

JBST

Child theme Development

<http://www.jbst.eu/>

7 April 2014 by Bass Jobsen (bass@w3masters.nl / @bassjobsen)

Table of Contents

Decision tree:.....	2
Example:.....	5
Getting started.....	6
Header.....	7
Fonts.....	8
Body	8
Footer.....	10
Homepage.....	12
our client section.....	14
Other pages.....	14
Development environment.....	15
Sample data:.....	15
Taking your theme in production.....	15
Style.css.....	15
LESS code.....	16
Theme options plugin.....	16
Customizer.....	16
Less example.....	17
The basics of the navbar.....	17
Set the colors of your customized navbar.....	18
Use predefined settings.....	18
Use the customizer.....	19
Alternatively set the colors with Less.....	19
Add a background gradient to your navbar.....	20
Give the links some some breathing space.....	20
Add separators between the navbar links	20
No separators for the first and the last link.....	21
Shorten your separators.....	22
Add some Indent to the links.....	23
Add an custom font.....	23
Your final Less code.....	23
Add an image slider to your website.....	25
Create an action hook.....	25
Built-in Less Mixins and Variables.....	26
Less variables.....	26
Less mixins.....	26

Decision tree:

When start writing code, try this steps (for every step if yes use it otherwise go to the next):

1. **can i use the build customizer?** (for example set the logo above the navbar and center). Note i'm building some options ready to set as default, in the function of your child theme write:

```
add_action('after_setup_theme','set_defaultoptions');
function set_defaultoptions() {
define('logo_image_position','in-nav');
define('logo_image','');
define('logo_outside_nav_text_align','left');
}
```

see also: <http://wordpress.stackexchange.com/questions/129479/alternatives-to-handle-customizer-settings>

2. **can you do your change with css / less only?**

Use the built-in LESS editor. This editor also accepts CSS (see also the github README for further instruction of ask)

You can also set every bootstrap less var here, see:

<https://github.com/twbs/bootstrap/blob/master/less/variables.less>

For example to change the default link color of your child theme, add to the LESS editor (or edit in less/custom.less): @link-color: orange;

Note if you have to add the same css / less again and again for different projects, consider step 3 for this changes.

3. start editing functions.php and / or create a template. Don't copy / paste functions and code from JBST's source, but use hooks, action and filters.

You will find a complete list of action hooks, filters and predefined settings on:

<https://github.com/bassjobsen/Boilerplate-JBST-Child-Theme>

an example:

The home page of <http://jbst.eu/> has some kind of jumbotron (see:

<http://getbootstrap.com/components/#jumbotron>)

To create this make a page frontpage_page.php or home.php

(http://codex.wordpress.org/Template_Hierarchy)

The content of this page will be:

```
remove_action( 'jbst_header', 'jbst_top_content_wrapper', 60 );
remove_action( 'jbst_footer', 'jbst_bottom_content_wrapper', 20 );
remove_action( 'jbst_footer', 'jbst_right_sidebar', 9 );
do_action( 'jbst_header' );
<!-- custom html for your -->
do_action( 'jbst_footer' );
```

So the only HTML you have to write in on the position of <!-- custom html for your --> of course you use Bootstrap's CSS to write this. Pretty cool, isn't?

NOTE if you have to add complex or repeating code on the <!-- custom html for your --> position. For example an image slider or a jumbotron again consider also **step 4 and further**

In some situation you will need a hook / action / filter which not exists yet. In such cases send me a email of what your need and we try to fit it in. Feel also free to post issue / pull request direct on Github.

Example: library/core.php now defines the navbar menu structure () on line 438 you will find:

```
'menu_class' => 'nav navbar-nav',
```

which setting the CSS classes for the navbar. I can imagine you want to add a custom class.

To accomplish this we change the line in core.php to:

```
'menu_class' => apply_filters('navbar_menu_class','nav navbar-nav'),
```

NB never change any of JBST's code for a particular project!! Also try where possible to keep the parent theme under version controle (git).

In your child theme's functions.php you can now use:

```
add_filter('navbar_menu_class', function($classes) { return $classes.' custom-navbarclass';});
```

In this specific case the first thing to check, can i also **fix it with step 2** (css / less)

4. create a custom post type

I found it will be relative easy to create a custom post type and use this to add / display an administrate list of items, such as sliders, features, team members etc.

The basic of a "custom post type" had been documented here:

<http://wordpress.stackexchange.com/questions/639/creating-an-image-centric-custom-post-type>

I have add two of such "custom post type" to my example theme (see attachment). You will find them in the folders /jbst-teammembers/ /jbst-features/.

The results are visible on the home page after activation. As you will see both show default content. Default content disappear as soon as you add an item to team members or features.

Both jbst-teammembers and jbst-features use the same code and structure basically. Teammembers is a little more sophisticated while jbst-features shows the option to limit the number of posts (features) to only three.

