19

MORE CSS TECHNIQUES

OVERVIEW

- Styling forms
- Style properties for tables
- CSS reset and normalizer
- Image replacement techniques
- CSS feature detection

Styling Forms

Use standard color, background, font, and box properties. Flexbox is a good tool for making the form adapt to available viewport space.

- **Text inputs**
  Change the appearance of the box with box properties. Style the text inside the box with color and various font properties.

- **`textarea`**
  Change font and line-height of text field. Prevent resizing with `resize: none;`.

- **Buttons** (submit, reset, button)
  Use box and font properties. Note that the default is border-box sizing with padding applied.

Styling Forms (cont'd)

- **Radio and checkbox buttons**
  Best practice is to leave them alone (or substitute your own JavaScript custom buttons).
- **Drop-down and select menus (`select`)**
  Specify the width and height. Best practice is to leave the option formatting to the browser. Some allow you to style the option text.
- **Fieldsets and legends**
  Apply box properties to **`fieldset`**. The **`legend`** element is difficult to style (try styling a `span` inside the `legend` element instead).

Styling Tables

To add **space within a cell**, apply the **`padding`** property to `td` or `th.`

To **add space between cells**, use the **`border-collapse`** and **`border-spacing`** properties.

`border-collapse`

Values: `separate`, `collapse`

`separate` allows space between cells. `collapse` combines borders with no space.

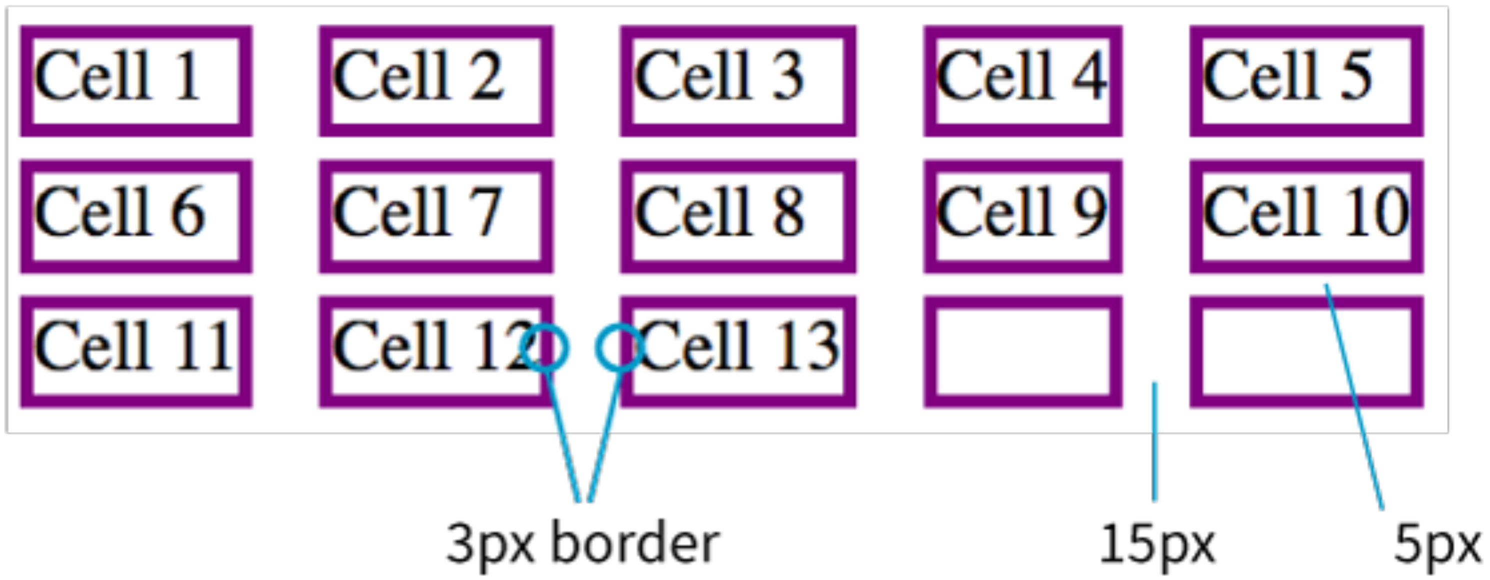
`border-spacing`

Values: *Horizontal-length vertical-length*

Specifies an amount of space between columns and rows when `border-collapse` is set to `separate`

