Lecture 7: Introduction to Cascading Style Sheets (CSS)

Dr. Jason Schanker

Introducing CSS

- CSS, which stands for Cascading Styling Sheets is used for styling your page.
- You tell how elements on your page should be styled via *rules* with *selectors* selecting the element(s) to style and *property-value* pair(s) which specify how the properties of the selected element(s) should be styled:
  - Example (Style all paragraph elements with a red background and a 1 pixel solid gray border):

```
p
{
    background-color: red;
    border: 5px solid gray;
}
```

- Note the form of the rule: selector first (`p` in this case) followed by a { followed by a series of property-value pairs in the form `property:value` , each terminated by a semicolon concluded with a closing }
- CSS rules can be specified inline in the HTML between `<style>` and `</style>` tags or placed in a separate .css file, which can then be linked in via the `link` element.
- Check out the beautiful lounge page with this styling (Remove /* and */ in the CSS, which is a comment that's ignored by the browser when rendering the CSS): http://codepen.io/anon/pen/GjmxyG

CSS Exercise 1 for the Head First Lounge

- As an example, see: http://codepen.io/anon/pen/kkyEEr
- Open the lounge.html file and add `<style>` and `</style>` tags in between the `<head>` and `</head>` tags. Then insert CSS rules in between the `<style>` and `</style>` tags to perform the following styling:
  - The paragraph text color should be maroon. You can specify the text color (foreground color) with the `color` property.
  - The heading elements (`h1` and `h2`) should have a sans-serif value for the font-family property and a gray (text) color.
  - **Note**: If you have multiple elements for which the same styling applies, you can list all of the selectors separated by commas in a single rule.