In both cases you can set menu order by hand. I think installation of a plugin to sort these items,

both <http://wordpress.org/plugins/simple-custom-post-order/> and <http://wordpress.org/extend/plugins/post-types-order/> seems to work well, should be independent of this code.

NOTE custom post types can be extend with custom fields, see: <http://wordpress.stackexchange.com/questions/17117/how-to-add-custom-fields-to-a-custom-post-type/131885>

If you think this is useful and helps faster and more stable development. We should disuse code re-use and sharing for this type of extensions. Option are use it under file control (if will have to write it to a class with setting in that case i think) or a code generator.

5. look for a plugin:

- the github README (<https://github.com/bassjobsen/jbst>) contains a list of Recommended and Supported Plugins check if you can use of them. If in this case check always b + c + d + e etc.
- don't use plugins which introduce additional javascript or css, in 99% bootstrap can fix your problem too (maybe you have to write a plugin / extension, step 5)
- don't use paid plugins (and always check if plugin's license fits the project)
- don't use plugin which are not updated recent

6. write a plugin / extension your self

- spent not longer than one to two hours on it
- write reusable code
- use the same basis structure again and again, for now use <https://github.com/bassjobsen/twitter-bootstrap-slider/> as a Boilerplate, (i will publish more basic code soon)

A plugin / extension does:

1. create a shortcode
2. adds a link to the dashboard (optional)
3. has a settings panel / store data in database and or save as media files (optional)
4. has documentation!!
5. if possible and useful for others should be published
6. has interaction with JBST hooks / actions (optional)
7. can be client specific

If you take <http://jbst.eu/> as an example again the jumbotron should be an extension and the features part under it too (display expo, marketplace).

In the last case you should build a short code [features] (maybe some options, like number, ids, random etc.) in the settings panel you can add new items, each item contains in this

case; title, description, icon, button text. It seems clear most of this code can be reused (or maybe the complete extension) to build for example a list of company members.

7. ask Bass what to do :) (bass@w3masters.nl)

Example:

This example describes how to set up a child theme using the decision tree above. You will be a child theme for the website shown below:



About Us

Unify is an incredibly beautiful responsive Bootstrap Template for corporate and creative professionals. It works on all major web browsers, tablets and phone. Lorem sequat ipsum dolor lorem sit amet, consectetur adipiscing dolor elit. Unify is an incredibly beautiful responsive Bootstrap Template for it works on all major web.

Donec id elit non mi porta gravida
Corporate and Creative
Responsive Bootstrap Template
Elit non mi porta gravida
Award winning digital agency

Award winning digital agency. We bring a personal and effective approach to every project we work on, which is why.
CEO Jack Bour

Meet Our Team



Jack Bour Chief
Executive Officer

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, justo sit amet risus etiam porta sem...



Porta Gravida
VP of Operations

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, justo sit amet risus etiam porta sem...



Kate Metus
Project Manager

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, justo sit amet risus etiam porta sem...



Donec Elit
Director, R&D Talent

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, justo sit amet risus etiam porta sem...

Our Clients



All files you will have to create should be create in the child theme's directory!

This design is pretty basic. In this example I expect the psd to html5 already done. Start with building the basis layout for a random page. You should create a file named page.php for this.

NOTE JBST wraps content in a container (bootstrap's grid) by default. In the design the page title (about us in the example picture) has a 100% width background color. Without this requirement there was not a reason to create a page.php in the first place. For the footer you will find the same kind of difficulties.

ALWAYS download the latest version before you start a new project:

<https://github.com/bassjobsen/jamedo-bootstrap-start-theme/archive/master.zip> also the examples below requires the latest version

Getting started

Make sure JBST is installed already.

After uploading the demo code to your theme directory (wp-content/themes) you should activate the theme in your dashboard (it is called "JBST Estate").

NOTE after activation of your theme you should recompile the LESS code first. In your dashboard choose appearance > LESS Editor and press “recompile”, the text area can be left empty!

Header

The header in this example can be built using the default header of JBST. You will have to set the logo outside navbar, with a default navbar. The navbar menu should be float to the right with CSS.

Create functions.php and add the code shown below to add the header setting discussed above:

```
add_action('after_setup_theme','set_defaultoptions');
function set_defaultoptions() {

    /* navbar */
    if(!defined('navbar_background_color'))define('navbar_background_color','white');
    if(!defined('navbar_border_color'))define('navbar_border_color','white');
    if(!defined('navbar_link_color'))define('navbar_link_color','black');
    if(!defined('navbar_linkhover_color'))define('navbar_linkhover_color','#0E8E4F');
    if(!defined('navbar_activelink_color'))define('navbar_activelink_color','#0E8E4F');
    if(!
defined('navbar_activebackground_color'))define('navbar_activebackground_color','white');

    if(!defined('navbar_search'))define('navbar_search',0);
    if(!defined('navbar_account'))define('navbar_account',0);
    if(!defined('navbar_cart'))define('navbar_cart',0);

    /* logo */
    if(!defined('logo_image_position'))define('logo_image_position','outside-nav');
    if(!defined('logo_image'))define('logo_image',get_stylesheet_directory_uri().'/logo.png');
}
}
```

