

## Design & Theme

Your store's theme is like a window dressing that can be changed for a season or promotion. In this section, you will learn about page layouts, how to make simple HTML changes, and apply a new theme to your store.

## Contents

Design Menu

Page Setup

HTML Head

Header

Footer

Page Layout

Standard Page Layouts

Storefront Examples

Layout Updates

Standard Block Layout

Layout Update Examples

Layout Update Syntax

Controlling Block Order

XML Load Syntax

Themes

Using the Default Theme

Installing a New Theme

Theme Assets

Scheduling Design Changes

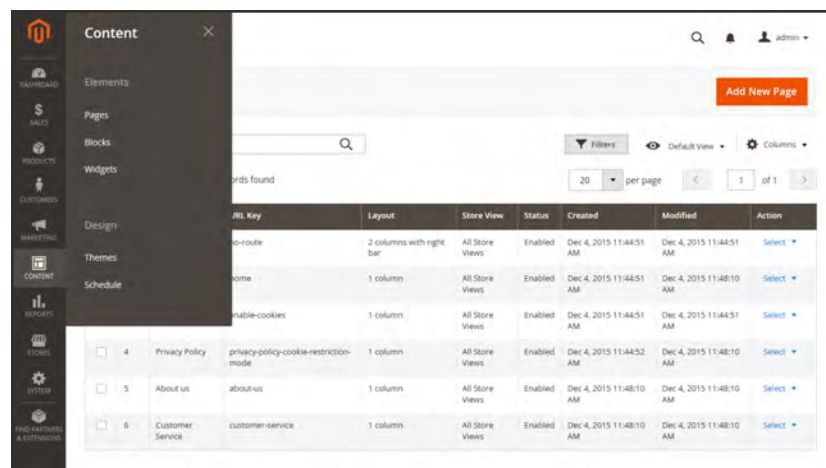


## CHAPTER 35:

# Design Menu

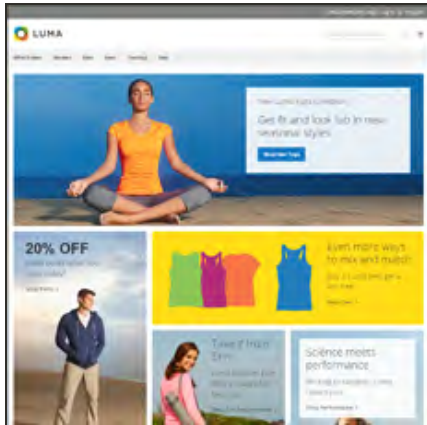
Magento provides a variety of easy-to-use design options that you can use to make simple changes to your store. In addition, you will find many professionally designed themes available on Magento Connect. Like the window dressing of your store, you can change the theme for the season or for a promotion.

More advanced users will appreciate the flexibility of working with Magento's object-oriented environment that assembles pages from separate components. After you understand the basics, you'll appreciate working in such a flexible and fluid environment. To learn more, see the [Magento Design's Guide](#).



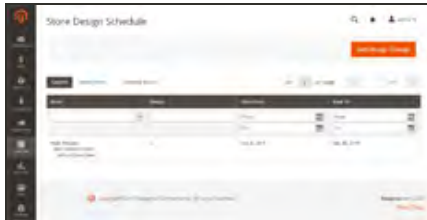
*Design Menu*

## Menu Options



## Themes

The theme determines the visual presentation of your store, and consists of a collection of layout files, template files, translation files, and skins.



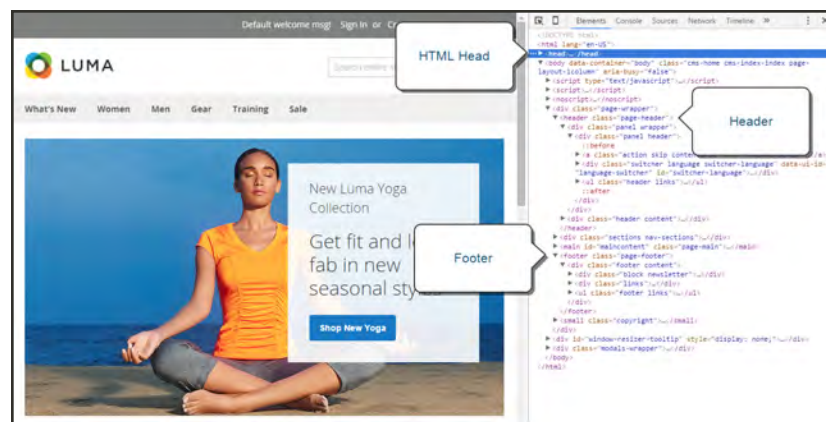
## Schedule

Themes can be activated for a period of time, according to a schedule. Use the schedule to plan theme changes in advance for a season or promotion.

## CHAPTER 36:

# Page Setup

The main sections of the page are controlled, in part, by a set of standard HTML tags. Some of these tags can be used to determine the selection of fonts, color, size, background colors, and images that are used in each section of the page. Other settings control page elements such as the logo in the header, and the copyright notice in the footer. The Page Setup sections correspond to the underlying structure of the HTML page, and many of the basic properties can be set from the Admin.



*HTML Page Sections*

## HTML Head

The settings in the HTML Head section correspond to the `<head>` tag of an HTML page, and can be configured for each store view. In addition to meta data for the page title, description, and keywords, the section includes a link to the favicon, and miscellaneous scripts.

Instructions for search engine robots and the display of the store demo notice are also configured in this section.

### To configure the HTML Head:

1. On the Admin sidebar, tap **Stores**. Then under **Settings**, choose **Configuration**.
2. In the panel on the left under **General**, choose **Design**.
3. If you have multiple stores or views, set the **Store View** in the upper-left corner to the store view where the configuration applies. (When configuring a specific store view, you must clear the Use Default checkbox after each field so new values can be entered.)
4. Expand ☑ the **HTML Head** section.
5. Update the fields as needed. (See the [Configuration Reference](#) for a description of each field,)
6. When complete, tap **Save Config**.

HTML Head

Favicon Icon

Choose File

No file chosen

Allowed file types: ICO, PNG, GIF, JPG, JPEG, APNG, SVG. Not all browsers support all these formats!

[STORE VIEW]

Default Title

Magento Commerce

[STORE VIEW]

Title Prefix

[STORE VIEW]

Title Suffix

[STORE VIEW]

Default Description

Default Description

[STORE VIEW]

Default Keywords

Magento, Varien, E-commerce

[STORE VIEW]

Miscellaneous Scripts

<link rel="stylesheet" type="text/css" media="all" href="{{MEDIA\_URL}}styles.css" />

This will be included before head closing tag in page HTML.

