

Sistemi informativi su Web

HTML e CSS (introduzione)

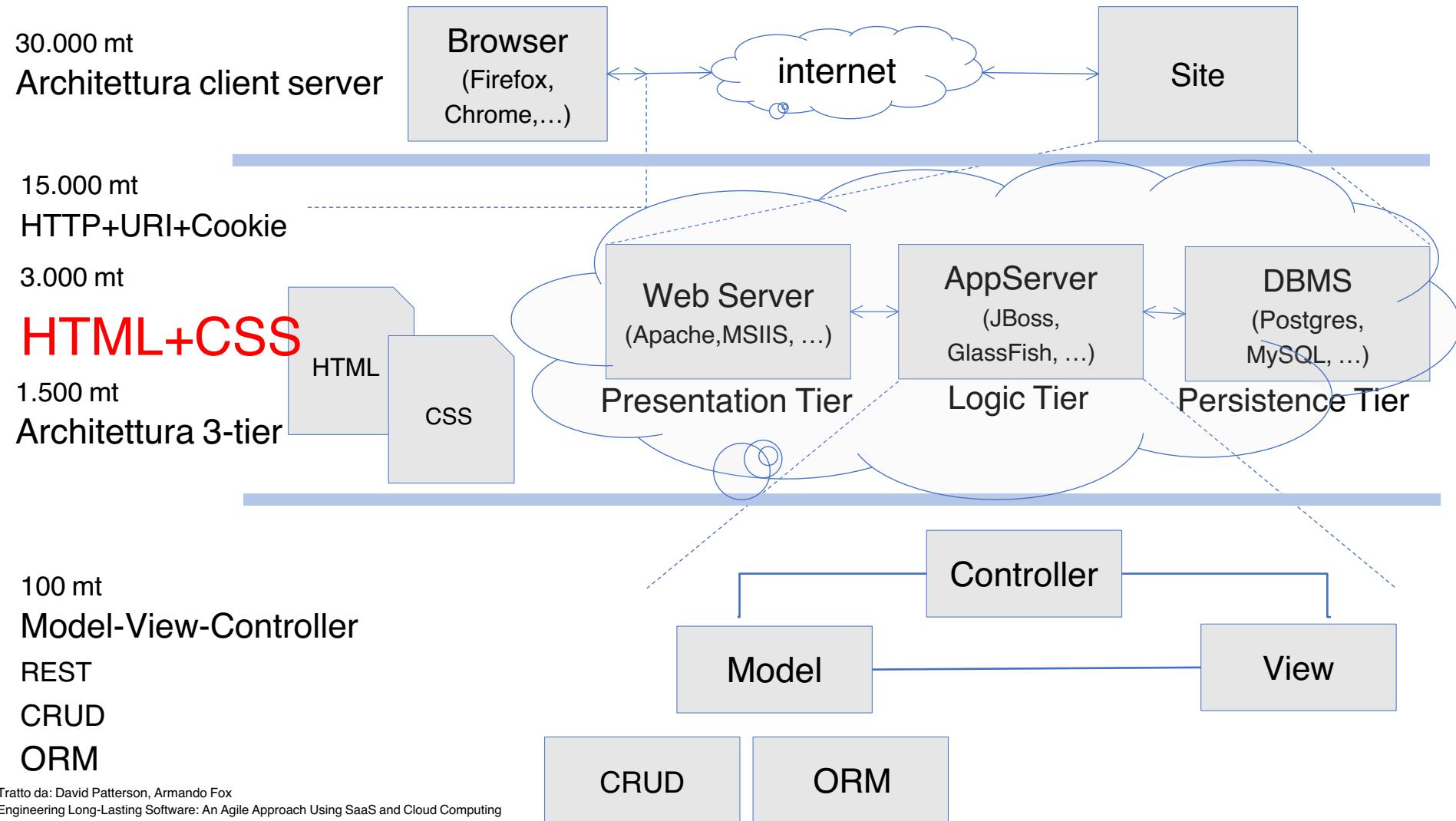
aa 2024-2025



Paolo Merialdo
Università degli Studi Roma Tre

CREDITS: these slides are based on original material by Victoria Kirst

Anatomia di un sistema informativo su Web



HTML (introduzione)

HTML

HTML (Hypertext Markup Language)

- Descrive il **contenuto** e la **struttura** di una pagina Web composta di blocchi chiamati **elementi**.

<**p**>

HTML è <**emem**>

<**img src="mia-foto.png" />**

</**p**>

HTML: struttura base di una pagina

(banale copia e incolla)

```
<!DOCTYPE html>
<html>
  <head>
    <title>SIW</title>
  </head>
  <body>
    ... contents of the page...
  </body>
</html>
```

Salvata in un file ***filename.html***

HTML: struttura base di una pagina

(banale copia e incolla)

Metadati: non
sono visualizzati
dal browser

```
<!DOCTYPE html>
<html>
  <head>
    <title>SIW</title>
  </head>
```

Contenuti
visualizzati nel
browser

```
  <body>
    ... Contenuti della pagina...
  </body>
</html>
```

} Es. **<title>**
appare nel titolo
della barra

Elementi HTML

<**p**>

HTML è <**em**>divertente!</**em**>
<**img** **src**=**"mia-foto.png"** />

</**p**>

- Un elemento ha tag di inizio e fine (<**p**> e </**p**>)
 - Contenuto (content): tutto ciò che è tra i tag di inizio e fine
- Un elemento può essere self-closing (**img**)
- Un elemento può avere attributi (**src**=**"mia-foto.jpg"**)
- Gli elementi possono contenere altri elementi (**p** contiene **eme img**)

Alcuni elementi HTML

(da usare dentro <body>)

Top-level heading h1, h2, ... h6	< h1h1 >>
Paragraph	< pp >
Image	< img src="cover.png" />
Link	< a href="http://google.com" >click here!</ a >
Strong (bold)	< strongstrong >
Emphasis (italic)	He's my < em >brother</ em > and all
Generic elements	< divspanspan > una semantica particolare</ div >

Percorsi assoluti e percorsi relativi

- I valori degli attributi **href** (dell'elemento `a`) e **src** (dell'elemento `img`) sono riferimenti a risorse Web, quindi URL.
Possiamo esprimere attraverso:
 - un percorso assoluto: specifica protocollo, dominio, risorsa. Esempi:
`http://www.google.com`
`http://uniroma3.it/didattica/`
 - un percorso relativo: specifica il percorso con riferimento al percorso della risorsa attuale

Percorsi relativi: esempio

- Supponiamo che nella pagina che si trova all'url <http://uniroma3.it/didattica/> siano presenti questi elementi:

```
<a href="offerta.html">  
<a href="corsi/legge.html">  

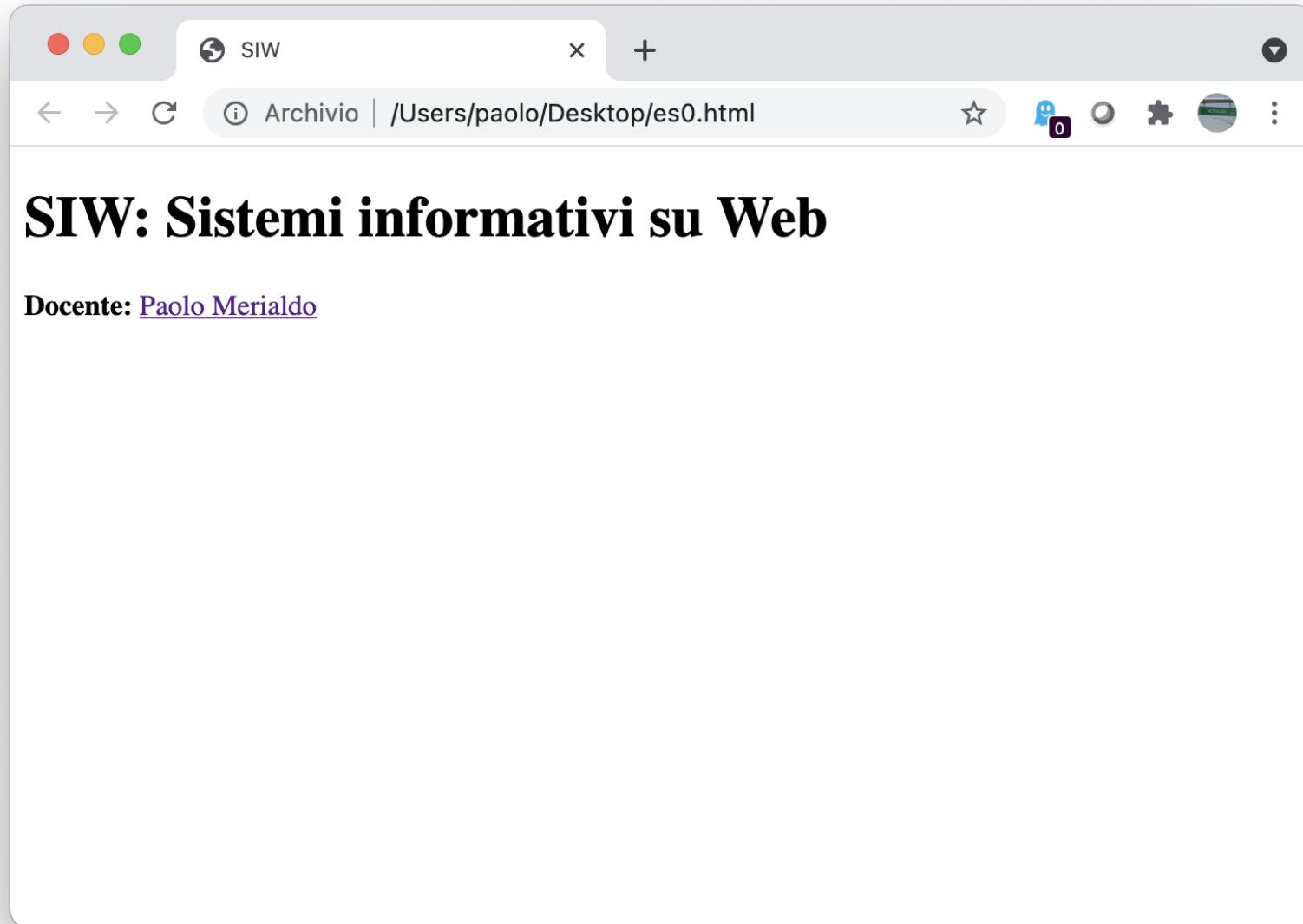
```

- I percorsi assoluti corrispondenti sono:

```
<a href="http://uniroma3.it/didattica/offerta.html"">  
<a href="http://uniroma3.it/didattica/corsi/legge.html"">  

```

Esercizio: semplice pagina web del corso



Esercizio: semplice pagina web del corso

Struttura base pagina HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>SIW</title>
  </head>

