Magento® U

Student Guide

Core Principles for Theming in Magento 2

magento.com/training/overview

v. 1.0

Magento® U

training@magento.com

magento.com/training/overview

magento.com/training/catalog/certification

Unit One Contents

About This Guide

This guide uses the following symbols in the notes that follow the slides.

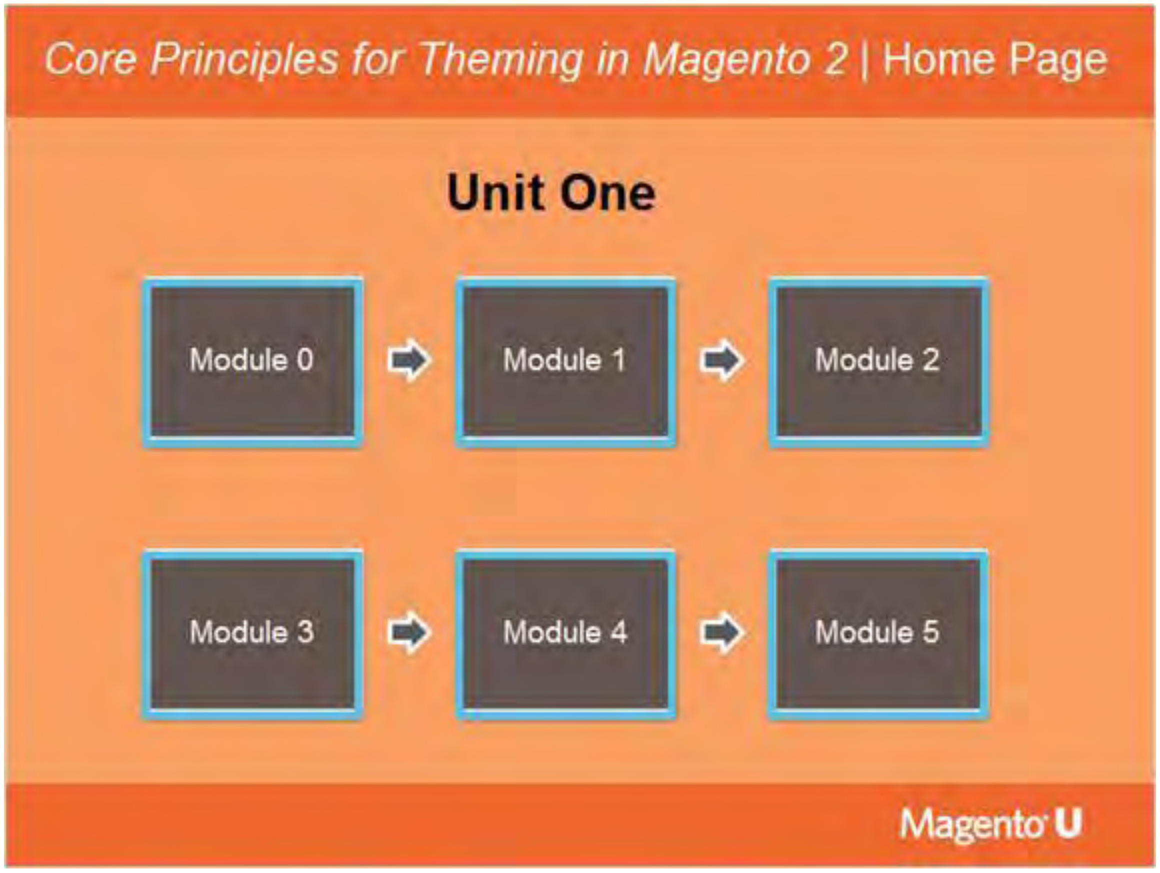
Symbol	Indicates…
📝	A note, tip, or other information brought to your attention.
❗	Important information that you need to know.
📄	A cross-reference to another document related to the course.

Core Principles for Theming in Magento 2

Magento® U

Core Principles for Theming in Magento 2

Introduction to the Home Page



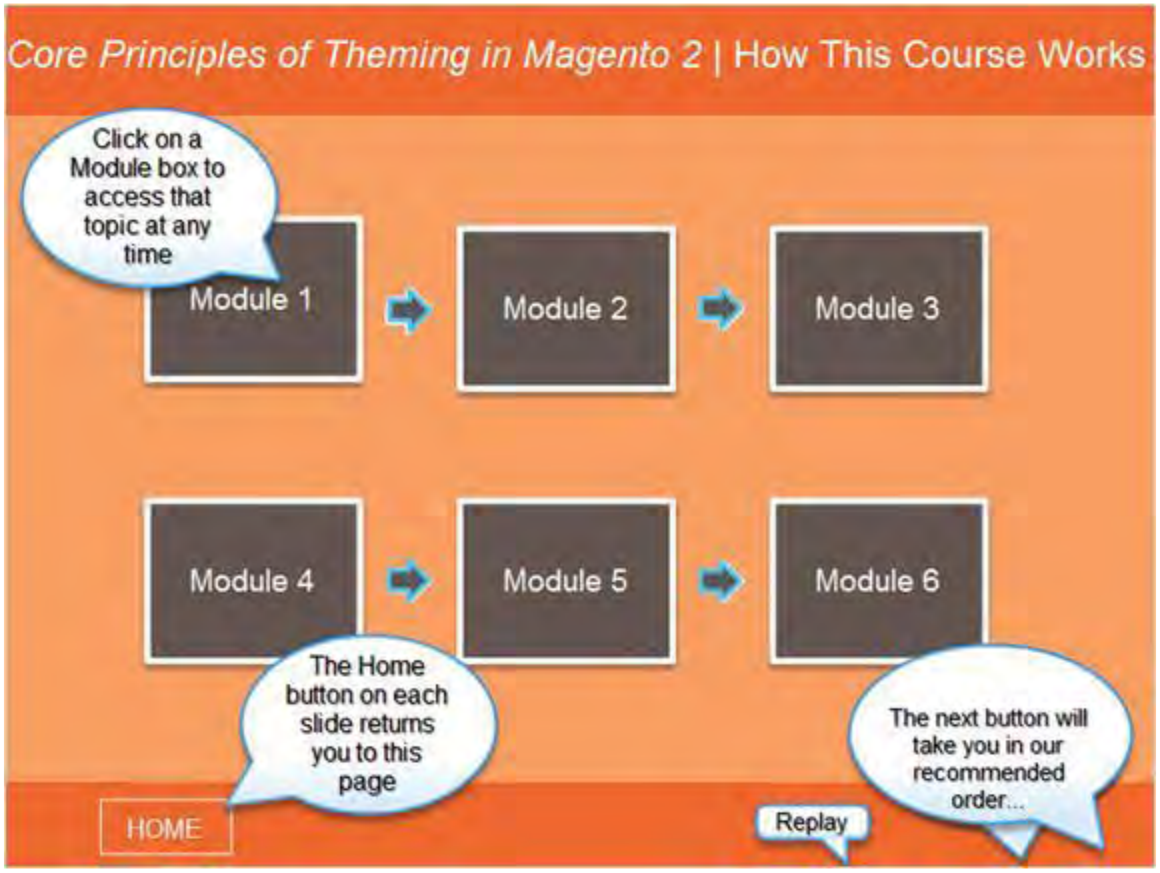
Core Principles for Theming in Magento 2 | Home Page

Notes:

Welcome to the Core Principles for Theming in Magento 2 course.

This course contains three units, each consisting of several modules. In Unit One, we will cover Module 0: Getting Started, Module 1: Overview of Theming, Module 2: Folder Structure, Module 3: About Static Files, Module 4: Deployment Process, and Module 5: Fallback Process. Unit Two covers Theme Architecture and Unit Three covers Layout XML, Blocks, and Templates.

How This Course Works


Core Principles of Theming in Magento 2 | How This Course Works
Click on a Module box to access that topic at any time
Module 1
Module 2
Module 3
Module 4
Module 5
Module 6
The Home button on each slide returns you to this page
The next button will take you in our recommended order...
HOME
Replay

Notes:

Each module can be accessed at any time by clicking on the appropriate box. The arrows indicate the proper sequence for taking the course the first time through. The Home button on each of the following course slides will return you to this page so you can replay a module or select a new one. The controls on the player allow you to replay a slide (backward curved arrow), proceed to the next slide, or return to the previous slide.

If you have any difficulty operating this course, or have any questions about the course, please email training@magento.com

Enjoy your course!

Introduction: Why Should I Take This Course?

Introduction

Why should I take this course?

To learn how to create and customize themes for Magento 2.

Magento U

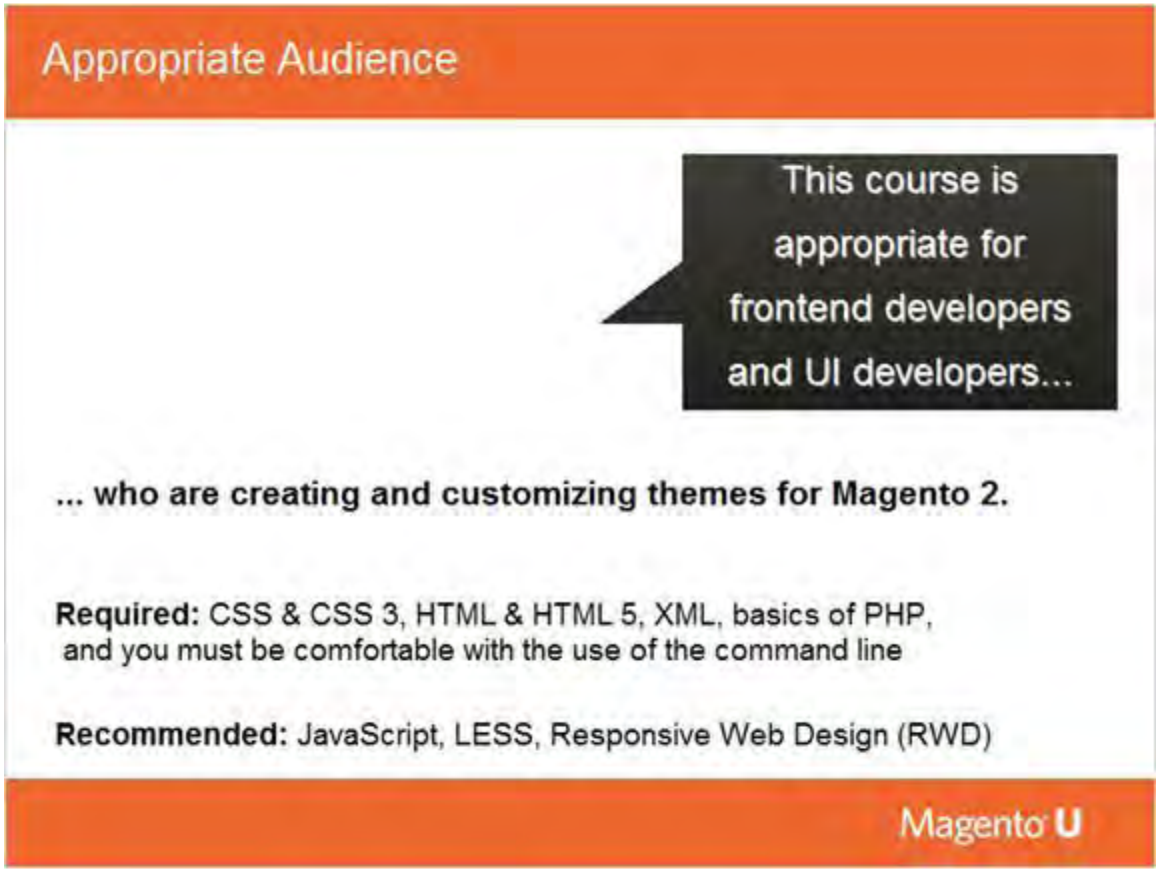
Notes:

This is an extensive, challenging course that presents essential concepts you need to learn about Magento 2 over a series of comprehensive units.

Be sure to give yourself enough time to go through the material at your own pace.

The built-in navigation allows you to repeat any slide or module as many times as you need, in any order you want.

Appropriate Audience



Appropriate Audience

This course is appropriate for frontend developers and UI developers...

... who are creating and customizing themes for Magento 2.

Required: CSS & CSS 3, HTML & HTML 5, XML, basics of PHP, and you must be comfortable with the use of the command line

Recommended: JavaScript, LESS, Responsive Web Design (RWD)

Magento U

Notes:

This course is appropriate for developers who are experienced with launching and extending the Magento 1 platform.

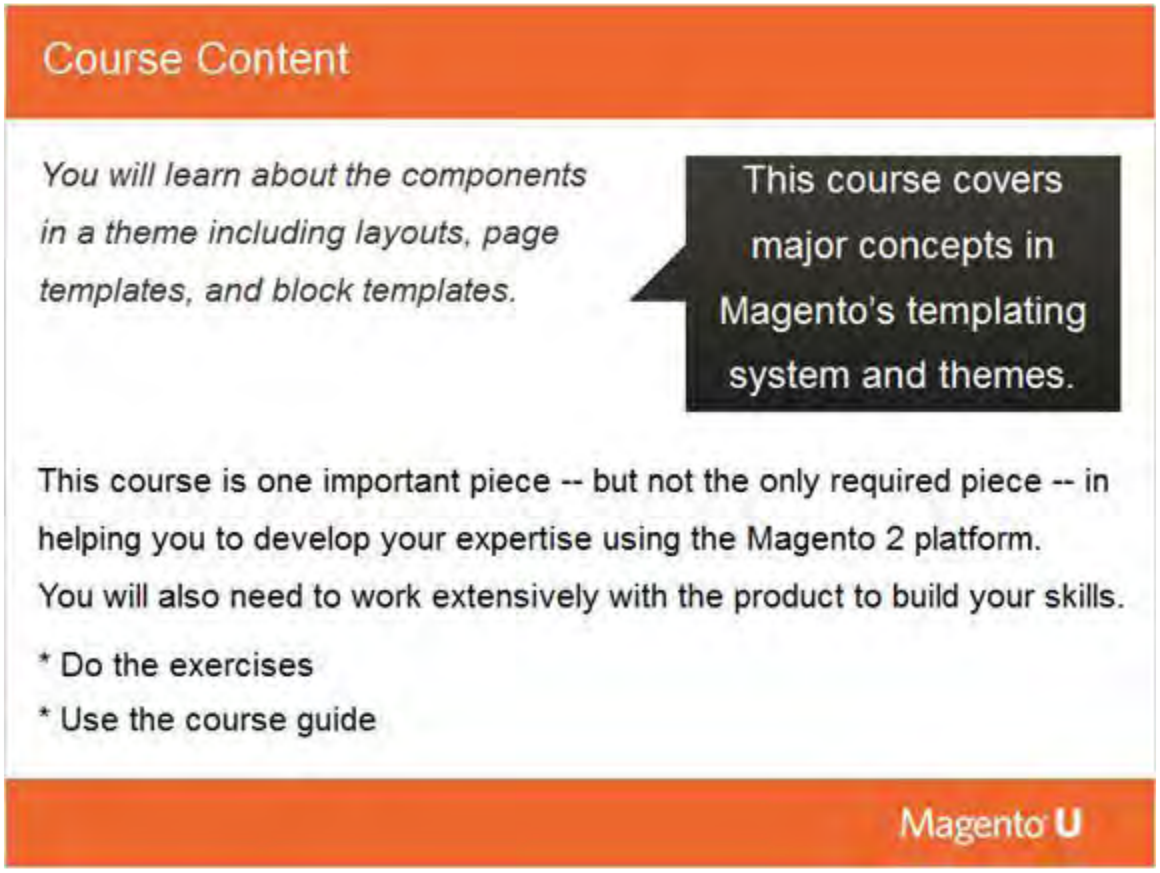
For this course, the following skills are required: CSS & CSS 3, HTML & HTML 5, XML, basics of PHP, and you must be comfortable with the use of the command line.

Knowledge of JavaScript and LESS are also helpful.

To enhance your understanding of this course, the following skills are desirable: Responsive Web Design (RWD), LESS, and JavaScript is strongly recommended but not required.

The depth of the content may prove challenging to new developers who have no familiarity with the Magento product.

Course Content



Course Content

You will learn about the components in a theme including layouts, page templates, and block templates.

This course covers major concepts in Magento's templating system and themes.

This course is one important piece -- but not the only required piece -- in helping you to develop your expertise using the Magento 2 platform.
You will also need to work extensively with the product to build your skills.

* Do the exercises
* Use the course guide

Magento U

Notes:

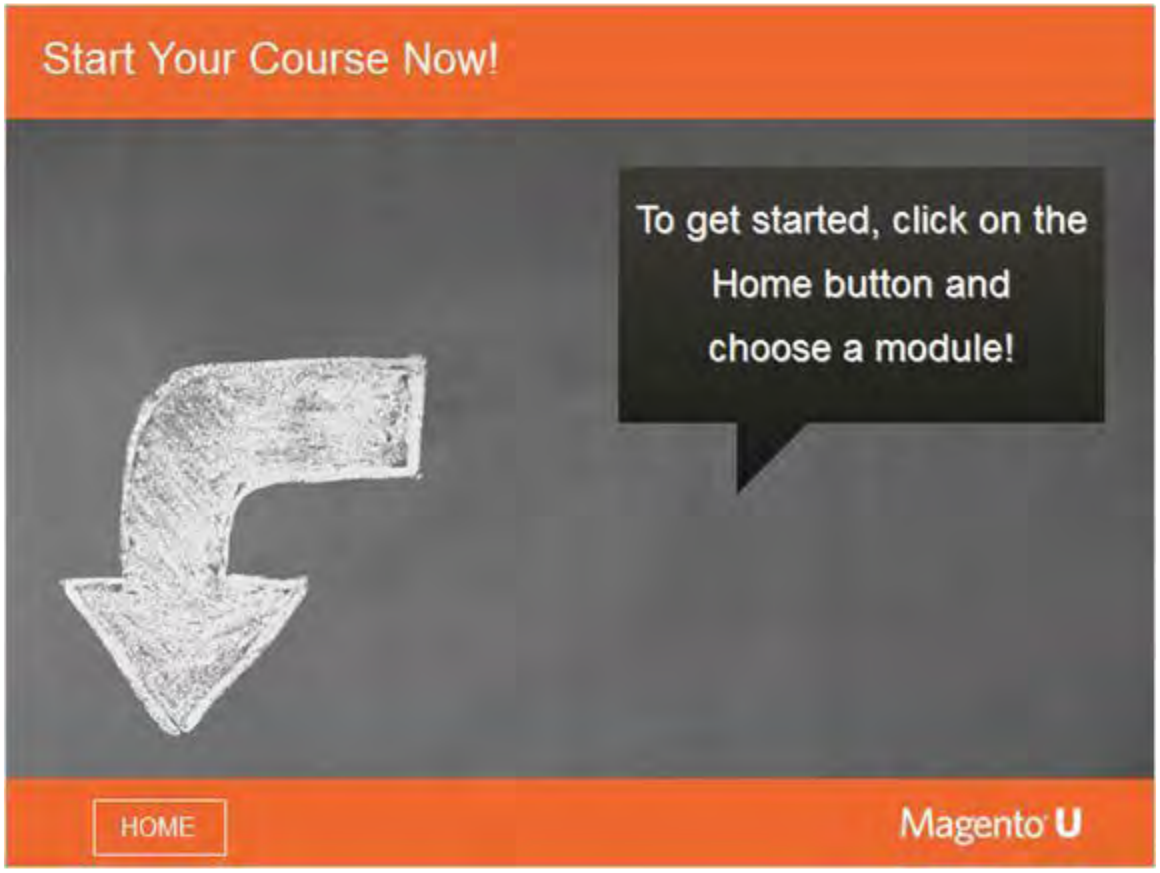
This course covers the fundamentals of Magento's templating system and how Magento themes work. You will learn about all the components of a theme, especially layouts, page templates, and block templates. You will learn how to customize both the look and feel and the functionality of a website at the theme level.

This course is one important piece -- but not the only required piece -- in helping you to develop your expertise using the Magento 2 platform. You will also need to work extensively with the product to build your skills.

Be sure to do all the exercises presented in the course, as these are key learning opportunities and provide practical, hands-on experience with the native Magento 2 installation. The course guide presents the narration in written form, while the slide highlights key concepts. Find the most effective way to use both to match your learning style. For example, you may want to read the slide first for a general grasp of the topic, then play the audio as you follow along with the full notes, to fill in all the details.

Remember, you can replay each slide as many times as needed to comprehend the material.

Start Your Course Now!



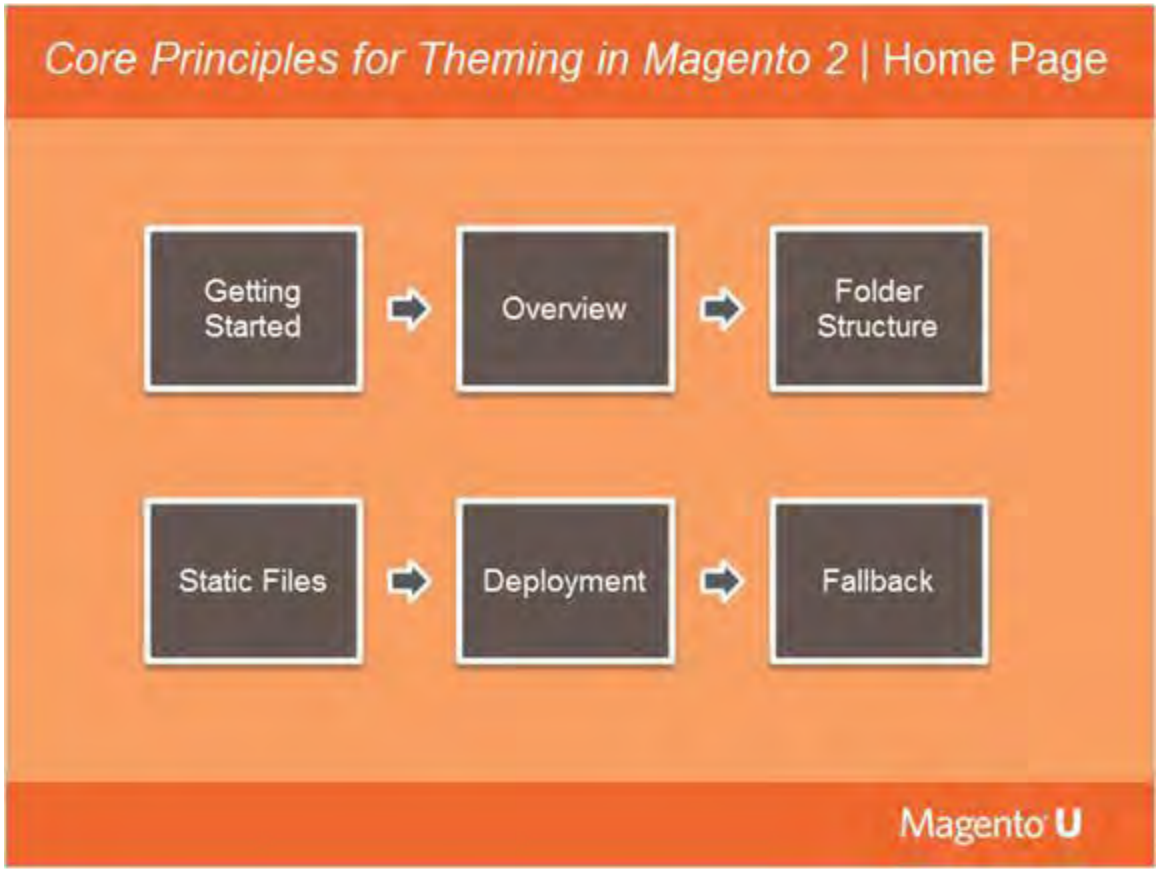
Start Your Course Now!

To get started, click on the Home button and choose a module!

Notes:

To get started with the course, click on the Home button now.

Unit One Home Page

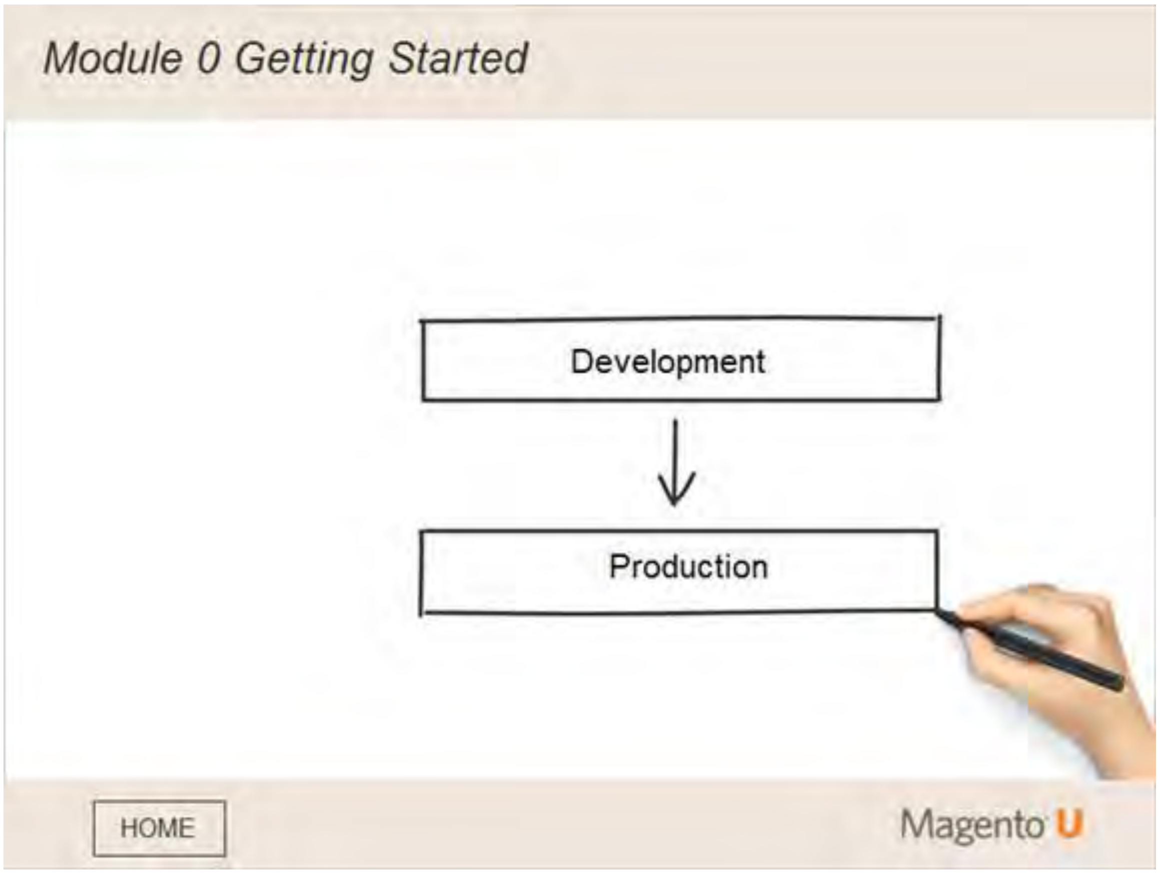

Core Principles for Theming in Magento 2 | Home Page

Notes:

There is no audio for this slide.

0. Module 0 Before We Get Started

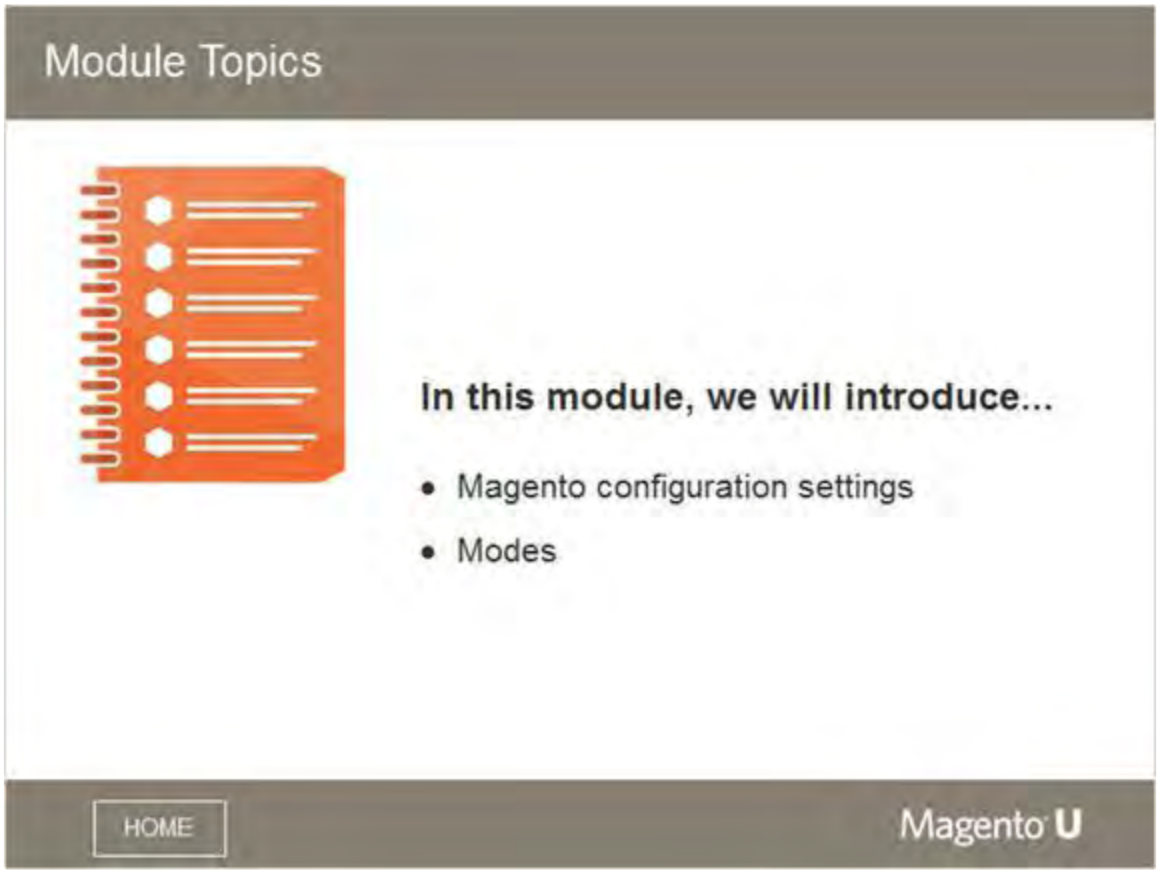
0.1 Module 0 Getting Started



Notes:

There are a few useful things to know about Magento before we dive in.

0.2 Module Topics



Module Topics

In this module, we will introduce...

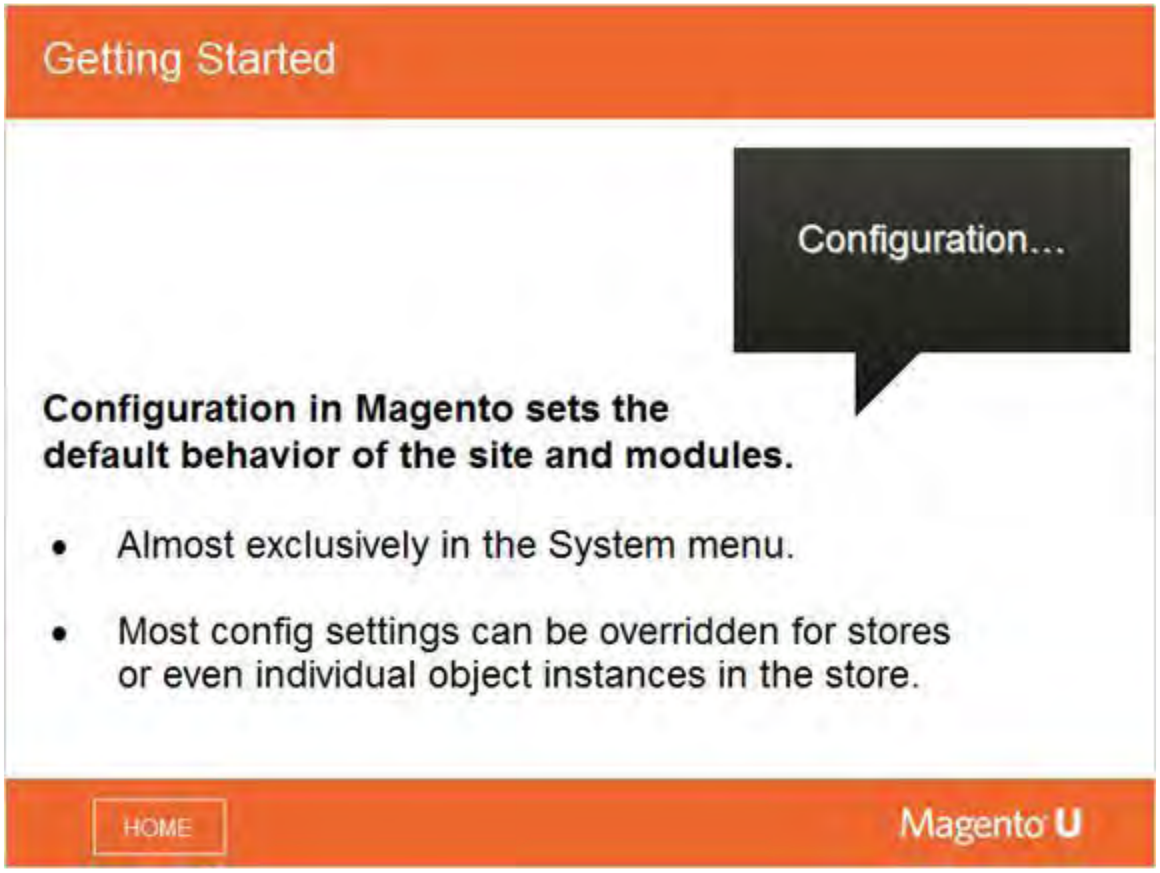
- Magento configuration settings
- Modes

HOME
Magento U

Notes:

In this module, we present Magento configuration settings and introduce modes.

0.3 Getting Started



Getting Started

Configuration…

Configuration in Magento sets the default behavior of the site and modules.

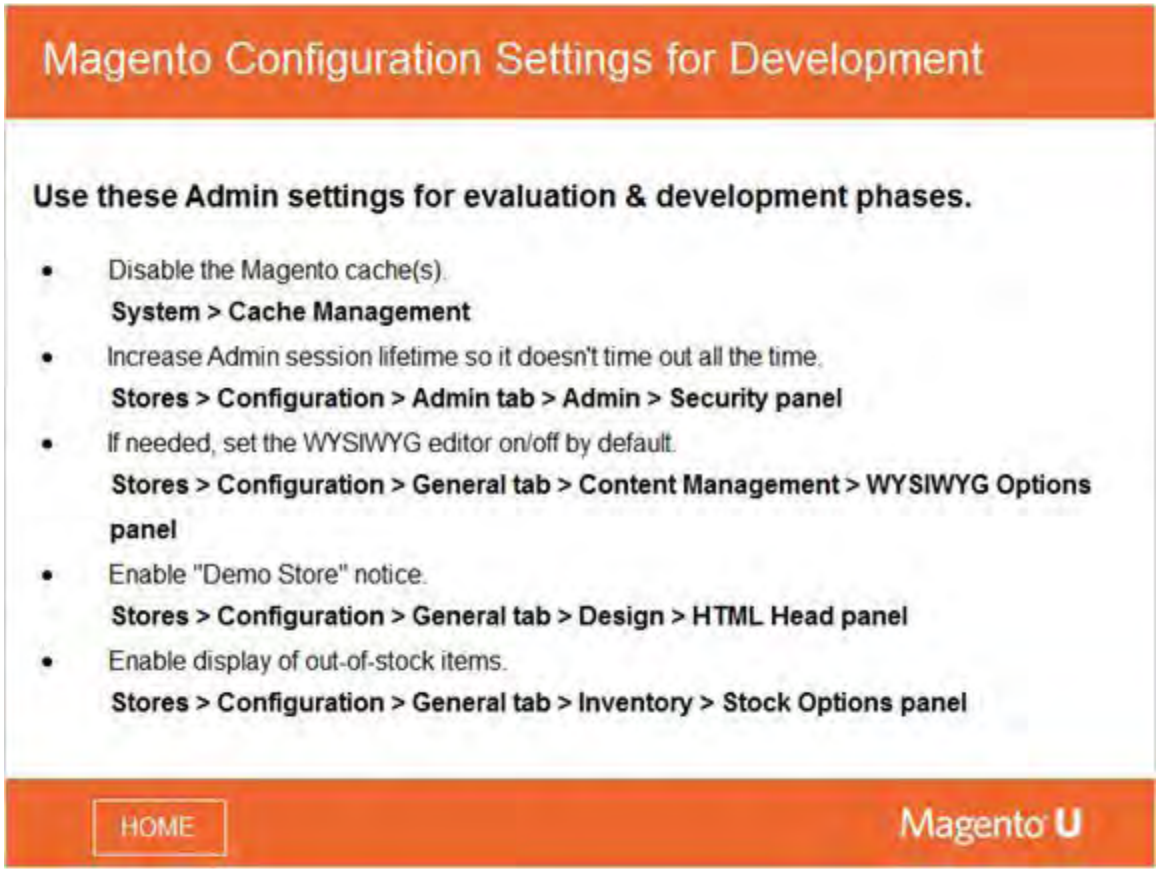
- Almost exclusively in the System menu.
- Most config settings can be overridden for stores or even individual object instances in the store.