[STORE VIEW]

Display Demo Store Notice

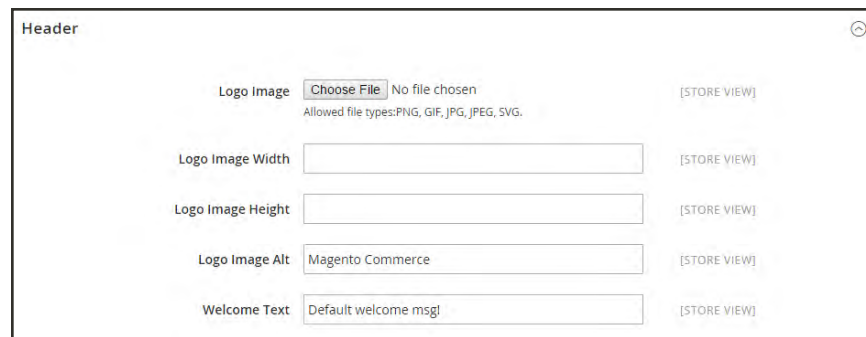
No

[STORE VIEW]

HTML Head

## Header

The Header section identifies the path to your store logo, and specifies the logo alt text and welcome message.




The screenshot shows the 'Header' configuration panel in the Magento Admin interface. The panel has a title 'Header' in the top left and a close icon in the top right. It contains five configuration rows, each with a label, a text input field, and a '[STORE VIEW]' link on the right. The first row is for 'Logo Image', featuring a 'Choose File' button, the text 'No file chosen', and a note 'Allowed file types: PNG, GIF, JPG, JPEG, SVG.'. The other four rows are for 'Logo Image Width', 'Logo Image Height', 'Logo Image Alt' (with the value 'Magento Commerce'), and 'Welcome Text' (with the value 'Default welcome msg!').

Field	Value	Action
Logo Image	Choose File   No file chosen <small>Allowed file types: PNG, GIF, JPG, JPEG, SVG.</small>	[STORE VIEW]
Logo Image Width		[STORE VIEW]
Logo Image Height		[STORE VIEW]
Logo Image Alt	Magento Commerce	[STORE VIEW]
Welcome Text	Default welcome msg!	[STORE VIEW]

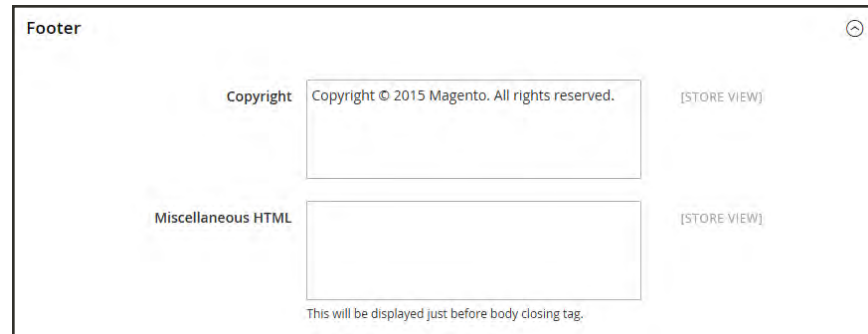
### Header

#### To configure the header:

1. On the Admin sidebar, tap **Stores**. Then under **Settings**, choose **Configuration**.
2. In the panel on the left, under **General**, choose **Design**.
3. Expand  the **Header** section. Then, make any changes necessary.
4. When complete, tap **Save Config**.

## Footer

The Footer configuration section is where you can update the **copyright notice** that appears at the bottom of the page, and enter miscellaneous scripts that must be positioned before the closing `<body>` tag..



The screenshot shows the 'Footer' configuration section. It has a title 'Footer' at the top left and a collapse icon at the top right. There are two main input areas: 'Copyright' and 'Miscellaneous HTML'. The 'Copyright' field contains the text 'Copyright © 2015 Magento. All rights reserved.' and has a '[STORE VIEW]' selector to its right. The 'Miscellaneous HTML' field is empty and also has a '[STORE VIEW]' selector to its right. Below the 'Miscellaneous HTML' field, there is a small note: 'This will be displayed just before body closing tag.'

*Footer*

### To configure the footer:

1. On the Admin sidebar, tap **Stores**. Then under **Settings**, choose **Configuration**.
2. In the panel on the left, under **General**, choose **Design**.
3. Expand ☑ the **Footer** section. Then, make any changes necessary.
4. When complete, tap **Save Config**.





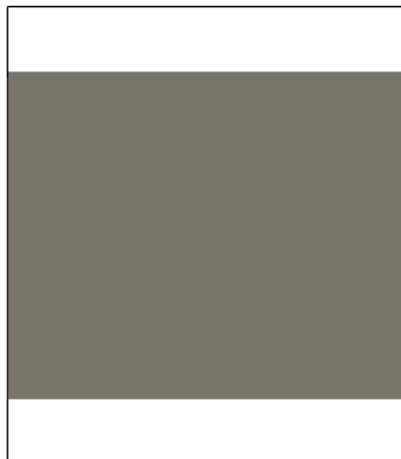
## CHAPTER 37:

# Page Layout

The layout of each page in your store consists of distinct sections, or containers, that define the header, footer, and content areas of the page. Depending on the layout, each page might have one, two, three columns, or more. You can think of the layout as the “floor plan” of the page.

Content blocks float to fill the available space, according to the section of the [page layout](#) where they are assigned to appear. You will discover that if you change the layout from a three-column to a two-column layout, the content of the main area expands to fill the available space, and any blocks that are associated with the unused side bar seem to disappear. However, if you restore the three-column layout, the blocks reappear. This fluid approach, or liquid layout, makes it possible to change the page layout without having to rework the content. If you are used to working with individual HTML pages, you will discover that this modular, “building block” approach requires a different way of thinking.

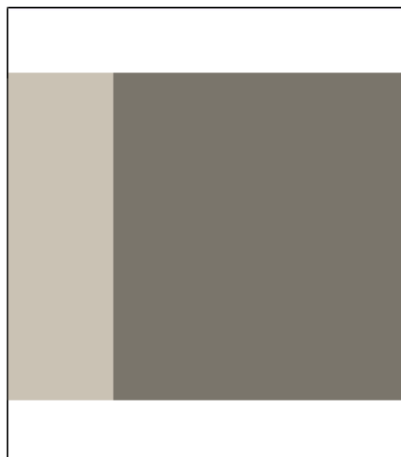
## Standard Page Layouts



### 1 Column

---

The “1 Column” layout can be used to create a dramatic home page with a large image or focal point. It’s also a good choice for a landing page, or any other page that has a combination of text, images, and video.



### 2 Columns with Left Bar

---

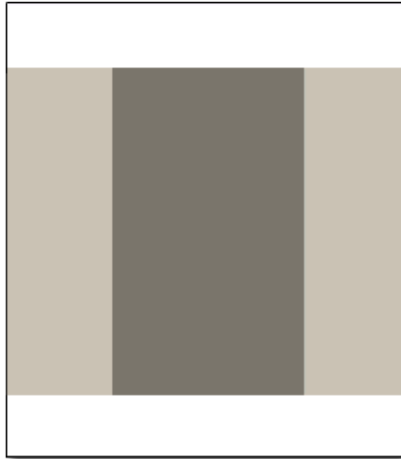
The “2 Columns with Left Bar” layout is often used for pages with navigation on the left, such as a catalog or search results pages with layered navigation. It is also an excellent choice for home pages that need additional navigation or blocks of supporting content on the left.



