

What are some good usability guidelines an average developer should follow? [closed]

Asked 16 years, 3 months ago Modified 7 years, 4 months ago

Viewed 2k times



28



As it currently stands, this question is not a good fit for our Q&A format. We expect answers to be supported by facts, references, or expertise, but this question will likely solicit debate, arguments, polling, or extended discussion. If you feel that this question can be improved and possibly reopened, [visit the help center](#) for guidance.

Closed 12 years ago.

I'm not a usability specialist, and I really don't care to be one.

I just want a small set of rules of thumb that I can follow while coding my user interfaces so that my product has decent usability.

At first I thought that this question would be easy to answer "Use your common sense", but if it's so common among us developers we wouldn't, as a group, have a reputation for our horrible interfaces.

Any suggestions?

usability

ui-guidelines

Share

Improve this question

Follow

edited Jul 31, 2017 at 17:16



Peter Mortensen

31.6k ● 22 ● 109 ● 133

asked Sep 9, 2008 at 1:00



Allain Lalonde

93.2k ● 71 ● 189 ● 238

16 Answers

Sorted by:

Highest score (default)

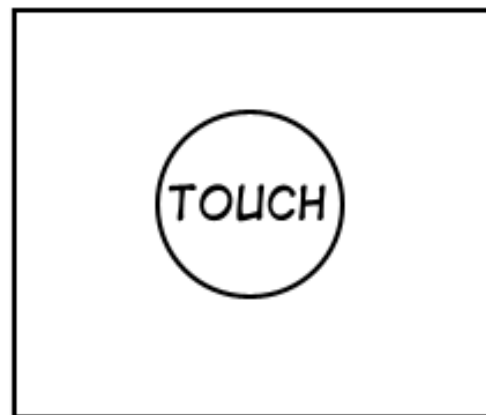




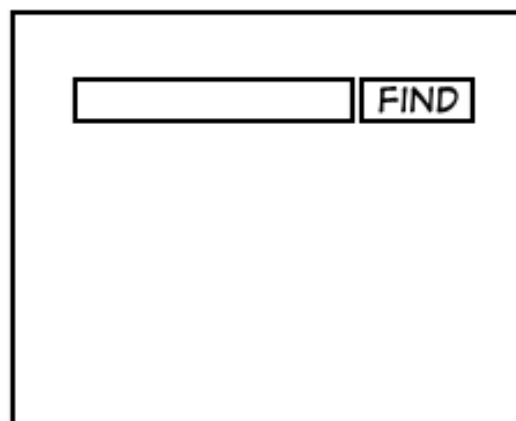
24



TYPICAL APPLE PRODUCT...



A GOOGLE PRODUCT...

A search bar with a text input field and a "FIND" button.

YOUR COMPANY'S APP...

FIRST NAME:	<input type="text"/>	TYPE CD:	<input type="text"/>	<div>4 - K AA2- DK9B KKA? CN3 AA-9</div>	
LAST NAME:	<input type="text"/>	TQP STAT:	<input type="checkbox"/>		
SSN:	<input type="text"/>	VER:	<input type="text"/>		
ID:	<input type="text"/>	CAT CD:	<input type="text"/>		
PHONE 1:	<input type="text"/>	CITY:	<input type="text"/>		
PHONE 2:	<input type="text"/>	STATE:	<input type="text"/>		
ADDR 1:	<input type="text"/>	ZIP:	<input type="text"/>		
ACCT #:	<input type="text"/>	ORD #:	<input type="text"/>		

OKAY APPLY SAVE UNDO HELP DELETE EDIT

[SELECT](#)[BROWSE](#)[ERRORS](#)

STUFFTHATHAPPENS.COM BY ERIC BURKE

Source: <http://stuffthathappens.com/blog/wp-content/uploads/2008/03/simplicity.png>

Share Improve this answer

edited Jul 5, 2012 at 1:09

Follow



ThiefMaster

318k ● 85 ● 603 ● 645

answered Sep 9, 2008 at 1:15



Marcio Aguiar

14.5k ● 6 ● 40 ● 42

1 This doesn't actually answer the question. – [Allain Lalonde](#)
Sep 17, 2008 at 19:12

Yeah, I'm sorry. But you can get some lessons from it: try to make it simple - or as others say, make it Zen. Like Steve Krug said in his book: Don't make me think! – [Marcio Aguiar](#)
Sep 18, 2008 at 7:29

