User Interface Design

What is User Interface

- Basically a page or screen in a software application
- A part of software that receives input
  - From the users
- Must have the facility to get input
  - Through keyboard
  - Mouse click
  - Voice
- What kind of Interface?
  - Interface for processing part of the software system
  - Sends data to the to the data processing module

Why is UI important

- The users of the of the system operates through this interface
- The performance of the system depends on the user interface
- Satisfaction of the user interface is important
- It is important to design UI first
- This is a sketch of what user wants
- Easy to convey the users what they will get

Usability in UI

- The effectiveness with which users can accomplish tasks in a (software) system, as measured by
  - Learnability: is it easy to learn?
  - Efficiency: once learned, is it fast to use?
  - Safety: are errors few and recoverable?

Relative importance of usability dimensions

- Depends on the user
  - Novices need learnability.
  - Experts need efficiency.
  - But no user is uniformly a novice or an expert.

- Depends on the task
  - Missile launchers need safety.
  - Subway turnstiles need efficiency.

Usability matters: the cost of getting it wrong

***50%** of all "malfunctioning" electronic devices returned to stores are in full working order, but users can't figure out how to operate them.*

[Elke den Ouden, 2006]

Three Mile Island: nuclear reactor meltdown caused by an ambiguous user interface

Designing UI: A good user interface is hard to design …

- You are not the user
  - Most software engineering is about communicating with other programmers.
  - UI is about communicating with users.
- Users are always right …
  - Consistent problems are the system's fault.
- Except when they aren't
  - Users don't always know what they want.

Achieving usability: best practices

- User testing and field studies
- Evaluations and reviews by UI experts
- Prototyping
  - Cheap, throw-away implementations
  - Low-fidelity: paper prototypes
  - Medium-fidelity: code prototypes

Key to success: good UI focuses on the user, not the developer or the system.

What is prototyping? Why do it?

- **Prototyping**: creating a scaled-down or incomplete version of a system to demonstrate or test its aspects.
- **Benefits** of prototyping:
  - aids UI design
  - help discover requirements
  - help discover test cases and provide a basis for testing
  - allows interaction with user to ensure satisfaction
  - team-building

Some prototyping methods

- Code prototyping
  - implement a "quick" / incomplete version of a UI
- Prototyping with UI builders (e.g., Visual Studio)
  - draw a GUI by dragging/dropping UI controls on screen
- **Paper prototyping**
  - a paper version of a UI

Why paper prototyping?

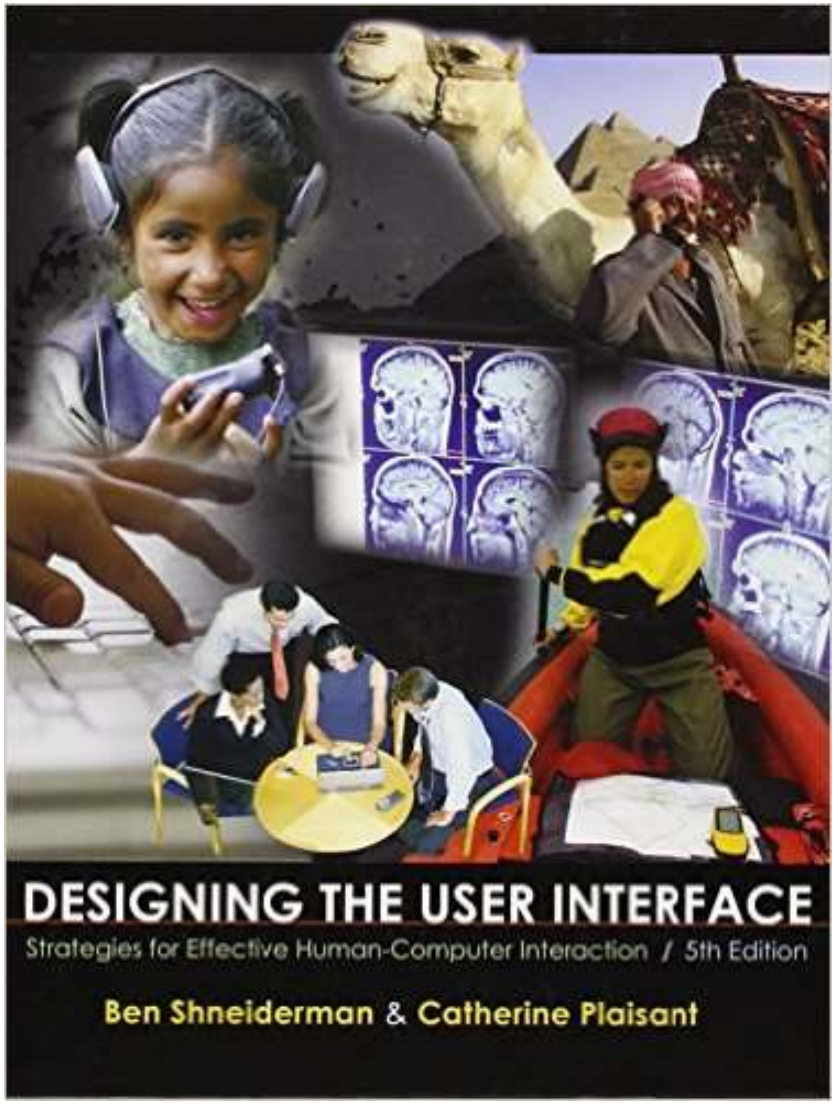
- Much faster to create and change than code
- More visual bandwidth (can see more at once)
- More conducive to working in teams
- Can be done by non-technical people
- Feels less permanent or final

When to do prototyping?

