6

BONUS MATERIAL

Putting Limits on Flexibility

This PDF contains information on JavaScript and IE techniques for limiting flexibility by simulating or adding min- and max-width properties.

USING JAVASCRIPT TO FIX IE'S CSS SHORTCOMING

All of the solutions we're covering rely on JavaScript, which is unfortunate, as it's usually ideal to fix CSS shortcomings with CSS. If some of your users don't have JavaScript enabled, these solutions of course won't work for them.

We have to use JavaScript here because there are simply no reliable ways to simulate `min-` and `max-width` in IE 6 and earlier without it. There is one CSS method for simulating `min-width`, but it's rather involved and doesn't solve the lack of support for `max-width`, so we won't be going over it here. If you want to check it out, Stu Nicholls explains his method at www.webreference.com/programming/min-width. A variation of this technique, by Bruno Fassino, can be found at www.brunildo.org/test/min-widthS.html.

EMBEDDING EXPRESSIONS IN THE CSS

■ **NOTE:** For more information on dynamic properties and their syntax, read the official Microsoft introduction to dynamic properties at http://msdn.microsoft.com/en-us/library/ms537634.aspx.

You can use a Microsoft-proprietary technology called *dynamic properties* to embed scripting inside your CSS file. Dynamic properties, which work only in IE, allow you to create formulas for dynamic or conditional values, also known as *expressions*.

To add an expression to your style sheet, simply add `expression()` in the place of the value of whatever property you want to make dynamic, and then add the appropriate JavaScript formula inside the parentheses. For instance, you could set a width value of 200 pixels on a `div` using the expression in this CSS rule:

```
div {
   width:expression(200 + "px");
}
```

■ **NOTE:** The code samples in this section have line breaks within the expressions to fit within the bounds of the book's pages. These line breaks aren't necessary for you to preserve in your own code—you can take them out, leave them in, or add them in different spots.

Of course, the formula for setting a minimum or maximum width is a lot more complicated. Here's how you can set a minimum width in pixels:

```
#wrapper {
   width:expression((document.body.clientWidth < 501)? "500px" :
"auto");
   min-width: 500px;
}
```

The `min-width` value is there simply for browsers other than IE 6 and earlier. The expression works by checking to see if the width of the `body` is less than 501 pixels. If so, it sets the width to 500 pixels. If not, it lets the width be "auto," or whatever it would be otherwise (in this case, 100 percent, since there's no width set anywhere). You could change this value of `auto` to an

explicit value like 90% if you wanted that to be the default before the min-width is reached.

The discrepancy between the 501 and 500 values in that expression is not a mistake; it's important that you keep the two pixel values different from each other by at least one pixel. If you don't do this, IE will freeze when you narrow your window past the min-width.

Here's an expression setting a max-width in pixels:

```
#wrapper {
   width:expression((document.body.clientWidth > 1001)? "1000px"
: "auto");
   max-width: 1000px;
}
```

The only difference here—other than the values, of course—is the "less than" sign has been replaced with a "greater than" sign in the expression.

You can combine the expressions for min- and max-width:

```
#wrapper {
   width:expression((document.body.clientWidth < 501)? "500px" :
(document.body.clientWidth > 1001)? "1000px" : "auto");
   min-width: 500px;
   max-width: 1000px;
}
```

■ **NOTE:** The page showing this completed technique is minmax_expression_pixels.html in the ch6_examples.zip file.

These expressions work when you're trying to set a min- or max-width in pixels, but they need to be a lot more complicated to work with em values. Here's one syntax you can use to set min- and max-width in ems:

```
#wrapper {
   width:expression(document.body.clientWidth > (60.5*(screen.
deviceXDPI/72))* parseInt(document.body.currentStyle.
fontSize)? "60em" : document.body.clientWidth < (30.5*(screen.
deviceXDPI/72))* parseInt(document.body.currentStyle.fontSize)?
"30em" : "100%");
   min-width: 30em;
   max-width: 60em;
}
```

As I said, explaining JavaScript is beyond the scope of both my skills and this book, so I won't even attempt to explain how this whole expression works; the full explanation is at Tom Lee's blog at http://tom-lee.blogspot.com/2006/03/how-many-pixels-in-em-part-2.html. The important things to note about the expression are simply these four values:

- 60.5
- 60em

- `30.5`
- `30em`

These are the only values you need to change on your particular page—the rest of the expression you can copy as-is without having to worry about how it works. Change the `60em` value to the `max-width` you want; change the `60.5` value to something that is slightly different from your `max-width` value (again, to avoid freezing the browser). Change the `30em` value to the `min-width` you want, with `30.5` being the slightly offset matched value.

A couple other things to note about this expression:

- It doesn't work in IE 5.x. If this matters to you, there's an even more complicated (!) version of the expression at Tom Lee's blog post that you can use instead.
- It doesn't work if you have any margins, padding, or borders on the element you're setting the expression on. You'll need to implement an even more insanely complicated expression if you want it to take margins, padding, or borders into consideration; this expression can be found in another of Tom Lee's blog posts at http://tom-lee.blogspot.com/2006/03/max-width-in-ie-using-css-expression.html.

■ **NOTE:** The page showing this completed technique is minmax_expression_ems.html in the ch6_examples.zip file.

All in all, using expressions is a good solution if you want to keep everything contained in the CSS, instead of turning to external scripts to fix CSS issues. Also, expressions may still work even when JavaScript is disabled in a user's browser (depending on what version of IE she is running and her security level). Expressions do have plenty of downsides, though:

- Unless you're a JavaScript developer, they're pretty complicated to work with.
- They don't validate. If this matters to you, hide the expressions inside an IE conditional comment. This is probably the best thing to do anyway, as it saves other browsers from having to download CSS that's totally useless to them.
- They can make IE sluggish, as it has to work almost constantly to see if values need to be recalculated.
- Sometimes they freeze IE even when you have made sure to differentiate the values by at least a pixel or two. This seems to be due to the expression interacting with other IE bugs that are triggered by certain CSS, but I can't tell you the exact conditions under which it will break—all I can tell you is expressions just plain don't work sometimes.

Because of these disadvantages, I never really bother with expressions, and turn to external JavaScripts instead.

USING EXTERNAL JAVASCRIPTS

A number of resourceful folks have come up with scripts to add `min-` and `max-width` support to IE 6 and earlier. There's no point in explaining the scripts—I'll just recommend a couple, and you can plug the one you want into your pages as you would any other external JavaScript file.

- **Doxdesk minmax.js by Andrew Clover (www.doxdesk.com/software/js/minmax.html).** This one is my favorite. All you have to do is add a link to the script, and it automatically finds any `min-width`, `max-width`, `min-height`, and `max-height` values in your CSS and applies these to the same elements in IE 6 and earlier. You don't have to do any customization in the script itself.
- **IE7.js by Dean Edwards (http://code.google.com/p/ie7-js).** This is actually an entire JavaScript library that corrects many of the CSS shortcomings in IE 6 and earlier, including the lack of `min-` and `max-width`. Because it does so much, it's a hefty file, so I wouldn't advise using it unless you need many of its functions. Even then, hide it from other browsers via conditional comments so they don't have to download it (and so your server won't have to keep serving it out unnecessarily).