Sprint PCS® Mobile Browser *Technology Paper*

Writing Consistent Mobile Browser Content on Sprint PCS Phones

Version 1.0
July 2004

Table of Contents

1 Introduction

WAP 2.0 defines a set of technologies that includes client provisioning, telephony and messaging services, push, synchronization, and wireless protocols, and a markup language. The WAP 2.0 markup language is the subject of this paper.

Sprint uses cellphones from different manufacturers, including Hitachi, LG, Samsung, and Sanyo. Even though all of these cellphones are WAP 2.0 phones, there are some differences in browser behavior that developers must be aware of.

The goal of this paper is to provide developers with a quick reference on how to create WAP 2.0 markup language-based applications that run consistently across all of Sprint micro-browsers. To accomplish this goal, this paper introduces XHTML Basic, XHTML Mobile Profile with examples.

1.1 Target Audience

This paper is targeted at developers, beginners to advanced, wanting to quickly understand how to create consistent browser-based applications across all Sprint phones.

This paper assumes the reader is familiar with WML 1.1 and basic XHTML.

1.2 About this document

This document attempts to summarize XHTML behavior across Sprint micro-browsers. For this, the most commonly used tags are discussed, and examples are provided. But always test your XHTML pages on real devices, as there are differences in behavior not only across micro-browsers, but also between the real device and the emulators. Feel free to try the examples below, which exercise the common XHTML elements.

2 Document Conventions

This paper contains explanations of various XHTML modules and tags. The modules and tag names are denoted by use of a `mono-spaced` typeface in the text, as is source code. File names and directory path names are also denoted with this typeface.

Descriptions of developer tool GUI elements, such as buttons and menus, are presented in **bold** typeface. The user's interactions with these elements are also presented in this typeface.

- Important information and suggestions will be presented as bulleted text and highlighted, as shown here.

3 Overview of Wireless Application Protocol (WAP) 2.0 Markup Language

WAP 2.0 does not really define a new markup language. To understand the meaning of this, lets cover some background information.

During the mid 1990s, there was a company called as Unwired Planet, which is today called Openwave. This company understood the power of mobile devices, and created a toolkit and technology that allowed developers to create browser-based applications for

handheld devices. Part of this technology was the Handheld Device Markup Language (HDML). HDML was successful, but it was a proprietary technology. Other companies such as Nokia, Sony Ericsson and Nippon Telephone and Telegraph (NTT) also had their own proprietary markup languages and technologies.

During the late 1990s, these aforementioned companies, together with AT&T, joined forces in creating an industry standard for browser-based mobile applications. The result was the Wireless Application Protocol (WAP) 1.0, which included Wireless Markup Language (WML).

During the same time, the World Wide Web Consortium (W3C), creators of HTML, was pushing XHTML, a reformulation of HTML in XML, and with this reformulation the idea of XHTML modules. This modularization effort broke XHTML into a set of basic modules, each module addressing different markup language needs and complexities. One of these modules is the XHTML Basic module that defines the most basic set of HTML tags that serve as the base for more complex markup languages such as XHTML 1.0.

Recently the WAP forum adopted XHTML Basic for its WAP 2.0 specification, but with the addition of a number of tags not included in XHTML Basic. The result is what is known as the XHTML Mobile Profile (MP), an XML markup language that provides most of the same capabilities found in WML 1.1. These relationships are illustrated next:

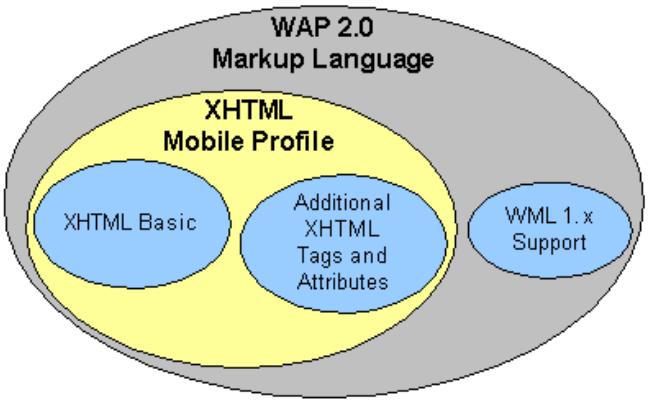


Figure 1. Relationship between WML 1.1, XHTML Basic, and XHTML Mobile Profile

In addition to the adoption of XHTML Mobile Profile, WAP 2.0 also mandates backward compatibility with WML 1.1, either via an external browser, or via WML XML namespace.

