Cascading Style Sheets

Web development I: Front-end engineering

Why CSS

UNIVERSITÉ DU LUXEMBOURG

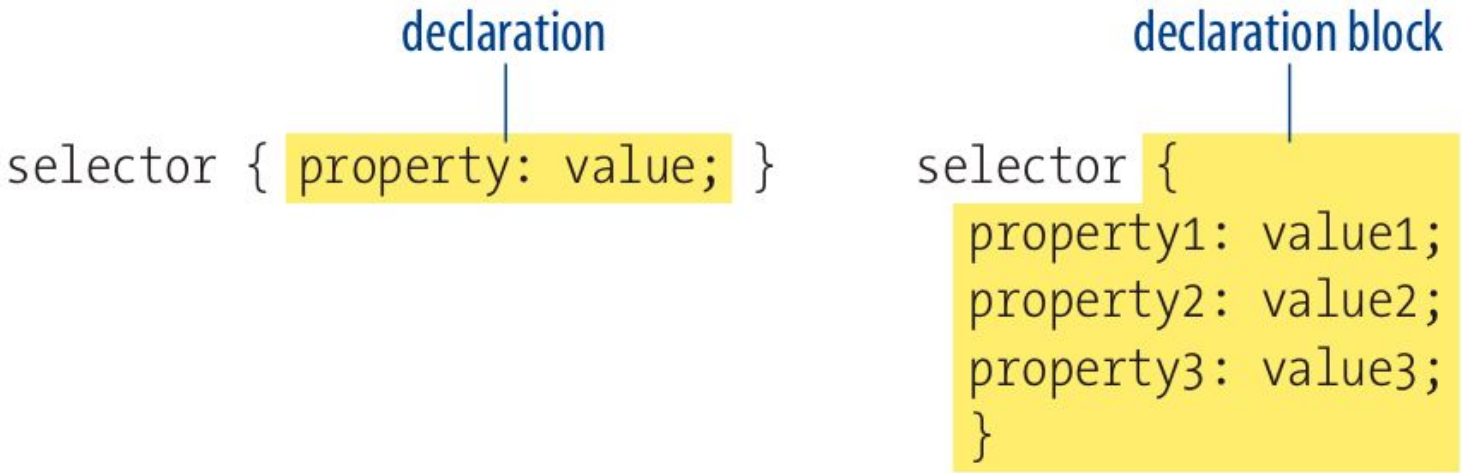
Precise type and layout controls

Less work

More accessible sites

Reliable browser support

Writing CSS


declaration
selector { property: value; }
declaration block
selector {
property1: value1;
property2: value2;
property3: value3;
}

Adding CSS

UNIVERSITÉ DU LUXEMBOURG

External file:

```
<link href="/path/to/file.css" rel="stylesheet" />
```

Embedded in the HTML:

```
<style>
@import url("/path/to/file.css");
p { color: red; }
</style>
```

Inline (*bad practice*):

```
<p style="color:red">Hello</p>
```

Some notes

UNIVERSITÉ DU LUXEMBOURG

Whitespace (in)significance:

```
p { color: red; }
```

```
p {
 color: red;
}
```

```
p{color:red}
```

Comments:

```
/* This is ignored. */
```

Grouped selectors:

```
p, h1, h2 { … }
```

Inheritance

Unstyled paragraph

It's the *back* of the note that's driving me crazy.

