OXID eShop mobile theme and customization

Copyright

Conventions

The following conventions are used:

`proportional font with grey background`
: For user input, code and URLs

Grey italic
: For filenames, file paths and other italic highlighting

Bold
: For controls and navigation paths

Bold red
: For warnings and important hints

Contact

OXID eSales AG
Bertoldstrasse 48
79098 Freiburg
Germany
Fon: +49 (761) 36889 0
Fax: +49 (761) 36889 29
Represented by the Management Board: Roland Fesenmayr (CEO), Andrea Seeger
Supervisory Board: Harald Fuchs (Chairman)
Commercial Register Freiburg
No. HRB 701648

Table of contents

1 Purpose

The purpose of this document is to show how to customize mobile theme with ease. General suggestions and guidelines on how to change mobile theme can be found in this document. Here you can find information about:

- Styling mobile theme
- How OXID eShop mobile theme switch works
- Using OXID eShop mobile theme switch

2 Extending mobile theme

Mobile theme is written in LESS stylesheet language and later compiled to css file, which is then included in templates. These LESS files should be extended by creating custom imports file, importing necessary theme components (or ***oxid.less*** containing all components) and overwriting variables and classes or adding new ones.

2.1 Using LESS in your shop

LESS is dynamic stylesheet language, which extends CSS with dynamic behaviour. While it is considerably easier to read and work with than CSS, it still has to be compiled to CSS in order for shop to use defined styles. There are few ways to use LESS files in your shop:

1. First one and recommended by us is using lessphp (`http://leafo.net/lessphp/`) or other LESS generator to generate css file, place it in ***/out/{theme}/src/css/oxid.css*** and use it instead of ***oxid.css*** file.
2. Second option is to use OXID module to generate css file on the fly, which will be released soon.
3. As a last option use ***less.js*** provided by `http://lesscss.org/`.

2.2 Naming conventions

File and folder names should be lowercase and separated by underscore:

```
skin_components_folder/skin_component.less
```

For class and mixin names use lowercase words, hyphen separated:

```
.my-class {
    .my-mixin();
}
```

2.3 Extending LESS file structure

Mobile theme's LESS files are placed in ***/out/mobile/src/less*** folder. To extend them – create prefixed imports less file ***{skin_prefix}component.less*** and add all imports to it. Files should be structured like this:

```
components
  component.less
{prefix}_components
```

```
  {prefix}_component.less
domain_components
  domain_component.less
{prefix}_domain_components
  {prefix}_domain_component.less
oxid.less
{prefix}.less
variables.less
{prefix}_variables.less
```

2.4 Creating custom imports file

There are two ways to create custom imports file:

1. Import only ***oxid.less*** file and then import custom files like that:

   ```
   // import base file
   @import "oxid.less"
   // skin specific variables and mixins
   @import "{prefix}_variables.less"
   @import "{prefix}_mixins.less"
   // skin specific
   @import "{prefix}_components/{prefix}_component.less"
   @import "{prefix}_domain_components/{prefix}_domain_component.less"
   ```

   This way theme variables and components can be easily extended or added. This is how most of the time theme should be extended.

2. However if some components have to be rewritten from scratch or will not be used, ***oxid.less*** can be changed with skin specific imports. Then ***{prefix}name.less*** file should include all necessary bootstrap, components, mixins and custom files like this:

   ```
   // Import bootstrap and bootstrap fixes
   @import "bootstrap/bootstrap.less";
   @import "glyphicons.less";
   @import "navs.less";
   // Import theme's default variables and mixins
   @import "variables.less";
   @import "mixins.less";
   // Skip components which has to be overwriten completely
   @import "components/main_nav.less";
   @import "domain_components/scaffolding.less";
   // Import skin variables and mixins
   @import "{prefix}_variables.less"
   @import "{prefix}_mixins.less"
   // Import new skin components
   @import "{prefix}_components/{prefix}_component.less"
   @import "{prefix}_domain_components/{prefix}_domain_component.less"
   ```

2.5 Changing variables

All theme variables are stored in ***variables.less*** file and can be overwritten by importing skin variables file (***{prefix}_variables.less***), where values can be changed.

So if `@border-width: 1px;` is defined in our ***variables.less*** file, by adding the same variable with different value `@border-width: 5px;` to ***{prefix}_variables.less*** file – this value will be changed everywhere where `@border-width` is used. Even when redefining variable after using it in a style, value is set to the last defined value:

```
@border-width: 1px;
border-width: @border-width; // will be set to 5px
@border-width: 5px;
```

Since fonts, paddings, margins, colours and most of other properties are moved to variables, most of the time it should be possible to change the look of a theme by simply changing its variables values.

