

From: Web Site Usability Handbook (2000)

By: Mark Pearrow **CHAPTER**

6

HEURISTIC EVALUATION

Any sufficiently advanced bug is indistinguishable from a feature.

—Rich Kulawiec

WHAT IS A HEURISTIC?

One dictionary definition of the word *heuristic* is as follows:

Heuristic, n.: Involving or serving as an aid to learning, discovery, or problem solving by experimental and especially trial-and-error methods (heuristic techniques; a heuristic assumption); also, of or relating to exploratory problem-solving techniques that utilize self-educating techniques (as in the evaluation of feedback) to improve performance (a heuristic computer program).¹

If that definition still sounds a little opaque to you, think of it a different way: A heuristic is a rule of thumb. Heuristic evaluation, then, is the act of estimating the state of usability of a Web site by applying well-known rules of thumb to the site and deriving a score for the site based on how closely it fulfills the requirements of these rules.

¹This definition was found on www.m-w.com.

Normally, the way a heuristic evaluation works is that two to five “expert evaluators” examine a site, taking special note of the way the site adheres to or violates the heuristic list items. To avoid peer bias, each evaluator does this work in isolation (not in the same room at the same time as the other evaluators, that is).² Each evaluator comes up with a list of usability problems that he or she finds; in almost every case, the usability problem exists because one or more of the heuristics have been violated. After the evaluation session, the evaluators share their results and begin to form recommendations for change.³



The main selling point of heuristic evaluation is that it can be done quickly, often be done on a tight budget. Since most companies want virtually instant turnaround on Web-based projects, this tool is a perfect fit for helter-skelter schedules. You can produce a relatively comprehensive heuristic evaluation in two to three days.

²Nor on a remote desert island, although sometimes that wouldn't be so bad!

³You can read more about this process at www.useit.com/papers/heuristic/heuristic_evaluation.html.

TEN USABILITY HEURISTICS

The pioneer of heuristic evaluation, Jakob Nielsen, developed a list of 10 heuristics in collaboration with his colleague Rolf Molich in 1990. Needless to say, this list of heuristics predates the popularity of the World Wide Web. Much of the list is still quite pertinent in the context of the Web, however; some items are a little dated. Additionally, the list lacks new heuristics that should probably be included. Much has been written on this topic. For now, we discuss the original 10 heuristics; later we'll look at some new heuristics that can be applied. The list that makes up the first portion of this chapter comes from Nielsen's article, *Ten Usability Heuristics*.⁴

Heuristic #1: Visibility of System Status

This simple principle says that the user should always know what state the Web site is in at any given moment. This rule is most applicable to transactional Web pages, such as electronic commerce forms. The user needs to know where he or she is in terms of the ordering process (“Is my order complete?” “Did I successfully cancel that order?” “Am I shopping again?”).

Additionally, users need to know where they are in the Web site, especially when the main site contains many microsites. This can be especially challenging when many different groups are responsible for the upkeep of the microsites (see heuristic #4 for more on this topic). A clearly labeled system for navigation helps users know where they are.

This heuristic also concerns the all-important idea of feedback: You should always provide the user with some way of knowing what is going on. In the early days of Web transactions, many users were billed multiple times for a charge that they had only authorized once. The reason? Slow Web servers and a lack of system status visibility led users to believe that their transactions had not worked, so they clicked “Submit”

⁴You can find this article at www.useit.com/papers/heuristic/heuristic_list.html.

again. And again. Clearly, you must provide some sort of feedback mechanism to let users know what's happening between transactions.

Very recently I was browsing a very big computer company's site and found myself in the middle of a transaction. This multibillion-dollar company provided me with a system status indicator—a little animated GIF that flashed like a progress bar. It was cheap, it was simple, but it let me know that something was happening, and that's all I needed to feel secure that my transaction was being handled.

Heuristic #2: Match the System to the Real World

This heuristic, as designed by Nielsen and Molich, is actually two heuristics in one. Here's what their paper says about this rule:

The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

The two parts to this heuristic are as follows:

- Use the natural language of the user, not jargon or system terms.
- Follow real-world conventions; provide natural mappings.

The first concept is critical. I have seen hundreds of usability problems come from violating this principle. What is meant by employing the user's natural language is avoiding jargon or terms that would otherwise expose the inner workings of the system. This also means using the terms that the user would normally use to describe an action, a piece of information, and so on.

One of the most famous violations of this principle involves the Northeastern University (NEU) Online learning system. NEU had decided to put many of its courses in an electronic format for delivery via the Web. The company NEU employed was fairly new to the area of user interface design.

In the company's design for the students' view of the course material, the student could navigate from one week's lecture to the next (each week's material was called a *module* via the main course menu. This part almost made sense.

However, when the student got deep into the lesson material, the button that would bring the student back to the main course menu was marked "Exit." No one was able to return to the course menu, because Exit meant something very different to everyone who was using the system: it meant to, well, exit—to leave the system, to no longer use it.

Fortunately, the crack expert designers and engineers put a team on the case of the mislabeled button. After considerable research, they came up with a new label to solve the problem. Instead of Exit, they labeled the button "Return." (See Figure 6.1.)

