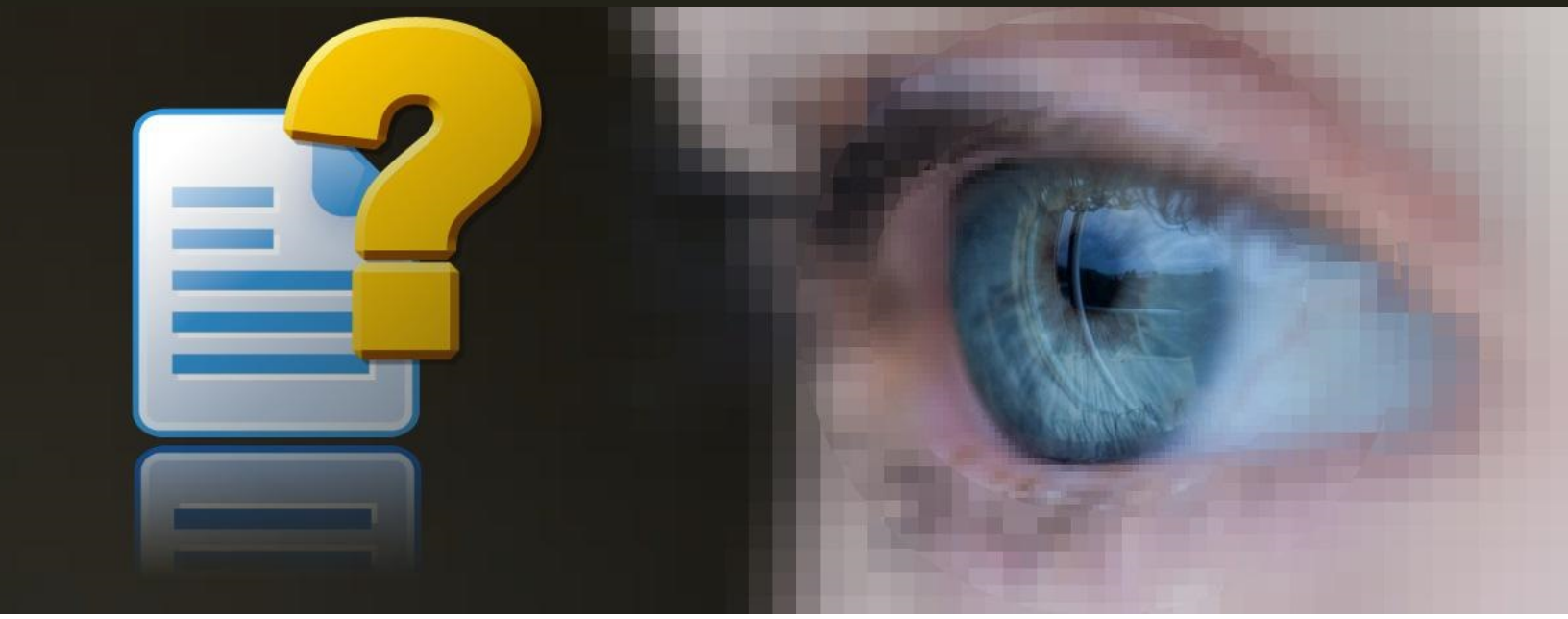
Zooming in on HTML Help



How to get Microsoft HTML Help ready for ultra-high resolution displays

By Alexander Halser, EC Software GmbH

Published in June 2015 with Help & Manual

Cover picture courtesy of pixabay.com

HTML Help in a Retina World **3**

Understanding the problem **4**

1 Suddenly broken? 5
2 Make text and other items larger 9
3 No zoom in HTML Help 11
4 About points, pixels and DPI 19

Implementing zoom in HTML Help **25**

5 Two ways to look at it 26
6 Applying zoom 28
7 Printing problems 37
8 The "em" solution 41

Summary **47**

9 HTML coding ahead 48
10 Links 50

Index **51**

For the last 20 years, Windows desktop applications have existed in a relatively unchanged display environment. The average display size grew during this period and the predominant aspect ratio changed from 4:3 to 16:9. A pixel, however, was still a pixel and a logical pixel still corresponded to a physical dot on the screen.

This was possible because the physical size of computer displays grew at about the same rate as their resolution. For example, compare your first 14" CRT monitor with its 1024 x 768 pixel resolution to a typical modern 24" consumer display with a HD resolution of 1920 x 1080. The physical size of each pixel - and the distance between the physical dots on the screen - is approximately the same. There may be a difference, but it is not huge.

Modern Ultra-HD displays ("4K displays") change the situation significantly, however, putting many, many more pixels into the same space. If nothing else changes, the letters in text will be the size of pinheads on such displays, and nobody has sharp enough vision to be able to read it comfortably. Everything on the display needs to be scaled for human consumption: Texts, headings, images and the entire user interface.

Most Windows developers have started to make their Windows desktop applications ready for the new high-resolution displays. The challenges involved in attaining full compatibility vary depending on the development environment. But whether you are programming in .NET, C#, C++, Delphi or another language, there is one thing we all have in common: The documentation of the software, the online help, needs to be ready for high-resolution displays as well as the programs themselves.

The standard Windows help format, HTML Help (or *CHM* files) is HTML-based and displayed by the *Microsoft HTML Help Viewer*. That's OK, you might think: CHM is HTML. The CHM viewer is really a browser, and browsers have zoom. So no problem, right?

Unfortunately, that isn't true, and this is something developers need to deal with if they want to deliver accessible documentation on high-resolution displays. This guide explains the important display differences between stand-alone browsers and the Microsoft HTML Help Viewer. It illustrates the problems encountered when displaying HTML Help on high-resolution monitors and suggests a variety of solutions you can use, depending on your HTML sources.

Happy 4K coding! :-)

Alexander Halser, May 2015

About the author

Alexander Halser is the founder of and senior developer at *EC Software GmbH*, the makers of the **Help +Manual** help authoring suite. You can reach him directly by email to alexander.halser@ec-software.com or a visit http://www.helpandmanual.com.

Understanding the problem

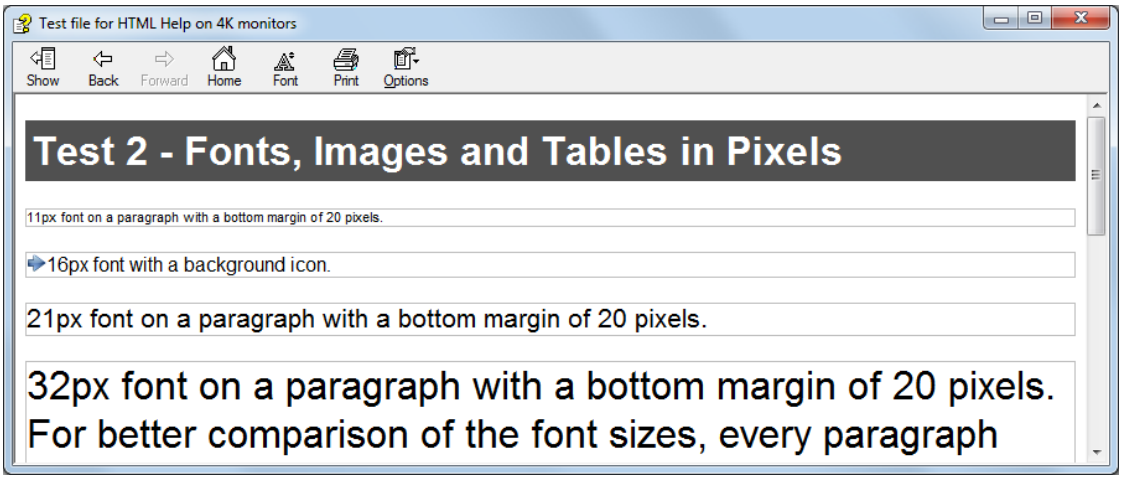
What is broken in HTML Help? There is no automatic zoom, and scaling on ultra-high resolution monitors doesn't work in the same way as in stand-alone web browsers. This chapter explains the rendering quirks of embedded Internet Explorer, which is the browser engine used in the HTML Help Viewer.

Picture courtesy of pixabay.com

1 Suddenly broken?

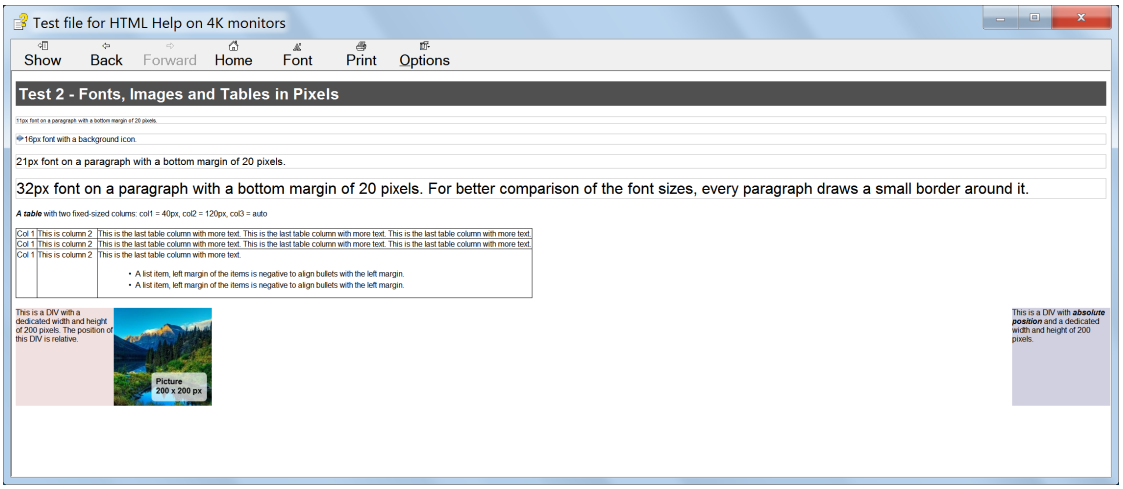
You have written a Windows software application and distributed this software with lovingly crafted documentation in the Microsoft HTML Help/CHM format. This has worked perfectly for years. However, chances are that very soon you will get a support call from a customer saying that they can't read the documentation because everything is too small. This customer most probably has an ultra-high resolution ("4K") monitor – or a modern high-resolution portable computer like a Microsoft Surface Pro or a Lenovo Yoga Pro.

This is what you see when you view the CHM help file on your standard display:



HTML Help file on a regular monitor

And this is what the customer sees on a hi-res display (both help windows are approximately the same size):



HTML Help file on an ultra-high resolution monitor

Are responsive HTML pages affected as well?

This can happen. It depends on the HTML code. It is not a lack of responsiveness that breaks the display of HTML Help on high-resolution monitors. The problems are caused by differences in the HTML rendering engine of the HTML Help Viewer compared to that of a current stand-alone browser.

I will outline the differences in detail in the next chapters. But the most important point is that if there are any "*px*" (pixel) dimensions or sizes in the CSS or the HTML source that you have compiled the CHM file from, your help file will most probably be affected by these problems to a greater or lesser extent. Any HTML element with a dedicated height, width, position, padding or margin in pixels, or images without explicit height and width dimensions, will not display correctly.

```
<img src="./img/screen1.png">

<img src="./img/screen1.png" width="635" height="397" />

<img src="./img/screen1.png" style="width:635px; height:397px" />

<div style="width:400px; height:auto; margin:20px auto">...</div>
```

If you have anything like the examples above in the HTML source of your CHM files, then you will have a problem.

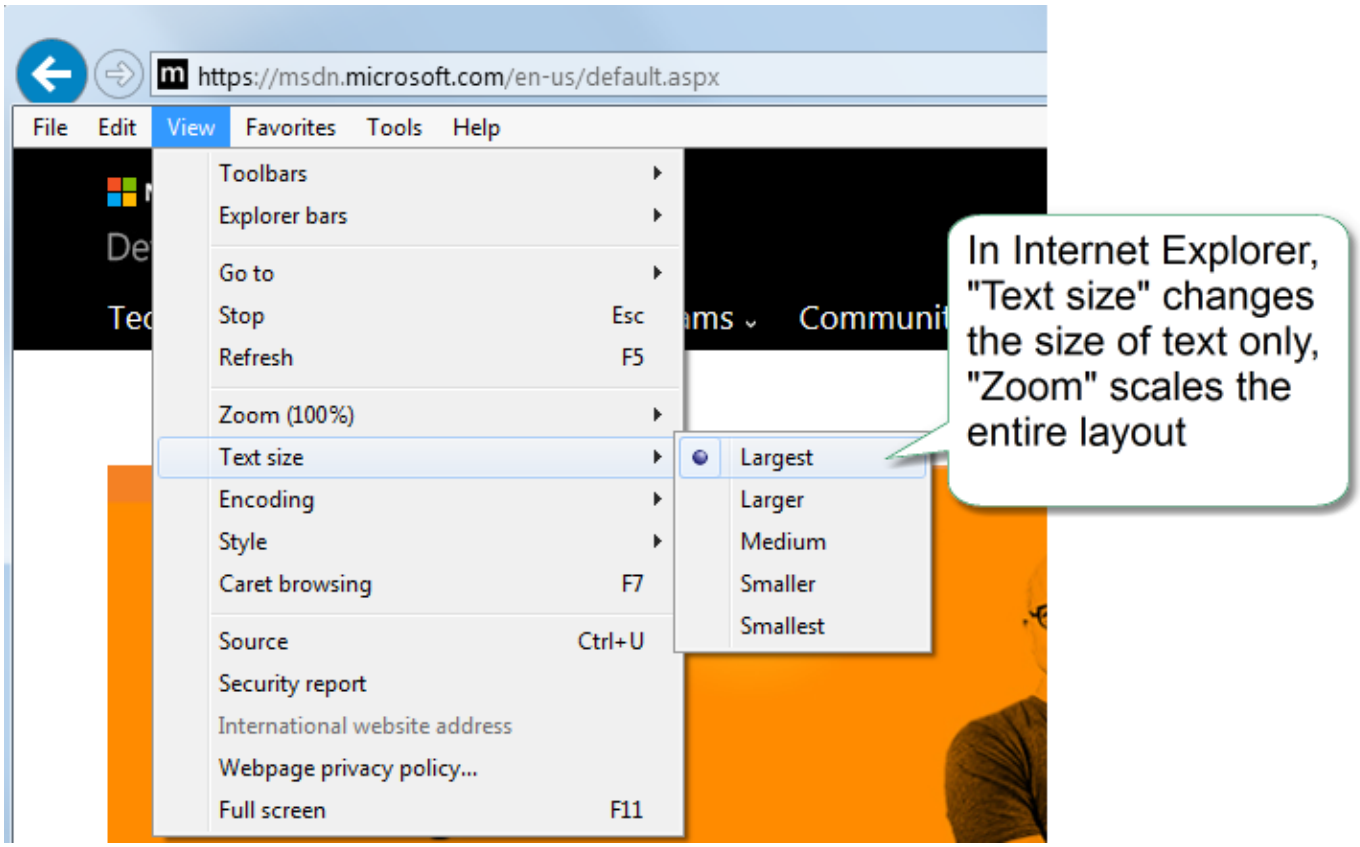
Does a dedicated help authoring tool help?

If you use Help & Manual, yes. This is where an authoring tool can be *really* helpful!

You can fix it with your favorite HTML editor as well, and in the next few chapters I will explain how to do that. But a dedicated authoring tool like *Help & Manual* will save you hours, if not days of work. Here is a shortcut link to see how we are doing it.

What end users can do about broken help files

The HTML Help viewer runs an embedded version of Microsoft Internet Explorer (MSIE) and MSIE has an option to increase or decrease text sizes. This was available in older versions of MSIE as well. It changes the display size of text only, on contrast to the newer *Zoom*, which scales the display of the entire HTML page.



