Getting started with theme development

Setting up your local environment

Now that you intend to building a theme for PrestaShop, you are better off keeping all your development work on your machine. The main advantage is that it makes it possible for you to entirely bypass the process of uploading your files on your online server in order to test your changes. Another advantage is that a local test environment enables you to test code without the risk of breaking your production store. Having a local environment is the essential first step in the path of web development.

> ⚠ The following content assumes you're a developer and you want to create a theme or a module.

Installing PrestaShop

We advise you to install PrestaShop using Git and Composer.

Then, open a command line on your (empty) working directory, then:

```
1. git clone https://github.com/PrestaShop/PrestaShop.git
2. composer install
```

Using git you can choose your PrestaShop version: `git checkout 1.7.0.0`. Also we would warn you to test your final result with a zip release, just for safety (since vendor version might be slightly different).

> ⚠ If you haven't done it yet, we strongly recommend you to read our article Set Up Your Git For Contributing

Building your .gitignore file

A gitignore file is a must-have for any Git-versioned project, as it specifies intentionally untracked files that Git should ignore.

What to ignore

Generally, you shouldn't version the following types of files:

- Temporary files (such as cache files)
- Generated files (such as minified CSS or retrieved XML files)
- Files with credentials or personal information (such as `settings.inc.php`)
- OS and IDE-related files (such as .DS_Store or .idea/)
- assets/css/*
- assets/js/*
- node_modules/

We suggest that you build your own using http://gitignore.io.

> If you are building a full project for a client, you can read our article on building a gitignore for PrestaShop.

Create your theme from the Starter Theme

When you want to create a theme, the best way is to use the Starte rTheme as a base theme.

Create a new folder under `themes/` then download the Starter Theme and copy its files in your new folder.

Download the Starter Theme

Create your theme.yml file

First of all, you need to rename `config/theme.dist.yml` to `config/theme.yml` and edit it according to your theme's name.

```
name: YOUR_THEME_DIRECTORY_NAME
display_name: YOUR THEME NAME
version: 1.0.0
author:
  name: "PrestaShop Team"
  email: "pub@prestashop.com"
  url: "http://www.prestashop.com"

meta:
  compatibility:
      from: 1.7.0.0
      to: ~
```

Once it' done you'll be able to select your theme in your back office.

Starter Theme

PrestaShop 1.7 introduces a new way for designers to create their theme from scratch: the Starter Theme. The default theme for PS 1.7 is based on the Starter Theme.

For pretty much every CMS, the default theme is used as a framework to build custom theme: designers have to rework the default theme and reshape it into what they want to display. Sometimes that means having to spend a lot of time removing all the CSS rules and JavaScript code from the default theme, and rewriting everything. This means a LOT of work before even starting to actually create something original.

This means that a lot of themes are tied to the default theme's technical choices, because this way of working makes it hard to make your own choices. For instance, since the default theme uses Bootstrap, it's hard to use Foundation.

With the Starter Theme, the PrestaShop team decided to build a skeleton theme that will give you a kickstart for your custom theme, with all the minimum code (essential template files, markup and JavaScript code) and enough freedom to make your own choices. You can choose to use Bootstrap, Foundation or Blueprint. The Starter Theme is not opinionated: there is no decision made to use either one library or another.

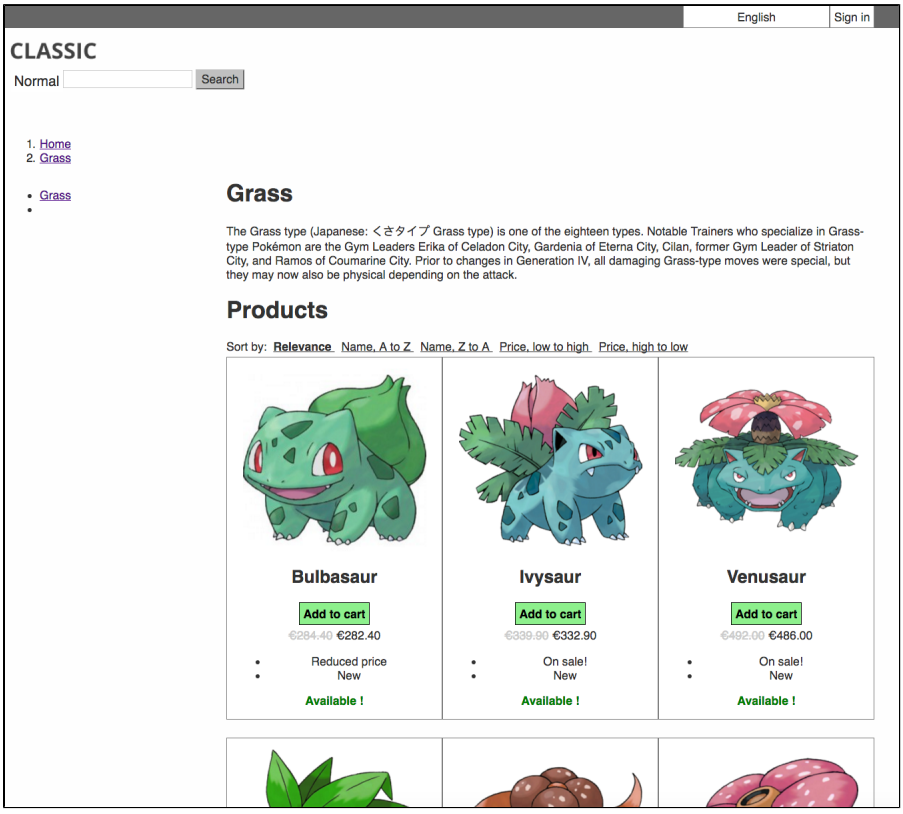
By using the Starter Theme as the foundation for your custom theme, everything is ready for you, you *ju st* have to create upon it.