HOME
Magento U

Notes:

First, let's talk about configuration. Configuration in Magento sets the default behavior of the site and modules. Config settings can be overridden for stores or object instances.

0.4 Magento Configuration Settings for Development



Magento Configuration Settings for Development

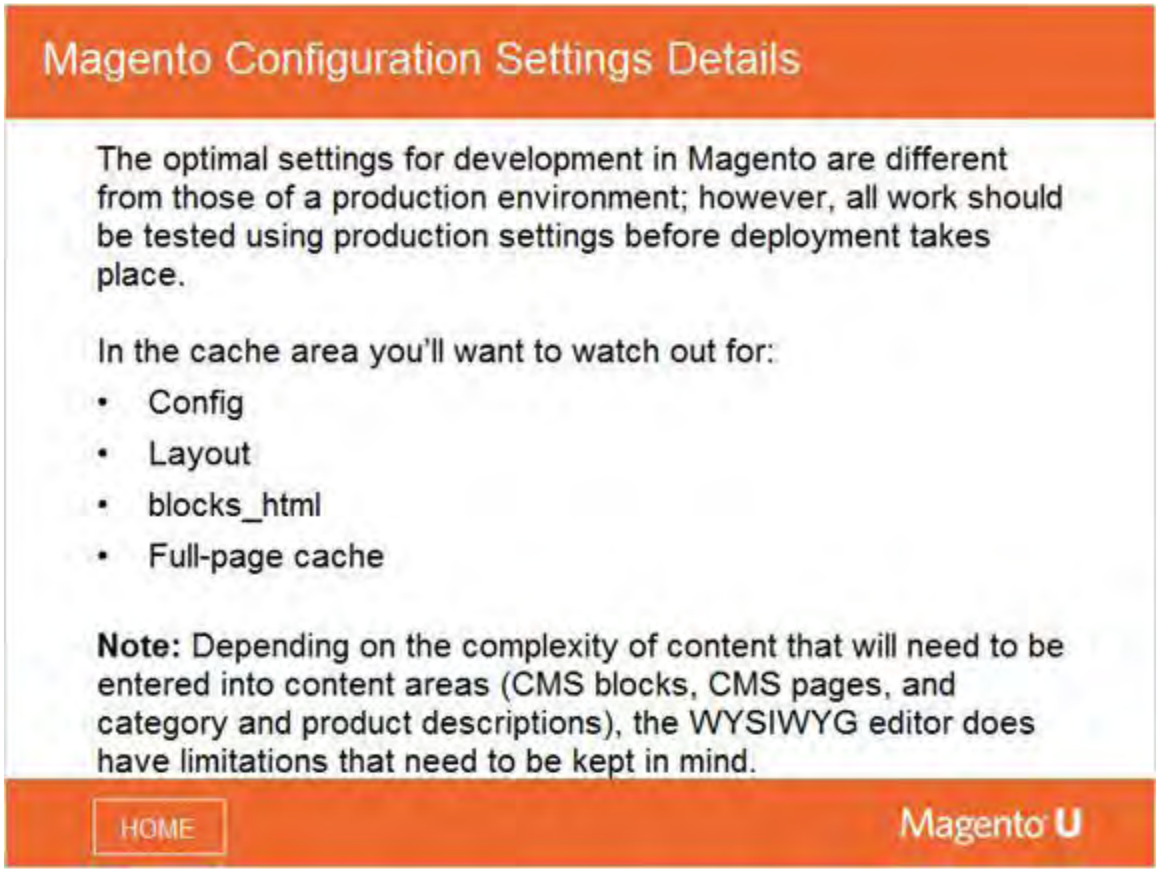
Use these Admin settings for evaluation & development phases.

- Disable the Magento cache(s).
  System > Cache Management
- Increase Admin session lifetime so it doesn't time out all the time.
  Stores > Configuration > Admin tab > Admin > Security panel
- If needed, set the WYSIWYG editor on/off by default.
  Stores > Configuration > General tab > Content Management > WYSIWYG Options panel
- Enable "Demo Store" notice.
  Stores > Configuration > General tab > Design > HTML Head panel
- Enable display of out-of-stock items.
  Stores > Configuration > General tab > Inventory > Stock Options panel

Notes:

Use these Admin settings for evaluation and development phases. You can refer back to this slide later. You will want to change these settings for production.

0.5 Magento Configuration Settings Details


Magento Configuration Settings Details

The optimal settings for development in Magento are different from those of a production environment; however, all work should be tested using production settings before deployment takes place.

In the cache area you'll want to watch out for:

- Config
- Layout
- blocks_html
- Full-page cache

Note: Depending on the complexity of content that will need to be entered into content areas (CMS blocks, CMS pages, and category and product descriptions), the WYSIWYG editor does have limitations that need to be kept in mind.

Notes:

Best Practice: The optimal settings for development in Magento are different from those of a production environment; however, all work should be tested using production settings before deployment takes place.

In the cache area you'll want to watch out for:

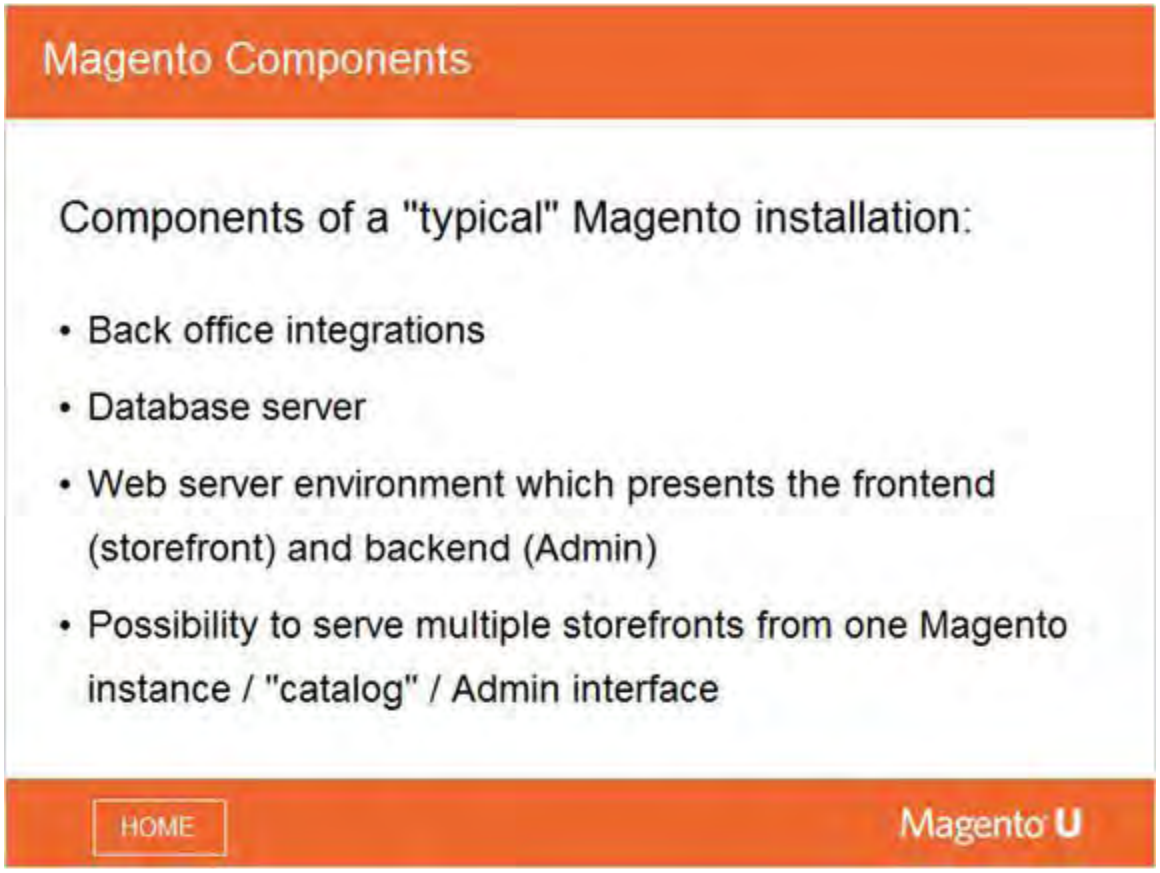
- Config
- Layout
- blocks_html
- Full-page cache

You can turn on Developer mode using the .htaccess file:

```
SetEnv MAGE_MODE developer
php_flag display_errors 1
```

Depending on the complexity of content that will need to be entered into content areas (CMS blocks, CMS pages, and category and product descriptions), the WYSIWYG editor does have limitations that need to be kept in mind.

0.6 Magento Components



Magento Components

Components of a "typical" Magento installation:

- Back office integrations
- Database server
- Web server environment which presents the frontend (storefront) and backend (Admin)
- Possibility to serve multiple storefronts from one Magento instance / "catalog" / Admin interface

HOME
Magento U

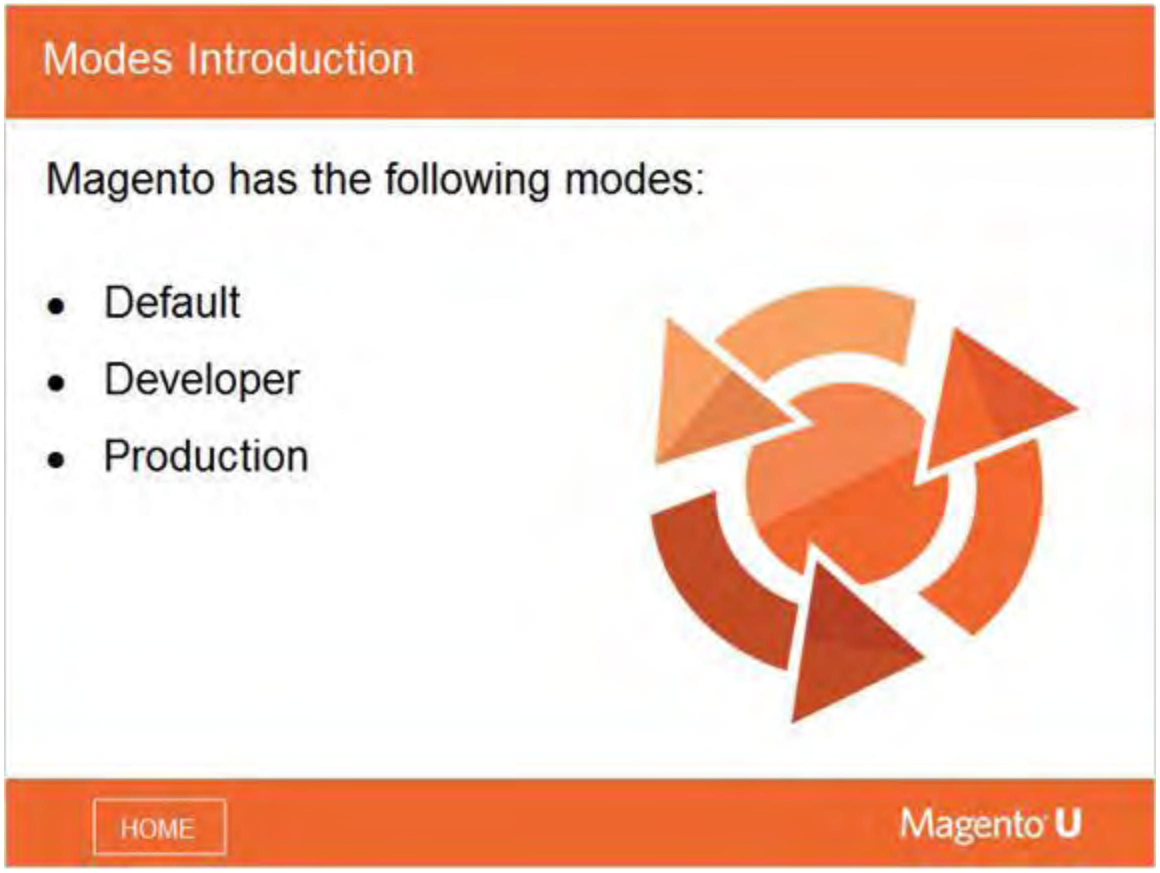
Notes:

Here is a quick review.

Components of a "typical" Magento installation:

- Back office integrations
- Database server
- Web server environment which presents the frontend (storefront) and backend (Admin)
- Possibility to serve multiple storefronts from one Magento instance / "catalog" / Admin interface

0.7 Modes Introduction



Modes Introduction

Magento has the following modes:

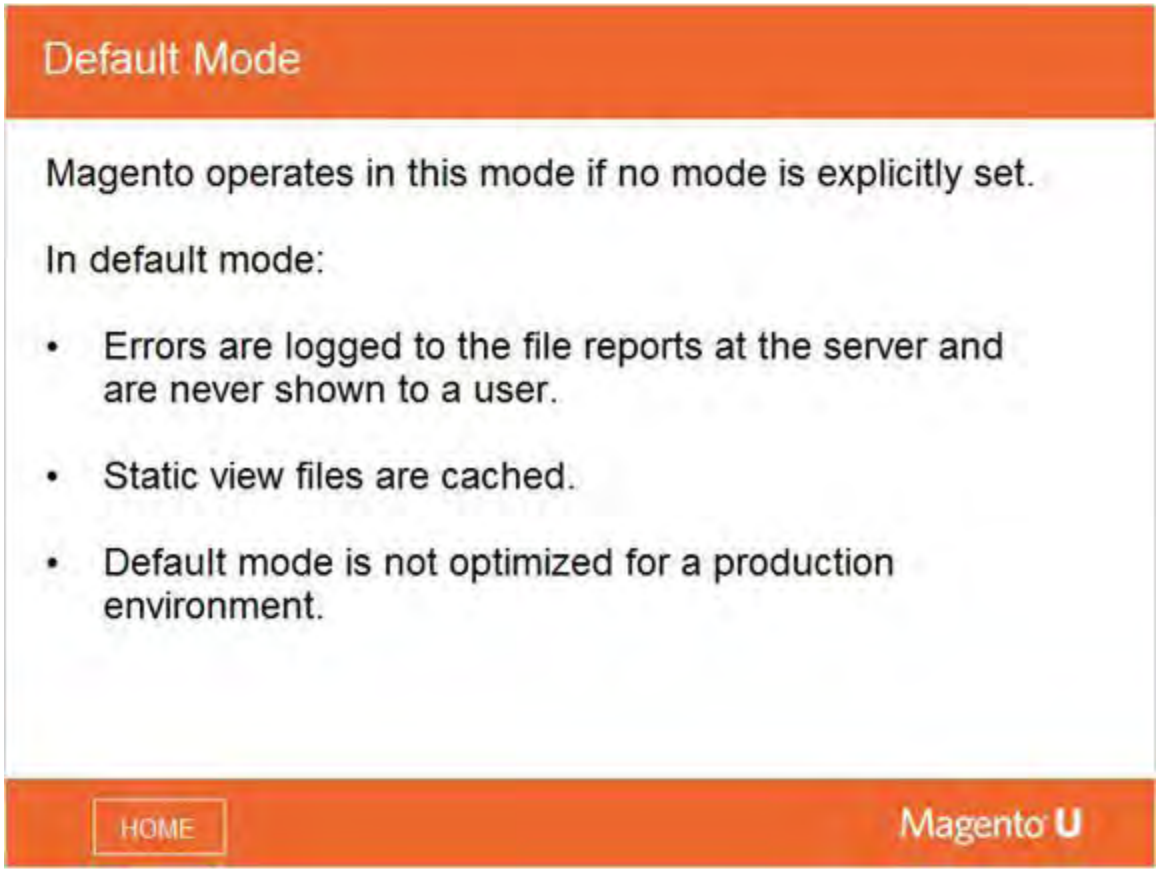
- Default
- Developer
- Production

HOME
Magento U

Notes:

Magento has three modes used for development and production. Each mode is optimized for its purpose.

0.8 Default Mode



Default Mode

Magento operates in this mode if no mode is explicitly set.

In default mode:

- Errors are logged to the file reports at the server and are never shown to a user.
- Static view files are cached.
- Default mode is not optimized for a production environment.

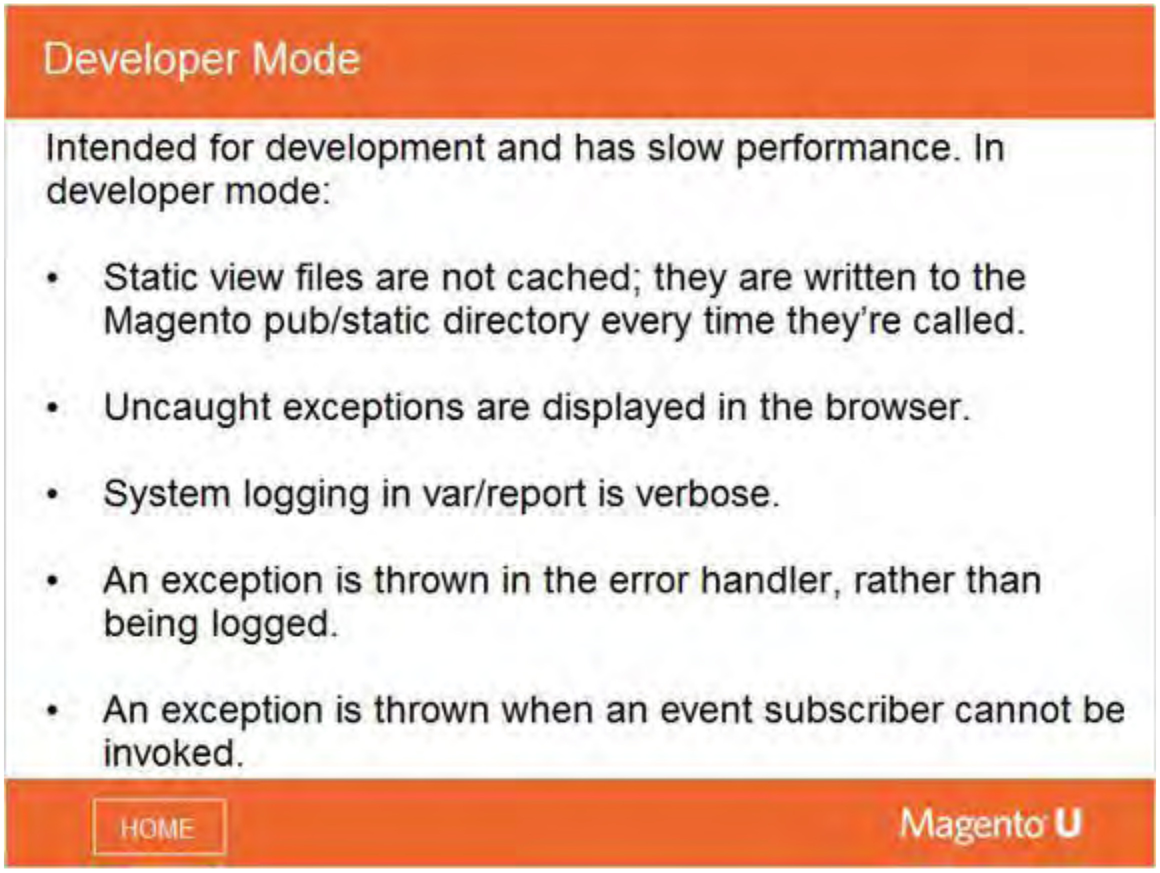
HOME
Magento U

Notes:

Default mode is not optimized for a production environment, primarily because of the adverse performance impact of static files being cached rather than materialized.

In other words, creating static files and caching them has a greater performance impact than generating them using the static files creation tool.

0.9 Developer Mode



Developer Mode

Intended for development and has slow performance. In developer mode:

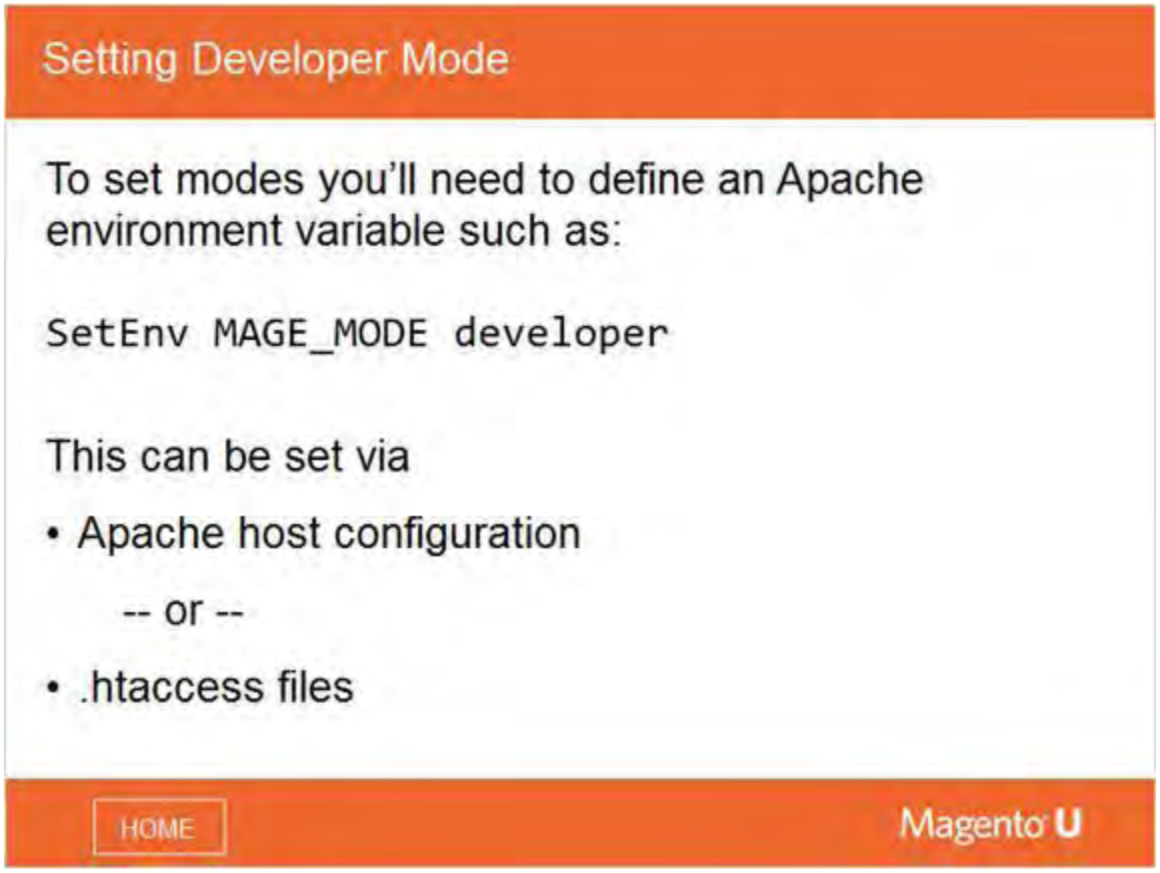
- Static view files are not cached; they are written to the Magento pub/static directory every time they're called.
- Uncaught exceptions are displayed in the browser.
- System logging in var/report is verbose.
- An exception is thrown in the error handler, rather than being logged.
- An exception is thrown when an event subscriber cannot be invoked.

Notes:

Developer mode provides more logging, and exceptions are captured to assist the developer while the site is being developed.

No performance optimization is done.

0.10 Setting Developer Mode



Setting Developer Mode

To set modes you'll need to define an Apache environment variable such as:

```
SetEnv MAGE_MODE developer
```

This can be set via

- Apache host configuration

  -- or --

- .htaccess files

HOME
Magento U

Notes:

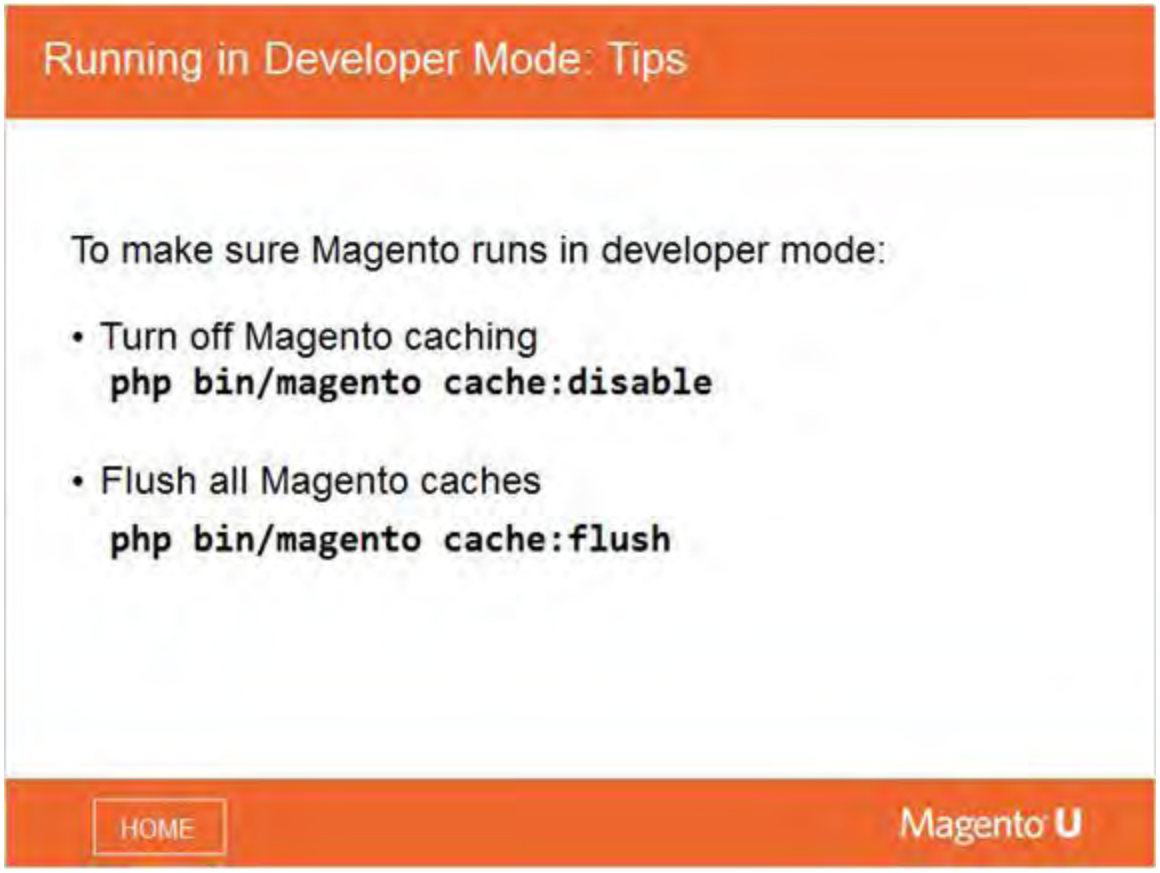
To set the developer mode you'll need to define an Apache environment variable such as:

```
SetEnv MAGE_MODE developer
```

This can be set via:

- Apache host configuration
  - or -
- .htaccess files

0.11 Running in Developer Mode: Tips



Running in Developer Mode: Tips

To make sure Magento runs in developer mode:

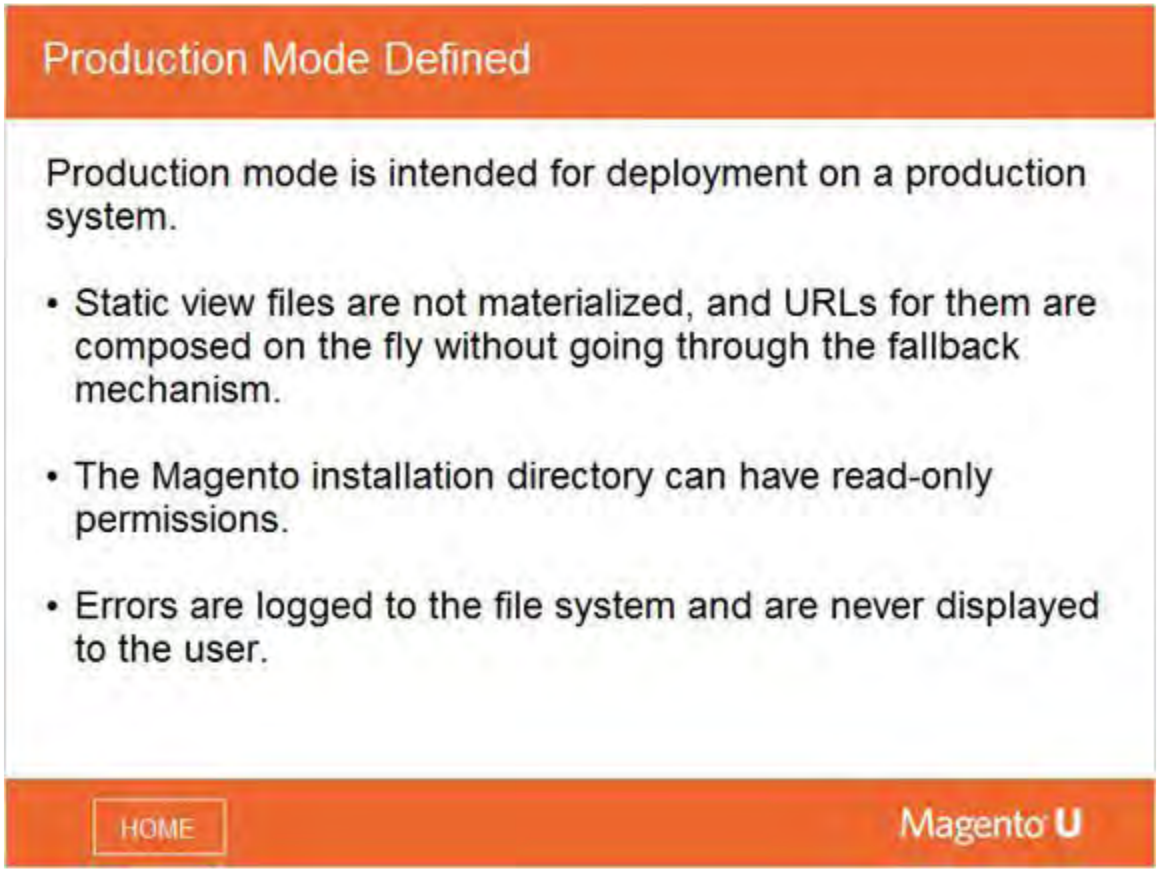
- Turn off Magento caching
```
php bin/magento cache:disable
```
- Flush all Magento caches
```
php bin/magento cache:flush
```

HOME
Magento U

Notes:

Earlier in this course, you saw more detail about developer mode.

0.12 Production Mode Defined



Production Mode Defined

Production mode is intended for deployment on a production system.

- Static view files are not materialized, and URLs for them are composed on the fly without going through the fallback mechanism.
- The Magento installation directory can have read-only permissions.
- Errors are logged to the file system and are never displayed to the user.

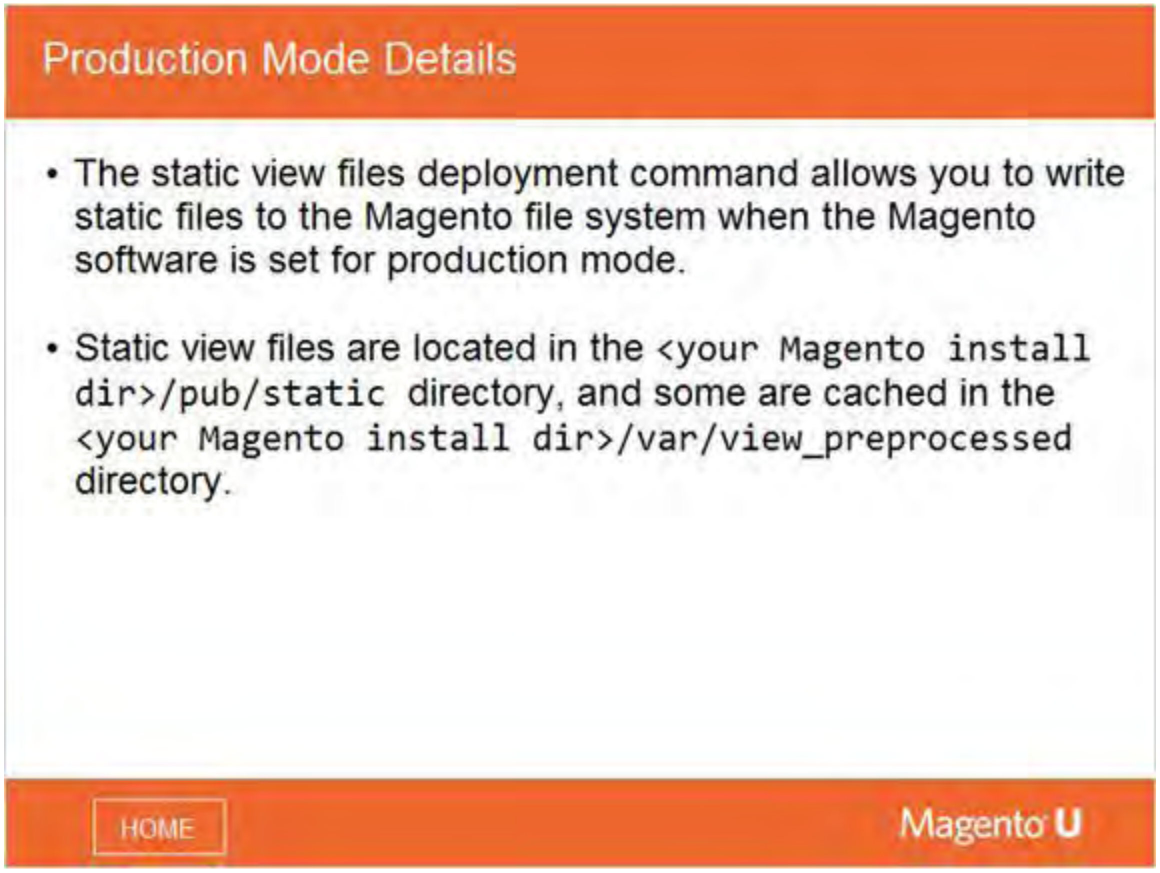
HOME
Magento U

Notes:

Production mode is optimized for performance in a few ways.

- Static view files are not materialized, and URLs for them are composed on the fly without going through the fallback mechanism.
- The Magento installation directory can have read-only permissions.
- Errors are logged to the file system and are never displayed to the user.

0.13 Production Mode Details



Production Mode Details

- The static view files deployment command allows you to write static files to the Magento file system when the Magento software is set for production mode.
- Static view files are located in the `<your Magento install dir>/pub/static` directory, and some are cached in the `<your Magento install dir>/var/view_preprocessed` directory.

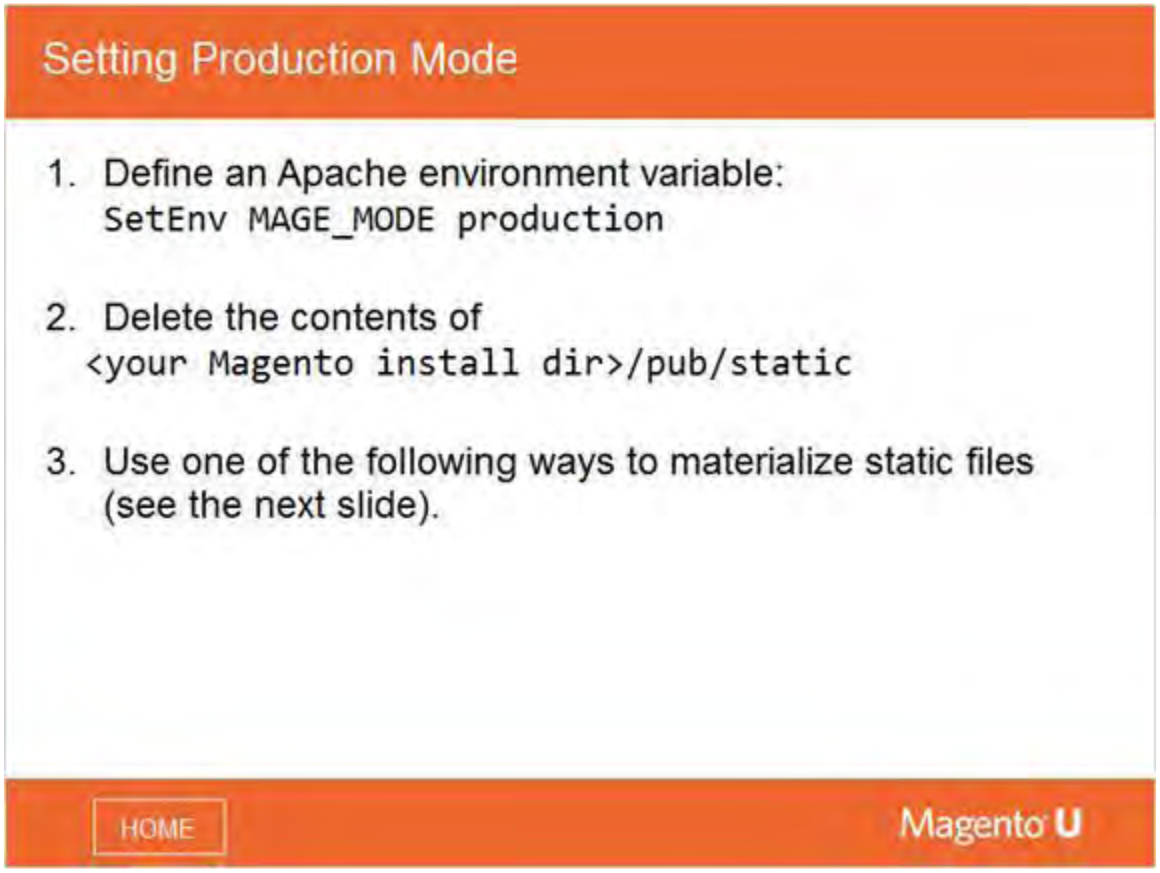
HOME
Magento U

Notes:

The static view files deployment command allows you to write static files to the Magento file system when the Magento software is set for production mode.

