CMPT 165
Advanced XHTML + CSS – Part 4

June 8th, 2015

More on `class`

An element may belong to multiple classes

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

<p class="class_dgreen">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. </p>

<p class="class_dred class_serif">Nullam aeterno
liberavisse nec id. Doming efficiendi liberavisse no pri.
Per ea alterum expetenda sententiae, quo et rebum
nominati dissentiunt, quis dice doming. </p>
```

```
.class_serif
{
font-family: serif;
}
.class_fantasy
{
font-family: fantasy;
}
.class_dgreen
{
    color: #030;
}
.class_dred
{
    color: #600;
}
```

More on `class`

Q1) What if:

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

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

Ans: fantasy. Order in CSS matters, i.e. "class_fantasy" overrides "class_serif"

Q2) What if:

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

Ans: same effect. Order in markup does not matter!

Key concepts

Today:

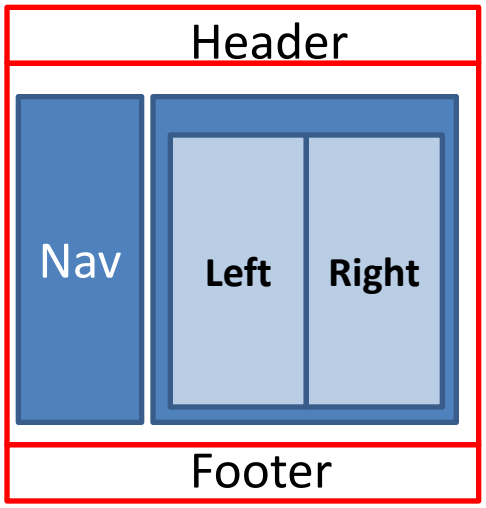
- More on contextual selectors

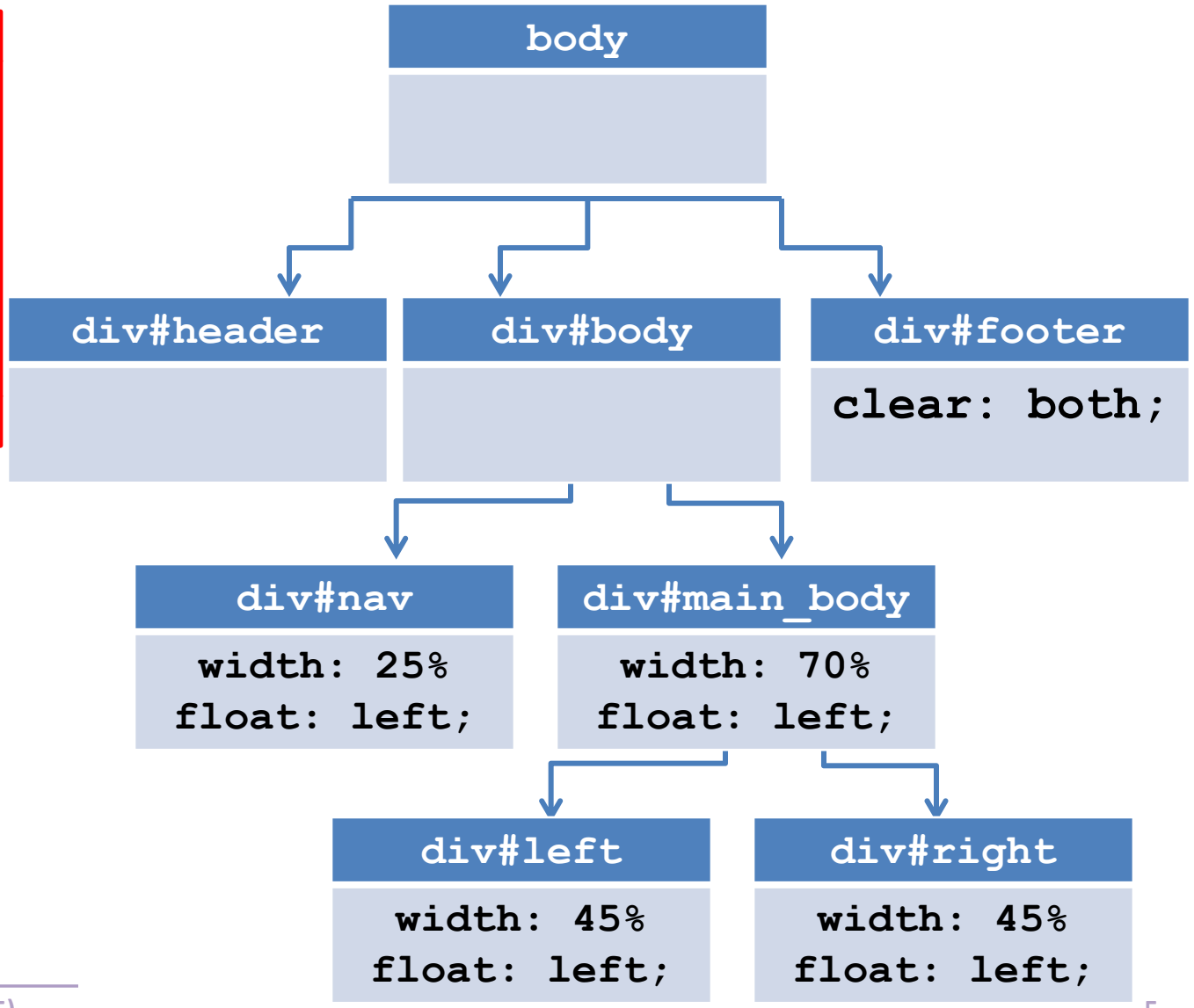
  +

  >

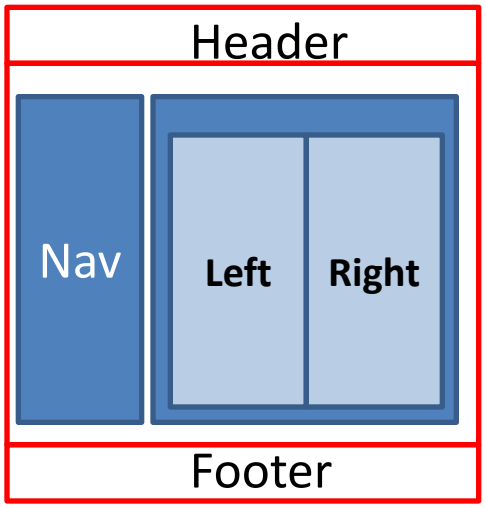
- Inheritances and specificity

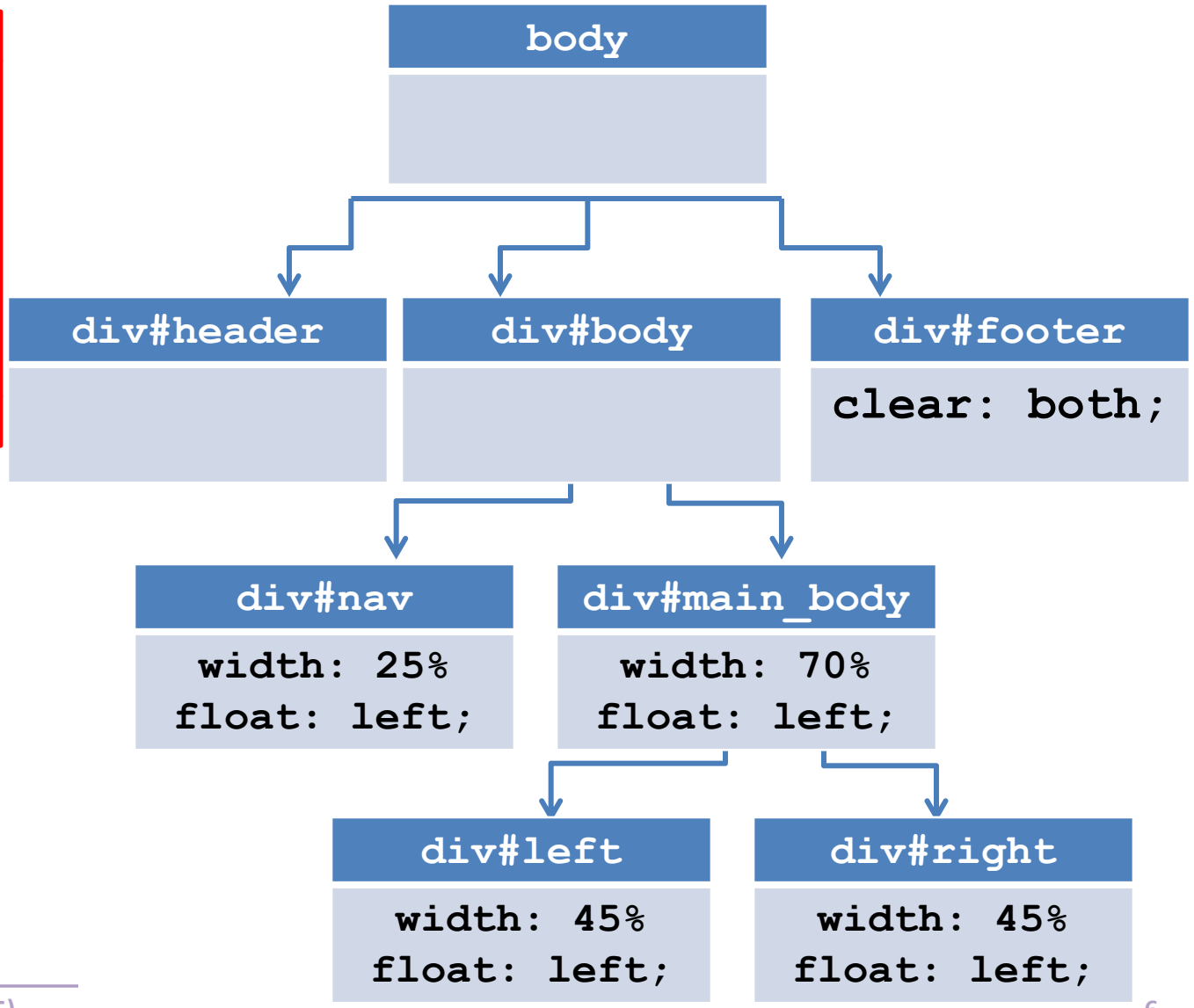
Tree diagram





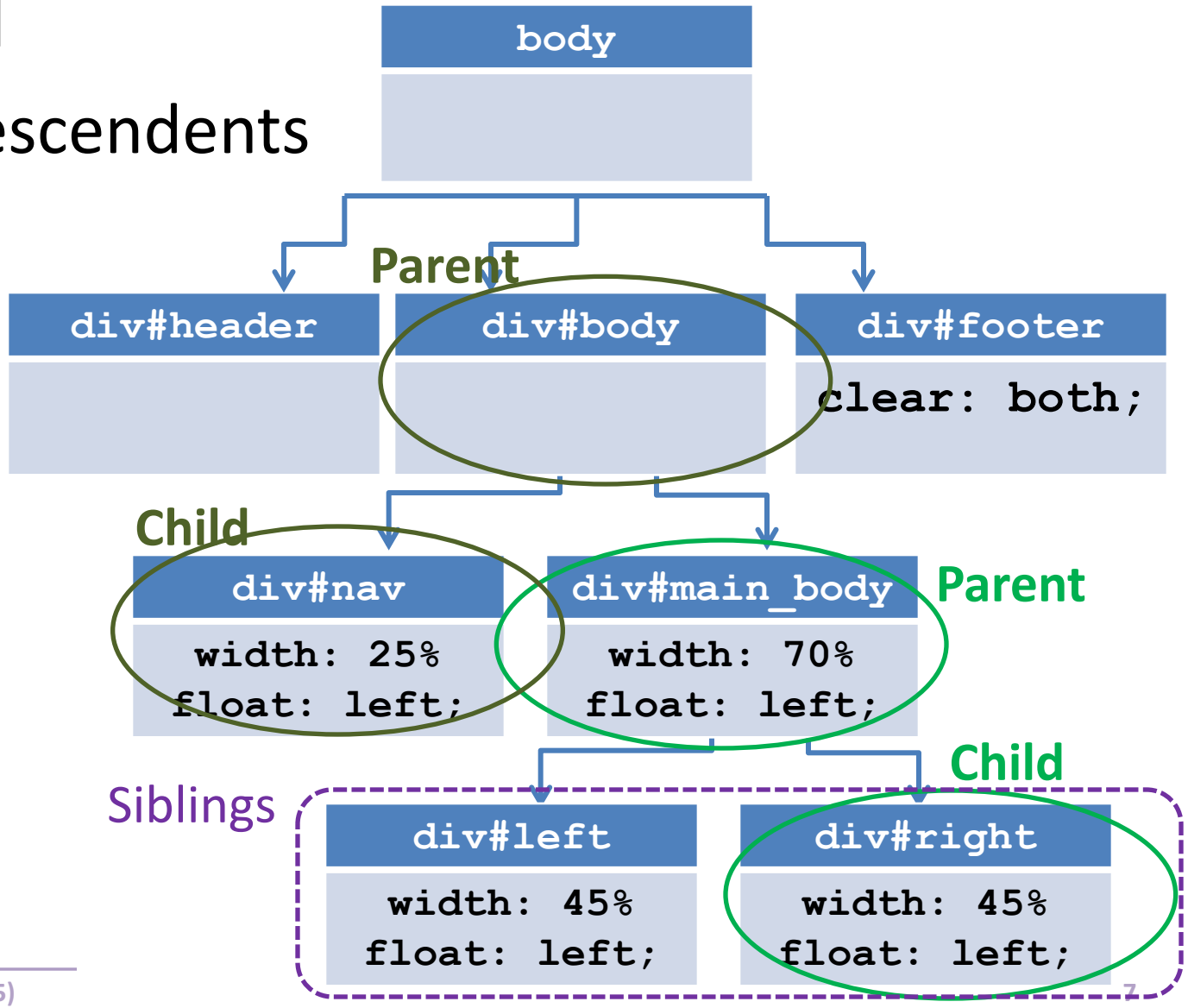
Tree diagram





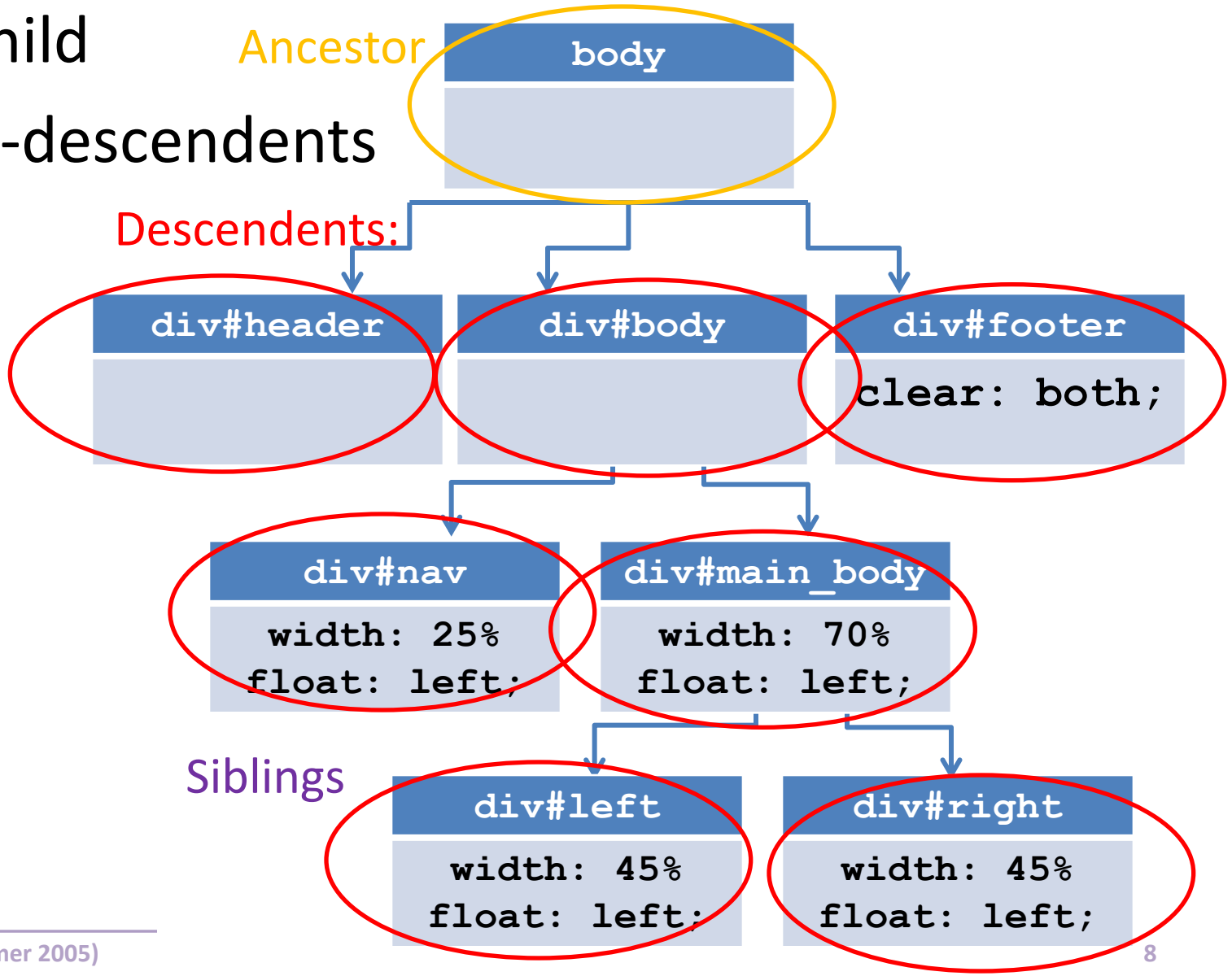
Examining effects of stylerules via relationships

- Parent-child
- Ancestor-descendents
- Siblings



Examining effects of stylerules via relationships

- Parent-child
- Ancestor-descendents
- Siblings



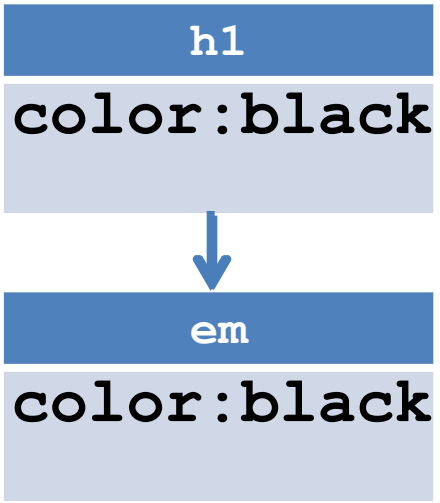
Style inheritance

A section of index.html

