

css

**INHERITANCE**

**Let's start  
with the  
document  
tree**

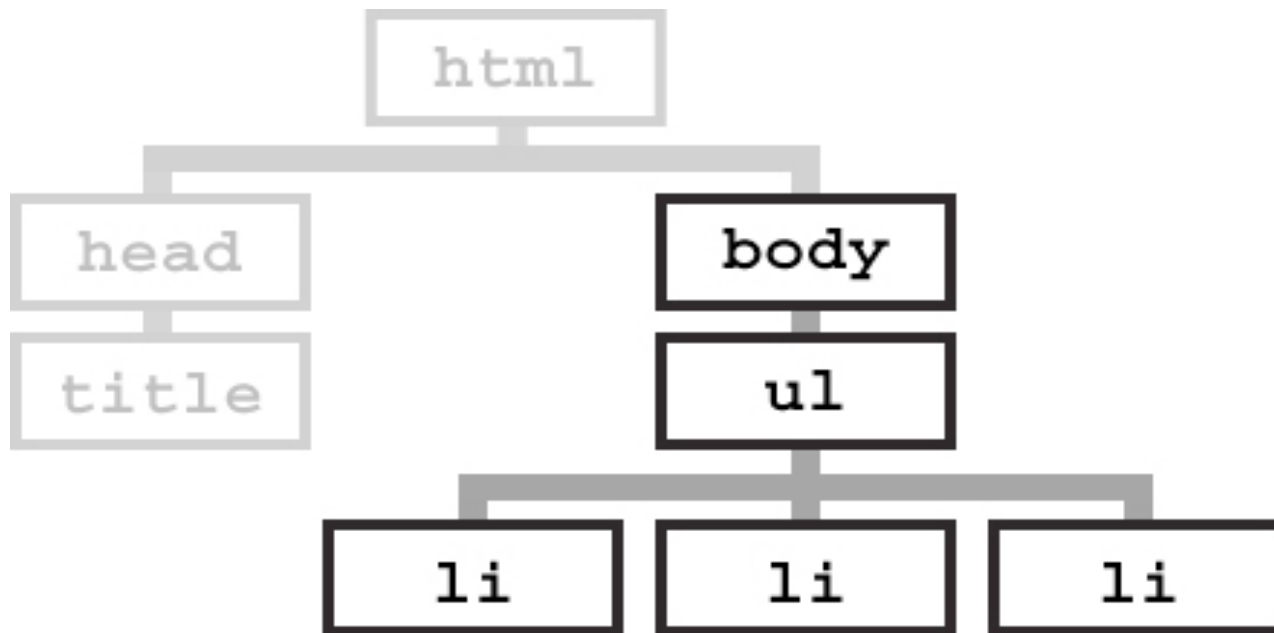
Before we explore inheritance, we  
need to understand the  
**document tree.**



All HTML documents are **trees**.



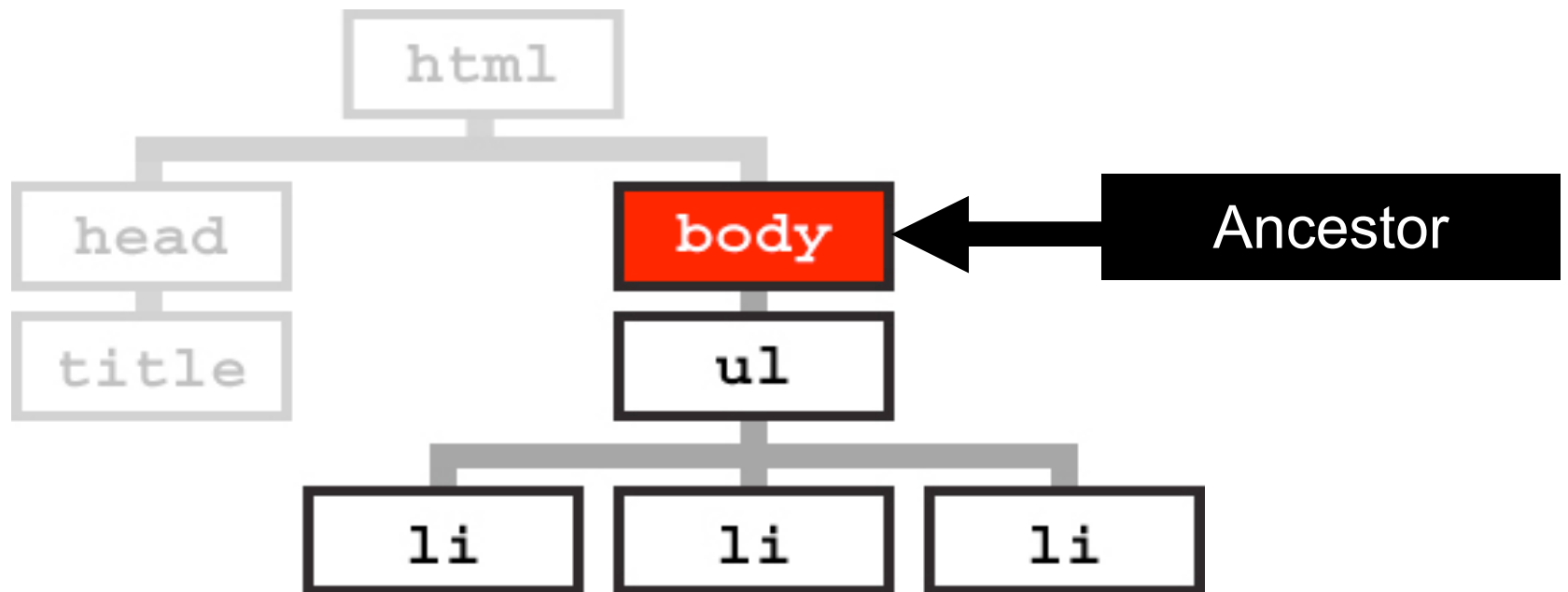
Document trees are made from  
**HTML elements.**



