Color and Images

Lecture 16
Colors in CSS

- Use: fonts, borders, backgrounds
- Provides semantic signal:
  - **Green** – go, success, complete, solution
  - **Red** – stop, failure, incomplete, problem
  - **Yellow** – yield, warning, attention
- Helps to set mood/emotion/tone:
  - **Bright** – cheerful, playful, positive
  - **Dark** – somber, serious, negative
  - **Warm** – energetic, alert, active
  - **Cool** – calm, tranquil, peaceful
Elementary Color Theory

- Combination of
  - Physics (eg wavelengths in nm)
  - Biology (perception of “red” vs “yellow”)
- Visible spectrum: 390-700nm
  - Nonspectral colors (eg pink, brown, white) require presence of *multiple* wavelengths
Color Perception

- Human eyes have 3 types of cones
  - Respond to different wavelengths (LMS)
- Color = cone response
Metamerism

- Different (continuous) spectra that stimulate our eyes in identical ways
  - Consequence: Different spectra with indistinguishable (to humans) color

- Example: white
  - Spectrum 1: all wavelengths equally present
  - Spectrum 2: three wavelengths present, stimulating LMS cones equally

- Consequence: continuous spectrum can be projected down to 3 dimensions
Color Mixing

- There are two ways to combine colors
  1. Subtractive: Color is a *filter*
     - Mixing = filter out both
     - Used for printing (& dyes, paints, gels)
  2. Additive: Color is a *light source*
     - Mixing = sum
     - Used for monitors
CMYK: Subtractive Color Mixing

- Filters transmit different *spectra*
  - Mixture transmits the *product* of both
  - Mix all three primaries = black
- Primary colors: cyan, magenta, yellow
  - Black (K) added for quality and cost
  - Traditional set (RYB) popular for painting

Primary yellow (transmits R & G) (absorb B)
Colors as Filters

Yellow: Filters out (only) blue

Rosi et al., Euro. J. of Physics, 37(6), 2016
Additive Color Mixing: RGB Cube

The primary colors are red, blue, and green. The secondary colors are cyan, magenta, and yellow. The primary colors can be mixed to produce all other colors. The RGB cube is a useful tool for understanding color mixing and color theory. The RGB values for white and black are #fff and #000, respectively. (Image credit: http://www.flickr.com/photos/ethanhein/3103830956/)
HSL Color Wheel (100% Sat.)

HSL Grid for Red (ie 0,x,y)

- (0,75%,88%)
- (0,100%,50%)
- (0,0%,25%)
CSS Color Values

- Keywords: case-insensitive identifiers
  - red, navy, firebrick, chocolate
- RGB as decimal (0-255), percentage, or hex
  - `rgb (255,0,0) /*pure red*/`
  - `rgb (100%,0%,0%)`
  - `#ff0000`
  - `#f00 /*expand by repeating digit*/`
- HSL (Hue, Saturation, Light)
  - Hue (0-360) is angle on color wheel: 0 is red, 120 green, 240 blue
  - Saturation & light are both %'s
  - `hsl (0,100%,50%) /*full bright red*/`
- Alpha channel (transparency): 1 is opaque!
  - `rgba (255,0,0,0.5)`
  - `hsla (0,100%,50%,1)`
### Color Keywords

