Cascading Style Sheets

It was still 1999...

The W3C made a decision: they'd standardize a subset of HTML and then would abandon the child. No more HTML. Ever.

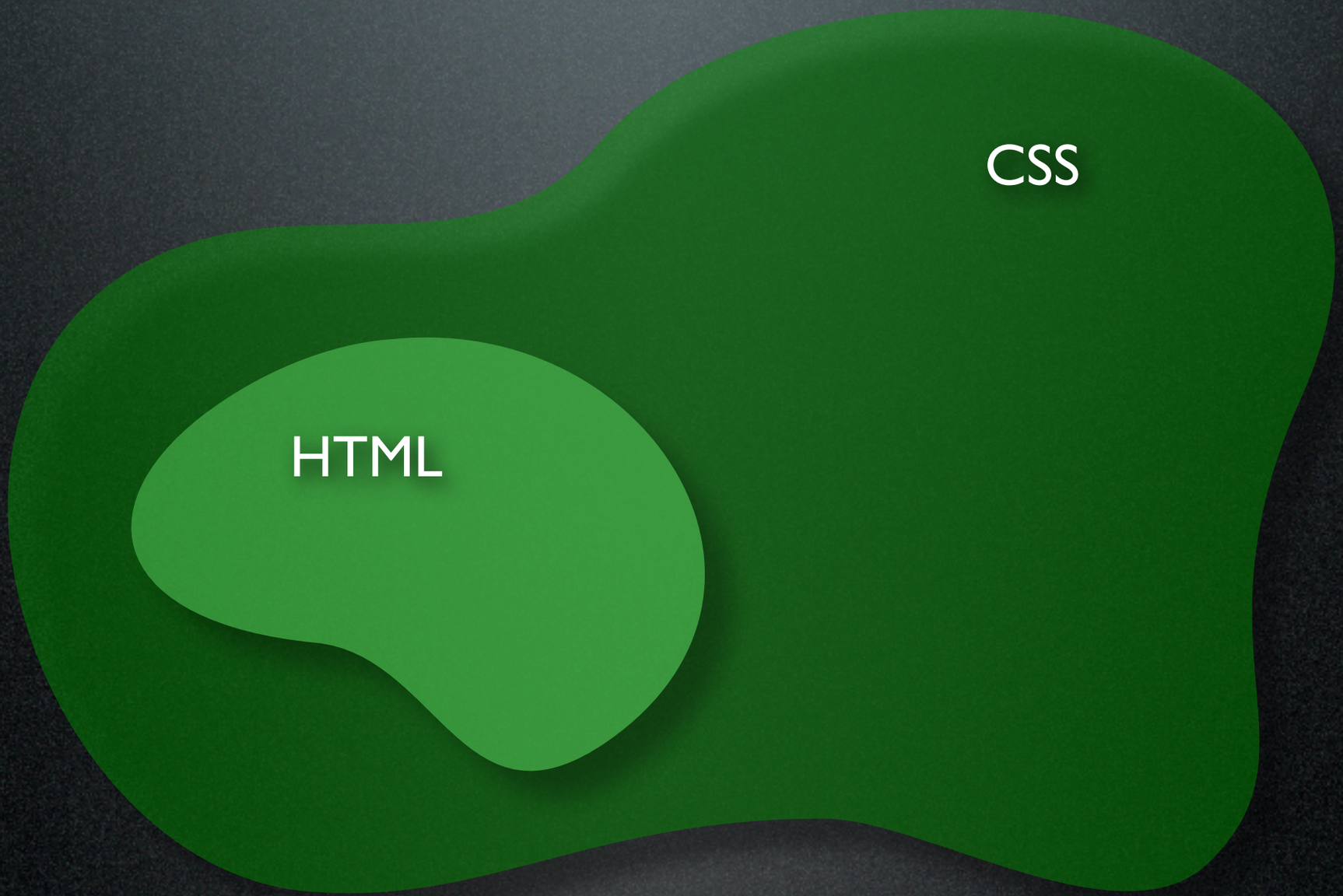
And they said, in effect, "OK, if you want stylization and presentation effects, then we'll give you stylization and presentation effects. But we're going to do it right."

The result was Cascading Style Sheets (CSS). CSS is a parallel "language" to HTML that allows minute and precise control over every visual aspect of HTML. CSS allows you to control parts of HTML that HTML itself could never control. The result was a two-part structure:

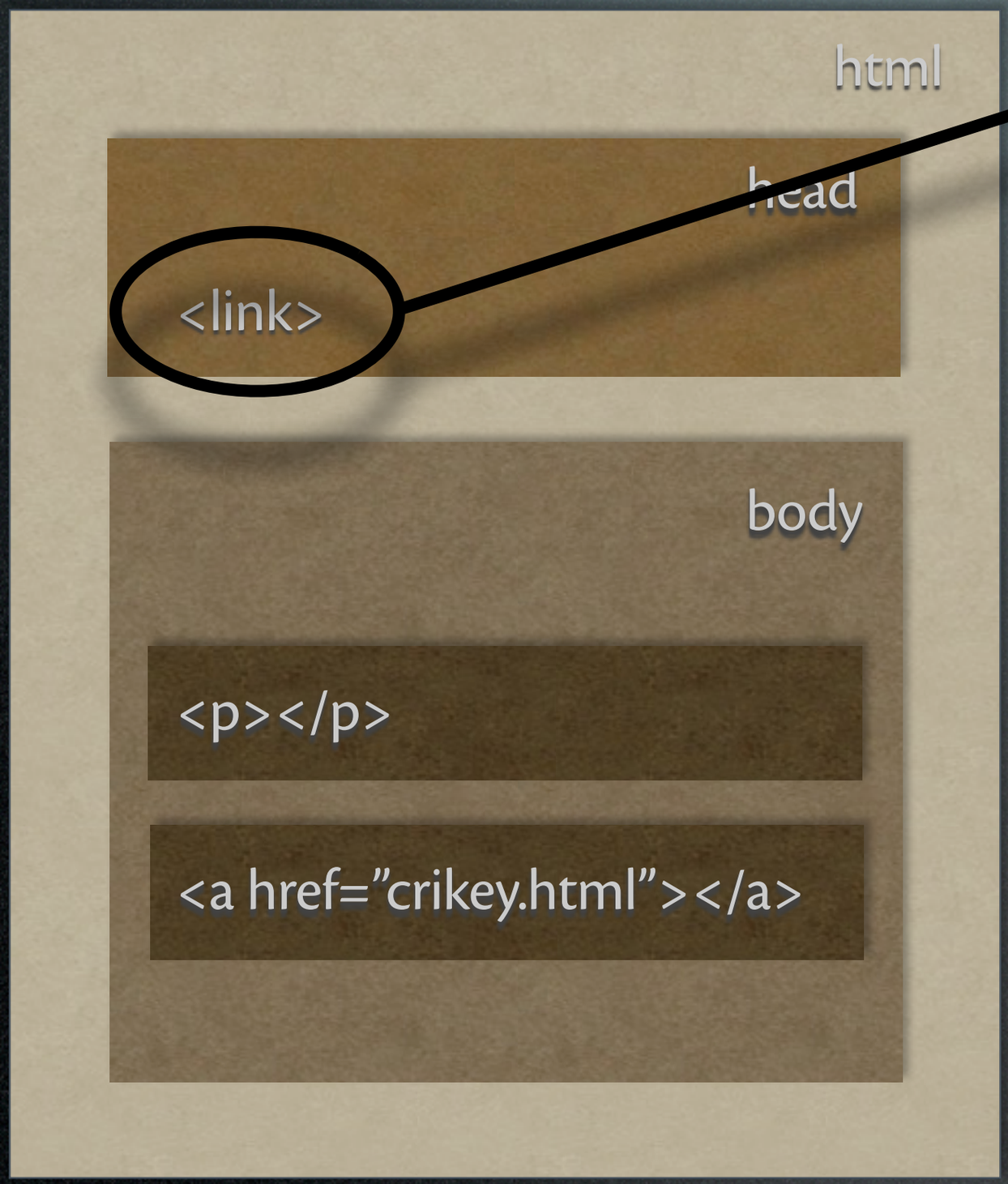
HTML	CSS
structural	presentational

CSS & HTML

CSS not only modified but actually *extended* the presentational abilities of HTML. That's why I recommend you have a good CSS reference. Learning HTML by itself is no longer sufficient and, arguably, *learning CSS is more important than learning HTML.*



What's CSS?




```
body {
        background: blue;
        color: yellow;
}

p {
        font-family: serif;
        line-height: 1.5 em;
}

h1, h2, h3 {
        margin-left: 3 em;
}
```

The CSS is just another plain text document that tells the browser how to draw things. It's connected to the webpage via the <link> tag.

The <link> Tag

The <link> tag tells the browser that it should look for an external document that's important to this webpage. The <link> tag must go inside the <head> section and it is an empty tag; it has no closing counterpart. It requires three attributes:

<link rel="stylesheet" href="*address_here*" type= "text/css">

Relationship: the browser should look for this document's *stylesheet*. (note: it's one word!)

The <link> Tag

The <link> tag tells the browser that it should look for an external document that's important to this webpage. The <link> tag must go inside the <head> section and it is a solo tag; it has no closing counterpart. It requires three attributes:

<link rel="stylesheet" href="*address_here*" type= "text/css">

Relationship: the browser should look for this document's *stylesheet*. (note: it's one word!)

Hypertext Reference: either relative or absolute addressing is OK.

The <link> Tag

The <link> tag tells the browser that it should look for an external document that's important to this webpage. The <link> tag must go inside the <head> section and it is a solo tag; it has no closing counterpart. It requires three attributes:

<link rel="stylesheet" href="*address_here*" type= "text/css">

Relationship: the browser should look for this document's *stylesheet*. (note: it's one word!)

Hypertext Reference: either relative or absolute addressing is OK.

Type: it's a plain text document full of CSS stuff.

The <link> Tag

The <link> tag tells the browser that it should look for an external document that's important to this webpage. The <link> tag must go inside the <head> section and it is a solo tag; it has no closing counterpart. It requires three attributes:

<link rel="stylesheet" href="*address_here*" type= "text/css">

Relationship: the browser should look for this document's *stylesheet*. (note: it's one word!)

Hypertext Reference: either relative or absolute addressing is OK.

Type: it's a plain text document full of CSS stuff.

CSS Syntax

The syntax for CSS is slightly different from HTML. OK, a lot different:

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

Where:

selector	is an HTML tag without the < > brackets
property	is a tag attribute *defined in CSS terms* (CSS has many more attributes than HTML does!)
value	is a number or a colour or a pre-defined keyword — the "specifics" of this definition.

