Progressive Enhancement in the Real World

John Wells
Underscore Ltd
9 Little Portland Street
London W1W 7JF, UK
+44 (0)20 7631 8400
johnw@underscore.co.uk

Chrisina Draganova
London Metropolitan University
100 Minories
London EC3N 1JY, UK
+44 (0)20 7320 1709
c.draganova@londonmet.ac.uk

ABSTRACT
Progressive Enhancement is a modern approach for developing web documents that are accessible across any browser or device that has access to the Internet. Based on the idea of separating a document's content, presentation, and behaviour, progressive enhancement embraces accessibility, semantics, forward-compatibility and usability. This poster illustrates a real world implementation of the progressive enhancement technique by following the steps in building a restaurant food menu that elegantly scrolls within a fixed page design.

Categories and Subject Descriptors
H.5.4 [**Hypertext/Hypermedia**]: User issues

General Terms
Design, Experimentation, Human Factors.

Keywords
Progressive enhancement, Web design, CSS.

1. INTRODUCTION
This poster illustrates a real world implementation of the progressive enhancement technique by following the steps in building an interactive food menu for the Food Fix web site [1]. The term progressive enhancement was introduced by Steven Champeon [2] in 2003. It provides a strategy for building web documents that are accessible on any browser or device connected to the Internet, based on the idea of separating a document's content, presentation and behaviour. The implementation of the progressive enhancement approach requires developing three layers:

- Content Layer - using semantic XHTML markup, accessible by all devices.
- Presentation Layer - using externally linked CSS, giving a branded look and feel for modern devices.
- Behaviour Layer - using externally linked JavaScript, adding a touch of elegance and interaction for devices capable of running JavaScript.

The benefits of applying the progressive enhancement approach in web design and development include: improved semantics, accessibility and SEO (Search Engine Optimisation), gaining a wider audience, improved performance, reduced costs for maintenance, and easier incorporation of more advanced features and extensions.

HT'07, September 10-12, 2007, Manchester, United Kingdom.
ACM 978-1-59593-820-6/07/0009.

2. LAYER 1: CONTENT
This layer contains the basic markup of the food menu, free of any embedded presentation styles and behaviour, saved in an XHTML file. This content is available to any browser or device, regardless whether CSS and JavaScript are supported. Figure 1 illustrates the appearance of the basic XHTML rendered with default browser settings. The tags used in this layer reflect the structure and semantic meaning of the document. For example, the first and second level heading tags <h1> and <h2> are used to establish the relevant sections on the page. The menu's various sections are appropriately organised in tables, and the respective menu items and prices are placed in the table rows and cells. The full source code for the entire Food Fix web site is available for download in [3].

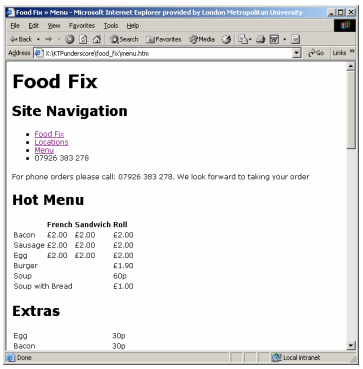

Figure 1. Food Fix Menu – Basic XHTML.

3. LAYER 2: PRESENATION
This layer consists of two CSS stylesheets: basic.css and modern.css. The purpose of the basic.css is to define styles for font, colour, text, and backgrounds that can be handled by any CSS enabled device. The second modern.css stylesheet provides the styles for features that can be handled by modern browsers and other "screen" devices. The two stylesheets are linked to the XHTML document using the following code:

```
<link rel="stylesheet" type="text/css" href="basic.css" media="all" />
<style type="text/css"
media="screen">@import/**/"modern.css";</style>
```

The first line ensures that the basic.css is available for all browsers and devices, new and old. The second line uses a "hack" technique that only allows certain modern browsers to render the modern.css styles [4]. Figure 2 illustrates the scrolling food menu rendered according to the rules defined in the two cascading stylesheets.