After this set other navbar's details by CSS in style.css:

```
/* give the navbar a border */

.navbar-default {
border-bottom: 5px solid #F5F0F0;
}
}
```

```

/* float the navbar menu to the right */

.navbar-nav {
float: right;
}

/* let the logo overlap the navbar */

.logo-outside-nav #logo-link-container img {
border-bottom: 8px solid #18C350;
position: absolute;
z-index:1000;
margin-left:-30px;
}
.logo-outside-nav.container {min-height:46px;}

```

Fonts

The navbar menu has a special font, to set a font:

- copy {font-name}.eot, {font-name}.svg, {font-name}.ttf and {font-name}.woff to wp-content/themes/{your-childtheme}/assets/fonts/ (use <http://fontquirrel.com/>)
- add the font by LESS. JBST has a built-in mixin for @facefont to use this add: .include-custom-font('{font-name}'); in this case .include-custom-font('good_times_rg-webfont'); to wp-content/themes/{your-childtheme}/less/custom.less
- Run the LESS compiler after your changes; Appearance > LESS Editor

Instead of writing your LESS code to less/custom.less you can also add it to the LESS compiler direct. Enter your code in the text box (Appearance > LESS Editor).

Include-custom-font('GoodTimes','good_times_rg-webfont'); adds a font with the name GoodTimes. The second parameter good_times_rg-webfont should always equal the name of the font files in assets/fonts/.

After this steps you can use in your CSS to set the font of the navbar's menu:

```

/* set navbar's menu font */

.navbar-default .navbar-nav > li > a {
font-family: GoodTimes;
font-size: large;
}

```

Body

As mentioned before the page title (about us in the example picture) has a 100% width background color. For this reason we can't use the default page structure, which includes content-page.php via content.php. So create page.php.

First delay the action which start the content and wraps all of it in a Bootstrap container div:

```
remove_action('jbst_header','jbst_top_content_wrapper',60);
add_action('jbst_before_content_page','jbst_top_content_wrapper',1);
```

No you can call the header: `get_header()`;

Start a loop: `while (have_posts()) : the_post(); ... endwhile;` inside this loop you can wrap the page title in some custom HTML:

```
<div class="pagetitle_area">
  <div class="container">
    <div class="row">
      <div class="col-md-12">
        <div class="pagetitle">
          <h1><?php the_title();?></h1>
        </div>
      </div>
    </div>
  </div>
</div>
```

Add some additional styles in `style.css` to style this page header:

```
.pagetitle_area {
  background-color: #EBF3EE;
  margin-bottom: 45px;
  margin-top: 20px;
  min-height: 90px;
}
.pagetitle_area .pagetitle {
  margin: 25px 0;
}
.pagetitle_area .pagetitle h1
{
  color: #020202;
  font-family: myriadpro-semibold-webfont;
  font-size: 28px;
}
```

After this header you can add the page content:

```
<?php do_action( 'jbst_before_content_page' ); ?>
<div class="entry-content">
<?php if ( has_post_thumbnail() ) { // check if the post has a Post Thumbnail assigned to
it.
```

```

the_post_thumbnail('large',array('class'=>'img-responsive img-page-class'));
}
?>
<?php the_content(); ?>
<?php wp_link_pages( array( 'before' => '<div class="page-links">' . __( 'Pages:',
'jamedo-bootstrap-start-theme' ), 'after' => '</div>' ) ); ?>
</div><!-- /.entry-content -->

```

Remember you hooked `jbst_top_content_wrapper` into `jbst_before_content_page`

After the loop the page can be end as used to:

```

do_action( 'jbst_after_content_page' );
get_footer();

```

Footer

The width of the footer can easily set to full page with with the option. Also the font color and default background color can be set this way. You can add this option by appending them to the `set_defaultoptions` function in `functions.php`:

```

if(!defined('footer_width'))define('footer_width','full-width');
if(!defined('footer_bg_color'))define('footer_bg_color','#6A6F75');
if(!defined('footer_text_color'))define('footer_text_color','#fff');
if(!defined('footer_link_color'))define('footer_link_color','#fff');

```

After doing this, the copyrights should be displayed with a different full width background color. Unfortunately the copyright don't have a full page width container but are wrapped in a Bootstrap container.

`/jamedo-bootstrap-start-theme/footer.php` shows you `jbst_footer` had been hooked by:

- * @hooked `jbst_right_sidebar` - 9
- * @hooked `jbst_bottom_content_wrapper` - 20
- * @hooked `jbst_footer_area` - 30
- * @hooked `jbst_body_close` - 40

`jbst_footer_area` displays the footer HTML. This function is not hooked now (suggestions for alternative hooks are welcome) `jbst_footer_area` calls three actions: `jbst_footer_start_content()`, `jbst_footer_widgets()`, `jbst_credits()` and `jbst_footer_end_content()`.