  <body>
    ...
  </body>
</html>
```

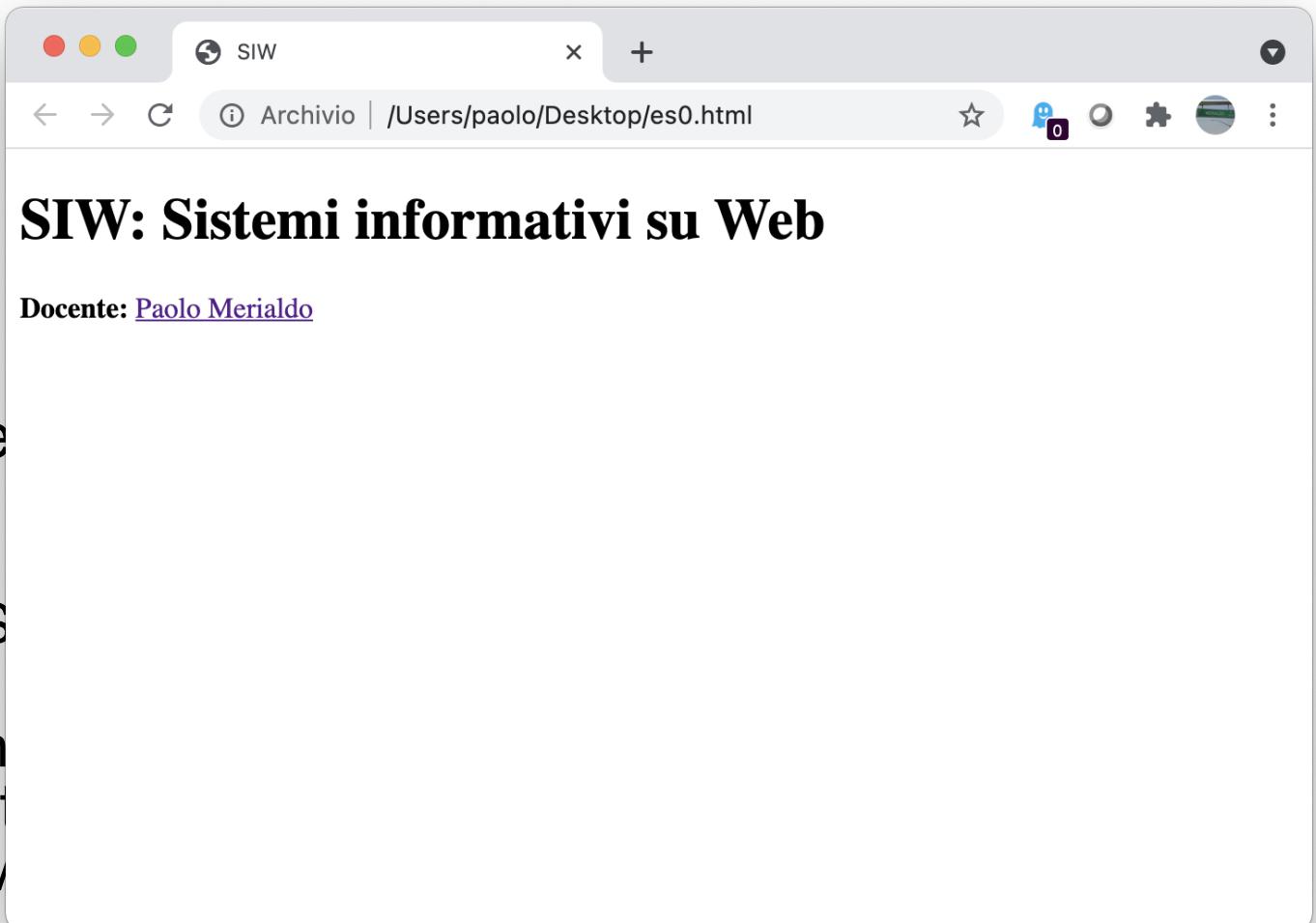
Contenuto testuale della pagina

SIW: Sistemi informativi su Web
Docente: Paolo Merialdo

[JSBin](#)

Soluzione

```
<!DOCTYPE html>
<html>
  <head>
    <title>SIW</title>
  </head>
  <body>
    <h1>SIW: Sistemi informativi su Web</h1>
    <p>
      <strong>Docente:</strong> Paolo Merialdo
    </p>
  </body>
</html>
```



Dettagli "strani"

- Gli spazi bianchi (spazi e invio) collassano in uno spazio ...

```
<h1>SIW:          Sistemi informativi su Web</h1>
<p>
  <strong>Docente:</strong>
    <a href="http://meraldo.inf.uniroma3.it/">
      Paolo Merald
    </a>
  </p>
```

- L'elemento successivo al contenuto di **<h1>** è a capo, la stessa cosa non vale per ****.

Alcuni elementi HTML

Top-level heading: **h1, h2, ... h6**

```
<h1>Moby Dick</h1>  
<h2>Or, the Whale</h2>
```

Moby Dick

Or, the Whale

Paragraph: **p**

```
<p>Call me Ishmael.</p>
```

Call me Ishmael.

Alcuni elementi HTML

Strong (rafforzato): **strong**

```
<strong>Be BOLD</strong>
```

Be BOLD

Emphasis (italico): **em**

```
He's my <em>brother</em> and all
```

He's my *brother* and all

Alcuni elementi HTML

Image: **img**

```

```



Link: **a** (NB: non **link**, **a** sta per anchor)

```
<a href="google.com">click here!</a> click here!
```

Un concetto fondamentale

block vs **inline** display

HTML

HTML (Hypertext Markup Language)

- Descrive il **contenuto** e la **struttura** di una pagina Web composta di blocchi chiamati **elementi**.

```
<p>
    HTML is <em>divertente!!!</em>
    
</p>
```

HTML prevede tre TIPI di elementi (più i metadati)

Tipi di elementi HTML

Le specifiche di HTML assegnano ciascun elemento HTML ad una di queste categorie:

- **block**: ha altezza e larghezza
Es. `<p>`, `<h1>`, ``, ``
- **inline**: non hanno altezza e larghezza
Es. `<a>`, ``, ``
- **inline block**: contenuto inline con altezza e larghezza
``
- **metadata**: informazioni sulla pagina, non visibile
`<title>`, `<meta>`

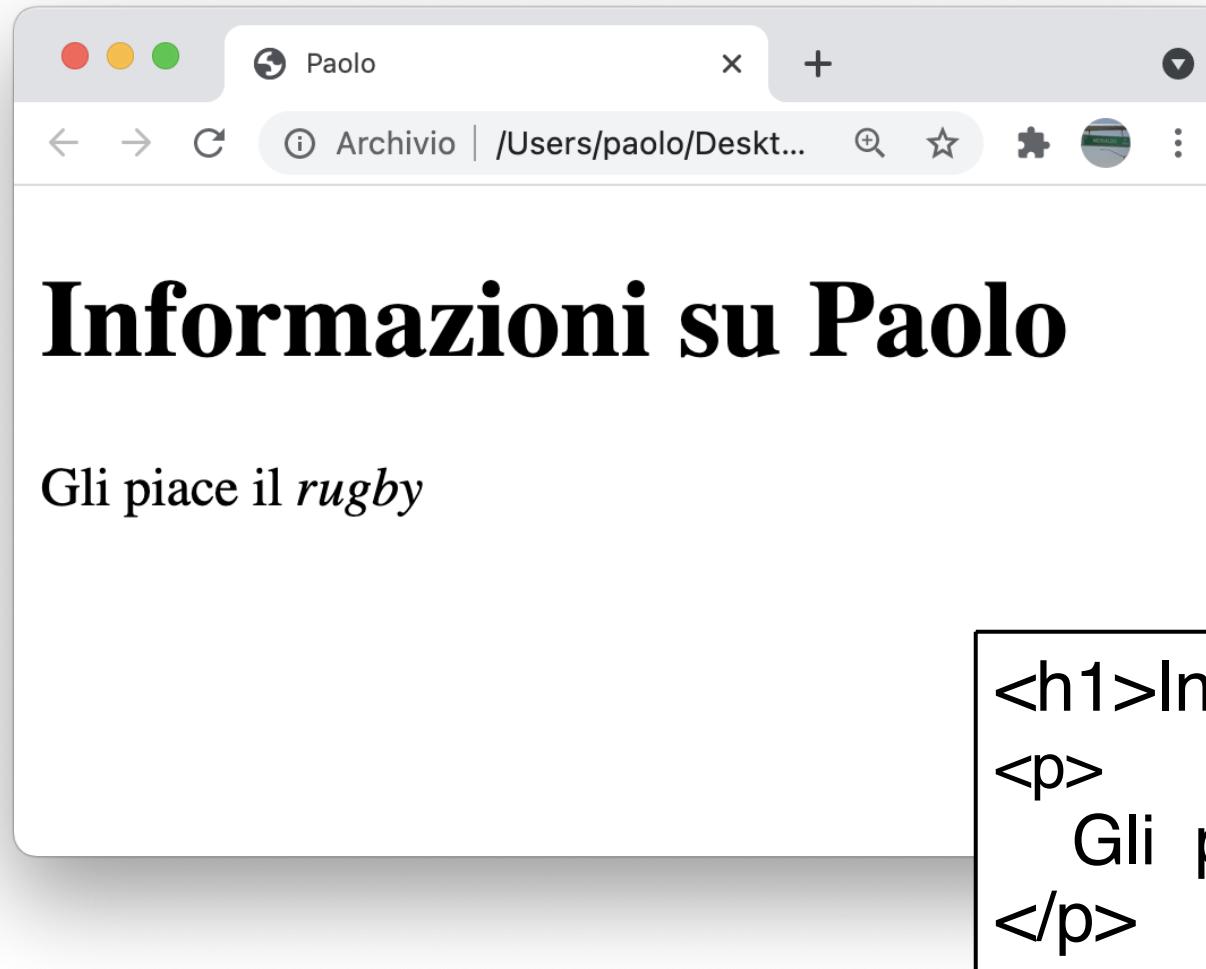
Elementi Block

Esempi: <p>, <h1>, ,

- Occupano l'intera larghezza della pagina (fluttuano dall'alto al basso)
- Hanno altezza (**height**) e larghezza (**width**) modificabili
- Possono avere elementi block o inline come figli



Esempio: Block

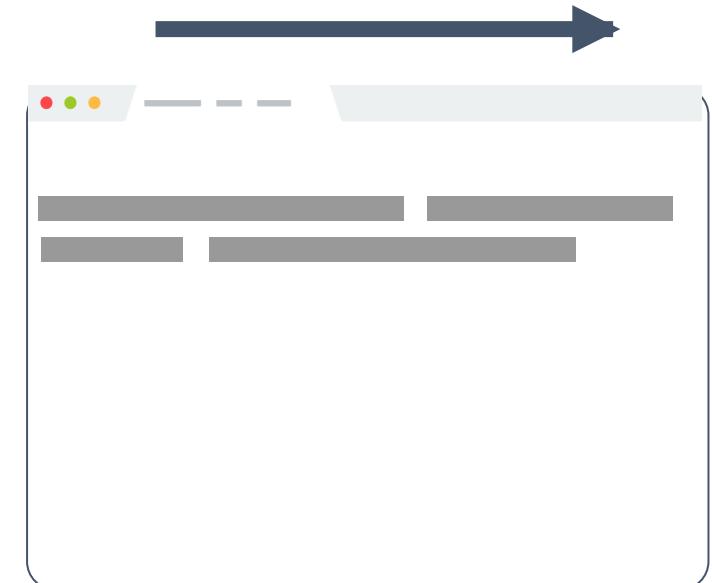