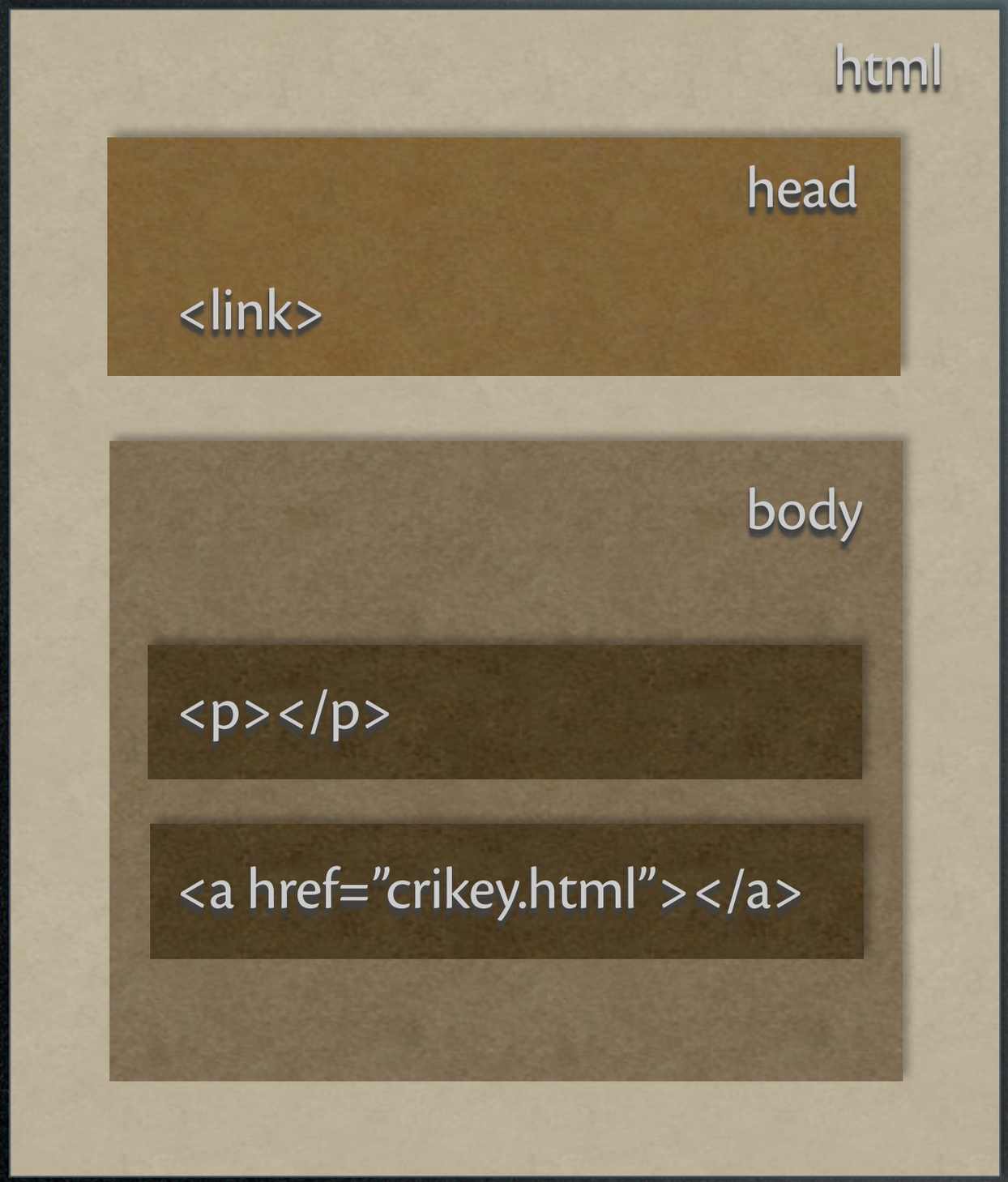
```
p { background: blue; }
```

CSS Syntax

Any tag ("selector") can define multiple properties. For clarity, I usually put each one on a separate line and indent it. Notice that each property-value pair ends in a semi-colon. And note that this is a moment when computer geeks are immaculately organized:

```
p {
    background: blue;
    font-family: Georgia, Times, serif;
    text-align: right;
    border: 4px green dotted;
    padding: 2em;
    color: white;
}
```

HTML Containers



We've already seen that properly formed HTML produces a series of nested containers — the opening and closing tags mark the boundaries of the box.

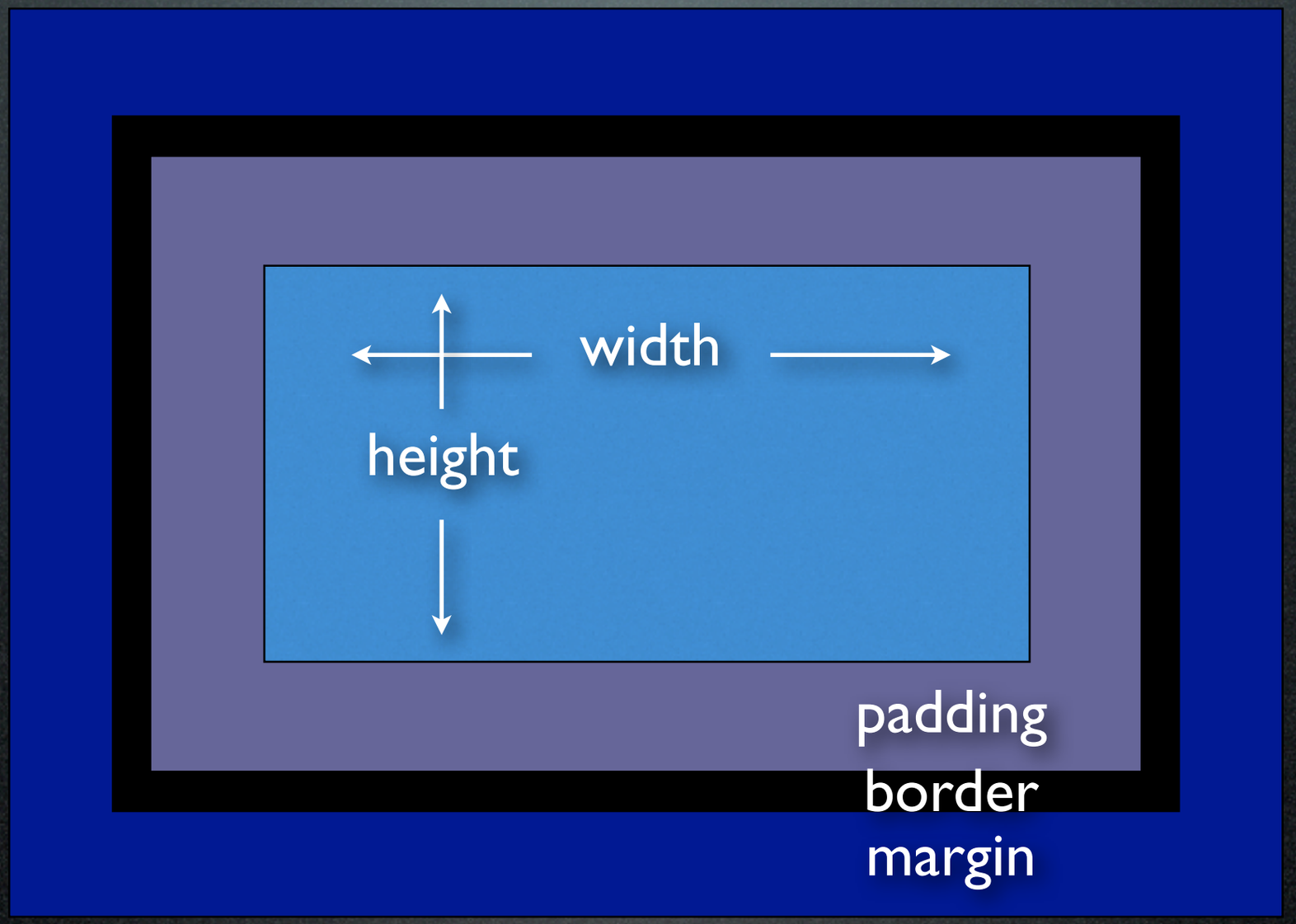
That concept is *crucial* to CSS for two reasons:

1) the style sheet tells the browser how to draw the boxes for each tag. It uses the "*box model*."

2) properties defined in a big box also apply to all the smaller boxes inside it. That's the "*cascade*."