```
<h1>I <em>can</em> do it.</h1>
```

style.css

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



On browser window:

I *can*?do it.

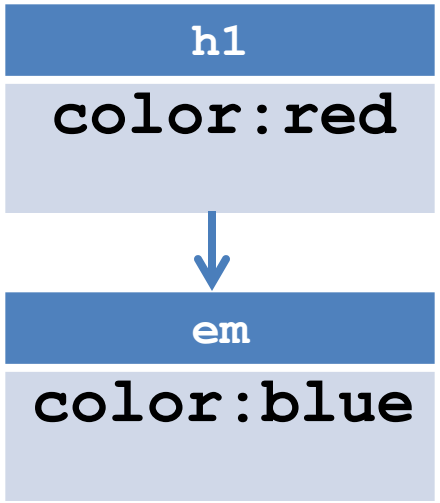
Style inheritance

A section of index.html

```
<h1>I <em>can</em> do it.</h1>
```

style.css

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



On browser window:

I *can* do it.

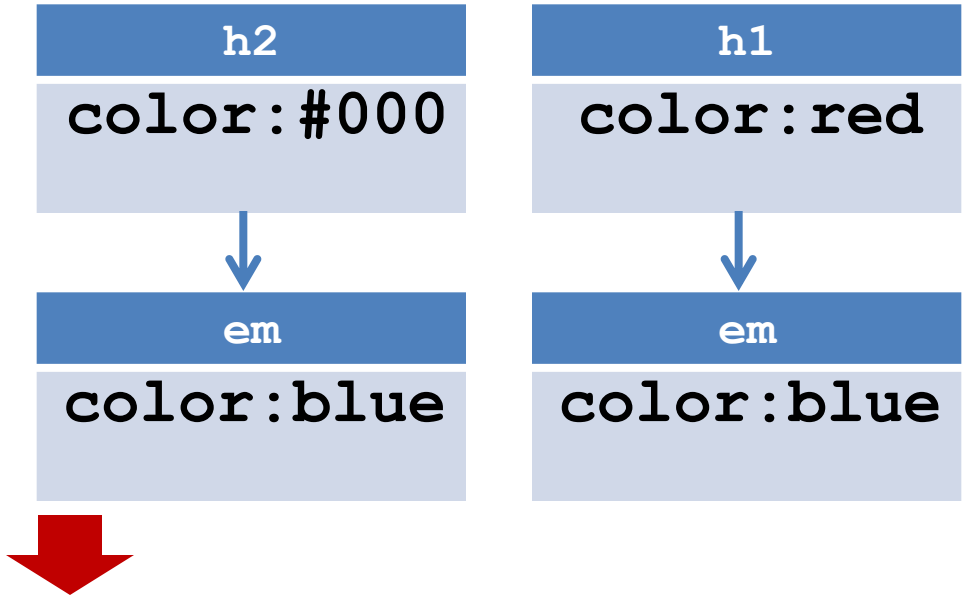
Style inheritance

A section of index.html

```
<h1>I <em>can</em> do it.</h1>
<h2>I <em>cannot</em> do it.</h2>
```

style.css

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



I *can* do it.
I *cannot* do it.

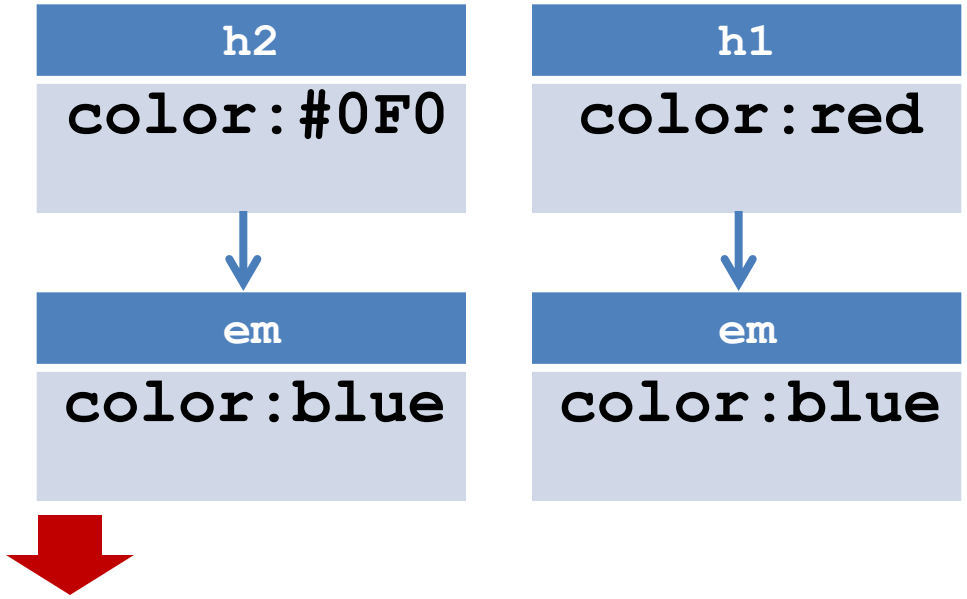
Style inheritance

A section of index.html