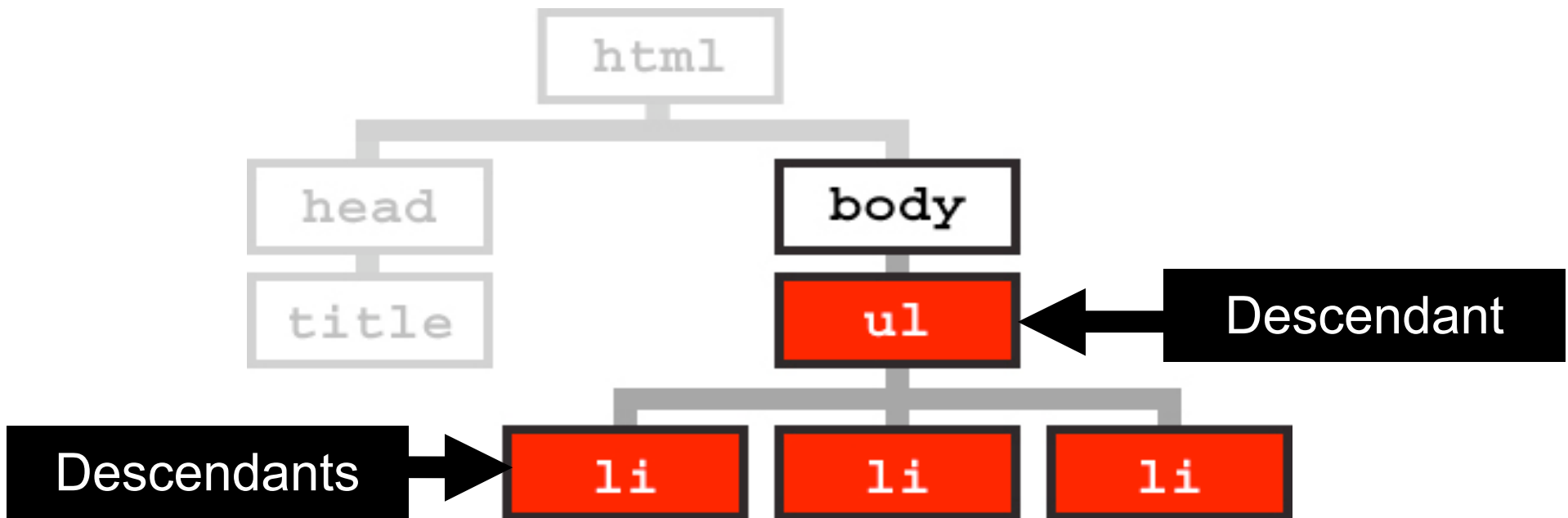
The document tree is just like  
your **family tree**.



An **ancestor** refers to any element that is connected but further up the document tree.

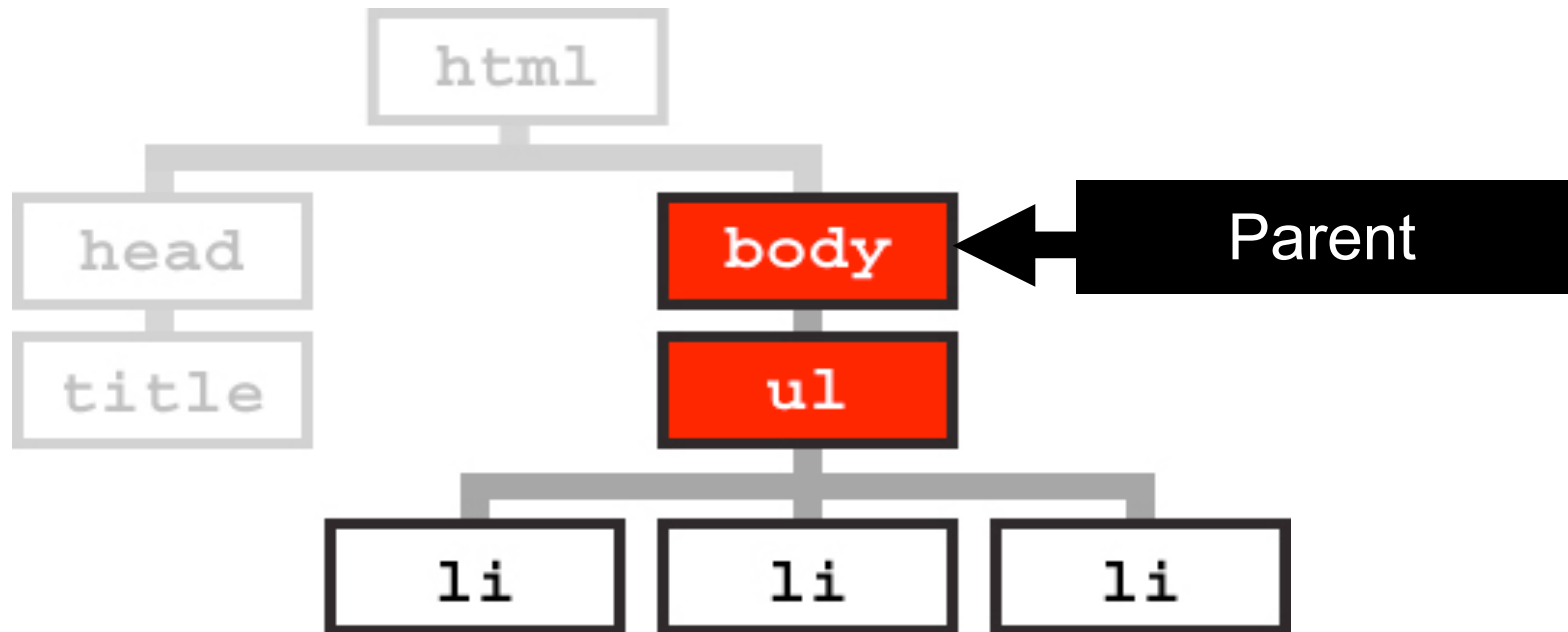


A **descendant** refers to any element that is connected but lower down the document tree.

