



I hope that you'll appreciate my use of the waterfall both as a way to calm your frayed nerves and to provide you with a metaphor for the style sheets...

In this document, I'd like to step back a little and explain a little bit about how html and CSS work together, which should give you a better sense of what you're doing when you're creating pages and styling them. I'm borrowing freely from existing content on the Web and putting it in my own language here. We'll look at two small exercises at the end of the didactic piece

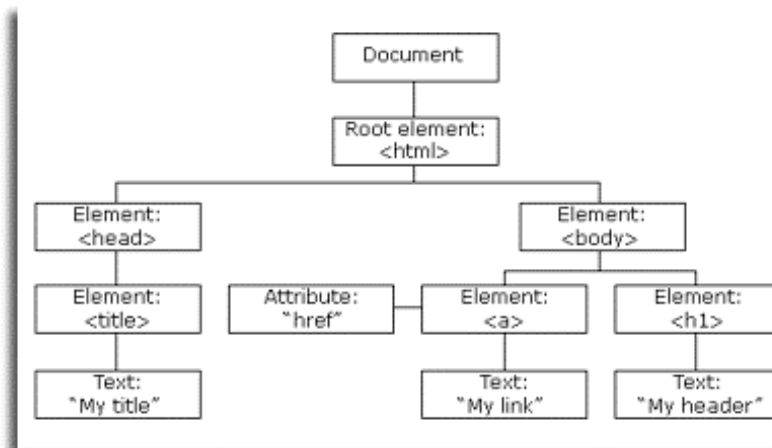
Elements

The best way to think of an element in html is as an invisible box that gets things put inside of it. *All html tags are elements*: that is, they are invisible boxes that have content placed inside of them. Because the box has its own name, the browser knows how to make the content appear on the screen. The graphic below, which I've nicked from Wikipedia, illustrates the breakdown of an element:



- The **element** is the invisible box (you don't "see" an element on an html page—you see its contents or you see something that describes the element, like a background color)
- The browser knows what kind of element this is because of the **tag** (in this case, it's a <p> tag, or paragraph tag)
 - Let's say the element looks just like this: <p>This is a paragraph.</p> When you use a tag that has no styling, then the browser will render the tag in the way that it's been programmed to do.
 - It's only when you modify the tag in some way (by giving it an **attribute**, as you see above) that the browser will make the tag look differently than it ordinarily would. Attributes describe the tag, as an adjective describes a noun.
- When we use style sheets, we are modifying the tag (or element) in some way. Basically what we're saying is, "Make the content in the invisible box appear *my way* rather than the way that the computer would ordinarily have it appear."
- There are 2 kinds of elements in html:
 - Block-level: these always appear on a new line (unless you give them a different attribute using CSS)—examples include <h1>, <h2> (and all of the <h> tags), <p>, and <div>
 - In-line: just as their name indicates, these elements can be on the same line as other elements—examples include (for italics), (for bold), and <a> (for hyperlinks)
- For more on this element concept, Wikipedia has a pretty nice entry at: http://en.wikipedia.org/wiki/HTML_element

The Document Tree



- The image above (<http://www.w3schools.com/html/dom/default.asp>) is a document tree based on the html Document Object Model, or DOM, which is basically a way of organizing elements in any document
 - Starting from the top of the tree, each of the boxes represented above contains the boxes that come beneath it
- If we were to look at the html code for the document tree above, it would probably look something like this:

```
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <a>My link</a>
    <h1>My header</h1>
  </body>
</html>
```

- Remember: all html tags are elements—that is, they are all boxes. So, when you look at the document tree, you realize that html documents are boxes that contain other boxes
 - The <html> tag is the **root element**—it's the element (or box) that contains all other elements (or boxes). Another way of saying this is that it has no *ancestors*, it only has *descendants* (if you were to put this in family tree terms, which is exactly what a document object model does).
 - There are only two elements that can go inside of the <html> element: the <head> element and the <body> element. Everything else that ever shows up on a web page goes inside of one of these elements
 - The <head> tag for any html page is the invisible part of the page—it contains metadata about the page, but is not visible
 - The <body> tag is where all of the visible content on an html page goes
- The document tree shows how elements are descendants of one another. That is, elements that are **children** (or descendants) of other elements are going to be affected by the properties of that **ancestor**
 - So, in our example above, whatever you do to the <body> tag (or element) is going to affect all of the tags (or elements) that are inside of

it because they are its descendants. For example, if you set the typeface of the <body> tag to be Verdana, then the typeface for every element inside of that will be Verdana unless you override it by using a style sheet

- This concept of ancestors and descendants is important to remember because as you create styles that add new attributes to elements, those attributes will affect the appearance of the descendant elements
- Maxdesign has a really nice tutorial on the html document tree at: http://css.maxdesign.com.au/selectutorial/document_tree.htm

Cascading order and inheritance

Cascading styles work with the document object model in order to define how things will look on the html page. Just as the elements (or tags) on a page influence the elements that are inside of them, style sheets also have an “order of precedence,” which works as follows (from lowest to highest priority):