- During or after requirements but before design
  - helps uncover requirements and upcoming design issues
  - shows us **what** is in the UI, but also shows us details of **how** the user can achieve goals in the UI

UI Design Considerations

Schneiderman's 8 Golden Rules

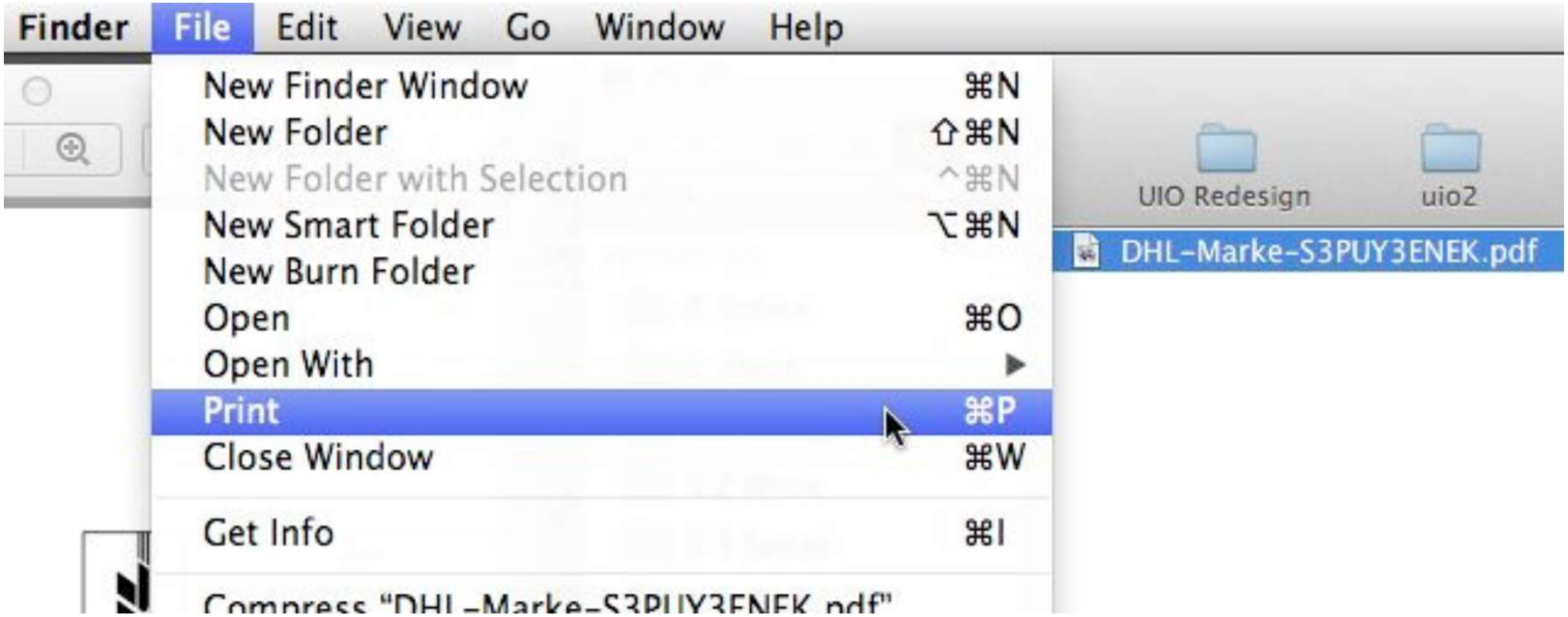


Rule 1: Strive for consistency.

Consistent sequences of actions should be required in similar situations; identical terminology should be used in prompts, menus, and help screens; and consistent commands should be employed throughout.

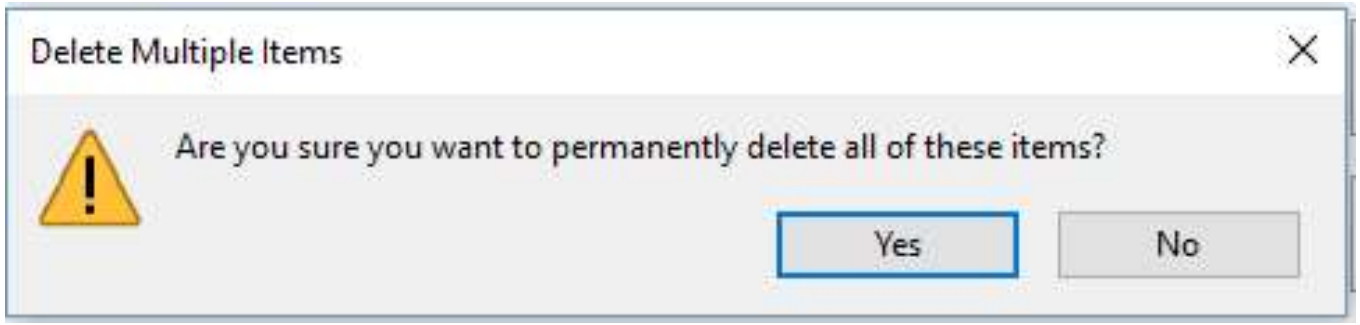
Rule2: Enable frequent users to use shortcuts

As the frequency of use increases, so do the user's desires to reduce the number of interactions and to increase the pace of interaction. Abbreviations, function keys, hidden commands, and macro facilities are very helpful to an expert user.



