Colors and Backgrounds

List of Topics

- Color names in CSS
- RGB color values
- Foreground and background colors
- Pseudo-class/element selectors
- Tiling background images
- Color gradients

Color Names

- The CSS specification defines a set of standard color names
  - 16 in CSS 2
  - 17 in CSS 2.1 (orange was added)
  - 130 in CSS 3
- See http://www.learningwebdesign.com/colornames.html for a complete list

RGB Color Values

- Much less limited than color names
  - covers millions of colors
- The RGB color model works by combining red, green, and blue light
  - each color channel varies from 0 (none) to 255 (full), or by percent
  - (255, 255, 255) = pure white

Specifying RGB color

- List each value on a scale of 0–255:
  `color: rgb(200, 178, 230);`
- List colors as percentage values:
  `color: rgb(78%, 70%, 90%);`
- Use 6-digit hexadecimal (base 16) values:
  `color: #C8B2E6`

The Joy of Hex

- Hexadecimal (base 16) has 16 digits
  - 0–9, A–F
- Very compact representation for numbers
- Use division to convert base 10 to base 16
  - Divide # by 16: quotient is first digit, remainder is second digit

RGBa Color

- RGBa color adds an "alpha" (transparency) value
  - 0 = fully transparent, 1 = fully opaque
- Use four values in color specification:
  `color: rgba(0, 0, 0, .1);`
- Doesn't work in IE version 8 and earlier

Foreground Color

- An element's *foreground* consists of its text and border
  - Use the `color` property:

```
color: #508C19;
```

Background Color

- Background color fills the canvas, the content area, and any padding (space) around the element
- Use the `background-color` property
- Use this tag in the `body` element to color the entire page

Opacity

- CSS 3 adds an `opacity` property (so you don't have to use RGBa color):

```
opacity: .5;
```

- `opacity` takes a value between 0 and 1
- This doesn't work with IE 8 and earlier

Pseudo-Class Selectors

- Some elements with multiple states belong to the same "class"
  - e.g., hyperlinks, hover state, un/checked
- The browser keeps track of the state of these elements
- We can style them using the : character

Link Pseudo-Classes

- Links are a dynamic pseudo-class
  - their value varies based on user actions rather than the page markup
- `:link` applies a style to unclicked links
- `:visited` applies a style to visited links
- `a:link { color: maroon; }`

User Actions

- :focus applies when an element is selected and ready for input
- :hover applies when the mouse pointer is over the element
- :active applies when the element is in the process of being clicked or tapped

Quick Test

- What will the following CSS style rules do?

```
a:hover { color: maroon;
  background-color: yellow; }

a:active { color: red;
    background-color: blue; }
```

Combining Pseudo-classes

- When you use all five link state styles, order matters!
- Proper order:
  :link, :visited, :focus, :hover, :active
- Mnemonic: "LoVe For Hell's Angels"

```
a { text-decoration: none; } /* turn off underlines */
a:link { color: maroon; }
a:visited { color: gray; }
a:focus { color: maroon; background-color: #ffd9d9;}
a:hover {color:maroon; background-color: #ffd9d9;}
a:active {color:red; background-color: #ffd9d9;}
```

Other Pseudo-Classes

- There are 18 more pseudo-class selectors
  - structural, user interface, and more
- We won't go into them now
- See Appendix B of the Robbins book for more detail

Pseudo-Element Selectors

- Four pseudo-element selectors
  - insert fictional elements into document
- Use :: or : to indicate these
- first letter/line, before, and after

First Letter and Line

- :first-line
  - applies a (limited) style rule to the first line of the specified element
- :first-letter
  - applies a limited style rule to first letter of the specified element

```
p:first-line { color: maroon; }
```

Before and After

- Insert *generated content* before or after an element without actually adding it to the source document
  - e.g., automatic counters, code to display URLs next to links when printing

```
p:before {font-weight: bold;
          color: purple;
 content: "Once upon a time: ";}
```

Attribute Selectors

- Target elements based on their attributes
  - this avoids lots of `class` or `id` markup
- *element*[*attribute*] — targets any element with a particular attribute

```
img[title] { border: 3px
                     solid; }
```

More Attribute Selectors

- element[attribute="exact value"]
  - exact attribute value selector
  - values are case-sensitive in IE 7
- element[attribute~="value"]
  - partial-attribute value selector
  - targets attributes that contain "value"

Still More Selectors

- element[attribute|="value"]
  - hyphen-separated attribute value selector
  - targets hyphen-separated values ("en-us")
- element[attribute^="first part of value"]
  - beginning substring attribute value selector (new in CSS 3)

Just Two More Selectors...

