UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

WEB TECHNOLOGIES — LECTURE 03

CSS: CASCADING STYLE SHEETS

Luigi Libero Lucio Starace, PhD
luigiliberolucio.starace@unina.it
https://luistar.github.io
https://www.docenti.unina.it/luigiliberolucio.starace

PREVIOUSLY, ON WEB TECHNOLOGIES

- We have learned how to write HTML documents
- HTML is concerned with **structure** and **semantics** of documents
- HTML is saying nothing at all on the **appearence** of documents

- An **** element specifies that its content should be emphasized
- It's not saying **how** the emphasizing part should be done
  - Emphasis might be conveyed using *italics*, different colors or backgrounds.

CSS: CASCADING STYLE SHEETS

- A **rule-based, declarative** language for specifying how documents should be presented to users.
- A **stylesheet** is a set of **Rules**, each defined as follows
- The `selector` specifies which HTML elems are affected by the rule
- Rules contain a set of **declarations**, in the form of **property-value pairs**, which specify the style to apply

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

CSS è un linguaggio dichiarativo basato su regole, per specificare come i documenti devono presentari all'utente.
Un foglio di stile è un insieme di regole, ognuna definita come segue.
Il selector specifica quali elementi HTML sono affetti dalla regola.
Le regole contengono un insieme di dichiarazioni, nella forma di coppie proprietà-valore, che specificano lo stile d'applicare.

CSS: FIRST EXAMPLE

```
<h1>Hello CSS</h1>
<p>
  Our <em>first</em> page
  with <em>style</em>!
</p>
```

```
h1 {
  color: red;
  font-size: 50px;
}

em {
  color: blue;
}
```

Ma le prime pagine web che sviluppiamo hanno
degli stili predefiniti!
- gli heading sono più grandi e mostrati in bold.
- <p> inizia una nuova linea, sono mostrati
in corsivo, in bold, <a> sono sottolineati
e blu, <ul> ha i bulloni e così via...

Questo perché i browser hanno i propri stili. Sono spesso indicati come stili user-agent. Questi default sono grosso modo lo stesso per
browser differenti, ma esistono alcune

DEFAULT USER AGENT STYLES

- But the first web pages we developed have some styling!
  - Headings are bigger and shown with a bold face…
  - **<p>** starts on new lines, **** are displayed in italic, **** in bold, **<a>** are underlined and blue, **<ul>** have bullets, and so on…
- That's because browsers apply their own, basic styles to every page!
- They are often referred to as **user agent styles**
- These defaults are roughly the same across different browsers, but some **differences** exist (and we'll get back to that!)

INCLUDING STYLESHEETS IN WEB PAGES

Styling can be included in HTML documents in different ways

- Using `<link>` elements in the `<head>` of the document
  - The `rel="stylesheet"` attribute specifies the relation between the current document and the linked document
  - The `href="style.css"` attrib. specifies the URL of the stylesheet to load
  - Same mechanism as `<img>`: browser will make an additional HTTP request to fetch the stylesheet before rendering the page

```
<head>
  <meta charset="UTF-8">
  <title>CSS</title>
  <link rel="stylesheet" href="style.css">
</head>
```


Stilizzare può essere incluso nei documenti HTML in modi diversi. Usando element <link> nella testa del documento:
- il rel="stylesheet" è un attributo che specifica la relazione tra l'attuale documento e quello collegato.
- href="style.css" è un attributo che specifica il link URL del foglio di stile da caricare.
- lo stesso meccanismo funziona per <img>, il browser farà una richiesta HTTP addizionale per recuperare il foglio di stile prima di caricare la pagina

Le regole CSS possono essere definite in elementi <style> nella <head>.
È generalmente preferibile usare fogli di stile
esterni e <link>.
Quali possono essere le ragioni?

INCLUDING STYLESHEETS IN WEB PAGES

- CSS rules can also be defined in `<style>` elements in the `<head>`
- It is generally preferable to use external stylesheets and `<link>`
  - Can you think of some reasons why?

```
<head>
  <meta charset="UTF-8">
  <title>CSS</title>
  <style>
    h1 {
      color: red;
      font-size: 50px;
    }
  </style>
</head>
```


Lo stile può essere inserito globalmente a un qualsiasi elemento HTML e si usa l'attributo style.
Il valore dell'attributo style è una sequenza di dichiarazioni separate da ;.
Queste dichiarazioni di stile si applicano solo allo

STYLING HTML ELEMENTS

- HTML elements can also be styled inline, using the `style` attribute
- The value of the style attribute is a sequence of declarations, separated by «**;**»
- These styling declarations apply **only to the specific element** bearing the attribute

```
<em style="color: fuchsia; font-weight: bold;">inline style</em>
```

CSS: INLINE STYLES

```
<h1>Hello CSS</h1>
<p>
  Our <em style="color:fuchsia;font-weight: bold;">first</em>
  page with <em>style</em>!
</p>
```

```
h1 {
  color: red;
  font-size: 50px;
}

em {
  color: blue;
}
```

SELECTORS

I selettori sono una parte chiave di CSS. Sono specifici agli elementi a cui si applicano le regole CSS. I selettori CSS non sono usati solo per lo stile! -Quando si usa .js per scrivere pagine web dinamiche si possono usare per selezionare elementi con cui interagire - quando si fa il testing web automatizzato, possono essere usati per determinare quali elementi il test deve interagire -quando facendo scraping/crawling si possono usare per selezionare gli elementi che

SELECTORS

- Selectors are a **key** part of CSS
- They specify to which elements a CSS rule applies
- CSS selectors are not only used for styling!
  - When using JavaScript to make web pages dynamic, they can be used to select which elements to interact with
  - When doing automated web testing, they can be used to determine which elements the test needs to interact with
  - When doing scraping/crawling, they can be used to select the elements that contain the information we want to extract

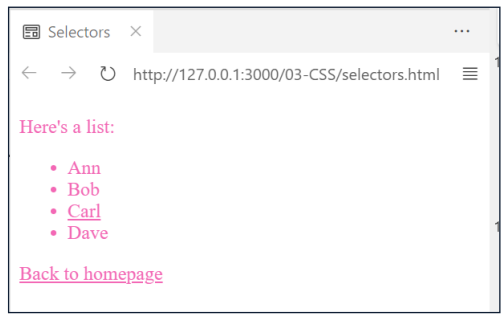
SIMPLE CSS SELECTORS

There exist **five** kinds of simple selectors:

- **Universal** selector (a.k.a **wildcard**). Matches any element.

```
* {
  color: hotpink;
}
```

```
<p>Here's a list:</p>
<ul>
  <li>Ann</li>
  <li>Bob</li>
  <li><a href="/car/">Carl</a></li>
  <li>Dave</li>
</ul>
<a href="/">Back to homepage</a>
```

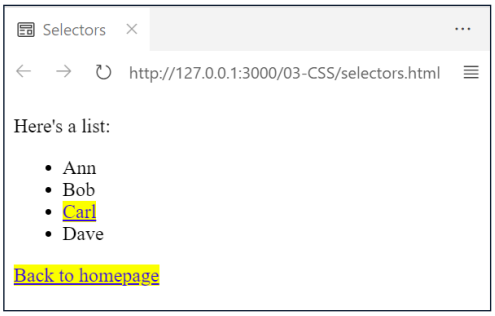


SIMPLE CSS SELECTORS

- **Type** selector. Matches all element of a given type (i.e., tag name)
- The selector is simply the name of the tag to match

```
a {
  background: yellow;
}
```

```
<p>Here's a list:</p>
<ul>
  <li>Ann</li>
  <li>Bob</li>
  <li><a href="/car/">Carl</a></li>
  <li>Dave</li>
</ul>
<a href="/">Back to homepage</a>
```

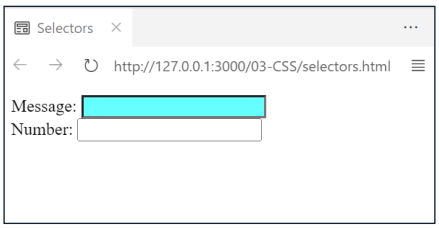


SIMPLE CSS SELECTORS

- **Id** selector. Matches the element with the given **`id`** attribute.
- Selector has the form ***`#ElementId`***

```
#msg {
  background: cyan;
}
```

```
<form>
  <label for="msg">Message: </label>
  <input id="msg" type="text" name="msg"><br>
  <label for="num">Number: </label>
  <input id="num" type="number" name="num">
</form>
```

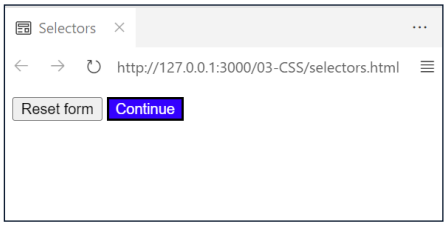


SIMPLE CSS SELECTORS

- **Class** selector. Matches the element with the given **`class`** attribute.
- Selector has the form ***`.classname`***

```
.primary {
  background: blue;
  color: white;
}
```

```
<button>Reset form</button>
<button class="primary btn">Continue</button>
```



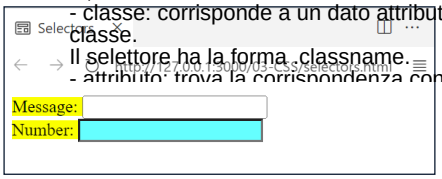
SIMPLE CSS SELECTORS

- **Attribute** selector. Matches the element with a certain attribute.
- Selector has the form **`[attribute]`** or **`[attribute=value]`**

Esistono 5 tipi di selettori semplici:
- universali (wildcar), recuperano qualsiasi elemento
- tipo: recuperano tutte le corrispondenze di un dato tipo
(come il nome di tag). Il selettore è semplicemente
il nome di cui si deve fare il match col tag
- id: recuperano l'elemento di un dato attributo id, ha la forma #ElementId ed è unico.
- classe: corrisponde a un dato attributo di classe.
Il selettore ha la forma .classname.
- attributo: trova la corrispondenza con

```
[for]{ /*all elems with a for attribute*/
  background: yellow;
}
[type='number']{ /*all elems with type=number*/
  background: cyan;
}
```

```
<form>
  <label for="msg">Message: </label>
  <input id="msg" type="text" name="msg"><br>
  <label for="num">Number: </label>
  <input id="num" type="number" name="num">
</form>
```



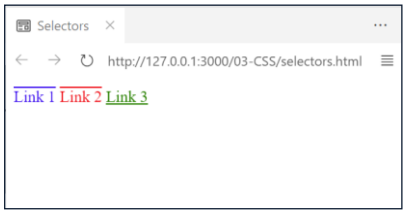
SIMPLE CSS SELECTORS

Gli operatori addizionali permettono il matching parziale con i valori degli attributi

- Additional operators (*=, ^=, $=) allow **partial matching** with attribute values

```
[href*='programming']{ /*contains 'programming'*/
  text-decoration: overline;
}
[href^='https']{ /*start with 'https'*/
  color: red;
}
[href$='.it/']{ /*ends with '.it/'*/
  color: green;
}
```

```
<a href="http://bookofprogramming.com/">Link 1</a>
<a href="https://programming.net/">Link 2</a>
<a href="http://webtechnologies.it/">Link 3</a>A
```



COMPLEX CSS SELECTORS: COMPOUNDS

È possibile combinare selettori semplici per
avere un controllo più granulare nella pagina,
fatto tramite selettori compositi.
Basicamente si seleziona l'intersezione dei

- It is possible to combine selectors to get fine-grained control
- This is done by concatenating selectors
- Basically select the **intersection** of the involved selectors

```
a[target='_blank'] {
  color: red;
}
a.my-class{
  color: green;
}
a[href*='programming'].my-class {
  background: yellow;
}
```

```
<a href="http://bookofprogramming.com/" target="_blank">Link 1</a>
<a class="my-class" href="https://programming.net/">Link 2</a>
<em class="my-class">Hello</em>
```

COMPLEX CSS SELECTORS: COMBINATORS

- Combinators are used to select elements based on their position in the document (remeber that HTML documents can be seen as **trees**!)
- Syntax is: `selector1 combinator selector2`