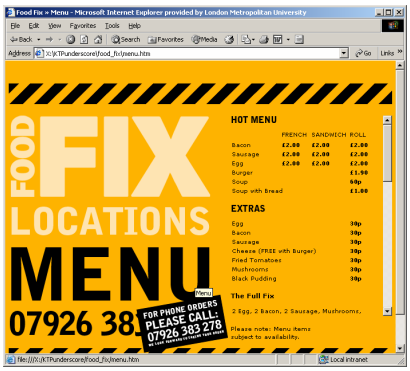

Figure 2. Food Fix Menu – With CSS.

The entire food menu section is placed between the div tags:

```
<div id="menu_scrolling">…</div>.
```

The positioning and the scrolling functionality of the menu is provided by the CSS rule defined for the "menu_scrolling" id in the modern.css:

```
#menu_scrolling {position: absolute; height: 350px; width: 282px;
 overflow: auto;}.
```

4. LAYER 3: BEHAVIOUR

This layer consists of an external JavaScript file, which overrides the default scrolling behaviour provided by the CSS overflow property, and introduces a more elegant and interactive way to scroll through the food menu. The scrolling functionality was developed using the Mootools JavaScript framework [5], which provides compatibility with major browsers and makes writing JavaScript code more efficient. The effect is that as a user clicks either the 'up' or 'down' buttons, the food menu gracefully scrolls one "view" worth of distance in the desired direction. Using MooTools, the menu's overflow property is set to hidden, and 'up' and 'down' buttons are revealed. Then event handlers are attached to these buttons, which fire off the scrolling behaviour when clicked. For example, the scroll up is achieved by using the following code:

```
$('button_up').extend({  onclick: function() {
  var menuScrolling = new Fx.Scroll('menu_scrolling', {duration:
1000, transition: Fx.Transitions.sineOut});
  var scrollTop = $('menu_scrolling').scrollTop;
  if(scrollTop > 0)
    menuScrolling.scrollTo(0, scrollTop - moveBy); } });.
```

Figure 3 illustrates the appearance of the menu page rendered by a JavaScript-enabled browser.

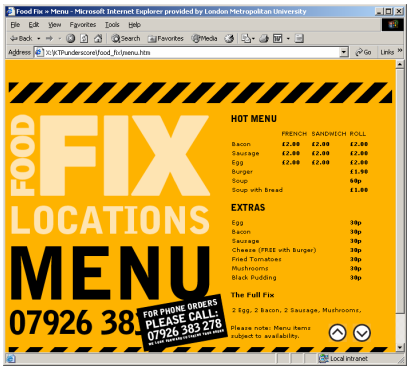

Figure 3. Food Fix Menu – With CSS & JavaScript.

5. CONCLUSIONS

The ideas presented here give a practical example of how to apply the guiding principles of the progressive enhancement methodology in developing a web site. By adding layers of presentation and behaviour on top of an already solid foundation of semantic structure and content, the end user enjoys the most modern of browsing experience possible for the device being used.

The progressive enhancement approach could be loosely considered as an application of the Model-View-Controller architectural pattern [6]. In this scenario, the data embedded in the content layer represents the model, and the browser software represents the controller. The view will depend on the browser software, which selects how to render the content depending on the features it supports. For example, a handheld device may only display the basic markup, while a modern browser will render the content using the embedded presentation and behaviour layers.

In this case study, the content layer includes both data and basic markup. This layer could be further divided in two layers that contain the data in some standard format (e.g. XML), and the markup as part of an XSLT stylesheet. However, these technologies are still not consistently supported by modern devices.

6. ACKNOWLEDGMENTS

Our thanks to the team at Undesrcore Design, who developed the Food Fix web site.

7. REFERENCES

[1] Food Fix web site, http://food-fix.co.uk/menu.php, 2007.

[2] Champeon, S. *Progressive Enhancement and the Future of Web Design,* http://www.webmonkey.com/03/21/index3a.html, 2003.

[3] Food Fix source code, http://labs.underscore.co.uk/food-fix/food-fix.zip, 2007.

[4] *Hide CSS from browsers :: @import,* http://imfo.ru/csstest/css_hacks/import.php, 2007.

[5] Mootools, http://www.mootools.net, 2007.

[6] Model View Controller, http://www.phpwact.org/pattern/model_view_controller, 2006.