Another instance of violation of this heuristic involved a client of mine, a large bank. The users of the bank's Web site were unable to find information that was lumped under a particular heading. After doing a usability study with real users, we found that the users did not understand the meaning of the name of the heading. Hence, they would not look at information under the confounding heading unless they had scoured the rest of the site first.

We later found that the name of the heading was originally something much more sensible to the users, but that it had been changed by a vice president because the word didn't sound "upscale" enough. Our job then became to convince the VP that his ego was less important than millions of dollars in business.

The moral to the story is that being too clever can often lead to stranded, confused, or angry customers. Being clever is almost never a good idea, even if you are in the business of being clever.

Likewise, you should eliminate all industry jargon from your site so that users must interface only with terms they know and use. Although this example overlaps a little with heuristic #9 below, error messages should not contain diagnostic, debugging codes used by programmers. For example, you should not print a message that says, "Error: The server encoun-



TMG5382

Java Script

Module 1 Course Materials

Page 1 of 1

Week 1 Course Materials

- [Lecture Notes](#)
- [Handouts](#)
 - [Handout 1: A Comparison of JavaScript and Java](#)
- [Exercise Code](#)
 - [Exercise 1-1](#)
 - [HTML Template](#)
 - [Exercise 1-2](#)
 - [Exercise 1-3](#)
 - [Exercise 1-5](#)
- [Definition Terms](#)
 - arithmetic operators
 - assignment operator
 - code
 - comment
 - concatenation operator
 - data types
 - function
 - hard-code
 - HTML skeleton
 - interpreter
 - LiveScript
 - loose typing
 - operators
 - overloaded
 - self-documenting
 - string
 - strong typing
 - var

FIGURE 6.1 An example of a designer not matching a control to a user's own natural language. This is the vastly improved version of this system. © 2000 Northeastern University Online.

tered server error 9812791. The SQL statement was malformed. Please mail the following stack trace to your system administrator.”

Instead, you should provide a user-friendly version of the error message, and instead of making the user report the error, you should write robust code that can self-report these kinds of electronic mishaps!⁵

⁵After all, there are refrigerators now that can e-mail a service technician when service is needed. The Web should be at least as useful as a refrigerator.

One other note on language: don't use machine-generated translations for your site. Although translation tools like the one found at *babelfish.altavista.com*⁶ are hilarious and lots of fun to play with, in the wrong hands they can yield some questionable translations. It's also not a good idea to try to translate a page with only a dictionary if you don't speak the source and target languages. In defense of the *babelfish* site, Systran has done an excellent job of making an automated machine translator of human languages. However, until computers possess a similar type of "understanding" of language as humans use it, you may still favor a human translator over a machine.

The second part to this dual heuristic, "Match the system to the real world," refers to a concept called *natural mappings*, covered thoroughly by Donald A. Norman in his book *The Design of Everyday Things*.⁷ A natural mapping is the approximation of the "real world" by a control interface.

Some examples of natural mappings include your car's steering wheel. Turning the wheel to the left causes the car to turn to the left, and so on. Another natural mapping is the console of a VCR or tape recorder. Pressing the Rewind button, which points in the backward direction,⁸ causes the tape to run backward.

Once an interface such as the VCR or tape player becomes ubiquitous, it makes sense to base new interfaces on the old to help leverage the

⁶I have been in tears before after playing with this site. Here's what to do: Enter a sentence in English, then translate it into French. Then take the French translation, and translate it back into English. Next, take that English translation and translate it into German. Then, back to English.

⁷This is one book that you absolutely must read. It's both entertaining and informative; it's a fairly quick read, and it will forever change the way you look at everyday objects, including Web pages.

⁸It points backward to Westerners who read from left to right, that is. Note that the direction may not intrinsically make sense to people who read primarily Arabic, Hebrew, or any other language written from right to left. However, through use, even contradictory controls can be learned. But this is a bad idea generally.

user's existing schema.⁹ This helps reduce the amount of retraining that needs to be done. See heuristic #4 for more detail.

Heuristic #3: User Control and Freedom

This heuristic probably evolved from too many dead-end, noncancelable dialog boxes. The basic premise of this heuristic is that you should always do the following:

- Provide clearly marked exits
- Support undo and redo transactions
- Make it harder to perform irreversible actions

What this means in the context of the Web is that for any given transaction state that a user might be in on your site (in the middle of a purchase, for example), there must be a way for the user to gracefully back out that doesn't involve crashing the browser.

Additionally, users should be able to undo and redo actions. This can be a bit tricky to support via Web programming, but it is a very worthwhile feature. An example of undo might be allowing users to take an item out of their shopping carts after they put it there ("Gee . . . do I really need that DVD of the movie *Tron*?"). You should also allow redo ("Now that I think about it, yes, I *do* need that *Tron* DVD!")

At the same time, you should use the principle of constraint to make it hard to commit irreversible actions. If the effects of any action are in some fashion irreversible, you need to go to great lengths to let the user know about it in advance.

Additionally, you should know when to use *modal dialog boxes*¹⁰—that is, dialog boxes that block the user's ability to do anything else until the dialog box has been acknowledged. Be warned that such widgets can be very frustrating to a novice user, since he or she may lose track of the

⁹A *schema*, to be brief, is a mental model of something.