```
<h1>Informazioni su Paolo </h1>
<p>
    Gli piace il <em>rugby</em>
</p>
```

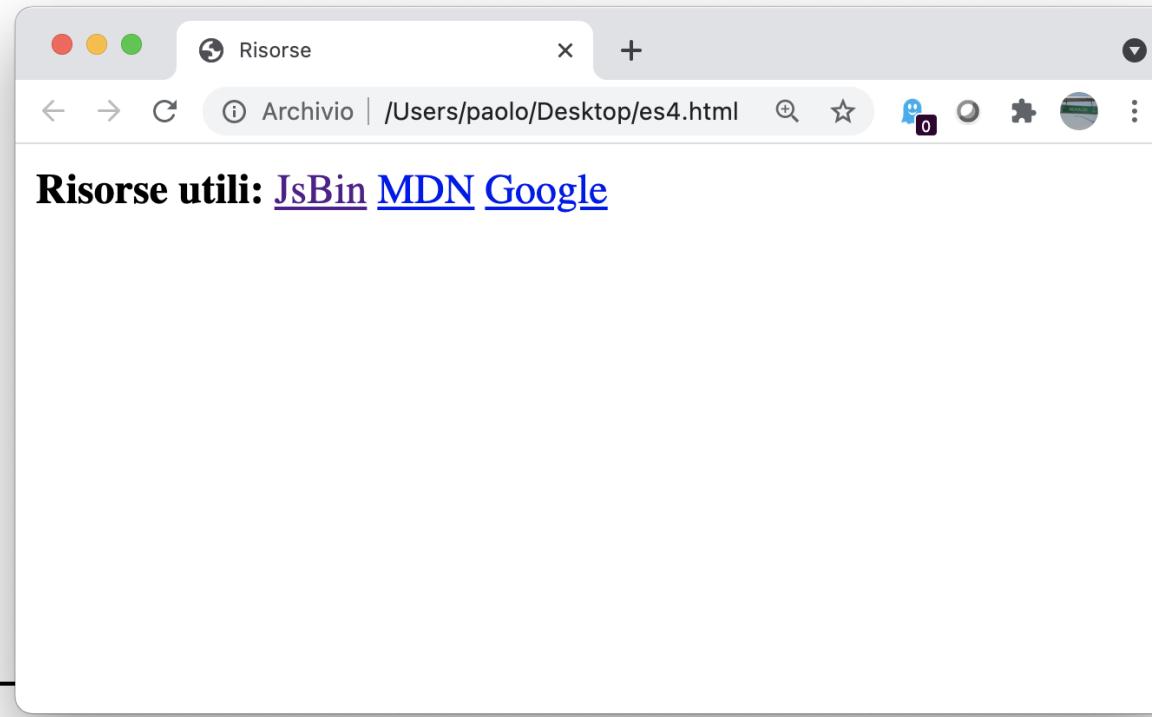
Elementi Inline

Esempi: <a>, ,

- Occupano la larghezza necessaria a visualizzare il contenuto
- Fluiscono da sinistra a destra
- **Non si possono** modificare **height e width**
- **Non si possono** avere elementi block come figli
- **Non è possibile modificarne il flusso**



Esempio: Inline

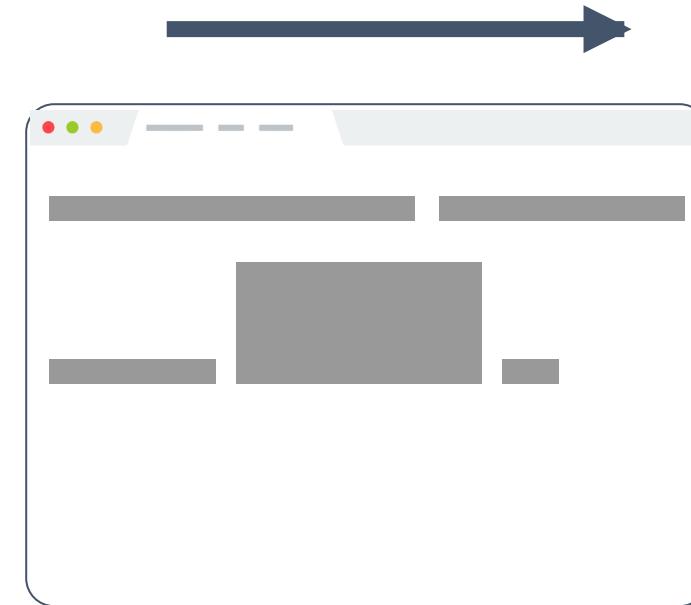


```
<body>
  <strong>Risorse utili:</strong>
  <a href="https://jsbin.com">JsBin</a>
  <a href="https://developer.mozilla.org/en-US/">MDN</a>
  <a href="http://google.com">Google</a>
</body>
```

Elementi inline-block

Esempi: , qualunque elemento con
display: inline-block;

- La larghezza (width) corrisponde alla dimensione del contenuto
- Il flusso e' sulla linea
- **Si possono impostare** height e width
- **Può avere** un elemento block come figlio
- **Può essere posizionato** con opportune proprietà CSS (**float** e position)



Esempio: Inline-block

Q: Come appare nel browser?

```

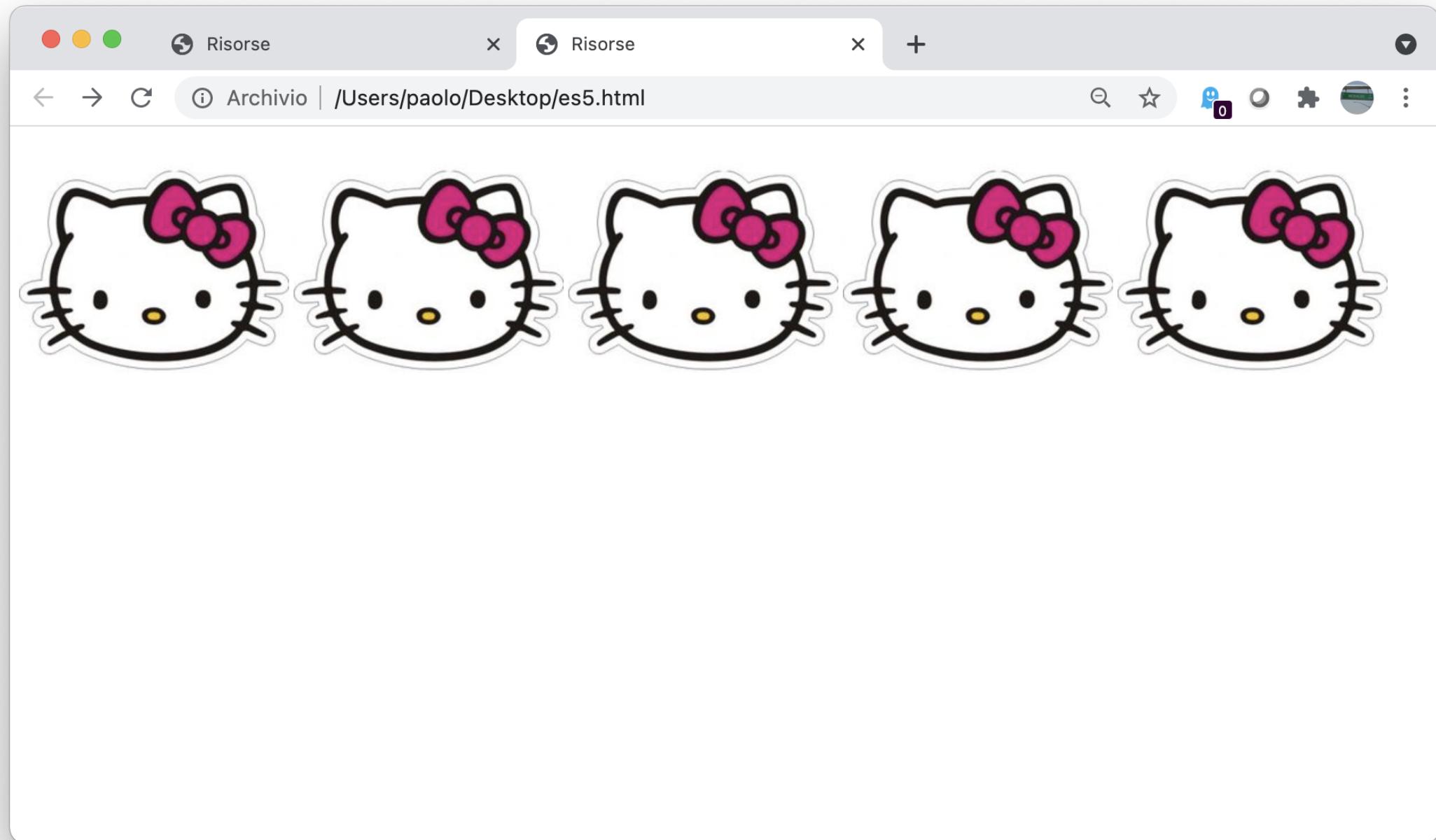




```

<http://i.imgur.com/WJToVGv.jpg> =





FORM HTML

(introduzione)

Form HTML

- Le form HTML permettono di creare interfacce utente per raccogliere dati di input
- I dati vengono raccolti da "controlli"
- Ciascun controllo ha un nome e un valore
 - Il nome è specificato nel codice HTML
 - Il valore corrisponde all'input fornito dall'utente
- La form è associata ad un URL che identifica la risorsa del server che processerà i dati raccolti
- Le coppie nome-valore dei controlli della form sono inviati alla applicazione con una stringa (query string)
- La stringa può essere inviata con il metodo HTTP POST o GET

L'elemento FORM

- Elemento fondamentale: **form**
 - Attraverso gli attributi **action** e **method** permette di specificare
 - l'URL della risorsa che gestisce i dati raccolti (può essere assoluto o relativo)
 - il metodo HTTP della richiesta (nella quale si inviano i dati alla applicazione)
 - All'interno dell'elemento form ci sono elementi corrispondenti ai controlli (oltre ad altri eventuali elementi HTML). Ogni controllo ha un attributo **name**

Esempio

```
<!DOCTYPE html>
<html>
<head>
  <title>form</title>
</head>
<body>
  <h1>Aggiungi un film</h1>
  <form action="/movie" method="POST">
    <p>
      Titolo: <input type="text" name="title"/>
    </p>
    <p>
      Anno:<input type="text" name="year"/>
    </p>
    <p>
      <button type="submit">Conferma</button>
    </p>
  </form>
</body>
</html>
```

Aggiungi un film

Titolo:

Anno:

Conferma



Esempio