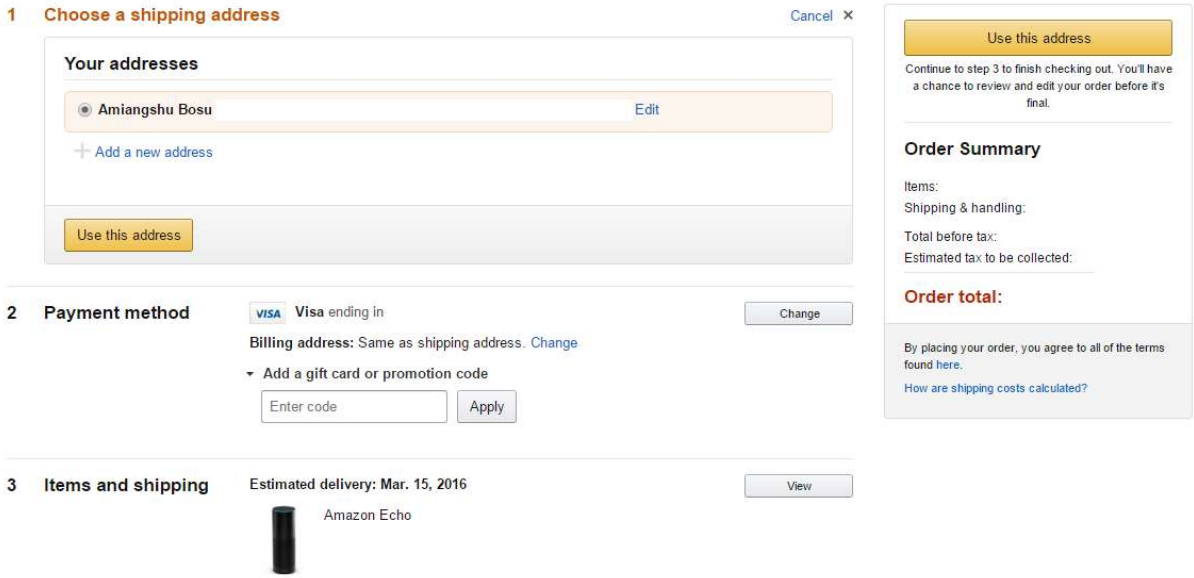
Rule 3: Offer informative feedback

For every operator action, there should be some system feedback. For frequent and minor actions, the response can be modest, while for infrequent and major actions, the response should be more substantial.