- Four different combinators exist in CSS:
  - Descendant selector (space)
  - Child selector (**>**)
  - Adjacent sibling selector (**+**)
  - General sibling selector (**~**)

I combinatori sono usati per selezionare elementi basati sulla loro posizione nel documento (ricorda che i documenti HTML possono essere visti come alberi!).
Sintassi:

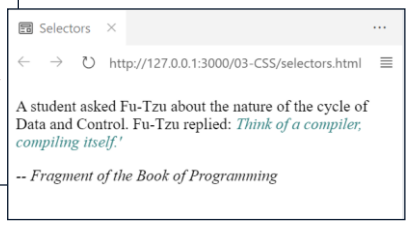
Esistono 4 combinatori diversi in CSS:
- selettori discendenti (spazio)
- selettori di figli diretti, quelli che si trovano direttamente "sotto" (>)
- selettore di fratello adiacente (+)
- selettore del fratello successivo (tilde)

COMBINATORS: DESCENDANT SELECTOR

- Syntax: `selectorA selectorB`
- Semantics: match all elements that match `selectorB` and are a contained within (i.e., are a descendant of) an element matching `selectorA`

```
section em {
  color: teal;
}
```

```
<section>
  <p>
    A student asked Fu-Tzu about the nature of
    the cycle of Data and Control. Fu-Tzu replied:
    <em>Think of a compiler, compiling itself.'</em>
  </p>
</section>
<em>-- Fragment of the Book of Programming</em>
```



Semantica: ottiene tutti gli elementi che corrispondono al selectorB e sono contenuti dentro all'elemento corrispondente a selectorA (ad esempio, sono discendenti di). Nell'esempio si vede come

Semantica: voglio tutti gli elementi che hanno un
match con un selectorB e sono direttamente contenuti (ad esempio, sono un figlio diretto di)
un elemento matchante in selectorA.
La regola dell'esempio si applica solo a

COMBINATORS: CHILD SELECTOR

- Syntax: `selectorA > selectorB`
- Semantics: match all elements that match `selectorB` and are a **directly** contained within (i.e., are a direct child of) an element matching `selectorA`

```
main > em {
  color: teal;
  font-variant: small-caps;
}
```

```
<main>
  CSS <em>selectors</em>:
  <p>We <em>like</em> 'em.</p>
</main>
```

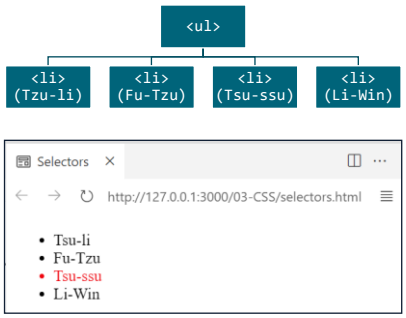
Semantica: voglio tutti gli elementi che hanno match
con un selettoreB e sono tutti figli successivi e adiacenti a un elemento matchante nel selettoreA.
Nell'esempio viene colorato di rosso solo l'elemento
Tsu-ssu, il 3o elemento.

COMBINATORS: ADJACENT SIBLINGS

- Syntax: `selectorA + selectorB`
- Semantics: match all elements that match `selectorB` and are a **next adjacent siblings** of an element matching `selectorA`

```
.master + li {
  color:red;
}
```

```
<ul>
  <li>Tsu-li</li>
  <li class="master">Fu-Tzu</li>
  <li>Tsu-ssu</li>
  <li class="disciple">Li-Win</li>
</ul>
```



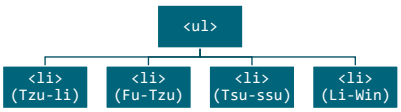
Semantica: voglio tutti gli elementi che soddisfano il selettoreB e che sia successivi (non direttamente)
di un elemento matchante selettoreA.
Nell'esempio voglio tutti gli elementi che hanno classe
disciple e che siano successivi a un elemento di

COMBINATORS: GENERAL SIBLINGS

- Syntax: `selectorA ~ selectorB`
- Semantics: match all elements that match `selectorB` and are a **subsequent siblings** of an element matching `selectorA`

```
.master ~ li.disciple {
  color:red;
}
```

```
<ul>
  <li>Tsu-li</li>
  <li class="master">Fu-Tzu</li>
  <li>Tsu-ssu</li>
  <li class="disciple">Li-Win</li>
</ul>
```



CSS SELECTORS: PSEUDO-CLASSES

Gli elementi HTML possono essere in stati differenti,
ad esempio per le interazioni utente o per le relazioni con altri elementi.
I selettori CSS pseudoclassi iniziano con :, e permettono di stilizzare gli elementi in base al loro stato:
- stati interattivi (risultati dall'interazione con l'utente)
- stati storici (usati per ricordare quali link sono stati visualizzati)
- stati di forma (specifica di interazioni con forme)
- stati derivanti dalle relazioni con altri elementi

HTML elements can be in different **states**, for example because of **user interactions** or because of their relation with other elements.

Pseudo-classes selectors start with « **:** », and allow to style elements based on their **state:**

- **Interactive states** (resulting from user interaction)
- **Historic states** (used to «remember» which links were visited)
- **Form states** (specific of interaction with forms)
- States deriving from **relations with other elements**

- :hover seleziona gli elementi su cui un dispositivo
di puntamento è piazzato (mouse).
- :active dà corrispondenza allo stato con cui un elemento sta attivamente interagendo (come la pressione di un pulsante)
- :focus dà corrispondenza allo stato in cui un elemento (come un link o un campo di inserimento)
su cui si deve focalizzare (come l'appena

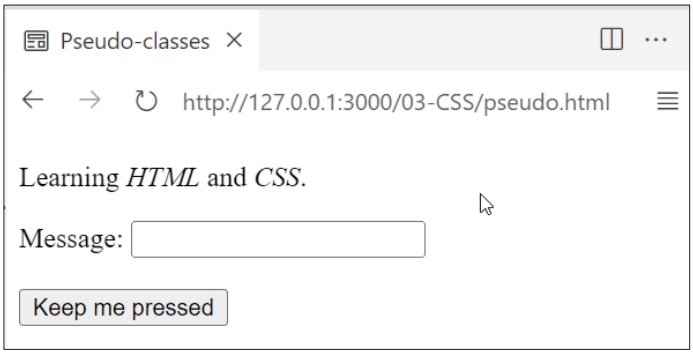
PSEUDO–CLASSES: INTERACTIVE STATES

- **:hover** selects the elements on which a pointing device (i.e.: mouse) is placed over
- :**active** matches the state in which an element is actively being interacted with (e.g.: button is being pressed)
- **:focus** matches the state in which an element (e.g.: a link or an input field) has focus (i.e.: is currently selected) in the web page

