

# TITLE: THE UX book

---

*Subtitle: Creating a better experience*

Preface: About User Experience

Foreword: By someone

Introduction: The whole book in 12 pages

## *Principles of Interaction Design*

Chapter 1: Alternate Paths to Enlightenment

**Chapter 2: The UI Five in Detail**

Chapter 3: Information Architecture

Chapter 4: Mental Models

Chapter 5: Respond Quickly and Nicely

Chapter 6: Make It Beautiful

Chapter 7: Keep It Simple, Keep It Safe

## *The Process of Design*

Chapter 8: Good vs. Great

Chapter 9: You vs. The HiPPO, (Highest Paid Person's Opinion)

Chapter 10: Good testing vs. bad testing

Chapter 11: Public sites vs. Web Applications

Chapter 12: Design Group vs. Engineering Group

## *Enabling technologies and techniques*

Chapter 13: The jQuery Idea

Chapter 14: The Adobe Idea (Flex, Flash and AIR)

Chapter 15: Progressive Enhancement

Chapter 16: Semantic Markup

## *Summary and Inspiration*

Chapter 17: The Interaction Design Manifesto

Recommended Reading & Resources

Acknowledgments

Index

## Chapter 2

### *The UI Five in Detail*



*I have not failed. I've just found 10,000 ways that won't work.*  
- Thomas Edison

---

*I ain't no doctor, but I knows when I'm losing me patients.*  
- Popeye

---

## Muddling through Nature

Thomas Edison was a genius, but more importantly, he had patience. People in general do not have this quality. Patience allowed Edison to find the myriad of ways that didn't work on a particular invention and yet he kept trying to find that one way that finally succeeded. In this world, God (a.k.a. Laws of Nature) is the ultimate User Experience Designer. These laws (or designs) force scientists to muddle, to experiment and try every possibility, until one works. Muddling is a natural instinct of human beings and has taken us from the cave to the field to the skyscraper. Muddling is how we progress as a society. However, Muddling is also an extremely frustrating experience for most people.

When you design and build your web application, you are God. You get to decide how the world works. You get to decide laws of action and reaction. If you decide a button does not do anything, then the user must live in that reality. Each decision you make lays down the laws of nature for your application. Granted, users don't generally live in a single application, but these days, the majority of people's lives are spent in front of a screen.

If you want to be an evil, vengeful deity, then you should foster superstition\* and reign vengeance down on the user for every misstep. You should make them feel guilty for every mistake. However, if you want to be a loved, compassionate supreme being, then you should design with the user in mind. Design so that they don't have to muddle all the time. Design so that mistakes are translated into the correct answer as if by miracle. My plea to you is to be a good designer. Create a just and pleasant world for the user to live. Make the world fun and friendly, with soft surfaces so that they don't hurt themselves.



This chapter defines the tools to build your new utopia. It describes in details the controls you should use to allow a user to figure out the path to success on their first try. It also establishes patterns and best practices for these controls. The toolkit includes: Buttons, Menus, Keyboard Shortcuts, Context Menus, Drag and Drop.



**\*** Superstition in users happens when a user is muddling and does something unrelated (like take a drink from a coffee cup) while simultaneously pressing the correct button in your application. The user mistakenly correlates the drinking of the coffee with the success of the application. From that point forward the user will make sure to drink coffee while pressing that button. Don't laugh. This happens every single day.

## Buttons

Buttons are the most overused of all user interface elements. If I had a nickel for every time I heard, “Oh, we can just add a button for that”, I could retire in luxury. The reason they are so common is that they are the easiest of all of the interface elements to add. Slap an anchor link in your HTML and “bam!” instant functionality.

There are several different flavors of buttons. The first is a simple text link. It can have any kind of styling but the basic fact is that it has text only. [Print](#) On the opposite end of the spectrum you can have an image to represent the functionality to save space.  In most tests the combination of word and icon is most recognizable.  [Print](#)

There are lots of icon libraries out there to help lend color and clarity to your buttons. Choose your images wisely. You are creating a mental link for the user to understand what that button does. Furthermore, when you are creating buttons, think about how they work. Do they open up a modal dialog? Do they have the ability to undo? It is wise to create standards to that buttons work in a predictable fashion.

