile fare diversamente** (e.g. link a risorsa esterna)

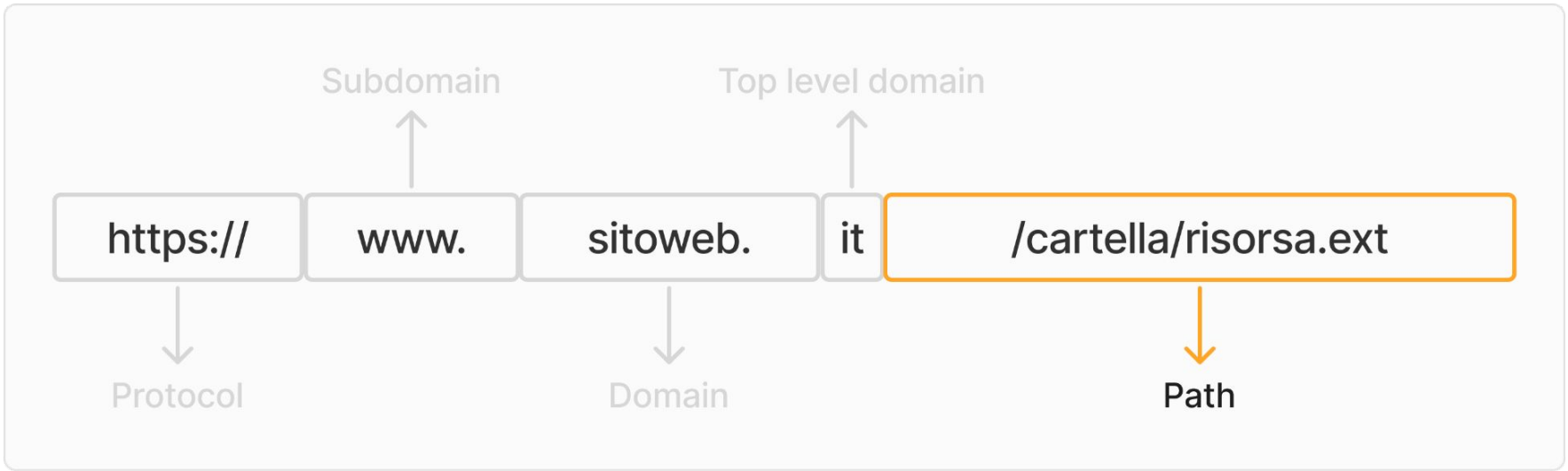
Absolute path

URL



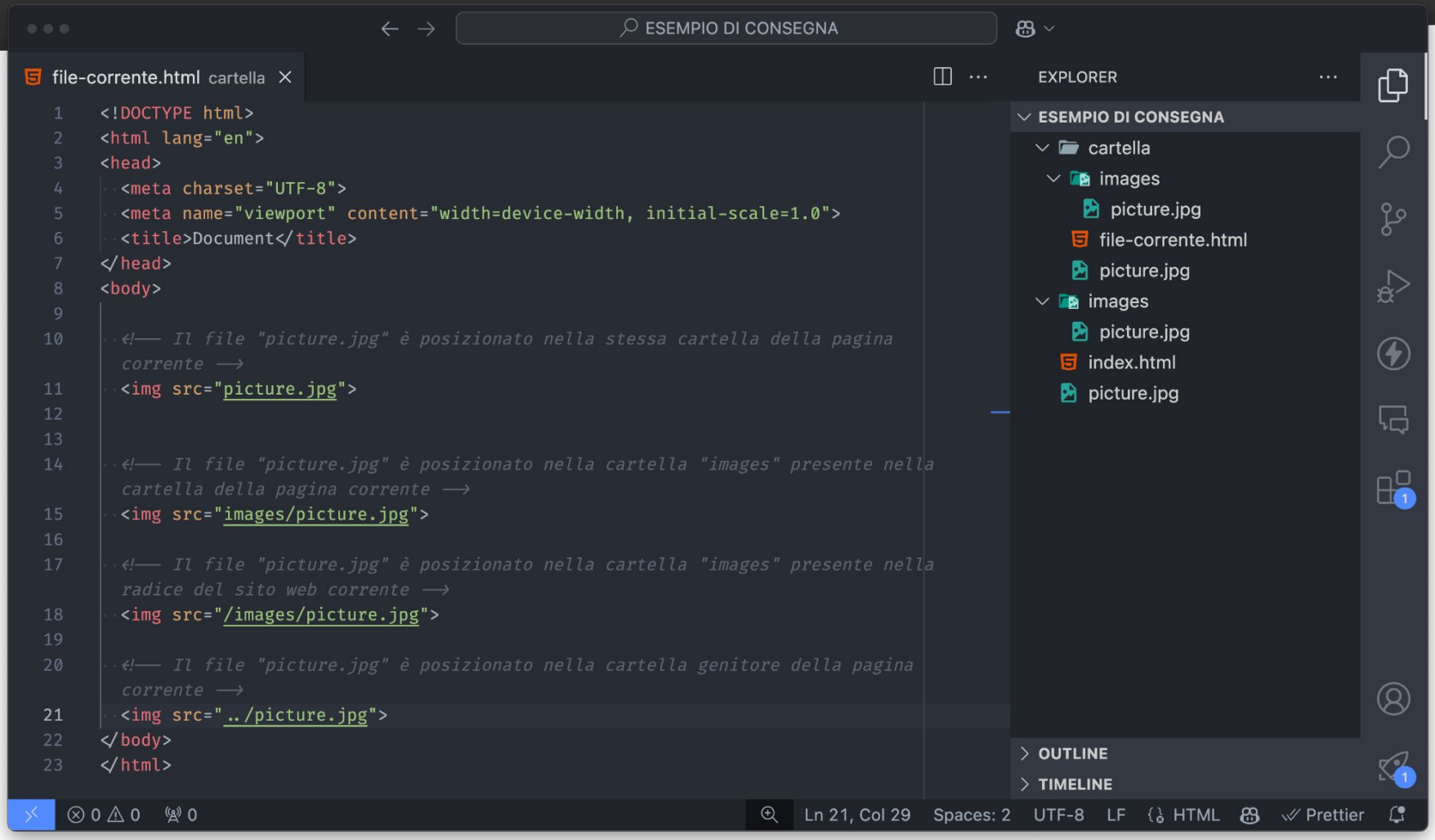
Relative path

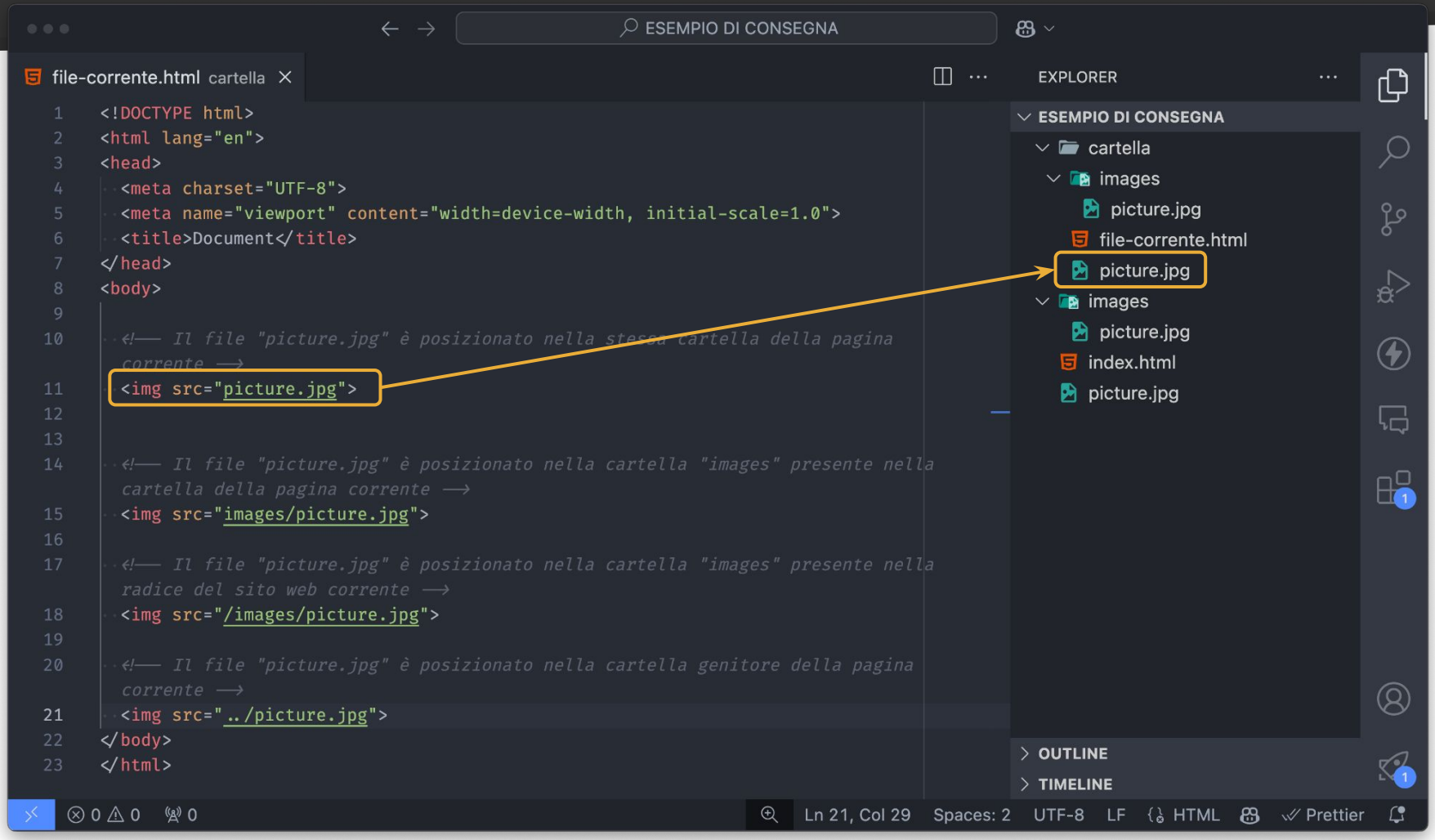
URL