¹⁰For an example of creating a modal dialog box in JavaScript, see developer.netscape.com/viewsource/goodman_modal/goodman_modal.html.

window that is blocking them, causing them to become frustrated when their attempts to click or make anything else work fail. Modal dialogs are good when your Web application presents a finite task, such as getting the user's confirmation of a potentially irreversible act¹¹ or the submission of a large amount of data.

All of this harks back to the fact that users of computer systems generally feel intimidated and isolated,¹² and many people fear getting trapped in programs and interfaces that cause them to “bungle” something. Providing the user with many ways to reverse actions and providing restraints to deter potentially irreversible actions gives the user a safety net. A more relaxed user is a happier user.

Heuristic #4: Consistency and Standards

Imagine that you have just purchased a new car. To open the doors on the car, you learn that you have to push in on three separate buttons while turning a knob on the door a quarter-turn counterclockwise. Once inside the car, you discover (after consulting the user manual) that the ignition is in the passenger side floorboard and that you activate it by putting the key in and turning it while pushing a knob on the gearshift.

Once you're up and running, the air conditioning and heat are controlled by separate consoles, but the fan has a single control knob. To increase the amount of heat, the heat slider must be pushed to the left, while A/C is increased by pushing its individual slider to the right. The fan knob must be turned in the opposite direction of the slider to actuate airflow.

Sound like fun? No? Well, what if you knew that this particular design allowed you to have a car that costs half the price of a comparable

¹¹Remember, there should be no irreversible acts in the first place. However, where technological limitations force your hand, you need to give the user a chance to opt out before the action is committed.

¹²Not you, of course! You're an expert computer user.

ergonomic car? Still wouldn't buy it? Rightfully so. Such a car would not only be a disadvantage to the poor driver, it would be a menace to society because of all the accidents it would cause.

The reason that few automobile makers brazenly disregard common control system design is that it would severely work against them. The fact that the industry has settled on some consistent standards means that users (drivers) have little relearning to do when they get a new car.

Software developers have settled on some similar standards (partially because operating system developers have facilitated doing so), which enable users to quickly "find their way around" on new software.

On the other hand, some software vendors have chosen to go in the other direction entirely. Take, for example, the program Kai's Power Tools by MetaCreations. This program has an interface that looks much more like the dashboard of an alien cruise ship than a computer program! The creator of the program threw traditional standards out the window (pun intended) to make a bold, new, "fun" interface. The program is aimed at graphic designers who want to have fun while they work.

Would this approach work with tax software? Just imagine . . .

The truth of the matter is that standards allow users to survive in an otherwise churning sea of rapidly changing technology. Note that there isn't a well-known, widely adopted standard for Web page layout. The Web applications that we create are usually not based on a style guide or any kind of accepted norm. Ad hoc standards are starting to emerge as dominant organizations gain recognition. Again, think Amazon.com. Its model for shopping online has become a de facto standard without the aid of a publicly available style guide.

The notion of widespread standards for Web style and implementation has gotten the attention of at least some people. The Web Standards Project (WSP), whose site can be found at www.webstandards.org, is one group that is pushing for such standards. Of course, the best-known Web standards organization is the World Wide Web Consortium (W3C, whose URL still confounds me: www.w3.org). The W3C steers

the development of most of the critical Web standards, such as HTML, Extensible Markup Language (XML), DOM, and many others. However, the W3C does not usually concern itself with higher-level issues such as interface layout, style (not CSS style sheets, but rather tasteful implementation), or any other such areas. For this reason, other external standards organizations might be necessary to accomplish the goal of standards organization in these outlying areas.

Heuristic #5: Error Prevention

No matter how well you document your Web applications, no matter how nice your search engine is, no matter how great your online help system can be, none of these items holds a candle to simply making your site easy to understand and use.

The same outfit that brought us the now-famous Exit button also produced a hefty user guide for its Web site-based content authoring tool. The tool was nonintuitive enough to merit some lengthy explanation. If you need to produce a manual so that people can use your Web site, something is really wrong!

Furthermore, it is much more desirable to simply prevent error conditions than to make lavish and information-heavy error messages. In the case of the Web, you should include automated spiders in your bag of site administration tools to ferret out broken links. As charming and warm a welcome as it is, "Error 404" would be a welcome omission from almost every user's world.

Heuristic #6: Recognition Rather Than Recall

Donald Norman's *The Design of Everyday Things* discusses an experiment involving total recall vs. recognition. A group of people was asked to draw a U.S. one-cent piece (a penny) and to include all the features they could remember. Try this task yourself! Although no one got the penny quite right, all of the participants could pick out a penny from a fistful of change. How?

Humans are rarely required to remember all the features of any object by rote memory. Almost always, the world offers cues that help the human discriminate among various stimuli. As Dr. Norman points out, critical information can exist in one of two places:

- In the head (memorized information)
- In the world (labels, “obvious” information, and so on)

Part of the success of an interface rests on how well the information needed to operate it is distributed between these two options. Via recall (knowledge in the head), users are required to “just know” something. This is a good thing in small doses, but the more information you can delegate to the interface—without compromising its coherence and simplicity—the better.

A common way to delegate information to the interface is through labels. If a user must recognize many different icon buttons to perform a task, it’s best if there is a legend describing the function of those buttons, or some other alternative to rote memorization. Note that it is very easy to clutter an interface with too many labels; a type of interface parsimony is the answer.