Let’s look at samples of buttons to see how they evolved and how they have been abused.

### CASE STUDY: MICROSOFT OFFICE

Perhaps the most scrutinized and discussed user interface in history. It has been the standard by which every other program is judged. This is perhaps unfair to Microsoft. Sure, a lot of their programs have annoying UI tendencies and Microsoft certainly has made plenty of mistakes. However, who among us has not made the same mistakes at some point? Therefore, I want to rip apart their UI in the most respectful way. Who knows, maybe I will design the next version of Word? Well, I can dream anyway.

Microsoft Word popularized the toolbar concept (The first toolbar appeared on the Xerox Alto computer in 1973). Toolbars are strips at the top or side of the program that has buttons which, when clicked, would interact with the contents in the main area. The problem with this approach has been scalability, which leads to “button overload”. The image on the right demonstrates what happens when you overuse this method.

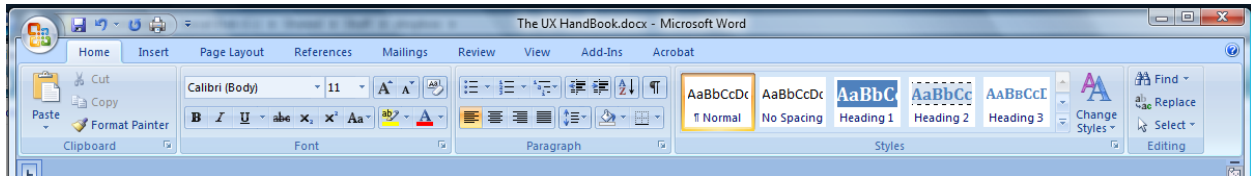
Things that look different  
should act different;  
Things that look the same  
should act the same.

- *Larry Marine*



Notice how your eyes automatically jump to certain areas in the button wilderness. Buttons with icons and words get the most attention.

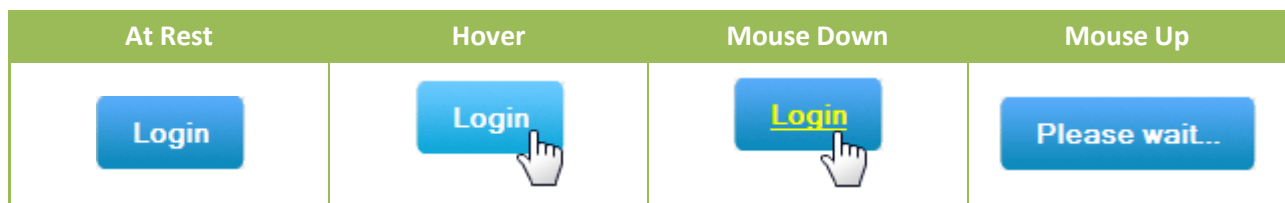
In the latest version of Microsoft Word 2007, Microsoft has tried to address this problem. They pioneered a concept called the “Ribbon” which organizes all of the buttons into tabbed sections. This is an extremely controversial UI experiment, but Microsoft assures us that they have tested it thoroughly.



My own experience with it tells me that it has pros and cons. With the ribbon, the user is given fewer choices about where items can live, which diminishes flexibility and customization. Therefore, I find myself playing “hide and seek” with a certain buttons. Often, it takes me a long time to find the control I am looking for. Lastly, I miss the predictability and usefulness of the menu bar. You can no longer use any menus to find controls. All of these effects just drive home the point that buttons can quickly get out of control and harm the user experience.

It’s not all doom and gloom. Buttons are very useful. They are easily discoverable and have a good basic affordance. They look clickable. That, by itself, is half the battle.

Buttons can actually have many states. There are many interesting moments and looks that a button should have. The idea of “interesting moments”\* was pioneered by Bill Scott. The concept is to think deeply about all of the subtle details in interaction design. Even when clicking a button, there are many interesting moments.



