NN/g Nielsen Norman Group
Evidence-Based User Experience Research, Training, and Consulting

Response Times: The 3 Important Limits

by Jakob Nielsen on January 1, 1993
Topics: Human Computer Interaction Web Usability Application Design

Summary: There are 3 main time limits (which are determined by human perceptual abilities) to keep in mind when optimizing web and application performance.

Excerpt from Chapter 5 in my book Usability Engineering, from 1993:

The basic advice regarding response times has been about the same for thirty years [Miller 1968; Card et al. 1991]:

- **0.1 second** is about the limit for having the user feel that the system is **reacting instantaneously**, meaning that no special feedback is necessary except to display the result.
- **1.0 second** is about the limit for the **user's flow of thought** to stay uninterrupted, even though the user will notice the delay. Normally, no special feedback is necessary during delays of more than 0.1 but less than 1.0 second, but the user does lose the feeling of operating directly on the data.
- **10 seconds** is about the limit for **keeping the user's attention** focused on the dialogue. For longer delays, users will want to perform other tasks while waiting for the computer to finish, so they should be given feedback indicating when the computer expects to be done. Feedback during the delay is especially important if the response time is likely to be highly variable, since users will then not know what to expect.

Normally, response times should be as fast as possible, but it is also possible for the computer to react so fast that the user cannot keep up with the feedback. For example, a scrolling list may move so fast that the user cannot stop it in time for the desired element to remain within the available window. The fact that computers can be too fast indicates the need for user-interface changes, like animations, to be timed according to a real-time clock rather than being timed as an indirect effect of the computer's execution speed: Even if a faster model computer is substituted, the user interface should stay usable.

In cases where the computer cannot provide fairly immediate response, continuous feedback should be provided to the user in form of a **percent-done indicator** [Myers 1985]. As a rule of thumb, percent-done progress indicators should be used for operations taking more than about 10 seconds. Progress indicators have three main advantages: They reassure the user that the system has not crashed but is working on his or her problem; they indicate

approximately how long the user can be expected to wait, thus allowing the user to do other activities during long waits; and they finally provide something for the user to look at, thus making the wait less painful. This latter advantage should not be underestimated and is one reason for recommending a graphic progress bar instead of just stating the expected remaining time in numbers.

For operations where it is unknown in advance how much work has to be done, it may not be possible to use a percent-done indicator, but it is still possible to provide running progress feedback in terms of the absolute amount of work done. For example, a system searching an unknown number of remote databases could print the name of each database as it is processed. If this is not possible either, a last resort would be to use a less specific progress indicator in the form of a spinning ball, a busy bee flying over the screen, dots printed on a status line, or any such mechanism that at least indicates that the system is working, even if it does not indicate what it is doing. Note added for the web version of this essay: Most web browsers fail in providing useful progress bars, since they don't communicate what percentage of the *entire download* for a page has been completed.

For reasonably fast operations, taking between 2 and 10 seconds, a true percent-done indicator may be overkill and, in fact, putting one up would violate the principle of display inertia (flashing changes on the screen so rapidly that the user cannot keep pace or feels stressed). One could still give less conspicuous progress feedback. A common solution is to combine a "busy" cursor with a rapidly changing number in small field in the bottom of the screen to indicate how much has been done.

See Also:
Article about website response times and how to improve them.

Web-Based Application Response Time

Update added 2014: I keep getting questions like this, so I decided to answer it here.

> **Q:** "You mention many times that response time is important, and there are tons of tools to measure response time, but what is an acceptable web based application's response time? What is a user's tolerance, not for a shopping experience, but for an interactive application?"

A: I wish we could eradicate the term "web-based application" because it distracts from the real issue, which is one of application UI design (we have several full-day courses on this topic). We don't have special guidelines for applications implemented in C++ relative to apps implemented in JavaScript. The fundamental usability recommendations are the same, no matter the implementation, since we are discussing user experience, not coding.

Therefore, the **response time guidelines for web-based applications are the same as for all other applications**. These guidelines have been the same for 46 years now, so they are also not likely to change with whatever implementation technology comes next.

0.1 second: Limit for users feeling that they are **directly manipulating** objects in the UI. For example, this is the limit from the time the user selects a column in a table until that column should highlight or otherwise give feedback that it's selected. Ideally, this would also be the response time for sorting the column — if so, users would feel that *they* are sorting the table. (As opposed to feeling that they are *ordering* the computer to do the sorting for them.)

1 second: Limit for users feeling that they are **freely navigating** the command space without having to unduly wait for the computer. A delay of 0.2–1.0 seconds does mean that users notice the delay and thus feel the computer is "working" on the command, as opposed to having the command be a direct effect of the users' actions. Example: If sorting a table according to the selected column can't be done in 0.1 seconds, it certainly has to be done in 1 second, or users will feel that the UI is sluggish and will lose the sense of "flow" in performing their task. For delays of more than 1 second, indicate to the user that the computer is working on the problem, for example by changing the shape of the cursor.

10 seconds: Limit for users **keeping their attention** on the task. Anything slower than 10 seconds needs a percent-done indicator as well as a clearly signposted way for the user to interrupt the operation. Assume that users will need to reorient themselves when they return to the UI after a delay of more than 10 seconds. Delays of longer than 10 seconds are only acceptable during natural breaks in the user's work, for example when switching tasks.

See Also:
Article on time scales in user experience.

References

Card, S. K., Robertson, G. G., and Mackinlay, J. D. (1991). The information visualizer: An information workspace. *Proc. ACM CHI'91 Conf.* (New Orleans, LA, 28 April-2 May), 181-188.

Miller, R. B. (1968). Response time in man-computer conversational transactions. *Proc. AFIPS Fall Joint Computer Conference* **Vol. 33**, 267-277.

Myers, B. A. (1985). The importance of percent-done progress indicators for computer-human interfaces. *Proc. ACM CHI'85 Conf.* (San Francisco, CA, 14-18 April), 11-17.