CMPT 165
Advanced XHTML + CSS – Part 5

June 10th, 2015

New concepts from last class

- Inheritance and specificity

- Contextual selectors

  +

  >

CSS rules

Inheritance

"is a way of propagating [cascading] property values from parent elements to their children."

-- CSS3 working draft specs

Specificity

- Allows us to override inherited rules
- Every stylerule has a weight on specificity
- Higher weight → more "rule" power

Contextual selectors: immediate child

Represented by symbol >

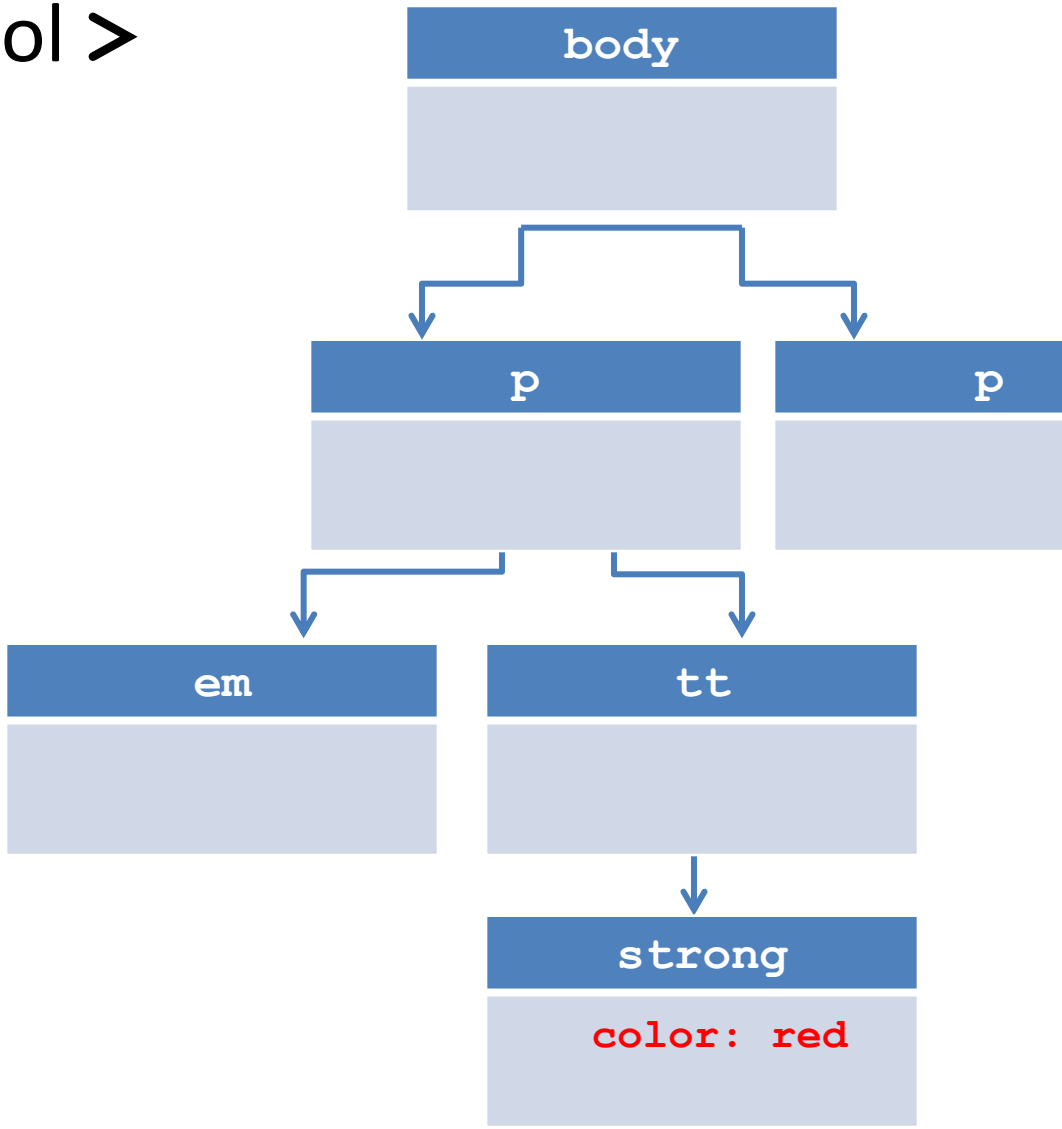
```
p strong {
   color: red;
}
```

More specific rule:

```
tt > strong {
   color: red;
}
```

Even more specific:

```
p > tt > strong {
   color: red;
}
```



Contextual selector: immediate child

```
<ol>
  <li>Item
    <ul>
      <li> Item </li>
      <li> Item </li>
      <li> Item
        <ul>
          <li>Item</li>
          <li>Item</li>
        </ul>
      </li>
    </ul>
  </li>
  <li>Item
    <ul>
      <li> Item </li>
      <li> Item
        <ol>
          <li>Item</li>
        </ol>
      </li>
    </ul>
  </li>
</ol>
```

 is not immediate child of <ul> (immediate child is <li>)

1. Item
    - Item
    - Item
    - Item
        - Item
        - Item
2. Item
    - Item
    - Item
        1. Item

```
ul ol {
  color: red;
}
```