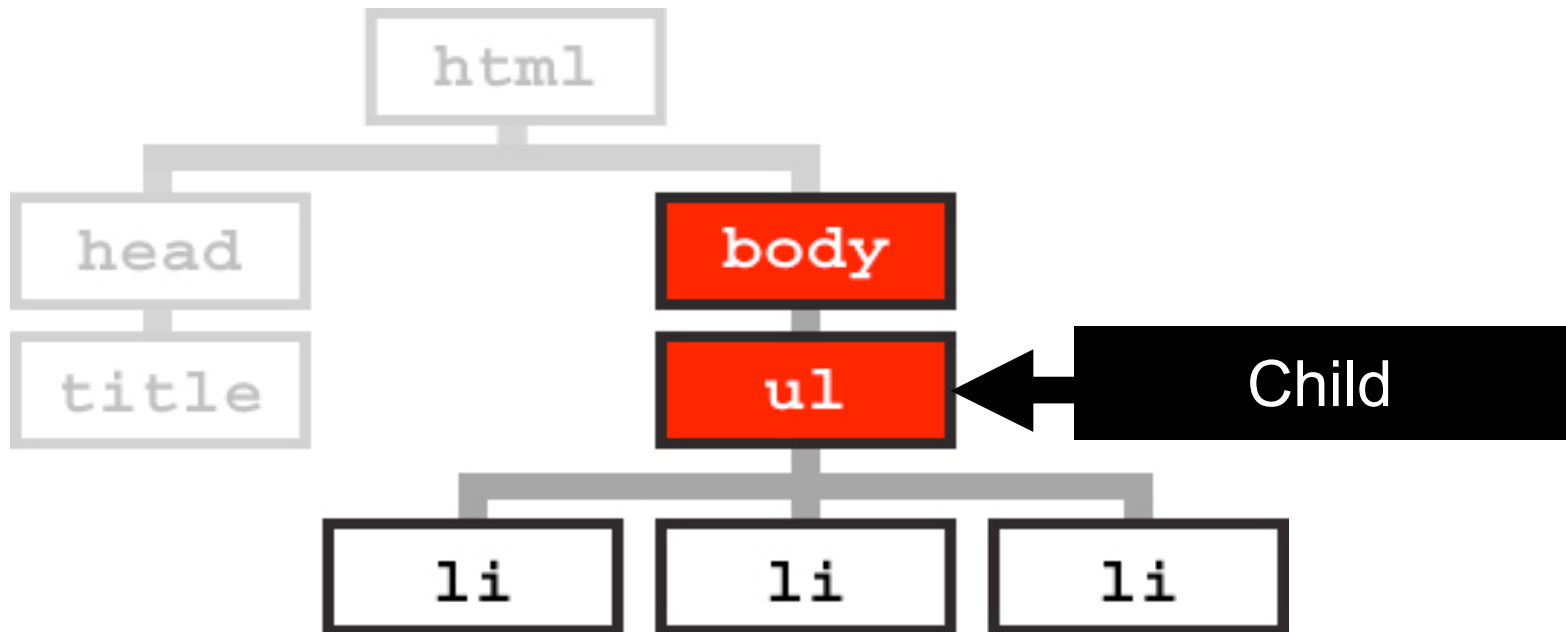




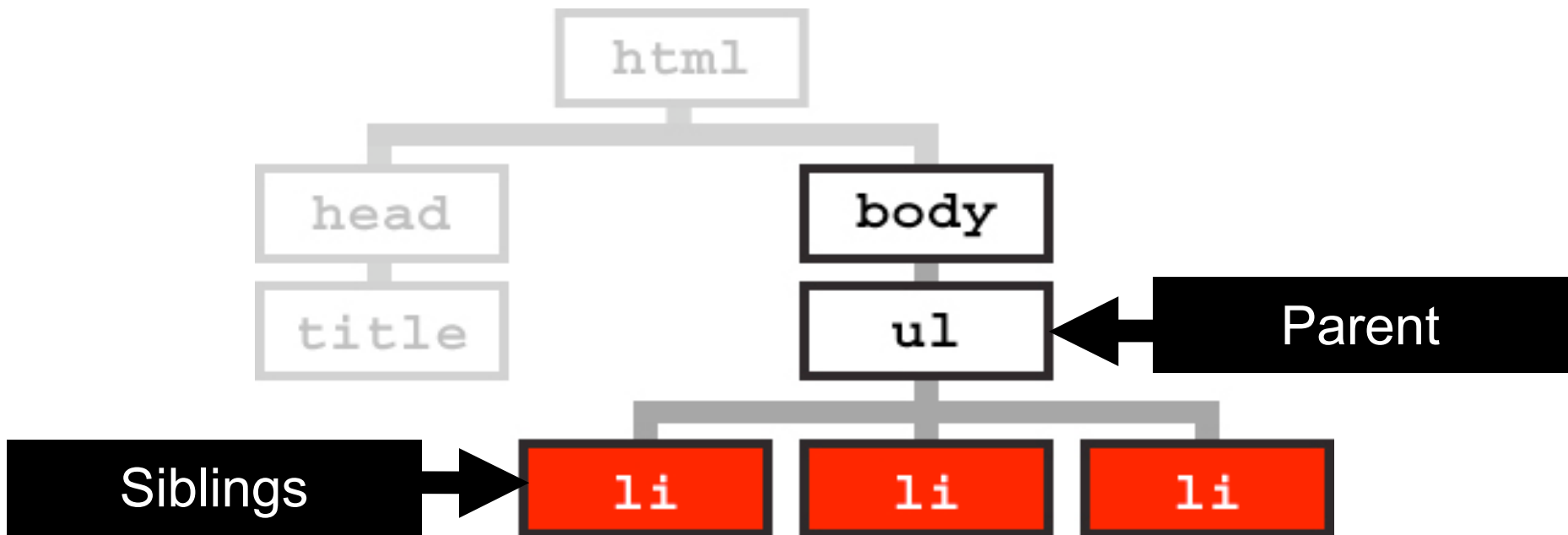
A **parent** is an element that is connected & directly above an element in the document tree.



A **child** is an element that is connected & directly below an element in the document tree.



A **sibling** is an element that shares the same parent as another element.



**Next, a bit  
about  
CSS rules**

We also need to understand the  
**basics of CSS rules** before  
exploring inheritance.



CSS rules tell browsers  
**how to render** specific elements  
on an HTML page.



CSS rules are made up of  
**five components.**



The **selector** "selects" the elements on an HTML page that are affected by the rule.

A diagram showing a CSS rule inside a window-like frame. The rule is 'p { color: red; }'. The 'p' is highlighted with a yellow background. A red arrow points from a black box labeled 'Selector' to the 'p'.

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

**Selector**



The **declaration block** is a container that consists of anything between (and including) the brackets.

A diagram illustrating a CSS declaration block. It shows a code editor window with a line of CSS code: 'p { color: red; }'. The text '{ color: red; }' is highlighted in yellow. A red arrow points from a black box labeled 'Declaration block' to the highlighted text. The code editor window has a grey title bar and a vertical scrollbar on the right side.

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

**Declaration block**

The **declaration** tells a browser how to render any element on a page that is selected.

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

**Declaration**

The **property** is the aspect of that element that you are choosing to style.

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



Property

The **value** is the exact style you wish to set for the property.

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



A diagram illustrating the concept of a 'value' in CSS. It shows a code editor window with the CSS rule `p { color: red; }`. The word `red` is highlighted with a yellow background. A red arrow points from a black box labeled 'Value' below to the highlighted word `red`.

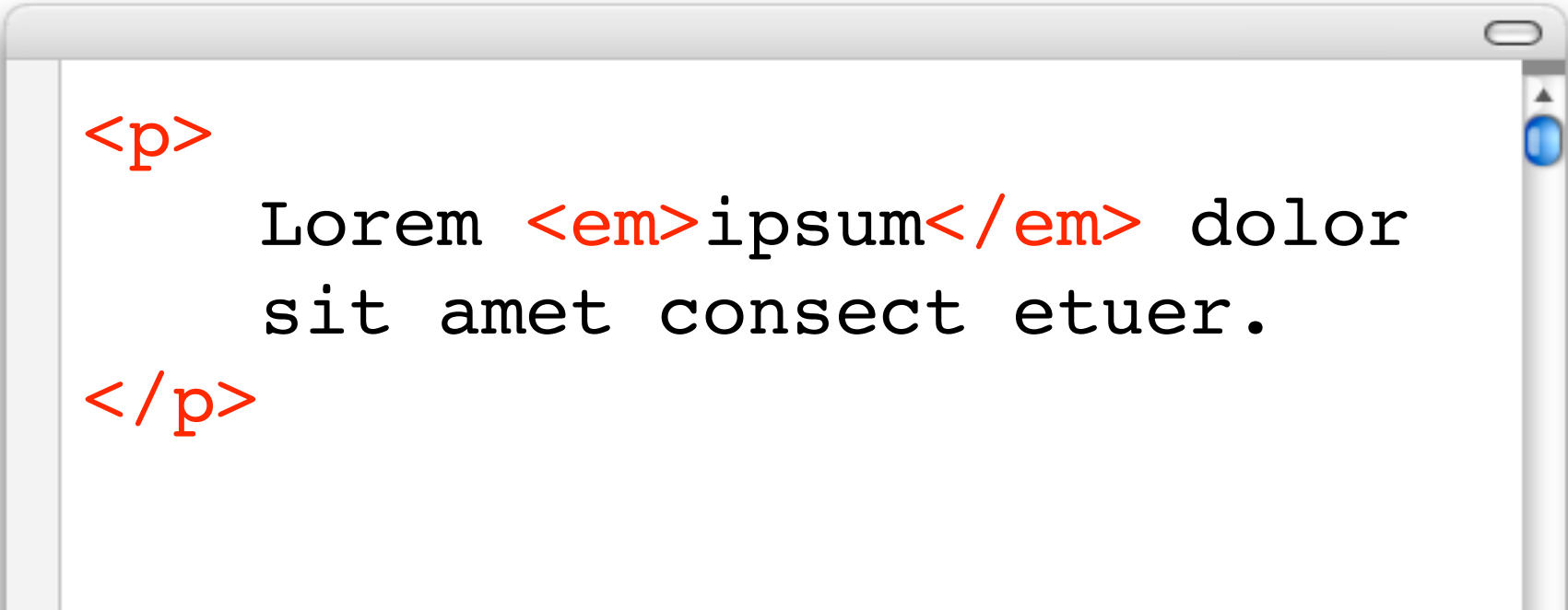
Value

**Now...**  
**what is**  
**inheritance?**

Inheritance is where specific CSS properties are **passed down** to descendant elements.

