
Magento2 Frontend Development

Created by
Ramakant Prabhutendolkar

Created on
22.05.2017

Courtesy
Keyur Patel

Topics to be discussed.

1) THEMES

Points to be discussed in this topic are *Theme overview, Theme Structure, Theme Inheritance, Apply and configure a storefront theme*

2) LAYOUT

Points to be discussed in this topic are *Layout instructions, Layout file types, Extend a layout, Override a layout, Common layout customization tasks*

3) TEMPLATES

Points to be discussed in this topic are *Templates customization walkthrough, Templates basic concepts, Illustration of customizing templates*

4) CASCADING STYLE SHEETS (CSS)

Points to be discussed in this topic are *Include CSS, CSS preprocessing, Compile LESS with Grunt, Magento UI library, Using jQuery UI styles, Customizing styles illustration, Using custom fonts, Add custom CSS preprocessor*

Frontend development prerequisites

You need a working Magento installation and the following browser versions installed on your device:

Storefront and Admin:

- Internet Explorer 11 or later, Microsoft Edge, latest-1
- Firefox latest, latest-1 (any operating system)
- Chrome latest, latest-1 (any operating system)
- Safari latest, latest-1 (Mac OS)
- Safari Mobile for iPad 2, iPad Mini, iPad with Retina Display (iOS 7 or later), for desktop storefront
- Safari Mobile for iPhone 4 or later; iOS 7 or later, for mobile storefront
- Chrome for mobile latest-1 (Android 4 or later) for mobile storefront

where *latest-1* means one major version earlier than the latest released version.

To use this guide, you must be familiar with:

- CSS and CSS 3, HTML and HTML 5
- XML
- JavaScript, Responsive Web Design (RWD)

Conventional notations

The following relative paths are used for modules and themes:-

<theme_dir>

Theme directory. Usually used when talking about custom themes, or any theme in general. For Magento out of the box frontend themes, the absolute path usually is one of the following:

- *app/design/frontend/Magento/<theme>*
- *vendor/magento/theme-frontend-<theme>*

<module_dir>

Module directory. When talking about a particular Magento module, also notation similar to the following is used: <Magento_Checkout_module_dir>

For Magento modules, usually one of the following:

app/code/Magento/<Module>

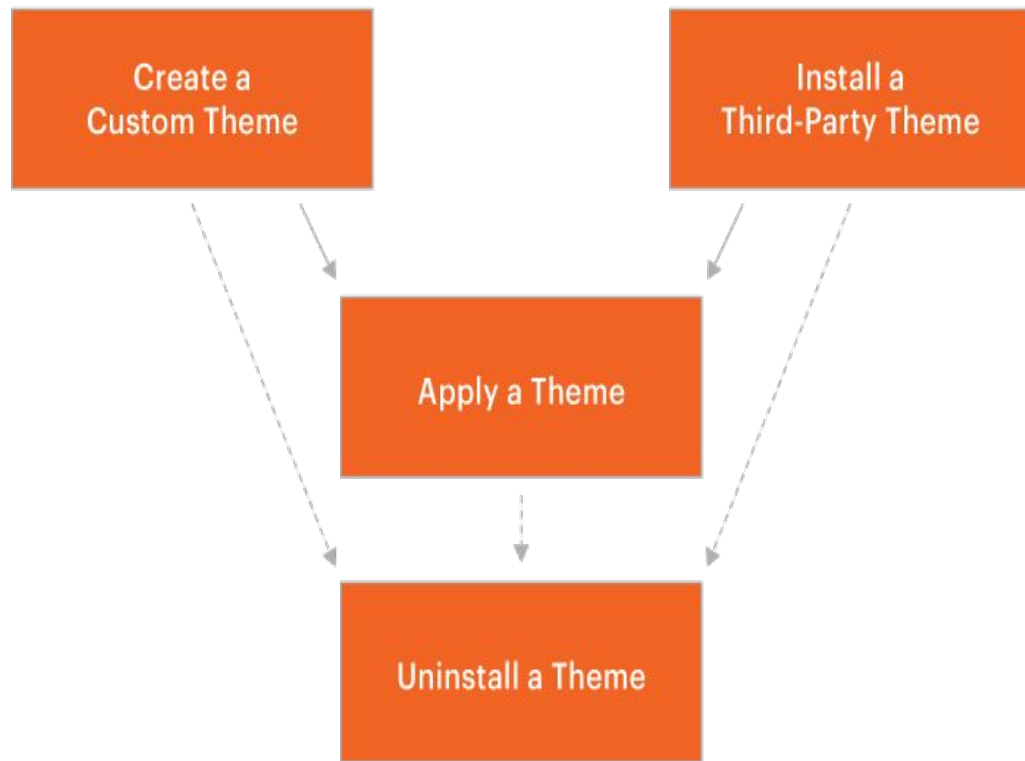
vendor/magento/module-<module>-<name>

Themes overview

Themes are designed to override or customize view layer resources, provided initially by modules or libraries.

