

Standardize displays on Web portals running on Firefox3 and Internet Explorer 7

Skill Level: Intermediate

Judith M. Myerson (jmyerson@bellatlantic.net)
System Engineer and Architect

11 Nov 2008

Do Firefox3 and Internet Explorer 7 look different? What's the best way to get these browsers to behave the same way? Should you use pixels or em values? Regular developerWorks author Judith M. Myerson shows you how to standardize displays on Web portals running on Firefox and IE. She gives tricks and tips for using em values to make the job of developing the contents of portals, including Asynchronous JavaScript + XML (Ajax) applications, much easier.

Introduction

In my previous developerWorks articles (see [Resources](#) for links), I talk about how you can monitor network redundancy and failover and create programs to solve network problems; what pitfalls to avoid when you develop Ajax applications, including some helpful solutions for optimizing browser differences; give you some solutions for speeding up Ajax applications; and show you some application-strengthening tools to improve security problems within your Ajax applications.

In this article, I talk about how running a Web portal in IE 7 and Firefox 3 can give different results. IE 7 has default fonts that you can resize and use, while Firefox 3 lets you set default fonts without resizing them. To get the portals to display fonts and tables the same way on both browsers, you need to resize them in em. Within this article, I show how em is *relative* to the base font you can set, while px is *absolute* to the base font based on the visual display of an object on the screen. I also give you tricks and tips in using the em values to make the job of developing the contents of Web portals, including Ajax applications, much easier.

The problem: pixels

People stare at paragraphs longer than they do at images. They follow these paragraphs with their eyes, while the only time they stare at images is when they see the images at an art gallery. Because people follow the text with their eyes, they will notice how Web pages running on IE 7 and Firefox 3 present the text differently if the browsers are using pixels to define absolute font sizes. If the differences are large, they can be annoying.

Additionally, when you zoom the page of pixels, the sizes grow disproportionately, and you end up with different text fields covering each other. The whole layout is completely messed up! All because of the pixels (actually the dots). They are unscalable and fixed in size.

The answer: M relativity

To get around the sizing problem, I use em measurement, which will always be relatively sized and scalable. I use em as a percentage of the font size of the parental element.

As shown in Listing 1, the top-most `<div>` layer has a computed font size of 16px as the parental element. The second `<div>` layer shows the child font size is changed to 0.75em. The computed child font size of 12px is achieved by multiplying 16px by .75. In the third `<div>` layer, the child font size in the second layer becomes the new parent font size. Multiplying this parent font by 0.75 results in a smaller font size of 9px.

Listing 1. Parent and child font sizes

```
<body>
<div style="font-size:16px;" >
  <div style="font-size:0.75 em;">
    <div style="font-size:0.75 em;">
      </div>
    </div>
  </div>
</body>
```

In addition to font, you can use the em value for padding, margin, border, and wherever you can set up sizes.

What's the font size?

The default font size for IE 7 and Firefox 3 is 16px, which is equivalent to 1 em. IE

represents this font size as "medium" text. You can reduce this font size by assigning 62.5% to the `font-size` property of the body selector (also known as the block-level element) like shown in Listing 2.

Listing 2. Reduce font size

```
<style type="text/css">
body {font-size: 62.5%}
</style>
```

This assignment takes 16px down to 10px. The reduced font size remains at 1 em. In Firefox3, you can change the default size to a larger text for your higher-resolution screen. Table 1 suggests font sizes for three screen resolutions.

Table 1. Fonts for higher screen resolution

Font size	Screen resolution
20px	1280x960
22px	1400x1050
28px	1792x1344

All translate to almost the same physical size as the font size of 16px at 1024x758. If your screen resolution is not in the table, use this equation to find out what font size you should use:

width of screen resolution/64 = font size

for example,

1280/64 = 20px

Design the document

Consider a document layout of two columns—the main content and a sidebar. The document is supplemented with a navigation menu as the header and company information as the footer.

Listing 3. Document layout

```
<body>
<div id="navigation"> ... </div>
<div id="main_content"> ... </div>
<div id="side_bar"> ... </div>
<div id="footer"> ... </div>
</body>
```

Assume the default base font of 16px has been reduced to 62.5% for the body

selector, shown in Listing 4.

Listing 4. Document style

```
<style>
body {font-size: 62.5%}

#navigation {font-size:1em}
#main_content {font-size:1.2em}
#side_bar {font-size:1em}
#footer {font-size:0.9em}
</style>
```

This gives you a document where text in the navigation and sidebar is displayed at 10px, the main content is 12px, and the footer is 9px.

Compute em equivalents

The em values in Table 2 assume the browser default setting of a 16px font size. It also assumes that in the body selector, the value of the font size property is set to 1em.