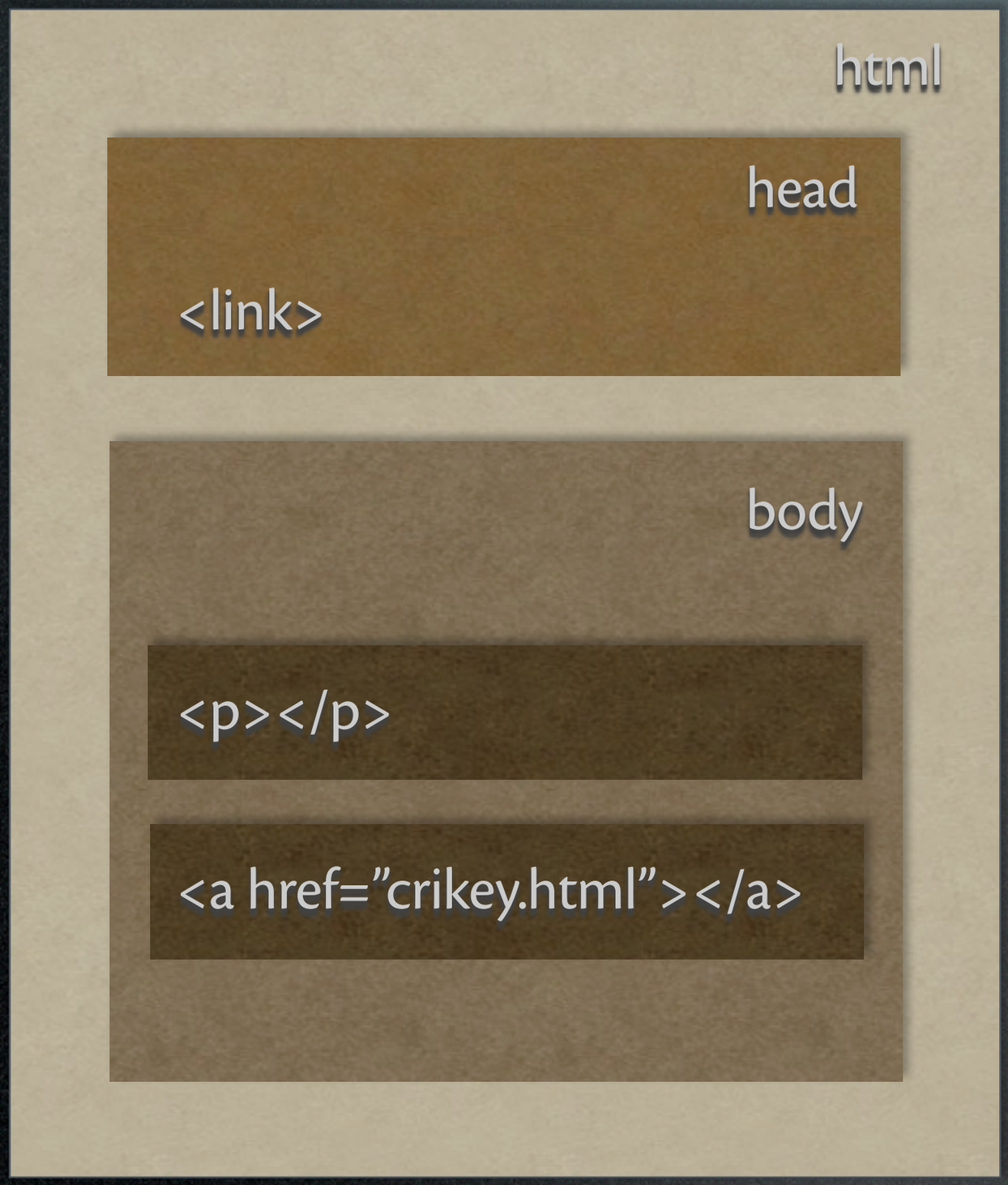
The Box Model

All boxes in CSS have the following components:



Notice that the complete box is bigger than is specified by the width and height!

HTML Boxes

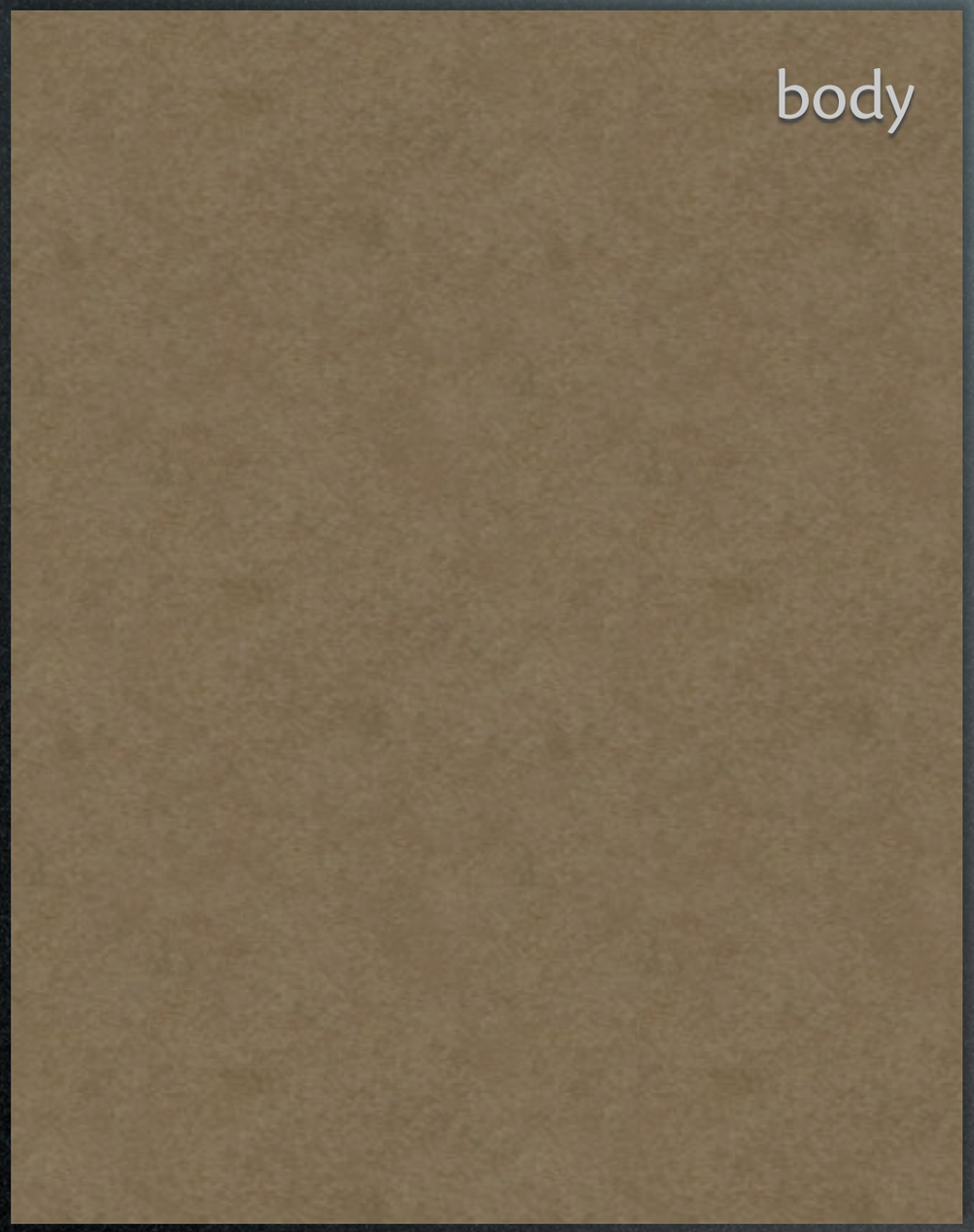


Since nothing in the <head> section gets displayed, it makes no sense to try to stylize it with CSS. No one ever does it.

Many browsers do not recognize that <html> is a real box, so no one in CSS-land ever stylizes <html> either.

That leaves us with <body> as the biggest box that we can stylize in our CSS.

