Device Independence and the Web[1]

Mark Butler, Fabio Giannetti, Roger Gimson, Tony Wiley
Information Infrastructure Laboratory
HP Laboratories Bristol
HPL-2002-249
October 2nd , 2002*

The Web is steadily increasing its reach beyond the desktop to devices ranging from mobile phones to domestic appliances. This rapidly expanding accessibility is largely due to the Web's foundation in open protocols and markup languages, which offer the most widely implemented global infrastructure for content and application access. HTML's original aim was to provide a device-independent markup language that was based on document semantics. It identified document elements such as headings, paragraphs, and lists without specifying presentation. Early on, however, browser developers introduced many ad hoc presentation-specific elements and attributes to HTML that blurred the distinction between semantics and presentation. A presentation created for a large-screen device, for example, can theoretically be displayed and interacted with on a small-screen device because they use the same markup. Practically, however, it might be too hard to use simply because the author did not create the presentation with a small form-factor in mind. Device independence is an attempt to regain some of Web publications' original intent. Web standards are now encouraging a renewed distinction between semantics and presentation through styling languages such as Cascading Style Sheets (CSS) and Extensible Stylesheet Language Formatting Objects (XSL-FO) for adding information to Web output, and interaction markup languages such as XML Forms (XForms) for input. As the number of devices accessing the Internet increases, the problem of creating presentations for each device type grows worse. Ideally, authors would need to create only one version of their Web content. Then, during the delivery and rendering process, adaptation software would create a presentation to match the delivery device's capabilities.

* Internal Accession Date Only
Approved for External Publication
IEEE Internet Computing – September/October 2002 – pp 81-86

Device Independence and the Web[1]

Mark Butler, Fabio Giannetti, Roger Gimson, and Tony Wiley
HP Labs, Bristol

HPL-2002-249

The Web is steadily increasing its reach beyond the desktop to devices ranging from mobile phones to domestic appliances. This rapidly expanding accessibility is largely due to the Web's foundation in open protocols and markup languages, which offer the most widely implemented global infrastructure for content and application access.

HTML's original aim was to provide a device-independent markup language that was based on document semantics. It identified document elements such as headings, paragraphs, and lists without specifying presentation. Early on, however, browser developers introduced many ad hoc presentation-specific elements and attributes to HTML that blurred the distinction between semantics and presentation. A presentation created for a large-screen device, for example, can theoretically be displayed and interacted with on a small-screen device because they use the same markup. Practically, however, it might be too hard to use simply because the author did not create the presentation with a small form-factor in mind.

Device independence is an attempt to regain some of Web publications' original intent. Web standards are now encouraging a renewed distinction between semantics and presentation through styling languages such as Cascading Style Sheets (CSS) and Extensible Stylesheet Language Formatting Objects (XSL-FO) for adding information to Web output, and interaction markup languages such as XML Forms (XForms) for input.

As the number of devices accessing the Internet increases, the problem of creating presentations for each device type grows worse. Ideally, authors would need to create only one version of their Web content. Then, during the delivery and rendering process, adaptation software would create a presentation to match the delivery device's capabilities.

---

[1] **IEEE Internet Computing – September/October 2002 – pp 81-86.**

These are the key questions:

- How can we express a Web application independent of the delivery device?
- How can we adapt device-independent applications to suit delivery device capabilities?
- How can authors retain some control over the final presentation of their content?

Existing Web technologies can provide partial answers to these questions, and researchers and practitioners are now working on technologies to provide fuller solutions. Here, we offer an overview of both.

Why device independence?

Currently, authors typically design Web applications for the PC browser and screen. To exert maximum control over the final appearance, they often base Web page layouts on tables, specified using absolute pixel positioning. Because adapting such a site for a small display is effectively impossible, authors must create a parallel site to accommodate these devices. Figure 1 shows similar content adapted for different devices. As the variety of Web-connected devices increases, however, creating a separate site for each kind of device is both economically and administratively impractical.

