Pop-Ups & Pop-Up Blockers

Contents

List of Figures

List of Tables

Pop-Ups & Pop-Up Blockers

Introduction

For the last several years, consumers on the Web have been increasingly subjected to a barrage of offers, surveys, and demos that “pop-up” on their screen demanding attention. For some users, clicking away these unwanted intrusions has seemed like a part time job. Not surprisingly, a new industry has been born out of users’ frustration—pop-up blockers.

And a new industry it certainly is! A quick look at the site, *Pop-Up Killer Review* (http://www.popup-killer-review.com/), shows not only a large number of blockers on the market (95 at this writing), but a growing variety of approaches—trainable pop-up blockers, intelligent pop-up blockers, pop-up killing browsers, and others.

While the arrival of this army of potential defenders may do something to save the sanity of consumers, it provides a significant challenge for Web site designers and developers who are trying to provide their users with honest, relevant information. Blockers can’t always discriminate between a pop-up containing an annoying ad and one that provides field help or instructions for filling out a form. Unless we at AT&T thoroughly understand the underlying technology of pop-ups and pop-up blockers, we may be inadvertently creating content that will never reach its intended audience.

In this white paper, then, the members of AT&T Technical Information Design and Development (TIDD) provide information about pop-ups and pop-up blockers that will help AT&T deliver its message but also ensure that it acts responsibly towards consumers.

Pop-Ups

Although pop-ups often appear to occur without intervention from users, they are in most cases triggered by JavaScript code on the Web page tied to specific movements or "events." Events can include such seemingly innocuous actions as:

- Opening a Web page or window (onLoad)
- Closing a Web page or window (onUnLoad)
- Clicking on a link (onClick)
- Placing the cursor on a specified element (onMouseOver)
- Removing the cursor from a specified element (onMouseOut)
- Submitting a form (onSubmit)

Once the conditions specified in the JavaScript have been met (you've placed your mouse on a banner, for instance, or you've tried to close the offending window), the JavaScript can then trigger an action or set of actions. One of these actions can be to launch (or re-launch) a pop-up. The timing of this launch can be controlled by an additional piece of JavaScript so as to "fool" some pop-up blockers into thinking that the action has been initiated by a user rather than a machine.

As it is currently being used, the term pop-up can mean almost any window launched over or under an existing browser window, including windows launched by an HTML e-mail or by third-party applications running on your computer. Later, we will need to offer some distinctions among kinds of pop-ups. But from the standpoint of technology, pop-ups can be launched in at least seven different ways. All of them can be blocked by the most extreme pop-up blockers. And not surprisingly the most commonly used methods for creating pop-ups are the ones attacked by the largest number of pop-up blockers.

- By opening a window via JavaScript (window.open).
- By linking to a new page while the first stays open (target="new").
- By calling for a hitherto invisible "layer" or Web page section (div) via JavaScript, often incorporating Flash to create an overlay or floating effect.
- By launching a player such as Flash, QuickTime, RealPlayer, or Windows Media Player.
- By launching an executable file (e.g. Gator) with an http-equiv Refresh meta element, or a JavaScript.
- By launching Windows Messenger Service messages on Windows XP and 2000 machines.
- By a third-party application already installed on your PC which launches a "fatherless" browser window.

Pop-Up Blockers

There are now nearly 100 different pop-up blockers available to users. No two are exactly the same. But for the sake of obtaining a basic understanding of the industry, we can divide them into three basic types:

- Extreme
- Intelligent
- Trainable

Some blockers are a mixture of types.

The Extreme

As its name suggests, the Extreme blocker takes the approach that the only good pop-up is a dead one. It kills *all* new browser windows, including those spawned with or without the use of client-side scripting, with or without the user's consent (i.e., a click), as well as the "fatherless" browser windows spawned by third-party applications.

Extreme pop-up blockers are typically enabled as soon as they are installed. Some integrate their controls directly into the browser menu or toolbars; others provide a stand-alone interface.

Most blockers of this type, but not all, will alert the user to the presence of a blocked pop-up window, by one of the following:

- Making a sound and/or presenting an icon when a pop-up window has been blocked.
- Presenting an icon when users place their cursor over a link, which would spawn a new window.

And when they do notify users to the presence of a pop-up, many of them are gracious enough to allow users to enable or disable pop-up blocking on the fly, often with a control key sequence.

NOTE:

As part of their scorched earth policy, these blockers may also clear browser history, cache and cookies, and block the IP addresses and hosts of known pop-up offenders.

