

# Mobile Friendly Library Websites

**Going mobile:** how to make your library website a more satisfying experience for visitors with cell phones, PDAs, and other mobile devices

A quick guide prepared by Fleur Helsingor  
(fhelsing@library.berkeley.edu)  
Web worker for the Engineering Library and Library Collections  
University of California, Berkeley  
<<http://www.lib.berkeley.edu/ENGI/>>

**June 2009, Updated July 2010**

## Contents:

- Introduction
- Design considerations: what works, what doesn't
- Step 1: Install mobile device emulators to test your site
- Step 2: Switch to XHTML
- Step 3: Create your "handheld media" external style sheet
- Step 4: Update your current "screen media" style sheet
- Step 5: Clean up the code in your web pages
- Step 6: Add the new style sheet to your pages
- Step 7: Add mobile friendly styles to the content of your pages
- Getting ready to launch!

## Introduction

A few years ago, if you wanted to make your website accessible to visitors with cell phones, handheld computers, and other mobile devices, you had to create a separate website for them using a special markup language such as the Wireless Markup Language (WML) instead of HTML. There's no need to do that now — modern mobile device browsers can handle conventional web pages coded in Extensible HTML (XHTML). You can use Cascading Style Sheets (CSS) to present your existing website's content in a mobile friendly way.

If you wish, you can still create a separate website for mobile device users. If you're creating a new website from scratch and your pages will contain a lot of tabular data or multimedia-rich content, you may want to create two separate websites. If your current website uses frames, and you don't plan to eliminate the use of frames, you should create a separate website for mobile device users.

Otherwise, I suggest that you use XHTML and CSS instead, and update your existing web pages so that the same content can display well on mobile devices as well as on desktop and laptop computers.

Since the Engineering Library's existing website has hundreds of pages and I figured that most, if not all, of the pages contain content of interest to mobile device users, I decided to use multiple external style sheets to serve up the site's existing content to a variety of users. Each page on the website calls three external style sheets:

```
<link rel="stylesheet" media="screen" type="text/css" href="Oengi.css" />
<link rel="stylesheet" media="print" type="text/css" href="Oengi_print.css" />
<link rel="stylesheet" media="handheld" type="text/css" href="Oengi_handheld.css" />
```

If the user is viewing the page on a desktop or laptop computer (which I'll refer to as "screen media" in this guide), the browser loads the first CSS document. When the user wants to print a copy of the page, the browser loads the printer friendly version ("print media"). The third CSS document is loaded when the user is viewing the page on a mobile device ("handheld media").

In addition, you can also provide style sheets for other kinds of displays, including projectors, televisions, Braille readers, TTY devices, and screen readers. I wouldn't be surprised to see even more media added in the future!