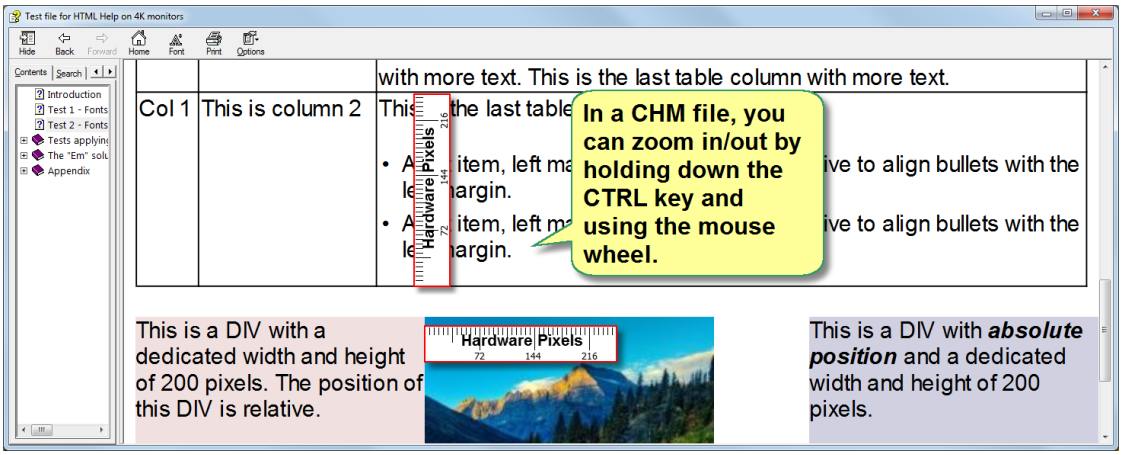
View menu in Internet Explorer

This option is hidden in the *View* menu. To change it, you first need to display the menu bar (right click on the title bar in MSIE, and then activate the "Menu bar" option in the context menu).

Whether or not this works will depend on the HTML code in the page being displayed. The text size option has no effect on fixed-sized fonts defined in *pixels* or *points.* It only works if you have used relative fonts whose size is defined in *ems* or *percent*. The option is set in Internet Explorer but its effect is global, and it is used by the embedded version of MSIE used by the HTML Help viewer as well. Any change to the settings in the stand-alone MSIE on a system will also affect the display of help pages in HTML Help, although in HTML Help, the option is not directly accessible.

The second option is a real zoom. The stand-alone version of Internet Explorer automatically defaults to a zoom value that matches the resolution of the user's Windows system. Any change to this zoom is permanent and saved in the Windows registry, so that MSIE will start with this zoom value the next time.

The zoom is available in HTML Help as well. Again it is not directly accessible, but you can still change it from within the HTML Help Viewer. To change the zoom in HTML Help, you click in the HTML viewer window (to put the mouse/keyboard focus there) and then increase or decrease the zoom by holding down the CTRL key and turning the mouse wheel. Alternatively, you can use the keyboard shortcuts **[CTRL]** + **[0]** for the default zoom and **[CTRL]** + **[+]** and **[CTRL]** + **[-]** to increase and decrease the zoom, respectively.



HTML Help on a "normal" monitor, manually zoomed in

None of these actions really solves the problem on ultra-high resolution monitors, however. Unlike the text size option, the zoom setting is not saved. Even if the user changes it, the HTML Help Viewer defaults to a zoom of 100% every time a CHM help file is opened. In addition to this zoom in HTML Help works differently than zoom in a stand-alone web browser.

Quick Summary

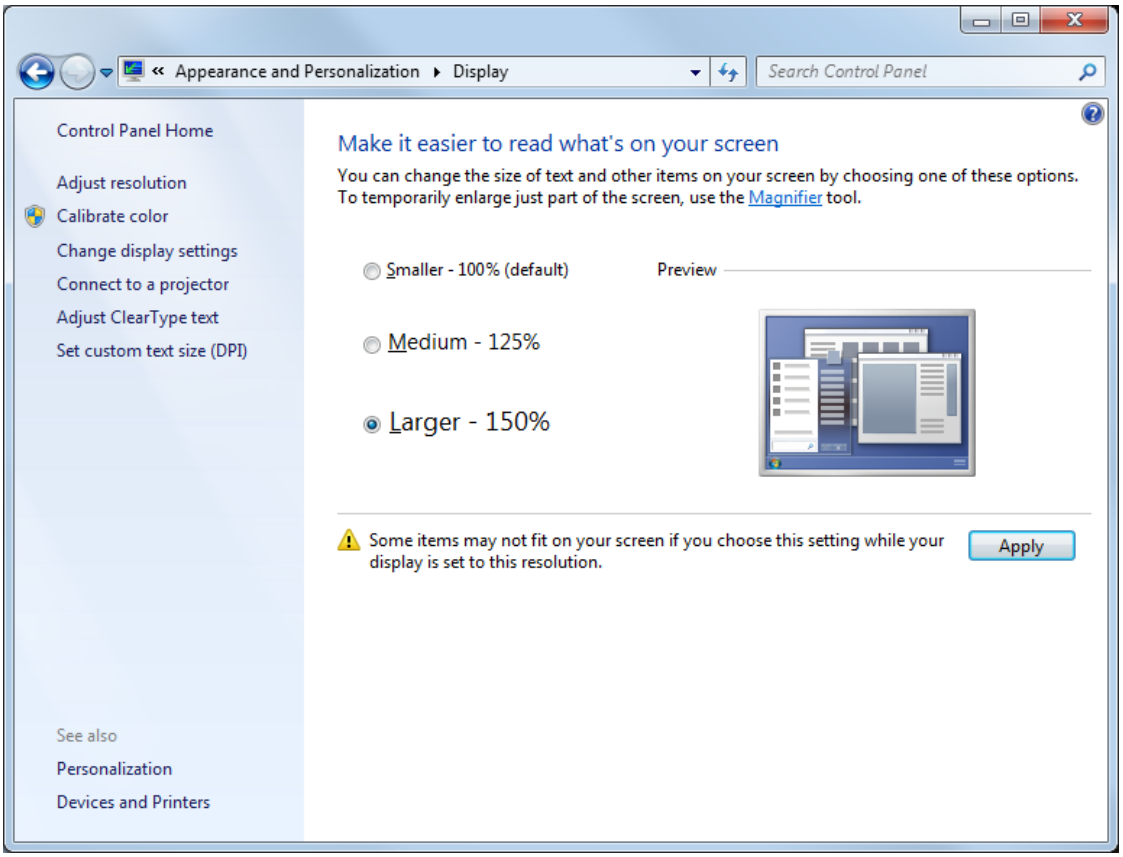
The hardware pixels on ultra-high resolution monitors are smaller than the pixels on regular monitors, creating a sharper image. Current web browsers adapt to this by displaying HTML pages with the appropriate zoom setting, but the Microsoft HTML Help Viewer does not.

While the end user can (temporarily) change the zoom in a HTML Help file, this does not really solve the problem. The zoom in HTML Help works differently.

The necessary fixes can be applied with a standard HTML editor and the next chapters will show, how to do that. But a help authoring suite like Help & Manual can save you a lot of time in this case.

2 Make text and other items larger

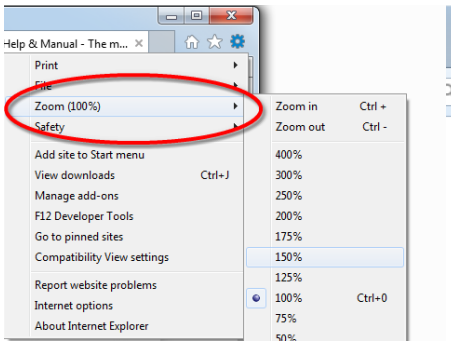
The hardware pixels on an ultra-high resolution monitor are so small, that text that is displayed at "normal" sizes, would be barely readable. To compensate this, Windows offers a display setting to "*Make it easier to read what's on your screen*":



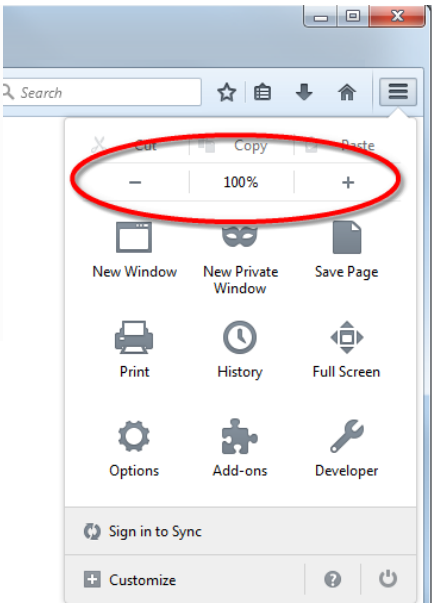
Windows control panel display settings

The result of this setting is that application windows, icons and texts will be enlarged on the screen, while at the same time the fine raster of hardware pixels ensures sharper rendering of the content.

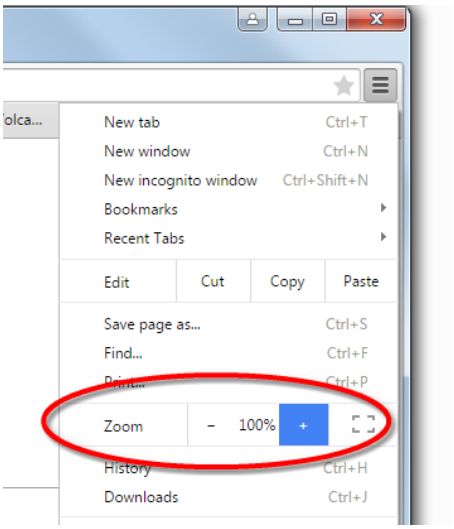
Web browsers adjust to this setting by displaying HTML pages with a zoom factor, that normally matches the text magnification of Windows (though it can be changed in the browser).



Zoom in Internet Explorer



Zoom in Firefox



Zoom in Google Chrome

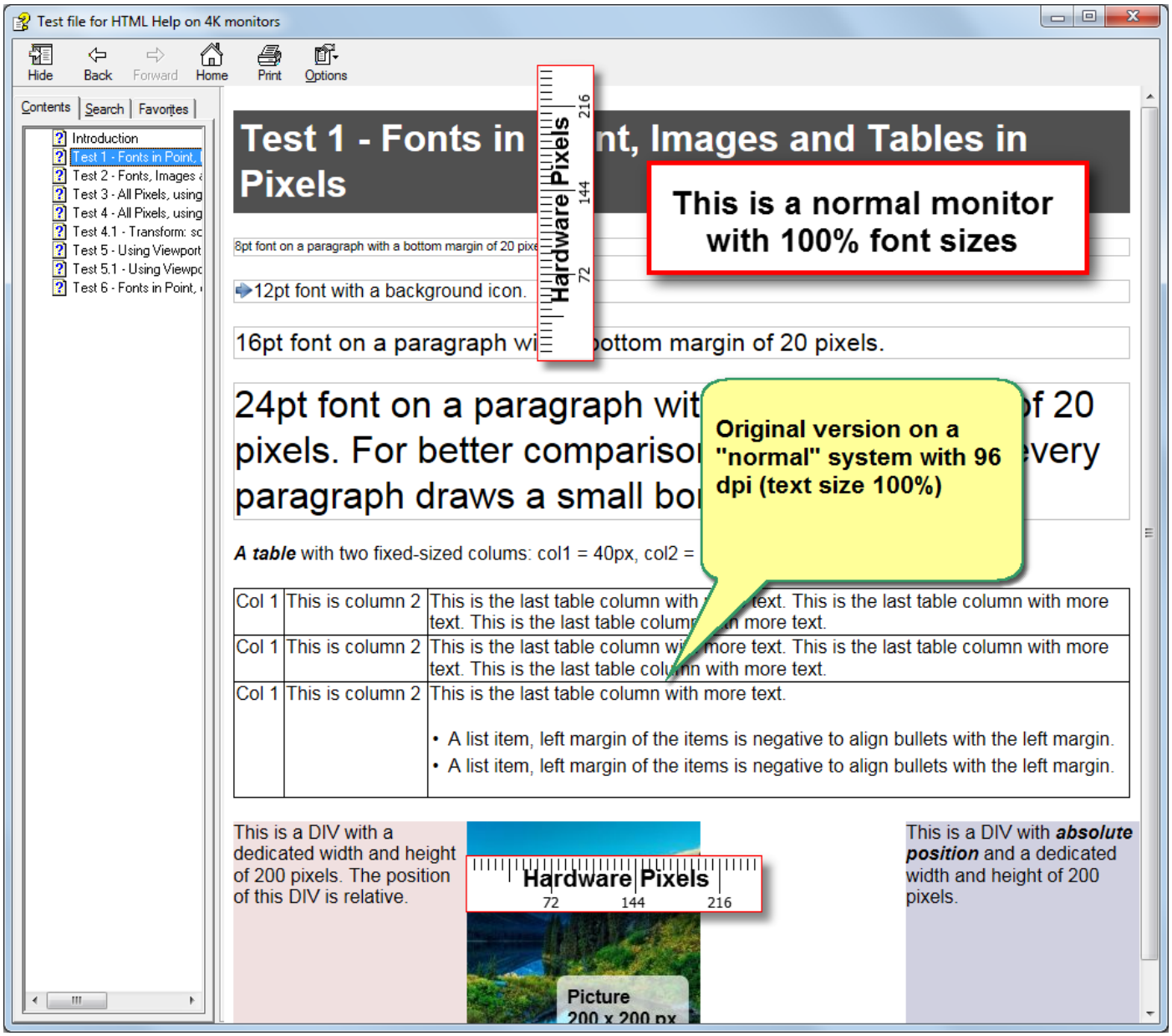
You normally don't worry about the zoom, because the browser handles it correctly. No matter how a HTML page is encoded, if it uses font sizes in pixels, point, inches or another measuring unit, the zoom will do it right. The browser's zoom is in fact a complex feature and we will examine some details in the following two chapters, when we compare the display of a decent web browser with the display of the same HTML page inside the Microsoft HTML Help Viewer.

The Microsoft HTML Help Viewer behaves differently than a stand-alone browser. It does not have a zoom button and, unfortunately, defaults to a 100% zoom, even if the "make everything larger" setting in Windows is set to 150% or 200%.

3 No zoom in HTML Help

This tutorial comes with a test HTML Help file "**HTML-Help-Test-For-4K.chm**" including source code. When you open this help file on a Windows system with "normal fonts" (100%), all test pages in this help file will look exactly the same.

