CSS cont.

October 5, Unit 4

Padding

- We can add borders around the elements of our pages
- To increase the space between the content and the border, use the *padding* property
- padding is shorthand for all of the padding properties:
  - padding-left
  - padding-right
  - padding-top
  - padding-bottom

Padding, cont.

- You can specify all 4 of the padding properties in one go
  - padding: 2cm 3cm 4cm 1cm;
    - (top, right, bottom, left)
- More complicated specifications are also possible
  - padding: 2cm 4cm 3cm;
  - Top has padding of 2cm, right and left of 4cm, and bottom of 3cm
- If you want equal padding for all 4 sides, simply specify a single number:
  - padding: 3cm;

Classes

- Let’s say that you need two types of paragraphs for your page
  - Perhaps main text and some sort of commentary or side notes
  - The commentary could be smaller, in a different font, or a different color
- But, since both the main text and the commentary are paragraphs, they’d both need to use the “p” selector
- We can use *classes* to differentiate between the two types of paragraphs

Classes, cont.

- With style sheets we can specify different *classes* for the same tag
- Classes are defined using the *class* attribute
  - Can be applied to any tag
- Can create any number of classes for a tag
- Saves us from having to write inline style information repeatedly

CSS Rules with Classes

- The selector for tags with classes has the form:
  - tag.class
- Example of a paragraph selector with a class:

```
p.sidenote {
   font-size: smaller;
   color: #FF33FF;
}
```

Using the Class Attribute

- So now we’ve defined a special class of paragraphs
  - The class is sidenote
- In the HTML, we can specify a paragraph with this class using the *class attribute*
- Ex:

<p class = “sidenote”> This is a sidenote…</p>

Selectors and Class Selectors

- Even if we have a special class of a tag, such as the **p.sidenote**, we can still specify css rules for the **p selector**
- Style information specified by the p selector will also apply to the sidenote class
- If the same property is defined both in the p, and the p.sidenote selectors with different values, the p.sidenote value will take precedence

Example with Multiple p Selectors

```
p {
    color: orange;
    font-family: serif;
}

p.sidenote {
    color: red;
    font-size: smaller;
}
```

```
<p class = “sidenote”>
    Here is some
    sidenote text </p>
```

Would display something like:

```
Here is some sidenote text
```

Example with Classes

- In class example

IDs

- *IDs* are like classes in that they allow a developer to specify a different category for the same tag
- Identifiers must be unique
  - They can be used once, and only once, on each page
- Can think of it like naming an element
- Instead of using the class selector, we'll use an *ID selector*
- To apply it to a tag, we'll use the *id* attribute

CSS Rules with Identifiers

- An id selector has the form:

  selector#identifier
- Example with an ordered list:

```
ol#contents   {
      font-family: sans-serif;
      list-style-type: upper-roman;
}
```

Using the id Attribute

- Now we have a specific identifier for an ordered list
  - Like a table of contents
- To use the CSS rule we created use the id attribute

```
<ol id = “contents”>
    <li>Introduction</li>
    <li>Middle</li>
    <li>Conclusion</li>
<ol/>
```

IDs as Anchors or Fragments

- Since IDs can only be used once per page, we can use them as *anchors* or *fragments*
- Fragments specify a specific location on the page
- Can be used to “jump” readers right to that section of the page
- Becomes part of a URL

Using Fragments

- Using the silly ordered list example, imagine that the introduction, middle, and conclusion are all on the same page
- Each is given an ID
- Instead of scrolling, we can turn those list items into links to jump to the right part of the page
- General form of the URL page.html#id
- Perhaps we'd have a link like:

```
<li><a href="mypage.html#conclusion">Conclusion</a></li>
```

Choosing Classes vs. IDs

- If the element appears (or could appear) more than once on a page
  - Use a class
- If the element will only appear once
  - Could use either
- If you want to use a fragment to jump to that part of your page
  - Use an identifier
- If you want to modify the element with JavaScript
  - Use an identifier
- Can use both a class and an identifier in the same tag
  - Perhaps simply using the identifier as a fragment

Using IDs Regularly

- If your page requires vertical scrolling, it’s a good idea to use id names
  - Allows users to link to relevant parts of your page
- Document with many headings
  - May be a good idea to use id names for your headings
    - Helps users linking to relevant parts
    - May allow you to use fragments in the future’
  - Coursepack suggests giving unique id names to your h2 tags

ID Example

- In Class Example

Specifying Color

- We already know how to specify color as a 6 digit hexadecimal number
  - #FF0000 : red
  - #FFFFFF : white
  - #2AF3C4 : some bright blue-green color
- And how to specify a color by its name
  - red, white, black, etc.
- With style sheets, though, we can specify colors in a number of other ways as well

Shorthand Hex

- For web-safe colors they all use duplicate digits for each component
  - ex. #FF0000, #33CCDD, #6699AA
- With style sheets we can shorten the hex color to 3 digits by reducing each component to a single digit:
  - ex. #F00, #3CD, #69A
- We can specify non-web-safe colors as well
  - #700 = #770000
  - #3C2 = #33CC22

Specifying Colors Using RGB

- So far you've had to convert from rgb to hex to use colors with html
- CSS allows you to specify colors using their rgb values
- When specifying rgb we use the value rgb (rValue, gValue, bValue)
- ex. background-color: rgb(120,19,245);

Transparent

- One of the reasons we want to use style sheets is to make it easier to change our websites
- Let’s say that all of our paragraphs use a white background and the background color on the site is blue
- But, we want to create a special class of paragraphs without the white background
- Can use the *transparent* value

```
p.plainParagraph  {
        background-color: transparent;
}
```

Setting Colors

- If you specify any colors in your site, specify them all
- You should specify:
  - background-color (obvious, I hope)
  - color (font color)
  - a:link (link)
  - a:visited (visited link)
  - a:active (active link)
- Why?
  - If you only specify some, a user’s default browser settings could render your site unreadable

Hyperlink Colors

- It is quite easy to change the colors of your hyperlinks
- Often the blue/purple default clashes with your color scheme
- Or if using a blue background, become unreadable
- It’s okay to change your link colors provided they still look like links

Changing Link Colors

- a:link – color of the link
- a:visited – color of visited link
- a:hover – mouseover color
- a:active – color of the active link

- Should specify these in this order if using *hover*
- *hover* must come after *visited* and *link*
- *active* must come after *hover*

Example with Changing Link Colors

```
body {
            color: #999999;
            background-color: white;
   }
a:link { color: #FF33CC}
a:visited {color: rgb(120,20, 0)};
a:hover {color: red};
a:active {color: #A31200};
```

In Class Example

- Changing all colors