2.6 Changing components

Components used in theme are divided to domain components and reusable components. Domain components are footer, header, and checkout etc. - components which usually consist of other components and belong to domain level. Reusable components are buttons, alerts, form elements and other items, which can be reused in shop and does not belong to domain level. All these components can be extended or changed by creating prefixed components directory, adding component's less file named after existing component (as defined in LESS files structure part) and necessary parts changed like this:

```
.steps {
    max-width: @steps-max-width;
    .step-name {
        display: table-cell;
    }
}
```

Before changing any component, first check if it is possible to do so by changing its variables.

2.7 Handling different resolutions (media queries)

There are two types of media queries used in mobile theme. To make scaling easier, mixins with media queries were introduced and are used in most of the components to make them scale depending on the resolution. When this is not enough, components can have their own media queries.

2.7.1 Scaling with Mixins

Media queries mixins are used when element has to increase its width, height, padding or other properties depending on resolution. These mixins can be used like original css properties:

```
.class {                        // user defined class
    .padding(10px, 5px, 3px);
}
```

Currently there are mixins for font-size, line-height, width, height, paddings and margins. If these mixins are not enough, new mixins can be created at custom mixins file.

2.7.2 Responsive media queries

When component has to change more than just scaling, media queries can be added to the component itself. Extending components with additional media queries is almost like extending a component, with the difference `@media screen and (min-width:  @screen-large)` which wraps stylesheet like this:

```
.steps .step-name {
    @media screen and (min-width:  @screen-large) {
        display: table-cell;
   }
}
```

2.8 Additional LESS information

For more information about LESS and its guidelines, refer to LESS CSS website at:
`http://lesscss.org/`

3 Theme switch functionality

Theme switch is a new module which allows users to assign a different theme for mobile device. In order to use a new theme for mobile, user needs to activate theme switch module, and in its settings define mobile theme name. Keep note that the theme must exist in shop, and the name must be valid, in order for theme switcher to work properly.

3.1 Theme switch configuration options

Theme switch module comes with `onActivate(), onDeactivate()` methods, which adds customizable options for mobile theme. Once theme switcher is enabled, administrator can go to linked theme to see that theme's options. With these options it is very easy and simple to change image sizes or other mobile theme options. As for changing and customizing the theme, you can look at information provided here:

```
http://wiki.oxidforge.org/Tutorials#.5B4.5.5D_Understanding_theme_management_in_OXID_eShop_from_4.5.0
http://wiki.oxidforge.org/Tutorials#.5B4.5.5D_Customizing_Your_Online_Storefront_with_OXID_eShop_4.5.0_Themes
```

4 Adding complete module support for mobile

Due to changes to template files in mobile theme, some modules might not work with a mobile theme from the get-go. However it is possible to get modules working for theme without needing to patch or alter the module. You can add specific blocks for modules in a theme switcher, which allows for module developers to have different module looks on mobile devices. With this, you don't need to make additional patches for module itself, when it can be added to module switcher.

To have an example of how to make a module look differently on mobile device, look at PayPal implementation. For any shops using PayPal module, the new mobile theme switcher has changes for PayPal insuring that module works as well as it does in desktop theme.

4.1 Changing block templates to look different on mobile device

It is possible to use custom blocks for mobile devices in a module, and one of possible ways is to override blocks like it was done for PayPal module.
To add custom module block templates for mobile device, in order for it to look better on mobile, follow these steps:

1. Edit mobile theme switcher ***metadata.php***, located in ***/modules/oe/oethemeswitcher***, override blocks like this:

```
'blocks' => array(
        …
array('template' => 'page/checkout/payment.tpl',    'block'=>'mb_select_payment',
      'file'=>'views/mobile/blocks/oepaypalpaymentselector.tpl'),
    ),
```

   Use mb_ prefix for block name, so it's clear that this block is for mobile.
2. 'file' array entry shows where new template file will be. Create template file in that location, and write desired functionality within the file.
3. Change desired blocks (for example select_payment) name to have prefix mb_ (mb_select_payment) where you want blocks to be replaced in theme files.

4.2 Adding desired mobile device functionality in other ways

You can add different looks and feels for your module in another way. You can use `getActiveThemeId()` method to get active theme, and then add desired functionality for that.

5 Adding user agents

Mobile theme switch currently supports most of the popular mobile devices. New device support can be added by adding new user agent. To do so, go to ***/modules/oe/oethemeswitcher/core*** and edit file ***oethemeswitcheruseragent.php***.

```
protected $_sMobileDevicesTypes =
'iphone|ipod|android|webos|htc|fennec|iemobile|blackberry|symbianos|opera mobi';
```

Just add new device by inserting new value after the last like this

```
…iemobile|blackberry|symbianos|opera mobi|newuseragent';
```

and check if it works with new device.

6 Responsive images and responsive translations

While current theme does not offer responsive images and translations, there are future plans to implement that. In the meantime, if you think that you know how to implement such functionality in a way you find best, you can try and suggest that via committing a contribution to GitHub. Please take note, that if you want to offer a good solution, you need to make sure it follows common coding standards, or coding standards used by OXID.