```
<form action="/movie" method="POST">
```

Risorsa (sul server)
che processa i dati
raccolti dalla form

Motodo HTTP
attraverso il quale
i dati raccolti
dalla form
vengono mandati
al server

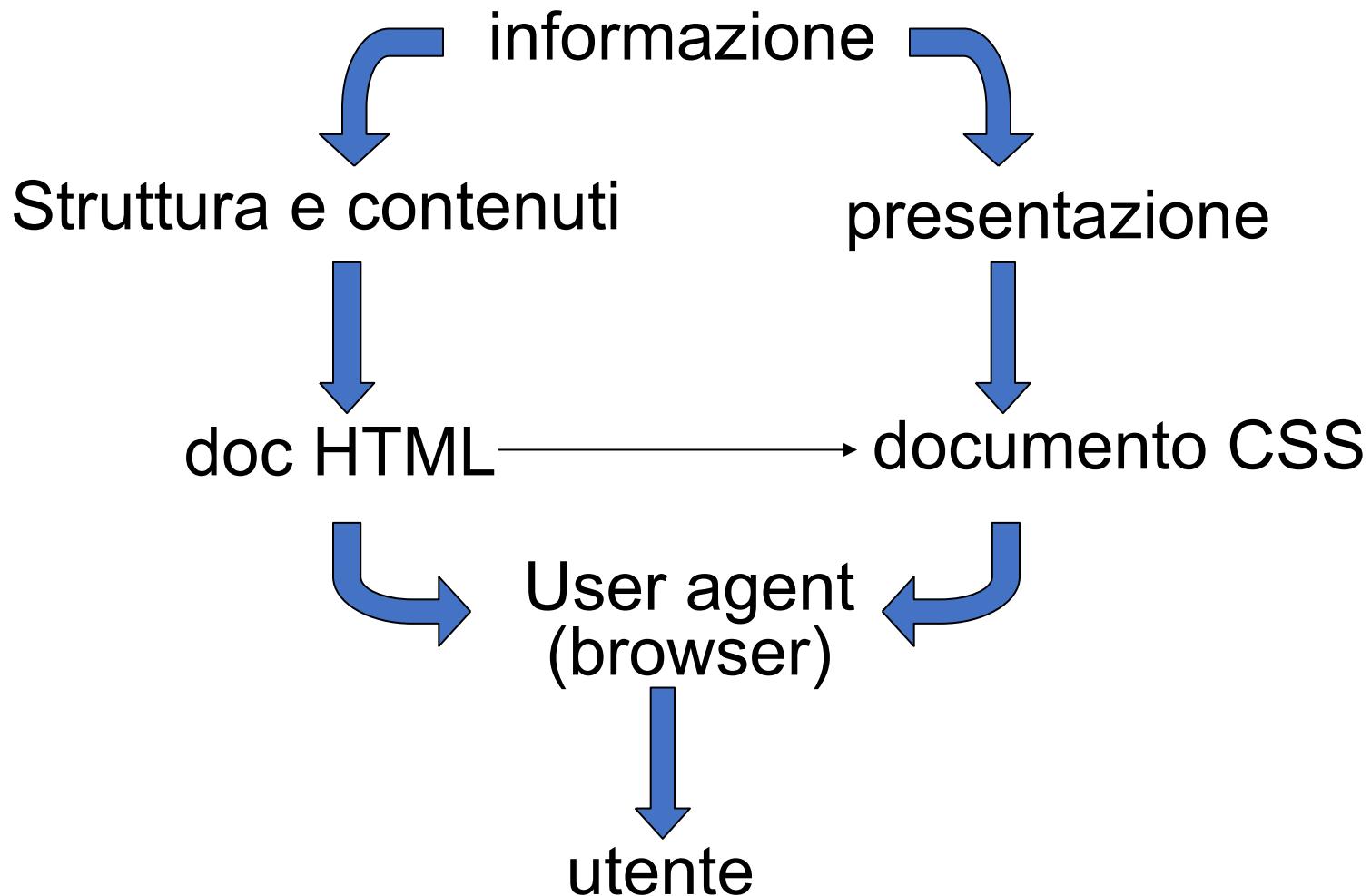
```
<input type="text" name="title"/>
```

Casella di testo

Nome

CSS (introduzione)

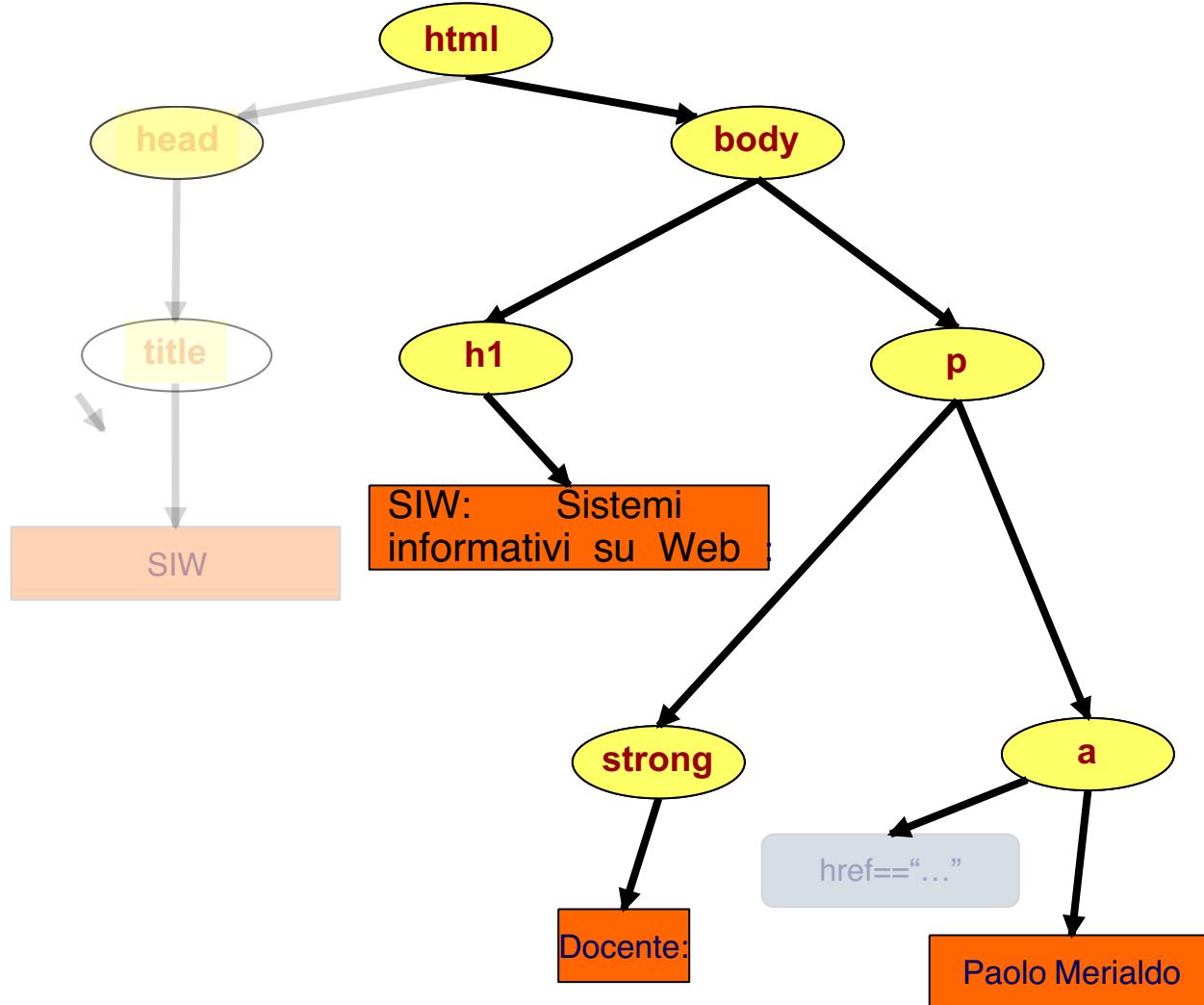
Filosofia HTML + CSS



CSS: idea generale

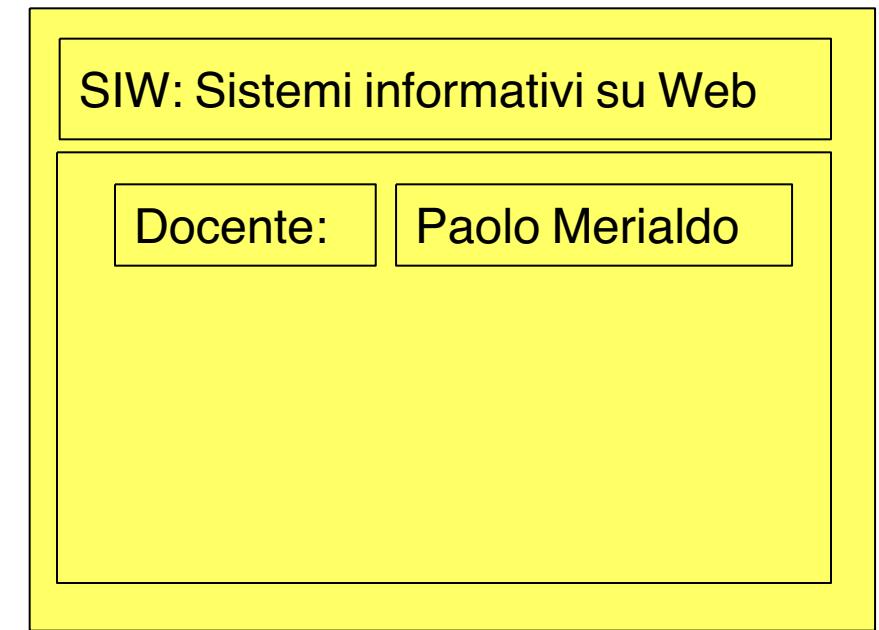
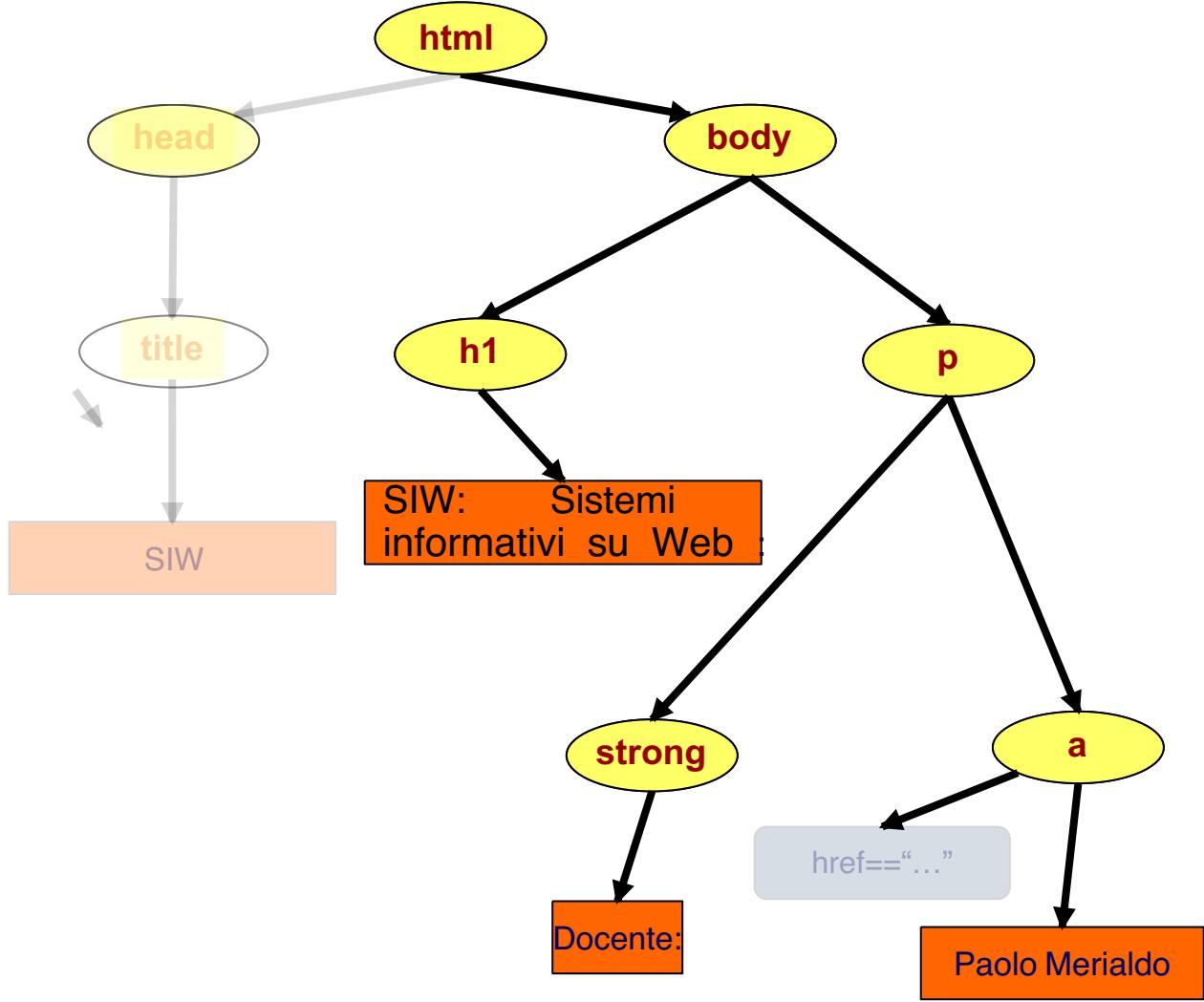
- I documenti HTML corrispondono ad alberi
- In fase di visualizzazione, la nidificazione dei nodi corrisponde ad una nidificazione di riquadri (“box”)
- Un foglio di stile CSS specifica un insieme di regole che definiscono aspetto e disposizione dei riquadri del documento

HTML+CSS



```
<!DOCTYPE html>
<html>
  <head>
    <title>SIW</title>
  </head>
  <body>
    <h1>SIW: Sistemi informativi su Web</h1>
    <p>
      <strong>Docente:</strong>
      <a href="http://merialdo.inf.uniroma3.it/">
        Paolo Merialdo
      </a>
    </p>
  </body>
</html>
```

HTML+CSS



CSS

css Cascading Style Sheets

- Descrive **aspetto e disposizione** degli elementi di una pagina web
- Composto da **regole CSS**, che definiscono insiemi di stili

```
selector {  
    property: value;  
}
```

CSS

Un foglio di stile CSS è composto di **regole di stile**:

```
selector {  
    property: value;  
}
```

selector: Specifica l'elemento HTML a cui applicare lo stile.

property: nome dello stile CSS.

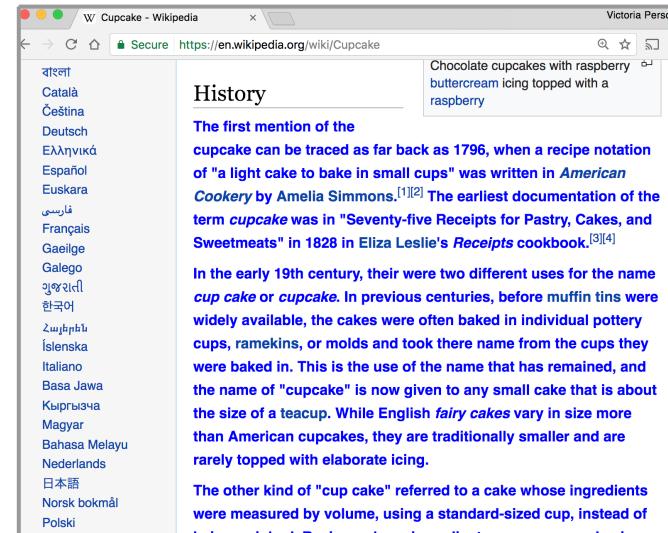
value: valore dello stile CSS.

Si salva in un file di testo
filename.css file.

CSS