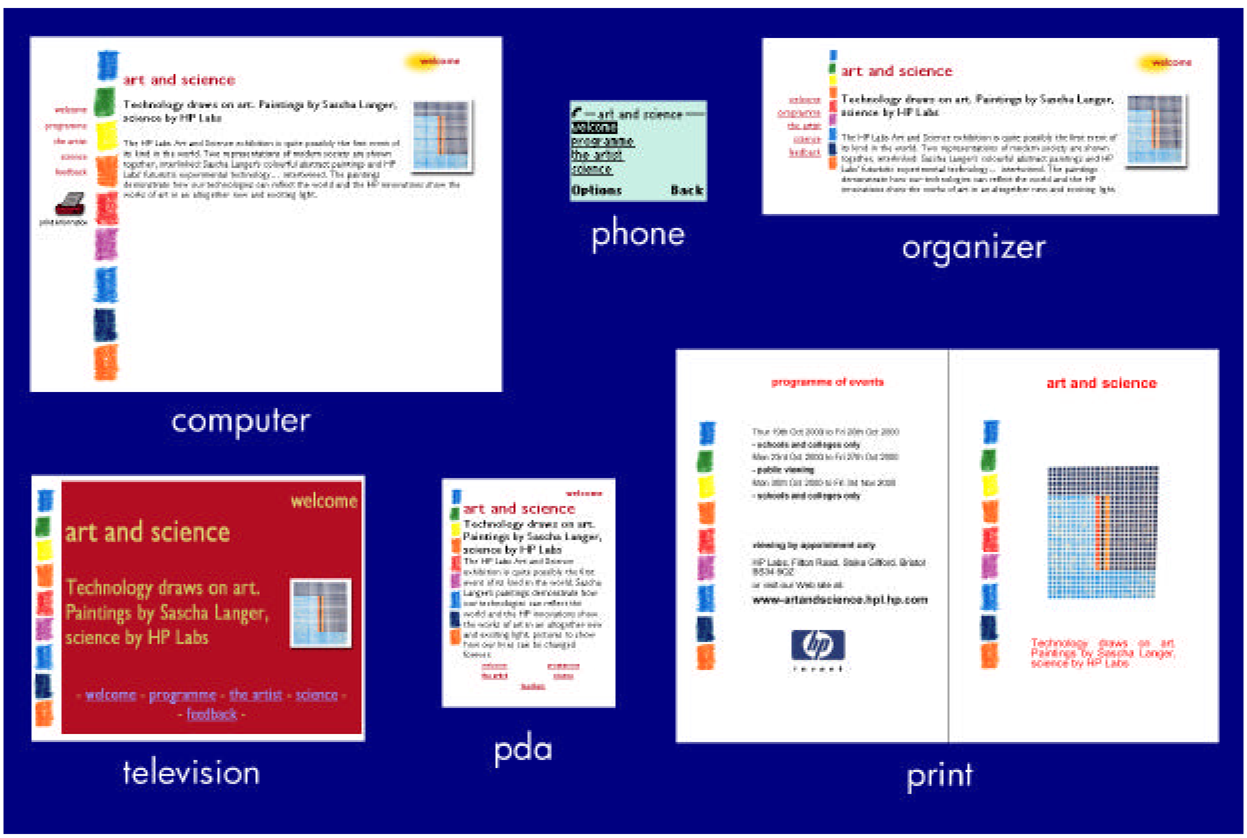

Figure 1: Content adapted for different devices. Although device display capabilities overlap in some cases, other factors, such as text entry capabilities, can vary widely.

The device-independent approach to design lets authors support many different devices without the high overhead. In fact, it's often unnecessary for authors to create completely different content for each device. The display capabilities of phones and PDAs, for example, overlap in some cases — though their text entry capabilities can differ widely. The key concept in achieving greater device independence is to match the rendering of application content and interaction to device capabilities.