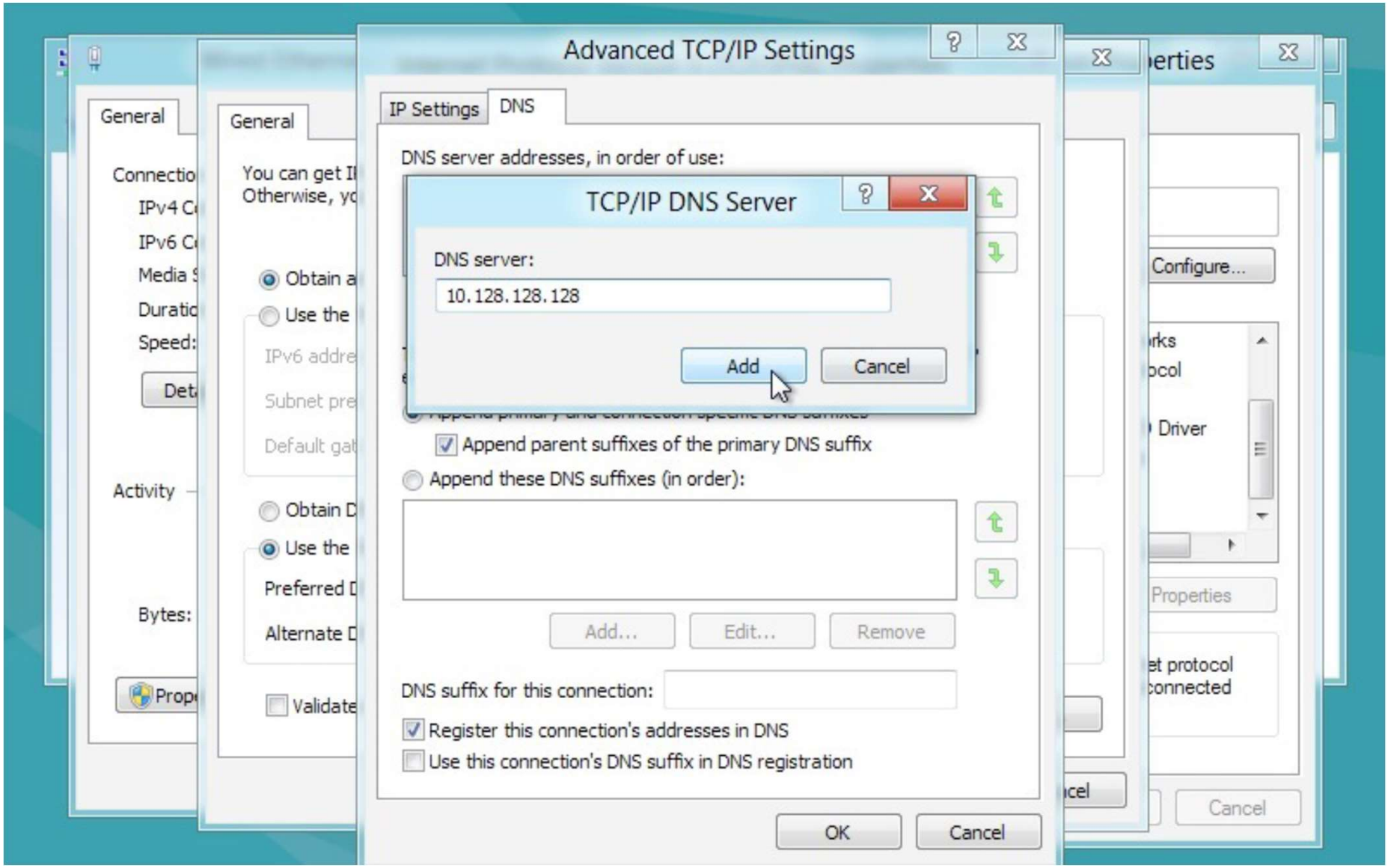


Rule 4: Design dialog to yield closure

Sequences of actions should be organized into groups with a beginning, middle, and end. The informative feedback at the completion of a group of actions gives the operators the satisfaction of accomplishment, a sense of relief, the signal to drop contingency plans and options from their minds, and an indication that the way is clear to prepare for the next group of actions.