Rest is defined as a button that is enabled, but not engaged. In other words, it is a button sitting there, waiting to be clicked. It can have icons or not, but it must look clickable. This can be achieved with a variety of styles, such as underlining, borders, and 3D. Anything that makes the user think, “Hey! That looks like I can click on it.” In the above example, we achieve that by rounding the corners and applying a subtle fade. This is all it takes to make the object look clickable.

\* <http://looksgoodworkswell.blogspot.com/2005/12/storyboarding-interesting-moments.html>

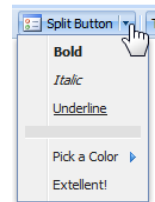
Hover is a reinforcing agent for the user. If the button does something when you put your mouse over it, then the system is enticing the user to click on it. In the above example, we made the background a lighter color. Small changes like this are very noticeable to the user. Reserve the really big (red) changes for errors.

Mouse down is the moment that the user clicks on the mouse, but doesn't yet release. Interestingly users often keep the button down for a few seconds as they think, "Do I *really* want to press this thing? What if it does something bad?" This moment for the button is also referred to as "active".

Mouse-up is the final stage. The button has been clicked. A good technique is to disable the button and indicate that the system is working on the request. Of course, this depends on how quick your system is. There is no need for this step if the effect is instantaneous. However, if there is a delay, then the "please wait" function would be wise.

Other interesting moments are applicable for more advanced buttons. A button that is normally enabled, but currently does not work is called "disabled". Usually these kinds of buttons look "dead", which if you can imagine a dead button, it would be ashen colored and lifeless. No hover (active) state for these fellows.

One special kind of button is a combination of a button and menu, otherwise known as a "mutton" or "split button". These guys are buttons that have a drop down trigger off to the right. They can be used for a variety of applications. Another special kind of button is a toggle. A toggle can be pushed in or out. Imagine a power button on an older stereo. If the button is pressed in, then the stereo was on, if it is out then the stereo was off.

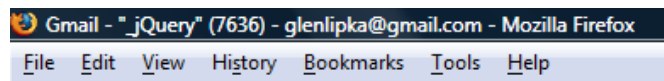


A question you might be asking is, "Which button do I use in which cases?" That's a good question. It's one that each interaction designer struggles with each day. No one has a set of rules that makes decision making as simple as looking up the answer. This is the part where experience as a UX designer will teach you which tools fit which cases.

My own process is to apply different choices to a rapid prototyping tool, like paper or PowerPoint. Then I imagine the user clicking on those choices. Empathy is a critical factor in determining the correct UX choice. This is discussed in more detail in Chapter 4, Mental Models.

## Menus

Menus were invented at Xerox Parc in the 1970's. (It seems like everything was invented there!) They are very small strips at the top of most programs. The dominant wording for Windows programs starts with File, Edit and View and ends with Help. Menus are extremely powerful for the simple reason that they can hold an enormous amount of functionality and not take up screen real estate. However, the space is not infinite and you would be wise to keep the menus from getting too long.

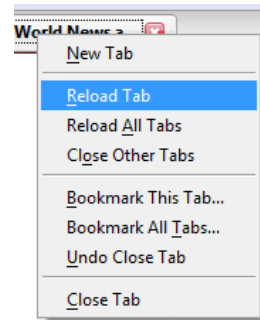


Menus have the ability to span off sub menus as can be seen in the figure to the right. This is always done with a triangle indicating the hierarchy of the menus. The subtle interactions are very important to make sure the user doesn't get frustrated with the menu. In this case we can identify many detailed features.

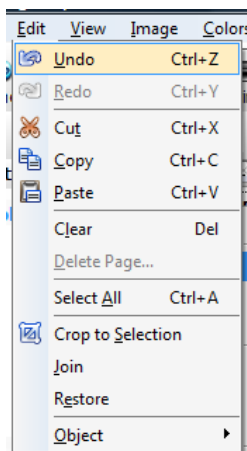


The first feature is that each word in the menu has an underlined letter. This allows for keyboard shortcuts in the menu. This is an extremely good idea in that it doesn't take up any more room in the UI, but enables a user with a keyboard preference to work quickly and effectively. Notice also next to the word "Reload" that "Ctrl+R" is listed right next to it. This enables another keyboard shortcut that would instantly perform that task. These keyboard features will endear you to power users, but not to intermediate or beginners. However, don't underestimate the advanced folks out there. They can really help your product if you let them.