PSEUDO–CLASSES: INTERACTIVE STATES

```
<p>Learning <em>HTML</em> and <em>CSS</em>.</p>
Message: <input type="text"><br><br>
<button>Keep me pressed</button>
```

```
em:hover {
  background: yellow;
}
input[type='text']:focus{
  background: cyan;
}
button:active{
  background: blue;
  color: white;
}
```



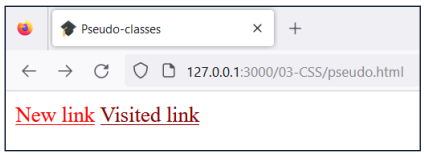
- :link seleziona i link che non sono stati visitati
- :visitati seleziona i link che sono appena stati

PSEUDO–CLASSES: HISTORIC STATES

- **:link** selects links that have not been visited yet
- **:visited** selects links that have already been visited

```
:link{
  color: red;
}
:visited{
  color: darkred;
}
```

```
<a href="./js/">New link</a>
<a href="./css/">Visited link</a>
```

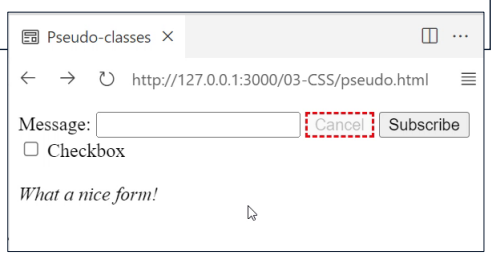


Historic states are **not supported** in the *Live Preview* browser within Visual Studio Code! Open the page in Firefox to check them out.

PSEUDO–CLASSES: FORM STATES

```
<div>
  <label for="mail">Message:</label><input id="mail" type="email">
  <button disabled>Cancel</button><button>Subscribe</button>
</div>
<input type="checkbox"> Checkbox<br><br>
<em>What a nice form!</em>
```

```
:disabled {
  border: 2px dashed red;
}
:invalid {
  color: red;
}
:checked ~ em {
  color: deeppink; font-weight: bold;
}
```



Technically, there can be email address without a dot. For example, user@localhost or user@com are valid addresses!

Collegate alla posizione in cui si trovano gli elementi.
- :first-child e :last-child selezione il primo/ultimo figlio tra un insieme di fratelli.
- :only-child si usa per selezionare gli elementi senza fratelli.
- :first/last-of-type si possono usare per selezionare
elementi che sono il primo/ultimo figlio di un insieme di fratelli, considerando solo elementi dello
stesso tipo.
- :nth-child/type(n) si può usare per selezionare elementi che sono all'ennesima posizione tra gli

PSEUDO–CLASSES: POSITION RELATIONS

- **:first-child** and **:last-child** select the first (last) child among a set of siblings.
- **:only-child** can be used to select elements that have no siblings.
- **:first-of-type** and **:last-of-type** can be used to select elements that are the first (last) child among a set of sibling, considering only elements of the same type.
- **:nth-child(n)** and **:nth-of-type(n)** can be used to select elements that are in the n-th position among their siblings.
  - **Indexing in CSS starts at 1!**

PSEUDO–CLASSES: POSITION RELATIONS

```
<p>
  <em>Ann</em> <strong>Bob</strong> <em>Carl</em>
</p>
<p>
  <strong>Ann</strong> <em>Bob</em> <strong>Carl</strong>
</p>
```

Ann **Bob** Carl
Ann Bob **Car**

PSEUDO–CLASSES: POSITION RELATIONS

```
<p>
  <em>Ann</em> <strong>Bob</strong> <em>Carl</em>
</p>
<p>
  <strong>Ann</strong> <em>Bob</em> <strong>Carl</strong>
</p>
```

Ann **Bob** Carl
Ann Bob **Car**

```
em:last-child {
  color: red;
}
```

Ann **Bob** Carl
Ann Bob **Car**

PSEUDO–CLASSES: POSITION RELATIONS

```
<p>
  <em>Ann</em> <strong>Bob</strong> <em>Carl</em>
</p>
<p>
  <strong>Ann</strong> <em>Bob</em> <strong>Carl</strong>
</p>
```

Ann **Bob** Carl
Ann Bob **Car**

```
em:last-of-type {
  color: red;
}
```

Ann **Bob** Carl
Ann Bob **Car**

Sto cercando tutti gli elementi di tipo em che sono
anche gli ultimi elementi di quel tipo, quindi
si elezionano solo il 1o Carl e il 2o Bob

Si seleziona solo Ann, perché è il primo figlio
em ad apparire nel paragrafo

PSEUDO–CLASSES: POSITION RELATIONS

```
<p>
  <em>Ann</em> <strong>Bob</strong> <em>Carl</em>
</p>
<p>
  <strong>Ann</strong> <em>Bob</em> <strong>Carl</strong>
</p>
```

Ann **Bob** Carl
Ann Bob **Car**

```
em:first-child {
  color: red;
}
```

Ann **Bob** Carl
Ann Bob **Car**

I primi del tipo em nei 2 paragrafi sono Ann e Bob

PSEUDO–CLASSES: POSITION RELATIONS

```
<p>
  <em>Ann</em> <strong>Bob</strong> <em>Carl</em>
</p>
<p>
  <strong>Ann</strong> <em>Bob</em> <strong>Carl</strong>
</p>
```

Ann **Bob** Carl
Ann Bob **Car**

```
em:first-of-type {
  color: red;
}
```

Ann **Bob** Carl
Ann Bob **Car**

PSEUDO–CLASSES: POSITION RELATIONS

```
<p>
  <em>Ann</em> <strong>Bob</strong> <em>Carl</em>
</p>
<p>
  <strong>Ann</strong> <em>Bob</em> <strong>Carl</strong>
</p>
```

