Hacking Firefox Boot Camp

chapter 1

by Mel Reyes

in this chapter

- ☑ Installing the Document Inspector
- ☑ Editing text tools
- ☑ Using the JavaScript Console
- ☑ Your profile explained
- ☑ Backing up before hacking

Gearing up to hack Mozilla Firefox is as simple as understanding some basic Internet browser features, installing some tools, and being able to find files on your computer system. Then it gets just a tad more complicated. This chapter starts things off by discussing the different methods for hacking Firefox that are covered in this book and how things will progress. Try not to get bogged down with the onslaught of topics that are covered here, because many of them are covered in depth throughout the book.

If you understand some basic web programming tools, such as CSS, JavaScript, and XML, you are one step ahead of the game. Conversely, if you are not well versed in these technologies, you will find plenty of examples and references to guide you along your hack training.

First, we cover some of the key tools you should use to get an edge when hacking Firefox. Tools covered include the Document Inspector, basic text editors, and JavaScript Console. A good portion of this chapter helps you find your personalized Firefox settings in your Profile directory and then highlights the key features of most of the files. As you continue to read this book, you will tap into many of the key components of your profile. Then we will approach the different methods of hacking the browser using some of the functionality included with the browser, such as about:config and the JavaScript Console. Finally, you'll learn the basics of changing your preferences and interface by manually hacking the prefs.js, user.js, userChrome.css, and userContent.css files. After getting all your gear, you will begin your quest to understand the core technologies involved in hacking just about every aspect of Mozilla Firefox.

Installing the Document Inspector Gadget

Out of the box, the Firefox Installer has two installation modes: Standard and Custom. If you have already done a Standard installation, you will be missing a key hacking and programming component: the Document Inspector, or DOM Inspector.

The Document Inspector extension is a development tool used to analyze the Document Object Model (DOM) of web pages or the Firefox interface, and is very useful in digging deeper into the core structure of web pages, the Firefox browser window, and browser elements. Currently, this browser development tool is available only from Firefox's main installation process. Later in this chapter and throughout the book, you will begin to see how web page document model standards fit into Firefox's interface customization.

So you want to install the Document Inspector (also called the DOM Inspector), but you already have Firefox installed? No problem. Simply reinstall Firefox, but instead of choosing the Standard installation type, choose the Custom installation type.

Follow the prompts until you get to the Select Components screen. Select Developer Tools, by clicking the checkbox as shown in Figure 1-1, to install the Document Inspector tool.

FIGURE 1-1: Install the Document Inspector tool

Once you have completed this installation or reinstallation, you will notice the DOM Inspector in your Tools menu is now available to all profiles on the system. You can use this tool as a resource for dissecting bits and pieces of web pages and the Firefox interface. Figure 1-2 shows the DOM Inspector view of a web page that is currently loaded in the main browser. Note that the hierarchy for the currently loaded web page is displayed in the left-hand panel, with each level or node grouped by the HTML-defined hierarchy and code. Additionally, details on the currently selected node are displayed in the right-hand panel; this panel becomes useful when hacking the Firefox interface.