Table 2. Em equivalents>

Font size	em equivalent	1px in ems
11px	0.689	0.091
12px	0.750	0.083
13px	0.814	0.077
14px	0.875	0.071
15px	0.938	0.067
16px	1.000	0.063
17px	1.064	0.059
18px	1.125	0.056
19px	1.188	0.053
20px	1.250	0.050

To get an em equivalent, do this:

$$(\text{child font size}/\text{parent font size}) = \text{child em}$$

For example, 12px/16px = .75em

To get an em equivalent in one pixel, do this:

$(1/\text{font size}) \times \text{required pixels} = \text{em equivalent}$

For example, $(1/14\text{px}) \times 1\text{px} = .071 \text{ em}$

What's the em equivalent of 770px?

Listing 5 shows the font size property for the body selector as 1 em. You assume this is equivalent to the browser default font size of 16px. So, to calculate the em equivalent of 770px, use the following equation.

$$(1\text{px}/16\text{px}) \times 770\text{px} = 48.125\text{em}$$

You apply the result to the width property in the `div` selector.

Listing 5. Set width in ems

```
body{
font-size:1em;
text-align:center;
}

div{
width:48.125em;
margin:0 auto;
}
```

The code says that the `div` selector inherits a property to center the text. This element automatically assigns ems equally to the right and left margins based on the size of the calculated width of the text.

Wrap it around

Now, I'll give "wrap" as the name for the ID to identify a control layer in the top-most `<div>` layer, as shown in Listing 6.

Listing 6. Identify a control layer

```
<body>
<div id="wrap">
.
.
.
</div>
</body>
```

I want it to be 1000px wide to comfortably fit a higher-resolution screen. What is 1000px in ems? Let's find out.

I know that the parent element (`<body>`) has a default font size of 16px. The child layer (`<div id="wrap">`) will inherit it. I can calculate what 1px is in ems using

the following math equation:

$$(1 / \text{parent font-size}) \times \text{required pixel value} = \text{em value}$$

Substituting, I get

$$(1/16\text{px}) \times 1000 = .0625\text{em} \times 1000 = 62.50\text{em}$$

Now, take a look at three selectors in Listing 7. They are `div#wrap`, `p`, and `h1`.

Listing 7. Three selectors

```
<style>
body {font-size: 1 em}

div#wrap{
width: 62.50em;
margin: 1.5em auto;
border: 0.063em solid #ccc;
background: #fff;
}

p{
font-size: 0.750em;
line-height: 1.5em;
margin: 1.5em;
}

h1{
font-size: 1.125em;
}

</style>
```

This listing says the font size differences having the same em values are due to the hierarchical layout of different child selectors. A child font size in one selector becomes a parent font size in another selector in the lower level in the hierarchy of selector sets. This is unlike Listing 2, which shows the font size differences as having the same em values.

div#wrap selector

In the `div#wrap` selector, I apply the width in 62.50 ems (1000px) and center the layer using the auto left and right margins. I give it a .063 (1px) gray border with a white background and align the text to the left.

p selector

The paragraph has inherited a 1em (16px) font size from its parent, `<div id="wrap">`. I already know that 1px is 0.0625em. I then use the same math to get the em value for 12px:

$$(1/16\text{px}) \times 12\text{px} = 0.750\text{em}$$

Next, I apply the result to the `font-size` property in the `p` selector, as shown in Listing 8.

Listing 8. Apply font-size property

```
p{
  font-size: 0.750em;
}
```

To calculate the desired line height and margin of 18px for the basic leading I do this:

$$(18\text{px} / 12\text{px}) = 1.5\text{em}$$

This equation says the line height and margin are 1 and a half times the font size.

Leading (pronounced "led-ing")

Leading, in old printing methods, was the insertion of lines of lead underneath rows of text. It is expressed as `line-height`, but instead of adding space below, it increases the space above and below each line of text.

h1 selector

The `<h1>` has inherited a 1em (16px) font size from its parent, `<div id="wrap">`. So, I already know what 1px is from before: 0.0625em. To set 18px font size for the `<h1>`, do this:

$$(1/16) \times 18 = .0625 \times 18 = 1.125\text{em}$$

Size up images in ems

When text sizing, assume that the size of an image should be a multiple of the basic leading. Listing 9 shows the image has a width and height of 90px (18px x 5).

Listing 9. Sizing an image

```
<body>
<p>

.
.
.
</p>
</body>
```

This listing says the image is a child of the paragraph—its parent— so I know that

the image has inherited a font size of 12px. Therefore, to calculate the image width and height in em I use:

$$(1/12) \times 90 = 7.5\text{em}$$

The image has right and bottom margins of 18px and is floated left in the paragraph text. I know that 12px/18px is 1.5 ems for the parent paragraph, as shown in Listing 10.