### 2 Columns with Right Bar

---

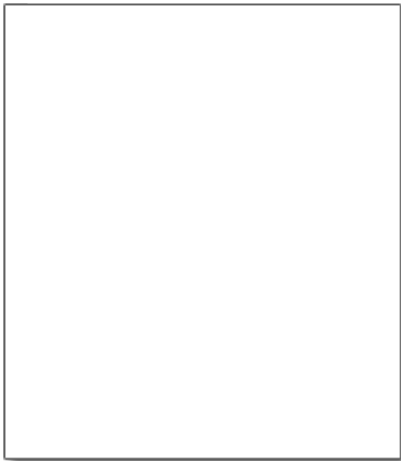
With a “2 Columns with Right Bar” layout, the main content area is large enough for an eye-catching image or banner. This layout is also often used for Product pages with blocks of supporting content on the right.



### 3 Columns

---

The “3 Column” layout has a center column that is wide enough for the main text of the page, with room on each side for additional navigation and blocks of supporting content.



### Empty

---

The “Empty” layout can be used to define custom page layouts. To learn more, see the [Magento Designer’s Guide](#).

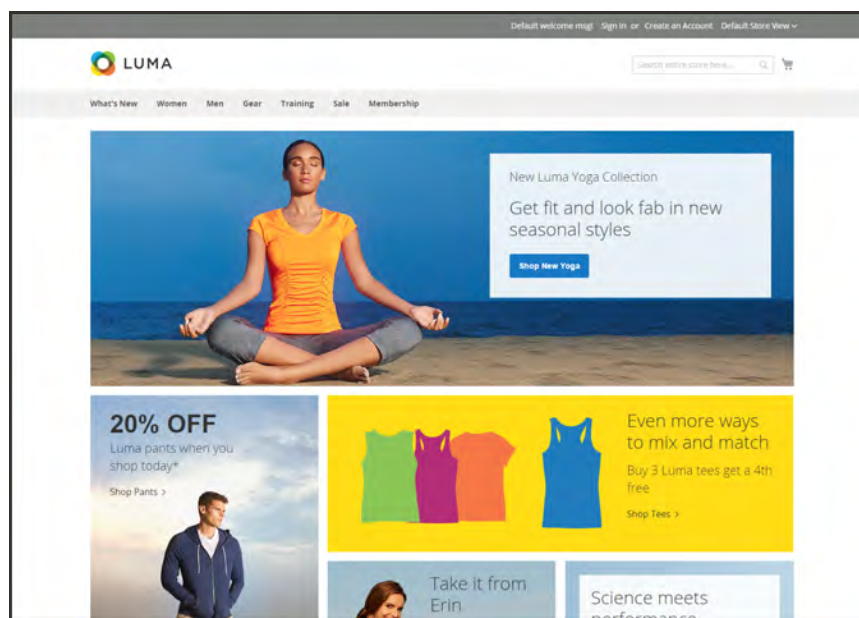
## Storefront Examples

The column dimensions are determined by style sheet of the theme. Some themes apply a fixed pixel width to the page layout, while others use percentages to make the page respond to the width of the window or device.

Most desktop themes have a fixed width for the main column, and all activity takes place within this enclosed area. Depending on your screen resolution, there is empty space on each side of the main column.

### 1 Column Layout

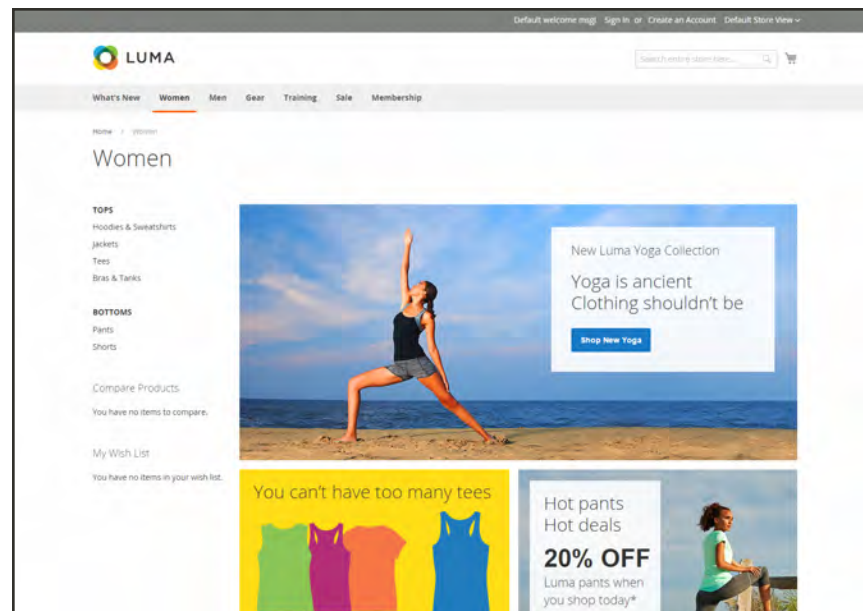
The content area of a “1 Column” layout spans the full-width of the main column. This layout is often used for a home page with a large banner or slider, or pages that require no navigation, such as a login page, splash page, video, or full-page advertisement.



*1 Column Layout*

## 2 Columns with Left Bar

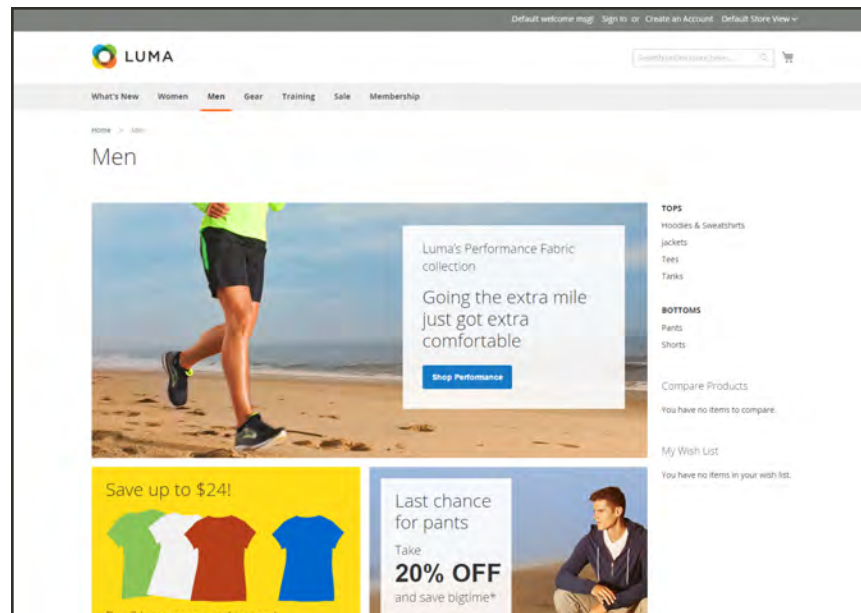
The content area of this layout is divided into two columns. The main content column floats to the right, and the side bar floats to the left.