Heuristic #7: Flexibility and Efficiency of Use

You can’t please everyone all the time. It’s a fact. However, you should endeavor to make your Web site customizable, if that’s feasible, so that users can make things the way they feel is most efficient for them. For example, if you have a search engine on your site, you should initially expose only the most minimal set of input elements for conducting a search. You should also provide an advanced search feature for intermediate and power users as an option.

It’s generally considered good design to make default interfaces as simple as the task will allow, with more powerful (and optional) features hidden but available to the knowing intermediate or expert user via accelerators. An *accelerator* is a feature available through a keyboard shortcut, key combination, or some other subtle mechanism. Accelerators are

designed to facilitate repetitious or common tasks for users who are comfortable enough to use them.

Note that this heuristic can be over-applied all too easily. My great-aunt was complaining once about a dreadful thing that had happened to her while using Microsoft Word. She had been typing a letter to her friends, and, after using a dash somewhere in the body, the program began to format her document for her automatically—in a fashion that was not what she wanted! The program was too “smart” to be helpful.

I assured her that it wasn't her fault (users almost always blame themselves!) and produced a recent *Macworld* issue in which a reader had mailed in a similar complaint. The “fix” was to disable the “feature.” So be careful that you don't create the Sorcerer's Apprentice Effect¹³ in your “helpful” designs!

Heuristic #8: Aesthetic and Minimalist Design

This heuristic is really just saying not to throw in the kitchen sink and everything else when all it takes to get the point across is a simple word or image. Adding items to a page does not automatically make it better; they usually make it worse if the addition isn't carefully thought out.

There are many reasons for adopting this minimalist stance. First, bandwidth is a critical resource, so adding content that isn't critical will work against you. Second, the average attention span (especially in Americans) is short, so the more you have on a page, the less likely each item is to be noticed. Finally, the more information on a page, the poorer the signal-to-noise ratio.

Note that minimalist design does not mean that in order to have a truly usable site, you must have an incredibly ugly, 1994-ish looking site, although many usability specialists have sites that take this approach. The truth of the matter is that you cannot have just form or function; you need both. Truly excellent sites combine both aspects of design.

¹³Watch the Disney classic *Fantasia* if you don't know what I am talking about.

Moving away from less complex systems usually results in more chaos and more errors; a carefully designed and appropriately grown site can include good looks and good usability. Designs such as these don't happen overnight; they are the result of iterative redesign.

Heuristic #9: Help Users Recognize, Diagnose, and Recover from Errors

One of my pet peeves is poorly designed Web forms, especially forms that do not employ client-side validation. Many times I have filled out forms that scrolled down the screen for miles and then submitted them, only to get an error message in return—but a well-hidden error message! I have seen many Web forms that return the same screen the user filled out, plus an obscure error message hidden somewhere on the page. Sometimes, the programmer was thoughtful enough to make the error message appear in a novel color (red, usually).

If a user makes a mistake on your site—by improperly filling out a form, by entering a botched query string, or whatever—you should have an obvious method for letting the user know about it. Consider using JavaScript to do client-side form validation for browsers that are capable of it. (Always design your server-side programs to be robust against unexpected input, though.) See the Javascript Package on the CD-ROM for more information.

When reporting an error to the user, make sure that you observe heuristic #2. Use the user's natural language to report the error condition, and explain to the user how to fix the problem.

Heuristic #10: Help and Documentation

This is the last heuristic because if you have observed the other nine, you shouldn't need too much of this final one. No matter how well designed your site is, you'll still need to employ some sort of online help, even if it is just a list of FAQs that you keep updated. If the user needs documentation to use the help system, you're in big trouble.

The topic of writing great online documentation and help is outside the scope of this book, but here are some basic tips that come from the *Apple Developer's Guide*¹⁴ to get you started:

- Your list of topic areas should be a logical outline of the guide file contents, similar to the table of contents for a book.
- Choose a method of organization that makes sense from the user's standpoint, not the system designer's. If in doubt, list topics alphabetically.
- For your main help instructions, topic names should form a complete question or statement from the user's point of view—for example, "How do I sign on?"
- The topic name should always focus only on the main goal that the user wants to achieve and not on any choices associated with that goal.
- Use a help heading that begins with "Why can't I . . ." for topics that explain why the user cannot perform a certain action (for example, "Why can't I print a file?").
- Use a help heading that begins with "How do I . . ." for topics that show the user how to accomplish a task (for example, "How do I create a custom dictionary?").

SOME ADDITIONAL HEURISTICS FOR THE WEB

It's important to understand that the preceding list of heuristics is a good starting point, but it is by no means the only such list that is valid. Additionally, as the technology used to implement Web documents evolves, the heuristics will also necessarily evolve. The list that you end up using may consist of just a subset of the ones we've examined, a combination of those and the ones I suggest in this section, or even a hybrid with your own special heuristics that come from your personal experience.

¹⁴You can find the *Guide* at www.devworld.apple.com/techpubs/mac/AppleGuide/AppleGuide-17.html.

Now that you have seen the original heuristics as per Jakob Nielsen, here are some newer suggestions for Web-specific evaluations. Some of these were mentioned earlier in Chapter 4.

Additional Heuristic #1: Chunking

Chunking is a cognitive tool that humans use to simplify their perception of the environment. When many pieces of similar information are presented in a spatially related fashion (i.e., close together), the perceptual system distills them into larger pieces of information called *chunks*.