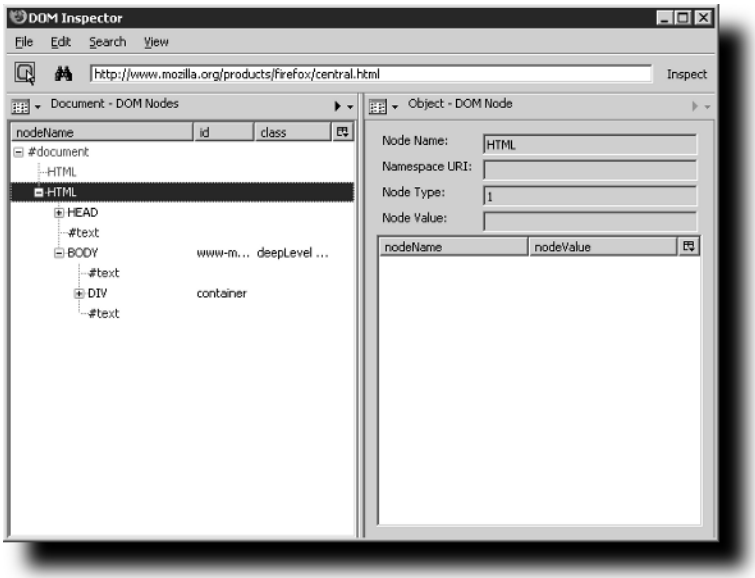

FIGURE 1-2: DOM Inspector document tree and object properties

Occasionally, I have noticed that running the DOM Inspector on a fresh install or reinstall did not yield the desired results or did not work at all. To correct this, I have tried either uninstalling Firefox and then reinstalling with the Developer Tools option enabled, or creating a new Firefox profile. Unless you are running an older version of Firefox that prompts you if you want to delete all the program files, the Firefox uninstaller retains all the supporting plugins and other files that it might need. If prompted to delete all Firefox program files, do not confirm this prompt; doing so will require reinstallation of plugin support for features such as Macromedia Flash, Shockwave, QuickTime and/or RealPlayer. Historically, uninstalling and reinstalling and/or creating a new profile have been the two methods that I have used to resolve mysterious Firefox issues when I could not consistently reproduce them.

For information on how to use the Profile Manager to create a new Firefox profile, visit the incredibly useful MozillaZine Knowledge Base at `http://kb.mozillazine.org/Profile_Manager`.

While having a pretty hierarchy tree of your HTML is nice, the real benefit of the DOM Inspector is using it to hack Firefox itself. Firefox is built on a cross-platform extensible user interface language called XUL, which is based on XML standards and was created to support Mozilla applications. The user interfaces for the Mozilla Suite, Firefox, Thunderbird, and Sunbird all use XUL to create and display the user interface. This interface foundation is the core element that helps all these programs run on different operating systems. The interface is a collection of object definitions used to create each of the elements on the screen.

Using the DOM Inspector can easily help you walk through the hierarchy used to create the actual windows displayed by Firefox. To load the browser window's XUL hierarchy, just follow these steps:

1. In the main browser window navigate to any external web site, such as `http://www.mrtech.com`.
2. Open the DOM Inspector from the Tools menu. At this point, the web site opened in the previous step will be parsed.
3. From the File menu, choose the entry from the Inspect a Window menu option that corresponds with the site opened in the first step, in this case, "MRTech.com - Mozilla Firefox."

After following these steps, the nodes or page elements for the main browser window are loaded and available for visual inspection, as shown in the left-hand Document - DOM Nodes panel of Figure 1-3. For future reference, you can use the following location or URL for the main Firefox browser window to quickly browse the node tree: `chrome://browser/content/browser.xul`.

After you have the nodes listed in the left panel for the browser.xul page, just click on Inspect to the right of the location bar to open a window to browse the actual page on the bottom half of the screen (also shown in Figure 1-3).