In this menu, the user can use the mouse, the keyboard or several advanced combinations. The functionality is separated into five major sections. And then within the menu, there is a sub-organization using thin gray lines or separators. Organizing a menu is incredibly hard work. The craft is called Information Architecture and is discussed in length in the next chapter. But notice quickly, how the word "Tab" is used in the menu to the right. User's eyes will scan down the left side of the words. Make sure to put the verb or differentiating word there. It will help the user see clearly what choices are available.



Lastly in this example, notice the words in the menu are pretty long. A menu allows for much longer words than in a button. Use this power sparsely. It is always better to be concise. Language is a scalpel, not a broad sword. Find the words that best help the user understand what the menu option means. This is discussed, as well in chapter 4, Mental Models.



Some menus can be accentuated with icons. This doesn't mean you need to put icons next to every menu item. Use icons for the more important ones, or at least for the ones that you have a clear and unambiguous graphic. It is also good practice to put the icons in a strip on the left so that the words still line up and can be easily scanned. There is nothing worse than a difficult to read set of functions. Make sure the selected item is clearly highlighted as well.

On critical aspect of menu usability is related to sub-menus. The problem is that when a user travels from an item on the menu to a sub-menu, the sub-menu often disappears. This shortcoming has scared many product managers away from sub-menus permanently. This is related to the technology behind it and the tricky "neck" of the menu. The following use case explains the problem.

## CASE STUDY: IBM.COM AND MSNBC.COM

A tale of two menus: One on IBM.com and one on MSNBC.com. Both menus help the user find what they are looking for and both also have submenus. These sites have a lot of content, so sub-menus were unavoidable. However, one of the menus forgot to deal with the neck and thus has hampered usability.

Of course, this is difficult to demonstrate without motion, but I will give it a try. In the IBM menu, the user puts their mouse over the choice “By Industry”, which creates a sub-menu. Notice the lack of triangle to indicate the hierarchy. (Already, this is a flawed menu.) However, the user then sees the choice “Consumer products” and decides that this is indeed what they want to see. The straight line cannot help but cut straight through the menu choice for “By business need”. This makes the sub-menu containing “Consumer products”, (the actual object of desire for the user), disappear! God, (in this case the designer) decided that it would be very funny to dangle a menu choice in front of the user and then yank it away at the last second. The only way to actually get to the choice would be to follow a right angle approach, directly to the right and then straight down. This is actually pretty hard to do, even for advanced users.



A more nuanced implementation is required. Notice this version below from MSNBC.com. The sub menus are clearly marked with triangles on the right side of the menu. Additionally, there is a different visual treatment to further differentiate the hierarchy. Most importantly, as the user moves their mouse from the main menu to a sub-menu, the menu does not disappear. This is achieved with a timer that is hooked into the menu that leaves it open for half of one second (500ms). This allows the menu to be tolerant of messy mouse movements. The user is happier and does not feel that the menu is unusable.



Details like this often take programming effort, but they are well worth the price. Usability is not something comes for free. You must invest in the technology and the details. In the end your customers will achieve higher task completion rates and be more likely to get to their destination.

## Keyboard shortcuts

Keyboard shortcuts are the least used of any UI element in web applications. This is a shame because they are often the quickest way to do something. Functions like control-s for saving are often left out of web applications even though they make perfect sense. In older browsers, this may have been difficult, but in today's world, there is no excuse for not encouraging keyboard shortcut behavior.

One interesting problem that has evolved is the absences of a traditional keyboard. Cell phones, game consoles, ATM machines, Car Stereos have all become embedded computer systems with very sophisticated functionality. Most of them, however, have no real keyboard. This creates a significant challenge to the interaction designer who wants to enable the user, but has limited hardware.