```
p {  
    color: blue;  
    font-weight: bold;  
}
```

Questa regola dice che:
"Tutti gli elementi < p > nella pagina devono essere blue in bold face"



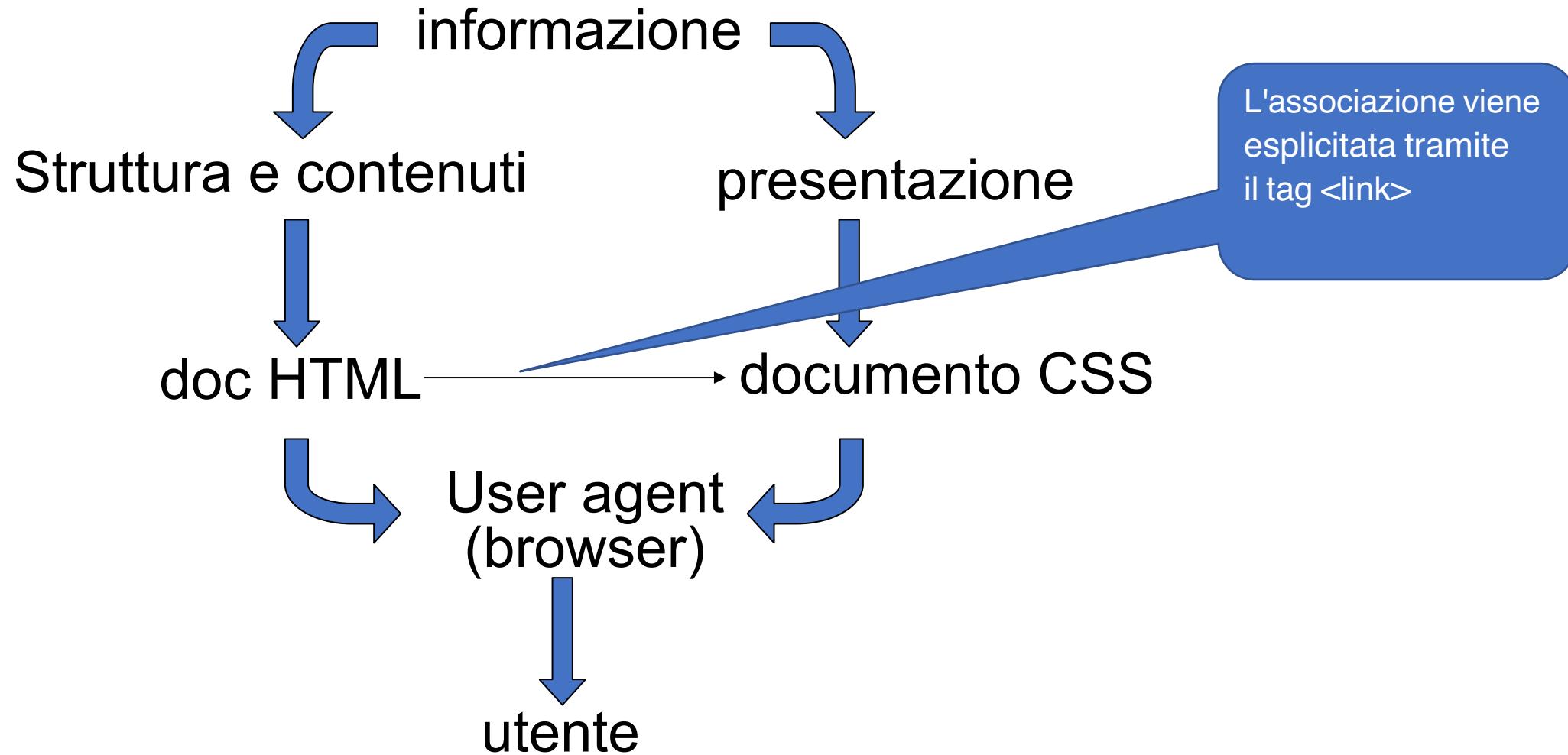
Associare un CSS alla pagina HTML

(banale copia e incolla)

```
<!DOCTYPE html>
<html>
  <head>
    <title>SIW</title>
    <link rel="stylesheet" href="filename.css" />
  </head>

  <body>
    ... contenuto della pagina...
  </body>
</html>
```

Filosofia HTML + CSS



Alcune proprietà CSS

Ci sono più di [500 CSS proprietà](#)! Vediamone alcune:

Font face (mdn)	font-family: Helvetica;
Font color (mdn)	color: gray;
Background color (mdn)	background-color: red;
Border (mdn)	border: 3px solid green;
Text alignment (mdn)	text-align: center;

Alcune proprietà CSS

- [Mozilla Developer Network](#) (MDN) è il miglior riferimento per gli elementi HTML e per le proprietà CSS
- La specifica W3C è molto difficile da leggere (è pensata per sviluppatori di browser, non per sviluppatori web)

Principali modalità per definire i colori CSS

140 nomi predefiniti ([lista](#))

color: black;

rgb() and **rgba()**

color: rgb(34, 12, 64);

color: rgba(0, 0, 0, 0.5);

Hex values

color: #00ff00;

color: #0f0;

color: #00ff0080;

- "a" sta per alpha channel: è un valore di trasparenza
- Generalmente sono preferibili le forme più descrittive:
 1. Nomi predefiniti
 2. rgb / rgba
 3. Hex

Aiutiamoci con CSS per rivedere
block vs inline display

Tipi di elementi HTML

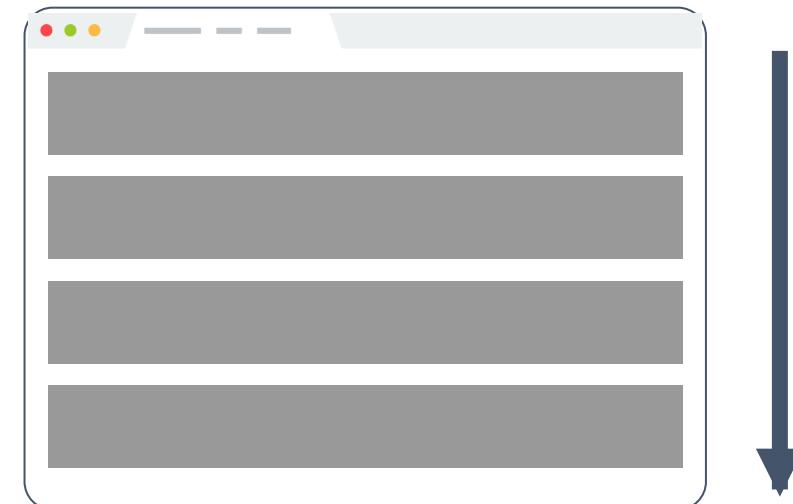
Le specifiche di HTML assegnano ciascun elemento HTML ad una di queste categorie:

- **block**: ha altezza e larghezza
Es. `<p>`, `<h1>`, ``, ``
- **inline**: non hanno altezza e larghezza
Es. `<a>`, ``, ``
- **inline block**: contenuto inline con altezza e larghezza
``
- **metadata**: informazioni sulla pagina, non visibile
`<title>`, `<meta>`

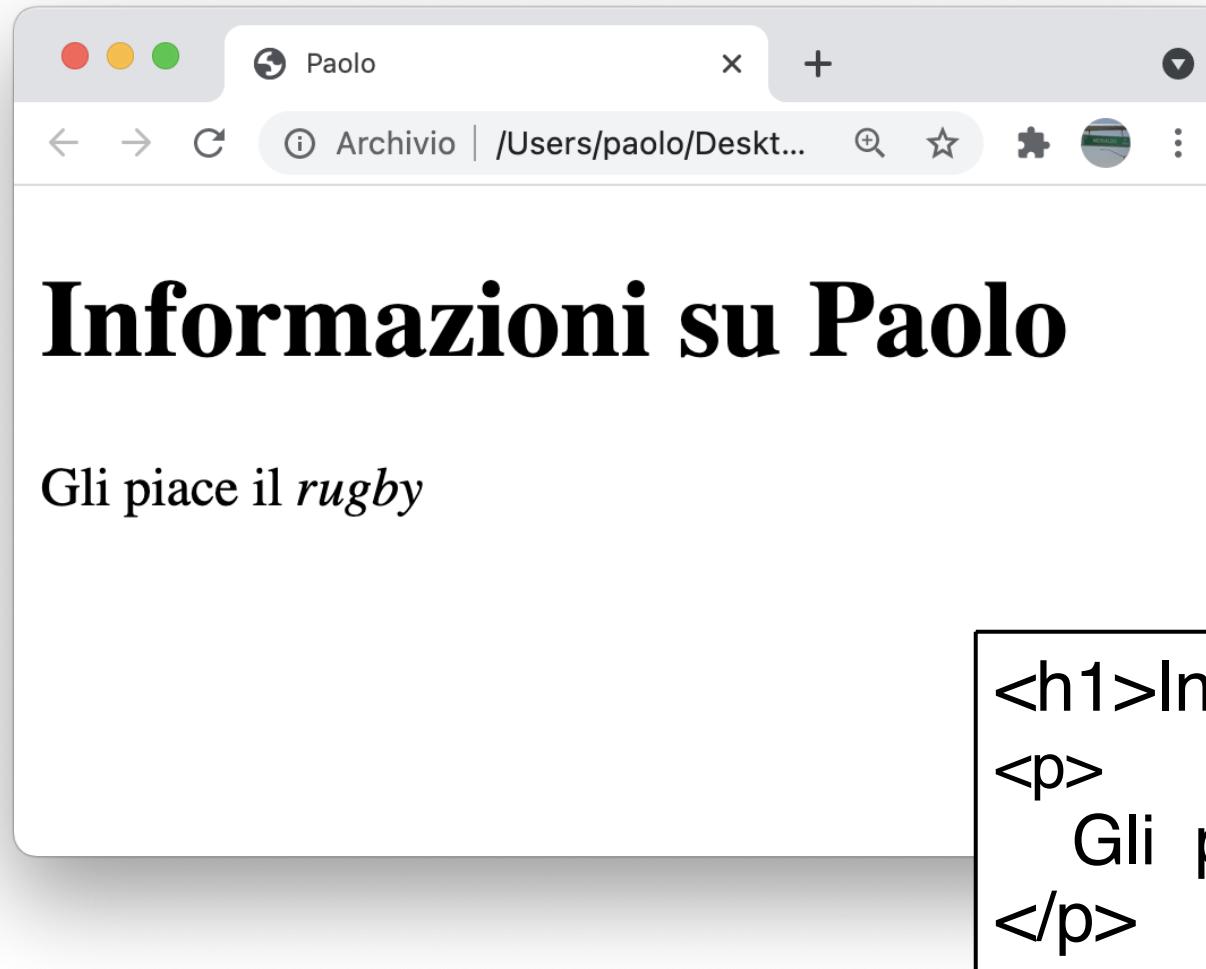
Elementi Block

Esempi: <p>, <h1>, ,

- Occupano l'intera larghezza della pagina (fluttuano dall'alto al basso)
- Hanno altezza (**height**) e larghezza (**width**) modificabili
- Possono avere elementi block o inline come figli



Esempio: Block



```
<h1>Informazioni su Paolo </h1>
<p>
    Gli piace il <em>rugby</em>
</p>
```

Aggiungiamo stile:



```
h1 {
  border: 5px solid red;
}
```

Q: come si vede nel
browser?