The Intelligent

The Intelligent blocker uses simulated human behavior to determine which windows to block. If a would-be pop-up window is spawned as the result of a click, the Intelligent blocker assumes the user wants the window and allows it to spawn. If a would-be pop-up window is spawned as the result of a window loading or some other unrelated action, the Intelligent blocker assumes the user does not want the window, since he or she made no explicit request for it.

Since they take a less aggressive approach in general, the Intelligent blockers are *not* usually enabled when they are installed. The user must take some action to

enable them. Most of them integrate their controls directly into the browser menu or toolbars.

Most pop-up blockers of this type will alert the user to the presence of a blocked pop-up window, usually with a sound or an icon indicating that a pop-up window has been blocked. And like the Extreme blockers, most of the Intelligent blockers also allow the user to enable or disable pop-up blocking on the fly with a control sequence.

NOTE:

These blockers may also clear browser history, cache and cookies, and block the IP addresses and hosts of known pop-up offenders.

The Trainable

The Trainable pop-up blocker gives the most control to actual users, but it also requires the most work to set up. It allows users to choose with considerable specificity the types of pop-up windows to be blocked. The “undesirability” of the windows can be based on one or more criteria:

- The mechanisms that spawn them, e.g., JavaScript, VBscript.
- The applications that spawn them, e.g., Outlook, Netscape Messenger.
- The sites that spawn them, e.g., Yahoo, Amazon, MSN.

A user, for instance, could train a “Trainable” pop-up blocker to eliminate unsolicited new windows from yahoo.com, but allow new windows spawned as the result of a click on att.com.

Because the whole point of this class of blockers is to respond the way you specify, they are not usually enabled when installed. The user must take some action to enable them. Most integrate their controls directly into the browser menu or toolbars.

Most blockers of this type, but not all, will alert the user to the presence of a blocked pop-up window, by one of the following:

- Making a sound and/or presenting an icon when a pop-up window has been blocked.
- Presenting an icon when users place their cursor over a link, which would spawn a new window.

And when they do notify users to the presence of a pop-up, most of them allow users to enable or disable pop-up blocking on the fly, often with a control key sequence.

They also allow the user to give the blocker another training session. (“Oh, yeah, here’s another site that’s out of control. Time to add it to the blocked list.”)

NOTE:

These blockers may also clear browser history, cache and cookies, and block the IP addresses and hosts of known pop-up offenders.

The Popularity of Pop-Up Blockers

How serious is the threat of pop-ups and pop-up blockers to AT&T's web content? Consider the following:

- In July 2003, Web users experienced 7.3 billion pop-up ads, an increase of 189% from July 2002 (*internetnews.com,* August 21, 2003).
- A recent survey conducted by iVillage found that over 90% of its users identified pop-ups as the most frustrating aspect of the Web site (*internetnews.com,* September 11, 2003).
- Blockers are taking on a higher profile and becoming easier to acquire as major players on the Web, such as Google, AOL and Earthlink, make them available to their users.
- A recent study estimates that 20-25% of all Web users are using pop-up blocking software (*internetnews.com,* September 11 & 16, 2003).

As the frustration with pop-ups continues to climb and the availability of pop-up blockers grows, it's easy to foresee the trend--the use of pop-up blockers will only increase.

How Do Customers Get Pop-Up Blockers?

Customers usually get their pop-up blockers through one of three channels:

- Third-party applications, which are downloaded and installed by the customer, and integrated with the Web browser. These are often offered through the user's local Internet Service Provider. As mentioned above, Google, AOL, and Earthlink all make them available to their users. Our own ISP, AT&T Worldnet, for instance, also has one.
- Third-party application suites – usually a group of Internet security and privacy-focused utilities, one of which is an integrated pop-up blocker.
- Web browsers and e-mail clients, which include pre-installed pop-up blockers. Netscape 7, Mozilla 1.5, and AOL 9 all include built-in pop-up killers.

Many of these pop-up blockers are available at download.com (search on "pop-up blocker"); they are also accessible from cnet.com.

Recommendations

Since blockers cannot always discriminate between good and bad content, and all of AT&T's sites can be potentially affected by the misbehavior of one site owner, TIDD strongly recommends that AT&T adopt a common policy towards pop-up windows. This should include:

- Restricting the use of pop-ups on AT&T sites to the circumstances described below. If AT&T as a whole acts with a sense of responsibility in this area, users will feel less inclined to block our sites.
- Eliminating or strongly discouraging pop-ups that are not initiated by users, i.e., the ones that seem to pop-up on their own.
- Ensuring that pop-ups do not provide links to other pop-ups.
- Developing AT&T standard icons and messaging to alert customers when important content is contained in a pop-up. If a customer is already using a blocker, AT&T needs to provide enough information on the "source" page about the potential information contained in the pop-up to convince customers of the wisdom of unblocking it.

