



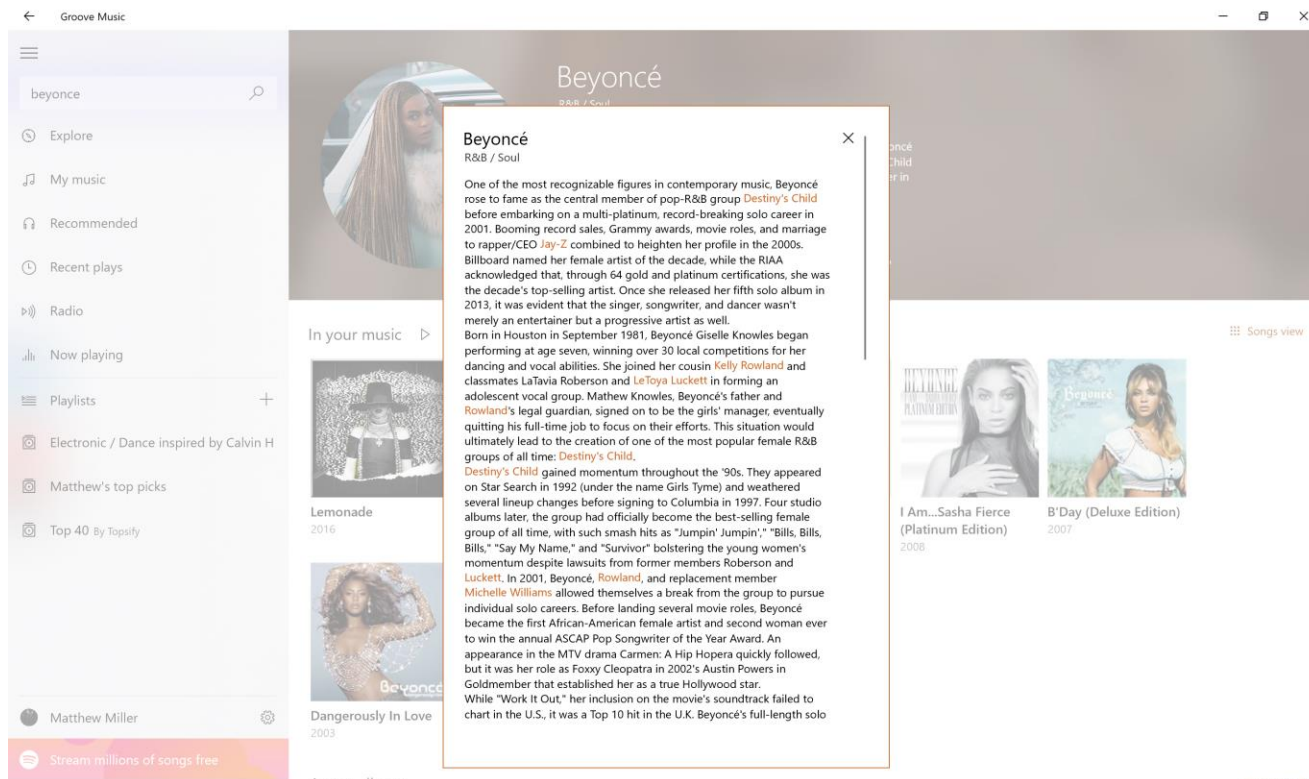
Learning Objectives

- Overview the Heuristic Evaluation Process
- Learn the Heuristics
- Perform Heuristic Evaluation
- Avoid Common Design Pitfalls by Following Design Principles

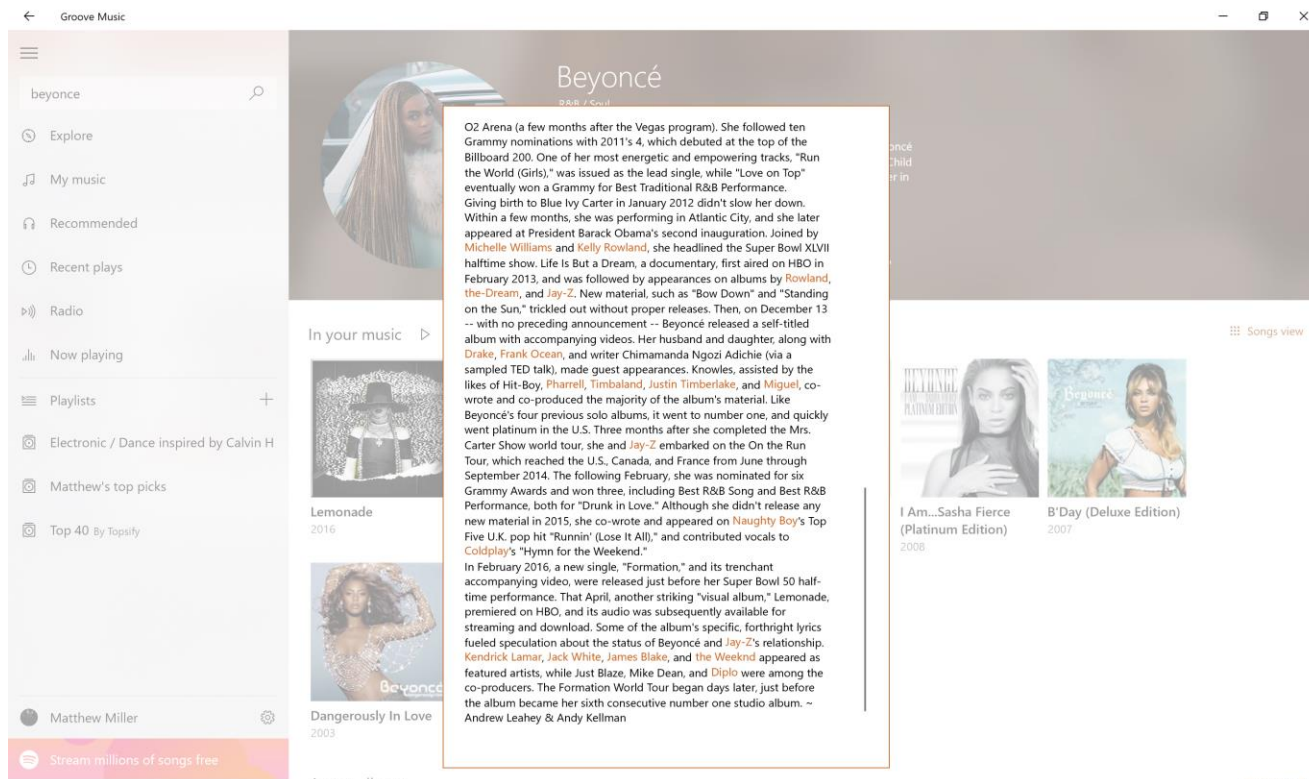


Interface Hall of Shame





Disappearing Close Button



Disappearing Close Button

NSERC Electronic Appli x

https://ebiz.nserc.ca/nserc_web/jaws/nserc_form200/n_cst_f200_person_profile/of_web_po

Save	Preview	Contact Us	Help	Instructions	Logout
------	---------	------------	------	--------------	--------

Location of Tenure >

Form

Application Profile

Person Profile

Addresses

Academic Background

Experience

Awards

Location of Tenure

S&F Information

Thesis

Key Words

Outline of Proposed Research

Justif. for Eligibility of Proposed Research

Contributions/ Statements

Transcripts - Direct

Transcripts - University

Reports on the Applicant

University Comments

Proactive Disclosure

Proactive Disclosure

Form 201 - Location of Tenure

Indicate where you would like to hold this award, in order of preference.

Note: The 'List...' button contains a list of postsecondary institutions.

Proposed location of tenure

Institution / organization Saskatchewan List...
Name if not available (100 chars)
Department Computer Science
Name if not available (100 chars)
Proposed supervisor Regan Mandryk (60 chars)
Program of study Ph.D. Comptuer Science: Human Computer Interaction (100 chars)

Second location (if appropriate)

Institution / organization Use the 'List...' button List...
Name if not available (100 chars)
Department Use the 'List...' button
Name if not available (100 chars)
Proposed supervisor (60 chars)
Program of study (100 chars)

Third location (if appropriate)

Country - Microsoft Edge

https://ebiz.nserc.ca/servlet/jaws/nserc_caches/n_cst_popup_service/o

Country

Clear entry Close without selecting

[CANADA](#)

[UNITED STATES](#)

[AFGHANISTAN](#)

[ALBANIA](#)

[ALGERIA](#)

[ANDORRA](#)

[ANGOLA](#)

[ANTARCTICA](#)

[ARGENTINA](#)

[ARMENIA](#)

[AUSTRALIA](#)

[AUSTRIA](#)

[AZERBAIJAN](#)

[BAHRAIN](#)

[BANGLADESH](#)

Scholarship Application



Heuristic Evaluation





Heuristic Evaluation

- Developed by Jakob Nielsen
- Helps find usability problems in a UI design by systematic inspection
- Small set (3-5) of evaluators examine UI
- independently check for compliance with usability principles ("heuristics")
- different evaluators will find different problems
- evaluators only communicate afterwards
- findings are then aggregated
- can perform on working UI or on sketches

Heuristic Evaluation Process

- Evaluators go through UI several times
 - inspect various elements
 - compare with list of usability principles
 - consider other principles/results that come to mind
- Usability principles
 - Nielsen's "heuristics"
 - supplementary list of category-specific heuristics
- Use violations to redesign/fix problems

Heuristic Evaluation Process

- Heuristic evaluation is one of the most straight-forward evaluation methods. The evaluation has three stages:
 1. Briefing session
 2. Evaluation period
 3. Debriefing session
- Sharp, Rogers, and Preece. (2007). Interaction Design: beyond human-computer interaction (2nd Ed.), John Wiley and Sons: West Sussex. pp. 700-701.

1. Briefing Session

- The experts are told what to do. A prepared script is useful as a guide and to ensure each person receives the same briefing.

2. Evaluation Period

- Each expert typically spends 1-2 hours independently inspecting the product, using the heuristics for guidance.
- The experts need to take at least two passes through the interface.
 - The first pass gives a feel for the flow of the interaction and the scope.
 - The second pass allows the evaluator to focus on specific interface elements in the context of the whole product, and to identify potential usability problems.

2. Evaluation Period (Cont)

- If the evaluation is for a functioning interface, the evaluators need to have some specific user tasks in mind so that exploration is focused.
 - Suggesting tasks may be helpful but many experts do this automatically.
 - However, this approach is less easy if the evaluation is done early in design when there are only screen mockups or a specification; the approach needs to be adapted to the evaluation circumstances.
- While working through the interface, specification or mockups, a second person may record the problems identified, or the evaluator may think aloud. Alternatively, they may take notes themselves. Experts should be encouraged to be as specific as possible and to record each problem clearly.

3. Debriefing Session

- The experts come together to discuss their findings and to prioritize the problems they found and suggest solutions.

