

Structure Benefits All

Aaron Leventhal
IBM, 14 Crosby Street
Arlington, MA
781-643-5083
<http://www.mozilla.org/access>
aleventh@us.ibm.com

ABSTRACT

Accessibility for the Dynamic Web is now possible due to new standards being developed at the W3C and being implemented in Firefox. The technology allows today's web pages to contain additional markup describing semantics. An often-cited benefit of this technology is the ability to describe scripted widgets with dynamic behaviour. However, another major benefit is to differentiate the sections of a web page, via human-readable labels or predefined semantics such as "main", "contentinfo", "navigation" and "search". Marking the sections of a web page offers significant improvement for users who need access to today's web with a small device or an assistive technology.

Categories and Subject Descriptors

H.1.2 [Models and Principles]: User/ Machine Systems—*human factors, human information processing*; K.4.2 [Computers and Society]: Social Issues—*assistive technologies for persons with disabilities*

General Terms

Human Factors, Design, User Agents

Keywords

Accessibility, Semantics, Dynamic HTML, User Agents

1. INTRODUCTION

This history of software development is filled with unsuccessful attempts to converge onto a single popular technology that can provide write once, run anywhere, while most likely unattainable. However, the web today is getting closer to that reality – or, at least to write once, test and debug everywhere. There are now reasonable choices for developing applications that work on both the desktop and on mobile devices, albeit each with a different cost/usability tradeoff.

This paper will discuss techniques for authoring websites that work for a variety of users on a variety of devices.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

W4A at WWW2006 23-26 May 2006, Edinburgh, UK
Copyright 2006 ACM 1-59593-281-x/06/05 ...\$5.00.

The focus is on making mainstream websites work better on small devices, using simple techniques which provide accessibility advantages for free.

2. DECIDING WHICH MARKUP TO USE

First, lets establish that we are talking about content which will be served in either HTML, XHTML or a variant such as XHTML-MP (mobile profile). In reality the mobile web is converging on HTML, which is nearly a superset of the other two languages. Why? All mobile browsers need to support HTML for competitive reasons – its become a must have feature that end users can browse the real web. Because of this, mobile developers are not necessarily seeing a driving need to use a more restrictive markup dialect when targeting mobile platforms.

Depending on the situation a developer faces, it is reasonable to use HTML, XHTML Basic or XHTML-MP, but in fact this choice matters less and less as the mobile web evolves. Those developers writing from scratch, concerned with standards or who believe the future belongs to true device independence via XForms, X+V or SVG, should currently choose XHTML Basic. Those developers who simply wish to be in sync with the web today, may simply follow basic HTML authoring techniques for small devices and accessibility.

3. DECIDING HOW MUCH TO TAILOR THE CONTENT FOR SMALL DEVICES

A bigger choice point is how much effort will be put into tailoring the content for small devices.

3.1 The Easiest and Least Effective Choice: Do nothing

In whats unfortunately the most likely scenario, the author will not concern themselves with the fact that small devices may use of their content. What happens to this content on a small device? It depends on the content, the browser and the device.

Opera's Small Screen Rendering

Two pieces of software consistently do a good job formatting mainstream web pages for small devices Opera and Opera Mini[1]. These two browsers use Operas SSR (Small Screen Rendering), and generally do a great job of unrolling the content into a long column that makes maximum use of the screen size[2]. The SSR rendering removes the need for users to scroll back and forth horizontally in order to read the

page. Navigation through the content can be accomplished just with vertical scrolling. Unfortunately Operas SSR may not always provide a satisfactory version of the content, if the developer:

- Structured the document out of order from its natural reading order
- used images which are too large to work on a small screen (and require too much download bandwidth)
- used dynamic effects like pop out submenus which may not fit on the screen
- relied on mouse pointers there may be no pointing device or none which produces events like mouseover
- used frames

In addition, not all mobile browsers have as advanced technology as SSR for dealing with real web pages. Even if SSR handles a given web page perfectly, it is very likely that the layout will not be satisfactory on other mobile browsers.

This kind of technology will greatly help format web sites for many devices, without requiring the author to consider every possible configuration. However, it cannot be even close to optimal without some cooperation from the author.

And, although SSRs layout is an improvement, it does not reorder the content to show the most important items first. It is still very difficult to find a web page that does not require paging down through many screens before reaching the main content. It can also be difficult to just find the search box for a web site.

