FONT SIZING WITH REM

Introduction

Determining a unit of measurement to size our text can be a topic of heated debate, even currently. Unfortunately, there are still various pros and cons that make the various techniques less desirable. It is just a matter of which less-desirable is most desirable.

There are two main techniques that are extolled:

1. Size with px
2. Size with em

Let us review these two approaches before I reveal the magical third.

Sizing with px

In the early days of the web, we used pixels to size our text. It is reliable and consistent. Unfortunately, users of Internet Explorer—even in IE9—do not have the ability to change the size of the text using the browser function of increasing or decreasing font size. For those concerned about the usability of their site, this may be a big deal. Recent versions of IE include zooming, which increases the size of everything on the page—a feature that is also available in most other mainstream browsers, too. This has helped mitigate the issue to a degree.

I, personally, have been of the camp that px-based layouts provide the consistency I prefer, and users have enough tools available to adjust their view that accessibility is less of a concern. But I digress. What else can we do?

Sizing with em

That whole inability to resize text in IE has been a continuing frustration. To get around that, we can use em units. Richard Rutter's article, How to size text using ems, was probably the first I read of this approach, way back in 2004. (Wow, it has been almost seven years.)

The technique modifies the base font-size on the body using a percentage. This adjusts things so that 1em equals 10px, instead of the default 16px. To set the font-size to the equivalent of 14px, set it to 1.4em.

```
body { font-size:62.5%; }
h1   { font-size: 2.4em; } /* =24px */
p    { font-size: 1.4em; } /* =14px */
li   { font-size: 1.4em; } /* =14px? */
```

The problem with em-based font sizing is that the font size compounds. A list within a list is not 14px, it is 20px. Go another level deeper and it is 27px! These issues can be worked around by declaring any child elements to use 1em, avoiding the compounding effect.

```
body { font-size:62.5%; }
h1   { font-size: 2.4em; } /* =24px */
```

```
p  { font-size: 1.4em; } /* =14px */
li { font-size: 1.4em; } /* =14px? */
li  li, li p /* etc */ { font-size: 1em; }
```

The compounding nature of em-based font-sizing can be frustrating so what else can we do?

Sizing with rem

CSS3 introduces a few new units, including the rem unit, which stands for "root em". If this has not put you to sleep yet, then let us look at how rem works.

The em unit is relative to the font-size of the parent, which causes the compounding issue. The rem unit is relative to the root—or the html—element. That means that we can define a single font size on the html element and define all rem units to be a percentage of that.

```
html { font-size: 62.5%; }
body { font-size: 1.4rem; } /* =14px */
h1   { font-size: 2.4rem; } /* =24px */
```

I am defining a base font-size of 62.5% to have the convenience of sizing rems in a way that is similar to using px.

But what pitiful browser support do we have to worry about?

You might be surprised to find that browser support is surprisingly decent: Safari 5, Chrome, Firefox 3.6+, and even Internet Explorer 9 have support for this. The nice part is that IE9 supports resizing text when defined using rems. (Alas, poor Opera, up to 11.10, at least) has not implemented rem units yet.)

What do we do for browsers that do not support rem units? We can specify the fall-back using px, if you do not mind users of older versions of Internet Explorer still being unable to resize the text (well, there's still page zoom in IE7 and IE8). To do so, we specify the font-size using px units first and then define it again using rem units.

```
html { font-size: 62.5%; }
body { font-size: 14px; font-size: 1.4rem; } /* =14px */
h1   { font-size: 24px; font-size: 2.4rem; } /* =24px */
```

And voila, we now have consistent and predictable sizing in all browsers, and resizable text in the current versions of all major browsers.

Note: Dec 13, 2011: Opera 11.60 now supports the rem unit.

Reference

PUBLISHED MAY 01, 2011 UPDATED DECEMBER 13, 2011
CATEGORIZED AS HTML AND CSS
Snook, Jonathan. "Snook.ca." Font Sizing with Rem -. 01 May 2011. Web. 09 Feb. 2016.
SHORT URL: http://snook.ca/s/997