Heuristics

Heuristics

- H2-1: visibility of system status
- H2-2: match between system & the real world
- H2-3: user control & freedom
- H2-4: consistency and standards
- H2-5: error prevention
- H2-6: recognition rather than recall
- H2-7: flexibility and efficiency of use
- H2-8: aesthetic and minimalist design
- H2-9: help users recognize, diagnose & recover from errors
- H2-10: help and documentation

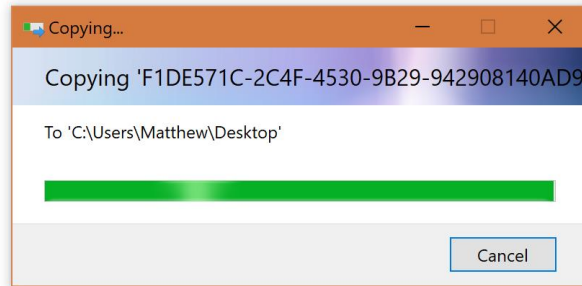
H2-1: Visibility of System Status

- Keep users informed about what is going on
- Example: pay attention to response time:
 - 0.1s: no special indicators needed, why?
 - perceived as “instantaneous”
 - 1s: user’s flow of thought stays uninterrupted, but delay noticed
 - 10s: duration if user to stay focused on action
 - >10s: for longer delays, use percent-done progress bars
 - user will want to perform other tasks while waiting



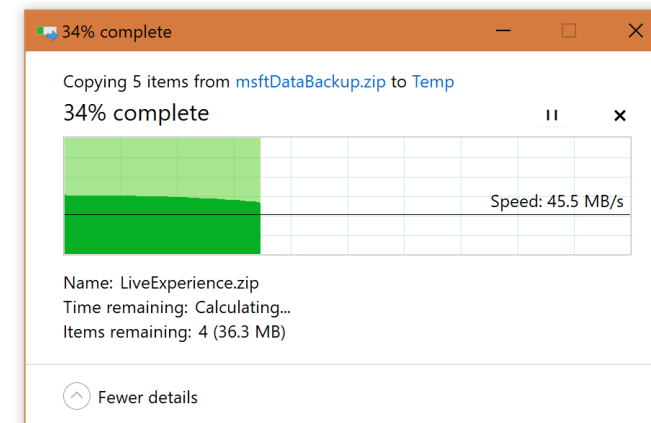
Working on updates
Part 2 of 3: Installing features and drivers
50% complete

H2-1: Examples



Copying from a Media Transfer Protocol (MTP) device on Windows 10

No indication of overall progress

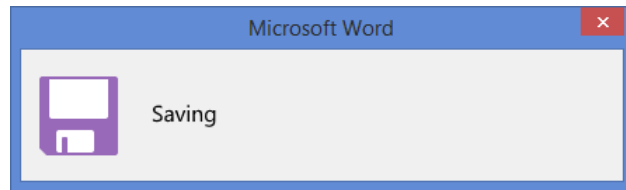


Copying from a zip archive on Windows 10

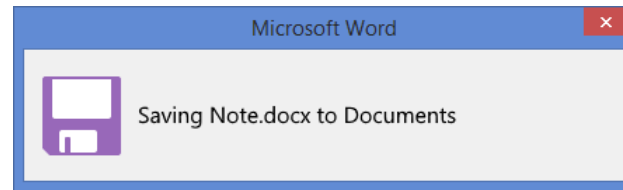
Full indication of progress, files remaining, etc.

H2-1: Examples

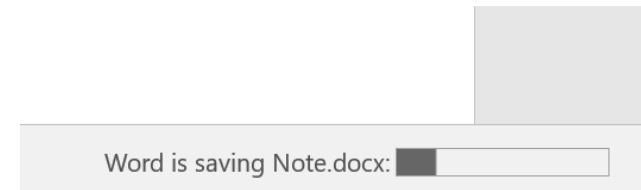
Be as specific as possible given the context



Less Specific

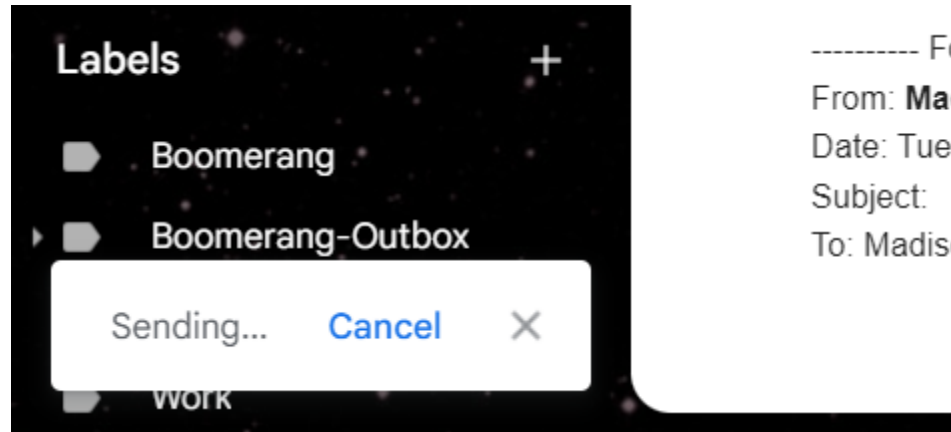


More Specific



In Context

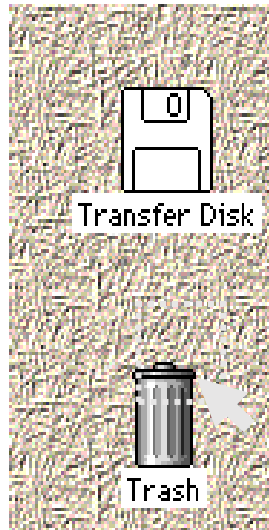
H2-1: Examples



H2-2: Match Between System & Real World

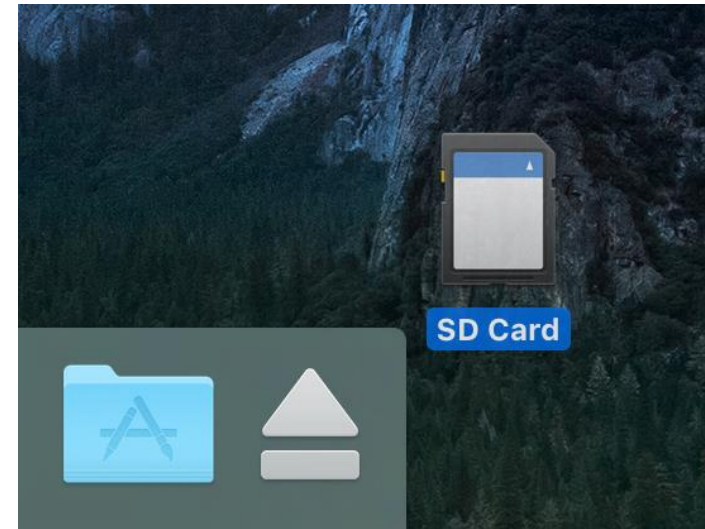
- Speak the user's language
- Follow real-world conventions

H2-2: Examples



Early Mac OS: Drag disc to trash to eject

Doesn't match convention

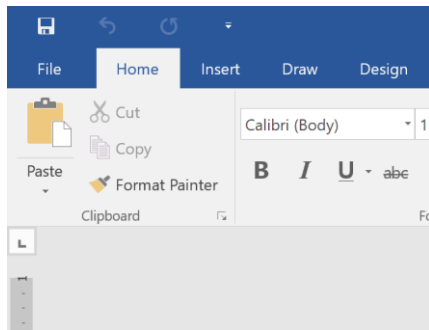


macOS: Trash bin turns into eject button

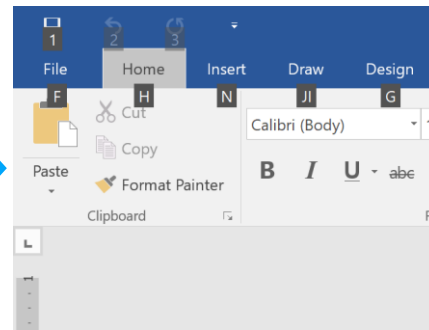
Better match, but still not obvious

H2-2: Examples

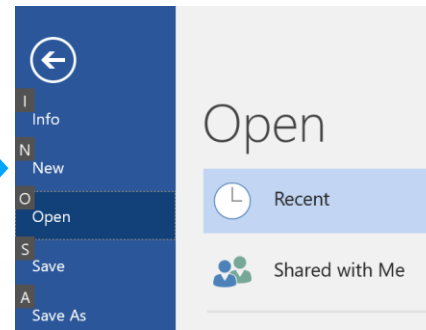
- Use meaningful mnemonics, icons & abbreviations



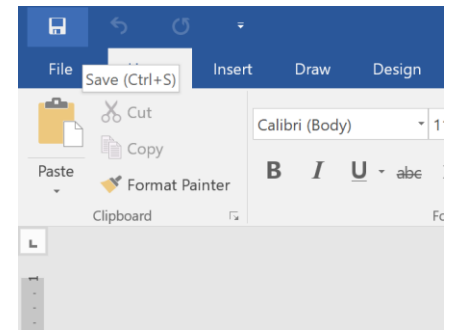
Need to save
from file menu



'F' for file

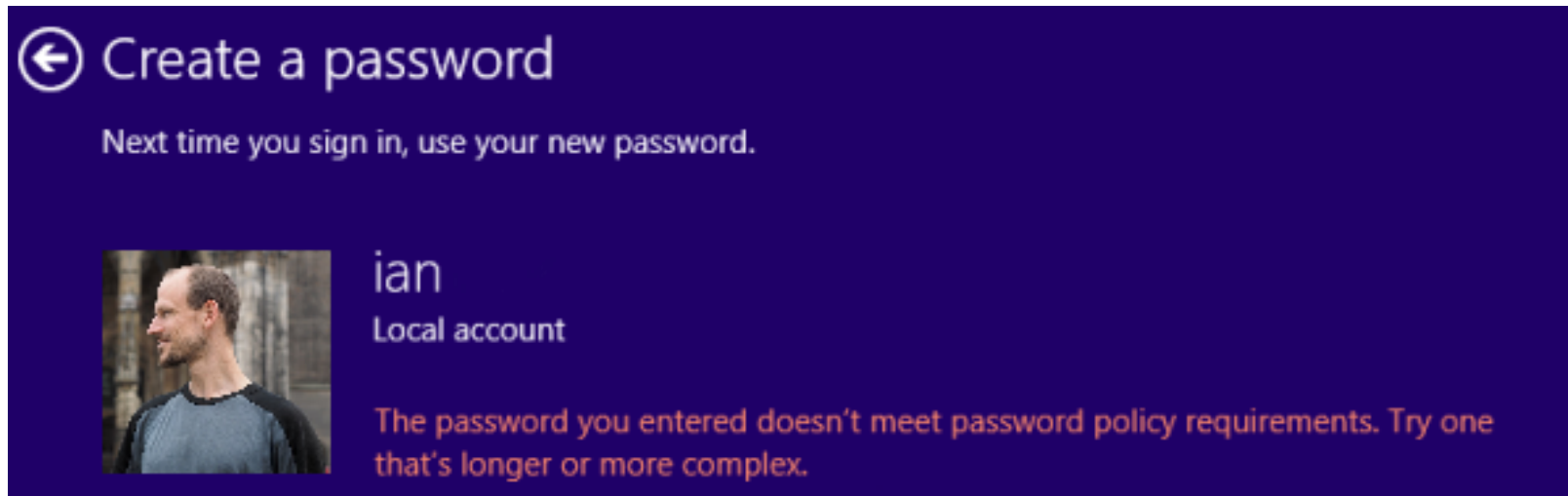


'S' for save



Familiar icon for
saving file. Ctrl-S
uses 'S' for save.