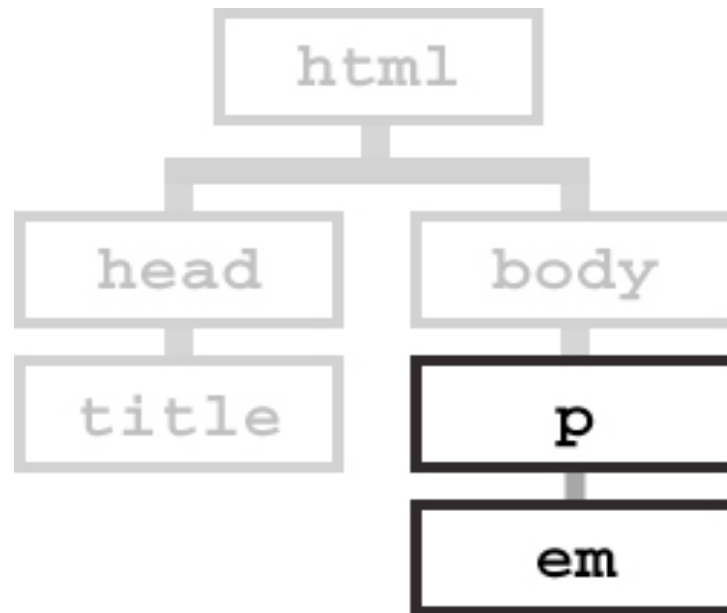


To see inheritance in action, we will use the **HTML code** below:

A screenshot of a code editor window with a light gray border and a white background. The window contains HTML code. The opening paragraph tag <p> is on the first line. The second line contains the text 'Lorem' followed by an italicized 'ipsum' (indicated by the <em> and </em> tags) and the text 'dolor sit amet consectetur.' The closing paragraph tag </p> is on the third line. The code is color-coded: the tags are red and the text is black.

```
<p>  
    Lorem <em>ipsum</em> dolor  
    sit amet consectetur.  
</p>
```

Note that the `<em>` element **sits inside** the `<p>` element.





We will also use this CSS code.  
Note that the `<em>` element has  
**not been specified.**

A screenshot of a code editor window with a light gray border and a white background. The code is written in a monospaced font. The text 'p {' is in black, 'color:' is in red, 'red;' is in red, and '}' is in black. There is a vertical scrollbar on the right side of the editor.

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

In a browser, the `<p>` and `<em>` elements will **both be colored red.**



But why is the `<em>` element  
**colored red** when it has not been  
styled?



Because the `<em>` element has **inherited** the color property from the `<p>` element.

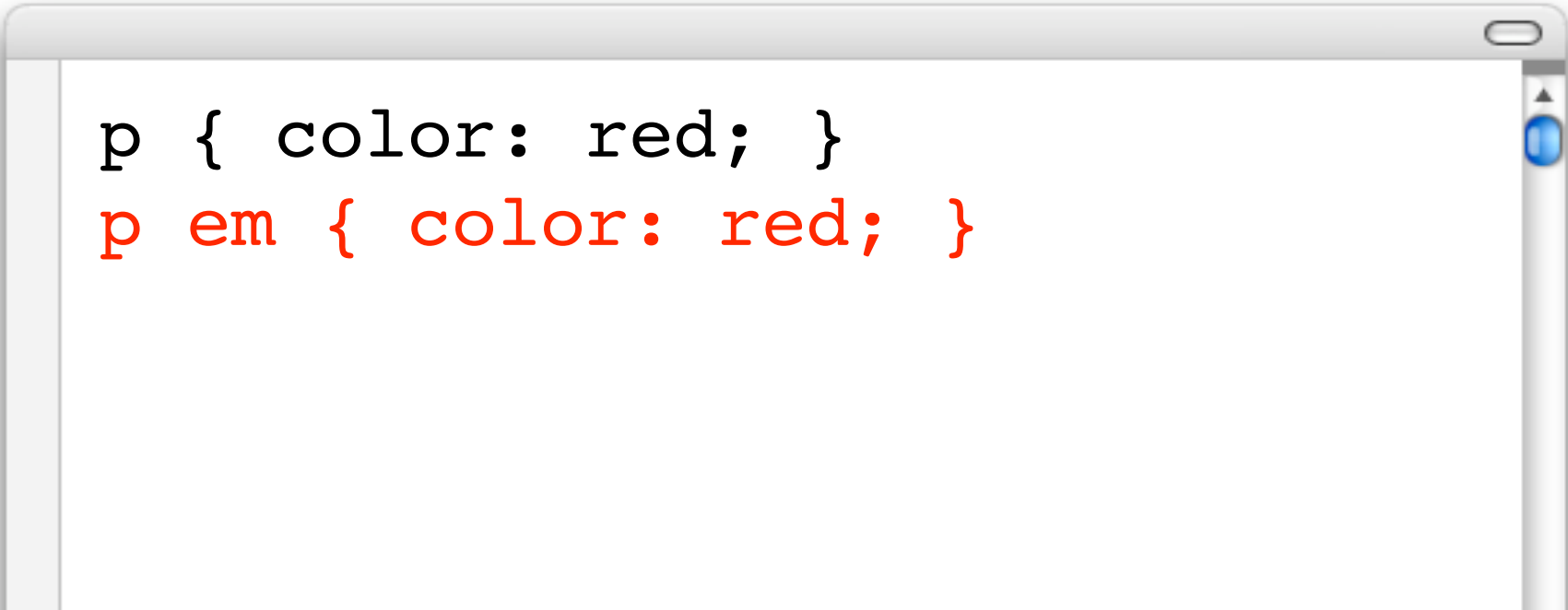


**Why is  
inheritance  
helpful?**

Inheritance is designed to **make  
it easier** for authors.