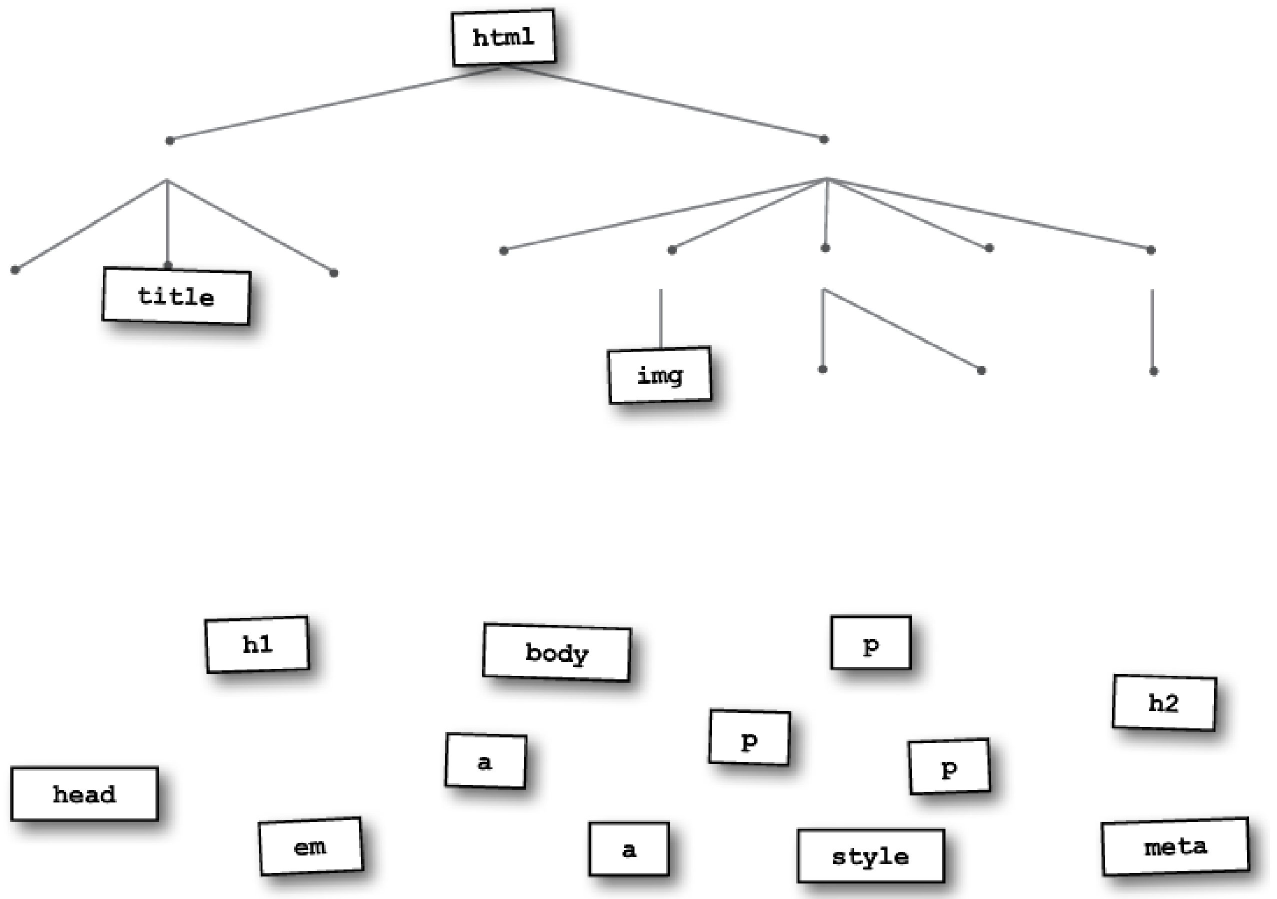
CSS Exercise # 2

- Make the `border-bottom` property of the `h1` elements have a value of `1px solid black`. The difference between setting an underline style and a bottom border is that a bottom border can extend past the text it’s below.
    - Note: You can have multiple CSS rules in which an `h1` element is selected; it is good to do so when an h1 element shares some property values with one or more elements but not all of them. In this case you “factor out” the shared properties in a separate rule so if you ever want to change the common properties, you can do so in one place.
        - Example: Only h1 elements should have a bottom border, but both h1 and h2 elements should have a sans-serif value for the font-family property and a gray (text) color