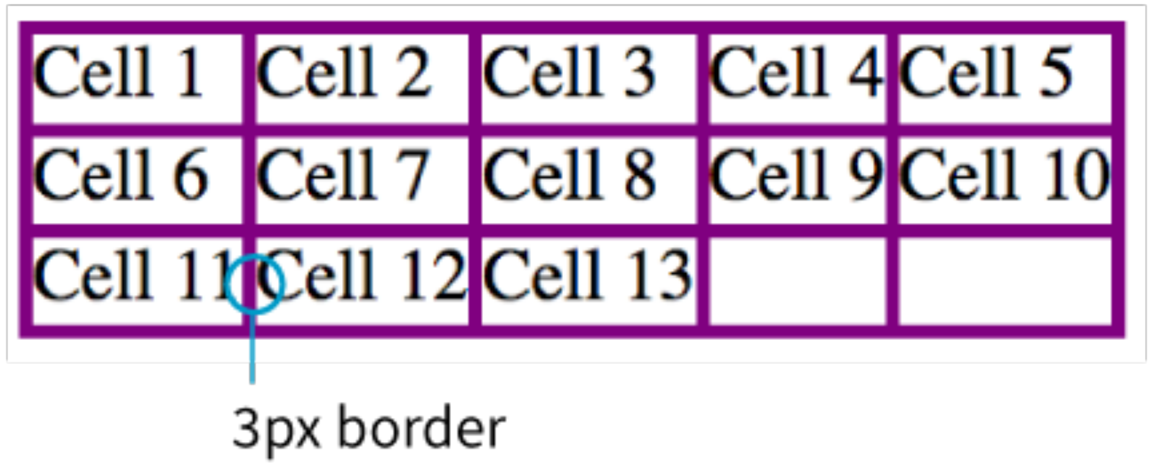
Styling Tables (cont'd)

```
td {
  border: 3px solid purple;
}
table {
  border-collapse: separate;
  border-spacing: 15px 5px;
  border: none;
}
```



Cell 1	Cell 2	Cell 3	Cell 4	Cell 5
Cell 6	Cell 7	Cell 8	Cell 9	Cell 10
Cell 11	Cell 12	Cell 13		

```
td {
  border: 3px solid purple;
}
table {
  border-collapse: collapse;
  border: none;
}
```



Cell 1	Cell 2	Cell 3	Cell 4	Cell 5
Cell 6	Cell 7	Cell 8	Cell 9	Cell 10
Cell 11	Cell 12	Cell 13		

Styling Tables (cont'd.)

`empty-cells`

Values: `show`, `hide`

When borders are `separate`, this property indicates whether empty cells display their backgrounds and borders.

`caption-side`

Values: `top`, `bottom`

Specifies on which side the `caption` element displays.

`table-layout`

Values: `auto`, `fixed`

Calculates the width of the table based on width values (`fixed`) or the minimum width required for the content of the table (`auto`).

“A Clean Slate” with CSS Reset

A **CSS reset** is a collection of styles that overrides *all* user agent styles (browser defaults) and gives you a neutral starting point for your own styles.

