Planning for Tiki 13

Find a list of the members of "Team Bootstrap" * here *

all Teams: tiki.org - Memberlists

Steps for preparing for Bootstrap compatibility in Tiki 13

(Bootstrap 3.1 will be released under an MIT license so will be compatible with Tiki's license.)

Boostrap v.3.0.1 comes soon and it is dual licensed (Apache 2 and MIT) see https://github.com/twbs/bootstrap/issues/10050
This is good enough for us and so v3.0.1 can be included in trunk?
gezza: I guess it is not good enough, they say " all new contributions to Bootstrap will be dual-licensed as Apache 2 and MIT"
(http://blog.getbootstrap.com/2013/10/29/bootstrap-3-0-1-released/)
Exactly: "We've been looking to move to the MIT license for quite some time, and today's release takes us that much closer. Starting with v3.0.1, all new contributions to Bootstrap will be dual-licensed as Apache 2 and MIT. The intent is to move the entire project (including all prior contributions) to the MIT license in a future version (hopefully v3.1.0)." — luci

Decisions that need to be made:

Table of contents

CSS - file organization revamp

- Replace lite.css (now obsolete IMO) and design.css and feature-specific files in css/ directory with function-specific files mirroring the bootstrap .less partials.
  -> Torsten: NOT obsolete IMHO
  - Well, I meant in the sense that if the Bootstrap grid is used to lay out all the Tiki content, lite.css isn't needed. Not to say it doesn't have any, but what advantage does lite.css offer over using only the Bootstrap grid (just to be sure about this and so I'm being a Devil's advocate, to prevent unneeded duplication)? Also, for people with Bootstrap experience, maybe it's good to offer a path that's "pure" Bootstrap. But of course lite.css can still be available. That's why I mentioned maybe having options in L&F. Anyway, I want to make a schematic of CSS "stack" options.
  - -> Bootstrap does not know our module zones (especially #col2, #col3) at all and that is what lite.css takes care of: defined in layout_view.tpl and specified in lite.css. The thing is, that you need not to touch Bootstrap at all, to use the columns and that this is an easy to use and existing method to provide any content across pages and features, in column when wide screen and vertically stacked when narrow screen.
    - As a grid framework, Bootstrap doesn't need to know about Tiki's specific content such as the module zones or #col1,2, and 3. That's what the layout_view.tpl file is for: just assign bootstrap.css classes to those columns to make them part of the responsive grid. Bootstrap.css doesn't need to be touched.
  - -> I am not holding on the file lite.css itself, but the questions are,
    a) if we keep the method or find a new method to define column module zones
    b) if we keep, where we define it css-wise
    c) if we are already clearly at the point, that we do not touch bootstrap.css and it's various decendants (free themes) in itself, but build Tiki around bootstrap, so we literally can just plug in and out such themes in one single file and maybe very few simple adjustments
    => The workflow for good basic design must be very reliable and very easy!

- Globalize selectors to minimize the use of single-instance rules (design details such as white space and color).
  agree
- Question: What is the best-practices final form of the CSS files?

CSS - use Less CSS pre-compiler?

- It seems best to take advantage of the power of CSS pre-compiling if the workflow can be worked out, but (for comparison):
- e107 just uses compiled bootstrap.css
- Joomla apparently has install code to fetch bootstrap.css but doesn't include it.
- Some new CMSs support Less pre-compiling of stylesheets, but none of the major ones seem to.
- Another question: Should theme CSS files override core files, or should themes be made by editing .less files and *replacing* core files (for example in the same way that the customized (http://getbootstrap.com/customize/) and demo (http://bootswatch.com/) bootstrap themes are done)?
- Torsten => afaik the performance goes down, when using uncompiled LESS files, cause they have to be compiled by the server at every pageload (maybe relieved a bit by caching).
  LESS and other pre-compilers are meant to make it easier to develop new themes and to make theme more consistent.

Providing the option to use LESS compiled files directly might be an additional feature, but in my point of view, IF we want that, this should be some of the last steps (kind of a top up, maybe nice to have or show feature "Tiki can even this ...)