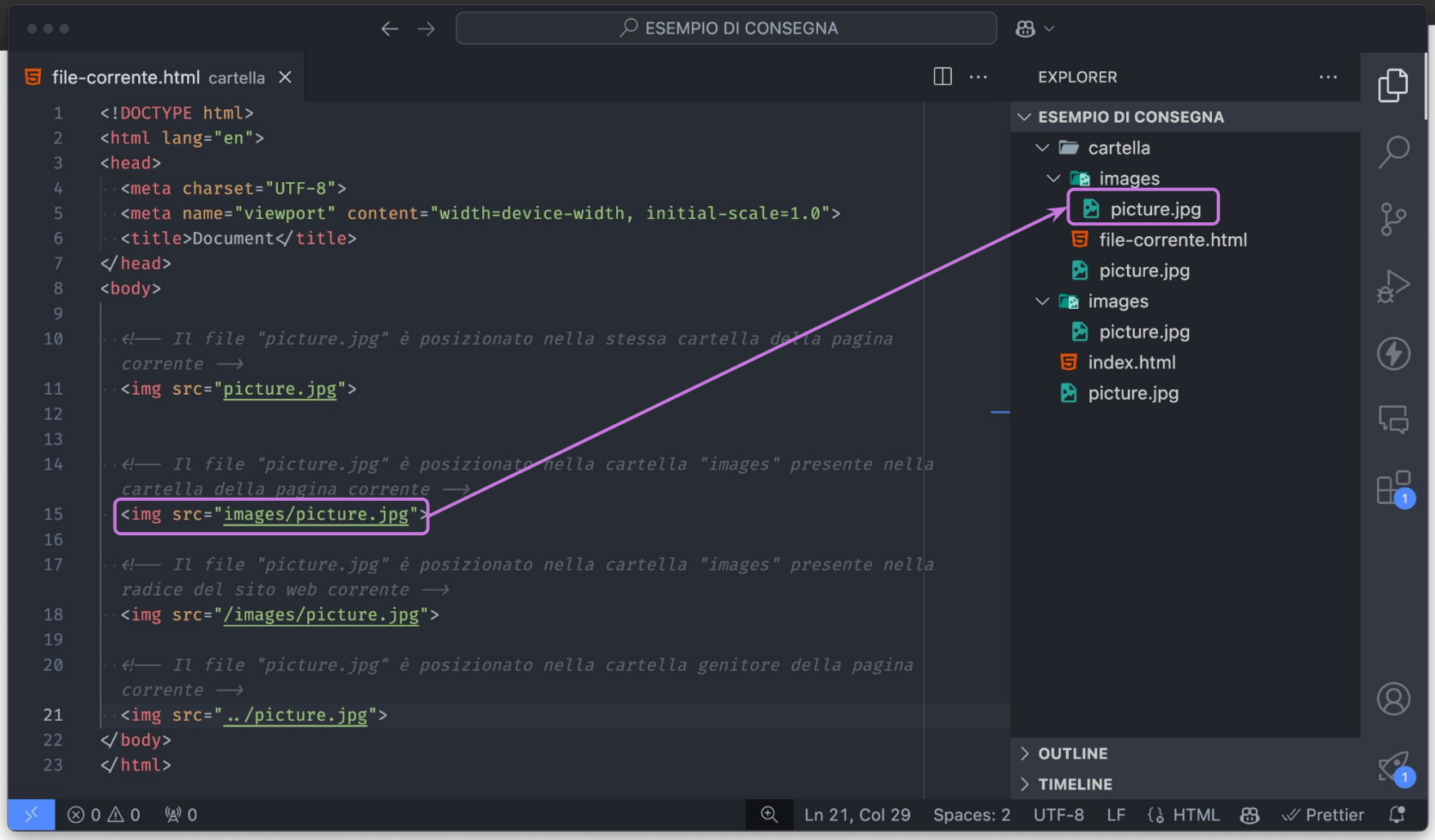


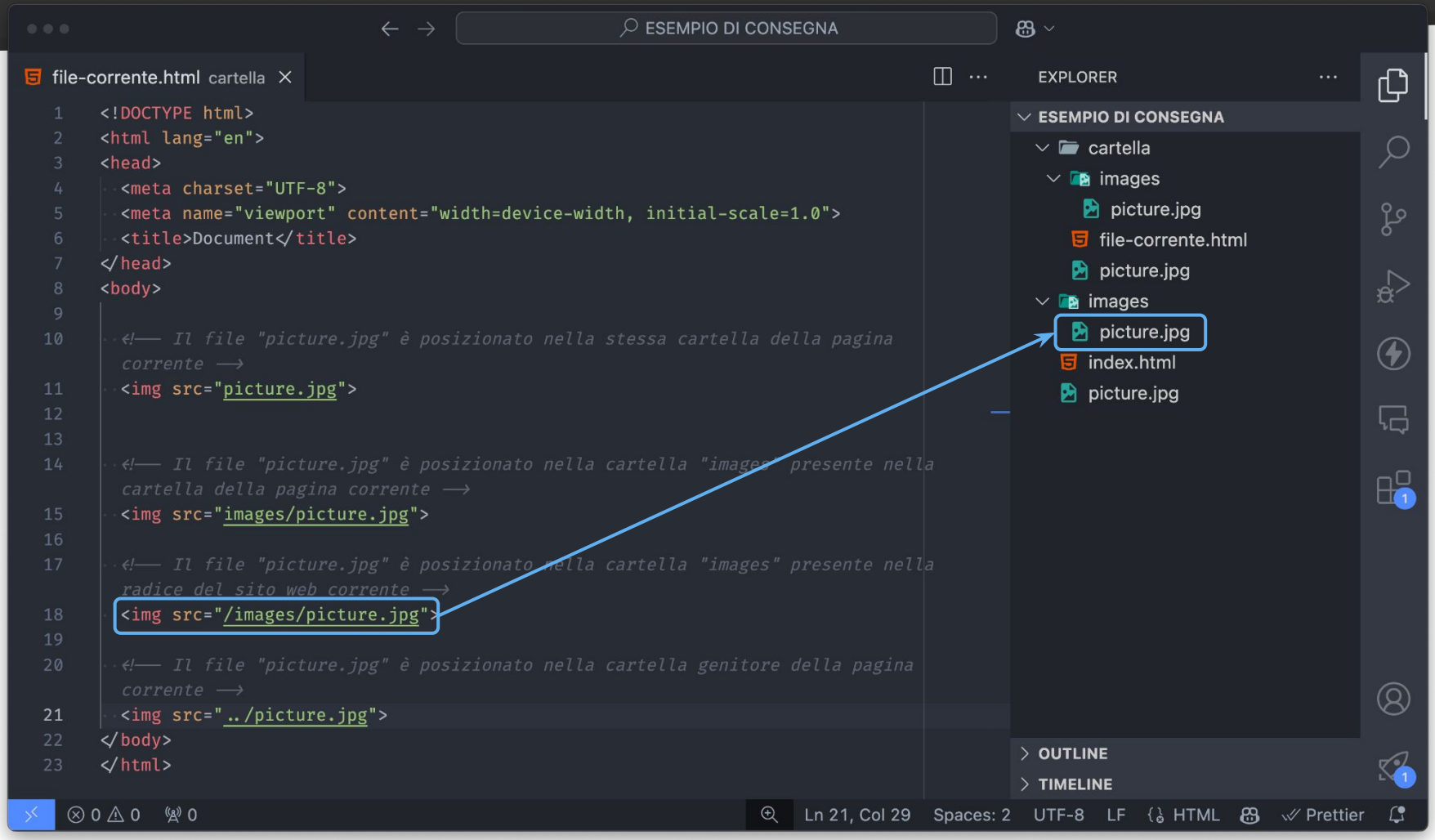
Tipi di relative path

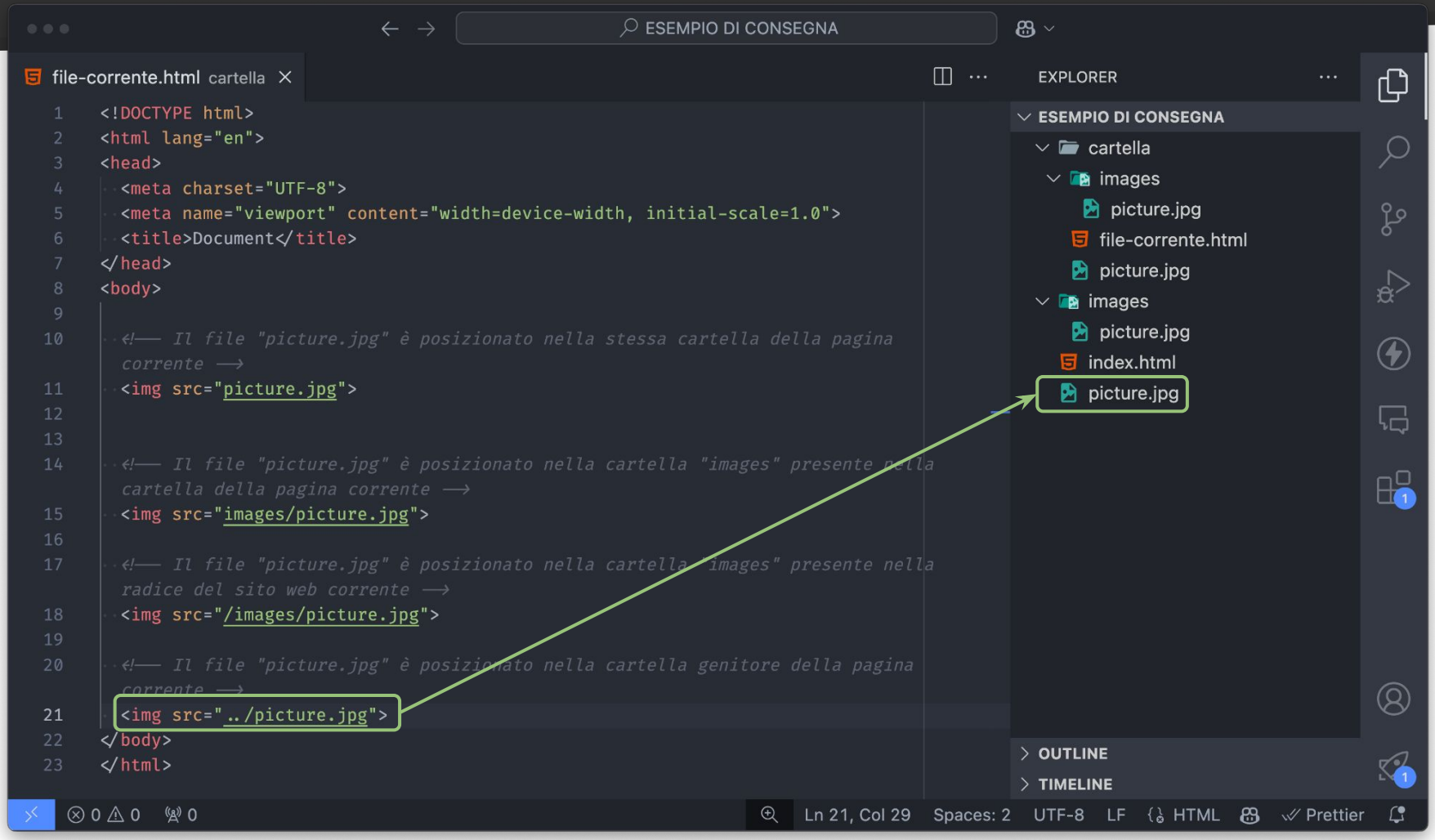
Path	Descrizione
`<img src="**picture.jpg**">`	Il file “picture.jpg” è posizionato nella stessa cartella della pagina corrente
`<img src="**images/picture.jpg**">`	Il file “picture.jpg” è posizionato nella cartella “images” presente nella cartella della pagina corrente
`<img src="**/images/picture.jpg**">`	Il file “picture.jpg” è posizionato nella cartella “images” presente nella radice del sito web corrente
`<img src="**../picture.jpg**">`	Il file “picture.jpg” è posizionato nella cartella genitore della pagina corrente