Resizable Images with Transition

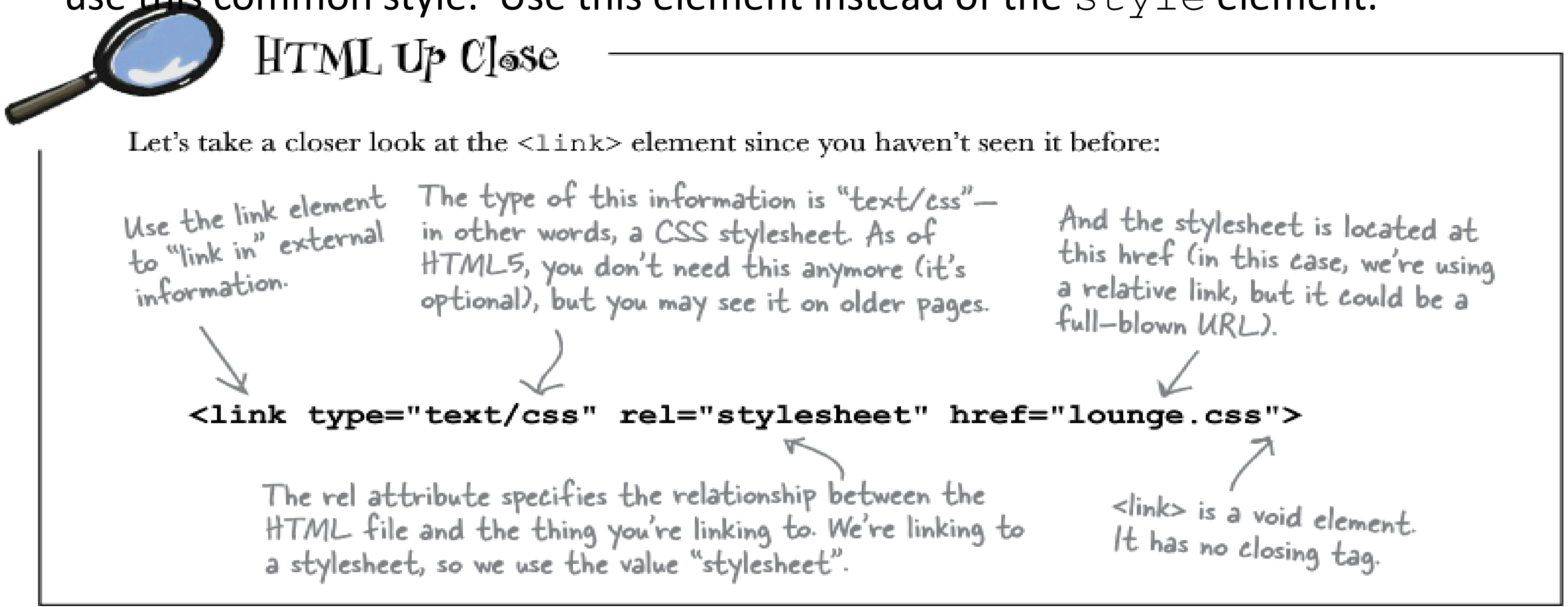
- See: http://codepen.io/anon/pen/EgmEOW
- CSS width and height overrides width and height specified for image in HTML.
- `auto` for height means the height will scale in proportion with the width.
- `:hover` is an example of a pseudo-class; by appending it to the `img` selector, the rule will be applied whenever an image is being hovered over.
- **CSS3 transition**: By giving a time interval for the `transition` property in the `img:hover` rule, the height and width of the image will adjust gradually instead of immediately when the image is hovered over. Similarly, by providing a transition property in the `img` rule, the image will gradually return to its initial size when removing the mouse cursor.