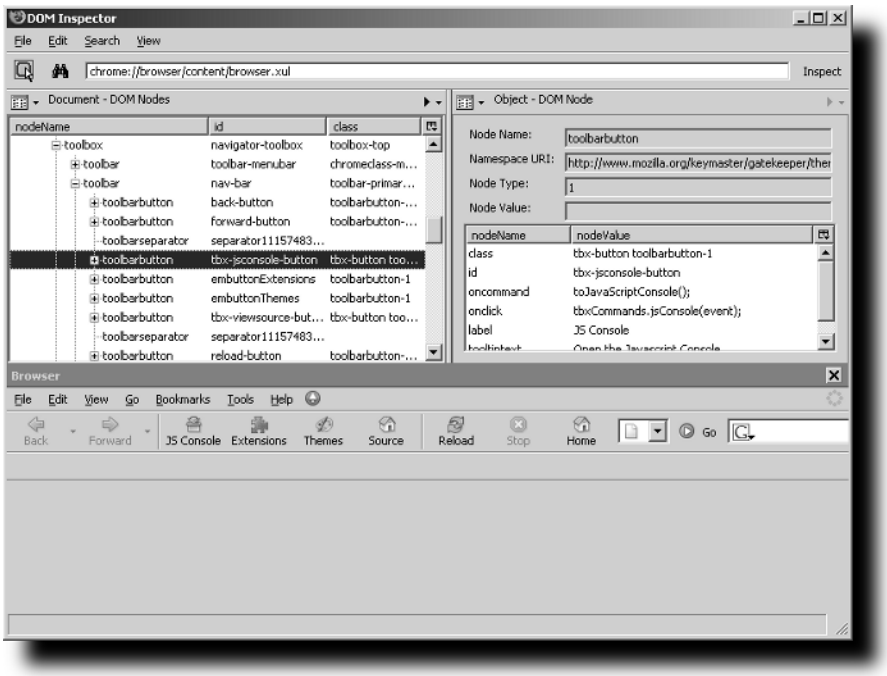

FIGURE 1-3: DOM Inspector with Firefox's browser.xul loaded

To find the internal name or id of a specific Firefox window element, I like to use the Select Element By Click option from the Search menu. Once you have selected this, you can click on any of the screen elements on the bottom half of the window and the DOM Inspector jumps to the actual definition for that element within the hierarchy. There you can easily access the internal id associated with the element and use that for future coding or manipulation.

The Select Element By Click option works only after you have clicked on Inspect next to the location bar.

Using the DOM Inspector and Figure 1-3, I will now explain how XUL is used to build the main Firefox browser window. As you see from this figure, there is a XUL object or node called toolbox with an id of navigator-toolbox. This object defines the top-level toolbar container on the main browser window. This container holds the three individual toolbar objects that are visible in the main window. They are toolbar-menubar, nav-bar, and PersonalToolbar. Digging deeper, the nav-bar toolbar object has a toolbarbutton object defined as back-button. This object holds the object information for the Back button, which is displayed on the browser window's navigation toolbar, and the fun continues from there with the rest of the interface XUL definitions.

All in all, the DOM Inspector is the most useful tool to begin digging around and understanding the interface elements that make up the Firefox windows.

Editing and Programming Text Tools

Another tool you will need is a good text-file editor. While the basic text editor that comes with the operating system works for some people, I have found that more functionality is desirable when working with web or Firefox files. Choose an editor with good code syntax highlighting or with other advanced options.

Key attributes to look for in a good programming editor or interface include the following:

- Is it actively developed?
- Can it support Windows, UNIX, Mac OS, and Unicode text-file formats?
- Does it have customizable tab stops or multi- or tabbed-file support?
- Is it free?

Using the editor provided by your operating system may work for you for now, but you may find yourself being a little more productive if you opt for a more up-to-date editor. Several good freeware text editors are actively developed and contain features that even the most diehard vi expert could grow to appreciate and love. Additional coverage on better programming editors is available in Chapter 16. One text editor that I have used in the past is EditPad, which works on Windows and Linux-based systems. I have also used the following Windows-based editors: Notepad++, PSPad, and the quick and simple Win32Pad.

You also have a few options for Linux distributions, including KDevelop, Nedit, Kate for KDE, or GEdit for GNOME. Apple Mac users have a lot of options for editors, including BBEdit, jEdit, and Mellel.

In addition to these editors, you can download and install the chromeEdit extension, featured in Chapter 2, for basic editing of the user.js, userChrome.css, and userContent.css files. For more information or to download chromeEdit, visit `http://cdn.mozdev.org/chromedit/`.

To download any of the aforementioned editors, just visit their sites:

- **BBEdit:** `http://www.barebones.com/products/bbedit/`
- **EditPad:** `http://www.just-great-software.com/`
- **jEdit:** `http://www.jedit.org/`
- **Kdevelop:** `http://www.kdevelop.org/`
- **Mellel:** `http://www.redlers.com/`
- **Nedit:** `http://www.nedit.org/`
- **Notepad++:** `http://notepad-plus.sourceforge.net/uk/site.htm`
- **PSPad:** `http://www.pspad.com/en/`
- **Win32Pad:** `http://www.gena01.com/win32pad/`

For more options and information on programming editors and software, visit `http://www.thefreecountry.com/programming/editors.shtml`.

Using the JavaScript Console

The JavaScript Console is a very handy debugging tool, is a built-in feature of Firefox, and does not require special installation. If you are a web developer or are planning on creating Firefox extensions, the JavaScript Console is the tool that you want to tap into to make sure you use the proper JavaScript syntax and to help you find your coding bugs.

To open the console, select JavaScript Console from the Tools menu. The console shows you three different types of information: Errors, Warnings, and Messages.