Downloading the Starter Theme

The Starter Theme is available on GitHub: https://github.com/PrestaShop/StarterTheme

If you download the Starter Theme and select it as the theme for your store, you will see minimalist theme with an overly simplistic (ugly?) style. This is only for development purpose. You should NOT use the Starter Theme as is, and you should NOT use its default CSS rules nor include them in your theme: please delete all files inside *_dev/css*.

Here is a screenshot of the Starter Theme with dev style:



⚠ The jQuery v2 library is loaded by the core.js file.

Please note that if you want to sell your theme on the PrestaShop Addons marketplace, there are some specific requirements. For instance, Addons-distributed themes MUST use Bootstrap 4.

Modify. Don’t override.

When you want to create a new theme, copy and paste all files from the Starter Theme inside your empty theme directory. Then you start modifying it, and building your own theme.

Do not use it as a parent theme, you will only run into trouble and waste your time.

Once you removed all style in `_dev/css`, your theme should like this:

Read also: http://build.prestashop.com/tag/starter-theme/

Theme organization

Directory structure

A PrestaShop theme is a set of files which you can edit in order to change the look of your online shop.

Here are a few important tidbits:

- All themes have their files located in the /themes folder, at the root of PrestaShop’s folder.
- Each theme has its own sub-folder, in the main themes folder.
- Each theme is made of template files (.tpl), image files (.jpg, .png and such), one or more CSS files (.css), and usually JavaScript files (.js).
- Each theme has a preview.png image file in its folder, enabling the shop-owner to see what the theme looks like directly from the back office, and select the theme appropriately.

The best way to learn how to create a theme for PrestaShop 1.7 is to dive into the Starter Theme.

Here is its organization, which is explained further below.

```
.
 CONTRIBUTING.md
 README.md
 _dev
    css
       ...
    js
       ...
    package.json
    webpack.config.js
 assets
    css
       ...
    img
       ...
    js
        ...
 composer.json
 config
    theme.yml
 modules
    ...
 plugins
    ...
 preview.png
 templates
     _partials
        ...
     catalog
        _partials
           ...
        listing
           ...
        ...
     checkout
        _partials
           ...
        ...
     cms
        _partials
           ...
        ...
     contact.tpl
     customer
        _partials
           ...
        ...
     errors
        ...
        static
            ...
     index.tpl
     layouts
        layout-both-columns.tpl
        layout-content-only.tpl
        layout-error.tpl
        layout-full-width.tpl
        layout-left-side-column.tpl
        layout-right-side-column.tpl
     page.tpl
     wrapper.tpl
```

The folders are used this way:

Folder	Purpose

_dev	Contains the raw development files for your SCSS, JavaScript and image assets. They are to be compiled using Webpack, and turned into production assets.
assets	Contains the production assets, compiled by Webpack from the _dev files.
config	Contains configuration file. By default, it only has the theme.yml file.
module	Contains either theme-specific modules, or the theme's version of native modules' template files. For instance, the `themes/classic/modules/ps_categorytree/views/templates/front/ps_categorytree.tpl` file replaces the Category module's own `modules/ps_categorytree/views/templates/front/ps_categorytree.tpl`
plugins	Your custom smarty plugins
templates	Contains the template files themselves (.tpl), mostly in contextual sub-folders (catalog, checkout, cms, etc.). The _partials folder contains "partial templates", which means parts that can used by / included into several templates: header.tpl, breadcrumb.tpl, footer.tpl, etc. This prevents redundant code blocks, and makes themes easier to maintain.

Required templates and libraries

Required templates

When you install/enable a theme, PrestaShop checks if the theme is valid: it looks for the theme.yml file (and checks its content), its declared compatibility, and the existence of some files.

There is a list of files that need to exists, even if they're empty. Please see dedicated documentation to know what makes a theme valid.

It could be that you've built some sort of groundbreaking theme and it doesn't exactly work like the Starter Theme does. For instance, if you don't have a product page, then you don't need the product.tpl file. In that case, you just have to create an empty product.tpl file. Be nice to the next developer and add a comment indicating where the code related to products can be found ;)

Required libraries

jQuery v2.1 is loaded by the core (bundled in `core.js`) file, but no other libraries, since the idea is that the Starter Theme should not be opinionated.