Exercise: Fill in the HTML element tree for the lounge page



Linking style sheets

- We often want consistent styling among the many web pages that can make up our site. Inserting the common CSS across the pages is a *bad* idea because if we ever wanted to change this style, we'd be modifying every page.
- Can create an external .css file (stylesheet) and then insert a `link` element in the `head` element of the HTML in each page to tell the browser that we want to use this common style. Use this element instead of the `style` element.



HTML Up Close

Let's take a closer look at the `<link>` element since you haven't seen it before:

Use the link element to "link in" external information.

The type of this information is "text/css"—in other words, a CSS stylesheet. As of HTML5, you don't need this anymore (it's optional), but you may see it on older pages.

And the stylesheet is located at this href (in this case, we're using a relative link, but it could be a full-blown URL).

```
<link type="text/css" rel="stylesheet" href="lounge.css">
```

The rel attribute specifies the relationship between the HTML file and the thing you're linking to. We're linking to a stylesheet, so we use the value "stylesheet".

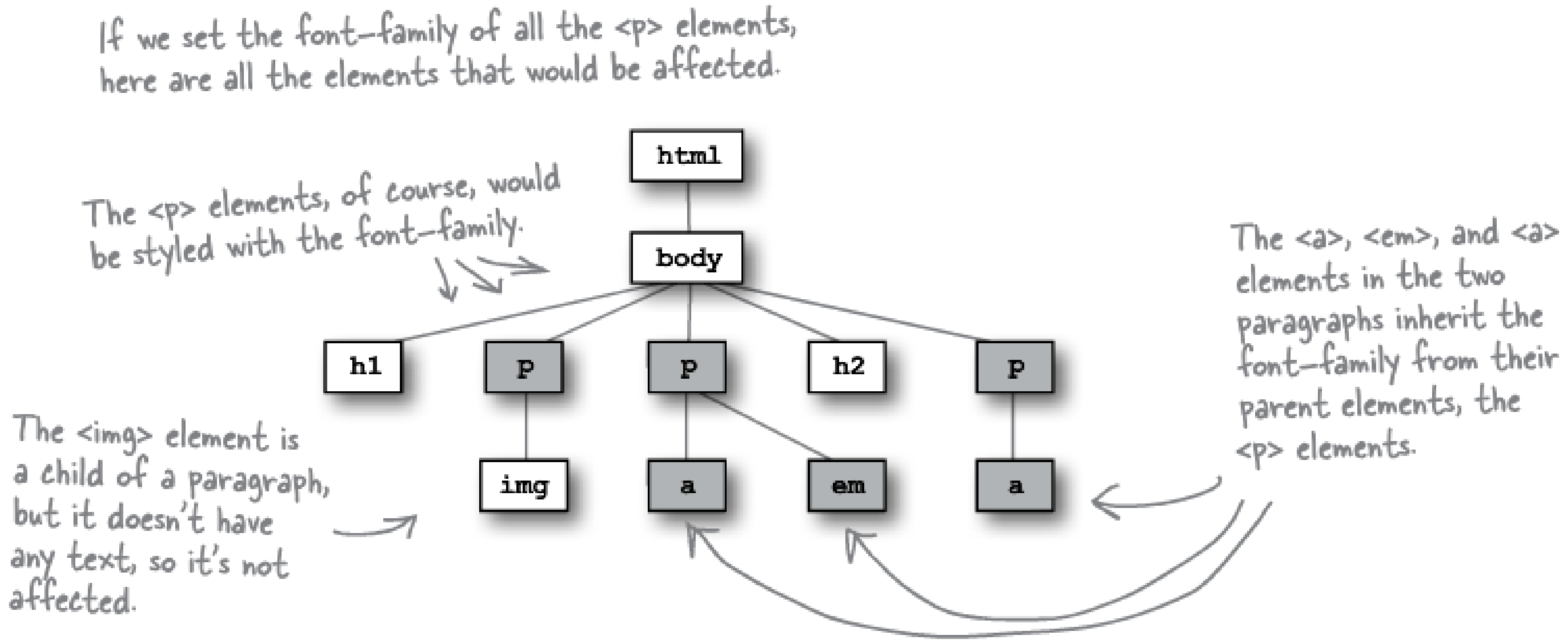
<link> is a void element. It has no closing tag.

Exercise: External style sheet

- Cut all of the CSS from between the `<style>` and `</style>` tags and paste it in a new file, lounge.css. Then remove the `style` element and replace it with the `link` element from the previous page. Finally, link to the external stylesheet from the elixir.html and directions.html files.

CSS Inheritance

- If the value for an element's property is not specified directly, then it may be inherited from its "most immediate" parent (parents are more immediate than grandparents who are more immediate than great grandparents, etc.) for which this styling information is specified.


If we set the font-family of all the <p> elements, here are all the elements that would be affected.
The <p> elements, of course, would be styled with the font-family.
The <img> element is a child of a paragraph, but it doesn't have any text, so it's not affected.
The <a>, <em>, and <a> elements in the two paragraphs inherit the font-family from their parent elements, the <p> elements.
html
body
h1
p
p
h2
p
img
a
em
a

Inheritance Exercises

- Modify the stylesheet so that *all* of the text uses the `sans-serif` font-family. Be sure to remove any unnecessary property-value pairs.
- To override inheritance, simply specify the *new* value you want that element’s property to assume in a selector for that element. Now make `em` elements have a `font-family` of `serif` instead by overriding its `sans-serif` inheritance.

Classes

- In cases where you don't want to style elements exclusively by type (e.g., *all* paragraphs, *all* primary headings, etc.), you can form your own collections of elements to style as a group by assigning the same value to each of their `class` attributes.
    - Each class name should be one word.
    - You may list multiple classes for elements by separating the names by spaces (e.g., `blue` and `description` classes: Use `<p class = "description blue">` or `<p class = "blue description">`
    - In the CSS, you specify classes with dot and the class name (e.g., `.blue` selector to specify all types of elements belonging to the blue class) or if you only want to style classes for certain types of elements, precede the `.` with the element type (e.g., `p.blue` to only select *paragraphs* belonging to the blue class).

Exercise on classes

- Modify the HTML by adding appropriately valued class attributes. Then create CSS rules with class selectors so that all of the gains get styled with green text and losses with red text. If the gain or loss is very large (greater than 1000 or less than -1000), make it bold.

```
<!doctype html>
<html>
  <head>
    <title>List of Gains/Losses</title>
    <meta charset = "utf-8">
    <link type = "text/css" rel = "stylesheet" href = "stocks.css">
  </head>
  <body>
  <h1>Stocks</h1>
  <ul>
    <li>Apple: -$2534.86</li>
    <li>Google: $145.87</li>
    <li>Red Hat: $2734.45</li>
    <li>IBM: -$41.15</li>
    <li>Microsoft: $324.59</li>
  </ul>
  </body>
</html>
```

Possible Interview Question (Going beyond the code): Why might this not be such a great idea for styling losses and gains?