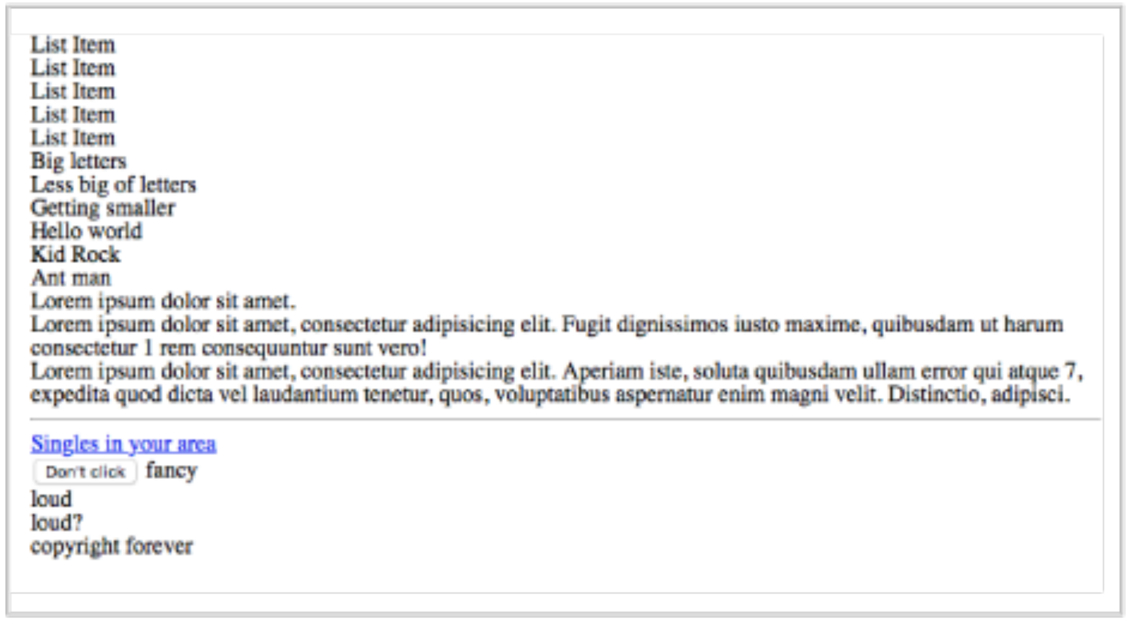
Example: An excerpt from Eric Meyer’s CSS Reset that removes margins, padding, and borders and sets the font to 100% for all elements:

```
/* http://meyerweb.com/eric/tools/css/reset/
    v2.0 | 20110126 License: none (public domain)*/
html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, /*... MORE ELEMENTS...*/ section, summary,
time, mark, audio, video {
   margin: 0;
   padding: 0;
   border: 0;
   font-size: 100%;
   font: inherit;
   vertical-align: baseline;
}
/* Styles continue... */
```

Normalize.css

Normalize.css is a reset style sheet that retains some initial browser styles but tweaks them for consistency across browsers. It prevents needing to write styles for every element.