Static view files are located in the <your Magento install dir>/pub/static directory, and some are cached in the <your Magento install dir>/var/view_preprocessed directory.

0.14 Setting Production Mode



Setting Production Mode

1. Define an Apache environment variable:
   `SetEnv MAGE_MODE production`
2. Delete the contents of
   `<your Magento install dir>/pub/static`
3. Use one of the following ways to materialize static files (see the next slide).

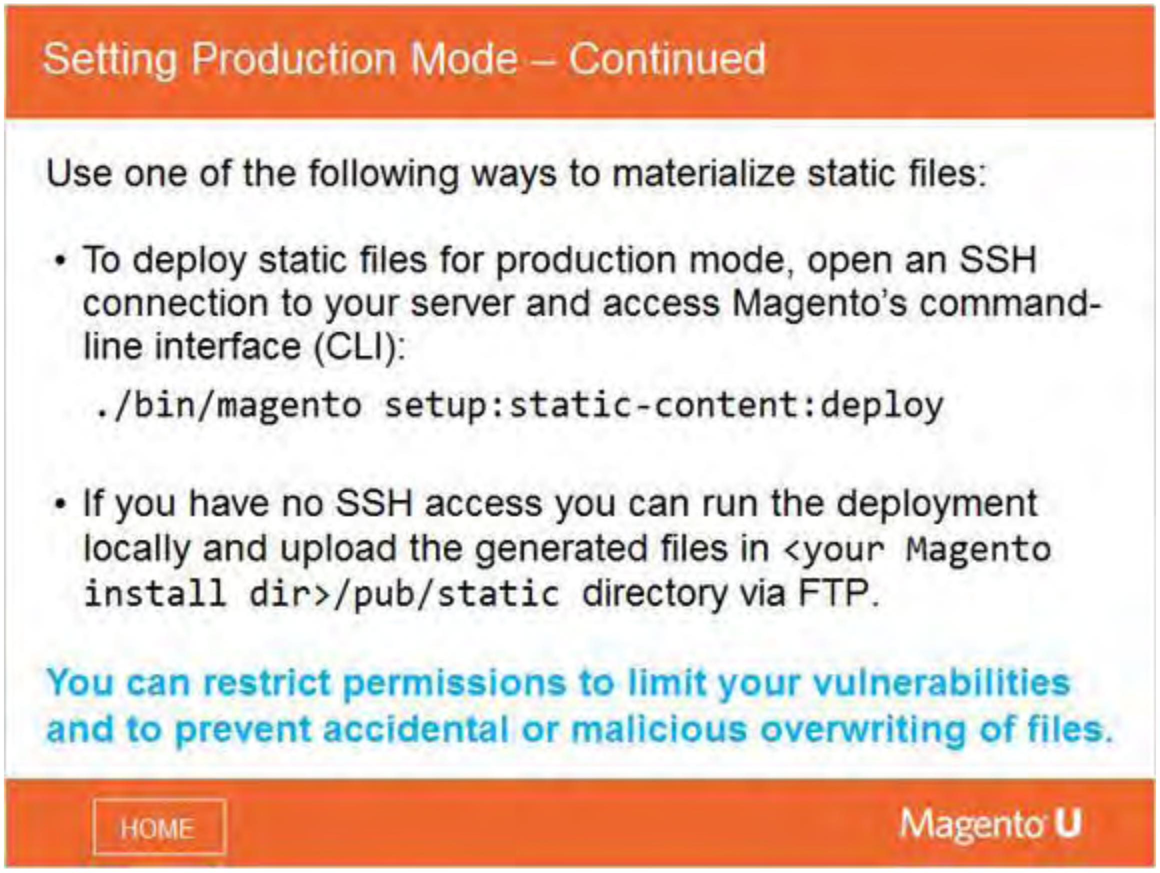
HOME
Magento U

Notes:

To set production mode:

1. Define an apache environment variable:
   `SetEnv MAGE_MODE production`
2. Delete the contents of <your Magento install dir>/pub/static
3. Finally, you'll need to materialize static files.

0.15 Setting Production Mode - Continued



Setting Production Mode – Continued

Use one of the following ways to materialize static files:

- To deploy static files for production mode, open an SSH connection to your server and access Magento's command-line interface (CLI):

```
./bin/magento setup:static-content:deploy
```

- If you have no SSH access you can run the deployment locally and upload the generated files in `<your Magento install dir>/pub/static` directory via FTP.

You can restrict permissions to limit your vulnerabilities and to prevent accidental or malicious overwriting of files.

Notes:

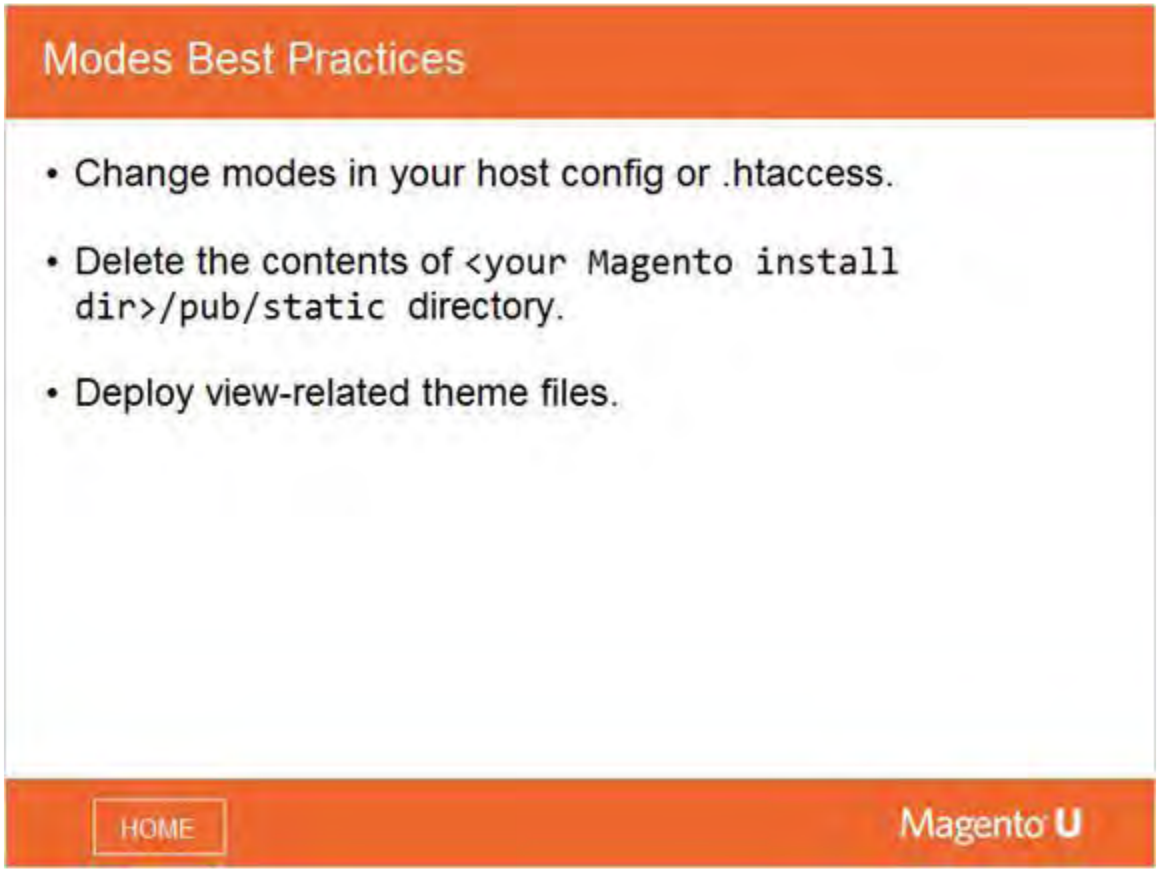
To deploy static files, do one of the following:

Open an SSH connection to your server and access Magento's command-line interface (CLI):
```
./bin/magento setup:static-content:deploy
```

If you have no SSH access you can run the deployment locally and upload the generated files in <your Magento install dir>/pub/static directory via FTP.

You can restrict permissions to limit your vulnerabilities and to prevent accidental or malicious overwriting of files.

0.16 Modes Best Practices



Modes Best Practices

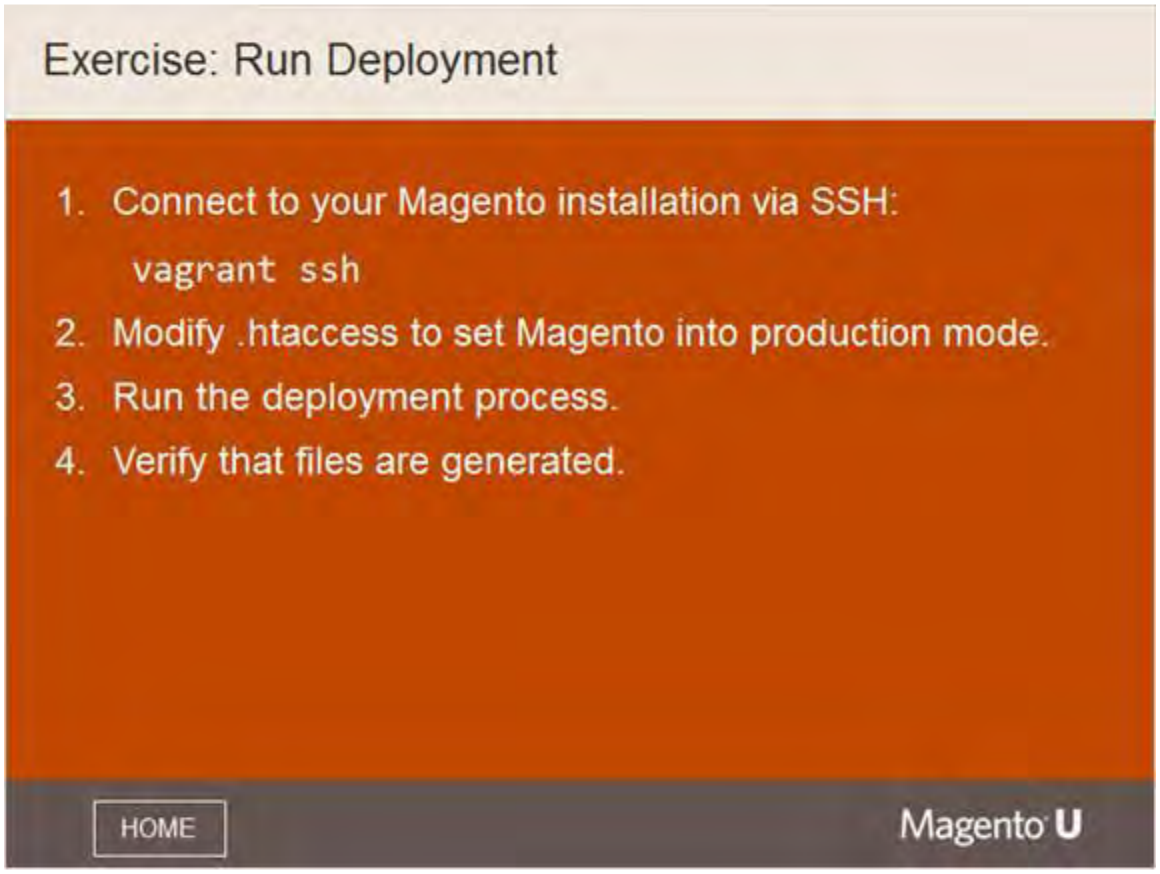
- Change modes in your host config or .htaccess.
- Delete the contents of `<your Magento install dir>/pub/static` directory.
- Deploy view-related theme files.

HOME
Magento U

Notes:

Keep these best practices in mind throughout the course.

0.17 Exercise: Run Deployment



Exercise: Run Deployment

1. Connect to your Magento installation via SSH:

```
vagrant ssh
```

2. Modify .htaccess to set Magento into production mode.
3. Run the deployment process.
4. Verify that files are generated.

HOME
Magento U

Notes:

Follow the exercise instructions shown on the slide.

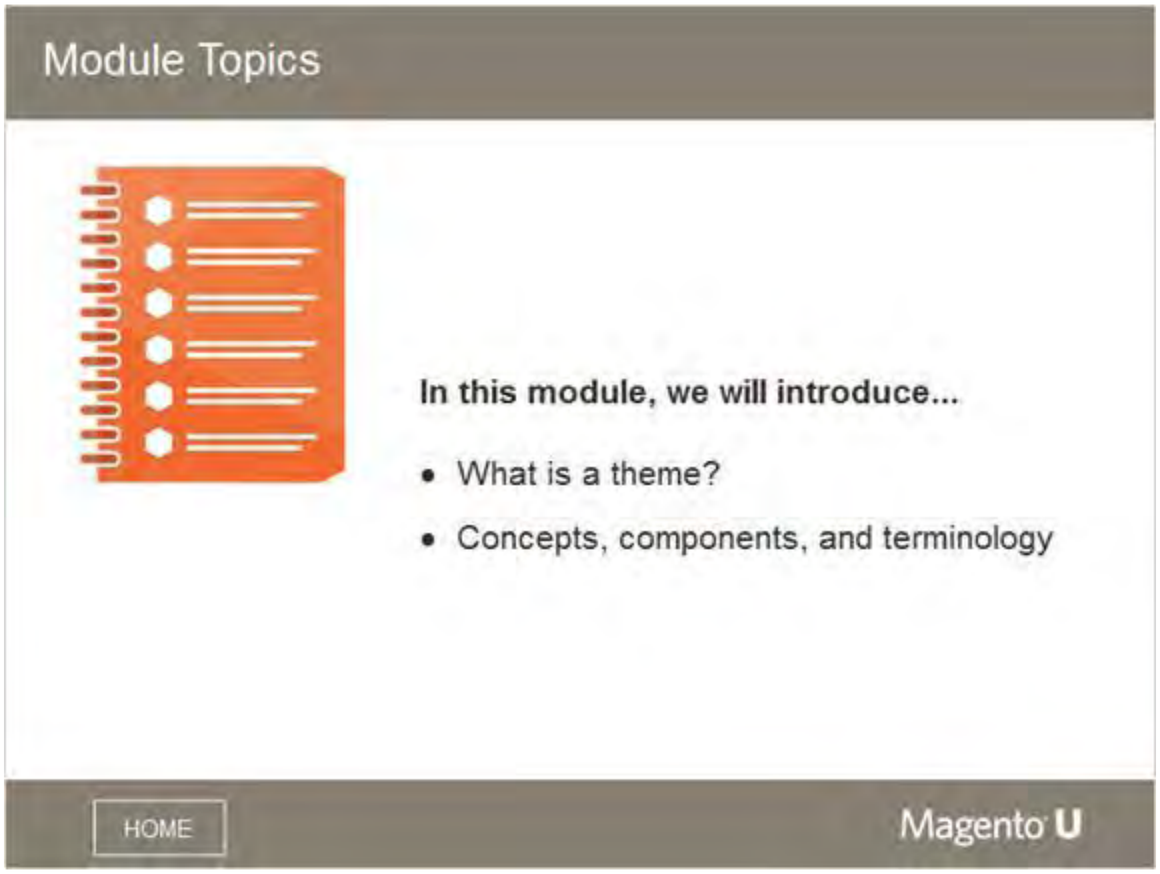
1. Module 1 Overview

1.1 Module 1 Overview of Theming

Notes:

We will now discuss an overview of theming. You will learn about a typical Magento installation, some key terminology, and where to locate various types of files.

1.2 Module Topics



Module Topics

In this module, we will introduce...

- What is a theme?
- Concepts, components, and terminology

HOME
Magento U

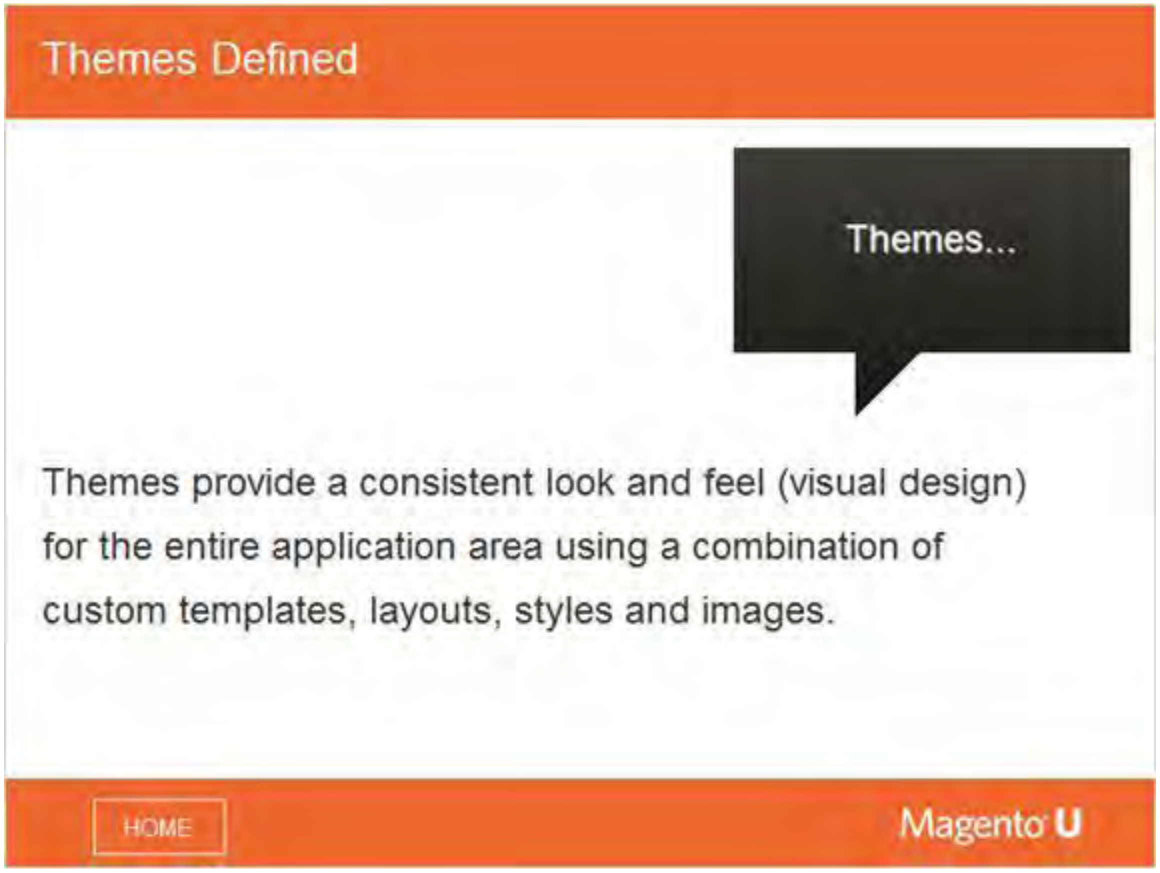
Notes:

In this module, we present an overview of the rendering system of Magento 2, focusing on folder structure, static files, deployment process and understanding fallback.

Each of these aspects will be taught first in isolation, to make important concepts easier to understand.

Then, the course will address how all the pieces fit together, using best practices.

1.3 Themes Defined



Themes Defined

Themes...

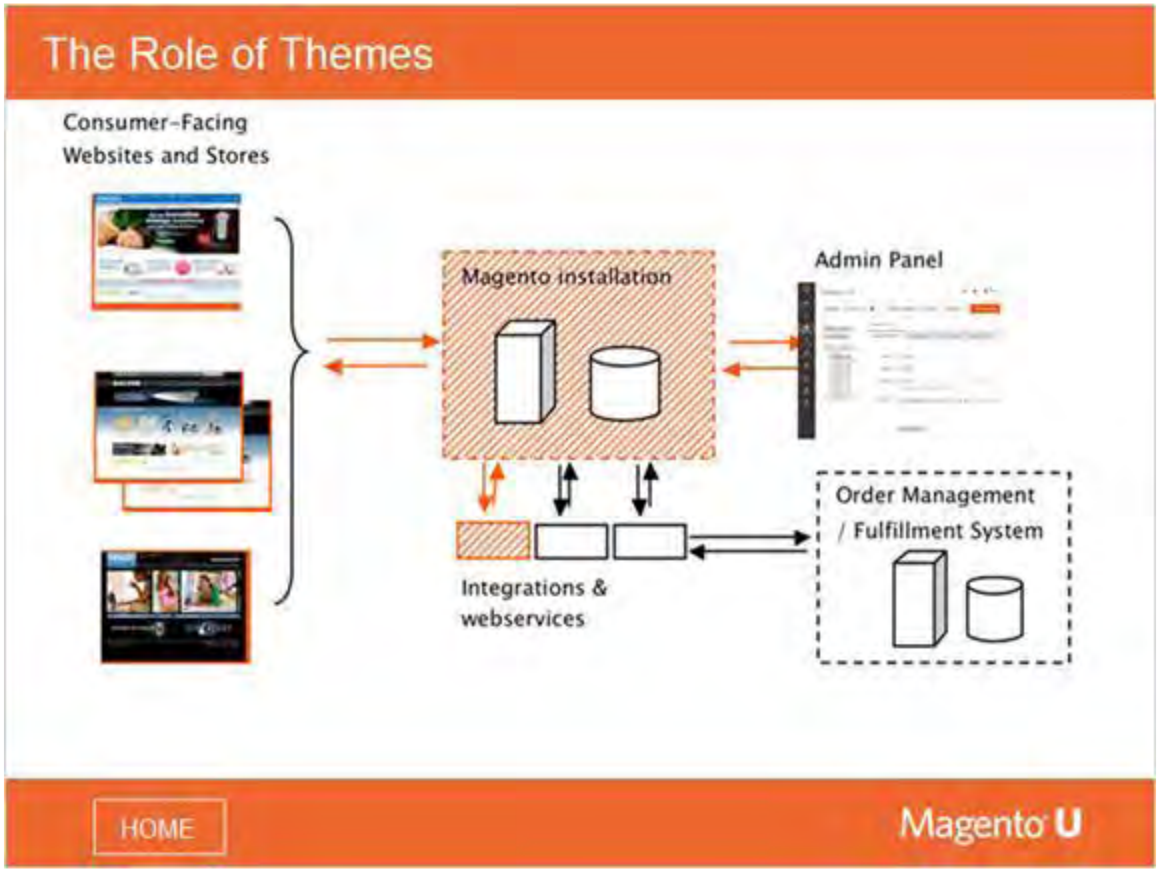
Themes provide a consistent look and feel (visual design) for the entire application area using a combination of custom templates, layouts, styles and images.

HOME
Magento U

Notes:

A theme is a component of a Magento application which provides a consistent look and feel (visual design) for the entire application area (for example, storefront or Magento Admin) using a combination of custom templates, layouts, styles or images.

1.4 The Role of Themes


The Role of Themes
Consumer-Facing Websites and Stores
Magento installation
Admin Panel
Order Management / Fulfillment System
Integrations & webservices
HOME
Magento U

Notes:

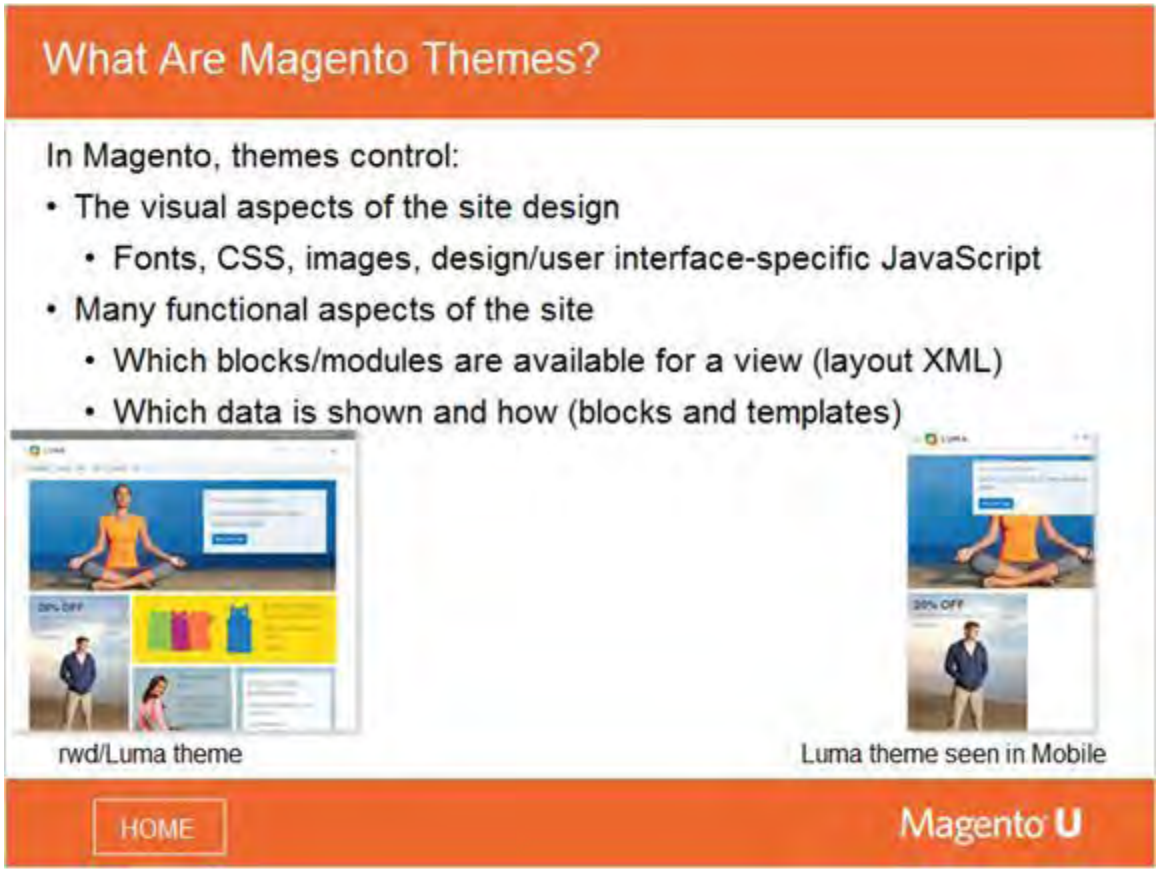
Themes are designed to override or customize view layer resources, provided initially by modules or libraries. Themes are implemented by different vendors (frontend developers) and intended to be distributed as additional packages for a Magento system similar to other components.

Components of a "typical" Magento installation include:

- Back office integrations
- Database server
- Web server environment, which presents the frontend (storefront) and backend (Admin)
- The possibility to serve multiple storefronts from one Magento instance / "catalog" / Admin interface

Any perception of complexity in Magento's theming structure should be balanced against the functional benefit that the system provides.Through Magento's theming engine the frontend developer has the option of discretely sharing or customizing design elements across multiple storefronts.

1.5 What Are Magento Themes?



What Are Magento Themes?

In Magento, themes control:

- The visual aspects of the site design
  - Fonts, CSS, images, design/user interface-specific JavaScript
- Many functional aspects of the site
  - Which blocks/modules are available for a view (layout XML)
  - Which data is shown and how (blocks and templates)

rwd/Luma theme

Luma theme seen in Mobile

HOME
Magento U

Notes:

A theme is a component of a Magento application that provides a consistent look and feel for the entire application area (for example, storefront or Magento Admin) using a combination of fonts, custom templates, layouts, styles or images. **Remember that the theming system is part of the larger whole of the framework.**

In addition to controlling the visual and function aspects of a site, themes allow the presentation layer to be independent of business logic and functionality.

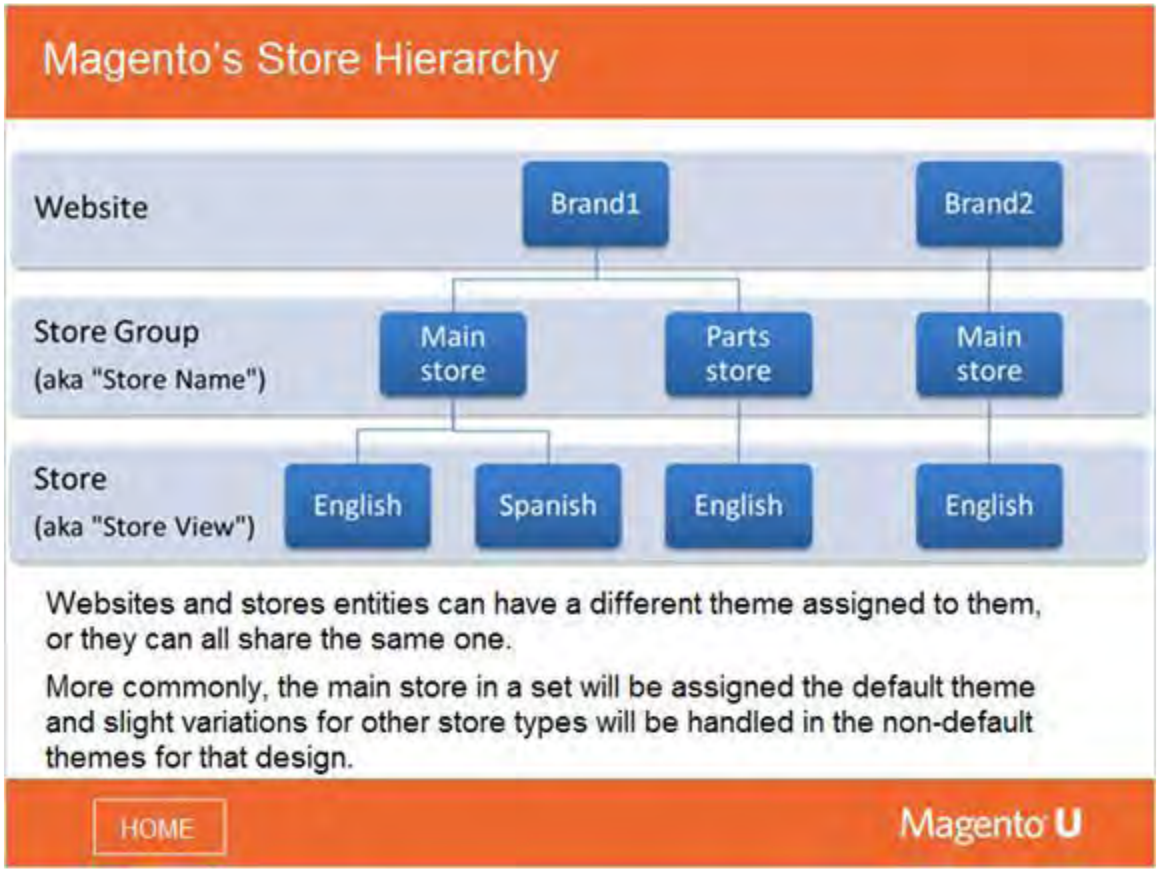
In Magento, themes govern presentation in two ways:

- Non-informational/non-output (via web, including CSS, images, and visual-specific JavaScript)
- The presence, format, and position of output (via layout XML, block classes, and templates)

We can relate the tasks that frontend developers do to the Magento theming components that are responsible:

- **Appearance-only:** Some implementations and changes can be entirely appearance-related.
  Examples: Changing background images, element dimensions, font style and colors, etc.
- **Layout XML-only:** Some implementations may involve adding or removing output from one or more views.
  Examples: Adding, removing, and arranging blocks in a sidebar; removing a block from output; etc.
- **Block-only:** Some implementations may involve arranging output within a given area.
  Examples: Changing markup, testing data states to vary output, arranging child block output in a template
- Most implementations will involve more than one of the above.
- **Above all, there are often multiple ways to implement something.**

1.6 Magento's Store Hierarchy



Websites and stores entities can have a different theme assigned to them, or they can all share the same one.

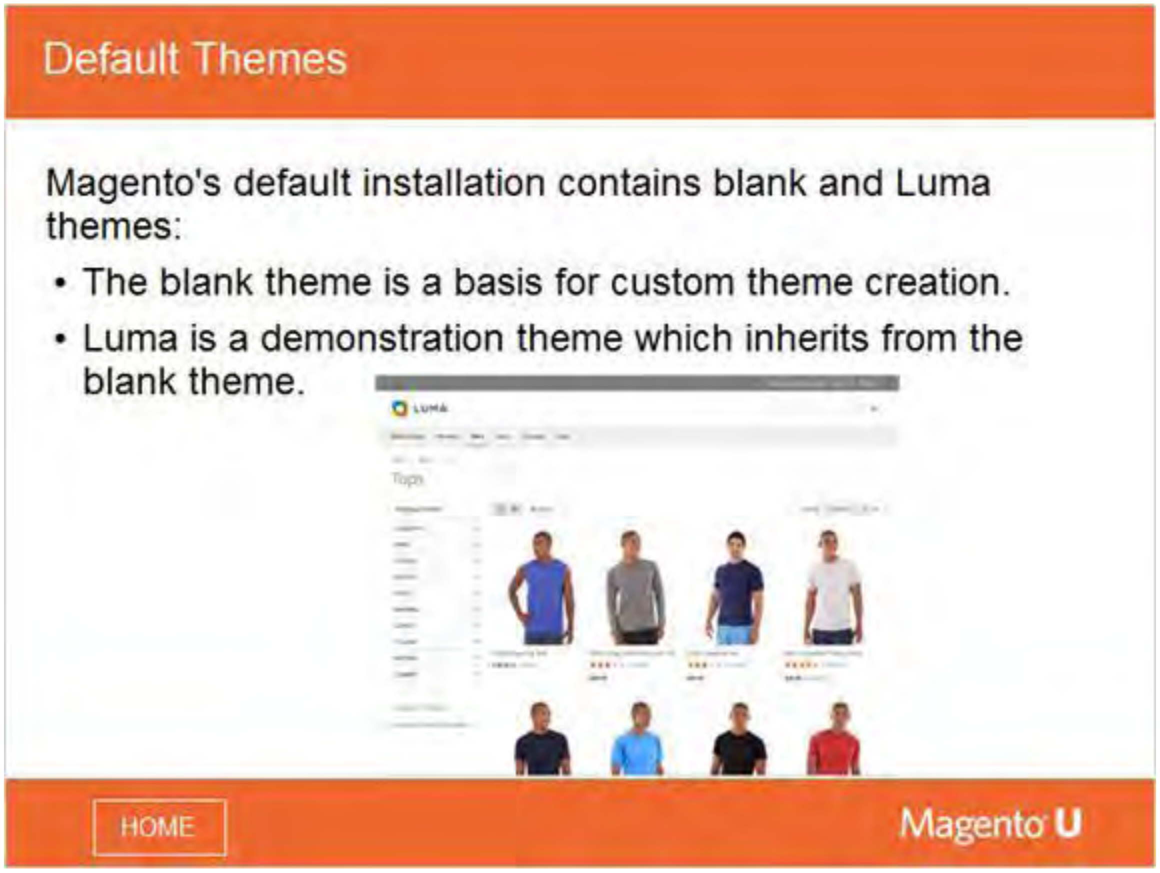
More commonly, the main store in a set will be assigned the default theme and slight variations for other store types will be handled in the non-default themes for that design.

Notes:

Once you understand how stores and themes work together, you can understand how to create themes appropriate to your store structure.

The main point to understand from this slide is the different configuration scope objects and their relationship to each other.