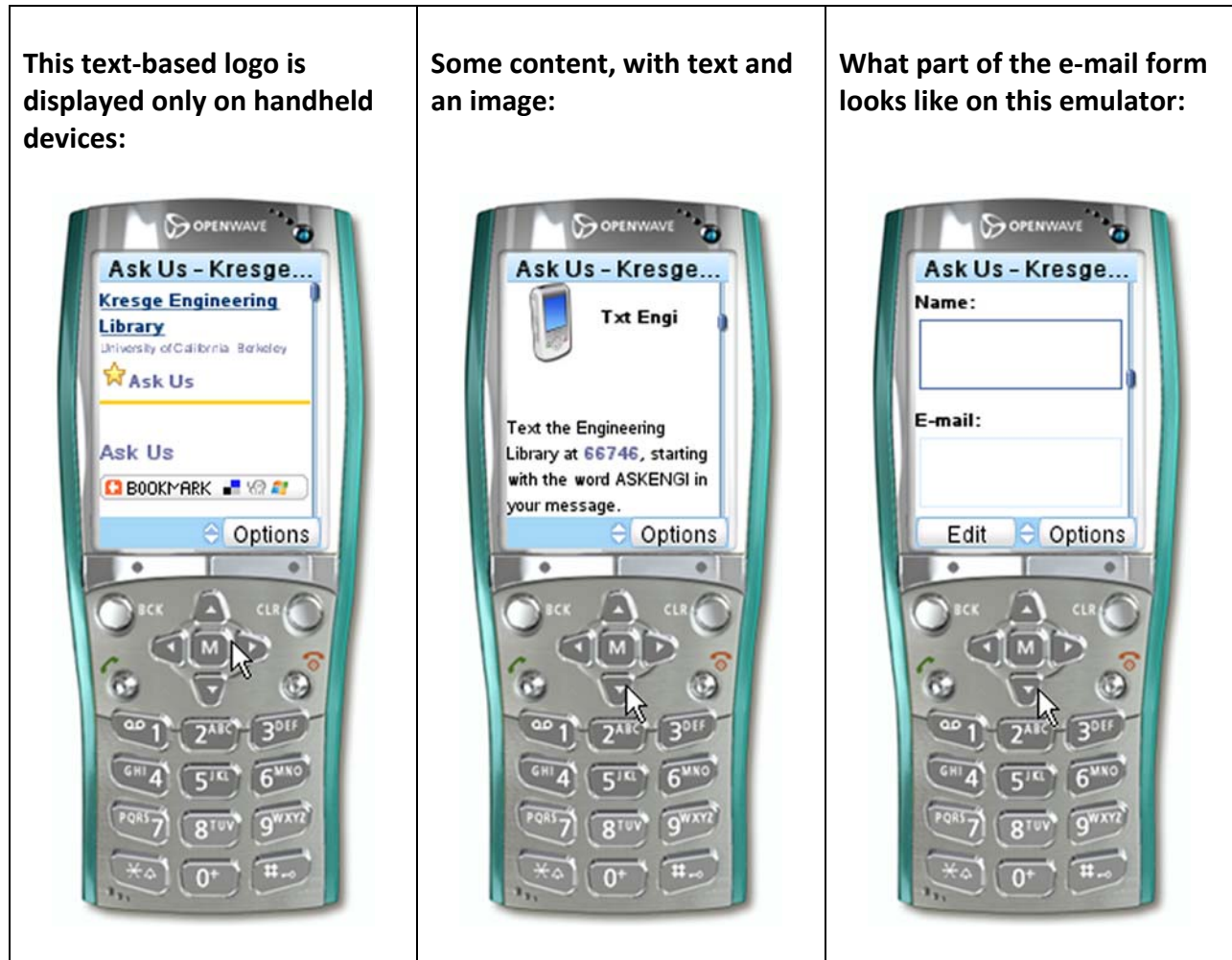
### For more information:

See "Style sheets for handhelds", chapter 13 in *HTML, XHTML, and CSS, sixth edition: Visual quickstart guide*, by Elizabeth Castro. Peachpit Press, ©2007. ISBN 0321430840  
<http://www.peachpit.com/store/product.aspx?isbn=0321430840>

"Changing the design of a page for wireless devices with the Handheld Media type: Design techniques to turn any page into a wireless friendly page," an article by Jennifer Kyrnin at About.com  
<http://webdesign.about.com/od/wireless/a/aa070307.htm>



This is what the Engineering Library's **Ask Us** page looks like when it's displayed on a cell phone emulator, using the library's "handheld media" style sheet (**0engi\_handheld.css**):

