Development Manual

Table of Contents

[Introduction 2](#_Toc165906821)

[Toolbar Architecture 2](#_Toc165906822)

[Basic Operations: Move and Engage 2](#_Toc165906823)

[Contexts and the Context Manager \*\*\*John comment 2](#_Toc165906824)

[The Main Toolbar 2](#_Toc165906825)

[Sub Toolbar Components 2](#_Toc165906826)

[Technologies, Libraries, and References 3](#_Toc165906827)

[Prototype.js 3](#_Toc165906828)

[Firefox Structure and Plugins 3](#_Toc165906829)

[Essential Components 3](#_Toc165906830)

[Highlighting 3](#_Toc165906831)

[Event Capturing 3](#_Toc165906832)

[Surf Mode (Single Input Mode) 3](#_Toc165906833)

[Auto Mode (Auto Iteration Mode) 3](#_Toc165906834)

[Sound Manager 3](#_Toc165906835)

[SubToolbars 3](#_Toc165906836)

[Links SubToolbar 3](#_Toc165906837)

[Navigation SubToolbar 4](#_Toc165906838)

[Scrolling SubToolbar 4](#_Toc165906839)

[Surf Mode SubToolbar 4](#_Toc165906840)

[Developing and Extending the Hawking Toolbar 4](#_Toc165906841)

[Helpful Resources 4](#_Toc165906842)

[Meet the Developers 4](#_Toc165906843)

[XUL and Firefox Tutorials and References 5](#_Toc165906844)

# Introduction

The Hawking Toolbar Project's goal is to design an extension for Mozilla’s Firefox browser to enable users to view and interactively navigate the content of the web using a limited input interface. The target input interface for which the toolbar is designed is a one or two switch interface. The physical switch will be mapped to an event in the browser such as a key press or mouse click. These switches will allow the user to interact with the toolbar which in turn will allow them to view web content as well as browser functionality such as clicking links, scrolling, or interacting with the browser history.

This document contains an overview of the basic design and implementation of the toolbar. It begins with an overview of the architecture and a list of technologies and references used, then moves on to describe some of the basic components and how they were implemented, and finally finishes with documentation on how to extend the current toolbar to add additional functionality.

# Toolbar Architecture

## Basic Operations: Move and Engage

The Hawking Toolbar is designed around two basic input mechanisms: move and engage. These two mechanisms allow the user to access and interact with all of the functionality of the toolbar and thus content of the web page the user is visiting.

## Contexts and the Context Manager \*\*\*John comment

A Context within the Hawking Toolbar is an abstraction that is designed to present the user with a set of functionalities that can be accessed using the move and engage architecture. Conceptually, a Context is simply a list of functions that the user can iterate through using the Move input and then choose one of the functions from the list using the Engage input. A Context is implemented in the code as a ContextList

The Context Manager is a JavaScript class

## The Main Toolbar

The main toolbar serves as the main interface from which all other toolbar functionality can be accessed. It is simply a context that contains buttons which either directly activate functionality of the Hawking Toolbar or enter another context using the Context Manager to access additional functionality.

## SubToolbar Components

Families of functionalities are made known to the user via the creation of sub toolbars. For instance, the Navigation sub toolbar contains a family of functions (Forward, Back, Refresh) that are accessed via buttons on the Navigation sub toolbar. Sub toolbars are Contexts that can be passed to the Context Manager and thus are built upon the move and engage architecture and linked to the main toolbar by a button within the context of the main toolbar. By engaging this button, the user changes scope to the subtoolbar and the main toolbar is hidden. Scope is returned to the main toolbar by clicking an exit button on the subtoolbar that calls the FireHawk.UnScope() method implemented in “hawkingbar.js”.

# Technologies, Libraries, and References

## Prototype.js

The Hawking Toolbar makes use of the prototype.js library found at:

<http://prototype.conio.net/>

This library serves as a JavaScript Framework that eases development and allows easier interaction with the DOM. The main purpose of using the prototype framework is to access its ability to work in a more structured object like thought process than used with standard JavaScript style.

## Firefox Structure and Plugins

Describe how Firefox is structured

# Essential Components

## Event Capturing

## Highlighting

The main purpose of using the Hawking Toolbar is to navigate web pages by detecting and clicking links using input switches. The Highlighter class implemented in hawkinghighlight.js provides supports highlighting within a window in Firefox. The Highlighter class creates an HTML <div> element and then uses absolute positioning and component resizing to place this highlighter div

## Surf Mode (Single Input Mode)

Surf Mode or Simple Mode is a reduced functionality mode in which the toolbar only iterates through the links available in

## Auto Mode (Auto Iteration Mode)

## Sound Manager

# Essential SubToolbars

## Links SubToolbar

DOM id: “HawkingSBPageNav”

The Links SubToolbar provides buttons that allows the user to iterate forwards or backwards through a list of available links on the current webpage the user is viewing and simulate a mouse click on a link using the following functions implemented in the FireHawk class:

FireHawk.HawkingPageNext()

FireHawk.HawkingPageClick()

FireHawk.HawkingPagePrev()

## Navigation SubToolbar

DOM id: “HawkingSBHistoryNav”

The Navigation SubToolbar allows the user to access the browser’s history and move forward or backward through it as well as refresh the current page or go to the browser’s homepage. This functionality is achieved by using an onclick attribute to call functions defined in the browser.js file provided as part of the Firefox chrome. The functions headers are listed below:

function BrowserForward(aEvent, aIgnoreAlt)

function BrowserBack(aEvent, aIgnoreAlt)

function BrowserReload()

function BrowserReloadSkipCache()

function BrowserHome()

## Scrolling SubToolbar

DOM id: “HawkingSBScroll”

The Scrolling Subtoolbar implements four buttons that allow the user to scroll up, down, left, and right. Each calls a corresponding function within the FireHawk class to perform scrolling using an onclick attribute. The functions called are:

FireHawk.htbScrollUp()

FireHawk.htbScrollDown()

FireHawk.htbScrollRight()

FireHawk.htbScrollLeft()

## Surf Mode SubToolbar

DOM id: “HawkingSBLiteracy”

The Literacy SubToolbar simply serves as a visual notice that the user has currently scoped into surf mode instead being in the scope of the main toolbar. Since the scope is in surf mode, the Move and Engage capturing will iterate through the surf mode context instead of another toolbar context. The toolbar also has a return or exit button that allows an administrative user to click and exit back to the main toolbar, returning scope to the main toolbar context.

# Developing and Extending the Hawking Toolbar

How to develop and implement….steps needed?

# Helpful Resources

## Meet the Developers

John Foushee

John Foushee is a Class of 2008 computer science major at the University of North Carolina at Chapel Hill originally from Charlotte, NC.

Andrew Hulbert

Andrew Hulbert is a Class of 2008 computer science major at the University of North Carolina at Chapel Hill originally from Snow Hill, NC.

Brian Louden

Brian Louden is a Class of 2009 computer science major at the University of North Carolina at Chapel Hill originally from Raleigh, NC.

## XUL and Firefox Tutorials and References

XUL Planet **-** <http://www.xulplanet.com/>

XUL Planet offers tutorials for basic XUL programming and structure as well as an extensive XUL component and element reference. It is a must read before programming in XUL as should be used as documentation source of XUL components for reference during development.