Designing, Localizing, and Customizing Web-based Embedded Assistance

Scott DeLoach
User First Services
scott@userfirst.net

Abstract

Embedded user assistance (UA) is any type of instructional or conceptual information that appears within a software application window. It includes both embedded help (help topics that appear within the application), field labels, and page overviews. This paper outlines the benefits of embedded UA, four techniques for providing embedded UA, and embedded UA solutions for localization and customization issues.

Keywords: *user assistance, online help, embedded help, web-based help, localization, customization*

Overview

An external help system is an excellent way to provide in-depth conceptual information and procedures. However, external help systems often fail to meet users' needs because they are a passive, external resource. With an external help system, the user has to realize they need assistance, open the help, locate the relevant information, and return their focus to the application. For many questions, this process feels like too much effort for the perceived benefit. As a result, users often do not use external help systems because they do not see a sufficient return on investment (ROI) [1]. The basic goal of online help is to help the user and get them back on task as quickly as possible [2, 3, 4, 5]. External help systems often fail to meet this goal.

Most user questions should be answered at the point of use (or rather, the point of confusion) using embedded assistance. To the user, embedded assistance provides a large benefit with practically no effort. In fact, most users cannot distinguish between embedded user assistance and the application it supports [6]. After using embedded assistance, users are much more likely to click on help links and spend more time using the external help system.

Four Techniques

There are four general techniques for providing embedded UA on the Web:

- overviews
- field labels
- popup windows
- dedicated UA panels

The following sections describe how each of these techniques can be used to answer specific types of user questions.

Conceptual Questions: Page and Section Overviews

Page overviews can be used to introduce an application page and to reassure users that they are on the correct page. If a page is divided into multiple sections, each section may have its own overview.

The Neiman Marcus website (neimanmarcus.com) provides a rarely-seen but very effective approach to page overviews. The search page, as seen in figure 1, provides a description of the search tool with callouts and descriptions directly below the search area. This information is presented as a large graphic, so the user cannot click inside the instructional fields.

The Neiman Marcus website provides an excellent overview of its search tool (Figure 1). However, one improvement might be to emphasize the difference between the actual search area and the instructional search area. Using a different background color or enclosing the UA area in a box might reduce user confusion.

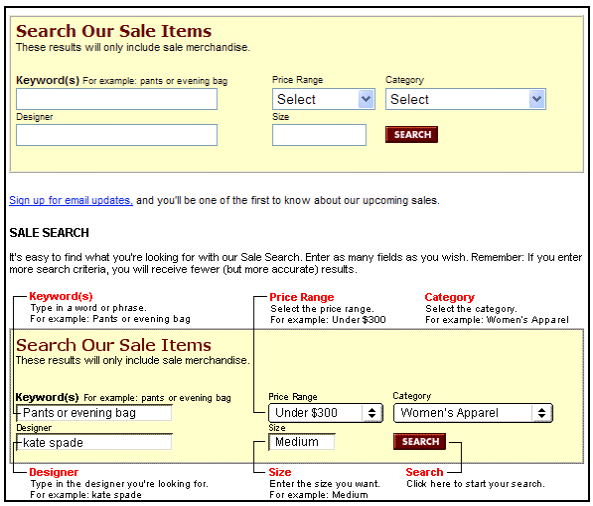

Figure 1. a page overview (neimanmarcus.com)

Basic Questions: Field Labels and Tips

The most common question that users have is "What do I type into this field?". Unfortunately, poorly written field descriptions are a major reason that users think that help systems are not useful. It only takes a few field descriptions like "First Name – Enter your first name" to convince users that opening the help is a waste of time.

In most cases, the user only has a basic question such as, "What is the maximum number of characters I can type into this field?". These questions should be answered as part of the field label. In figure 2, the password field label informs the user that their password must be at least six characters without spaces.