The Box Model



Like all other boxes, body has the following components:

background
color (it's a *foreground* color*)
margin (outside the border)
border
padding (inside the border)
width (inside everything)
height (inside everything)

* The foreground color applies to fonts and borders

The “Cascade”

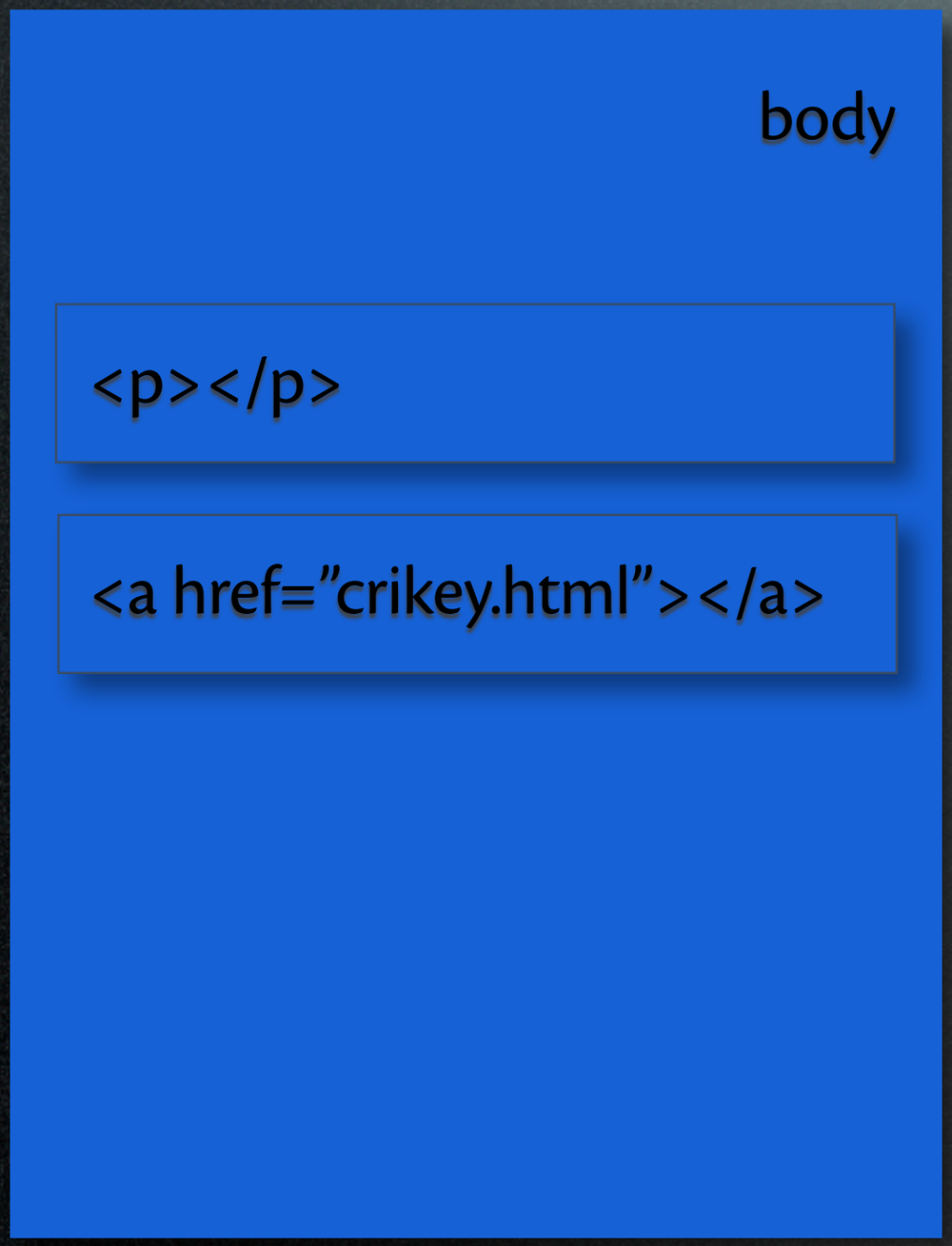
I am free to define some or all of those box characteristics in my CSS. Anything left undefined will be displayed according to the browser’s default display settings — and remember, those can vary from browser to browser.

I can also add some other definitions that describe the page’s typography. They’re not part of the box model per se, but they are additional CSS features over which I have control:

```
body {
    background: blue;
    margin: 2 em;
    font-family: Trebuchet, “Trebuchet MS”, sans-serif;
    line-height: 1.5em;
}
```

Any sub-containers will also have a blue background, use a sans-serif font, and be word-processor-esque 1.5 line spaced.