Magento application provides two design themes: **Luma**, as a demonstration theme, and **Blank** as a basis for custom theme creation.

There are no restrictions on using the demonstration Luma theme for a live store, but if you want to customize the default design, you need to create a new theme.



Prerequisites- Theme creation

For the sake of compatibility, upgradability, and easy maintenance, do not modify the out of the box Magento themes. To customize the design of your Magento store, create a new custom theme.

Set your Magento application to the developer mode. The application mode influences the way static files are cached by Magento. The recommendations about theme development we provide in this chapter are developer/default-mode specific.

Create a new storefront theme

- Create a directory for the theme under `app/design/frontend/<your_vendor_name>/<your_theme_name>`.
- Add a declaration file `theme.xml` and optionally create etc directory and create a file named view.xml to the theme directory.
- Add a composer.json file.
- Add registration.php.
- Create directories for CSS, JavaScript, images, and fonts.
- Configure your theme in the Admin panel.

```
app/design/frontend/<Vendor>/  
|— <theme>/  
| |— etc/  
| | |— view.xml  
| |— web/  
| | |— images  
| | | |— logo.svg  
| |— registration.php  
| |— theme.xml  
| |— composer.json
```

Refer this link for more info [click](#)

Uninstall a storefront theme

Prerequisites

Set your Magento application to the developer or default mode.

Make sure that the theme is not applied on the storefront. To do this, in the Admin panel navigate to **Content > Design > Configuration** and make sure that your custom theme is not applied for any store view.

Make sure that the theme is not defined as a parent for any registered theme. To do this, in the Admin panel, navigate to **Content > Design > Themes**. Make sure that your theme is not mentioned in the Parent Theme column. If it is mentioned, you need to uninstall the child theme first.

Three ways to unInstall Theme

- 1) Uninstall a manually added theme
- 2) Uninstall a theme package if Magento was installed using Composer
- 3) Uninstall a theme extension

Magento theme structure

Magento theme location

Storefront themes are conventionally located under **`app/design/frontend/<Vendor>/`**. Though technically they can reside in other directories.

For example Magento built-in themes can be located under **`vendor/magento/theme-frontend-<theme_code>`** when a Magento instance is deployed from the Composer repository.

Each theme must be stored in a separate directory:

```
app/design/frontend/<Vendor>/  
├── <theme1>  
├── <theme2>/  
└── <theme3>...
```

[Refer this link for more info click](#)

```
<theme_dir>/  
├── <Vendor>_<Module>/  
│   ├── web/  
│   │   ├── css/  
│   │   │   └── source/  
│   ├── layout/  
│   │   └── override/  
│   └── templates/  
├── etc/  
├── i18n/  
├── media/  
├── web/  
│   ├── css/  
│   │   └── source/  
│   ├── fonts/  
│   ├── images/  
│   └── js/  
├── composer.json  
├── registration.php  
└── theme.xml
```

Apply a theme

To apply a theme:

- In Admin, go to CONTENT > Design > Configuration. A Design Configuration page opens. It contains a grid with the available configuration scopes.
 - In the configuration record corresponding to your store view, click Edit.
 - On the Default Theme tab, in the Applied Theme drop-down, select your newly created theme. Click Save.
 - If caching is enabled, clear the cache.
 -
 - To see your changes applied, reload the store front pages.
- In Admin, go to CONTENT > Design > Configuration
 - In the configuration record corresponding to your store view, click Edit.
 - On the Design Rule tab, click Add New User Agent Rule.
 - In the Search String box specify the user-agent using either normal strings or regular exceptions (PCRE). In the Theme Name drop-down list select the theme to be used for matching agent.
 - Click Save.
 - If caching is enabled, clear the cache.
 - To see your changes applied, reload the store front pages.

Refer this link for more info click

Apply a theme

Clear the cache

If caching is enabled in Magento Admin, you must clear the cache after you apply the theme, add a design exception, add a logo, and perform other tasks.

A system message notifies you that invalidated cache types must be refreshed.

1. Click **System > Cache Management**.
2. Clear the invalid cache types.



Configure images properties for a theme

The properties for the images displayed on the product pages are defined by the gallery widget options. The options of the widget can be configured in the theme `view.xml`

Configure image properties in view.xml

The conventional location of view.xml for a theme is:

`<theme_dir>/etc/view.xml`

In view.xml, image properties are configured in the scope of `<images module="Magento_Catalog">` element

```
<images module="Magento_Catalog">
...
</images/>
```

Image properties are configured for each image type defined by the `id` and `type` attributes of the `<image>` element:

```
<images module="Magento_Catalog">
    <image id="unique_image_id" type="image_type">
        ...
    </image>
</images/>
```

For more details, view the [Gallery widget](#) topic.