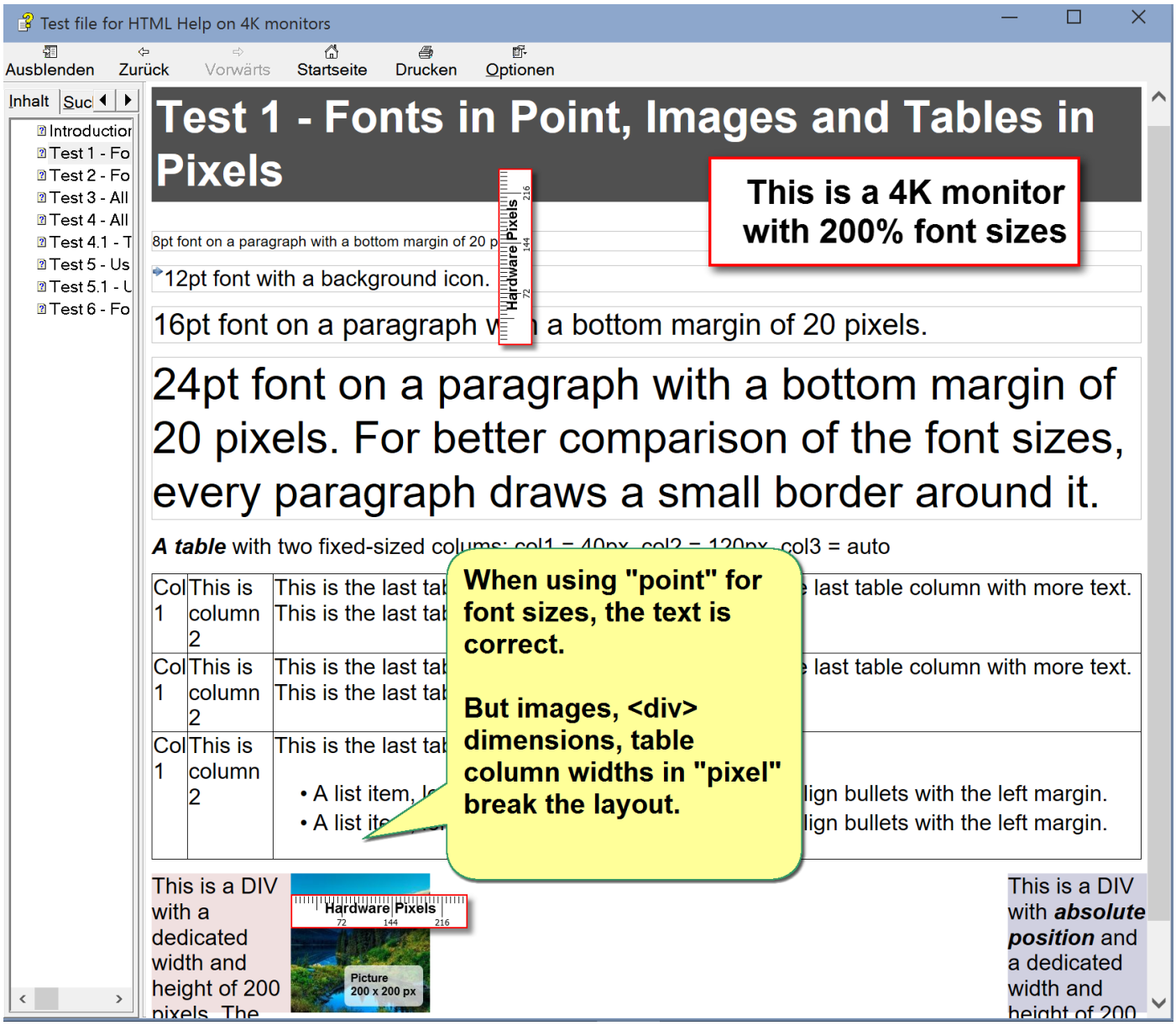
They look like this:



A text with a 16-point font will look like a 16-point font normally looks like (it is 20 hardware pixels high) and a element or a picture with a width of "200px" will be 200 hardware pixels. Table columns with fixed sizes approximately match the size of the text.

If you view this HTML page in a stand-alone browser, it will look the same. When you increase the browser's zoom - and this will be the case on an ultra-high resolution display - everything in the window will be enlarged: text sizes, pictures, boxes with "200px" width will become 300 or 400 hardware pixels wide. In other words: the browser retains the desired layout, it *scales everything proportionally*.

And here's how the same HTML Help file looks on a Windows system with 200% font sizes:

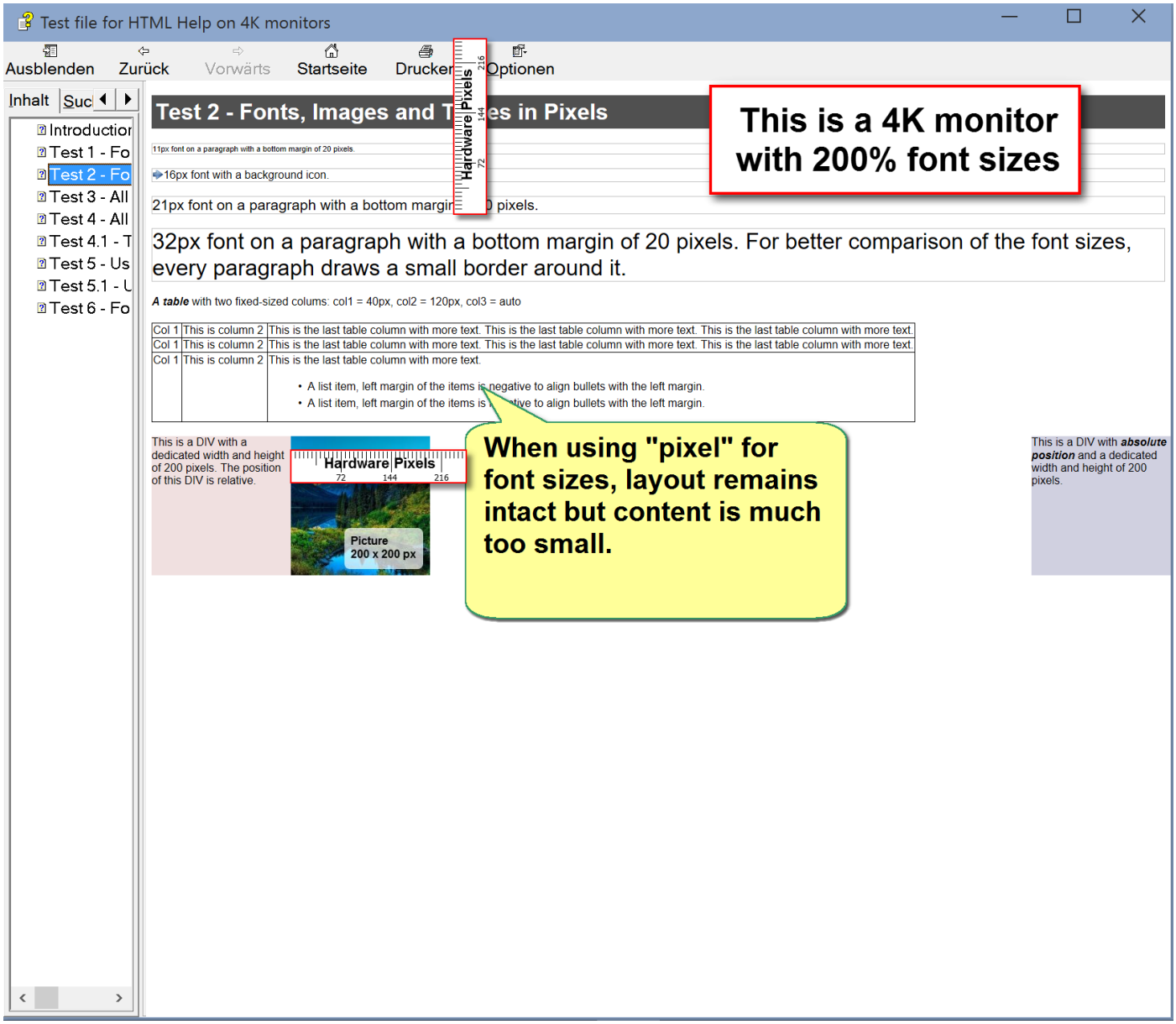


CHM help page with font sizes in point

Look closely at the screenshot above... font sizes in "point" are as large as they should be. Why a font size specified in point automatically gets larger - more on that later. But all items that are specified with "pixel" values (typically images, client-side image hotspots, table column widths, fixed-positioned HTML elements, you name it) are still displayed in **hardware pixels**! Except for relatively simple texts, this will break the layout in most cases and, of course, images and screenshots in the help pages are much too small.

A stand-alone browser would also enlarge all "pixel"-sizes and scale the content proportionally. HTML Help, unfortunately, does not.

Would we be better off if we specified font sizes in pixels as well? Let's see how this looks like...



CHM help page with font sizes in pixel

That's even worse! When everything is in pixels including font sizes, the layout stays intact, but it's barely readable, because the "pixels" are hardware pixels! In the second screenshot it really becomes obvious that *in a HTML Help file there is no zoom*.

What's happening?

To understand HTML Help, you need to know that the **Microsoft HTML Help Viewer** hosts an ***embedded version of Internet Explorer*** to display HTML help pages. Even if the user's default web browser is Firefox, Opera or Chrome, in HTML Help it's always Internet Explorer (MSIE) that displays the help page. HTML Help uses the MSIE that it finds on the system. This might be IE7 on a Windows XP system, or it might be IE11 on Windows 8.1.

Regardless of the MSIE version installed, the help pages in HTML Help have *three things in common*:

1. It is Internet Explorer 7 that you are looking at
2. The zoom of the embedded MSIE defaults to 100%
3. The zoom works differently than the zoom in a stand-alone browser

Embedded IE = IE7

The Internet Explorer embedded in HTML Help always runs in IE7 compatibility mode! It could be older as well. But even if MSIE is the default web browser on the system and the user has installed the latest MSIE 11.0.98.*something*, the embedded viewer displays the HTML page as if it was IE7.

This is important to know! When looking for solutions for the missing zoom later, we are going to explore CSS tweaks, media queries, scripting and other options, that may not be supported by IE7. To make the embedded IE behave like a more modern version of MSIE, we can elevate it with a HTML meta tag.

To elevate the embedded IE add a `<meta http-equiv="X-UA-Compatible" ...>` tag to the header for each HTML page in your CHM file.

Example:

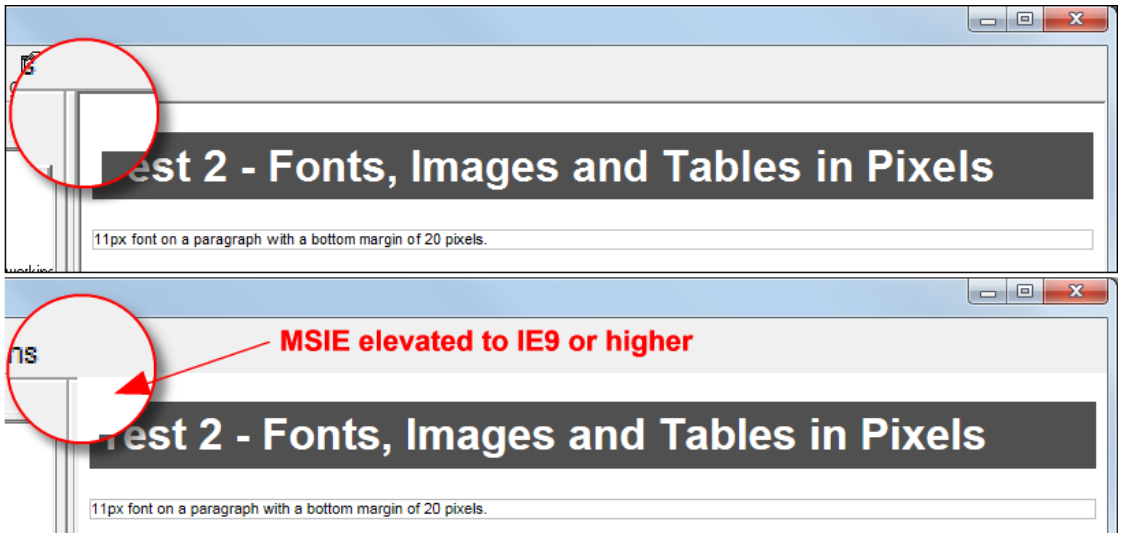
```
<!DOCTYPE HTML>
<html>
<head>
   <title>Test HTML page</title>
   <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
   <meta http-equiv="X-UA-Compatible" content="IE=8" />
   ...
```