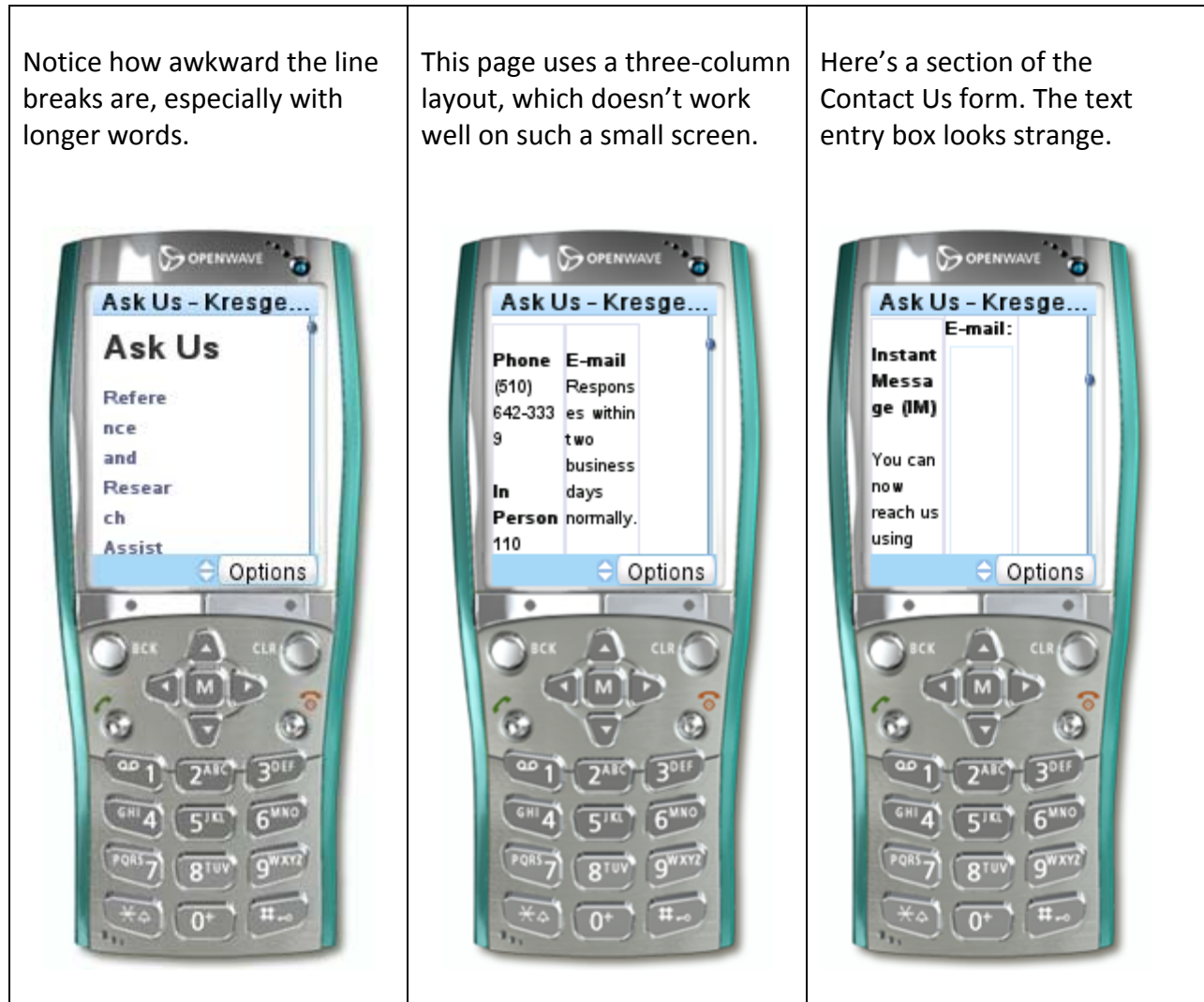


---

**Note:** both devices are loading the same web page. The style sheets are controlling how the content is displayed, and each device's web browser chooses which one to use.

---

By way of comparison, this is what the **Ask Us** page looked like *before* I put a style sheet in place for visitors with handheld devices.



Fortunately, there is a way to fix these kinds of problems, and the solution involves the use of XHTML and CSS. There are also several site design considerations that you should keep in mind as you work toward creating a more user friendly website for your visitors with mobile devices.

