Mobile Web
Developer's Guide

Part I: Creating Simple Mobile Sites for Common Handsets

DotMobi Mobile Web Developer Guide

Part I: Creating Simple Mobile Sites

Published by mobile Top Level Domain (mTLD), Ltd. 11 Exchange Place, IFSC, Dublin 1, Ireland

Contributing Authors: Ronan Cremin, Jo Rabin, Brian Fling, D. Keith Robinson

Printing History: March 2007: First Edition

1. Introduction

The Web has revolutionized how we interact with and publish information, but up to now it has only been accessible to people with desktop devices. Web-enabled mobile phones now extend the expected global reach of the Web to three times that of today, touching one-third of the population[1] of the planet.

The goal of this guide is to provide developers and site owners with enough knowledge to get started with the creation of Web content for mobile users. It covers the benefits of publishing for mobile users, how mobile delivery differs from desktop delivery and how to design for the mobile context.

Before now, Web publishing for mobile users has been something of a mystery, partly because of a lack of information. This guide forms part of dotMobi's efforts to change this by providing authoritative and comprehensive guides, best practices and methods and other material describing how to publish for mobile.

The guide aims to provide an introduction for those not familiar with the Mobile Web. It contains techniques and information required to create a basic site that will work well on the majority of phones. It is not an encyclopedia of past and present technologies and techniques, but provides a place to start. Although using a .mobi domain is recommended as a clear way to indicate to the user that a site is mobile-friendly, the advice presented here applies to any mobile site.

[1] T-Mobile, Credit Suisse First Boston and Pyramid Research report

After reading this guide, you should have a firm understanding of the basics of mobile presentation: what to do and what not to do. This should help you determine how your organization can mobilize its online strategy.

Part II of this guide will address more advanced topics such as adaptation of content to suit the capabilities of more advanced devices and provide a better experience for their users. We provide an introduction to this in Chapter 9. Going further with Adaptation.

The Benefits of the Mobile Web

Some of the more important benefits of the Mobile Web are:

- It can enable access to information, any time and anywhere there is cell phone coverage. By freeing information from the restrictions of a desk or search for a nearby WiFi hotspot, people can quickly retrieve and exchange information.

- It provides vast connectivity. One-third of humankind currently has access to the Internet through a mobile device. This number is twice as many as the number of Internet-connected personal computers (PCs). By 2010, it's expected that half of the planet's population will have access to the Internet through a mobile device.[2]

- It enables services to take advantage of mobile device capabilities such as clicking on a phone number to call it or add it to the device address book.

---

[2] Informa Telecoms & Media (2007)

- It can provide location-sensitive content. Location technologies can enable location-sensitive information be provided to a user. This can reduce the steps required for the user to reach useful content, and so makes it accessible with the least amount of effort

DotMobi

mTLD Mobile Top Level Domain, Limited (usually known as dotMobi) is the ICANN-appointed global registry for the .mobi domain. Backed by 13 leading mobile and Internet organizations, .mobi addresses the need for seamless access to the Internet from mobile phones.

DotMobi is the first and only top level domain dedicated to users who access the Internet through their mobile phones. With four mobile phones purchased for every one personal computer, there's a world of people for whom the mobile phone is the main access point to the Internet. And those users can trust that a Web site is compatible with their mobile phone when that site's address ends in .mobi.

In a fast-growing mobile society, businesses, organizations and individuals need to reach and interact with their customers while mobile. A .mobi address allows them to bypass the constraints of geography, mobile service providers and handsets to reach their audience.

For content providers, it opens a new and more profitable revenue stream, without having to rely on mobile service provider portals.

Conversely, for mobile service providers, the .mobi domain provides them with the ability to take advantage of profitable data services while ensuring a good

user experience, and in return, gain customer loyalty. This is because the dotMobi company ensures a predictable, consistent experience on a mobile phone by encouraging site owners to use dotMobi Switch On! TM Guides, based on World Wide Web Consortium (W3C) recommendations.

DotMobi, the company behind the .mobi domain, is backed by 13 of the world's leading mobile and Internet players – the same companies who have delivered the promise of today's information society: Ericsson, GSM Association, Google, Hutchison, Microsoft, Nokia, Orascom Telecom, Samsung Electronics, Syniverse, Telefónica Móviles, TIM (Telecom Italia Mobile), T-Mobile and Vodafone.

References and Resources

- List of interesting dotMobi sites: http://showcase.mtld.mobi/

2. Mobile Web Strategy

When considering a Web presence targeted at mobile users, the first question to ask is, "Why should it be mobile?" Simply having the content and the ability to mobilize it is not on its own, enough – you should focus on providing content that is directly useful to a mobile user and takes account of the differences in device characteristics.

Some recent mobile handsets can render existing PC-focused web sites. However, the user experience is often sub-optimal – the most relevant content for the mobile user may be difficult to access. Some particularly rich sites may not render at all on the mobile and viewing the site may be slow and expensive.

After asking "Why?", ask "What content should I make mobile?" "What need do I serve by making my content available to mobile users?" "What value does it add to their lives from a mobile perspective?"

The key point here is that certain things may not make sense in the mobile context. For example, completely re-creating the interface of content-driven Web site on a mobile device isn't a good way to start a mobile project. A sports-based Web site that provides stories and scores might work well with text-based scores for mobile devices. However, it's likely that when presenting the information on a mobile device you will probably want to change the navigation structure and you might want to have condensed versions of the stories.

The example emphasizes the need for starting with basic user goals in creating a successful mobile strategy – take away any content that doesn't support the basic user goals and use cases. You should constantly ask, "Why should it be mobile?"

to keep your mobile presence aligned with user goals and to build a compelling mobile experience.

That said, bear in mind that some users will want to experience the full desktop version of your site while mobile, and should not be prevented from doing so – see chapter 9. Going further with Adaptation.

Balancing Goals and Constraints

You should start a mobile project by defining the overall goals for the site. Consider grouping these into the following three categories for easier management and using their related questions to define goals:

- **Business**: These come from your project's business goals and should be clearly defined before you begin. What are your organization's goals? How does a mobile presence help your organization achieve its goals? Does it improve my current business reach? Does it provide new business opportunities? Is there an opportunity for innovation? Depending on your business case, you could choose anything between providing a great experience on all mobile devices, or creating a basic web site with free tools in a short time.

- **User**: User goals (and sometimes constraints) reflect your audience's goals. How does your audience benefit from a mobile presentation? What tasks will they accomplish with your mobile content? Does the immediacy of mobile help the user? How will they interact with your content? Context is big in mobile, so the key is to understand your users' wants and needs. Ensure that you value the attention, time and money of your audience, i.e. value your customer.

- **Technical**: Technology and resource opportunities and constraints. Look at these goals as a triangle or three-legged stool. When making decisions, try to find and maintain a balance on all three sides: business, user and technical. If you lean on one side too much or too little, then the other sides suffer.

User goals usually make a great place to start, and this is especially true when considering mobile.

User-centered Design

User-centered design is a popular and smart way to approach interaction design. Understanding the needs of users helps humanize the process and keeps your project in check with their goals. Users know best and will cast their vote by giving their attention to your site – or not.

A user-centered design approach fits especially well with mobile. Start with context. Think about how and where people will interact with your content or application. What content would they want to get through a mobile device? Because of technical constraints and attention limitations, in most cases, the answer becomes clear.

As an example, consider making a restaurant's site mobile friendly. People accessing the site from a mobile would probably want a textual menu, location and directions, contact information and hours of operation. Remember that mobile users often usually require information that they can action - so, as an example, providing the day's specialties on the site would be a good feature to offer – with the phone number on the same page. Technological constraints

mean that you should verify the capabilities of the user's device before offering features like online ordering and a PDF-based menu.

Thinking about how someone interacts with a mobile device and the context around that provides the best first step in creating an effective mobile experience.

Know Your Audience

Knowing your audience is an important principle in developing your mobile strategy. It is important that you understand what your audience is looking for so you can anticipate how they will want to navigate your site, and so provide them with quick and efficient navigation.

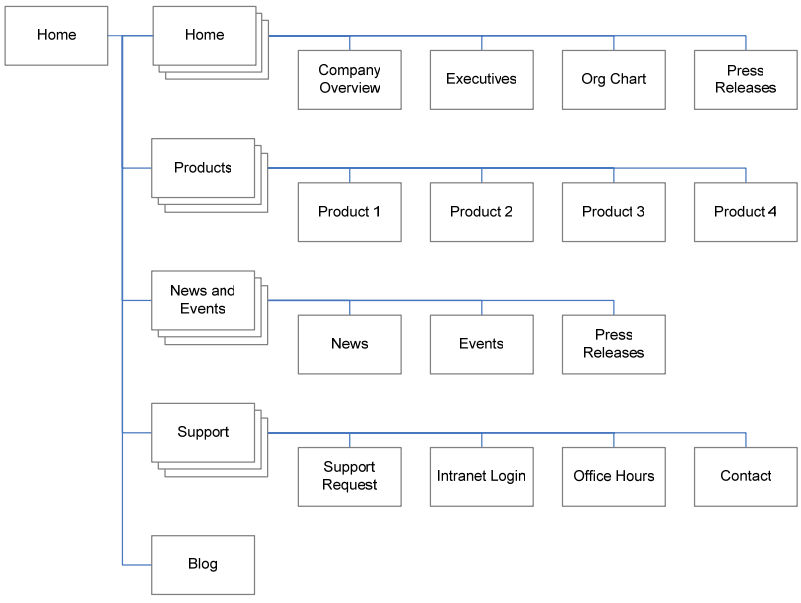
Analyzing demographic and market data on your target audience helps with identifying the needs of your mobile user, but nothing works better than engaging them in conversation. Talk to friends and family on how they would use your content on a mobile device, and then branch out to friends of friends.

Professional Web developers use techniques such as focus groups to help them.

3. Mobile Information Architecture

In the mobile context it is especially important to structure information as simply as possible. Placing the right information in the right place is an important part of providing a usable experience; getting it wrong means providing a poor experience.

While building the mobile information architecture, think about users' "click investment". Even at 3G speeds data retrieval on a mobile phone takes significantly longer than we are accustomed to on a desktop PC. It doesn't take long for users to become frustrated with lengthy sequences of retrievals, and give up. So getting this right is extremely important.



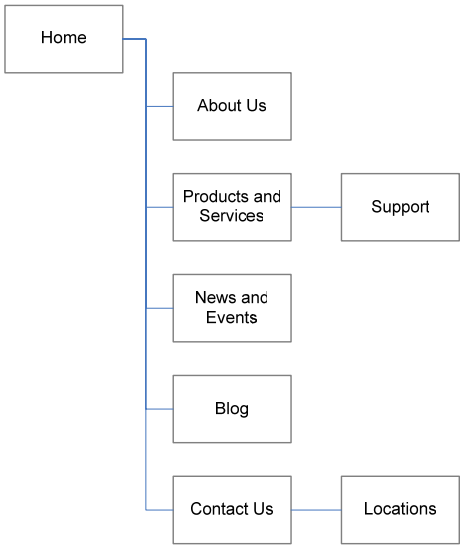
On the other hand, a user presented with a clear goal and intelligent ways to reach it, may tolerate latency issues and the time taken to reach the goal. This is best done by setting user expectations. Every link should use clear labels to communicate to the user what to expect on the resulting page, thereby lowering the risk of click disappointment.

Keep It Simple

The best advice for creating a strong mobile information architecture is to keep it as simple as possible. The following two approaches work well when structuring your IA for mobile presentation:

- Limit choices. Take the content that's relevant to a mobile user and discard the rest. This results in a simple and focused IA that cuts down the risk of the user getting lost. This approach works well with small, focused sites.

- Create a simple site drill-down architecture, nesting content into well-labeled categories. While this sounds straightforward, it's necessary to plan carefully before taking this approach. A typical Web site has sub-pages - users follow a link to reach the sub-page. This is drilling down to find the information the user seeks on the Web site.