ESEMPIO DI CONSEGNA
file-corrente.html cartella
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Document</title>
7 </head>
8 <body>
9
10 <!-- Il file "picture.jpg" è posizionato nella stessa cartella della pagina corrente -->
11 <img src="picture.jpg">
12
13
14 <!-- Il file "picture.jpg" è posizionato nella cartella "images" presente nella cartella della pagina corrente -->
15 <img src="images/picture.jpg">
16
17 <!-- Il file "picture.jpg" è posizionato nella cartella "images" presente nella radice del sito web corrente -->
18 <img src="/images/picture.jpg">
19
20 <!-- Il file "picture.jpg" è posizionato nella cartella genitore della pagina corrente -->
21 <img src="../picture.jpg">
22 </body>
23 </html>
EXPLORER
ESEMPIO DI CONSEGNA
cartella
images
picture.jpg
file-corrente.html
picture.jpg
images
picture.jpg
index.html
picture.jpg
OUTLINE
TIMELINE
0 0 0
Ln 21, Col 29
Spaces: 2
UTF-8
LF
HTML
Prettier


ESEMPIO DI CONSEGNA
file-corrente.html cartella
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Document</title>
7 </head>
8 <body>
9
10 <!-- Il file "picture.jpg" è posizionato nella stessa cartella della pagina corrente -->
11 <img src="picture.jpg">
12
13
14 <!-- Il file "picture.jpg" è posizionato nella cartella "images" presente nella cartella della pagina corrente -->
15 <img src="images/picture.jpg">
16
17 <!-- Il file "picture.jpg" è posizionato nella cartella "images" presente nella radice del sito web corrente -->
18 <img src="/images/picture.jpg">
19
20 <!-- Il file "picture.jpg" è posizionato nella cartella genitore della pagina corrente -->
21 <img src="../picture.jpg">
22 </body>
23 </html>
EXPLORER
ESEMPIO DI CONSEGNA
cartella
images
picture.jpg
file-corrente.html
picture.jpg
images
picture.jpg
index.html
picture.jpg
OUTLINE
TIMELINE
Ln 21, Col 29
Spaces: 2
UTF-8
LF
HTML
Prettier


ESEMPIO DI CONSEGNA
file-corrente.html cartella
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Document</title>
7 </head>
8 <body>
9
10 <!-- Il file "picture.jpg" è posizionato nella stessa cartella della pagina corrente -->
11 <img src="picture.jpg">
12
13
14 <!-- Il file "picture.jpg" è posizionato nella cartella "images" presente nella cartella della pagina corrente -->
15 <img src="images/picture.jpg">
16
17 <!-- Il file "picture.jpg" è posizionato nella cartella "images" presente nella radice del sito web corrente -->
18 <img src="/images/picture.jpg">
19
20 <!-- Il file "picture.jpg" è posizionato nella cartella genitore della pagina corrente -->
21 <img src="../picture.jpg">
22 </body>
23 </html>
EXPLORER
ESEMPIO DI CONSEGNA
cartella
images
picture.jpg
file-corrente.html
picture.jpg
images
picture.jpg
index.html
picture.jpg
OUTLINE
TIMELINE
Ln 21, Col 29
Spaces: 2
UTF-8
LF
HTML
Prettier


ESEMPIO DI CONSEGNA
file-corrente.html cartella
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Document</title>
7 </head>
8 <body>
9
10 <!-- Il file "picture.jpg" è posizionato nella stessa cartella della pagina corrente -->
11 <img src="picture.jpg">
12
13
14 <!-- Il file "picture.jpg" è posizionato nella cartella "images" presente nella cartella della pagina corrente -->
15 <img src="images/picture.jpg">
16
17 <!-- Il file "picture.jpg" è posizionato nella cartella "images" presente nella radice del sito web corrente -->
18 <img src="/images/picture.jpg">
19
20 <!-- Il file "picture.jpg" è posizionato nella cartella genitore della pagina corrente -->
21 <img src="../picture.jpg">
22 </body>
23 </html>
EXPLORER
ESEMPIO DI CONSEGNA
cartella
images
picture.jpg
file-corrente.html
picture.jpg
images
picture.jpg
index.html
picture.jpg
OUTLINE
TIMELINE
Ln 21, Col 29
Spaces: 2
UTF-8
LF
HTML
Prettier


ESEMPIO DI CONSEGNA
file-corrente.html cartella
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4 <meta charset="UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1.0">
6 <title>Document</title>
7 </head>
8 <body>
9
10 <!-- Il file "picture.jpg" è posizionato nella stessa cartella della pagina corrente -->
11 <img src="picture.jpg">
12
13
14 <!-- Il file "picture.jpg" è posizionato nella cartella "images" presente nella cartella della pagina corrente -->
15 <img src="images/picture.jpg">
16
17 <!-- Il file "picture.jpg" è posizionato nella cartella "images" presente nella radice del sito web corrente -->
18 <img src="/images/picture.jpg">
19
20 <!-- Il file "picture.jpg" è posizionato nella cartella genitore della pagina corrente -->
21 <img src="../picture.jpg">
22 </body>
23 </html>
EXPLORER
ESEMPIO DI CONSEGNA
cartella
images
picture.jpg
file-corrente.html
picture.jpg
images
picture.jpg
index.html
picture.jpg
OUTLINE
TIMELINE
Ln 21, Col 29
Spaces: 2
UTF-8
LF
HTML
Prettier

GERARCHIA DEI TAG

Il tag <body>

Il <body> contiene tutti gli elementi del documento HTML destinati a comparire nella pagina