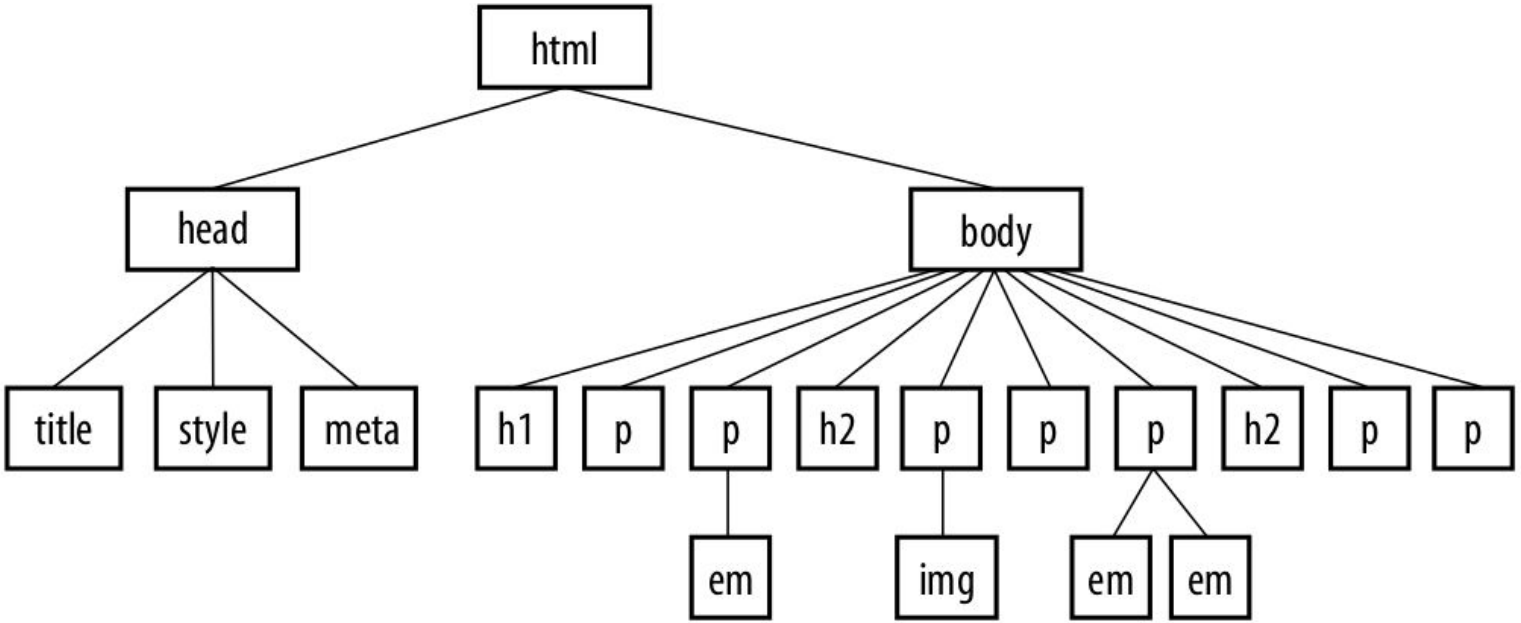
```
p {font-size: small; font-family: sans-serif;}
```

Paragraph with style rule applied

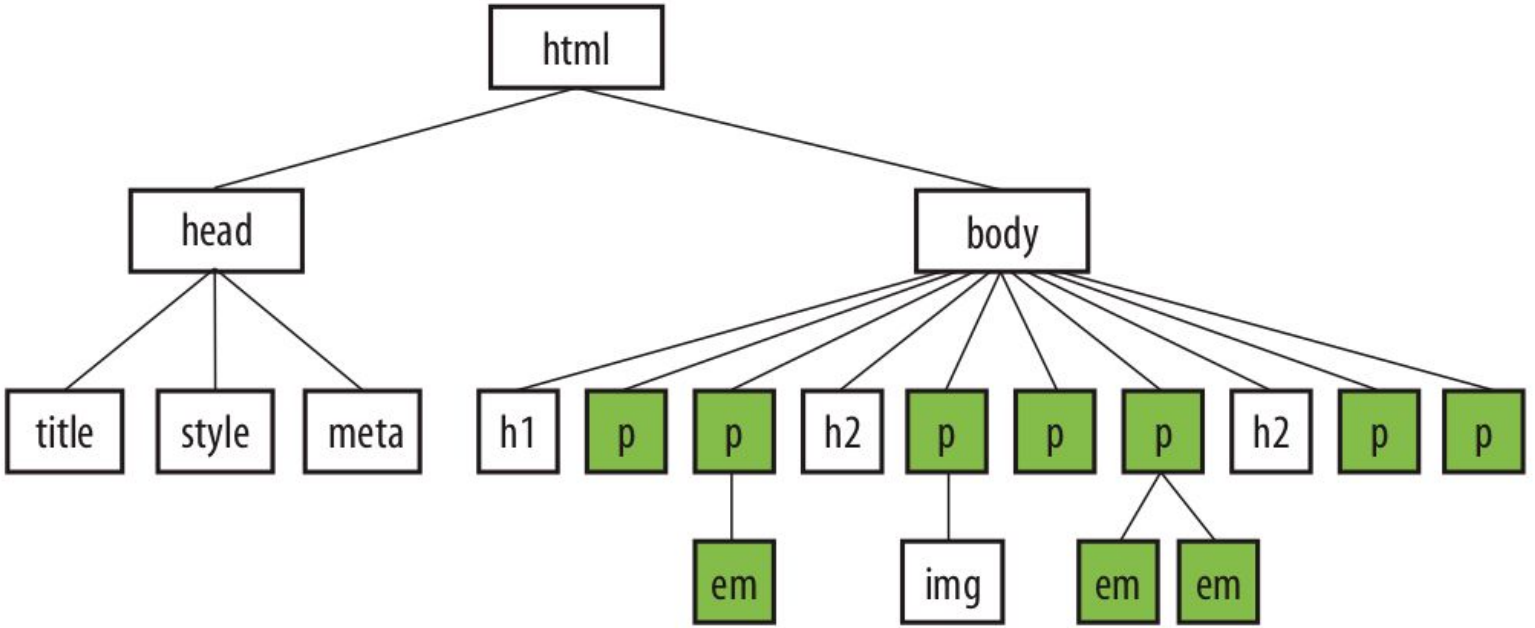
It's the *back* of the note that's driving me crazy.