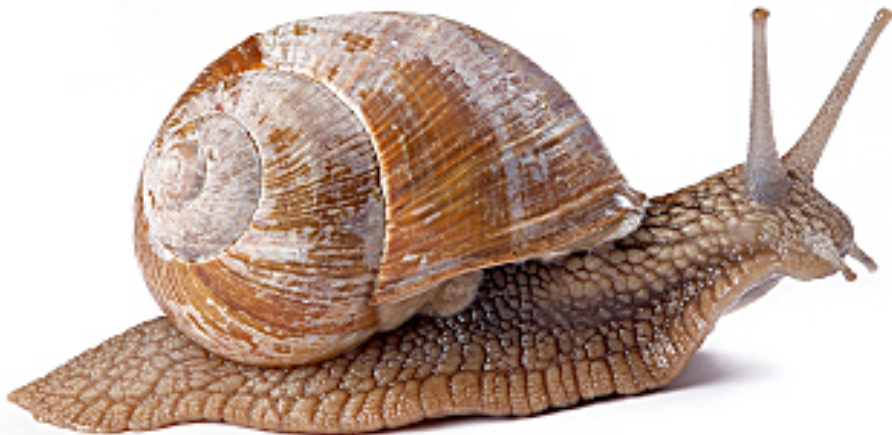


Otherwise we would need to specify properties for **all descendant elements**.



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

CSS files would be **much larger in size**, harder to create and maintain as well as slower to download.





**Are all CSS  
properties  
inherited?**

No. All CSS properties are  
**not inherited!**



If every CSS property was  
inherited, it would make things  
**much harder** for authors.



Authors would have to **turn off**  
unwanted CSS properties for  
descendant elements.



For example, what would happen  
if the **border property was**  
**inherited** by default...



and we then **applied a border** to  
the `<p>` element?

A screenshot of a code editor window with a light gray title bar and a vertical scrollbar on the right. The code is written in a monospaced font, with the property name and opening brace in black and the values in red.

```
p { border: 1px solid red; }
```

The `<em>` inside the `<p>` would also have a **red border**.



Luckily, borders are **not inherited**, so the `<em>` would not have a red border.





Generally speaking, only  
properties that **make our job  
easier** are inherited!



**So, which  
properties are  
inherited?**

The following CSS properties **are**  
**inherited...**



azimuth, border-collapse, border-spacing,  
caption-side, color, cursor, direction, elevation,  
empty-cells, font-family, font-size, font-style,  
font-variant, font-weight, font, letter-spacing,  
line-height, list-style-image, list-style-position,  
list-style-type, list-style, orphans, pitch-range,  
pitch, quotes, richness, speak-header, speak-  
numeral, speak-punctuation, speak, speech-  
rate, stress, text-align, text-indent, text-  
transform, visibility, voice-family, volume, white-  
space, widows, word-spacing

Yikes! That is a **lot of properties.**



To simplify things, let's take a look  
at some of the  
**key groups** of properties.



**Text-related properties** that are  
inherited:



azimuth, border-collapse, border-spacing,  
caption-side, color, cursor, direction, elevation,  
empty-cells, font-family, font-size, font-style,  
font-variant, font-weight, font, letter-spacing,  
line-height, list-style-image, list-style-position,  
list-style-type, list-style, orphans, pitch-range,  
pitch, quotes, richness, speak-header, speak-  
numeral, speak-punctuation, speak, speech-  
rate, stress, text-align, text-indent, text-  
transform, visibility, voice-family, volume, white-  
space, widows, word-spacing



**List-related properties** that are  
inherited:



azimuth, border-collapse, border-spacing,  
caption-side, color, cursor, direction, elevation,  
empty-cells, font-family, font-size, font-style,  
font-variant, font-weight, font, letter-spacing,  
line-height, **list-style-image, list-style-position,**  
**list-style-type, list-style,** orphans, pitch-range,  
pitch, quotes, richness, speak-header, speak-  
numeral, speak-punctuation, speak, speech-  
rate, stress, text-align, text-indent, text-  
transform, visibility, voice-family, volume, white-  
space, widows, word-spacing

And, importantly, the  
**color property** is inherited:



azimuth, border-collapse, border-spacing,  
caption-side, **color**, cursor, direction, elevation,  
empty-cells, font-family, font-size, font-style,  
font-variant, font-weight, font, letter-spacing,  
line-height, list-style-image, list-style-position,  
list-style-type, list-style, orphans, pitch-range,  
pitch, quotes, richness, speak-header, speak-  
numeral, speak-punctuation, speak, speech-  
rate, stress, text-align, text-indent, text-  
transform, visibility, voice-family, volume, white-  
space, widows, word-spacing

**Is font-size  
inherited?**

The simple answer is “yes”.  
However, font-size is inherited in  
**a different way** to many other  
properties.



Rather than the actual value  
being inherited, the **calculated  
value** is inherited.



Before explaining how font-size inheritance works, we need to look at **why font-size is not directly inherited.**





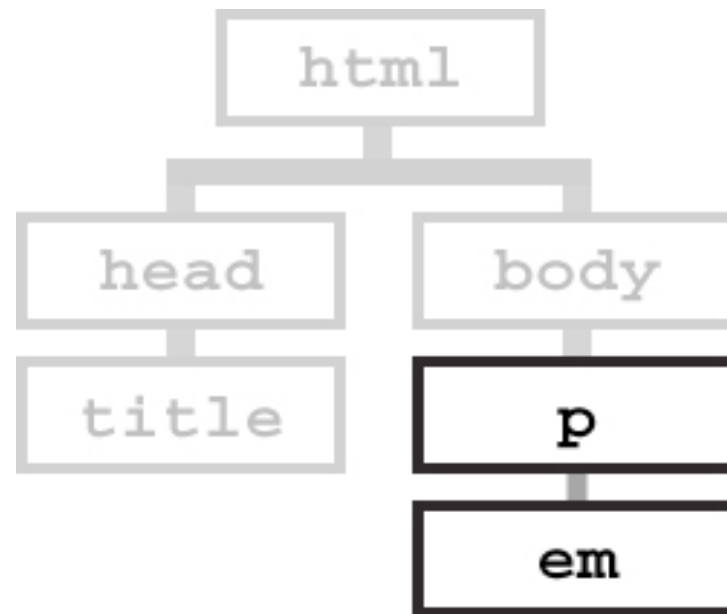
Let's start with the  
**same sample** of HTML code we  
used earlier:

```
<p>
```

```
    Lorem <em>ipsum</em> dolor  
    sit amet consectetur.
```

```
</p>
```

As before the `<em>`  
**sits inside** the `<p>`.



Now, a font-size is applied to the **<p> element only**. The <em> has not been specified.