In addition to easing author overhead, device independence offers users several benefits. Accessibility, for example, is a fundamental concern, and in some countries a legal necessity. Users must be able to interact with the Web in ways that suit their abilities. Offering options that let users replace images with text, present text as speech, or interact using voice or special input devices can benefit a wide range of users. Different circum-stances might also alter the way users want to interact. A user in a car, for example, might switch from visual to audio-only interaction while driving. By making content adaptability a fundamental design aspect, Web site designers can both increase user options and better accommodate presentation preferences.

Device-Independence Technologies

Currently, designers are using many different approaches to achieve better device independence for Web presentations. The existing approaches fall into three broad categories — intermediate, client-side, and server-side — depending on who controls the adaptation process.

Intermediate adaptation

To avoid changing either the server that provides content or the client that consumes it, intermediaries in the content delivery chain can offer limited adaptation. Transcoding proxies, for example, can transform image formats or even subsets of markup languages. This gives data-enabled phones access to Web sites by either omitting a server's full-resolution color images or transcoding them into low-resolution or monochrome versions, depending on the phone's display capabilities.

Intermediaries typically lack special information about content, and thus their adaptation abilities are limited. This is usually no problem for individual images, which typically include resolution and size metadata. Because authors are increasingly marking up content with presentation rather than semantic information, however, it's much harder to transcode anything but trivial markup and still get an acceptable result. Transcoding HTML into Wireless Markup Language (WML), Compact HTML (cHTML), or Handheld Device Markup Language (HDML) for phones is only possible for some constructs. Literally transcoding Web pages that use "hidden semantics" — such as tables to control layout or text embedded within images — typically produces unusable results.

Intermediate adaptation can help reduce origin server loads, but it is only fully successful when it's based on both knowledge of target device capabilities and author-provided metadata and adaptation hints.

Client-side adaptation

Adaptation can also occur in the content delivery device (typically the Web browser). The advantage here is that the adaptation code usually has direct access to the device's capabilities.

Some client-side adaptations are independent of content directives. Many browsers, for example, let users increase or decrease text display size. However, this can have unexpected consequences for author-defined layouts. Client-side adaptations can also occur based on directives within the content. An example of such author-controlled client-side adaptation is the use of CSS, which authors often use to style HTML elements. CSS is equally applicable, in browsers that support it, for styling Extensible Hypertext Markup Language (XHTML), Scalable Vector Graphics (SVG), or even plain XML content. Separating style from content is accepted good practice and lets authors provide different styles to suit different devices.

CSS media types (not to be confused with Internet MIME media types) are names that identify different device types, such as screen, handheld, TV, print, projection, aural, and Braille. In CSS, authors can define different styling rules for different media types. Figure 2 shows an example style sheet, with variable attributes such as foreground and background colors, and text fonts that let the browser adjust the presentation for conventional screens, printers, small handheld devices, televisions, or projection displays.

On smaller devices, authors can use CSS media types to omit display of parts of a Web page. Of course, delivering a complete page suitable for any device, then omitting large parts of it, is not an effective use of delivery bandwidth.

```
/* media-dependent foreground/background colours */
@media screen, print, handheld
  { body { color: black; background-color: white } }
@media projection, tv
  { body { color: white; background-color: blue } }

/* media-dependent font family */
@media screen, handheld, tv, projection
  { body { font-family: helvetica,sans-serif } }
@media print
  { body { font-family: times,serif } }

/* media-dependent text appearance */
@media screen, print
  { body { font-size: 12pt } }
@media handheld
  { body { font-size: 10pt; line-height: 80% } }
@media projection, tv
  { body { font-size: 16pt; font-weight: bold } }

/* increase width of borders on tv to avoid flicker */
@media tv
  { body { border-width: thick } }
```