## Design considerations: what works, what doesn't

Most XHTML elements are supported within mobile device web browsers. The six basic headline elements, `<h1>` through `<h6>`, work well, as do the paragraph element (`<p>`) and the various list elements. Support for XHTML-based forms is also quite good. You'll be able to make use of the usual text boxes, radio buttons, check boxes, and other types of form fields.

I've also found that **Server-Side Includes (SSIs)** are well supported. If you have chunks of content that you use on multiple pages on your website, you can continue to use SSIs.

**Font support** can be spotty. In some mobile device browsers, only one font face is supported, with limited font size support. Handheld device browsers can support bold text, but support for italic text may be poor. My advice:

- Go ahead and style your text as you wish using CSS, but be aware that some of the handheld device browsers may render your text in a limited way.
- Use percentages or ems to specify font sizes rather than pixels.

In my experience, only three commonly used content types cause serious display problems on handheld devices.

**Large images:** since mobile device screens can be very small, mobile device browsers will attempt to resize large images so that they will fit on the screen. Even though the browser is displaying a small image, the actual file size remains large and the image may load very slowly.

Fortunately, it's easy to use CSS to hide large images to keep them from loading in the mobile environment. There's a very useful attribute and value pair ("display : none") for use within CSS documents that you can put into place to hide content from display on mobile devices (or other hardware, for that matter).

To do this, you can create a custom class to add to your external "handheld media" style sheet, and this custom class can be used whenever you need it:

```
.hideThis { display : none; }
```

Note that the class name begins with a dot (**.hideThis**, for example). The dot tells the browser that you're creating a custom CSS class and that this class can be used many times within the document.

The "**display : none**" attribute and value pair coding instructs a mobile device's browser to ignore the content in question and to not display it, while a desktop PC or laptop would still be able to display it.

For example, if you've created a class called ".hideThis" and you want to hide a large image:

```

```

**Note:** Small images, such as the RSS feed icon or UC-eLinks buttons, can work well in the mobile web environment so feel free to keep them in place. Some well-chosen small graphics can be beneficial to users at any screen size.

**Nested lists** can get pretty ragged-looking on a small display. Try to keep your lists to only one level of nesting. You may need to recast your lists — try replacing the outer level of nesting with paragraph XHTML elements instead of list elements.

**Tables** do not display well on the smallest mobile devices, so you'll need to avoid using table-based coding to lay out your pages. Use CSS layout techniques instead. If you're not familiar with CSS page layout techniques, I'll be providing a simple example later in this guide.

If you have a large table containing tabular data, you can hide it from view on mobile devices while making it available for "screen media" users:

```
<div class="hideThis"> <!--hide this table from display on mobile devices -->
<table>
...
</table>
</div> <!--end of hiding this table from mobile devices -->
```

**Since mobile device users may not be using a mouse or other pointing device**, they may benefit from having access to a special table of contents or other supplementary navigation. However, you may not feel the need to provide these features to "screen media" users. In this case, you can add a style to your "screen media" style sheet to hide such content from users with desktop or laptop computers.

```
.handheldContent { display : none; }
```

This style can be used whenever it is needed and as many times as needed. For example, to create a table of contents for mobile device users only:

```
<div class="handheldContent"> <!--hide this ToC from display on "screen media" -->
<p><strong>Contents:</strong></p>
<ul>
...
</ul>
</div> <!--end of hiding this ToC from "screen media" -->
```

**JavaScript** is poorly supported in mobile web browsers, so you should avoid its use or provide an alternative for users who can't use JavaScript or have it turned off. This practice would also help some disabled users as well as handheld device users.