<table>
<thead>
<tr>
<th>aliceblue</th>
<th>antiquewhite</th>
<th>aqua</th>
<th>aquamarine</th>
<th>azure</th>
<th>beige</th>
</tr>
</thead>
<tbody>
<tr>
<td>bisque</td>
<td>blanchedalmond</td>
<td>chartreuse</td>
<td>chocolate</td>
<td>coral</td>
<td>brown</td>
</tr>
<tr>
<td>burlywood</td>
<td>cadetblue</td>
<td>cyan</td>
<td>darkblue</td>
<td>darkcyan</td>
<td>darkgoldenrod</td>
</tr>
<tr>
<td>darkgray</td>
<td>darkgreen</td>
<td>darkkhaki</td>
<td>darkmagenta</td>
<td>darkolivegreen</td>
<td>darkorange</td>
</tr>
<tr>
<td>darkorchid</td>
<td>darkred</td>
<td>darksalmon</td>
<td>darkseagreen</td>
<td>darkslateblue</td>
<td>darkslategray</td>
</tr>
<tr>
<td>darkturquoise</td>
<td>darkviolet</td>
<td>deepink</td>
<td>deepskyblue</td>
<td>dimgray</td>
<td>dodgerblue</td>
</tr>
<tr>
<td>firebrick</td>
<td>floralwhite</td>
<td>forestgreen</td>
<td>fuchsia</td>
<td>gainsboro</td>
<td>ghostwhite</td>
</tr>
<tr>
<td>gold</td>
<td>goldenrod</td>
<td>gray</td>
<td>green</td>
<td>greenyellow</td>
<td>honeydew</td>
</tr>
<tr>
<td>hotpink</td>
<td>indianred</td>
<td>indigo</td>
<td>ivory</td>
<td>khaki</td>
<td>lavender</td>
</tr>
<tr>
<td>lavenderblush</td>
<td>lawngreen</td>
<td>lemonchiffon</td>
<td>lightblue</td>
<td>lightcyan</td>
<td>lightcyan</td>
</tr>
<tr>
<td>lightgoldenrodyellow</td>
<td>lightgray</td>
<td>lightgreen</td>
<td>lightpink</td>
<td>lightsalmon</td>
<td>lightsaegreen</td>
</tr>
<tr>
<td>lightskyblue</td>
<td>lightslategray</td>
<td>lightsteelblue</td>
<td>lightyellow</td>
<td>lime</td>
<td>limegreen</td>
</tr>
<tr>
<td>linen</td>
<td>magenta</td>
<td>maroon</td>
<td>mediumaquamarine</td>
<td>mediumblue</td>
<td>mediummorchid</td>
</tr>
<tr>
<td>mediumpurple</td>
<td>mediumseagreen</td>
<td>mediumslateblue</td>
<td>mediumspringgreen</td>
<td>mediumturquoise</td>
<td>mediumvioletred</td>
</tr>
<tr>
<td>midnightblue</td>
<td>mintcream</td>
<td>mistyrose</td>
<td>moccasin</td>
<td>navajowhite</td>
<td>navy</td>
</tr>
<tr>
<td>oldlace</td>
<td>olive</td>
<td>olivedrab</td>
<td>orange</td>
<td>orangered</td>
<td>orchid</td>
</tr>
<tr>
<td>palegoldenrod</td>
<td>palegreen</td>
<td>paleturquoise</td>
<td>palevioletred</td>
<td>papayawhip</td>
<td>peachfuzz</td>
</tr>
<tr>
<td>peru</td>
<td>pink</td>
<td>plum</td>
<td>powderblue</td>
<td>purple</td>
<td>rebeccapurple</td>
</tr>
<tr>
<td>red</td>
<td>rosybrown</td>
<td>royalblue</td>
<td>saddlebrown</td>
<td>salmon</td>
<td>sandybrown</td>
</tr>
<tr>
<td>seagreen</td>
<td>seashell</td>
<td>sienna</td>
<td>silver</td>
<td>skyblue</td>
<td>slateblue</td>
</tr>
<tr>
<td>slategray</td>
<td>snow</td>
<td>springgreen</td>
<td>steelblue</td>
<td>tan</td>
<td>teal</td>
</tr>
<tr>
<td>thistle</td>
<td>tomato</td>
<td>turquoise</td>
<td>violet</td>
<td>wheat</td>
<td>white</td>
</tr>
<tr>
<td>white</td>
<td>yellow</td>
<td>yellowgreen</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Color Depth

- "Depth" = # of bits in representation
  - 8 bits → 256 different colors
  - 24 bits → 16,777,216 different colors (eg 8 bits each for r,g,b)
- Alpha may be (incorrectly) included
  - rgba is a point in 4-dimensional space
- Problem: image color depth > display color depth
  - Quantization: each pixel gets closest available color (leads to banding)
  - Dithering: add noise, which looks better!
Quantization of Continuous Function

The diagram illustrates the quantization process of a continuous analog signal. The analog voltage is converted into a binary equivalent, representing the discrete values generated by an A/D converter at each sampling point. The continuous analog signal is approximated by a staircase function, which reflects the quantized values. The figure also indicates the time axis, showing the progression of sampling points.
Quantization vs Dithering

original

quantized

dithered
Quantization vs Dithering

Original Image

GIF without dithering

GIF with dithering

HTML `<img>` Tag Attributes

- `src`: location of image file
- `width`, `height`:
  - Area in *window* to reserve for image
  - Image is *scaled* to those dimensions
  - These attributes affect browser flow, regardless of when/if image is displayed
- `alt`: text to show if graphic can not be displayed or seen (i.e., alternative)
- `title`: text to *augment* displayed graphic (e.g., tooltip)
Image Representation

- Raster vs vector graphics
  - Raster: stored pixel-by-pixel
  - Vector: mathematical description

- Compression of raster images
  - Lossy: better compression, lower quality image
  - Lossless: largest file size, best quality
Major Formats

- **GIF**
  - Raster graphics, lossy compression (oldest)
  - 8 bit, basic transparency (on/off)
  - Frame-based animation (groan)
  - Good for small file size, crisp lines, logos

- **JPEG**
  - Raster, lossy compression
  - 24 bit, no transparency
  - Good for photos, gradual gradients

- **PNG**
  - Raster, lossless (but still often good) compression
  - Variable depth, full alpha transparency
  - Good replacement for GIF (but no animation)

- **SVG**
  - Vector graphics (newest)
  - Good for crisp lines, simple logos, graphs
Scaling Images

- Vector graphics scale perfectly
- Raster images should be *pre-scaled*
  - Width (height) attributes of image tag should match actual width (height) of image
  - Why?
Alternative: CSS

```
.deleteButton {
  background: 
    -webkit-linear-gradient(top, 
      #be6868 0%, #941b17 50%, 
      #880d07 50%, #be483c 100%);
  border: 3px solid #000;
  color: #fff;
  cursor: pointer;
  font-size: 15pt;
  padding: 10px 34px;
  text-shadow: 0 -1px 0 #000;
  border-radius: 13px;
  box-shadow: 0 1px 0 #454545;
}
```