BrowserBot: An Online Browser Code Conversion Tool

Sara Aycock
Department of Computer Science and Engineering
Notre Dame, IN 46556
Saycock11@gmail.com

Abstract. Cross browser compatibility is an important aspect of a web-based application. Since some aspects of coding have to be different for different browsers, it is beneficial to have a web tool that translates code meant for a certain web browser into code compatible with another browser. This paper discusses the added benefits of using BrowserBot, a web tool that converts code into multi-compatible code rather than writing multiple versions of code by hand.

1 Introduction

Cross browser compatibility, or multi-browser compatibility, is a requirement of almost every web site implemented today. There are over 100 different browsers and browser versions that web users can use today, although most are outdated and hardly used.[4] Browsers sometimes have different software coding standards than each other. Programs written for a certain browser may have some features that will not be compatible with other browsers.

Most popular browsers include: Internet Explorer, Firefox, Safari, and Opera. A website that is cross-compatible is able to be correctly executed with all browsers. Most websites are created to be at least multi-browser compatible with all the major browsers. The main features that are not compatible include Cascading Style Sheets (CSS), JavaScript, and even some HTML code. Additional features include ActiveX objects and third party toolkits.

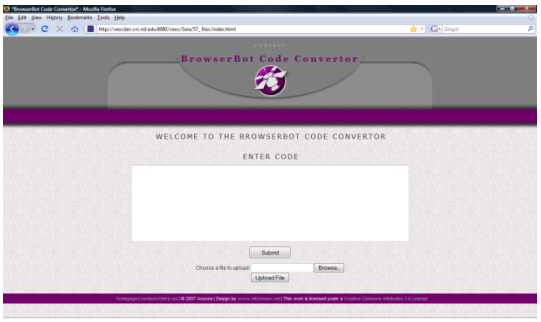
A browser code converter is a useful tool for web site designers. Designers can write their code to conform to software requirements for a certain browser. They then can enter their code into the converter and receive an updated code that is compatible with all the major web browsers. This application is an open source program so that anyone could use it and improve it if they can.

2 BrowserBot Architecture

We created the BrowserBot as a web-based application consisting of a PHP processor and a web interface. The web interface is standard HTML, CSS, and JavaScript, and the PHP processor is a series of PHP logic files. Below is a screenshot of the main user interface.

Figure 1: The Interface

3 Design



The PHP processor takes a user input of code and performs several scans of this code to make it multi-browser compliant. It then returns a multi-compliant browser code to the user. There are two ways that a user can upload code: through a file or by directly entering the code into a textarea. (see figure 1).

3.1 Assumptions

In order for BrowserBot to function properly all inputted code has to be validated and have correct syntax. An incorrect syntax could break the application causing more incorrect code to be outputted.

3.2 Limitations

Currently the Browser Code Converter only converts code that is compatible with Mozilla Firefox 3.6 into code that is compatible with Internet Explorer 8. The converter solves three major differences between the coding of Internet Explorer and Firefox.

3.3 Conversions

3.3.1 document.getElementById

The converter searches for any instance of the getElementById method which returns an object by its id. This is a method that is compatible with Firefox, but not with some versions of Internet Explorer. The converter adds a getElementById function at the end of the code and adds a string of document.getElementById after each getElementById method.This way the code is now compatible with both Mozilla and Internet Explorer.

```
function getElement(aID){
   return (document.getElementById(aID) ?
   document.getElementById(aID):
   document.all[aID])
}//end function getElement
```

3.3.2 <tbody>
Some versions of Internet Explorer require a tag called <tbody> in the code after a <table> tag in order for the table to be read. The converter inserts <tbody> tags after all <table> tags and closing <tbody> tags before each closing <table> tag. This allows tables to be viewed using an Internet Explorer browser.

3.3.3 getAttribute("className")
Some versions of Internet Explorer require that the *getAttribute* method uses the *className* attribute rather than the *class* attribute. The converter finds all instances of this and adds an if statement and a *className* method. For example, if the original code:

```
var test = myobject.getAttribute("class") ;
```

is inputted into BrowserBot then the ouput will be the following:

```
if(document.all)
 var test= myobject.getAttribute("className")
else
 var test = myobject.getAttribute("class") ;
```

4 Results

We applied the BrowserBot to the creation of the Miami-Dade virtual Emergency Operations Center (vEOC) program. The vEOC is a web-based training program for the Miami-Dade Emergency Operations Center.[8, 9, 14] It simulates emergency situations, allows users to respond to these emergencies by interacting with each other and the computer. The program also evaluates the actions taken by the users to determine whether or not they took the correct actions. Below is a screenshot of the vEOC main user interface.

Figure 2: vEOC interface

The majority of the program's code was written to be compatible with the Mozilla Firefox browser; however the program's clients in Miami-Dade are using Internet Explorer to run the program. The code had to be updated to be multi-browser compatible. Using BrowserBot results in a more efficient conversion of the code.

5 Future Work

There is much potential work to be done on the converter in the future. Many methods in Mozilla do not cross over to Internet Explorer and they should be added to the converter. Also eventually the converter should be able to make a code compatible to all the major browsers instead of just Firefox to Internet Explorer.