Drill-Down Recommendations

Here are a few suggestions to keep in mind when considering a drill-down approach for your site:

- Limit categories as much possible: Users become increasingly disoriented as they go deeper into a site. With a Mobile Web site, their tolerance often stops at about five levels – the fewer levels the better You should adjust this number upwards when targeting more capable devices.

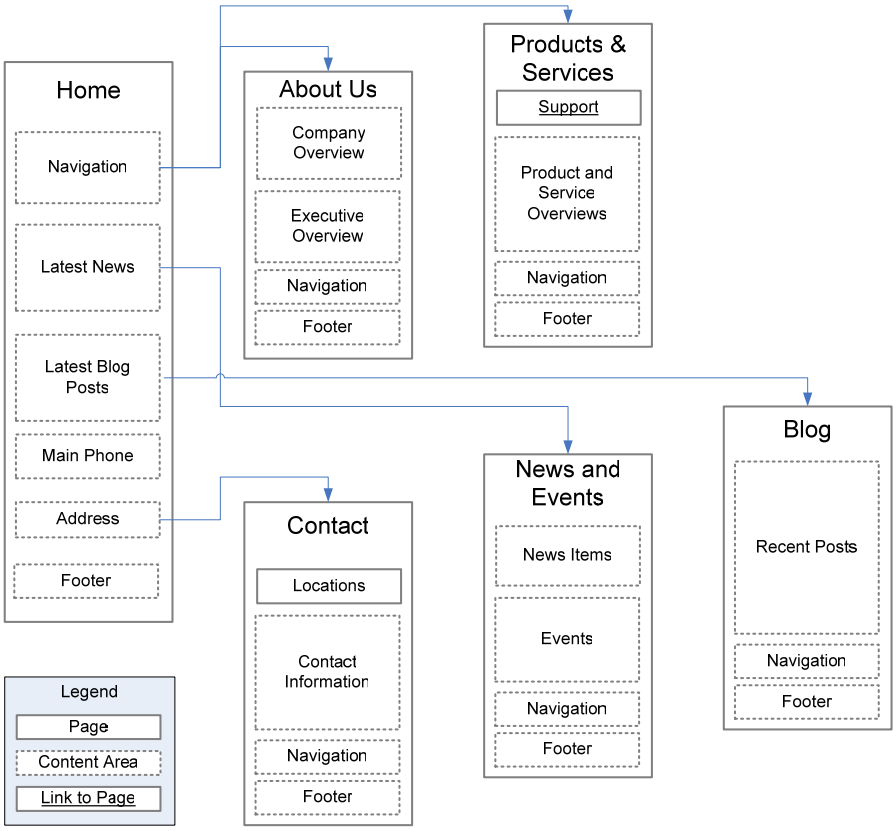
- Attempt to limit links to ten per page unless you know that the device can present more information well: You should code your links with accesskeys, so that the user can use the phone keypad to navigate links. We recommend a assigning up to ten accesskeys to any page (0–9) to ensure compatibility with older devices. You should adjust this number upwards when targeting more capable devices that can display this information better.

> **Jargon**
>
> **Accesskey**: An attribute of XHTML elements, like A and INPUT, the accesskey allows users to use the numerical keypad of a mobile device to quickly navigate to areas of the document or site.

- Provide at least one content item with each category page i.e. avoid empty links: Giving users at least one sample of the content within a category is a good way to make sure that users get to the right place. Consider placing a link to the featured content with a one or two sentence overview. The shorter, the better.

- Prioritize links by activity or popularity: This is often referred to as "deck placement." Sorting links in order by frequency of access ensures that the most sought-after content appears first on the list. You improve a user's

chances of getting to the right destination. The drawback, however, is that lower listed and newer items won't perform as well. You can work around this by maintaining some editorial control and featuring specific categories or content within the category.



Once you determine your content requirements and define your category structure and labels, compile them into a simple site map. This gives you a high-level overview of your information space. Beware: It won't give you a

complete idea of how the real site interaction will work – testing on real mobile devices is always the best way to check this.

Click Streams

An important difference between developing for mobile and developing for the desktop is creating the right flow of information for the user. Due to the limited screen size, the mobile designer might need to spread out information into multiple pages rather than present it on one page. A well-designed click stream is an important concept in mobile design and development.

This method is effective in showing what actions users will likely take to reach their goals and is similar to the model used on PC-oriented Web sites when long articles are published. By combining the concepts of a wire-frame and a traditional click stream, we can more accurately recreate how the user will interact with our information.

Jargon

Deck: A term to describe a mobile Web site. Its origins are from WAP 1.0 and WML development which uses the metaphor of "cards" as pages.

More specifically, a Carrier Deck is used to describe a Web presence maintained by a mobile service provider. When you access the Web from a mobile device, the first page you see is often referred to as the carrier deck.

Deck Placement: The term used to describe where a third-party vendor Web site or application will appear on the Carrier Deck. Default order of content on most Carrier Decks is determined by sales. New items often have temporary "Top-Deck Placement"

Mobile Service Providers

If you plan to have your site accessed through a mobile service provider portal – often referred to as a carrier deck – then note that creating a click stream is a crucial and often required deliverable.

They usually want to see an early click stream prior to development and may make suggestions to improve usability. Some providers require you to submit a

click stream or site map when requesting consideration for the provider's portal or better deck placement.

Prototyping

While click streams give you a way to visualize how your mobile information architecture will work in an ideal world, it doesn't show what you could do better. If you plan to test the usability of your information architecture with potential users (as you always should), creating a prototype is a cheap and easy method of finding potential and costly problems early on in the design and development process. There are two main types of prototypes, paper prototypes and HTML prototypes.

Paper Prototype

A paper prototype provides a simple way to gather valuable feedback on the interaction of your site. You can create a paper prototype in a couple of hours and present it almost anywhere. It can be tricky to find people to participate, but you can usually muster a few volunteers from your colleagues, friends and family.

The most valuable point of a paper prototype exercise comes from having a dialog with participants. Ask how they use their mobile phone. When? Where? Using their answers, you can determine what would be valuable information for them to access through their mobile phone and why they would use it. Using your paper prototype, have a prepared script of tasks for

Jargon

Paper Prototype: Taking printed or sketched wire frames and presenting them to users asking them to perform a series of tasks. The facilitator acts as the "computer" changing the pages as the user makes selections.

Paper prototyping is an excellent method of doing early usability testing for mobile interfaces.

them to perform. Start the activity by explaining what goal you want them to reach. For example, "Find today's movie times." Use this time to interject conceptual asides like, "Is this something you have done, or considered doing on a mobile phone?" Their experiences or perceptions about mobile technology offer valuable insight as to how they will perform the task.

Keep the tasks simple and avoid confusing them. Most importantly, leave room in your script to improvise. If they get stuck on a particular screen, draw a new script on the spot and see if it works better. Don't rush from participant to participant. After each study, you gain insight and find answers to some problems. Keep modifying your script and screens for each participant until it's right.

HTML Prototypes

The HTML prototype is another method of prototyping that can take a little more time to create than paper prototypes, but they can provide greater flexibility in testing. Create each of the primary screens just as you would a paper prototype, but instead build them with HTML. This lets you to link each of the screens together creating a more realistic example of your interaction.

You can remotely test a simple HTML prototype through any desktop Web browser or on the mobile device itself. While this testing method works in an observed environment like a paper prototype, it's harder to make changes on the spot. The real benefit comes from the ability to email the prototype to others asking them to perform one or two general tasks. The participant provides feedback either by email or a survey.

HTML prototypes offer a handy way to conduct iterative development, especially in a medium that's difficult to test. Getting feedback from real people while developing naturally steers the project toward a much more usable and valuable experience.

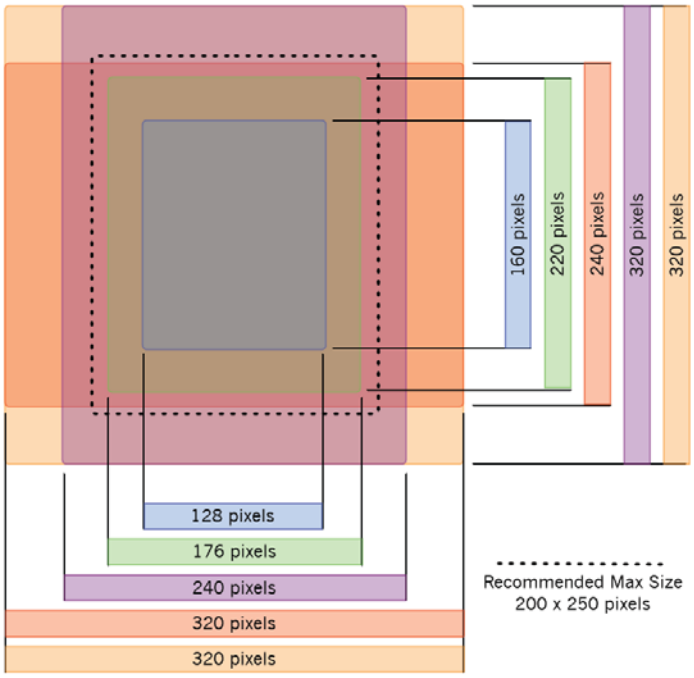
4. Mobile Web Design

While designing for mobile may sound limiting compared to the sophistication of designing for desktop-oriented Web sites, there is nonetheless great scope for creating useful and appealing designs for mobile.

You can create rich mobile experiences but care needs to be taken to match the design to the capabilities of the requesting device: a site is only as good as the browser that displays it. Devices and browsers are already very good, and improving rapidly, but you can still expect to encounter hurdles like slow load times, reduced device compatibility and inconsistent stylesheet support.

Designing for Different Screen Sizes

Outside of the Mobile Web we're lucky to have a manageable number of different screen resolutions – most PCs support resolutions of approximately 1024 x 768 pixels and have a full keyboard and mouse. With the Mobile Web on the other hand, there is a lot more diversity in the physical attributes of the devices – screen sizes and keyboard layouts vary hugely across the range of devices currently in use.



By separating these screens into different classes of devices you can narrow the number of screen sizes to worry about and so reduce the complexity of the design hugely.

You should also bear in mind that it's the screen width more then the screen length, which defines the usability and attractiveness of the outcome. Images that look good in a lower-end phone with a low resolution screen may fill only half or a third of the screen of a high resolution phone, and may not be useful without zooming.

This comes back to knowing your target audience and the devices they're likely to have. Once you know this, you're better equipped to decide what screen size(s) to target when doing the design.

Part II of this guide will go into a lot more detail on the tools and methods for dynamically adapting content for the requesting device.

Designing for the Right Device

When designing for mobile, think about the different *classes* of devices. The line between devices is not well defined, making designing for mobile more challenging. Instead, the boundaries shift constantly. Despite this, some simple guidelines exist to help you determine what device class to target. The mobile devices available today can be broken down in to a few broad classes:

1. **Feature Phones**: These are the most common device type. Feature phones usually come in candy bar, clamshell or slider form. They have a 12-key layout and typically come with voice, messaging and data capabilities. Most feature phones sold in the past three years also come with built-in digital cameras and media players. Companies typically target these phones to the general consumer.

2. **Smart Phones**: Smart phones share the same features as a feature phone with two primary differences: Its ability to run additional third-party applications and a slightly larger screen. Smart phones typically use a more full featured operating system and companies market them as them as advanced multimedia devices to consumers or as productivity devices to the business sector.