Today the WAP forum is part of the Open Mobile Alliance, Ltd.

3.1 XHTML Basic and Mobile Profile

As previously mentioned, WAP 2.0 defines XHTML Mobile Profile, which consists of the tags defined by the XHTML Basic Module with the addition of other XHTML tags and attributes. Table 1 summarizes the XHTML Basic and XHTML Mobile Profile tags:

Table 1. XHTML Mobile Profile Tags and Attributes

XHTML Basic Module	Element
Base	base (specifies a URL against which relative URLs in the same document are resolved. It is contained in the <head> element.
Basic forms	form, input, label, select, option, textarea
Basic tables	caption, table, td, th, tr
Hypertext	a
Image	img
Link	link
List	dl, dt, dd, ol, ul, li
Meta information	meta
Object	object, param
Structure	body, head, html, title
Text	abbr, acronym, address, blockquoate, br, cite, code,dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p, pre, q, samp, span, strong, var
Other XHTML Modules	**Element / Attribute**
Forms (partial)	fieldset, optgroup
Legacy (partial)	start attribute on ol, value attribute on li
Presentation (partial)	b, big, hr, i, small
Style Sheet	style element
Style Attribute	style attribute

*Table source: WAP 2.0 XHTML Specification (WAP-277-XHTMLMP-20011029-a, Version 29-Oct-2001)

3.2 Key Differences between WML 1.x and XHTML

Most of the features found in WML 1.1 are also available in XHTML. This makes migrating from WML 1.1 to XHTML pretty straightforward. The following summarizes some key differences:

- XHTML may contain only one <body> tag per page. There is no concept of card decks in XHTML; card decks were introduced in WAP 1.0 to maximize the use of wireless bandwidth. WML applications with multi-card decks need to be divided into separate XHTML pages and linked together.
- WML uses a binary communication protocol. XHTML is typically transmitted in plain text, unless encrypted.
- XHTML does not support for client side *events*, *variables,* or *scripting*.
- Quick navigation is done using different tags.

The next table summarizes the mapping between *common* WML 1.1 to XHTML tags:

Table 2. WML 1.1 to XHTML Mobile Profile Tag Mapping

WML Markup	XHTML MP Markup To Use
<!– comment -->	Same
<wml> ... </wml>	<html> ... </html>
<head> ... </head>	<head> ... </head>
<card> ... </card>	To migrate, replace WML <card> tags with XHTML <body> tags. If <card> has a class, such as in <card class="class">, use within the <body>. Note that in XHML there can only be a single <body> tag per page. For multi-deck WML applications, you would create one XHTML file per card deck, and link these pages together via hyperlinks.
<a> <b> <big> <form> <i> <img> <p> <small> <table> <tr> <td> <u>	Use *same* element name in XHTML. Note that attributes may or may not be the same. Please refer to WML 1.1 and XHTML documentation.
<do>	Use <input type="button">. For quick navigation in XHTML, use access key instead of using WML <do> tag, which provides similar navigational capabilities.
<anchor>	Use <a id="uniqueID">
<prev/>	N/A – use a link to previous XHTML page.
<refresh/>	Use meta header, for example <meta HTTP-EQUIV="Refresh" CONTENT="5; URL=http://www.xyz.com/abc.html">
<noop/>	N/A
<go/>	N/A – use a link to specified XHTML page.
Variables	Support for global variables and data sharing can be accomplished via server-side scripting.

4 Overview Of Sprint WAP 2.0 Phones and Browsers

The next table summarizes the Sprint PCS Phones by device manufacturer. In particular, notice that Sprint supports three major browsers: Openwave, Teleca (AU), and Netfront.

Table 3. Sprint Phones Browsers

Device Manufacturer	Browser
LG	Openwave 6.1
Samsung	Teleca (AU) (MIC 1.1.4)
Hitachi	Openwave 6.1
Sanyo	Access Netfront 3.0
Nokia	Openwave 6.2.2
Audiovox	Openwave 6.3 or Microsoft IE

The following summarizes the general rendering behavior per browser for the most commonly used XHTML element tags:

- **Text elements**
  - `<abbr>, <acronym>, <address>, <br>, <code>, <div>, <em>, <h1>, <h2>, <h3>, <h4>, <h5>, <h6>, <p>, <pre>, <q>, <span>, <strong>, <b>, <big>, <hr>, <i>, <small>.`
  - The Teleca (AU) browser inserts extra line breaks after the `<h1-h6>` elements.
  - Other inconsistencies are related to use of different font types and text indentation across the different browsers.