Listing 10. Image size properties

```
<style>
p img{
width: 7.5em;
height: 7.5em;
margin: 0 1.5em 1.5em 0;
float: left;
}
</style>
```

Avoid inherited shrinkage

Assume the parent font size is 12px. If so, I calculate an em value for 10px as 0.833. Then I add the result to the font size property like shown in Listing 11.

Listing 11. Inherited shrinkage example

```
<style>
#main_content LI {font-size:0.833em}
</style>
```

This means all main content list items should be displayed at 10px. I use the same equation to achieve this:

$$10 / 12 = 0.833$$

But what happens when one list contains another? It gets smaller. Why? It's because any list item in the #main_content div should 0.833 times the size of its parent. To prevent this inherited shrinkage, use this selector set:

```
LI LI {font-size:1em}
```

This says that any list item inside another list item should be the same size as its parent (the other list item). You should use a whole set of child selectors to prevent confusion during development: LI LI, LI P, TD P, BLOCKQUOTE P {font-size:1em}.

A similar job needs to be done on forms and tables to force form controls and table cells to inherit the correct size: INPUT, SELECT, TH, TD {font-size:1em}.

Revisit p selector

Now, I'll modify the elements in Listing 2 to avoid font size shrinkage for the paragraphs and lists when one parent list contains a child list. First, I remove the font-size property from the p selector in Listing 2. Then, as shown in Listing 12, a child selector set of p li li for the same font size.

Listing 12. Inherited shrinkage example

```
<style>

body {font-size: 1 em}

div#wrap{
width: 62.50em;
margin: 1.5em auto;
border: 0.063em solid #ccc;
background: #fff;
}

p li li {font size: 0.750 em}

p{
line-height: 1.5em;
margin: 1.5em;
}

h1{
font-size: 1.125em;
}

</style>
```

Conclusion

You'll need a team of developers, testers, and potential users to get the IE 7 and Firefox3 to standardize displays on Web portals by adopting em instead of pixels in all of your designs. To get the browsers to behave in the same way, you must plan ahead on creating, testing, and deploying Web portals using the em values. Include in your plan an em calculator that you can use or develop for integration into a browser or even a Web design application. Resolving these issues makes your job of standardizing displays on Ajax applications much easier.

Resources

Learn

- Go to [W3C's Cascading Style Sheets](#) on authoring tools, specifications, browsers, test suites, core styles, tutorials, and more.
- In the article, "[Get Nagios for your Ajax applications](#)," (developerWorks, August 2008) find out how you can monitor network redundancy and failover and create programs to solve network problems.
- In the article, "[Optimized and predictable Ajax applications](#)," (developerWorks, October 2007) learn what pitfalls to avoid when you develop Ajax applications, including some helpful solutions for optimizing browser differences.
- "[Speed up your Ajax applications while dodging Web services vulnerabilities](#)" (developerWorks, August 2007) suggests some solutions for speeding up Ajax applications.
- In the article, "[Ajax security tools](#)," (developerWorks, May 2008) discover some application-strengthening tools to improve security problems within your Ajax applications.
- The [Work with Web services in enterprise-wide SOA series](#) by Judith M. Myerson offers information on how to work with Web services in enterprise-wide SOAs.
- Browse the Judith M. Myerson's series, [Use SLAs in a Web services context](#), which has details on service-level agreements.
- Want more information on Ajax tools? Read about them in "[Survey of Ajax tools and techniques](#)" (developerWorks, July 2007).
- Read "[Tight coupling Web services in the SOA](#)" (developerWorks, January 2008) to get a look at the pros and cons of both tight and loose coupling Web services and the resulting change in scale that comes from tight coupling.
- Read Judith M. Myerson's [The Complete Book of Middleware](#), which focuses on the essential principles and priorities of system design and emphasizes the new requirements brought forward by the rise of e-commerce and distributed integrated systems.
- Get the business insight and the technical know-how to ensure successful systems integration by reading [Enterprise Systems Integration, Second Edition](#).
- Visit the [technology bookstore](#) for books on these and other technical topics.
- Stay current with [developerWorks technical events and webcasts](#).

Get products and technologies

- [IBM trial products for download](#): Build your next development project with IBM trial software, available for download directly from developerWorks.

Discuss

- [developerWorks blogs](#): Get involved in the developerWorks community.

About the author

Judith M. Myerson

Judith M. Myerson is a systems architect and engineer. Her areas of interest include middleware technologies, enterprise-wide systems, database technologies, application development, network management, security, RFID technologies, and project management.

Trademarks

IBM
Rational