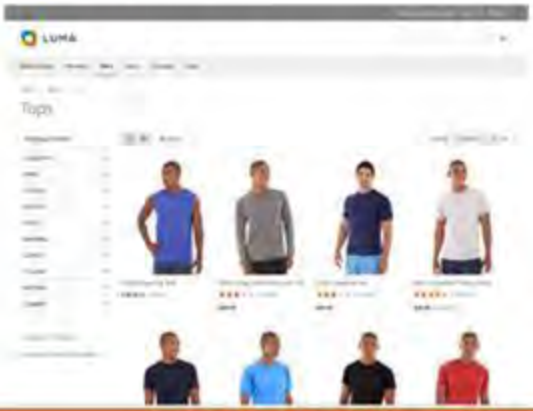
1.7 Default Themes



Default Themes

Magento's default installation contains blank and Luma themes:

- The blank theme is a basis for custom theme creation.
- Luma is a demonstration theme which inherits from the blank theme.



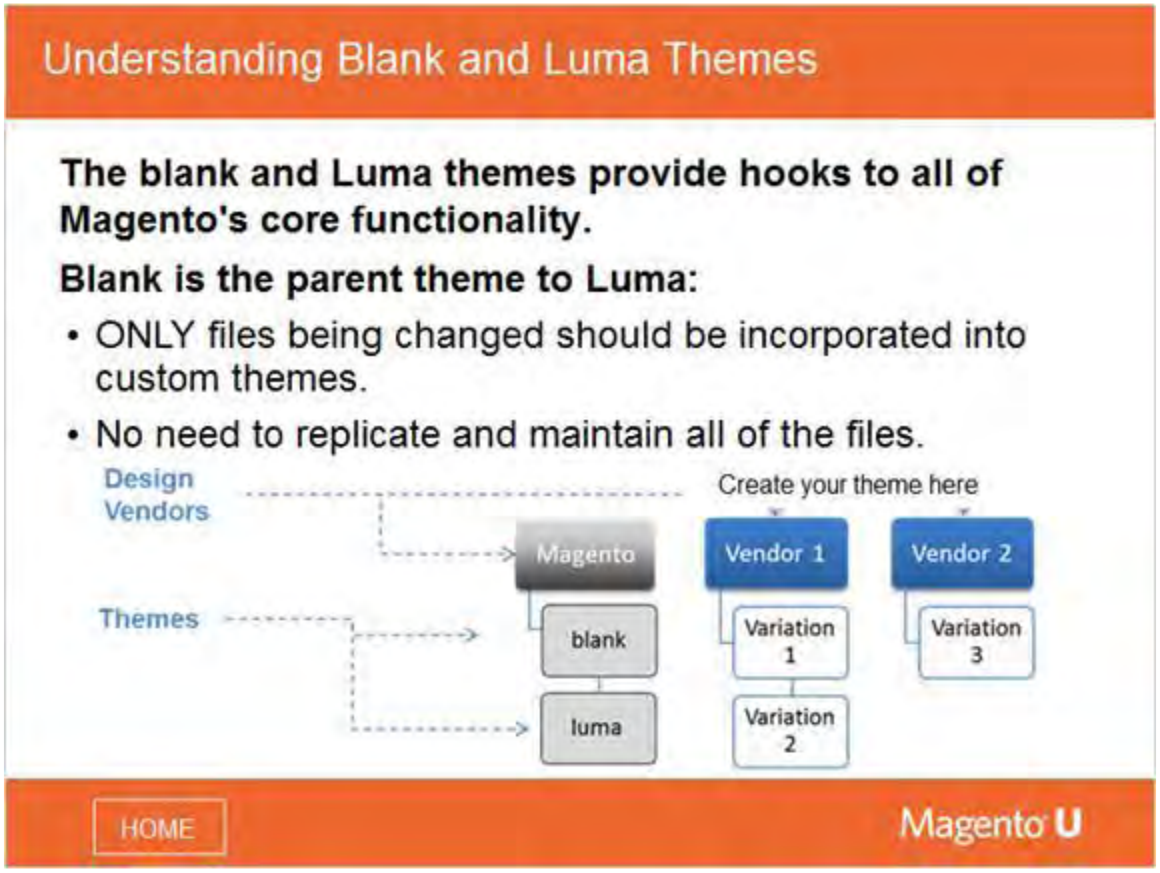
HOME

Magento U

Notes:

Magento provides two default themes that you can use as a basis to create your own themes.

1.8 Understanding Blank and Luma Themes



Understanding Blank and Luma Themes

The blank and Luma themes provide hooks to all of Magento's core functionality.

Blank is the parent theme to Luma:

- ONLY files being changed should be incorporated into custom themes.
- No need to replicate and maintain all of the files.

Design Vendors
Create your theme here
Magento
Vendor 1
Vendor 2
Themes
blank
Variation 1
Variation 3
luma
Variation 2
HOME
Magento U

Notes:

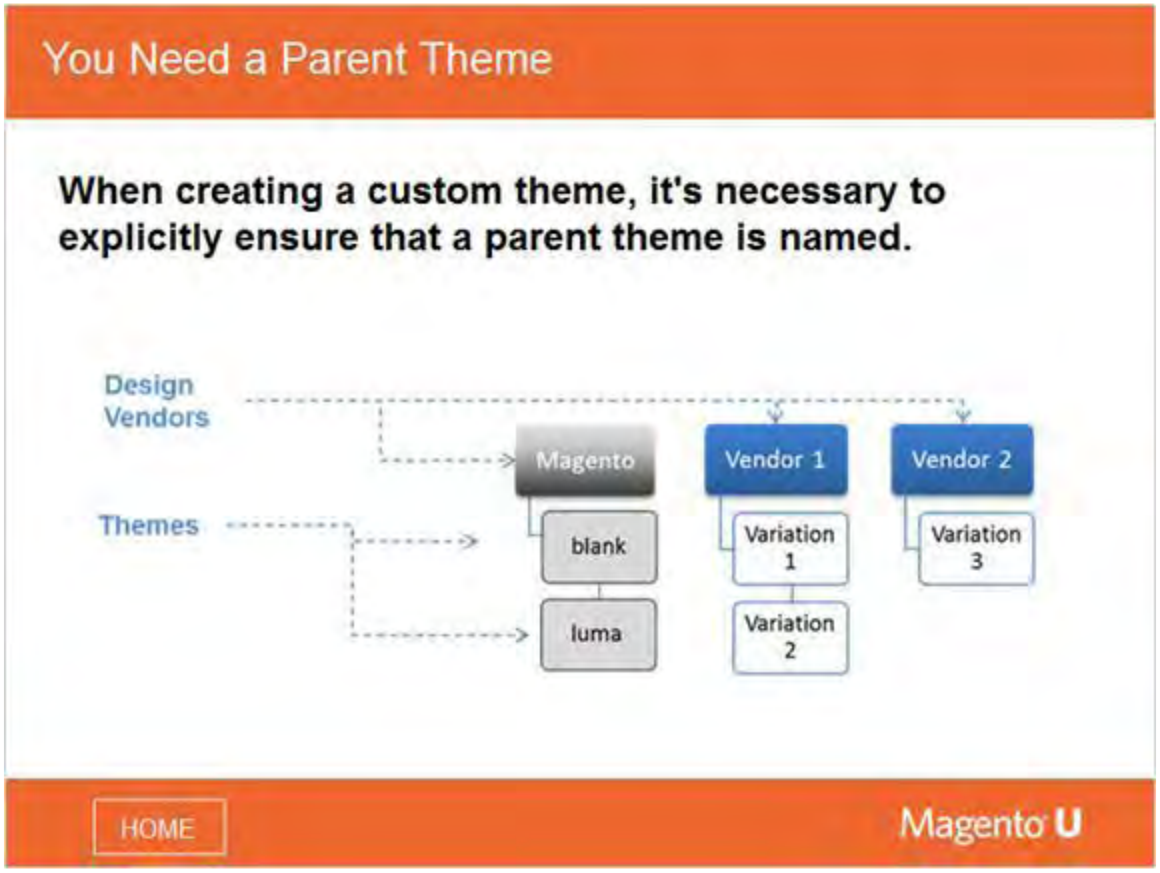
When you create your theme, you need to designate a parent theme. Your theme then only needs to incorporate files that are changed from the parent theme.

A parent and a child theme can belong to different vendors. For example, your custom theme can inherit from the Magento blank theme.

The blank theme is a basis for custom theme creation. When you inherit from the blank theme, there is no need to copy all its files. You need to override only those files that require changes, as is done in the Luma theme.

The Luma theme is a demonstration theme which inherits from the blank theme.

1.9 You Need a Parent Theme



You Need a Parent Theme

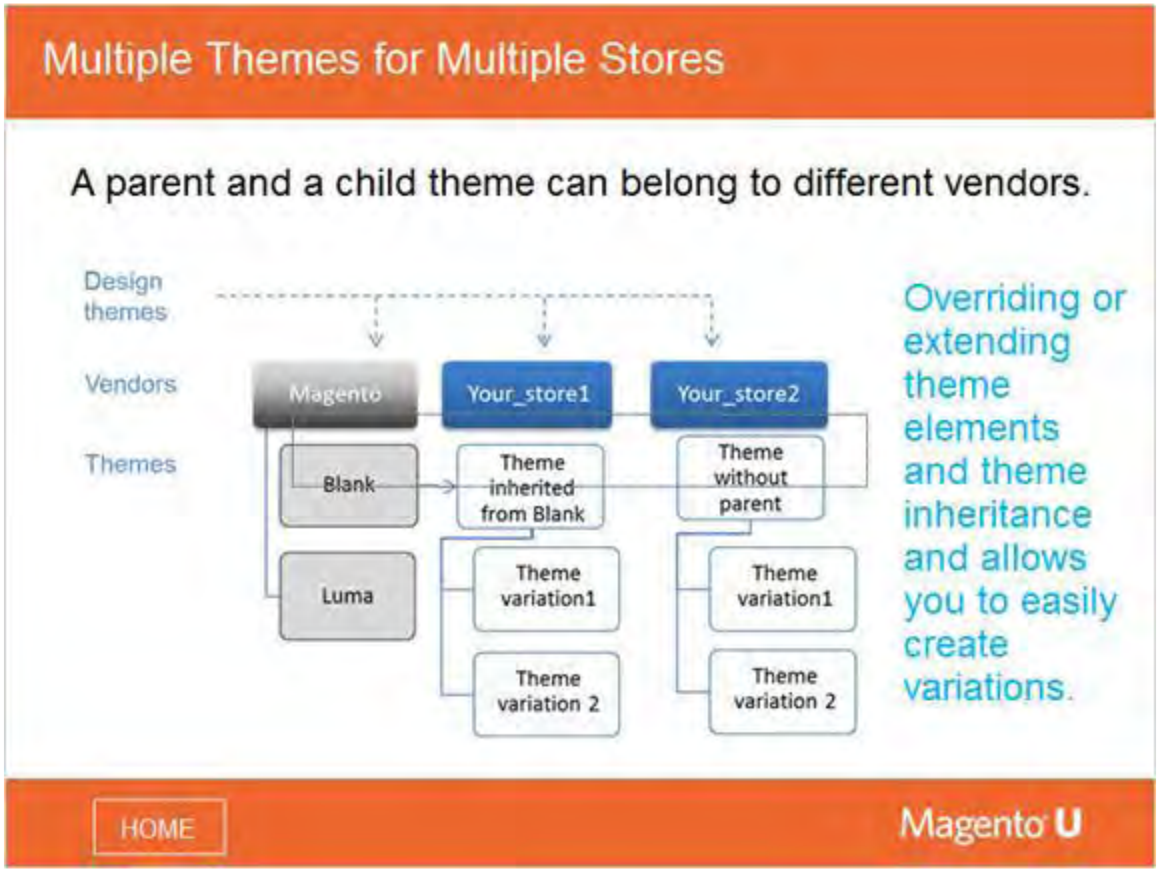
When creating a custom theme, it's necessary to explicitly ensure that a parent theme is named.

Notes:

You can see how theme inheritance works here. Luma’s parent theme is blank -- or another way to state this is that luma inherits from blank.

Depending on your design, inherit from either the blank or luma theme. Then you can create additional customizations in your theme.

1.10 Multiple Themes for Mulitple Stores



Multiple Themes for Multiple Stores

A parent and a child theme can belong to different vendors.

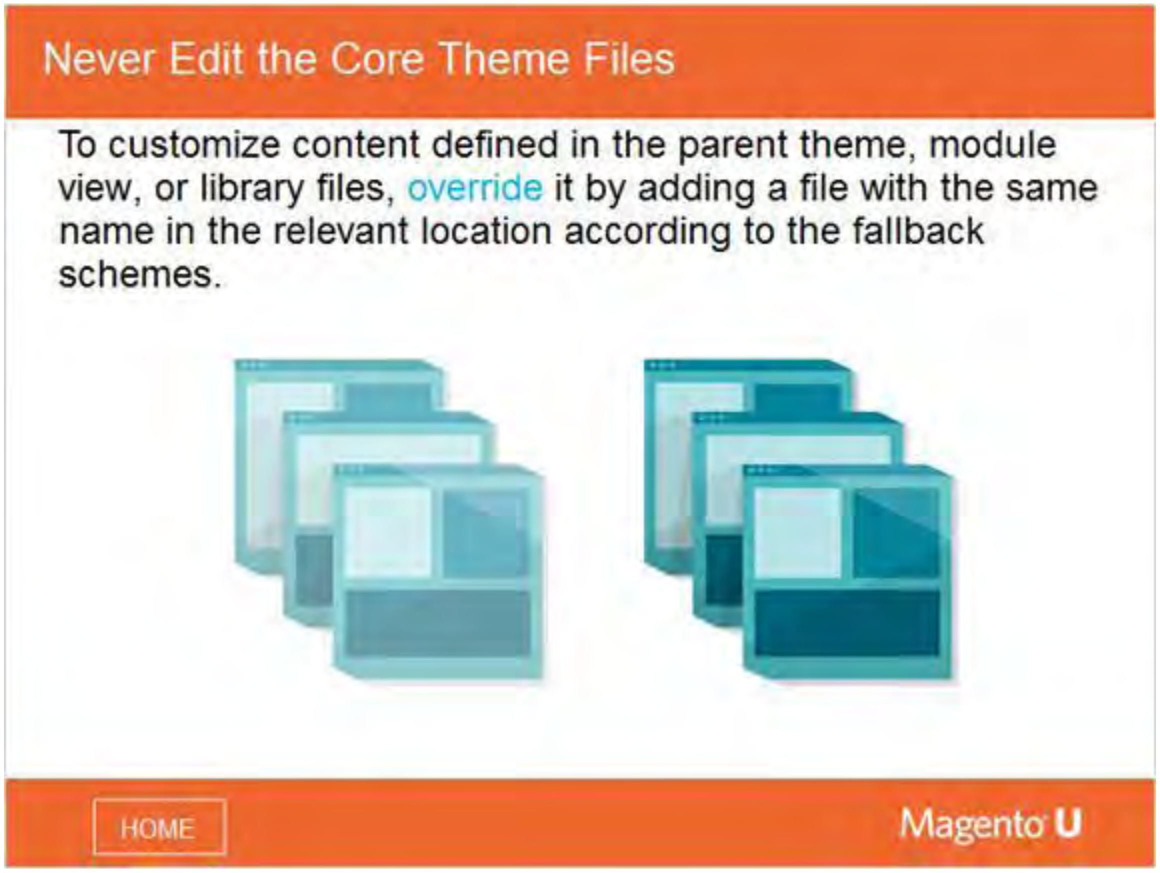
Overriding or extending theme elements and theme inheritance and allows you to easily create variations.

Notes:

We'll talk about the file structure soon but for now, realize that you have the ability to use elements from other existing themes in a new theme that you create.

You can override static assets, templates, or layouts. Overriding or extending theme elements and theme inheritance allows you to easily create variations.

1.11 Never Edit the Core Theme Files


Never Edit the Core Theme Files

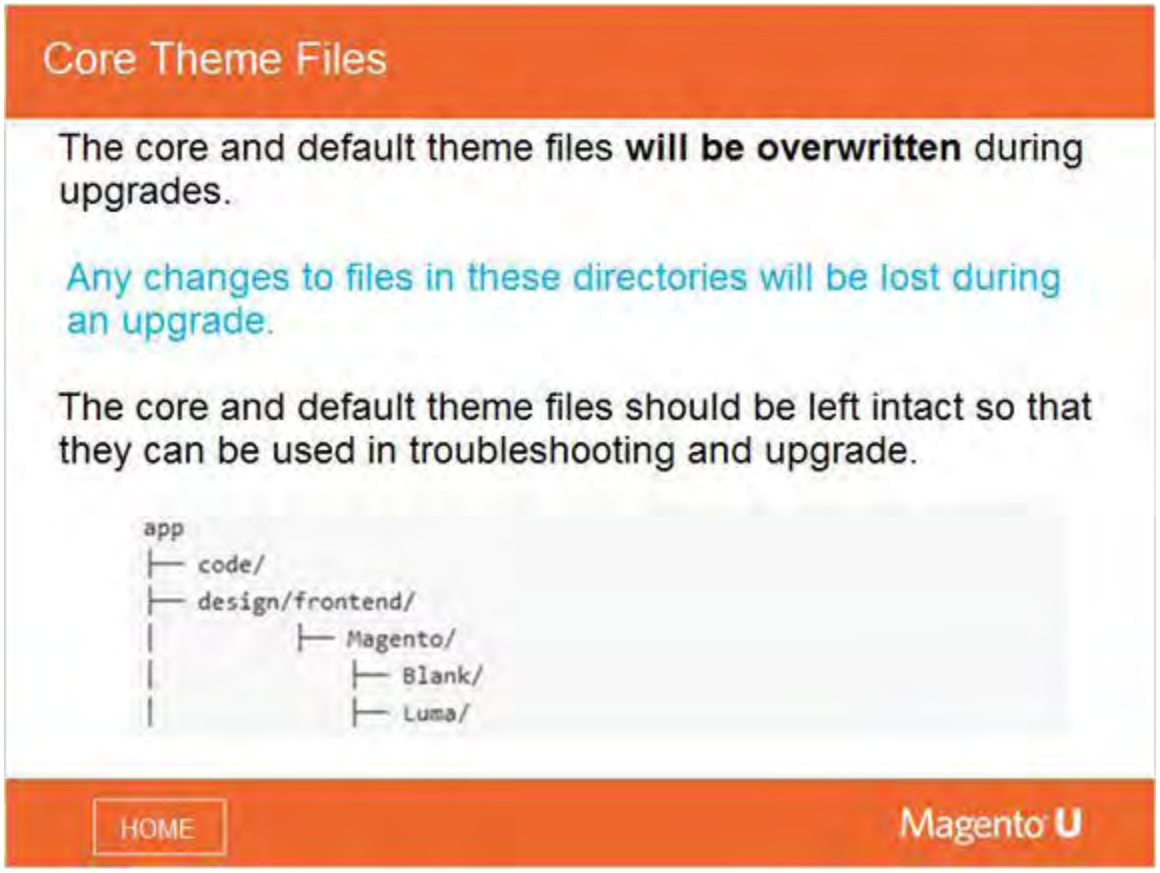
To customize content defined in the parent theme, module view, or library files, override it by adding a file with the same name in the relevant location according to the fallback schemes.

HOME
Magento U

Notes:

To customize content defined in the parent theme, module view, or library files, **override** it by adding a file with the same name in the relevant location according to the fallback schemes.

1.12 Core Theme Files



Core Theme Files

The core and default theme files **will be overwritten** during upgrades.

Any changes to files in these directories will be lost during an upgrade.

The core and default theme files should be left intact so that they can be used in troubleshooting and upgrade.

```
app
├─ code/
├─ design/frontend/
|         ├─ Magento/
|             ├─ Blank/
|             ├─ Luma/
```

HOME
Magento U

Notes:

Following these best practices ensures the easiest possible upgrade path:

- NEVER directly edit files provided by third parties, especially those from the core team. Copy specific files to a custom theme and effect changes therein.
- Changes to externally provided files will be overwritten in an upgrade.

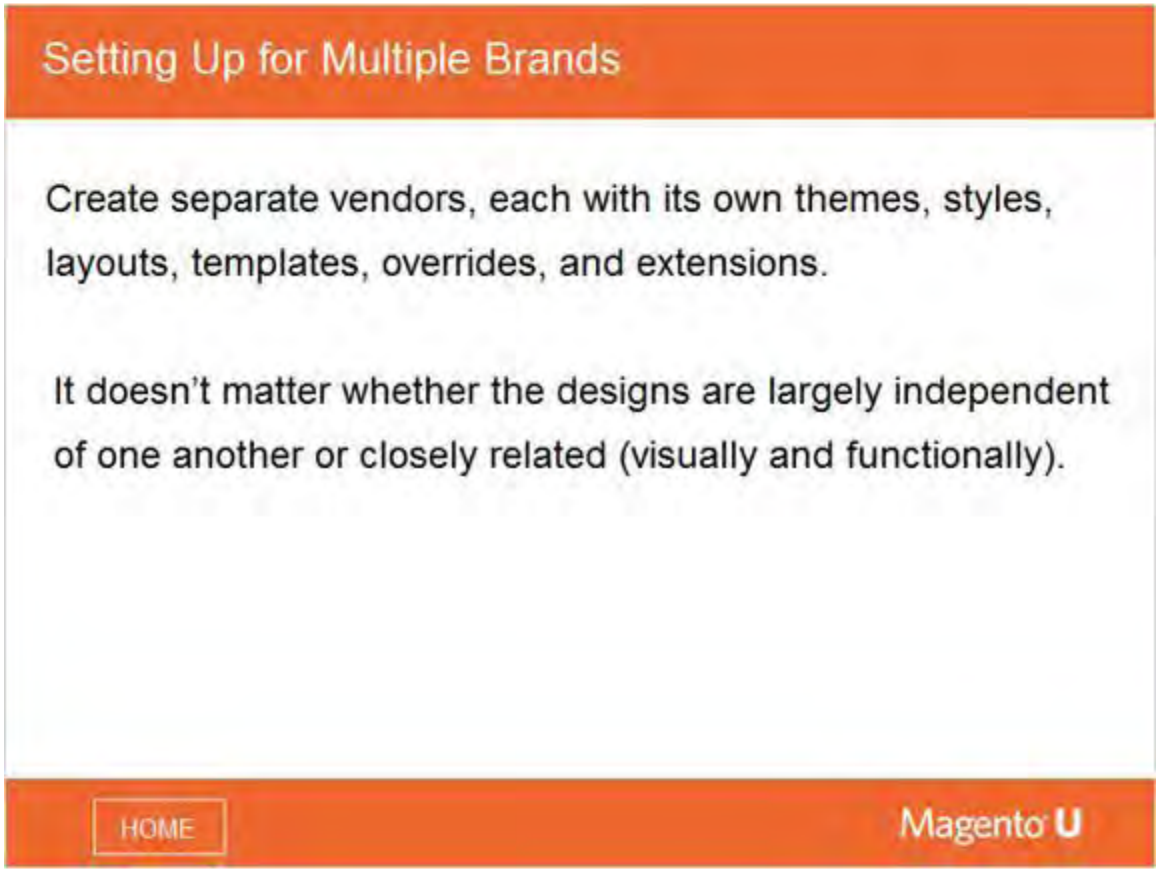
Upgrades in Magento affect theming in two ways:

- Upgrades (of Magento core and of third-party modules) simply replace **all** existing files from the core with their new versions. Because of this, core files **should never be modified, because they will be overwritten.**
- The blank and Luma themes are under Magento as the vendor in the file structure, to indicate that their files shouldn't be overwritten. Also, app/design/frontend/Magento vendor should NOT be used for placing new themes.

Upgrades require letting updated functionality and new elements be presented. Populating custom themes with only those files that are being customized provides the following advantages:

- Eases the upgrade process by allowing changes to automatically come through when the core/third-party files are overwritten
- Allows easy identification of the files that have been customized, drastically simplifying the process of merging external updates to the customized versions

1.13 Setting Up for Multiple Brands



Setting Up for Multiple Brands

Create separate vendors, each with its own themes, styles, layouts, templates, overrides, and extensions.

It doesn't matter whether the designs are largely independent of one another or closely related (visually and functionally).

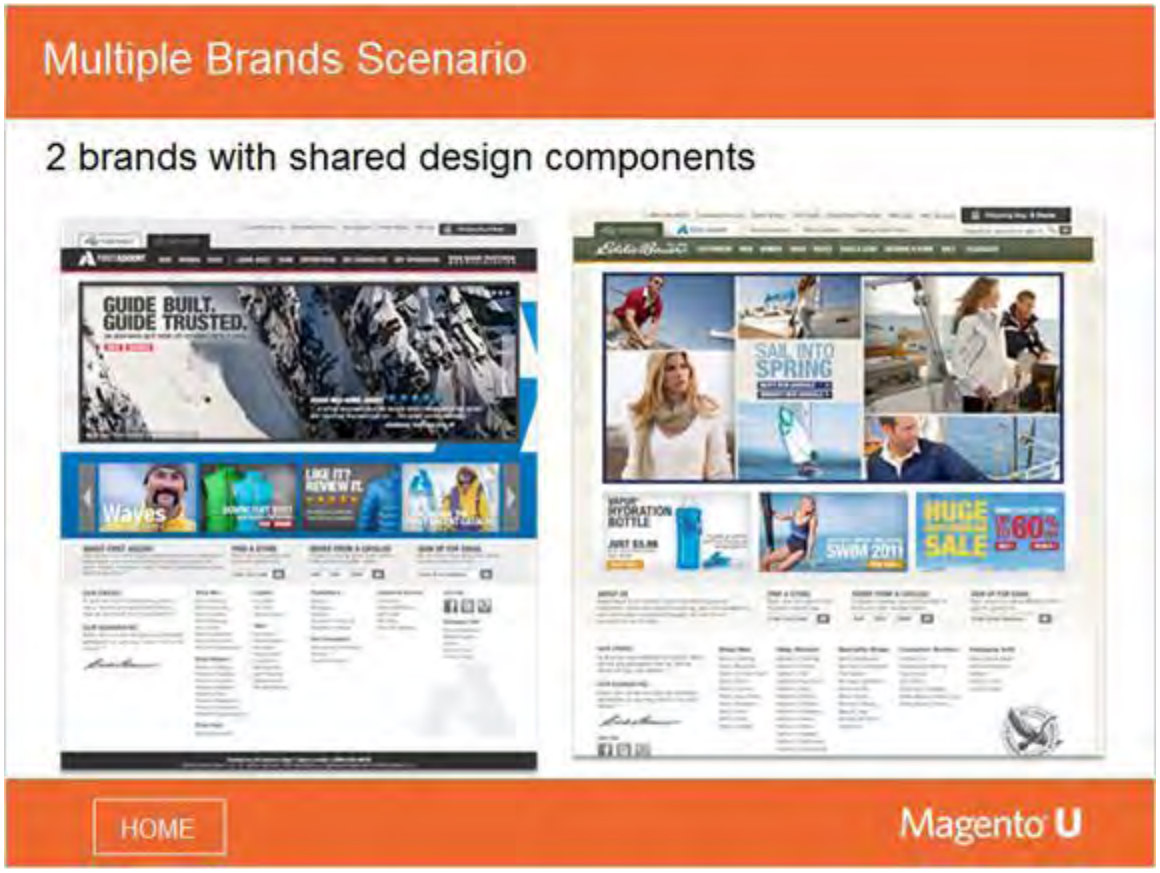
HOME
Magento U

Notes:

Create separate vendors, each with its own themes, styles, layouts, templates, overrides, and extensions. It doesn't matter whether the designs are largely independent of one another or closely related, both visually and functionally.

Even when you only have one design for multiple brands, you'll still end up with multiple themes. Only the concept of theme.xml's parent configuration might be different.

1.14 Multiple Brands Scenario



Multiple Brands Scenario

2 brands with shared design components

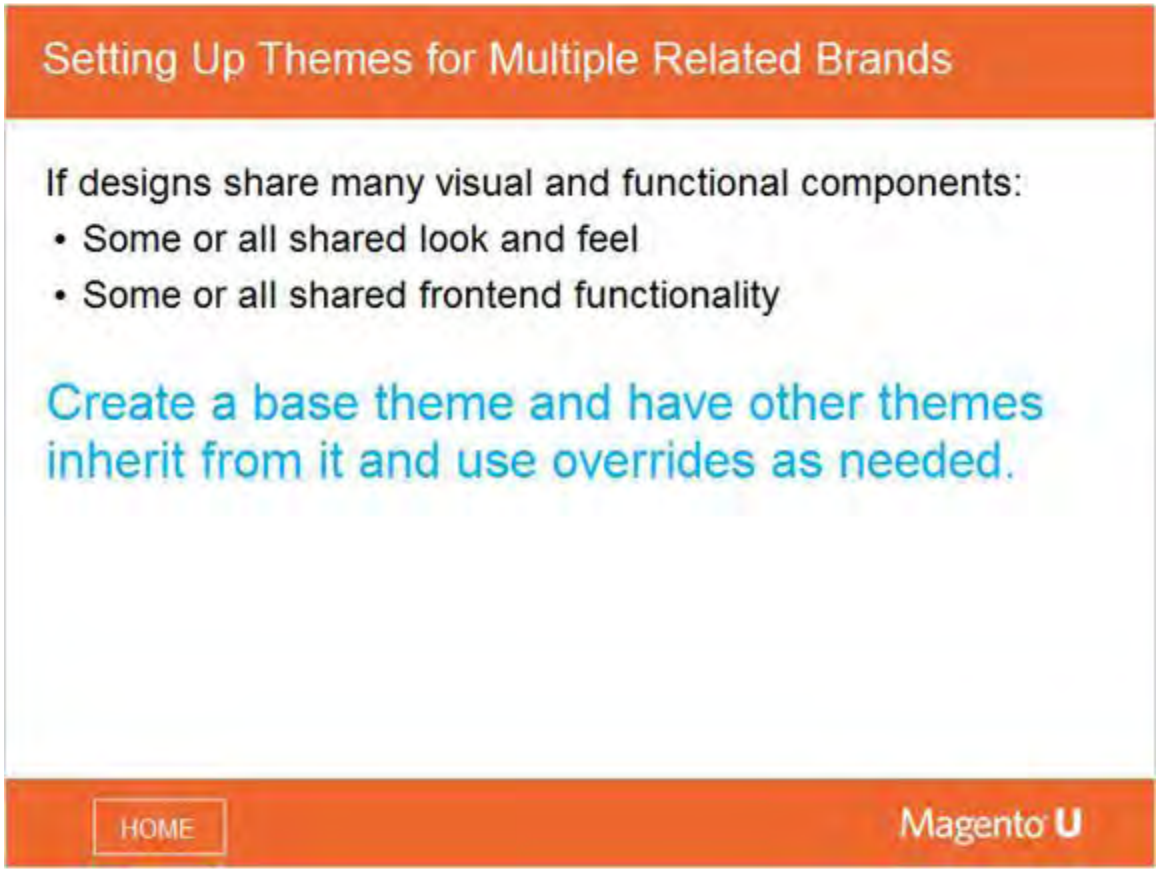
Notes:

This slide presents one instance being used for two websites with several shared user interface elements. The fallback model in Magento allows for theming work to obey the principle of **DRY ("Don't Repeat Yourself").** Each will have its own theme but use parent themes, extensions, and overrides to handle the differences.

Other examples include:

- International retailers with similarly branded stores but slight differences between countries
- Many retailers with outlet stores that are similar but less flashy than the main store

1.15 Setting Up Themes for Multiple Related Brands



Setting Up Themes for Multiple Related Brands

If designs share many visual and functional components:

- Some or all shared look and feel
- Some or all shared frontend functionality

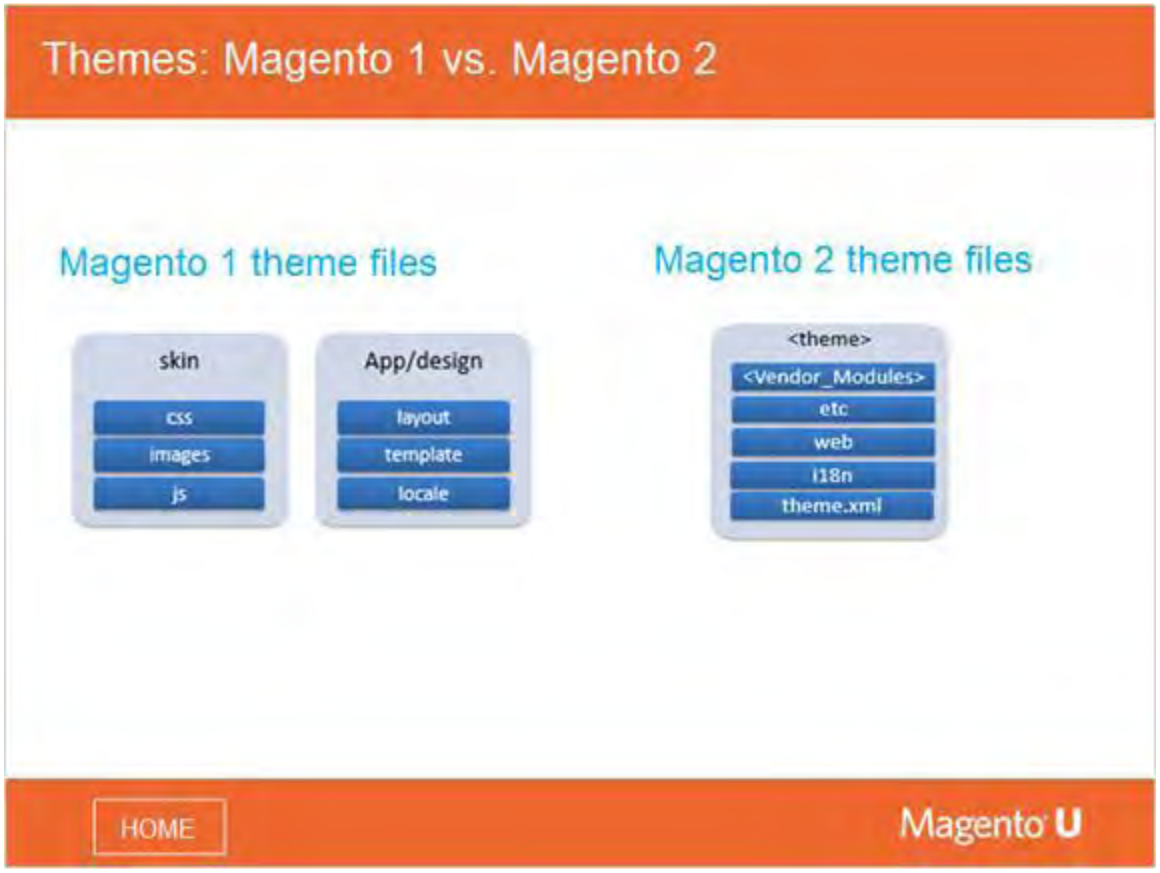
Create a base theme and have other themes inherit from it and use overrides as needed.

HOME
Magento U

Notes:

Creating a theme is easy. Create a base theme and have other themes inherit from it, and use overrides as needed. The details about extending and overriding themes will be covered in later in this course.

1.16 Themes: Magento 1 vs Magento 2

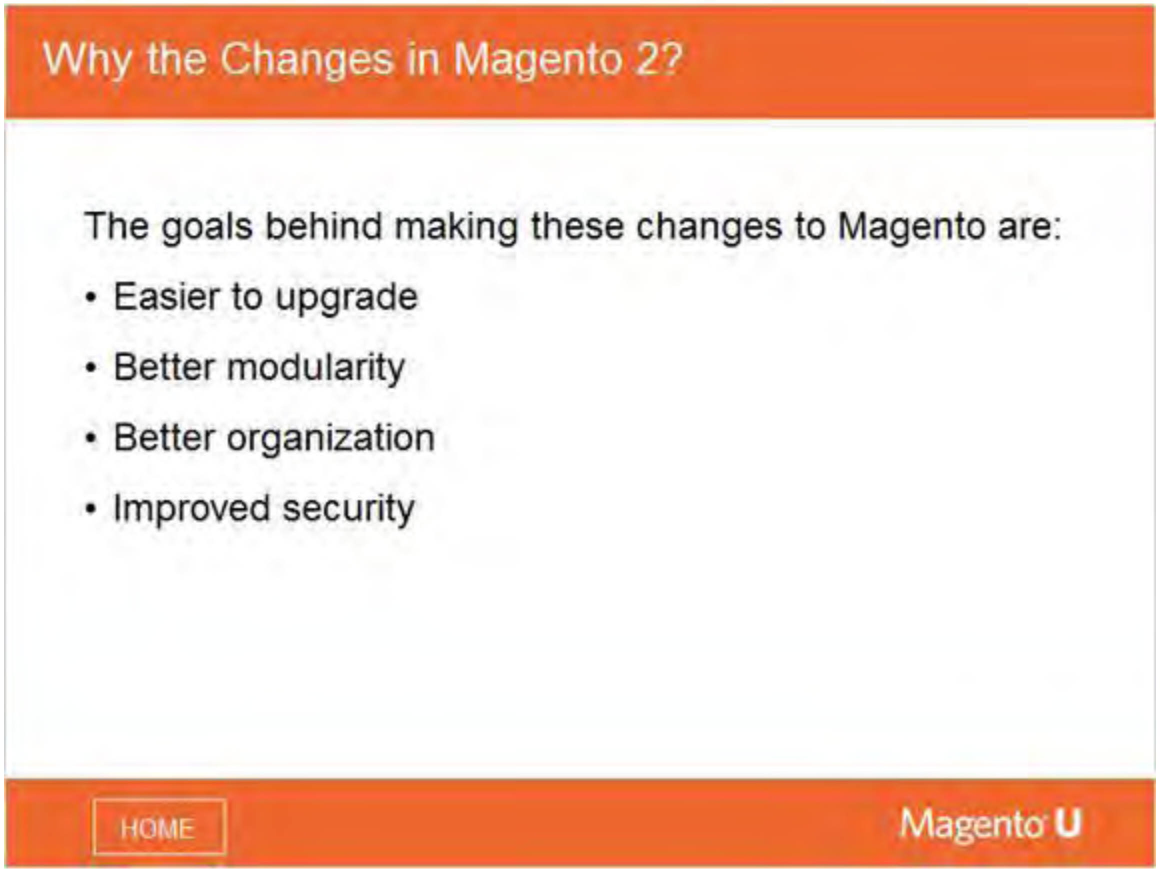


Notes:

In Magento 1, themes were kept in separate folders -- skin and app/design. In Magento 2, they are kept in a single folder.