- **List elements**
  - `<ol>, <ul>, <li>.`
  - The Netfront 3.0 browser renders correctly all the tested list types.
  - The Teleca (AU) browser does not support: 1) roman-numerals (letters are displayed instead), and 2) images as list item decorations.
  - The Openwave 6.1 browser does not support roman-numerals (decimal numbers are displayed instead) for list item decorations.

- **Table elements**
  - `caption, <table>, <td>, <th>, <tr>.`
  - The Netfront 3.0 browser does a very good job with tables.
  - The Teleca (AU) and Openwave 6.1 browsers properly display the table tags, but the table cell rendering is not as neatly done as in the Netfront browser.

- **Definition List elements**
  - `<dl>, <dt>, <dd>`
  - All browsers consistently rendered these tags.

- **Input elements**
  - `<form>, <input>, <label>, <select>, <option>, <textarea>.`
  - The rendering of the input elements varies across browsers.

- Simple pull-down menu (`select/option`) behaves consistently across all three browsers.
- Grouped options (`select/optgroup`) behave differently in the Netfront 3.0 browser, which does not group input options, as the Openwave 6.1 and Teleca (AU) browsers do.
- Simple text input (`<input type="text">`) behaves consistently across browsers.
- Multi-line text input (`<textarea>`) behaves as expected on Netfront 3.0 and Openwave 6.1, but not in Teleca (AU), which displayed only a single text box for text entry.
- Radio buttons and checkboxes behave consistently across browsers.
- The Openwave 6.1 browser was the only one that supported images in the submit button (`<input type="submit">`).

- **Image element**
  - `<img>`.
  - All tested browsers support GIF and JPG images.

- **Styles**
  - All common styles work consistently across browsers.

In summary, of all the test browsers, the Netfront 3.0 is the most well behaved browser.

5 Writing Consistent WAP 2.0 Applications Across Sprint PCS Phones

Writing consistent applications is not that difficult, as long as you stick to the XHTML MP tags, and you keep in consideration the rendering peculiarities associated with some of the tags. Even though some of the tags may not render exactly across different browsers, the rendering is close enough.

5.1 Commonly used XHTML Mobile Profile Tags

The next table highlights the most frequently used XHTML tags:

Table 4. XHTML Mobile Profile Commonly Used Tags

XHTML Basic Module	Element
Base	`base`
Basic forms	`form, input, label, select, option, textarea`
Basic tables	`caption, table, td, th, tr`
Hypertext	`a`
Image	`img`
Link	`link`
List	`dl, dt, dd, ol, ul, li`
Meta information	`meta`
Object	`object, param`
Structure	`body, head, html, title`

Text	abbr, acronym, address, blockquote, br, cite, code,dfn, div, em, h1, h2, h3, h4, h5, h6, kbd, p, pre, q, samp, span, strong, var
Other XHTML Modules	**Element / Attribute**
Forms (partial)	fieldset, optgroup
Legacy (partial)	start attribute on ol, value attribute on li
Presentation (partial)	b, big, hr, i, small
Style Sheet	style element
Style Attribute	style attribute

5.2 Tags to Avoid and Other Considerations

There are not many tags to avoid. Writing consistent applications is more about understanding the limitations associated with some of the XHTML elements with respect to how these are rendered across different browsers:

- Avoid using proprietary tags such as Openwave XHTML extensions.
- Use of tables: because of limited screen size, tables may not render consistently across browsers; tables may truncate or wrap around. An alternative to tables is using *definition lists* (<dl>, <dt>, <dd>) for minimal layout control, as demonstrated in the tables section. If you must use tables, try not to use more than two columns, and use small fonts to ensure all the text fits well without wrapping.
- Use of images: images provide a rich user interface. But not all browsers support all the different types of image formats (PNG, JPEG, GIF, animated GIF, BMP). PNG, JPEG and GIF are pretty standard. Animated GIF are not. Because of limited screen size, use images that fit on all Sprint PCS Phones; refer to table 2 for a summary of wireless phone characteristics. PNG is the preferred image format to use.
- Text formatting: <abbr>, <addr>, <acronym>, <blockquote>, <cite>, , <pre>, <q>, and even <p> may not render consistently across browsers. For example, some browsers render some of the above elements using a bold font, while other may use italic.
- Use of lists: while the default list (numeric) type renders decorations consistently across browsers, non-default list type, such as alphabetic, roman number, or image types do not. Use the default list type to ensure consistency.
- Some browsers do not wrap fields. To maintain flow consistency, add a line break
 between <label> and input element that follows it, as well as between field elements.
- Do not design your pages to a specific screen size since screen characteristics vary from handset to handset.

5.3 Cascading Style Sheets

Style sheets define rules for how content should be rendered by the browser. Style sheet information can be defined externally, internally or inline. Most of the CSS properties are optional.

- External style sheets are the preferred way to associate styles to a document, as it provides for a single, central point to modify the look and feel of your whole application.

Style sheets are very powerful and convenient, but complex style sheets may require excessive processing, affecting the application's performance. The following summarizes the style sheet properties by category:

- Color and Background style properties: to set the foreground and background color, background image, and other background properties.
- Font style properties: to set the font family, style, size and other font properties.
- List style properties: to set the list type, position, image, and other list properties.
- Text style properties: to set the text indentation, alignment, decoration, and other text properties.
- Other style properties that aren't used as frequently, but are mentioned here for completeness: Padding, Border, Margin, Visual Formatting
- Please refer to the following for more information: XHML Mobile Profile and CSS Reference

The style properties that are used the most, such as the ones related to color, font and text styles, all behave consistently across browsers.

For more information on XHTML Mobile Profile and CSS please refer to the reference documentation in the Sprint developer's web site.

5.4 Use a Consistent Page Layout

Consistent page behavior is as important as choosing the right markup language tags to use. A recommended layout follows. This layout breaks the page into four main (`<div>`) sections: 1) header, 2) highlights or summary, 3) content, and 4) navigation:

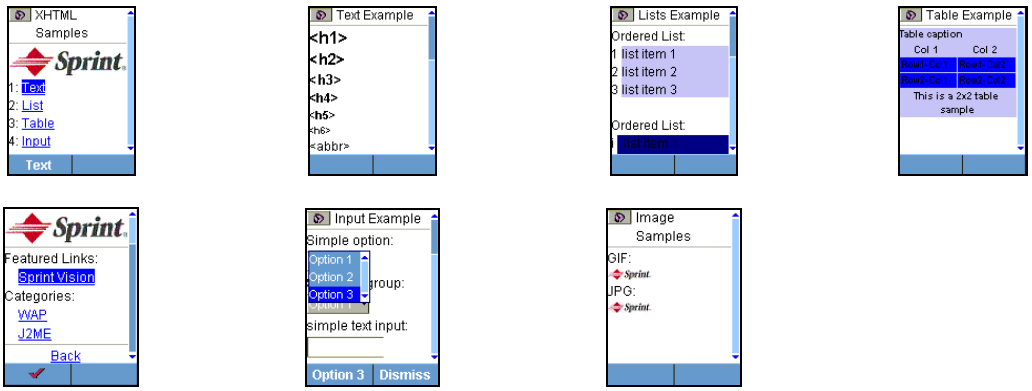
Header
Highlights
Content
Navigation Links (i.e. Back)

The above recommended layout is illustrated in section 6.5 Using Definition Lists.

6 Using WAP 2.0 Consistently Across Sprint PCS Phones

This section summarizes how to use XHTML to write consistent applications. For this, the following XHTML sample application is covered:

Table 5. Sample Application XHTML Pages.



All of the source for the sample application can be found in the archive file `SprintXHTMLSampleApp.zip`.

6.1 Using Hypertext and Image Elements

Not all browsers support all the images formats (PNG, JPG, BMP, GIF, and animated GIF). JPEG and GIF are pretty standard. Animated GIF are not. Also the use of images can degrade performance, as image download can be a time-consuming operation.