The problem is that most websites position the link navigation sections to the top and left of a site. Since these are usually first in the document, SSR displays them first even though they are arguably the least important content. Opera recommends putting most the navigation links to the right side of the page. Unfortunately most web sites go against this recommendation, and most authors are not willing to change their design because of this recommendation. As we discuss at the end of this article, it is currently possible to mark navigation, search and main content sections, and ultimately this will allow mobile users to find what they need more quickly.

Example of Small Screen Rendering

Here is the ideal before (desktop) and after (on a small device) of Small Screen Rendering:



Figure 1: The Eclipse web site on the desktop, before transformation.

The Eclipse example shows the reasonable limit for SSR for the usual case. The normal layout from figure 1 changes dramatically in figure 2, which shows the content rendered vertically: a header, then the search box, then a dozen links

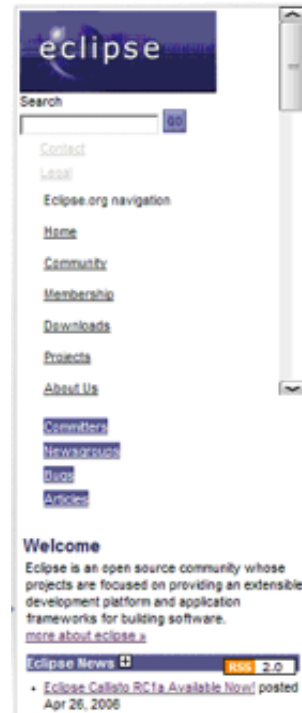


Figure 2: The Eclipse web site rendered in SSR.

followed by the main content. The main content is not on the screen when you first enter the site using Opera. However, the user has only a small distance to scroll down before the main content begins.

Many things can and usually will go wrong, and even in this ideal case usability is impaired by all of the links cluttering the beginning of the page, rather than coming after the main content.

Although far from perfect, innovations such as Small Screen Rendering do prove that a lot is possible, even when dealing web content as it exists today. If only desktop screen magnification software such as ZoomText, MAGIC and Lunar knew this trick, it would help low vision users quite a bit! As it stands low vision users often have a difficult time finding what they're looking for on web sites, as they pan and scroll both horizontally and vertically finding the link you want can be like an exercise in mowing the lawn. It will be interesting to see how browsers on small devices evolve their reformatting techniques.

3.2 A More Effective Choice: Follow Basic Tips

The following tips have been collected from Operas developer pages[3], with italicized comments provided by the author. Again Opera has shown leadership in the field by putting together a complete, simple to understand resource for developing content which works well on small devices. These techniques are not expensive to implement in a web-site being developed from scratch. Adding to their usefulness, over 90 of these tips are also good for accessibility.

Coding tips

1. Use terse, efficient markup – *also good for accessibility!*

2. Avoid frames – *also good for accessibility!*
3. Avoid pop-ups – *also good for accessibility!*
4. Avoid using proprietary features, or use fallbacks *also good for accessibility!*
5. Specify image height and width Images load slowly on small devices – this helps the web browser know ahead of time how much screen real estate they will need
6. Use alternative text on images – *also good for accessibility!* It is common to browse with images off because this greatly decreases the wait and cost for the end user.
7. Have fallbacks for JavaScript and dynamic effects – *also good for accessibility!*

Testing Tips

1. Test in Opera, in Small-Screen view (shift+F11) *also good for accessibility!*
2. Test with graphics turned off *also good for accessibility!*
3. Test with JavaScript turned off – *also good for accessibility!*
4. Test with no mouse – *also good for accessibility!*

Small-Screen design tips

1. Design with document order in mind – *also good for accessibility!*
2. Design the small-screen version for maximum readability – *also good for accessibility!*
 - (a) Use headings. These can be used to the users advantage, via navigation shortcuts which move the user between headings *also good for accessibility!*
 - (b) Avoid tables for layout use CSS instead *also good for accessibility!*
 - (c) Do not rely on color in text it will all be shown black on light gray *also good for accessibility and printing*
3. Only use images suited for a small screen, hide the rest in mobile media style sheet Large images do not work well on a small screen, so some images are removed and others are scaled. Images slicing will not work.
4. Be careful with the use of colors, font sizes, and alignment – *also good for accessibility!*

Modern devices usually have a color screen, but they offer less contrast than a normal PC screen. Devices are also often used in sunlight and other difficult conditions while PC screens are used in more controlled environments. Reducing screen contrast can be used to conserve battery length. For these reasons you should use good contrast between foreground and background. Expect subtle color differences to disappear.