Now I feel guilty that it's on the top. haha :) – [Marcio Aguiar](#)
Sep 18, 2008 at 8:08

1 I think it's a pretty good answer. Make it as simple and intuitive as you possibly can, particularly for parts of the interface that users will interact with dozens of times per day. Save the scary forms for the administrators and the "once in a blue moon" functions. – [CMPalmer](#) Sep 23, 2008 at 1:12

2 Actually, this answers the question very well. The point of the pictures is to say "Look at Apple and Google, who have very successfull products. Now look at the simplicity of their user interfaces and compare that with your own business application." Just because it's a line of business application doesn't mean it should have a simple, clean, easy-to-use interface. No, this answer doesn't give a set of rules to follow,

other than "keep it simple", but it is still a good answer.

– [Scott Dorman](#) Oct 10, 2009 at 2:57



Read [Don't Make Me Think by Steve Krug](#). It is a great starting point, and an easy short read.

12



EDIT: This is mainly for web usability though, but it would still be a good read even if you are doing rich clients.



Share Improve this answer

answered Sep 9, 2008 at 1:03



Follow



[Mike Stone](#)

44.6k ● 30 ● 114 ● 140



Just two things, really:

7



1. "A user interface is well-designed when the program behaves exactly how the user thought it would" -

[quoted](#) from Joel Spolsky's [User Interface Design For Programmers](#)



2. Put your designs in front of a user. A real end-user is best, but for lightweight, rapid feedback, you can't beat hallway usability testing i.e. grab a co-worker.



If you remember Joel's advice and make sure you get feedback on whatever you do **and act on it** i.e. iterate, you'll not go too far wrong. And I would echo the recommendation for Steve Krug's [Don't Make Me Think](#) - it's probably the best work-related book I've read, bar none, and is just as applicable to desktop software as websites.

Hope this helps.

Share Improve this answer
Follow

answered Sep 17, 2008 at 17:10



Mal Ross

4,701 ● 4 ● 36 ● 46



5



- Don't make things work in a different way than your users are expecting (i.e. breaking the "back" button when using Ajax in web forms)
- Follow the K.I.S.S principal

Really, any rules someone posts will be a variation on the theme: **Don't Make Your Users Think**

"Don't Make Me Think" has already been posted, see also [Design of Everyday Things](#) and [Designing with Web Standards](#) which are also great for light usability reading.

Share Improve this answer
Follow

edited Sep 9, 2008 at 1:09



Rob Allen

17.7k ● 6 ● 53 ● 70



4

Avoid [modes](#). It's frustrating to a user when input works sometimes but not others, or does different things at different times.



Share Improve this answer

edited Sep 9, 2008 at 2:38

Follow



answered Sep 9, 2008 at 2:21



Patrick McElhaney

59.1k ● 41 ● 137 ● 169



4



The single most important piece of advice I'd give someone is to work on the UI first. Pen and paper and all. That way, you won't subconsciously couple buttons to functions, input fields to variables, etc.

The best UI might be a pain to code, and if your backend code is mostly written, it will sabotage your thinking.

Other than that, I'd point to [Apple's Human Interface Guidelines](#). Of course, if your platform is not OS X, take the OS X sections with a lot of salt. What works in OS X might not work on Windows. You should embrace your platform's idioms.

OS X stuff aside, that document has some pretty good starting points on the fundamentals.

Share Improve this answer

edited Jul 31, 2017 at 17:18

Follow



Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Sep 9, 2008 at 1:07



Jordi Bunster

4,906 ● 3 ● 29 ● 22



4



Here are some simple rules:

- Fewer clicks are better.
- Frequently used features should be easier to find.
- Features for "advanced" users can be harder to find than the ones above.

Think about the number of mouse/keyboard clicks it takes a user to get to something.

PS - please don't tell the Microsoft Office 2008 people about this; the poor little guys would cry themselves to sleep tonight! :)

Share Improve this answer

Follow

edited Jul 31, 2017 at 17:19



Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Sep 17, 2008 at 19:55



a. brooks hollar

623 ● 3 ● 9



1



Think about the users that will use your app. Why are they using it and in which context?