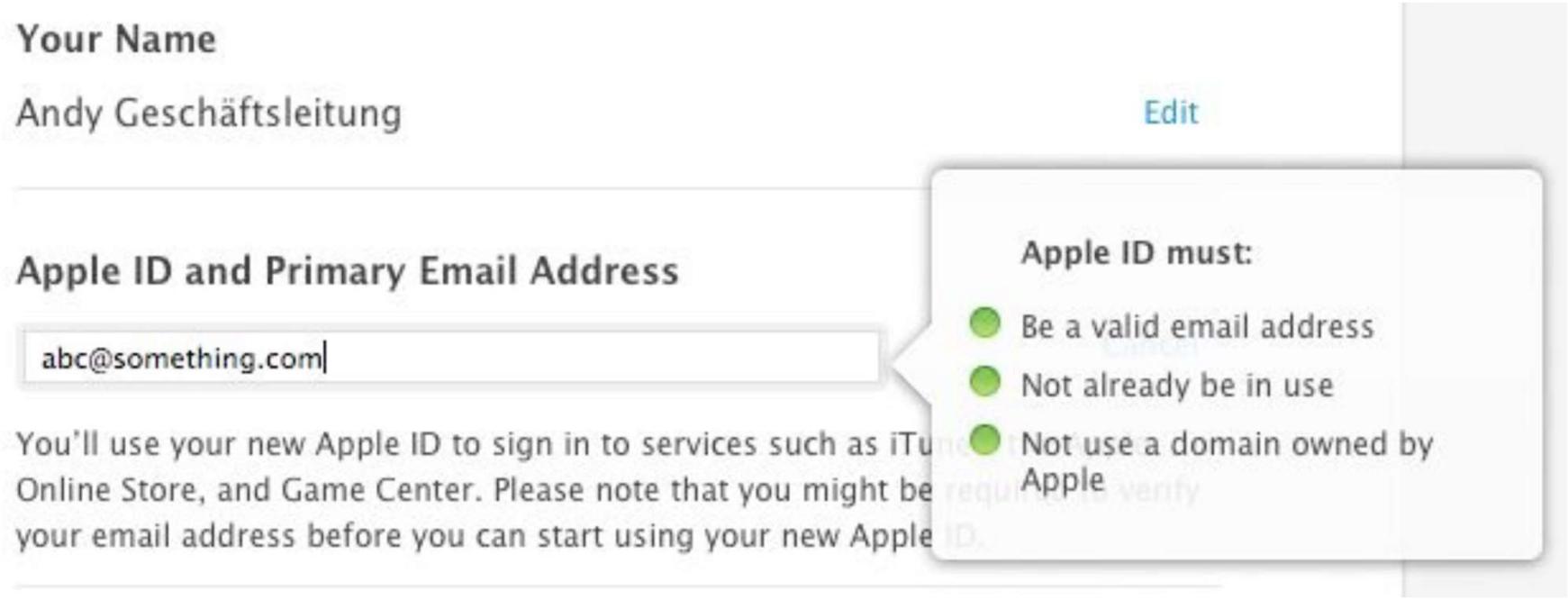


Rule 4: Design dialog to yield closure



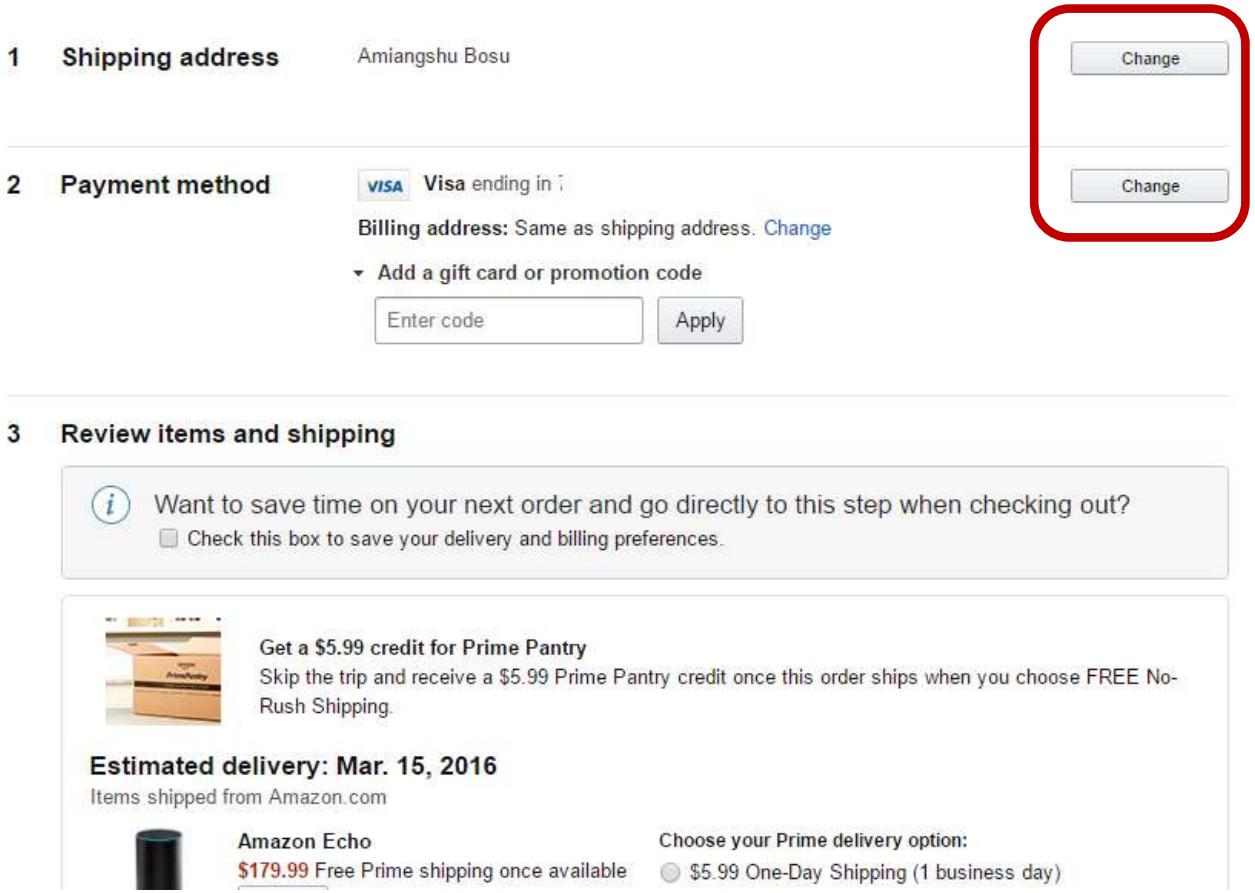
Rule 5: Offer simple error handling

As much as possible, design the system so the user cannot make a serious error. If an error is made, the system should be able to detect the error and offer simple, comprehensible mechanisms for handling the error.