```
<h1>Informazioni su Paolo </h1>
<p>
  Gli piace il <em>rugby</em>
</p>
```



Informazioni su Paolo

Gli piace il *rugby*

Elementi Block

Si estendono su tutta la larghezza della pagina

Fluiscono dall'alto al basso

```
<h1>Informazioni su Paolo </h1>
<p>
  Gli piace il <em>rugby</em>
</p>
```

```
h1 {
  border: 5px solid red;
}
```

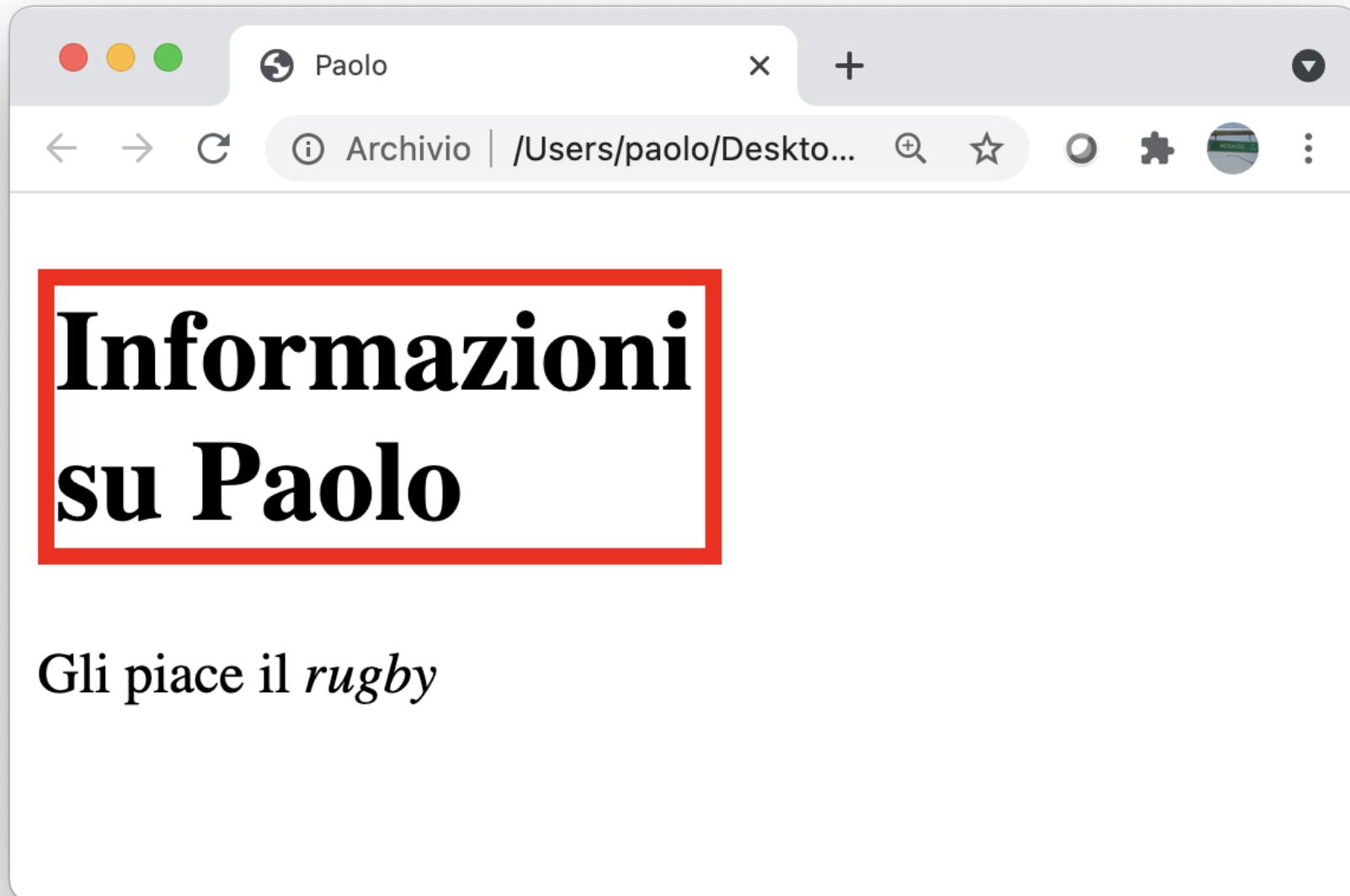


Q: Come cambia nel browser?

```
h1 {  
    border: 5px solid red;  
    width: 50%;  
}
```



```
<h1>Informazioni su Paolo </h1>  
<p>  
    Gli piace il <em>rugby</em>  
</p>
```



Informazioni su Paolo

Gli piace il *rugby*

Elementi Block

Si estendono su tutta la larghezza della pagina

Fluiscono dall'alto al basso

Hanno una larghezza che può essere modificata

```
<h1>Informazioni su Paolo </h1>
<p>
  Gli piace il <em>rugby</em>
</p>
```

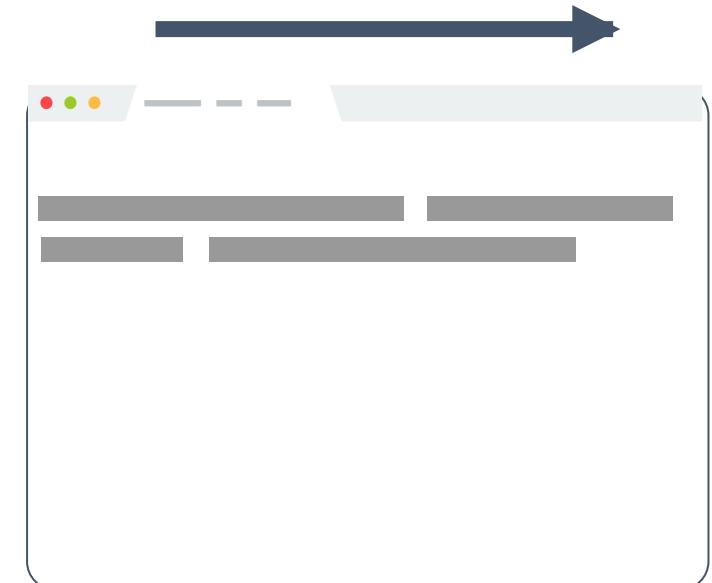
```
h1 {
  border: 5px solid red;
  width: 50%;
}
```



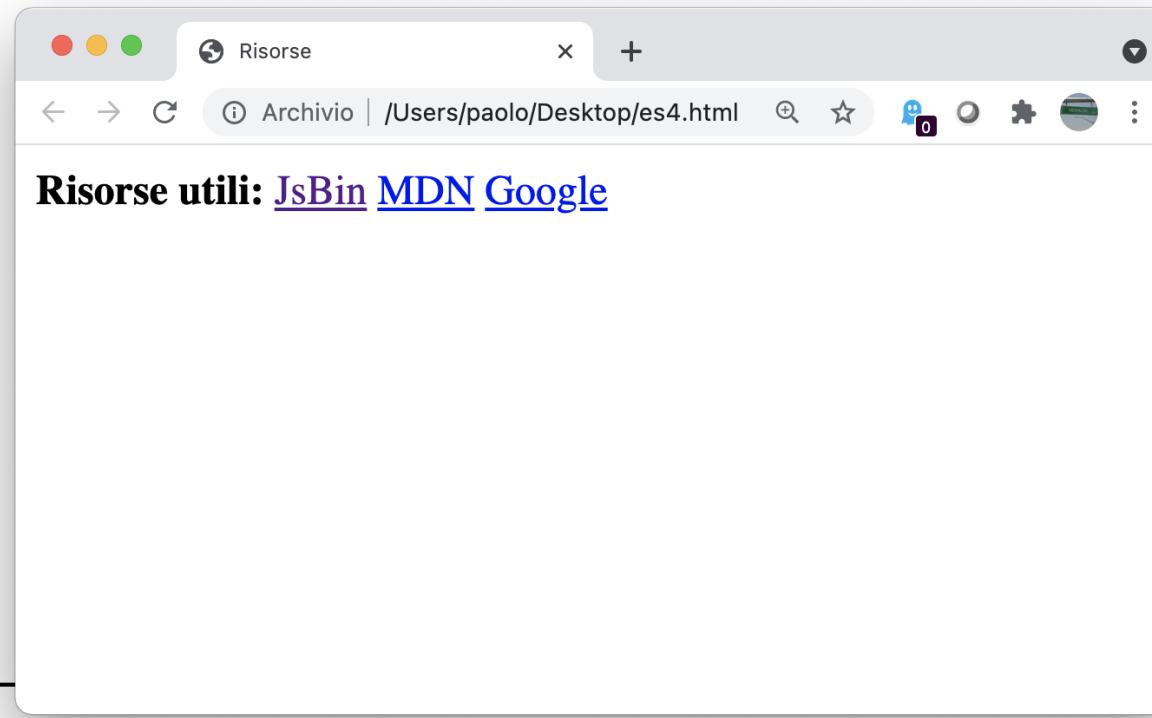
Elementi Inline

Esempi: <a>, ,

- Occupano la larghezza necessaria a visualizzare il contenuto
- Fluiscono da sinistra a destra
- **Non si possono** modificare **height e width**
- **Non si possono** avere elementi block come figli
- **Non è possibile modificarne il flusso**



Esempio: Inline



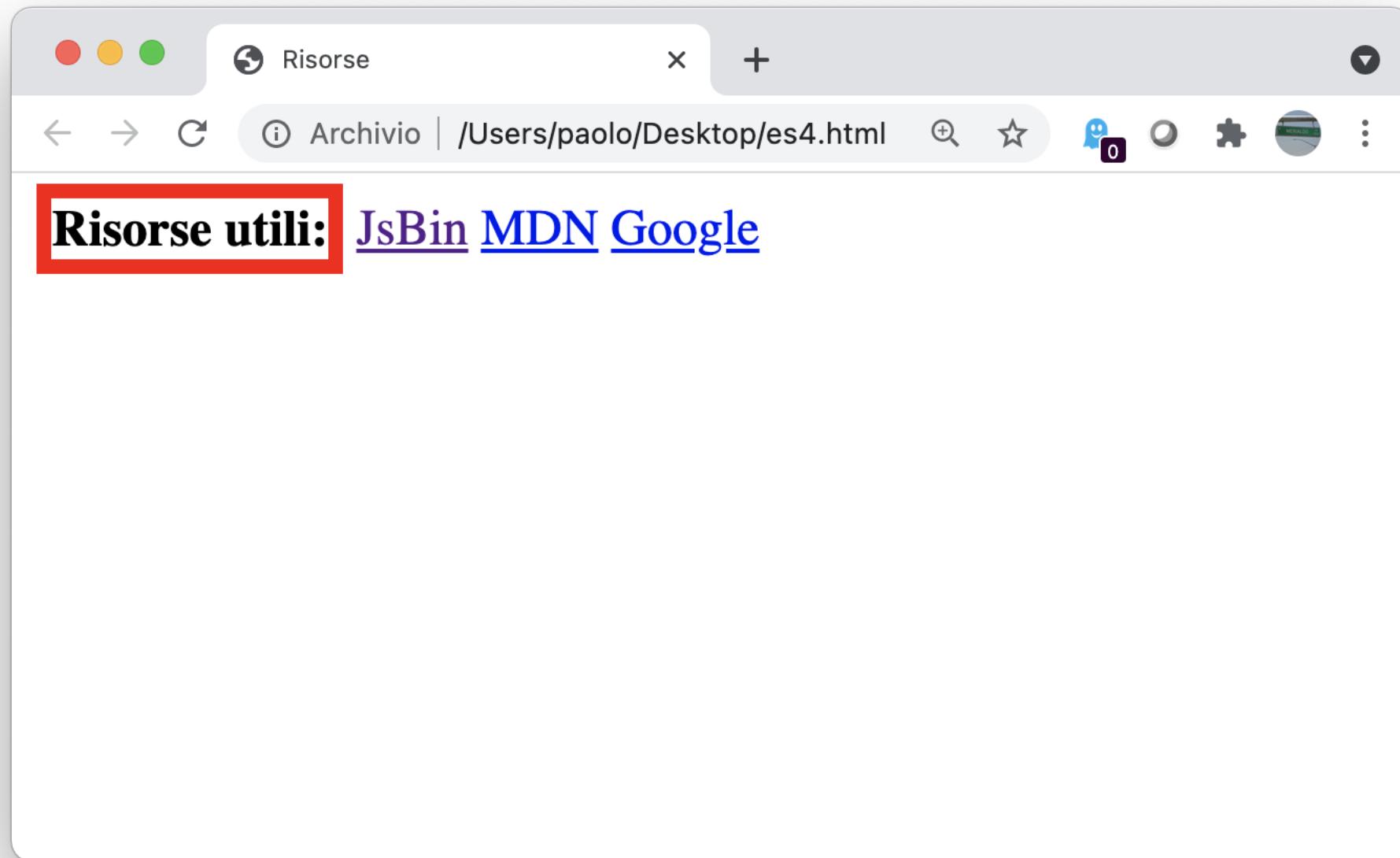
```
<body>
  <strong>Risorse utili:</strong>
  <a href="https://jsbin.com">JsBin</a>
  <a href="https://developer.mozilla.org/en-US/">MDN</a>
  <a href="http://google.com">Google</a>
</body>
```

Q: come appare nel browser?



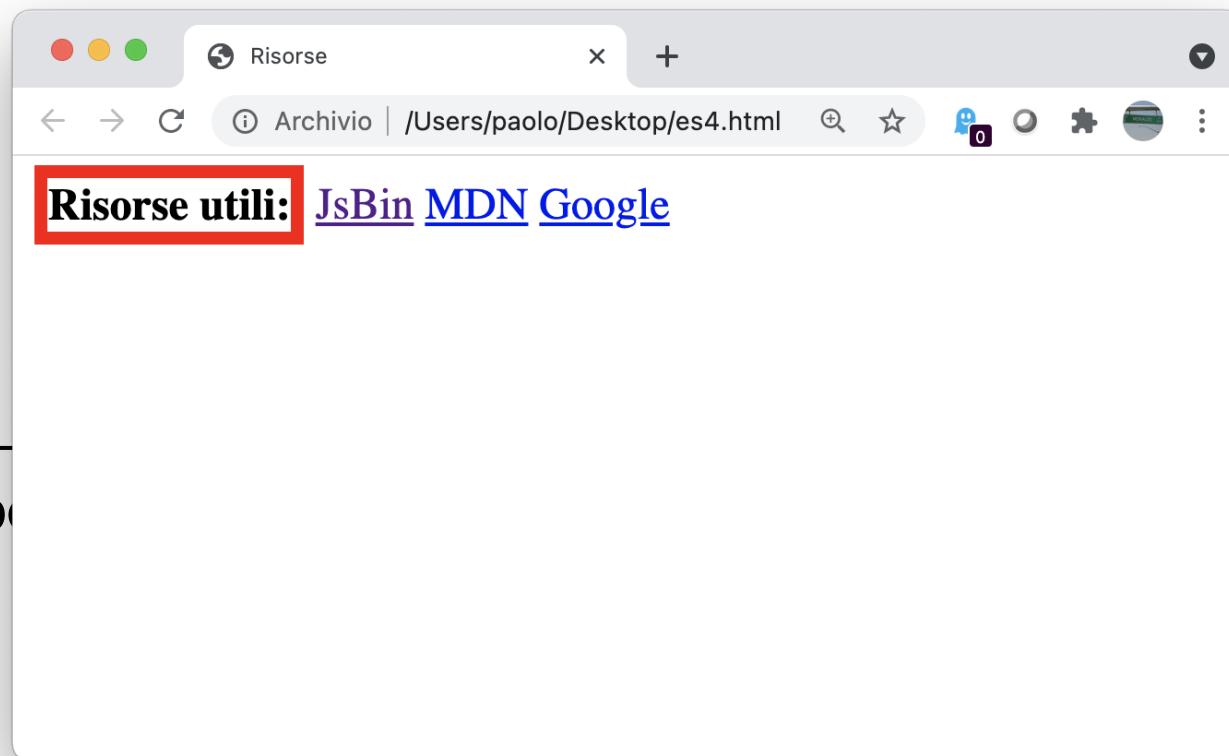
```
strong {  
  border: 5px solid red;  
  width: 1000px;  
}
```