Magento 2 allows you to override and extend files as needed to make customizations, rather than duplicate a large number of files and modify them.

1.17 Why the Changes in Magento 2?



Why the Changes in Magento 2?

The goals behind making these changes to Magento are:

- Easier to upgrade
- Better modularity
- Better organization
- Improved security

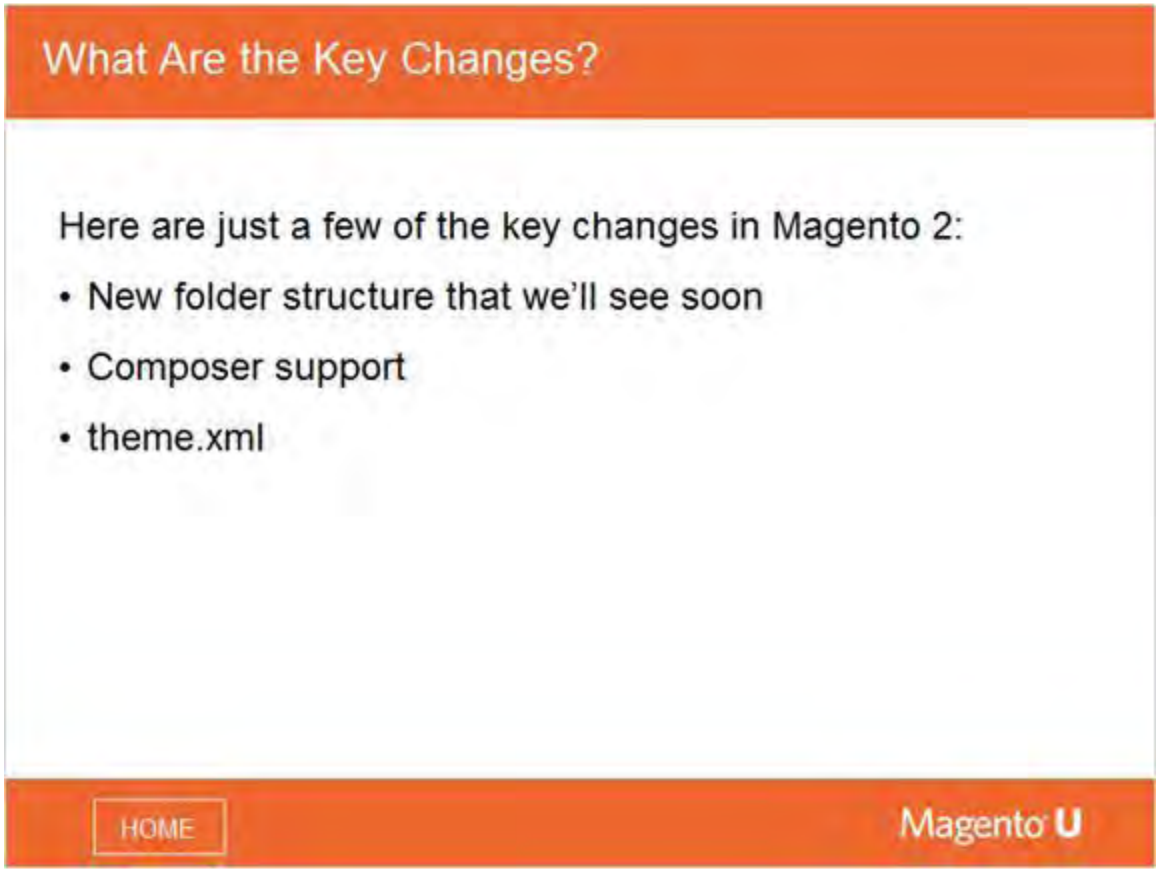
HOME
Magento U

Notes:

The reasons behind the changes in Magento 2 are to update the technology, improve performance, improve the ease of upgrades, simplify external integrations, provide better modularity, and allow better organization.

As a developer, you no longer need to worry about changing the core files.

1.18 What Are the Key Changes?



What Are the Key Changes?

Here are just a few of the key changes in Magento 2:

- New folder structure that we'll see soon
- Composer support
- theme.xml

HOME
Magento U

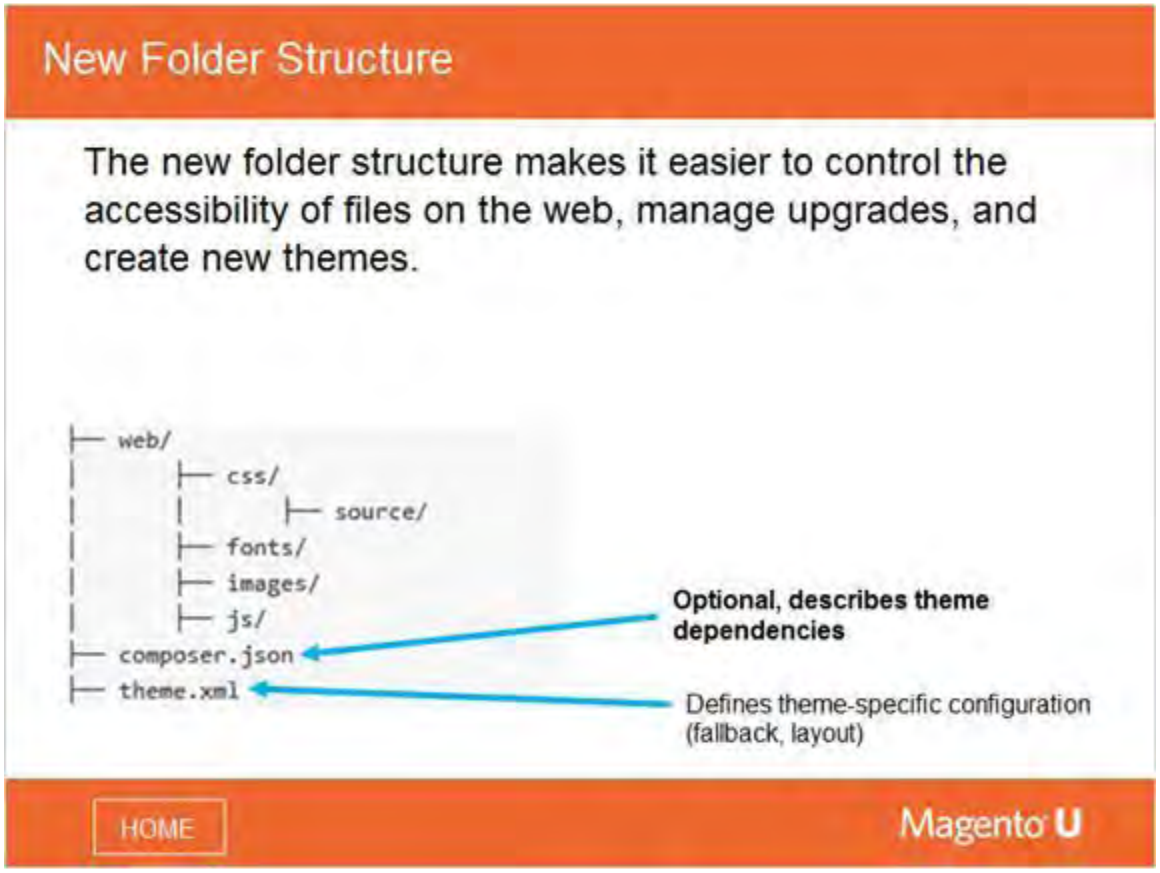
Notes:

These are just a few of the changes you'll see.

Some other changes that we'll discuss in other units of this course include:

- Magento 2 now uses the LESS CSS pre-processor, which enables easier and faster code changes and maintenance.
- Magento UI Library simplifies the process of Magento theme creation and customization.

1.19 New Folder Structure



New Folder Structure

The new folder structure makes it easier to control the accessibility of files on the web, manage upgrades, and create new themes.

```
├─ web/
|        ├─ css/
|        |       ├─ source/
|        ├─ fonts/
|        ├─ images/
|        ├─ js/
├─ composer.json
├─ theme.xml
```

Optional, describes theme dependencies
Defines theme-specific configuration (fallback, layout)
HOME
Magento U

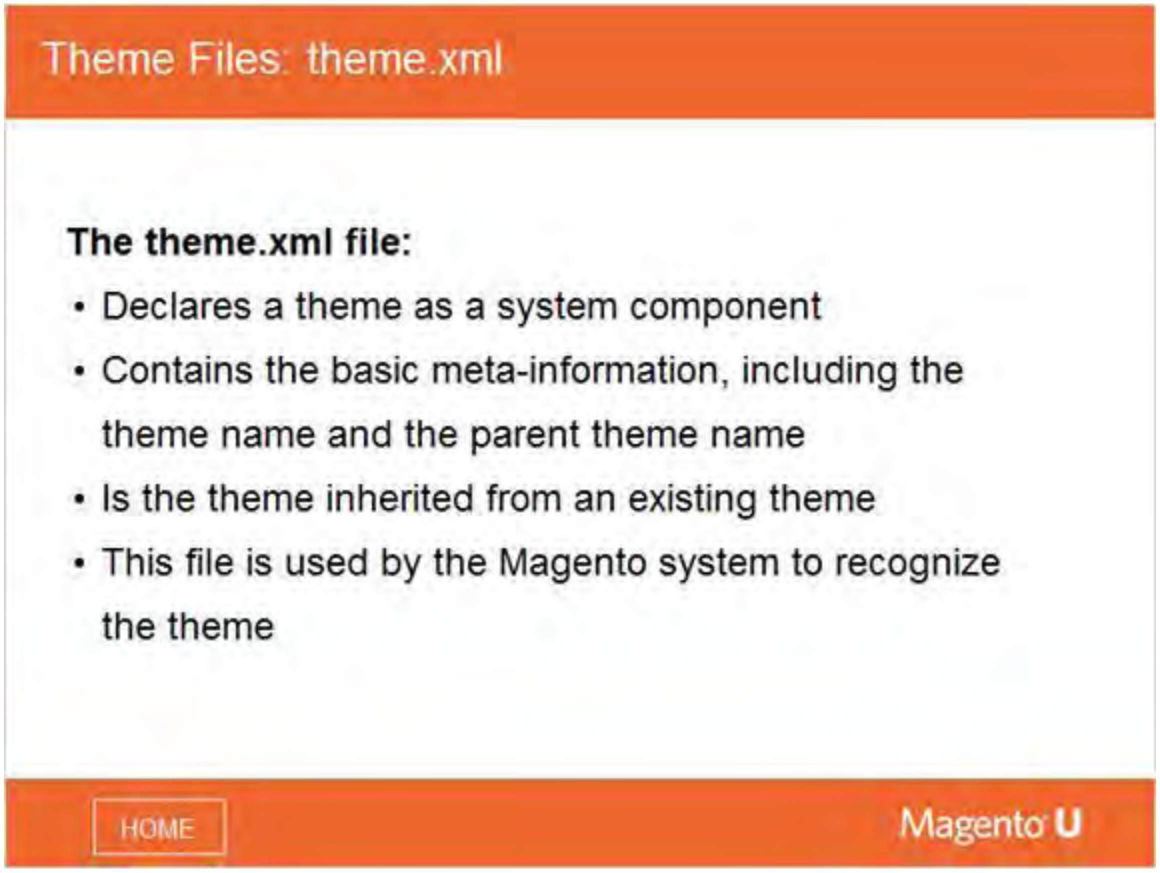
Notes:

Magento 2 uses a new folder structure to make file management for many tasks easier.

You can now extend and override the files in a parent theme to create a new theme. The new organization makes upgrades much cleaner too.

We'll cover the details of the new folder structure in depth in its own module.

1.20 Theme Files: theme.xml



Theme Files: theme.xml

The theme.xml file:

- Declares a theme as a system component
- Contains the basic meta-information, including the theme name and the parent theme name
- Is the theme inherited from an existing theme
- This file is used by the Magento system to recognize the theme

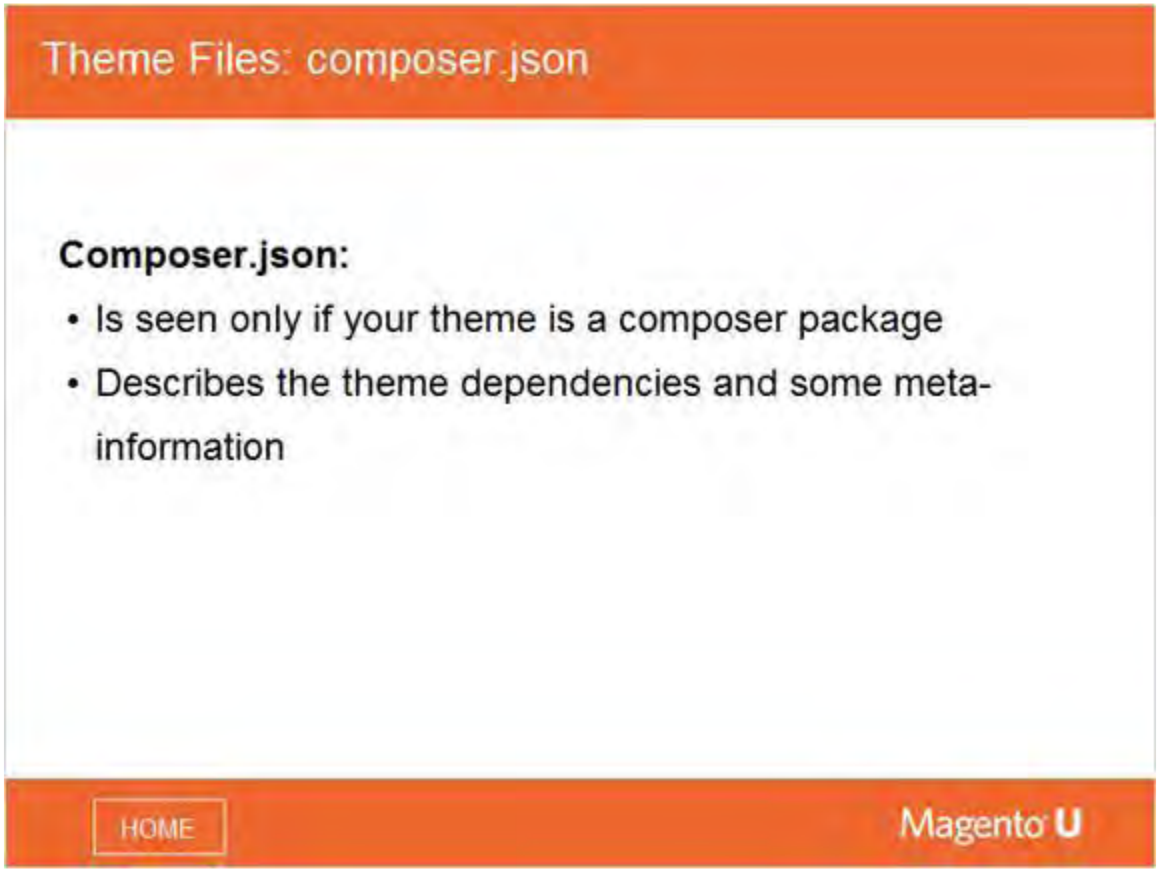
HOME
Magento U

Notes:

The theme.xml file is the file used by Magento to recognize the theme. It contains the basic information about a theme including the theme name and the parent theme name.

We'll take a closer look at its contents later.

1.21 Theme Files: composer.json



Theme Files: composer.json

Composer.json:

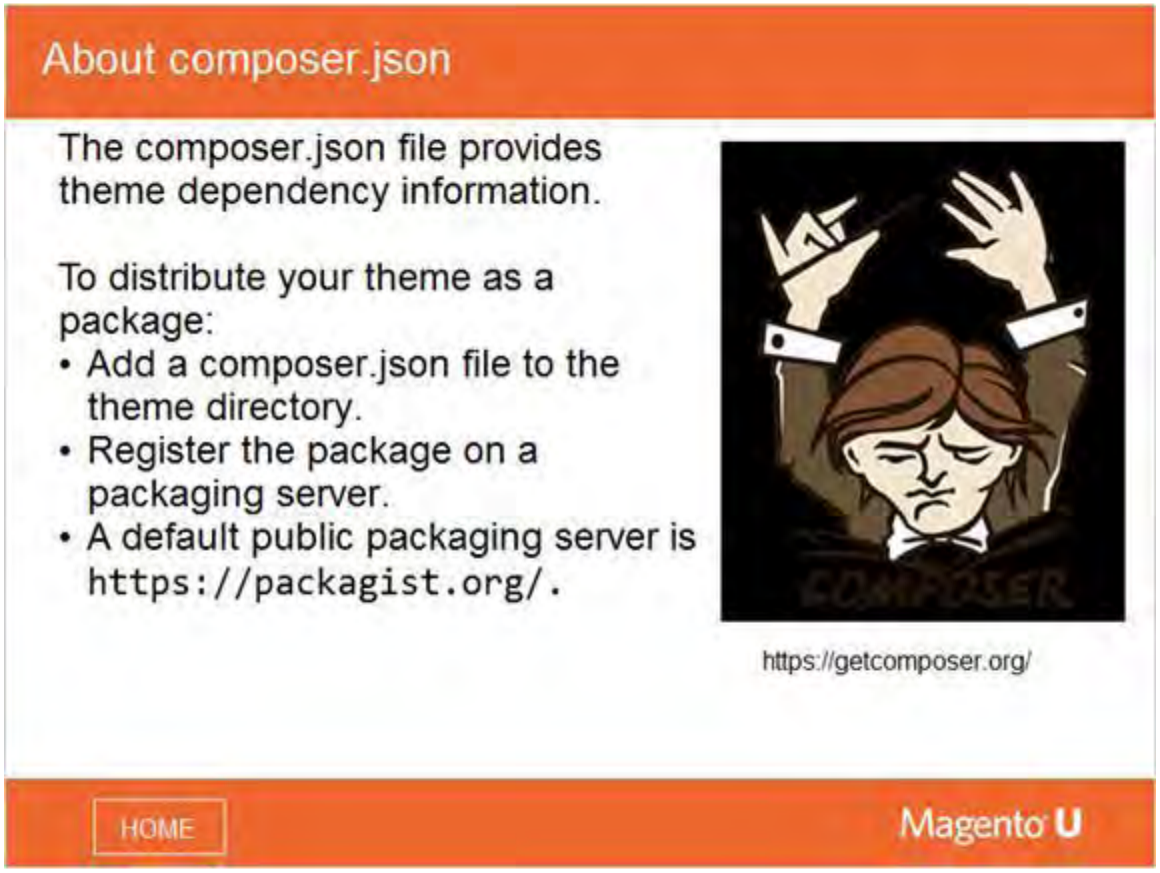
- Is seen only if your theme is a composer package
- Describes the theme dependencies and some meta-information

HOME
Magento U

Notes:

Magento 2 allows you to optionally use composer to provide theme dependency information. If you are providing your theme as a composer package, you'll see this file.

1.22 About composer.json



About composer.json

The composer.json file provides theme dependency information.

To distribute your theme as a package:
- Add a composer.json file to the theme directory.
- Register the package on a packaging server.
- A default public packaging server is `https://packagist.org/`.

https://getcomposer.org/

HOME

Magento U

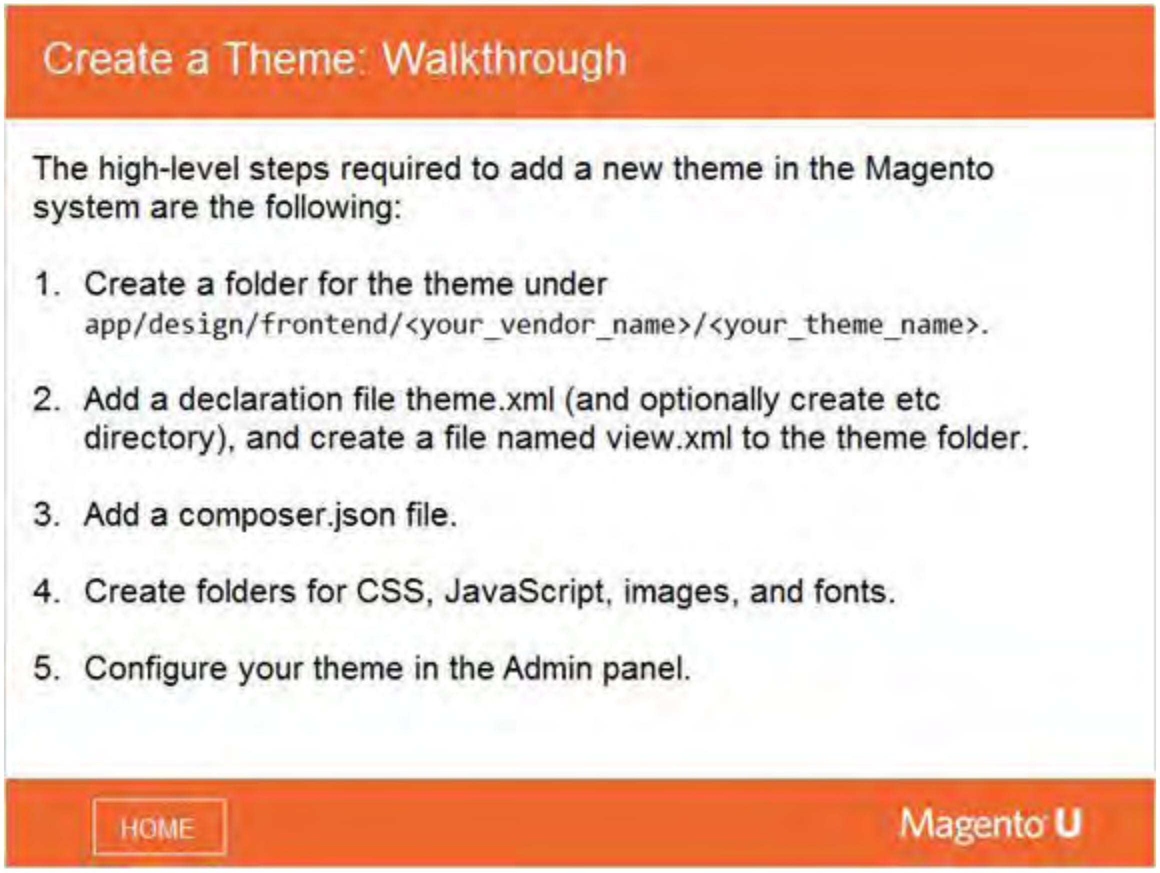
Notes:

To distribute your theme as a package:

- Add a composer.json file to the theme directory.
- Register the package on a packaging server.

To learn more about using composer with Magento, see http://devdocs.magento.com/guides/v2.0/frontend-dev-guide/themes/theme-create.html#fedg_create_theme_composer

1.23 Create a Theme: Walkthrough



Create a Theme: Walkthrough

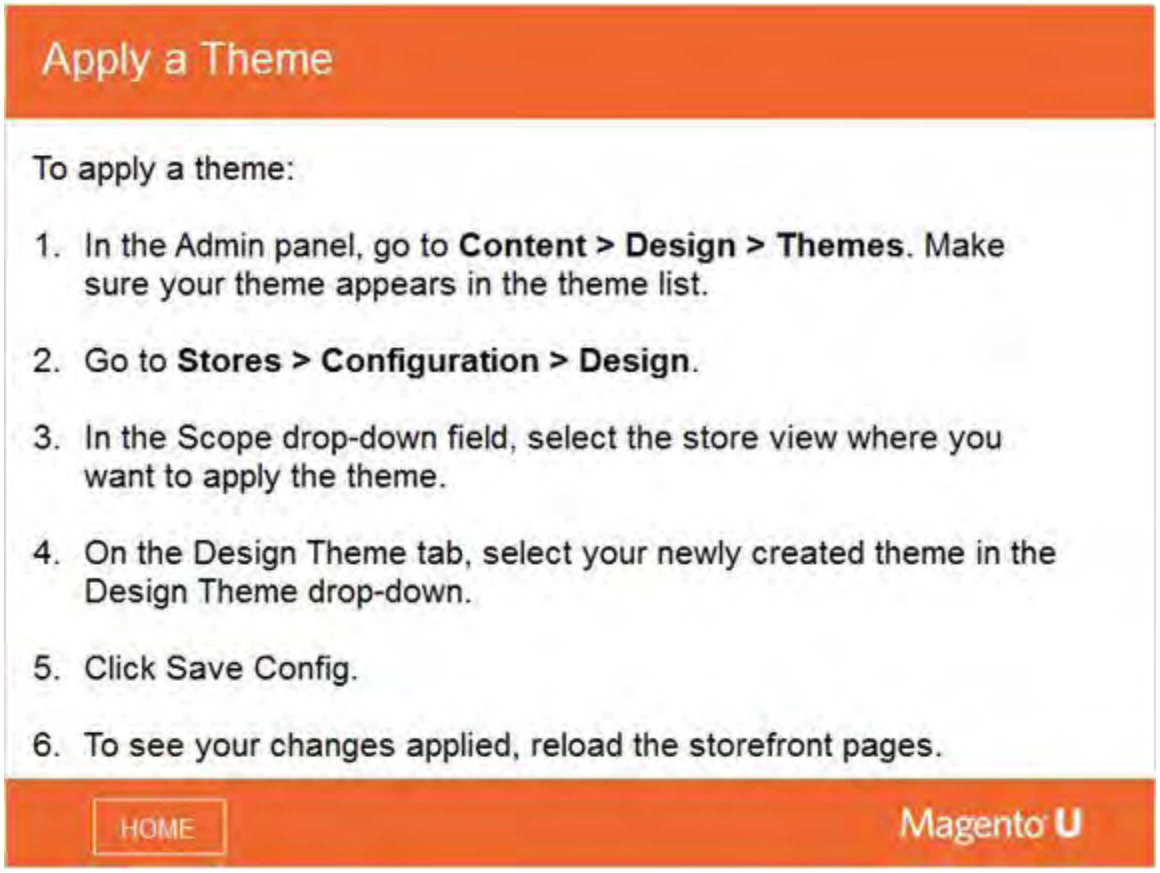
The high-level steps required to add a new theme in the Magento system are the following:

1. Create a folder for the theme under `app/design/frontend/<your_vendor_name>/<your_theme_name>`.
2. Add a declaration file theme.xml (and optionally create etc directory), and create a file named view.xml to the theme folder.
3. Add a composer.json file.
4. Create folders for CSS, JavaScript, images, and fonts.
5. Configure your theme in the Admin panel.

Notes:

Your new theme can be a standalone new theme, or can inherit from the default or any other existing one.

1.24 Apply a Theme



Apply a Theme

To apply a theme:

1. In the Admin panel, go to **Content > Design > Themes**. Make sure your theme appears in the theme list.
2. Go to **Stores > Configuration > Design**.
3. In the Scope drop-down field, select the store view where you want to apply the theme.
4. On the Design Theme tab, select your newly created theme in the Design Theme drop-down.
5. Click Save Config.
6. To see your changes applied, reload the storefront pages.

Notes:

After you add your theme to the file system, you can apply it to your store. You apply a theme in the Admin.

If caching is enabled in your Magento Admin panel, you must clear the cache to see the changes applied. You might also need to manually delete all the published static files in pub/static/frontend.

When the Magento system cache is enabled, you must clear it each time to see your design changes reflected on the storefront. To avoid this, disable certain system cache types while you make design changes.

To do this:

1. In the Admin, go to System > Tools > Cache Management.
2. Select the Layouts, Blocks HTML output, View files fallback, View files pre-processing, and Page Cache cache types.
3. In Actions, select Disable and click Submit. The selected cache types now show a red bar in the status area that reads DISABLED.

2. Module 2 Folder Structure

2.1 Module 2: Folder Structure

Notes:

In this module, we’ll take a closer look at the folder structure of Magento 2.

2.2 Module Topics

Module Topics

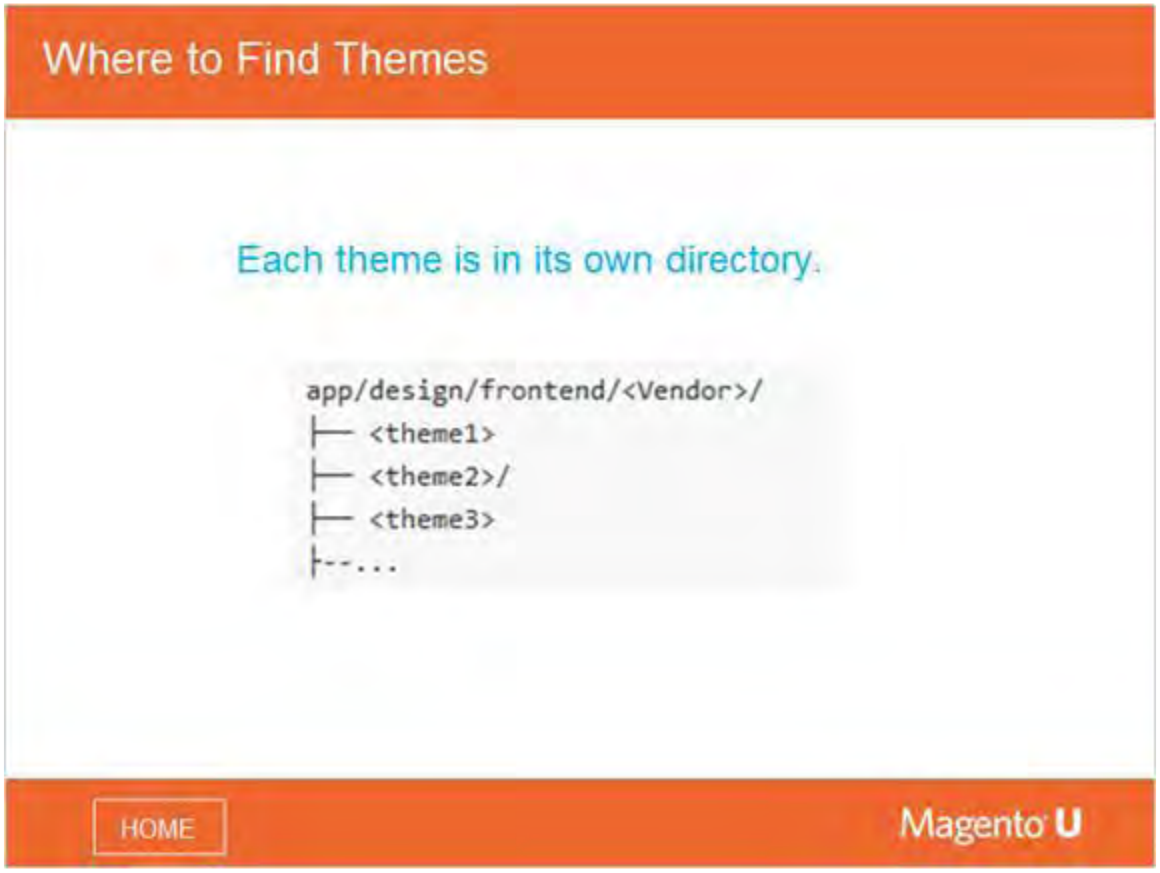
In this module, we will introduce...

- The Magento 2 folder structure
- Locating themes, templates, layouts, styles

Notes:

This section introduces what Magento themes are, and what they control in Magento implementations. We'll also look at what that translates to in the file system and in the Admin panel.

2.3 Where to Find Themes



Where to Find Themes

Each theme is in its own directory.

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

HOME
Magento U

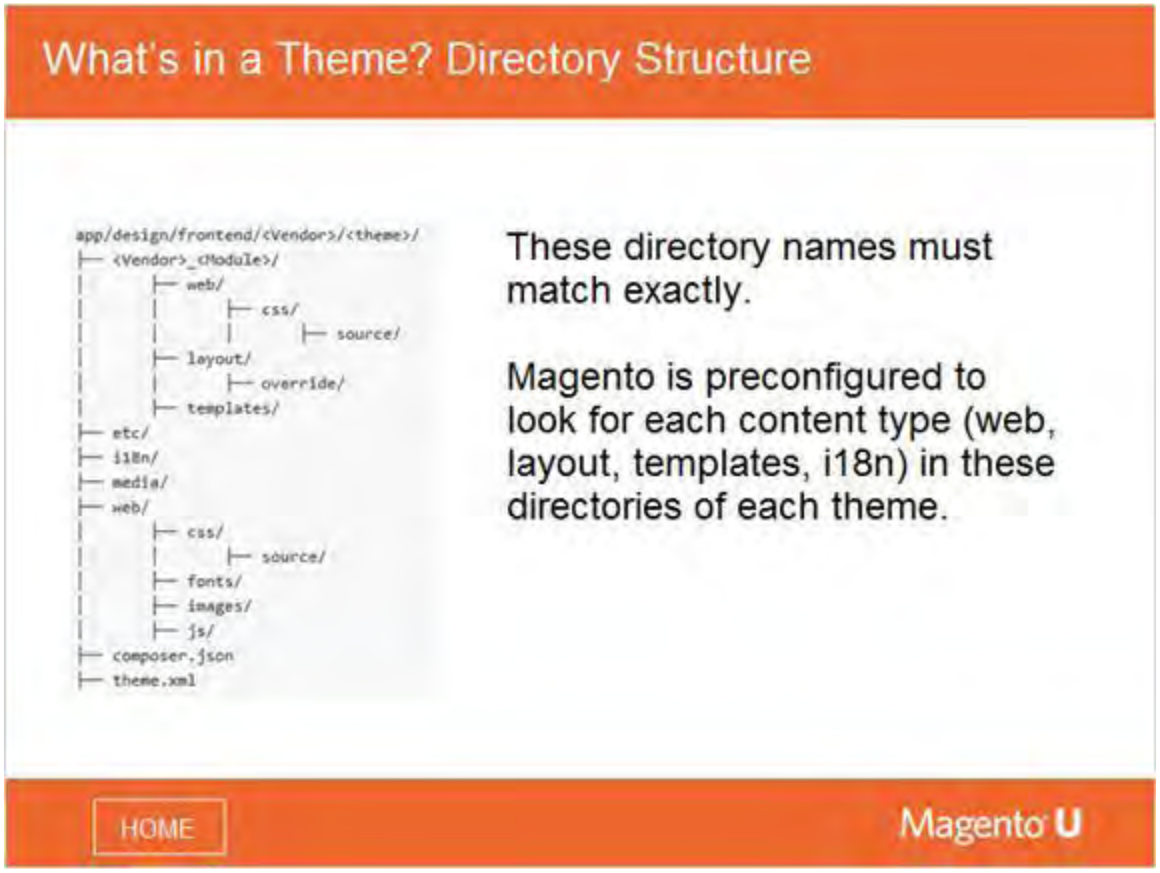
Notes:

All Magento storefront themes are located under app/design/frontend/<Vendor>/.

Each theme is in its own directory.

Everything in that theme is in the theme folder or in the folder of the parent theme. We will discuss parent themes at length later.

2.4 What's in a Theme? Directory Structure



What's in a Theme? Directory Structure

```
app/design/frontend/<Vendor>/<theme>/
├── <Vendor>_<Module>/
|       ├── web/
|       |       ├── css/
|       |       |       ├── source/
|       ├── layout/
|       |       ├── override/
|       ├── templates/
├── etc/
├── i18n/
├── media/
├── web/
|       ├── css/
|       |       ├── source/
|       ├── fonts/
|       ├── images/
|       ├── js/
├── composer.json
├── theme.xml
```

These directory names must match exactly.

Magento is preconfigured to look for each content type (web, layout, templates, i18n) in these directories of each theme.

HOME
Magento U

Notes:

You'll need to become familiar with this new directory structure.