The emphasized text (em) element is small and sans-serif even though it has no style rule of its own. It *inherits* the styles from the paragraph that contains it.

Inheritance and document structure

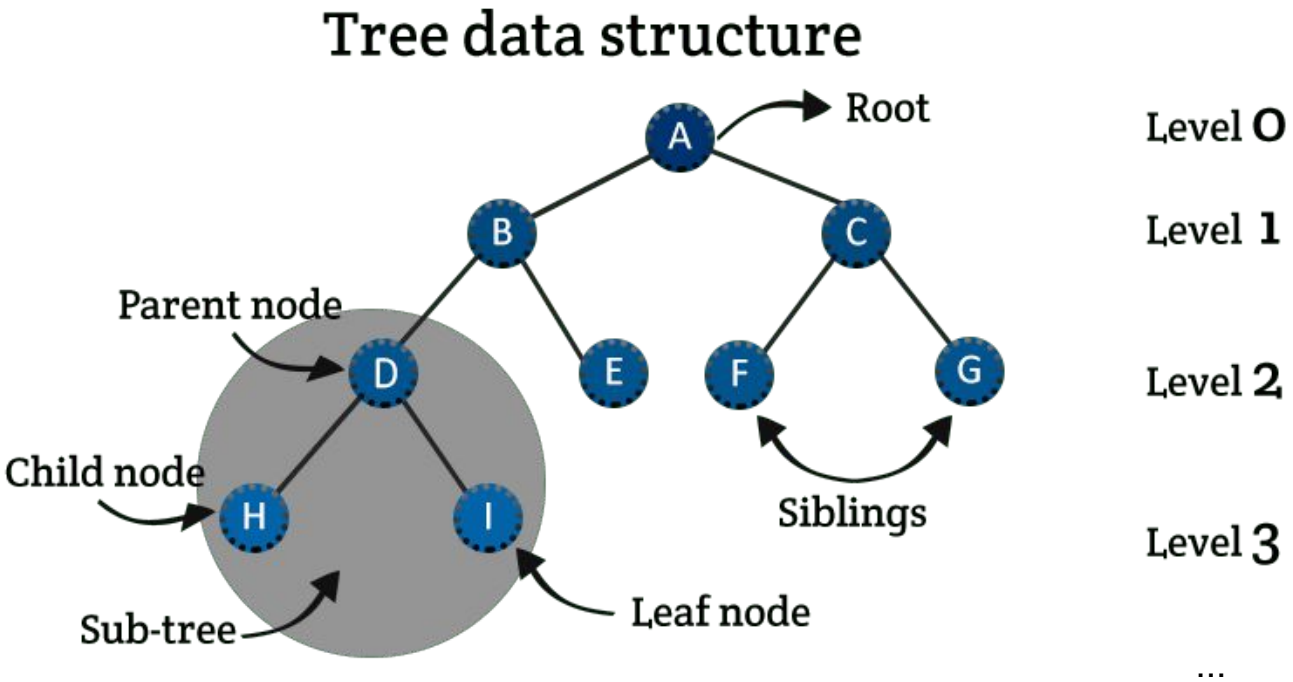


Inheritance and document structure




```
p {font-size: small; font-family: sans-serif;}
```

Document **tree** structure



Contextual selectors

Descendant selector (note the whitespaces):

```
ul li a { … }
```

Child selector:

```
p > em { … }
```

Adjacent sibling selector:

```
h1 + p { … }
```

General sibling selector:

```
h1 ~ h2 { … }
```

Class and ID selectors

UNIVERSITÉ DU LUXEMBOURG

Class notation: dot (.) + class name

```
.warn { color: orange; }
```

Classes can be shared across HTML elements:

```
<p class="warn">Hi</p>

<div class="warn">Bye</p>
```

ID notation: hash (#) + id

```
#top { color: blue; }
```

Only one ID is allowed in the whole HTML document!

```
<h1 id="top">Welcome</h1>
```

Combining selectors

UNIVERSITÉ DU LUXEMBOURG

Increase **specificity**:

```
main > section p.warn { … }
```

Conflicting styles: the cascade

Style information is passed down (“cascades” down) until it is overridden by a style command *with more specificity* (weight).