```
<h1>I <em>can</em> do it.</h1>
<h2>I <em>cannot</em> do it.</h2>
```

style.css

```
h1 {
        color: red;
}
em {
        color: blue;
}
h2 {
        color: #0F0;
}
```



I *can* do it.
I *cannot* do it.

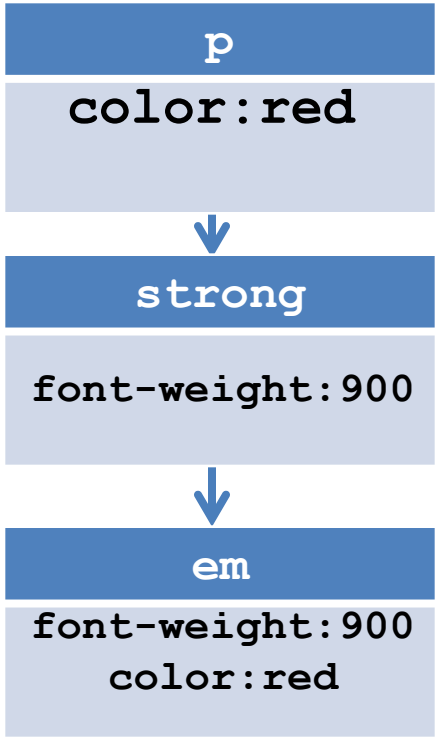
Style inheritance

A section of index.html

```
<p><strong>I <em>can</em> do it.</strong></p>
```

style.css

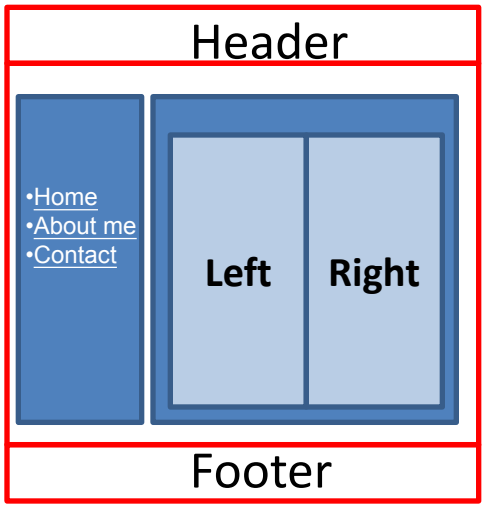
```
strong{
      font-weight: 900;
}
p{
      color: red;
}
```

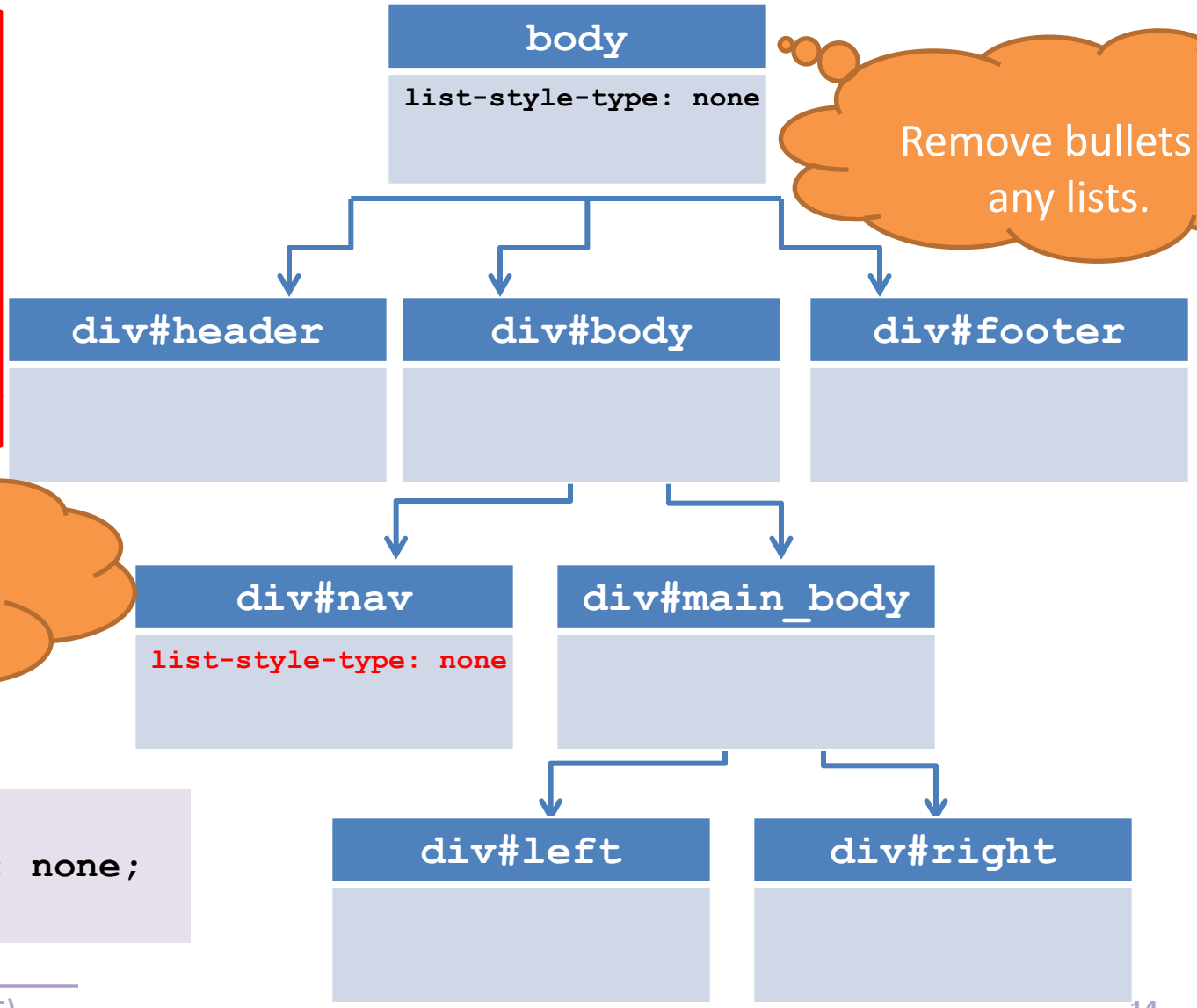


em inherits from all of its ancestors:

- **strong**
- **p**

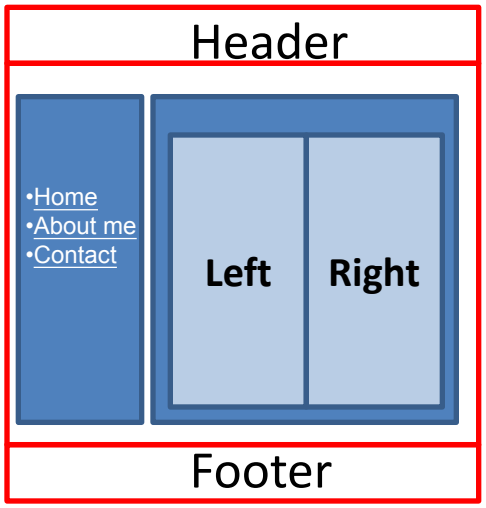
Contextual selector

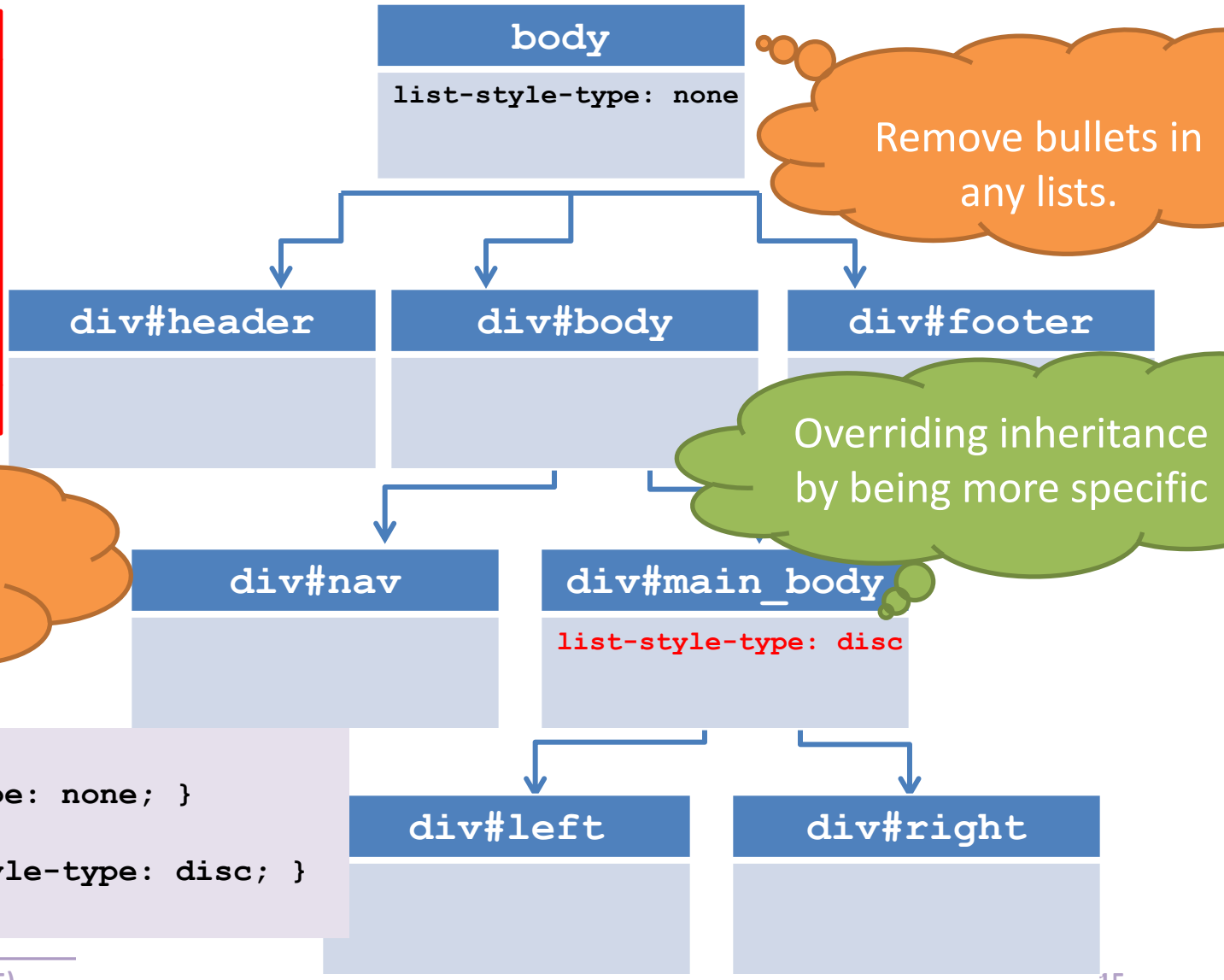