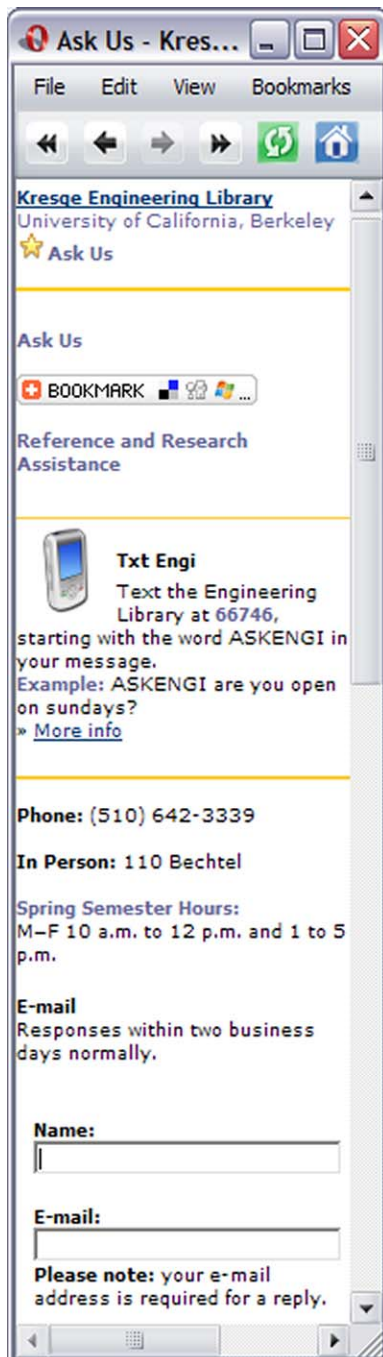
For example, here's a code chunk from the Engineering Library's page footer:

```
<div class="small">Copyright (C) 2009 The Regents of the University of California. All rights
reserved.
<br /><script type="text/javascript" src="engiEmail2.js"> </script>
<noscript>Document maintained by the Kresge Engineering Library.</noscript> ...</div>
```

In this example, an external script, **engiEmail2.js**, is called into action. If JavaScript is turned off or unsupported by the browser, the chunk of text contained between the **<noscript>** and **</noscript>** elements is displayed instead.

However, this doesn't mean that all scripts cause problems. For example, the CGI-based scripts that we're using on the Library Web to activate our forms usually work as expected on mobile devices. You can keep them in place.

## Step 1: Install mobile device emulators to test your site



Screen capture using the Opera browser in "small screen" mode

There is a wide variety of platform and browser combinations in the mobile web environment, more than a hundred such combinations. Screen sizes vary widely, too, from cell phones with tiny screens in the 100-150 pixel range to pocket PCs with screens that are 600 pixels wide. To make things a little easier for you during the design process, you can install **mobile device emulators** on your PC and view the files on your hard drive as you're working on them. There are several options available.

Recent versions of **Dreamweaver** (CS3 or later) offer a suite of mobile device emulators built into the package. Education discounts are available.

[http://www.adobe.com/devnet/devices/articles/introducing\\_device\\_central.html](http://www.adobe.com/devnet/devices/articles/introducing_device_central.html)

Download the **Opera browser**. You can view your pages in small screen mode (use the **View** menu > **Small Screen** option). There's also an Opera for Mobile version of the browser, and it's installed on millions of handheld devices. Opera is free of charge as of this writing.

<http://www.opera.com/>

### Tips:

- For the ultimate emulation experience, minimize your Opera browser window to a 150-pixel width!
- To locate emulator software for specific devices, do a Google search for "mobile emulator."

## Step 2: Switch to XHTML

If you're still using HTML 4 (or an older version of HTML), you should upgrade your code to XHTML. I recommend upgrading to **XHTML 1.0 Transitional** — this version of XHTML is more forgiving than the Strict version, and if you're used to HTML 4 the learning curve won't be so steep.

**Why do you need to do this?** Because of the wide variety of handheld devices and mobile browsers that are available, your code should comply with the current standards that are being promoted by the World Wide Web Consortium (W3C). Also, mobile device browsers support Extensible HTML (XHTML) rather than HTML. If you're using HTML 4 or an even older version of HTML, your code may display poorly, if it displays at all.