`jbst_footer_widgets` is hooked by `jbst_footer_show_widgets` and `jbst_credits()` by `jbst_custom_credits()`.

You should remove this last hook in `footer.php`:

```
remove_action('jbst_credits', 'jbst_custom_credits',10);
```

The above will remove the copyrights but left the HTML intact. Reset this HTML by adding to style.css:

```
.site-info {  
padding: 0;  
margin: 0;  
border:0;  
}
```

After doing this you can hook some custom HTML with the copyrights in jbst_footer_end_content.

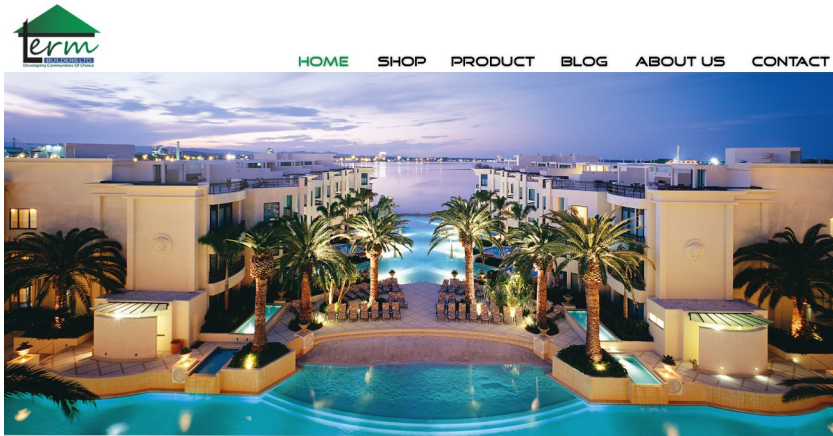
```
function copyright_full_width()  
{  
    ?>  
    <div class="footer_bottom_area" style="background-color:#3E4753;padding: 15px 0;  
text-align:center;border-top:2px solid white;">  
        <div class="container">  
            <div class="row">  
                <div class="<?php echo JBST_GRIDPREFIX; ?>12">  
                    <div class="footer_bottom">  
                        <?php echo jbst_custom_credits(); ?>  
                    </div>  
                </div>  
            </div>  
        </div>  
    </div>  
    <?php  
}
```

```
add_action('jbst_footer_end_content','copyright_full_width');
```

Now the footer will fits your design too. Although the footer seems to miss useful hooks to redesign it, the result is semantically correct. The additional content will be wrapper inside the <footer>.

Homepage

The homepage is this example has a different design:



The screenshot shows a website homepage for 'erm'. At the top left is the 'erm' logo with a green house icon. To the right is a navigation menu with links: HOME, SHOP, PRODUCT, BLOG, ABOUT US, CONTACT. Below the menu is a large image of a resort pool at night. Underneath the image is a text block: 'Unify is a clean and fully responsive incredible Template.' followed by a paragraph of placeholder text and a green 'Purchase Now' button. Below this are three feature boxes: 'Fully Responsive' with a green arrow icon, 'HTML5 + CSS3' with a gear icon, and 'Launch Ready' with a wine glass icon. Each box contains a paragraph of placeholder text. At the bottom, there are three section headers: 'Recent Works', 'Welcome to Term Builders', and 'Our Clients', each followed by a horizontal line and a large grey rectangular area.



Fully Responsive

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus etiam sem...



HTML5 + CSS3

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus etiam sem...



Launch Ready

Donec id elit non mi porta gravida at eget metus. Fusce dapibus, tellus ac cursus commodo, tortor mauris condimentum nibh, ut fermentum massa justo sit amet risus etiam sem...

Recent Works

Welcome to Term Builders

Our Clients

Please notice the steps described here are not included in the sample code yet.

To build this homepage. Create front-page.php. In this file you can set a default layout by:

```
global $jbst_layout;  
$jbst_layout = 'full-width';
```

The navbar will have not a border here. To fix this you can add some additional CSS to style.css, note the <body> gets a extra class 'home' automatic.

The header image can be add by a image slider with one image maybe or some custom HTML. Prefer to use here: <https://github.com/bassjobsen/twitter-bootstrap-slider>. See also: <http://getbootstrap.com/javascript/#carousel>

Also for the features, Recent works, welcome to team builders and the our client section, custom html or an extension / plugin (see **step 5 of the decision tree**) **TODO** (not included in the example code for now). It's important to develop and use re-usable code here.

In the basics front-page.php will look:

```
<?php  
/*  
Template Name: Home Estate  
Description: Add Jumbotron to your home page  
*/  
global $jbst_layout;  
$jbst_layout = 'full-width';  
get_header();  
?>  
[bootstrap_slider]  
<?php  
do_action( 'jbst_before_content_page' );  
get_header();  
?>  
[features]  
[recent_works]  
[team builders]  
<?php  
  
do_action( 'jbst_after_content_page' );
```

```
get_footer();
?>
```

our client section

In fact the “our client section” is on every page too, so you should add it to the footer.