Book Exercise

- Assign each of the first 3 drink paragraphs on the elixir.html page to its own class using a name to match the drink that it describes (i.e., greentea, raspberry, and blueberry classes) and then add appropriate rules in lounge.css so that each paragraph text color matches the color of the drink it describes (i.e., green, blue, and purple). Note the cranberry paragraph is already the correct text color by the CSS rule:

```
p {
    color:maroon;
}
```

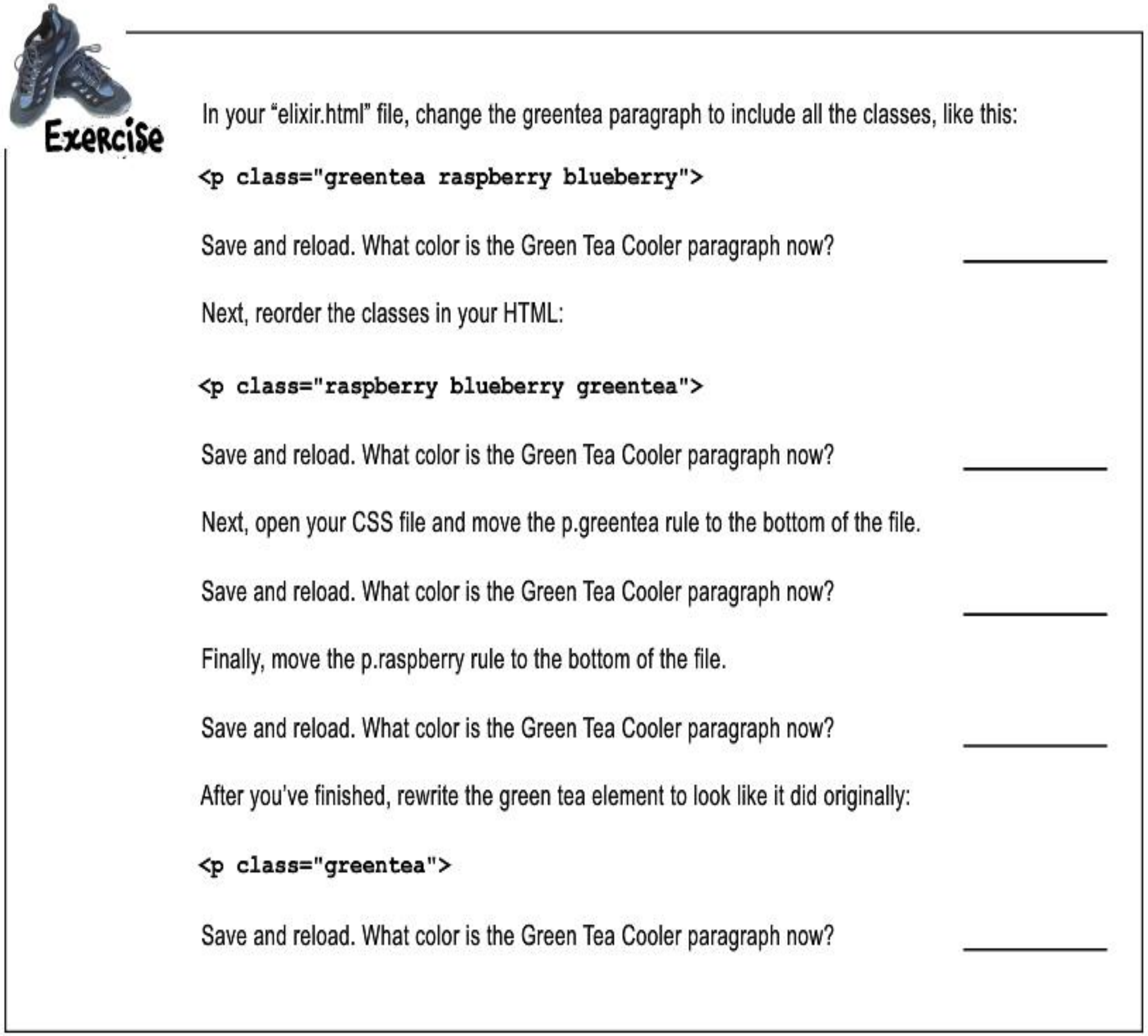
- **Question on conflicting rules**: The green tea paragraph is still a paragraph so how does the browser decide between the above rule and the newly added rule specifying different text colors?