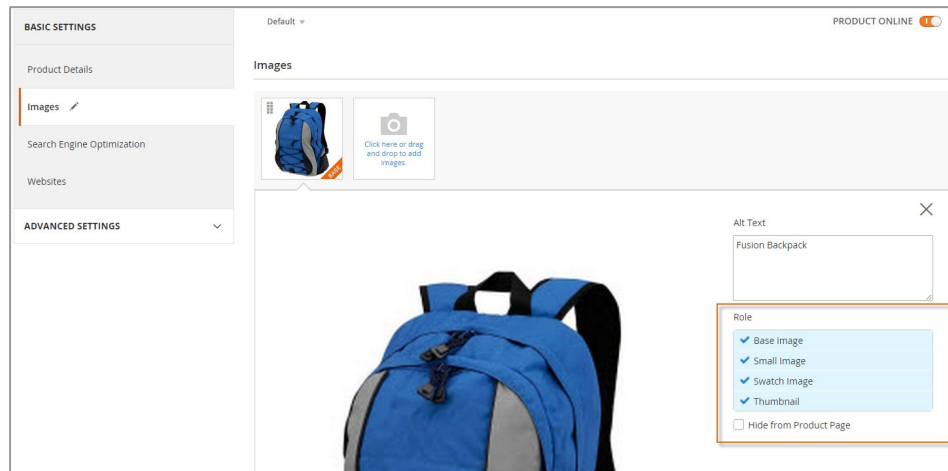
Configure images properties for a theme

Additional Info

Id: Can have any value, but in out-of-the- box Magento themes id's are meaningful and describe the location of an image.

type:

- **image** - corresponds to the Base Image role in the Magento Admin
- **small_image** - corresponds to the Small Image role in the Magento Admin
- **swatch_image** - corresponds to the Swatch Image role in the Magento Admin
- **swatch_thumb** - corresponds to the Swatch Image role in the Magento Admin.
- **thumbnail** - corresponds to the Thumbnail Image role in the Magento Admin



```
<images module="Magento_Catalog">
  <image id="unique_image_id" type="image">
    <width>100</width> <!-- Image width in px -->
    <height>100</height> <!-- Image height in px -->
  </image>
</images>
```

Refer this link for more info click

For more details, view the Gallery widget topic.

Theme inheritance

Theme inheritance enables you to easily extend themes and minimize the maintenance efforts. You can use an existing theme as a basis for customizations, or minor store design updates, like holidays decoration.

The level of theme inheritance is not limited. Theme inheritance is based on the fallback mechanism, which guarantees that if a view file is not found in the current theme, the system searches in the ancestor themes, module view files or library.

Set a parent theme

A parent theme is specified in the child theme theme.xml declaration file. Like theme phtml, static files and layout can be inherited.

E.g. `app/design/frontend/<vendor_name>/<theme_name>/theme.xml`

```
<theme xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:Config/etc/theme.xsd">
    <title>Orange</title>
    <parent>Magento/blank</parent>
    <media>
        <preview_image>media/preview.jpg</preview_image>
    </media>
</theme>
```

Locate templates, layouts, and styles

Locate templates

To locate the template that is responsible for a specific part of the storefront or Admin, you can use Magento built-in template hints.

To enable template hints:

- Click **Stores > Configuration > ADVANCED > Developer**.
- In the Scope dropdown in the upper-left corner select the view for which you want the template hints.
- In the Debug tab, set Template Path Hints for storefront to Yes. To enable path hints for Admin set Template Path Hints for Admin to Yes.
- To save the changes, click Save Config in the upper-right corner.

Override templates

The fallback scheme for templates is the following (module context is always known for them):

- Current theme templates:
`<theme_dir>/<Namespace>_<Module>/templates`
- Ancestors themes templates, recursively, until a theme with no ancestor is reached:
`<parent_theme_dir>/<Namespace>_<Module>/templates`
- Module templates:
`<module_dir>/view/frontend/templates`

In case of custom module magento will look for file into
`app/code/<module_dir>/view/frontend/templates` **after**
`<theme_dir>/<Namespace>_<Module>/templates`

Locate templates, layouts, and styles

Locate templates

After you have determined the module, you can search for the layout in the following locations:

- `<current_theme_dir>/<Namespace>_<Module>/layout/`
- `<parent_theme(s)_dir>/<Namespace>_<Module>/layout/`
- `<module_dir>/view/frontend/layout/`
- `<module_dir>/view/base/layout/`

Locate styles

Perform the search according to the following fallback scheme:

- Theme styles `<current_theme_dir>/web/css/`
- Module theme styles `<current_theme_dir>/<Namespace>_<Module>/web/css/`
- Parent theme styles `<parent_theme_dir>/web/css/`
- Parent theme Module styles `<parent_theme_dir>/<Namespace>_<Module>/web/css/`
- Module styles for the frontend area `<module_dir>/view/frontend/web/css/`
- Module styles for the base area `<module_dir>/view/base/web/css/`

Layout instructions

Use layout instructions to:

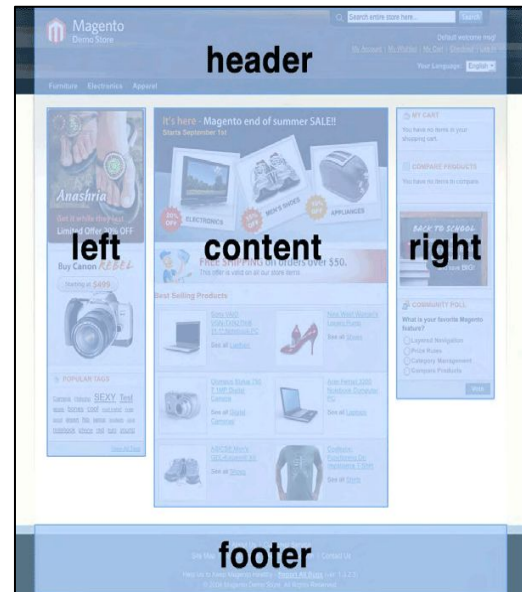
- Move a page element to another parent element
- Add content
- Remove a page element

Common layout instructions:

Use the following layout instructions to customize your layout:

- | | |
|---|---------------|
| ● <block> | ● <move> |
| ● <container> | ● <remove> |
| ● before and after attributes | ● <update> |
| ● <action> | ● <argument> |
| ● <referenceBlock> and <referenceContainer> | ● <arguments> |

[Refer this link for more info click](#)



Layout file types

For a particular page, its layout is defined by two major layout components: *Page layout file and Page configuration file*.

- A page layout file defines the page wireframe, for example, *one-column layout*.
- Page configuration is also an .xml file. It defines *the detailed structure (page header, footer, etc.), contents and page meta information, including the page layout used*. Page configuration features both main elements, blocks of particular classes and containers.
- We also distinguish the third type of layout files, generic layouts. They are .xml files which define the contents and detailed structure inside the <body> section of the HTML page markup. These files are used for pages returned by AJAX requests, emails, HTML snippets and so on.

Page layout files conventional location

Conventionally page layouts must be located as follows:

Module page layouts:

<module_dir>/view/frontend/page_layout

Theme page layouts:

<theme_dir>/<Namespace>_<Module>/page_layout

Page layouts declaration

To be able to use a layout for actual page rendering, you need to declare it in *layouts.xml*. Conventionally layout declaration file can be located in one of the following locations:

Module layout declarations: *<module_dir>/view/frontend/layouts.xml*

Theme layout declaration:

<theme_dir>/<Namespace>_<Module>/layouts.xml

Common Layout Instructions

<container>

A structure without content that holds other layout elements such as blocks and containers.

Details: A container renders child elements during view output generation. It can be empty or it can contain an arbitrary set of <container> and <block> elements.

```
<container name="header.container" as="header_container" label="Page Header Container"
htmlTag="header" htmlClass="page-test" before="main.content" />
```

<action>

Details: Used to set up the execution of a certain method of the block during block generation; the <action> node must be located in the scope of the <block> node.

```
<block class="Magento\Module\Block\Class" name="block">
    <action method="setText"> <argument name="text" translate="true" xsi:type="string">Text</argument></action>
    <action method="setEnabled"><argument name="enabled" xsi:type="boolean">true</argument></action>
</block>
```

Common Layout Instructions

`<referenceBlock>` and `<referenceContainer>`

Updates in `<referenceBlock>` and `<referenceContainer>` are applied to the corresponding `<block>` or `<container>`.

For example, if you make a reference by `<referenceBlock name="right">`, you're targeting the block `<block name="right">`.

Attributes:

```
<referenceBlock name="block.name" remove="true" />
```

```
<referenceContainer name="container.name" display="false" />
```

The display attribute is optional and its default value is true.

You are always able to overwrite this value in your layout. In situation when remove value is true, the display attribute is ignored.