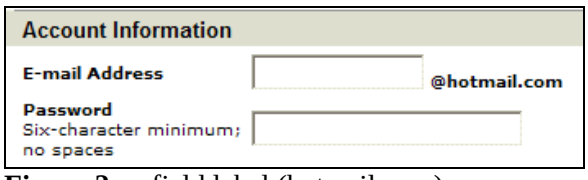

Figure 2. a field label (hotmail.com)

Exceptions, Photos, and Related Fields: Popup Windows

Obviously, not all field-level questions can be answered with field labels. These questions are often based on exceptions such as "What if I don't have a middle name?" Other questions, such as "How do these fields relate to each other?" or "What is the difference between these two items?", are best answered using a table, list, or photograph.

The Dodge website (dodge.com) uses embedded DHTML windows to explain related fields (Figure 3). These grouped UA topics allow the user to answer multiple questions at the same time, and they make it much easier to see how different fields are related to each other.

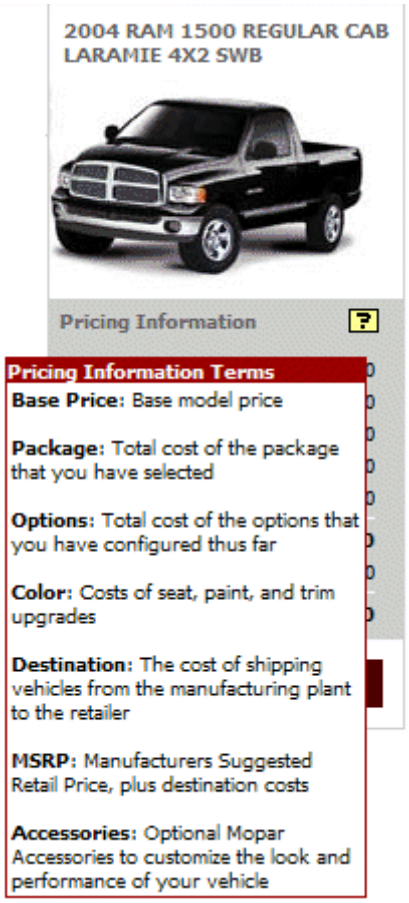

Figure 3. a related fields popup box (dodge.com)

The Dodge website also uses popup windows to provide photographs of truck parts and accessories (Figure 4). These popups allow the user to verify that they have selected the right item. In figure 4, the user has clicked on the "Next Generation 4.7L Magnum® Engine" link to open the popup window. This website is designed to help people purchase cars and trucks, and the photographs do an

excellent job of education the user while marketing the products.

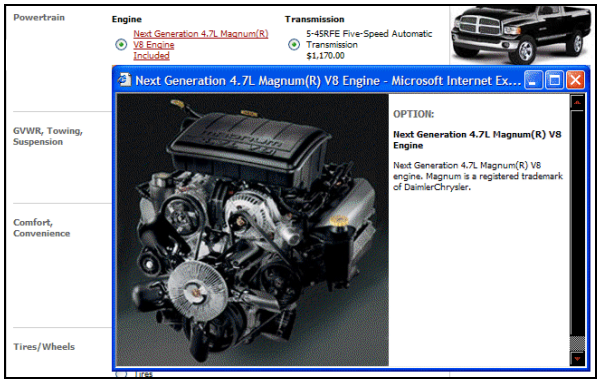

Figure 4. a graphical popup window (dodge.com)

Advanced Questions: UA Panels

UA panels provide enough space to answer advanced questions, such as "Why do you need this information?" They normally appear and disappear when the user clicks on a link. However, some applications provide dedicated UA panels.

In figure 5, the Alamo website (alamo.com), includes a dedicated UA panel to assist customers as they rent a car. The UA panel's topics focus on why the user should provide the requested information, how this information is used, and how to handle special cases. The UA panel is large enough to answer numerous questions, and it can link to help topics in an external help window if needed.