Specificity: More details later

- When rules conflict, the "most specific" one is used.
  - If no CSS rules exist for an element, we use the rule applying to the element's most immediate parent (if any such rules exist).
  - Given a rule which selects the element's class and one which selects the element's type (e.g., `.greentea` vs. `p`), we use the class rule (`.greentea`).
  - Given a rule which selects the element's class *and* type and one that selects the element's class alone (e.g., `p.greentea` vs. `.greentea`), we use the one that selects the class and the type (`p.greentea`)
- Exercises (for thought):
  - Suppose we were to replace raspberry by <em>raspberry</em> in the description of the raspberry drink, what would its text color be? Why? What would be the text color if we were to then add a rule with an `em` selector specifying a color of maroon? What if the rule instead only selected `em.raspberry` elements?

Book exercise on specificity: Tiebreakers

- Based on your tests, does the order that you specify the classes matter? What about the order that you specify the CSS rules? How do each of these orderings matter or does it not matter?



In your "elixir.html" file, change the greentea paragraph to include all the classes, like this:

```
<p class="greentea raspberry blueberry">
```

Save and reload. What color is the Green Tea Cooler paragraph now? ________

Next, reorder the classes in your HTML:

```
<p class="raspberry blueberry greentea">
```

Save and reload. What color is the Green Tea Cooler paragraph now? ________

Next, open your CSS file and move the p.greentea rule to the bottom of the file.

Save and reload. What color is the Green Tea Cooler paragraph now? ________

Finally, move the p.raspberry rule to the bottom of the file.

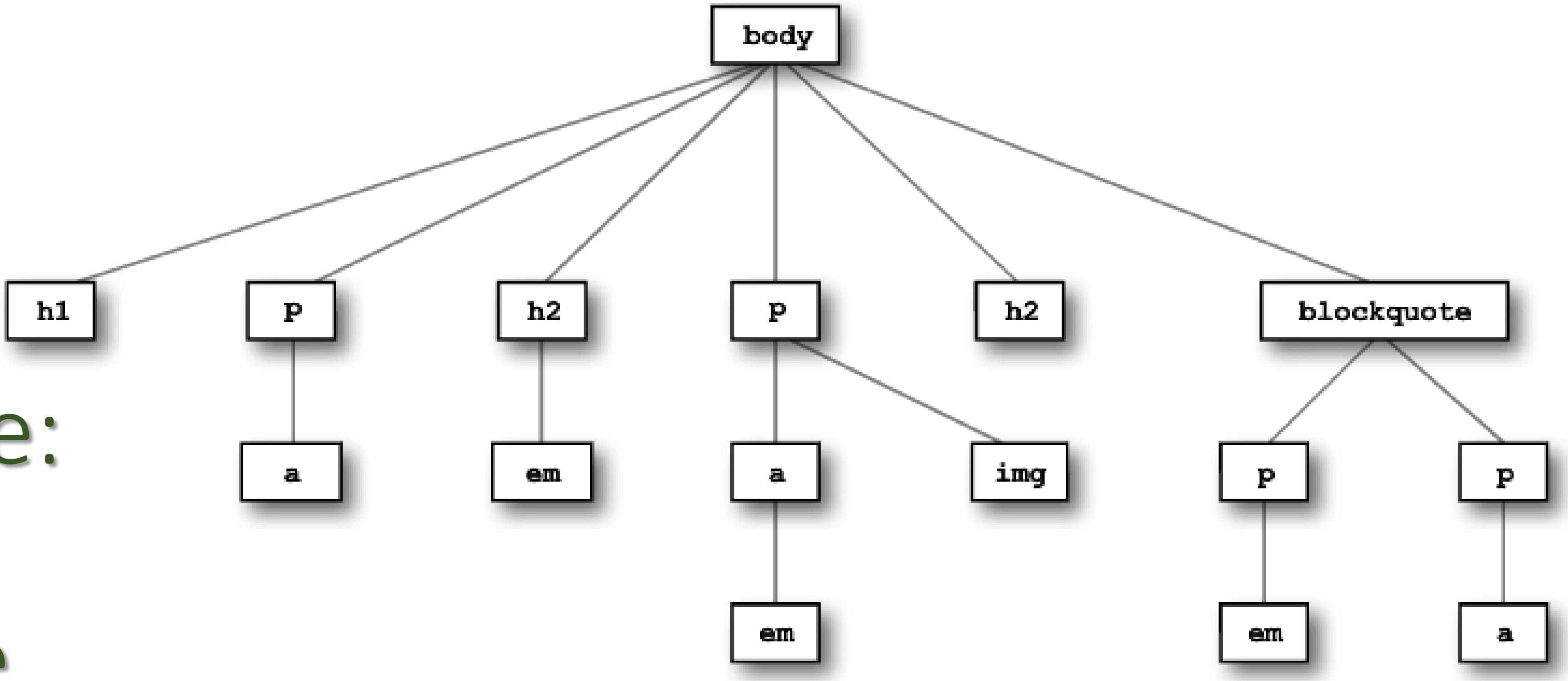
Save and reload. What color is the Green Tea Cooler paragraph now? ________