### CASE STUDY: NINTENDO WII

For the holidays this year, my family got a Nintendo Wii. These are marvelous innovative devices that have become very popular. After seeing their market share evaporate to Sony (Playstation) and Microsoft (xBox), Nintendo had to change the rules if they wanted to stay relevant. They reinvented the gaming console. Rather than awesome 3D graphics and speed, they focused in interaction design and how people play games. The result is something brand new and very exciting. However, not all is good in the land of Mario.

There is no built in keyboard, yet there are many places where typing is needed. On cell phones, even though there is not a keyboard, there are enough buttons to make typing possible, albeit slightly tortured. On a Wii, you just have a remote control. The infrared light on the control is seen by a sensor on top of the TV. This sensor is very sensitive and can see controller movements within millimeters. You point the controller at the screen and click in a text area. The "virtual" keyboard is now present.



As you move your mouse over the letters, the individual key gets bigger, so it's easier to click on it. This interface is hard to use, and I wouldn't advise it for anyone with shaky hands. My own hands are fine, but I had a terrible time typing without error. Nintendo can improve this interface tremendously with the addition of "shortcuts". When typing in a URL (the Nintendo has an internet browser built in), there should be a shortcut to add ".com" to the end of a word. This could be done with any combination of the buttons on the controller. The left-arrow key should control backspace. The B button should put the keyboard in shift mode. The space bar could be another button.

The point of keyboard shortcuts is to find the most common things a user does and make those things easier, if you know the shortcut. Beginners won't know about the shortcut, but they are going slow anyway. The intermediate user desperately wants to find productivity gains. Similar to the games themselves, the designer has to give the user a lot of combination buttons and gestures that achieve a certain goal. Use any game and you will notice that there are lots of motions and movements that the designers use to enhance the game. All I am saying is that they should apply that same technique to the keyboard. And you, the reader, should think about your own application and how you can add these gestures and shortcuts to help the user go faster. The goal is speed.

I feel the need;  
The need for speed.  
- Goose, Top Gun

### CASE STUDY: GRIDS

Another more tangible use case (for web applications) is the common grid. Grids have been around since the beginning of time. The table is a perfect tool for showing information in a structured and orderly fashion. In the last 15 years, the technology around the grid has improved dramatically. There are many features of a grid that are extremely useful. Keyboard shortcuts are one area that has often been overlooked in web application for grids.

One of the problems has been the technology to enable this sort of interaction. Recently, advances in JavaScript libraries and other technologies make this a much easier task. For example, notice the grid to the left from ExtJS.com.



A screenshot of a web application grid titled "Framed with Checkbox Selection and Horizontal Scrolling". The grid has four columns: "Company", "Price", "Change", and "% Change". It lists various companies with their respective prices and changes. A mouse cursor is hovering over the "AT&T Inc." row, which is highlighted in grey. There are checkboxes in the left margin of each row, and a vertical scrollbar is visible on the right side.

Company	Price	Change	% Change
3m Co	\$71.72	0.02	0.03
Alcoa Inc	\$29.01	0.42	1.47
Altria Group Inc	\$83.81	0.28	0.34
American Express Company	\$52.55	0.01	0.02
American International Group, Inc.	\$64.13	0.31	0.49
AT&T Inc	\$31.61	-0.48	-1.54
Boeing Co.	\$75.43	0.53	0.71
Caterpillar Inc.	\$67.27	0.92	1.39
Citigroup, Inc.	\$49.37	0.02	0.04
E.I. du Pont de Nemours and Company	\$40.48	0.51	1.28
Fxvnm Mobil Com	\$68.16	-0.43	-0.64

The selection model is clear: You can click on any row and it will select. Additionally, you can hold the shift key down to select a series of rows. Hold the control key down to select individual rows. Selected rows are lit and look slightly different than how the row looks when in hover state.

The user can also use the arrow keys on the keyboard to navigate the grid. By pressing the spacebar you can select many specific rows without ever using the mouse. This is incredibly important for user experience success. Different people will want to use the system in different ways. In this case, there is no easy way to tell the system which rows to select without the shift-control modifier keyboard shortcuts. The only alternative the user has is to individually click each row, no matter how many there might be.