1. Item
    - Item
    - Item
    - Item
        - Item
        - Item
2. Item
    - Item
    - Item
        1. Item

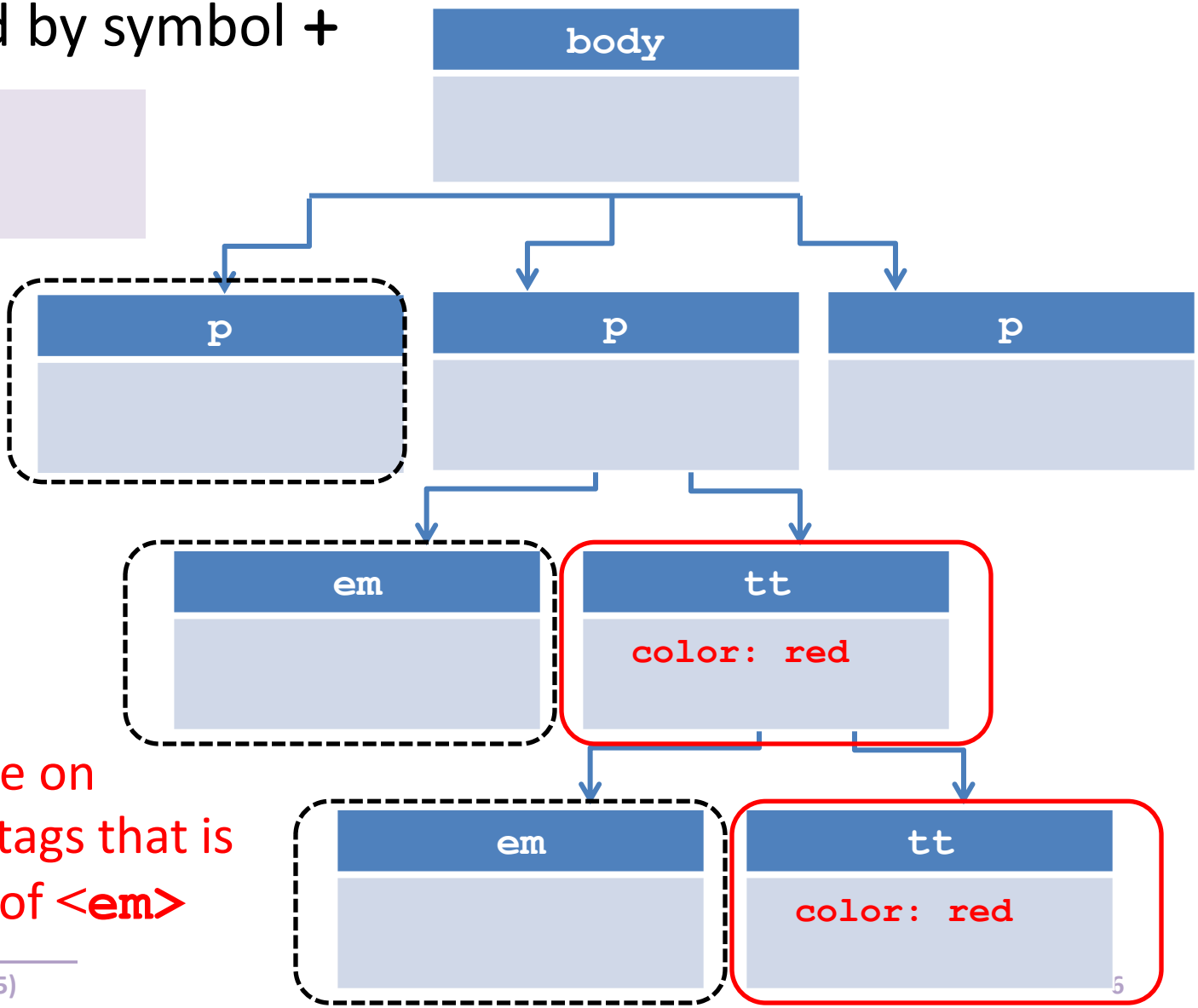
```
ul > ol {
  color: red;
}
```

?

Contextual selectors: next child

Represented by symbol **+**

```
em+tt {
  color: red;
}
```



Apply style rule on **all** **<tt>** tags that is the next child of ****

Today's Agenda

- Overriding styles
- Pseudo-elements
- Pseudo-classes

Overriding stylerules: inline style specifications

```
<h1 class="class_fantasy class_serif">Hello!</h1>
```

```
.class_serif
{
font-family: serif;
}
.class_fantasy
{
font-family: fantasy;
}
```

```
<h1 style="font-family:sans-serif" class="class_fantasy class_serif" >Hello!</h1>
```

- Inline style overrides…
- Use sparingly!

Recall 3 ways to specify styles:

a) Inline style
b) Style in <head></head>
c) External stylesheet with <link>

!important

- Yet another way to override specificity of a declaration

```
.class_serif
{
font-family: serif;
}
.class_fantasy
{
font-family: fantasy;
}
```

```
.class_serif
{
font-family: serif !important;
}