H2-2: Examples



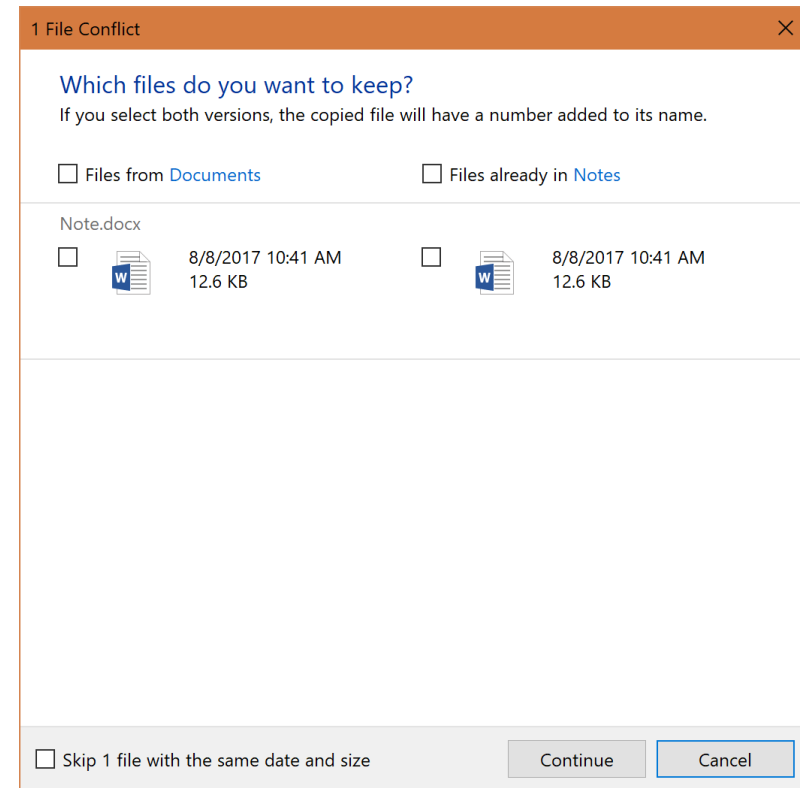
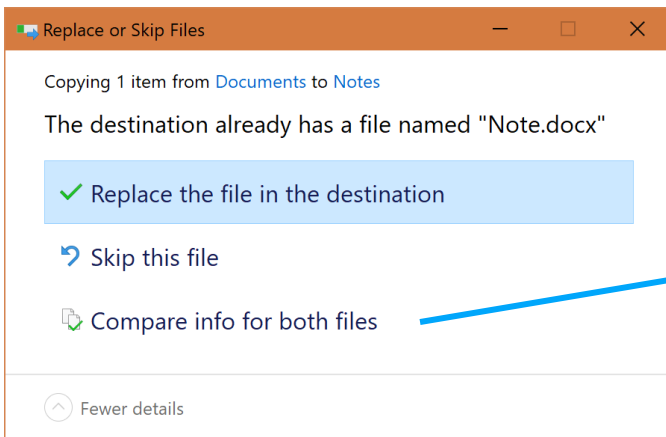
Counter Example: What is a "local account"?
What makes a password "more complex"?

Could use "offline account" and "that uses numbers or symbols" instead.

H2-3: User Control & Freedom

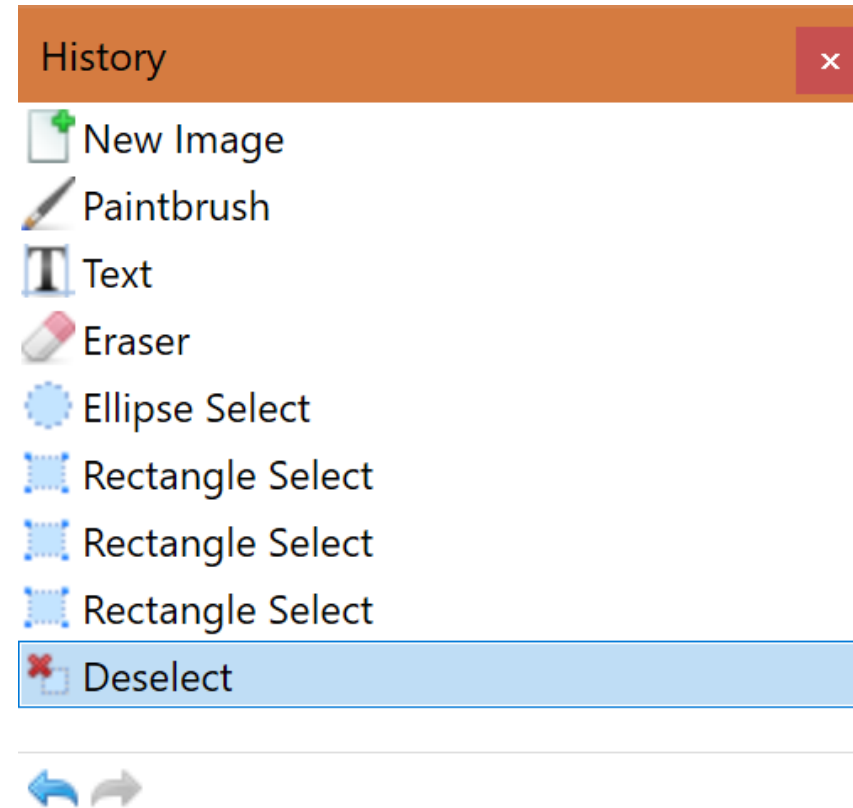
- Provide clearly marked exits for mistaken choices (undo, redo)
- Don't force down fixed paths
- Strategies:
 - **Cancel button** - for dialogs waiting for user input
 - **Universal undo** - can get back to previous state
 - **Interrupt** - especially for lengthy operations
 - **Quit** - for leaving program at any time
 - **Defaults** - for restoring a property sheet

H2-3: Examples






H2-3: Examples

- Image editor undo stack with selection actions included. Where applications allow complex selections, even deselection might be a mistake.

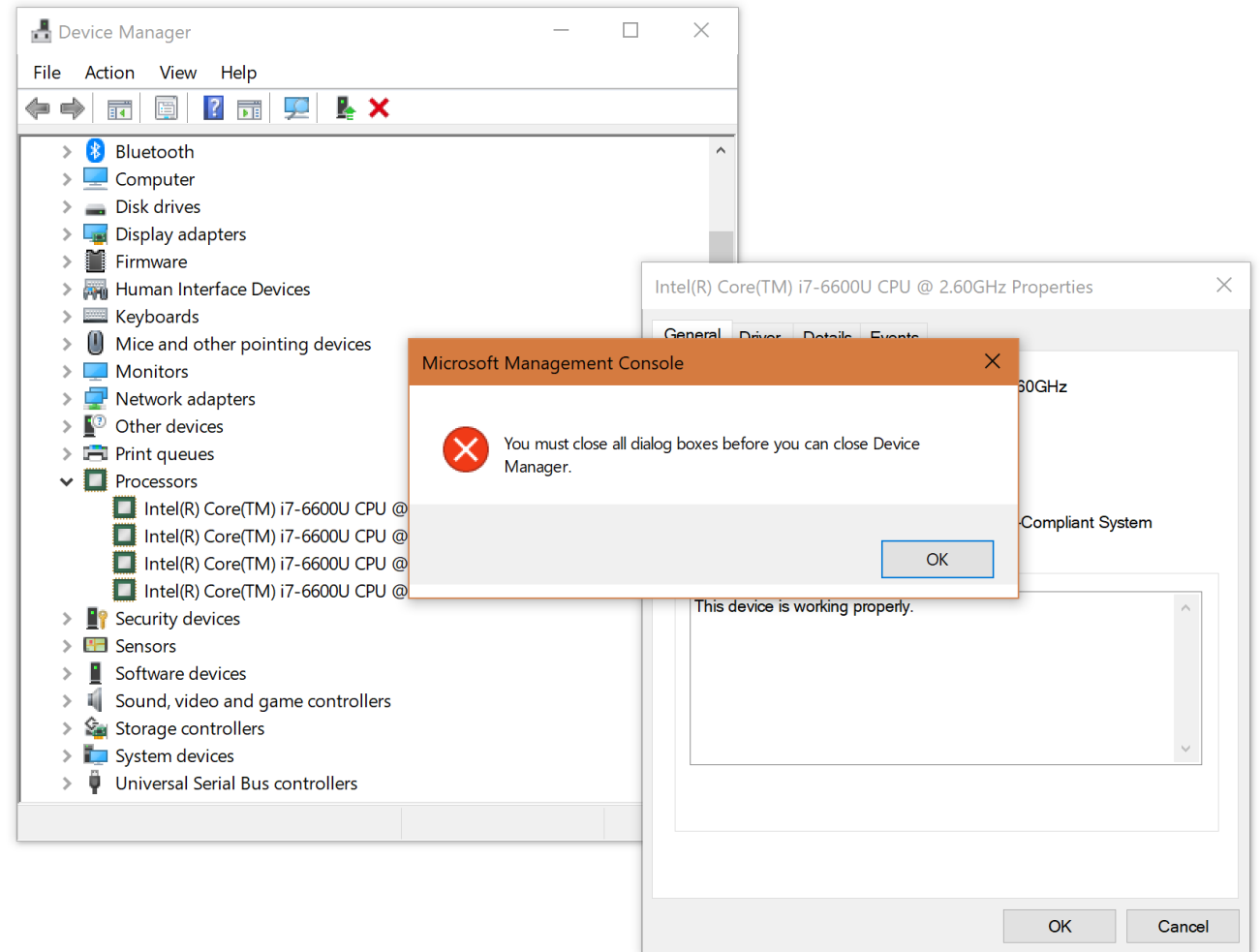


H2-3: Examples

Shopping Cart		Price
Select all items		
<input checked="" type="checkbox"/>	<div><p>AmazonBasics Office Stapler with 1000 Staples - Black #1 Best Seller in Manual Office Staplers In Stock Ships from and sold by Amazon.ca ✓prime <input type="checkbox"/> This will be a gift Learn more Size : 1-pack Pattern Name: Stapler Qty: 6 Delete Save for later See more like this Share</p></div>	\$8.52
<input type="checkbox"/>	<div><p>Totoro Reel Key Holder Only 3 left in stock. Sold by KimoSpace_Inc and Fulfilled By Amazon.ca. ✓prime <input type="checkbox"/> This will be a gift Learn more Qty: 1 Delete Save for later Share</p></div>	\$22.75
<input checked="" type="checkbox"/>	<div><p>Timeless Skin Care 20% Vitamin C + E Ferulic Acid Serum - 1 oz - Lightweight, Non-Greasy Formula - Use Daily to Brighten, Restore & Correct Skin - Recommended for All Skin Types Only 3 left in stock. Ships from and sold by Pictor Star Gift options not available. Learn more Qty: 1 Delete Save for later See more like this Share</p></div>	\$39.19
Subtotal (7 items):		\$90.31

