A LIST APART

Illustration by **Kevin Cornell**

A Pixel Identity Crisis

by **Scott Kellum** · January 17, 2012

Published in Graphic Design, Mobile/Multidevice, Interaction Design

A note from the editors: This article was augmented post production with vendor prefix information to cover Webkit-based and Opera browsers.

The pixel has always been the smallest unit in screen-based design. Because it's been indivisible, it is the concrete unit of measurement among screen-based designers. The phrase "a pixel is a pixel is a pixel" has been adopted to help print designers not used to fixed-screen density understand the concept. Because of this consistency, web designers have adopted pixels over points and other units to build websites.

Now that hardware is changing and pixel densities are growing, pixels are struggling to find relevance as the stable unit they once were. Browser zooming is one thing and has been covered on QuirksMode *(http://www.quirksmode.org/blog/archives/2010/04/a_pixel_is_not.html)*. But what is a pixel on high resolution devices today? Why does the 640px — 960px iPhone 4 claim to be 320px — 480px in the browser? The truth is that there are two different definitions of pixels: they can be the smallest unit a screen can support (a hardware pixel) or a pixel can be based on an optically consistent unit called a "reference pixel."

The hardware pixel

Most of us are familiar with the hardware pixel. It's the smallest point a screen can physically display and is usually comprised of red, green, and blue sub-pixels. Light from these three sub-pixels is mixed to create the colors we see. Because the hardware pixel relates to a physical element on a screen it cannot be stretched, skewed, or subdivided. These properties make the hardware pixel like the atom: the unit of design on which we build everything.

The reference pixel and splitting atoms

Things are changing for the pixel. The w3c currently defines the reference pixel as the standard for all pixel-based measurements *(http://www.w3.org/TR/CSS2/syndata.html#length-units)*. Now, instead of every pixel-based measurement being based on a hardware pixel it is based on an *optical reference unit* that might be twice the size of a hardware pixel. This new pixel should look exactly the same in all viewing situations. The beauty of using a reference pixel is that it takes proximity to a screen into account. When using a phone that you held close, a reference pixel will be smaller on the screen than a projection you view from a distance. If the viewer holds their phone up so it is side-by-side with the projection, the pixel sizes should look identical no matter the resolution or pixel density the devices have. When implemented properly, this new standard will provide unprecedented stability across all designs on all platforms no matter the pixel density or viewing distance.

Reference pixels are amazing, but now we have two conflicting definitions. Android devices have a new unit *(http://developer.android.com/guide/practices/screens_support.html)*, called the "density independent pixel" or "dip," which developers can use to distinguish the number of optical pixels an item spans. This allows developers to use hardware pixels for crisp graphics and patterns by using px or the new relative definition of a pixel for text sizing and consistent proportion across devices by using `dip`. Splitting the definition of a pixel into two units is great for Android but the web has to deal with years of pixel based designs across all kinds of devices. It would be wonderful if web developers had these units as well, but without shifting the current definition of a pixel entirely, the pixel-based web would break.

Imagine if the iPhone 4 told a website it was actually 640px; text would render at half size, such that without zooming, that website would be nearly impossible to read. For all those mobile sites that disable zooming it would be impossible to use.

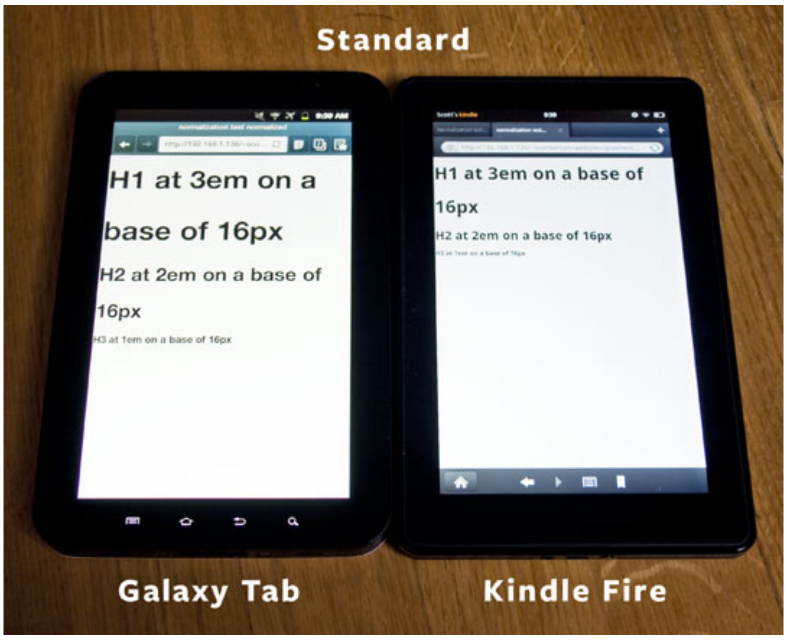
Standard
H1 at 3em on a base of 16px
H2 at 2em on a base of 16px
H3 at 1em on a base of 16px
Galaxy Tab
Kindle Fire

Fig. 1: Standard pixel sizes on the Galaxy Tab and Kindle Fire.

We also don't know what each device considers a pixel and this is a problem. For example, the original Galaxy Tab and the Kindle Fire have the exact same screen size and resolution; but pixels on the Tab, which adjust for the reference pixel, are measured at 1.5 times that of the Fire, which uses hardware pixels. It is impossible to develop for new devices without having one to test on because the only specs released speak in terms of hardware pixels. A developer might look up information on these two devices and think that if the screens are the same and they both use the Webkit rendering engine their website will look the same. Sadly this is not the case; as far as a website is concerned, the Galaxy Tab is 400px — 683px and the Kindle Fire is 600px — 1024px.

