UX

What is UX?

User Experience (UX) is a relatively new term, and as such it's definition has been a bit fluid. When it comes to software, a preferred definition is:

User Experience Design (UXD or UED or XD) is the process of enhancing user satisfaction by improving the usability, accessibility, and pleasure provided in the interaction between the user and the product.
-Wikipedia, User experience design

This definition is only part of the equation, namely the software. A new developer for example, needs a lot more than just the software itself. When visiting our website, they need to know at a glance what it is they're looking at so that they choose to investigate further, they need an easy setup, proper introductory documentation, etc. That's why this definition is helpful as well:

"User experience" encompasses all aspects of the end-user's interaction with the company, its services, and its products."
-Nielsen Norman Group Design Consultancy, The Definition of User Experience

This definition includes not only the Tiki interface, but also the impression we get from the logo and website, the clarity of our messaging, the quality/usability of our documentation, etc.

One of the better examples in this regard is the Laravel project, https://laravel.com/. The website is simple and uncluttered, the messaging is simple, the docs are well structured and clear.

Purpose of this Page

This page is meant to help in the discussion of what makes up Tiki's UX and how it can be improved. The goal is to reach some definition of what is 'proper' Tiki UX, so that developers have guidelines when creating and fixing features. Having a concrete definition would also make UX issues become bugs and therefore prevent discussions on every single UX commit, if it is a [FIX] or an [ENH] and where it therefore may be committed to. This would hopefully facilitate UX-fixes in released (especially LTS) versions, so that Tiki won't be judged by a crude interface for five years to come.

Principles of UX

Minimalism

Minimalism is achieved by reducing a design to only the most essential elements ... It's about taking things away until nothing else can be removed without interfering with the purpose of the design.
-Cameron Chapman, Principles Of Minimalist Web Design, With Examples

There is a bit of intuition around minimalism, and many of us have different tolerances to the quantity of elements on screen. A developer's mind for example, is tuned to face clutter and complexity and quickly derive meaning. End users however may not possess that same super power. It's therefore important to keep interfaces clean, and help direct the user complete their chosen task.

When faced with a interface that seems cluttered, and unintuitive several questions can be asked: Can it be reduced? Can it be restructured to feel more intuitive? Can we make the core/frequently used features more evident, while hiding the more complex and intimidating features behind an "advanced" area?

For more actionable information, visit Tyler Tate, Minimizing Complexity In User Interfaces.

Simple for a beginner, Powerful for the power user

Tying in to minimalism, core functions should should be immediately visible. In contrast, the less

frequently a feature is used, the more expert knowledge is required, the less it should be emphasized.

A good example of this is google's advanced search. For most people, searching with google is simply typing a search parameter into the search field, and clicking search, nothing else. For power users however, they may want to have finer control over their search query. If you look at Google's search interface, there is a gear icon in the top right corner. Contained within is a dropdown menu with an advanced search option. The non-power user may never even notice this gear, but it's still accessible for the power user who know what they're looking for.

Conclusion: Any functionality that is not critical to a beginner level user, used frequently, or core to the feature should be de-emphasized, but still be accessible to the power user.

Programming Abstractions vs User Abstractions

It's important to make a distinction between programming abstractions, and abstractions for the end user. For instance, if a user wishes to order a product online, the developer needs to create a form and submit the data so that it can be stored in the database. For the developer, the logical solution is to add a button to that form with the label "Submit". While this makes sense to a developer, from the end user's perspective, they are not submitting data via a form, they are shopping. In the user's mind this is likely analogous to physically shopping in a store. The checkout process may be analogous to being at the cash register. So for that user, rather than "submitting", they are finalizing an order. The button label for them might be "Complete Order", "Place Order", "Finalize Payment", or something more clever.

Where this becomes more critical is warning messages. If you were to fill out that form and there was an error in some of the information provided, writing "validation error" would confuse the end user. A better error message would be "Invalid Postal Code", or "Oops, there is an error with the Postal Code" depending on the tone and voice of the application.

Conclusion: Make sure to differentiate between user abstractions and code abstractions. Keep developer abstractions in the code, and create abstraction users can understand.

Make it Safe to Experiment

When a someone is a new user, or even a power user trying out a new feature, it's important make it safe for the them to experiment. If there are time consuming consequences to trying out a new feature and deciding to revert back to the original setup, this will take a mental toll on the user and discourage them from trying new features in the future.

Simple undo functionality should be implemented, and most often it should be located within the same context that it was enabled or setup.

Conclusion: Make it safe for the user to experiment. Make it simple to revert back to their original setup. Support undo functionality where possible.

User Interface

This is the biggest(?) and easiest part of the UX discussion to grasp, since it is quite tangible and practical. Still there is no formal definition of how the Tiki UI should look like, which creates inconsistencies that affect the users' emotions and attitudes.

Fixing such inconsistencies would already go a long way towards creating a better UX, like:

- Context menus should always appear either on click or on hover
- Buttons with similar functionality should be placed in similar places on different pages

Not about consistency, but preconceiving the average user's way of thinking and expectations:

- Essential/standard functionalities should be reachable directly and not hidden somewhere. They should meet "User's Expectation" and "Industries standards" as it is the natural way of communication and interaction between the user and Tiki. When a user see a B symbol in a toolbar he expect the Bold option. When a user see the "f" symbol in a page he expect it is related to Facebook.
    - I'm not a pro on this and I don't know how to put this, but I am sure that there are for example classification systems for UI elements that can be used to get to a common understanding of how an interface works and should be designed.
    - There needs to be a balance between keeping all the gadgets and icons in plain view, on the one hand, and avoiding too much clutter and complexity on the page, on the other. This is the issue regarding dropdown menus, for example. The views are probably subjective on this, as it involves usability and also visual aesthetics.

Some related pages of previous attempts:
TemplateStylingProposals
Templates Best Practices

Other things than UI

Probably general look and feel goes in here, even though it is the way the UI looks. But having elements on pages properly aligned instead of just having them hang around where they end up after templating gives me a much better feeling and doesn't create this nagging thought in the back of my head: "Did anyone bother with this?!".

- I think one reason for things not appearing to be aligned correctly is that Tiki has a lot of configuration flexibility, and alignment might have been tested in every combination of options. For example, a site logo in the top module zone can be inside a div with no padding or margins (*nobox=y*) or in a standard module box (in a div class="panel panel-default"), or in a div with a grid class (like div class="col-xs-4"). The white space around the icon is different in each case. But it's true that in the "out of the box" default setup, things should be nicely aligned.

Usability

Forms forms forms - is filling the forms easy enough for users or is it annoying and cumbersome?
Menus - is it too complicated to edit them? Is it easy enough to use them? Expand submenus on click or on hover?
Dropdowns - click or hover?
Buttons? Separated or grouped? When to use what?
Modals?
Action feedback?
What are the best practices? Should there always be a choice?
Consistency!

Speed

Loading speed and overall performance also contributes to UX.

Please expand

This page has been started by a total UX-n00b, that recognizes a good UX/UI when he sees it, but doesn't know how to put it into words. Please help him, help all the other developers and help Tiki! 😃