```
li {
  list-style-type: none;
}
```

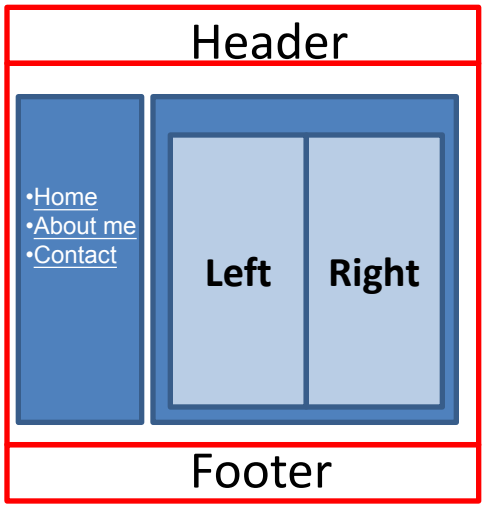
Option2: override with higher specificity

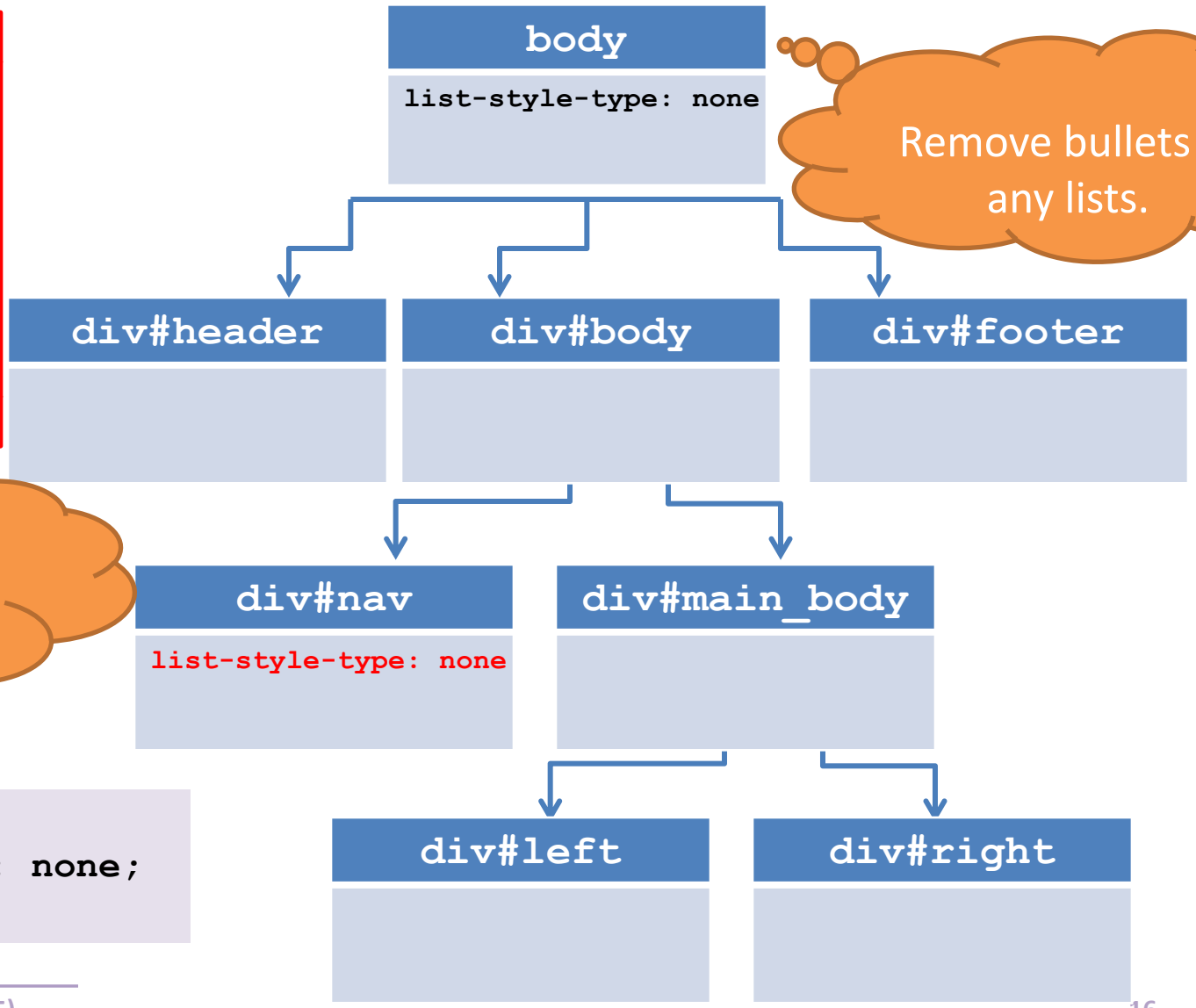