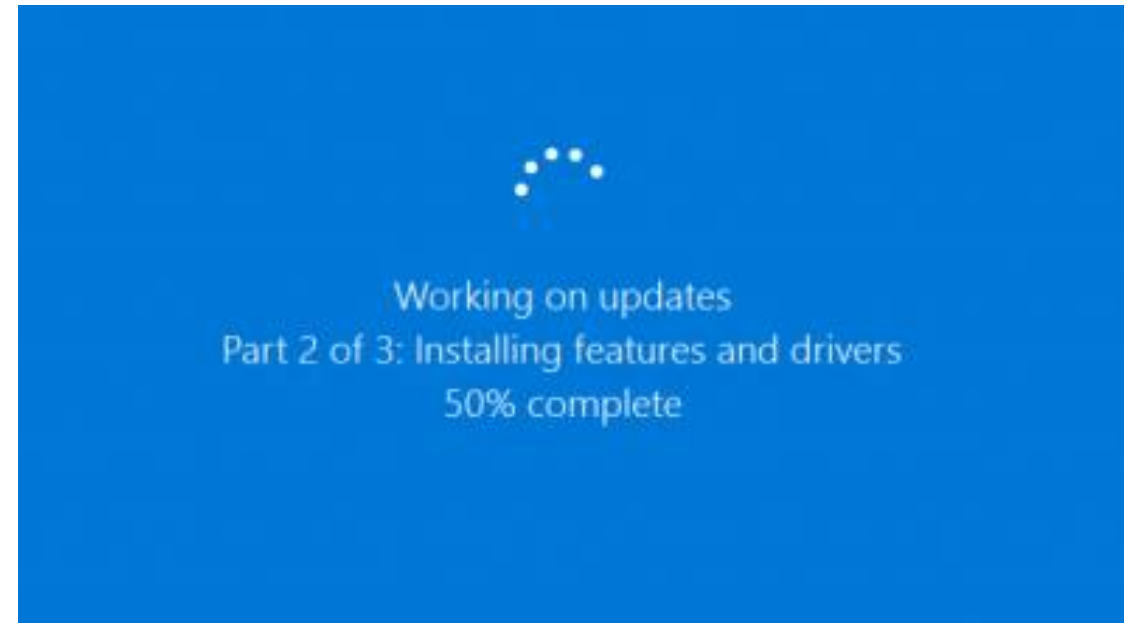
H2-3: Examples

- Counter example:
application cannot be
quit without closing all
dialogue boxes first



H2-3: Examples

- Counter example: cannot interrupt lengthy update install process

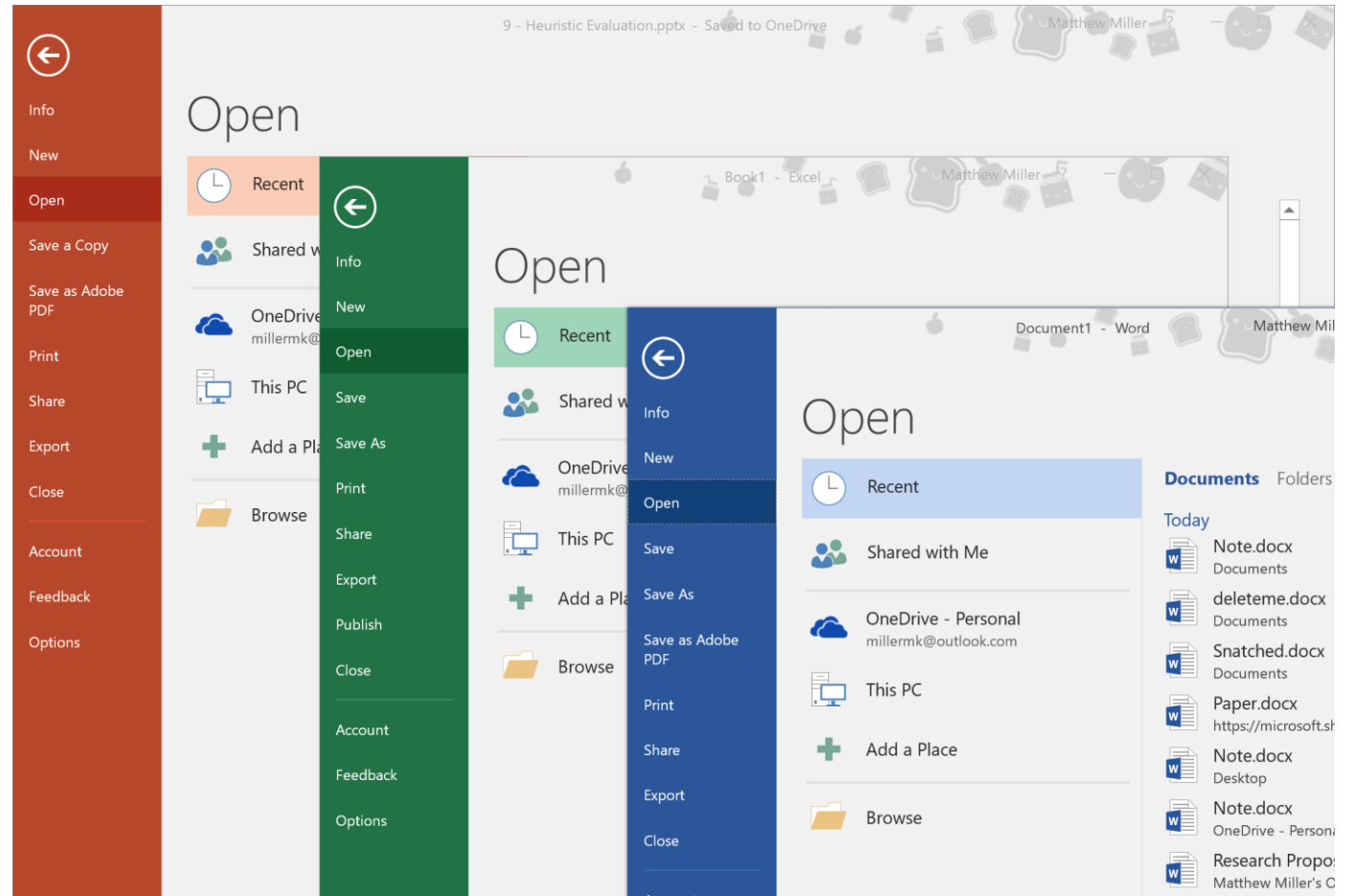


H2-4: Consistency & Standards

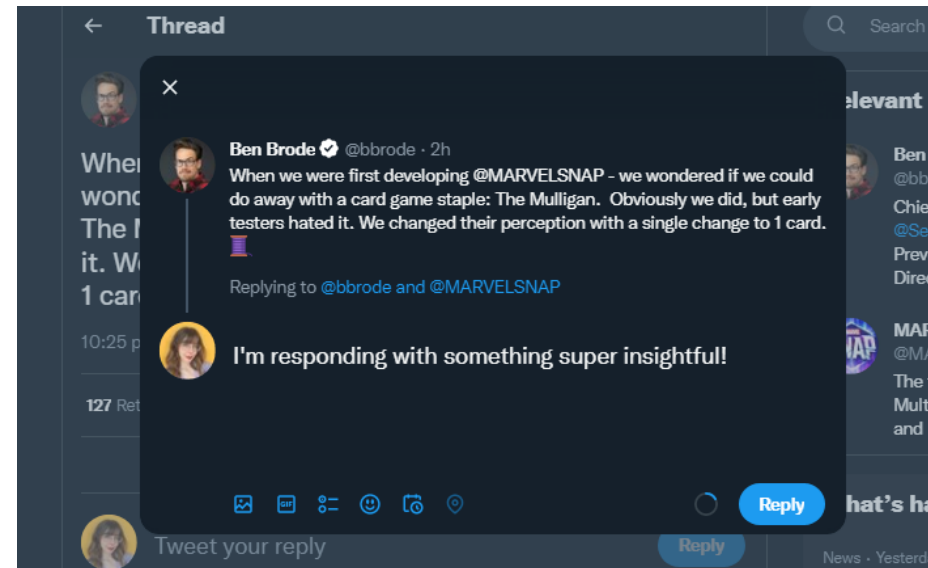
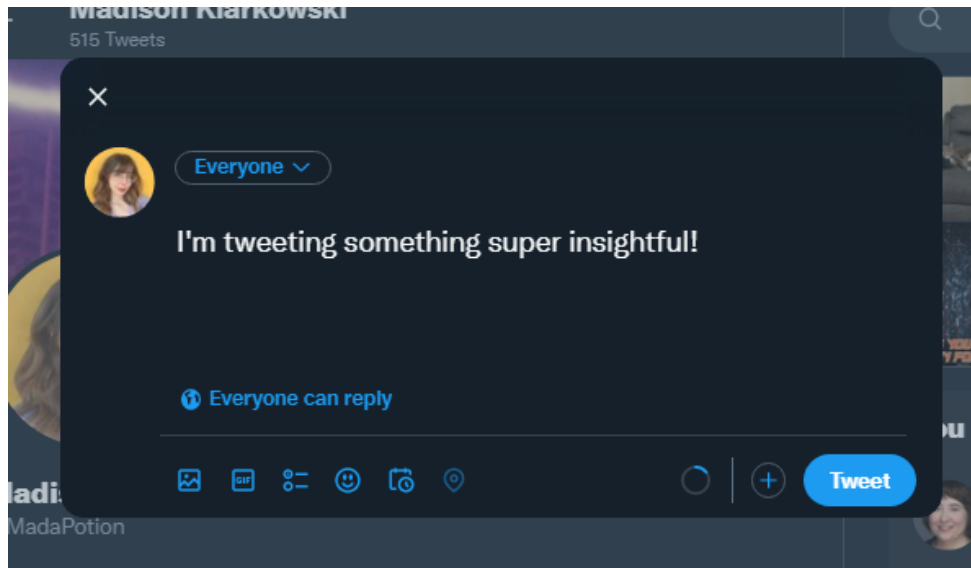
- Consistency of effects (predictability)
 - Same words, commands, actions should always have the same effect in equivalent situations
- Consistency of language and graphics
 - Same info/controls in same location on all screens/dialog boxes
- Consistency of input
 - Require consistent syntax across complete system