When to Use Pop-Ups

Pop-ups should only be used when at least one of the following conditions applies (see also the *When Not to Use Pop-Ups* section):

- Presentation of the new content in the same window would seriously disrupt the current task.
- The user would need to move back and forth between the new content and the content in the parent window.

Applying the criteria above, pop-ups can be effectively used to display the following:

- A definition
- An explanation
- Instructions
- Help information
- Application-like content (e.g., AT&T BusinessDirect)

Under some circumstances, the following types of content may also meet the criteria:

- A demo
- A site tour
- A tutorial
- A game
- A survey or poll

☞ **IMPORTANT:**

Using pop-ups to display the latter set of content examples should be done with caution, because they will not always satisfy the criteria above. In particular, it should be noted that “not wanting the user to leave our site” is not a valid reason for using a pop-up, because it does not address the task with which the user is currently engaged.

Usability Considerations

In addition to the previously mentioned concerns regarding pop-up blockers, there are a number of ways in which the use of pop-ups can lead to usability problems:

- Users may fail to notice that content has been displayed in a new window.
- They may become confused when they cannot display previously viewed pages using the browser’s Back button.
- They may be confused if windows are inadvertently closed, minimized, or obscured by other windows. The risk of this problem increases when pop-ups can be launched from other pop-ups.
- Pop-ups that feature the same global navigation (e.g., home page, search, help links) as the parent window can create confusion. If a user clicks an item in the pop-up’s global navigation, will they see the new destination in the parent window, the current pop-up, or a new pop-up?

The bottom line is that pop-ups should only be used when they clearly represent the best and only solution to the problem. Too often designers have become used to using pop-ups when another alternative was available to them. Here are some examples of single-browser approaches to scenarios where a designer may be tempted to use pop-ups:

- Users often compare information. In these cases, present the subject information side-by-side in a table instead of forcing the user to open pop-ups to perform the comparisons manually.
- A Web application that fulfills orders might launch a detailed summary of an order in a separate pop-up. Rather, present the detailed summary as a page loaded in the same browser window.

How to Use Pop-Ups

Pop-ups should only be presented in response to a request by the user—typically, a mouse click (not a mouseover). Pop-ups should *not* be presented as a result of an unrelated action such as leaving a site. This is precisely the kind of pop-up behavior that annoys users and accounts for the popularity of blockers.

In the following sections, we provide further implementation recommendations organized according to three major types of pop-ups:

- Minimal Chrome

  A minimal chrome pop-up window is one with little or no “chrome,” that is, without the additional browser button bar, location bar, or status bar. Because it is not a “standard” browser window, It is typically launched with JavaScript or some other client-interpreted scripting language, and is blocked by most pop-up blockers. See Figure 2.

- Full Browser (launched with JavaScript)

  A full browser pop-up window (launched with JavaScript) is one that appears with the standard browser toolbars. Launching the window with JavaScript (or some other client-interpreted script language) enables the web developer to control the size, location, and other parameters of the pop-up, but it also makes it easier to block with most pop-up blockers. See Figure 3.

- Full Browser (launched without JavaScript).

  A full browser pop-up window (launched without JavaScript) is one that appears with the standard browser toolbars. Because it is *not* launched with JavaScript or some other client-interpreted scripting language, it is only blocked by some of the more extreme pop-up blockers. See Figure 3.

The following decision tree can help you decide which implementation of a pop-up is most appropriate for your situation. Detailed implementation instructions are included in separate sections devoted to the three pop-up types.