- No, the best-practice method is for the dev to compile the Less files into CSS before they and the output CSS files are committed to SVN. There's no performance hit at all; the files for distribution are just produced differently. Whether to switch to Less earlier or later kind of depends on how the current CSS is reorganized, IMO.
  - gezza: IMHO precompiling should be used only by devs for now
    - I agree, but would like to know if even devs are ready to use Less.
      - gezza: me not yet 😬

If Less pre-compiling is used...

Less and CSS - determine the best workflow.

Proposed:

- Commit .less files and compiled .css files
- Edit only .less files
- Resist the temptation to make quick edits in .css files. Reference: http://jeffcroft.com/blog/2012/feb/23/many-ways-to-use-css-preprocessors/
- Torsten: => Maybe a good point, but do you really want to force our user (equals customers in some respect) to learn LESS when they want or need to customise a few bits and pieces of their Tiki instance?
  - Users won't need to learn Less. Less would be used by the dev team to generate the packaged CSS files; theme authors could use Less if they wanted to, but wouldn't need to. Site admins/Tiki users could still edit CSS as usual, overriding the default Tiki files as they do now. (Example: Bootstrap.css itself is generated with Less, but users override the file rules if they like, with CSS editing.

RTL language support

- We should figure on supporting RTL from the start; discussion: https://github.com/twbs/bootstrap/issues/9913, theme-based solution: https://github.com/morteza/bootstrap-rtl.
- Torsten: => yesterday or so, I remind, that in an IRC chat it was said, that there is a solution?
  - Coming in Bootstrap 3.1, right?

Icon Fonts

- Should icon fonts such as Glyphicons (http://getbootstrap.com/components/) (or similar, such as Fontawesome (http://fortawesome.github.io/Font-Awesome/), which may work better - need to research/decide) be supported by default?
  Or if not by default, maybe it can be a L&F option, or else documentation provided as to what to modify (supporting icon fonts means linking to the files and adding, for example, '' or '' in the template).
- Torsten: => Since long I wish an option to simply switch icon sets.
  apart from that I strongly opt for Font-Awesome, as Font-Awesome is completely free (have to check licence!!! Afaik possible!)
  Glyphicons are not cross browser compatible (Chrome problems) and not completely free (free and pro

version)
    - gezza: imho there should be a UI and a configuration "layer" between the tpls and the actual icons, so in the tpl we should have only a generic sysname as the icon id like "save" and than the actual icon that is used as "save" is determined depending on the configuration. Icon set schemes could be predefined and shipped, for example Tiki could ship a scheme called "Tiki default icons", where "save"=/img/icons/disk.png. You could associate an icon scheme with a theme. Performance can be a question as this would requre additional processing.
      => that is actually an exact description of what I aswell think, we need. Thx gezza for the brief description and you have my +1
        - Flexibility like that sounds great, but I wonder how it can be done. Icon fonts require a class in the HTML, right? But maybe it can be done.

Template (.tpl) files

- Add Bootstrap classes to existing Tiki HTML where they make sense (under way with buttons, etc.).
- Reorganize as part of file tree organization
  Torsten: => Need a Coder/Designer Webinar!
- Confirm that templates/layouts is the place for page layout alternatives
  Torsten: => Confirmed. Here is the start point <-> layout_view.tpl
  templates/layouts/customlayoutname/layout_view.tpl

gezza: I think it should be in 2 files, I wrote some thoughts about it at the bottom of http://themes.tiki.org/Bootstrap+Themes+Transition
briefly: imho header and footer part of the current layout_view should remain the same in all themes. the body section (where you create the grid) should be moved out to a layout_body.tpl. Every theme gets an own layout_body.tpl.

    - gezza, by header and footer, do you mean the lines that load HTML head at the top and the javascript lines at the bottom? I'm not sure what you want to exclude.
        - gezza: yes, I meant those. Maybe there is a better solution, my concern is that layout development should focus only on the grid definition, but if we leave other functionalities that can change as Tiki evolves it can become hard to maintain all the grid layout files. See my example at the end of https://themes.tiki.org/BS+Themes+Transition, I think it should be that simple to introduce a new layout. (still some unnecessary items left there). A dev should just place the existing Tiki module zones into his desired grid and that is it. Or is it not that simple? my local tests work fine with this approach, but maybe I dont see the whole picture..