In the early days of telephone research, it was discovered that human beings could keep roughly seven, plus or minus two, unrelated bits of information in their working memory at once—information, for example, such as digits. This is the reason U.S. phone numbers have seven digits.

But note that chunking has a profound effect on phone-number memorization. For example, I can't help but think of area codes as single pieces of information, since they have meaning to me. I know that (617) is the area code for most of Boston and surrounding areas; I know that 72206 is the ZIP code for the Arkansas governor's mansion. Chunking is hard at work in these examples, making those otherwise meaningless bits of information seem much more manageable and meaningful.

It is also known that assigning semantic cues to otherwise meaningless stimuli is very effective for facilitating recall. Take, for example, the experiment conducted with "doodles." Check out www.doodles.com; it's fun and informative.

So how do you make this information a part of your Web site? If you have a complex page that has much more than the magic 7 ± 2 items, you should consider chunking as a partial solution. Try to cluster like pieces of information near one another, and provide white space around them to make them visually distinct.

Additional Heuristic #2: Use the Inverted Pyramid Style of Writing

Journalists know a lot about this technique. The idea is that you should put the most important information in a text-heavy page or segment at the very top, leaving less important details for later in the text. This way, users have to read a minimal amount of information before they can move on. Reading on-screen text is painful (more on this in a moment), so reducing the amount of text a user has to read is a great benefit.

Although a discussion of online writing style is out of the scope of this book, here are some general suggestions for creating good inverted pyramid-style text. Always remember that you want to wrap up the most important details in the first sentence. Subsequent sentences should contain information of decreasing importance. How do you determine the importance of a piece of information? There is no exact answer, nor is there a formula to calculate importance. What you can do is refer to this well-known list of news values, which is the sort of list that journalists use to prioritize information in their articles:

- **Impact: Information has impact if it affects a large number of people.** If information will impact a significant number of your users, it is important. If the information details how a raccoon broke into the employee lunchroom and scarfed Bill's peanut-butter-and-bologna sandwich, that's not really a high-impact story (although it might qualify for the "strange" category listed below!)
- **Timeliness: Information has timeliness if it happened recently.** The meaning of "recently" varies, depending on the type of publication. For a weekly newspaper, "recently" would be anything that had happened this week. For news services such as CNN, anything older than 24 hours is certainly old news. On the Web, you should strive for a very small window of currency, at least on par with news channels. Of course, this will vary based on your business and your company; maybe nothing interesting happens at your company!

- **Prominence: Information has prominence if it involves a well-known person or organization.** When a major company becomes a client of yours, that's important. When your company makes a \$10 million donation to a Children's Hospital, that's important. When Jane falls off the roof after chasing a raccoon through your office building, that's, erm, not as important.
- **Proximity: Information has proximity if it involves something that happened somewhere nearby.** This is a classic journalism news value, but its utility is somewhat lost on the Web. Define "local" on the Web—Earth? So unless your user population is confined geographically (which is very possible, depending on the nature of your site), you have an open field here.
- **Conflict: Information has conflict if it involves some kind of disagreement between two or more people.** Alas, many humans, especially Americans, are drawn to conflict, hence the popularity of many confrontational "talk" shows and professional wrestling. I do not condone gratuitous violence on a Web site, but a hot debate over a topic that might be appropriate for and of interest to your users is at least moderately important.
- **Weirdness: Information has weirdness if it involves something unusual or strange.** People tend to be drawn in by information that is bizarre. Take, for example, tabloid headlines such as "Techno-Squid Eats Parliament" or "JFK Fathered My Half-Alien Elvis Baby." Some Web journals have used this principle as their primary gimmick. Things that are novel tend to grab people's attention. Be careful, though, that you don't sink into sleaze or cheap effects.
- **Currency: Information has currency if it is related to some general topic many people are already talking about.** This one is a double-entendre, since people are also generally interested in "currency"—i.e., money! There is something to be said for tossing around buzzwords, but at the same time, there is much to be said for credibility and competence.

Additional Heuristic #3: Important Information Belongs "Above the Fold"

To borrow yet another concept from journalism, the most important information on a Web page must go near the top of the page. Depending on how paranoid you want to be, you should preview your work at a resolution of 640 by 480 on a 14-inch monitor to ensure that the most important information is visible without scrolling.

If you don't believe me right now, you will after you do a few usability tests: A frightening number of users are either blissfully unaware that they can scroll vertically in a browser or they opt not to. Some people can't scroll easily in either dimension (for example, if they are using WebTV, which cannot scroll horizontally). Hence, you should make sure that the vital bits of your message go on the first screenful.

Additional Heuristic #4: Avoid Gratuitous Use of Features

Just because a cool new technological toy exists does not mean that you are obligated to incorporate it into your site. Don't use frames unless you really need to and are ready to pay the price; frames are still problematic for most users. The same can be said of Flash, Java applets, JavaScript, or anything else that you might hear is "the latest and greatest."

Recently, Northeastern University made a major revision to its web site (www.neu.edu) to include numerous Flash animations. In my opinion the animations don't do anything to truly make the site a better tool for its patrons. After being teleported through two "entry portals" a la David Siegal,¹⁵ the NEU site takes you to a page that is full of Java applets, image-based navigation, and scriptlets that were generated by GoLive. So the site is 100 percent buzzword compliant. But if you are visually impaired, the game is over. There is absolutely no textual content on the "home" page for a speech synthesizer to read aloud.