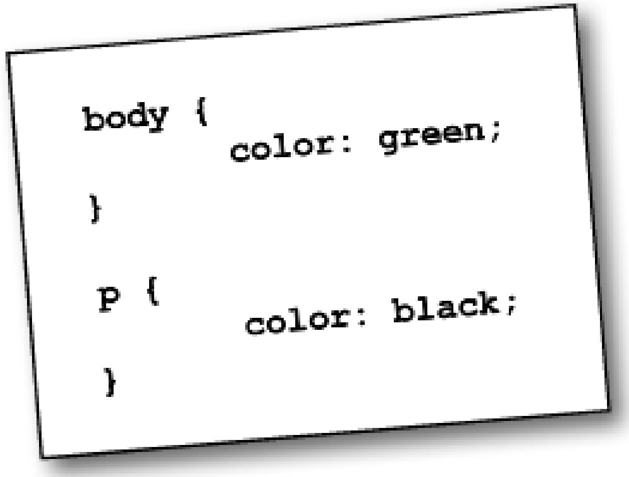
After you've finished, rewrite the green tea element to look like it did originally:

```
<p class="greentea">
```

Save and reload. What color is the Green Tea Cooler paragraph now? ________

Book exercise: Which elements are green?





```
body {
        color: green;
}

p {
        color: black;
}
```

Here's the CSS. Use this to determine which of the above elements hit the jackpot and get the green (color).

