Multiscreen Help Authoring

How to deal with the explosion in device sizes

The potential screen width range that we may need to support, now extends from two inches to eighty.

At the Adobe Thought Leadership Day in Chicago, I talked about the challenges and opportunities of supporting applications that will be displayed on devices of a wide variety of sizes.

For the last thirty years or so the designers of personal computing software have had a fairly consistent canvas to work. Viewing screens for most computers have been in a range of approximately 10-14 diagonal inches. Screen resolutions in the past twenty years have been fairly static as well. It was possible to deliver user experience in one presentation flavor. The User Interface (UI) would scale automatically from desktop to laptop and even to netbook. User Assistance (UA) designers enjoyed not having to worry very much about how information would look and feel on different displays.

All that changed with the iPhone in 2007 and was further affected with the introduction of Android in 2008. Since then, there has been an explosion of device types and sizes. Most of that has been with devices smaller than a typical desktop/laptop PC. Now we see emergence of app development for large screen home entertainment systems as shown in Figure 1. The potential screen width range that we may need to support, now extends from two inches to eighty.

One of the things we must to do is start thinking about what this means for the design and delivery of User Assistance. Whether we are doing Help for native OS apps or presenting content in a mobile browser, we need to address the wide and growing fragmentation of devices.

What we don't want to do is take the approach of having the content designed for the desktop squeezed into a screen the size of a candy bar. We've all seen this on our personal mobile devices. Words and images presented in a tiny page display that requires a lot of pinching and zooming to read what has been squished beyond recognition. Hopefully we can entertain a more graceful and efficient adjustment. Ideally, we should strive to match the amount and the type of content with a particular device.

Partnered with CSS3 and JavaScript, and with the support of browser makers, HTML5 offers a way to single-source content in an intelligent and scalable way.

There are other issues to deal with beyond varying screen resolutions. Certain UI elements and OS components are proprietary to individual platforms and we may need to make adjustments for those. While tap and double tap are used commonly across touch devices, other interactions may differ from one platform to another. The recent lawsuit won by Apple against Samsung has already resulted in Android changing certain user interface elements to be different from the iPhone.

Another difference between the desktop and mobile devices is how they are used. It is right in the word 'mobile'. A smartphone is used to get quick glimpses at only the most important information needed at that time. Common sense tells us that trying to read and touch a device while in motion is more difficult than being at your desk.

All of the above issues point out the need to adjust the presentation of User Assistance to best fit the profile of individual devices. However, crafting individual solutions for this explosion of devices isn't practical. Even organizations the size of Microsoft and Apple don't have enough resources to support hundreds of different devices.

One possible solution lies in the emergence of HTML5. Partnered with CSS3 and JavaScript, and with the support of browser makers, HTML5 offers a way to single-source content in an intelligent and scalable way.

Using HTML elements, we custom tag common objects in our user interface, we craft style sheets for major device 'types', and we use another web technology—media query to match the device with the right style sheet. The result can be a single set of source files that can be automatically transformed to look and work differently on different devices.

For example, assume we are supporting a web-based application. In addition to the desktop, our customers would like to use this content on a phone, a 7" tablet, and a 10" tablet.

The central deliverable is the content repository—the files that make up our web app. This component will be the same, regardless of how many device categories you want to support. This is also the part you are likely to be very familiar with. This scenario is shown in Figure 2.

Figure 2

In Adobe RoboHelp, you have controls for conditional text and user-defined variables.

Two new components are sets of cascading style sheets and associated media queries designed to manipulate single-source content based on the device type it is being displayed on. The design of these style sheets and media queries is a very important new skill. (For more info on media queries, see w3.org/TR/css3-mediaqueries/@media1).

You will need to design your style sheets so that they optimally adjust the content based on the screen dimensions of the device and its form of user interaction. This approach is often called 'responsive design'.

Large screens can show more information and have a more robust set of navigation tools. With smaller displays, you may want to hide or scale certain features. For example, the navigation pane may need to be minimized or hidden completely for a phone. The content may also need to be adjusted if the primary interaction is by touch rather than a mouse.