.class_fantasy
{
font-family: fantasy;
}
```

```
<h1 class="class_fantasy class_serif">Hello!</h1>
```

- Applies to individual declaration only

```
.class_serif
{
        font-family: serif !important;
        font-size: 12pt !important;
}
.class_fantasy
{
        font-family: fantasy !important;
}
```

- Order still matters!

Sources of stylerules and their specificities

Sources

1. Author (you)
2. User agent (default settings given by browser)
3. User
   - Provide his/her own stylesheet… or none
   - How-to in FireFox: http://webdesign.about.com/od/css/ht/htcssuserfirefo.htm

Highest specificity:

1. User **`!important`** declarations
2. Author **`!important`** declarations
3. Author normal declarations
4. User normal declarations
5. User agent

a) Inline style
b) Header style
c) External stylesheet

Today's Agenda

- Overriding styles
- Pseudo-elements
- Pseudo-classes

Pseudo-elements

"Pseudo": not actual, unreal

Pseudo-element: not actual element but behaves like one

Usage: style specific parts of an element

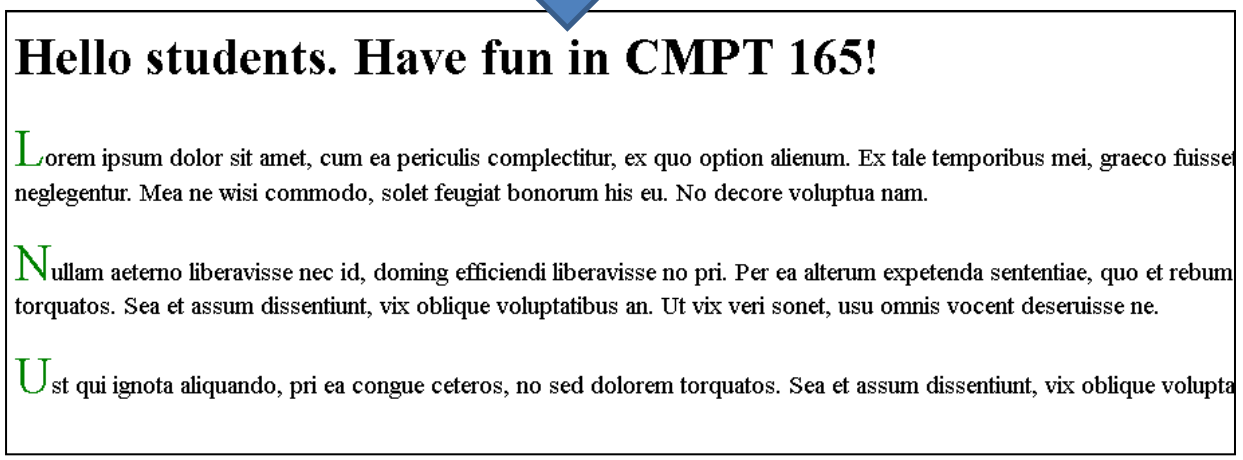
Ones you may find most useful:

- `:first-letter`
- `:first-line`
- `:before`
- `:after`

Pseudo-elements

Style the first letter, first line of an element

```
p:first-letter{
    color: green;
    font-size: xx-large;
}
```



Pseudo-elements

Style the first letter, first line of an element

```
p:first-letter{
        color: green;
        font-size: xx-large
}
p:first-line{
        color: blue;
        font-variant: small-caps;
}
```

Hello students. Have fun in CMPT 165!

LOREM IPSUM DOLOR SIT AMET, CUM EA PERICULIS COMPLECTITUR, EX QUO OPTION ALIENUM. EX TALE TEMPORIBUS MEI, GRAECO FUISSET omittam an vel, sed ex brute decore. Ea putant fuisset patrioque eum, atqui integre vivendum cum ex, cu sea sadipscing neglegentur. Mea ne wisi commodo, solet feugiat bonorum his eu. No decore voluptua nam.

NULLAM AETERNO LIBERAVISSE NEC ID, DOMING EFFICIENDI LIBERAVISSE NO PRI. PER EA ALTERUM EXPETENDA SENTENTIAE, QUO ET rebum nominati dissentiunt, quis diceret rationibus id pri. Ut qui ignota aliquando, pri ea congue ceteros, no sed dolorem torquatos. Sea et assum dissentiunt, vix oblique voluptatibus an. Ut vix veri sonet, usu omnis vocent deseruisse ne.

Pseudo-elements

Insert additional content before or after an element

`:after`

`:before`

```
blockquote:after {
    content: "--L.T.";
}
```

Lorem ipsum dolor sit amet, ut sea eripuit euismod. Id pri nostro option vidisse, vim no explicari la concludaturque an. Vix no debitis singulis comprehensam. Cibo facer viris quo in, duo alia partem consequat ea. Vel soleat epicurei omittantur ex, wisi nostrud denique at quo, te quo quod laboramus. Sensibus splendide ius eu, et habemus nominati vix. Eam no nonumy explicari, est idque nonumy intellegebat in. -- L.T.

Pri debitis adipisci quaestio in. Prima deserunt vituperata pro ea, ex vis ullum dicit semper. Unum duis dolorum et cum, his in lorem nonumes disputando. Eam iusto efficiantur ea, ius rebum primis patrioque ne. -- L.T.

Ne dolore aliquip appellantur vel, eam noster scripta dolorum id, pro ad sonet minimum tincidunt. Ei vix quod principes pertinacia, ut quo falli antiopam. Officiis adipiscing. -- L.T.

Pro facete voluptatum id, euismod quaerendum cotidieque an his, iusto forensibus mel cu. Sonet postulant accommodare vix ex. Eu reque noster antiopam mei, ut munere persecuti expetendis per. Pri legere aliquam utroque cu. Feugait pertinax contentiones duo an. Justo electram cum id, vix ea reque menandri, regione aliquid appellantur mel ut. -- L.T.