Ann **Bob** Carl
Ann Bob **Car**

```
em:nth-child(2) {
  color: red;
}
```

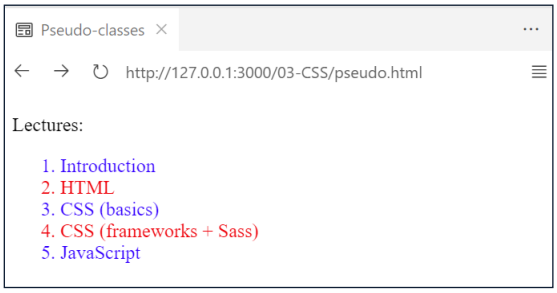
Ann **Bob** Carl
Ann Bob **Car**

Si possono usare le pseudoclassi anche usando espressioni matematiche che permettono di rendere
omogena la visualizzazione dello stile,
ad esempio si può usare even e odd per costruirsi

PSEUDO–CLASSES: POSITION RELATIONS

```
<p>Lectures:</p>
<ol>
  <li>Introduction</li>
  <li>HTML</li>
  <li>CSS (basics)</li>
  <li>CSS (frameworks + Sass)</li>
  <li>JavaScript</li>
</ol>
```

```
li:nth-child(even){
  color: red;
}
li:nth-child(odd){
  color: blue;
}
```



Gli pseudoelementi possono essere usati per indicare parti specifiche del contenuto di un dato elemento HTML, senza aggiungere ulteriore markup HTML.
La sintassi di uno pseudoelemento è ... .
selector è un selettore CSS per l'elemento target.
pseudo-element è uno degli pseudoelementi supportati dai selettori
- ::first-letter colpisce la prima lettera del contenuto all'elemento al livello del blocco.
- ::first-line colpisce la prima linea al livello di blocco del livello.
- ::selection colpisce il contenuto che è attualmente selezionato dall'utente.
- ::before crea un elemento che è il primo figlio dell'elemento selezionato.
- ::after crea un elemento che è l'ultimo figlio dall'elemento selezionato.

PSEUDO—ELEMENTS

Pseudo-elements can be used to target specific parts of the content of a given HTML element, without adding extra HTML markup

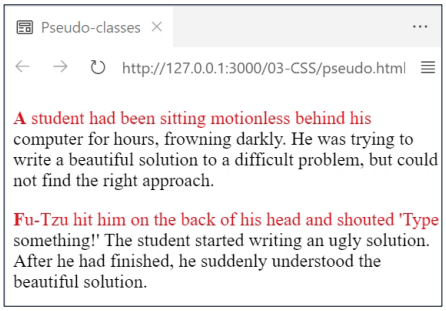
The sintax of pseudo-element selectors is **selector::*pseudo-element***

- **selector** is a CSS selector for the target element
- ***pseudo-element*** is one of the supported pseudo-element selectors:
  - **::first-letter:** targets the first letter of the content of a **block-level** element
  - **::first-line:** targets the first line of the content of a **block-level** element
  - **::selection:** targets the content that is currently selected by the user
  - **::before:** creates an element that is the **first child** of the selected element
  - **::after:** creates an element that is the **last child** of the selected element

PSEUDO—ELEMENTS: EXAMPLES

```
<p>A student had been sitting motionless behind  his computer for hours,
frowning darkly. He was trying to write a beautiful solution to a difficult
problem, but could not find the right approach.</p>
<p>Fu-Tzu hit him on the back of his head and shouted 'Type something!' The
student started writing an ugly solution. After he had finished, he suddenly
understood the beautiful solution.</p>
```

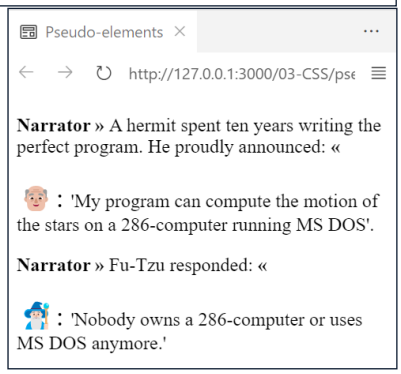
```
p::first-letter {
  font-weight:bold;
}
p::first-line{
  color: red;
}
p:last-child::selection {
  background: red;
  color: white;
}
```



PSEUDO-ELEMENTS: EXAMPLES

```
<p class="narrator">A hermit spent ten years writing the perfect program. He
proudly announced: </p>
<p class="hermit">'My program can compute the motion of the stars on a 286-
computer running MS DOS'.</p>
<p class="narrator">Fu-Tzu responded:</p>
<p class="fu-tzu">'Nobody owns a 286-computer or uses MS DOS anymore.'</p>
```

```
.narrator::before {
  content: "Narrator » "; font-weight: bold;
}
.narrator::after {
  content: " «"; font-weight: bold;
}
.hermit::before {
  content: "🧓 : "; font-size: 24px;
}
.fu-tzu::before {
  content: "🧚 : "; font-size: 24px;
}
```



THE CASCADE

THE CASCADE IN CASCADING STYLE SHEETS

- Sometimes, two or more rules might apply to the same element
- These rules might be **conflicting**, i.e., assign different values to the same property (e.g.: `color`)
- **The cascade** is the algorithm used to **resolve** such **conflicts**
  - **Input:** a set of conflicting properties that apply to a given element
  - **Output:** a single, cascaded, property to actually apply
- The cascade considers **4 key aspects, in order:**
  1. **Origin and Importance**
  2. **Layers**
  3. **Specificity**
  4. **Position and order of appearence** of the rule

A volte succede che 2 o più regole si applicano allo stesso elemento
Queste regole possono andare in conflitto, ad esempio si assegnano
diversi valori alla stessa proprietà.
Il cascade è l'algoritmo usato per risolvere questi
conflitti:
- input: un insieme di proprietà in conflitto che si applica a dato elemento.
Output: un singlo, cascaded, proprietà da applicare attualmente.
Il cascade considera 4 elementi chiave, in ordine:
- Origine e importanza
- livelli
- specificità

THE CASCADE: ORIGIN

- The CSS we write (a.k.a. **authored CSS**) is not the only one being applied to a web page
- We've already mentioned that **user agent styles** exist
  - The stylesheets that are included by browsers by default
- Other styles (a.k.a. **local user styles**) might be added by specific browser extensions or from the operating system level
  - For example, for accessibility purposes
  - Visually-impaired persons might want to use high-contrast color schemes, with larger fonts, etc.