*2 Columns Left Bar*

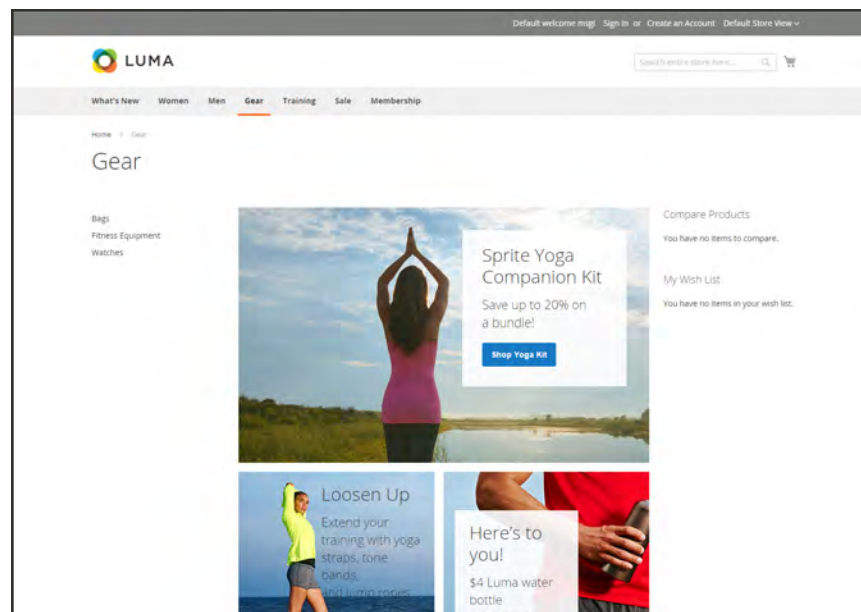
## 2 Columns with Right Bar

This layout is a mirror image of the other two-column layout. This time, the side bar floats to the right, and the main content column floats to the left.

*2 Columns Right Bar*

### 3 Columns

A 3-column layout has a main content area with two side columns. The left side bar and main content column are wrapped together, and float as a unit to the left. The other side bar floats to the right.

*3 Columns*

## Layout Updates

Before you begin working with custom layout updates, it is important to understand how the pages of your store are constructed, and the difference between the terms layout and layout update. The term **layout**<sup>1</sup> refers to the visual and structural composition of the page. However, the term **layout update**<sup>2</sup> refers to a specific set of XML instructions that determines how the page is constructed.

The XML layout of your Magento Community Edition 2.0 store is a hierarchical structure of blocks. Some elements appear on every page, and others appear only on specific pages. You can see how these structural blocks are referenced by examining the layout update code for your home page. To do so, simply open your home page in edit mode, and choose the Design tab to view the Page Layout section. Depending on the theme, it might contain instructions to remove blocks, unset blocks, and add blocks by referencing specific areas of the page layout.

In many cases, the same result can be achieved with the Frontend App tool. To place a block of content as a frontend app, you must identify the page, and the location on the page where you want the block to appear. You can use the Frontend App tool to place a block on most any page in your store, including the home page and all content pages. However, to place a block in the sidebar of a specific page, you must make the change by entering code as a layout update.

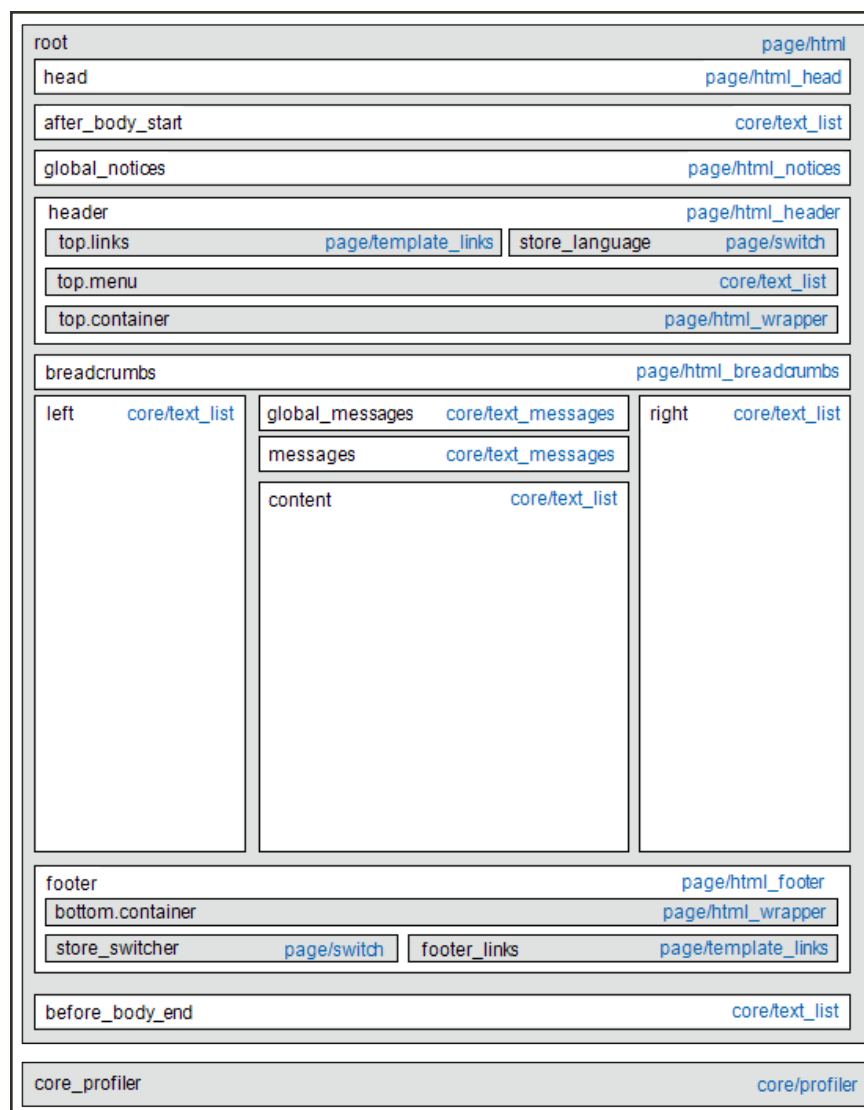
---

<sup>1</sup>The visual and structural composition of a page.

<sup>2</sup>A specific set of XML instructions that determines how the page is constructed.

## Standard Block Layout

In the following diagram, the block names that can be used to refer to a block in the layout are black, and the block types, or block class paths, are blue.