Gli elementi nel body avranno una struttura a “matrioska”, contenendosi l’un l’altro

Gerarchia dei tag (1)

Il fatto che i tag si contengano l'un l'altro crea una gerarchia

```
<div>  <!-- Genitore (parent) -->
  <span>Lorem ipsum</span>  <!-- Figli (child) di <div> -->
  <img class="img-responsive" />   <!-- Fratelli (siblings) tra loro -->
</div>
```

Gerarchia dei tag (2)

```
<html>  <!-- Elemento radice (root) -->
<body>  <!-- Antenato (ancestor) di <span> e <img> -->
  <div>
    <span>Lorem ipsum</span>  <!-- Discendenti (descendants) di <body> -->
    <img class="img-responsive" />
  </div>
</body>
</html>
```

Indentazione dei tag

Allo sviluppatore conviene indentare il codice in una maniera che rispetti la gerarchia dei tag.

Questo non è necessario per l'interpretazione da parte del browser, ma rende lo sviluppo più comodo.

```
<div id="wrapper">
  <div id="header">
    <ul>
      <li>Menu Item</li>
      <li>Menu Item</li>
      <ul>
        <li>Sub Menu Item</li>
        <li>Sub Menu Item</li>
      </ul>
    </ul>
  </div>

  <div id="main-content">
    <p>lorem ipsum</p>
  </div>

  <div id="sidebar">
    <p>lorem ipsum</p>
  </div>
</div>
```

Ereditarietà

Lo standard fissato dal W3C per la rappresentazione dei documenti strutturati si chiama **Document Object Model** (DOM).

La gerarchia dei tag è un meccanismo potente e necessario nell'applicazione degli **stili** (CSS).

Gli stili sono caratterizzati da **ereditarietà**.

Molte proprietà impostate per un elemento, dunque, saranno ereditate dai suoi discendenti.

Ma cosa sono gli stili?

INTRODUZIONE AI CSS

Cascading Style Sheets

L’acronimo **CSS** sta per **Cascading Style Sheets** (fogli di stile a cascata); è un linguaggio di stile per i documenti web

I CSS istruiscono un browser su come il documento debba essere presentato all’utente, per esempio definendone i font, i colori, le immagini di sfondo, il layout, il posizionamento delle colonne o di altri elementi sulla pagina, etc.

Nell’HTML standard (senza CSS)

Un tempo gli stili venivano definiti direttamente nell’HTML. Funzionava in questo modo:

1. le istruzioni di formattazione del testo dovevano essere inserite direttamente nel tag <body>, associate al tag da formattare tramite attributi dei tag HTML o tag HTML aggiuntivi
2. dovevano essere ripetute a ogni variazione della formattazione (ad ogni tag) e per ogni istruzione serviva un attributo

```
<p><font color="#FF0000" size="10">
    <b>tutto questo testo avrà corpo 10,
    colore rosso e grassetto</b>
</font></p>
```


tutto questo testo avrà corpo 10, colore rosso e grassetto

Con i CSS, invece...

1. Si definisce la regola di formattazione una sola volta; si inserisce il CSS nel tag <head> o in un file esterno con estensione “.css”
2. Nel foglio di stile, la regola CSS può essere applicata direttamente a un tag (ad esempio, possiamo assegnare determinati stili a tutti gli elementi “p”)
3. Oppure, può essere associata a un nome arbitrario. In questo caso, nel file HTML possiamo applicarla tramite un attributo (class o id) di uno specifico tag

Con i CSS, invece...

esempio

Per esempio, nel tag <head> o in un file css esterno si definiscono gli stili

```
/* styles.css */

p {
  font-size: 15px;
  font-weight: bold;
  color: #FF0000;
}

.blu {
  color: #0099FF;
}
```

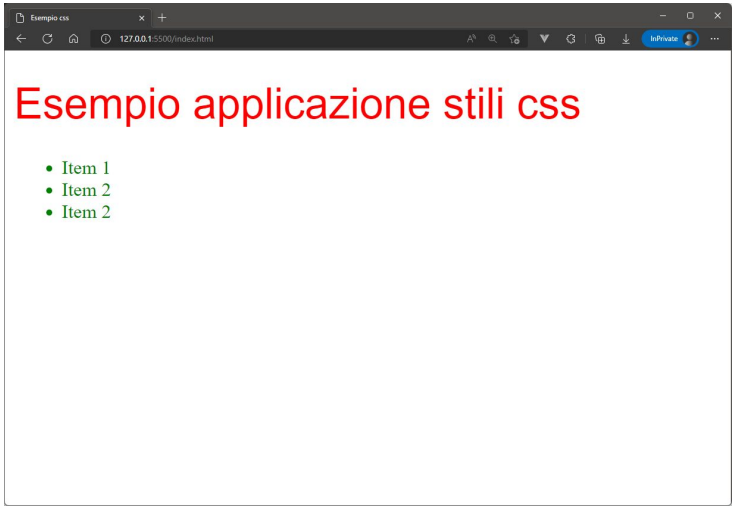
e nel <body> si applicano le regole

```
<!-- index.html -->

<p>questo testo avrà
dimensione 15 pixel,
grassetto, colore rosso
(#FF0000)</p>

<p class="blu">questo testo
avrà dimensione 15 pixel,
grassetto, colore blu
(#0099FF)</p>
```

Esempio di CSS




```css
/* styles.css */

body {
  background: white;
  color: black;
}

li {
  color: green;
}

h1 {
  color: red;
  font: 36px Arial, sans-serif;
}
```