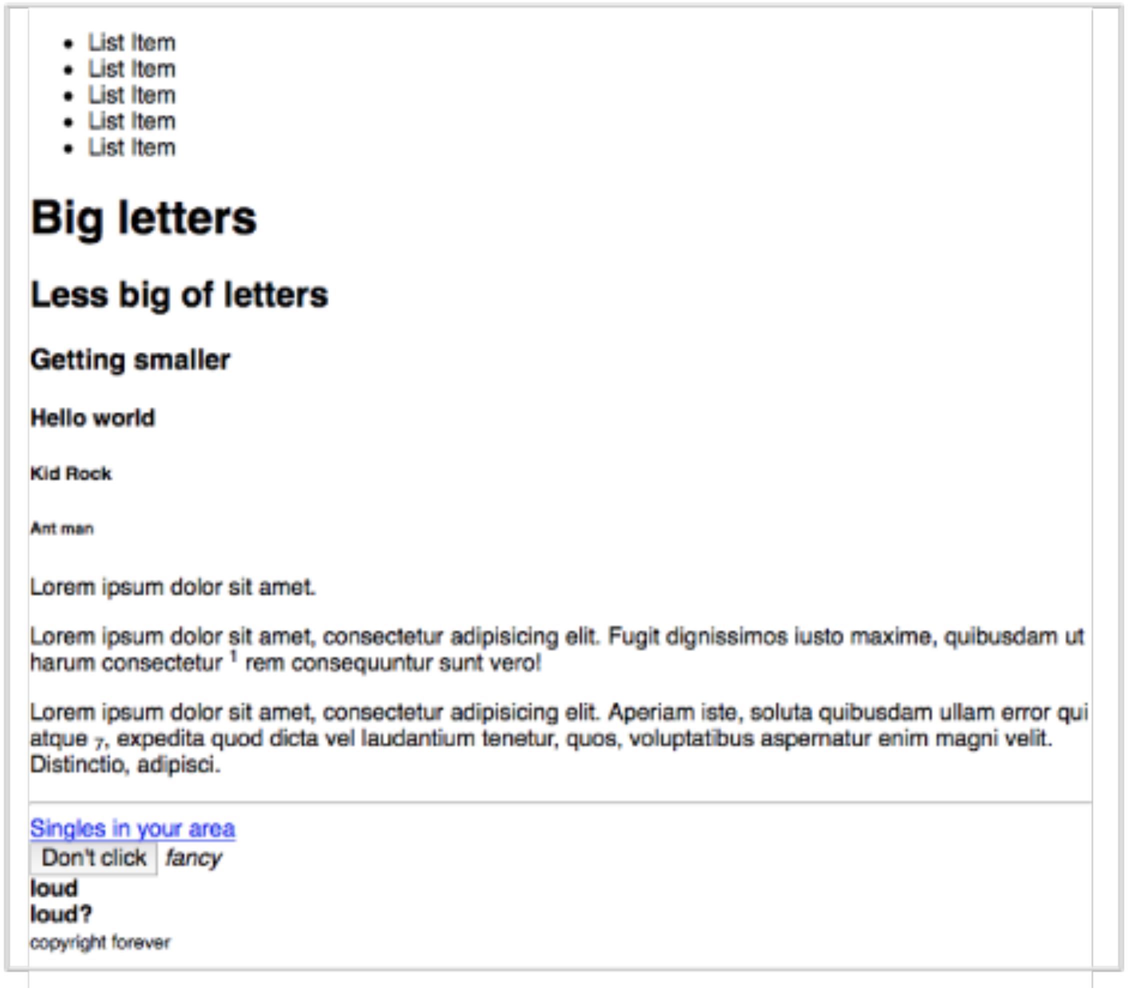
CSS reset



List Item
List Item
List Item
List Item
List Item
Big letters
Less big of letters
Getting smaller
Hello world
Kid Rock
Ant man
Lorem ipsum dolor sit amet.
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Fugit dignissimos iusto maxime, quibusdam ut harum consectetur 1 rem consequuntur sunt vero!
Lorem ipsum dolor sit amet, consectetur adipisicing elit. Aperiam iste, soluta quibusdam ullam error qui atque 7, expedita quod dicta vel laudantium tenetur, quos, voluptatibus aspernatur enim magni velit. Distinctio, adipisci.

Singles in your area
Don't click fancy
loud
loud?
copyright forever

Normalize.css



- List Item
- List Item
- List Item
- List Item
- List Item

Big letters

Less big of letters

Getting smaller

Hello world

Kid Rock

Ant man

Lorem ipsum dolor sit amet.

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Fugit dignissimos iusto maxime, quibusdam ut harum consectetur [1] rem consequuntur sunt vero!

Lorem ipsum dolor sit amet, consectetur adipisicing elit. Aperiam iste, soluta quibusdam ullam error qui atque $_7$, expedita quod dicta vel laudantium tenetur, quos, voluptatibus aspernatur enim magni velit. Distinctio, adipisci.

Singles in your area
Don't click *fancy*
loud
loud?
copyright forever

Image Replacement Techniques (IRT)

IRT replaces text with an image (like a logo) while ensuring it's accessible to screen readers and search engines.

There are many techniques. The "Phark" method is popular:

1. Put the image in the background of the sized text element.

2. Move the text itself out of the browser window with a large negative `text-indent`.

NOTE: Consider whether the `alt` attribute on an `img` element may suffice before using an IRT.