- Decide on bundled options - classic, responsive, responsive variants a la Drupal panels(https://drupal.org/sandbox/apmsooner/1805170) and a naming convention. Options should include:
    - Classic (current pre-bootstrap layout - will be modified/simplified as gadgets-to-modules progresses)
    - Default (Bootstrap-compatible implementation of classic)
        - As a superset of Bootstrap layout, this supports legacy Tiki behavior: module zones and left and right columns, column show-hide, etc.
          Torsten: => module zones and left and right columns have no problems with bootstrap.
          Only some change in lite.css is needed: limit effects of lite.css wits for wide viewports with @mediaqueries in lite.css and use relative grid width for the columns
          + This can also be done with Bootstrap's container elements (that is, making a "fixed-width" content area in a wide viewport - or am I misunderstanding this point?

@mediaqueries should have the same measurements as in styles/bootstrap/bootstrap.css (where ever this file will be placed finally).
        - Simple alternative (one big page-data area, for the most flexible layout- no default content)
          Why not just deactivate all module zones to achieve that?
          Sure, that may be better, but I'm thinking of cases where maybe you want module zones on some pages but not all, etc.
        - ?? Other layout views, or "panels"
- Globalize Tiki feature elements
  This may be a good opportunity to "globalize" or "unify" the elements of the various Tiki features; that is, to revamp the layout elements of wiki pages, blog and forum posts, cms articles (at least) so their titles, content body and so on are consistent, with consistent CSS selectors and so on. Any needed style variation can be provided via the body class for the feature. This would greatly simplify theming.
- Torsten: => Great idea! +1

Related tasks

- Revamp Smarty blocks and other code to support Bootstrap construction and classes .
    - Tabs
    - Menus
      How does the menu revamp fit into this transition?
      Torsten: => menu needs a complete overhaul - need a Coder/Designer webinar
    - Etc.
- Modules
    - Move remaining tiki-center gadgets, etc. to modules
        - Page bar, wiki action icons
        - Column show/hide icons
        - Etc.
    - It should be easy to add the bootstrap grid class to a module. Maybe instead of the actual class names, something more human-readable should be used, such as "1/6 width". Options also include "row", "container", etc. (what else should be used from bootstrap.css? Anything?
      Torsten: => maybe not needed! Just apply the grid-class to the object (or mostly containing DIV) and Grid is working
      I have to document that later, when I am back from travel
      I'm looking forward to your documentation because I don't know how a grid column (box, etc.) can be set to be, for example, 1/6 of the container width without being assigned the class - are we talking about the same thing?
    - About responsive behaviors of specific modules, what is the best behavior for a site logo? Probably to be replaced by something smaller, or to shrink, or disappear completely and only the site title be used.
      Torsten: => optional responsive images would be more than awesome, but that would be kind of a task ;-)
      Images can scale in CSS3 so are already responsive in that sense; Bootstrap also has rules for object display/no-display according to media query - though this doesn't stop the image download, which is a drag, but as Marc indicated, there is server-side control of resources depending on display device, maybe for Tiki 14.
- Tables
  If tables are too wide, they will cause their grid div to overlap the divs to the right. The Bootstrap solution

is to put the table in a div with class="table-responsive", to give the table a horizontal scroll bar. Thus the table will stay contained in the grid div and the design will remain responsive (http://getbootstrap.com/css/#tables-responsive). However, this behavior is specified by a media query rule in bootstrap.css for displays narrower than 767px. Tiki has tables that are too wide for their column even in PC displays, so we may need another class and rule for these cases.