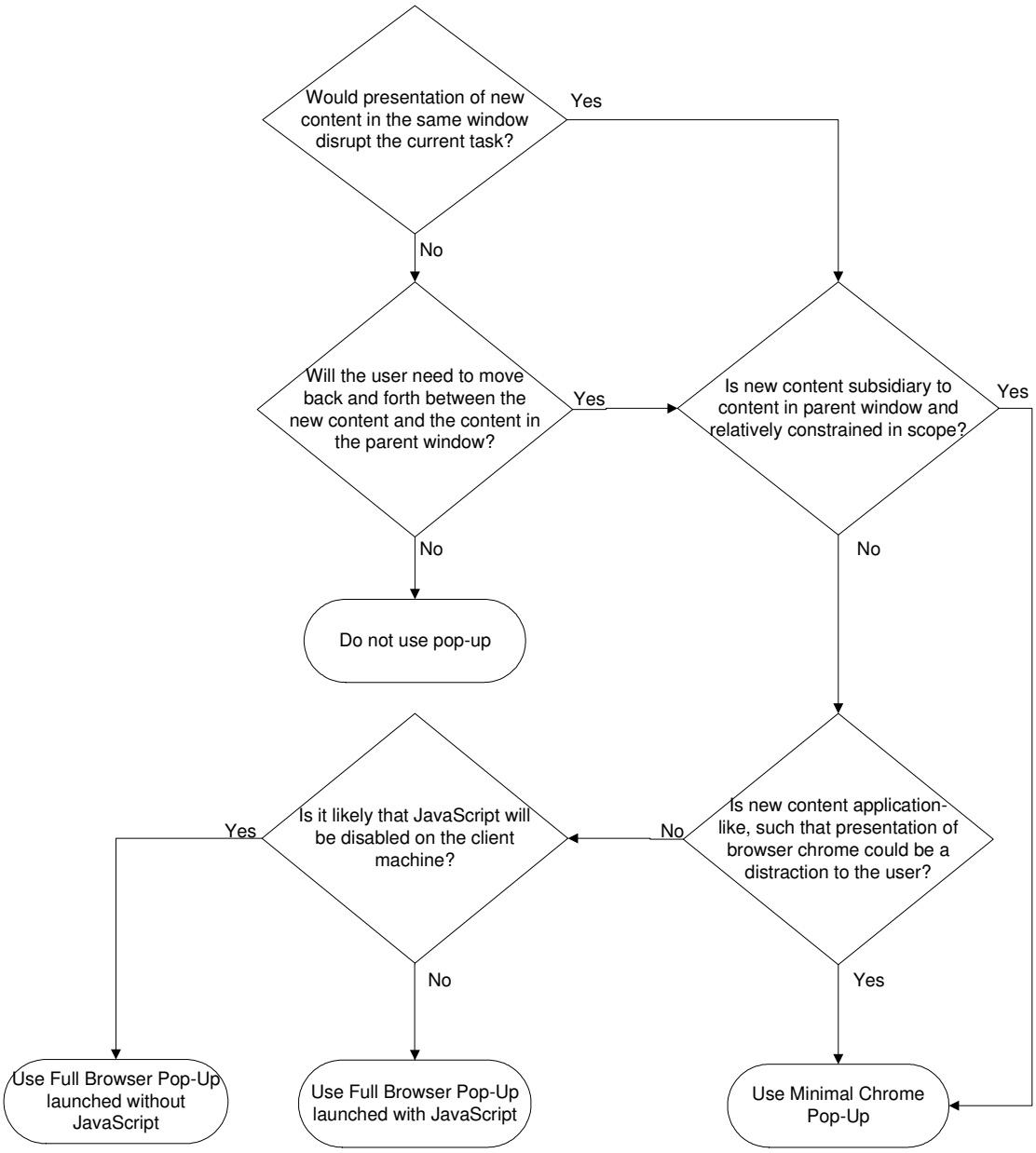


Figure 1. Pop-Up Decision Tree

Minimal Chrome Pop-Up

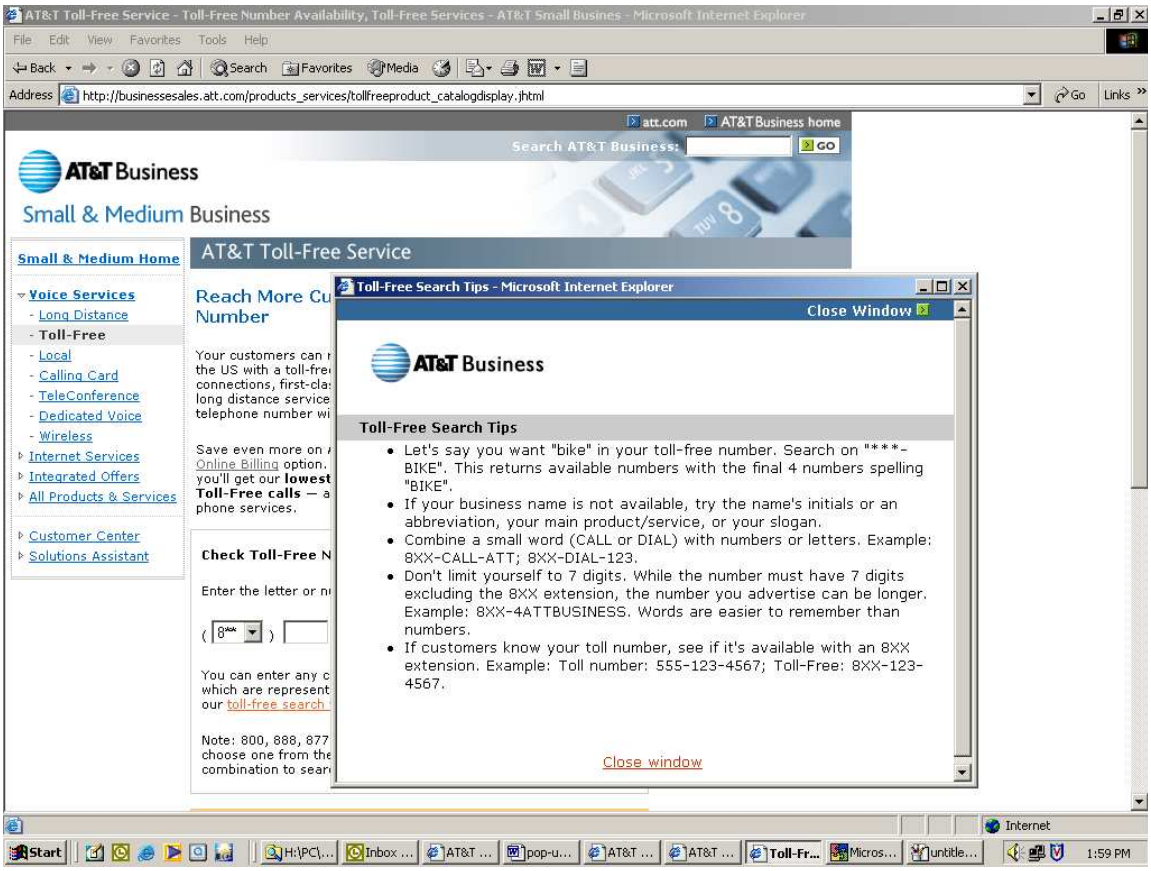


Figure 2. Minimal Chrome Pop-Up

When to use:

Use to display subsidiary and constrained information such as definitions, explanations, instructions, Help information, etc. Also useful for application-like content where the user will need to move back and forth between the new content and the content in the parent window, and browser chrome (the tool bar, menu bar, location bar and status bar) would be a distraction.

Characteristics:

- Launched with JavaScript.
- Should be given focus when launched, so it is not hidden from the user.
- Should be positioned such that the pop-up is fully visible within the user's screen area.
- Content should appear with a title or header.

- Content should not contain links to launch other pop-ups.
- Should be noticeably smaller than the parent window so that the parent window is not hidden.
- Should include a Close button (in addition to the browser's Close Window button).
- Pop-up attributes:
  - Height: No more than 80% of the screen (no more than required by content; may be greater than 80% if content is application-like and user will only need to return to the parent window on an infrequent basis).
  - Width: No more than 80% of the screen (no more than required by content, and should not require horizontal scrolling to view content; may be greater than 80% if content is application-like and user will only need to return to the parent window on an infrequent basis).
  - Location bar: No.
  - Menu bar: No.
  - Resizable: Yes.
  - Scrollbars: Should always be enabled to accommodate variable font sizes selected by users. This will result in scrollbars appearing only when the size of the content exceeds the size of the pop-up.
  - Status bar: No.
  - Tool bar: No.

Limitations:

- Will be blocked by many pop-up blockers.
- Requires that client-side script is enabled.
- Does not display the location (URL) of the content.
- Does not display the browser's Back button, so the user cannot use it.
- Does not display the browser's Print button or menu bar option (but users can reach the Print option available via right-clicking).
- Must be designed to avoid the need for horizontal scrolling.

Full Browser Pop-Up (Launched with JavaScript)