`jbst_bottom_content_wrapper` calls `after_page_content` by an `do_action('after_page_content')`. You can use this action to add custom html after the container which holds the content and sidebars:

Add in your `footer.php`:

```
function our_clients_section()
{
    ?>
    <div class="client_area" style="background-color:#EBF3EE;padding: 10px 0;
margin-bottom:20px;">
        <div class="container">
            <div class="row">
                <div class="<?php echo JBST_GRIDPREFIX; ?>12">
                    <div id="clients">
                        <h3>Our Clients</h3>
                        [clients]
                    </div>
                </div>
            </div>
        </div>
    </div>
    <?php
}

add_action('after_page_content','our_clients_section');
```

Other pages

This site got a limited number of pages which all differ in some details, the contact page have a contact form, the about page shows team members.

For the theme you can decide to create templates for the limited number of different pages. This templates will be a copy of `page.php` with some modifications.

Also try to find a solutions where code reuse and good user interaction are in balance.

If you design for a specific client you could try to make page.php more dynamic. Cause you know the url's (or can influence them).

For example create page-about.php, where “about” is your page slug (see also http://codex.wordpress.org/Template_Hierarchy#Page_display):

```
<?
define('about',1);
include('page.php');
?>
```

In your page.php you can now use: `if(defined('about')){}`

Development environment

In your wp-config.php set WP_DEBUG to true: `define('WP_DEBUG', true);`

Install <http://wordpress.org/extend/plugins/debug-bar/>

Every time you have to recompile your LESS code, WordPress will ask you for your credentials. Consider to add these to wp-config.php too:

```
define( 'FS_METHOD', 'ftpext' );
define( 'FTP_USER', 'bass' );
define( 'FTP_PASS', 'password' );
```

see also: http://codex.wordpress.org/Editing_wp-config.php#WordPress_Upgrade_Constants

Sample data:

<https://wpcom-themes.svn.automattic.com/demo/theme-unit-test-data.xml>

WooCommerce includes sample data too (included with the plugin)

Taking your theme in production

It's highly recommended to test your theme before taking in production, the list of plugins below could help you to test:

- <http://wordpress.org/plugins/theme-check/>
- <http://wordpress.org/plugins/monster-widget/>

Also consider to do a theme unit test: http://codex.wordpress.org/Theme_Unit_Test

Style.css

Make sure your style.css is up to date, choose a proper name for your child theme. If appropriate Add a license file. Last but not least add a screenshot.png (880 x 660 pixels) of your child theme.

LESS code

If you start selling your child theme or install it for a client it is not necessarily to add the LESS editor and code. To remove it:

1. Make sure you are happy with the last compiled version of your CSS
2. Copy the content of `/wp-content/themes/jamedo-bootstrap-start-theme/library/assets/css/wpless2css.css` into `/wp-content/themes/{your_childtheme}/style.css` or import an copy of this file.
3. Disable LESS for your child theme with a predefined setting: `define('jbst_less',0);`
4. `remove less/*`

Theme options plugin

Also in JBST this plugin is optional. Most child themes should not need it too, so consider to disable this plugin (or references too it). When writing not possible in the current version of JBST by default, expected soon.

Customizer

JBST add many options to the Customizer. Most of them are not useful for child themes. Settings are added with `add_action('jbst_add_to_customizer','.....')`; consider to remove the ones which are not useful for your child theme. Or remove them all at once:

```
remove_action('customize_register', 'jbst_customizer');
```


Less example

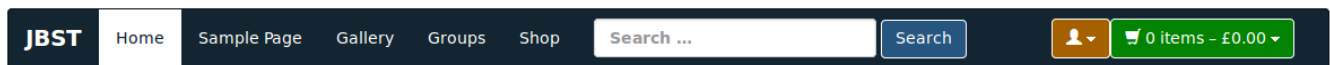
In this example you will build a navbar which looks like that shown below:



You can find an example of these navbar by visiting: <http://www.menstrualcups.eu/>

The basics of the navbar

After activation your child theme (Download the Boilerplate JBST Child Theme from: <https://github.com/bassjobsen/Boilerplate-JBST-Child-Theme>) your navbar will look like:



JBST's navbar has been build with Bootstrap's navbar component. Please visit <http://getbootstrap.com/components/#navbar> to read it's documentation. The HTML structure and classes are the same as Bootstrap's default, beside some additional code to meet web accessibility standards, the complete code can be found on: <https://github.com/bassjobsen/bootstrap-a11y-theme>.