¹⁵David Siegal is the author of the book *Creating Killer Web Sites*.

Any time you feel the temptation to use toys like these, make sure that you are truly adding value to the site, not just arbitrarily adding more fluff for the sake of keeping up with the Web-Joneses.

Additional Heuristic #5: Make Your Pages “Scannable”

Many early Web site designs drew heavily from printed products such as brochures, magazines, and other traditional media. These sites featured vast quantities of text in the same fashion that you would find in the traditional media, since most companies felt that this was the correct approach. Now that we're a few generations of site design wiser, it's pretty well known that this is absolutely the wrong approach.

It is now widely accepted that users of Web sites read those sites in a vastly different fashion than they would read a book, a newspaper, or magazine: They scan. Instead of diving into chunks of text and trying to understand it all, Web site users look quickly for keywords, hyperlinks, and other important eye-catching features in order to progress to the next important page. Heavy textual content on the Web is usually ignored completely or, in the best case, it is printed for easier reading. In many cases, if you must use large amounts of text, it's wise to consider an alternative media type, such as Adobe Portable Document Format (PDF).

Web pages that are easy for users to scan have several common characteristics. Some, although not all, of the characteristics are as follows:

- Sans serif fonts
- Short, concise self-describing hyperlinks that are set apart by white space
- Brief, easily digestible paragraphs that follow the inverted pyramid style

Sans serif fonts are held to be more readable on-screen than serif fonts, for many reasons. In traditional typography, the body of a document is normally set in a serif font because it is believed that serif fonts are generally more readable over the long haul. Sans serif fonts are typically much more prominent optically and so are often used to set headlines in printed material (see Figure 6.2).

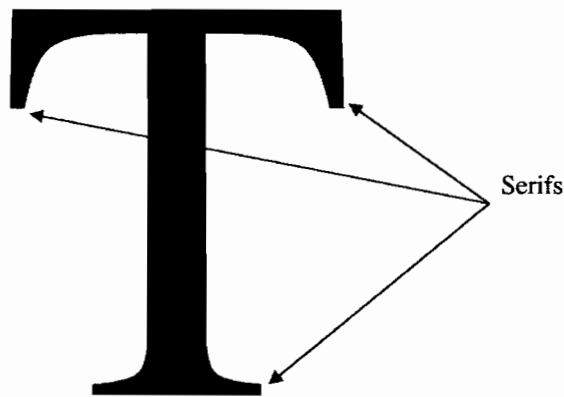


FIGURE 6-2 *This is an example of a serif font. The little “tails” on the ends of the strokes are called serifs. A sans serif font, logically enough, lacks these tails.*

However, the combination of screen flicker and poor resolution makes serif fonts hard to read on a computer monitor. This limitation will likely dissipate as soon as high-resolution (300 pixels per inch or higher) displays become commonplace.

The original Web authors planned on hypertext links being unobtrusive, inline items that would not scream for attention. However, since the birth of the Web it has been pretty obvious that users don't grasp the inline-link concept; these kinds of links are frequently overlooked. The popularity of the hyperlink “Click Here” is both frightening and pervasive. Many users I have seen participate in usability studies have been almost oblivious to links that don't say “Click Here.”

Therefore, links that stand out from surrounding text seem effective. Burying links inside text may net you lots of errors in navigation on your site.

Additional Heuristic #6: Keep Download and Response Times Low

One of the most pervasive complaints about Web sites has been the same since the beginning of the World Wide Wait: sluggish web site

response. Even a well-designed and otherwise usable site can be severely downgraded if the download time is high. Sluggish site behavior can be caused by many things, including client-side bandwidth limitations, server load, and network congestion. Another major culprit is graphic obesity. This heuristic is one that lends itself to automated evaluation (see the "Automated Aids to Heuristic Evaluation").

Is that it?

Not exactly. Again, this list is a nonexhaustive one, but it summarizes some of the most important usability heuristics you are likely to need to do a good evaluation. Do you have suggestions for new heuristics? Pass them along to the community, because they could become widely adopted. I encourage each of you to explore this technique as an addition to your toolbox and to expand and update the heuristic list as technology evolves.

HEURISTIC EVALUATION METHODOLOGY

Now that you have a list of potential heuristics against which you can measure your site, it's important to know how to go about gathering information and making sense of it. One critical thing to note is that heuristic evaluation works best when approximately five experts are involved in the evaluation process. According to Nielsen, the bang-for-the-buck curve begins to flatten out at around five evaluators. If you have too few evaluators, however, you will end up catching only a fraction of the usability problems on the site.

The Environment

Each evaluator should view the site in isolation, as mentioned before. You should plan on the evaluation taking from one to two hours to complete. In the event that the evaluation of the complete site cannot be done in this time frame, plan on breaking the evaluation goals up over two or more sessions in order to keep individual session times to two hours or less.

Some evaluators will have a very methodical manner about examining the site; they might move in a linear fashion from one end of the site to the other. Other evaluators might have a more random approach. The end result, fortunately, is that not everyone will find all the same usability problems, resulting in more problems discovered, so there is much to be said for diversity of technique!