H2-4: Examples

- Consistent file menu across PowerPoint, Excel, and Word

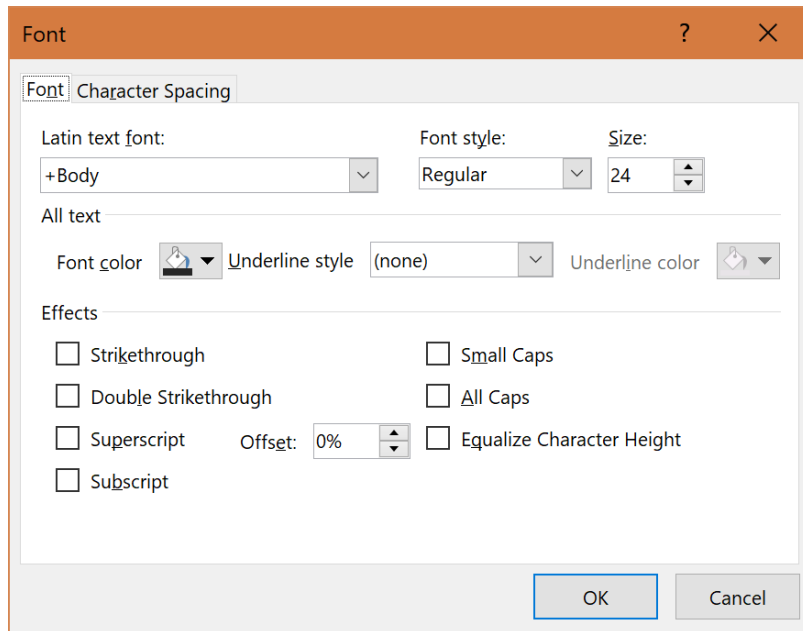


H2-4: Examples

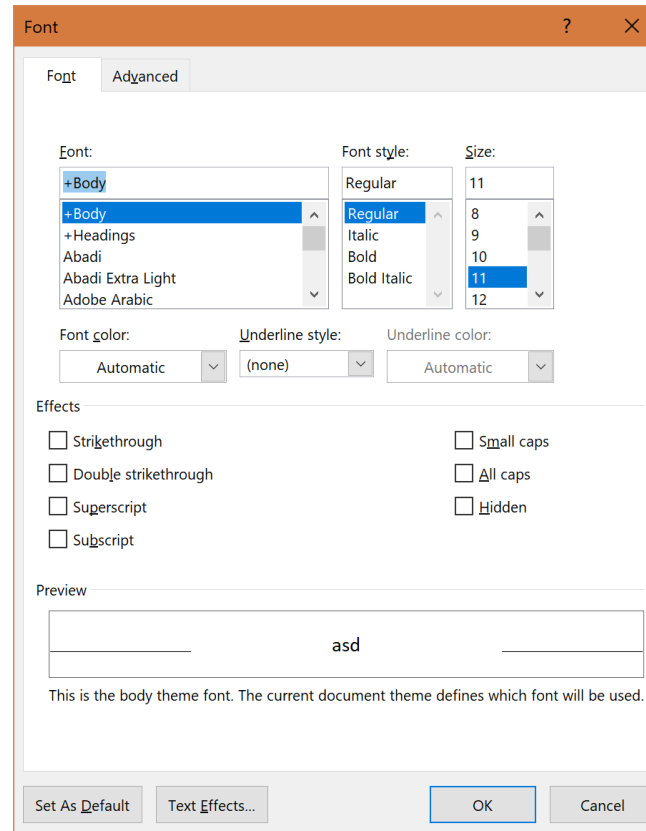


H2-4: Examples

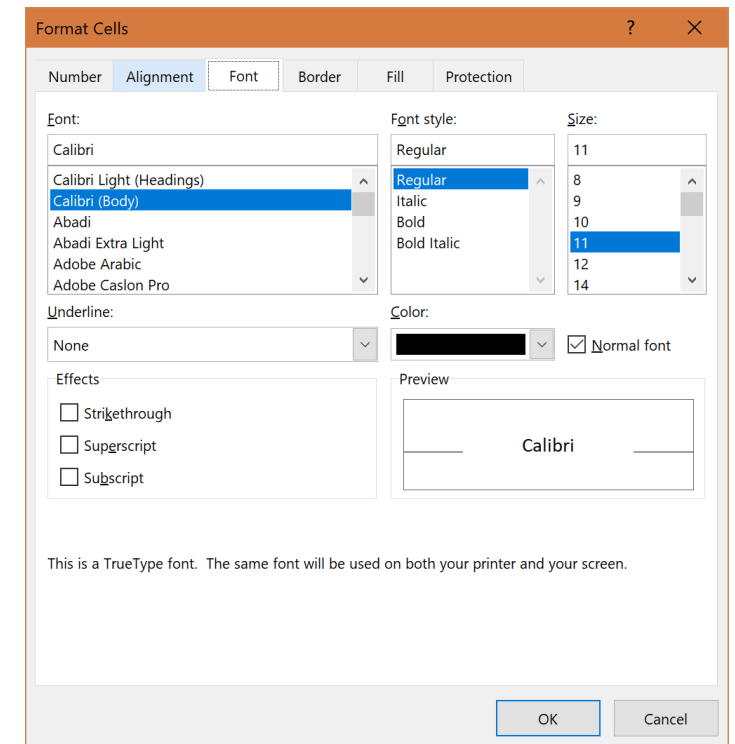
- Counter example: Microsoft Office Font Dialogues



Microsoft PowerPoint Font Menu



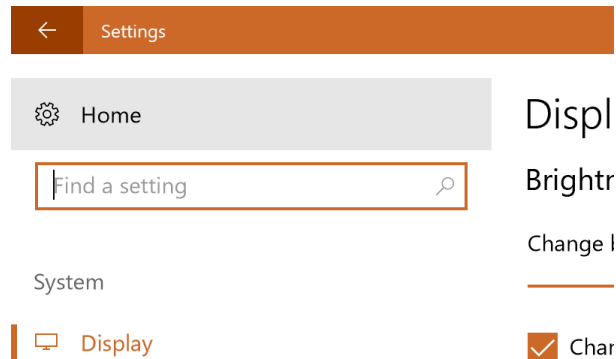
Microsoft Word Font Menu



Microsoft Excel Font Menu

H2-4: Examples

- Counter example: Back button in Windows



in the title bar



or the taskbar

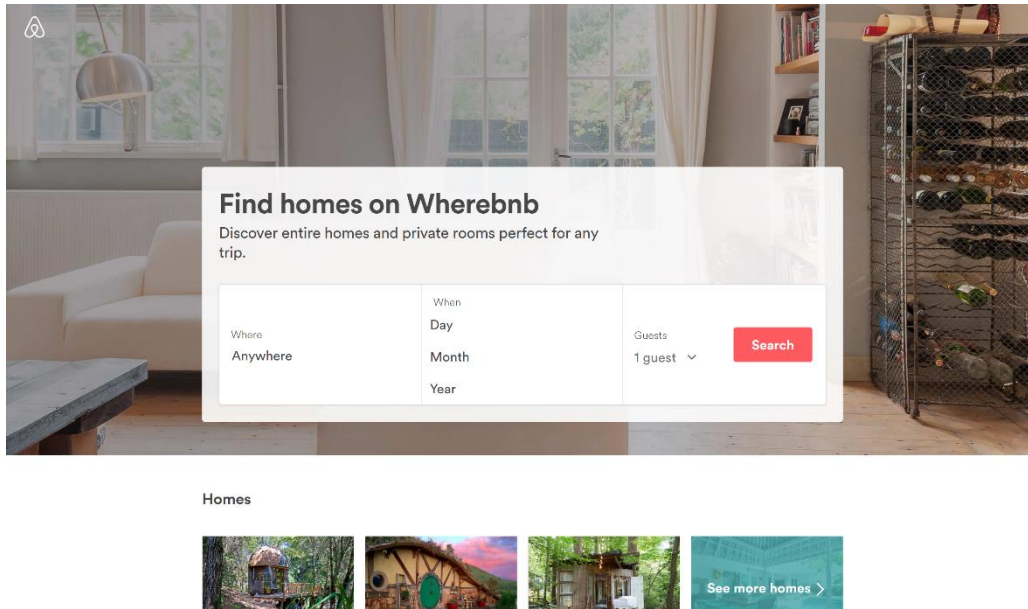


or the app itself

H2-5: Error Prevention

- Try to make errors impossible
 - Only allow legal data or legal commands
- Provide reasonableness checks on input data
 - E.g. on entering order for office supplies, 5000 pencils is an unusually large order. Do you really want to order that many?
 - Did you forget to attach something?

H2-5: Examples

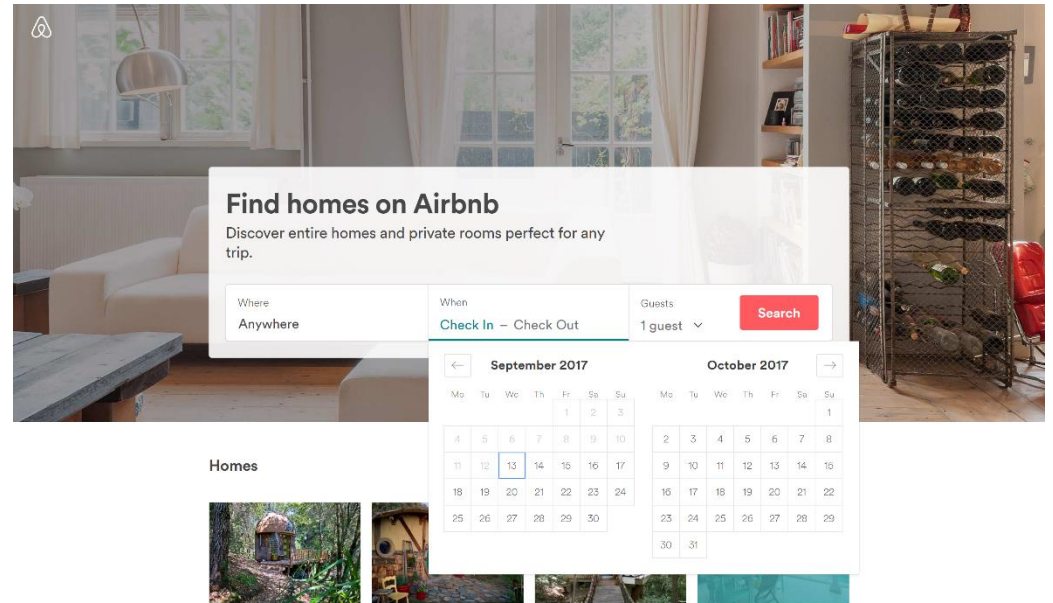


The screenshot shows the 'Find homes on Wherebnb' search interface. It features a background image of a modern living room. The search form is a white box with the following fields:

- Where:** A text field with the placeholder 'Anywhere'.
- When:** A section with three stacked text fields: 'Day', 'Month', and 'Year'.
- Guests:** A dropdown menu showing '1 guest'.
- Search:** A red button.

Below the search form, there is a section titled 'Homes' with three small thumbnail images of different types of accommodations. A 'See more homes >' link is located at the bottom right of this section.

Individual text
fields for date allow
for errant data



The screenshot shows the 'Find homes on Airbnb' search interface. It features a background image of a modern living room. The search form is a white box with the following fields:

- Where:** A text field with the placeholder 'Anywhere'.
- When:** A section with two stacked text fields: 'Check In' and 'Check Out'.
- Guests:** A dropdown menu showing '1 guest'.
- Search:** A red button.

Below the search form, there is a section titled 'Homes' with three small thumbnail images of different types of accommodations. A 'See more homes >' link is located at the bottom right of this section.

Overlaid on the bottom right of the search form is a calendar widget showing the months of September and October 2017. The calendar has a grid layout with days of the week (Mo, Tu, We, Th, Fr, Sa, Su) and dates. The date '13' in September is highlighted with a blue border.