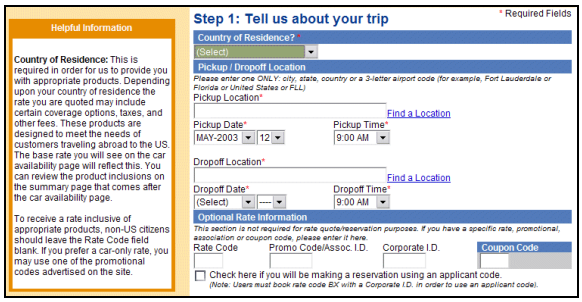

Figure 5. a dedicated UA panel (alamo.com)

Since the UA panel is always on the screen, the Alamo website uses JavaScript to automatically describe the field that has focus. The user never has to ask for help, and relevant assistance is always available on the screen.

In figure 6, The Bank of America website (bankofamerica.com) uses a dedicated UA panel to answer common user questions. Section-specific questions and short answers are provided within sections (for example, "Why aren't all of my payees listed here?"). By answering common user questions in the application, the Bank of America site increases user success.

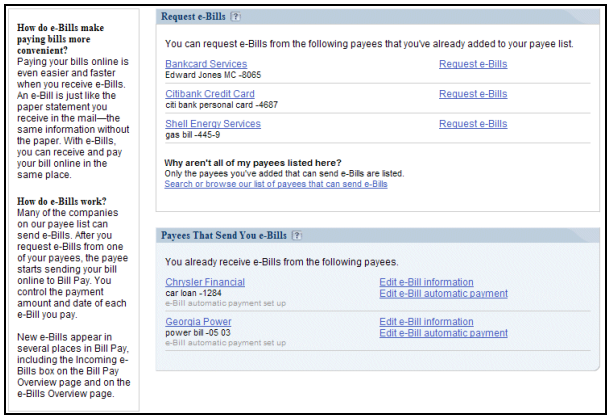

Figure 6. a dedicated UA panel (bankofamerica.com)

Supporting Multiple Languages

The most effective method of supporting multiple languages is storing the UA text in a database. Table 1 shows a simple approach for providing field-level UA in English and Spanish. The UA text for each language is stored in a separate column. The application finds the field using the fieldID column, selects the correct language, and displays the UA text in the application.

Table 1. Database columns for English and Spanish UA topics

FieldID	UAEnglish	UASpanish
ProjNum	The project number is used by accounting to track projects. It is automatically generated.	El número del proyecto es utilizado dando cuenta para rastrear proyectos. Se engendra automáticamente.
StartDate	The start date is the date of the project kick-off meeting. The format is MM/DD/YYYY.	La fecha del comienzo es la fecha de la reunión del comienzo del proyecto. El formato es MM/DD/YYYY.

Supporting User Preferences

Many users prefer to adjust the application to better meet their needs. For example, users may want to control a UA panel's location, size, or the type of information it provides. For example, a user might prefer a UA panel that appears on the right-hand side, is 200pixels wide, and displays information for expert users.

There are two popular techniques for remembering user preferences. The first technique is to store them in a browser cookie on the user's PC. The second technique is to store them in a database on the application's Web server.

Using a Cookie Cookies are small packets of information that are stored on a user's PC in a text file. If the user's PC has a cookie for a website, the website can read and edit the cookie's information as needed. Some browsers or browser security settings do not allow cookies, so cookies do not work for all users. In addition, cookies are stored by PC not by user. If the user uses a different PC, their preferences would not be available.

Using a Database Using a database to store preferences can be more complicated than using cookies, but it offers a number of advantages. A database is not a security risk, so it should work with almost any browser. The database approach is also user-specific, so the user can use any PC and still see their preferences. The downside of the database approach is that it can increase the server load and reduce performance.

Supporting Content Customization

One of the biggest challenges facing technical communicators is documenting an application that can be customized by the user. Most Web-based applications can be customized at either a site-wide or user level (or both). Technical communicators rarely know how the application has been customized, so they are forced to provide online help that describes the base system. In addition, customers often have specific information that they would like to add to the help. In most online help systems, this is not possible.