Figure 3 shows what the underlying single-source framework would look like. The Help topics or other content are contained in one repository. The overall layout and display of terms is processed by four different style sheets. Depending on your authoring tool, you can use various methods to tag content for use in one or more of the 'buckets'.

In addition to adjusting the look and feel of the content, the CSS can control the language through conditional text. Selectors can be set to hide and reveal variants such as 'tap' and 'click' based on the computing device and the interaction device. The 'More details' section of this article shows the four output targets separately so you can see them a bit better than in Figure 2.

In Adobe RoboHelp, you have controls for conditional text and user-defined variables to help manage this effort. I'll get into the details of working with RoboHelp in an upcoming article.

Using just these four 'buckets' supports a significant percentage of the installed user base of devices. It is likely that they can continue to do so for several years (Figure 4). However, it will not be difficult to support emerging, significant screen sizes with a few new buckets. We may have ones for home entertainment screens, automotive displays, and information kiosks.

When using an array of style sheets as described here, it can be very helpful to establish a parent style sheet (Figure 5). This defines an overall look and feel for the family of deliverables. The typeface, color palette, and background can be defined and managed in one master file. The child style sheets inherit the global attributes and fine-tune them for their individual media types. RoboHelp supports parent/child CSS.

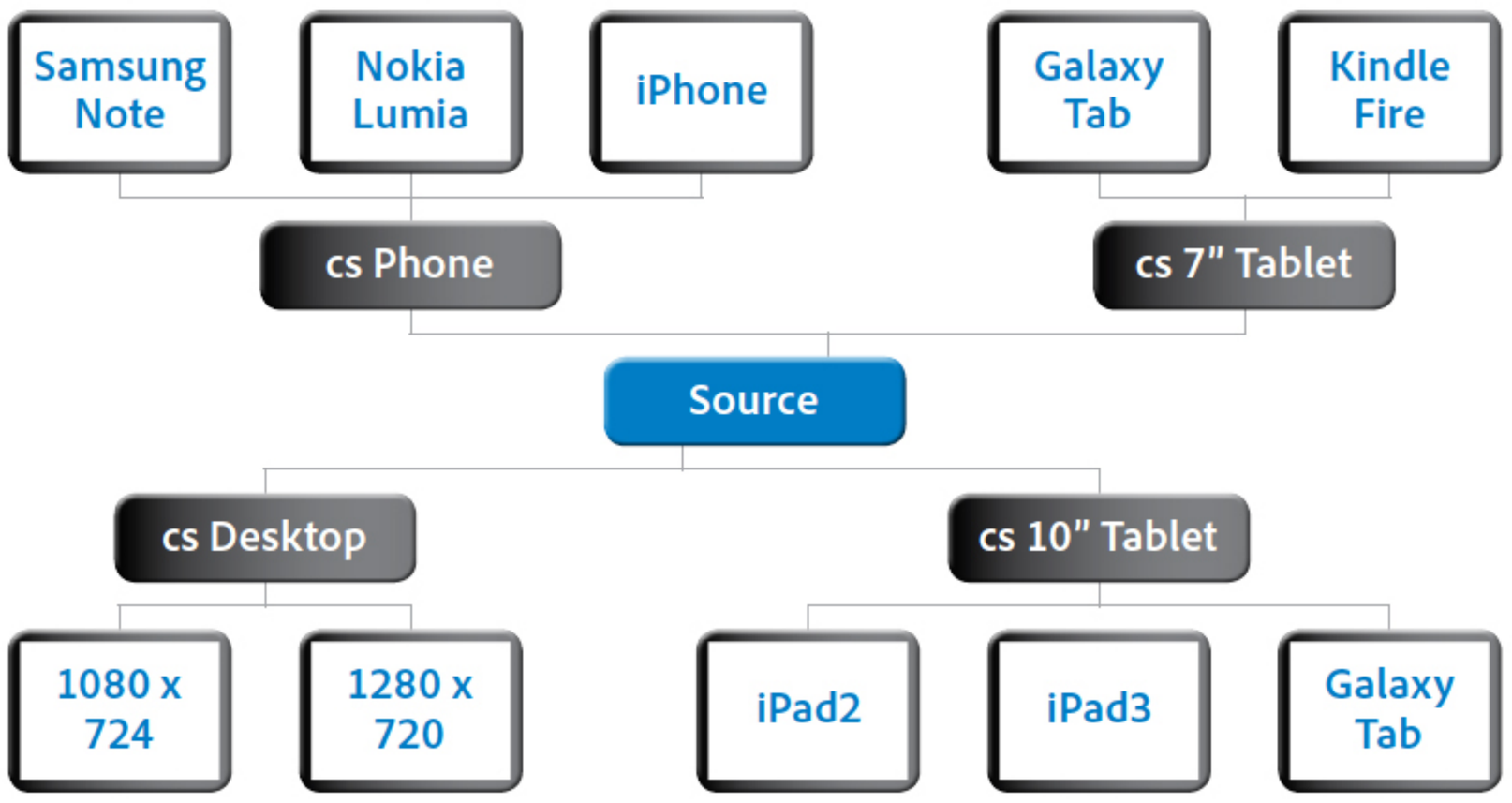


Figure 3

When using an array of style sheets, it can be very helpful to establish a parent style sheet... RoboHelp supports parent/ child CSS.