To customize the navbar with Less your can use Bootstrap's navbar variables:

```
//== Navbar
```

```
//
```

```
###
```

```
// Basics of a navbar
```

```
@navbar-height: 50px;
```

```
@navbar-margin-bottom: @line-height-computed;
```

```
@navbar-border-radius: @border-radius-base;
```

```
@navbar-padding-horizontal: floor((@grid-gutter-width / 2));
```

```
@navbar-padding-vertical: ((@navbar-height - @line-height-computed) / 2);
```

```
@navbar-collapse-max-height: 340px;
```

```
@navbar-default-color: #777;
```

```
@navbar-default-bg: #f8f8f8;
```

```
@navbar-default-border: darken(@navbar-default-bg, 6.5%);
```

```
// Navbar links
```

```

@navbar-default-link-color:      #777;
@navbar-default-link-hover-color:  #333;
@navbar-default-link-hover-bg:    transparent;
@navbar-default-link-active-color: #555;
@navbar-default-link-active-bg:   darken(@navbar-default-bg, 6.5%);
@navbar-default-link-disabled-color: #ccc;
@navbar-default-link-disabled-bg: transparent;

// Navbar brand label
@navbar-default-brand-color:      @navbar-default-link-color;
@navbar-default-brand-hover-color: darken(@navbar-default-brand-color, 10%);
@navbar-default-brand-hover-bg:   transparent;

// Navbar toggle
@navbar-default-toggle-hover-bg:  #ddd;
@navbar-default-toggle-icon-bar-bg: #888;
@navbar-default-toggle-border-color: #ddd;

```

Bootstrap defines two classes for a navbar, a `.navbar` class and a `.navbar-default` or `.navbar-inverse` class. The first `.navbar` class sets the structure and the second class sets the styling of the navbar. By default there are two styles `.navbar-default` and `.navbar-inverse`. JBST doesn't define a new style but overwrites the `.navbar-default` styles.

Set the colors of your customized navbar

JBST offers you 3 methods to set the colors of the navbar. You can use the predefined setting, the customizer or define your colors in LESS directly. Customizer settings overwrite per-defined settings and LESS code overwrites all other settings.

Use predefined settings

To set the colors according the example you could use the predefined settings shown below:

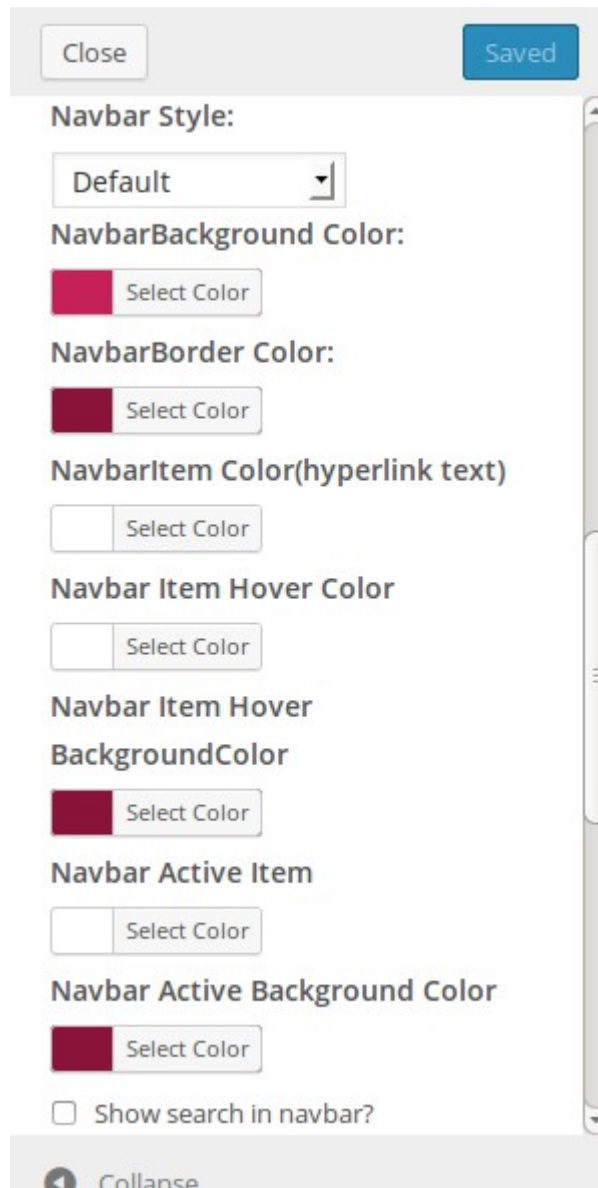
```

if(!defined('navbar_background_color'))define('navbar_background_color','#c62059');
if(!defined('navbar_link_color'))define('navbar_link_color','white');
if(!defined('navbar_linkhover_color'))define('navbar_linkhover_color','white');
if(!defined('navbar_activelink_color'))define('navbar_activelink_color','white');
if(!
defined('navbar_linkhoverbackground_color'))define('navbar_linkhoverbackground_color','#8a1339');
if(!
defined('navbar_activebackground_color'))define('navbar_activebackground_color','#8a1339');

```

Please notice you do not define the border color, this color is set automatically by: `@navbar-default-border: darken(@navbar-default-bg, 6.5%);`

Use the customizer



The same colors can be set with the customizer as shown in the image above.