Utilizing Operas tips in your mainstream web site is the least expensive solution which still works fairly well.

On the minus side, it still really provides the user with too much information for a small screen. The user will often need to scroll vertically to find the important content. Thus, this technique will not bring millions of mobile customers to a portal.

3.3 A Bit More Optimal: Create a Mobile Style Sheet

SSR is clever enough that it will not take action when a mobile style sheet is present. In that case SSR assumes that the mobile style sheet has been designed for efficient mobile browsing.

Creating a mobile style sheet has a few advantages:

1. The author can code to remove content which is redundant, purely decorative or of lesser importance
2. The author has complete control over layout
3. This solution does decrease the download size significantly in that images with the CSS rule display: none will not be loaded. However, the HTML document will still be loaded.

Here is some sample code to insert a mobile style sheet:

```
<!-- Stylesheet for devices -->
<link rel="stylesheet"
      href="phone.css"
      media="handheld">
```

Unfortunately, it is still difficult to avoid having the navigation links show up at the top of the content without removing them altogether, if the desktop version has them to the top or left. Web authors could put link navigation panels at the end of their document and still have them rendered at the top or left, but are unlikely to do so because CSS positioning is not rendered consistently in desktop browsers.

Worst of all, not all mobile browsers will pick up on the mobile style sheet.

3.4 Optimal but Expensive: Create an Alternate Version

Finally, if the mobile market for a particular website is crucial, the alternate version of the content is most likely needed. This will provide the ideal solution for mobile users, in terms of small download size, most readable layout and best usability, an alternative version must be created. This alternative version will be tested and optimized for a variety of mobile devices. Providing a unique URL for mobile users is not recommended, in that mobile users will not as easily remember the special URL.

It is better to detect the presence of a small screen browser by checking the user agent string or the UA Profile. It is difficult, but it can be done. The server can then be configured to present the alternative version of the content.

Providing an alternate version is an option for content providers with a large commercial stake in the mobile space. And, as the accessibility world has discovered, alternative versions of content often lag behind their mainstream counterparts functionality.

3.5 A Newly Evolving Compromise: Add Semantics

The solutions discussed up to this point all have concrete advantages and disadvantages. The first three options are affordable for a wide variety of organizations and developers. The fourth option, creating an alternative version of the content, is only practical where significant resources are available to develop, test and maintain that version.

On the other hand, the affordable options all commonly share one important problem they tend to result in lost and frustrated users. Its a bit too tedious to find what you need with a small window within the elongated column of content. If it takes too long to find what they want, the user is most likely going to give up before completing the task.

For example, try to find the main content on the following BBC front page (for this exercise pretending that BBC does not have a mobile version) (see Figure 3).

Start in the Main Content

Clearly, reaching the BBC sites main content requires too much effort. The biggest usability hurdle with this and many sites is allowing the small device user to see or navigate to what they want quickly.

This is also a known problem for users with disabilities. Screen magnification, screen reader and keyboard-only users all have difficulty getting to the main content of a website efficiently after a page loads.

The main content is generally the first place you want to visit, so it would be ideal if screen magnifiers, screen readers, keyboard navigation extensions and small device browsers could all start the user in the main content.

The Importance of Landmarks

All of these user types also need streamlined navigation to other key features of a web site, such as the main search box or a navigation links panel. It would be excellent if the user agents for small devices and users with disabilities could provide shortcuts for navigating to these parts of a page.

Setting Landmarks

Fortunately, there is a potential way to enable these features because web sites can mark up the main content, navigation panel or search field as such. The necessary semantics are called document landmarks. The most important document landmarks are defined in the XHTML 2 [5] Role Access module:

main This defines the main content of a document.

secondary This is any unique section of the document. In the case of a portal, this may include but not be limited to: show times; current weather; or stocks to watch.

navigation This is the navigation bar on a web document. This is typically a list of links to other pages on the site or other areas of the same document.

banner A banner is usually defined as the advertisement at the top of a web page. The banner content typically contains the site or company logo and other key advertisements for the site.