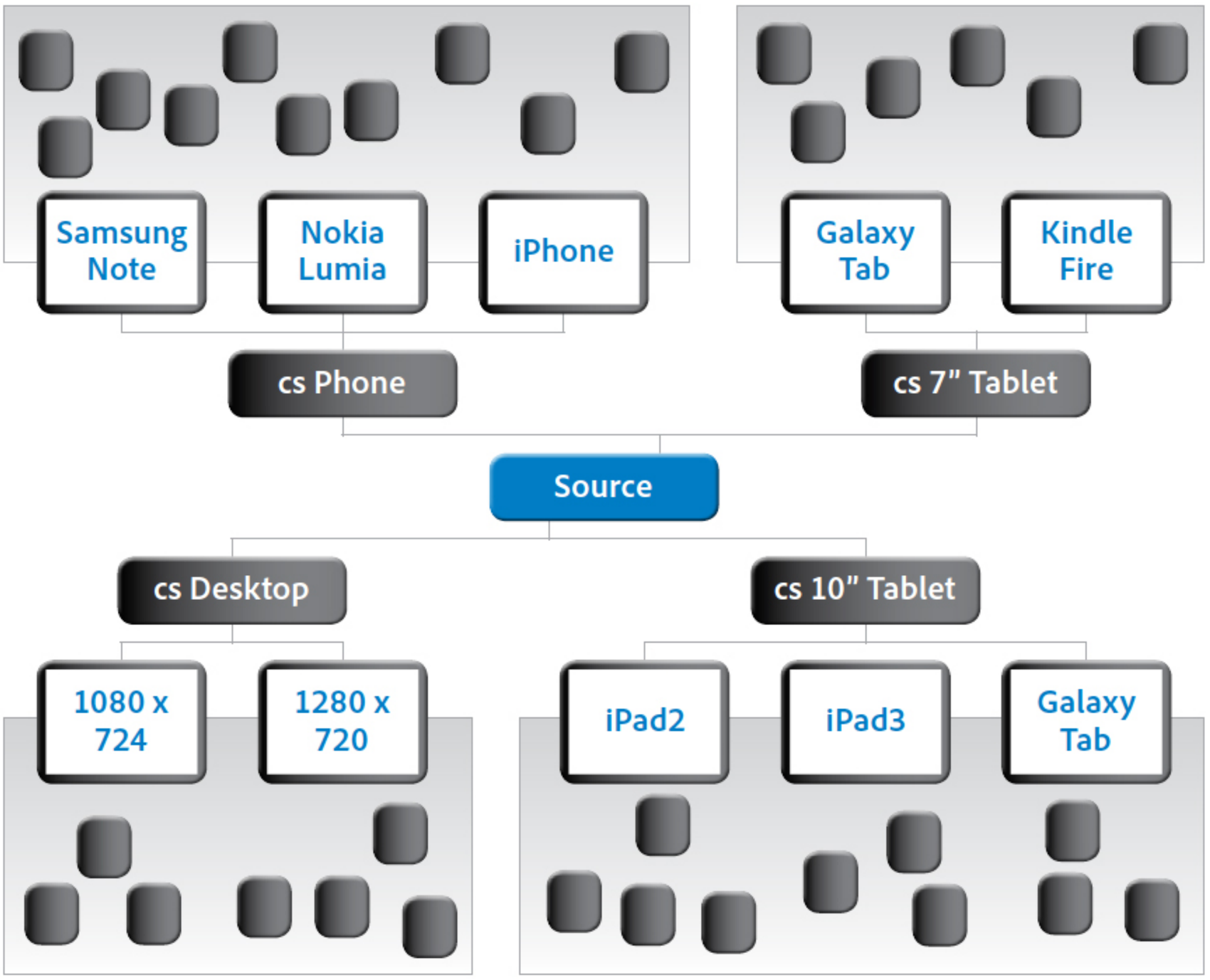


Figure 4

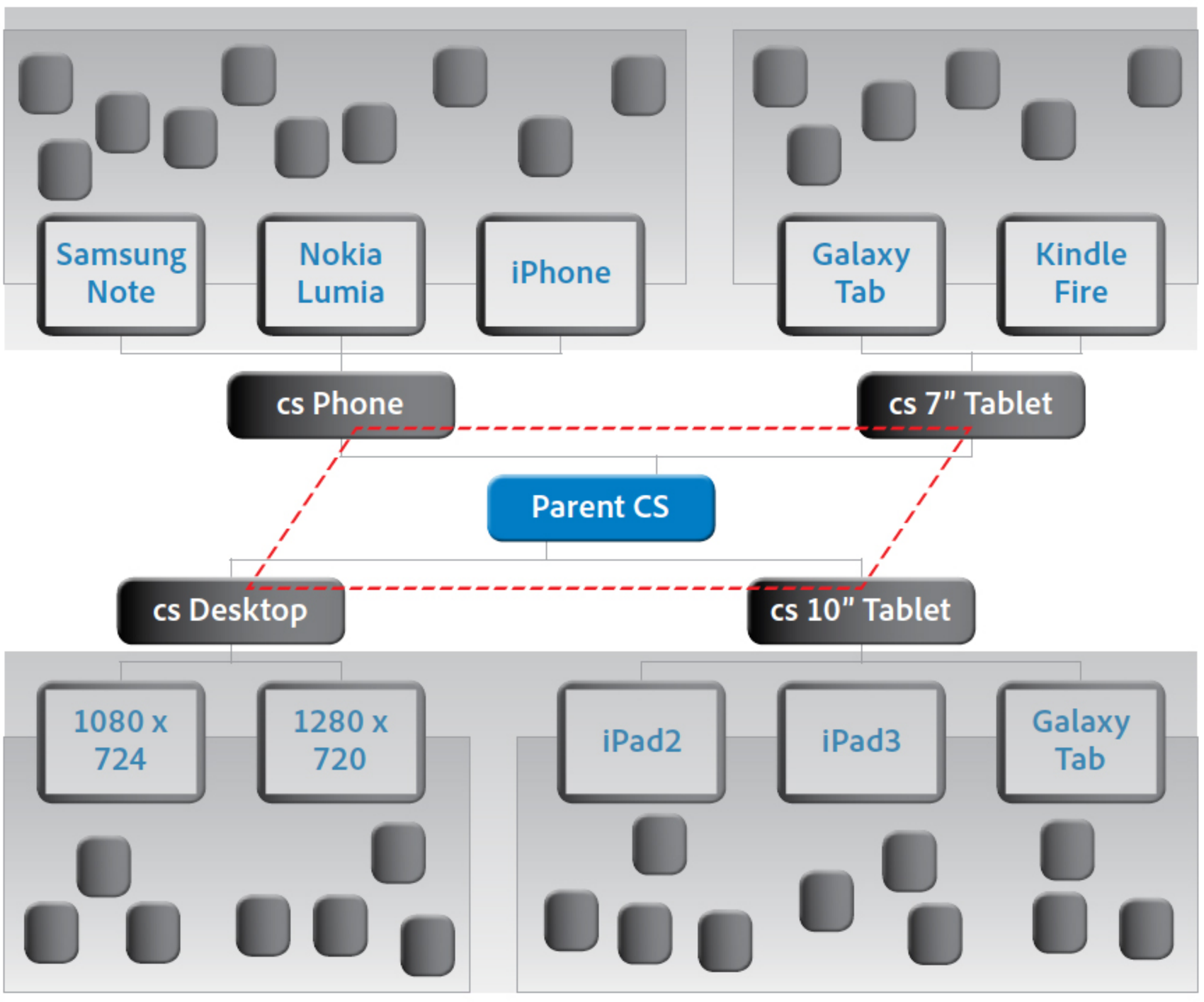


Figure 5

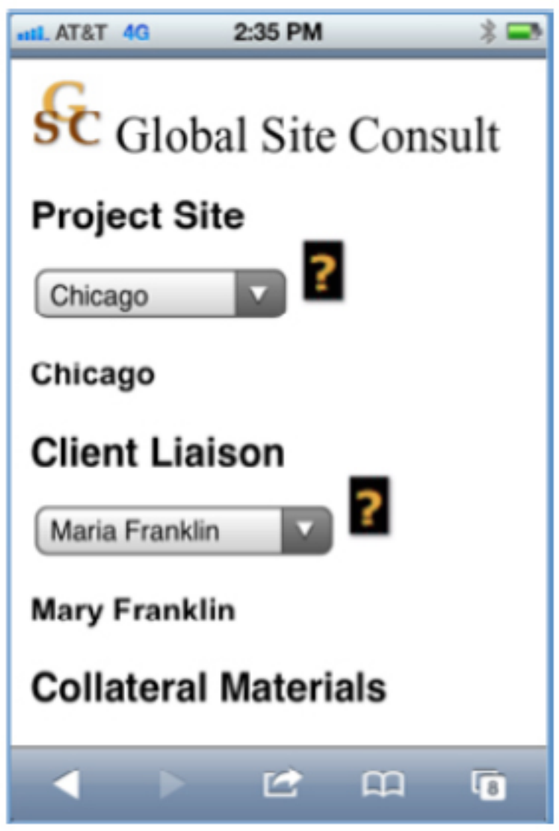
You need to design style sheets so that they optimally adjust the content based on the screen dimensions of the device...

More details

Desktop

10" (iPad)

7" (Nexus 7)



iPhone

A User Assistance professional needs to find ways to support an ever-increasing set of devices and interaction types.

Conclusion

Mobile devices will soon be the dominant method for mainstream consumers to work with software applications. During this transition, a User Assistance professional needs to find ways to support an ever-increasing set of devices and interaction types. HTML5, Cascading Style Sheets, Media Queries, and tools like RoboHelp can together provide a method for doing so.

As you move forward, consider the following:

- Adopt a system that gracefully adapts its information presentation.
- Simplify and focus your content development with a single-source core.
- Adjust to varying device types by separating presentation from content with HTML, CSS, and Media Query.
- As a final note, look into the 'mobile-first' approach being espoused by Luke Wroblewski and others.

Biography

Joe Welinske is the President of WritersUA.

Joe Welinske is the president and founder of WritersUA—a company devoted to providing training and information to User Assistance professionals. The WritersUA Conference draws hundreds of attendees each year from around the world to share the latest in User Assistance design and implementation. The free content on the WritersUA website attracts over 20,000 visitors each month. Joe most recently published 'Developing User Assistance for Mobile Apps'. He is currently teaching courses at Bellevue College and UC Silicon Valley. Joe received a B.S. in Industrial Engineering from the University of Illinois in 1981, and an M.S. in Adult Instructional Management from Loyola University in 1987.

Adobe

Adobe Systems Incorporated
345 Park Avenue
San Jose, CA 95110-2704
USA
www.adobe.com

XXXXXXXX 9/12