*Standard Block Layout*



## Block Descriptions

BLOCK TYPE	DESCRIPTION
<code>page/html</code>	There can be only one block of this type per page. The block name is "root," and, it is one of the few root blocks in the layout. You can also create your own block and name it "root," which is the standard name for blocks of this type.
<code>page/html_head</code>	There can be only one block of this type per page. The block name is "head," and it is a child of the root block. This block must not be removed from layout.
<code>page/html_notices</code>	There can be only one block of this type per page. The block name is "global_notices," and it is a child of the root block. If this block is removed from the layout, the global notices will not appear on the page.
<code>page/html_header</code>	There can be only one block of that type per page. The block name is "header," and it is a child of the root block. This block corresponds to the visual header at the top of the page, and contains several standard blocks. This block must not be removed.
<code>page/html_wrapper</code>	Although included in the default layout, this block is deprecated, and only is included to ensure backward compatibility. Do not use blocks of this type.
<code>page/html_breadcrumbs</code>	There can be only one block of this type per page. The name of this block is "breadcrumbs," and it is a child of the header block. This block displays breadcrumbs for the current page.
<code>page/html_footer</code>	There can be only one block of this type per page. The block name is "footer," and it is a child of the root block. The footer block corresponds to the visual footer at the bottom of the page, and contains several standard blocks. This block must not be removed.
<code>page/template_links</code>	There are two blocks of this type in the standard layout. The "top.links" block is a child of the header block, and corresponds to the top navigation menu. The "footer_links" block is a child of the footer block, and corresponds to the bottom navigation menu. It is possible to manipulate the template links, as shown in the examples.

**Block Descriptions (cont.)**

BLOCK TYPE	DESCRIPTION
<code>page/switch</code>	There are two blocks of this type in a standard layout. The "store_language" block is a child of the header block, and corresponds to the top language switcher. The "store_switcher" block is a child of the footer block, and corresponds to the bottom store switcher.
<code>core/messages</code>	There are two blocks of this type in a standard layout. The "global_messages" block displays global messages. The "messages" block is used to display all other messages. If you remove these blocks, the customer won't be able to see any messages.
<code>core/text_list</code>	This type of block is widely used throughout Magento, and is used as a placeholder for rendering children blocks.
<code>core/profiler</code>	There is only one instance of this type of block per page. It is used for the internal Magento profiler, and should not be used for any other purpose.

## Layout Update Examples

The following blocks types can be manipulated with custom layout instructions. Each action must be specified using the full syntax of the instruction. In the following examples, a simplified notation is used to refer to each action, which corresponds to the full syntax of the instruction.

### Full Syntax

```
?  
  
<!-- Action can be specified inside either a <block>  
or <reference> instruction. -->  
  
    <action method="someActionName">  
        <arg1>Value 1</arg1>  
        <arg2>Value 2</arg2>  
        <!--    -->  
        <argN>Value N</argN>  
    </action>  
  
<!--    -->
```

### Simplified Syntax

```
?  
  
someActionName($arg1, $arg2, ..., $argN)
```

## page/template\_links

### Syntax

ACTION	DESCRIPTION
<code>addLink(\$label, \$url, \$title, \$prepare</code>	Adds another link to the end of the list of existing links. Just specify the <code>\$label</code> (link caption), <code>\$url</code> (link URL) and <code>\$title</code> (link tooltip), and you'll see a new link in the corresponding place. The <code>\$prepare</code> parameter must be "true" if you want the URL to be prepared, or converted to the full URL from the shortened URL. For example, the new page becomes <code>BASE_URL/newpage</code> if prepared.
<code>removeLinkByUrl(\$url)</code>	Removes a link from the block by its URL. Note that the URL must be properly specified and exactly match corresponding URL of the link you want to remove.

## cms/block

### Syntax

ACTION	DESCRIPTION
<code>setBlockId(\$blockId)</code>	Specifies the ID of a CMS block, so its content can be fetched and displayed when the page is rendered.

```
?
<!--...-->

<reference name="content">
    <block type="cms/block" name="additional.info"
as="additionalInfo">
        <action method="setBlockId"><id>additional_info</id></action>
    </block>
</reference>
<!--    -->
```

**core/text**

A core/text block can be used to enter free form text directly into the template.

**Syntax**

ACTION	DESCRIPTION
<code>addText(\$textContent)</code>	Specifies text to be rendered as the block's content. After the text is specified, the layout update instructions must continue to be a valid XML statement. If you use HTML tags as part of the text, it is recommended to use:  <code>&lt;![CDATA[...]]&gt;</code>

```
?
<!--...-->

<reference name="content">
    <block type="core/text" name="test.block">
        <action method="addText">
            <txt><![CDATA[<h2>ATTENTION!</h2><p>Check your options
carefully before you submit.</p>]]></txt>
        </action>
    </block>
</reference>
<!-- -->
```

**page/html\_welcome**

This block can be used to duplicate the “Welcome, <USERNAME>!” message that appears in the header block. When the user is not logged in, the welcome message specified in the configuration appears.

## Layout Update Syntax

Custom layout updates can be applied to product category pages, product pages, and content page to achieve a variety of results, such as:

<code>&lt;block&gt;</code>	Create new block.
<code>&lt;reference&gt;</code>	Update existing content.
<code>&lt;action&gt;</code>	Assign actions to blocks.
<code>&lt;remove&gt;</code>	Remove blocks.

Any change made to the layout is applied when the associated entity—which can be either a product, category, or CMS page—becomes active in the frontend of the store.

Custom layout update instructions consist of well-formed XML tags, without the `<?xml ...>` declaration and root tag. As with normal XML, every tag must either be empty or properly closed, as shown in the following examples:

```
<tag attribute="value" />
<tag attribute="value"> ... </tag>
```

**<block>**

Creates a new block within the current context. Layout block nesting defines the ordering of block initialization location of the blocks on the page.

**Syntax**

NAME	VALUE
type	<p><b>*</b> block class path</p> <p>An identifier of the block class path that corresponds to the class of the block. See the list of the available block types below.</p>
name	<p><b>*</b> block name identifier</p> <p>A name that can be used to address the block in which this attribute is assigned. If you create a new block with the name that is the same as one of the existing blocks, your newly created block substitutes the previously existing block. See the list of names of existing blocks below.</p>
before	<p>block name   '-'</p> <p>Is used to position the block before a block with the name specified in the value. If "-" value used the block is positioned before all other sibling blocks.</p>
after	<p>block name   '-'</p> <p>Is used to position the block after a block with the name specified in the value. If "-" value used the block is positioned after all other sibling blocks.</p>
template	<p>template filename</p> <p>A template filename used for the specific block type. As you have no way to see the list of template files, use whatever template value is demanded for every block type listed below.</p>
as	<p>block alias</p> <p>An alias name by which a template calls the block in which this attribute is assigned. Sometimes it's necessary to specify the alias for a specific block type.</p>

**<reference>**

Changes the context for all included instructions to a previously defined block. An empty <reference> tag if of no use, because it affects only the instructions which are children.

**Syntax**

NAME	VALUE
name	* block name A name of a block to reference.

**<action>**

Used to access block API, in other words, call block's public methods. It is used to set up the execution of a certain method of the block during the block generation. Action child tags are translated into block method arguments. The list of all available methods depends on the block implementation (e.g. public method of the block class).

**Syntax**

NAME	VALUE
method	* block method name A name of the public method of the block class this instruction is located in that is called during the block generation.

**<remove>**

Removes an existing block from the layout.

**Syntax**

NAME	VALUE
name	* block name The name of the block to be removed.