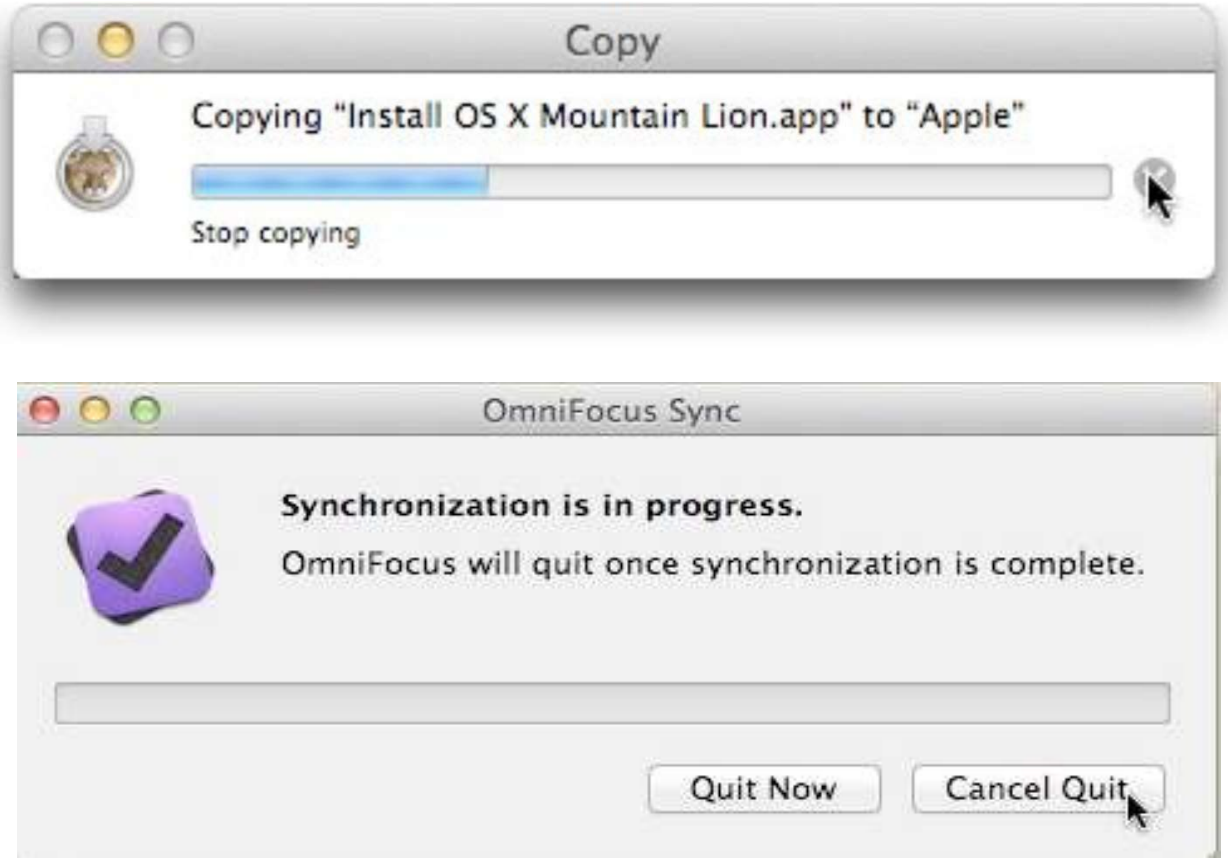
Rule 6: Permit easy reversal of actions

This feature relieves anxiety, since the user knows that errors can be undone; it thus encourages exploration of unfamiliar options. The units of reversibility may be a single action, a data entry, or a complete group of actions.



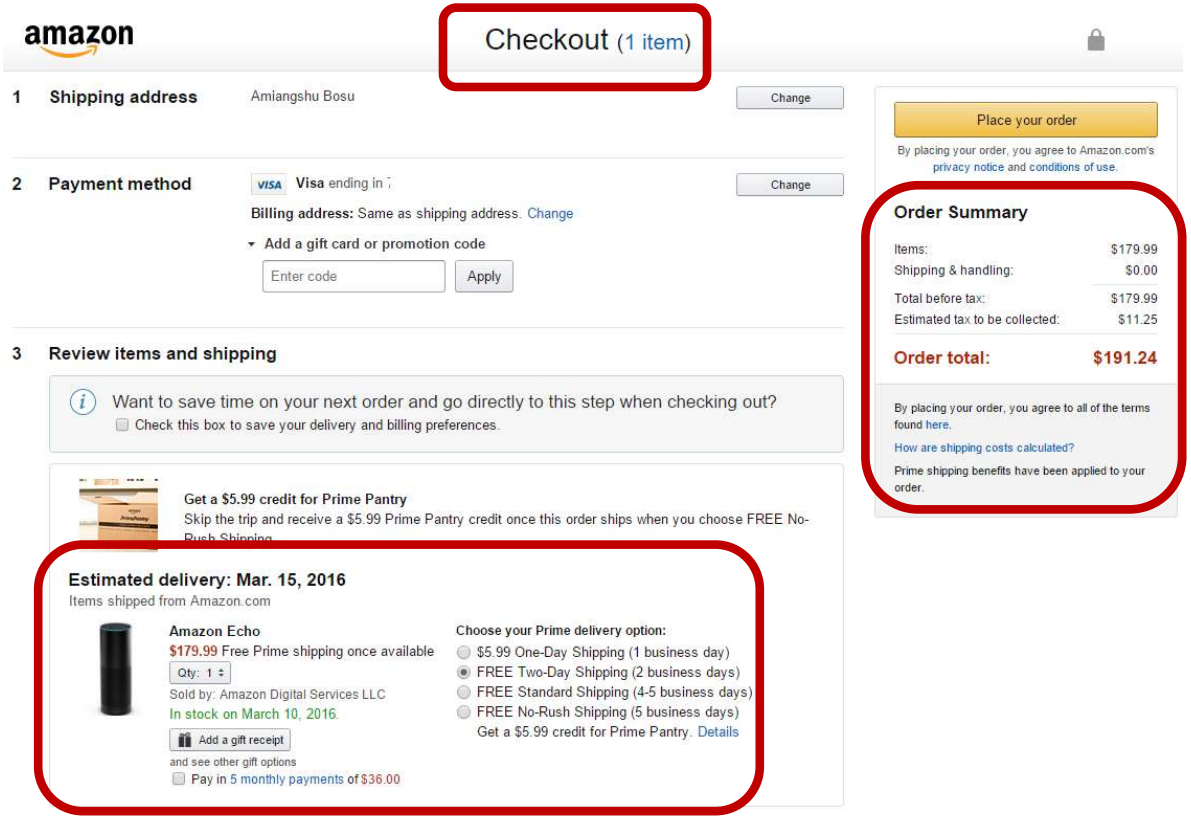
Rule 7: Let the user be in control.