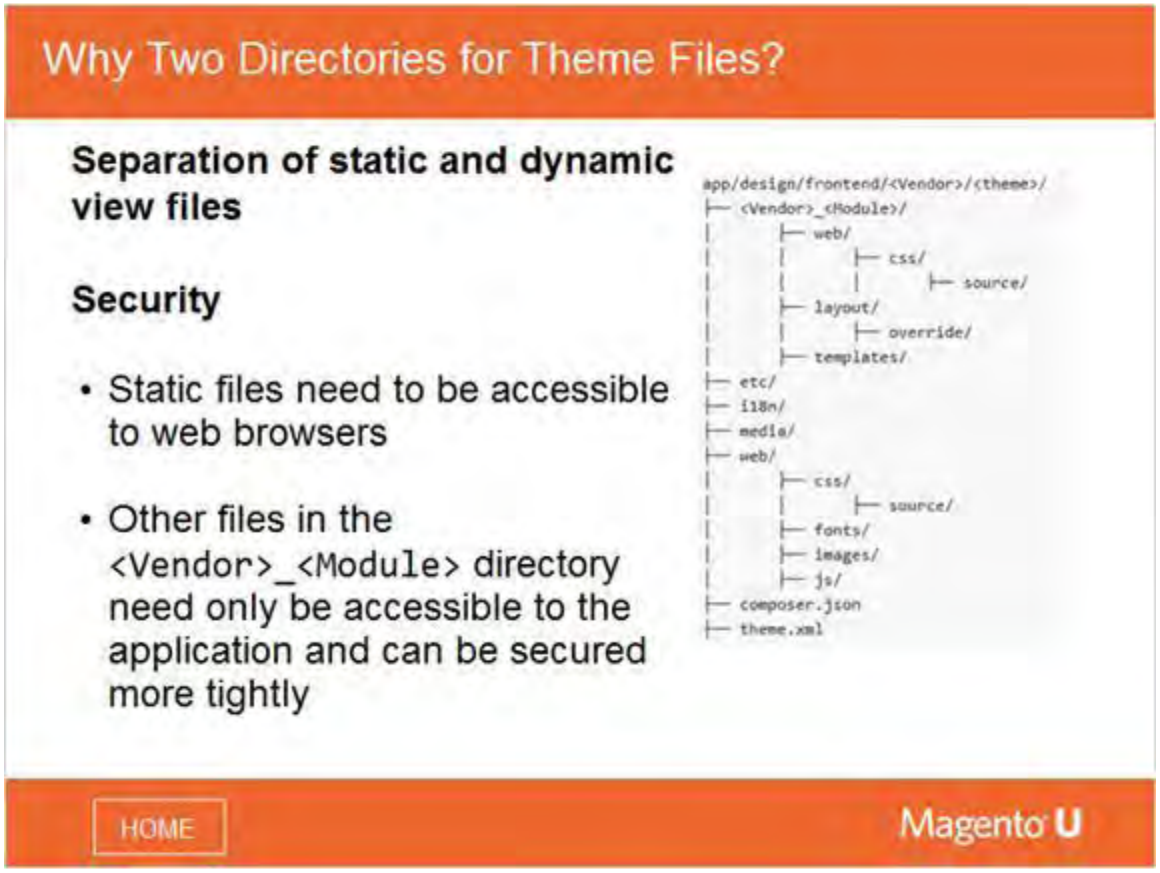
All Magento storefront themes are located under app/design/frontend/<Vendor>/.

All theme files are organized in type-specific directories (for example, templates in "templates") underneath a folder with the same name as the theme, taken from the configuration.

Themes can inherit from a parent theme. You don't need to duplicate the inherited information. For example, the “etc” folder is mandatory, when the theme has no parent. It contains a view.xml file which configures storefront picture sizes. If the theme has a parent, this file can be inherited from the parent, and is not required to exist in the child theme.

The file composer.json is optional and will exist only if the theme is a Composer package.

2.5 Why Two Directories for Theme Files?



Why Two Directories for Theme Files?

Separation of static and dynamic view files

Security

- Static files need to be accessible to web browsers
- Other files in the `<Vendor>_<Module>` directory need only be accessible to the application and can be secured more tightly

```
app/design/frontend/<Vendor>/<theme>/
├── <Vendor>_<Module>/
|       ├── web/
|       |       ├── css/
|       |       |       ├── source/
|       ├── layout/
|       |       ├── override/
|       ├── templates/
├── etc/
├── i18n/
├── media/
├── web/
|       ├── css/
|       |       ├── source/
|       ├── fonts/
|       ├── images/
|       ├── js/
├── composer.json
├── theme.xml
```

Notes:

There are separate directories for the static and dynamic view files so that you can keep your dynamic files secure.

Static view files
A set of theme files that are returned by the server to a browser as is, without any processing, are called the *static files* of a theme. These include fonts, images, CSS content.

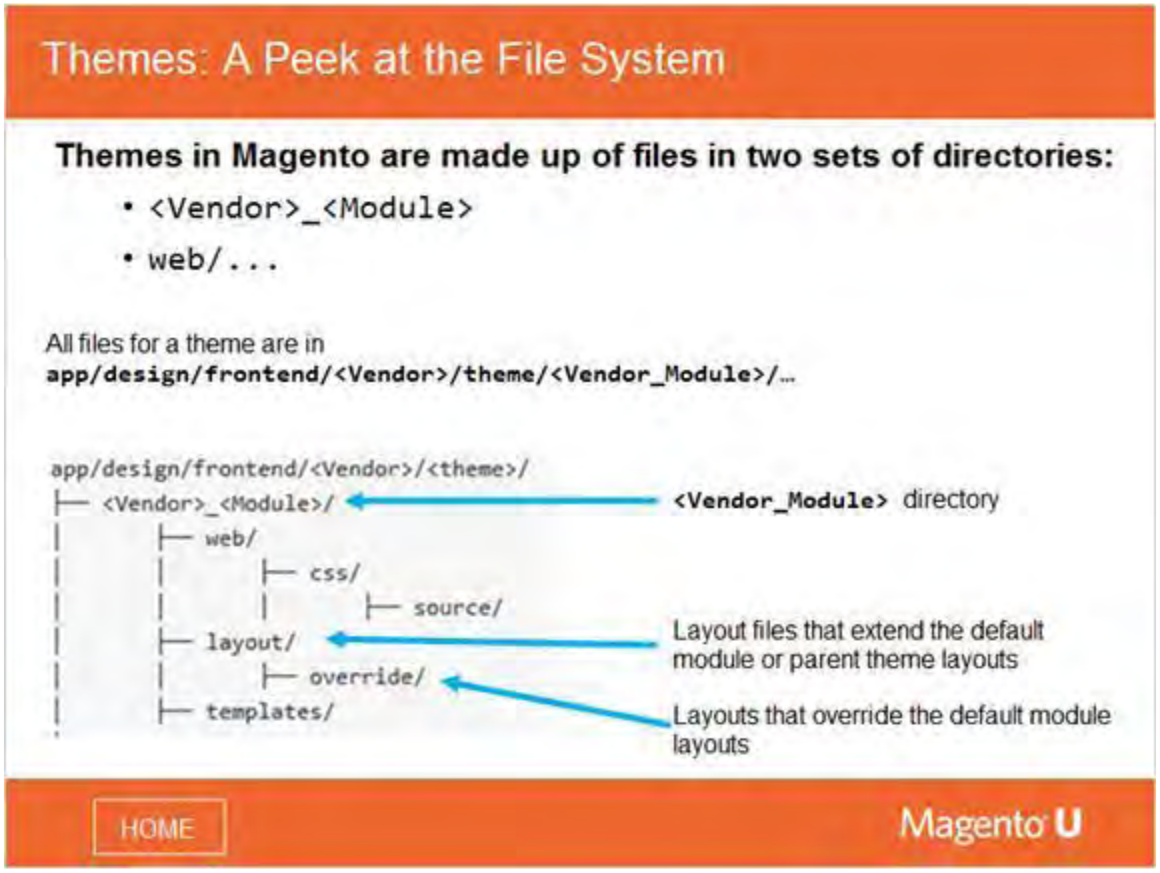
Dynamic view files
View files that are processed or executed by the server in order to provide a result to the client are referred to as *dynamic files*. These include .less files, templates, and layouts.

Key difference: Static vs. dynamic
The key difference between static files and other theme files is that static files appear on a web page as references to the files, whereas other theme files take part in the page generation, but are not explicitly referenced on a web page as files.

Public static files
Static view files that can be accessed by a direct link from the storefront are referred to as *public theme* files. Public static files are published to the /pub/static/frontend/<Vendor>/<theme>/<language>/ directory and its subdirectories.

There is a deployment process that is used to make these files accessible on the web. We'll discuss that in a later module.

2.6 Themes: A Peek at the File System



Themes: A Peek at the File System

Themes in Magento are made up of files in two sets of directories:

- `<Vendor>_<Module>`
- `web/...`

All files for a theme are in
`app/design/frontend/<Vendor>/theme/<Vendor_Module>/…`

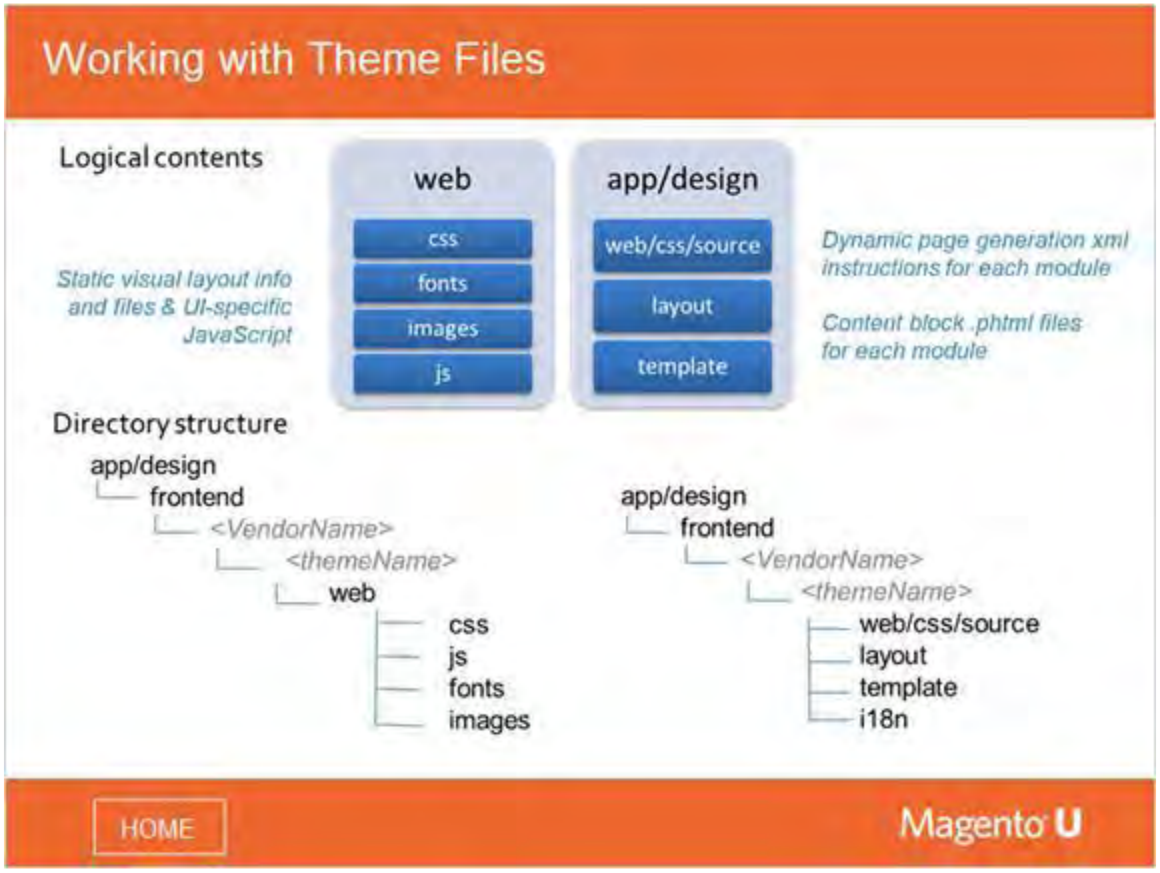
```
app/design/frontend/<Vendor>/<theme>/
├─ <Vendor>_<Module>/
|      ├─ web/
|      |      ├─ css/
|      |      |      ├─ source/
|      ├─ layout/
|      |      ├─ override/
|      ├─ templates/
```

<Vendor_Module> directory
Layout files that extend the default module or parent theme layouts
Layouts that override the default module layouts
HOME
Magento U

Notes:

Let's take a closer look. Everything in a theme is in the Vendor/theme folder. Note that there are many ./web folders – one for the theme itself, and many for the override directories.

2.7 Working with Theme Files



Working with Theme Files

Logical contents

Static visual layout info and files & UI-specific JavaScript

Dynamic page generation xml instructions for each module

Content block .phtml files for each module

Directory structure

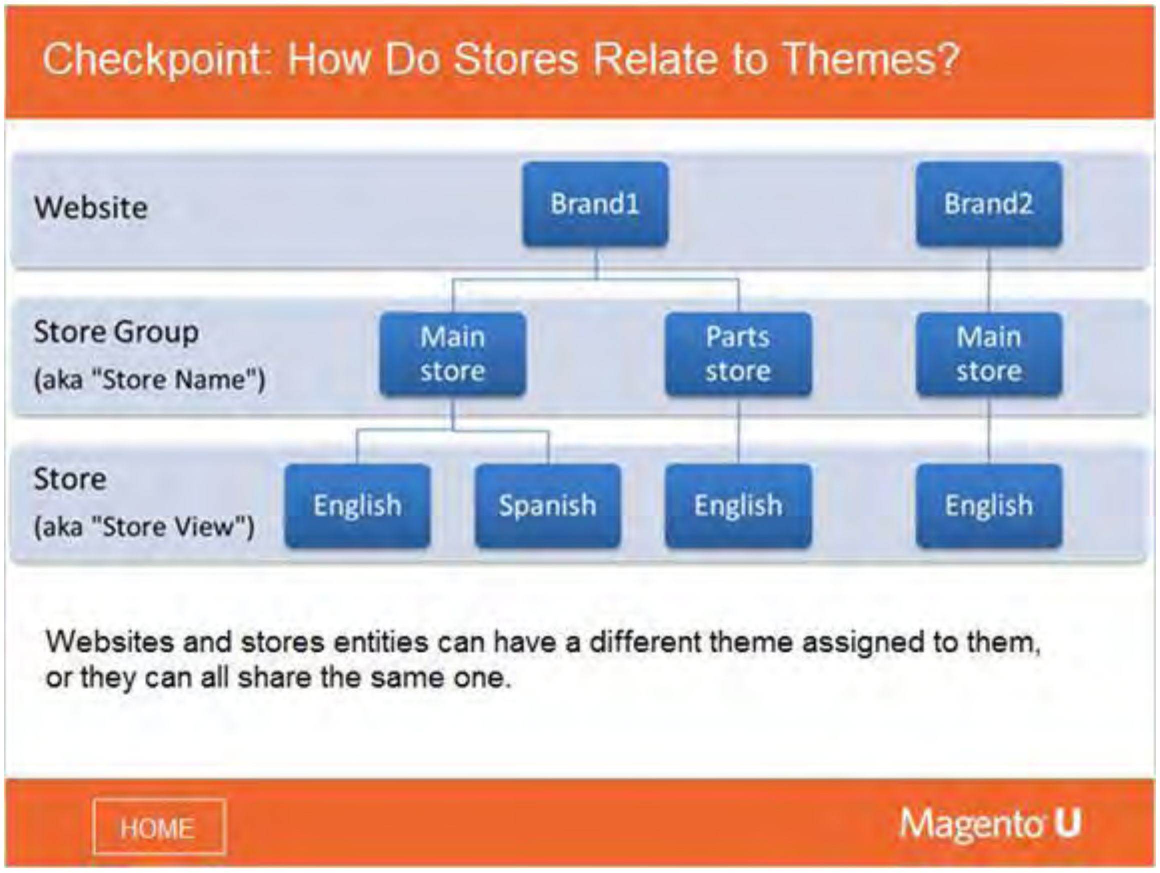
```
app/design
└── frontend
    └── <VendorName>
        └── <themeName>
            └── web
                ├── css
                ├── js
                ├── fonts
                └── images
```

```
app/design
└── frontend
    └── <VendorName>
        └── <themeName>
            ├── web/css/source
            ├── layout
            ├── template
            └── i18n
```

Notes:

Unlike layout XML and template files, which are declared relative to **layout** and **template** folders respectively, skin assets are always declared relative to the theme folder level.

2.8 Checkpoint: How Do Stores Relate to Themes?



Checkpoint: How Do Stores Relate to Themes?

Websites and stores entities can have a different theme assigned to them, or they can all share the same one.

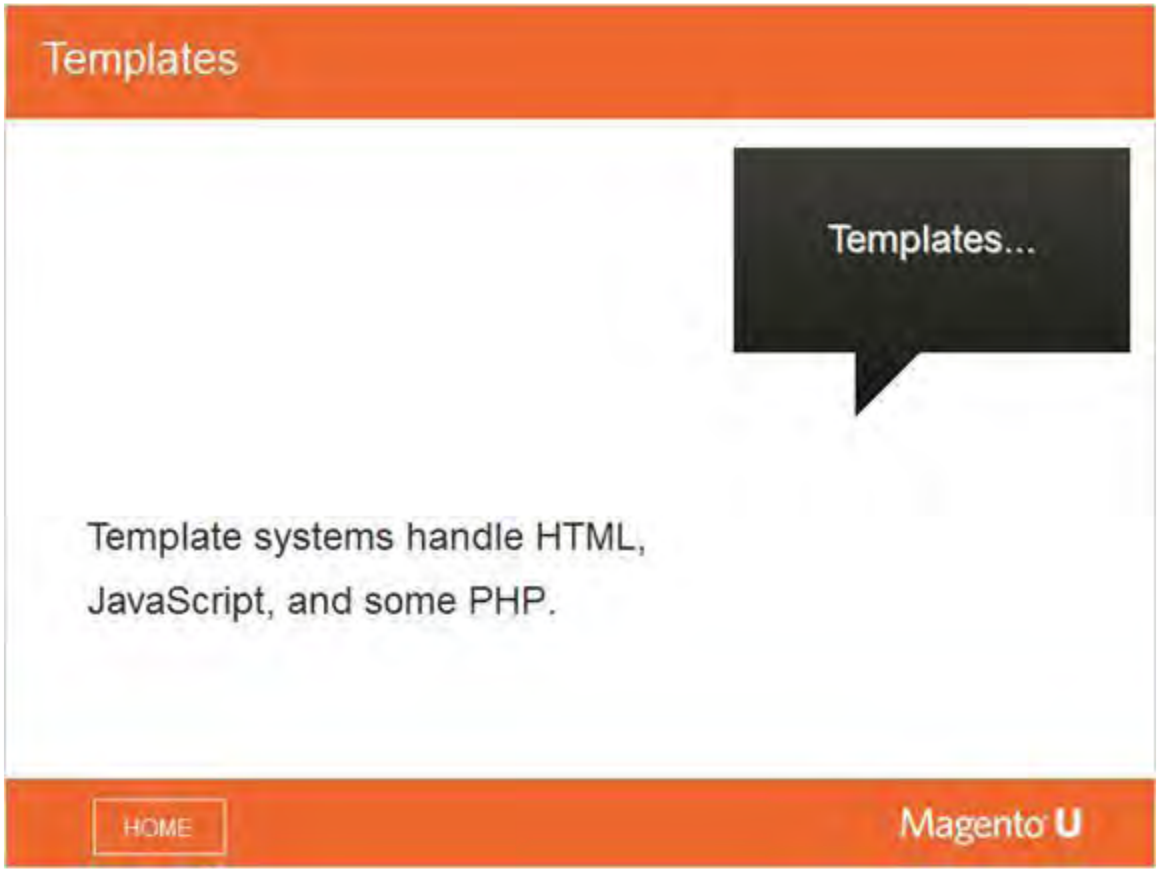
Notes:

It’s important to understand how stores and themes work together, so you can create themes appropriate to your store structure.

Typically, the main store in a set will be assigned the default theme and the other store types will be handled in the non-default themes for that design.

Themes are interchangeable without loss of content or layout of pages. They are installed by uploading theme folders via FTP or SSH and applying them using the backend Admin system.

2.9 Templates



Templates

Templates...

Template systems handle HTML, JavaScript, and some PHP.

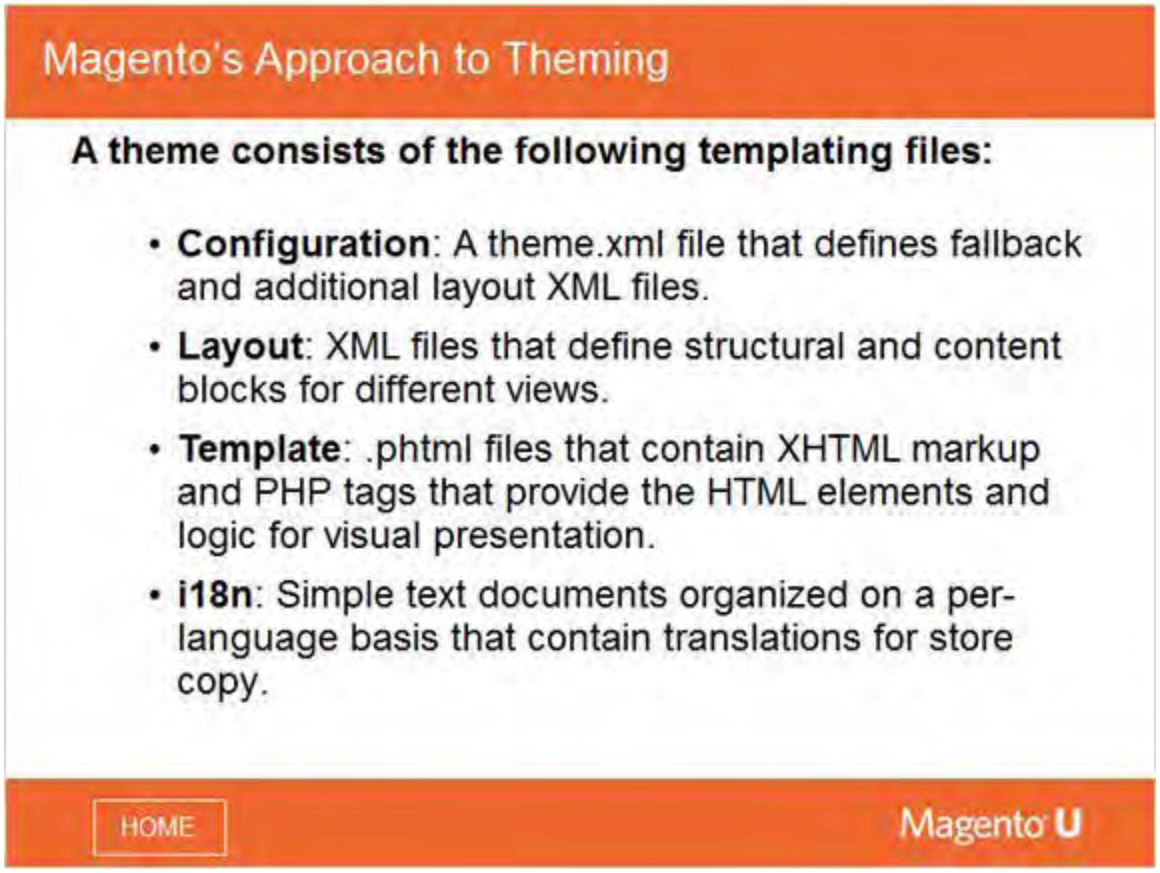
HOME
Magento U

Notes:

Templates in Magento are phtml files that contain HTML instructions.

In Magento 1, phtml is the only option; in Magento 2, it is possible to use any rendering system.

2.10 Magento's Approach to Theming



Magento's Approach to Theming

A theme consists of the following templating files:

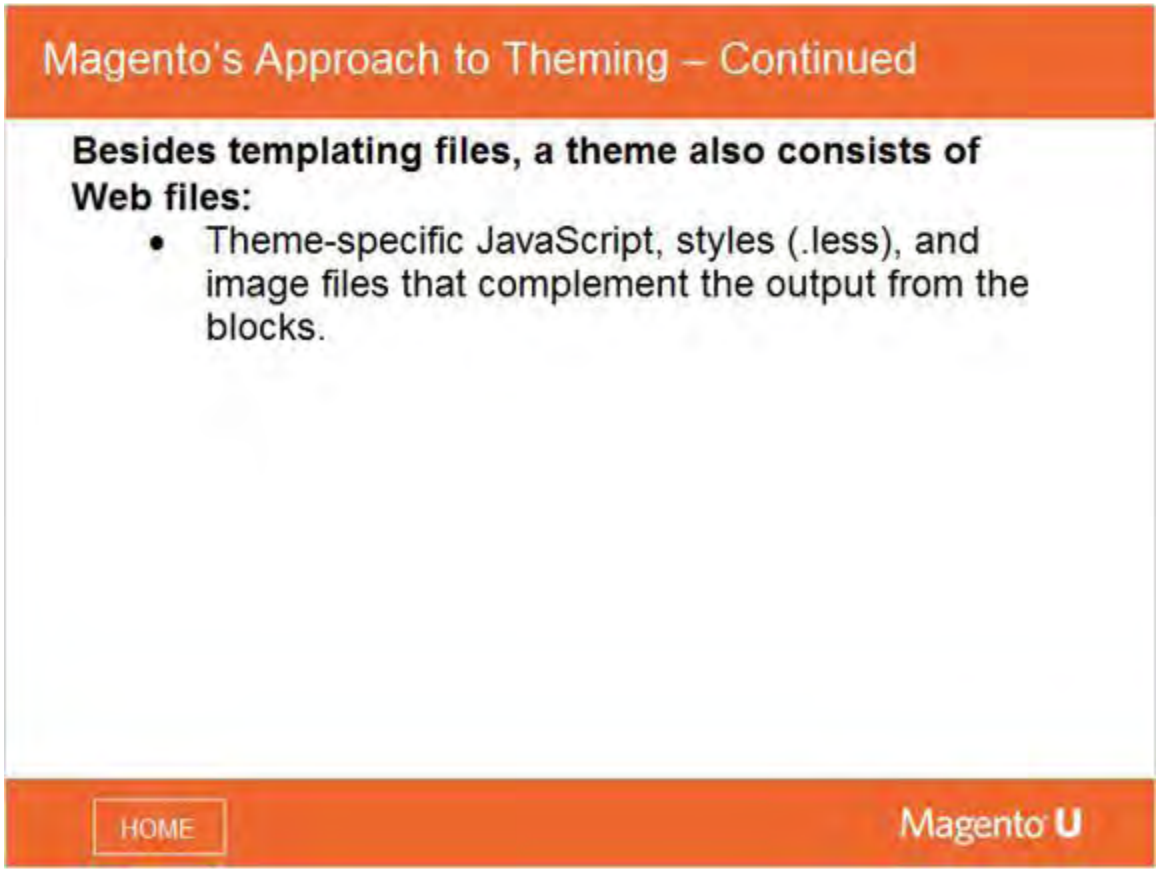
- **Configuration**: A theme.xml file that defines fallback and additional layout XML files.
- **Layout**: XML files that define structural and content blocks for different views.
- **Template**: .phtml files that contain XHTML markup and PHP tags that provide the HTML elements and logic for visual presentation.
- **i18n**: Simple text documents organized on a per-language basis that contain translations for store copy.

Notes:

This slide introduces the concept that **asset types are distinct from themes**.

Configuration, layout, template, and i18n files are only some of the files used in templates.

2.11 Magento's Approach to Theming - Continued



Magento's Approach to Theming – Continued

Besides templating files, a theme also consists of Web files:

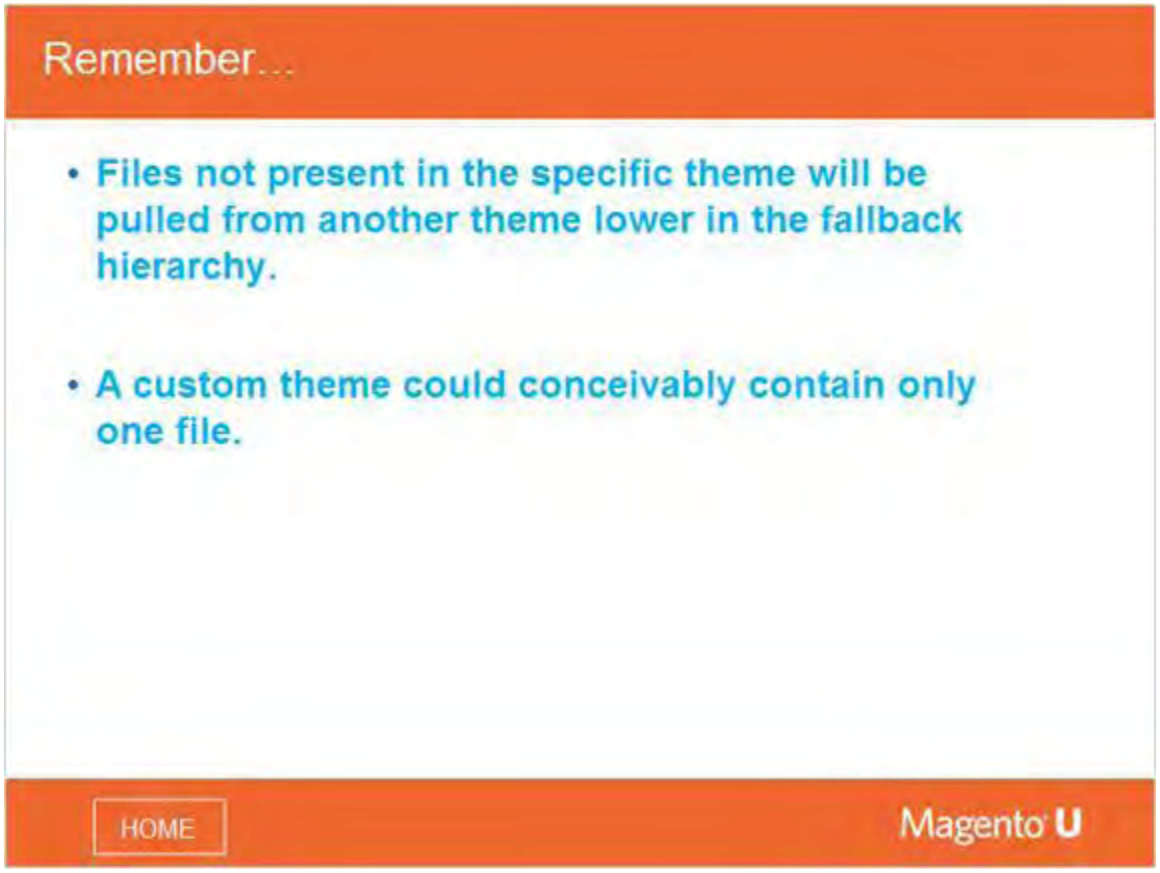
- Theme-specific JavaScript, styles (.less), and image files that complement the output from the blocks.

HOME
Magento U

Notes:

Web files include the static files (images, CSS, etc.) and files such as LESS files that create output files.

2.12 Remember...



Remember...

- Files not present in the specific theme will be pulled from another theme lower in the fallback hierarchy.
- A custom theme could conceivably contain only one file.

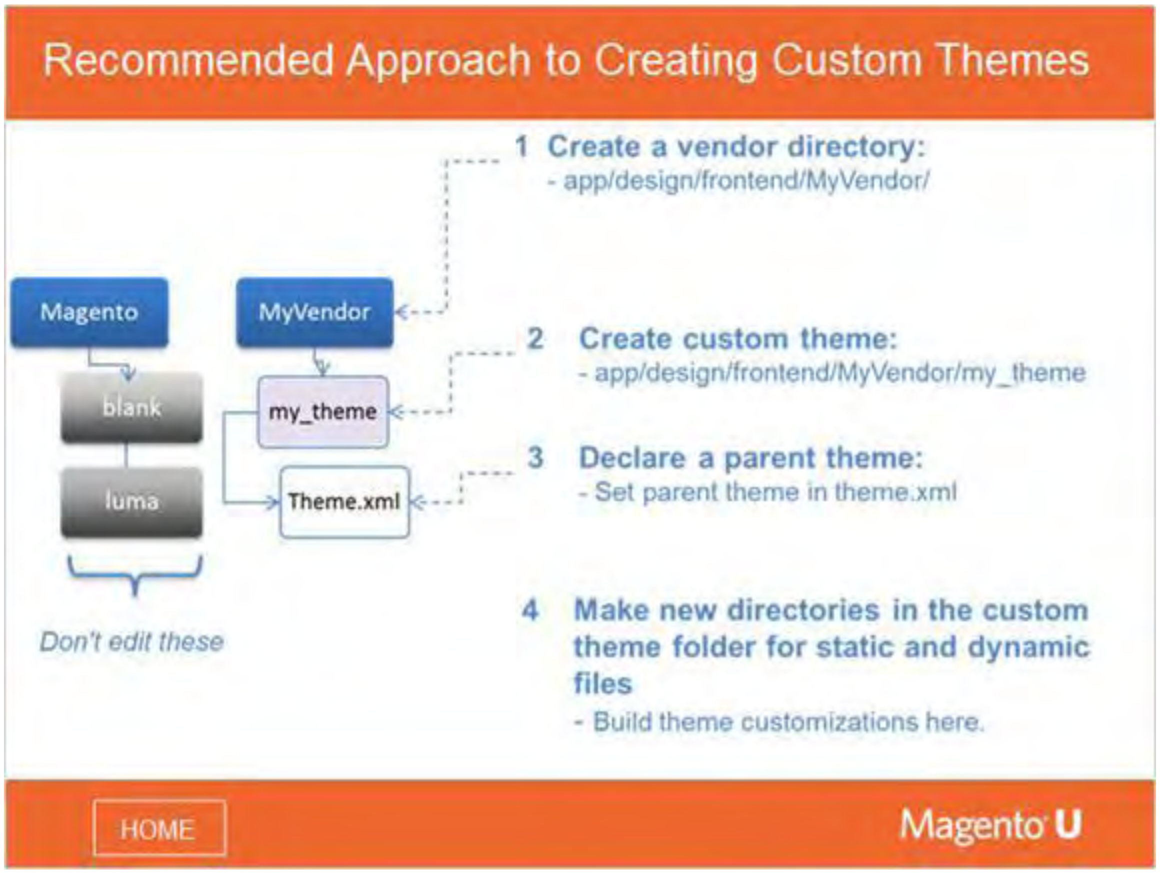
HOME
Magento U

Notes:

Remember:

- Files not present in the specific theme will be pulled from another theme lower in the fallback hierarchy.
- A custom theme could conceivably contain only one file.

2.13 Recommended Approach to Creating Custom Themes


Recommended Approach to Creating Custom Themes

1 **Create a vendor directory:**
- app/design/frontend/MyVendor/

2 **Create custom theme:**
- app/design/frontend/MyVendor/my_theme

3 **Declare a parent theme:**
- Set parent theme in theme.xml

4 **Make new directories in the custom theme folder for static and dynamic files**
- Build theme customizations here.

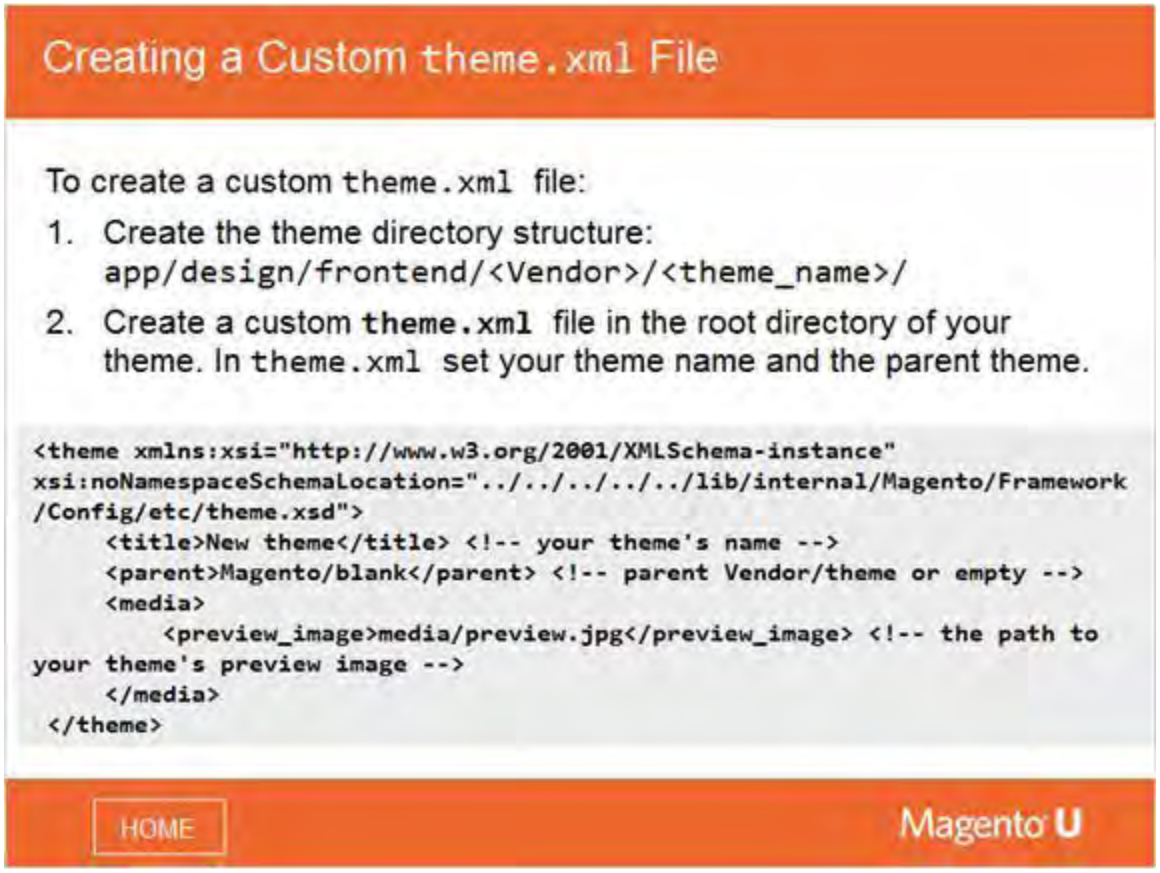
Notes:

Use this process to create your own custom themes.