**<extend>**

This instruction performs final modifications to blocks which are already part of the layout. Every attribute in the <block> instruction—except for the block name—is subject for change. In addition, the special attribute parent can be used to change the parent of the block. Simply put the name of the new parent block into the <extend> instruction, and the parent of the block that is referenced will be changed in the layout.



**Syntax**

NAME	VALUE
name	<div><div>*</div>block name</div> <div>The name of any block to be extended.</div>
*	<div>any other</div> <div>Any other attribute specific for the &lt;block&gt; instruction.</div>
parent	<div>block name</div> <div>The name of the block that should become a new parent for the referenced block.</div>
	<div><div>*</div>Indicates a required value</div>

## Controlling Block Order

Sometimes more than one content element is assigned to the same structural block. For example, there might be several block that appear in a sidebar. You can control the order of blocks by including a “before” or “after” positioning property in the code. To place a block either before, or after a specific block, replace the hyphen with the block identifier, as shown in the following examples:

`before="-"`      Places the block at the top of the sidebar, before other blocks.

`after="-"`      Places the block at the bottom of the sidebar, after other blocks.

### Code to Position Content Blocks

```
<block type="cms/block" before="-" name="left.permanent.callout">
<block type="cms/block" before="some-other-block"
name="left.permanent.callout">
<block type="cms/block" after="-" name="left.permanent.callout">
<block type="cms/block" after="some-other-block"
name="left.permanent.callout">
```

## XML Load Sequence

For developers, it is important to understand that blocks and layout updates must be loaded in the correct order, in keeping with the rules of precedence and **load sequence**<sup>1</sup> which determine how the page is rendered. Magento supports the following page layout scenarios:

### Scenario 1: Default Layout

The default layout consists of the visual elements that are visible from every page of the store. Whether it is a menu item, or a shopping cart block, each item has a **handle**<sup>2</sup> in the default section of the layout definition.

### Scenario 2: Changes to Specific Pages

The second case allows you to create a different layout for a specific page. The XML layout for specific pages is constructed in the same sequence that Magento loads modules, and is determined by the system configuration.

In addition to the instructions in the layout update files which are specific to each module, you can make a custom layout update that applies to a special case in the backend, and is merged each time the special case occurs.

---

<sup>1</sup>The order in which scripts are loaded into memory. To work correctly, some scripts must be loaded before others.

<sup>2</sup>In programming, a name used to reference an object.



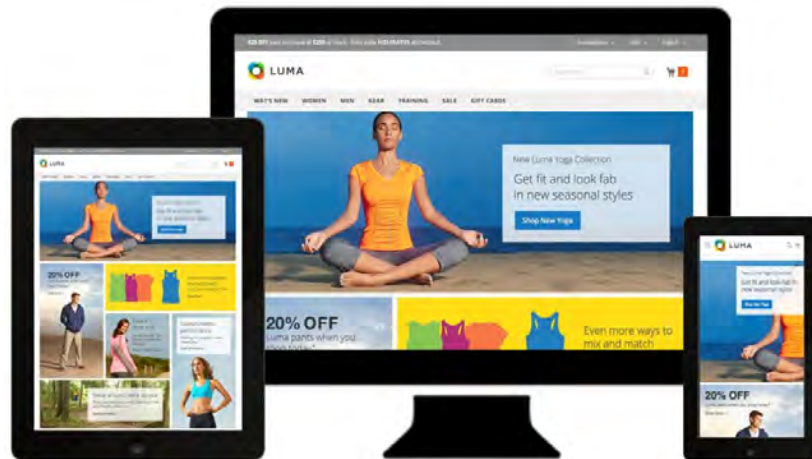


## CHAPTER 38:

# Themes

A theme is a collection of files that determines the visual presentation of your store. When you first install Magento Community Edition 2.0, the design elements of the store are based on the “Default” Theme. In addition to the initial default theme that comes with your Magento installation, there is a wide variety of themes that are available “off the shelf” on Magento Connect.

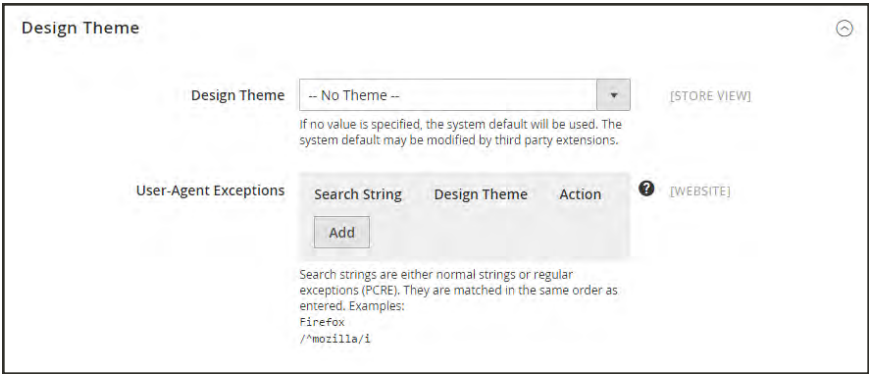
Magento themes include layout files, template files, translation files, and skins. A skin is a collection of supporting CSS, images, and JavaScript files that together, create the visual presentation and interactions that your customers experience when they visit your store. Themes and skins can be modified and customized by a developer or designer who has knowledge of Magento theme design and access to your server. To learn more, see the [Frontend Developer Guide](#).



*Luma Theme*

# Using the Default Theme

Magento’s default responsive theme renders the display of your storefront for different devices, and incorporates best practices for desktop, table, and mobile devices. To learn more, see the [Responsive Theme Developer’s Guide](#).



*Design Theme*

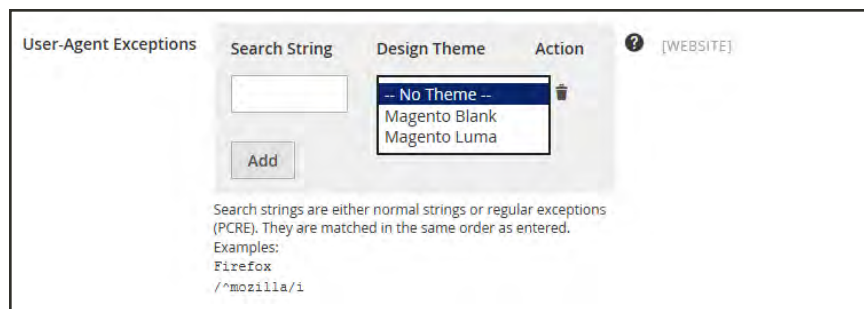
Some themes are designed to be used only with specific devices. When Magento detects a specific browser ID, or user agent, it uses the theme that is configured to be used for the specific browser. The search string can also include Perl-Compatible Regular Expressions (PCRE). To learn more, see: [User Agent](#).

```
Firefox
/^mozilla/i
```

### To view the default theme:

1. On the Admin sidebar, tap **Stores**. Then under **Settings**, choose **Configuration**.
2. In the panel on the left under **General**, choose **Design**.
3. Expand ☑ the **Design Theme** section.
4. Set **Design Theme** to the theme that you want to apply to the store.
5. If the theme is to be used for only a specific device, do the following:
  - a. Under **User-Agent Exceptions**, tap **Add**.
  - b. In the **Search String** field, enter the browser ID for the specific device.