Today's Agenda

- Overriding styles
- Pseudo-elements
- Pseudo-classes

Pseudo-classes

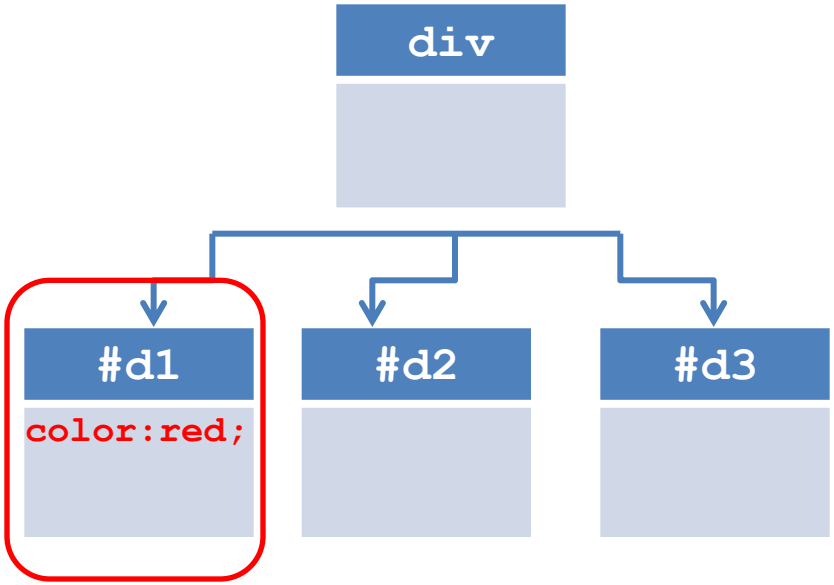
Pseudo-class: not actual class but behaves like one

E.g.

`:first-child`	first child of an element
`:last-child`	last child of an element

```
<div>
    <div id="d1">123456</div>
    <div id="d2">abc    </div>
    <div id="d3">abcdef</div>
</div>
```