To ensure consistent image rendering behavior use GIF or JPG formats. Also ensure the images are of proper size. Please refer to Table 2, which summarizes Sprint PCS Phone characteristics. The following XHTML file (`index.html`) displays an image at the top of the page, followed by a list of hyperlinks that uses rapid navigation using the `<accesskey>` that allows the user to press a key to select a specific list item:

Listing 1. Using Hyperlinks and Images

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
"http://www.wapforum.org/DTD/xhtml-mobile10.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>XHTML Samples</title>
    <link href="style.css" rel="stylesheet" type="text/css" />
    <!--meta http-equiv="refresh" content="30" /-->
  </head>
  <body>
    <p>
      <img src="sprint.gif" alt="Logo" /> <br/>
      1: <a accesskey="1" href="txt.html" title="Text">Text</a>
      <br>
      2: <a accesskey="2" href="lst.html" title="List">List</a>
      <br>
      3: <a accesskey="3" href="tbl.html" title="Table">Table</a>
```

```
      <br>
      4: <a accesskey="4" href="input.html"
         title="Input">Input</a>
      <br>
    </p>
  </body>
</html>
```

The result is the following:

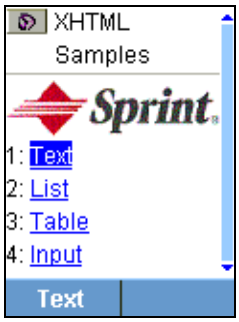


Figure 2. Using the Hypertext and Image Elements

The following XHTML snippet shows how to the use of both GIF and JPG image files:

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

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

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Image Samples</title>
    <link href="style.css" rel="stylesheet" type="text/css" />
    <!--meta http-equiv="refresh" content="30" /-->
  </head>
  <body>
    <p>
      GIF: <br/>
      <img src="sprintsm.gif" alt="GIF Logo" /> <br/>
      JPG: <br/>
      <img src="sprintsm.jpg" alt="JPG Logo" /> <br/>
    </p>
  </body>
</html>
```

The result is the following:

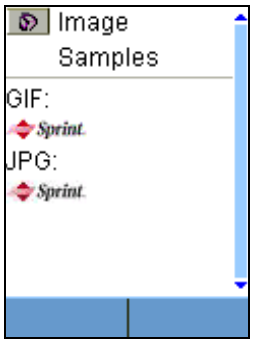


Figure 3. Using the JPEG and GIF Image Elements

6.2 Using Styles

As previously mentioned, style sheet information can be defined externally, internally or inline. External style sheets are the preferred way to associate styles to a document, as it provides with a single, central point to modify the look and feel of your application.

Externally defined style sheets are defined using the `<link>` element:

Listing 2. Defining an External Style Sheet

```
<head>
  <title>Table Example</title>
  <link href="style.css" rel="stylesheet" type="text/css" />
</head>
```

The following is the simple `style.css` style sheet used in our examples:

```
.nowrap {white-space:nowrap}
.centered {text-align:center}
.centered-small {text-align:center;font-size:x-small}
.smalltext {font-size:x-small;foreground-color:white}
.bluegray {background-color:#C0C0F0}
.navybackground  {background-color:navy}
.bluebackground {background-color:blue}
```

6.3 Using Text Formatting

The following elements behave consistently across browsers (except for minor inconsistencies such as extra line break is some of the browsers):

- <h1>
- <h2>
- <h3>
- <h4>
- <h5>
- <h6>
-
- <code>
-
- <b>
- <big>
- <i>
- <small>
- <hr>

The above are the most commonly used text formatting elements, and using these guarantees consistent rendering across devices. The following elements don't behave consistently across browsers:

- <abbr>
- <acronym>
- <address>
- <blockquote>
- <cite>
-
- <pre>
- <q>
- <var>

Inconsistency includes font style, size and/or indentation. The following code snippet highlights the use of the text-formatting elements:

Listing 3. Using the Text Formatting Elements

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

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

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
    <head>
        <title>Text Example</title>
        <link href="style.css" rel="stylesheet" type="text/css"
/>
```

```
    </head>
    <body>
        <div>
            <h1>&lt;h1&gt;</h1>
            <h2>&lt;h2&gt;</h2>
            <h3>&lt;h3&gt;</h3>
            <h4>&lt;h4&gt;</h4>
            <h5>&lt;h5&gt;</h5>
            <h6>&lt;h6&gt;</h6>