Genetics & Inheritance

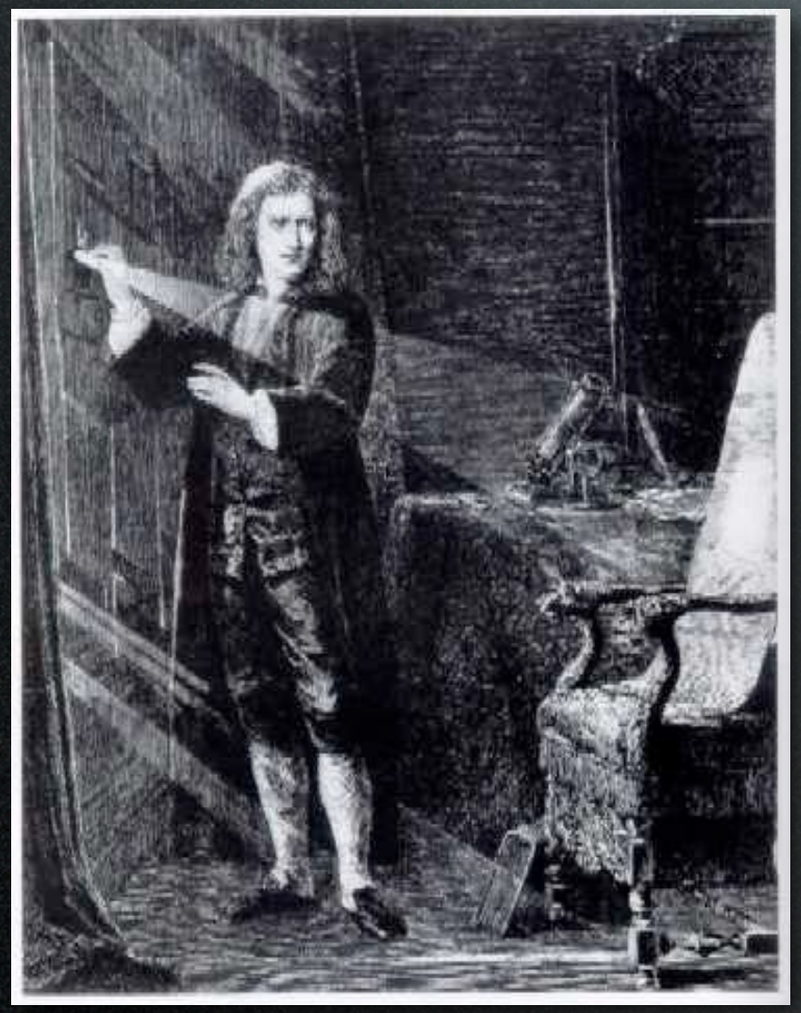


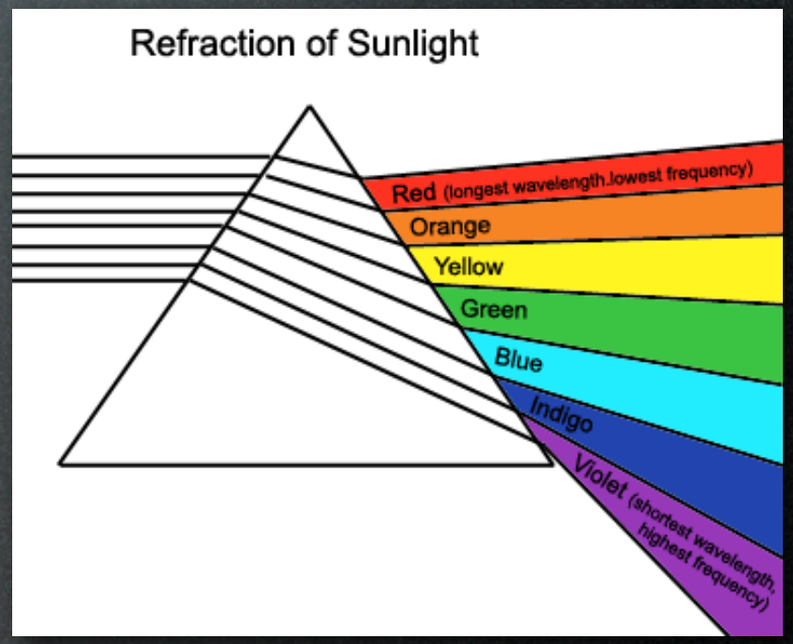
In this case, everything will be blue. All my text will be 1.5-spaced and, since I did not define a foreground color, the browser will be free to pick a font color. Black is a common default, but there's no guarantee of that.

The bigger container is called the parent and its smaller containers are its children. As in genetics, children inherit qualities from their parents. Some qualities (like margins, borders and padding) are not inheritable. That might seem counter-intuitive, but it's usually what you want. If not, you can tell any child to inherit any specific qualities from its parent.

A Brief Digression about Colours

(in which the ghost of Isaac Newton returns to haunt us)





A Quick Note about Color

HTML defined 16 popular colors that could be referenced by name. CSS added one more (orange), so there are 17 valid named colors in CSS. Some browsers recognize as many as 140 color names, but they're not standardized, so use them at your peril.

aqua	navy
black	olive
blue	orange
fuchsia	purple
gray	red
green	silver
lime	teal
maroon	white
	yellow

Mixing Colors

The three primary colors of light are Red, Green and Blue (ha! what did *you* think they were? *Pigment* and *light* are not the same!). Any color can be made by mixing various proportions of these three primary colors.

Computers can measure colors in a variety of ways—usually as a number between 0 and 255. To make matters worse, those numbers are usually represented as a *hexadecimal* number, which is base 16 (we normally count in base 10). Hexadecimal digits are 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Actually, this is just a shorthand so that colors can be identified by two digits rather than three. It saves space and typing.

Hex Triplets

The color is specified as a number: a preceding # sign followed by six hexadecimal digits. The pattern is this:

#RRGGBB

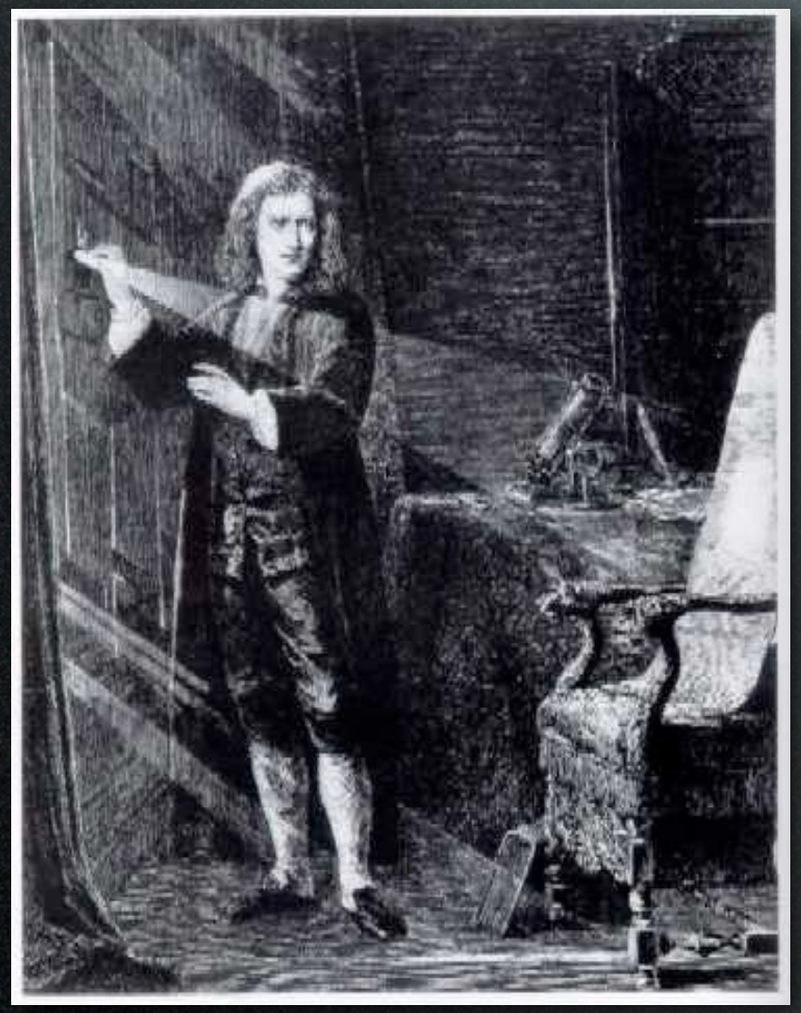
Each pair of digits represents the amount of that color to use — from none (00) to 100% (FF).