```
li{    list-style-type: none; }

#main_body{ list-style-type: disc; }
```

Option1: limit scope via contextual selector



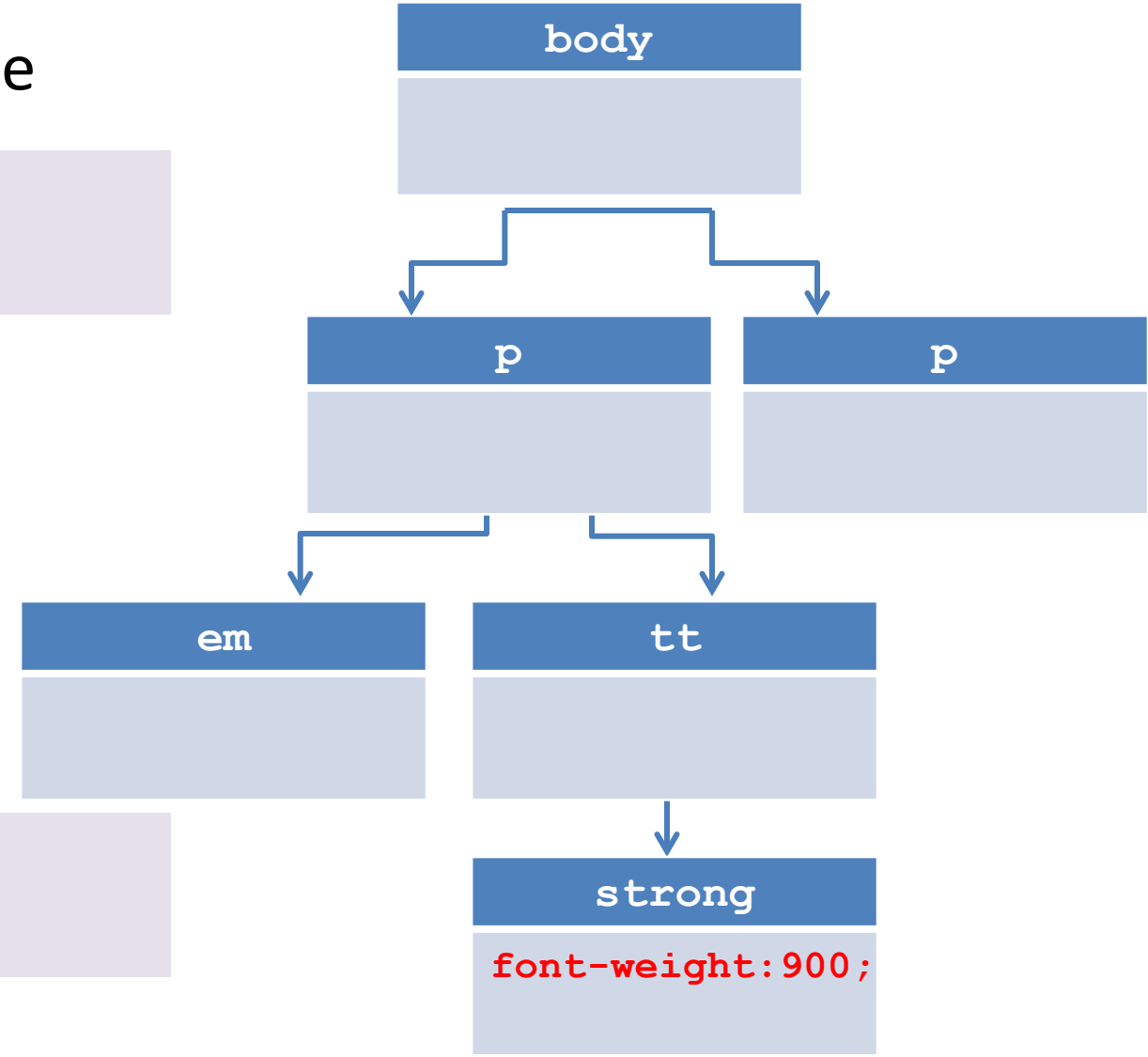



```
#nav li {
  list-style-type: none;
}
```

Contextual selectors

- Can be long-range

```
body p tt strong {
  font-weight: 900;
}
```



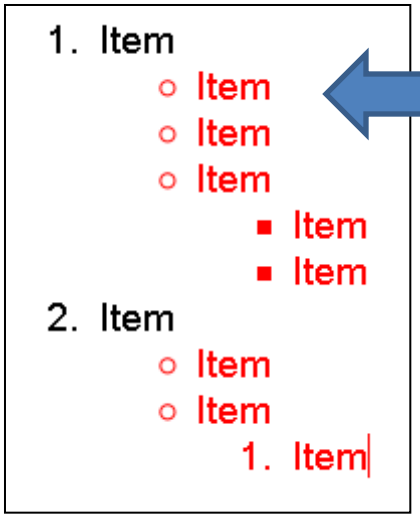
What about following?

```
body, p, tt, strong {
  font-weight: 900;
}
```

Contextual selectors

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

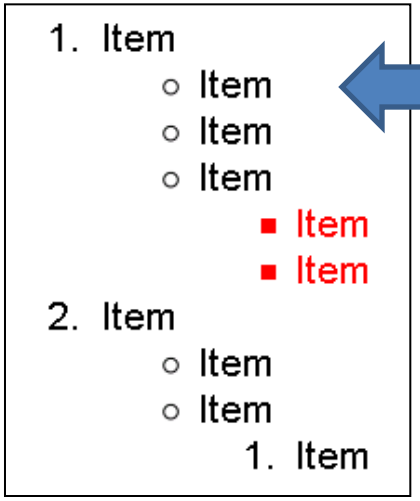
1. Item
   - Item
   - Item
   - Item
     - Item
     - Item
2. Item
   - Item
   - Item
     1. Item




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

This is equivalent to above:

```
ul …element… li {
  color: red;
}
```

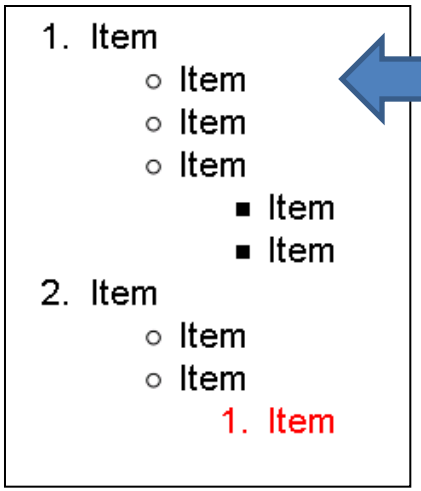