Common Layout Instructions

`<move>`

Sets the declared block or container element as a child of another element in the specified order.

```
<move element="name.of.an.element" destination="name.of.destination.element" as="new_alias"
after="name.of.element.after" before="name.of.element.before"/>
```

`<update>`

Includes a certain layout file. Used as follows: The specified handle is "included" and executed recursively.

```
<update handle="{name_of_handle_to_include}"/>
```

`<page_layouts>` and `<page>`

Page_layout is used for declaring template layouts and Page is used for style and declare all content block.

Layout Folder Structure

Folder structure for page layout files
which are under **Vendor Folder**

```
<Vendor>/magento
├── module-theme/view/frontend/
│   ├── layout/
│   │   ├── default.xml
│   │   ├── default_head_blocks.xml
│   │   └── page_layout/
│   │       ├── 1column.xml (template files)
│   │       ├── 2columns-left.xml
│   │       ├── 3columns.xml
│   └── layout.xml
```

Folder structure for page layout files
which are under **Theme Directory**

```
app/design/frontend/<Vendor>/<theme_dir>
├── Magento_Theme/
│   ├── layout/
│   │   ├── default.xml
│   │   └── default_head_blocks.xml
```

**Note: If there will be any changes required in base templates (e.g 1column.xml). Then only page_layout folder will be added to theme directory.*

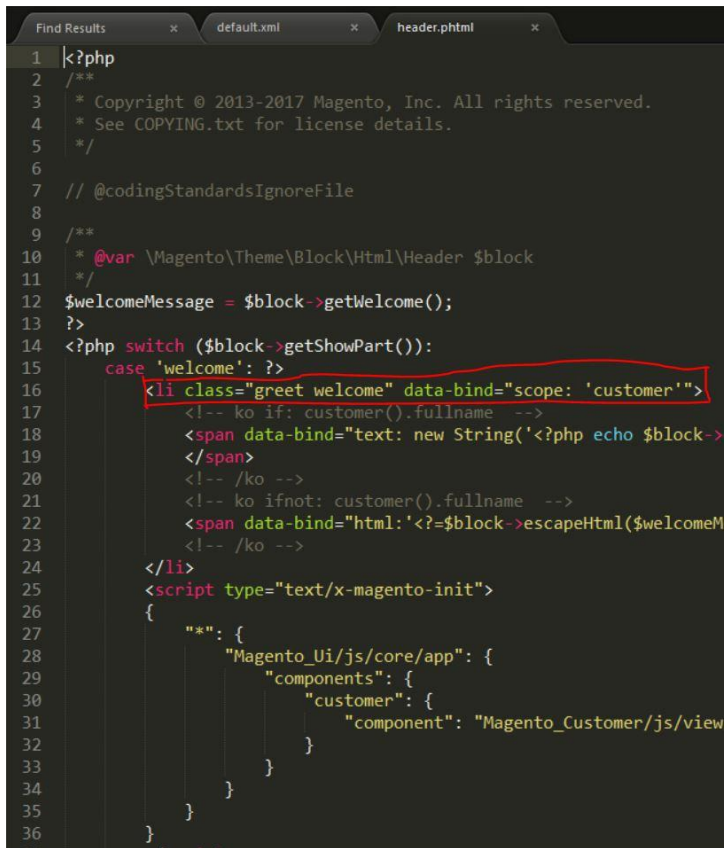
Template Customization Walkthrough

To Customize a Template:

- Locate the template which is associated with the page/block you want to change using template hints.
- Copy the template to your theme folder according to the template storing convention.
- Make the required changes.

To add a new template in a theme:

- Add a template in your theme directory according to the template storing convention.
- Assign your template to a block in the corresponding layout file.



```
1 |<?php
2 /**
3  * Copyright © 2013-2017 Magento, Inc. All rights reserved.
4  * See COPYING.txt for license details.
5  */
6
7 // @codingStandardsIgnoreFile
8
9 /**
10  * @var \Magento\Theme\Block\Html\Header $block
11  */
12 $welcomeMessage = $block->getWelcome();
13 ?>
14 <?php switch ($block->getShowPart()):
15     case 'welcome': ?>
16         <li class="greet welcome" data-bind="scope: 'customer'">
17             <!-- ko if: customer().fullname -->
18                 <span data-bind="text: new String('<?php echo $block->
19                 </span>
20             <!-- /ko -->
21             <!-- ko ifnot: customer().fullname -->
22                 <span data-bind="html: '<?=$block->escapeHtml($welcomeM
23             <!-- /ko -->
24         </li>
25         <script type="text/x-magento-init">
26         {
27             "": {
28                 "Magento_Ui/js/core/app": {
29                     "components": {
30                         "customer": {
31                             "component": "Magento_Customer/js/view
32                         }
33                     }
34                 }
35             }
36         }
```