Read more about assets management.

Theme.yml

The theme's theme.yml file defines all of the theme's configuration and meta information, such as its version number, layouts, compatibility range, hook configuration, etc.

Theme description

The theme's name MUST match its directory name. For instance, if the theme is named "My Awesome Theme" and its 'name' value is set to "my-awesome-theme", then the folder MUST be /my-awesome-theme .

Users will be able to choose the layout for each page from the theme's settings page. Layouts are automatically parsed from the theme's /templates/layouts folder, so this configuration key is optional, but it allows designers to provide some more user-friendly info than just a filename.

```
name: StarterTheme # The name must match the directory name
display_name: Starter Theme
version: 1.0.0
theme_key: 3c6e0b8a9c15224a8228b9a98ca1531d # Needed by PrestaShop Addons
author:
  name: "John Doe"
  email: "pub@prestashop.com"
  url: "http://www.prestashop.com"
meta:
  compatibility:
      from: 1.7.0.0
      to: ~
  available_layouts:
    layout-full-width:
      name: Full width layout
      description: Ideal for product pages to maximize picture size
    layout-left-side-column:
      name: One small left column
      description: Great for CMS pages to show advertisements on the side
```

Global settings

Configuration

You can have the theme change the configuration of PrestaShop when the theme is enabled.

```
global_settings:
  configuration:
    PS_QUICK_VIEW: false
    NEW_PRODUCTS_NBR: 4
    PS_PNG_QUALITY: 8
```

Modules

You can have the theme enable or disable modules when the theme is enabled.

```
global_settings:
  modules:
    to_enable:
      # All modules below are enabled when
      # the theme is enabled (and installed if needed).
      # They are disabled when the theme is disabled.
      - my-custom-module
      - yippeeslider
    to_disable:
      # All modules below are disabled when the theme is enabled.
      # They are re-enabled when the theme is disabled.
      - homeslider
      - blockwishlist
```

Since PrestaShop 1.7.0.0, theme activation can also reset modules:

```
global_settings:
  modules:
    to_enable:
       ...
    to_disable:
       ...
    to_reset:
      # All modules below are reset when the theme is enabled.
      - blockreassurance
      - blockwishlist
```

You can have the theme create hooks and attach modules to custom and existing hooks when the theme is enabled.

```
global_settings:
  hooks:
    custom_hooks:
      - name: displayFooterBefore
        title: displayFooterBefore
        description: Add a widget area above the footer
    modules_to_hook:
      displayHeaderTop:
        # displayHeaderTop will have exactly the following
        # modules hooked to it, in the specified order.
        # Each module in this list will be unhooked
        # from all other display hooks it is hooked to.
        - blocklanguages
        - blockcurrencies
        - blockuserinfo
      displayHeaderMiddle:
        # displayHeaderMiddle will have whatever is currently hooked to it
        # kept hooked to it, and blocksearch will be appended
        # to the list (or moved to the end if already hooked there).
        - ~
        - blocksearch
      displayHeaderBottom:
        # displayHeaderBottom will have blocktopmenu and blockcart
        # prepended to it.
        - blocktopmenu
        - blockcart
        - ~
      displayFooter:
        - blocknewsletter
      displayLeftColumn:
        # blockcategories is hooked on all pages on displayLeftColumn
        - blockcategories
        # blocktags is hooked on displayLeftColumn on all pages
        # except "category" and "index"
        - blocktags:
            except_pages:
              - category
              - index
```

Image settings

```
global_settings:
  image_types:
    cart_default:
      width: 80
      height: 80
      scope: [products]
    small_default:
      width: 125
      height: 125
      scope: [products, categories, manufacturers, suppliers]
    medium_default:
      width: 300
      height: 300
      scope: [products, categories, manufacturers, suppliers]
    large_default:
      width: 500
      height: 500
      scope: [products]
    home_default:
      width: 250
      height: 250
      scope: [products]
    category_default:
      width: 960
      height: 350
      scope: [categories]
    product_listing:
      width: 220
      height: 220
      scope: [products, categories, manufacturers, suppliers]
    large_banner:
      width: 960
      height: 400
      scope: [categories]
```

All the settings below can be changed through an interface in the theme's back office interface, and only depend on the theme/shop combination.

When the theme.yml file is parsed by PrestaShop, the 'theme_settings' configuration key is copied to a file named settings_n.yml, where 'n' is the id of the shop where the theme is installed (settings_123456. yml, for instance).

When the configuration is changed through the back office interface, only the settings_n.yml file is updated - the theme.yml file remains unchanged.

```
global_settings:
  theme_settings:
    default_layout: layout-full-width
    layouts:
      # Specific layout for some pages
      identity: layout-left-side-column
      order-confirmation: layout-left-side-column
```

Dependencies

When making a theme you may want to add features with custom modules. It's important that these modules are installed with your theme. These modules should be declared as dependencies so you're sure prestashop will export them when creating your theme zipball.

So far themes only have modules dependencies.

```
dependencies:
  modules:
    - xx_customslider
    - xx_customproductpage
```