A screenshot of the same web application grid as above, but with several rows selected. The selected rows are highlighted in blue. The selected rows are: "Altria Group Inc", "American Express Company", "American International Group, Inc.", "AT&T Inc", "Boeing Co.", and "Caterpillar Inc.". The mouse cursor is now hovering over the "Citigroup, Inc." row.

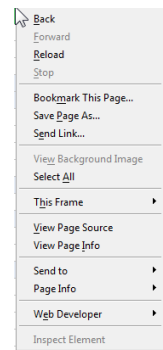
Company	Price	Change	% Change
3m Co	\$71.72	0.02	0.03
Alcoa Inc	\$29.01	0.42	1.47
Altria Group Inc	\$83.81	0.28	0.34
American Express Company	\$52.55	0.01	0.02
American International Group, Inc.	\$64.13	0.31	0.49
AT&T Inc	\$31.61	-0.48	-1.54
Boeing Co.	\$75.43	0.53	0.71
Caterpillar Inc.	\$67.27	0.92	1.39
Citigroup, Inc.	\$49.37	0.02	0.04
E.I. du Pont de Nemours and Company	\$40.48	0.51	1.28
Fxvnm Mobil Com	\$68.16	-0.43	-0.64

Grids have many features to consider beyond keyboard shortcuts. Sorting, pagination, column choosing, grouping, inline editing, updating and filtering are just some of the advanced features that a good grid employs. Depending on the application, grids can make or break your user experience. Make sure to give them ample time for UI features in your planning. Also, use well tested grids like Microsoft Excel and online design patterns to choose the best features for you.

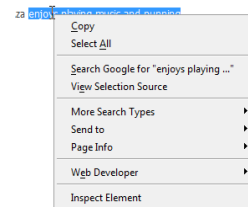
As you may have noticed, the theme of this chapter is not to go through each user interface element and detail a design pattern for them. The theme is to find all of the ways that the user can interact with your system and make sure that you employ them all. Clicking on buttons and menus with the mouse or using the keyboard are the most common, but there are other dynamic ways for the user to communicate to you. They both involve using the mouse in a more sophisticated way. The first is using right mouse button and the second is to hold the mouse button down, while moving the mouse somewhere else.

## Context Menus

“Context menus” are what displays when you right click on something in an application. Go to any webpage in your favorite browser and click the right mouse button in some open space. The context menu is like a normal menu, except for the word “context”. This means that the menu itself is supposed to have a specific relationship with the thing you are clicking on. If you right-click on an image, you should be able to download the image. If you right-click on some highlighted text, it gives you a very different menu. The contextual nature of the menu is what makes it so powerful. It can change the choices to be appropriate to the object being clicked.



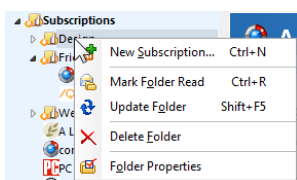
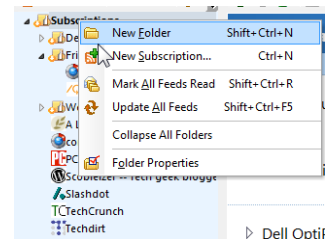
Imagine if there were real world context menus. If I right-clicked on a baby it might give me the options to feed, change diaper and burp. The context menu for a car would give a very different menu. Be careful to make sure that all of the context menus are discoverable and make sense. If the user doesn't understand the method to your madness, they will think everything (or nothing) is right-clickable. Things you should enable context menus on are:



- Tree nodes (objects and folders)
- Rows in a grid
- Grid headers
- Special objects

### CASE STUDY: FEEDDEMON

Tree nodes are the most obvious place to add context menus. Whenever your application has a tree, most likely there are things you can do with the nodes or leaves of that tree. A feed reader is a perfect example of this kind of interface. Often feeds are organized into folders and sub-folders. Notice in the figure to the right how the context-menu is specific to the node that was clicked on.



Now, when we click on the first folder, we get a very different menu. The choices are almost completely different. In fact, there is only one choice that remains the same. For each type of item in the tree, the program gives the user a different a contextually useful set of choices.