jQuery, JavaScript

- How to resolve redundancy of existing scripts and new Bootstrap JavaScript?
- Is jQuery mobile still needed? Should it be an optional alternative to Bootstrap (which supports mobile devices by default)?
  + Torsten: => in my personal point of view we do not need a mobile perspective any more (I never got it really running with perspectives)
  + the existing solution seems to me more like a workaround
- Especially in light of Jonny's comments, I think we can conclude that jQuery Mobile will be replaced by Bootstrap. Now, what about jQuery-UI?
    - gezza: I think it is related to the question of how to manage the styling of externals (stuff in the vendor directory). To achieve unified visual experience, we need to do something with the various styling of vendor sources, but we dont want to touch source codes in the vendor directory, so we need a layer either to first purify and than to add (or just simply add) boostrap classes on the fly. This dilemma is known if you google it, you will find many threads on this. A project to address this is http://addyosmani.github.io/jquery-ui-bootstrap/. Not sure yet about the right solution.

Compare Responsive Solutions:

- jQuery Mobile vs Bootstrap (with flatui)
- jQuery Mobile with a responsive framework .. Foundation or Bootstrap
- percent of use Bootstrap vs. Foundation
- Phonegap or Bootstrap or jQuery Mobile

Touch & Gestures

- Should we pick a lib?
    - SuperFish now supports Touch

Workflow for theme developers, etc. in and after Tiki 13

- Regarding themes, we need to think about the entry points of designers and implementers. We need to support a range of skill levels and dev time availability regarding themes.
    - Just install a pre-made bootstrap theme and have it work.
        - In this case, we probably need to provide the grid and responsive class rules, as the pre-made bootstrap themes don't necessarily cover these. But pre-made themes are often stand-alone, not needing the default bootstrap.css. This needs to be researched further.
        - Maybe there should be some switches in L&F, to indicate whether the full bootstrap.css is needed (maybe the custom theme replaces it).
        - These methods are for working in CSS, not Less files.
- Make a new theme compatible with bootstrap.

    - One approach: working in CSS only.
    - If appropriate/necessary, we could provide instructions on how to best use the getbootstrap.com theme customizer (http://getbootstrap.com/customize/).
    - Another approach: working with Less.
- Make or upgrade a legacy Tiki theme.
    - Document the mapping of Bootstrap.css to Tiki legacy CSS selectors.

Mail to dev-list 22 Oct. 2013

has to be merged into the page structure, when I am back from travelling.
Cheers
Torsten

Hi Gary, Luci and all the Tiki-to-Bootstrap crew,

I think right now it is not yet a point to throw out superfish- why should we?
Independently of that question we need either a new menu-administration and module_menu or (what I'd prefer) some extension of the existing menu/module_menu for the menu options, bootstrap offers.

The next four days I will be mostly away and have only few time spots to communicate.
So I write all in the mail and copy the mail at the end of
https://themes.tiki.org/Planning+for+Tiki+13 and comment some points
When I am back, I want to merge the mailcontent into the appropriate sections of the page - if somebody wants to do before, feel free. Sorry, I have to leave the house and travel in a few minutes.

There is a lot in bootstrap's HTML, which has not yet a representation in Tiki. I am sure, some additional or refactored Wiki-Plugins will do the job.

In this respect the (hopefully growing) Tiki-to-Bootstrap crew needs to put the heads together with those people who are (or will be) on the Module/WikiPlugin development - especially when the direction is to merge them.

lite.css

Please do not missunderstand me Gary. It is not important, if we keep the file lite.css itself or merge the rules somewhere else. I did throw out some of lite.css already and changed the rest.

The core is following: We still need column definitions - responsible it must be alright, in which file ever they will be written, Ok ... but in the end it must still be defined somewhere.

I do not see a point to delete #col2 and #col3 out of any layout scheme, as they are optionally activated/deactivated anyway.

Luci did make a very good job when he created the file (afaik it was him who invented lite.css !?) and I

mainly use his basic idea, whilst the measurements are changed to responsive BS3 grid.

CSS structure

The css-structure in my demo-installation is as following:

/css
(kept untouched)

/styles/bootstrap/boostrap-default-css-subdirectories
(for ex.: /grid, /carousel, /jambotron, etc)

/styles/bootstrap/bootstrap-default-css-files.css
(for ex.: /bootstrap.css, ...)