```
ul li ul {
  color: red;
}
```

Contextual selectors

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

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




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

This is equivalent to above:

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

Contextual selectors: immediate child

Represented by symbol >

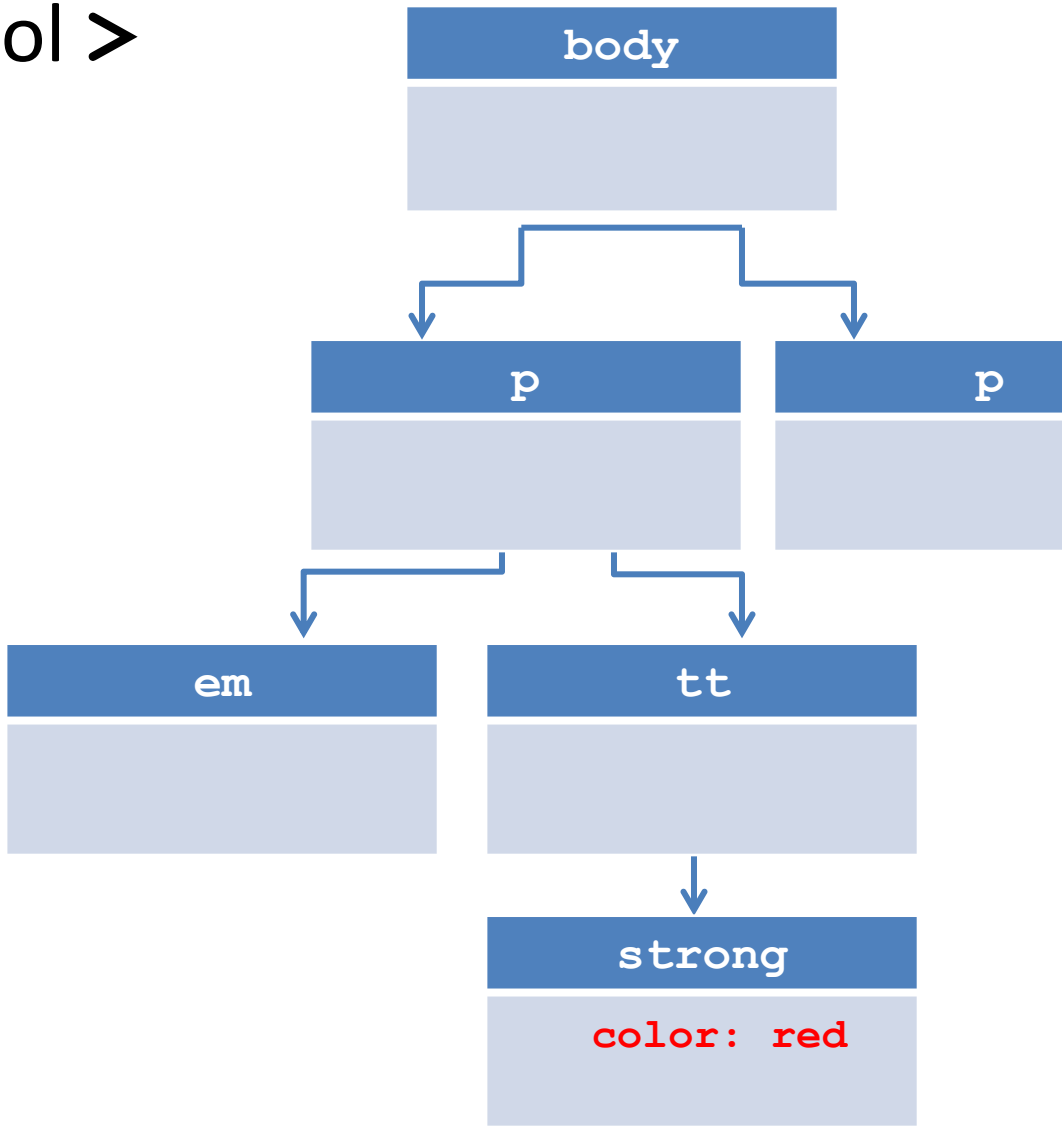
```
p strong {
   color: red;
}
```

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

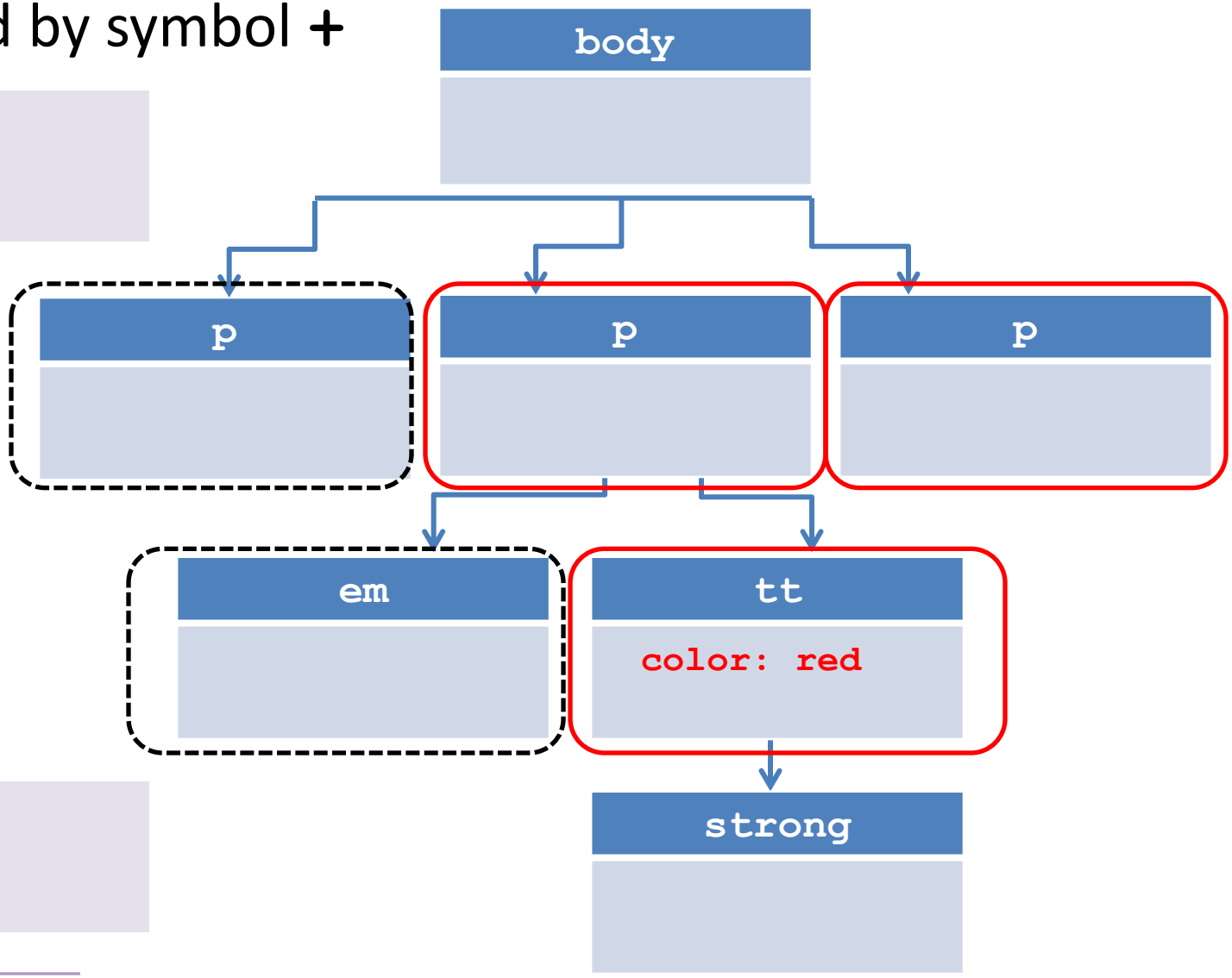
```
ul > ol {
  color: red;
}
```

?

Contextual selectors: next child

Represented by symbol **+**

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

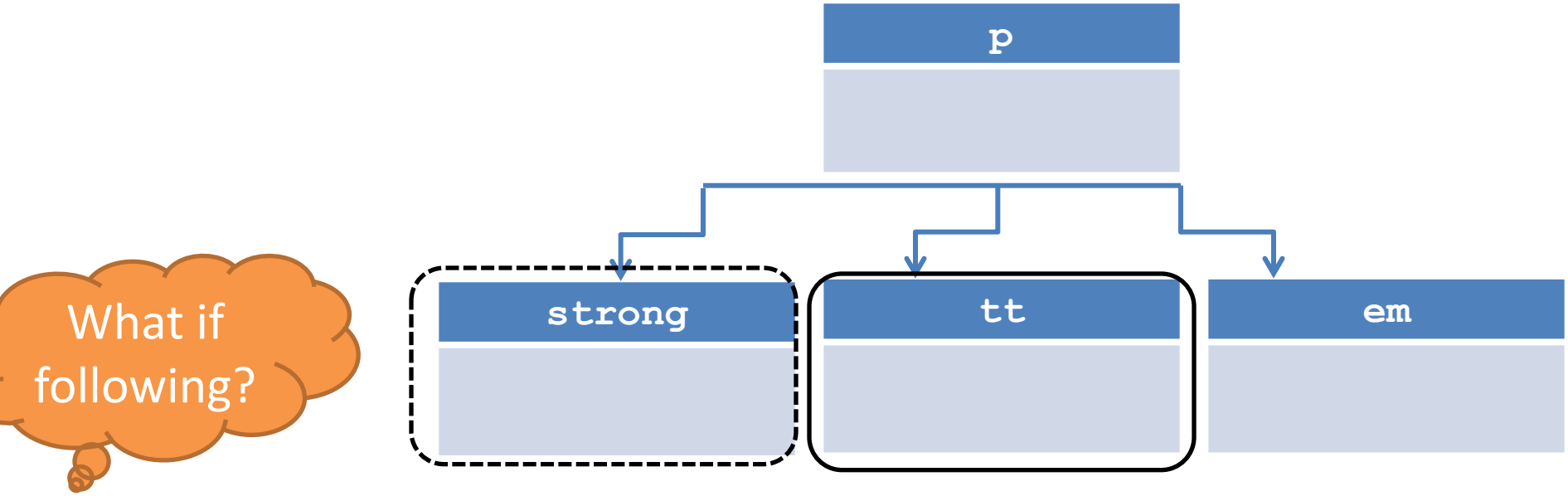


What if following?

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

Contextual selectors: next child

Represented by symbol **+**




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

CSS rules

Inheritance

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

-- CSS3 working draft specs

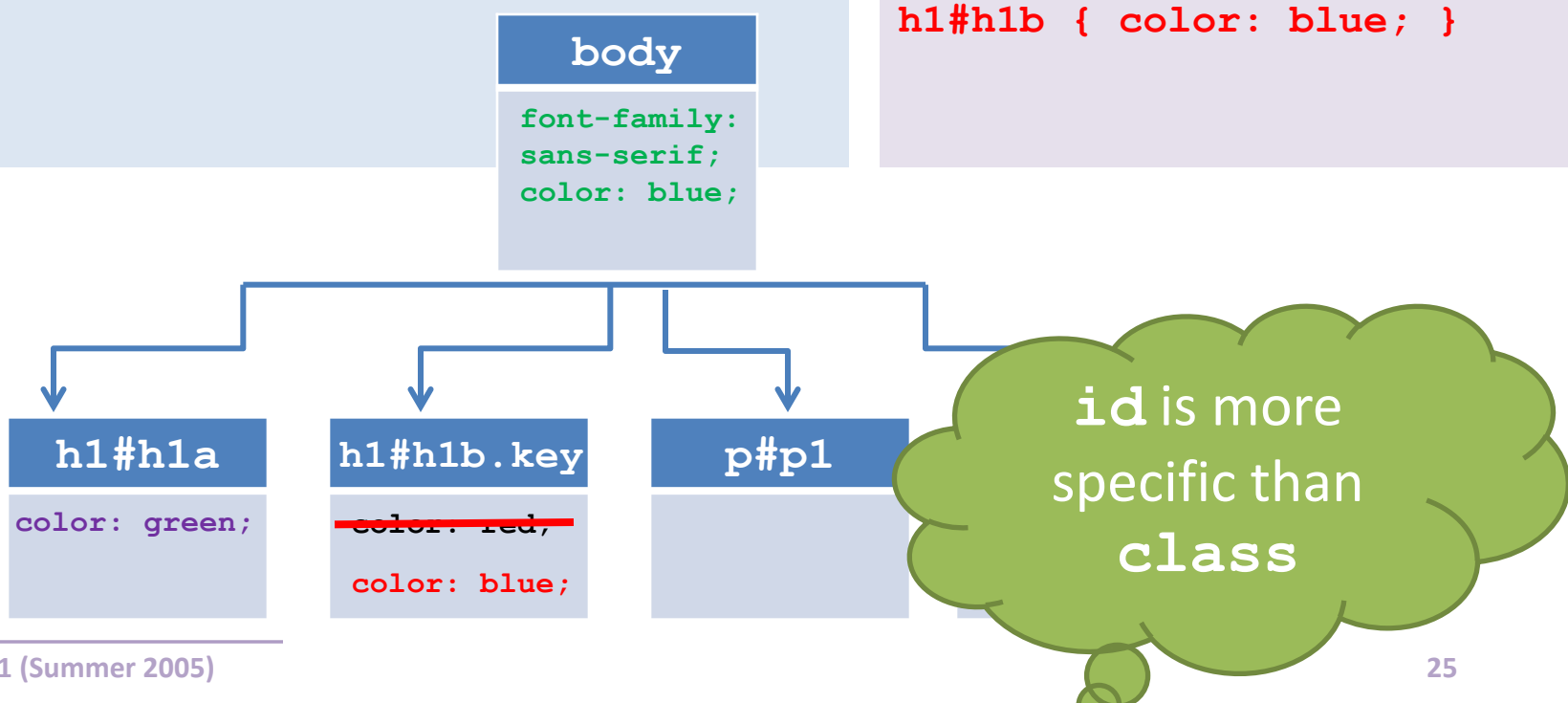
Specificity

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

Specificity of stylerules