**Drupal users:** go ahead and skip to the next step. The Drupal development community has made it easy to create XHTML-based websites right out of the box!

**Dreamweaver** has a built-in code conversion feature. Open the document that you want to upgrade, and use:

**File** menu > **Convert** > **XHTML**

**If you code by hand:**

- There's a quick guide to learning what's different about XHTML in **Chapter 16** of *HTML & XHTML: The Definitive Guide*, 6th Edition. This O'Reilly book is available online at Safari:  
<http://proquest.safaribooksonline.com/0596527322>
- Take a look at "You already write for cell phones," a brief introduction to XHTML Basic by Jennifer Kyrnin.  
<http://webdesign.about.com/cs/xhtmllxml/a/aa073001a.htm>

## Step 3: Create your “handheld media” external style sheet

Cascading Style Sheets get their name from the way they behave. Style designations literally cascade downward, from external styles to embedded styles to inline styles.

First, the browser reads in the styles from the external CSS document, if one is being called. Then it reads in any styles that are embedded in the **<head>** section of the page. If an embedded style conflicts with a style in the external CSS document, the browser will discard the style from the external document and use the embedded style. If an inline style is declared within the page’s **<body>** section, the browser will use the inline style instead of the style that was declared in an embedded or external style sheet.

For example, if you specify a first-level (**<h1>**) headline using an inline style with a very large font size, the mobile device will try to display the huge headline even if an **<h1>** headline style was declared with a smaller font size in the external “handheld media” style sheet. To avoid such conflicts, define the giant-sized headline styles only within your “screen media” external style sheet.

This doesn’t mean that you need to always avoid embedded or inline styles. For example, if you have one paragraph on one page that you want to display using a blue color, it should work well on a mobile device so feel free use an inline style in such a case.

### Two basic guidelines:

- Create style definitions for every commonly used style in both external style sheets.
- Make at least two general purpose styles, such as **.hideThis** (for the **handheld.css** document) and **.handheldContent** (for the **screen.css** document), for use as needed. By the way, these names should be unique so that you won’t inadvertently hide something that you meant to display.

### If you would like to learn more about CSS:

A five-lesson self-paced online course is available at About.com. You’ll receive the lessons in e-mail, and the course is free of charge. There’s no requirement to buy a textbook or submit homework assignments.

<http://webdesign.about.com/c/ec/30.htm>

*CSS: The Definitive Guide*, 3rd Edition, by Eric A. Meyer. This book is available to UCB users at Safari:  
<http://proquest.safaribooksonline.com/0596527330>

**Dreamweaver users:** to learn more about creating and using style sheets within the Dreamweaver environment, take a look at the Missing Manual series. There should be an edition available for your particular version of Dreamweaver.

<http://search.oreilly.com/?q=dreamweaver+missing+manual>

**If you’re using Dreamweaver 8**, there’s an online course available at **E-Learn Berkeley**. Log on at <http://blu.berkeley.edu/> and navigate to E-Learn (it’s in the **Self Service** section). When you arrive at the site, search for the “Working with Cascading Style Sheets in Dreamweaver 8” course.

To help get you started, I've created a basic "handheld media" style sheet:

```
/* Style sheet for "handheld */
/* media" (handheld.css) */

body {
  margin : 0px 0px 0px 0px;
  font-family : verdana, helvetica,
    sans-serif;
}

/* library banner */
#header { display : none; }

/* special logo for use only */
/* on mobile devices */
#logo {
  font-size : 100%;
  color : #666699;
  background-color : #ffffff;
}

/* single column of content */
/* on handheld devices */
#container { width : 100%; }
#content { width : 100%; }

/* site navigation will not */
/* display on handheld devices */
#navigation { display : none; }

/* page footer */
#footer {
  width : 100%;
  background-color : #ffffff;
  color : #000000;
  font-size : 70%;
  padding-bottom : 2px;
}

/* document title */
h1 { font-size : 80%; }

p, ol, ul { font-size : 78%; }

/* hide any content that */
/* shouldn't be displayed on */
/* mobile devices */
.hideThis { display : none; }

/* end of handheld.css */
```

## Notes:

If your library makes use of a large banner image, you can have handheld devices display a text-based banner — the **#logo** style — instead of the large banner.