/styles/bootstrap/lite.css
(reduced to the core variant with mediaquery and grid-based relative width)

/styles/bootstrap/cssmenus.css
(variant with a third responsive type, file maybe just temporary in use -> cssmenu_horiz, cssmenu_vert + cssmenu_responsive)

/styles/bootstrap/bootswatchthemes.css
(each bootswatchtheme is a complete customised and minimised replacemet of the original bootstrap.css.
Every theme from bootswatch.com is shipped as one single minified file named "boostrap.css".
So I renamed each of them after download to themename.css and put here besides the original bootstrap.css
...
Most likely the same with themes from other sources - we will try that anyway - maybe some extra workflow is necessary for bootstrap themes from Wordpress or Drupal? ... we have to check that)

/styles/themename.css
(one for each downloaded bootstrap theme, contain the @include rules for the base-css-files like styles/bootstrap/lite.css styles/bootstrap/themename.css
This file needs only two lines and provides the place to customise the original downloaded bootstrap-variation:
@import url("tikitobs3.css");
@import url("bootstrap/themename.css");
/* +++ from here space to customise themename.css +++ */
)

/styles/tikitobs3.css
(a "translation" file and kind of layer between /styles/themename.css and styles/bootstrap/themename.css
It @includes lite.css and my suggested left hidden admin area for all bootstrap themes.
...
Similar approach than lite.css and design.css from the old layout.
might be merged and/or renamed and/or placed somewhere else.
...
Some bits of design.css went inside this file aswell and here the standard cssmenu is becoming responsive

(hence not flipping like the nice bootstrap menu).
...
Not knowing yet, this kind of file might be useful in different variations like wordpress-to-tikibs.css, drupal-to-tikibs.css, etc.?
...
@import url("bootstrap/lite.css");
@import url("leftflip/leftflip.css");
/* +++ from here a number of adjusted necessary oldstyle css rules +++ */
)

The actual theme-option system is still working for bootstrap aswell, but not yet in use and as usal the following would be the place to put theme-specific files like themespecific background images or themespecific preview images and theme options:

/styles/themename
/styles/themename/option
etc.

For comprehensive background images or themespecific preview images or the custom logo I use an additional folder
/styles/grafics

Since Tiki 12, the layout scheme is a preference in tiki-admin.php look and feel.

A custom layout_view.tpl is placed here besides classic:
/templates/layouts/classic
/templates/layouts/bootstrap
/templates/layouts/any-other-layout-basis
...
Each folder placed here will appear in the Look and Feel administration dialogue as optional Layout scheme.
...
In this file (layout_view.tpl) I added a bootstrap grid class to each one of the divs #col2 and #col3.
(only two times a copy/paste)
This pretty much together with the changed lite.css provides us with a responsive column design.

So you see, mainly not too much changed to make it working without losing existing classic functionality.

The css has still some work to be done, but in my point of view we have mainly the following fronts to fight:

Roadmap thoughts (random order)

- Include the bootstrap jQuery and make the advanced elements working
- 'Take' the HTML based features one by one and either merge the new functionalities in existing wiki-plugins or code new wiki-plugins and maybe extend modules (at least necessary for menu!)
- Identify and change css rules in all our templates that can be expressed with existing bootstrap rules
    - listing them to be able to maybe convert the most used old-style themes like fivealive and jqui

- Identify css rules that cannot be represented with bootstrap rules (feature specific stuff) and streamline them to the bootstrap way of consistency where it makes any sense
    - listing them to be able to maybe convert the most used old-style themes like fivealive and jqui

- Identify why certain bootstrap elements like buttons have strange unintended effects like shrinking size on hover and fix it

- Code/structure an easy to use standard procedure to include external jQuery features
    - for ex.: responsive slideshows or flipcards, ...
    - try to figure out how to provide existing data to those features
    - where to put after download, how to activate
    - maybe mainly a more distinct documentation and best practice needed?

I think, that is pretty much enough for today,
cheers
Torsten

Related

- http://confoo.ca/en/2014/session/the-new-css-layout
- http://sixrevisions.com/mobile/methods-mobile-websites/