3. **PDAs**: These devices — evolved from the PDAs of the '90s — now often include voice, messaging, and data capabilities. PDAs have much in

Jargon

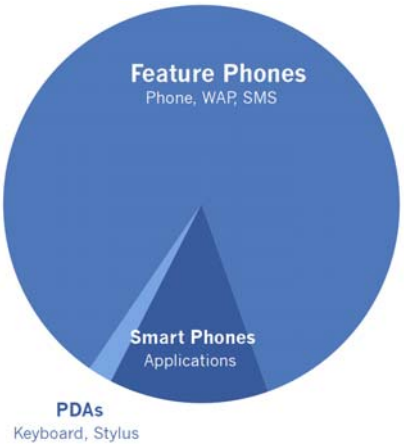
Candy bar: a phone design that uses a single rectangular form with the keypad arranged below the screen

Clam shell: a phone design that folds in half, with the screen on one half and the keypad on the other.

common with the smart phone but differ in that much of their functionality is primarily oriented towards organizational tasks rather then voice communications. Another difference is that PDAs often include QWERTY keyboard and stylus in place of the 12-key layout on normal phones. They also feature a larger screen that can often switch between portrait and landscape mode.

4. **Voice-Only Phones**: These devices are typically extremely low-cost phones aimed at developing markets and are not relevant in the context of the Mobile Web.

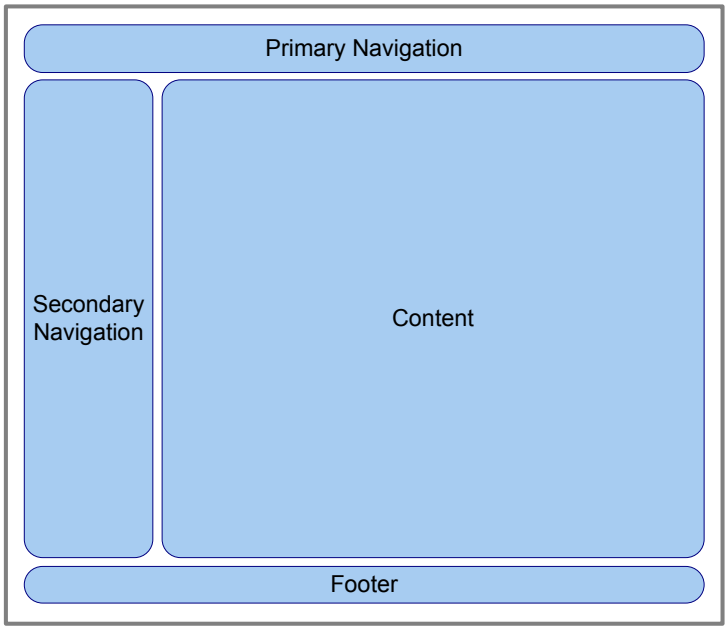
Feature Phones lead the market by a large margin but bear in mind that the borderline between the Feature Phones and Smart Phones is constantly shifting towards the Smart Phone category – the newest Feature Phones are often equal in functionality to yesterday's Smart Phones.



Mobile Web Navigation Paradigms

We are accustomed to a variety of navigation schemes on the Web such as tabbed navigation schemes and menus located on the sides of the main page content.

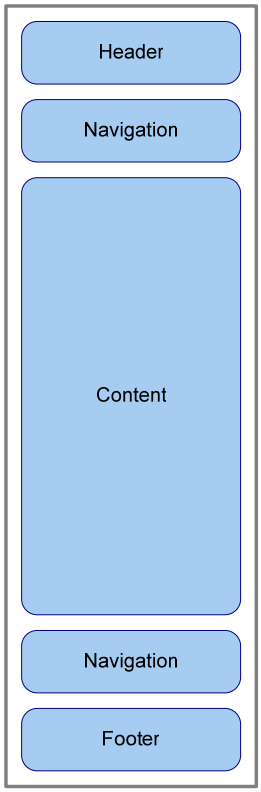
These schemes give us useful visual clues about where we're located within the site and provide reference points on how to navigate within the site. Obviously, this is more difficult to do in a mobile context because of the mobile devices' limited screen size and navigation capability constraints. While it isn't impossible to use desktop style navigation scheme like tabs on a mobile device, they generally do not work as well as they do on the desktop due to the limited device screen size and pointing capabilities.



The preferred and most common method of creating mobile navigation schemes is to use a simple vertical list of options, often assigning and listing the corresponding numbers (0–9) to the accesskeys for keypad navigation. You can design this list in many ways using stylesheets or images. You should consider supporting more advanced navigation methods for higher-end phones to ensure a rich user experience on these devices.

Showing multiple levels of navigation within your list usually doesn't work well because it gives users too many options and consumes valuable screen area. A better way is to show only the options related to the page they're viewing.

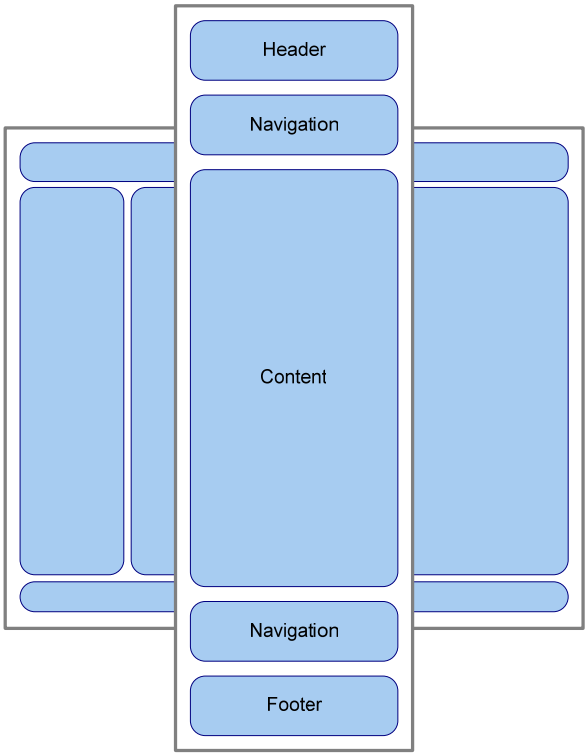
However, you should provide escape points, either as links to the next section, to the parent section, to the home page or all of the above. These links usually work best at the bottom of the page and allow the user to move on without scrolling back to the top of the screen.



Layout & Information Design

Not only does the desktop Web rely on navigation schemes we have taken for granted, but also we also tend to design for it in landscape mode, where the pages are wider than they are tall. Designing for the mobile requires switching the thinking to portrait mode where the content is typically taller than it is wide.

Most landscape layouts and navigation schemes – like horizontal tabs and columns of text – don't well work on the mobile. Instead, think of the mobile like a page in a book with a portrait orientation. So use a single column with text that's left justified.



References & Resources

- Mobile Design Patterns (http://patterns.littlespringsdesign.com)
- Mobile Usability: How Nokia Changed the Face of the Mobile Phone, by Christian Lindholm, Turkka Keinonen (McGraw-Hill Professional)

5. Mobile Web Standards

Many different interpretations exist for the term "Web standards." People overuse this term, creating confusion regarding its real meaning.

"Web standards" can refer to the actual specification of how a language or technology works. An industry standards body, such as the World Wide Web Consortium (W3C), produces these specifications.

"Web standards" can also describe the techniques of applying the language or technology as recommended by the standards body. These are essentially "best practices" or a development philosophy. One example of this is the strict separation of structural markup from visual presentation using cascading stylesheets (CSS).

There have been accepted industry Web standards and best practices for mobile development in place since the late '90s. The currently accepted mobile standards continue to evolve along with mobile devices.

However, mobile devices throughout the world vary in how they render content. This resembles the differences between Netscape and Internet Explorer in the late '90s or even Firefox and Internet Explorer today. However, the mobile device space is more fragmented: instead of having a few major mobile browsers, there are many different many browser types and thousands of variants.

The good news is that of all the various browsers, many are simply variations of a previous version. The inconsistency between the browsers is often minimal and not worth the effort to work around.

The bad news is that this problem is not likely to disappear any time soon. For as long as software comes built-in the device from its manufacturer, it's difficult to upgrade. This is especially true for lower-end phones that do not normally allow you to update the built-in applications. The resulting rendering inconsistencies will likely continue as an annoyance for years to come.

That said, this situation will improve as time passes, just as it did with desktop browsers. With increasing adoption, mobile browser developers will continue to improve their browsers, helping stabilize the situation. Furthermore, there is now a broad consensus on the Web standards that should be supported by mobile browsers and support for this baseline set of standards is improving all of the time.

This section of the guide focuses on the most important mobile standards including XHTML-Basic and Wireless CSS.

Wireless Application Protocol (WAP)

WAP was the first widely deployed set of standards for the Web on mobile. WAP is actually a suite of standards that cover both the page markup format (WML) and the protocols used to serve it (WTP, WTLS etc.) The WAP standards suite is managed by the Open Mobile Alliance (OMA).

There have been two major evolutions of WAP.

WAP 1.0 was the dominant standard in the earlier days of the Mobile Web and nearly all mobile service

Jargon

WAP (Wireless Application Protocol): A standard for applications that use wireless communication like mobile phones. Most modern phones support WAP 2.0, which uses XHTML-MP as the primary markup language while WAP 1,0 used WML.

WML (Wireless Markup Language): An XML language used to specify content and user interface for WAP devices. Often mistakenly referred to as WAP 1.0 given it is its default markup language.

providers and mass-market mobile phones in North America and Europe retain support for at least WAP 1.0.

WAP 2.0 is the current version of the standard. One of the major goals of WAP 2.0 was to bring mobile devices closer to the desktop by adopting the following changes:

- Support for the standard Internet communication protocols such as TCP/IP and HTTP rather than the proprietary protocols used by WAP 1.0.
- Adoption of XHTML-MP as the primary markup language.

Wireless Markup Language (WML)

WML was the core markup language of WAP 1.X. WML is an XML-based markup language that differs significantly from HTML (the markup language used in the Web). The term "deck" first applied to WML sites since each interaction or page is a "card" as shown in the following example:

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//PHONE.COM//DTD WML
1.1//EN""http://www.phone.com/dtd/wml11.dtd" >
<wml>
  <card id="main" title="First Card">
    <p mode="wrap">This is a sample WML page.</p>
  </card>
</wml>
```

Jargon

XHTML-MP: A subset of XHTML, used as a markup language for wireless application development. Since it's the default markup language of WAP 2.0, it's often incorrectly referred to as WAP 2.0.

With the development of XHTML-MP, WML is now deprecated, but continues to serve as the fall-back markup language for the WAP 2.0 specification. For this reason, mobile service providers often require creating Web sites with WML as a "safe mode" for older mobile browsers. Since WML is widely supported and is fairly consistent about browser

rendering, it serves this purpose well. The downside is that WML has more limited design capabilities than XHTML and does not support the richer features of more modern mobile devices.

XHTML Mobile Profile / XHTML Basic

XHTML-MP (Extensible Hypertext Markup Language - Mobile Profile) is a specialization XHTML designed to incorporate features useful to mobile devices. XHTML-MP 1.0 was defined by the OMA and is an extension of the original W3C-inspired XHTML Basic 1.0.

Over time, OMA has developed XHTML-MP and now has a proposed 1.2 version of its specification. Thanks to recent alignment efforts between the W3C and the OMA, the proposed W3C XHTML Basic 1.1 and XHTML-MP 1.2 are virtually indistinguishable.

Since both XHTML Basic and XHTML-MP are subsets of XHTML, the transition to producing mobile-friendly content need not be difficult for developers – standard development tools such as desktop browsers and integrated development environments can be used for mobile authoring, and developers do not need to understand a completely new language.

XHTML Basic 1.1 is set to become the standard level of support on mobile devices – at present XHTML-MP is the most widely supported dialect. An XHTML-MP 1.0 browser, and any XHMTL browser (such as a PC browser) will properly render a site coded in XHTML Basic 1.1.

Wireless CSS

XHTML-MP comes with a mobile-friendly means of using CSS to separate presentation from the markup, just like on the desktop. The OMA-managed Wireless CSS standard is a subset of CSS and is also part of the WAP 2.0 specification. Note that Wireless CSS is not backwards compatible with WML.

You can add Wireless CSS to your document the same way as you would for a normal HTML document. Link to an external global stylesheet using the following line:

```
<link href="external.css" rel="stylesheet" type="text/css" />
```

Insert styles at the document head the following example shows:

```
<style>
  p {
    font-size: small;
  }
</style>
```

Wireless CSS supports a lot CSS attributes, but not all of them. Note that more advanced styling techniques won't likely work across multiple mobile browsers. The best advice is to keep your CSS as simple as possible.

Like XHTML, OMA and W3C are working towards producing a harmonized mobile version of CSS called CSS-MP. Work on this is still at a preliminary stage so for now, stick with WCSS.

Remember that confusingly, XHTML-MP comes from OMA, and that CSS-MP comes from the W3C.

The Benefit of Web Standards Techniques

Using common Web standard development techniques means separating markup and presentation. Controlling every element of the visual presentation using style sheets frees content from the medium giving you greater flexibility and control.

Previously, developers used tables (and occasionally nested tables) in HTML markup to hold small sliced up images create the desired, and often pixel-perfect, design. Unfortunately, whenever a site needed a design change, tables required developers to make large changes to the coded markup. Removing tables simplified the process of changing the design's look, feel and interface for designers and developers.

In the early days of the Web standards movement, not all Web browsers supported the extensive use of stylesheets for presentation. In this case, the Web standards developer semantically coded pages meaning they coded the content in the same order as it appeared on the screen. The approach lets users with older browsers that poorly supported CSS to read and use the site.

This merge of techniques has benefited the Mobile Web. Developers produced content for the desktop screen that could easily be adapted for the mobile screen. Thanks to the fact that XHTML-Basic is derived from XHTML, phones supporting XHTML-Basic can often support simple XHTML. In other words, a site coded in XHTML may also work on a mobile device.

Please note that the one-size-fits-all approach represents the most basic approach to mobile design. If you have capabilities and tools to develop content that recognizes the characteristics of the device and serves specially formatted content to it, this is perfectly OK under the dotMobi guidelines.

W3C Initiatives

In spring of 2005, the World Wide Web Consortium (W3C) created the Mobile Web Initiative (MWI) which consists of several working groups. The group's goal was to increase recognition of standards and best practices of publishing to the Mobile Web.

The W3C MWI, of which dotMobi is a sponsor, includes industry leading mobile service providers, handset makers, a variety of mobile publishers and mobile developers.

The initial groups are Mobile Web Best Practices (BPWG) and the Device Description Working Group (DDWG).

The Mobile Web Best Practices Working Group's goal is "develop a set of technical best practices and associated materials in support of development of Web sites that provide an appropriate user experience on mobile devices."

Jargon

User Agent Profile (UA-Prof): A link to an XML-based description of the requesting device sent with each HTTP request. The User Agent Profile tells the webserver about the requesting device's basic capabilities so the server delivers the right content to it.

Content Adaptation (or Adaptation): The process of dynamically optimizing content to the restrictions of the requesting device. In other words, an adaptation model relies on the mobile device's user agent profile to tell the server to deliver markup or images based on the browser, screen size and device capabilities. Mobile service providers typically require submitted mobile sites use the adaptation model to get a place on the carrier deck0.

The Device Description Working Group's goal has a more technical focus "to enable the development of globally accessible, sustainable data and services that provide device description information applicable to content adaptation." The organization addresses the problems with inconsistent or incomplete User Agent Profiles (UA-Profs) that are required to execute content adaptation to specific mobile devices.

W3C Mobile Web Best Practices

The W3C Mobile Web Best Practices Working Group (BPWG) has made a concerted effort to understand and address the majority of issues that face the Mobile Web user. The goal of the group is to advocate a variety of coding principles and publishing best practices to developers, publishers and mobile service providers.

The W3C Mobile Web Best Practices: Basic Guidelines (MWBP) is the first deliverable of the working group and echoes the recognized best practices established since 2000 in the Mobile Web community.

The recommendations and principles of this guide embrace and extend the W3C Mobile Web Best Practice recommendations.

Default Delivery Context

The Default Delivery Context is defined by the BPWG as the minimal environment in which the Web can be experienced. Many of the practices in the Basic Guidelines document make reference to the DDC in their recommendations. Designing Web sites with the DDC in mind helps ensure interoperability with devices of this kind. However, only having those devices in mind might mean that content providers miss the opportunity to provide a better experience on more capable devices

mobileOK

mobileOK is the second deliverable from the W3C Mobile Web Best Practice group. The goal is to create machine-readable labels and a mobileOK trustmark to indicate that the Mobile Web site adheres to the Best Practices

recommendations. While optional, the mobileOK scheme is intended to help increase advocacy, education and adoption of a people-centered Mobile Web experience. The mobileOK scheme currently supports 2 levels, mobileOK Basic and mobileOK, corresponding to two different levels of testing required. mobileOK scheme is appropriate for low end devices. MobileOK is a useful tool to help site builders to check that their work well on low-end devices but developers should not limit themselves to targeting such limited devices.

This guide's recommendations and principles support the W3C Mobile Web Best Practices and the mobileOK scheme. The dotMobi MobiReady Report validates and ensures sites are compliant with the mobileOK scheme as well as providing additional helpful information.

dotMobi Registrant Rules

dotMobi has worked in conjunction with all major mobile service providers, World Wide Web Consortium (W3C) and the Mobile Web development community to develop recommendations and best practices meant to simplify Mobile Web site development.

DotMobi wants to ensure that all .mobi domains provide the best possible experience to mobile users. Therefore, dotMobi requires all registrants to adhere to the following rules:

- All dotMobi sites must be available in the standard XHTML-Mobile Profile 1.0 or later (e.g. XHTML Basic 1.1), and this must be the default presentation, unless the site knows that the device supports something else.

- A mobile user must be able to access a dotMobi site from http://www.domain.mobi as well as http://domain.mobi
- A dotMobi Web site must not use frames.

In addition to these mandatory rules, dotMobi recommends that developers follow W3C Best Practice guidelines, but will not enforce these.

References & Resources

- WAP 2.0 Specification (http://www.wapforum.org/what/technical.htm)
- WAP 2.0 Technology White Paper (http://www.wapforum.org/what/WAPWhite_Paper1.pdf)
- XHTML-MP 1.0 Specification (http://www.openmobilealliance.org/tech/affiliates/wap/wap-277-xhtmlmp-20011029-a.pdf)
- Comparison of XHTML Mobile Profile and XHTML Basic http://pc.dev.mobi/?q=node/119)
- W3C Mobile Web Best Practices Working Group (http://www.w3.org/2005/MWI/BPWG/)
- W3C mobileOK Scheme 1.0 (http://www.w3.org/TR/mobileOK/)
- MobiReady Report (http://ready.mobi)
- W3C Mobile Web Best Practices Basic Guidelines (http://www.w3.org/TR/mobile-bp/)

- dotMobi Switch On Web Developers Guide (http://pc.mtld.mobi/documents/dotmobi_Switch_On_Web_Developer_Guide3.html)
- XHTML-MP Specifications (http://www.openmobilealliance.org/tech/affiliates/wap/wap-277-xhtmlmp-20011029-a.pdf)
- Wireless CSS Specifications (http://www.openmobilealliance.org/release_program/docs/Browsing/V2_2-20040609-C/OMA-WAP-WCSS-V1_1-20040609-C.pdf)
- XHTML Modularization (http://www.w3.org/TR/2001/REC-xhtml-modularization-20010410/)

6. Getting Started with XHTML

Creating basic mobile-specific Web sites using XHTML is as simple as coding a basic HTML site. Since both XHTML Basic and XHTML-MP are similar to XHTML, it only requires a few changes to format content for the mobile context.

In the next section, we cover the XHTML development recommendations for and best practices for mobile, but we start with the basics of creating XHTML pages.

If you are familiar with XHTML and the basic principles of Web standards, then you should know most of this already.

Let's walk through examples of a mobile-specific page beginning with a look at the character encoding and doctype that start off our page.

Always Use the Correct Encoding & Doctype

Character Encoding

Ensuring the use of the correct character encoding and doctype makes sure that the page renders as expected.

The XML character encoding directive tells the browser how characters on the page should display. It appears on the first line of each XHTML Basic page as the following example shows:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

Doctype

The document type (doctype) tells the browser how the page needs to be rendered, including the rules and how strictly to follow these rules.

Here is an example of a doctype declaration for XHTML Basic 1.1:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.1//EN"
"http://www.w3.org/TR/xhtml-basic/xhtml-basic11.dtd">
```

Always Use Well-formed Code

For those not familiar with XHTML, the first thing to know is that all code should validate (according to the doctype) and be well-formed (a valid XML document).

Here is a basic run down of the rules of well-formed XML as well as the key differences between XHTML and HTML.

- All elements should be closed, e.g. `<br/>`.

  These are also acceptable: `<br></br>`, and `<br />`.

- All non-empty elements should be closed:

  ```
  <p>Example Text</p>
  ```

- All elements must be properly nested:

  ```
  <em><strong>Example Text</strong></em>
  ```

- The `alt` attribute must be used for all images:

  ```
  <img src="image.png" alt="Image Description" />
  ```

- Text should appear within a block level element and not directly in the body: `<body><p>Example Text</p></body>`

- Inline elements should always nest with block level elements:

```
<h2><em>Example Text</em></h2>
```

- All attributes should appear within quotes:

```
<p class="names"/>
```

- All elements and attributes should use lowercase:

```
<p class="Sm">Example Text <hr noshade="true"/></p>
```

There are many online Web tools that you can use to validate your markup including the MobiReady Report and the W3C Validation service. There are also browser-based tools such as the HTML Validator Firefox Extension that can be used throughout the development cycle to keep an eye on your pages.

Always Avoid Using Tables for Layout

We can now add our content in the body of our document, but first we need to add structural elements to contain each logical section, a header, footer and the main body, for example.

With HTML 4, it was common practice to use tables to control the layout of content. This technique, however, constricts the use of our markup by integrating presentational layout into our code. While this doesn't seem like a critical issue, it becomes a big problem when the page is viewed in multiple mobile browsers.

Instead, use XHTML-friendly `<div>` elements to logically contain our content for later styling to control the presentation. Since we usually display text in a single section, the structure is straightforward with a content container in the middle of a header and footer:

```
<body>
  <div id="header">
```

```
      <!-- Header placeholder -->
    </div>
    <div id="content">
      <!-- Content placeholder -->
    </div>
    <div id="footer">
      <!-- Footer placeholder -->
    </div>
  </body>
```

Place Navigation in the Content Body

Unlike on the desktop, it usually isn't a good idea to have a navigation list on every page. Given the vertical orientation of the mobile page, you should show only navigation that's relevant to the page, thereby reducing page weight and scrolling. Thus, the navigation goes into the content body:

```
<div id="content">
  <ol>
    <li><a href="news.html">News</a><li>
    <li><a href="products.html">Our Products</a></li>
    <li><a href="customers.html">Our Customers</a></li>
    <li><a href="about.html">About Us</a></li>
    <li><a href="contact.html">Contact Us</a></li>
  </ol>
</div>
```

Use accesskeys in the Primary Navigation

The primary navigation should include an assigned `accesskey` that corresponds to a keypad number key whenever possible:

```
<li><a href="news.html" accesskey="1">News</a></li>
```

This code links the News item to the "1" key on the mobile keypad and displays the number 1 by it (if the <li> it is part of is the first in the list, of course). Obviously, navigation that exceeds the number of keys on the keypad makes it

difficult to provide accesskeys for lists with more than ten items. While not a requirement for all links, accesskeys are useful for primary navigation.