How do the concepts from the previous slides apply to the setup on this slide?

Later, after you have viewed the module on fallback, come back and see if you can determine the fallback order from my_theme.

2.14 Creating a Custom theme.xml File



Creating a Custom `theme.xml` File

To create a custom `theme.xml` file:

1. Create the theme directory structure:
   `app/design/frontend/<Vendor>/<theme_name>/`
2. Create a custom **`theme.xml`** file in the root directory of your theme. In `theme.xml` set your theme name and the parent theme.

```
<theme xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../../../../../lib/internal/Magento/Framework
/Config/etc/theme.xsd">
     <title>New theme</title> <!-- your theme's name -->
     <parent>Magento/blank</parent> <!-- parent Vendor/theme or empty -->
     <media>
         <preview_image>media/preview.jpg</preview_image> <!-- the path to
your theme's preview image -->
     </media>
 </theme>
```

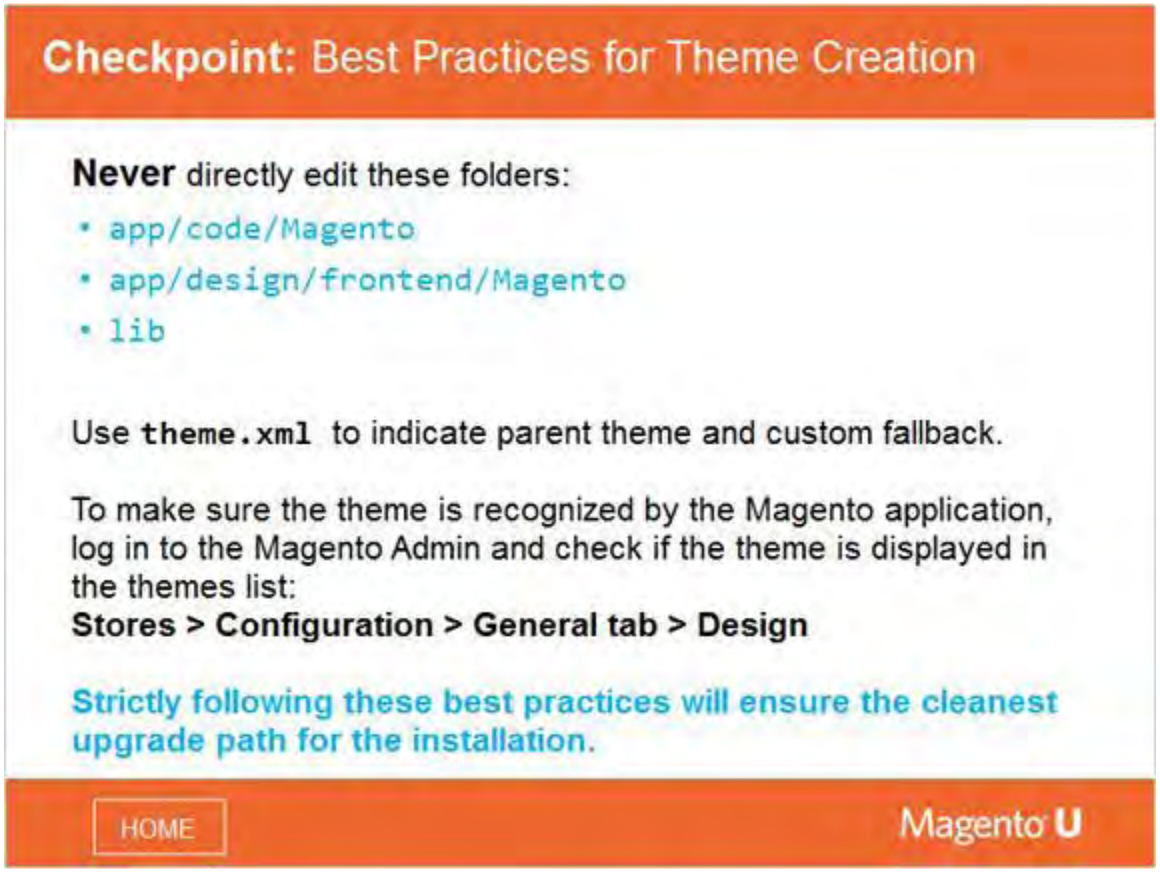
HOME

Magento U

Notes:

Here is an example of a custom theme.xml file. Use this slide as a comprehension check of the relevant concepts presented thus far.

2.15 Checkpoint: Best Practices for Theme Creation



Checkpoint: Best Practices for Theme Creation

Never directly edit these folders:

- `app/code/Magento`
- `app/design/frontend/Magento`
- `lib`

Use **`theme.xml`** to indicate parent theme and custom fallback.

To make sure the theme is recognized by the Magento application, log in to the Magento Admin and check if the theme is displayed in the themes list:
Stores > Configuration > General tab > Design

Strictly following these best practices will ensure the cleanest upgrade path for the installation.

Notes:

Be sure that you understand these concepts before going on.

3. Module 3 Static Files

3.1 Module 3 Static Files

Notes:

In this module, we'll take a closer look at static files.

Static files are separate from the layout and template files that refer to them.

3.2 Module Topics

Module Topics

In this module, we will introduce...

- Static file types: CSS, less, fonts, images, js files

HOME
Magento U

Notes:

In this module, we present the different static file types, where they can be found, and describe how they are separate from layout and templates.

3.3 Assets

Assets

Assets are all of the files used within a theme…

-- and yet --

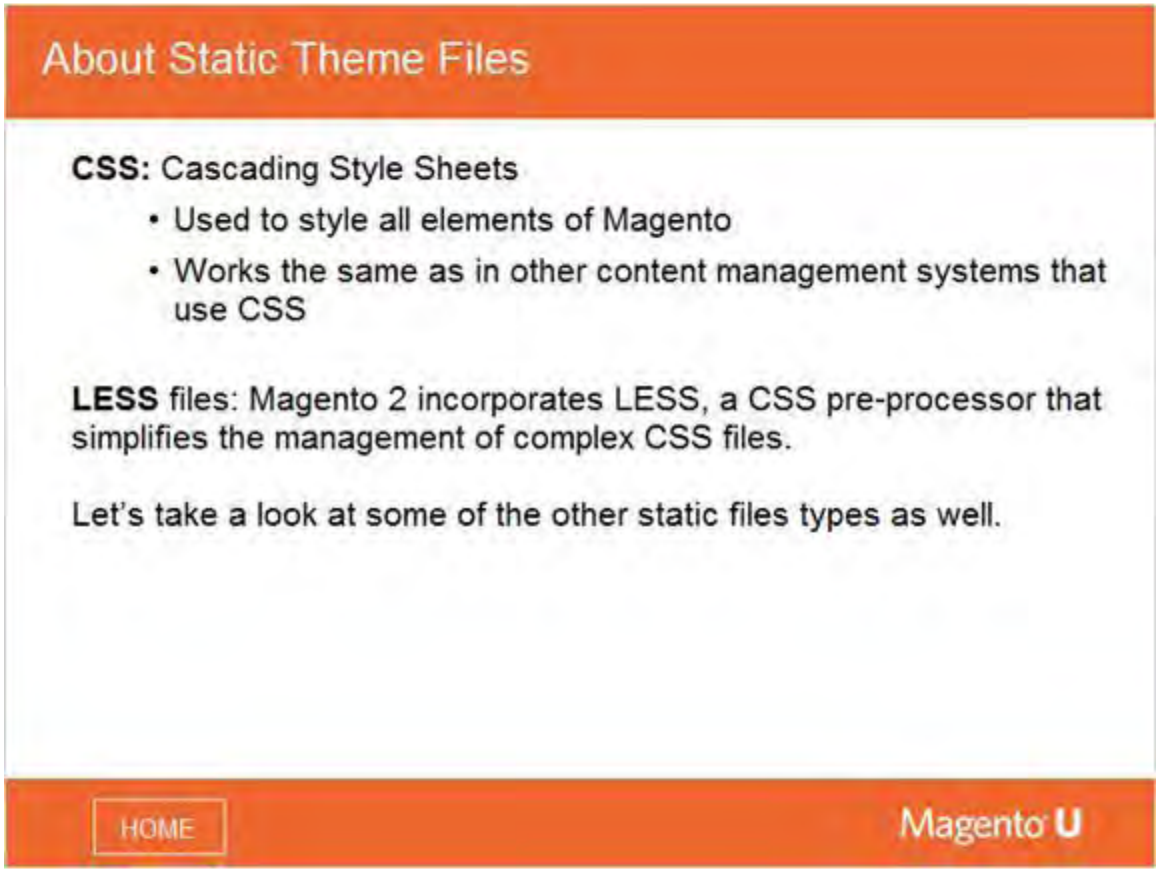
Assets are distinct from themes.

Assets…
HOME
Magento U

Notes:

Assets are all of the files used within a theme and yet…
…assets are distinct from themes.

3.4 About Static Theme Files



About Static Theme Files

CSS: Cascading Style Sheets

- Used to style all elements of Magento
- Works the same as in other content management systems that use CSS

LESS files: Magento 2 incorporates LESS, a CSS pre-processor that simplifies the management of complex CSS files.

Let's take a look at some of the other static files types as well.

HOME
Magento U

Notes:

Static files are separate from the layout and template files that refer to them. Commonly used static file types include image files, font files, CSS files, LESS files, and so on.

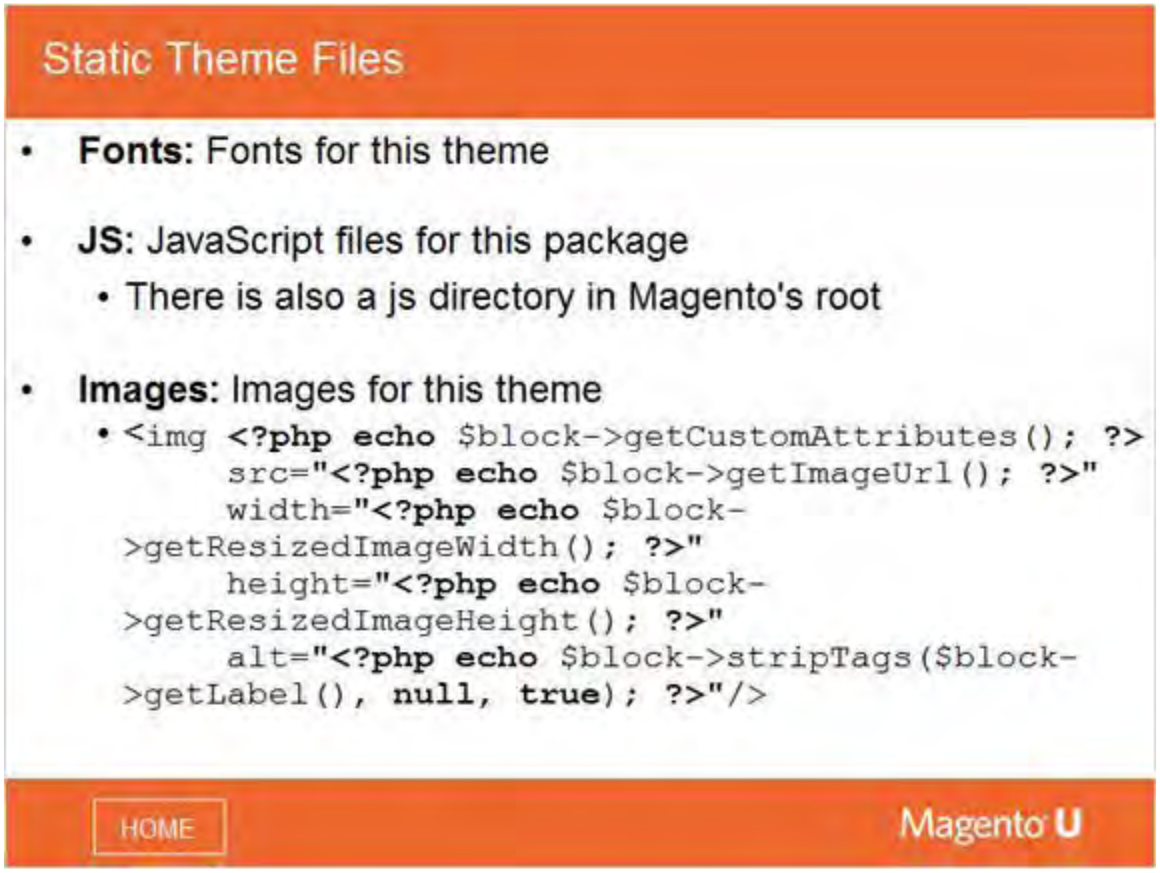
To define styles of a Magento store, you can use both CSS and LESS stylesheets.

CSS files are stylesheets in a Magento theme should contain relative paths

SASS and LESS are CSS authoring techniques and technologies that are relatively recent developments in the world of web design, and as such are not covered in this class. Participants wishing to use these will need to refer to examples online.

We will discuss more about LESS in another unit of this course.

3.5 Static Theme Files



Static Theme Files

- **Fonts:** Fonts for this theme
- **JS:** JavaScript files for this package
  - There is also a js directory in Magento's root
- **Images:** Images for this theme

  -
    ```
    <img <?php echo $block->getCustomAttributes(); ?>
            src="<?php echo $block->getImageUrl(); ?>"
            width="<?php echo $block-
    >getResizedImageWidth(); ?>"
            height="<?php echo $block-
    >getResizedImageHeight(); ?>"
            alt="<?php echo $block->stripTags($block-
    >getLabel(), null, true); ?>"/>
    ```

HOME

Magento U

Notes:

There are two general locations for JavaScript in a Magento file system:

- ./js files are for custom and third-party JavaScript libraries or scripts
- JavaScript files that are theme-specific belong in theme JS directories, especially those that use DOM selectors (refer to opcheckout.js).

Theme-related images should be stored in the theme, especially the following:

- Images referenced relatively in CSS and JavaScript files
- Images that contain text that may need translating

All files must be explicitly added for the necessary links and tags to be rendered - they are not added automatically simply by just existing.

3.6 Where to Find Static Theme Files

Where to Find Static Theme Files

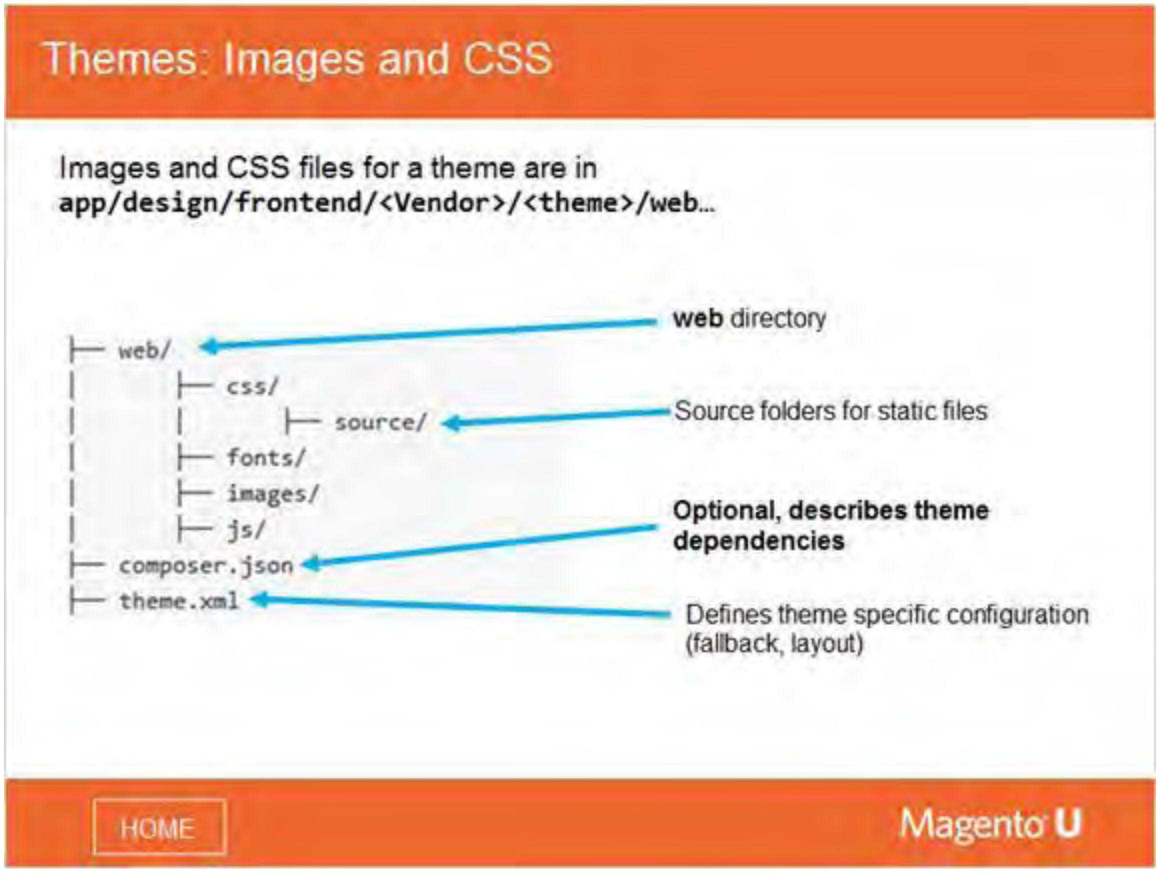
Images belonging to entity data (products, pages, etc.) and uploaded through the Admin will be found in subfolders in the **./pub/media/** directory.

HOME
Magento U

Notes:

Images belonging to entity data (such as products, pages, etc.) and uploaded through the admin will be found in subfolders in the **./pub/media/** directory.

3.7 Themes: Images and CSS Files



Themes: Images and CSS

Images and CSS files for a theme are in
`app/design/frontend/<Vendor>/<theme>/web…`

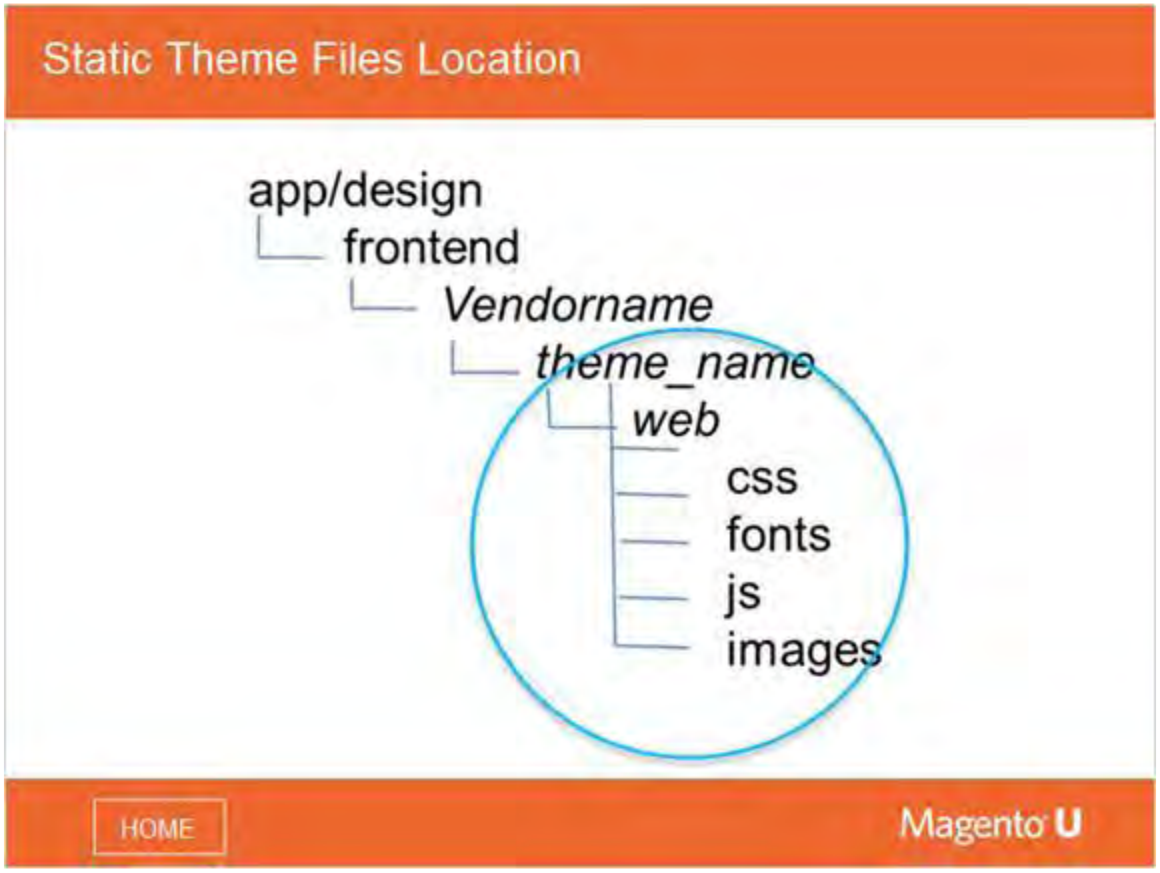
```
├─ web/
|       ├─ css/
|       |       ├─ source/
|       ├─ fonts/
|       ├─ images/
|       ├─ js/
├─ composer.json
├─ theme.xml
```

web directory
Source folders for static files
Optional, describes theme dependencies
Defines theme specific configuration (fallback, layout)
HOME
Magento U

Notes:

Here are the folders used for static files.

3.8 Static Theme Files Location



Static Theme Files Location

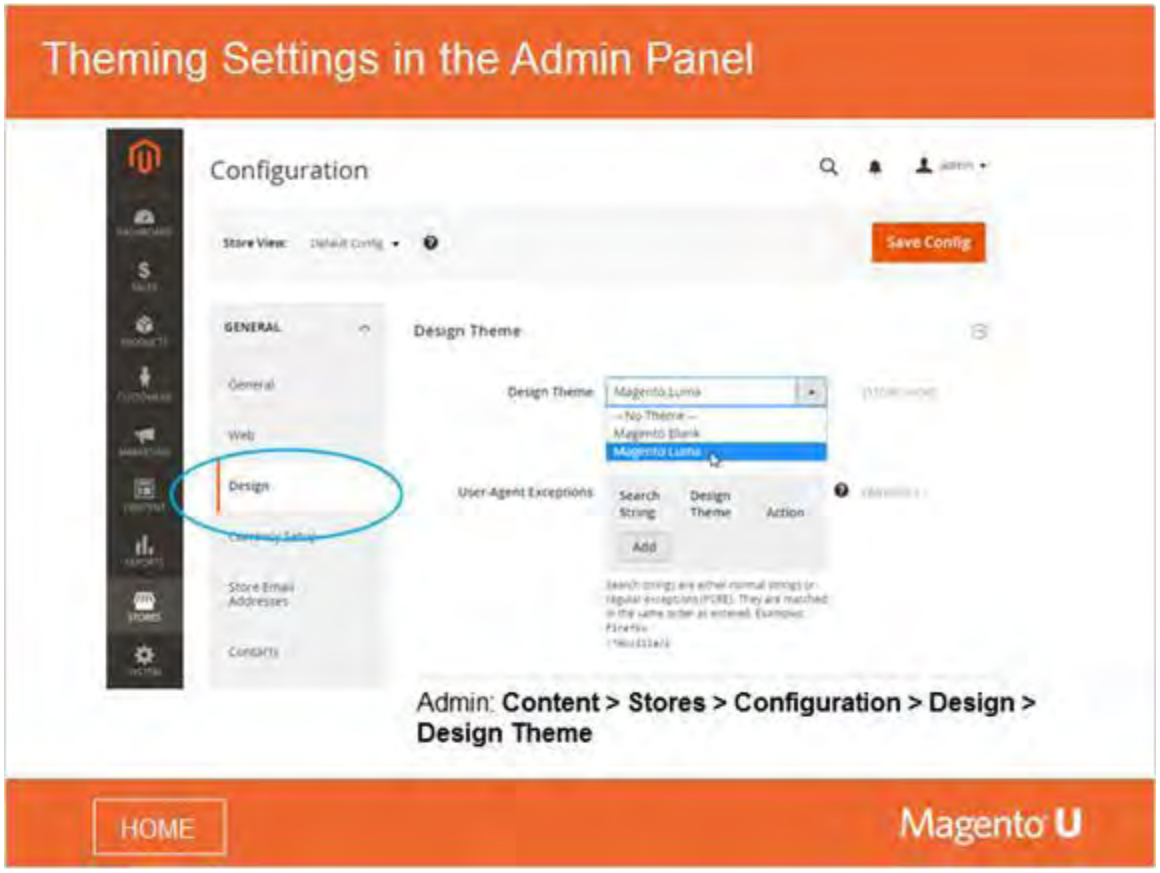
```
app/design
└── frontend
    └── Vendorname
        └── theme_name
            └── web
                ├── css
                ├── fonts
                ├── js
                └── images
```

HOME
Magento U

Notes:

Here is where you'll find the static files.

3.9 Theming Settings in the Admin Panel


Theming Settings in the Admin Panel

Admin: Content > Stores > Configuration > Design > Design Theme

Notes:

Content > Design > Themes displays a list of themes where you can view theme details and a preview.

To apply a theme:

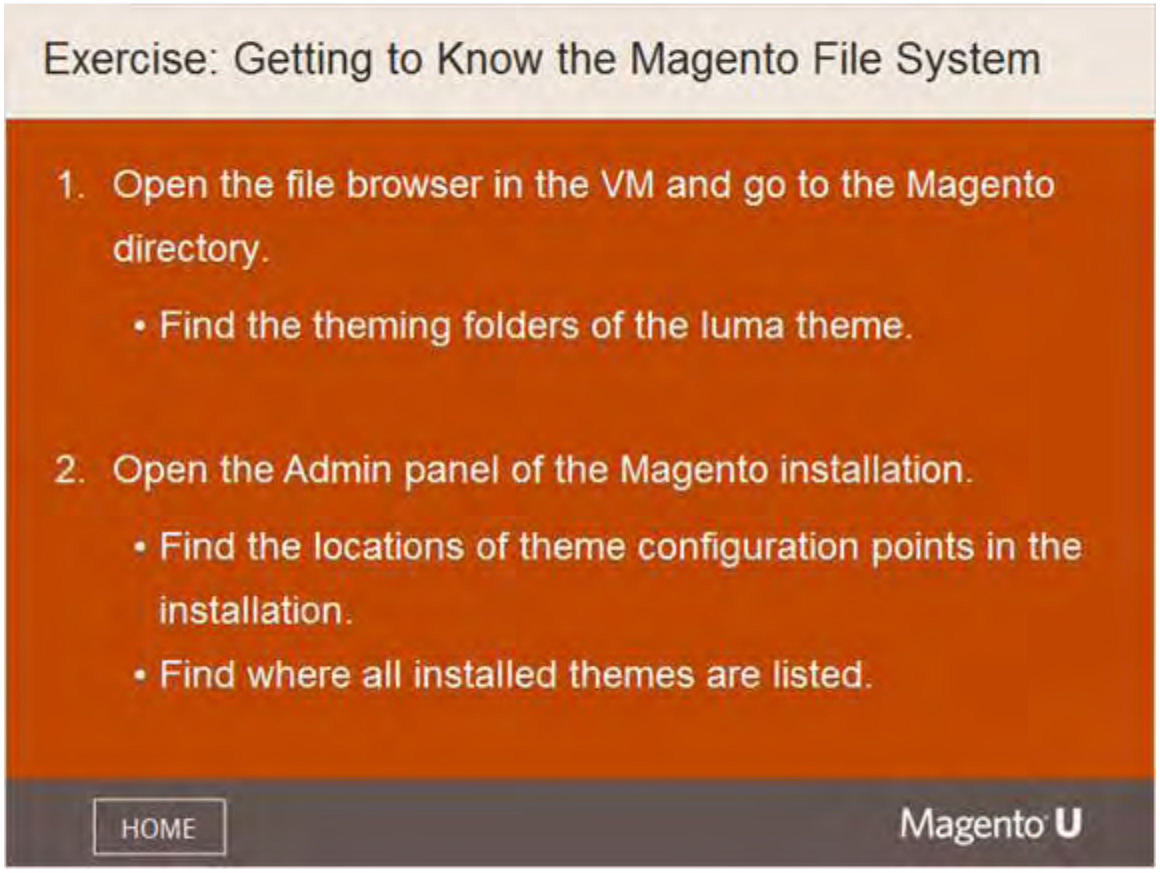
1. Go to **Stores** > **Configuration** > **Design**.
2. In the **Scope** drop-down field, select the store view where you want to apply the theme.
3. On the **Design Theme** tab, select your newly created theme in the **Design Theme** drop-down.
4. Click **Save Config**.
5. To see your changes applied, reload the storefront pages.

Stores > Configuration is where many things affecting functionality and storefront display are set.

- In the **Store View** drop-down field, select your website.
- On the **Design Theme** tab select your theme.

Values saved here are stored in the core_config_data table. We will explain more about these fields later in the course. We will also cover fallback and theme hierarchy later.

3.10 Exercise: Getting to Know the Magento File System



Exercise: Getting to Know the Magento File System

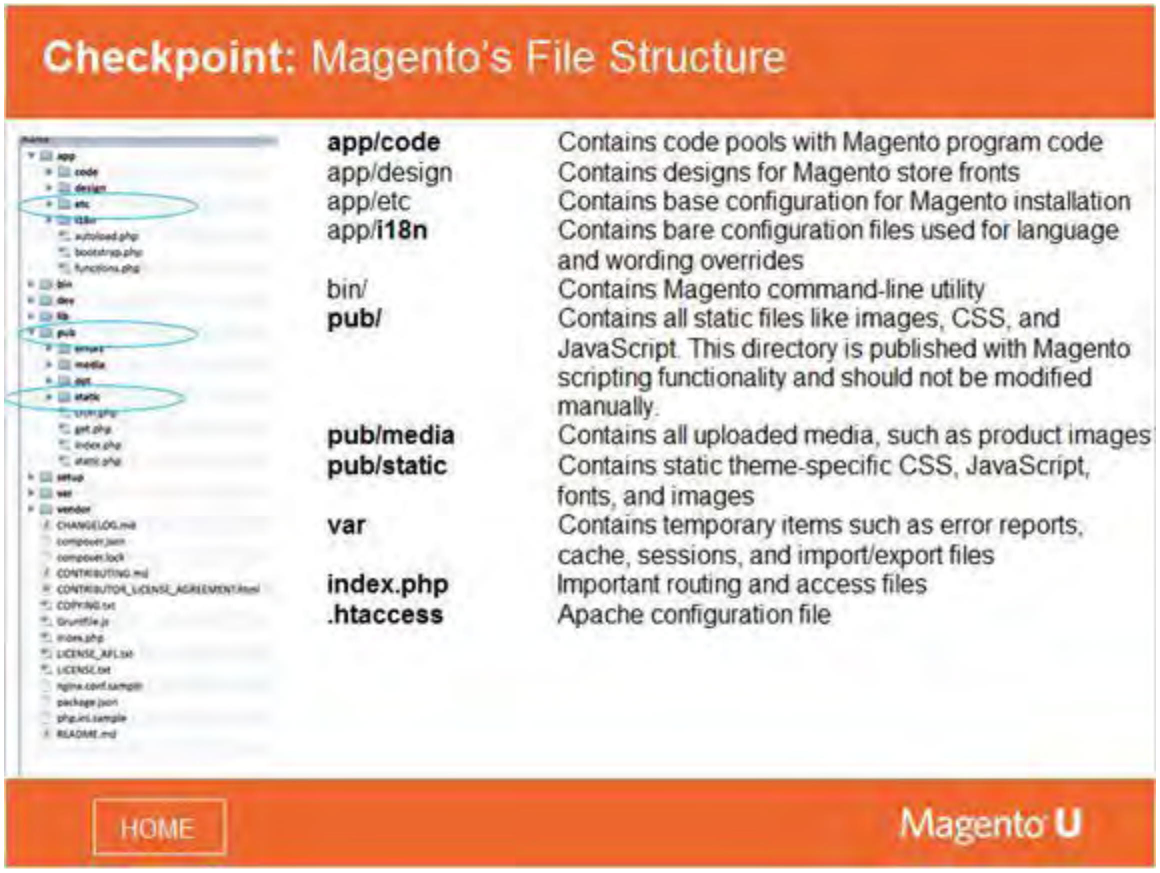
1. Open the file browser in the VM and go to the Magento directory.
   - Find the theming folders of the luma theme.
2. Open the Admin panel of the Magento installation.
   - Find the locations of theme configuration points in the installation.
   - Find where all installed themes are listed.

HOME
Magento U

Notes:

To complete this exercise, follow the instructions on the slide.

3.11 Checkpoint: Magento's File Structure



Checkpoint: Magento's File Structure