Alternatively set the colors with Less

```
@navbar-default-bg: #c62059;  
@navbar-default-link-color: white;  
@navbar-default-link-hover-color: white;  
@navbar-default-link-active-color: white;  
@navbar-default-link-hover-bg: #8a1339;
```

```
@navbar-default-link-active-bg: #8a1339;
```

You can enter this Less code into the built-in Less Compiler (Appearance >> Less Compiler) or copy it into `wordpress/wp-content/themes/{your-childtheme}/less/customs.less`

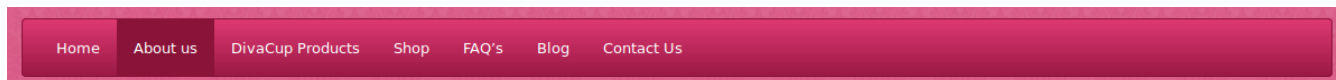
In both cases you have to run the compiler Appearance >> Less Compiler >> Recompile Less Code to apply your changes.

Add a background gradient to your navbar

After setting the navbar colors in the previous step you can add a gradient to the navbar with the Less code shown below:

```
#jbst-top-nav {
    #gradient > .vertical(lighten(@navbar-default-bg,10%),darken(@navbar-default-
    bg,10%));
}
```

You will only have to add this code to the compiler, after recompiling your code the navbar will look like:



To learn more about Less, please read the documentation which can be found on: <http://lesscss.org/>

In the above code the gradient is build with a mixins from Bootstrap: `#gradient > .vertical()`; sets a vertical gradient on the selector `#jbst-top-nav`. The start- and end color are calculated with the Less built-in function `lighten()` and `darken()`.

Give the links some some breathing space

To meet the requirements of example navbar add some padding to the links:

```
#jbst-top-nav {
    #gradient > .vertical(lighten(@navbar-default-bg,10%),darken(@navbar-default-
    bg,10%));
    a {
        padding: 20px 30px;
    }
}
```

As you see, Less allows you the nest your CSS selectors.

Add separators between the navbar links

In this example you create separators by adding a left and a right border to the links (a):

```
a {
```

```
padding: 20px 30px;
border-left: 1px solid lighten(@navbar-default-bg,10%);
border-right: 1px solid darken(@navbar-default-bg,10%);
}
```

No separators for the first and the last link

To exclude the first and last link you can use the `::first-child` and `::last-child` pseudo classes. Unfortunately the `a` selector is always the first and last child of the listitem (`li`). To solve this you can apply the removal of the border of the `li` selectors:

```
& li {
  &:first-child a { border-left: 0 solid;}
  &:last-child a { border-right: 0 solid;}
}
```

Your complete Less code now will look that shown below:

```
@navbar-default-bg: #c62059;
@navbar-default-link-color: white;
@navbar-default-link-hover-color: white;
@navbar-default-link-active-color: white;
@navbar-default-link-hover-bg: #8a1339;
@navbar-default-link-active-bg: #8a1339;
#jbst-top-nav {
  #gradient > .vertical(lighten(@navbar-default-bg,10%),darken(@navbar-default-
bg,10%));
  a {
    padding: 20px 30px;
    border-left: 1px solid lighten(@navbar-default-bg,10%);
border-right: 1px solid darken(@navbar-default-bg,10%);
  }
  & li {
    &:first-child a { border-left: 0 solid;}
    &:last-child a { border-right: 0 solid;}
  }
}
```

Shorten your separators

As you have seen the separators in the preceding have the same height as the navbar. The first requirements show shorter separators. To meet this requirement too, you will have to change your Less code into:

```
#jbst-top-nav {
  #gradient > .vertical(lighten(@navbar-default-bg,10%),darken(@navbar-default-
bg,10%));

  .nav > li {
    padding: 10px 0;
    &:hover,&.active,&:focus {
      #gradient > .vertical(lighten(@navbar-default-link-hover-bg,10%),darken(@navbar-
default-link-hover-bg,10%));
      a {
        background: none;
      }
    }
  }
  a {
    border-left: 1px solid lighten(@navbar-default-bg,10%);
    border-right: 1px solid darken(@navbar-default-bg,10%);
    padding: 10px 30px;
  }

  &:first-child a { border-left: 0 solid;}
  &:last-child a { border-right: 0 solid;}
}
}
```

The preceding code apply the `:hover` and `.active` states on the `li`-element:

```
&:hover,&.active,&:focus {
  #gradient > .vertical(lighten(@navbar-default-link-hover-bg,10%),darken(@navbar-
default-link-hover-bg,10%));
}
```

and remove these state from the a-elements:

```
a {  
  background: none;  
}
```

Finally the border to create the separators is apply on the a-elements too.

Add some Indent to the links

Adding indent is as simple as writing:

```
#jbst-top-nav > .container-fluid {  
  padding-left: 100px;  
}
```

Add an custom font

Your example navbar use the Bariol Regular font. You can download this font by visiting: <http://www.bariol.com/index.html>. After downloading the font you can use Font Squirrel's WebFont Generator to create a web font: <http://www.fontsquirrel.com/tools/webfont-generator>

Copy the files you download from Font Squirrel to wordpress/wp-content/themes/{your-childtheme}/less/customs.less. This directory now should contain: bariol_regular-webfont.svg, bariol_regular-webfont.woff, bariol_regular-webfont.eot, bariol_regular-webfont.ttf. Notice all files have the same name, you will have to use this name in the mixin below.