When first opened, JavaScript Console shows all messages for your current browser session, as illustrated in Figure 1-4. The console shows errors only if there are any; this includes errors for all sites visited since Firefox was last opened up. If there are no messages displayed, Firefox has not encountered JavaScript errors on any of the pages you have navigated to so far.

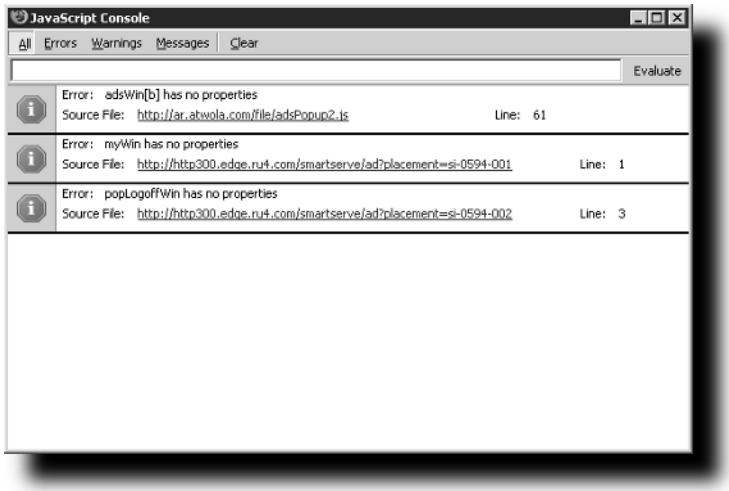

FIGURE 1-4: JavaScript errors displayed in console window

A key feature of the console is its ability to jump to the offending code if you click the Source File: link just below the error message. Doing so opens the View Source window directly to the line number referenced in the message, as shown in Figure 1-5.

FIGURE 1-5: Source code of offending JavaScript code

Chapter 15 dives deeper into using the console and covers how to enable some of the advanced debugging preferences. The chapter also shows you how the JavaScript Console is a good area to display status messages while debugging and creating your extensions.

What and Where Is My Profile?

Your settings are stored in a Firefox directory or profile, which Firefox creates right after your first install. Your profile contains all your Firefox-specific settings, including but not limited to the following:

- Extensions
- Themes
- Bookmarks
- Saved form values
- Saved passwords

Additionally, your profile contains any imported settings from Internet Explorer, Netscape 6/7, or Mozilla browsers.

Losing any part of your profile can be extremely annoying; Chapter 2 covers how to hack settings in your Firefox profile.

To work with your current profile manually, you first need to find the root directory where your personal settings are stored. To do this, you must follow the directions specific to your operating system, shown in the next section. This operating system–dependent "settings" directory is referred to as your %UserPath% as we continue. From there, you will be able to find the path and directory structure in which Firefox has stored your user profile.

Finding Your User Path

Each operating system has a different directory to which it saves your user settings. Most applications take advantage of this operating system "user path" to store their settings, so as not to collide with other users who might log into the same computer. Firefox does the same; it uses this directory to create the user's profile. The challenge is that each operating system uses a different naming and directory structure to store these files and settings. Making life even more complicated, different versions of the same operating system (for example, Windows) use different structures. Peruse the following subsections to find the operating system you are currently using, and read how to find your user path.

Firefox Profile Name History Lesson

Throughout its development cycle, Firefox has been through a few changes. Earlier development and testing builds saved the profile and settings to the following Phoenix directory:

```
%UserPath%\Phoenix\Profiles\
```

Phoenix was the original name for the Mozilla browser–only project; this made perfect sense for the profile directory name. Even though the project name changed to Firebird for legal reasons, the Phoenix Profile directory persisted. Finally, after additional legal and copyright wrangling, the name Firefox was born. Not too long after this, the development version of Firefox included an automated migration of most of the profile entries and files from the Phoenix directory to the Firefox directory:

```
%UserPath%\Firefox\Profiles\
```

But it did not end there. The final decision made was that for new installations the root profile directory should live in harmony with the core common Mozilla directory structure and eventually become the following:

```
%UserPath%\Mozilla\Firefox\Profiles\
```

So if you have been testing Firefox for a long time now, you may have two or three directories, but only one is your current working Profile directory.

Using Windows?