Figure 3: The BBC website unrolled via Opera's Small Screen Rendering. The main content shows up somewhere around the 3rd screen full.

contentinfo This is information about the content on the page. For example, footnotes, copyrights, links to privacy statements, etc. would belong here.

note The content is parenthetic or ancillary to the main content of the resource.

seealso Indicates that the element contains content that is related to the main content of the page.

search This is the search section of a web document. This is typically a form used to submit search requests about the site or is a more general Internet wide search service.

Additional landmark roles are being defined by W3Cs WAI Protocols and Formats Working Group (PFWG).

The landmarks can be set in a number of ways. In XHTML 2 (and soon in XHTML 1.x), it is possible to simply define these roles on the element using the role attribute, for example:

```
<div role=maincontent>Todays main story involves  
</div>
```

In HTML, the same landmark can also be defined in the <head>, like this:

```
<link rel=maincontent  
      href=#maincontent>
```

In addition, it would be straightforward to develop a new Microformat which would allow navigation roles to be used in HTML via the class attribute, as in:

```
<div class=axis x2:maincontent>  
  Todays main story  
</div>
```

A similar class attribute technique will very likely be used this way by DHTML authors marking widget roles such as menu, tabpanel or slider [6].

Sprinkling this additional markup into existing XHTML 1.x or even HTML content allows the content to have additional semantic information, yet does not affect the rendering of the content. This approach fills a crucial niche it allows the same content to be used everywhere, so the more expensive option 4 of developing new content will not always be necessary. In addition, it can greatly help accessibility.

Screen magnifiers, screen readers and keyboard navigation software can direct users to the main content or important sections of a web page automatically, or via a keystroke. In addition, use of a richer set of landmark roles will allow the content to be presented in alternative ways for users with learning and cognitive disabilities.

Next Steps — Industry Support

The standards involved in landmarks are still under development, but some user agent support already exists. Firefox, for example, provides information about landmarks to screen readers and screen magnifiers. The next step is to work with the manufacturers of these assistive technologies to include support in their products.

Another crucial step for the industry is to make this important technique useful and prominent within the realm of phones and PDAs.

The author believes this success is only a matter of time. The industry can build on its successes with dynamic widget role support in screen readers such as JAWS and Window-Eyes. Web pages developed using roles can now provide desktop-like user interfaces to visually impaired Firefox users, as well as the mainstream [7].

As the user agent support becomes more of a reality, developers can begin truly working on techniques to develop content meant for both desktops and small devices.

4. CONCLUSION

Accessibility for the Dynamic Web is now possible due to new standards being developed at the W3C and being implemented in Firefox. The technology allows today's web pages to contain additional markup describing semantics. An often-cited benefit of this technology is the ability to describe scripted widgets with dynamic behavior. However, another major benefit is to differentiate the sections of a web page, via human-readable labels or predefined semantics such as "main", "contentinfo", "navigation" and "search". Marking the sections of a web page, in principle a simple concept, can potentially offer significant gains for web usability with handheld devices and assistive technologies.

5. REFERENCES

- [1] R. Beattie. Opera Mini: Best Mobile Web Browser Bar. None publicly available at <http://www.russellbeattie.com/notebook/1008770.html>
- [2] M. Wilton-Jones. Mobile browser rendering, publicly available at <http://www.howtcreate.co.uk/operaStuff/devices/>
- [3] Opera Inc. Making Small Devices Look Great, publicly available at <http://my.opera.com/community/dev/device/>
- [4] Developers Home, Tutorial about Detecting User Agent Types and Client Device Capabilities, publicly available at <http://www.developershome.com/wap/detection/>
- [5] Axelsson, J, Birbeck, M., Dubinko, M., Epperson, B., Ishikawa, M. McCarron, S., Navaro, A., Pemberton, S. XHTMLTM 2.0 W3C Working Draft (May 2005) publicly available at <http://www.w3.org/TR/2005/WD-xhtml2-20050527/>
- [6] M. Pilgrim, B. Gibson, A. Leventhal. Embedding Accessibility Role and State Metadata in HTML Documents, publicly available at <http://www.w3.org/WAI/PF/adaptable/HTML4/embedding-20060318.html>
- [7] A. Leventhal. Accessible DHTML, publicly available at http://developer.mozilla.org/en/docs/Accessible_DHTML