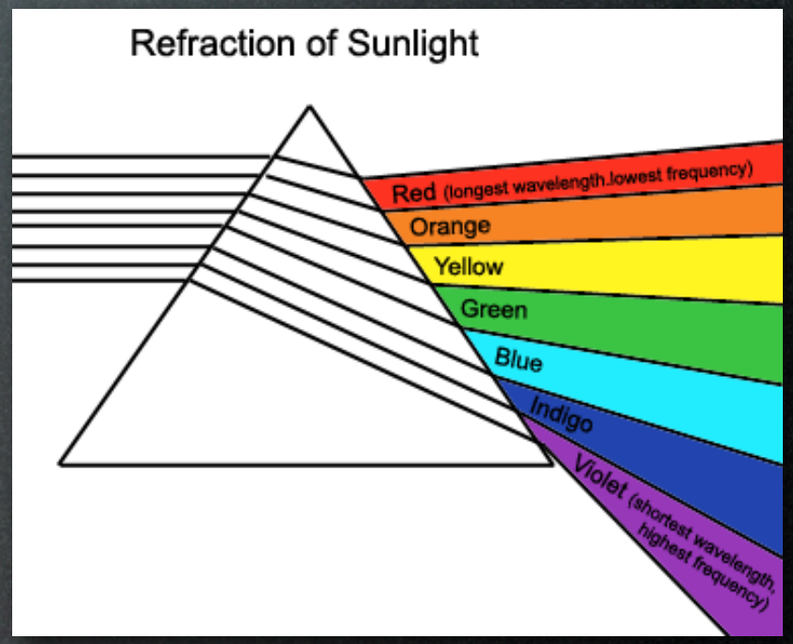
So pure red is synonymous with #FF0000. Pure green is synonymous with #00FF00. Pure blue is synonymous with #0000FF.

In contrast to pigments, mixing all the colors together makes white (#FFFFFF). Subtracting out all the colors makes black (#000000). Higher numbers mean a lighter color; lower numbers mean a darker color.

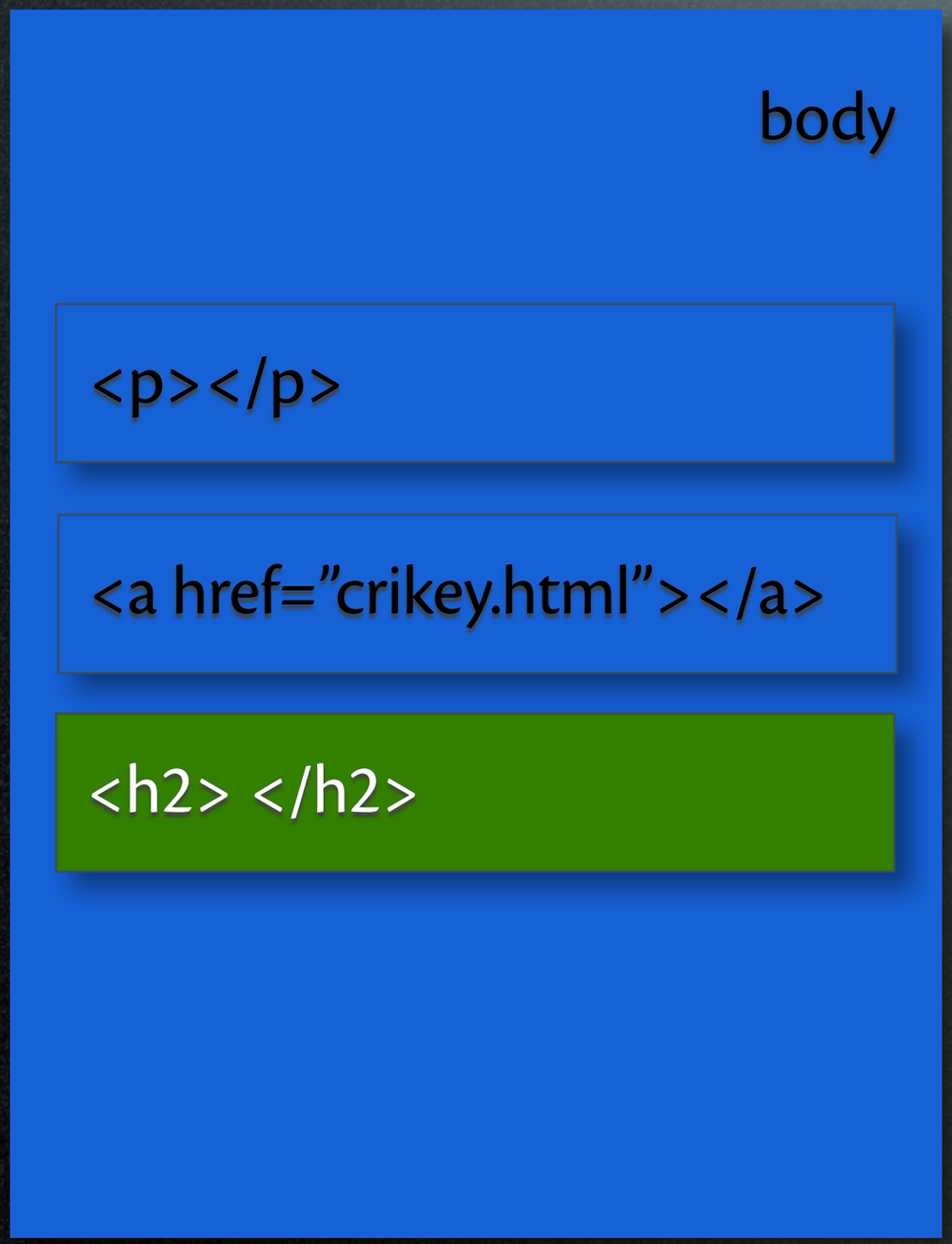
End of Digression

(Isaac Newton rests again in peace)





Smaller Boxes Win



We've seen how big box qualities are inherited by smaller child boxes. But any child can "override" a parent's definition.

For example, I can add this to my CSS to modify the page:

```
h2 { background: green;
     color: white;
}
```