            <abbr>&lt;abbr&gt;</abbr> <br/>
            <acronym>&lt;acronym&gt;</acronym> <br/>
            <address>&lt;address&gt;</address> <br/>
            <blockquote>&lt;blockquote&gt;</blockquote> <br/>
            <cite>&lt;cite&gt;</cite> <br/>
            <code>&lt;code&gt;</code> <br/>
            <em>&lt;em&gt;</em> <br/>
            <pre>&lt;pre (preserve)&gt;</pre> <br/>
            <q>&lt;q (quotation)&gt;</q> <br/>
            <strong>&lt;strong&gt;</strong> <br/>
            <var>&lt;var&gt;</var> <br/>

            <b>&lt;b&gt;</b> <br/>
            <big>&lt;big&gt;</big> <br/>
            <i>&lt;i&gt;</i> <br/>
            <small>&lt;small&gt;</small> <br/>
        </div>
        <hr/>
        <p>
            Paragraph block. <span class="smalltext">Span block
within the paragraph block.</span>
        </p>
    </body>
</html>
```

The result is the following:

Figure 4. Using the Text Formatting Elements

6.4 Using Tables Consistently

Tables provide a convenient way to arrange information. But tables require special attention. If you use too many columns, or large fonts, the table becomes hard to read, as the browser may wrap the table contents. In fact, the use of tables is discouraged in favor of definitions lists (see next section), which provides a simple but effective way to layout your markup language elements.

The following XHTML file (`tbl.html`) renders a table. To minimize wrapping around, the table doesn't use more than 2 columns, and uses small fonts, which are set via an external style sheet:

Listing 4. Using the Table Elements

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
"http://www.wapforum.org/DTD/xhtml-mobile10.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
  <head>
    <title>Table Example</title>
    <link href="style.css" rel="stylesheet" type="text/css" />
  </head>
  <body>
    <div class="bluegray">
      <table class="smalltext">
        <caption align="left">Table caption</caption>
        <tr>
            <th> Col 1 </th>
            <th> Col 2 </th>
        </tr>
        <!-- Define first row -->
        <tr>
          <td class="bluebackground">
            Row1-Col1
```

```
            </td>
            <td class="bluebackground">
              Row1-Col2
            </td>
          </tr>
          <!-- Define second row -->
          <tr>
            <td class="bluebackground">
              Row2-Col1
            </td>
            <td class="bluebackground">
              Row2-Col2
            </td>
          </tr>
          <tr> <th colspan=2>This is a 2x2 table sample</th> </tr>
        </table>
      </div>
    </body>
</html>
```

The result is the following:

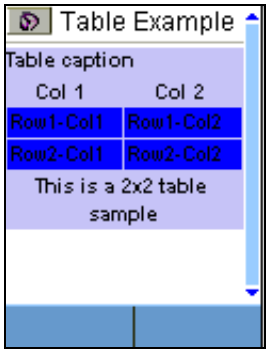


Figure 5. Using the Table Elements

6.5 Using Definition Lists

As previously mentioned, the use of tables for page layout is discouraged because of inconsistent table rendering across devices. This inconsistency is typically due to varying screen sizes across devices. Instead you can use definition lists within divisions (`<div>`) which behave consistently across browsers. Definition list allows you to indent content as well:

Source	Output
<dl> <dt>Featured Links:</dt> <dd>Link 1</dd> <dt>Categories:</dt> <dd>Category 1</dd> </dl>	Featured Links: Link 1 Categories: Category 1

The following XHTML file (`dl.html`) organizes information using a definition list. It also organizes the page by breaking it into four main sections: header, summary, content, and navigation:

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

<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile
1.0//EN"

"http://www.wapforum.org/DTD/xhtml-mobile10.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">