Experienced operators strongly desire the sense that they are in charge of the system and that the system responds to their actions. Design the system to make users the initiators of actions rather than the responders.



Rule 8: Reduce short-term memory load on the user.

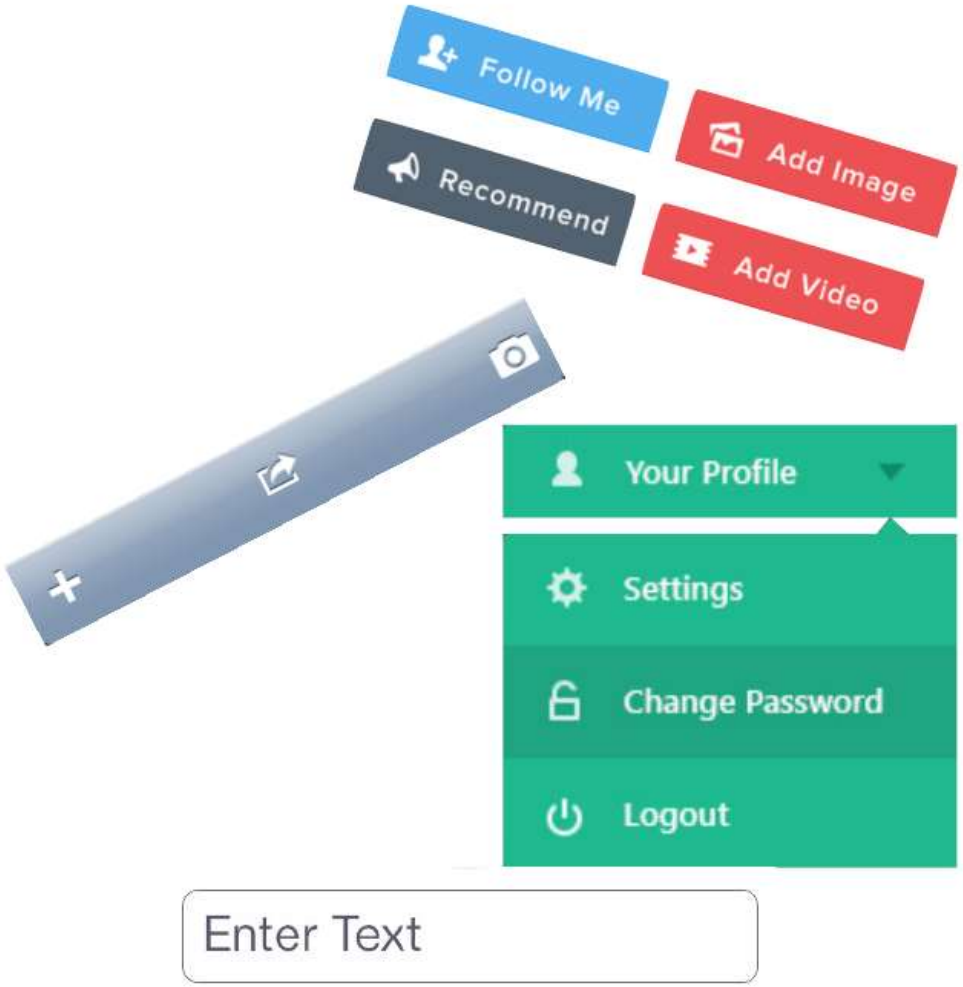
The limitation of human information processing in short-term memory requires that displays be kept simple, multiple page displays be consolidated, window-motion frequency be reduced, and sufficient training time be allotted for codes, mnemonics, and sequences of actions.



UI Design components

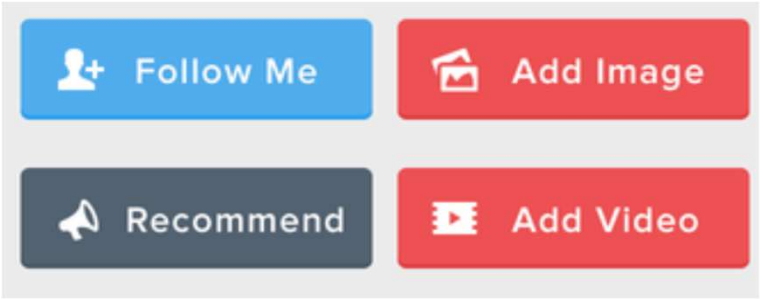
When to use?

- a button?
- a check box?
- a radio button?
- a text field?
- a list?
- a combo box?
- a menu?
- a dialog box?