Il CSS che scrive (authored CSS) non è l'unico a essere applicato alla pagina web.
Abbiamo appena menzionato che esistono gli stili user-agent. I fogli di stile che sono inclusi nel browser di default.
Altri stili (local user styles) possono essere aggiunti
da specifiche estensioni browser o dal livello dell'OS.
Ad esempio per aspetti di accessibilità.

La regola important può essere usata per aggiungere
più importanza a una proprietà dentro una regola CSS.
!important è aggiunto alla fine della dichiarazione della proprietà.
L'importanza aggiunge un ruolo significativo nella

THE CASCADE: IMPORTANCE

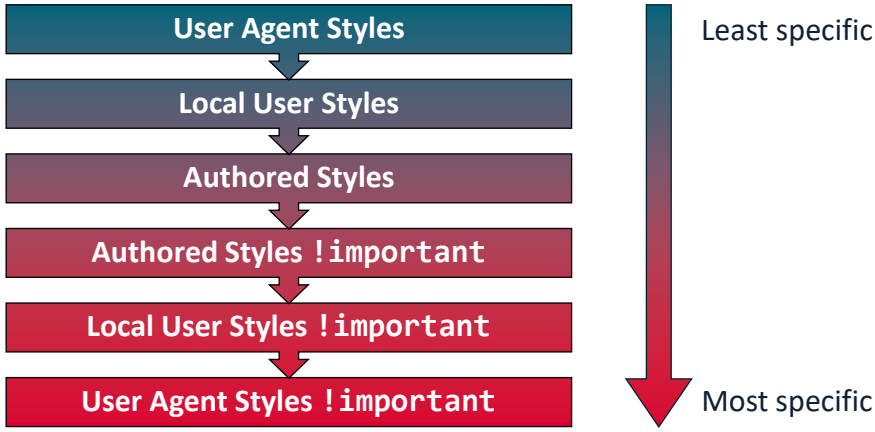
- The `!important` rule can be used to add more importance to a property inside a CSS rule
- `!important` is simply added at the end of the property declaration

```
h1 {
  color: red !important;
}
```

- Importance plays a significant role in the cascade

THE CASCADE: ORIGIN–IMPORTANCE

From the lest specific origin to the most specific one

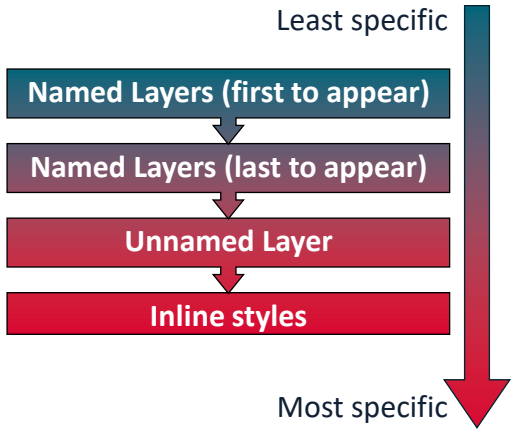


Se dentro ogni bucket di origine/importnaza possono
esserci multipli layer cascade.
Con stili authored:
- i layer custom possono essere definiti usando @layer regola. I layer custom che sono dichiarati dopo hanno maggior priorità.
Tutti gli altri CSS (in <style> o importti con <link>) appartengono a layer senza nome.
Gli stili inline appartengono a un layer separato e hanno la priorità più alta.

THE CASCADE: LAYERS

Within each origin/importance bucket, there can be multiple cascade **layers**

- Within authored styles:
  - Custom layers can be defined using **@layer** rule (we won't see that)
    - The custom layers that are declared later have higher priority
  - All other CSS (in **<style>** or imported with **<link>**) belongs to a unnamed layer
  - Inline styles belong to a separate layer and have the highest priority



THE CASCADE: SPECIFICITY

Quando 2 regole sono in conflitto:
- appartengono allo stesso bucket origine-importanza
- appartengono allo stesso layer
Si considera la specificità.
L'idea è che il selettore più specifico dovrebbe vincere.

When two conflicting rules:

- Belong to the same origin-importance bucket, and
- Belong the same layer

Specificity is considered.

- The idea is that the **most specific** selector should win

```
.primary {
  color: blue;
}
```

VS

```
h1 {
  color: red;
}
```

```
<h1 class="primary">Cascading!</h1>
```

CSS definisce come calcolare la specificità di un selettore.
- ignora il selettore universale
- conta il numero dei selettori id (A)
- conta il numero di classi, attributi e selettori di pseudoclassi (B)
- conta il numero di tipi e selettori di pseudoelementi (C).
La specificità è una tripla (A,B,C) calcolata come sopra.

THE CASCADE: SPECIFICITY

CSS defines how to calculate the specificity of a selector:

- Ignore the universal selector
- Count the number of id selectors (=A)
- Count the number of class, attribute and pseudo-classes selectors (=B)
- Count the number of type and pseudo-element selectors (=C)

The specificity is a numeric triple **(A, B, C)** computed as above

THE CASCADE: SPECIFITY EXAMPLES

- A: Number of id selectors
- B: Number of class, attribute and pseudo-classes selectors
- C: Number of type and pseudo-element selectors

Selector	Specificity (A, B, C)
`#id`	(1, 0, 0)
`em.master[target]`	(0, 2, 1)
`#navbar ul li a.nav-link[href*='/']`	(1, 2, 3)
`article.item section p::first-letter`	(0, 1, 4)
`a:hover`	(0, 1, 1)
`*`	(0, 0, 0)

I confronti sono fatti considerando le 3 componenti
in ordine:
- la specificità con un valore di A maggiore è più specifico.
- se 2 A pareggiando, allora si deve vedere quale B
vince.
- se 2 B pareggiano, allora si deve vedere quale C
vince.
se tutti i valori pareggiano, allora hanno la

THE CASCADE: COMPARING SPECIFICITIES

Comparisons are made by considering the three components in order:

- the specificity with a larger **A** is more specific;
- if the two **A** are tied, then the specificity with a larger **B** wins;
- if the two **B** are also tied, then the specificity with a larger **C** wins;
- if all the values are tied, the two specificities are **equal**.