Now you can use JBST's built-in custom font mixins and create a custom font with Less:

```
.include-custom-font("BariolRegular";"bariol_regular-webfont");
```

Your final Less code

```
.include-custom-font("BariolRegular";"bariol_regular-webfont");  
@navbar-default-bg: #c62059;  
@navbar-default-link-color: white;  
@navbar-default-link-hover-color: white;  
@navbar-default-link-active-color:white;  
@navbar-default-link-hover-bg: #8a1339;  
@navbar-default-link-active-bg: #8a1339;  
#jbst-top-nav {  
  font-family: BariolRegular;  
  #gradient > .vertical(lighten(@navbar-default-bg,10%),darken(@navbar-default-  
bg,10%));  
  a {  
    padding: 20px 30px;
```

```

border-left: 1px solid lighten(@navbar-default-bg,10%);
border-right: 1px solid darken(@navbar-default-bg,10%);
}
& li {
  &:first-child a { border-left: 0 solid;}
  &:last-child a { border-right: 0 solid;}
}
.container-fluid {
  padding-left: 100px;
}
}

```

or:

```

.include-custom-font("BariolRegular";"bariol_regular-webfont");
#jbst-top-nav {
  #gradient > .vertical(lighten(@navbar-default-bg,10%),darken(@navbar-default-
bg,10%));
  font-family: BariolRegular;
  .nav > li {
padding: 10px 0;
&:hover,&.active,&:focus {
  #gradient > .vertical(lighten(@navbar-default-link-hover-bg,10%),darken(@navbar-
default-link-hover-bg,10%));
  a {
background: none;
}
}
a {
border-left: 1px solid lighten(@navbar-default-bg,10%);
border-right: 1px solid darken(@navbar-default-bg,10%);
padding: 10px 30px;
}

&:first-child a { border-left: 0 solid;}
&:last-child a { border-right: 0 solid;}

```



```
    }  
    .container-fluid {  
padding-left: 100px;  
    }  
}
```

Add an image slider to your website

Also this step starts with the creating of a child theme. Instruction can be found by visiting <https://github.com/bassjobsen/Boilerplate-JBST-Child-Theme>:

1. Download, install (do not active) JBST <https://github.com/bassjobsen/jamedo-bootstrap-start-theme/archive/master.zip>
2. Copy [the Boilerplate files](#) to `wordpress/wp-content/themes/Boilerplate-JBST-Child-Theme/`
3. rename the folder from step 2 and open 'style.css' and change the theme info (name, author, description, etc)
4. activate your child theme in your Dashboard Appearance > Themes

After the above installation of the child theme you will have to download and active the the image slider plugin. The plugin can be found on: <http://wordpress.org/plugins/twitter-bootstrap-slider/>

Visit https://codex.wordpress.org/Managing_Plugins to find out how to install a plugin.

Create an action hook

All JBST's action hooks and filters are described on: <https://github.com/bassjobsen/Boilerplate-JBST-Child-Theme>. Now open `functions.php` of your child theme and add the lines shown below to it:

```
add_action( 'jbst_header','bass_show_image_slider',45);  
function bass_show_image_slider()  
{  
    do_action('insert_bootstrapslider');  
}
```

Notice you can use WordPress' built-in theme editor (Appearance > Editor) to change `functions.php` as shown below:



The preceding code will add the slider above the navbar, not wrapped inside a .container and so full screen. Depending on your navbar settings there will be a padding above the image slider, you can remove this with Less:

```
body {padding-top:0px;}
```

To add the slider under the navbar you can use:

```
add_action('jbst_header','bass_show_image_slider',55);
```

Notice here the priority changed from 45 to 55.

Built-in Less Mixins and Variables

Less variables

```
@stylesheet_directory_uri //equals get_stylesheet_directory_uri().
@custom-font-dir //equals get_stylesheet_directory_uri()./assets/fonts/.
```

```
@font-family-base //bootstrap Less variable, sets the main font on the body
```

```
@headings-font-family //bootstrap Less variable
```

```
@navbar-default-font-family
```

```
@logo_font_family
```

```
@footer-bg-color
```

```
@footer-text-color
```

```
@footer-link-color
```

```
@footer-link-hover-color
```

Less mixins

```
.include-custom-font(@family: arial,@font-path, @path: @custom-font-dir, @weight: normal,
```

@style: normal);

Where @font-path should be set to the name of the font file without extension. So for instance, with .include-custom-font('Calibri','calibri-webfont'); you font directory (wordpress/wp-content/themes/{your-child-theme}/assets/fonts/) should contain: calibri-webfont.svg calibri-webfont.eot calibri-webfont.ttf and calibri-webfont.woff.