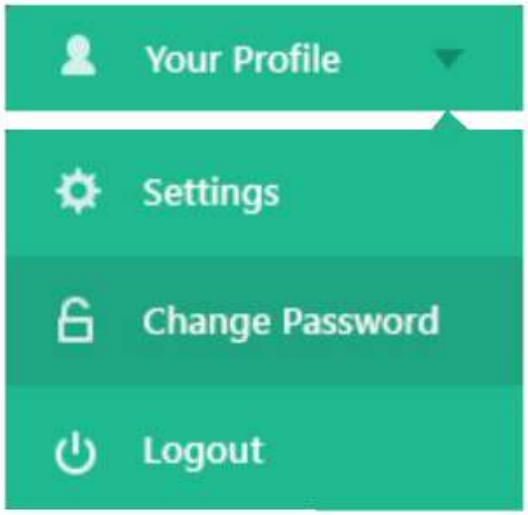
UI design: buttons, toolbars, menus

- Use **buttons** for single independent actions that are relevant to the current screen.
  - Use button text with verb phrases such as "Save“ or "Cancel", not generic: "OK", "Yes", "No"
  - Use Mnemonics or Accelerators (Ctrl-S)



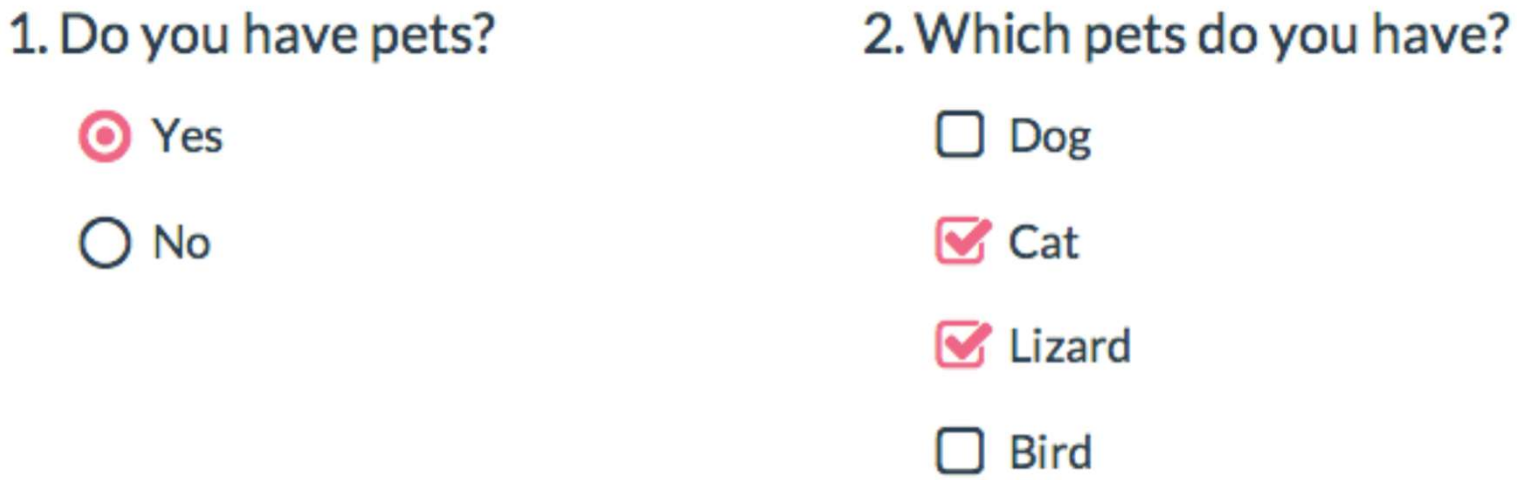
- Use **toolbars** for common actions.

- Use **menus** for infrequent actions that may be applicable to many or all screens.
  - Users hate menus! Try not to rely too much on menus. Provide another way to access the same functionality (toolbar, hotkey, etc.)



UI design: check boxes and radio buttons

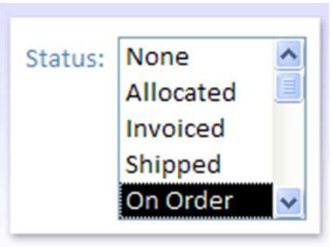
- Use **check boxes** for independent on/off switches.
- Use **radio buttons** for related choices, when only one choice can be activated at a time.



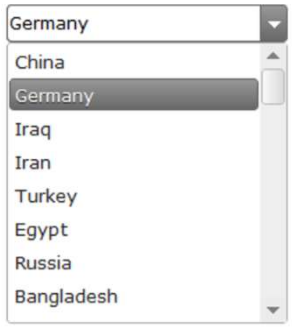
UI design: text fields, lists, combo boxes, sliders

- Use **text fields** (usually with a label) when the user may type in anything they want.

- Use **lists** when there are many fixed choices (too many for radio buttons); all choices visible on screen at once.



- Use **combo boxes** when there are many fixed choices; don't take up screen real estate by showing them all at once.



- Use a **slider** or **spinner** for a numeric value.

UI design: dialogs and panes

- Use a **tabbed pane** when there are many screens that the user may want to switch between at any moment
- Use **dialog boxes** or **option panes** to present temporary screens or options
  - "modal" dialog box prevents any other action