If you are using Windows, your user directory should look similar to this:

- **Windows 2000/XP:** C:\Documents and Settings\<LOGINNAME>\Application Data\
- **Windows NT:** c:\Windows\Profiles\<LOGINNAME>\
- **Windows 9x/ME:** C:\Windows\Application Data\Mozilla\Firefox

The drive (C:\ above) and location of the default Windows directory may vary based on your custom installation.

Using Linux/UNIX?

If Linux/UNIX is your modus operandi, you should expect to find your Firefox profile in a directory similar to ~/.mozilla/firefox.

Using Mac OS?

Finally, for all you Apple aficionados, your directory structure should be something similar to ~/Library/Application Support/Firefox.

Now that you have found your user directory, this will now be referred to as %UserPath% and will be used to track down where Firefox has stored your profile.

For official information on locating your Firefox profile, visit `http://www.mozilla.org/support/firefox/edit#profile`.

Express Pass to Your Profile Path

One nice feature that Firefox finally enabled is human-readable settings for the profile.ini file with the direct or relative path to the current profile(s). Prior to this, profile information was stored in binary format only, and automating and scripting Firefox profiles was difficult to do. The profiles.ini file lives in the now common path for Firefox Profiles, which is as follows: %UserPath%\Mozilla\Firefox\.

The profiles.ini file will look similar to the following if this was the first time you installed Firefox:

```
 [General]
StartWithLastProfile=1

[Profile0]
Name=default
IsRelative=1
Path=default\zsryldfv.slt
```

In this first example, notice that the `IsRelative` setting is equal to 1, which is a Boolean toggle for true. This means that the path is relative to the common Mozilla Firefox path of %UserPath%\Mozilla\Firefox\, so the full directory path would look something like %UserPath%\Mozilla\Firefox\default\zsryldfv.slt.

Note that `zsryldfv` in the path is a randomly generated directory name and varies from system to system. If you had previously installed earlier builds of Firefox that stored the profiles in other places, the profiles.ini file might look something like this:

```
[General]
StartWithLastProfile=1

[Profile0]
Name=default
IsRelative=0
Path=%UserPath%\Firefox\Profiles\default\zsryldfv.slt
```

Moreover, you will notice that `IsRelative` is zero or false, so the Path entry in the file reads as-is, or absolute, and that is where you will find your current profile.

Unhide Your %UserPath% and Enable File Extensions for Windows

For Windows systems such as Windows 2000 and XP, the %UserPath% may be hidden, and a file's extensions may not be visible. To correct this situation on these systems, just follow these steps:

1. Open Windows Explorer by selecting the Run option from the Start menu.
2. Enter **explorer.exe** and press OK.
3. On the menu bar, select Tools➪Folder Options, and in the View tab uncheck the "Hide extensions for known file types" option.
4. Then check the "Show hidden files and folders" option and click OK at the bottom of this dialog box.

At this point, the file listing should refresh, and hidden directories and file extensions will be available within all application and file/folder dialogs.

Backing It Up Before Hacking It Up

As with any hack or modification to a program, being able to restore to a previously working state is critical. Luckily, Firefox hacking and modifications are primarily text file based and can usually be restored very easily. For the most part in this book, we will not be hacking the binary or low-level executables of Firefox. However, you are introduced to hacking several key text files to either hack or repair your system.

This section prepares you to prepare your system for hacking and quickly points out how to back up your extensions, themes, and critical files such as your profile, and so on.

Saving the Installer, Extensions, and Theme Files

In preparation for any hacking adventure, make sure if you have to rebuild that you have all the necessary files that you previously used.

1. Make sure you create a Backup directory in a reliable location. Best practices dictate that you create a Backup folder either on your desktop or in a common backup location. This is where you want to store backups of your preferences, extensions, and any other supporting files.
2. Make sure you save the original installation file for Firefox. This will come in handy when you want to reinstall a fresh copy of the base application. Even though you probably will not do this often, there are some sections in this book where you will want to reinstall.

3. Review your currently installed extensions by going to the Extensions manager in the Tools Menu (choose Tools ➪ Extensions). If you have none, you are all set. If you do have extensions installed, you should do the following.

    a. Go down the list of extensions in the Extensions window, right-click each extension, and choose Visit Home Page from the right-click menu.

    b. Almost every extension's support page should allow you to download the XPI or extension file by right-clicking on the download or install links and saving the file to the Backup folder you created in Step 1.

    c. Some sites use JavaScript code to install their extension. For these, you will just have to bookmark the site and revisit them in case of emergency.