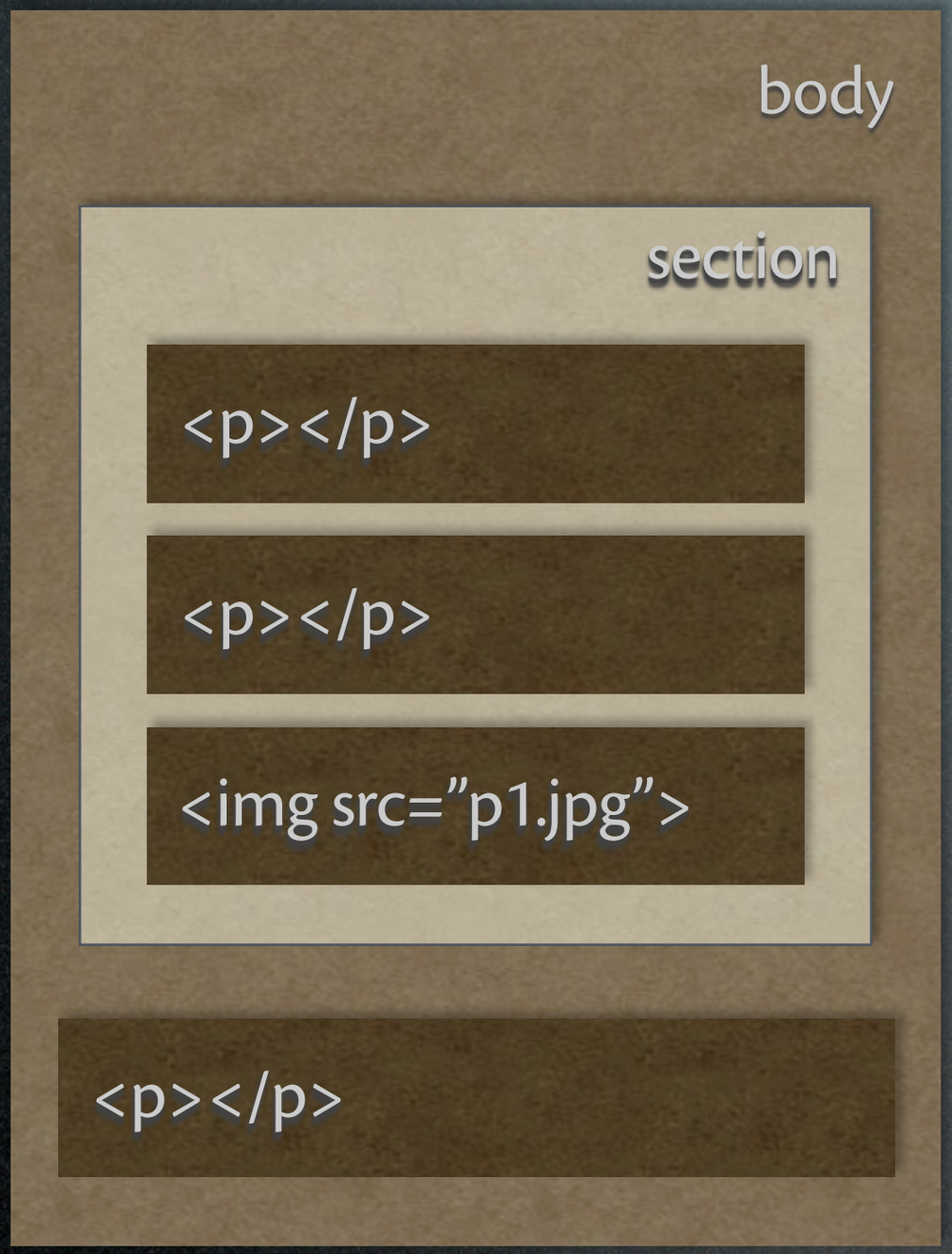
A smaller box can modify or supplement any of the qualities it may have inherited from its parent.

CSS Design Strategy

Using a CSS, you'll follow these general tactics:

1. Define the most universal, omnipresent qualities in the biggest possible box.
2. Define smaller boxes only when they are either *different from* the bigger box or when they must *add new qualities* to those already inherited from the bigger box.

Grouping Elements



What if I wanted somehow to treat a group of boxes as one unit?

Well, I can. I can use a <section> or container to group the elements. These containers are normally invisible, but I can stylize them with a CSS.

HTML5 has many grouping containers like <section> and . A <section> is a *block element*; a is an *inline element*. Both are invisible unless you specify otherwise. See our HTML handout for more specifics.

Block	Inline
Block-level elements generate an element box that (by default) fills its parent element's content area and cannot have other elements at its sides. In other words, it generates "breaks" before and after the element box. The most familiar block-level elements from HTML are p and div. Replaced elements can be block-level elements, but they usually are not. (*CSS Reference*, p. 9)	*Inline-level elements* generate an element box within a line of text and do not break up the flow of the line. The best inline element example is the a element in HTML. Other candidates would be strong and em. These elements do not generate a "break" before or after themselves and so they can appear within the content of another element without disrupting its display. (*CSS Reference*, p. 9)
<hr> <section>	<img>

Identifying Your Boxes

Any container can be identified via a label. It's very common to label 's and 's, but any and all tags can take an attribute that labels it:

id="*some unique name*"
class="*some group name*"

Examples:

<section class="warning"> </section>

Here are the rules:

id names have to be UNIQUE; there can be only ONE per HTML document (most browsers don't check, but a validator will).

classes can have multiple occurrences; class is a useful way to identify elements that may repeat throughout a web page.

Class & ID in CSS...

CSS can stylize id's and classes with a special syntax:

```
.warning { color: red; }                /* class selector */
#footnotes { font-size: smaller; }      /* ID selector */
```

Here's the mnemonic I use:

CD-HI

... which stands for

Class = Dot ; Hash = ID

Using Classes

Remember the display property? What if we wanted to center only certain <img> tags on our webpage, but not all of them?

```
img.centered {
    display: block;
    margin-left: auto;
    margin-right: auto;
}
```

Here any <img class="centered"> tag will take on this presentational style. A standard <img> tag unadorned by a class attribute will remain *inline*, which is the default presentational style for that tag.

Pseudo-Classes

It'd be great if we could add classes to HTML "on the fly" — for example, to add classes to <a> tags signifying links that have or have not been visited yet so we could stylize them differently:

```
<a href="goofus.html" class="visited">Goofus</a>
<a href="crikey.html" class="unvisited">Crikey</a>
```

CSS is already there: we can use "pseudo-classes," which are kinds of "phantom" or "ghost" classes. They are inferred or implied by the status of an element, but aren't coded specifically into the HTML. The browser's "history" cache keeps track of links that have been visited, for example, so we don't need to keep track of that. But we can use that information to stylize links:

```
a:visited { color: orange; text-decoration: none; }
```

Pseudo-Classes

Pseudo-class selectors begin with a colon (:) and *NO SPACE IS ALLOWED BETWEEN THE COLON AND THE NAME*. Putting a space after the colon will confuse the browser and it will ignore your CSS pseudo-class styles.

:link — unvisited links
:visited — visited links
:hover — in a mouse-over state
:active — in a mouse-down (click) state

Order is Important!

Remember, in a CSS the last rule wins. Stylizing links with pseudo-classes can be tricky because you have to define them in the proper sequence so that mouseovers and clicks behave properly. The sequence above is correct and fits the mnemonic:

LoVe — HAte

Tip: Flush the Cache

Web browsers scan HTML documents to see if any external documents are needed by this web page. A CSS is one such external document, so the browser sends another request to the server, downloads the CSS, and *sticks it into the cache for future reference*. The browser now references the cache — and does not return to the server — whenever it needs that particular CSS.

So what's the problem? If you're working on a web page and change the CSS, your changes *may not* always show up when you reload the page. Why not? Because the browser can't easily tell that the CSS document in the cache is not the current version.

Solution? Flush the cache. (Experiment with your browser to see how.)

In some browsers, you can also hold down shift and click refresh. The browser will fetch the most current version of the CSS.

Top 7 CSS Tips

1. Remember the syntax: selector { property: value; }
2. Learn the valid properties from a good CSS book or the W3 Schools site; CSS is its own language.
3. Force browsers into Standards Mode with a DOCTYPE tag.
4. Remember the mnemonics:
   - CD: “Hi!”
   - LoVe – HAte
5. No spaces after colon (:) in pseudo-class selectors!
6. Build your own boxes with the invisible tags <section> (block) and (inline).
7. Flush the cache and reload the page if CSS changes don’t take effect.