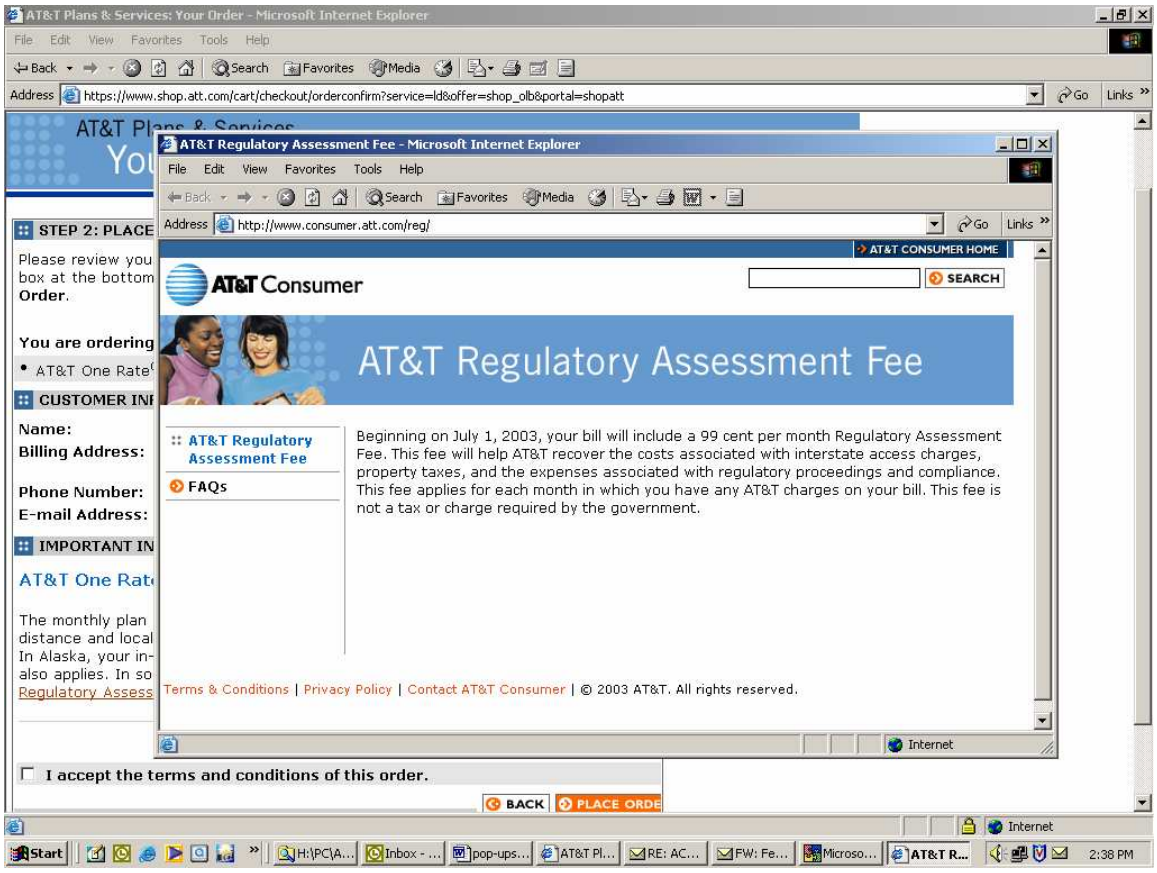


Figure 3. Full Browser Pop-Up

When to use:

Use to display content from an external Web site or another AT&T Web site when presentation of the new content in the same window would disrupt the current task, or the user will need to move back and forth between the new content and the content in the parent window.

Use a Full Browser Pop-Up instead of a Minimal Chrome Pop-Up only when at least one of the following applies:

- There is a significant amount of content to be presented in the pop-up (more than it is practical to reproduce on your Web site), and the content cannot be dynamically reformatted for presentation in a Minimal Chrome Pop-Up.
- It is not practical to reproduce the content on your site because of problems with maintaining the currency of the content in more than one place.

Characteristics:

- Launched with JavaScript.
- Should be given focus when launched, so it is not hidden from the user.
- Should be positioned such that the pop-up is fully visible within the user's screen area.
- Should be sized appropriately for content.
- Content should appear with a title or header.
- Content should not contain links to launch other pop-ups.
- Similar content should be displayed using the same window name, to avoid confusing the user with multiple windows.
- Should be noticeably smaller than parent window so that parent window is not hidden.
- Pop-up attributes:
  - Height: About 80% of the screen.
  - Width: About 80% of the screen.
  - Location bar: Yes.
  - Menu bar: Yes.
  - Resizable: Yes.
  - Scrollbars: Should always be enabled to accommodate variable font sizes selected by users. This will result in scrollbars appearing only when the size of the content exceeds the size of the pop-up.
  - Status bar: Yes.
  - Tool bar: Yes.

Limitations:

- Will be blocked by many pop-up blockers.
- Requires that client-side script is enabled.
- Display of the Links bar cannot be enabled.
- Must be designed to avoid the need for horizontal scrolling.

Full Browser Pop-Up (Launched without JavaScript)

When to use:

Use to display content from an external Web site or another AT&T Web site when presentation of the new content in the same window would disrupt the current task, or the user will need to move back and forth between the new content and the content in the parent window.

Use if there is a significant probability that JavaScript may be disabled on the client machine, or when hiding the parent window is not a concern.

Use a Full Browser Pop-Up instead of a Minimal Chrome Pop-Up only when at least one of the following applies:

- There is a significant amount of content to be presented in the pop-up (more than it is practical to reproduce on your Web site), and the content cannot be dynamically reformatted for presentation in a Minimal Chrome Pop-Up.
- It is not practical to reproduce the content on your site because of problems with maintaining the currency of the content in more than one place.