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

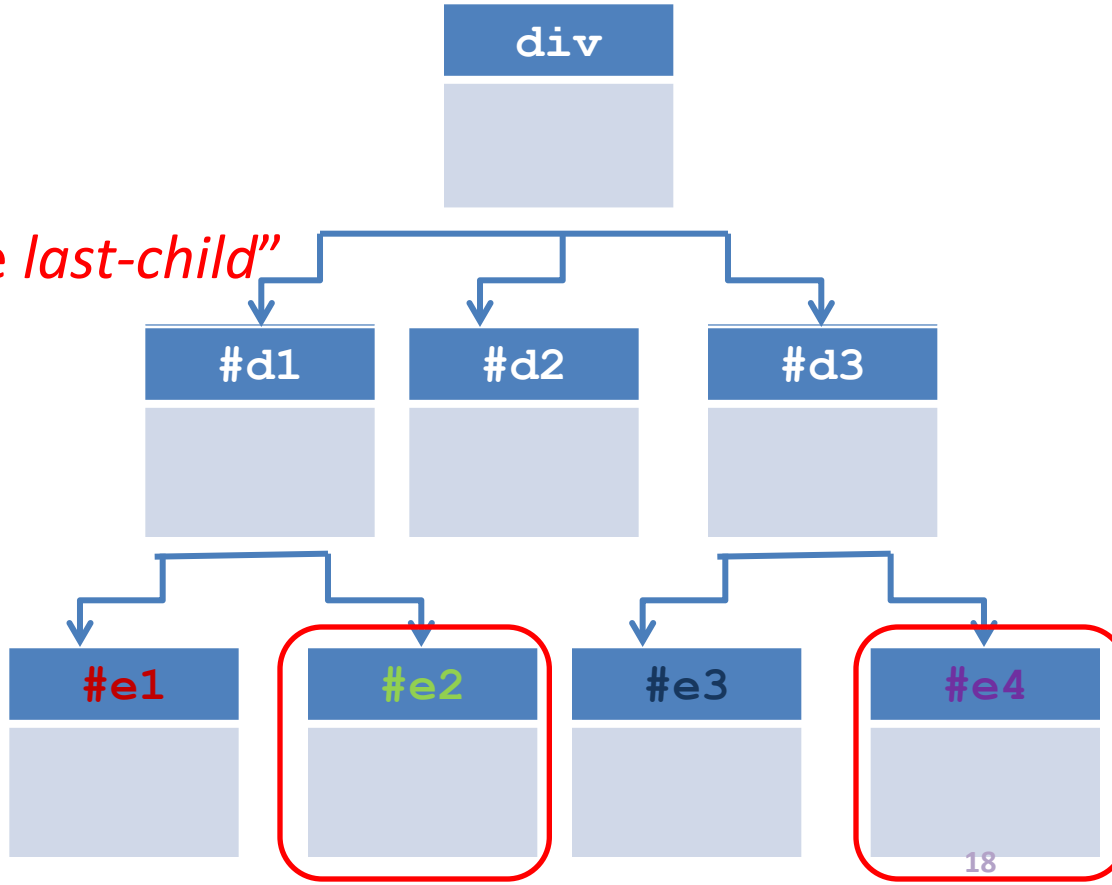


:last-child

```
<div>
    <div id="d1"><em id="e1">a</em>  <em id="e2">b</em></div>
    <div id="d2">                                      </div>
    <div id="d3"><em id="e3">c</em>  <em id="e4">d</em></div>
</div>
```

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

"Select all **** tags that are *last-child*"



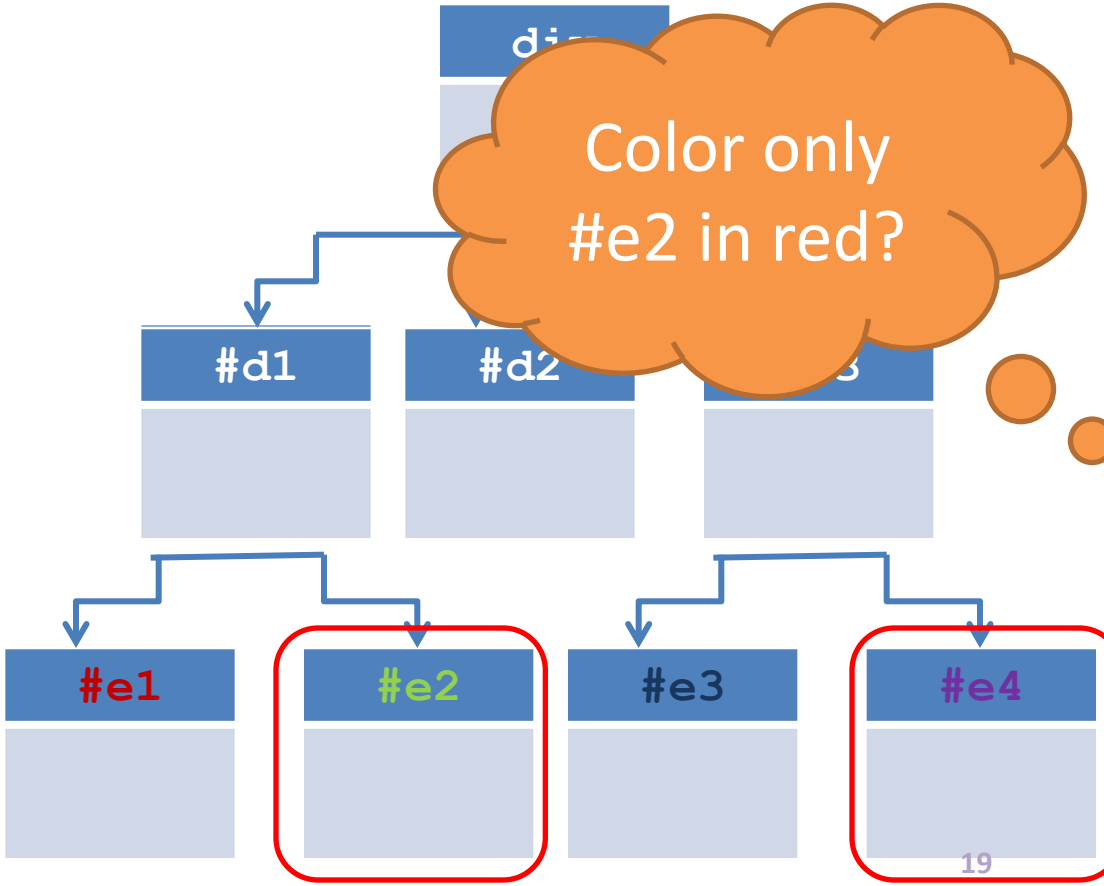
Browser window:

a b
c d

:last-child

```
<div>
    <div id="d1"><em id="e1">a</em>  <em id="e2">b</em></div>
    <div id="d2">                                        </div>
    <div id="d3"><em id="e3">c</em>  <em id="e4">d</em></div>
</div>
```