THE CASCADE: SPECIFICITY

Selector #1	Specif. #1	Selector #2	Specif. #2	Winner
`a[target]`	(0, 1, 1)	`.list a`	(0, 1, 1)	Draw
`#msg`	(1, 0, 0)	`input[type].inp`	(0, 2, 1)	#1
`#nav > #brd a.lk`	(2, 1, 1)	`em.foo.bar.light`	(0, 3, 1)	#1
`[id='nav'] a`	(0, 1, 1)	`#nav a`	(1, 0, 1)	#2

Quando 2 proprietà:
- appartengono allo stesso bucket
- allo stesso layer
- hanno la stessa specificità
Allora vince l'ultima regola che appare!

THE CASCADE: POSITION AND APPEARENCE

When two properties:

- Belong to the same origin/importance bucket
- Belong to the same layer
- Have the same specificity

The **last rule** to appear has the highest priority

```
h1 {
  color: red;
}
h1 {
  color: blue;
}
```

Questa regola si applica allo stesso foglio di stile e sull'ordine in cui i fogli di stile appaiono

THE CASCADE: POSITION AND APPEARENCE

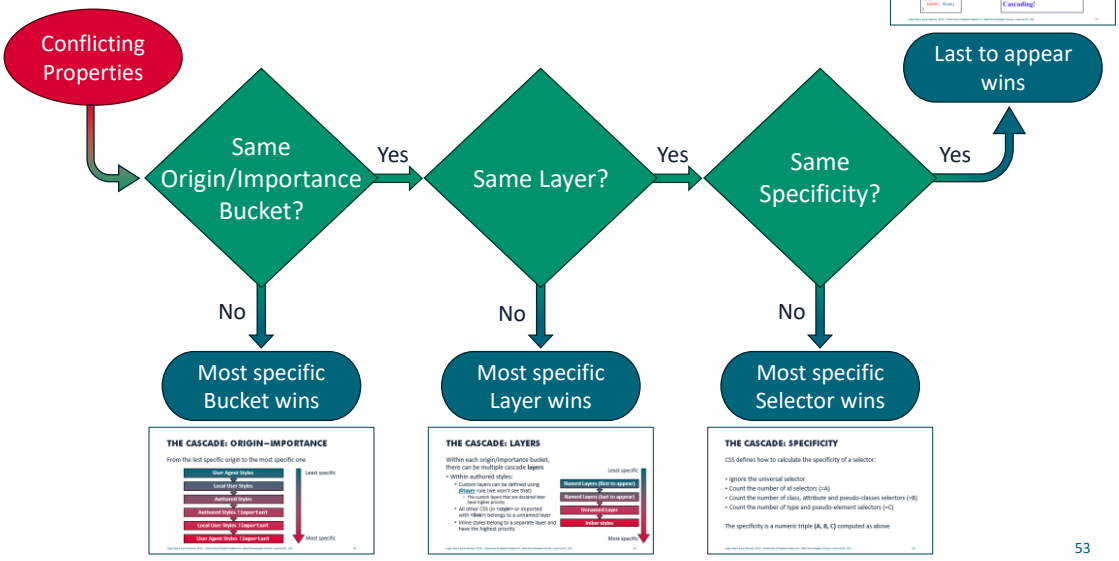
- This rule applies within the same stylesheet, and on the order in which stylesheets appear

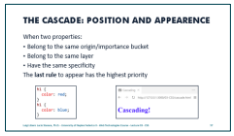
```
<head>
  <meta charset="UTF-8">
  <title>Cascading</title>
  <link rel="stylesheet" href="blue.css">
  <link rel="stylesheet" href="red.css">
</head>
<body>
  <h1>Cascading!</h1>
</body>
```

```
h1 {
  color: blue;
}
```

```
h1 {
  color: red;
}
```

THE CASCADE: OVERVIEW

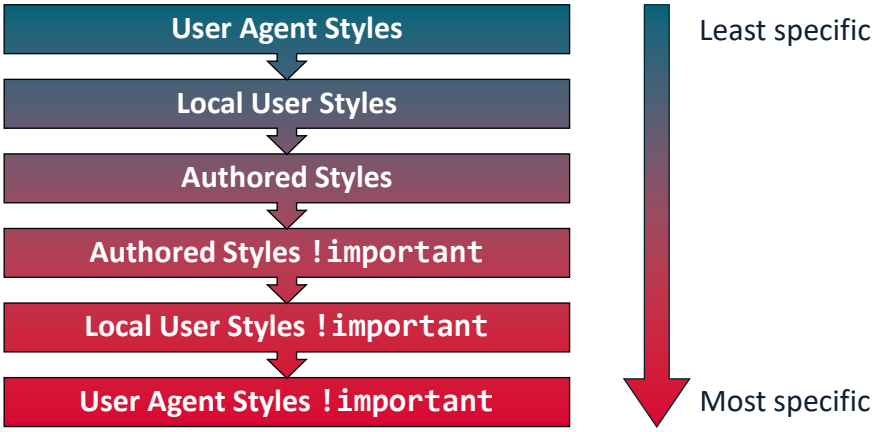




53

THE CASCADE: ORIGIN–IMPORTANCE

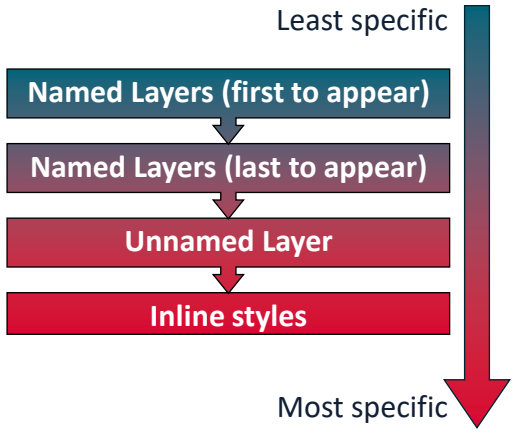
From the lest specific origin to the most specific one