Phark Image Replacement Technique

The markup:

```
<h1 id="logo">Jenware</h1>
```

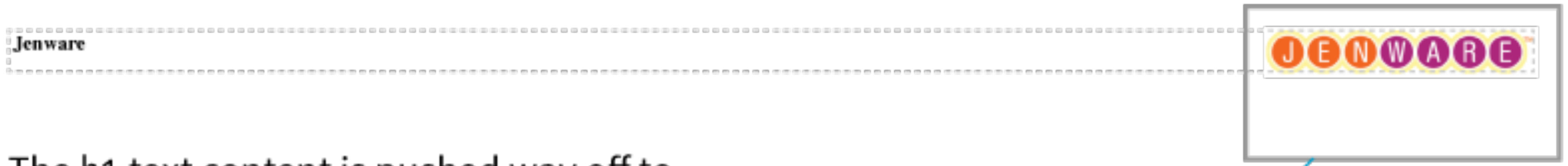
The style rule:

```
#logo {
  width: 450px;
  height: 80px;
  background: url(jenware.png) no-repeat;
  text-indent: -9999px;
}
```

What users see:

What is actually happening:

`text-indent: -9999px;`



The h1 text content is pushed way off to the left, outside the browser window.

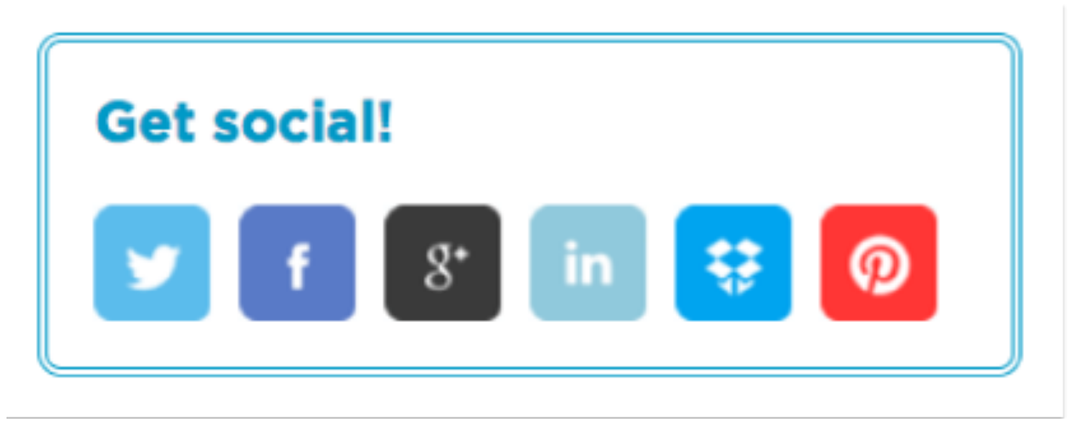
Browser window

CSS Sprites

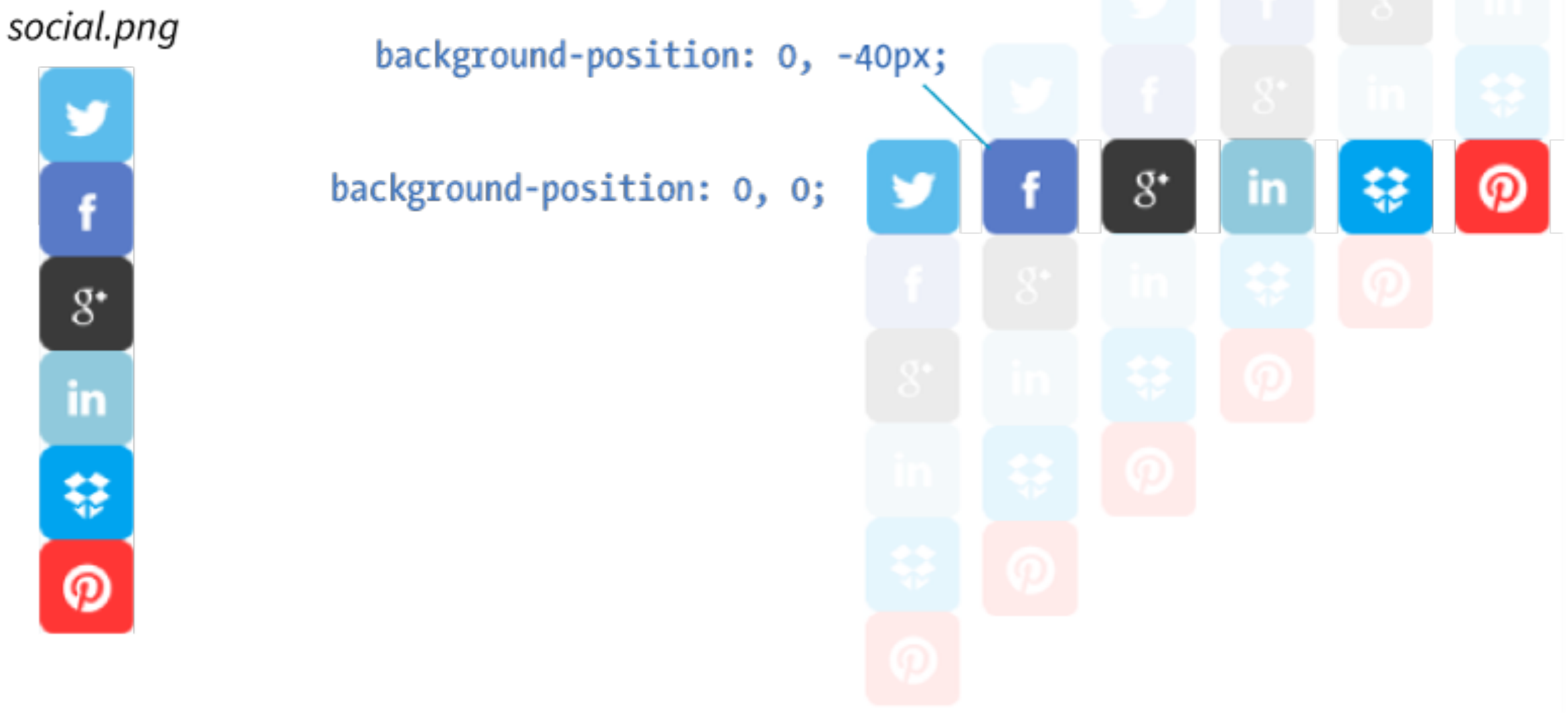
- A **sprite** is an image file that contains multiple images.
- Putting many small images in one image file reduces calls to the server and **improves performance**.
- Use the image as a background image and **control which portion of it is visible** with the `background-position` property:

```
li a {
   display: block;
   width: 40px;
   height: 40px;
   background-image: url(social.png);
}
li a.twitter { background-position: 0 0;}
li a.fb { background-position: 0 -40px;}
li a.gplus { background-position: 0 -80px;}
li a.linkedin { background-position: 0 -120px; }
li a.dropbox { background-position: 0 -160px; }
li a.pinterest { background-position: 0 -200px; }
```

CSS Sprites (cont'd)



The separate icons in this panel are contained on one sprite image (*social.png*) that is postioned in each a element.



CSS Feature Detection

It takes a while for new CSS features to be supported in all browsers. **Feature detection** lets you test for support of a particular feature and provide appropriate fallbacks for non-supporting browsers.

Two common detection methods:

- **CSS feature queries** (`@supports` rule)
- **Modernizr** (JavaScript library)

NOTE: In the past, styles were delivered based on the browser version. Testing for individual features is a better approach.