Characteristics:

- Launched with the target attribute on a form or an anchor element.
  - The recommended method is to give the window a name; for example: target = “newWindowpane”. Windows launched with the same name from the same browser will open in the same window. This will avoid confusing the user with multiple windows.
  - Another method (not preferred) is to specify target = “_blank”, which will open a nameless new window.
  - Avoid using “_search” and “_media”. On Windows machines, these are reserved for other functions.
- Content should not contain links to launch other pop-ups.
- Pop-up attributes cannot be specified.

Limitations:

- Size, focus, positioning, and chrome cannot be specified when the pop-up is launched.

When Not to Use Pop-Ups

To ensure that AT&T customers have a rewarding (not annoying) user experience when they visit our sites, designers should avoid using pop-ups in the following situations.

- Without a request from the user (typically, a mouse click). This includes the following:
  - When the document in the parent window loads or unloads.
  - When the parent window is closed.
  - When the parent window loses or gains focus.
  - When the setTimeout method is used to open a new window after a set amount of time.
  - When a user positions the cursor on or moves the cursor away from a particular element.
- When users are starting a new task and leaving the old task or content behind (e.g., signing up for a service).
- When the sole reason for the use of a pop-up is “not wanting the user to leave our site.”

Standard Icon and Messaging

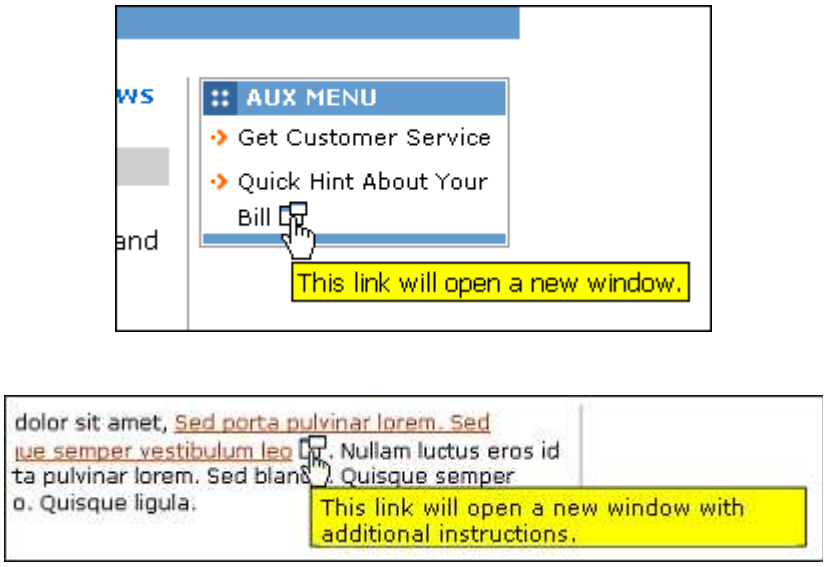
Not all pop-up blockers alert their users that a pop-up is being blocked. Some that do alert their users also allow them to control the intrusiveness of the notification (e.g. you can turn off an auditory alert). In any case, the notification is, of course, generic (it doesn’t tell you the nature of the content) and it is always after the fact.

For all these reasons, we believe it advisable for AT&T sites to take a more user-friendly and proactive approach to indicating the presence of pop-up content. This means alerting users whenever a click will result in a pop-up window.

For text links, TIDD recommends placing a small icon immediately after a link that launches a pop-up window. The ALT text asociated with the icon should read “This link will open a new window.” And if it is not already clear from the link text on the Web page, the ALT text should also explicitly indicate the nature of the content contained in the pop-up.

The following illustrations show examples of TIDD’s recommended icon and ALT text.

TIDD has developed this icon in two sizes--for standalone and for in-line links. Both are available at http://tidd.att.com/tidd/services/pop-up.html



.

Figure 4. Recommended Icon Use

If, for whatever reason, the pop-up icon cannot be used with a text link, the TITLE attribute on the link should alert the user to the pop-up and its contents as explained above.

If the pop-up will be spawned by clicking on another image, you do not need to use the pop-up icon. Simply use the ALT attribute on the image to alert the user to the pop-up and its contents as explained above.

Pop-Up Code and Best Practices

The Pop-Up Function

When coding a pop-up function, the following practices should be followed:

- Consider the width and height of the user's screen. The window size you specify should not exceed the size of the screen. A good practice is to determine the size of the user's screen and then size the window to 80% of that. If you're launching a window with full browser chrome, subtract an additional 120px.
- Omit window placement.