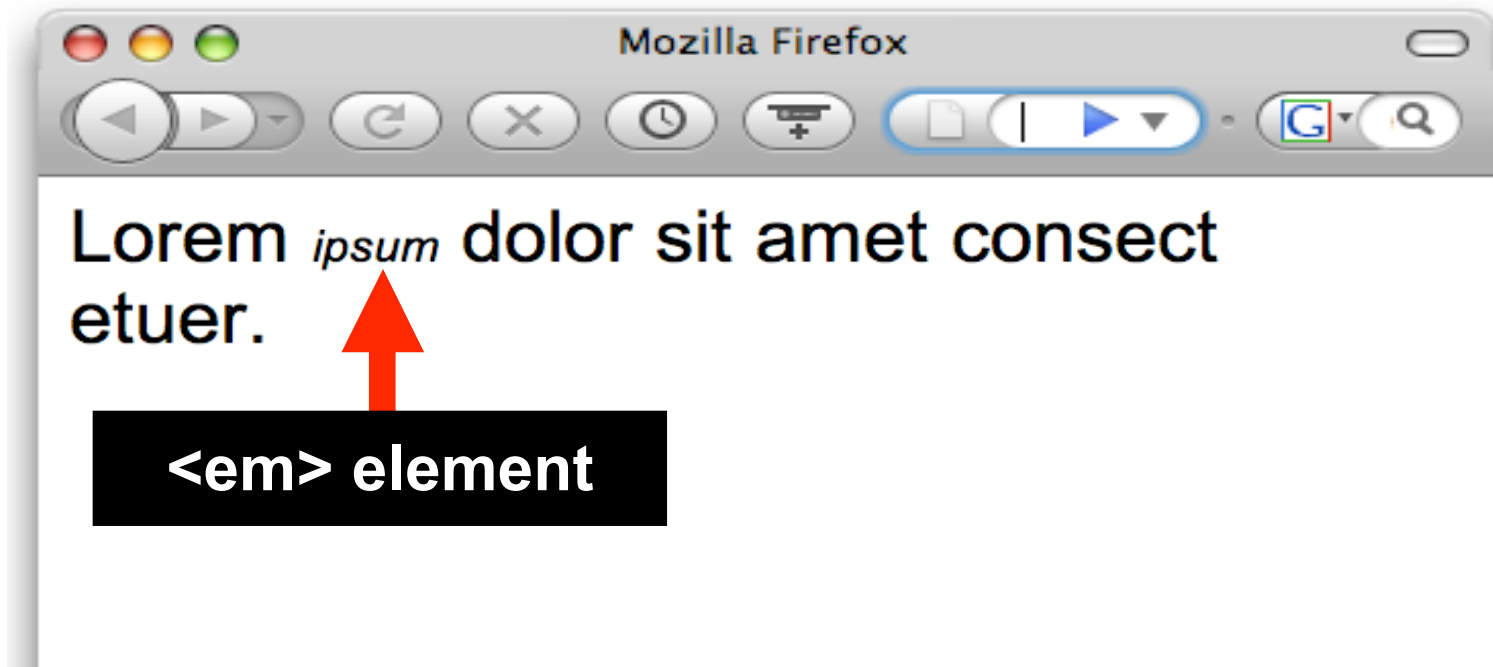
A screenshot of a code editor window with a light gray border and a white background. The code is written in a monospaced font. The text 'p {' is in black, 'font-size:' is in red, '80%;' is in red, and '}' is in black. There is a vertical scrollbar on the right side of the editor.

```
p { font-size: 80%; }
```

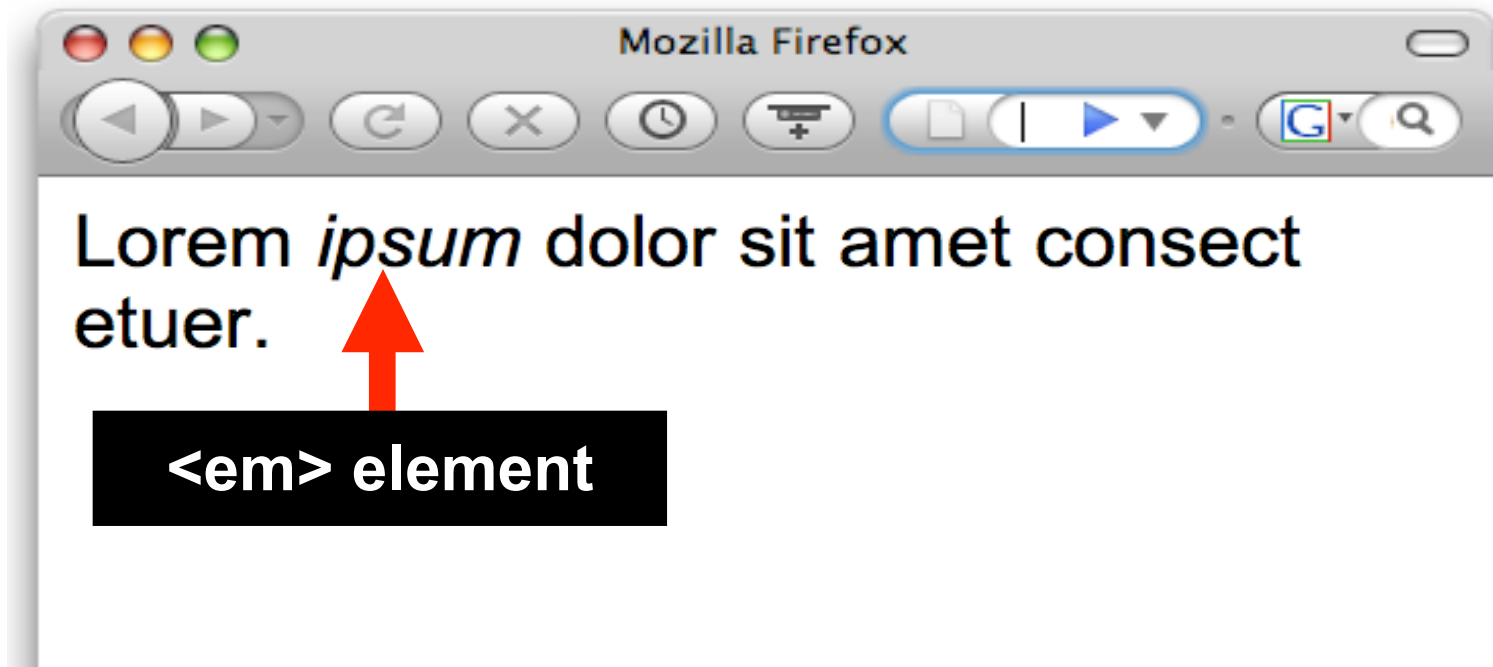
If the font-size value of 80% were  
to be inherited, the  
<em> would be sized to **80%**  
of the <p> element...



and the rendered document  
would **look like this**:



However, **this is not the case!**  
The `<em>` is the same size as the  
`<p>`.



So how does inheritance work for  
**font-size**?



Let's look at **three examples**  
in action.





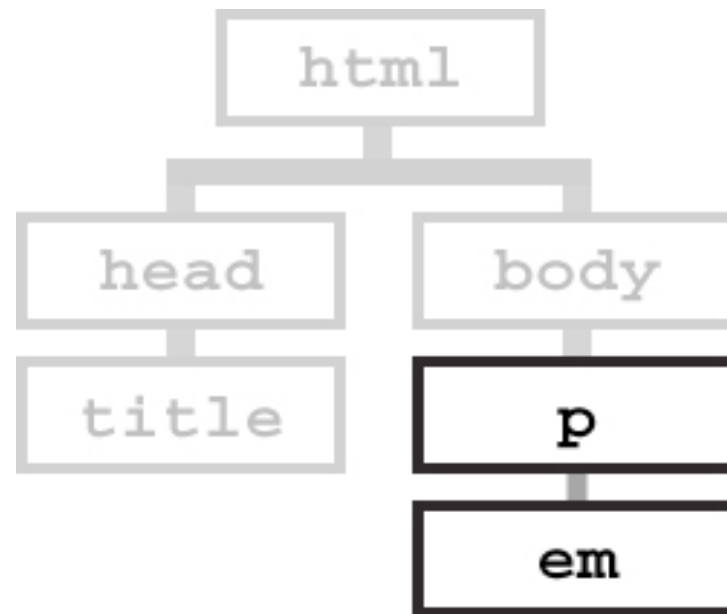
We will use the **same HTML code** as before:

```
<p>
```

```
  Lorem <em>ipsum</em> dolor  
  sit amet consectetur.
```

```
</p>
```

Which produces the same **document tree** as before.