Figure 2: CSS style sheet with device-dependent styles. By adjusting display colors and text fonts, the presentation is suitable for a variety of devices, including PCs, printers, handhelds, and projection displays.

```
<?xml version="1.0"?>
<xsl:stylesheet
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:param name="accept"/>
  <xsl:param name="deli-capabilities"/>

  <xsl:template match="/">
    <xsl:if test="contains($accept,'wml')">
      <xsl:call-template name="wmldevice"/>
    </xsl:if>
    <xsl:if test="not(contains($accept,'wml'))">
      <xsl:call-template name="htmldevice"/>
    </xsl:if>
  </xsl:template>

  <xsl:template name="htmldevice">
    <html>
      <head>
        <title>Example HTML page</title>
      </head>
      <body>
         <xsl:call-template name="capabilityTest"/>
      </body>
    </html>
  </xsl:template>

  <xsl:template name="wmldevice">
    <wml>
      <card title="Example WML card">
        <xsl:call-template name="capabilityTest"/>
      </card>
    </wml>
  </xsl:template>

  <xsl:template name="capabilityTest">
    <p>
      Device capability test supported by DELI in Cocoon.
      <xsl:if test="$deli-capabilities/browser/ColorCapable">
        This device can display colors.
      </xsl:if>
    </p>
  </xsl:template>

</xsl:stylesheet>
```

Figure 3: Extensible Stylesheet Language Transformation (XSLT) style sheet. In this example, the style sheet uses HP's Delivery Context Library (Deli) capability handling for Cocoon.

Server-side adaptation

Server-side adaptation offers maximum author control over the delivered content, including the ability to radically change the content amount and its styling, navigation, and layout. In order to produce the most appropriate adaptation, however, the server must have sufficient information about the delivery context, including the delivery device's capabilities.

There are many techniques for achieving server-side adaptation. Web site designers have been delivering different versions of their content for years to accommodate nonstandard browser implementations. The author might provide multiple versions of the content and let the server select which to deliver; more typically, the author will include explicit variants within server-side coding, such as within Java Server Pages.

A standards-based approach — supported, for example, by the Apache Cocoon server — uses an Extensible Stylesheet Language Transformation (XSLT) to generate appropriate delivery markup from a common XML content representation. Figure 3 shows an example, outlining a transformation using different templates, depending on whether the target device accepts WML markup or the default HTML. The author might provide several XSL style sheets to generate markup suitable for different delivery devices; the server then selects the appropriate style sheet based on the delivery context.

Delivery Context

Wherever adaptation takes place, it must be based on information about the delivery context. This can include the delivery device's capabilities, the delivery network's characteristics, user preferences, and other optional application-specific parameters such as users' preferred language or their location.

Some of this information is available in the Web page request's standard HTTP protocol header (see Figure 4). Currently, there are no standards for expressing arbitrary device capabilities in the

```
User-Agent: Mozilla/2.0 (compatible; MSIE 3.02; Windows CE; PPC; 240x320)
UA-OS: Windows CE (POCKET PC) - Version 3.0
Accept-Encoding: gzip, deflate
UA-CPU: ARM SA1110
UA-pixels: 240x320
```

(a)

```
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.0; en-GB; rv:0.9.4.1)
Gecko/20020508 Netscape6/6.2.3
Accept: text/xml, application/xml, application/xhtml+xml, text/html;q=0.9,
image/png, image/jpeg, image/gif;q=0.2, text/plain;q=0.8, text/css,
*/*;q=0.1
Accept-Encoding: gzip, deflate, compress;q=0.9
Accept-Language: en-gb
```

(b)

Figure 4: Example HTTP request headers.(a) A PDA's HTTP request header from a Pocket IE 2002 browser; (b)A PC's HTTP request header from a Netscape 6.2.3 browser.

header, though servers can use existing headers to try guessing the client's nature. The Accept header field, for example, identifies the response's acceptable representations (MIME types), while the User-Agent header field identifies the requester's Web browser.