- Will the majority be pro users that know the domain in which the application is used and use the app a lot? Then don't be afraid of adding a lot of data to the screens as long as it arranged logically for users

(normally that is not in alphabetical order :-). Think trade screens for stock borkers or airplane cockpits.


- Are users occassional users? Keep it simple. Avoid context switches (keep all/as much as possible of necessary data for a task on the screen at each time). Don't break expectations of how gui widgets normally work. Design for failures.
- Anything in between? Allow users to grow in the UI. Track usage so you can later determine where users seem to spend the most time so you can improve the most used areas of your app.
- Test your app on friends and colleagues (the corridor test) to see if they are able to use it efficiently.

That's a start.

Share Improve this answer

answered Sep 17, 2008 at 7:37

Follow

 **mbesso**
715 ● 4 ● 13



I suggest to read these [blog_posts](#) from the [Enso](#) creators.

1



Of course they repeat guides/ideas/advices from books such as

[The Design of Everyday Things](#) and [About Face](#), but nevertheless, the posts contain quite a few insights and (IMO) they are a good read.



Share Improve this answer

Follow

answered Oct 9, 2009 at 12:40



[Nick Dandoulakis](#)

43.1k ● 16 ● 105 ● 138



0

What information does your user need, put that on the screen and nothing else. If you cannot define what the user needs - get another user.



Share Improve this answer

Follow

answered Oct 10, 2009 at 2:14



[Nicholas Jordan](#)

656 ● 3 ● 6



0

Remember that your application will be one of many the user will have to deal with. Don't do things just to be different or kewl. Don't come up with unusual graphics, behaviors, terminology, or interactions. Use the standard OS controls, conventions, utilities, and behaviors.



Let your app interoperate with other apps; allow cutting and pasting of data, save your data in formats other apps can read, and allow importing data from other apps instead of using your UI.

If you are making a desktop app, do not try to take over the user's computer. Leave the user's Documents folder, task bar, and application preferences alone. Don't change anything already installed on the computer. Allow scripted or command-line interactions.

If you're making a web app, do not try to take over the browser. Do not try to subvert the standard menu bars, history, layout, or fonts. Allow the user to change the page using Javascript.

Share Improve this answer

answered Oct 10, 2009 at 2:40

Follow



Dour High Arch

21.7k ● 30 ● 77 ● 93



0



(1) Common actions should require as little effort as possible and should be obvious; on the other hand, actions that are rarely needed can be require a lot of steps and can be hidden behind menus and dialogs. To be able to do so, you should always describe what the user will want to do with the application by listing **use cases**.

(2) A UI should be selfdocumenting. The manual should be integrated in the application's dialogs and menu's, as users don't read separate manuals. For example, the keyboard shortcut should be shown in the menu item representing the action it is associated with.

Share Improve this answer

answered Mar 12, 2010 at 9:18

Follow



Dimitri C.

22.4k ● 21 ● 88 ● 103



Provide keyboard shortcuts for power users (even if it is as simple as "hit enter to search")

0

Don't put too much on screen at once.



If you pop up a messagebox, your users generally won't ever read it.



Share Improve this answer
Follow

answered Mar 12, 2010 at 9:25



Jamie

611 ● 1 ● 5 ● 20



0

In addition to the other recommendations here, I'd recommend [Designing Interfaces by Jenifer Tidwell](#) as a good way of becoming familiar with UI conventions. Also, [The inmates are running the asylum By Alan Cooper](#) is excellent for providing an insight into how to approach interaction design.



Share Improve this answer
Follow

answered Mar 12, 2010 at 10:32



Matt Lacey

65.6k ● 12 ● 93 ● 144



0

A good follow on to Don't Make Me Think is [Robert Hoekman's Designing the Obvious](#). It's more focused on web applications, as opposed to web sites like in Krug's.



Share Improve this answer
Follow

answered Mar 12, 2010 at 10:38



roryf

30.1k ● 17 ● 83 ● 105





0



- **Simple** is better than complex
- Complex is better than complicated (eliminate 'nested ifs')
- **Intuitive** (good elements needs no explanation)
- **Follow the convention** (for example, underlined means link, red means error, tab goes to next field, etc.)
- Use semantics to apply the logic (header reads first, paragraphs next)
- whitespace is important

Share Improve this answer

Follow

edited Jul 31, 2017 at 17:20



Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Mar 12, 2010 at 9:48



takeshin

50.6k ● 32 ● 123 ● 165
