CSS Techniques

CSE/ISE 102: Introduction to Web Design and Programming
Stony Brook University

Topics We'll Cover

- CSS Reset
- Image Replacement
- CSS Sprites
- Styling Tables
- Basic Responsive Web Design
- Web Fonts

Reference: Robbins, Chapter 18

CSS Reset

A Clean Slate

- Remember that browsers have their own built-in stylesheets (*user-agent* style sheets)
- These style sheets' settings vary from browser to browser, and can affect your own styles through inheritance

Starting Fresh

- A ***CSS reset*** is a collection of style rules that override all user agent styles and provide a neutral starting point
  - One example is Eric Meyer's "Reset CSS" tool, available at http://meyerweb.com/eric/tools/css/reset/
  - Just place a copy of these styles at the start of your own style sheet

Image Replacement

When Text Isn't Enough…

- Before Web fonts, Web designers needed to fall back to images when they wanted text in a font fancier than Times or Helvetica
- What if a Web font isn't sufficient for your needs?
  - e.g., a company logo or icon-based links

Wholesale Replacement

- Completely replacing text with an image isn't a great idea
  - it affects accessibility and usability
  - it hurts search engines' ability to index your page

A CSS-Based Solution

- Use an image as the element's background
- Keep the text, but use CSS to shift it out of the way so that it's invisible to the user
  - Screen readers and search engines will still see it

The Kellum Technique

- Uses `text-indent` to push the text content all the way to the right and out of sight

```
h1#logo {
  /* image dimensions */
  width: 450px;
  height: 80px;
  background: url(image.png) no-repeat;
  text-indent: 100%;    /* shift text */
  white-space: nowrap; /* hide long text */
  /* hide anything that falls outside the h1
     element box */
  overflow: hidden;
}
```

CSS Sprites

Improving Performance

- Every external element in a Web page (stylesheet, images, etc.) generates a separate HTTP request
- If we can reduce the number of these requests, we can improve the performance of a Web site
- One way to do this is to combine images into a single larger image
  - display small parts of the large ***sprite*** image

Sample Markup:

```
<ul>
  <li><a href="" class="hide twitter">Twitter</a></li>
  <li><a href="" class="hide fb">Facebook</a></li>
  <li><a href="" class="hide gplus">Google+</a></li>
  <li><a href="" class="hide linkedin">LinkedIn</a></li>
</ul>
```

CSS Markup:

```
.hide {
  text-indent: 100%;
  white-space: nowrap;
  overflow: hidden;
}

li a {
  width: 29px;
  height: 18px;
  background-image: url(social.png);
}

li a.twitter { background-position: 0 0; }
li a.fb { background-position: 0 -20px; }
li a.gplus { background-position: 0 -40px; }
li a.linkedin { background-position: 0 -60px; }
```

Styling Tables

Table Styling

- Most table styling can be done with CSS properties that we've already covered
- There are a few additional table-centered CSS properties that can be helpful
  - Specifically, these deal with borders

Border Properties

- `border-spacing:` *length length*
  - specifies how much horizontal and vertical space to insert between the borders of adjacent cells
- `border-collapse: separate | collapse`
  - borders of adjacent cells "collapse" and only one border is drawn (intervening space is removed)

Separated Borders Example

```
table {
  border-collapse: separate;
  border-spacing: 15px 5px;
  border: none;
}

td {
  border: 2px solid purple;
}
```

Collapsed Borders Example

```
table {
  border-collapse: collapse;
  border: none;
}

td {
  border: 2px solid purple;
}
```

Empty Cells

- You can hide the borders of empty cells in tables with separated borders
  - Empty cells may contain carriage returns and spaces only (no text, images, or non-breaking spaces)
- `empty-cells: show | hide`

Responsive Web Design

Responsive Web Design

- Uses CSS to adapt a page's layout based on the screen size of the viewing device
- For example, one-column layout for a smartphone, wider margins on a tablet, and multiple columns for a desktop/laptop (or landscape tablet)

Core Components

- Fluid layout
- Flexible images
- CSS media queries
- Also includes the viewport `meta` element

Viewports

- Mobile browsers render Web pages on a canvas called the ***viewport***, and then shrink that viewport down to fit the device screen width
  - e.g., iPhones use a 980px viewport shrunk down to 320px
- As developers, we can control the size of the viewport:

```
<meta name="viewport" content="width=device-width,
initial-scale=1">
```

Fluid Layouts

- Use percentage values rather than fixed widths for elements, so elements resize proportionally
- It isn't possible to account for every device's screen dimensions, so Web designers generally target a few major classes
  - Fluid layouts handle the in-between sizes

Making Images Flexible

- This is easy:

```
img {
    max-width: 100%;
}
```

- This causes images to scale proportionally to the width of their container
- Just make sure your `<img>` tags don't have `width` or `height` attributes, or it won't work correctly!

Media Queries

- Allow us to deliver different styles based on the media type: `print, speech, handheld, braille, projection, screen, tty, tv` (or `all`)
- Media queries can also evaluate specific device features, like viewport width, screen width, orientation, or screen resolution

Features You Can Query

`width`	`orientation`	`color`	`grid`
`height`	`aspect-ratio`	`color-index`	
`device-width`	`device-aspect-ratio`	`monochrome`	
`device-height`	`resolution`	`scan`	

Media Query Format

- `@media` *target-media-type* [`and` (media feature and value being tested)]*

```
@media screen and (min-width: 480px;)
{
    /* style rules for devices that pass
this test */
}

@media screen and (max-width: 700px) and
(orientation: landscape;)
{ … }
```

Media Queries in the Document Head

- We can also use media queries to conditionally load CSS stylesheets based on media properties
- Use the `media` attribute in the `link` element:

```
<link rel="stylesheet"
href="2column-styles.css"
media="screen and (min-width: 780px)">
```

A Mobile-First Strategy

- Use a form of progressive enhancement:
  - order your styles from smallest device to largest/ most capable device
- Mobile-first media queries tend to begin with the `min-` prefix
- Baseline styles come first, followed by small device styles, then enhanced styles for larger browsers

Tricky Problems

- How do we choose breakpoints?
  - one option is to choose based on pixel dimensions of popular devices
- How do we make images responsive?
  - don't want to load hi-res images on slow mobile connections
- One size doesn't fit all
  - JavaScript can be used to load content conditionally
- Other limitations
  - Sometimes, it may be better to build separate mobile and desktop sites

Web Fonts

Web Font Options

- There are two ways to provide specific fonts to your users:
  - Host your own Web fonts
  - Use a font embedding service

Self-Hosting Steps

- Find a font
  - be careful about licensing for Web usage!
- Save the font in multiple formats
  - one way is to use Font Squirrel's @font-face Generator
- Upload your font(s) to your server
  - they usually go in the same directory as your CSS files

Web Font Code

```
@font-face {
font-family: 'Font_name';
src: url('myfont.eot') format('embedded-opentype'),
url('myfont.woff') format('woff'),
url('myfont.ttf') format('truetype'),
url('myfont.svg') format('svg');
}


p { font-family: Font_name;  }
```

Use a Font Embedding Service

- These companies/services sometimes charge money, but they handle all of the preceding hassle for you
  - Google Web Fonts, Adobe Typekit, fonts.com