Calendar widget
doesn't allow for
errors

H2-5: Error Prevention

- Two kinds of errors:
 - Mistakes
 - conscious deliberations lead to an error instead of correct solution
 - Slips
 - Unconscious behaviour gets misdirected en-route to satisfying goal (e.g. drive to store, end up at work)
 - Shows up frequently in skilled behavior, usually due to inattention
 - Often arises from similar actions

H2-5: Error Prevention

- Generic system responses for errors:
- 1. Gag
 - Deals with errors by preventing the user from continuing
 - e.g., cannot get past login screen until correct password entered

H2-5: Error Prevention

- Generic system responses for errors:
- 2. Warn
 - Warn people that an unusual situation is occurring
 - e.g., audible bell, alert box
 - When overused, becomes an irritant

H2-5: Error Prevention

- Generic system responses for errors:
- 3. Do Nothing
 - Illegal action just doesn't do anything
 - User must infer what happened
 - enter letter into a numeric-only field (key clicks ignored)
 - put a file icon on top of another file icon (returns it to original position)

H2-5: Error Prevention

- Generic system responses for errors:
- 4. Self-Correct
 - System guesses legal action and does it instead
 - Can lead to a problem of trust (e.g. autocorrect)

H2-5: Error Prevention

- Generic system responses for errors:
- 5. Lets talk about it
 - system initiates dialog with user to come up with solution to the problem
 - E.g. Compile error brings up offending line in source code
 - E.g. Google search

H2-5: Error Prevention

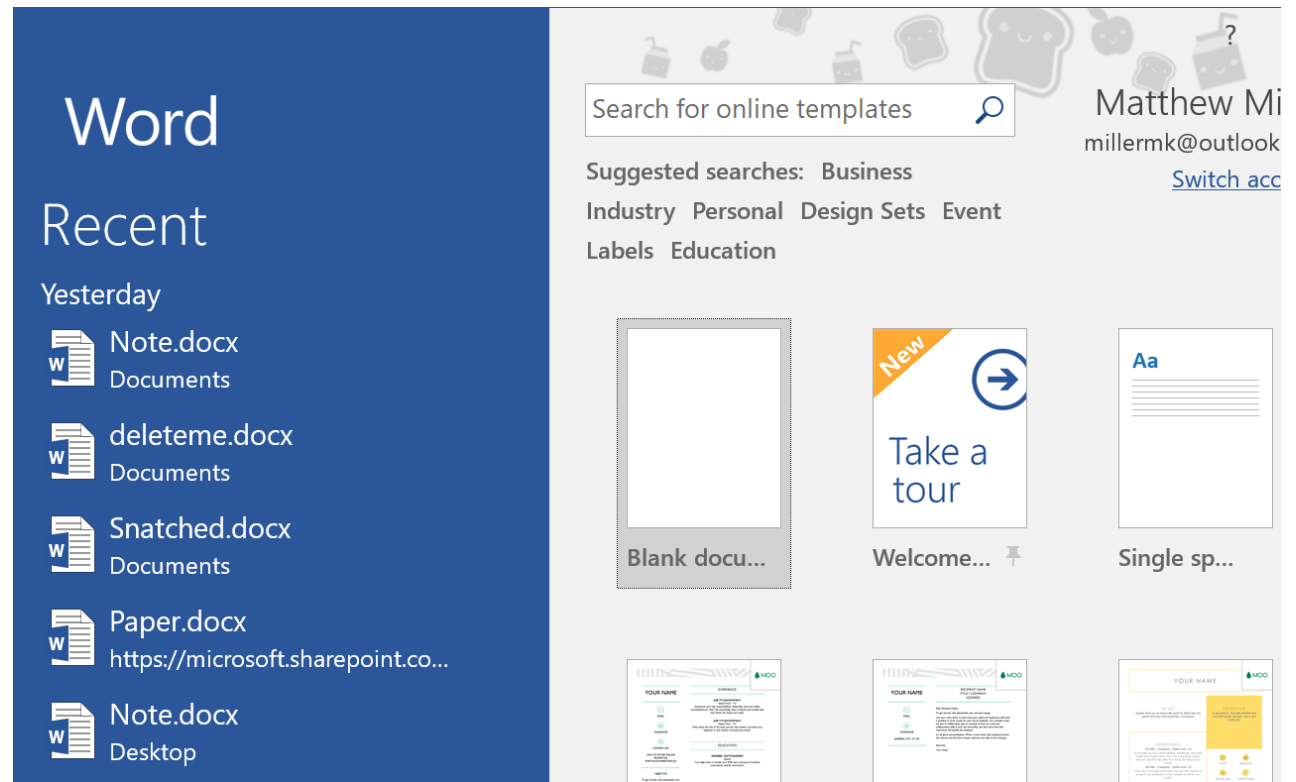
- Generic system responses for errors:
- 6. Teach me
 - System asks user what the action was supposed to have meant
 - Action then becomes a legal one

H2-6: Recognition Rather than Recall

- Computers good at remembering things, people are not!
- Promote recognition over recall
 - menus, icons, choice dialog boxes vs commands, field formats
 - relies on visibility of objects to the user (but less is more!)

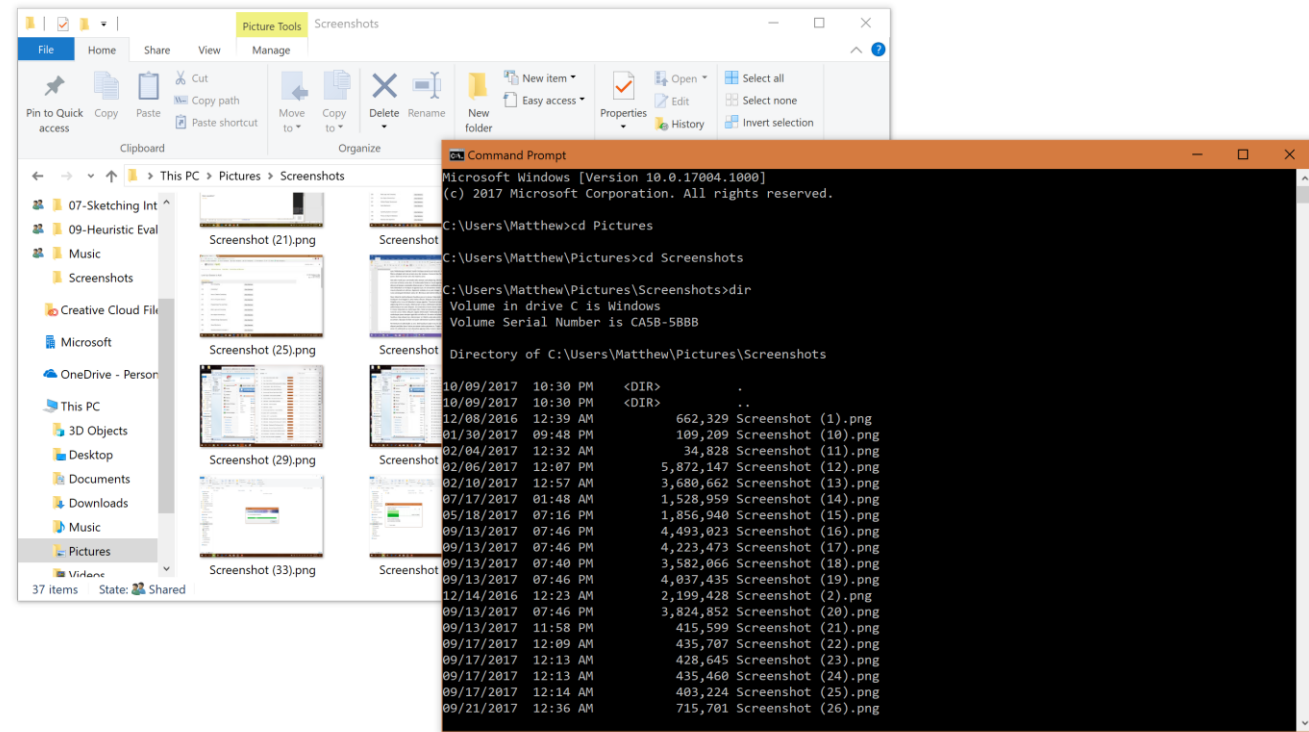
H2-6: Examples

- Recent files in Word allow for recognition rather than recalling what you were working on or where it is stored



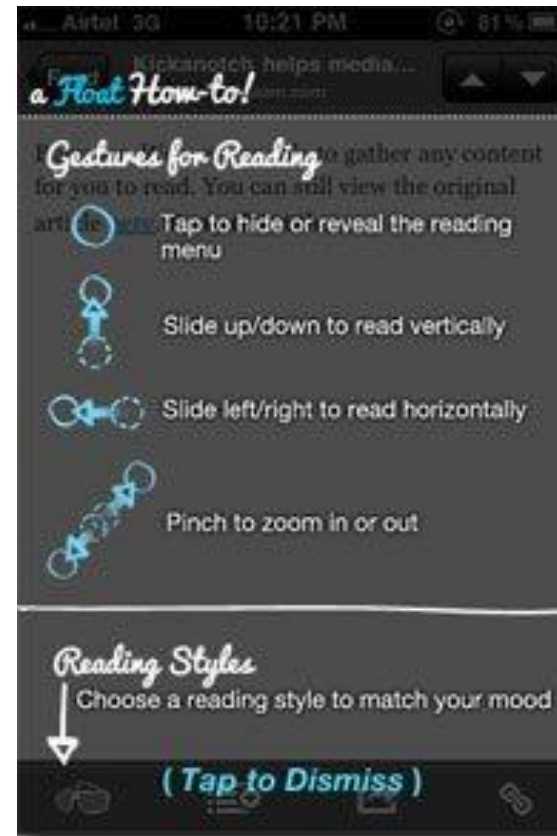
H2-6: Examples

- GUI interfaces allow us to recognize files by icon while command-line interfaces require recall



H2-6: Examples

- Counter example: First-run tutorials in apps which require users to recall a lot of info on using the app.
- Where possible, use clearer labels & affordances instead, or introduce commands as relevant.

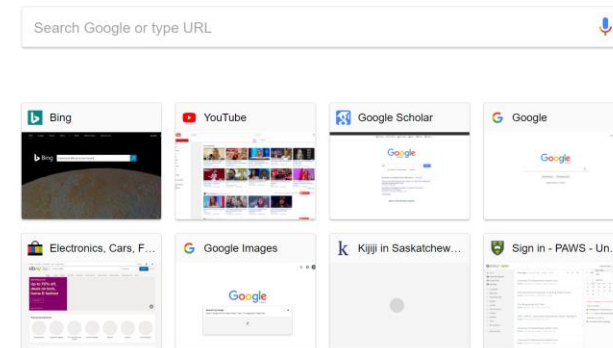


H2-7: Flexibility and Efficiency of Use

- Experienced users perform frequent operations quickly
 - E.g. approximately 60% of page views on the web are revisits
- Strategies:
 - keyboard and mouse accelerators
 - abbreviations
 - command completion
 - menu shortcuts
 - function keys
 - double clicking vs menu selection
 - type-ahead (entering input before the system is ready for it)
 - navigation jumps
 - e.g., going to window/location directly, and avoiding intermediate nodes
 - history systems