Conventional templates location

Templates are stored in the following locations:

Module templates:

```
<module_dir>/view/frontend/templates/<path_to_templates>
```

Theme templates:

```
<theme_dir>/<Namespace>_<Module>/templates/<path_to_templates>
```

Here <path_to_templates> might have several levels of directory nesting, or might be empty. Examples:

```
<Magento_Catalog_module_dir>/view/frontend/templates/product/widget/new/content/new_grid.phtml
```

```
<Magento_Checkout_module_dir>/view/frontend/templates/cart.phtml
```


Templates overriding

For template files with the same name, the following is true: theme templates override module templates, and those of a child theme override parent theme templates.

This mechanism is the basis of the template customization concept in Magento application: to change the output defined by a certain default template, you need to override one in your custom theme.

Overriding templates is described with more details in the Theme Inheritance article.

Root template

In Magento there's a special template which serves as root template for all storefront pages in the application: `<Magento_Theme_module_dir>/view/base/templates/root.phtml`.

Unlike other templates, root.phtml contains **the doctype specification and contributes to <head> and <body> sections of all pages rendered by Magento application**. But similar to other templates, root.phtml can be overridden in a theme.

Customize Email Templates

Email templates are stored in the `<module_dir>/view/<area>/email` directory of their respective modules. For example, the template for the new order transactional email for the Sales module is located in `<Magento_Sales_module_dir>/view/frontend/email/order_new.html`.

You can add custom templates as physical files in your custom theme or create them using the Magento Admin. Both approaches are described in the following sections.

- Customize email templates using a theme
- Customize email templates using the Magento Admin

Customize email templates using a theme

Override email templates by creating templates in a new directory in your custom theme, using this pattern: `<theme_dir>/<ModuleVendorName>_<ModuleName>/email`. For example, to override the New Order email template, create a template named `order_new.html` in the `<theme_dir>/Magento_Sales/email` directory.

Customize Email Templates

Customize email templates using a theme

Template fallback is supported for email templates, so parent themes of your current theme are searched for templates.

Customize email templates using the Magento Admin

Any templates configured in the Magento Admin take precedence over default or theme-based templates.

1. In the Magento Admin, navigate to **MARKETING > Communications > Email Templates**
2. Click [Add New](#) Template.
3. If you want to use a default template as a starting point, in the Load default template section, choose the template and click Load Template. The path to the configuration settings for each default template displays in the Currently Used For field in the Template Information section.

Make note of this path because you will need it later when you configure this new template to be used instead of the default template.

Customize Email Templates

Customize email templates using the Magento Admin

- In Template Name, enter a name to identify the template in the Magento Admin.
- In Template Subject, add plain text to use as the Subject of the emails sent using the template you create. This field can contain system variables.
- Customize template content. For details, see the section on customizing content.
- In Template Styles, optionally add CSS styles for the template. These styles are added inside of a `<style>` tag in the `<head>` of the email. Typically, you'll use the LESS files to make style changes to emails because some email clients don't support styles in `<style>` tags.
- Click Save Template.

The screenshot shows the 'New Template' form in the Magento Admin interface. At the top, there are navigation links: 'Back', 'Reset', 'Convert to Plain Text', 'Preview Template', and a red 'Save Template' button. Below this is a 'Load default template' section with a dropdown menu set to 'New Order' and a 'Load Template' button. The 'Template Information' section shows the 'Currently Used For' path: 'Stores -> Configuration -> Sales Emails -> Order -> New Order Confirmation Template (Default Config)'. The 'Template Name' field contains 'Test order'. The 'Template Subject' field contains the Magento variable `{{var store.getFrontendName()}}: New Order # {{var order.increment_id}}`. Below these fields is an 'Insert Variable...' button. The 'Template Content' field contains a complex HTML template with PHP and Magento variables, including a table for a greeting and a section for account information. The 'Template Styles' field is currently empty.

New Template

← Back Reset Convert to Plain Text Preview Template Save Template

Load default template

Template New Order

Load Template

Template Information

Currently Used For Stores -> Configuration -> Sales Emails -> Order -> New Order Confirmation Template (Default Config)

Template Name * Test order

Template Subject * {{var store.getFrontendName()}}: New Order # {{var order.increment_id}}

Insert Variable...