1. *Browser default:* All browsers default to render elements (tags) according to a particular style, which is how they will appear on the screen unless they are overridden by styles higher up in the list
2. *Styles specified by the viewer to the browser:* You can specify the way in which your browser will render certain elements, overriding the default settings
3. *External style sheet:* An external style sheet can be linked to many html pages in order to style them uniformly
4. *Internal style sheet:* These are the styles that appear inside of the <head> tag and will style only the elements on one particular page
5. *In-line style:* You can apply a style to an individual element in-line. That is, you might choose to style just one particular instance of an element in one place while all the other remain as they are

Further, there are internal styling priorities for CSS:

1. **Element:** All elements default to style (“the look and feel”) in a certain way. So, for example, <h1> is bold, large, and on its own block (starts on a new line). Using CSS, you can create a new rule (and save it on an internal style sheet or an external style sheet) and redefine any html tag to make it appear as you choose
2. **Class:** A class is a way of overriding html tags when you want to change tags only in certain instances. **A class is essentially just like an ID, only you can use it as many times as you want on the same page.** For instance, let’s say (and this is arbitrary) you wanted every third instance of <h1> to be red and 300% of the regular text size on the page. Rather than redefine <h1> with a new rule, which will apply to every instance of <h1>, we can create a class and

apply it only to those instances of `<h1>` that we want. So, we could create a class called "redh1" (this is arbitrary—you can call classes whatever you want) and then apply it to only the `<h1>` that we're interested in at that moment. So, the code might then look something like this after it's applied:

```
<html>
  <head>
    <title>The red one</title>
    <style type="text/css">
      <!--
        .redh1 {
          font-size: 300%;
          color: #FF0000;
        }
      -->
    </style>
  </head>
  <body>
    <h1>this is h1 tag as it normally
    appears</h1>
    <h1 class="redh1">this is the h1 tag
    as it appears when we modify it with
    a class</h1>
  </body>
</html>
```

In the browser the above code would appear as follows:



- In the red box in the code above is the **internal style**: the class that you have created here will only work on this page. **Classes are always preceded by a “.” in the style sheet**
 - In the green box in the code above, we have applied the class “redh1” to one <h1> tag while leaving the other alone
3. **ID**: An ID is used to create a **unique element that appears ONLY ONCE on a page**. Whereas you can use classes as many times as you like, you can use an ID with a specific name only once per page. You can turn most html tags (or elements) into an ID. For example, let’s add a table with two cells to the page that we have created above. We are going to style the <h1> tags in each of the table cells differently, so we’ll give each cell a unique ID. Below is what the code will look like:

```
<html>
  <head>
    <title>The red one</title>
    <style type="text/css">
      <!--
        .redh1 {
          font-size: 300%;
          color: #FF0000;
        }
        1. #cellh1 {
          font-size: 140%;
          color: #FFFFFF;
          background-color: #3366CC;
        }
        2. #cell2 h1 {
          font-size: 110%;
          color: #FFFF00;
          background-color: #66CC66;
        }
      -->
    </style>
  </head>
  <body>
```

```
<h1>this is h1 tag as it normally appears</h1>
<h1 class="redh1">this is the h1 tag as it appears
when we modify it with a class</h1>
<table width="80%" border="0">
  <tr>
    3. <td id="cell1"><h1>This is h1 in the ID
      "cell1"</h1></td>
      <td id="cell2"><h1>This is h1 in the ID
      "cell2"</h1></td>
  </tr>
</table>
</body>
</html>
```

And below is what the code above will look like in a browser:



□ Here are some things to notice about the code:

1 & 2. In #1 and #2 in the code above: Note the two IDs in the internal style sheet (once again, because you are defining these styles on this page, they will only apply to this page). In the style sheet, IDs are always preceded by a “#” sign. In this style, we’ve created two ID’s “#cell1” and “#cell2.” Then, we got even more specific by adding the <h1> tag after each of the IDs. In effect, what we’ve said to the browser is, “Whenever you see some element that has the ID “cellx” associated with it, if it has an <h1> tag inside of it, then make the <h1> tag look this way.” All of the other <h1> tags on your html page will remain as they are because they are not

inside of one of those IDs—that’s what the image of the browser above is showing you. All of the text on that screen is content within an `<h1>` tag, but because those elements are modified (look at the code in the style sheet—the ID and the class are attributes {look back to page 1 if you need to be reminded of what this means} of the element, which means they are adjusting the look of the element according to how they have been defined) by different classes, IDs, and styled tags, they all look different.

3. In the code above, #3 shows you the ID being applied to each of the two table cells (table cells are `<td>` tags). The `<h1>` element (or tag) is a **descendant** of the ID “cellx” and is therefore being modified by it. The `<h1>` tag in each of those cells will only look that unique way when they are located within an element modified by the ID

This can be done with **any html element** so that you can become extremely specific about how you want certain elements to appear when they are contained within other elements

More on CSS

Following are some links to some good shortcut tips:

- World Wide Web Consortium tutorial on styling text with CSS: <http://www.w3schools.com/css/default.asp>
- Text appearance with CSS: <http://www.vordweb.co.uk/css/text-style-sheet.htm>
- General CSS: <http://lesliefranke.com/files/reference/csscheatsheet.html>