Handheld devices would display all of the content in a single column even though "screen media" devices would display it in two columns. The site navigation that would normally be displayed in the second column would not be loaded or displayed on handheld devices.

The page footer styles would look a bit different on handheld devices.

This is a custom class style that could be used to hide any content from view on handheld devices, including large images and data tables, as needed.

## Step 4: Update your current “screen media” style sheet:

Here’s a very simple style sheet, to serve as an example:

```
/* Style sheet for "screen media" (screen.css) */
body {
    margin : 0px 0px 0px 0px;
    font-family : verdana, helvetica, sans-serif;
}

#header {
    width : 750px;
    float : left;
}

/* Special logo used only on handheld devices (hide from "screen media") */
#logo { display : none; }

/* the page layout uses a two-column format */

#container {
    width : 750px;
    float : left;
}

#content {
    width : 600px;
    float : left;
}

#navigation {
    width : 150px;
    float : left;
}

/* page footer */
#footer {
    width : 750px;
    float : left;
    background-color : #ccccdd;
    color : #000000;
    font-size : 70%;
    padding-bottom : 6px;
}
/* document title */
h1 { font-size : 120%; }

p, ol, ul { font-size : 78%; }

/* used for content that should display only on handheld devices */
.handheldContent { display : none; }

/* end of screen.css */
```

---

## Why are some of the styles using pound signs or dots in their names?

Custom style names that begin with pound signs (#) are known as “ID” styles, and they are used only once in a web page. In this case, only one content section would be used in each page, so I created a **#content** ID style type.

Style names that begin with dots are “class” styles, and such styles can be used as many times as needed within a web page. Since I may need to use the **.hideThis** style more than once in any given page, I’ll make it a class style rather than an ID style.

---

## Step 5: Clean up the code in your web pages

At this stage in the process, you should take a close look at all of the pages on your website. Keep an eye out for any embedded or inline styles that might cause display problems on small screens. You may need to modify your external style sheets to accommodate these styles.

If you’re using tables to lay out any of the content of your pages, such as a web-based form, you’ll probably need to recast the content so that you can eliminate the table elements.

**If you code by hand and would like to create a form without using a table**, take a look at **Recipe 7.14** in *CSS Cookbook*, 2nd Edition, by Christopher Schmitt. This book is available online at Safari: <http://proquest.safaribooksonline.com/0596527411>

In the previous version of the Engineering Library website, I used a table-based layout to create charts of article databases on several of the pages. These charts displayed the name of each article database (with a link), the coverage, the description, and links to guide documents (if available). I realized that I really didn’t need to use a table —paragraphs and line breaks could serve the same purpose. A simple layout like this should work well on both platforms.

---

### **Compendex UCB access only**

1884–present

Indexes over 5,000 journals, conferences, technical reports, and other materials related to the engineering and technical literature.

» [Quick Guide \(HTML\)](#)

» [Quick Guide \(PDF\)](#)

### **INSPEC UCB access only**

1898–present

Indexes over 4,000 scholarly journals, conference proceedings, books, reports, and dissertations in physics, electrical engineering and electronics, computers and control, and information technology.