Search strings are matched in the order they are entered.



*User-Agent Exceptions*

- c. Repeat the process to enter additional devices.
6. When complete, tap **Save Config**.

## Installing a New Theme


When you first install Magento, the design elements of the store are based on the “Default” theme. You can modify the theme, add themes created by others, or create new ones. To learn more, see the [Designer’s Guide](#).

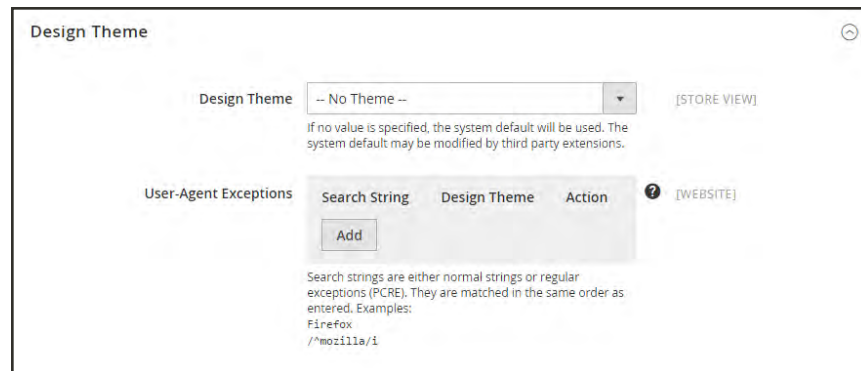
**Magento Connect** contains a wide selection of extensions that you can install to enhance the appearance of your store. The following example shows how to add a mobile theme from Magento Connect.

### Step 1: Install a New Theme

1. On the Admin sidebar, tap **Find Partners & Extensions**.
2. Under Magento Marketplace, tap **Visit Magento Marketplaces**.
3. Find the Magento 2 theme that you want to install, and follow the instructions to install the theme on your server.

### Step 2: Apply the Theme to Your Store

1. On the Admin sidebar, tap **Stores**. Then under **Settings**, choose **Configuration**.
2. In the panel on the left under **General**, choose **Design**.
3. Expand  the **Themes** section.
4. Set **Design Theme** to the new theme.



**Design Theme**

Design Theme  [STORE VIEW]

If no value is specified, the system default will be used. The system default may be modified by third party extensions.

User-Agent Exceptions

Search String	Design Theme	Action
<input type="button" value="Add"/>		

Search strings are either normal strings or regular exceptions (PCRE). They are matched in the same order as entered. Examples:  
Firefox  
/\*mozilla/i

[WEBSITE]

*Design Theme*

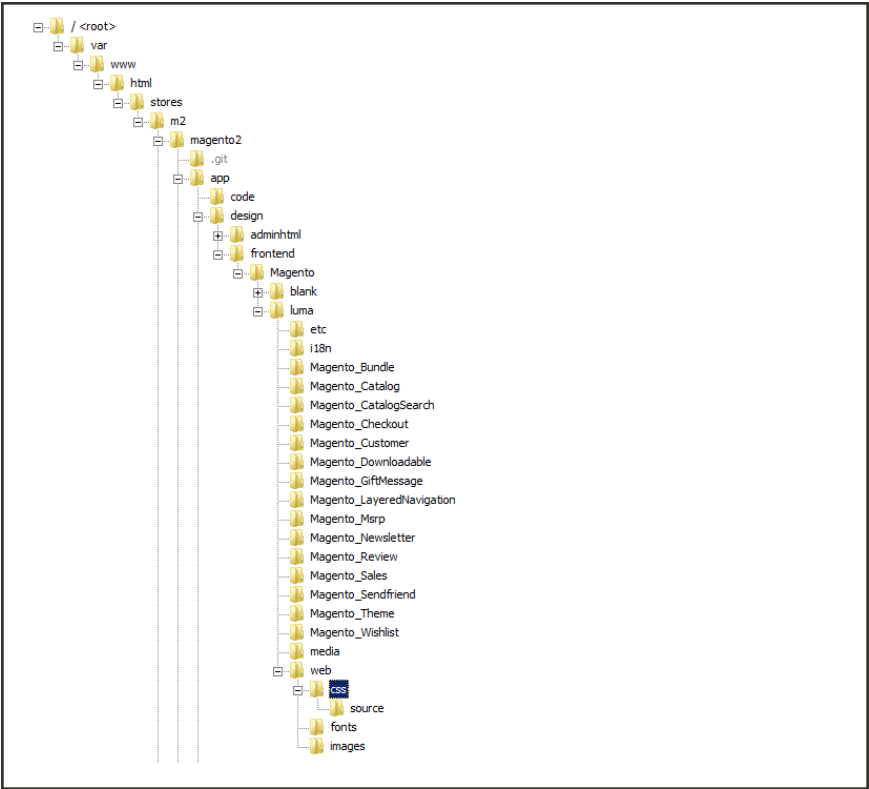
5. When complete, tap **Save Config.**



# Theme Assets

For a standard installation, the assets associated with a theme are organized in the web folder at the following location below the Magento root.

[magento\_root]/app/design/frontend/Magento/[theme\_name]/web



Theme Assets

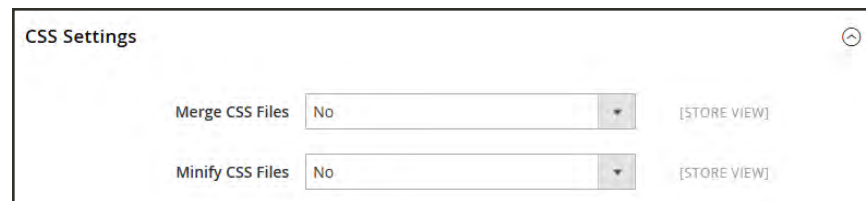
FOLDER	DESCRIPTION
CSS	Contains the CSS files that control the visual styling that is associated with the skin.
fonts	Contains any additional fonts that are used by the theme and that are not available by default on most systems.
images	Contains all images used by the theme, including buttons, background images, and so on.

## Merging CSS Files

As part of an effort to optimize your site and reduce page load time, you can reduce the number of separate CSS files by merging them into a single condensed file. If you open a merged CSS file, you'll find one continuous stream of text, with line breaks removed. Because you can't edit the merged file, it's best to wait until you are out of the development mode, and no longer making frequent changes to the CSS.

### To merge CSS files:

1. On the Admin sidebar, tap **Stores**. Then under **Settings**, choose **Configuration**.
2. In the panel on the left under **Advanced**, choose **Developer**.
3. Expand ☺ the **CSS Settings** section.