- element[attribute$="last part of value"]
  - ending substring attribute value selector
- element[attribute*="any part of value"]
  - arbitrary substring attribute value selector — looks for provided text in any part of the attribute's value

Background Images

- The `background-image` property adds a background image from a specified URL
  - URL is relative to the location of the CSS style rule
  - Images that don't fit are tiled by default

    body {background-image: url(star.gif); }

Tiling Direction

- By default, images tile left and right, up and down
- We can change this using the `background-repeat` property
  - possible values: repeat, repeat-x, repeat-y, no-repeat

Background-position

- This property specifies the position of the *origin image* in the background
  - values: *length measurement*, *percentage*, left, center, right, top, bottom
  - keyword values are usually used in pairs (e.g., left bottom); missing keywords are assumed to be `center`

Background Attachment

- Use `background-attachment` to keep the background fixed in position, even when scrolling
  - values: `scroll`, `fixed`

Background Shorthand

- Use the background property to specify all styles in one declaration

```
body { background: white
       url(arlo.png) no-repeat
       right top fixed; }
```

Multiple Backgrounds

- Use comma-separated lists to apply multiple values for background-image
  - Be sure to use comma-separated lists for other background property values as well

```
body {
       background:
            url(image1.png) left top no-repeat;
            url(image2.png) center center no-repeat;
            url(image3.png) right bottom no-repeat;
     }
```

Color Gradients

- Can be applied anywhere an image may be applied: background-image, border-image, and list-style-image
- Two types:
  1. linear gradients
  2. radial gradients

Linear Gradients

- `linear-gradient()` provides the angle of the line and one or more *color stops* (points of pure color)
- Angle is specified in degrees (`0deg` points upward, and increases clockwise) or using keywords: to top, to right, to bottom, to left
- e.g., linear-gradient(to top, yellow, green);

Radial Gradients

- Radiate out from a point in a circle
- radial-gradient notation describes:
  - the shape (circle or ellipse)
  - position of the center of the gradient
  - size (radius length or keyword)
    - closest-side, farthest-side, closest-corner, farthest-corner, cover, contain

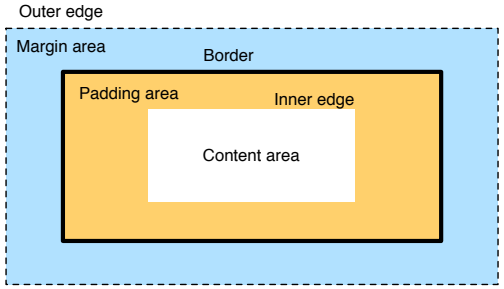
Radial Gradient Example

```
body {
        background-image:
              radial-gradient
              (center contain
               yellow green);
      }
```

Working with Boxes

The Element Box

- Every element in a document generates a rectangular *element box:*



Specifying Box Dimensions

- Calculated automatically by default
- We can also specify `width` and `height`
  - can add `padding` and `margin` as well
- Two ways to do this:
  - content box sizing (default)
  - `box-sizing` (applies to *border box*)

Border-Box Sizing

- Must set `box-sizing` to `border-box` in the style sheet
- Doesn't work directly in many browsers
  - Safari, Chrome, and Firefox need `-webkit-` or `-moz-` prefixes:

```
-moz-box-sizing: border-box;
```

Padding

- Can add padding to all sides, or just one side
  - `padding-top`, `padding-right`, `padding-bottom`, `padding-left`, or just `padding`
- Use TRouBLe mnemonic here as well

Borders

- Available border styles: none, dotted, dashed, solid, double, groove, ridge, inset, and outset
- Can be applied one side at a time: border-top-style, border-right-style, border-bottom-style, border-left-style

Border Width

- Use `border-width` (or side variants) to specify the thickness of a border
  - e.g., border-top-width, border-left-width
- Must include `border-style` to make border visible (defaults to `none`)

Border Color

- Use `border-color` or side variants (e.g., `border-top-color`) to set the color of a border, or to make it transparent
  - Transparent borders are useful for rollover (`:hover`) effects — they maintain the space where the border will appear when mouse is not over the element