Error Severity Ratings

As each evaluator progresses through the site, they should make a log of usability problems they find, with specific notes about the page on which each problem was located, including the page URL. It is also critical that each problem be assigned a *severity rating*, which is a way of gauging how destructive the error is to the overall functionality of the site. This kind of severity rating allows the usability team to prioritize the components of change that they will recommend as a result of the evaluation. Usually a scale of 1 to 4 is used to rate the problems that are encountered:

1. Cosmetic problem only; need not be fixed unless extra time is available on project
2. Minor usability problem; fixing this should be given low priority
3. Major usability problem; important to fix, so should be given high priority
4. Usability catastrophe; imperative to fix this before product can be released

Debriefing

After all the expert evaluators have rated and scored the site, it is not uncommon for all the evaluators to meet with a moderator to discuss what happened in the test sessions. Individual evaluators can discuss their own subjective experiences with the site; this is also an excellent time for the moderator to ask probing questions about specific areas of usability violations noted on the evaluators' logs. You could even choose to audio- and videotape the debriefing sessions for future analysis.

During the debriefing, it is very likely that the expert evaluators will have suggestions for change, as well as new ideas and new questions that will help the usability team further explore the site's usability. The feedback gathered at the debriefing should be transcribed and presented in the summary report that results from this evaluation.

Assembling Data

Putting your results together and making sense of them is just as important as, if not more important than, gathering the data. Therefore, you need to analyze your data along several lines.

Quantitative Measures

What was the overall number of unique usability problems that were found in this round of heuristic evaluation? If you have a baseline (obtained from previous heuristic evaluations), what is the delta (change) from the last measurement? How many errors did each evaluator find?

Heuristic Violation Proportion Graph

Make a bar graph to show how many violations per heuristic were found. Notice any trends that occur; for example, a huge spike in the "user's natural language" heuristic could mean that there was not enough early interaction with the real user community for the site's designers to develop a user-oriented vocabulary.

Severity Proportion Chart

Make a chart to show how many errors of each class were found. A high proportion of type 4 errors (the severest type) could serve as a catalyst to make the "powers that be" approve major changes for the site. This graphic also gives the team an easy-to-understand snapshot of the overall health of the site's usability (see Figures 6.3 and 6.4).



FIGURE 6.3 An example of a chart showing heuristic violation proportions.

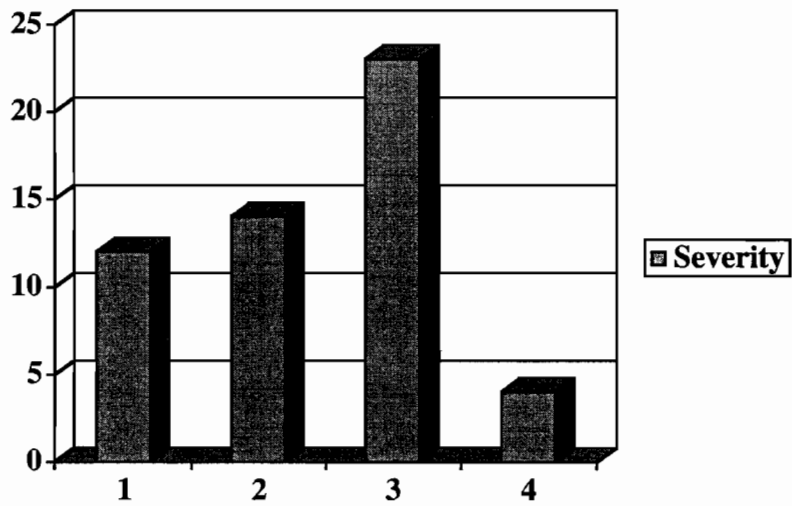


FIGURE 6.4 An example of a chart of severity types in an heuristic evaluation.

Thematic Trends

Were there any obvious trends in the located usability problems? For example, if the evaluators found that the terminology used to label various parts of the site was too jargon-filled, falling outside the user population's natural vocabulary, you would again have powerful evidence to support the need to rethink this area and to solicit additional feedback from the user community.

Putting It All Together

Finally, you need to summarize all your suggestions into bullet points so that the team has a clear set of objectives for improved usability. Be sure to cite specific instances and URLs rather than general concepts. Make this list almost impossible to *not* understand. The level of detail that you should include is a stylistic and workflow-dependent decision. The most important idea here, though, is to be specific enough that the "fixes" can be understood and carried out by someone not on your immediate team of evaluators.

Some Criticism of Heuristic Evaluation

Heuristic evaluation is a tool that assuredly has a place in your toolbox. Remember, though, that it should by no means be the only trick up your sleeve! You might hear several criticisms from others in your organization. Take heart from the following excerpt, which comes from Jakob Nielsen's landmark publication, *How to Conduct a Heuristic Evaluation*:¹⁶

Heuristic evaluation does not provide a systematic way to generate fixes to the usability problems or a way to assess the probable quality of any redesigns. However, because heuristic evaluation aims at explaining each observed usability problem with reference to established usability principles, it will often be fairly easy to generate a revised design according to the guidelines

¹⁶You can find this article at www.useit.com/papers/heuristic/heuristic_evaluation.html.

provided by the violated principle for good interactive systems. Also, many usability problems have fairly obvious fixes as soon as they have been identified.