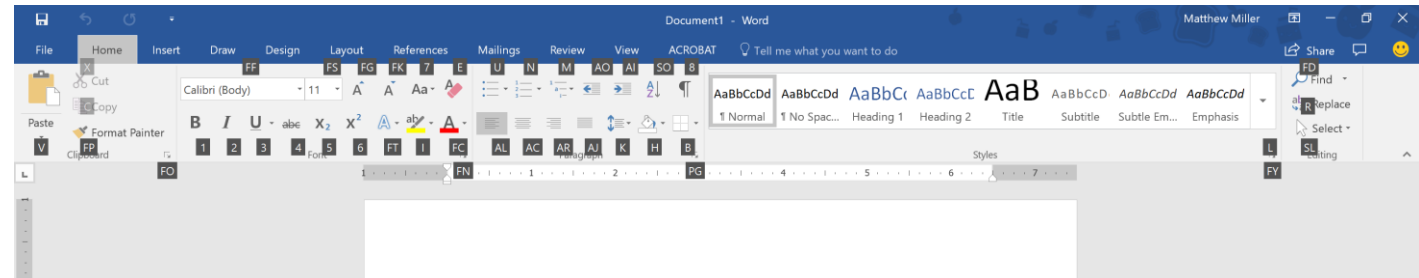
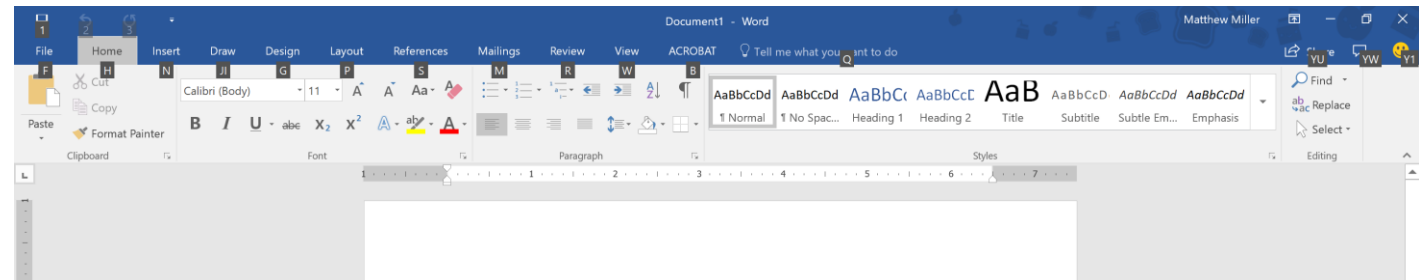
H2-7: Examples

- Browser places frequent sites in new tab, stores history



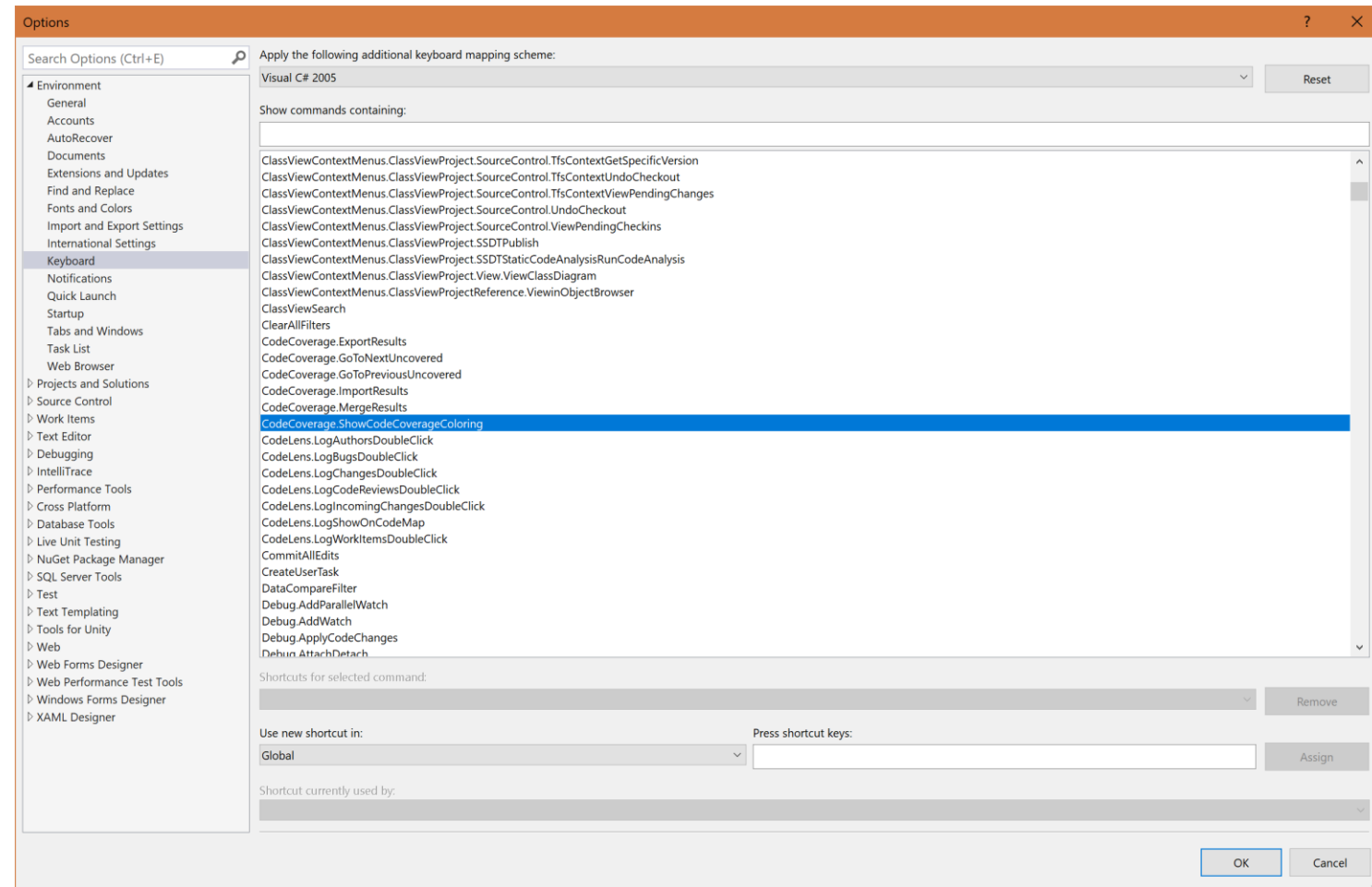
H2-7: Examples

- Keyboard shortcuts for consumer or professional apps



H2-7: Examples

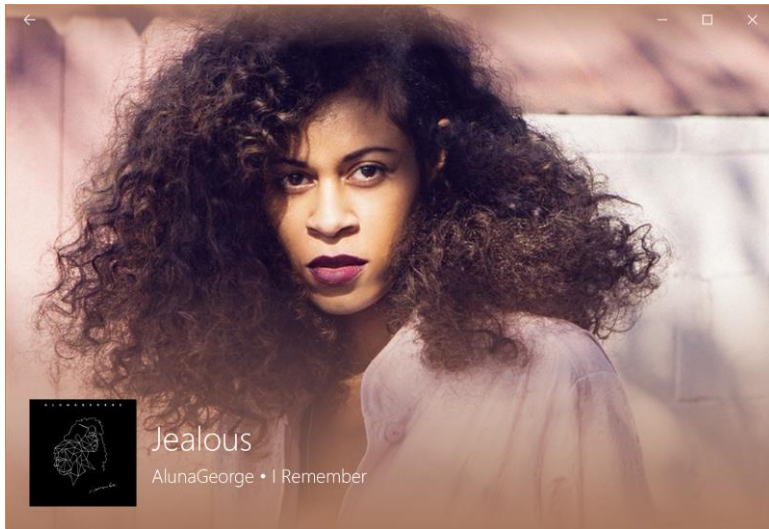
- Keyboard shortcuts for professional or technical applications (fully customizable)