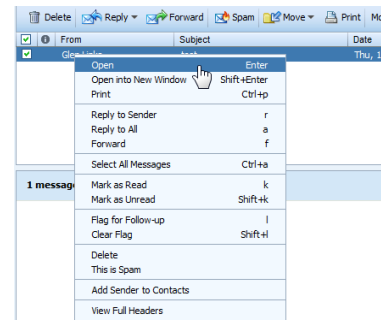
On the one hand, this is powerful, but it can also be confusing. A game that users often play is called “Hide and Seek”. The game works by putting an important feature in a context menu. Then make sure that the context menu only appears when you click a very specific kind of object. That is when the user looks for the function all throughout the application. It sounds like fun, but unless you are in elementary school, “Hide and Seek” has long since lost its amusement.

The balance is sometimes difficult to find. Should you include extra items in your menu just so that users can find them easier? This sometimes leads to menu choice overload, where literally dozens of choices are shoved in the users face. Then they can’t decide or scan the results. You have to keep your menus short enough to scan. The best advice is to keep focusing on the intermediate user. Advanced users are rare and unrepresentative and beginners wouldn’t try right-clicking anyway. The intermediates are the bread and butter. Make them happy and you will be golden.

As stated earlier, grids are also good fodder for right-clicking. By right-clicking on a row of a grid you are describing to the system what you want to happen to that row. Email is a perfect example of this effect.

### CASE STUDY: YAHOO MAIL

Yahoo has worked hard on its email client over the years. They have been a pretty solid innovator for online email clients. In some cases, they are pushing the envelope and trying new things. Notice that the menu has keyboard shortcuts for all kinds of functions, even though this is an online web application. In other cases, they forget the basics, like button-overload, demonstrated in the example to the right.

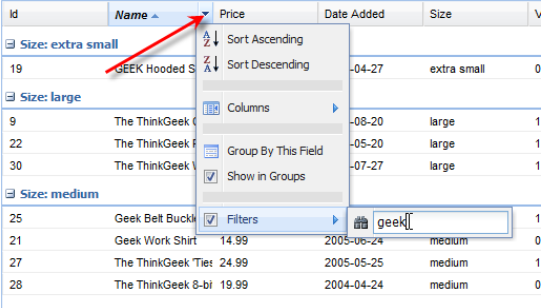


It is a very powerful feature indeed to enable rich email management. The problem is that the menu is too complicated and not differentiated. Had they added some icons and possibly removed some choices, it would make the usability of this menu increase dramatically. The main functions are not called out. Clear a flag seems to be equally important as delete or reply. A prioritized list with icons to help scan for them would be the best improvement. Although, we critique Yahoo in this example, Google’s email client has no right-clicking available at all. There are some nice things Gmail provides, but context menus are not one of them.

The combination of right-clicking and keyboard shortcuts in a grid is one of the most powerful UI dynamics around. Unfortunately, its applications are limited to specific kinds of interactions. Email seems to be the most widely used. However, if you find a good instance where the interaction would fit, you would have an advantage of many examples to draw from.

One area that Yahoo Mail left out was right-clicking on a header of a grid. Being able to click on the header of a grid is most useful when the contents of the grid need to be sorted or filtered.

The example to the right shows an online grid example where the user can choose many different options related to the contents. Microsoft Excel also has many advanced options for headers of a table. In this example, the menu only appears when you click the trigger. It would be better to have that menu appear with right-clicking or the trigger. That way, it would be more discoverable. More importantly, it maintains the mental model of the context-menu. Specifically, right-clicking on items in the UI would display contextually relevant options and features.



Id	Name	Price	Date Added	Size	V
Size: extra small					
19	THE GEEK HOODED S		-04-27	extra small	0
Size: large					
9	The ThinkGeek C		-08-20	large	1
22	The ThinkGeek F		-05-20	large	1
30	The ThinkGeek V		-07-27	large	1
Size: medium					
25	Geek Belt Buckl				1
21	Geek Work Shirt	14.99	2005-06-24	medium	0
27	The ThinkGeek Tie	24.99	2005-05-25	medium	1
28	The ThinkGeek 8-bi	19.99	2004-04-24	medium	0

## Drag and Drop

When to use

Interesting moments (yahoo library)