Rule order: Last rule always wins

```
<style>
p { color: red; }
p { color: blue; }
p { color: green; }
</style>
```

```
<style>
p { color: red;
    color: blue;
    color: green; }
</style>
```

Conflicting styles: the cascade

UNIVERSITÉ DU LUXEMBOURG

From less to more *priority*:

1. Default browser styles
2. User style settings (set as “reader style sheet”)
3. Linked external style sheet (added with `<link>` element)
4. Imported style sheets (added with `@import`)
5. Embedded style sheets (added with `<style>` element)
6. Inline style information (added with `style` attribute)
7. Any style rule marked as `!important`

Conflicting styles: the cascade

UNIVERSITÉ DU LUXEMBOURG

From less to more *priority*:

1. Universal selector: `* { … }`
2. Individual selectors
3. Contextual selectors
4. Class selectors
5. ID selectors
6. The `!important` flag.

Pseudo-class selectors

UNIVERSITÉ DU LUXEMBOURG

`:link`	Never clicked element
`:visited`	Already clicked element
`:hover`	Mouse pointer over the element
`:active`	Element being clicked
`:focus`	Element selected and ready for input

Pseudo-class selectors

UNIVERSITÉ DU LUXEMBOURG

Samples of my work:

- Pen and Ink Illustrations
- Paintings
- Collage

a:link
Links are maroon and not underlined.

Samples of my work:

- Pen and Ink Illustrations
- Paintings
- Collage

a:focus
a:hover
While the mouse is over the link or when the link has focus, the pink background color appears.

Samples of my work:

- Pen and Ink Illustrations
- Paintings
- Collage

a:active
As the mouse button is being pressed, the link turns bright red.

Samples of my work:

- Pen and Ink Illustrations
- Paintings
- Collage

a:visited
After that page has been visited, the link is gray.

Pseudo-class selectors in forms

UNIVERSITÉ DU LUXEMBOURG

`:enabled`	Element is enabled
`:disabled`	Element is disabled
`:checked`	Element is checked (radio, checkbox)
`:required`	Element is required
`:autofill`	Matches content filled by the browser
`:invalid`	Element has invalid content (e.g. malformed email)

More pseudo-class selectors

`:root`	Useful to declare CSS variables
`:empty`	Element with no content (and no whitespace)
`:target`	Element ID matches URL fragment
`:not()`	Do not match given selector(s)
`:lang()`	Match elements based on language
`:has()`[1]	Match *all* elements with forgiving selectors
`:is()`[1]	Match *any* elements with forgiving selectors
`:where()`[1]	Same as `:is()` but using 0 specificity

[1] Not fully supported yet in all major browsers

More pseudo-class selectors

UNIVERSITÉ DU LUXEMBOURG

`:first-child`	First element in group of siblings
`:last-child`	Last element in group of siblings
`:only-child`	Element with no siblings
`:first-of-type`	First element in group of siblings
`:last-of-type`	Last element in group of siblings
`:only-of-type`	Element has no siblings of the same type

More pseudo-class selectors

`:nth-child()`	Matches based on position among siblings
`:nth-last-child()`	… counting from the end
`:nth-of-type()`	Matches based on type among siblings
`:nth-last-of-type()`	… counting from the end

Pseudo-element selectors

UNIVERSITÉ DU LUXEMBOURG

`::first-line`	Matches first line of (block-level) element
`::first-letter`	Matches first letter of (block-level) element
`::before`	Creates a first child of the element
`::after`	Creates a last child of the element
`::selection`	Matches highlighted content

Attribute selectors

UNIVERSITÉ DU LUXEMBOURG

`element[attribute]`	Simple attribute value selector
`element[attribute = "value"]`	Exact attribute value selector
`element[attribute ~= "value"]`	Partial attribute (*word*) value selector
`element[attribute |= "value"]`	Beginning hyphen-separated value selector
`element[attribute ^= "value"]`	Beginning substring value selector
`element[attribute $= "value"]`	Ending substring value selector
`element[attribute *= "value"]`	Arbitrary substring value selector

How many CSS properties are there?

UNIVERSITÉ DU LUXEMBOURG

“541 distinct property names, based on 73 technical reports.”

— https://www.w3.org/Style/CSS/all-properties.en.html