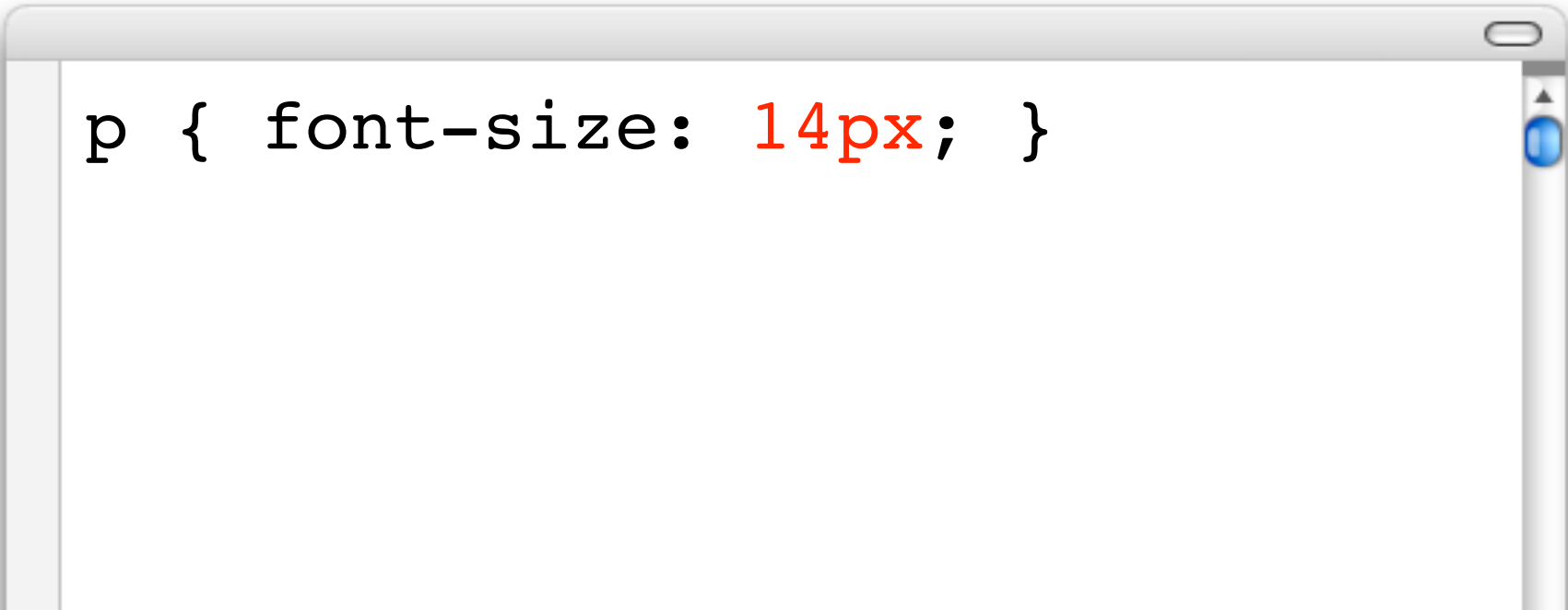


# Example 1:

## Pixels

# The <p> element has been given a font-size of **14px**.

Note: pixels are not recommended for sizing fonts due to accessibility issues associated with older browsers such as IE5 and IE6.

A screenshot of a code editor window with a light gray border and a white background. The code is written in a monospaced font. The text 'p { font-size: 14px; }' is displayed, with '14px' highlighted in red. The editor has a vertical scrollbar on the right side.

```
p { font-size: 14px; }
```

This pixel value (14px) overrides the browsers default font-size value (approx 16px). **This new value is inherited by descendants.**

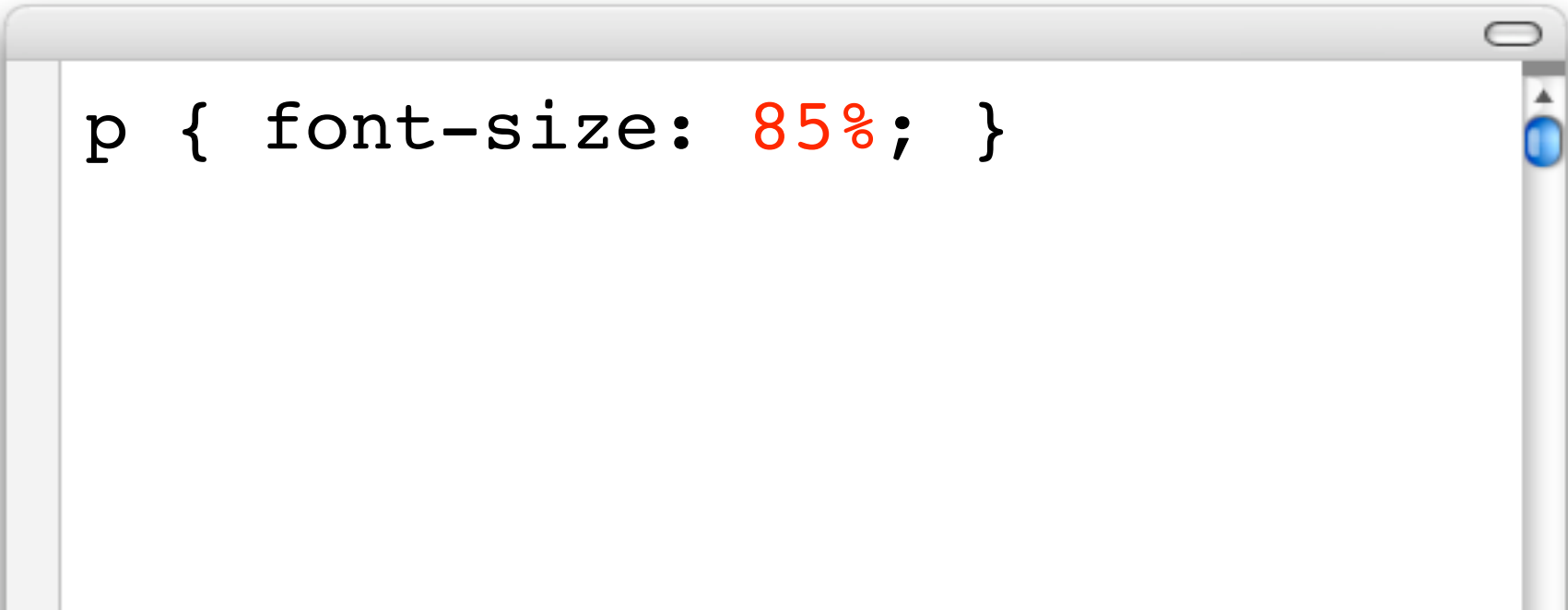


So, the `<em>` element inherits the  
**14px value.**

element	value	calcuated value
default font size	approx 16px	
<code>&lt;body&gt;</code>	unspecified	approx 16px
<code>&lt;p&gt;</code>	14px	14px
<code>&lt;em&gt;</code>	unspecified	inherited value = 14px

# Example 2: Percentage

The <p> element has been given  
a font-size of **85%**.