NOTE:
The function below checks for screen size and allows you to open a browser with full chrome if you specify a 1 for the Z value.

Table 1. Pop-Up Code Example

```
function newPop(A,B,X,Y,Z) { // url, name, width, height, full = 1

// width and height
windowWidth = (X < screen.width)?X:(screen.width * .8);
windowHeightTemp = (Y < screen.height)?Y:(screen.height * .8);
windowHeight = (Z == 1)?windowHeightTemp - 120:windowHeightTemp;

// specifications
winSpecsMini = "scrollbars=yes,resizable=yes,status=no,location=no,menubar=no,"+
"toolbar=no,width="+windowWidth+",height="+windowHeight;

winSpecsFull = "scrollbars=yes,resizable=yes,status=yes,location=yes,menubar=yes,"+
"toolbar=yes,width="+windowWidth+",height="+windowHeight;

winSpecs = (Z == 1)?winSpecsFull:winSpecsMini;

// launch
newWindow = window.open(A,B,winSpecs);
}
```

The Anchor Element

When launching a pop-up with a click the following practices should be followed:

- Use the pop-up icon as described earlier and make the icon *and* the associated text clickable.
- Use the ALT attribute associated with the icon to alert the user to the pop-up and to describe its contents if that is not already clear from the context. See Figure 4. Do not use the TITLE attribute of the text link for this purpose, unless the icon cannot be used.
- The onclick attribute of the <a> element should call the pop-up function, as well as a “return false.”
- The href attribute of the <a> element should point to the URL that would open in the pop-up window. This fail-safes the pop-up for people who have JavaScript disabled as well as when pop-up killers kill the onclick event.
- The target attribute of the <a> element should call the same name that the pop-up window is named.

Table 2. Pop-Up Example 2

Full Pop up
Mini Pop up

Terms and Definitions

Client-Side Scripting

- Active scripting: Microsoft's general term for all client-side technologies, including VBScript and JScript.
- Client-side scripting: A generic term for scripts such as JavaScript, ECMAScript, VBScript and other scripting languages that are interpreted by applications (such as a browser) on the client machine.
- JavaScript: A script language developed by Netscape used to create dynamic effects on Web pages, including pop-up windows, rollovers, and form validation. JavaScript is interpreted by the browser (client). It is the basis upon which the ECMAScript standard (now endorsed by both Netscape and Microsoft) is built.
- JScript: A script language developed by Microsoft used to create dynamic effects on Web pages. It now conforms to the ECMAScript standard. Equivalent in most ways to Netscape's more popular JavaScript.
- VBScript: A script language developed by Microsoft as part of its Visual Basic programming language. It provides functionality similar to JavaScript, but it will only work with Microsoft's Internet Explorer browser.

Related Technologies

- ALT/TITLE attribute: Attributes of the IMG element primarily used to describe an image on a Web page for people who are unable to view the image themselves. In many browsers, the ALT/TITLE attribute displays as a tooltip when the pointer is placed over it. Also referred to colloquially as an "ALT tag." The TITLE attribute can also serve the same purpose as an ALT attribute for images (though if you use TITLE, you should also use an ALT for older user agents).
- DHTML: A collective term for a combination of HTML and other Web technologies, including JavaScript, Cascading Style Sheets, and the Document Object Model (DOM), all of which enable great interactivity on a Web page without involving requests to the server.
- HTTP-EQUIV REDIRECT META ELEMENT: An HTML meta element used to redirect customers to new pages, or to launch executables.

Windows

- Auxiliary browser window: A window that is integrated into the Web browser; e.g., Internet Explorer's Search, Media, and History windows.
- Browser window: Any window spawned by the Web browser executable.
- Daughter window: A window spawned by a parent window, usually as a result of JavaScript or some other script language on the parent page which is then interpreted by the browser.

- Fatherless window: A window spawned independently of a Web browser, typically by a third-party application, using the browser executable or dll.
- Minimal chrome window: A daughter window with very little “chrome” (the tool bar, menu bar, location bar and status bar); chrome may be removed via client-side scripting.
- New window: A browser window launched after the initial start-up of the browser application, including daughter windows, fatherless windows, and windows spawned via CTRL-N or other means.
- Parent window. Any window that spawns another window.
- Pop-up window: A generic term for any window spawned by another window as well as fatherless windows.
- Pop-under window: A pop-up window that “pops up” beneath (instead of on top of) the parent window.
- Windows Messenger Service: A network service available to Windows XP and 2000 users, which allows them to send messages to other users on a network (the Internet is a network).