Rounded Corners

- Use border-radius for rounded corners
  - takes a length measurement or percentage as its value
- `border-top-right-radius 100px 50px;`
- Can also pick out specific corners
- Use a slash to separate horizontal and vertical radii when using shorthand version

Margins

- Use `margin-top`, `margin-right`, etc. to add space on the outside of the border
- Note: percentages are calculated based on the *width* of the parent element
  - if width changes, so will the margin
- Use the `auto` value to automatically calculate the margin necessary to fit or fill the available space

Margin Behavior

- Top and bottom margins of adjacent neighbors *collapse*; only largest value is used
- Top/bottom margins on inline elements only apply when left and right margins are set

Drop Shadows for Boxes

- Use `box-shadow` just as you would use `text-shadow`:

  hOffset vOffset blur spread color (inset)

  - `inset` makes the shadow appear *inside*
- Like `text-shadow`, you can apply multiple box shadows

Display Roles

- We already know about block and inline elements
- The `display` property can be used to change the way an element behaves
  - e.g., display a `<li>` element inline instead of as a block element
  - `display: none` hides an element

Display Options

- inline
- block
- list-item
- inline-block
- table
- inline-table
- table-column
- table-cell
- table-caption
- none

Display Example

- Turn a list into a horizontal navigation bar:

  ul.navigation li { display: inline; }
- Display anchor inline elements as block elements for width and height:

  ul.navigation li a { display: block; }

Page Layout with CSS

Normal Flow

- Text elements are laid out top to bottom in the order they appear in the source, and left to right
- Resizing the window causes block elements to expand or contract and inline elements to reflow
- Objects affect the layout of other objects around them

Floating

- The `float` property moves an element to the far left or right, so other content wraps around it
  - Use margin to add spacing
  - img { float: right; margin: 10px; }

Float Notes

- A floated element is like an island in a stream
  - remaining elements flow around floated stuff
- Floats stay in the content area of the containing element
- Margins are maintained

Floating Inline Elements

- Always provide a width for floated text elements
  - without this, the content area of the box expands to its widest width
- Floated inline elements behave like blocks
  - margins exist on all four sides
- Margins *DON'T* collapse on floated elements

Floating Block Elements

- Other blocks wrap around floated blocks
- Must provide a width for floated blocks
- Elements do not float higher than their reference in the source
  - To float an element at the top of the page, it must appear first in the source

Clearing Floated Elements

- The `clear` property prevents an element from flowing around (appearing next to) a floated element:
  - Apply this to the following element, not the floated element
  - values: left, right, both, none
    - specify on which side to clear floats

A Navigation Bar

```
<ul>
<li><a href="#">serif</a></li>
<li><a href="#">sans-serif</a></li>
<li><a href="#">script</a></li>
<li><a href="#">display</a></li>
<li><a href="#">dingbats</a></li>
</ul>
```

Navigation Bar CSS

```
ul { list-style-type: none;
     margin: 0;
     padding: 0; }

ul li { float: left; }

ul li a { display: block; }
```

Creating Columns

- Three ways to make a 2-column layout:
  1. Float one `div` and add a wide margin on the side of the text element that wraps around it
  2. Float both `div`s to the left or right
  3. Float one `div` to the left, and one to the right

Column Caveats

- Dependent on the order of the elements in the source document
  - floated element must appear *before* wrapped content
- Every float needs to have a specified width

```
#products {
            background-color: #FFFFFF;
            line-height: 1e.5em;
            padding: 1em 2%;
            border: double #FFBC53;
            margin: 0 2% 1em;
            clear: both;
            float: left;
            width: 55%;
          }
```

Positioning Basics

- Elements can be positioned relative to where they would normally appear in the flow
- Elements can be placed at a specific location on the page
- Elements can be positioned relative to the browser window (viewport)

Positioning Types

- Use the `position` property
  - static: normal positioning scheme
  - relative: moves box relative to its original position in the flow (space is preserved)
  - absolute: removes element from flow and positions it with respect to window
  - fixed: element always stays in one place

Position Offsets

- top, right, bottom, left
- Value determines distance the element should be moved *away* from that edge

Containing Blocks

- If positioned element is *not* contained in another positioned element, it will be placed relative to the *initial containing block* (created by the `<html>` element)
- If element has an ancestor with a relative, absolute, or fixed position, it will be positioned relative to *that* element

Stacking Order

- By default, elements stack in the order they appear in the source document
- Use `z-index` property to change order
  - value can be positive or negative
  - the higher the value, the higher the element will appear in the stack