CSS Feature Queries (@supports)

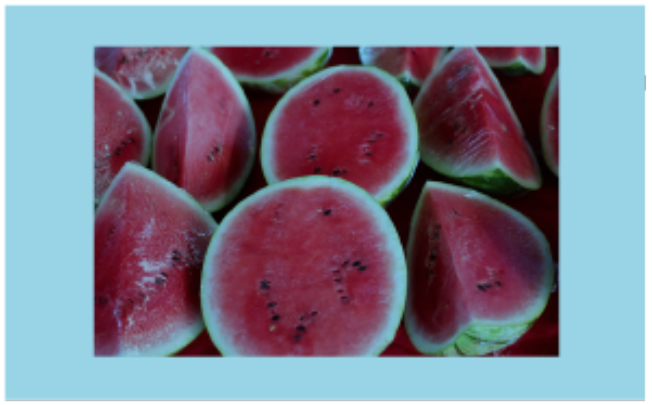
A **@supports rule** tests for support of a particular **property and value pair**. If the browser passes, the rules inside the brackets are applied:

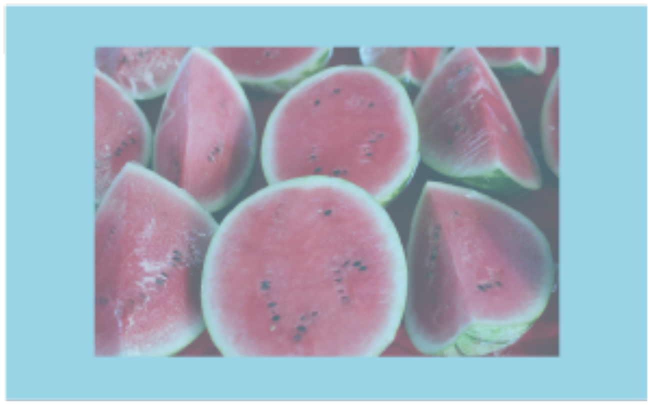
```
@supports (property: value) {
  /* Style rules for supporting browsers here */
}
```

TIP: Provide fallback styles for non-supporting browsers first and then override them with the preferred rules.

CSS Feature Queries (cont'd)

Original image (no effect)


As seen on browsers that support `mix-blend-mode: multiply;`


Fallback for non-supporting browsers (`opacity: .5`)

The `mix-blend-mode` property is specified for browsers that support it. Other browsers get a similar effect with the widely supported `opacity` property.

```
#container {
  background-color: #96D4E7;
  padding: 5em;
}
.blend img {
  opacity: .5;
}
@supports (mix-blend-mode: multiply) {
  .blend img {
    mix-blend-mode: multiply;
    opacity: 1;
  }
}
```

CSS Feature Queries (cont’d)

PROS:

- Doesn’t rely on JavaScript
- Can take advantage of cutting-edge properties safely

CONS:

- Limited browser support (for now)
- CAUTION: Some browsers that *don’t* support feature queries *do* support newer properties you might test for (for example, Flexbox). The new properties won’t apply if they are in an unsupported `@supports` rule.

NOTE: Feature queries are a great way to test for Grid support.

Modernizr

A JavaScript library that tests for HTML5 and CSS3 features.

It indicates the result (support or no support) with a **class name** applied to the **`html`** element (and in a JavaScript object for scripting):

```
<html class="js flexbox">
<html class="js no-flexbox">
```

Use the generated class name in the selector to separate styles for supporting and non-supporting browsers:

```
.flexbox nav {
  /* flexbox styles for the nav element here */
}

.no-flexbox nav {
  /* fallback styles for the nav element here */
}
```

Modernizr (cont'd.)

1. Go to modernizr.com to build and download the Modernizer.js script. (NOTE: You can customize the script to test for just the features you need.)

2. Put the *Modernizer.js* file in the same directory a the files for your site.

3. Add the script to the **`head`** of the HTML document, before linked style sheets or other scripts.

4. Add `class="no-js"` to the **`html`** element (it gets overwritten if there is JavaScript):

```
<html class="no-js">
<head>
  <script src="modernizr-custom.js"></script>
  <!--other scripts and style sheets -->
</head>
```

Modernizr (cont’d)

PROS:

- Easy to use, with thorough documentation
- Excellent browser support

CONS:

- Relies on JavaScript (which may not be enabled)
- Slightly slower than CSS feature queries