```
<body>  
  <strong>Risorse utili:</strong>  
    <a href="https://jsbin.com">JsBin</a>  
    <a href="https://developer.mozilla.org/en-US/">MDN</a>  
    <a href="http://google.com">Google</a>  
</body>
```



Gli elementi Inline ignorano width

non può essere modificata la larghezza



```
strong {  
    border: 5px solid red;  
    width: 1000px;  
    /* Non funziona; strong è  
       inline! */  
}
```

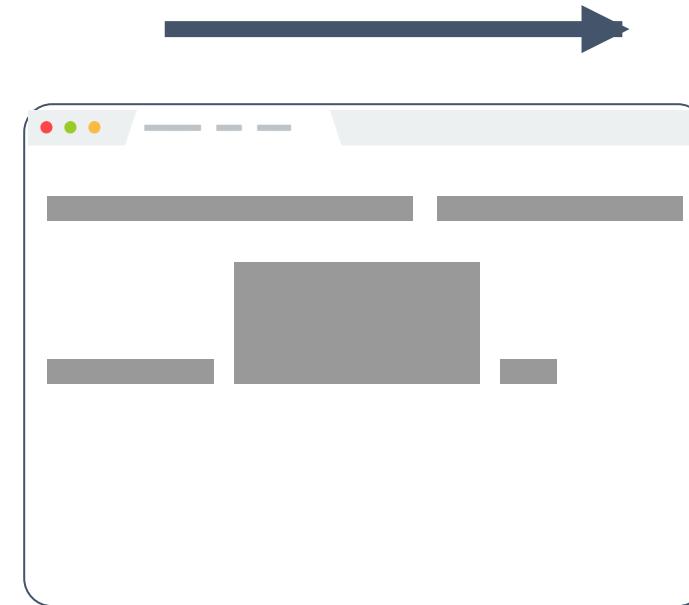
```
<b>  
    <a href="http://jsbin.com/...>JsBin</a>  
    <a href="http://mdn.mozilla.org/en-US/">MDN</a>  
    <a href="http://google.com">Google</a>  
</b>  
</body>
```

- **Non si possono** modificare height e width (vengono ignorate)

Elementi inline-block

Esempi: , qualunque elemento con
display: inline-block;

- La larghezza (width) corrisponde alla dimensione del contenuto
- Il flusso e' sulla linea
- **Si possono impostare** height e width
- **Può avere** un elemento block come figlio
- **Può essere posizionato** con opportune proprietà CSS (**float** e position)



Esempio: Inline-block

Q: Come appare nel browser?

```

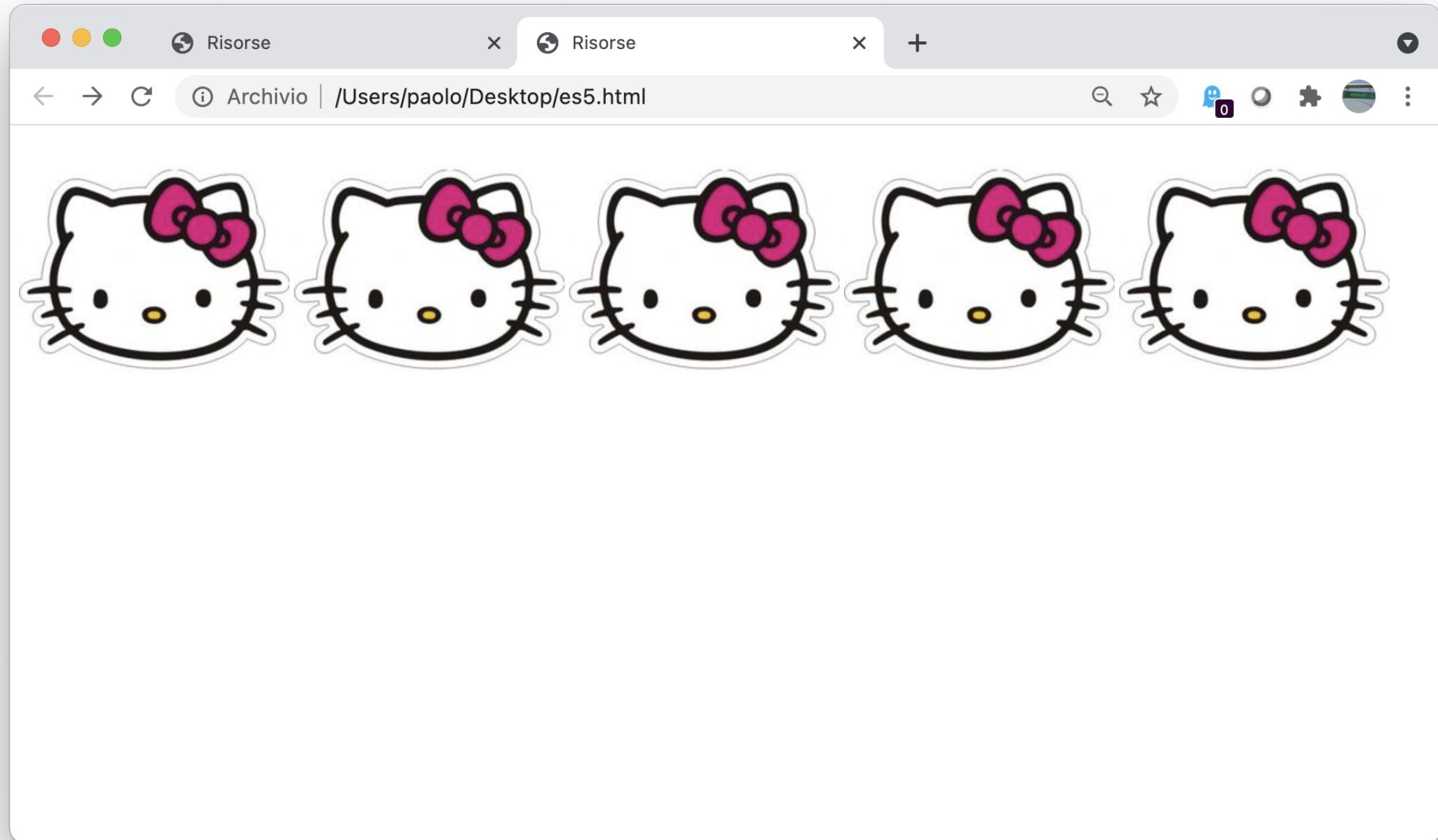




```

<http://i.imgur.com/WJToVGv.jpg> =





Esempio: Inline-block

```
img {  
    width: 50px;  
}
```

Q: Come appare nel browser?