Identifying break points and moving forward

It's safe to say there is a problem with pixels. But we can use media queries to identify the inconsistencies and adjust accordingly. We can use the `device-pixel-ratio` media query to identify devices with scaled pixels. We can combine that with dimensions to identify types of devices with some amount of accuracy. The iPhone 4 has a `device-pixel-ratio` of two, so it measures pixels as twice the size of a hardware pixel. Many Android devices have a `device-pixel-ratio` of 1.5, so things are scaled at one-and-a-half times larger than the hardware pixel. Currently this needs to be prefixed, but you can use this query to start identifying scaled devices.

Specificity is key when identifying classes of devices, but be careful: never be quite so specific as to single out a single device and attempt to target that in case other devices have the same problem. Taking a look at the original Galaxy Tab and devices like it, a `device-pixel-ratio` of 1.5 will only get us part-way there. Most Android phones today also have this `device-pixel-ratio` and conflicts there can lead to other problems. Querying the `device-width` at both orientations will tell us that this is larger than a phone, but not as large as other tablets that may come out in the future. We can use `device-width` to identify the screen, unlike `width` and `height` which only query the viewport. Using the device dimensions, we can gain much better insight into the hardware being used:

```
@media screen and (device-pixel-ratio: 1.5)
    and (device-width: 683px)
    and (orientation: landscape),
  screen and (device-pixel-ratio: 1.5)
    and (device-width: 400px)
    and (orientation: portrait)
```

Note that `device-pixel-ratio` needs to be vendor prefixed to work, so adding `-webkit-` and `-o-` will make it work in Webkit and Opera browsers.

Now let’s look at devices like the Kindle Fire, which has the exact same hardware resolution as the Tab, and uses hardware-based pixels. Dealing with devices like this are more difficult because they’re more likely to conflict with netbooks and other larger screens as well as the iPad with a similar resolution. Using the media queries `orientation` and `max-device-height` will highlight the devices we need. Devices usually give a height less than the actual `device-height` so to avoid 1024 x 768 tablets identifying those under 600 pixels tall will avoid those. Now we’re looking at tablet devices which have fairly consistent resolutions and sizes:

```
@media screen and (device-pixel-ratio: 1)
    and (device-width: 1024px)
    and (max-device-height: 600px)
    and (orientation: landscape),
  screen and (device-pixel-ratio: 1)
    and (device-width: 600px)
    and (max-device-height: 1024px)
    and (orientation: portrait)
```

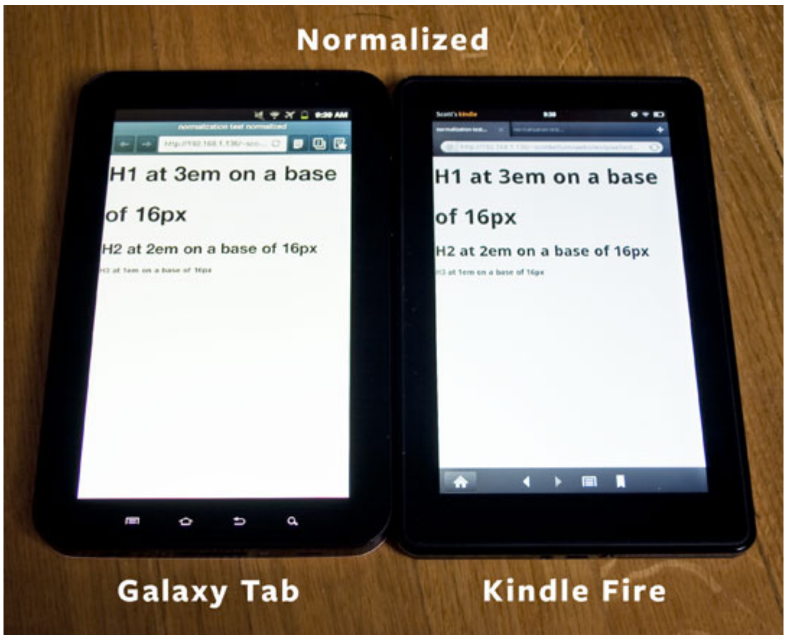
Normalized
H1 at 3em on a base of 16px
H2 at 2em on a base of 16px
H1 at 3em on a base of 16px
H2 at 2em on a base of 16px
Galaxy Tab
Kindle Fire

Fig. 2: Normalized pixels on the Galaxy Tab and Kindle Fire.

Using ems instead of pixels will give you a lot of control over how your site can scale—and not just on type—on everything that had a pixel-based value before. Now, all you need to do is change the `font-size` value on the `html` element. Switching the Kindle Fire to look like the Galaxy Tab is just a matter of multiplying the base size by 3/2 (1.5) on the Kindle Fire's media query. If you wish to use hardware-based pixels on the Galaxy Tab, multiply its base size by 2/3 (.666667). Because everything is based on ems, your site will scale relative to base `font-size`. Check out the full example in this Gist *(https://gist.github.com/1438467)*. Changing it depending on the context can improve consistency across platforms.

Managing multiple pixel definitions can be tricky. The pixel is the most basic unit and this shift to the optical pixel will cause some strange things to happen. Device makers are doing a fairly good job at identifying these issues and solving them so we don't have to, but something is always going to slip through the cracks or contain inconsistencies. Fortunately we have tools to identify and overcome these obstacles as long as we remain aware of this shift.

About the Author

Scott Kellum

Scott Kellum is the design director at Treesaver, a company in New York City that creates paginated publications using web standards. Keeping pace with a broad range of interests, Scott is a Sass enthusiast and amateur type designer; formally a draftsman at Darden Studio.

ISSN 1534-0295 · Copyright © 1998–2018 A List Apart & Our Authors