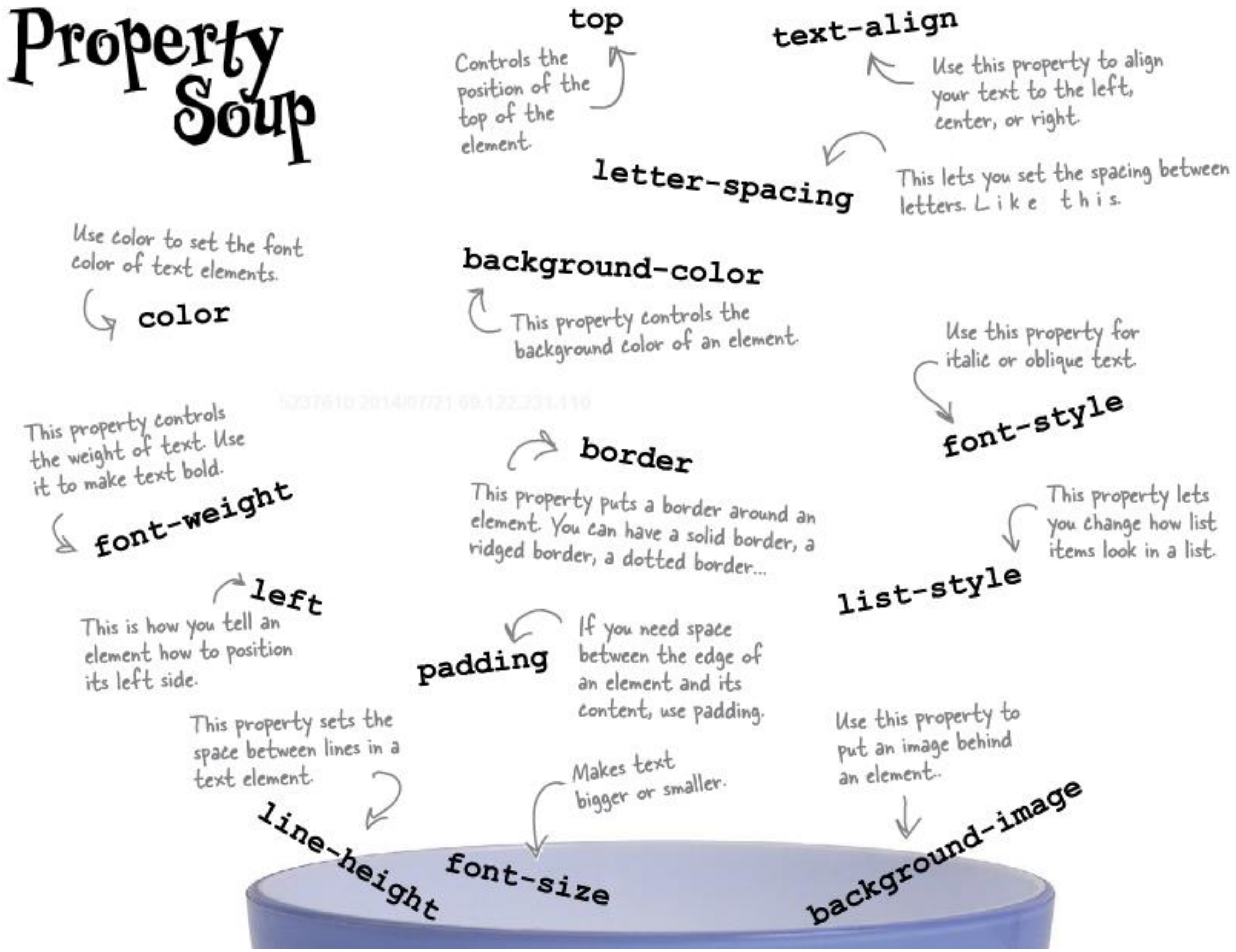
Book exercise: Finding errors in CSS

- Find all of the errors in the stylesheet style.css to the right.
- If you make an error in the CSS, usually all subsequent rules are ignored!
- There is a CSS validator:

http://jigsaw.w3.org/css-validator/

```
<style>
body {
    background-color: white
h1, {
    gray;
    font-family: sans-serif;
}
h2, p {
    color:
}
<em> {
    font-style: italic;
}
</style>
```

CSS Properties


Property Soup

top
Controls the position of the top of the element.

text-align
Use this property to align your text to the left, center, or right.

letter-spacing
This lets you set the spacing between letters. L i k e t h i s.

color
Use color to set the font color of text elements.

background-color
This property controls the background color of an element.

font-style
Use this property for italic or oblique text.

font-weight
This property controls the weight of text. Use it to make text bold.

border
This property puts a border around an element. You can have a solid border, a ridged border, a dotted border...

list-style
This property lets you change how list items look in a list.

left
This is how you tell an element how to position its left side.

padding
If you need space between the edge of an element and its content, use padding.

line-height
This property sets the space between lines in a text element.

font-size
Makes text bigger or smaller.

background-image
Use this property to put an image behind an element.

Notes

- This is primarily a summary of Chapter 7 of *Head First HTML and CSS*, 2nd Edition by Elisabeth Robson and Eric Freeman, 2012. It contains images, exercises, and code from the book.