Template Content *

```
((template config_path="design/email/header/template"))
<table>
<tr class="email-intro">
<td>
<div class="greeting">{{trans "Hi customer_name," customer_name=$order.getCustomerName()}}</div>
{{trans "Thank you for your order from %store_name%" store_name=$store.getFrontendName()}}
{{trans "Once your package ships we will send you a tracking number."}}
{{trans "You can check the status of your order by <a href='%account_url%'>logging into your account</a>."
account_url=$this.getUrl($store, $customer/account?) }}
</td>
</tr>
</table>
{{trans "If you have questions about your order, you can email us at <a href='mailto:%store_email%'>%store_email%'</a> or call us at <a href='tel:%store_phone%'>%store_phone%'</a>." store_phone=$store_phone [raw]}}
{{depend store_phone}}
{{trans "Our hours are <span class='no-link'%store_hours%'>%store_hours%'</span>." store_hours=$store_hours [raw]}}
{{/depend}}
</td>
</tr>
</table>
```

Template Styles

Customize Email Templates

Customize email templates using the Magento Admin

- Now that you have created a template, you must configure that template to be used:
 - If you haven't done so already, log in to the Magento Admin as an administrator.
 - Click STORES > Settings > Configuration > SALES > Sales Emails.
 - In the left pane, locate the section that contains the template you want to override. This is the section referenced by Currently Used For in your new template. (See step 3 earlier in this section.)
 - For example, if you created a "New Order" template, the configuration section is Order as the following figure shows.
 - Select your newly created template from the list.
 - Click Save Config.

General Settings

Order

Enabled Yes [STORE VIEW]

New Order Confirmation Email Sender Sales Representative [STORE VIEW]

New Order Confirmation Template Test order [STORE VIEW]
Email template chosen based on theme fallback when "Default" option is selected.

New Order Confirmation Template for Guest New Order for Guest (Default) [STORE VIEW]
Email template chosen based on theme fallback when "Default" option is selected.

Send Order Email Copy To [STORE VIEW]
Comma-separated

Send Order Email Copy Method Bcc [STORE VIEW]

[For more details refer this link](#)

Include CSS

How Magento stylesheet files are organized

Conventionally, CSS and LESS files are stored only in themes. Module directories do not contain any default styles.

In a theme directory, stylesheets are stored in the following locations:

- */<Namespace>_<Module>/web/css (Module-specific styles)*
- */web/css (Contains the following)*
 - **print.less:** used to generate styles for the printed version of store pages.
 - **_styles.less** - a composite file, which includes all LESS files used in the theme. The underscore sign ("_") in a file name conventionally means that a file is not used independently, but is included in other files.
 - **styles-m.less:** used to generate mobile-specific styles, includes _styles.less
 - **styles-l.less:** used to generate desktop-specific styles, includes _styles.less.
 - **/source:** this subdirectory contains LESS configuration files that invoke mixins from the Magento UI library
 - **/source/_theme.less:** overrides the default Magento UI library variables values.

Include CSS

In the Magento application, the recommended way to include stylesheets is to specify them in layout files.

Usually, the stylesheets you include should be available for all store pages. To achieve this, include your CSS in [default_head_blocks.xml of the Magento Theme module](#), which defines the default <head> page section for all Magento pages. The recommended way to do this is adding an extending default_head_blocks.xml in your theme, and including the required stylesheets in this file.

Your custom default_head_blocks.xml should be located as follows:

<theme_dir>/Magento_Theme/layout/default_head_blocks.xml.

To include a CSS file, add the `<css src="<path>/<file>" media="print|<option>" />` block in `<head>` section in a layout file.

Include CSS

`<path>` is specified relative to the theme web directory (`<theme_dir>/web`) For example, the following illustrates how stylesheets are included in the default Blank theme:

`<Magento_Blank_theme_dir>/Magento_Theme/layout/default_head_blocks.xml`

```
<page xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="urn:magento:framework:View/Layout/etc/page_configuration.xsd">
    <head>
        <css src="css/styles-m.css" />
        <css src="css/styles-l.css" media="screen and (min-width: 768px)" />
        <css src="css/print.css" media="print" />
    </head>
</page>
```


CSS preprocessing

The .less files from which the .css files included in layout are compiled. For example, in one of the layout files of the Magento Blank theme, the following .css files are included:

```
<head>  
  <css src="css/styles-m.css" />  
  <css src="css/styles-l.css" media="screen and (min-width: 768px)" />  
  <css src="css/print.css" media="print" />  
</head>
```

The root source files for the Blank theme:

- `<Magento_Blank_theme_dir>/web/css/styles-m.less`
- `<Magento_Blank_theme_dir>/web/css/styles-l.less`
- `<Magento_Blank_theme_dir>/web/css/print.less`

LESS compilation modes

In the Magento application, the following modes of compiling .less files to CSS are implemented:

Server-side LESS compilation.

This is the default compilation mode, and is the only option in production application mode. In this case the compilation is performed on the server, using the LESS PHP library.

Client-side LESS compilation.