Tipi di CSS

1. **CSS in linea (inline)**: quando lo stile viene specificato in prossimità del testo da formattare (nel tag body)
2. **CSS interni(embedded)**: quando gli stili vengono definiti all'interno della pagina html nel tag <head> e possono essere richiamati solo nel tag <body> della pagina stessa
3. **CSS esterni(external)**: quando gli stili sono definiti in un file esterno e possono essere condivisi da diverse pagine del sito; un singolo foglio stile contenente le regole di formattazione viene applicato a più pagine

CSS in linea (inline)

Quando lo stile viene specificato in prossimità del testo da formattare (nel tag body)

```
<h1 style="color: red; background: black;"> ... </h1>
```

(il loro utilizzo non è soltanto scomodo, ma a tutti gli effetti sconsigliato in quasi tutti i casi)

CSS interni (embedded)

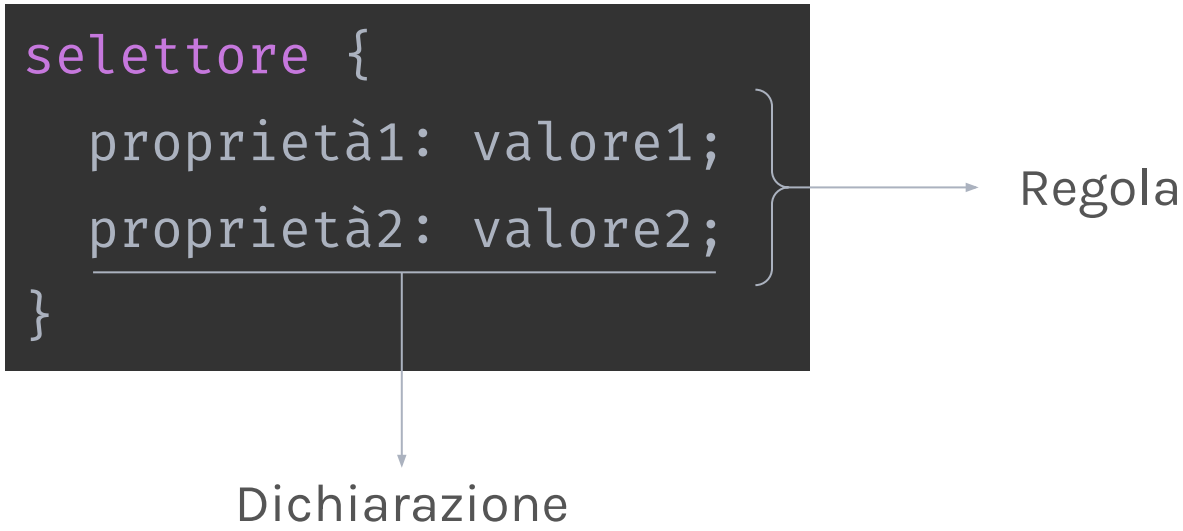
Quando gli stili vengono definiti all'interno della pagina html nel tag <head> e possono essere richiamati solo nel tag <body> della pagina stessa

```
<html>
   <head>
      <style type="text/css">
         body {background: white;}
         p {color: black;}
         …
      </style>
   </head>
   <body>
    …
```

CSS esterni (external)

Quando gli stili sono definiti in un file esterno e possono essere condivisi da diverse pagine del sito; un singolo foglio stile contenente le regole di formattazione viene applicato a più pagine

```
<html>
   <head>
      <link href="style.css" rel="stylesheet" type="text/css">
   </head>
   <body>
    ...
```


selettore {
proprietà1: valore1;
proprietà2: valore2;
}
Regola
Dichiarazione

I selettori di base

h1

#id

.class

```
h1, p, a {
  font-size: 2em;
}
```

La regola sarà applicata a tutti gli elementi <h1>, <p>, <a>

```
.classe1 {
  ...
}
```

La regola sarà applicata agli elementi con classe “classe1”

```
#identificatore1 {
  ...
}
```

La regola sarà applicata all’elemento con id “identificatore1”

```
h1#identificatore2 {
  ...
}
```

La regola sarà applicata all’elemento <h1> con id “identificatore2”

I selettori di relazione (1)

```
/* antenato discendente {dichiarazioni;} */
#contenitore p { color: white; }
```

La regola sarà applicata all'elemento <p> che è discendente dell'elemento con id "contenitore"

```
/* padre > figlio {dichiarazioni;} */
#box>p {
  color: white
}
```

La regola sarà applicata all'elemento <p> che è figlio dell'elemento con id "box"

>

+

~

antenato discendente

I selettori di relazione (2)

```
/* fratello + fratello adiacente {dichiarazioni;} */
h1+h2 {
  color: white;
}
```

La regola sarà applicata all'elemento <h2> posto immediatamente dopo <h1> (allo stesso livello di discendenza)

```
/* fratello ~ fratello {dichiarazioni;} */
h1~h2 {
  color: white;
}
```

La regola sarà applicata agli elementi <h2> posti dopo <h1> (allo stesso livello di discendenza)

>

+

~

antenato

discendente

I selettori: pseudoclassi

:first-child

:nth-child(n)

:not(selector)

```
p:first-child {
  color: white;
}

p:nth-child(2) {
  color: black;
}
```

Le regole saranno applicate, rispettivamente, al <p> che è primo figlio e al <p> che è secondo figlio

```
:not(p) {
  color: white;
}
```

La regola sarà applicata a tutti gli elementi che non sono <p>

I selettori: pseudoelementi

::after

::before

::first-letter

```
p::first-letter {
  color: white;
}
```

La regola sarà applicata alla prima lettera dei paragrafi <p>

```
p::after {
  content: "contenuto";
  color: white
}
```

Genera un elemento che conterrà “contenuto” dopo ogni tag <p>. Ogni elemento avrà color: white

```
p::before {
  content: "contenuto";
  color: white
}
```