  <head>

    <title>Def List Example</title>

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

  </head>

  <body>

      <div>

         <img src="sprint.gif" alt="GIF Logo" /> <br/>

         <hr/>

         <dl>

           <dt>Featured Links:</dt>

            <dd><a
href="http://vision.sprintpcs.com">Sprint Vision</a></dd>

         </dl>

      </div>

      <div>

         <dl>

           <dt>Categories:</dt>

            <dd><a
href="http://vision.sprintpcs.com">WAP</a></dd>
```

```
            <dd><a
href="http://vision.sprintpcs.com">J2ME</a></dd>
          </dl>
      </div>
      <hr/>
      <div align="center">
        <a href="./index.html">Back</a>
      </div>
  </body>
</html>
```

The result is the following:

Figure 6. Using the Definition List Elements

The method and content organization illustrated in this section is the recommended way to write an XHTML mobile content page.

6.6 Using Lists Consistently

Lists are very common in mobile applications. Lists behavior is consistent across browsers except for the list type, which allows the developer to label or decorate list items using numbers, letters, roman literals, images, circles, squares and so on. While the default list (numeric) type renders consistently across browsers, non-default list type, such as alphabetic, roman number, or image types do not, in which case requested type is ignored and default behavior is displayed. Using the default behavior guarantees consistent rendering across browsers. The following XHTML file (`lst.html`) renders different types of lists:

Listing 5. Using the List Elements

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
"http://www.wapforum.org/DTD/xhtml-mobile10.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
```

```
<head>
  <title>Lists Example</title>
  <link href="style.css" rel="stylesheet" type="text/css" />
</head>
<body>
  <div>
      Ordered List: <br/>
      <ol class="bluegray">
         <li>list item 1</li>
         <li>list item 2</li>
         <li>list item 3</li>
      </ol>
      <br/>
      Ordered List: <br/>
      <ol type="i" class="navybackground">
         <li>list item 1</li>
         <li>list item 2</li>
         <li>list item 3</li>
      </ol>

      <br/>
      Ordered List: <br/>
      <ol type="a" class="navybackground">
         <li>list item 1</li>
         <li>list item 2</li>
         <li>list item 3</li>
      </ol>
      <br/>
      UnOrdered List: <br/>
      <ul class="bluebackground">
         <li>list item 1</li>
         <li>list item 2</li>
         <li>list item 3</li>
      </ul>
      <br/>
      UnOrdered List: <br/>
      <ul style="list-style-image: url(sprintsm.gif)" type="a"
         class="bluegray">
         <li>list item 1</li>
         <li>list item 2</li>
         <li>list item 3</li>
      </ul>
```

```
    </div>
  </body>
</html>
```

The result is the following:

Figure 7. Using the List Elements

6.7 Using Input Elements Consistently

Input elements allow the user to enter data using text, radio buttons, and checkboxes. Inconsistencies across browsers include:

- `<select>` using `<optgroups>` displays the selection box differently across browsers. Use `<select>` with `<option>` instead.
- Submit button using image: `<input type="submit" src="image">`. Do not use images.
- Use of label <label> not wrapping input fields correctly. Add a line break `<br>` to ensure consistent flow.
- `<textarea>` works OK, but only displays a single text input line.

The following XHTML file (`input.html`) renders the different input elements:

Listing 6 – Using Input Elements

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
"http://www.wapforum.org/DTD/xhtml-mobile10.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
    <title>Input</title>
    <link href="style.css" rel="stylesheet" type="text/css" />
</head>
    <body>
        <div class="nowrap">
            <form action=http://www.host.com/servlet
```

```
method="post">
   <label>Simple option:
       <select name="selection">
           <option value="option1"
             title="Option 1">Option 1</option>
           <option value="option2"
             title="Option 2">Option 2</option>
           <option value="option3"
             title="Option 3"
             selected="selected">Option 3</option>
       </select>
   </label>
   <br/>
   <label>Select optgroup:
       <select name="selection">
           <optgroup label="Group1">
               <option value="option1"
                 title="Option 1">Option 1</option>
               <option value="option2"
                 title="Option 2">Option 2</option>
           </optgroup>

           <optgroup label="Group2">
               <option value="option1"
                 title="Option 1">Option 1</option>
               <option value="option2"
                 title="Option 2">Option 2</option>
           </optgroup>
           <option value="None"
             title="None"> None</option>
       </select>
   </label>
   <br/>
   <label>simple text input:
       <input type="text" name="name" size="8"
        maxlength="8"emptyok="false" title="Edit"/>
   </label>
   <br/>
   <label>textarea input:
       <textarea name="textarea" cols="12" rows="4"
        maxlength="64" title="Textarea">
           Textarea content
```