Composite capabilities/preferences profile (CC/PP) is a specific data format for expressing delivery context information and is the basis for the user agent profile (UAProf), which is part of the Open Mobile Alliance's current mobile phone standard. Figure 5 shows an example of a UAProf.

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:prf="http://www.wapforum.org/profiles/UAPROF/ccppschema-20010430#">
  <rdf:Description rdf:ID="ExamplePhone">
    <prf:component>
      <rdf:Description rdf:ID="HardwarePlatform">
        …
        <prf:BitsPerPixel>12</prf:BitsPerPixel>
        <prf:ScreenSize>190x200</prf:ScreenSize>
        <prf:ColorCapable>Yes</prf:ColorCapable>
        <prf:ImageCapable>Yes</prf:ImageCapable>
        <prf:Keyboard>PhoneKeypad</prf:Keyboard>
        <prf:Model>Example</prf:Model>
        <prf:StandardFontProportional>Yes</prf:StandardFontProportional>
        <prf:TextInputCapable>Yes</prf:TextInputCapable>
        …
      </rdf:Description>
    </prf:component>
    <prf:component>
      <rdf:Description rdf:ID="SoftwarePlatform">
        …
        <prf:CcppAccept>
          <rdf:Bag>
            <rdf:li>application/vnd.wap.wmlc</rdf:li>
            <rdf:li>application/vnd.wap.wmlscriptc</rdf:li>
            <rdf:li>image/vnd.wap.wbmp</rdf:li>
            <rdf:li>image/jpeg</rdf:li>
            <rdf:li>image/bmp</rdf:li>
            <rdf:li>image/gif</rdf:li>
            <rdf:li>image/png</rdf:li>
          </rdf:Bag>
        </prf:CcppAccept>
        …
      </rdf:Description>
    </prf:component>
  </rdf:Description>
</rdf:RDF>
```

Figure 5: Example user agent profile. UAProf is based on the composite capabilities/preferences profile (CC/PP), which proposes a framework for expressing delivery context information.

CC/PP uses XML namespaces so that different profiles can use different or even multiple vocabularies to describe context information. The data format is based on RDF and represents capabilities as a two-level hierarchy consisting of components and properties. The latest generation of mobile phones is already deploying the data format through UAProf. The W3C plans further work on protocols and vocabularies, and the Java Specification Request 188 Expert Group is defining a server-based application programming interface for CC/PP.

The "CC/PP Tool Support" side-bar gives pointers to tools that are currently available for handling CC/PP and UAProf. This support for conveying richer delivery context information between client and server opens the way to better content adaptation.

Authoring for device independence

Authors can use the technologies we've described to offer device-specific presentations based on common HTML or XML content. However, such technologies do not necessarily reduce the author's burden. Each Web site page, depending on its content, typically has a different layout that might require adaptation to match device capabilities. Authors might thus need a different style sheet for each page layout on each device type. For a large site with many pages — each of which must be delivered to many different devices — the authoring burden is still too great.

CC/PP Tool Support

Public-domain tools are now available to support the composite capabilities/ preferences profile (CC/PP) framework for expressing delivery context information.

- HP Labs' Delivery Context Library (Deli) Toolkit (http://delicon.sourceforge.net/) offers Java-based server-side support for CC/PP and UAProf, as well as conventional HTTP headers. Deli provides a simple API that lets servlet developers use CC/PP information without having to worry about data format, protocol, or processing issues. It also supports legacy devices so that developers can create CC/PP-aware applications today, even when most devices are not yet CC/PP-capable. As Figure 3 in the main article shows, Deli has also been integrated into the XML Stylesheet Language Transformation (XSLT) adaptation environment on the Apache Cocoon server (http://xml.apache.org/cocoon/developing/deli.html).
- Intel's CC/PP Toolkit (www.intel.com/pca/developernetwork) is available as an Intel PCA Developer Network software download and includes a client-side proxy and profile manager written in Java aimed at Pocket PC devices, as well as a server-side profile repository and Apache server module. The profile repository stores profiles in an SQL database, while the Apache module provides a CGI interface to adapt content and support content redirection, which lets the server use the CC/PP profile to select content variants in a similar way to HTTP content negotiation.