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



:last-child

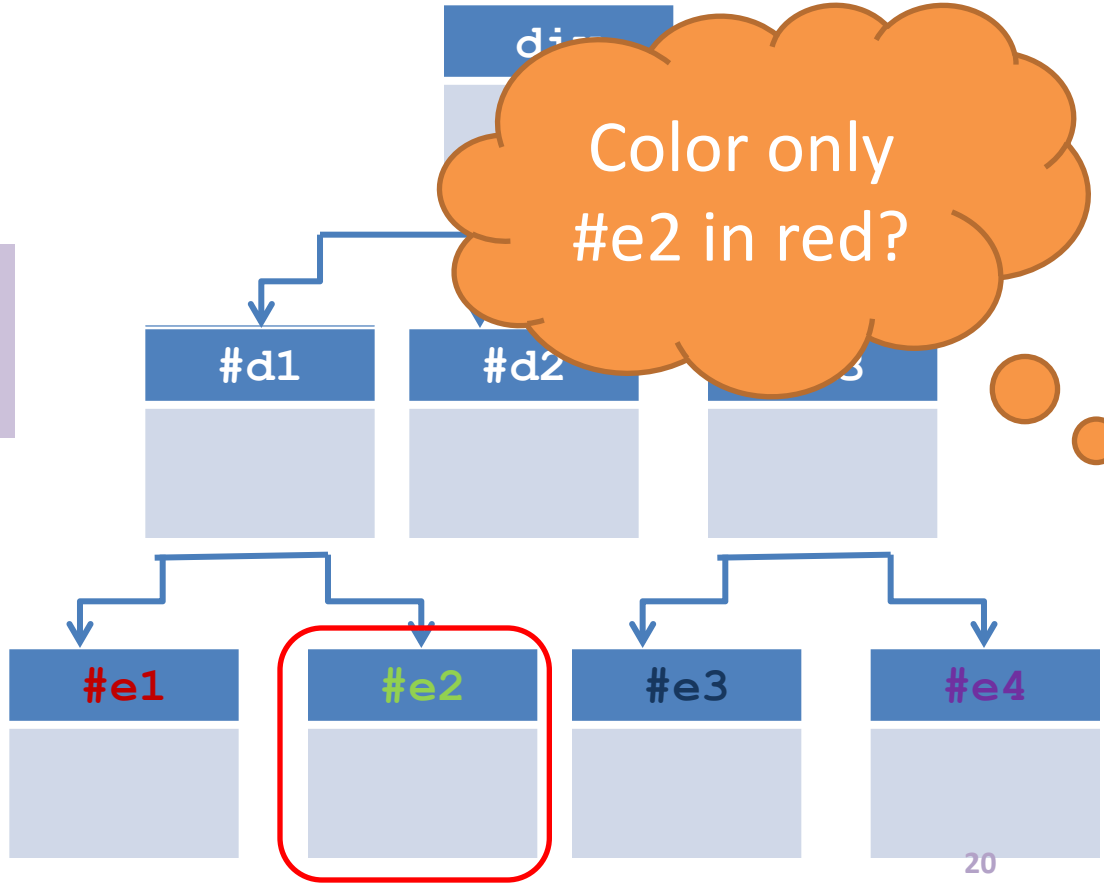
"Select **all** tags in #d1 that are *last-child*"

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

```
#d1 em:last-child{
    color: red;
}
```

Browser window:

a b
c d



More common pseudo-classes

- In anchor elements `<a>`

`:link`	unvisited link
`:visited`	visited link
`:active`	mouse clicked on link
`:hover`	mouse hover on link

- Called pseudo-classes, but they really refer to "state"
  - i.e. depends on users' action, rendered dynamically by user's browser

More common pseudo-classes

- Simple (good) way to add interact with user

```
a {
      text-decoration: none;
      color: #86759A;
}
a:link {
      font-weight: bold;
}
a:visited {
      font-weight: normal;
}
a:hover, a:focus, a:active {
      text-decoration: underline;
      /*Try on your own: text-decoration: overline; */
}
```

Demo

Notes on anchor-based pseudo-classes

- **`a:hover`** should come after **`a:link`** and **`a:visited`**
- **`a:active`** should come after **`a:hover`**
  - Know why? ☺
- Can join:

  `a:visited:hover`
- Order does not matter

:hover, :active

- Works for any element
  - E.g. lists, table rows, dividers, imgs, etc.