Use Ordered Lists for Navigation

Unlike on the desktop Web it isn't the best idea to have a navigation list on every page. Given the vertical orientation of the mobile page you should only show navigation relevant to the page, reducing page weight and scrolling. Instead we will add our navigation into the content body.

```
<div id="content">
  <ol>
    <li><a href="news.html" accesskey="1">News</a><li>
    <li><a href="products.html" accesskey="2">Our Products</a></li>
    <li><a href="customers.html" accesskey="3">Our Customers</a></li>
    <li><a href="about.html" accesskey="4">About Us</a></li>
    <li><a href="contact.html" accesskey="5">Contact Us</a></li>
  </ol>
</div>
```

For our home page, we can take certain liberties in providing a description for each link to let users know what to expect in each section. By wrapping the description into a `<span>`, we can use CSS to style it differently from the navigation:

```
<div id="content">
  <ol>
    <li>
      <a href="news.html">News</a>
      <span class="description">Read the latest about our products.</span>
    </li>
    <li>
      <a href="products.html">Our Products</a>
      <span class="description">Browse our product descriptions.</span>
    </li>
    <li>
      <a href="customers.html">Our Customers</a>
      <span class="description">View our customers.</span>
    </li>
    <li>
      <a href="about.html">About Us</a>
      <span class="description">What we do? How can we help
```

```
you?</span></li>
    <li>
      <a href="contact.html">Contact Us</a>
      <span class="description">Telephone, email and location
details.</span>
    </li>
</ol>
```

Linking Phone Numbers

One of the benefits of the Mobile Web is that its users primarily view it on a phone, allowing the user to quickly and easily make phone calls. It's an opportunity to help the user and save steps:

```
<a href="tel:+12065450210">+1 206 545-0210</a>
```

Like any hyperlink, any text could appear between the <a> element to initiate a call. However, the recommendation is to display the phone number.

Dealing with Forms can be Tricky

Entering data into a Mobile Web site can be a difficult and time-consuming process. To avoid wasting the user's time and causing frustration, use forms sparingly.

However, when using forms, reduce the required information as much as possible. The following creates a contact form with few fields:

```
<form method="post" action="process_comment.cgi">
  <dl>
    <dt>Your comment is about:</dt>
    <dd><input type="radio" id="cat1" value="website" accesskey="w" />
    <label for="cat1">Our <span class="accesskey">W</span>eb
Site</label></dd>
    <dd><input type="radio" id="cat2" value="product" accesskey="p" />
    <label for="cat2">Our <span
class="accesskey">P</span>roducts</label></dd>
    <dd><input type="radio" id="cat3" value="news" accesskey="n" />
```

```
    <label for="cat3">A <span class="accesskey">N</span>ews
Article</label></dd>
    <dd><input type="radio" id="cat4" value="other" accesskey="o" />
    <label for="cat3"><span class="accesskey">O</span>ther</label></dd>
    <dt><label for="comment">Your comment:</label></dt>
    <dd><textarea id="comment" name="comment" rows="5"
cols="20"></textarea></dd>
    <dt><label for="email">Your e-mail (optional):</label></dt>
    <dd><input type="text" name="email" id="email" /></dd>
    <dt><input type="submit" value="Send" /></dt>
  </dl>
</form>
```

With all the basics covered, the code looks like the following when putting it all together to create an XHTML Basic homepage.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.1//EN"
    "http://www.w3.org/TR/xhtml-basic/xhtml-basic11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>Site Name</title>
  <meta http-equiv="content-type" content="application/xhtml+xml" />
  <meta http-equiv="cache-control" content="max-age=300" />

  <link rel="stylesheet" type="text/css" href="style.css"/>
</head>
  <body>
  <!--
  All pages start with a common header and navigation bar so that users
can
  navigate back to earlier pages without needing to scroll right down to
the
  bottom of the page. Note that we still provide styling using XHTML
  elements where appropriate rather than by using the equivalent CSS
styles.
  This is so that the pages look their best on devices that do not support
  CSS.
  -->
  <div id="header">
    <h1><img src="logo.gif" width="100" height="50" alt="Company Logo"
/></h1>
    <p><small>Home Page</small></p>
  </div>

  <!--
  The page content is sandwiched between the common header and footer. In
  this case, our content is a set of links to sections. Because the
content
  of this page is static, it is sensible to provide access keys so that
  users can access links immediately. Each section is labeled by the key
  number corresponding to its access key.
  -->
  <div id="content">
```

```
    <ol>
      <li>
        <a href="news.html" accesskey="1">News</a>
        <span class="description">Read the latest news about
        <span class="company">Company</span> and our products.</span>
      </li>
      <li>
        <a href="products.html" accesskey="2">Our Products</a>
        <span class="description">Browse our product portfolio
        and request further information.</span>
      </li>
      <li>
        <a href="customers.html" accesskey="3">Our Customers</a>
        <span class="description">How <span class="company">Company</span>
        products help our customers succeed.</span>
      </li>
      <li>
        <a href="about.html" accesskey="4">About Us</a>
        <span class="description">What do <span class="company">
        Company</span> do? How can we help you?</span>
      </li>
      <li>
        <a href="contact.html" accesskey="5">Contact Us</a>
        <span class="description">Contact information: telephone, email
        and postal details plus a map of our location.</span>
      </li>
    </ol>
  </div>

  <!--
  The common footer provides the copyright statement for your company and
a
  second copy of the navigation bar so that users don't need to scroll
right
  back to the top.
  -->
  <div id="footer">
    <p><small>&copy; Company Ltd. All rights reserved.</small></p>
  </div>

</body>
</html>
```

The following is the associated CSS file `style.css` that provides the styling information for the page:

```
body { margin: 0; }
#header  { background: #e0ffe0; color: green; border-bottom: solid 1px
green; margin: 0 0 5px 0; padding: 5px; }
#header h1 { margin: 0 0 0 2px; }
#header p { margin: 0 0 0 10px; }
#footer  { background: #e0ffe0; color: green; border-top: solid 1px green;
```

```
margin: 10px 0 0 0; }
.company { font-weight: bold; }
.small { font-size: small; }
.description { font-size: small; display: block; }
hr { clear: both; border:solid; border-width:1px; border-bottom-
color:#007300; border-top-color:#ffffff; border-left-color:#ffffff;
border-right-color:#ffffff;}
a { text-decoration: none; font-weight: bold; color: green; }
```

References & Resources

- W3C Mobile Web Best Practices Basic Guidelines (http://www.w3.org/TR/mobile-bp/)
- Global Authoring Practices (http://www.passani.it/gap/)
- Firefox HTML Validator (http://users.skynet.be/mgueury/mozilla/)

7. Recommendations & Best Practices

The transition from understanding XHTML to developing with XHTML Basic (and XHTML MP) – the native markup language of the Mobile Web – can be simple in principle. The real complexity comes with supporting a variety of devices, older devices or trying to programmatically adapt content for different devices, which appears in the next section.

Good Mobile Web development only requires a developer to understand XHTML and the best practices collected over the years by the mobile development community. These practices come from common obstacles that arise when dealing with popular mobile browsers and devices.

Together with the mobile development community and the W3C Mobile Web Best Practices Working Group, dotMobi has compiled Mobile Web development best practices and recommendations. These recommendations aim to create the best possible user experience for the Mobile Web user.

dotMobi has also created the MobiReady Report (http://ready.mobi) for developers to verify that their Mobile Web site meets these recommendations. This section addresses each of the individual best practices for Mobile Web development along with explanation of the problem, the how and the why you should implement them.

Page Information

The following best practices are related to the overall page information rather than specific content within the page.

Character Encoding

Problem: Correct character encoding is critical in making sure pages render correctly on devices. Different document types require different character encodings. XML documents should always have a UTF-8 character set while documents served as MIME type `text/html` should use ISO 8859-1.

Solution: The following line shows how to set encoding correctly in an XML document:

```
<?xml version="1.0" encoding="UTF-8" ?>
```

If you do not specify the correct character encoding for your pages your page may display strange characters when rendered on a mobile device. The recommendation is to use UTF-8 encoding for maximum compatibility.

Browsers assume, by default, that pages delivered with the `text/html` MIME type should use ISO-8859-1. Either way you should set the character encoding explicitly: using the HTTP header and using the XML header.

If you develop Web pages in a Windows environment, note the default character encoding is often Windows CP 1252 – which is similar to, but not identical to ISO-8859-1.

XHTML MP 1.0 Doctype

Problem: Using the incorrect markup type, specified by the doctype, can cause content to render erratically or incorrectly.

Solution: For now, use XHTML MP 1.0 as the default markup for mobile-specific pages.

Applying the following XHTML MP 1.0 doctype tells mobile browsers how to render the content:

```
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
"http://www.wapforum.org/DTD/xhtml-mobile10.dtd">
```

Discussion: Since XHTML-MP is a sub-set of XHTML, many Mobile Web browsers render simple XHTML pages. However, for mobile-specific sites, the recommendation is to use only the elements defined in the XHTML-MP sub-set as your default markup. If you know that the accessing device supports a wider range of markup you can use this to provide a richer experience.

MIME Types

Problem: The MIME types sent my HTTP servers provide important information to browsers on how to treat a document. When sending incorrect MIME types with a document, it may cause the browser to incorrectly interpret and fail to render the document.

Solution: For XHTML-MP and Basic, the recommended MIME type is `application/xhtml+xml`. Unlike HTML, XHTML-MP shouldn't be served as `text/html` except in some very specific circumstances.

Administrators often set up Web servers correctly for more common document types such as HTML and CSS, but not for XHTML. Verify the addition of the XHTML to the list of MIME types on the Web server.

Page Titles

Problem: Page titles surrounded by the `<title>` element are an important, but frequently overlooked page element. Good titles increase the discoverability and usability of Web pages.

Solution: Add a short descriptive page title for easy identification while keeping in mind that the mobile device may truncate the title. Most devices use the page title as a default label for bookmarks, so a title helps the user identify content within bookmarks.

Discussion: It is common to use only the site name as the title, but it doesn't help users as much as it could. A helpful document title consists of the primary title, optionally followed by your site name as shown in the next example:

```
<title>Description of Page Content | Site Name</title>
```

Search engines primarily use page titles to identify content. Also, while naming your pages, think about how users will search for your content on search engines.

Use of Stylesheets

Problem: Use CSS to control the presentation of the page.

Solution: Using CSS stylesheets helps with consistency and centralizing styles, and lowers the overall page size.

Discussion: Many mobile browsers prioritize markup before presentation, loading stylesheets and images last. This sometimes causes markup to appear briefly without styles while the external stylesheet loads. Get around this by adding important styles to the document head alongside using an external stylesheet.

The recommendation is to always use CSS, separated from your markup. This allows you to control the page presentation while decreasing the page size and allowing the browser to cache the CSS for later use.

Objects or Scripts

Problem: Most mobile devices don't support embedded objects or scripts and it's usually not possible for users to install plug-ins to workaround this. Design content with this in mind. Even where a device does support scripting, steer clear of using it unless you can't find another way to reach your objectives. Use of client-side scripting increases power consumption and drains the battery faster.

Solution: Skip scripts and embedded objects.

Discussion: While many modern browsers support scripting it may be disabled so you should ensure that your page works well without it.

If you must rely on either scripting or embedded objects, apply an advanced device detection strategy, routing devices with better scripting support to pages specifically tailored to support those devices.

Auto Refresh

Problem: A Web page uses auto-refresh, but it incurs extra download times and data charges.

Solution: Omit creating periodically auto-refreshed pages unless you notify the user and provide a way to stop it or configure the refresh interval

Redirects

Problem: Using markup to redirect pages increases the load time that comes with downloading and processing another page.

Solution: If you need to use redirects, configure the server to perform redirects using HTTP 3xx codes.

Discussion: Avoid redirects as much as possible since they add both time and cost to for the user to reach the final page.

Caching

Problem: Using cached information sometimes reduces the need to reload resources such as images and stylesheets, thereby lowering download times and costs.

Solution: By specifying cache information on your mobile pages, you lower the number of times devices download common resources. This especially helps with resources like a stylesheet or logo. The following example shows how to use a `meta` directive to set a Cache-control header:

```
<meta http-equiv="Cache-Control" content="max-age=300"/>
```

Discussion: Not all devices support cache control, but caching is especially important for mobile devices due to the high network latencies typically experienced on mobile networks.

Note that it may make sense to wait until you complete development before adding cache information, otherwise you won't see development changes when caching does its job.

Structure

The following practices are related to the overall structure of a page.

Structure

Problem: Developing a Mobile Web site with poorly-formed and un-semantically-structured content.

Solution: It is good practice for documents to show structure with headings and sub-headings. This means coding in order and in a semantically correct fashion to ensure the code elements and the order in which they appear makes sense without manipulating the presentation. The following shows an example of semantic coding:

```
<h1>Top Level Heading</h1>
<h2>Second Level Heading</h2>
<p>Paragraph Body</p>
<h3>Third Level Heading</h3>
<p>Paragraph Body</p>
<h2>Second Level Heading</h2>
<p>Paragraph Body</p>
<h3>Third Level Heading</h3>
<p>Paragraph Body</p>
<h4>Fourth Level Heading</h4>
<p>Paragraph Body</p>
```

Discussion: Using structural markup, rather than formatting effects, makes it easier to modify content when it splits into several pages. Furthermore, structural markup potentially facilitates access to the sections of the document that a user wants. Use headings in accordance with the specification whenever applying

them. For example, they should properly nest based on their level, as shown in the previous example.

Tables

Problem: On smaller screens, tables often don't work well or render erratically.

Solution: Unless you know a device supports tables, shun using tables. Smaller tables with two or three columns work on most devices, but it's not a recommended approach.

Discussion: Use a definition list `<dl>` instead of a table to display data.

Nested Tables

Problem: Nested tables (tables within tables), like tables used for layout, don't work well in mobile design especially since they render poorly and increase the page size.

Solution: Instead of nested tables to control presentation, create well-formed XHTML and control the presentation with stylesheets.

Tables for Layout

Problem: Layout tables, such as nested tables, don't work well in mobile design due rendering inconsistencies.

Solution: The Web design industry considers using tables for layout a bad practice, particularly for mobile devices. It's more efficient to do page layouts with a style-based layout producing a layout that adapts well to the narrow screens. Table-based layouts combine presentation and markup making

development more difficult and adapting a page for other mediums almost impossible.

Discussion: Tables are cumbersome and difficult to support. Table-based layouts restrict your ability to adapt for multiple devices and increase page size. Applying a style-based layout adds flexibility while reducing page size.

Frames

Problem: Frames don't work well in mobile design because most devices don't support them, and they cause usability problems. Frames are not allowed in dotMobi sites.

Solution: Instead of framesets, apply server side includes (SSI) or other techniques for loading local content. Instead of using a frameset to hold a site within yours, use a link to the other site.

Content

The following best practices are related the main content of a page.

Number of Links

Problem: Too many links on a page makes it difficult for the user to navigate and read content. Many mobile browsers stop scrolling vertically on pressing the down key when a link appears in the currently viewed section of the page.

Solution: Try to limit links to 10 links per page and add accesskeys to links whenever possible, so that user can navigate with the keypad rather than having to scroll to the desired link.

Prioritize links by popularity so that frequently accessed links show up at the top. This creates a better experience by giving users quick access to what they want and ensures important content appears above the fold. When prioritizing links by popularity, care should be taken to ensure that new links are highlighted to ensure that they are noticed.

The end of each page should give the user some place to go. This could have a link to the parent category, links to related content, a link to return home or show the navigation list, anything to help the user move forward instead of scrolling to the top of the page for options.

Discussion: Design your content so that users reach frequently accessed information within a minimum number of page retrievals. However, it may lead to more page retrievals to reach less frequently accessed information. As a rule of thumb, users get frustrated if it takes more than four steps to reach their destination. Whether or not this is possible depends on the nature of the site and, in particular, the grouping of items in menus to provide logical themes.

Access Keys

Problem: Navigating a mobile site cam easily turn into a difficult and cumbersome experience.

Solution: Associating an accesskey attribute with each link gives the user an easy way to access the link using the device's keypad. They come in handy when used consistently across a site letting users jump quickly to their chosen sections without scrolling to find a link. The following shows how to associate an accesskey with a link:

```
<li><a href="link.html" accesskey="1">Link 1</a></li>
```

Discussion: You can have more than 10 links per page, but some devices may not have more than 10 buttons. When possible, create accesskeys for all navigational links. You can save some accesskeys by not using them for links appearing within content blocks. It may make sense to use a consistent set of accesskeys across a site so that, for example, the "0" key always goes back to the main menu regardless of what page you are currently on.

Free Text Input Controls

Problem: It's difficult for the user to enter content into free text input controls such as text boxes and text areas.

Solution: While these may be unavoidable in certain cases in forms that need information from the user, try to use text boxes and text areas as rarely as possible.

Cut the need for text entry by relying on radio buttons, select boxes and lists of links.

Default Input Mode

Problem: It's possible to limit the type of data that entered into an input field by defining the input mask or input mode using CSS. This makes it easier for users to enter information into a free text field.

Solution: Automatically set the input mode (alphanumeric or numeric) of the mobile device's keypad according to the input mask value. The following example limits the input to only numeric values.

```
<input type="text" style=' -wap-input-format: "*N"' />
```

The next example limits the input to alpha characters by capitalizing the first letter.

```
<input type="text" style=' -wap-input-format: "A*a"' />
```

Images

Image Sizes

Problem: Thanks to the diversity of mobile screen sizes, some text and block elements automatically wrap to the screen dimensions while large images fall off the screen.

Solution: Edit images so they're as small as possible in terms of pixel dimensions unless you know that the device supports bigger images. Most mobile device screens are about 120 pixels wide. Keep images narrower wider than the screen size unless there's no better way to represent the information.

Discussion: Unless you are using advanced device detection techniques, the width of an image should stay under 120 pixels. Note that if you simplify the image creation this way you should avoid use of images that are composed of dense information since this information may be lost on higher resolution displays where the image is rendered physically smaller.

Declare Image Dimensions

Problem: Not specifying the pixel height and width of an image forces the mobile device to calculate the values, and consequently increases render times and degrades performance.

Solution: Images such as bitmaps have an intrinsic pixel size. Telling the browser its size in advance prevents the browser from having to re-render the page when it receives the image. Letting the server resize the image cuts down the data transferred and the time it takes for the client to process and scale the image. If the specified width and height attributes match the intrinsic size, then the client doesn't resize the image.

Image Maps

Problem: Most mobile devices lack a pointing device like a mouse or rolling ball, making it difficult for users to use server-side image maps.

Solution: Omit image maps unless you know the requesting device supports them.

Alt Text

Problem: Downloading images considerably increases the time to load your page.

Solution: Creating pages that are readable without images lets your users browse your page in text-only mode. As a result, download times and costs go down. If the user has images turned on, textual descriptions help users assess the page's usefulness prior to the images' arrival.

Always provide alt text value for images.

Other Best Practices

Valid Markup

Problem: Using invalid XHTML Basic or Mobile Profile on mobile pages.

Solution: Use valid XHTML Basic or MP on mobile pages for maximum efficiency.

Discussion: Non-validating markup may not display correctly or efficiently on mobile devices. In some cases, especially on older phones, non-validating XHTML-MP won't render at all, leaving users with an error message in their browser. Note that different markup types are acceptable if you know that the requesting device supports them.

Pop-up Windows

Problem: Most mobile devices don't support pop-up windows.

Solution: Even when devices support pop-up windows, changing the current window confuses the user. Avoid the use of pop-up windows.

External Resources

Problem: The client device must separately download every external resource listed in a page (images, stylesheets and other objects). This adds time and cost to view a page.

Solution: Carefully consider the number of external resources you use, limit them and keep each resource's file size as small as possible without sacrificing usability across multiple devices.

Discussion: Most mobile browsers download each resource as a separate element, beginning with downloading and rendering markup, followed by stylesheets and images. Depending on network speed, the user may see the basic markup while external resources download. When the download finishes, the browser renders the page again with the included elements.

Total Page Download Size

Problem: Large pages take longer to load and increase data charges.

Solution: Keep markup, images, stylesheets and all external resource files small in size.

Discussion: The rule of thumb is to keep Mobile Web pages under 10Kb in size, counting both the markup and the included resources. Larger pages sizes of up to about 25Kb may make sense in certain situations – the key point is to remember that there is no hard limit here and that different situations may require different solutions.

References and Resources

- MobiReady Report (http://ready.mobi)
- W3C Validation Service (http://validator.w3.org/)

8. Mobile Publishing

Publishing is probably the most complicated element of the Mobile Web. People from the desktop Web development community need to adjust their view when developing for the Mobile Web.

Luckily, publishing to the Mobile Web does not have to be complicated. There is plenty of room in-between the simple and the complex for any developer and any project to find a home.

While it is easy to segue into the inner workings of device detection of the differences in browser rendering, in this section we focus on establishing a reasonable development baseline and easy ways to begin mobile publishing.

Supporting Devices & Browsers

Supporting devices and browsers can be difficult and time consuming Every year, the industry sells hundreds of different mobile device types/variants worldwide. Mobile service providers often customize mobile browsers on their devices, creating a large number of different deployed browsers.

As a rough guide, every major browser version has a lifetime of approximately 50 different device models, with various minor revisions.

Since most devices don't allow over-the-air updates, faulty devices can't receive regular upgrades leading companies to pull them from the market and replace them with a new version. This typically includes a revised version of the browser, perhaps exacerbating the problem.

Most mobile service providers recommend supporting devices for two years, a difficult task.

Simplifying Device Support

Supporting devices and browsers is a major concern for many developers from the Web community accustomed to dealing with differences in multiple Web browsers.

While supporting multiple mobile devices and browsers gets complex, the situation is not as bad as many believe. This belief comes from looking at the mobile browser landscape in the same way as desktop Web browser landscape.

Consider the following points:

- Only Mobile Web sites that are part of a mobile service provider portal must support all provisioned devices and browsers. Mobile Web sites outside of mobile service provider portals can support whatever devices and browsers the publisher wishes.

- Working to the capabilities of the W3C Default Delivery Context should ensure that a site will work with most existing browsers but remember that a better experience can be provided by recognizing the device and using its capabilities.

- Depending on your target audience and locale, older or poorly designed devices may not need to be supported. Drawing a line with device and browser support is relatively easy because those with older or poor quality devices typically don't use the Mobile Web. You can verify this by reviewing common user agents recorded in your Web server's access logs.

One way to approach this problem is to focus on five classes of devices that span a range of capabilities. Of the hundreds of devices available, supporting five mainstream devices makes a great place to start. Obviously supporting more devices ensures greater compatibility with more devices and developers should always endeavor to do this, but supporting five devices disparate devices can do the job. Here is an example:

- A Nokia Series 40
- A Motorola V series, (v3 aka RAZR, v600, v500, etc.)
- A modern Samsung and/or LG device
- A Smart Phone, like Nokia s60 (or Series 60)
- A PDA, like a Treo or Windows Mobile device

Site Naming

Mobile pages are just Web pages published to any Web server. Nevertheless, it can be problem a problem getting mobile users enter the URL for your site. The following options are some of the ways getting users to access a Mobile Web site:

- Use the mobile-specific .mobi top-level domain (instead of .com, .net or .org) to indicate that your site is mobile friendly.
- If you do not have a dotMobi domain, you may try to educate the user to enter a mobile-friendly URI e.g. example.com/mobile or mobile.example.com.

- Detect the mobile device automatically and redirect the user to a mobile-friendly location. In this case, the user simply enters domain.com. In this case ensure that you provide an option for the user to browse the PC version of the site – many users have advanced mobile devices may choose to access the richer version of the site

For more information on how to structure your site, when to redirect the user to a separate mobile-only only site, see Chapter 9. Going further with Adaptation.

dotMobi

The .mobi top-level domain is the ICANN approved top-level domain specifically for mobile devices giving publishers the option to use an alternate domain for their mobile site. Instead of using device detection, sub-domains or directors, mobile users go to example.mobi.

Much like the sub-domain approach, dotMobi sites sometimes use a separate server or virtual host, separating desktop sites from mobile sites on the same server.

If you own a dotMobi domain, you can route all traffic to it using device detection or server redirects.

Server Directory

You can publish Web sites written in XHTML Basic to most Web servers without re-configuring the server.

This publishing approach works in shared hosting environments, and it's compatible with most content management systems that can publish alternate templates.

Sub-domains

The sub-domains approach to publishing, similar to server directories, lets you publish a mobile site to a location on your server, mobile.example.mobi, wap.example.mobi or m.example.mobi, for instance.

Since most servers treat sub-domains as separate Web sites with a unique root directory and server directives, this approach is convenient for your Mobile Web site because it keeps desktop and Mobile Web sites separated.

It's possible to redirect traffic from a server directory to sub-domains to give the user easier access while taking advantage of separated environments. For example, entering example.mobi/mobile redirects the user to example.domain.mobi

Configuring Server MIME Types

The Internet Media Type – better known as MIME type – is an Internet standard for indicating the format of a message. It is used extensively in many Internet protocols. In the context of the Web in particular, it's used as part of the HTTP protocol to help the browser understand how to parse and render content that it receives.

For each request made from a browser to a Web server, HTTP requires that the server includes a Content-Type header identifying the format of the response. Upon receiving this Content-Type header, the browser can decide how to decode the response and render it in an appropriate form.

In the mobile context, the server sends MIME types to the mobile browser. Many Web servers, by default, do not have the correct MIME types defined for mobile markup standards. It is important set up the Web server to serve the right Content-Type headers for mobile markup languages.

At best this may cause the browser to render the page more slowly; at worst your pages may be unreadable or result the mobile device to produce an error message.

Use the value `application/xhtml+xml` for all recommended mobile content types (XHTML-MP and XHTML Basic).

Site Testing

Testing is an vital component of mobile publishing. Mobile testing is a big job considering the many devices and the differences in how content renders on the device. Fortunately, simple testing methods exist for testing your mobile Web site.

One tip before beginning: Create a simple page on the development server containing links to the URLs for the sites you plan to test. Bookmark this site on each device used in the test. This way you avoid having entering URLs into the devices many times over.

Desktop Testing

You can – and it's recommended – test your markup and stylesheets on a desktop browser before trying it on a device. Though desktop browsers have better CSS support, you can confirm the basic markup and stylesheets. Doing

testing on the desktop comes in handy in finding errors quickly and easily compared with searching for errors on the device.

Opera

The Opera desktop browser has a small screen view that mimics a mobile screen when loading a mobile stylesheet or condenses the page. Be aware that Opera's small screen view is a little wider than mainstream mobile devices.

Frames

Another way to test on the desktop is to create a Web page with an inline frameset or iframe, specify the dimensions to match your target mobile screen and add the URL of your mobile site like the following example:

```
<iframe src ="mobile/index.html" width="240" height="320"
style="border:1px solid;"> </iframe>
```

These steps create a reasonable representation of a mobile device on the desktop. You can even go as far as to wrap the frame with an image of a phone for further realism.

Firefox

When using advanced device detection methods, Firefox – with help from the User Agent Switcher extension – lets you change the User Agent HTTP header you send to the server. Once you add the data from the supported mobile user agents, you can test how each of your sites displays.

Other helpful all-purpose extensions for Firefox are the Web Developers Toolbar and Firebug (which provides very useful XHTML and CSS debugging).

Emulator Testing

Another method of testing a mobile site is to use a phone emulator. This is typically a desktop or Web-hosted application that mimics the device experience for a particular device or class of devices. The accuracy of phone and browser emulators vary from a perfect mimicking of the browser rendering to rough approximations.

While an emulator does not replace testing on an actual device, they are a very useful tool during development to do quick verification of how your code displays without loading it on a real mobile browser. Note that regardless of how perfectly a browser mimics the rendering of page on a real device, they can't reproduce the overall experience of using a real device, since factors such as network speed and latency are involved. For this reason, emulators are a very useful step in any testing program, but should never be used to replace real device testing.

Device Testing

While getting access to devices, networks and the right data plans is a challenge, the outcome is worth the struggle.

Other methods of testing are very useful, but nothing represents the final experience better than testing on a real mobile device during the development process, since this best recreates how the user will interact with your site.

If you have only one device, borrow phones from friends and family. Talk to vendors that may let you rent phones or use their device lab. If you don't have access to a vendor in your area, try going to a mobile service provider store and use the display phones for testing.

Prepaid SIM cards from multiple mobile service provider networks can be a developer's best friend. Switching prepaid SIMs allows you to test on multiple devices across multiple networks, without having to commit to multiple contracts with mobile operators.

Remote Access

Remote access services let you control an actual device remotely through your desktop. DeviceAnywhere is one of the few examples of such a service. The application taps into the actual device, which you rent through the company's software. It supports most of the devices sold in North America and Europe.

Remote access provides a compelling method for testing and has the added convenience of a desktop emulator as well as showing the displayed characteristics of the actual device. Although it doesn't replace the tactile lessons learned from using the real device, it's a solution for small publishers with limited resources. It deals with the problem of testing on many devices some of which are not supported by networks in their own countries.

Usability Testing

The recommended approach of testing Web sites with users, offers valuable insight. Testing is a process that should occur throughout the design and development process rather than being relegated to something to do upon the site's completion. Regardless of the site's size, verifying the discoverability, readability and usability of the site with actual users always creates a better site. This is the perfect time to quote Jared Spool's golden rule of usability testing, "Testing with one user is better than none."

References & Resources

- Opera Software AG (http://www.opera.com)
- DeviceAnywhere.com (http://www.deviceanywhere.com/)
- Firefox User Agent Switcher (http://chrispederick.com/work/useragentswitcher/)
- Firefox Web Developers Toolbar (http://chrispederick.com/work/webdeveloper)
- Firebug (http://www.getfirebug.com)
- dotMobi Online Phone Emulator (http://emulator.mtld.mobi/)
- Nokia Browser Simulator (http://www.forum.nokia.com/info/sw.nokia.com/id/db2c69a2-4066-46ff-81c4-caac8872a7c5/NMB40_install.zip.html)
- Openwave Phone Simulator (http://developer.openwave.com/dvl/member/downloadManager.htm?softwareId=23)
- Opera Mini Simulator (http://www.opera.com/products/mobile/operamini/demo.dml)
- Usable Products (http://www.usableproducts.com)

9. Going further with Adaptation

The range and diversity of devices on the market today presents a challenge and an opportunity for the content provider. The challenge: to achieve a workable, compelling experience across device types requires thought and careful planning. The opportunity: for content providers to stand out from those who don't, by presenting a compelling experience to a broader segment of their target demographic.

Most of this document focuses on getting the basic things right. This means that it pays more attention to working down to the limitations of basic devices as opposed to working up to the capabilities of the growing number of more fully featured devices. In doing this, dotMobi isn't implying that it encourages what is often referred to as a "least common denominator" approach.

A minimal approach has its place when the service targets users who have only basic devices or where the content is naturally simply structured and navigated. However, more and more users are adopting devices that come with advanced features and browsing technologies. Users in some markets who have experienced the Web on desktops have become accustomed to sophisticated experiences. Thus, more manufacturers and software vendors in the mobile space are building capabilities into their devices to meet those expectations.

Your job, as a content provider, is to exploit those capabilities to provide users with the best possible experience when using your service. In this chapter, we look at adaptation, which - broadly speaking - is the process of selecting and formatting content differently based on the differences in device types and circumstances of use.

User Choice

In chapter 2 we discuss how converting your desktop-based site to a miniature one for mobile users is not the right approach. This not only refers to the different capabilities of the users' devices, but also to the users' usually different needs when sitting at a desk compared to using a mobile device.

Throughout this guide, we have emphasized anticipating your users' interests while they're mobile and making it as easy as possible for them to reach the needed content. While this is an important rule of thumb, taking advanced steps in creating a mobile experience requires slightly reconsidering the rule of thumb.

Crucially, when we say "usually different," there are nonetheless some cases where some users may have particular reasons for accessing content usually associated with access from the desktop.

For example, lengthy research reports are, for most people, hard to read on a mobile device. When planning your site, you would not make access to such reports a priority. However, some users, because they need the information and do not have access to a more appropriate device may wish to access the report anyway. Some users in various demographics may not have access to a desktop experience at all.

For these and other reasons, devices coming into the market today anticipate that users, for whatever reason, may wish to access 'the desktop experience' while mobile.

It remains true that typical mobile devices cannot present a full desktop experience, as the size of the screen and more importantly, the lack of a full

keyboard and pointing device remain obstacles to text input and typical desktop navigation.

That said, it is important that in spite of anticipating users' common needs in your mobile experience, you do not prevent them from accessing aspects of your content designed primarily for the desktop experience.

While you should anticipate users' needs and provide them with what you think is an appropriate experience by default, you should allow them to over-ride your assumptions.

Include a link to the mobile experience on the desktop experience and include a link to the desktop experience on the mobile experience.

What Is Adaptation?

In earlier chapters we mention that the mobile space has a range of device types and content often renders better when tailored to specific device characteristics or features. Adaptation is about exploiting those features where they exist, while avoiding them in others. Another part of adaptation requires working around problems found in specific devices.

Adaptation can occur at the server or at the client side (the device). It's also possible for adaptation to take place in the network, and operators frequently do some adaptation between your Web site and the device. However, that form of adaptation is usually outside of your control.

Client Side Adaptation

Client side adaptation usually relies on using the media selectors associated with CSS. Stylesheets can provide a different experience for the same page on mobile by using the 'handheld' media type.

While this can be a useful form of adaptation, it is limited. The mobile downloads the same XHTML markup as a desktop browser, which might unnecessarily taking time and cost the user money with content that's ends up not being displayed.

To include different mobile and desktop stylesheets in your document, add link elements like the following:

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

Alternatively, screen and handheld styles can appear in the same style sheet, using the `@media` CSS rule.

For example, if you wanted to suppress the display of an image in the mobile presentation, identify the image element using the id attribute (calling it 'image1'), and then refer to it within the style sheet:

```
@media handheld {#image1
  {display: none}}
```

Note that most browsers will still download the image, and users will suffer the delay and cost of doing this. So client side adaptation with CSS is mainly useful when varying the presentation style, or order of the content.

Scripting is another technique for client side adaptation. In the desktop world scripting is often used to accommodate differences between browsers. However,

on mobile be especially care because as noted elsewhere in this document, low-end devices often include no or limited support for scripts. But for high end devices—where you know the device capabilities—this can be a handy technique.

Server Side Adaptation

Server side adaptation comes in various forms. At its simplest, it relates to using scripting to vary small parts of the page delivery. One example is changing the HTTP Content-Type header attached to the content. As previously discussed, it's not typical to use the content type `text/html` when delivering to mobile devices. On the other hand, when delivering such pages to some versions of Internet Explorer on the desktop, it can be more effective to use that content type. In this example, the adaptation would consist of a test to see which User Agent (browser) accesses the page and deliver the chosen header based on whether the browser is Internet Explorer or not.

The following is a simple (and incomplete) example of how to do this in the server side scripting language PHP:

```
<?php
$msie = strpos($_SERVER[“HTTP_USER_AGENT”],’MSIE’) > 0;
if (!$msie) {
  header(“Content-Type:
  application/xhtml+xml;
  charset=UTF-8”);}
else {
  header(“Content-Type: text/html;
  charset=UTF-8”);}
?>
```

Other examples of varying the content to better suit device characteristics include presenting an appropriately sized image for the device accessing the page. A good general guidance for low-end devices is to limit images to 120px width.

Those with high-end devices will have a much better experience if the image fits their screen and its color depth.

Because it wastes bandwidth to transfer images that are bigger than the screen can handle, it's good practice to resize images at the server prior to transmission.

Do this by preparing specifically sized images in advance and choosing the biggest size that will fit; or dynamically resize the image at the point of delivery.

Both approaches have advantages: The former is better when the image must preserve specific details (e.g. the ball in a sports photo). The advantage of dynamic resizing is its ability to fit an image exactly to the device.

Device Detection and Characteristics

To be able to resize images and carry out other kinds of adaptation requires that you know the characteristics of the devices.

Most devices (desktop and mobile) send information about the content formats and character encodings they support, but they usually don't provide enough details for a particular application. Some, but not many, devices present some information as non-standard HTTP headers. But in any In any case, it's impractical for devices to send all the information that is needed for adaptation in their headers.

The answer is a database of information known as a Device Description Repository. Currently, the standards world is doing a lot of work on this subject.

The W3C and OMA are working together to create a standard framework for storing useful information with dotMobi actively participating because of its importance to the Mobile Web.

In the meantime, other solutions exist for finding out device characteristics.

One of the better known is WURFL. This open source project maintains and updates a range of information about a multitude of devices. There are APIs which help you access this information from server side scripting.

Another source is the OMA-inspired UAPROF. UAPROF puts an additional header in the HTTP request containing the URL of a file with information about the requesting device. If it hasn't got the file already, the server can find this URL and request the information about the device.

Several companies provide commercial services around Device Descriptions. Mobile Wizards, for example, has collected thousands of UAProfiles and offers free and fee-based services for this information.

Other companies like Volantis, MobileAware, SevenVal, Drutt and Argogroup offer value-added services, including specialist online adaptation services.

Adaptation Strategies

Many adaptation strategies exist, but the one you choose depends on the nature of your content and budget. Such strategies include the following:

- One Size Fits All
- Minor Adaptation e.g. scaling pictures to screen size

- Redirection
- Unified

One Size Fits All

Sites with simple navigational structures can be designed so that they render well on both mobile and desktop devices. In the right circumstances, using a simple direct approach provides the most compelling experience across device types.

Minor Adaptation

An embellishment of the One Size Fits All approach, detecting device capabilities allows you to take advantage of advanced device features, to present better quality images and to include extra features for devices that you know support them.

Redirection

Sites that are more complex are harder to build in a way that makes that every page works well on different device types. This is especially so when the desktop experience has a more complex navigation. Additionally, navigation that designed with mobile user needs in mind can prioritize different aspects of the site and its content than the desktop experience and so make it easier for mobile users to reach content that is important to them.

One solution to this problem is to build a separate mobile site, which could potentially also include minor adaptation to take advantage of enhanced device features. In essence, this approach means that when devices access the home page, the server assesses whether the device is mobile or not.

To do this, the server reviews the User Agent HTTP header and compares the string with a list (such as the one provided by WURFL). There are other clues in the HTTP headers that help you determine if it's a mobile device. Such clues include the presence of a UAPROF, or the device reporting that it supports WAP content types, but beware they're not 100% accurate. Note that a dotMobi site should default to the mobile version of the page if the device is not recognized.

Having determined which experience is appropriate for the device in question, the server sends an HTTP 301 or 302 response containing the URL of the right page such as http://mobile.example.mobi or http://example.mobi/mobile.

The major drawback of this approach is that it separates the mobile content into an area of its own, making it hard to share bookmarks between mobile and desktop devices. It also makes it hard for mobile users to reach the desktop experience if they want or need to.

Another problem is that some more fully featured devices and some versions of Opera Mini (an adaptation solution) try to emulate desktop user agents in order not to route to the mobile experience, which can make it hard for mobile users to reach the mobile experience.

At a minimum, both the desktop-oriented and the mobile-oriented home page should link to the other experience to allow users to choose an alternative, and provide a backup in case device detection has incorrectly determined the device type.

In summary, this approach is probably best considered a temporary solution.

Unified

A unified approach attempts to address the drawbacks of the redirection approach by overlaying the URLs that apply to the various experiences onto each other. This way, bookmarks can be shared between the desktop and mobile. Pages that don't apply to one experience can redirect to pages that do apply.

This approach needs more planning and expertise than the others do, but it ultimately leads to more satisfactory user experiences and repeat visits to your site.

Various companies, including those aforementioned specialize in helping content providers with tools and services for implementing a complete adaptation approach.

Remember

Your assumptions about mobile user context (what they may need or the way they wish to experience it) should guide the design of the default mobile experience. However, do not prevent users from accessing content because of those assumptions, and allow them to choose a different experience. Their device may have been specifically designed to display desktop Web pages, so allow them to access such pages if they want to.

References and Resources

- WURFL (http://wurfl.sourceforge.net)
- Mobile Wizards (http://www.uaprofile.com)

Appendix A: Creating a Mobile-Friendly Site using only Stylesheets

Web sites coded with "Best Practices" in mind (Semantically-coded XHTML/CSS-based design without tables for layout) adapt to mobile devices much more easily than those that are not. Lists placed at the top for navigation, header elements properly providing context to the content and non-use of tables make it possible for the mobile user to read and use the site, often with few changes to code.

The reality is that it can take as little as adding only one line of code on desktop Web page to make it format in a mobile friendly way. Just apply a handheld media type for displaying the mobile stylesheet to the well-formed XHTML markup and the site turns into a site for mobile browsers.

The one line of code that often appears in XHTML documents is the following:

```
<link href="mainstyles.css" rel="stylesheet" type="text/css" />
```

This loads an external file called "mainstyles.css" containing the site's styles and applies it to the XHTML document. It loads the same stylesheet regardless of device type. To specify that only desktop Web browsers should use this stylesheet, just add "media=screen" as shown in the following example:

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

Since you're adding several stylesheets to a document, for clarity, name each stylesheet after the medium it's intended for. Start by renaming "mainstyles.css" to "screen.css" as in the next example:

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

Now add the mobile stylesheet, using the “handheld” media attribute:

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

When specifying different media types as shown in the next example, the browser conditionally loads the stylesheet designed for its medium. Desktop Web browsers, for example, use the screen stylesheet. When the user prints the page, however, the browser switches to the print stylesheet. The real magic happens when you load the same page on a mobile device. Recent Web browsers load the handheld stylesheet, styling the content specifically for the small screen.

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

Using only one line of code, you gain the ability to transform your markup for mobile devices. (Of course, you still have the task of creating the mobile stylesheet.)

Warning

This technique can only do so much. It can’t handle complex design using just CSS and it can’t remove large images using advanced techniques like image replacement.

File sizes of pages designed for bigger screens are significantly larger, so CSS needs to hide extraneous elements like headers and sidebars. While this improves the content’s readability, the user’s device still downloads the hidden content. The user pays for every kilobyte including the hidden content’s bits.

More importantly, this technique doesn't affect the mobile context in terms of providing information that's relevant to the users' physical location or mobility. Nonetheless, this provides an acceptable solution for those with well-coded Web sites who simply want to publish simple content and have it display well on mobile devices.

Part II of this guide will address more sophisticated method of publishing content including adaptation techniques to customize the content to the requesting device.

Glossary

accesskey

An access key allows a browser user to immediately jump to a specific part of a Web page via the keyboard. In most Web browsers, the user does this by pressing ALT (PC) or CTRL (Mac) followed by the appropriate character on the keyboard. On mobile phone this can typically be accomplished by pressing a numeric key directly.

ARPU (Average Revenue Per User)

Term regularly used in the mobile industry to describe the total revenue earned from voice, messaging, data or other activity per user or subscriber.

BREW®

Proprietary mobile device platform developed by Qualcomm. BREW® provides the richest mobile user interface that is widely available. All applications designed for the BREW® platform must pass National Software Testing Labs in order to made available on Carrier decks

First-tier carrier Verizon utilizes the BREW® platform in all their handsets. Several second-tier carriers also use BREW®.

Carrier

North American term used to refer to a mobile telecommunications company or their network (referred to as Operators elsewhere in the world).

cHTML

cHTML is a proprietary markup language based on HTML used by NTT DoCoMo in Japan and other countries. cHTML was the primary markup language of the i-Mode service

Clickstream

Used to refer to the series of clicks, or path, the user takes to reach a destination in an informational space. Often used to describe user behavior gathered from server logs, but also can be used in early planning, as in "creating the optimal clickstream."

Carrier Deck

Refers to the Web presence maintained by each carrier. When you access the Internet from a mobile device, the first page you see is often referred to as the carrier deck, but will also be used to describe all online carrier services that a user interacts with.

Deck Placement

The term used to describe where a third-party vendor WAP site, or application will appear on the Carrier Deck. Default order on content on most Carrier Decks is determined by sales. New items often have temporary "Top-Deck Placement"

The Catch-22 is due to the constrained screen size, the items that show at the top tend to have the highest sales. The items at the bottom see very little user attention.

Device Manufacturer

The term used to describe the manufacturers that make mobile devices, such as Nokia, Motorola, Samsung, etc.

i-Mode

i-Mode is the name given to NTT DoCoMo's Mobile Web service. This service was originally offered in Japan only but has now spread to several other countries.

Information Architecture (IA)

Term used to describe the discipline of creating an informational space. The discipline has its roots in Library Sciences, but has evolved to include many other forms of information. IA is a critical element of mobile design and often a carrier requirement.

J2ME (Java 2 Platform Micro Edition)

J2ME is mobile device platform based on a stripped-down implementation of Java.

Paper Prototype

The process of taking printed or sketched wire-frames and presenting them to a user asking them to perform a series of tasks. The facilitator acts as the "computer" changing the pages as the user makes selections.

Paper Prototyping is an excellent method of doing early usability testing for mobile interfaces.

Premium SMS

Premium SMS is a method of charging mobile-consumers for content delivered via SMS

Premium SMS messages can be mobile originated (MO) or mobile terminated (MT) depending on the server. With MO messages, the consumer is charged on

sending the message, with MT messages the consumer is charged on receipt of the message. In many countries around the world there is a controlled system of numbering so that users can be aware of the cost of PSMS messages before they send them e.g. 7XXXX might indicated a high cost message, 1XXXX may be the minimum band.

RSS (Really Simple Syndication)

A format widely used in the weblog space for syndication of content. RSS has several types of formats, RSS 1.0, RSS 2.0 and all of them are basic XML documents.

RSS strips presentation from content allowing publishers to reformat content for other purposes. Generally used for newsreaders or RSS aggregators, but can also be used to create mobile versions of Web content.

SMS (Short Messaging Service)

Broadcast messaging system that allows for subscriber to subscriber text based communications. Analogous to instant messaging.

Subscribers

Term used by carriers to describe a wireless customer. Subscribers is the correct term to use when referring to users in any communication that could be seen by carriers.

Walled Garden

This is a term used to describe a service (typically a Web portal) that users are either prevented from leaving or encouraged not to leave through obfuscation or financial incentives.

W3C (World Wide Web Consortium)

The W3C develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential. W3C is a forum for information, commerce, communication, and collective understanding.

XHTML (Extensible Hypertext Markup Language)

XHTML is a markup language that has the same depth of expression as HTML, but a stricter syntax. Whereas HTML is an application of SGML, a very flexible markup language, XHTML is an application of XML, a more restrictive subset of SGML.

WAP (Wireless Application Protocol)

Wireless Application Protocol or WAP is a standard for applications that use wireless communication. Its principal application is to enable access to the Internet from a mobile phone or PDA.