» [Quick Reference Guide \(HTML\)](#)

» [Quick Guide \(PDF\)](#)

---

## Step 6: Add the new style sheet to your pages

Now that your new “handheld media” style sheet is ready for use, you can add it to the **<head>** section of each of the pages on your site. If you’ve also created a new external “screen media” style sheet, you can put it in place as well.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html lang="en-US" xml:lang="en-US" xmlns="http://www.w3.org/1999/xhtml">
<head>
...
<link rel="stylesheet" media="screen" type="text/css" href="screen.css" />
<link rel="stylesheet" media="handheld" type="text/css" href="handheld.css" />
</head>
```

If you already have a link to a “screen media” style sheet in your pages, you’ll need to add the **media=“screen”** attribute and value pair so that the user’s browser will know which style sheet it should load for “screen media” users.

If you don’t specify the media, the default setting of “all” may be used. That may result in the browser ignoring the “handheld media” style sheet, even when a mobile device user is viewing your pages.

**Tip:** you can use the global search and replace feature to add the link to the new style sheet code to all of your pages at once. This works in non-WYSIWYG text editors, such as HomeSite and BBEdit. There’s a similar feature in Dreamweaver, as well.

### Drupal users:

- You might be interested in using the **Switchtheme** module — this module can be set up to let your users choose between your “screen media” theme and a “handheld media” theme.
- Take a look at the “mobile” theme — this theme was created with mobile device users in mind.
- If you’ve already created a custom theme (or plan to create a custom theme), you can add a link to an external “handheld media” stylesheet in your theme’s template documents.

## Step 7: Add mobile friendly styles to the content of your pages

To make the appropriate branding display as desired in each media type, you'll need to refer to both of them in every page on your site. Here's one way of doing this using the `<div>` XHTML element:

```
<!-- displays on desktop or laptop computers -->
<div id="header">

</div>

<!-- displays only on handheld devices -->
<div id="logo">
<!--this will make a clickable link back to the home page -->
<!-- but leave it off the home page -->
<p><strong><a href="index.html">Your Library Name</a></strong>
<br />University of California, Berkeley</p>
</div>
```

Since mobile device users may not be able to use a mouse or other pointing device, they may benefit from having access to a special table of contents or other supplementary navigation. However, you may not feel the need to provide these features to "screen media" users. Here's where you would use the `.handheldContent` style from your `screen.css` document:

```
<div class="handheldContent">
<p><strong>Contents:</strong></p>
<ul>
...
</ul>
</div>
```

Since the complete site navigation won't appear on handheld devices, I suggest that you add an additional link back to the home page so that these users would have an easier time moving around the site. A good place to add this link would be right above the footer on each page.

```
<p class="handheldContent"><strong><a href="index.html">Your Library's
Name</a></strong></p>
```

---

**Tip:** as you work, check your pages using your emulators. If you use the Opera browser or the Openwave browser emulator, you can check the pages from your hard drive.

**Note:** when you're referring to a custom "ID" or "class" style within page content, leave off the pound sign (#) or dot at the beginning of the name.

---

## Getting ready to launch!

When your new mobile friendly redesign is ready to go live, it's a good plan to do some online testing first. If you have a handheld device and an account with web access, that will come in handy and you can also use your emulators. If you work in public service, you might invite your regular clientele to do some testing with their own devices, and ask them to let you know if they notice any problems or have suggestions for improvements.

Depending on what your working environment is like, you may have two options for doing the initial testing.

- If you have an online testing area or sandbox, you can install everything there.
- Create copies of several of your new mobile friendly pages, and install them on your regular website as "testing" pages. I chose to do this for the Engineering Library website, with pages that I called "index\_testing.html", "ask\_us\_testing.html", and "tutorial\_testing.html." I also installed a testing version of a course guide and a subject guide.

When you're finished with these tests — and are satisfied with the results — you can do either a "soft" or a "hard" launch. Since the Engineering Library's website contains hundreds of pages, I chose to do a soft launch. This would make it possible for many users to put the site through its paces and give me a chance to resolve any problems that users report before announcing the new service to the world.

On March 30, 2009, I installed all of the mobile friendly pages in the Engineering Library's regular web space, at <http://www.lib.berkeley.edu/ENGI/>. This was a soft launch, with no formal announcements, although we did some word-of-mouth publicity and invited feedback from our regular visitors.

The hard launch happened as scheduled in August 2009, before the Fall Semester began. At that time, we publicized the new service.