## Schneiderman's Eight Golden Rules Will Help You Design Better Interfaces

Ben Schneiderman (born August 21, 1947) is an American computer scientist and professor at the University of Maryland Human-Computer Interaction Lab. His work is comparable to other contemporary design thinkers like Don Norman and Jakob Nielsen. In his popular book "Designing the User Interface: Strategies for Effective Human-Computer Interaction", Shneiderman reveals his eight golden rules of interface design:

- **Strive for consistency** by utilizing familiar icons, colors, menu hierarchy, call-to-actions, and user flows when designing similar situations and sequence of actions. Standardizing the way information is conveyed ensures users are able to apply knowledge from one click to another; without the need to learn new representations for the same actions. Consistency plays an important role by helping users become familiar with the digital landscape of your product so they can achieve their goals more easily.
- **Enable frequent users to use shortcuts.** With increased use comes the demand for quicker methods of completing tasks. For example, both Windows and Mac provide users with keyboard shortcuts for copying and pasting, so as the user becomes more experienced, they can navigate and operate the user interface more quickly and effortlessly.
- **Offer informative feedback.** The user should know where they are at and what is going on at all times. For every action there should be appropriate, human-readable feedback within a reasonable amount of time. A good example of applying this would be to indicate to the user where they are at in the process when working through a multi-page questionnaire. A bad example we often see is when an error message shows an error-code instead of a human-readable and meaningful message.


*The Windows Media Player designers should have remembered Ben Schneiderman's 3rd golden rule: Offer informative feedback. Poorly designed error messages often show an error-code that does not mean anything to the user. As a good designer you should always seek to give human-readable and meaningful feedback.*

- **Design dialogue to yield closure.** Don't keep your users guessing. Tell them what their action has led them to. For example, users would appreciate a "Thank You" message and a proof of

purchase receipt when they've completed an online purchase.

- **Offer simple error handling.** No one likes to be told they're wrong, especially your users. Systems should be designed to be as fool-proof as possible, but when unavoidable errors occur, ensure users are provided with simple, intuitive step-by-step instructions to solve the problem as quickly and painlessly as possible. For example, flag the text fields where the users forgot to provide input in an online form.
- **Permit easy reversal of actions.** Designers should aim to offer users obvious ways to reverse their actions. These reversals should be permitted at various points whether it occurs after a single action, a data entry or a whole sequence of actions
- **Support internal locus of control.** Allow your users to be the initiators of actions. Give users the sense that they are in full control of events occurring in the digital space. Earn their trust as you design the system to behave as they expect.
- **Reduce short-term memory load.** Human attention is limited and we are only capable of maintaining around five items in our short-term memory at one time. Therefore, interfaces should be as simple as possible with proper information hierarchy, and choosing recognition over recall. Recognizing something is always easier than recall because recognition involves perceiving cues that help us reach into our vast memory and allowing relevant information to surface. For example, we often find the format of multiple choice questions easier than short answer questions on a test because it only requires us to recognize the answer rather than recall it from our memory.

**10 Usability Heuristics for User Interface Design**

**#1: Visibility of system status**

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

**#2: Match between system and the real world**

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.

**#3: User control and freedom**

Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

**#4: Consistency and standards**

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

**#5: Error prevention**

Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

**#6: Recognition rather than recall**

Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

**#7: Flexibility and efficiency of use**

Accelerators — unseen by the novice user — may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users. Allow users to tailor frequent actions.

**#8: Aesthetic and minimalist design**

Dialogues should not contain information which is irrelevant or rarely needed. Every extra unit of information in a dialogue competes with the relevant units of information and diminishes their relative visibility.

**#9: Help users recognize, diagnose, and recover from errors**

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

**#10: Help and documentation**

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out, and not be too large.