A screenshot of a code editor window with a light gray title bar and a vertical scrollbar on the right. The code inside is a CSS rule for the 'p' element, with the font-size value '85%' highlighted in red.

```
p { font-size: 85%; }
```



The browsers default font-size (16px) and the percentage value (85%) are used to create a calculated value ( $16\text{px} \times 85\% = 13.6\text{px}$ ). **This calculated value is inherited by descendants.**



So, the `<em>` element inherits the  
**13.6px calculated value.**

element	value	calcuated value
default font size	approx 16px	
<code>&lt;body&gt;</code>	unspecified	approx 16px
<code>&lt;p&gt;</code>	85%	16px x 85% = 13.6px
<code>&lt;em&gt;</code>	unspecified	inherited value = 13.6px

# Example 3:

## EMs

# The <p> element has been given a font-size of **.85em**.

Note: Avoid using EMs for font-size values under 1em as IE5 renders these values in pixels instead of EMs (.8em is rendered as 8px).

A screenshot of a code editor window with a light gray border and a white background. The code is written in a monospaced font. The text ".85em" is highlighted in red. The window has a standard macOS-style title bar with a close button in the top right corner.

```
p { font-size: .85em; }
```

The browsers default font-size (16px) and the EM value (.85em) are used to create a calculated value ( $16\text{px} \times .85\text{em} = 13.6\text{px}$ ).

**This calculated value is inherited by descendants.**



So, the `<em>` element inherits the  
**13.6px calculated value.**

element	value	calcuated value
default font size	approx 16px	
<code>&lt;body&gt;</code>	unspecified	approx 16px
<code>&lt;p&gt;</code>	<b>.85em</b>	16px x .85em = <b>13.6px</b>
<code>&lt;em&gt;</code>	unspecified	inherited value = <b>13.6px</b>

Those examples were too simple.  
What about more complex  
examples using **different  
elements?**



# Example 4:



All elements have been specified using **percentage values**.

```
body { font-size: 85%; }  
h1 { font-size: 200%; }  
h2 { font-size: 150%; }
```

The browsers default font-size (16px) and the body percentage value (85%) are used to create a calculated value ( $16\text{px} \times 85\% = 13.6\text{px}$ ). This calculated value is inherited by descendants **unless new values are specified.**

# The **font-size inheritance** in action

element	value	calculated font-size
default font size	approx 16px	
<body>	85%	16px x 85% = 13.6px
<h1>	200%	inherited value 13.6px x 200% = 27.2px
<h2>	150%	inherited value 13.6px x 150% = 20.4px
<p>	unspecified	inherited value = 13.6px
<em>	unspecified	inherited value = 13.6px

**Using inheritance  
for efficiency**

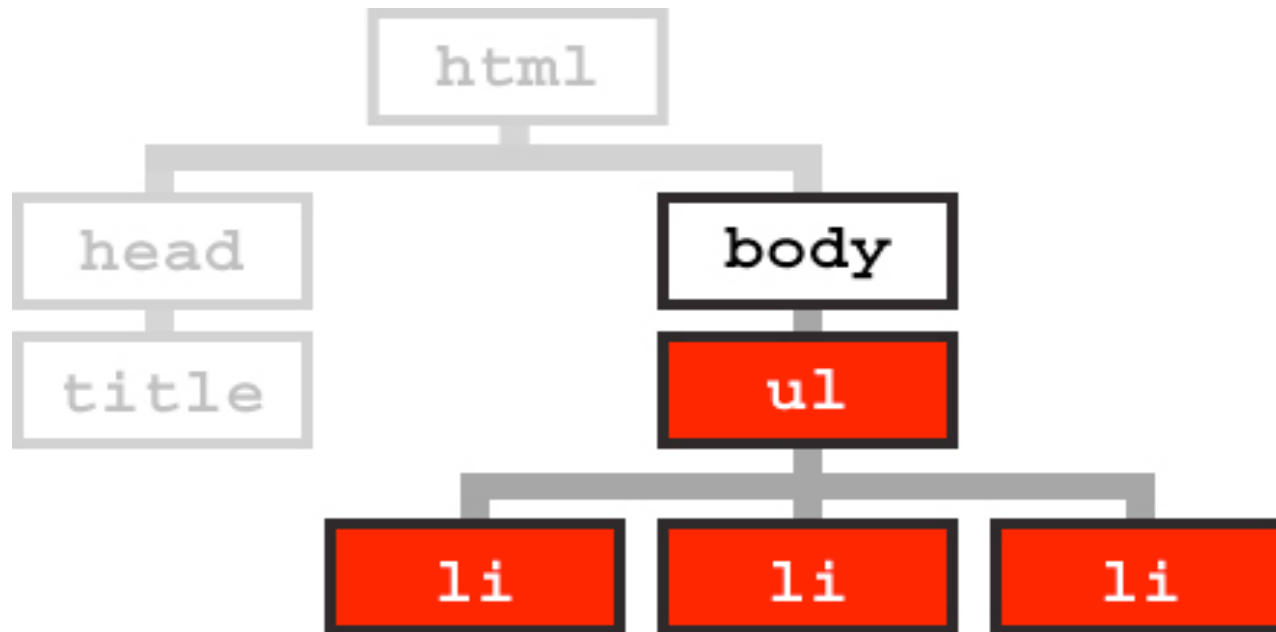
Authors can use inheritance to  
write **efficient CSS**.



For example, you can set the color, font-size and font-family on the **body element**.

```
body {  
    color: #222;  
    font-family: arial,  
    helvetica, sans-serif;  
    font-size: 90%;  
}
```

These properties will be **inherited**  
**by all descendant elements.**



You can then **override** the properties as needed, specifying new color values...



```
body {  
    color: #222;  
    font-family: arial,  
    helvetica, sans-serif;  
    font-size: 90%;  
}
```

```
h1, h2, h3 { color: green; }  
h4, h5, h6 { color: black; }
```

new **font-family values...**

```
body {  
    color: #222;  
    font-family: arial,  
    helvetica, sans-serif;  
    font-size: 90%;  
}
```

```
h1, h2, h3 { color: green; }  
h4, h5, h6 { color: black; }
```

```
h1, h2, h3, h4, h5, h6 {  
    font-family: georgia,  
    times, serif;  
}
```

and new **font-size values** as  
needed.

}

```
h1, h2, h3 { color: green; }  
h4, h5, h6 { color: black; }
```

```
h1, h2, h3, h4, h5, h6 {  
    font-family: georgia,  
    times, serif;  
}
```

```
h1 { font-size: 200%; }  
h2 { font-size: 150%; }  
h3 { font-size: 125%; }  
#footer { font-size: 90%; }
```

Now, go forth and  
**inherit the world!**



# We're done!

Downloaded from: <http://www.slideshare.net/maxdesign/css-inheritance-a-simple-stepbystep-tutorial>  
Presentation by: Russ Weakley