As you see, this technique was never really intended to be a highly empirical method for obtaining data and systematically transforming it into change; rather, it is a shoot-from-the-hip guerrilla tactic that helps jump-start almost any usability project. And this is not a bad thing at all!

The following sections summarize some criticisms you might hear from naysayers in your organization.

Real Users Aren't "Expert Evaluators"

Obviously, this technique revolves around the idea that expert evaluators "know what's best" for real users. Some people have criticized this technique because it does not attempt to approximate the actual user's frame of reference. My rebuttal has always been that regardless, the expert evaluators do in fact uncover usability errors that would have almost assuredly tripped up a "real" user. This speaks to the concept that I have only recently, and reluctantly, begun to embrace: Some usability is better than none, and it rarely matters how you come to find the problems, as long as you do find them.

Heuristic Evaluation Can't Find All "Show-Stopers"

This statement is absolutely true. There will almost assuredly be at least one obscure problem with your site that "expert" users won't be able to locate because they don't interact with the site the way a novice user might. It never ceases to amaze me how much trouble some people—even intelligent, relatively computer-literate people—have with their computers. I have seen several users perform the "click of death" on a computer before—that is, they click so rapidly on everything in their visual field that the computer freezes. I have watched at least one very intelligent user trash the contents of her most important directory—irrevocably—because she would not stop to read the warning messages that popped up on her screen.

for
l vari-
popu-
e to
back

nts so
sure
fake
at you
most
s" can
team

ox.
your
ation.
lsen's

ml.

When I tried to replicate the error just to figure out how it could have been avoided, I found it nearly impossible to duplicate. The novice user had usage habits that were simply alien to me; I would never have stumbled on the chance condition that she had so easily wrought. But remember that you will still have real users test your design, so everything should “come out in the wash.”

Automated Aids to Heuristic Evaluation

Although automated techniques should never be used to completely replace real user testing, they can be used quite effectively to augment an existing comprehensive usability plan. In particular, certain types of metrics tend to lend themselves to being evaluated in an automated fashion. Such data, collected automatically, can be a real asset to any usability plan. Remember that automated techniques allow you to acquire large amounts of data that would otherwise be too painful or not feasible to gather by hand.

In particular, automated testing of link integrity (searching for broken links), load times, types of content on a site (HTML, Java, JavaScript, other embedded content such as video or audio data), and the number of ways to traverse to a particular page are easily automated programmatically.

At least one company, WebCriteria,¹⁷ makes a living of implementing just such a technique. For a reasonable fee, you can get the company to do a metrics assessment of your site. The company's current product incarnation uses “Max,” an automated browsing agent that is supposed to traverse a site the way a normal human user would. Max is built on a well-known human factors model known as GOMS, which is a type of model human processor.¹⁸ I have not used the service personally. You might opt to outsource such a service to a company like this one, or you

¹⁷You can visit WebCriteria at www.webcriteria.com.

¹⁸For more information on Max and GOMS, see www.isl.nist.gov/iaui/vvrg/hfweb/proceedings/lynch/index.html.

could choose to write your own in-house programs to do site traversal and reporting for you.¹⁹

CHAPTER SUMMARY

- A heuristic is a rule of thumb.
- Jakob Nielsen developed heuristic evaluation several years before the Web became popular.
- Heuristic evaluation can be done rapidly and with little expense, compared with other testing methods.
- In order to do a heuristic evaluation, you need a list of heuristics to follow and approximately five evaluators.
- The list of heuristics you use might come from the original 10 developed by Nielsen and reviewed in this chapter, or you might choose other heuristics that you or other usability professionals derive from experience.
- Almost every usability problem can be traced to a violation of one or more heuristics.
- To assist in prioritizing change, you should also rate the severity of each usability problem.
- You may hear several types of criticism regarding heuristic evaluation, almost all of which can be dispersed by simply using heuristic evaluation in addition to, rather instead of, other evaluation techniques.
- Parts of heuristic evaluation can be automated using commercial and homegrown software.

HANDS-ON EXERCISES

1. Assemble a team of evaluators. You should shoot for five, but if you can even get one other person to help you with this task, it makes a huge difference.

¹⁹And if you do, may I suggest using Perl?

2. Perform a heuristic evaluation of your site. Decide ahead of time which heuristics you will use, and prepare scoring/comment sheets for your evaluators to use. You can use the sample scoring sheet included on the CD-ROM as a template.
3. After the evaluation, meet with all the members of your group to discuss the findings. What problems did you find that can be immediate action items (problems that obviously need to be fixed without the need for further user testing)? Which might require more input from users?
4. Do a Web search to find out what kinds of automated usability tools exist. Technology is evolving so quickly that the list will have changed by the time you read this book.

DISCUSSION TOPICS

1. What two other heuristics should be included in the heuristics list discussed in this chapter? Why? Note that there might be an unlimited number of heuristics that you could mention; there isn't necessarily a right or wrong way to answer this question.
2. Why even test users at all? If an evaluator is an expert, shouldn't users expect the expert to solve all the problems without their help?
3. If you were forced to use only heuristic evaluation to test your site, what kind of errors would you be most likely to uncover? What types of errors would you be likely to miss? Explain.
4. Do you think that the list of heuristics should vary completely from site to site, should they be completely standardized, or should there be a middle ground? Explain your answer. What are the pros and cons of each stance?