In the long run it would most likely be beneficial to program the code converter in a different language, preferably something like Perl since it is more efficient when searching files and strings.[1]

6 References

1. answerbag.com (2006). http://www.answerbag.com/q_view/16515. Accessed on July 12, 2010.

2. Becerra-Fernandez, I., Prietula, M., Madey, G., and D. Rodriguez (2007) Project Ensayo: a Virtual Emergency Operations Center for Disaster Management Research, Training, and Discovery. Presentation and Proceedings, *The First International Conference on Global Defense and Business Continuity (ICGD&BC 2007)*, San Jose, CA.

3. Becerra-Fernandez, I., Madey, G, Prietula, M., Rodriguez, D., Valerdi, R., and T Wright. (2008) Design and Development of a Virtual Emergency Operations Center for Disaster Management Research, Training, and Discovery, *HICSS-41*, Hawaii.

4. Freelancefolder.com (2009). http://freelancefolder.com/7-fresh-and-simple-ways-to-test-cross-browser-compatibility/. Accessed on July 12, 2010.

5. Ibm.com (2005). http://www.ibm.com/developerworks/web/library/wa-ie2mozgd/. Accessed on July 12, 2010.

6. impressivewebs.com (2009). http://www.impressivewebs.com/7-javascript-differences-between-firefox-ie/. Accessed July 27, 2010.

7. Morelli, R., A. Tucker, N. Danner, T. R. De Lanerolle, Heide, J. Ellis, O. Izmirli, D. Krizanc, and G. Parker. Revitalizing Computing Education Through Free and Open Source Software for Humanity. 2010.

8. Nikolai, C., Becerra-Fernandez, I., Prietula, M., and G. Madey (2009) Project Ensayo: Designing a Virtual Emergency Operations Center. *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, San Antonio, Texas.

9. Nikolai, C., Johnson, T., Becerra-Fernandez, I., and G. Madey (2010). Leveraging WebEOC in Support of the Haiti Relief Effort: Insights and Recommendations. *The 7th International Community on Information Systems for Crisis Response and Management (ISCRAM) Conference*, Seattle, WA.

10. quirksmode.org (2010). http://www.quirksmode.org/compatibility.html. Accessed on July 27, 2010.

11. Reloco.com (2006). http://www.reloco.com.ar/mozilla/compat.html . Accessed on July 12, 2010.

12. T. Johnson (2010) vEOC Usability Test. July 1.

13. Validator.w3.org (1994). http://validator.w3.org/#validate_by_upload. Accessed on July 12, 2010.

14. Veocdev.crc (2010). http://veocdev.crc.nd.edu:8080/veoc Accessed on June 7, 2010.

7 Appendix

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x
CSS 2	incorrect	incomplete	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	to be tested
CSS 3	minimal	incomplete	almost	almost	almost	almost	almost	almost	almost	almost	almost	almost	almost	almost	almost	almost	to be tested
DOM Core Node manipulation	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	to be tested
DOM HTML HTML tag manipulation	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	to be tested
DOM CSS Stylesheet manipulation	alternative	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	yes	to be tested
DOM Events	yes	to be tested	yes	to be tested	to be tested	to be tested	to be tested	to be tested	yes	to be tested	to be tested	to be tested	yes	to be tested	to be tested	to be tested	to be tested
CSS Object Model View Element dimensions, mouse coordinates, and miscellaneous	incomplete	almost	yes	almost	almost	almost	almost	almost	almost	almost	almost	almost	almost	almost	almost	almost	to be tested

CSS 2 has become the baseline of CSS support; without it a browser is decidedly backward.

The compatibility information here is about the CSS3 modules I test. It is not necessarily valid for the browsers' entire CSS3 support.

The W3C DOM Core module defines how to access, read and manipulate an XML document. Well-formed HTML documents are XML documents, so these methods and properties can be used to completely rewrite any HTML page, if you so wish.

Here you find details on how to find elements, how to create new ones, how to read out node information and how to change the structure of the document.

Though HTML documents are XML documents, they have a number of special features that the average XML document doesn't have. The W3C DOM HTML module defines these special cases and how to deal with them.

Here you find details on getting and setting properties of HTML elements, such as `className` or `id`. The `innerHTML` property is of prime importance to any DOM script.

Style sheets are part of the document, too (sort of). The W3C DOM CSS module gives access to style sheets and allows you to change a style sheet.

This module contains some browser incompatibilities, but they are of the cute kind. W3C and Microsoft define some different methods and arrays, but some simple object detection allows you to evade these problems.

Event support in all browsers.

- Konqueror lacks support for some common events.

This specification contains several age-old properties that all browser support but that never have made it to a W3C specification yet.

Selector	IE 5.5	IE 6	IE 7	IE8	IE9 pr3	FF 3.0	FF 3.5	FF 3.6	FF 4b1	Saf 4.0 Win	Saf 5.0 Win	Chrome 4	Chrome 5	Opera 10.10	Opera 10.53	Opera 10.60	Konqueror 4.x