4. Do the same thing for Themes that you did for Extensions. Just open the Themes window (choose Tools ➪ Themes), run down the list of Themes, right-click each extension and choose Visit Home Page from the right-click menu for each theme, and save all of the individual Java Archive (JAR) or themes files to the backup directory.

Backing Up Critical Files

Now that you have all the core installation files backed up, you can proceed by backing up your profile. To ease into hacking Firefox, I recommend using the free MozBackup tool for Windows systems, shown in Figure 1-6, to back up and then restore your Firefox profile. Linux and Mac users should focus on finding and backing up the profile directory completely, which is also an option for Windows users. Chapter 2 covers the use for some of the files that are nicely packaged by MozBackup, and below is a list that describes some of the key files.

FIGURE 1-6: MozBackup backup selection screen

For more information or to download MozBackup, visit `http://mozbackup.jasnapaka.com/`.

Some of the critical profile files include the following:

- **bookmarks.html:** Where all the bookmark entries are stored and can be viewed with any browser.
- **cookies.txt:** Contains all cookies currently stored for all sites.
- **pref.js:** Contains all of the Firefox settings and customizations that you have made — for example, changing the homepage, location of last download folder, and so on.
- **hostperm.1:** Contains cookie and image permissions that have been enabled.
- **formhistory.dat:** Contains autocomplete data for form fields on web pages.
- **user.js, chrome/userContent.css, and chrome/userChrome.css:** Are not created by default and should be backed up if you have created or modified them.

To make a backup of your Firefox profile on Windows systems using MozBackup, follow these steps:

1. Download and install MozBackup.
2. Close all Firefox windows and run MozBackup.
3. Click Next on the Welcome screen.
4. Select the "Backup a profile" option if not already selected and the Mozilla Firefox listing at the bottom of the Operation Type screen and then click Next.
5. Select the profile you want to back up. (Most installations will have only one profile listed.) You can also select a different path to save the file and then click the Next button.
6. Choose whether you want the backup password protected, and follow the prompts if you do.
7. Select the components that you want backed up. To save space and time, leave the Cache entry unchecked and then click Next.

At this point, the backup begins and a PCV file is created with the date as part of the filename — Firefox 1.0.3 (en-US) - 7.10.2005.pcv, for example. One reason I like this tool is that it uses standard ZLib or Zip file compression to bundle the files, not a proprietary format. This means that the file is compatible with any extraction program that supports Zip files. You can open the file directly in your compression program of choice, or just rename the file extension from .pcv to .zip and quickly scan through and extract specific files.

Additionally, you can run through the MozBackup file to selectively restore any of the files that have been bundled by selecting Restore a Profile from the Operation Type screen. On the next screen, you select the profile and the backup file to restore from and then proceed by picking the files to restore.

If you use a Linux or Mac system, or you just want to cover all bases, make sure that you can find your profile and make a complete backup of the profile directory before proceeding. Chapter 2 covers how to find your profile, or you can visit the MozillaZine Knowledge Base article here at `http://kb.mozillazine.org/Profile_Folder#Firefox`.

While there are other, less critical files that you might want to back up in the installation path for the main application, the files covered here are really the core user files for running Firefox; the rest are plugins and additions that are covered in Chapters 11, 13, and 14.

So now you are ready to hack, right? Keep in mind that the backup that you just completed is an early cut of your profile. You will go through several iterations, hacks, and modifications throughout this book, and you may eventually want to revert to a previous version. Keeping backups of major milestones and achievement points will help you restore to one of your more recent working profiles. I can't stress enough how annoying it is to lose months' worth of bookmarks, hacks, installed extensions, and settings because I was too lazy to do a backup.

Chapter 14 gives you some additional tools and methods for backing up, which should make life a little easier.

Summary

This chapter is geared to help set the foundation for the rest of the book. To do so, I wanted to focus on having an understanding of some of the basic tools, such as the DOM Inspector and JavaScript Console, which will be referenced throughout the chapters. Additionally, the purpose of a profile was explained, as well as how to find it. Finally, the importance of backing up installation files, extensions, and your profile before you begin hacking was stressed. With this quick run-through of Firefox basics, we can now move on to bigger and better things.