THE CASCADE: LAYERS

Within each origin/importance bucket, there can be multiple cascade **layers**

- Within authored styles:
  - Custom layers can be defined using **@layer** rule (we won't see that)
    - The custom layers that are declared later have higher priority
  - All other CSS (in **<style>** or imported with **<link>**) belongs to a unnamed layer
  - Inline styles belong to a separate layer and have the highest priority



THE CASCADE: SPECIFICITY

CSS defines how to calculate the specificity of a selector:

- Ignore the universal selector
- Count the number of id selectors (=A)
- Count the number of class, attribute and pseudo-classes selectors (=B)
- Count the number of type and pseudo-element selectors (=C)

The specificity is a numeric triple **(A, B, C)** computed as above

THE CASCADE: POSITION AND APPEARENCE

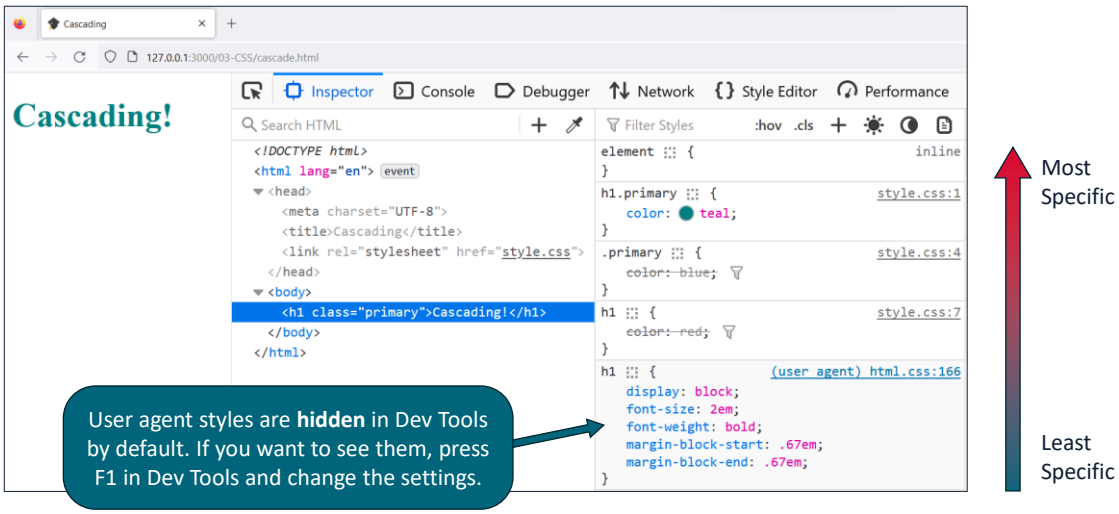
When two properties:

- Belong to the same origin/importance bucket
- Belong to the same layer
- Have the same specificity

The **last rule** to appear has the highest priority

```
h1 {
  color: red;
}
h1 {
  color: blue;
}
```

THE CASCADE IN BROWSER DEV TOOLS



INHERITANCE

INHERITANCE IN CSS

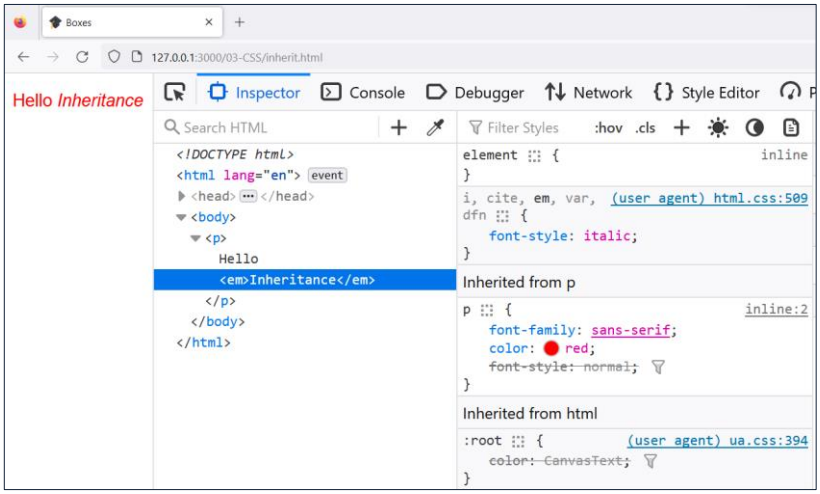
Alcune proprietà possono essere ereditate dagli
elementi antenati, se nessun valore specifico è impostato.
Le proprietà ereditabili includono colore, font-size,
font-family, font-weight, font-style.

- Some CSS properties can be inherited from ancestor elements, if no specific value is set
- Inheritable properties include **color**, **font-size**, **font-family**, **font-weight, font-style**

```
p {
  font-family: sans-serif;
  color: red;
  font-style: normal;
}
```

```
<p>
  Hello <em>Inheritance</em>
</p>
```

INHERITANCE IN CSS



Inherited properties have the lowest specificity of all styling methods

Le proprietà ereditate hanno la specificità più bassa di tutti i metodi di stilizzare.

ASSIGNMENT #2

Today's lecture comes with **Assignment #2**! In this assignment, you will:

- Do some practice with basic CSS
- Write some tricky CSS rules
- Test your knowledge of the Cascade algorithm

Note: the live HTTP server we setup in **Exercise 1** of **Assignment #1** will be handy for this assignment! Unless you are already familiar with HTTP servers and already know what you're doing, make sure you completed at least **Exercise 1** in **Assignment #1** before doing **Assignment #2**!

REFERENCES

- **Learn CSS**
  web.dev
  https://web.dev/learn/css/
  Sections: 1, 3 to 6, 14, 15
- **Introducing the CSS Cascade**
  MDN web docs
  https://developer.mozilla.org/en-US/docs/Web/CSS/Cascade
- **Flukeout: A game-based approach to learning CSS selectors**
  https://flukeout.github.io/