When your application is not in the production mode, you can set Magento to compile .less files in a browser, using the native less.js library

To set the compilation mode, do the following:

- In the Magento Admin, navigate to **Stores > Configuration > ADVANCED > Developer.**
- In the Store View drop-down field, select **Default Config.**
- Under Front-end development workflow, in the **Workflow type field,** select the **compilation mode.**
- To save the settings, click **Save Config.**
- Make sure that the same compilation mode is set for each configuration scope. That is, check the Front-end development workflow option having **switched the Store View drop-down field to the website scope first,** and then to the store view. Change the option to match the default config if it is different.

LESS compilation modes

Server-side LESS compilation

The following paragraph describes how the LESS preprocessor works in server-side compilation mode. For each CSS file included in the layouts, LESS preprocessor does the following:

- Checks if the requested .css file is found. If it is found, the preprocessor stops its execution. Otherwise, it proceeds to the next step.
- Changes the extension of the requested file to .less and tries to find the file using the Magento fallback mechanism. If the .less file is not found, LESS preprocessor stops its execution. Otherwise, it proceeds to the next step.
- Reads .less file contents and resolves @magento_import and default LESS @import directives.
- Resolves all paths in .less files to relative paths in the system using the Magento fallback mechanism. All files resolved by the LESS preprocessor are copied to var/view_preprocessed/less. Imported files are processed recursively.
- All source files are passed to the PHP LESS compiler. The resulting compiled .css files are published to pub/static/frontend/<Vendor>/<theme>/<locale>.

Styles debugging in server-side compilation mode

In server-side LESS compilation mode, to have your changes applied, clear `pub/static/frontend/<Vendor>/<theme>/<locale>` by deleting the directory in the file system, and reload the store pages to trigger compilation and publication.

LESS compilation modes

Client-side LESS compilation

The client-side compilation flow is similar to server-side. The difference is in the set of files, published to pub/static on the last step. In the client-side mode, the following files are published to the pub/static/frontend/<Vendor>/<theme>/<locale> directory:

- root source (.less) files with resolved @magento_import directive
- symlinks to the root source file that do not contain @magento_import
- symlinks to all other .less files imported recursively by the @magento_import and @import directives

For Practicals

Step to be considered

- Magento2 development Modes
- CLI imp commands for UI
- Installation
- Theme creation
- Theme configuration
- Module customization
 - Xml
 - Templates
 - Overrides
 - Magento template referring functions (php codes)
- Less creation
 - Compilation
 - XML configuration
- CMS Section
- Magento Admin few things

1. Steps to apply theme:

- a. Create theme, deploy static content then apply it in backend
- b. Default.xml has to be present in theme
- c. If two themes have same name magento will stop working in cmd as well as everywhere.

Magento Modes

Overview of setting Magento modes

To improve security and ease-of-use, we added a command that switches Magento modes from developer to production and vice versa.

Production mode also has better performance because static view files are populated in the `pub/static` directory and because of code compilation.

When you change to developer or production mode, we clear the contents of following directories:

```
var/cache  
var/di  
var/generation  
var/view_preprocessed  
pub/static
```

Exceptions:

- `.htaccess` files are not removed
- `pub/static` contains a file that specifies the version of static content; this file is not removed

Show Magento Modes

Command usage:

```
magento deploy:mode:show
```

A message similar to the following displays:

```
Current application mode: developer.
```

Change Magento Mode

Command usage:

```
magento deploy:mode:set {mode} [-s|--skip-compilation]
```

A message similar to the following displays:

```
Enabled production mode.
```

where,

{mode} is required; it can be either developer or production --skip-compilation is an optional parameter you can use to skip code compilation when you change to production mode.

Change to developer mode

When you change from production to developer mode, you should clear generated classes and Object Manager entities like proxies to prevent unexpected errors. After doing so, you can change modes. Use the following steps:

- If you're changing from production mode to developer mode, delete the contents of the **var/generation** and **var/di** directories:

```
rm -rf <your Magento install dir>/var/di/* <your Magento install dir>/var/generation/*
```

- Set the mode:

```
magento deploy:mode:set developer
```

- The following message displays:

```
Switched to developer mode.
```


CLI imp commands for Frontend

When you change from production to developer mode, you should clear generated classes and Object Manager entities like proxies to prevent unexpected errors. After doing so, you can change modes. Use the following steps:

- If you're changing from production mode to developer mode, delete the contents of the **var/generation** and **var/di** directories:

```
rm -rf <your Magento install dir>/var/di/* <your Magento install dir>/var/generation/*
```

- Set the mode:

```
magento deploy:mode:set developer
```

- The following message displays:

```
Switched to developer mode.
```