```
  
  
  
  

```

<http://i.imgur.com/WJToVGv.jpg> =

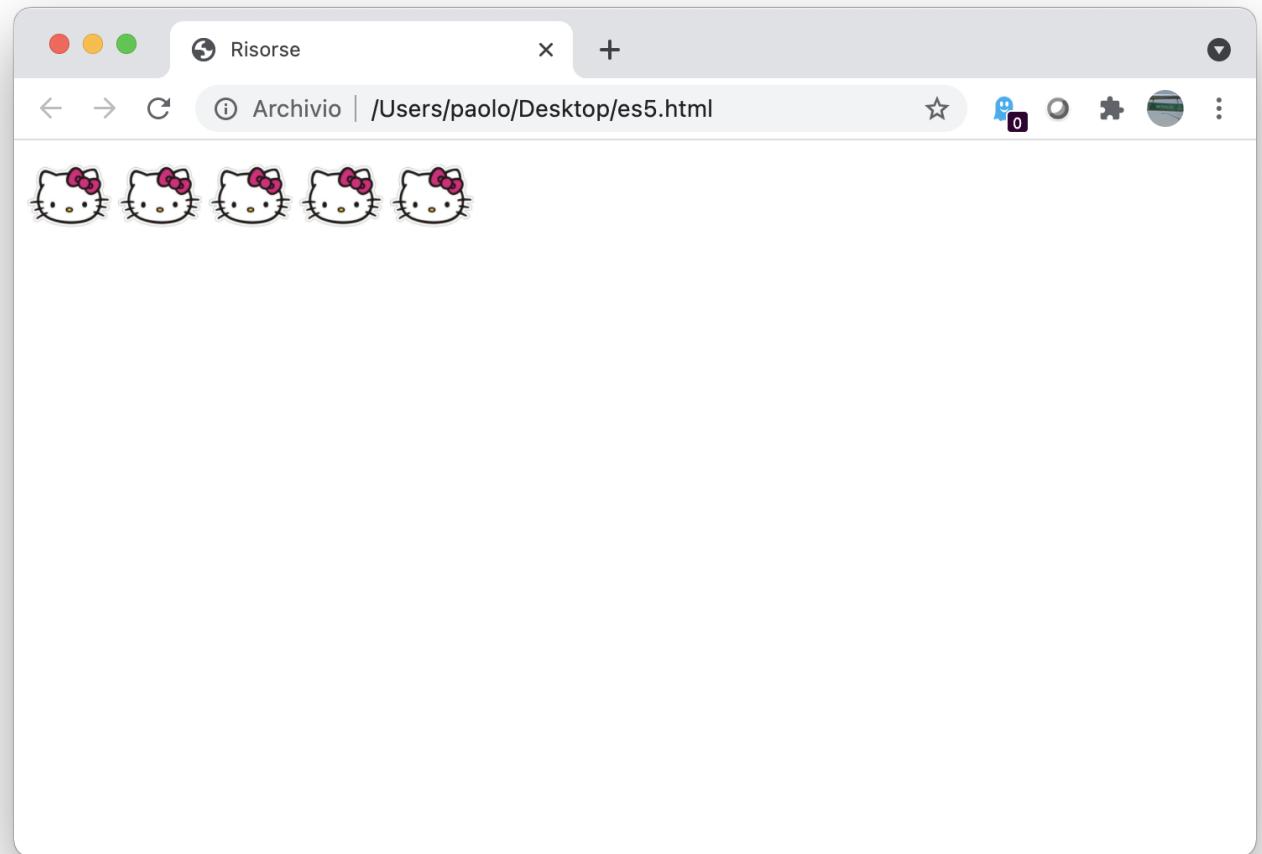


Inline-block

Hanno width e height; flows left to right

**Si possono impostare
height e width
inline-block scorre da
sinistra verso destra**

```
img {  
    width: 50px;  
}
```



Ricapitoliamo

block:

- scorre **top-to-bottom**;
- **è possibile impostare height e width**
- Esempi: <p>, <h1>, , , <table>

inline:

- scorre **left-to-right**;
- **non è possibile impostare height e width**
- Esempi: <a>, ,

inline-block:

- scorre **left-to-right**;
- **è possibile impostare height e width**
- Esempi:

Morale della storia (fino a qui):

Se il nostro CSS non funziona, controlliamo se stiamo applicando proprietà block-level ad elementi inline

Gli elementi neutri <div> e (molto usati)

<div>: elemento generico block

: elemento generico inline

Breve storia di HTML

Breve storia di HTML



**Tim Berners-
Lee**

- 1989: Tim Berners Lee crea il World Wide Web (WWW: pagine web e protocollo HTTP)
- 1994: viene creato il World Wide Web Consortium
 - "W3C": ha l'obiettivo di mantenere e sviluppare gli standards alla base del web
 - Prevede molti linguaggi:
 - HTML, CSS, DOM, XML, etc
- 1997: viene pubblicato "HTML4"
 - La prima versione stabile di HTML

"Degrading gracefully"

La specifica W3C di HTML indica alcuni [design principles](#). Uno di questi è il cosiddetto degrading gracefully



"Una scala mobile non si può mai rompere: al più diventa una scala"
(non si applica a Roma)

Questo è il motivo per cui i browsers fanno di tutto (best-effort) per renderizzare HTML e CSS non-standard ("invalidi").

Perchè non forzare "strict HTML"?

Abituati al rigore di un compilatore, troviamo strano che:

- I Browsers non solo non falliscono quando incontrano HTML/CSS non validi
- Ma addirittura presentano HTML/CSS apparentemente "corretto"

Q: Perchè i browser non respingono HTML/CSS mal scritto?

Perchè non forzare "strict HTML"?

Abituati al rigore di un compilatore, troviamo strano che:

- I Browsers non solo non falliscono quando incontrano HTML/CSS non validi
- Ma addirittura presentano HTML/CSS apparentemente "corretto"

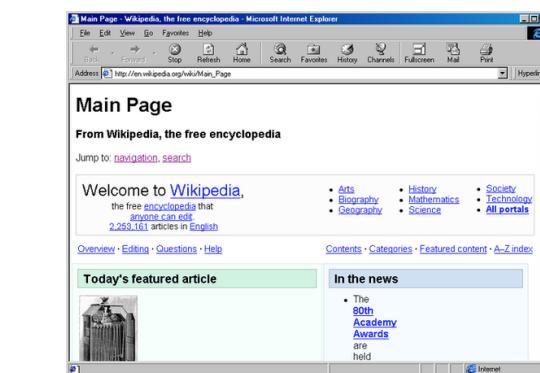
Q: Perchè i browser non respingono HTML/CSS mal scritto?

**R: c'è stato un tentativo per forzare la scrittura di codice
HTML/CSS valido...**

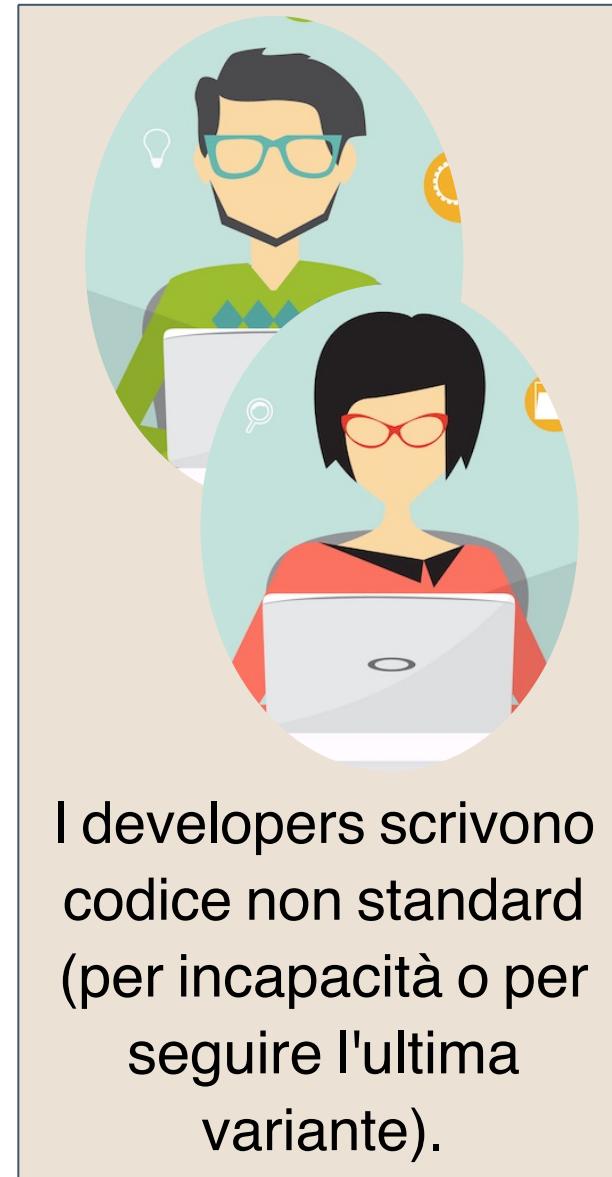
Il mondo nel 1997:



Gli standards
dicono una cosa



Ogni browser introduce
una nuova bellissima
variante



I developers scrivono
codice non standard
(per incapacità o per
seguire l'ultima
variante).

Il mondo nel 1997:

Nel 1997, la
confusione regna
sovra!

Gli standar-
dicono una cosa

una nuova "fichissima"
piccola variante

...opers scrivono
non standard
.incapacità o per
seguire l'ultima
variante).



Intorno al 2000:



W3C

Oops, "degrading
gracefully" è stato un
errore!

Browsers, smettete di
renderizzare pagine
non valide.

(Raccomandazione W3C: XHTML 1.1)

Intorno al 2000



W3C

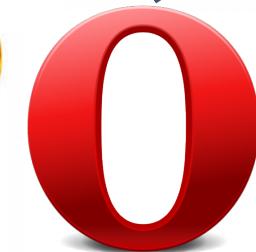
That would
break the
internet!

What?!?!

Sure
whatever



We can't
do that!



2004: WHATWG formed



Buttiamo via tutto e
ripartiamo da zero con
XHTML 1.1

(ha rotto circa 64 milioni di siti web)



WHATWG



Lavoriamo su HTML5
uno standard imperfetto
ma realistico

XHTML: idee di fondo

- XHTML = Riformulazione di HTML 4.0 in XML
 - formato e DTD XML (e non SGML):
 - i documenti XHTML corretti sono alberi validi rispetto ad un DTD XML fissato dal consorzio
 - il DTD descrive quali elementi e quali attributi possono comparire in una pagina XHTML
- Il mime type di un documento XHTML formalmente dovrebbe essere:
Content-Type: application/xhtml+xml

XHTML nella pratica

- Il mime type di un documento XHTML formalmente dovrebbe essere:
Content-Type: application/xhtml+xml
- Nella pratica, nonostante ci si sia iniziati ad uniformare alla sintassi di XHTML, si è continuato ad usare come mime-type:
Content-Type: text/html
- Questo per vari motivi. Il più importante legato alla gestione degli errori: un browser era tenuto ad adottare un approccio "draconiano" con un documento XML, mentre poteva continuare ad essere tollerante con un documento HTML
- Tradotto: nella pratica XHTML non è mai esistito!

Corriamo al 2021



- W3C ha rinunciato a XHTML 1.1 nel 2007
- W3C e WHATWG hanno rapporti amichevoli, ma sono entità separate

"HTML5" vs HTML

W3C mantiene [HTML5](#):

- Versione più stabile dell'HTML WHATWG
- Di solito copia quello che fa WHATWG dopo un po' di tempo

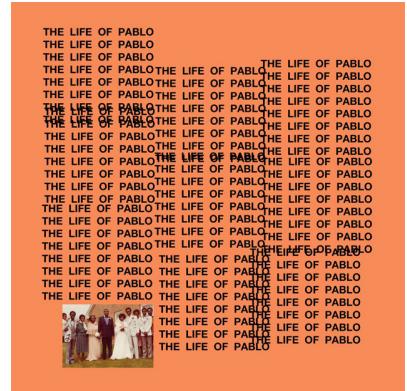


WHATWG mantiene [HTML: The Living Standard](#)

- Niente numeri, niente versioni
- Aggiornato frequentemente (oggi stesso!)
- La maggior parte dei browsers implementa WHATWG
- Quindi, non usiamo il termine "HTML5", ma semplicemente HTML



WHATWG



Cosa abbiamo bisogno di conoscere

Q: Chi mi dice quali elementi HTML posso usare?

- [MDN's list of HTML tags](#)

Q: Come faccio a sapere se un tag HTML (o una proprietà CSS, o una caratteristica JS) è implementata da tutti i browsers?

- consulta [caniuse.com](#)

Q: Perchè non posso usare HTML/CSS/JavaScript non-standard, anche se funziona in ogni browser?

Cosa abbiamo bisogno di conoscere

Q: Chi mi dice quali elementi HTML posso usare HTML?

- [MDN's list of HTML tags](#)

Q: Come faccio a sapere se un tag HTML (o una proprietà CSS, o una caratteristica JS) è implementata da tutti i browsers?

- consulta [caniuse.com](#)

Q: Perchè non posso usare HTML/CSS/JavaScript non-standard, anche se funziona in ogni browser?

- Perchè non è garantito che funzioni nel futuro
- Perchè non è garantito che funzioni su tutti gli "user agents" (non solo browsers)