- If properly used, a good/cool way to interact with user ☺

```
<div id="nav"> <!-- subdivider#1 -->
    <ul>
    <li><a href="demo.html" title="Home page">Home</a></li>
    <li><a href="demo.html" title="About me" >About me</a></li>
    <li><a href="demo.html" title="My notes">My course notes</a></li>
    <li><a href="demo.html" title="Contact">Contact</a></li>
    </ul>
</div>
```

Demo

Summary of stylerules specification

Ancestor-descendent:

```
selector1 selector2 … selectorN {
        property: value;
}
```

Immediate child:

```
selector1 > selector2  … > selectorN {
        property: value;
}
```

Siblings:

```
selector1 + selector2 … + selectorN {
        property: value;
}
```

Pseudo-element:

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

Pseudo-class:

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

In CSS3, distinction is made:

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

Specificity: same as class

Guidelines to defining stylerules

1. As start, use generic selectors
2. Order of selectors matter
3. Specificity are easier to edit and maintain
4. Add specificity as you go along

- Use the least number of selectors required to style an element
- Avoid `!important` and inline styling whenever possible
- Using advanced selectors gives you more control
- Don’t create unnecessarily complicated ones

Today's Highlights

- Specificity recapped
  - Specificity calculator???
    - http://specificity.keegan.st/
    - http://css-specificity.webapp-prototypes.appspot.com/
- Overriding with:

  1) inline stylerules

  2) `!important` … Use these sparingly!
- Pseudo-elements: just 4 key ones you need to understand
- Pseudo-class:
  - Various: http://www.w3schools.com/css/css_pseudo_classes.asp
  - Key ones you should exercise on your own:

    `:hover`

    `:active`

Brief notes on
Character Entities

Character entities

- End of Unit 4 of Study Guide
- <, > are part of Markup tags
- Character entities are used to display reserved characters

  E.g. non-breaking space: ` `

  "♪": `&#x266A;`
- Case sensitive; e.g. this won't work: `&Copy;`
- Don't forget to end with semicolon
- 3 ways to specify:

Approach	Starts with either…
name	&
decimal (dec)	&#
hexadecimal (hex)	&#x

Common mistakes

Reference

http://line25.com/articles/10-html-entity-crimes-you-really-shouldnt-commit

- Math:
  - "x", " / " keys for multiply sign should be avoided, use instead:

```
&times   &divide
```

  - Fractions: "1" "/" "4"

```
&frac14   &frac23
```

- Double quotes:
  - One on keyboard is use for measurement, e.g. 3'10"
  - Curly quotes should be used, e.g. "I can do it"

```
&ldquo; I can do it &rdquo;
```

Character entities: references

Links to key ones added on course webpage

Currency

http://www.w3schools.com/charsets/ref_utf_currency.asp

Math

http://www.w3schools.com/charsets/ref_utf_math.asp

Greek and Coptic Symbols

http://www.w3schools.com/charsets/ref_utf_greek.asp

Arrows

http://www.w3schools.com/charsets/ref_utf_arrows.asp

Miscellaneous

http://www.w3schools.com/charsets/ref_utf_symbols.asp

Misc.

User's own legibility stylesheet

```
html { background:#eef; }
body, html, * {
        color:black ! important;
        font-family:sans-serif;
}
body {
        padding:2em;
        background:white ! important;
        max-width:40em;
        line-height:1.1em;
        margin-left:auto;
        margin-right:auto;
}
b, strong {
        background-color:#eef;
        font-weight:normal;
}
a:link { color:blue ! important; }
a:active { color:red ! important; }
a:visited { color:navy blue ! important; }
```

Source: http://www.nimbustier.net/publications/web/user-css.html.en