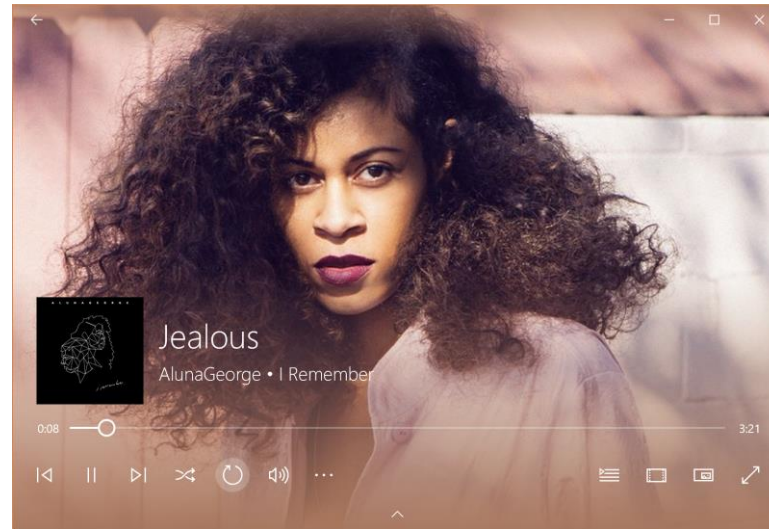
H2-8: Aesthetic and Minimalist Design

- No irrelevant information

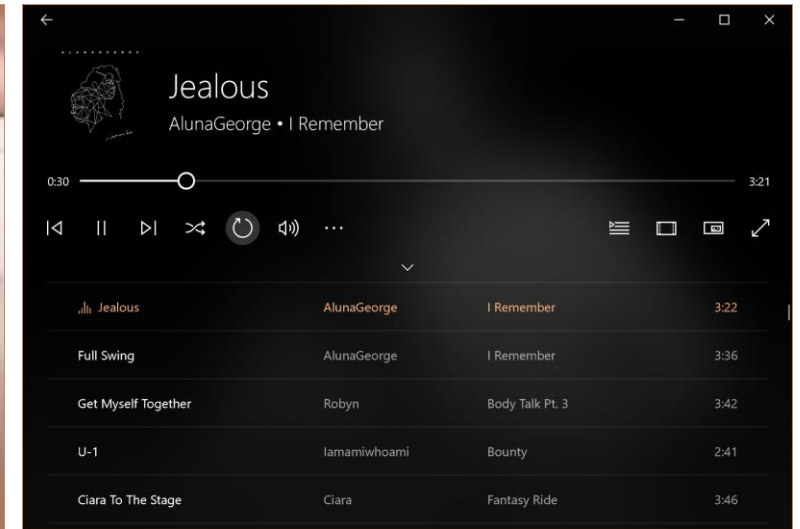
H2-8: Examples



Music app showing song,
artist, album only



Mouseover indicates
interaction, show relevant
controls



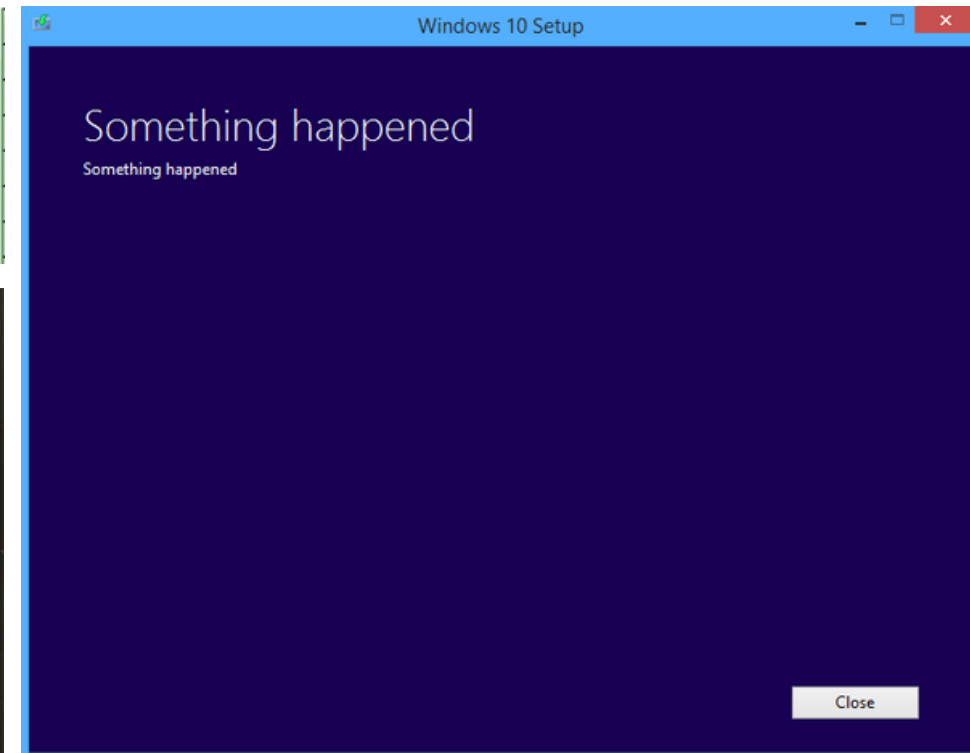
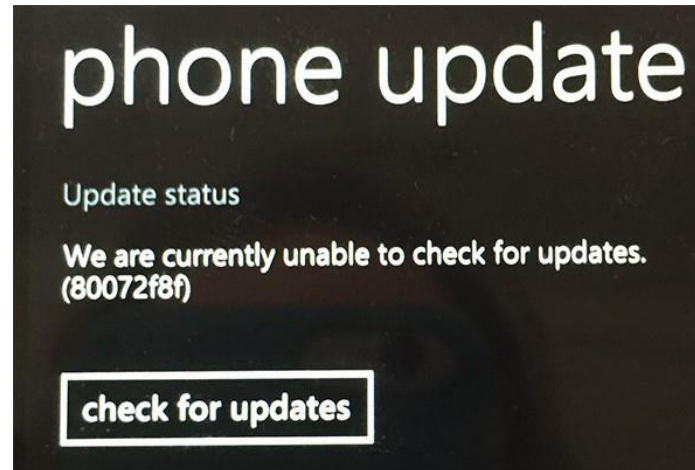
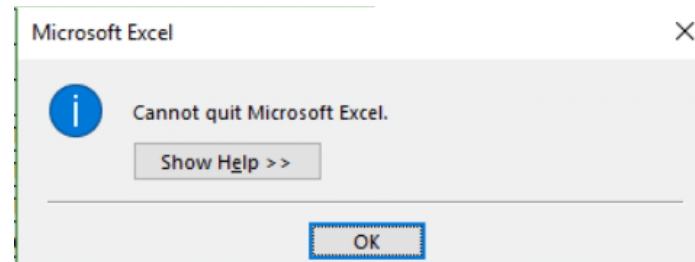
Upcoming tracks
available by swiping up

H2-9: Help Users Recognize, Diagnose, and Recover from Errors

- Use plain language for error messages
- Precisely indicate the problem
- Constructively suggest a solution

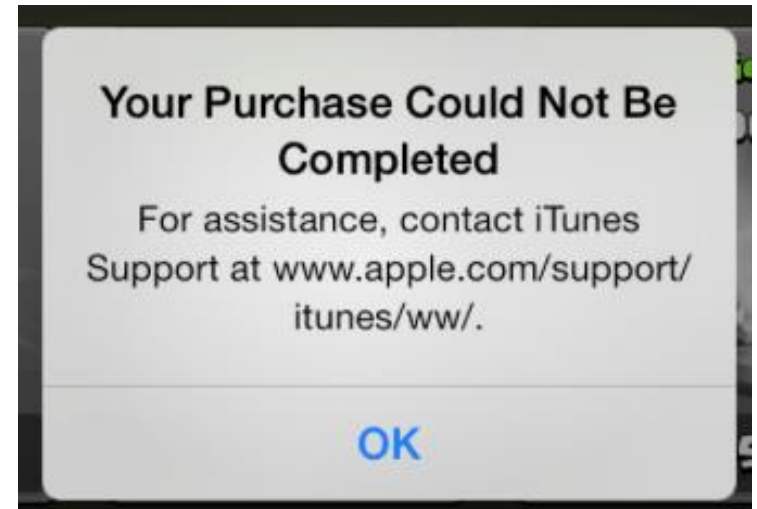
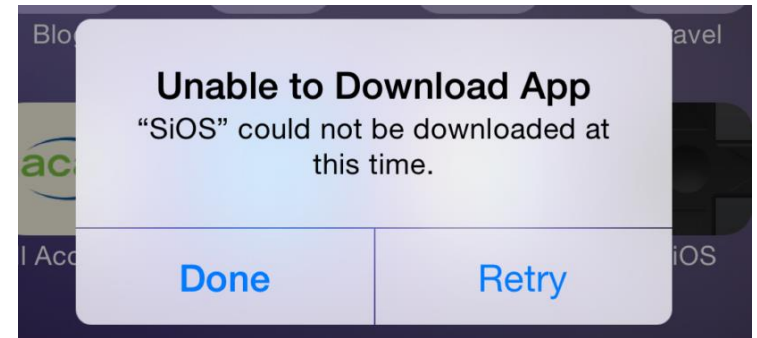
H2-9: Examples

- Counterexamples:
no indication of
what went wrong
or how to fix it



H2-9: Examples

- Errors say what happened and suggest a solution. More specifics could further improve:
 - E.g. "The app could not be downloaded because of an issue on our end, try again later"



H2-10: Help and Documentation

- Help is not a replacement for good design!
- Simple systems:
 - walk up and use; minimal instructions
- Most other systems
 - feature rich
 - simple things should be simple
 - learning path for advanced features

H2-10: Help and Documentation

- Many users do not read manuals (prefer to spend their time pursuing their task)
- Usually used when users are in some kind of panic
 - Provide accessible documentation (online/with application for digital systems)
 - Good search/lookup tools
 - Help specific to current context
- Sometimes used for quick reference
 - syntax of actions, possibilities...
 - list of shortcuts ...

Heuristic Evaluation Process

Heuristics (revised)

- H2-1: visibility of system status
- H2-2: match between system & the real world
- H2-3: user control & freedom
- H2-4: consistency and standards
- H2-5: error prevention
- H2-6: recognition rather than recall
- H2-7: flexibility and efficiency of use
- H2-8: aesthetic and minimalist design
- H2-9: help users recognize, diagnose & recover from errors
- H2-10: help and documentation

Phases of Heuristic Evaluation

- We initially cast the Heuristic Evaluation as three phases:
 1. Briefing session
 2. Evaluation period
 3. Debriefing session

Phases of Heuristic Evaluation

1. **Pre-evaluation training:** give evaluators needed domain knowledge & information on the scenario
2. **Evaluation:** individuals evaluates UI & makes list of problems
3. **Severity rating:** determine how severe each problem is
4. **Aggregation:** group meets & aggregates problems (w/ ratings)
5. **Debriefing:** discuss the outcome with design team

How to Perform Evaluation

- At least two passes for each evaluator
 - first to get feel for flow and scope of system
 - second to focus on specific elements
- If system is walk-up-and-use or evaluators are domain experts, no assistance needed
 - otherwise might supply evaluators with scenarios
- Each evaluator produces list of problems
 - explain why with reference to heuristic or other information
 - be specific & list each problem separately

How to Perform Heuristic Evaluation

- Example Problems:
 - Can't copy info from one window to another
 - violates "Minimize the users' memory load" (H1-3)
 - fix: allow copying
 - Typography uses different fonts in 3 dialog boxes
 - violates "Consistency and standards" (H2-4)
 - slows users down
 - probably wouldn't be found by user testing
 - fix: pick a single format for entire interface

How to Perform Heuristic Evaluation

- Why separate listings for each violation?
 - risk of repeating problematic aspect
 - may not be possible to fix all problems
- Where problems may be found
 - single location in UI
 - two or more locations that need to be compared
 - problem with overall structure of UI
 - something that is missing
 - common problem with paper prototypes

Severity Ratings

- Used to allocate resources to fix problems
- Estimates of need for more usability efforts
- Combination of
 - frequency
 - impact
 - persistence (one time or repeating)
- Should be calculated after all evals. are in
- Should be done independently by all judges

Severity Ratings

- Severity Rating Scale
 - 0 - don't agree that this is a usability problem
 - 1 - cosmetic problem
 - 2 - minor usability problem
 - 3 - major usability problem; important to fix
 - 4 - usability catastrophe; imperative to fix

Severity Ratings (Example)

- 1. [H1-4 Consistency] [Severity 3][Fix 0]
 - The interface used the string "Save" on the first screen for saving the user's file, but used the string "Write file" on the second screen. Users may be confused by this different terminology for the same function.

Debriefing

- Conduct with evaluators, observers, and development team members
- Discuss general characteristics of UI
- Suggest potential improvements to address major usability problems
- Dev. team rates how hard things are to fix
- Make it a brainstorming session
 - little criticism until end of session

Self Guided vs Scenario Exploration

- Self-guided
 - Open-ended exploration
 - Not necessarily task-directed
 - Good for exploring diverse aspects of the interface, and to follow potential pitfalls
- Scenarios
 - Step through the interface using representative end user tasks
 - Ensures problems identified in relevant portions of the interface
 - Ensures that specific features of interest are evaluated
 - But limits the scope of the evaluation - problems can be missed

Individuals vs Teams

- Nielsen recommends individual evaluators inspect the interface alone
- Why?
 - evaluation is not influenced by others
 - independent and unbiased
 - greater variability in the kinds of errors found
 - no overhead required to organize group meetings

Number of Inspectors

- Average over six case studies shows a single evaluator found:
 - 35% of all usability problems
 - 42% of the major problems
 - 32% of the minor problems
- Not great, but finding some problems with one evaluator is much better than finding no problems with no evaluators!

Number of Inspectors

- Number of inspectors needed varies according to:
 - difficulty of the interface being evaluated
 - the expertise of the inspectors
- Average problems found by:
 - Novice evaluators (no usability expertise) - 22%
 - Regular specialists (expertise in usability) - 41%
 - Double specialists (experience in both usability and the particular kind of interface being evaluated) - 60%
 - also find domain-related problems
- Tradeoff
 - Novices poorer evaluators, but cheaper!



Situating Heuristic Evaluation



HE vs. User Testing

- HE is much faster
 - 1-2 hours per evaluator vs. days-weeks
- HE doesn't require interpreting users' actions
- User testing is far more accurate (by def.)
 - takes into account actual users and tasks
 - HE may miss problems & find false positives
- Good to alternate between HE & user testing
 - find different problems
 - don't waste participants

Summary: (Dis)advantages of HE

- Advantages:
 - The “minimalist” approach
 - General guidelines can correct for majority of usability problems
 - Easily remembered, easily applied with modest effort
 - Discount usability engineering
 - Cheap and fast way to inspect a system
 - Can be done by usability experts
- Disadvantages:
 - Principles must be applied intuitively and carefully
 - Can’t be treated as a simple checklist
 - Subtleties involved in their use
 - Doesn’t necessarily predict users’/customers’ overall satisfaction
 - May not have same “credibility” as user test data

Summary: (Dis)advantages of HE

- Research results:
 - 3-5 evaluators usually able to identify 75% of usability problems
 - user testing and usability inspection have a large degree of non-overlap in the usability problems they find (i.e., it pays to do both)
- Cost-benefit:
 - usability engineering activities often expensive / slow; but some can be quick / cheap, and still produce useful results
 - usability inspection turns less on what is “correct” than on what can be done within development constraints
 - ultimate trade-off may be between doing no usability assessment and doing some kind

Further Reading

- Books
 - Usability Engineering, by Nielsen, 1994
- Web Sites & mailing lists
 - useit.com
 - UTEST mail list



We Try



We Try: System

- University of Saskatchewan course registration
- Mobile browser



We Try: Identify Problems

- Perform a HE of the U of S registration system on a phone

[illegible]