The W3C CC/PP Working Group offers a list of other related resources at www.w3.org/Mobile/CCPP.

We can reduce this burden by maximizing reuse of content, navigation, layout templates, and application logic across as many devices as possible. The ultimate goal is "single authoring," in which a sufficiently rich description of all application and presentation aspects is provided to permit automatic adaptation to any delivery device.

Currently, there are several proprietary device-independent markup languages and associated adaptation platforms that offer different approaches to a single-authoring solution. Also, within the W3C, several working groups are addressing aspects of authoring problems.

- The XForms working group has adopted a device-independent approach to defining application interaction through XML-based forms. The standard they are developing separates the semantic aspects of form field definition and submission from the presentation aspects of how the form appears to a user on a particular device.
- The Multimodal Interaction working group is investigating how authors can combine different modalities (visual, speech, and so on) to provide a richer user interface.

Related resources

The World Wide Web Consortium supports several standardization activities relevant to Web content across different devices.

W3C activities

- CC/PP — www.w3.org/Mobile/CCPP
- Device independence — www.w3.org/2001/di
- Multimodal interaction — www.w3.org/2002/mmi
- Style — www.w3.org/Style
- Web accessibility — www.w3.org/WAI
- XForms — www.w3.org/MarkUp/Forms/

Other resources

- Device-independence principles — www.w3.org/TR/di-princ
- Authoring scenarios for device independence — www.w3.org/2001/di/public/as/
- W3C Delivery Context Workshop — www.w3.org/2002/02/DIWS
- W3C Device Independent Authoring Techniques workshop — www.w3.org/2002/07/DIAT/
- CSS media types — www.w3.org/TR/ REC-CSS2/media.html
- Open Mobile Alliance's UAProf specification — www.wapforum.org/what/technical.htm
- Java Community Process JSR-188 specification — www.jcp.org/jsr/detail/188.jsp

- The Device Independence working group covers overall issues related to authoring and adaptation techniques, and is addressing some of the remaining gaps. It has produced working documents on device-independence principles and authoring scenarios, has held a delivery-context workshop, and is about to hold a workshop on device-independent authoring techniques.

Conclusion

Device manufacturers, users, and authors have differing needs and expectations when it comes to Web content. Web software and hardware manufacturers naturally try to differentiate their products by supporting a special combination of capabilities, but few can expect Web authors to create content for their product alone. Users, however, do expect to access the same content from any device with similar capabilities. Even when device capabilities differ, users might still want access to an adapted version of the content. Due to device differences, the adaptation might not produce an identical presentation, but device-independence principles suggest it should be sufficiently functional to let users interact with it successfully. Web application authors cannot afford to create multiple content versions for each of the growing range of device types. Authors would rather create their content once, and adapt it to different devices — but they also want to retain control of presentation quality.

Device independence is about trying to satisfy these differing needs, spanning the delivery path between author and user by way of diverse manufacturers' devices. The field's continued evolution within the broader Web standards framework aims to find solutions that are beneficial for all.

Authors

Mark Butler is a researcher at HP Labs Bristol. He is chair of the W3C CC/PP working group and creator of the DELI CC/PP toolkit.

Fabio Giannetti is a researcher at HP Labs Bristol. He is creator of FOA and WH2FO, authoring tools for XSL Formatting Objects.

Roger Gimson is a senior researcher at HP Labs Bristol. He is chair of the W3C Device Independence working group.

Tony Wiley is a project manager at HP Labs Bristol.

Readers can contact the authors at
{mark-h_butler,fabio_giannetti,roger_gimson,anthony_wiley}@hp.com