Bootstrap Themes Transition

About
This page has some preliminary ideas about Bootstrap classes transition targeted for Tiki 13 and onwards to help Tiki themers and designers.

Bootstrap files
Location: vendor/bootstrap/

Now in trunk (to be Tiki 13), all necessary Bootstrap files are part of the Tiki package and loaded by default. However, the Bootstrap Less files aren't included (yet) so you will have to get these separately if you want to use them to make a Tiki theme.

Main elements

Navbar
Could be used as a standard background for buttons.

Buttons
Default Tiki button (query, navigation, cancel action)

```html
class="btn btn-default"
```

Insert/update buttons (results in a commit to the database)

```html
class="btn btn-primary"
```

Delete, revert buttons (data gets deleted if or reverted to initial settings when pressed)

```html
class="btn btn-warning"
```

gezza: What do you think, would it make sense to introduce a functionality focused "tiki class" layer as well for the buttons to make styling flexible and development more consistent? I am thinking something like:

- btn-insert
- btn-update
- btn-delete
- btn-revert
- btn-cancel
- btn-query
- btn-navigation
- btn-icon
Notes

lib/
smarty_tiki/
function.button.php

btn class added (later updated to Bootstrap 3 style);

```html
<span class="button" ...>
</span>
```

renders now in mobile mode as

```html
<span class="btn-default button" ...> or <span class="btn-warning button"...>
</span>
```

otherwise

```html
<a class="btn-default" data-role="button" ...> or <a class="btn-warning" data-role="button"...
```

- issue #1: not all buttons are picked up for styling, for example the comment button on tiki-page_bar.tpl (it has span class="button")

- issue #2: in Tiki input types such as submit look like buttons, but they are not picked up by this function (eg: Create tracker button on tiki-list_trackers)

Dropdown

Select one option:

```html
class="form-control"
```

Select multiple options:

```html
multiple class="form-control"
```

Page layouts (grids)

These changes will be made in templates/layouts/bs_grid (or some name indicating Bootstrap grid compatibility. At issue is where to add class="row" container divs.

In parallel with this, presumably the idea of moving everything in the center column to modules will happen. This will simplify the template and give a lot of flexibility to page layout as well.

Overall page layout

This includes the top module zone, topbar module zone, side columns, center content (less div#role_main, which should be addressed separately), page bottom module zone (bottom of center column), and footer (bottom module zone).
gezza: The classes on layout-view.tpl are going to be simplified or we keep them? What is the plan for shadowlayers, fullscreen, fixedwith? Clearfix classes?

IMO (), we can have several layouts and the designer or site admin can choose which to use, or can add another. For example:

- Responsive_fivealive: This one enables replication of the current Fivealive style (has extra divs needed for full-width header and footer background graphics, but fixed-width content). Within each - header, middle, and footer - of course the areas are responsive.
- Responsive_middle: Simpler than the Fivealive layout, but has support for current Tiki layout of distinct header, middle and footer.
- Responsive_simple: Essentially one large container that depends on wiki code, etc. for layout as well as content.
- Other responsive variants

These templates/layouts/responsive_fivealive.tpl etc. layouts can be added or removed as we experiment with the possibilities.

In my opinion, there's no particular reason to keep old divs and div classes like the shadow divs in layouts other than Classic. If one goal of being Bootstrap compatible is to attract devs and designers who are familiar with Bootstrap, we should provide templates that are clean and familiar as much as possible, not cluttered. Also, Bootstrap provides for the "full-width" and "fixed-width" properties, so I don't think we need explicit divs/classes distinct from Bootstrap's.

Hmmm, -1 if you intend to remove Shadow Layers feature from me. At least keep the main shadow layer please.

How do we do custom wrappers easily?

For example it is very useful with "Categories used in templates (TPL)" preference. I use it a lot when I need something category specific:

Main shadow start:

```php
{if isset($objectCategoryIds) and in_array(32, $objectCategoryIds)}<div class="cat_wrapper"></div>{/if}
```

Main shadow end:

```php
{if isset($objectCategoryIds) and in_array(32, $objectCategoryIds)}</div>{/if}
```

Then in CSS:

```
.cat_wrapper .wikitext { /* Keep Layout Shadows feature and Categories
```
BTW, some probably-useful Bootstrap-compatible HTML templates for various page layout:
http://startbootstrap.com/ (Apache license); i.e., how to use Bootstrap divs in wiki pages, etc. for a nice site.

To keep these layouts simple I guess the layout_view.tpl has to be clean of any functionality other than layout arrangement. If there is other functionality, logics inside, than it would become hard to maintain them as one would need to copy functionality changes all over the different layout_view.tpl files. So these need to be moved out to separate tpls?:

- share
- tell a friend
- error report
- full screen?
- social networks
- google analytics
- endbody code?
- js?

At TikiFest Montreal/Ottawa last spring, we discussed (and I think agreed) moving those items to modules, to clean up the template and so they could be positioned flexibly.

Any thoughts on changes affecting Tiki module zones? (Top, Topbar, Pagetop, Left, Right, Pagebottom, Bottom). If someone new comes, this might be confusing having also header, main, middle, footer etc.

I think the current names are pretty descriptive and should be easy for newcomers to understand (maybe with the exception of "Bottom", which could be changed to "Footer"). It might be confusing that "Pagetop" and "Pagebottom" refer to the "middle center" of the content, and not the entire web page, so they could be changed if there are better words.

Proposal #1 (not sure if possible to do, but seems simple to follow)

The current layout_view can be separated into 3 areas:

1) header: everything until the start of the body tag. Designers don't need to work with this and it remains unified in all layouts when code evolves.
2) body_layout: this is where the designers create the layout. It contains only the grid layout with the module zones. See starter sample below. This way basically all the designers have to do is to create their favoured grid layout and place the module zones where they want. The only thing newcomers have to learn is that Tiki has some zones for content and they have to place them all into the grid.
3) footer: everything before the closing of the body tag that is not related to grid layout (JS file calling, endbody code, things from footer.tpl etc). Designers don't have to work with this, it remains unified in all layouts when the code evolves.

1) and 3) remains in one file and this file is considered as a system file. 2) goes into a separate file and gets called from the system file depending on the layout selection in theme administration.

Example: the current layout_view.tpl contains header, start of body, than an inclusion of body_layout.tpl
CSS selectors (classes vs. ids)

Styling of external sources

External sources come with their own styling. How is this neutralized and turned into Tiki styling? Vendor (external) source codes contain lots of their own formatting. This causes problems when trying to achieve a unified look and feel. For example, jquery-ui is nice, but has its own styling. Using Bootstrap, how will a jquery-ui dropdown look the same as a dropdown anywhere else in Tiki? Or a button?

One way is to simply add the needed classes, stop and override the usage of external CSS. This might work for buttons, tables, hyperlinks, etc, but how about divs? This can be done as well, but it does add bulk to the total CSS used for a page, since the external script's CSS has to be overridden by new rules that are added to make the page elements produced or affected by the external script consistent with the rest of the Tiki page.