```
<body>

    <h1 id="h1a">Heading A</h1>

    <h1 id="h1b" class="key">Heading B</h1>

    <p id="p1">This is a sentence.</p>

    <p id="p2">This is another sentence.</p>

</body>
```

```
body {
    font-family:sans-serif;
    color: blue;
    }

h1 { color: green; }
h1.key { color: red; }
h1#h1b { color: blue; }
```



Measuring specificity

Scoring system:

An id is worth 100 | Each class is worth 10 | Each element is worth 1

Selector	Id	Class	Element	Total
body	0	0	1	001
h1	0	0	1	001
h1.key	0	1	1	011
#h1b	1	0	0	100

```
body {
    font-family:sans-serif;
    color: blue;
    }
```

```
h1 { color: green; }
```

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

```
#h1b { color: blue; }
```

h1 with class "key" is more specific to:
h1 (001)
.key (010)

100 is larger than 011.
Last rule wins!

Measuring specificity

Selector	Id	Class	Element	Total
h2 strong	0	0	1+1	002

```
h2 strong
{
    font-family: serif;
}
h2 strong
{
    font-family: serif;
    font-size: 12pt;
} /*same specificity as above*/
```

rule

(depends on selectors, not number of declarations)

Measuring specificity

Selector	Id	Class	Element	Total
p em				
p.classA em				
#nav li				
#nav .red .key				
#main_body #left .key				
#main_body #left h1 tt em				
#nav, #body				
ol + ul				
table tr td ul > li				

Measuring specificity

Selector	Id	Class	Element	Total
p em	0	0	1+1	002
p.key em	0	1	1+1	012
#nav li	1	0	1	101
#nav .red .key	1	1+1	0	120
#main_body #left .key	1+1	1	0	210
#main_body #left h1 tt em	1+1	0	1+1+1	203
#nav, #body	1	0	0	100
ol + ul	0	0	1+1	002
table tr td ul > li	0	0	1+1+1+1+1	005

Measuring specificity

Selector		Class	Element	Total
p em		0	1+1	002
p.key em			1+1	012
#nav li	1	0	1	101
#nav .red .key	1	1+1	0	120
#main_body			0	210
#main_body			1+1+1	203
#nav, #body	1	0	0	100
ol + ul	0	0	1+1	002
table tr td ul > li	0	0	1+1+1+1+1	005

Why? Because `.classA` can be standalone, p.classA refers to 2 levels of specificity→ 011

Why? There are just 2 levels of specificity→ 101

Recall: Commas are used to separate independent selectors that have been grouped together.

Questions?