The screenshot shows the 'CSS Settings' configuration panel. It has a title bar 'CSS Settings' with a refresh icon on the right. Below the title bar, there are two settings. The first is 'Merge CSS Files' with a dropdown menu currently set to 'No' and a '[STORE VIEW]' button to its right. The second is 'Minify CSS Files' with a dropdown menu currently set to 'No' and a '[STORE VIEW]' button to its right.


*CSS Settings*

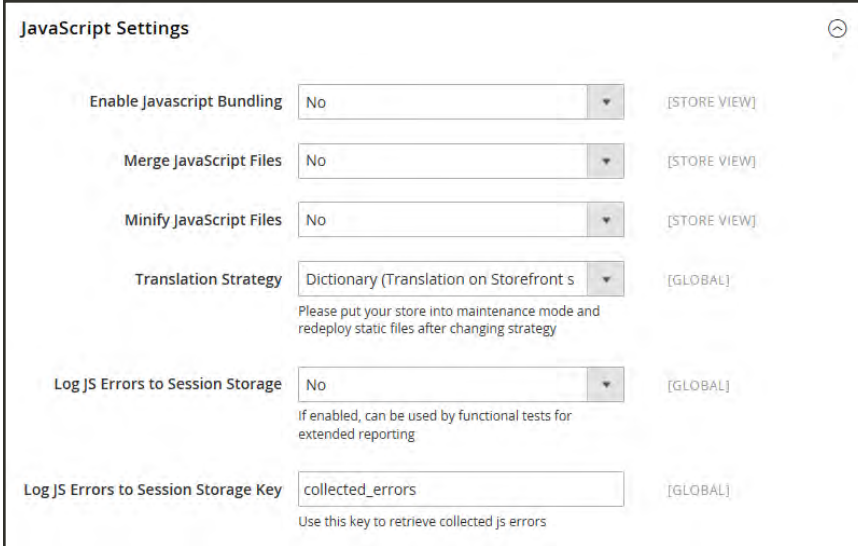
4. Set **Merge CSS Files** to “Yes.”
5. When complete, tap **Save Config**.

## Merging JavaScript Files

Multiple JavaScript files can be merged into a single, condensed file to reduce page load time. If you open a merged JavaScript file, you'll find one continuous stream of text, with line breaks removed. If you are finished with the development process, and the code contains no errors, you might consider merging the files.

### To merge JavaScript files:

1. On the Admin sidebar, tap **Stores**. Then under **Settings**, choose **Configuration**.
2. In the panel on the left under **Advanced**, choose **Developer**.
3. Expand  the **JavaScript Settings** section.



The screenshot shows the 'JavaScript Settings' panel. It contains several configuration options, each with a dropdown menu and a scope indicator. The options are: 'Enable Javascript Bundling' (set to 'No', scope '[STORE VIEW]'), 'Merge JavaScript Files' (set to 'No', scope '[STORE VIEW]'), 'Minify JavaScript Files' (set to 'No', scope '[STORE VIEW]'), 'Translation Strategy' (set to 'Dictionary (Translation on Storefront s)', scope '[GLOBAL]', with a note 'Please put your store into maintenance mode and redeploy static files after changing strategy'), 'Log JS Errors to Session Storage' (set to 'No', scope '[GLOBAL]', with a note 'If enabled, can be used by functional tests for extended reporting'), and 'Log JS Errors to Session Storage Key' (set to 'collected\_errors', scope '[GLOBAL]', with a note 'Use this key to retrieve collected js errors').

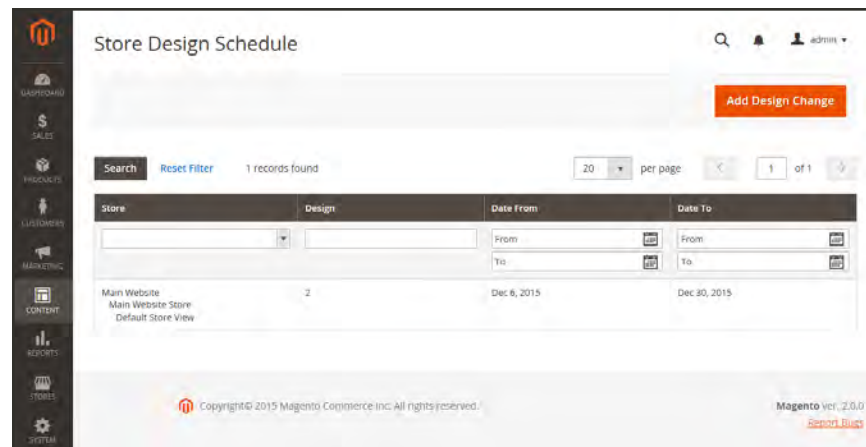
Setting	Value	Scope
Enable Javascript Bundling	No	[STORE VIEW]
Merge JavaScript Files	No	[STORE VIEW]
Minify JavaScript Files	No	[STORE VIEW]
Translation Strategy	Dictionary (Translation on Storefront s)	[GLOBAL]
Please put your store into maintenance mode and redeploy static files after changing strategy		
Log JS Errors to Session Storage	No	[GLOBAL]
If enabled, can be used by functional tests for extended reporting		
Log JS Errors to Session Storage Key	collected_errors	[GLOBAL]
Use this key to retrieve collected js errors		

*JavaScript Settings*

4. Set **Merge JavaScript Files** to “Yes.”
5. When complete, tap **Save Config**.

## Scheduling Design Changes

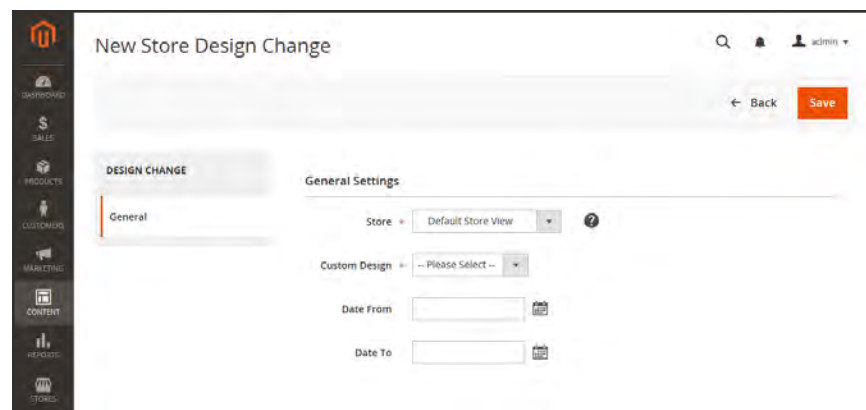
Design changes can be scheduled in advance, so they go into effect on schedule. You can use scheduled design changes for seasonal changes, promotions, or just to add variation.




*Store Design Schedule*

### To schedule a design change:

1. On the Admin sidebar, tap **Content**. Then under **Design**, choose **Schedule**.
2. Tap **Add Design Change**. Then under General Settings, do the following:



*New Design Change*

- a. Set **Store** to the view where the change applies.
- b. Set **Custom Design** to the theme, or variation of a theme, that is to be used.
- c. To define the period when the change is in effect, use the calendar  to choose the values for the **Date From** and **Date To** fields.
3. When complete, tap **Save**.