The best solution to these issues is to store the UA text in a database rather than in a help system and to use variables for field and screen names. For example if an application has a field labeled "Project Number," the technical communicator might write the following UA text (as seen in Table 1):

"The project number is used by accounting to track projects. It is automatically generated."

If the user changes the name of this field to "ID," the UA text will no longer match the application. The solution is to use a variable for the field name rather than static text. Many Web-based technologies, such as Active server pages (ASP), can read variables from a database. If the application can be customized, it must have a way of matching the field label to an internal ID. The same approach can be used to ensure that the UA text uses the correct label text (Table 2).

Table 2. Database columns for a field's ID and label

FieldID	**FieldLabel**
ProjNum	Project Number

In this example, the UA text can use the FieldID to find and display the correct field label.

If the UA text needs to be customized, we can extend this approach and store the UA text in the database, as seen in Table 2.

Table 2. Database columns for a field's ID, label, default UA text, and customized UA text

FieldID	**FieldLabel**	**UAText**	**UACustom**
ProjNum	Project Number	The project number is used by accounting to track projects. It is auto-generated.	The project number must begin with the office's code. For example, Atlanta should use the "ATL" prefix.

The sample database presented in Table 2 includes two UA text columns: UAText and UACustom. When the application is delivered to the client, the UACustom column should be empty. When the user customizes the UA text, the customized text is stored in the UACustom column. If the UACustom cell is empty, the the application displays the default text from the UAText cell. If the

UACustom cell is not empty, the application displays the customized UA text.

This approach has two advantages. First, the user never overwrites the default description, so they can always recover the default text if needed. Second, the application never overwrites the user's customizations. If the help is updated, the changes are stored in the UAText column.

Conclusion

User assistance should appear at the point of confusion rather than buried in a help topic. Ideally, UA assistance should include embedded UA (page overviews, field labels, and embedded help) and an external help system. The user's type of question and their experience level can be used to determine which approach will best meet the users' needs [7, 8]

Embedded UA also provides flexible solutions to localization, personalization, and customization challenges that will become more complex as Web-based products mature and attempt to reach a more global audience.

References

[1] J. Spool and T. Scanlon. "Making online information usable," in *HyperViews* vol. 3 no. 4, pp. 5-7, 1996.

[2] T. M. Duffy, B. Mehlenbacher, and J. E. Palmer. *Online Help: Design and Evaluation*. Norwood, NJ: Ablex Publishing Corporation, 1992.

[3] J. T. Hackos and D. M. Stevens. *Standards for Online Communication*. New York, NY: John Wiley & Sons, Inc., 1997.

[4] W. Horton. *Designing and Writing Online Documentation: Hypermedia for Self- Supporting Products*. New York, NY: John Wiley & Sons, Inc., 1994.

[5] H. N. Shirk. "Technical writers as computer scientists: The challenges of online documentation," in *Text, Context, and Hypertext: Writing with and for the Computer*. E. Barrett, Ed. Cambridge, MA: MIT Press, 1988, pp. 311-325.

[6] A. Sellen and A. Nichol. "Building user-centered online help," in *The Art of Human-computer Interface Design*. B. Laurel, Ed. Menlo Park, CA: Addison-Wesley, 1990, pp. 143-153.

[7] S. DeLoach. "Standards for Online Help," in *tekom Jahrestagung 2004*, Wiesbaden, Germany: Gesellschaft fuer technische Kommunikation e.V. 2004, pp. 35-37.

[8] S. DeLoach. "User-centered Design for Informational Websites," in *Proceedings of Technical Communication over Internet Symposium*, Taipei, Taiwan: Academia Sinica 2001, pp. 32-36.

About the Author

Scott DeLoach is a founding partner of User First Services, an Atlanta-based consulting company that specializes in designing and creating user assistance. Over the last ten years, Scott has presented over 50 papers on interface design, usability, WBT development, JavaScript coding, and online help development at conferences across the US and Canada and around the world. He has been developing online Help systems since 1992 and has been coding in JavaScript since 1997. Scott holds a Master's Degree in Technical and Scientific Communication from Miami University.