```
                    </textarea>
                </label>
                <br/>
                Radios buttons: <br/>
                <label>r1 <input type="radio" name="radio"
                value="radio1" title="Radio 1"
                checked="checked" /> </label>
                <label>r2 <input type="radio" name="radio"
                value="radio2" title="Radio 2"/> </label>
                <br/>
                Checkboxes: <br/>
                <label>c1 <input type="checkbox" name="checkbox"
                value="check1" title="Check 1"/> </label>
                <label>c2 <input type="checkbox" name="checkbox"
                value="check2" title="Check 2"/> </label>
                <br/>
                <label>Submit: <input type="submit"
                value="submit" title="Submit"
                src="sprintsm.jpg"/></label>
                <label>Reset:  <input type="reset" value="clear"
                title="Clear"/> </label>
            </form>
        </div>
    </body>
</html>
```

The result is the following:

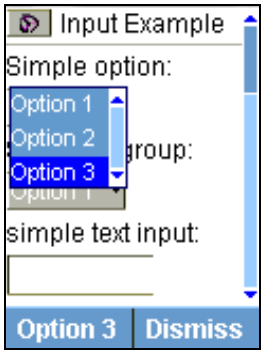


Figure 8. Using the Input Elements

7 Testing Tips

The following tips will help you during your testing phase:

- Always test your content first using the emulator. (Note that the Teleca AU browser does not have an emulator.) However, *always* perform final testing on the

target device.

- To avoid retyping your server test URL, bookmark the URL. Also, some browsers keep a URL history, which you can also use to quickly return to a previously used URL.
- Because the browser caches pages, it may not download recently changed XHTML pages from the server. To workaround this, force a refresh by going to the browser main menu, and select `refresh`. Or if you have access to your development web server, you can configure it to force immediate expiration of your pages. For example, in Apache web server make sure the `mod_expires` module is available. Then have an `.htaccess` file with the following:

  ```
  ExpiresActive On
  ExpiresByType text/html A0
  ```

  Make sure to reset your server configuration once you are done with your testing, as browser caching is very important to minimize page rendering latency.
- Please refer to the Sprint Developer's Web site where you can find useful resources such as XHTML style guides.

7.1 XHTML Validation

All web pages using Sprint Code Standards must validate as XHTML Transitional 1.0. The differences between HTML 4 and XHTML are few, the key difference being responsible code. Browsers have historically been forgiving of lax coding habits, such as not closing the `<p>` element.

Use the following steps as a checklist and your pages will validate as XHTML Transitional 1.0:

- All documents must have an XHTML `DOCTYPE` declaration. A Document Type Declaration is used to declare which version of HTML the webpage conforms to. Browsers may not render web page correctly if a proper `DOCTYPE` is not specified. The following must be at top of each page:

  ```
  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
  Transitional//EN"

  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
  transitional.dtd">

  <html xmlns="http://www.w3.org/1999/xhtml"
  xml:lang="en" lang="en">
  ```

- Content-type must be specified in `<head>`:

  ```
  <meta http-equiv="Content-Type" content="text/html;
  charset=iso-8859-1" />
  ```

- All tags and attributes must be lowercase:

  ```
  <BODY ID="sectionII"> must be <body id="sectionII">
  ```

- Attributes must have a value:

  ```
  <input type="checkbox" checked> becomes <input
  type="checkbox" checked="checked" />
  ```

- Attribute-value pairs must be contained within quotes. Values are case-sensitive, but are not required to be all lowercase as are tags and attributes:

  `<body id=sectionII>` becomes `<body id="sectionII">`
- All tags must close properly:

  `<p> must be closed by a matching </p>`
- Minimized tags must contain trailing space and a slash:

  `<img src="foo.gif">` becomes `<img src="foo.gif" />`

  `<br>` becomes `<br/>`
- No improper nesting of elements:

  `<span><strong>Text</span></strong>` becomes
  `<span><strong>Text</strong></span>`
- Properly encode non-ASCII, reserved or unsafe characters in the source code and in the URL encoding. For example:
  `&` is `%26`, < is `%3C`, > is `%3E`, and = is `%3D`
- The alt attribute in `<img>` tag is mandatory:

  `<img src="back.gif" alt="Back" />`

  If no `alt` value is relevant use `<img src="back.gif" alt="" />` but avoid this practice as much as possible per Accessibility Standards.
- To validate your page use *W3C HTML Validator* (http://validator.w3.org/check). Another option is to download a Windows-only program that will validate pages locally, for example you can use *A Real Validator* (http://arealvalidator.com/).

8 References

- Sprint developer's web site
  - Sprint PCS Vision [SM] Style Guide for XHTML Basic
  - Usability Requirements for XHTML Basic
- XHTML Basic and XHTML Modularization Overview
- XHTML Mobile Profile and CSS Reference