Genera un elemento che conterrà “contenuto” prima di ogni tag <p>. Ogni elemento avrà color: white

I selettori di attributo

```
el[attr]
el[attr=val]
el[attr~=val]
el[attr*=val]
```

```
a[rel] {
  background-color: yellow;
}
```

La regola sarà applicata a tutti i tag <a> che hanno un attributo di tipo rel

```
a[rel=nofollow] {
  background-color: red;
}
```

La regola sarà applicata a tutti i tag <a> che hanno un attributo di tipo rel con valore nofollow

```
a[title~=notizie] {
  color: white;
}
```

La regola sarà applicata ai tag <a> che hanno un attributo title con valore che contiene la parola "notizie"
(i.e. <a title="Ultime notizie">)

```
a[rel*=no] {
  color: white;
}
```

La regola sarà applicata ai <a> tag che hanno un attributo rel con valore che contiene la stringa "no"
(i.e. <a rel="nofollow">,
<a rel="norefferer">)

Raggruppare i selettori

```
p, div, a.class1 {
  background-color: yellow;
}
```

La regola sarà applicata a tutti i tag <p>, ai tag e ai tag <a> con classe “class1” (Da non confondere con i selettori per i discendenti)

el, el

Commenti

```
/*Commento nei CSS*/

/*Ad esempio, posso utilizzarlo per segnalare che qui iniziano i css del menù*/

/*MENU*/
```

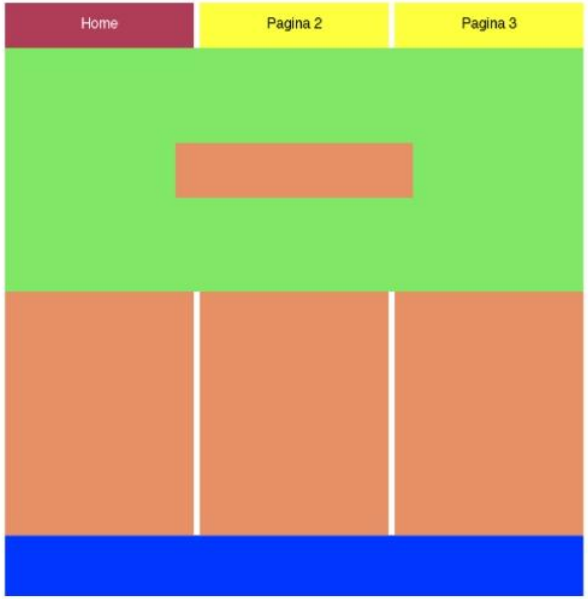
/* */

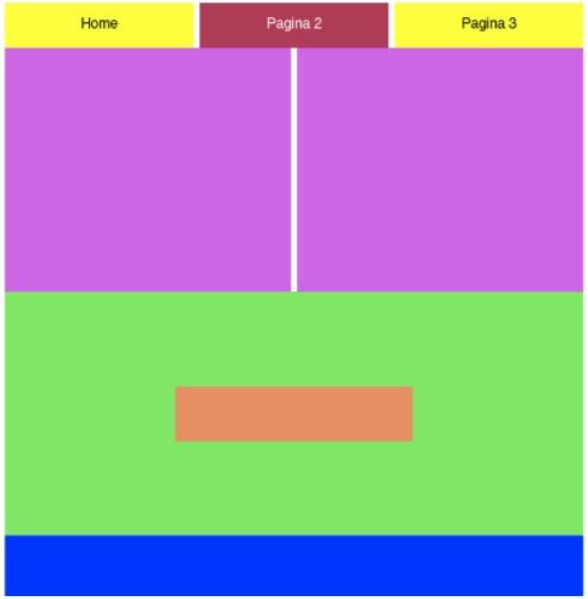
ESERCIZIO 1

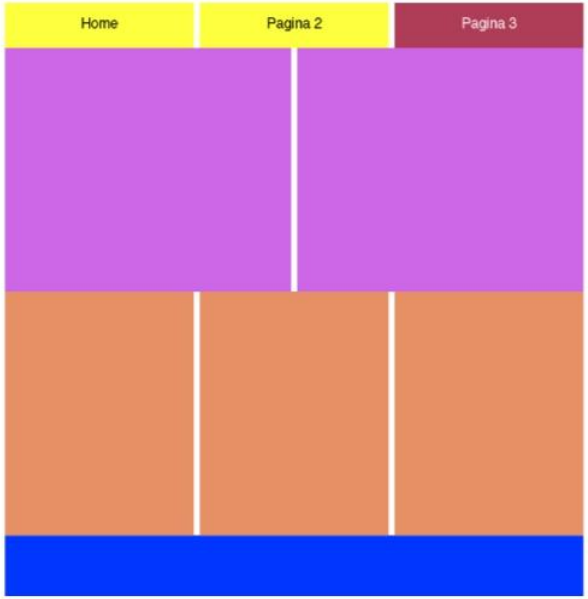
Creare 3 pagine come quelle nella slide successiva, seguendo la struttura in figura. Collegarle fra di loro usando dei link.

Inserire uno pseudoelemento dopo gli elementi di una classe a scelta della pagina.

Utilizzare selettori di relazione per assegnare stili al menù.


Home
Pagina 2
Pagina 3


Home
Pagina 2
Pagina 3


Home
Pagina 2
Pagina 3

RIFERIMENTI UTILI

- Visual Studio Code
- Editor online
- HTML elements reference
- CSS selectors reference
- CSS selector tester
- Tipi di File paths
- FreeCodeCamp
- Google

ESERCITAZIONE 1

FINE