The above elevates MSIE to IE8. IE9 and IE11 are similar:

```
<meta http-equiv="X-UA-Compatible" content="IE=9" />

<meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

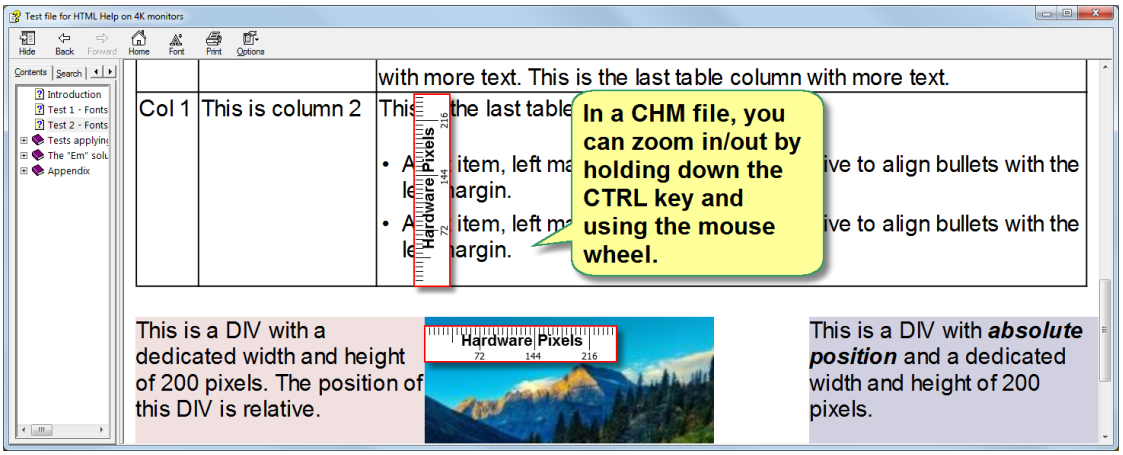
Why would you elevate the embedded browser to *IE8* and not to *IE9* or *IE-edge*? As soon as you elevate it to IE9 or higher, you will notice a small optical glitch - the frameless HTML viewer overlaps the boundaries of its parent panel on all 4 sides:



Is this a critical issue? The HTML Help viewer has never been a piece of beauty, after all! I do not know if it is critical and in all the years of working with HTML Help, I've never come across any other negative side effects, but you never know. IE8 has most of the features we need in our search for a zoom solution. IE9 and IE-edge are almost the same. The optical also occurs with some combinations of Internet Explorer 10 and Windows 8. If you don't mind the optical glitch, go for it.

Zoom always defaults to 100%

This *is* a critical issue! When you open the help file, the embedded IE defaults to a zoom of 100%. This is clearly visible in the screenshot above titled "*CHM help page with font sizes in pixel*". Is there no zoom at all? Oh, there is! Click into the HTML viewer area, then hold down the CTRL key and scroll the mouse wheel. It will zoom like a charm, at least if the MSIE version installed is newer than IE8.

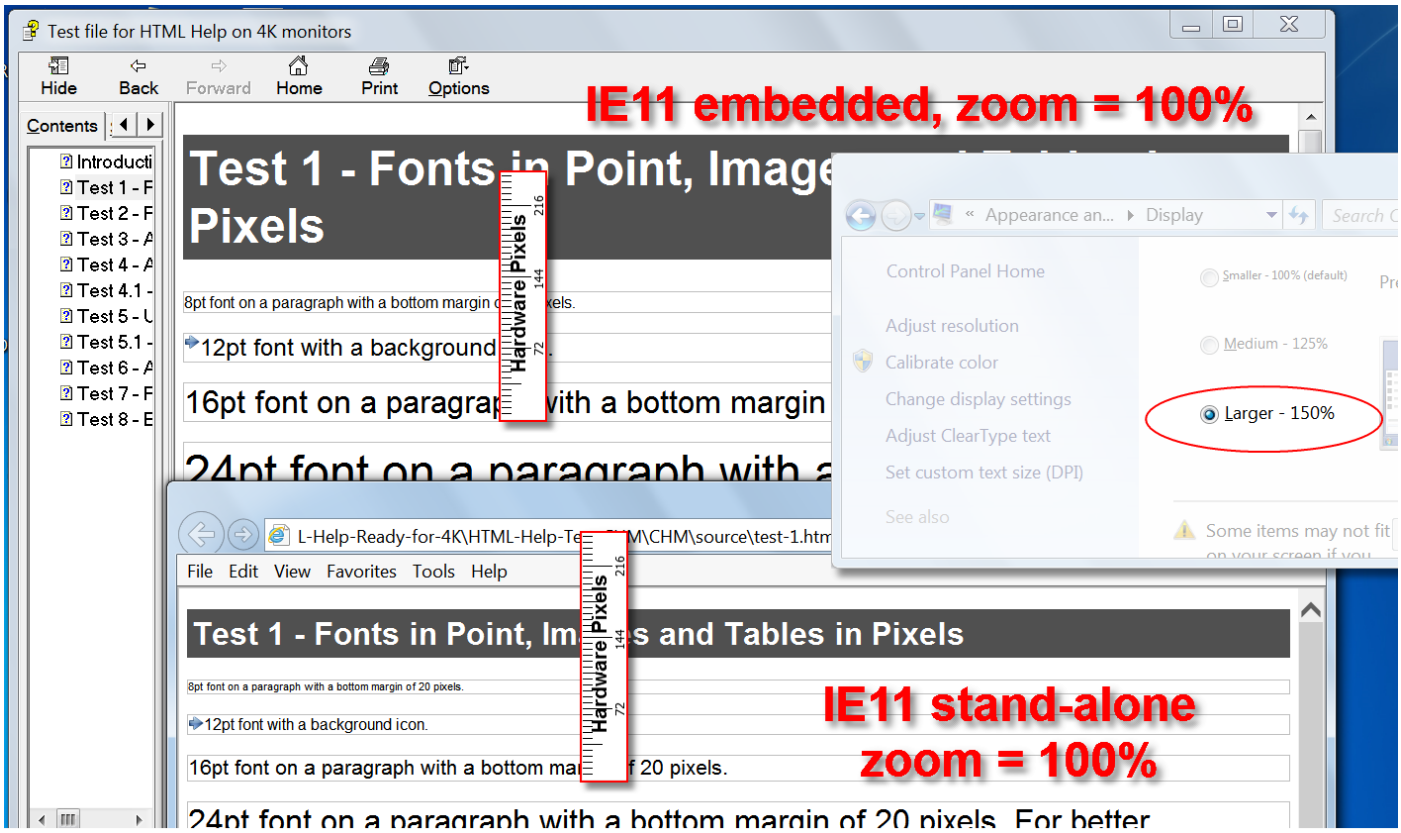


HTML Help on a "normal" monitor, manually zoomed in

However, the user needs (a) to know about CTRL+mouse and (b) re-adjust the zoom every time the help file gets re-opened. Unlike stand-alone MSIE, which saves the zoom value in the registry, the HTML Help Viewer forgets the temporary zoom and defaults to 100% the next time the help file gets opened. *Plus...*

Zoom behaves differently

This is another critical issue. When **font sizes are specified in point**, the **text appears enlarged**, although the zoom defaults to 100%. This is normal, you would think. But look how the very same HTML page is displayed in a stand-alone version of the same MSIE on the same system, also set to a zoom of 100%:



On a system with 150% text sizes (Windows setting), embedded IE and stand-alone IE display the same HTML page. Fonts are specified in point.

The stand-alone version of MSIE does **not enlarge text when fonts are specified in point**! It zooms the entire layout, no matter if fonts are specified in point or pixel. This ensures that text sizes and layout always match and never break. This little detail will become important in our search for a solution and is explained in the next chapter.

Quick Summary

The embedded MSIE in HTML Help behaves different than MSIE as a stand-alone browser. This affects the display size of text, images, and other layout elements.

In HTML Help, MSIE always starts with a zoom of 100%, even if this is much too small for a high-resolution display. You can change the zoom temporarily, but there is no way of storing it permanently. The display of fonts specified in point also differs from the stand-alone browser. This has an effect on the HTML Help zoom solutions we are going to explore. On a high-resolution monitor, the content is either too small or the layout is most likely broken.

4 About points, pixels and DPI

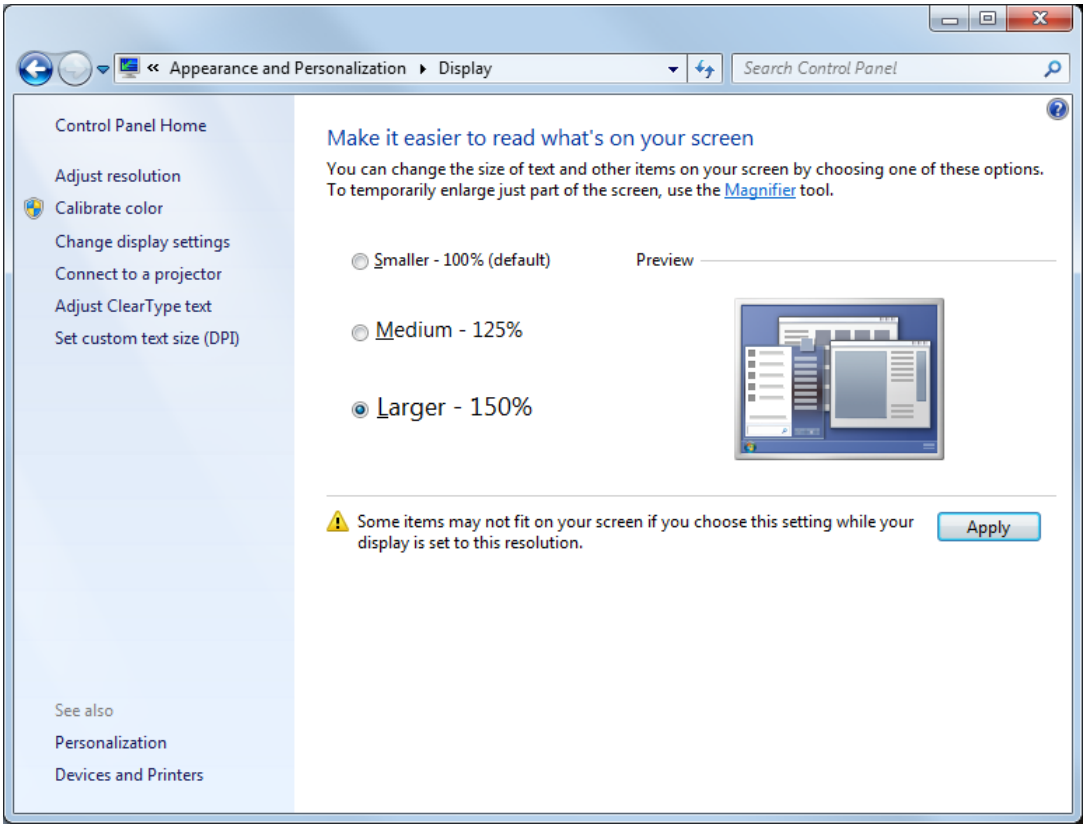
How much is an inch?

For a computer display, the physical inch is defined by the diagonal of the display. 24" inch is nice, 30" is better. Your flat screen TV probably has 50" or more. How many pixels the monitor can fit in a physical inch, defines the display resolution.

This is called ***dots-per-inch*** or ***DPI***. A Samsung 23" HD display (1920 x 1080) has 96 DPI, a 24" Dell Ultra-Sharp UP2414Q (3840 x 2160) has 184 DPI. This link helps you to calculate the DPI value of your own display.

How much is an inch in Windows?

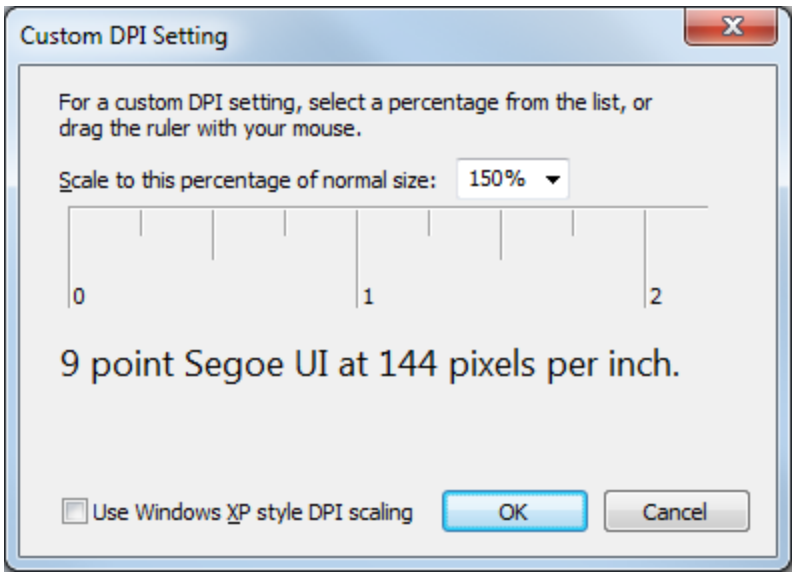
It depends. Normally, it is 96 pixels exact[1]. Regardless of the display. Windows has its own *logical DPI* setting, *which is independent from the physical size of the monitor*. The default is 96 DPI (this is 100%) and you can change this in the control panel's Display settings:



Windows control panel with logical DPI settings

Windows DPI values		
Setting in %	**DPI**	**Value is typical for**
100%	96	Standard monitors. In earlier versions of Windows, this setting was called "Normal fonts"
125%	120	Normal monitors for people with weaker eye-sight. In earlier versions of Windows, this setting was called "Large fonts"
150%	144	Large ultra-high resolution desktop monitors ("4K displays)
200%	192	Ultra-high resolution displays on smaller devices like the Microsoft Surface 3 tablet

This logical versus physical DPI is confusing, isn't it? Sure enough, it is. And for this reason, Microsoft meanwhile uses a different wording in order to not confuse consumers. They call it "*Make text larger or smaller*", which is a pretty accurate description of what the logical DPI value does. Nonetheless, you are still served the "DPI" term when you click "*Set custom text size*" in the dialog above.



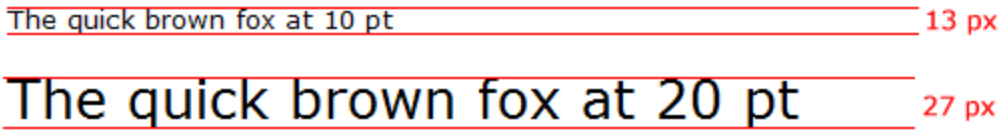
Dialog for a custom logical DPI value

In the picture above, the text size is set to 150%. The default DPI of 96 multiplied by 1.5 results in a new (logical) DPI value of 144. So, does this mean that Windows squeezes more pixels into one physical inch of the display? Of course not. It changes the formula, how text sizes are calculated.

Text is measured differently

Text is measured in *point* - that's also the value that you might use to define a font in a CSS style sheet. A *point* is defined as 1/72 inch. Always, everywhere. Point is a hard-wired entity like 1 inch = 1/12 of a foot.

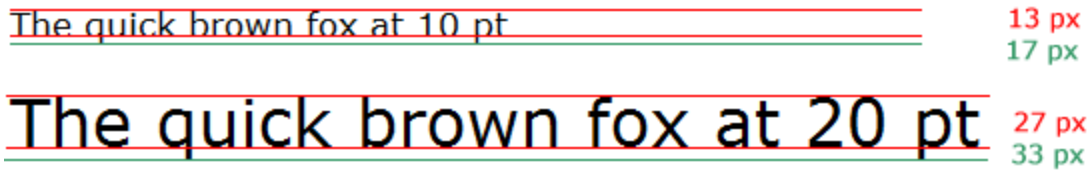
Take a 10 pt font for example. The height of the font will be 10/72 inches. When an inch has 96 DPI (or pixels), the font will be 10/72*96 pixels high. That's a rounded height of 13 pixels on screen. A 20 pt font is 20/72*96 = 27 pixels rounded.



10pt and 20pt fonts on a 96 dpi setting (Windows default)

So, what happens, when we *change the definition of an inch*? If our inch should be composed of 120 pixels instead of the default of 96? In Windows terms, this is a "medium" font size or 125% (compare with the control panel screenshot above).

If we define inch to be 120 pixels, the 10 pt font will result in a height of 17 pixels (10/72*120 = 17) and the 20 pt font will be 33 pixels (20/72*120 = 33). And because a pixel in Windows still means a physical pixel on the display, the result is bigger text, when measured in pixels.



10pt and 20pt fonts at 120 dpi (in Windows: 125%)

If a display has a high *physical* DPI value (= the pyhsical pixels are quite small), a regular *logical* DPI of 96 in Windows results in text that is so small, that it becomes hard to read. By increasing the *logical* DPI in Windows, text becomes larger in pixels, restoring a comfortable reading size on screen *and* a smoother font, because it's made up of more pixels.

Web browsers render with device-independent pixels

If you designed web sites in the late 1990ies, you certainly remember a puzzling difference in the layout of a web page, that looked pixel-perfect on Windows, but awkward in a Mac browser and vice versa. Because images and boxes and the layout in general were defined in pixels, the fonts in point. On Macs, font sizes

were calculated the same way as in Windows: pixel = point / 72 * DPI. Because DPI was 72 by default, points were effectively pixels on a Mac, but 1/3 larger on Windows. The point-based font sizes blew up every pixel-perfect web design.

That's why designers started to *use pixel for font sizes*, leading to lengthly discussions over the usability of web sites with fixed-sized fonts. Today, this is a thing of the past. It doesn't matter anymore, whether you define the font size for a HTML page in pixel, point or any other entity.

For a web browser, a CSS value specified in pixels ("px") is a *device-independent* pixel. For a Windows browser, an *inch always has 96 device-independent pixels*. If you specifiy a point "pt" value (say, 72 pt = 1 inch) , the browser calculates the text size in pixel based on 96 DPI, so the result is: 96 pixels on screen! But this pixel size is device-independent and represents a zoom of 100%. The browser will scale the pixels anyway, depending on the browser's zoom. Other CSS units of measurement like "in" (inch) "cm" (centimeter) are calculated the same way. This technique makes *pixel* and *point* CSS sizes proportional and has ended the usability discussions.

HTML Help renders with hardware-pixels

For the embedded MSIE in HTML Help, a CSS value specified as "px" is **not** a logical pixel, but a physical one! Because it defaults to a 100% zoom, "1px" in a CSS stylesheet is 1 hardware-pixel on screen. On an ultra-high resolution monitor, such a pixel is quite small. Point "pt" sizes, on the other hand, are calculated relative to the system-wide DPI setting.

The table below shows the calculation differences between a stand-alone browser and HTML Help. In HTML Help, mixing pixels ("px") and other CSS units like point ("pt") or inch ("in") on the same HTML page will not work, because HTML Help treats "px" as an absolute value, whereas "pt", "in" or "em" are calculated taking the system-wide Windows DPI into account.

On a Windows system with *192 DPI* (200%)			
CSS value	**Calculation**	**Browser zoom**	**Resulting size on screen**
Web browser calculates			
"100px"	= 100 device-independent pixels (dip)	200%	200 pixels
"12pt"	12/72***96** = 16 pixels (dip)	200%	32 pixels
"1.3in"	1.3***96** = 124.8 pixels (dip)	200%	250 pixels
HTML Help calculates			
"100px"	= 100 hardware pixels	100%	**100** pixels
"12pt"	12/72***192** = 32 pixels	100%	32 pixels
"1.3in"	1.3***192** = 250 pixels	100%	250 pixels
HTML Help calculates (if browser zoom is increased to 200%)			
"100px"	= 100 hardware pixels	200%	200 pixels
"12pt"	12/72***192** = 32 pixels	200%	**64** pixels
"1.3in"	1.3***192** = 250 pixels	200%	**500** pixels

Long story short ... even if we manage to automatically increase the default zoom of the embedded MSIE in HTML Help to the system-wide Windows DPI value (and we will do that), we will never get a coherent layout with a mix of "px" and "pt".

Why does HTML Help calculate sizes this way? This is pure speculation, but I guess, it's a legacy thing. Keep in mind that the CHM format was introduced in 1998 and has not changed since. The HTML Help viewer got a couple of updates over time, but none recently. There are still thousands of CHM files on the market, coded at at time when browsers did not have a zoom and people were arguing over the political correctness of encoding font sizes in pixels. So, HTML Help is still calculating the layout of a page like browsers did it in the old days of the Internet. Changing this behaviour would break them.

On the other hand, Microsoft has introduced so many awkward proprietary HTML tags over the years ("MOTW" *mark-of-the-web*, anyone?), that it wouldn't be asked too much to have one more of those tags for online help. My favorite would be a proprietary <meta> tag for CHM files:

```
<meta http-equiv="X-UA-Compatible" content="IE-ignore-and-behave-like-a-browser!"/>
```

But that's wishful thinking! 4K monitors are here, they are going to stay and HTML Help is not ready. Yet it is still the standard documentation format for Windows desktop applications. *Let's pimp it up!*

Quick Summary

In HTML and CSS, you can specify font sizes in pixels, point, or other CSS units of measurement, like "em". Text sizes in point are *normally* relative to the Windows system-wide DPI value. But web browsers use their own zoom functions and calculate a "point" based on 96 pixel, on order to not break a coherent layout mix of "px", "pt" and other units.

The embedded MSIE in HTML help, however, calculates a point based on the Windows DPI setting. If we want to have coherent text sizes and layout in HTML Help, we must not mix CSS pixel values "px" with other CSS units like "pt".

[1)] Where 96 DPI in Windows comes from

You might ask yourself, why 96 pixels is the default for a logical inch on Windows. Would it not be more intuitive, if a logical inch was defined as 72 pixels, so that 1 pixel is 1/72 inch, similar to the *point* entity? On Mac OS/X this is exactly the case. This article explains the 96 DPI mystery.

Implementing zoom in HTML Help

How can we make HTML Help zoom the content automatically? There are different ways to accomplish that and this chapter will explain the options we have.

Picture courtesy of pixabay.com

5 Two ways to look at it

As I have pointed out here, the zoom in HTML Help is different compared to the regular zoom function of a stand-alone browser. To get a consistent layout, we must not mix CSS "px" values with other entities like "pt", "em" or "in". That's true for both the solutions I am going to propose.

But we can look at the problem in two ways:

Automatically apply a browser zoom

Zoom is a straight-forward and simply solution that requires additional CSS settings, but otherwise no changes to the HTML code.

When we apply a zoom factor to the embedded IE window, **font sizes must be encoded with *pixels*** (CSS value "px"). Fonts in point values are already zoomed by HTML Help and applying a zoom to the entire page would double the magnification of the text.

Change all pixel values to relative units

This is the more sophisticated solution, but may require changes to a large part of your HTML code base.

If font sizes are defined in point (CSS value "pt") and it is not possible or desirable to change this to pixels, the only other solution we have is to **change other pixel settings** (image sizes, margins, absolute positions, etc) to **a relative encoding** that is scaled in a similar way as the point values.

The goal is to modify existing HTML Help files and make them compatible for 4K monitors, with the least amount of work possible. Depending on your HTML code base and how the code is structured - inline CSS or global style sheets, hard-coded layouts or responsive design - you might prefer the one or the other way to do it.

Test pages in the accompanying demo CHM file	
Test 1	This is a typical HTML page with fonts defined in point. Paragraph margins, images, table columns, fixed s are defined in pixels. The text of this page will scale in HTML Help on a high-resolution monitor. But other elements like pictures will not scale and appear too small.
Test 2	This is the same HTML page, but fonts are defined in pixels as well. This page will not scale at all. On a high resolution monitor, it will look similar to the screenshot shown in Suddenly broken.
Test 3	This is the same as Test 2, but the CSS on high-resolution monitors applies a zoom factor. The test page is explained here.
Test 4	This is the same as Test 2, but the CSS on high-resolution monitors changes the viewport. Description
Test 5	This is the same as Test 2, but the CSS on high-resolution monitors applies a scale transform.

Test pages in the accompanying demo CHM file	
	Description
Test 6	This is the same as Test 5 with modifications for print. Description
Test 7	This test page is based on Test 1, with changes to the HTML code. The test is described here.
Test 8	This test page is similar to Test 7 with modifications of the CSS. Description
Test 9	This test page is similar to Test 7 with modifications of the CSS. Description
Appendix The following 2 tests explore ways to change the zoom with Javascript and VBScript. They are not described in this tutorial, because they do not have any advantage over the CSS modifications used in the test pages 3 - 9. I have included them as a proof-of-concept.	
Test 10	Applying a scale transform with Javascript
Test 11	Changing the (real) browser zoom with VBScript

Testing HTML Help without a 4K monitor

An ultra-high resolution monitor is a gorgeous thing to look at. If you can afford one, get it! But you can test HTML Help for high resolutions without such a monitor as well.

The key difference between a normal and a "4K" monitor is: hardware pixels are smaller and without a magnification text would be barely readable. Windows has a system-wide setting called "Make text and other items smaller or larger". From a software's point of view, this is the major difference between a normal and an ultra-high resolution system. Set up a virtual machine for testing and change this system-wide setting to 200%. You will need to log off and on again after the change.

At 200%, on a normal monitor with a resolution of, say 1920 x 1080 pixels, the content in the virtual machine will appear - well, big... and some items might not fit on the screen. However, it is sufficient for testing. Increasing to just 150% might work as well. But I found 150% difficult, because the size differences we are going to explore are not as obvious with 150%. At 200%, it becomes crystal clear what's wrong.

6 Applying zoom

My first attempt to implement HTML Help zoom was - you guessed it - setting the `Zoom` property. This is a very old CSS property that only Internet Explorer understands. But in HTML Help, we are dealing with IE anway.

Using the zoom CSS property

The proprietary `zoom` CSS property was implemented with IE 5.5 already and when you apply it to the <body> tag, it scales the content of the entire HTML body.

Example CSS:

```
<style type="text/css">
   body {
      zoom: 2;    /* MSIE only */
      width: 50%
      background: #FFFFFF;
      font-family: Helvetica, Arial, sans-serif;
      font-size: 16px;
   }
</style>
```

In the style sheet, I set two properties: `zoom` magnifies the HTML <body> to 200% and at the same time, `width` reduces the maximum width of the HTML <body> to half the width of the browser window. Without setting width in combination with the zoom, the HTML page would be 200% wide, causing a horizontal scroll bar to appear. This is a clear indicator that we are *not really* setting the web browser's zoom, it is just a magnification of the <body> tag, an early *transform:scale()* in MSIE terms.

The `zoom` CSS property is not inherited, but will affect direct children of the <body> tag. HTML elements with absolute positioning (`position: absolute`) are not affected by the <body>'s `zoom` attribute. If you have any element with an absolute or fixed position in your HTML code, you will need to apply the zoom separately to these elements. My example pages contain one with absolut positioning to test this.

```
<style type="text/css">
   body {
      zoom: 2;    /* MSIE only */
      width: 50%
      background: #FFFFFF;
      font-family: Helvetica, Arial, sans-serif;
      font-size: 16px;
   }
   #absolutediv {
      zoom: 2;
      width: 50%
   }
</style>
```

But how do we know which zoom to apply at all? With media queries!

Media queries to find the right zoom value

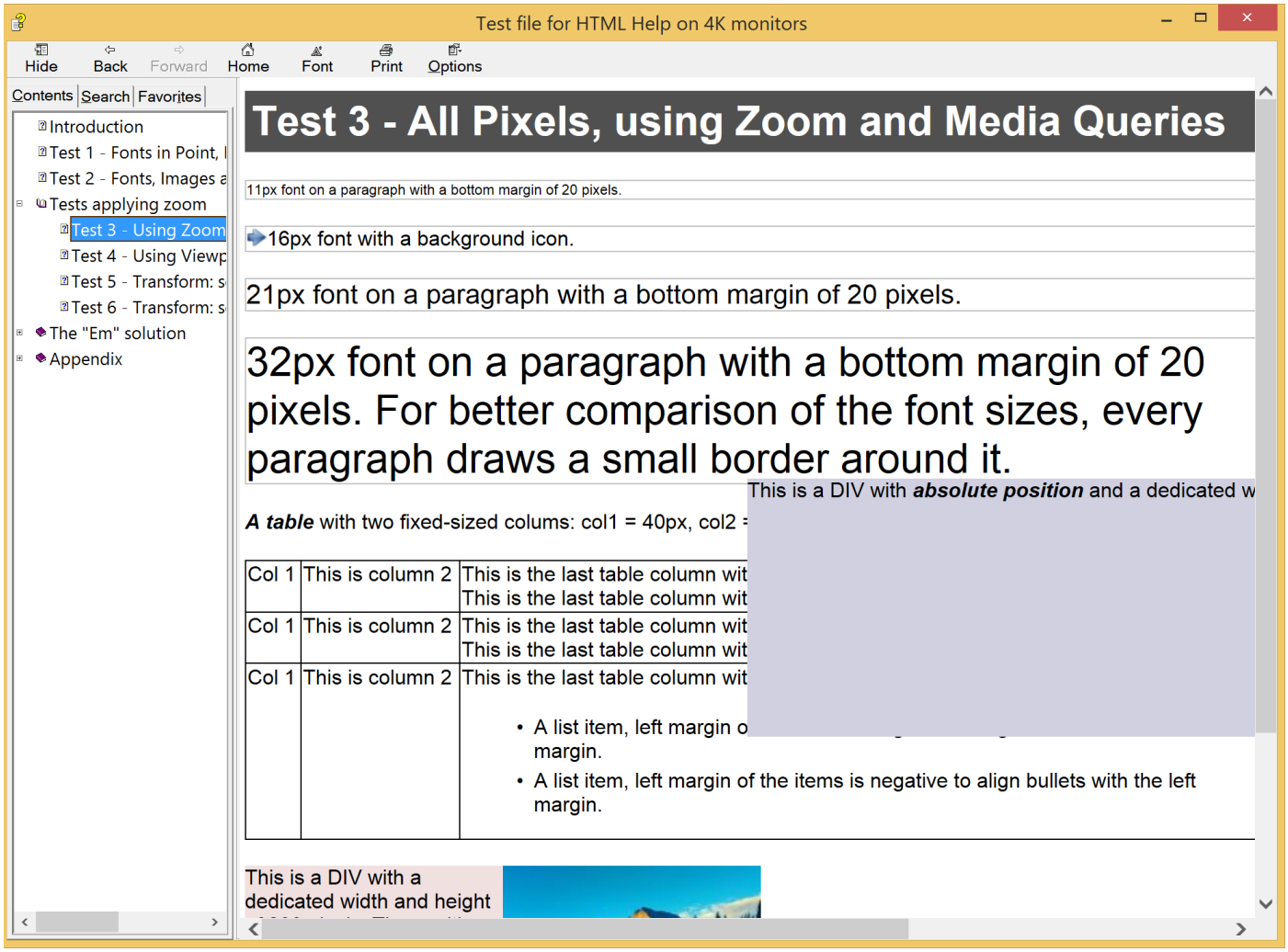
Media queries are supported in Internet Explorer from version 9 on. In a compiled HTML Help file, MSIE runs in IE7 compatibility mode by default. We need to elevate it to IE9 to be able to use media queries:

```
<!DOCTYPE HTML>
<html>
<head>
   <title>Test 3 - All Pixels, using Zoom and Media Queries</title>
   <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
   <meta http-equiv="X-UA-Compatible" content="IE=9" />
   <!-- IE9 is required for CSS media queries -->
```

Once this is in place, we can use media queries in the CSS below. In my example I've placed all CSS code directly in the HTML test pages to have a separate CSS for every page. You will certainly use a global CSS instead and link it to every HTML page. But the CSS is the same, whether it is inside the HTML page or external. This is the media query:

```
<style type="text/css">
    @media (min-resolution: 144dpi) {  /* setting for 150% */
      body { zoom: 1.5; width: 66.666% }
    }
    @media (min-resolution: 192dpi) {  /* setting for 200% */
      body { zoom: 2; width: 50% }
    }
</style>
```

The default CSS for the <body> tag does not specify a `zoom` value, it will default to 100% which is fine on a system with 96 DPI. If the media query delivers a screen with a higher DPI value than 96, we start to zoom in on the HTML body. That's how it looks like in the compiled CHM:



The <body> tag enlarged with the CSS zoom property

This is not satisfying, is it? What's wrong in the picture above is the absolute that does not work, although `zoom` is explicitely set for the absolute positioned as well. And the horizontal scrollbar is not desired, either. The `zoom` property looked promising, but it is obviously too old and not supported well. Let's try something else...

It's the viewport, stupid!

Why did I not think about this earlier? All those mobile devices desperately want a ***viewport*!** For a responsive website, setting the *viewport* is a must. Well, the difference to the `zoom` property above is subtle. Firstly, Internet Explorer supports the viewport CSS property in IE10 or higher. So if this CHM file is shown on a PC with MSIE9 installed, it will not work.

We need to elevate MSIE to IE10 or IE-edge to be able to change the viewport:

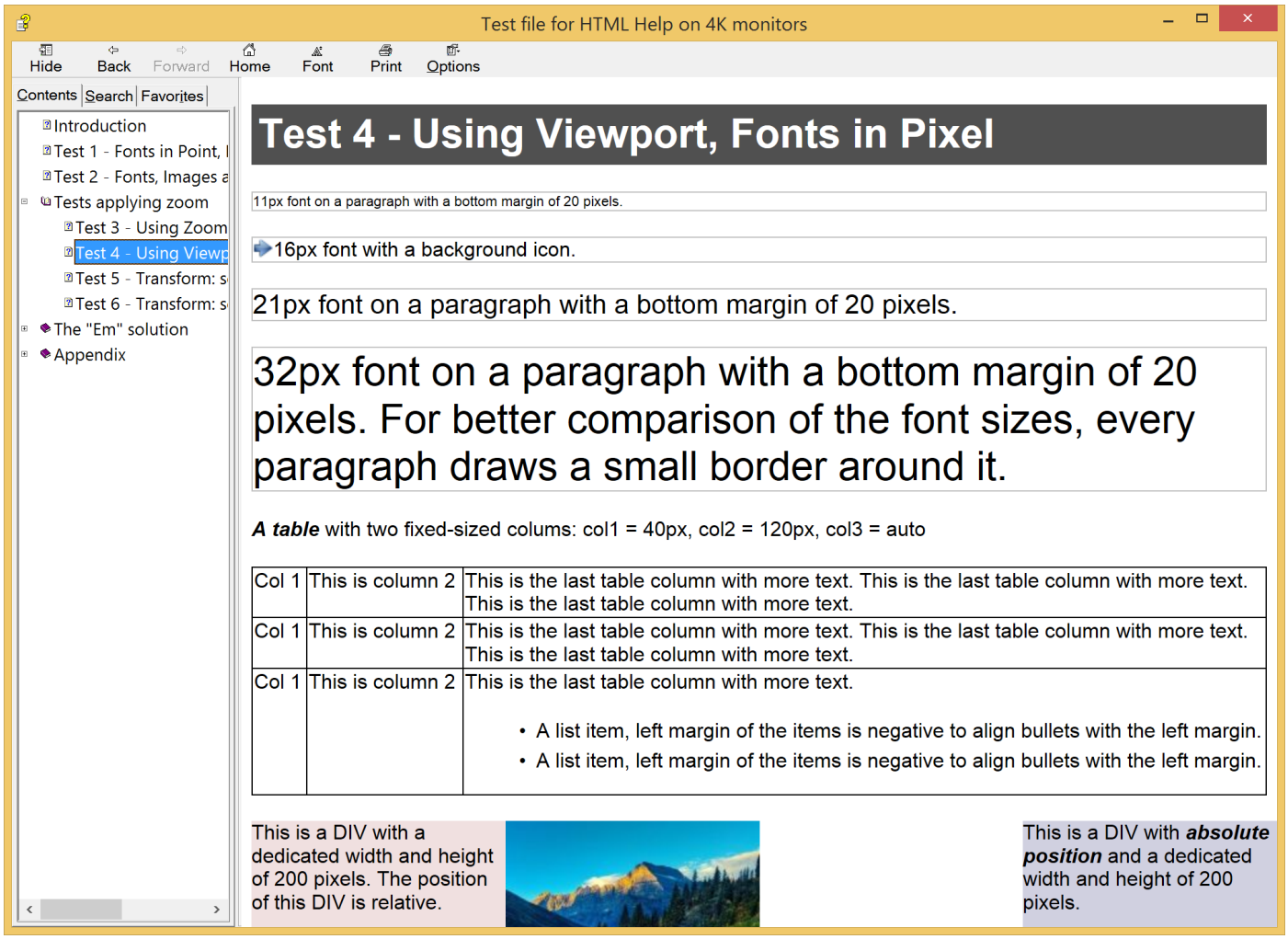
```
<!DOCTYPE HTML>
<html>
<head>
   <title>Test 4 - Using Viewport, Fonts in Pixel</title>
   <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
   <meta http-equiv="X-UA-Compatible" content="IE=edge" />
```

Even IE10 supports the `viewport` with the MS-prefix, only. Media queries are supported with IE9 already, so the part of the media queries will work. When setting `viewport`, it's most often used with fixed widths. Something like "*if screen min-width is between 1024 and 1366 px, set the viewport to 1024px*". That would work for a HTML layout that requires a minimum browser width of 1024px.

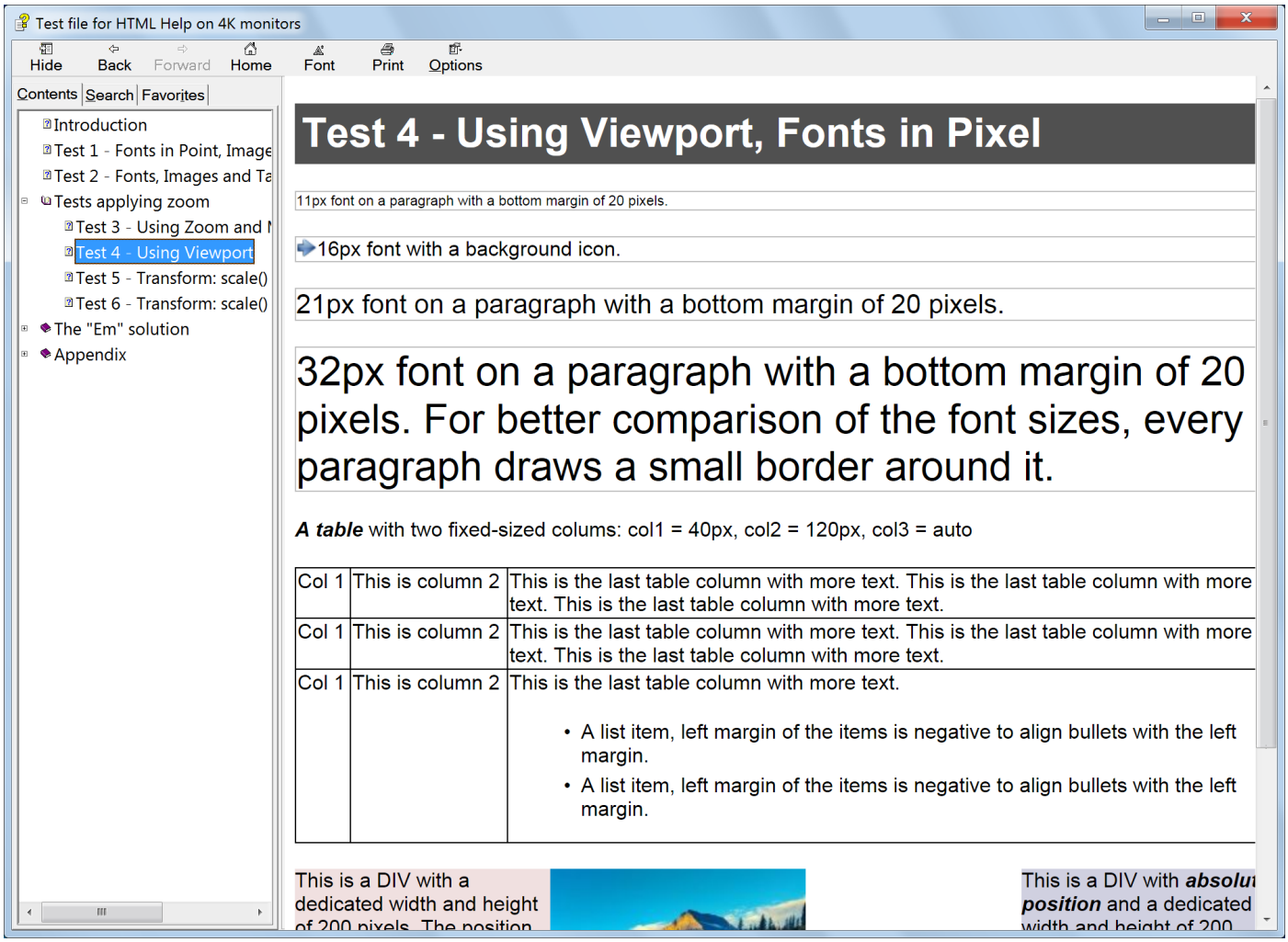
But the viewport also takes percentage values, which will scale the viewport relative to 100% browser width. Our new CSS is the following:

```
<style type="text/css">
    @media (min-resolution: 144dpi) {
      @-ms-viewport { width: 66.6667%; }
    }
    @media (min-resolution: 192dpi) {
      @-ms-viewport { width: 50%; }
    }
</style>
```

And this is how it looks like:



The <body> tag enlarged with the CSS viewport property (Windows 8.1 at 192 DPI)



The <body> tag enlarged with the CSS viewport property (Windows 7 at 192 DPI)

That's much better, isn't? Here, the absolute is correct. We don't need to add extra CSS for absolute positioned elements and the vertical position of the absolute is where it should be!

Have we found our zoom method? Take a close look at the screenshot above - there is something wrong with the right margin. On Windows 8.1, the vertical scrollbar is missing. And on Windows 7, the body tag extends to the very right border of the window and part of it is hidden by the vertical scrollbar. I was able to work around (most) of these glitches with a little extra CSS, but why bother with margins, when we can do a CSS3 transformation as well?

Using Transform: scale()

Surprisingly, CSS3 transforms (the 2D transforms) - were supported in IE9 already, though with the "MS-" prefix only. Along with media queries. We need to elevate it to IE9 to be able to use both:

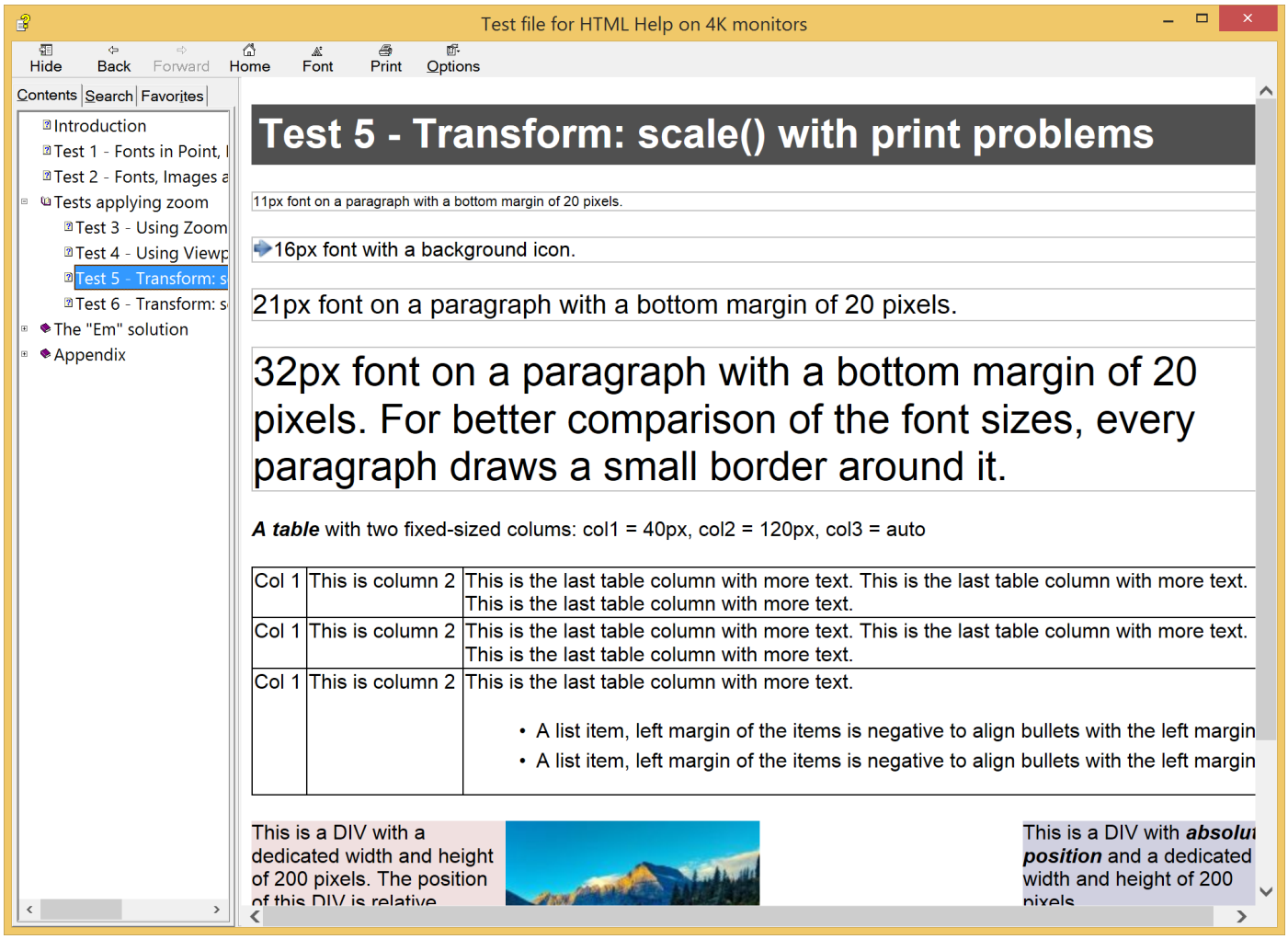
```
<!DOCTYPE HTML>
<html>
<head>
   <title>Test 5 - All Pixels, using Transform: scale()</title>
   <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
   <meta http-equiv="X-UA-Compatible" content="IE=9" />
```

The CSS for the 2D-transformation is almost as simple as that for the `viewport` above. You could apply the transformation to the <body> tag or to the <html> root tag. Which one is a matter of taste and might depend on the remaining HTML code. Here's why I prefer the <html> tag over <body>.

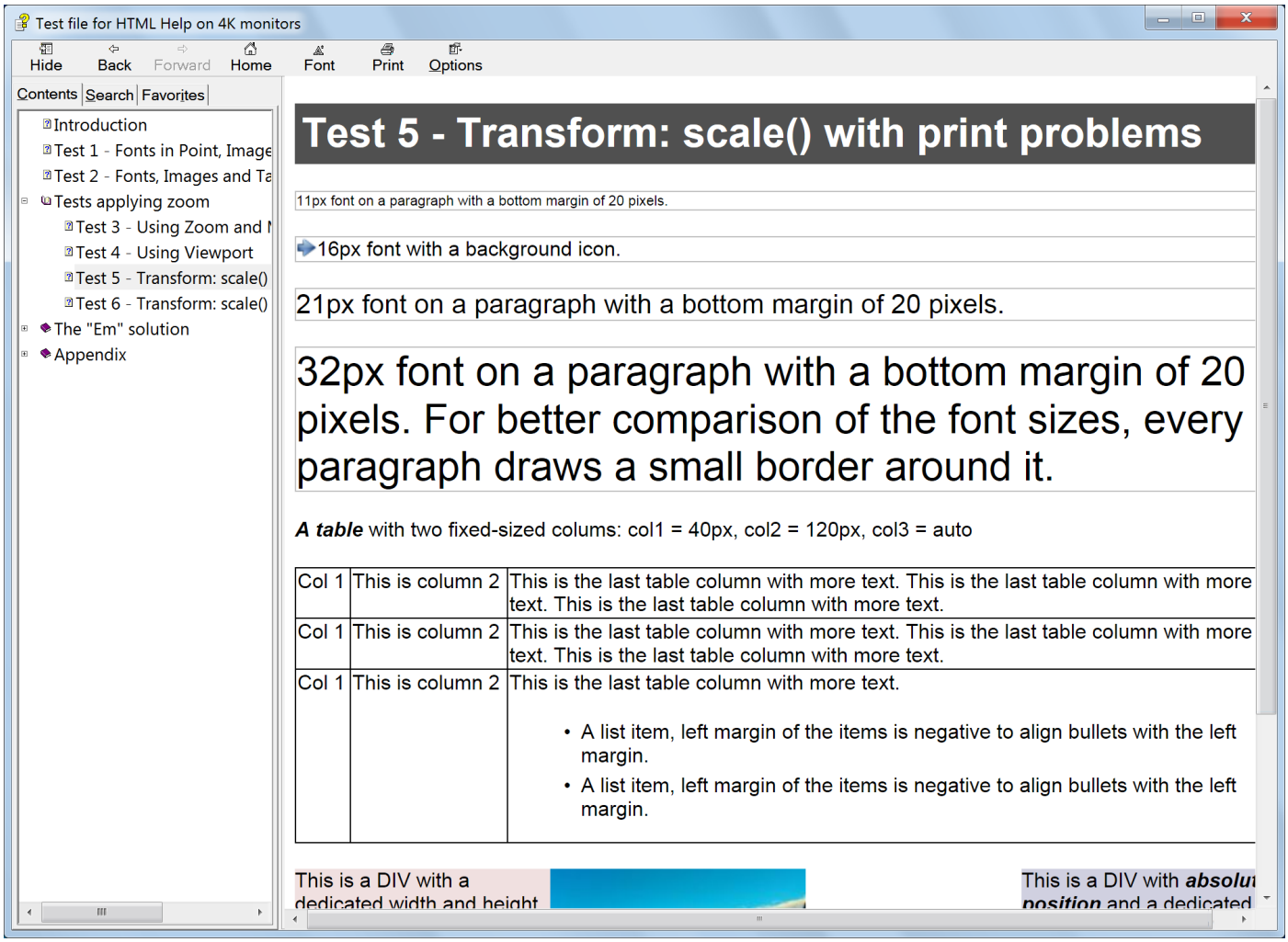
The `scale()` transformation uses a default origin of "50% 50%", so the center of the transformation is in the very center of the page. It will extend to negative margins to the top and the left, if we leave it this way. Setting the `transform-origin` to the top/left corner will produce the desired effect:

```
<style type="text/css">
    @media (min-resolution: 144dpi) {
      html {
          -ms-transform: scale(1.5);
          -ms-transform-origin: 0 0;
          width: 66.666%;
      }
    }
    @media (min-resolution: 192dpi) {
      html {
          -ms-transform: scale(2);
          -ms-transform-origin: 0 0;
          width: 50%;
      }
    }
</style>
```

Does it work on high resolutions? Pretty well!



The <body> tag enlarged with the Transform: Scale() property (Windows 8.1 at 192 DPI)



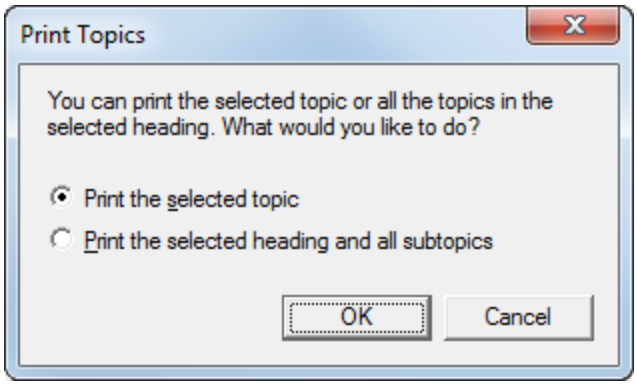
The <body> tag enlarged with the Transform: Scale() property (Windows 7 at 192 DPI)

Quick Summary

I have explained three ways to automatically zoom in on the entire <body> tag of a HTML page, using different CSS properties. Of the three options tested, the **Transform: Scale()** property yields the best results. Keep in mind that all three options require font sizes to be defined in *pixels*.

7 Printing problems

When all fonts are defined in pixels, the zoom solution with the CSS `transform:scale()` works quite well. On the screen. But how about printing? The HTML Help viewer has a pretty basic print function. The user can print a single topic or, if the selected TOC entry is a chapter, all topics in this chapter.



Print dialog in HTML Help

The print function of the HTML Help viewer is not very sophisticated, to say it politely. And it is notorious for various printing problems. But it can at least do a decent printout of single topics.

So I wanted to know if `transform:scale()` works with printouts as well. I compared a printout of the **Test-5 topic** on a system with 96 DPI (100%) with another printout of the same topic on a high-resolution system with 192 DPI (200%). They should both look approximately the same. And the more I was comparing, the more I become puzzled...

Something's broken with MSIE media queries

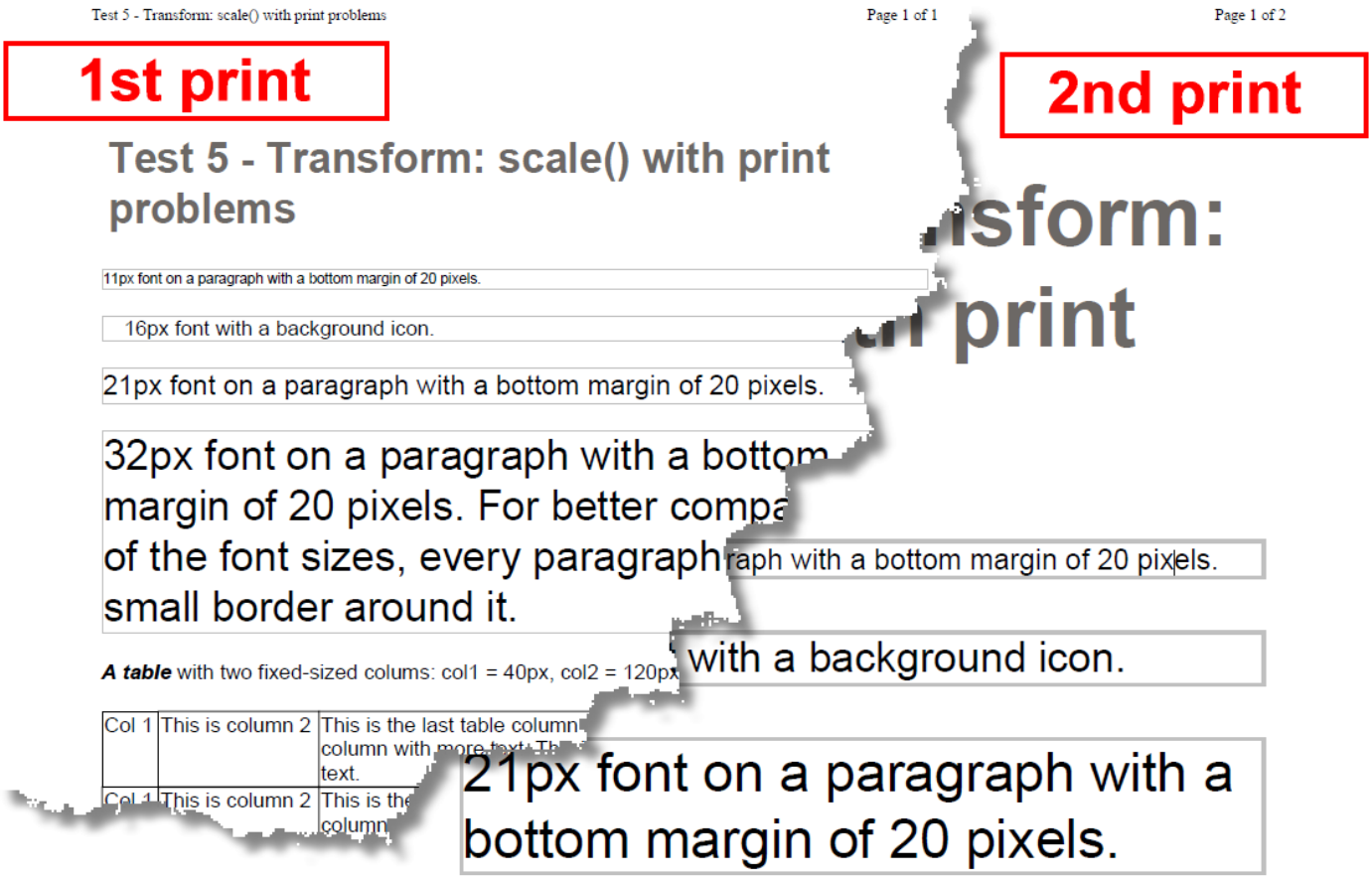
In my first attempt with `transform:scale()`, I kept the media queries for high resolutions to a bare minimum. But after getting really weired results with printing, I made them more strict. In the final version of **Test-5.html**, it's now the following code:

```
@media screen and (min-resolution: 192dpi) {   /* 200% */
  html {
     -ms-transform: scale(2);
     -ms-transform-origin: 0 0;
     width: 50%;
  }
}
```

This rule is clearly aimed on (a) screen display, not print and (b) just screens that use a minimum resolution of 192 DPI, which is 200% magnification. What will happen, if you print this topic, on a "normal" system with 96 DPI (100%)? It will print just normally, as expected? Let's give it a try...

1. Open the demo CHM file **HTML-Help-Test-For-4K.chm** and navigate to "**Test 5**"
2. Right-click the HTML content, from the context menu choose "Print". Then click OK to really print it.
3. Repeat step 2

The first printout will look normal. The second printout will be twice as large as the first. Any subsequent repetition of step 2 will produce the same result:



When printing the same topic twice, MSIE mixes up media queries

On the second printout, Internet Explorer clearly applies a media rule that should not be used at all. This bug can even be duplicated with MSIE in stand-alone mode, it doesn't apply to CHM files only. After a few more experiments, I decided to get things straight and put the high-resolution CSS into an external stylesheet. That's where it belongs, anyway.

The modifications applied to **Test-6.html** in the demo CHM file were these:

1. Put all 4K stuff into an external style sheet called *".\css\style4k.css"*
2. Reference the style sheet with *media="screen"*:

```
<link rel="stylesheet" href="./css/style4k.css" type="text/css" media="screen" />
```

And this one works as expected. *Done!*

Done? Wait a second...
This was about printing HTML Help *on 4K monitors*, wasn't it?

So, how does the printout from a high-resolution system look like? In a word: disappointing! The text is barely readable. It looks exactly like the **Test-2 topic**, in fact it is the same as Test-2, because the media queries that do the `transform:scale()` are no longer applied.

There's one interesting detail though: picture sizes and with absolute dimensions *specfied in pixels* do *appear at their correct size*! How come? Wasn't our embedded MSIE treating pixels like hardware pixels, so *everything* was just too small on a high-resolution display?

HTML Help print rendering is different again

We not have insight to the source of MSIE and its rendering engine; someone from Microsoft could probably tell us more about it. But we can test the results of different units of measurement. Let's compare the two reference topics (**Test 1** and **Test 2**) in my demo help file on both, a 96 DPI system and a 192 DPI high resolution system. We know that on 96 DPI, everything looks correct, HTML Help renders both, the *point*-font-sizes and the *pixel*-font-sizes as expected. Printing topics works as well.

On 192 DPI, things start to differ. On screen, the point-font-sizes grow with the system DPI, the pixel values do not, so the layout breaks. When printed, however, the print engine of MSIE treats both, the *point* and *pixel* values similarly, relative to the system DPI! It prints the page (almost) like the stand-alone Internet Explorer would print it. On screen it doesn't.

Let me put this into a table to wrap my own head around it:

HTML Help rendering on screen and print, 96 DPI and 192 DPI compared				
	On 96 DPI (100%)		On 192 DPI (200%, high-resolution)	
	Screen	Print	Screen	Print
Font sizes in point ("pt"), other dimensions in pixels ("px")	OK	OK	Text sizes correct, px-dimensions too small, layout breaks	OK (!)
Font sizes and other dimensions in pixels ("px")	OK	OK	Coherent layout, but content is much too small	Text too small, px-dimensions correct, layout breaks
Font sizes and other dimensions in pixels ("px"), `transform:scale()` zooms content	OK (no zoom)	OK (no zoom)	OK	Text size OK, px-dimensions too large, layout breaks, half of the page is cut off in print
Font sizes in point ("pt"), other pixel-based dimensions in device independent measuring units such as em ("em")	OK	OK	OK	OK

Conclusion

We need two separate style sheets for screen and print! The *screen stylesheet* defines font sizes in pixels and applies a zoom to the HTML body with `transform:scale()`, so that the content appears at the right size on a 4K monitor. The *print stylesheet* defines font sizes in points (instead of pixels), no zoom is applied to print.

The alternative is to change all pixel sizes to a *device-independent* unit. This solution has been proposed by *responsive-web* evangelists for years and is indeed the more sophisticated one. I called it the "em" solution.

Quick Summary

The print function In HTML Help was never really meant for printing documentation and is notorious for its bugs. Do not expect a pixel-perfect printout. Single topics, however, can usually be printed without problems.

On screen, the mix of font sizes in point with other elements in pixels will not work. In print, it will. To get it right you need 2 separate stylesheets for screen and print.

8 The "em" solution

"*An em is a unit in the field of typography, equal to the currently specified point size.*" (Wikipedia)

In HTML, an **em** is equal to the font height of the element it is used on. If you do not specify any font size in your CSS, the browser will assume a *default font size of 12pt*. This is 1em and that's 16 (hardware) pixels on a 96 DPI system (12/72*96 = 16). On a high-resolution system with 192 DPI, 1em is 32 (hardware) pixels.

We know that HTML Help - in spite of using a default browser zoom of 100% - scales font sizes defined in point proportionally to the system-wide DPI value. Fonts in point automatically grow with the display resolution. Pixel values do not, the *em* unit does.

In other words: **em** is the magic unit that grows with the font size in point. If we change all "px" values in the HTML code to "em", the em-dimensions will scale automatically, along with the text.

HTML code with pixel definitions:

```
<img src="./img/screen1.png">

<img src="./img/screen1.png" width="635" height="397" />

<img src="./img/screen1.png" style="width:635px; height:397px" />

<div style="width:400px; height:auto; margin:20px auto">...</div>
```

Changed code with em:

```
<img src="./img/screen1.png">

<img src="./img/screen1.png" width="635" height="397" />

<img src="./img/screen1.png" style="width:39.6875em; height:24.8125em" />

<div style="width:25em; height:auto; margin:1.25em auto">...</div>
```

Default value 16 px = 1 em

In **Test 7** of my demo help file, I replaced all px-based values with *em*. When recalculting the pixel sizes, I simply divided all pixels by 16. This works, because the font size of the <body> tag is 12pt, which is 16 pixels on a 96 DPI system. If the <body> tag had a different font size, 1em would be the equivalent of that size.

If you apply an *em* value to the CSS of a <h1> heading, the value in pixels is different. Because <h1> typically has a larger font size than the 12pt body font and *em* is always proportional to the font size of the

element it is used on:

```
<p style="font-size:12pt; margin-bottom:1em">  <!-- margin = 16px -->
<p style="font-size:16pt; margin-bottom:1em">  <!-- margin = 21px -->
<p style="font-size:20px; margin-bottom:1em">  <!-- margin = 20px (fixed) -->
```

So, one *em* is 16 pixels here and 21 pixels there, it all depends on what is going on in your stylesheets. Imagine a picture placed in a with a different font size. It was taxing enough to calculate width:39.6875em; height:24.8125em, but with a different font size it's a completely new calculation! If you rely on a pixel-perfect layout, working with *em* is like herding cats. Thanks to CSS3, however, there is a new family member, a close relative to *em*: the rem unit.

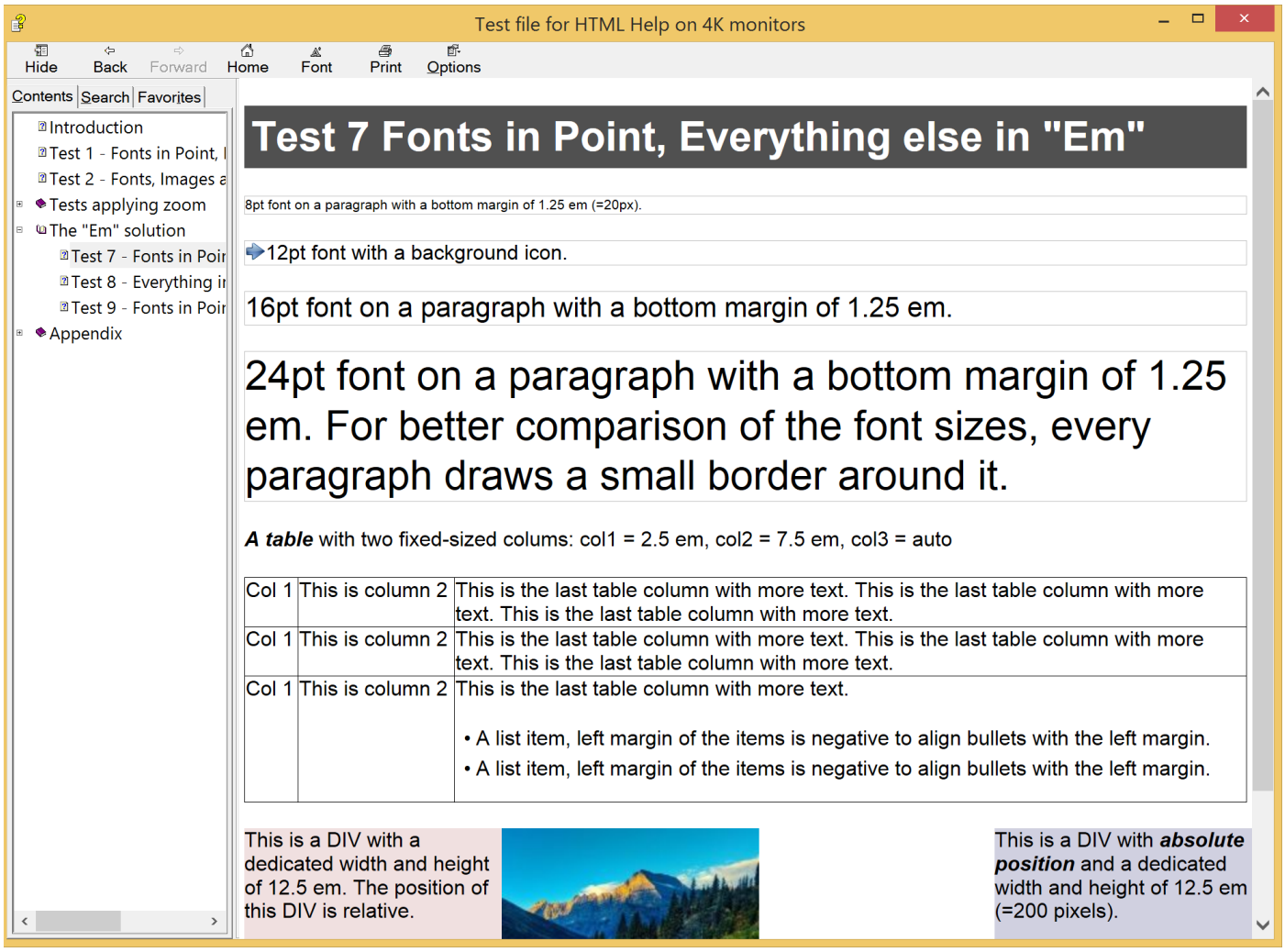
Default value 16 px = 1 em = 1 rem

In **Test 7** of my demo help file, I did not really use *em*. I've used *rem*, the *"root"-em*. 1 rem is equal to the font size of the root element, which is <body>. This makes the calculation a lot easier, because it does not change. If the CSS defines a 12pt font for the body, 1rem is always 16px, regardless of where it is used.

The *rem* unit is supported in Internet Explorer from version 9 on. In a compiled HTML Help file, MSIE runs in IE7 compatibility mode by default. We need to elevate it to IE9 to be able to use *rem*:

```
<!DOCTYPE HTML>
<html>
<head>
   <title>Test 3 - All Pixels, using Zoom and Media Queries</title>
   <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1" />
   <meta http-equiv="X-UA-Compatible" content="IE=9" />
```

Once this is in place, we can use use rem directly in the CSS. HTML Help will scale the entire content proportionally:



HTML Help on Windows 8.1 with 192 DPI, fonts in point

Redefining 'em

When coding HTML by hand, the *em/rem* solution is quite cumbersome, because you have to make a division by 16 (or a different divisor, depending on the font size of the body element), which isn't simple mental arithmetic. It would help if we could redefine, how many pixels an *em/rem* is. And that's easily possible!

In **Test 8** of my demo help file, I defined **1 em as 10 pixels** by setting the font size for the very root element, **<html>**. This is the CSS:

```
html { font-size: 62.5% }
body {
  background: #FFFFFF;
  font-family: Helvetica, Arial, sans-serif;
  font-size: 1.6em;
}
```

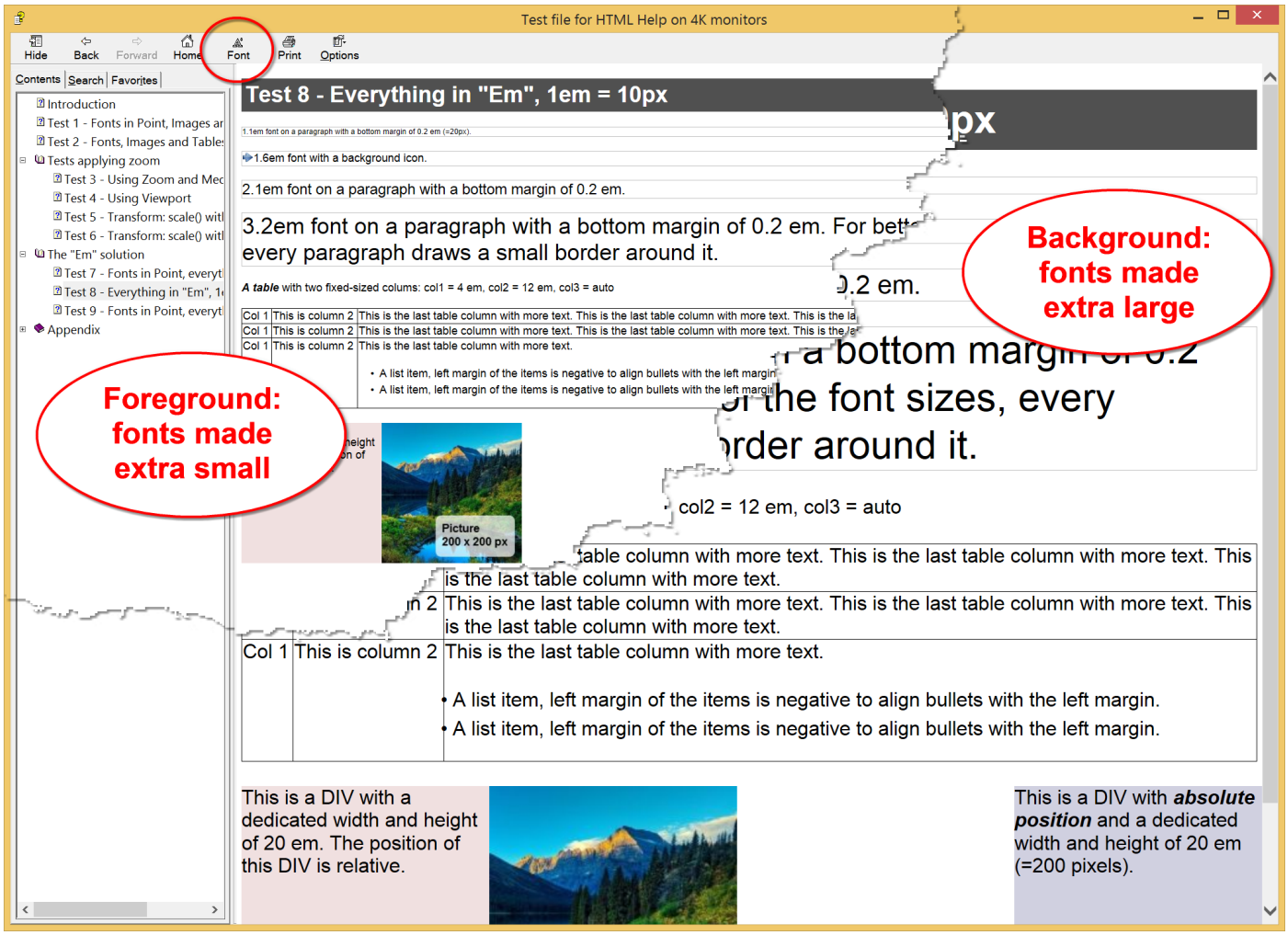
```
em       { font-weight: bold }
h1       { font-size: 3.2rem;
           background: #505050;
           color: #FFF;
           line-height: 120%;
           padding: 0.5rem }
p        { padding: 0;
           margin: 0 0 2rem 0;
           border: thin solid #c0c0c0;
           line-height: 120% }
p.plain { border: none }
.test1 { font-size: 1.1rem }
.test2 { font-size: 1.6rem;
          padding-left: 1.6rem;
          background:url(action32.png) left top no-repeat;
          background-size: 1.6rem 1.6rem; }
.test3 { font-size: 2.1rem }
.test4 { font-size: 3.2rem }
ul li { margin: 0 0 0.5rem -2.4rem }
```

The default font size of the root element is - if nothing else is specified in the stylesheet - 12pt, which is 100% and equal to 16px. To make it 10px by default, we need to change the root font size to 62.5%:

100% / 16 * 10 = 62.5%

The 16 pixels (or 10 pixels, respectively) are device-independent pixels. The browser assumes a default font size of 12pt. By changing this to 62.5% of 12pt, we define a base font size of 7.5pt which is equal to 10 hardware pixels on a 96 DPI system. This is much easier to calculate with - we just need to divide any "px" value in the HTML code by 10.

The result in HTML Help on a high-resolution system is the same, but has one additional benefit: because font sizes are now relative, the end user can adjust them for better readability[1].



HTML Help on Windows 8.1 with 144 DPI and relative font sizes, small and large compared

Other device-independent units

An *em/rem* is just one of several "device-independent" units you can use in HTML Help. What I mean with *device-independent* is that HTML Help will scale them. W3 defines them as *absolute lengths*, but HTML Help will scale them like font sizes in point:

cm	centimeters
mm	millimeters
in	inches
~~px~~	~~pixels~~ - **NOT THIS ONE!** This is the only unit that HTML Help really treats as a hardware pixel!
pt	points; 1pt is equal to 1/72nd of 1in

pc picas; 1pc is equal to 12pt

Limitations of device-independent units including "em/rem"

There are a few (really very few) HTML tags, that cannot be set up with CSS and scalable units, but require dedicated pixel values. Client-side image maps are one example. <object> tags or frames are another. This HTML code does not work with em/rem:

```
<img src="planets.gif" style="width:14.5rem; height:12.6rem" alt="Planets"
     usemap="#planetmap" />
  <map name="planetmap">
    <area shape="rect" coords="0,0,82,126" href="sun.htm" alt="Sun" />
    <area shape="circle" coords="90,58,3" href="mercur.htm" alt="Mercury" />
    <area shape="circle" coords="124,58,8" href="venus.htm" alt="Venus" />
  </map>

<object width="400" height="400" data="helloworld.swf"></object>
```

The <object> tag as well as <iframes> can be worked around with inline CSS. For the <area> tag there is no such workaround, because the `coords` attribute definitely wants pixel values.

Quick Summary

Compared to the zoom methods explored in the previous chapter, the "em" solution is superior. There are no optical glitches with scrollbars and margins, no need to tinker with separate print stylesheets. There are compatibility issues with a few HTML tags like image maps, objects and frames, which can be overcome though.

The "em" solution is stable and with *em* (though not with *root-em*), the HTML is compatible even with really old versions of Internet Explorer. The disadvantage of this method: it might involve a *lot* of HTML code changes.

[1] This article describes how to add a Font Size button to the toolbar of a HTML Help file:
https://support2.microsoft.com/default.aspx?scid=KB;EN-US;Q240062&

Summary

A summary of the zoom methods discussed in the previous chapters, along with their advantages and disadvantages and list of useful links to explore.

Picture courtesy of pixabay.com

9 HTML coding ahead

For many years there wasn't much progress in the Windows desktop world regarding high-resolution display, but recent hardware developments have brought some major changes. We are quickly moving towards a *retina world*. It's high time to adjust HTML Help and make it compatible with the new hardware requirements.

In this tutorial, I have outlined two general solutions. Both require changes to your HTML code or your CSS.

Applying "zoom" with transform: scale() is a relatively quick fix, because it can be applied with a global stylesheet. There are additional challenges associated with printing help pages from the HTML Help Viewer, but it can be done. This solution requires font sizes defined in pixel. It is possible that the CSS3 transformation might not work on old systems that don't have a recent version of Internet Explorer installed (the minimum is IE9).

The second way to do it is the "em" solution. This involves individual changes to most help pages and requires intensive testing. It's very easy – too easy – to overlook an individual or <img> tag that is still defined with pixel values. If you decide to go this route then this is probably the right time to get yourself a decent HTML editor, if you don't already have one...

And while we're on the subject of HTML editors, there is one more important question:

How does our own help authoring tool *Help & Manual* handle these problems?

In Help & Manual 7 you can choose how to encode sizes. This decision is global for your entire help project and may be configured differently for every output format (Help & Manual can output the same project to a wide range of different formats). A single change in the configuration of the help project is all it takes make the output compatible. Our default for HTML Help/CHM output is **point** encoding for font sizes and **inches** for all pixel-based dimensions.

> *Why inches? Inches are a very old unit that has been supported in Internet Explorer since version 3. The alternative encoding would be "ems". But ems are relative to the parent element's font size and the page layout templates in Help & Manual are flexible and user-defined. If the user creates a layout with nested elements with different font sizes, the em units used in the page content would be relative to their parent container elements and not to the body tag. We'd have to use rems instead, to make sure that all HTML skins work. And rems in turn require elevation of the embedded MSIE to IE9, which produces unpleasant optical glitches in the CHM viewer.*

Help & Manual produces differently encoded HTML for every output format. When publishing WebHelp, it generates clean HTML5. For eBooks in the ePUB and Mobi formats, it takes the limitations of eBook viewers into account. When publishing CHM files, it automatically generates HTML code that works around the bugs of HTML Help and the embedded Internet Explorer that the MS CHM Viewer uses. We consider dealing with these problems to be our job, we *don't bother you* with these details. You can focus on writing

your documentation. Help & Manual will make it just work!

Download a fully functional 30-day evaluation version of Help & Manual from:
http://www.helpandmanual.com

10 Links

Note

If you received this eBook in PDF format *only* and it did not include the HTML Help example CHM with source code, please download the **complete package from this link**:

http://download.ec-software.com/getting-html-help-ready-for-4k-monitors.zip

The following links were helpful when writing this guide (in no particular order):

http://caniuse.com/
Overview of browser support of HTML and CSS

https://msdn.microsoft.com/en-us/library/ms531209(v=vs.85).aspx
Internet Explorer support of Cascading Style Sheets

http://mshelpwiki.helpmvp.com/mshelp/html-help
MS Help Wiki

https://support2.microsoft.com/default.aspx?scid=KB;EN-US;Q240062&
How to add a Font Size button to the toolbar of a HTML Help file

https://geekycoder.wordpress.com/2010/01/12/solutions-to-enlarge-the-text-size-of-chm-without-tears/
Solutions to enlarge the Text Size of CHM without Tears

http://blogs.windows.com/bloggingwindows/2013/07/15/windows-8-1-dpi-scaling-enhancements/
Windows blog - Windows 8.1 DPI Scaling Enhancements

http://www.anandtech.com/show/7939/scaling-windows-the-dpi-arms-race/5
AnandTech - Scaling Windows - The DPI Arms Race

http://blogs.msdn.com/b/fontblog/archive/2005/11/08/where-does-96-dpi-come-from-in-windows.aspx
MSDN blogs - Where does 96 DPI come from in Windows?

- A -

About the author 3, 48
Alternative measuring units 41

- B -

Browser
Embedded MSIE rendering 11, 19
Stand-alone rendering 11, 19
Zoom 9, 11

- C -

Contact 3
CSS
Device-independent units 41
Em 41
Media queries 29
Transform: scale() 34
Viewport 31
Zoom 28
Zoom, when printed 37

- D -

Device-independent units 41
Display resolutions 9, 19
Dots per inch 19
DPI 19

- E -

Em
Limitations 46
Rem 42
Size redefined 43
The magic unit 41
Em vs. Rem 42
Embedded MSIE 11
Elevating the browser engine 15, 29
IE7 compatibility mode 15

- F -

Firefox 9
Font sizes
In Em 41
In Pixels 28
In Pixels, when printed 37
Point vs. Pixels 26

- G -

Google Chrome 9

- H -

HTML Help
Different scaling 11, 19
In a Retina World 3
In Help & Manual 48
Manual zoom 5
No Zoom 11, 19
On a 4K monitor 5
Print problems 37
Text sizes 5

- I -

Internet Explorer 9, 11

- L -

Links 50

- M -

Magic unit "em" 41
Media queries 29
Microsoft Help Wiki 50
MSIE 11
stand-alone 9

- P -

Pixels 19
Points 19

Print problems in HTML Help 37

- R -

Rem 42

- S -

Summary 48

- T -

Testing without a 4K monitor 26
Text sizes in HTML Help 5
Transform: scale() 34

- U -

UI scaling in Windows 9, 19

- V -

Viewport 31

- W -

What end users can do 5
Windows
    System-wide DPI 19
    UI scaling 19

- Z -

Zoom 9
    CSS property (proprietary) 28
    In HTML Help 19
    In stand-alone browsers 9, 19