```
Name
▼ app
  ▸ code
  ▸ design
  ▸ etc
  ▸ i18n
    autoload.php
    bootstrap.php
    functions.php
▸ bin
▸ dev
▸ lib
▼ pub
  ▸ errors
  ▸ media
  ▸ opt
  ▸ static
    cron.php
    get.php
    index.php
    static.php
▸ setup
▸ var
▸ vendor
  CHANGELOG.md
  composer.json
  composer.lock
  CONTRIBUTING.md
  CONTRIBUTOR_LICENSE_AGREEMENT.html
  COPYING.txt
  Gruntfile.js
  index.php
  LICENSE_AFL.txt
  LICENSE.txt
  nginx.conf.sample
  package.json
  php.ini.sample
  README.md
```

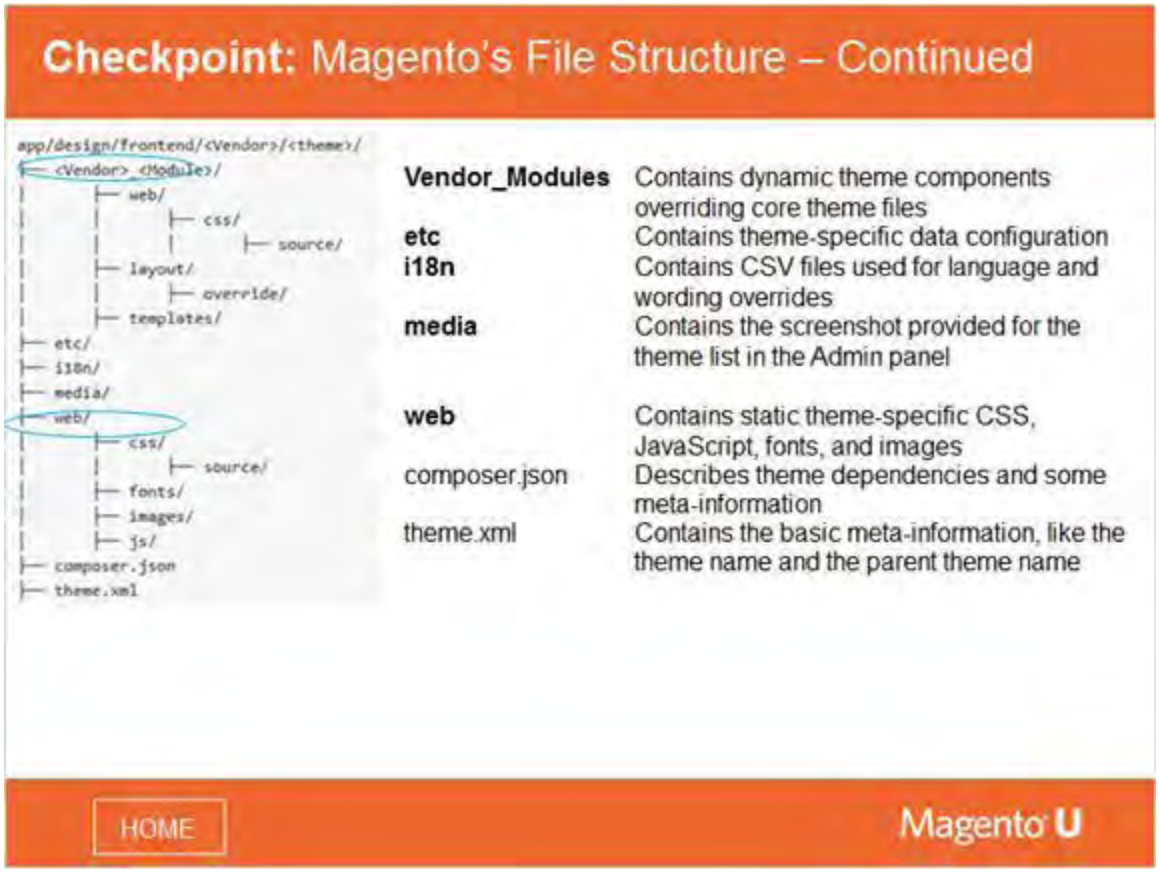
app/code	Contains code pools with Magento program code
app/design	Contains designs for Magento store fronts
app/etc	Contains base configuration for Magento installation
app/**i18n**	Contains bare configuration files used for language and wording overrides
bin/	Contains Magento command-line utility
pub/	Contains all static files like images, CSS, and JavaScript. This directory is published with Magento scripting functionality and should not be modified manually.
pub/media	Contains all uploaded media, such as product images
pub/static	Contains static theme-specific CSS, JavaScript, fonts, and images
var	Contains temporary items such as error reports, cache, sessions, and import/export files
index.php	Important routing and access files
.htaccess	Apache configuration file

Notes:

Two things to know on Magento’s file structure are:

- Magento has two general types of XML: config and layout. The app/etc folder contains config XML, and layout XML is located under a theme.
- If you do not see the .htaccess file in your web root, you must show hidden files. While the hotkeys for doing this depend on context, it's generally Control+H or Cmd+H.

3.12 Checkpoint: Magento's File Structure - Continued



Checkpoint: Magento's File Structure – Continued

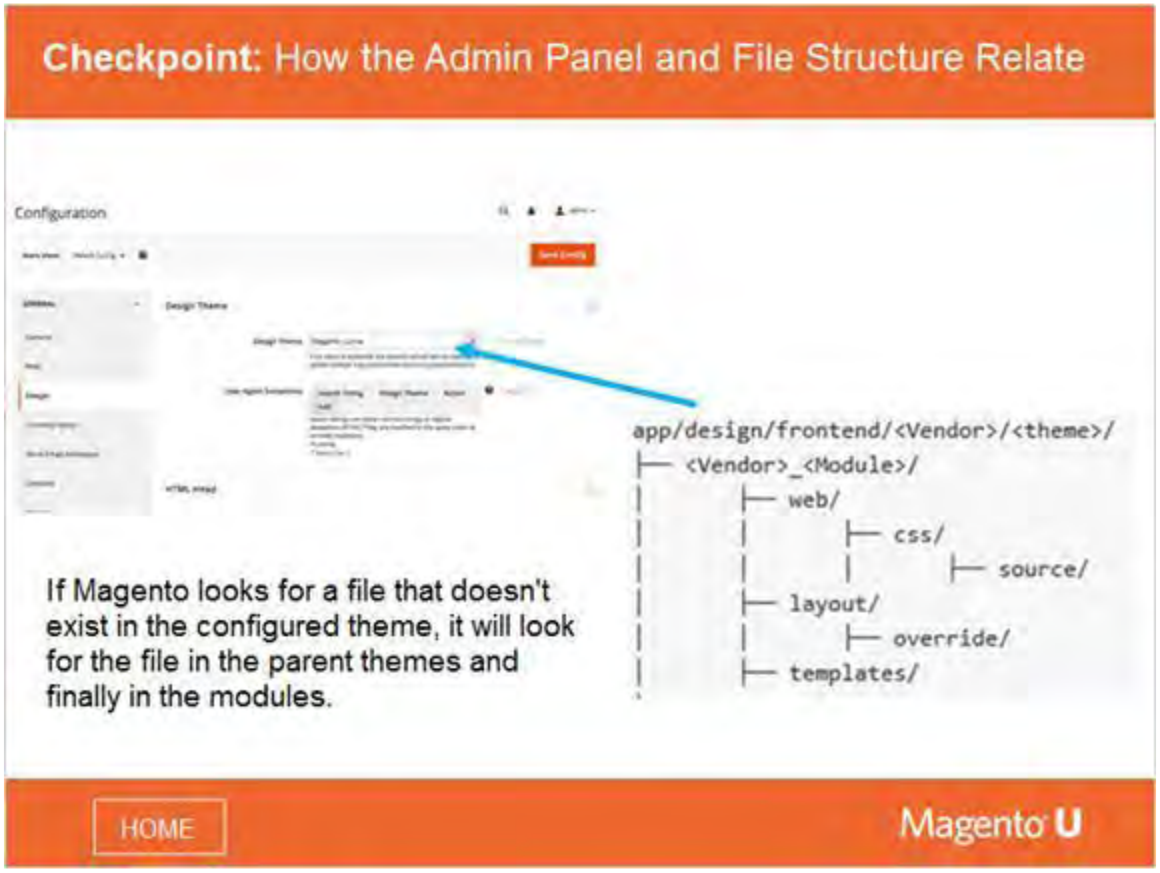
```
app/design/frontend/<Vendor>/<theme>/
├── <Vendor>_<Module>/
|       ├── web/
|       |       ├── css/
|       |       |       ├── source/
|       ├── layout/
|       |       ├── override/
|       ├── templates/
├── etc/
├── i18n/
├── media/
├── web/
|       ├── css/
|       |       ├── source/
|       ├── fonts/
|       ├── images/
|       ├── js/
├── composer.json
├── theme.xml
```

Vendor_Modules	Contains dynamic theme components overriding core theme files
etc	Contains theme-specific data configuration
i18n	Contains CSV files used for language and wording overrides
media	Contains the screenshot provided for the theme list in the Admin panel
web	Contains static theme-specific CSS, JavaScript, fonts, and images
composer.json	Describes theme dependencies and some meta-information
theme.xml	Contains the basic meta-information, like the theme name and the parent theme name

Notes:

You will want to become familiar with this structure so you know where to find specific types of files.

3.13 Checkpoint: How the Admin Panel and File Structure Relate



Checkpoint: How the Admin Panel and File Structure Relate

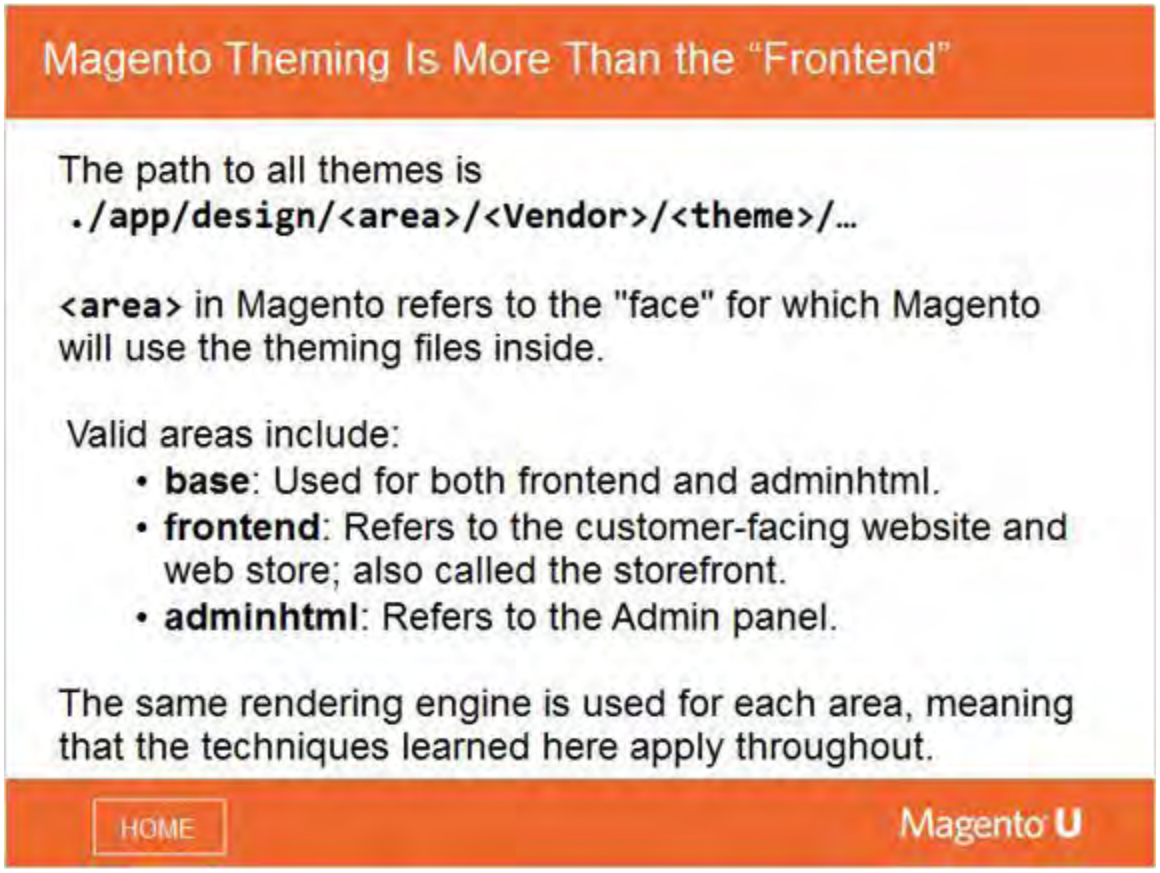
```
app/design/frontend/<Vendor>/<theme>/
├─ <Vendor>_<Module>/
|        ├─ web/
|        |        ├─ css/
|        |        |        ├─ source/
|        ├─ layout/
|        |        ├─ override/
|        ├─ templates/
```

If Magento looks for a file that doesn't exist in the configured theme, it will look for the file in the parent themes and finally in the modules.

Notes:

It's important to review the information on the slide to be sure that you understand the relationship between config settings and the file system. Clarifying this relationship now will help you understand more complex concepts later on, such as fallback.

3.14 Theming Is More Than the "Frontend"



Magento Theming Is More Than the "Frontend"

The path to all themes is
`./app/design/<area>/<Vendor>/<theme>/…`

`<area>` in Magento refers to the "face" for which Magento will use the theming files inside.

Valid areas include:

- **base**: Used for both frontend and adminhtml.
- **frontend**: Refers to the customer-facing website and web store; also called the storefront.
- **adminhtml**: Refers to the Admin panel.

The same rendering engine is used for each area, meaning that the techniques learned here apply throughout.

Notes:

There are other design areas besides "frontend" (adminhtml and base) which, though they are not covered by this course, still use the same theming engine.

For example, the Admin theme can be customized for clients using the knowledge you will gain from this course.

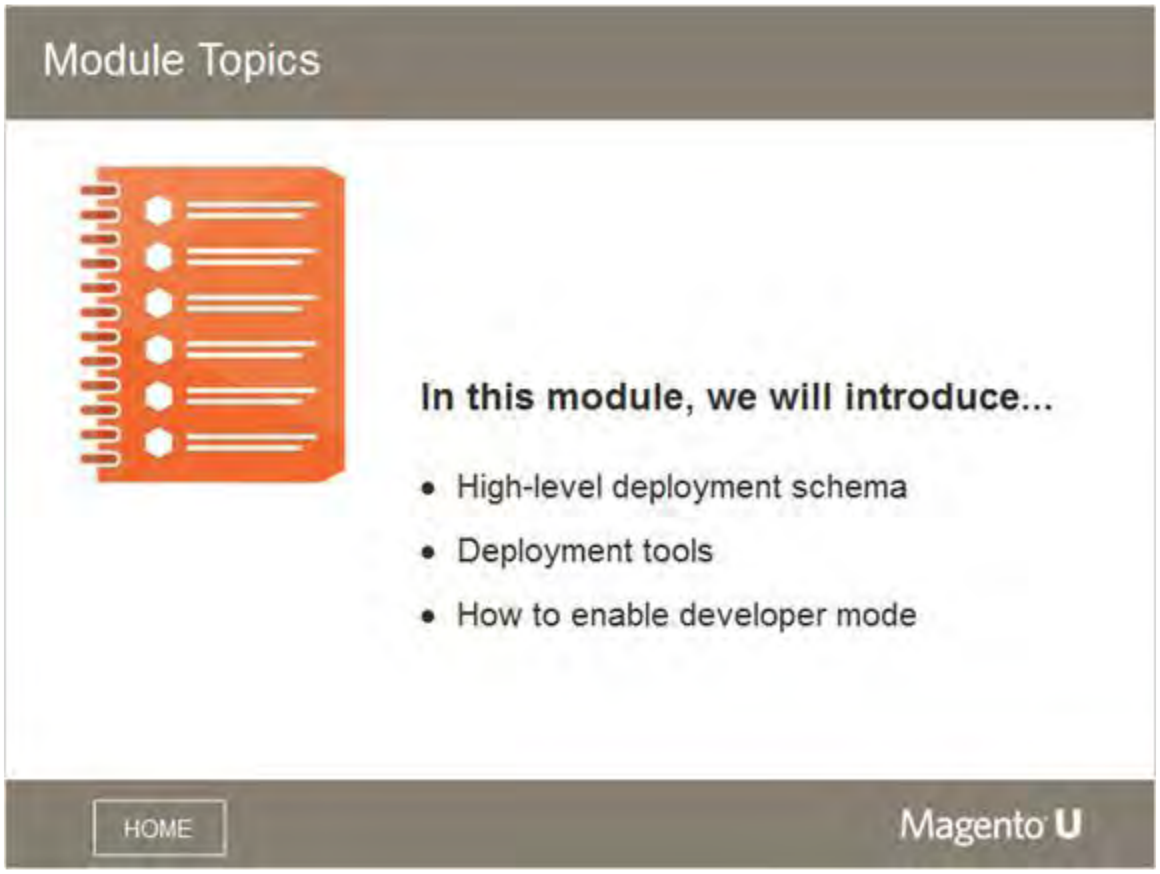
4. Module 4 Deployment

4.1 Module 4 Deployment

Notes:

Once you create your content, it’s time for deployment.

4.2 Module Topics



Module Topics

In this module, we will introduce...

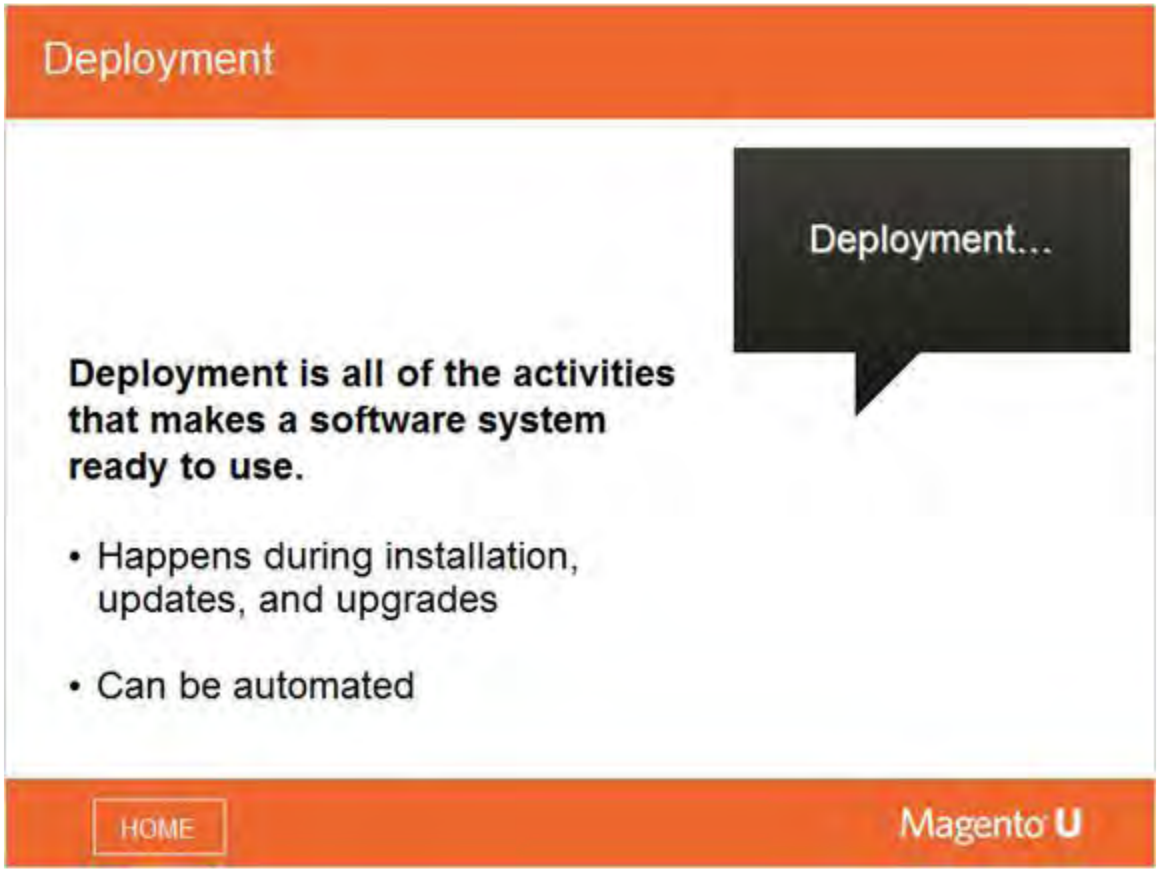
- High-level deployment schema
- Deployment tools
- How to enable developer mode

HOME
Magento U

Notes:

In this module, we present the high-level deployment process and the tools used.

4.3 Deployment



Deployment

Deployment is all of the activities that makes a software system ready to use.

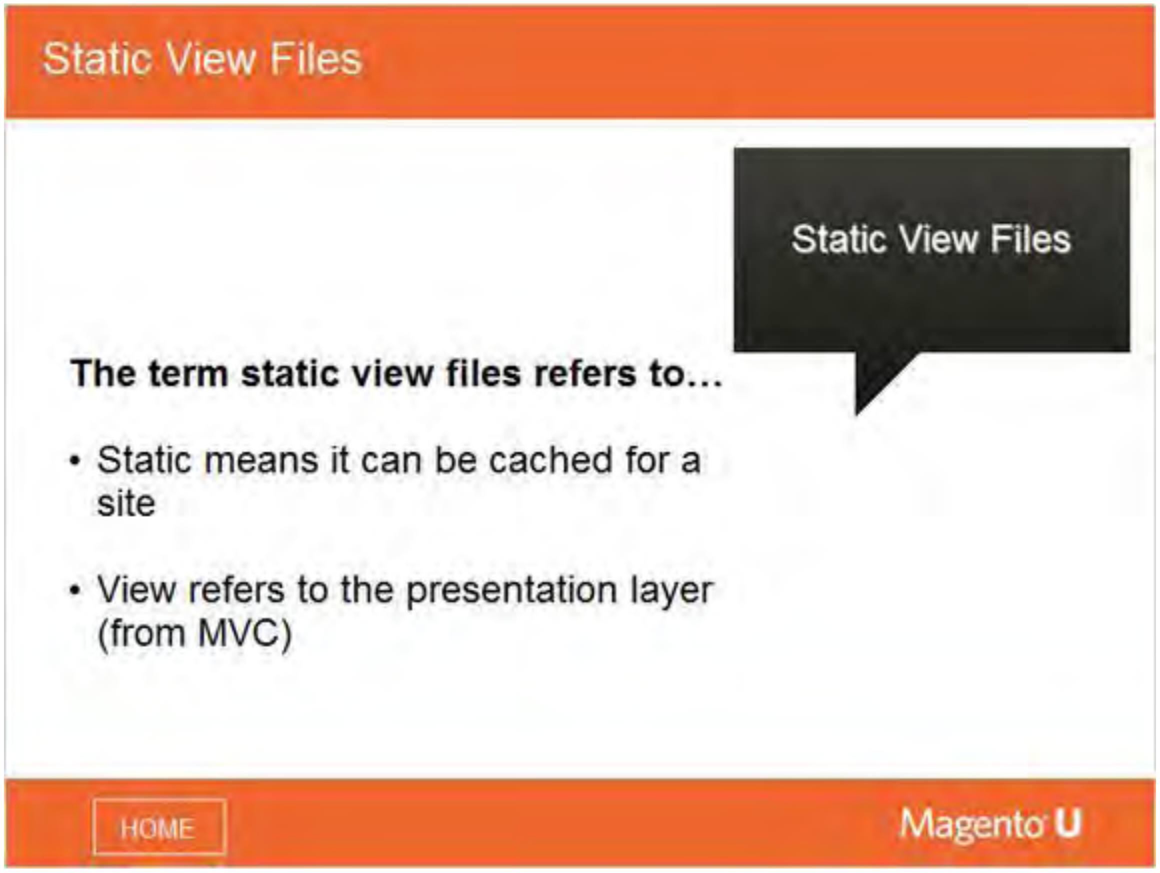
- Happens during installation, updates, and upgrades
- Can be automated

Deployment...
HOME
Magento U

Notes:

Website deployment is moving a website from a local environment to live servers. There are tools that can make the process easier and faster.

4.4 Static View Files



Static View Files

The term static view files refers to…

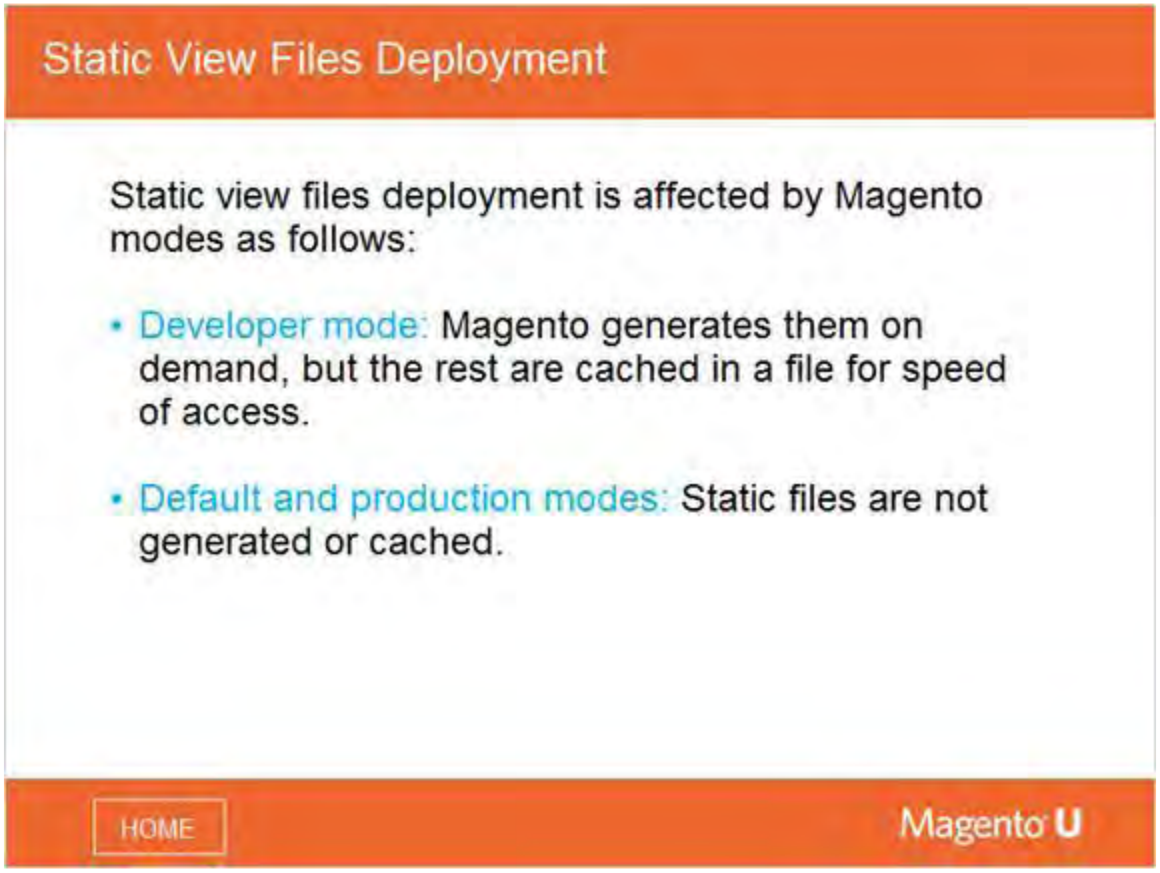
- Static means it can be cached for a site
- View refers to the presentation layer (from MVC)

Notes:

The static view files deployment command allows you to write static files to the Magento file system when the Magento software is set for production mode.

Static view files are located in the <your Magento install dir>/pub/static directory, and some are cached in the <your Magento install dir>/var/view_preprocessed directory as well.

4.5 Static View Files Deployment



Static View Files Deployment

Static view files deployment is affected by Magento modes as follows:

- Developer mode: Magento generates them on demand, but the rest are cached in a file for speed of access.
- Default and production modes: Static files are not generated or cached.

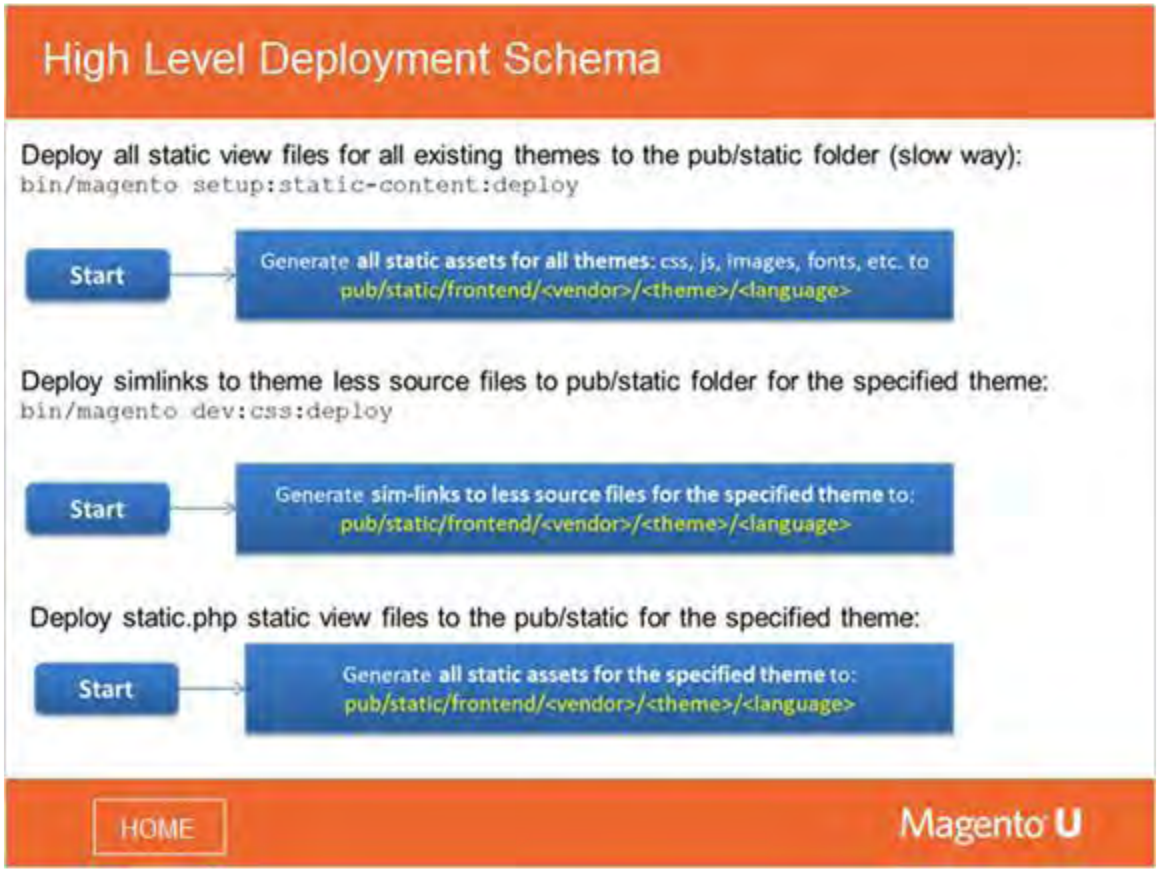
HOME
Magento U

Notes:

Static view files deployment is affected by Magento modes, as shown here.

You m**ust w**rite static view files to the Magento file system manually using the commands discussed in this module. After that, you can restrict permissions to limit your vulnerabilities and to prevent accidental or malicious overwriting of files.

4.6 High Level Deployment Schema



High Level Deployment Schema

Deploy all static view files for all existing themes to the pub/static folder (slow way):

```
bin/magento setup:static-content:deploy
```

Start
Generate all static assets for all themes: css, js, images, fonts, etc. to pub/static/frontend/<vendor>/<theme>/<language>

Deploy simlinks to theme less source files to pub/static folder for the specified theme:

```
bin/magento dev:css:deploy
```

Start
Generate sim-links to less source files for the specified theme to: pub/static/frontend/<vendor>/<theme>/<language>

Deploy static.php static view files to the pub/static for the specified theme:

Start
Generate all static assets for the specified theme to: pub/static/frontend/<vendor>/<theme>/<language>
HOME
Magento U

Notes:

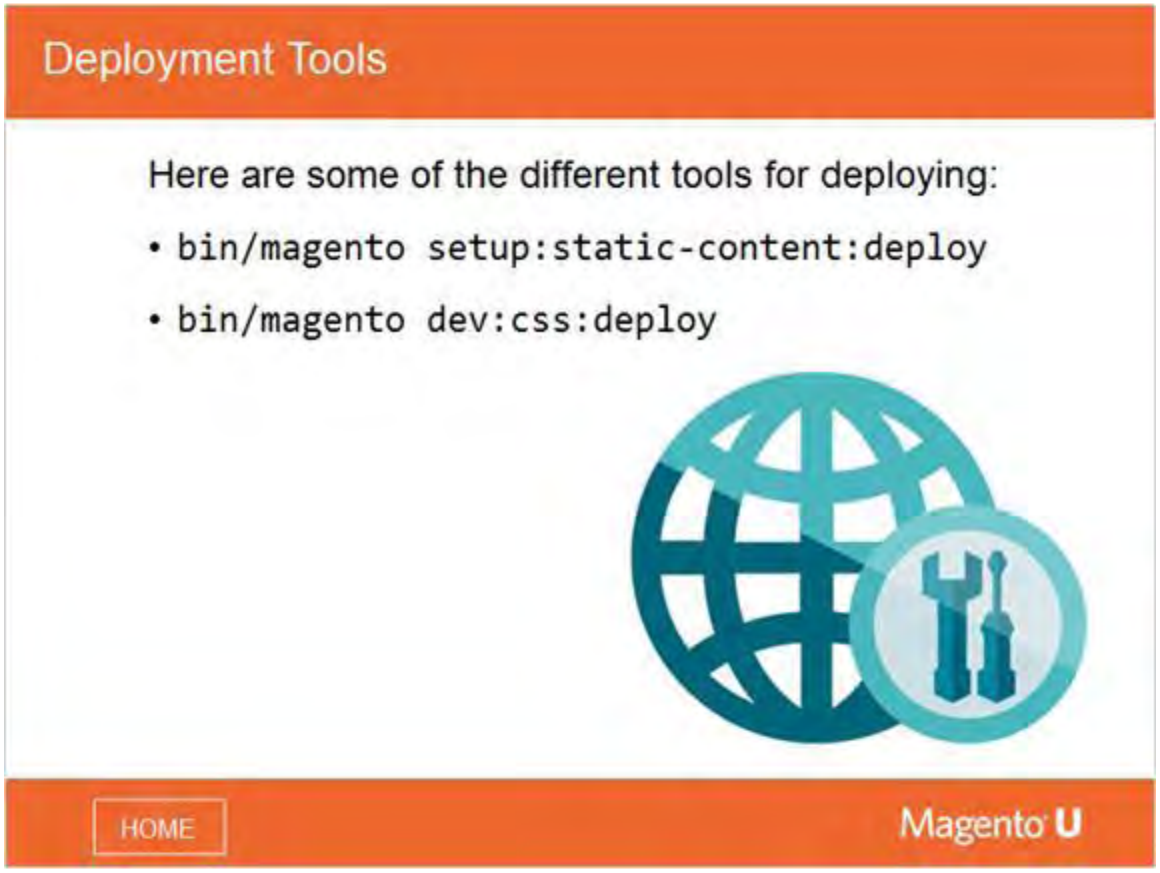
You can deploy files for a specific theme, for all existing themes, or you can deploy the simlinks to theme LESS source files. We cover LESS in more detail in a later unit of this course.

Use the first command shown here to generate the static view files for all existing themes to the pub/static folder. Note that this is the slowest method because it updates all files for all existing themes.

Use the second command to deploy simlinks to theme less source files. They are deployed to pub/static folder for the specified theme.

Use the third command to deploy static view files to pub/static for the specified theme.

4.7 Deployment Tools



Deployment Tools

Here are some of the different tools for deploying:

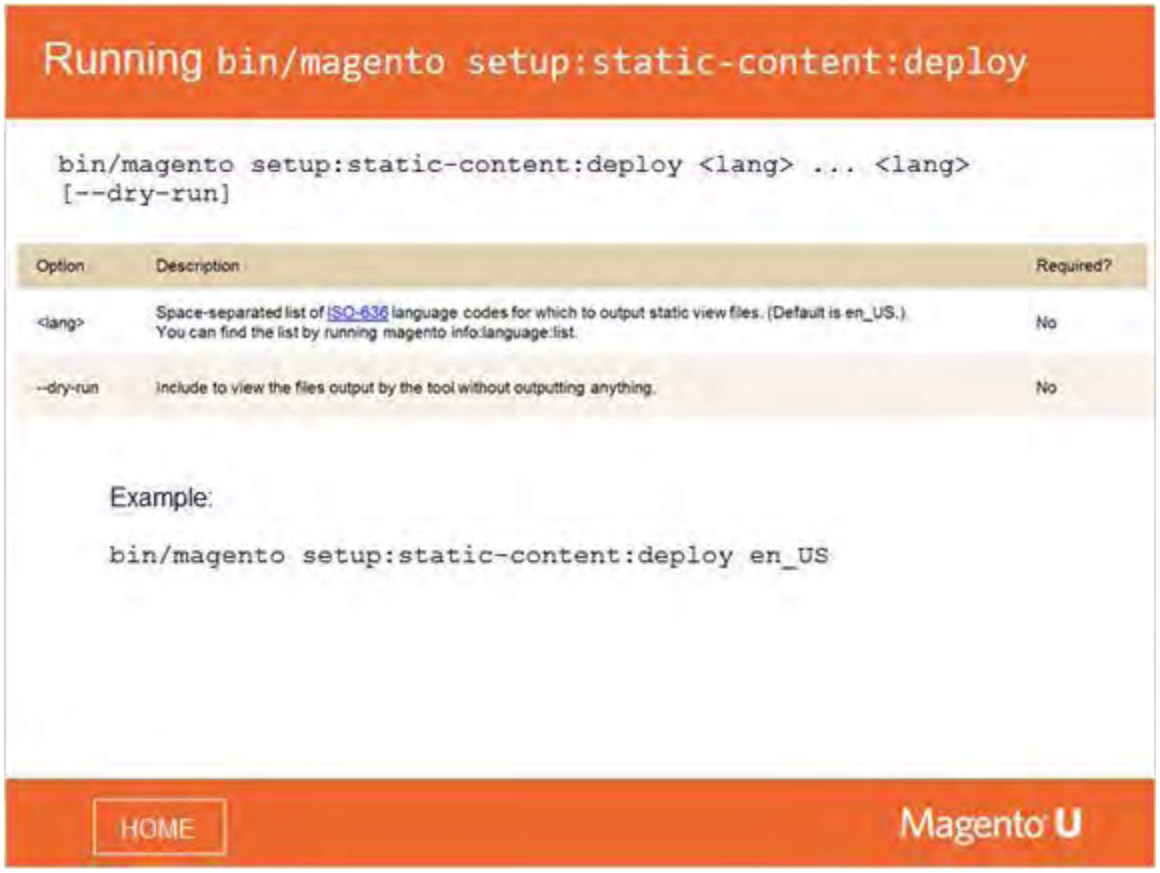
- `bin/magento setup:static-content:deploy`
- `bin/magento dev:css:deploy`

HOME
Magento U

Notes:

Let's take a closer look at these tools. You saw them in the previous slide, but we'll take a look at the available options for each one.

4.8 Running bin/magento setup:static-content:deploy



Running `bin/magento setup:static-content:deploy`

```
bin/magento setup:static-content:deploy <lang> ... <lang>
[--dry-run]
```

Option	Description	Required?
<lang>	Space-separated list of ISO-636 language codes for which to output static view files. (Default is en_US.) You can find the list by running magento info:language:list.	No
--dry-run	Include to view the files output by the tool without outputting anything.	No

Example:

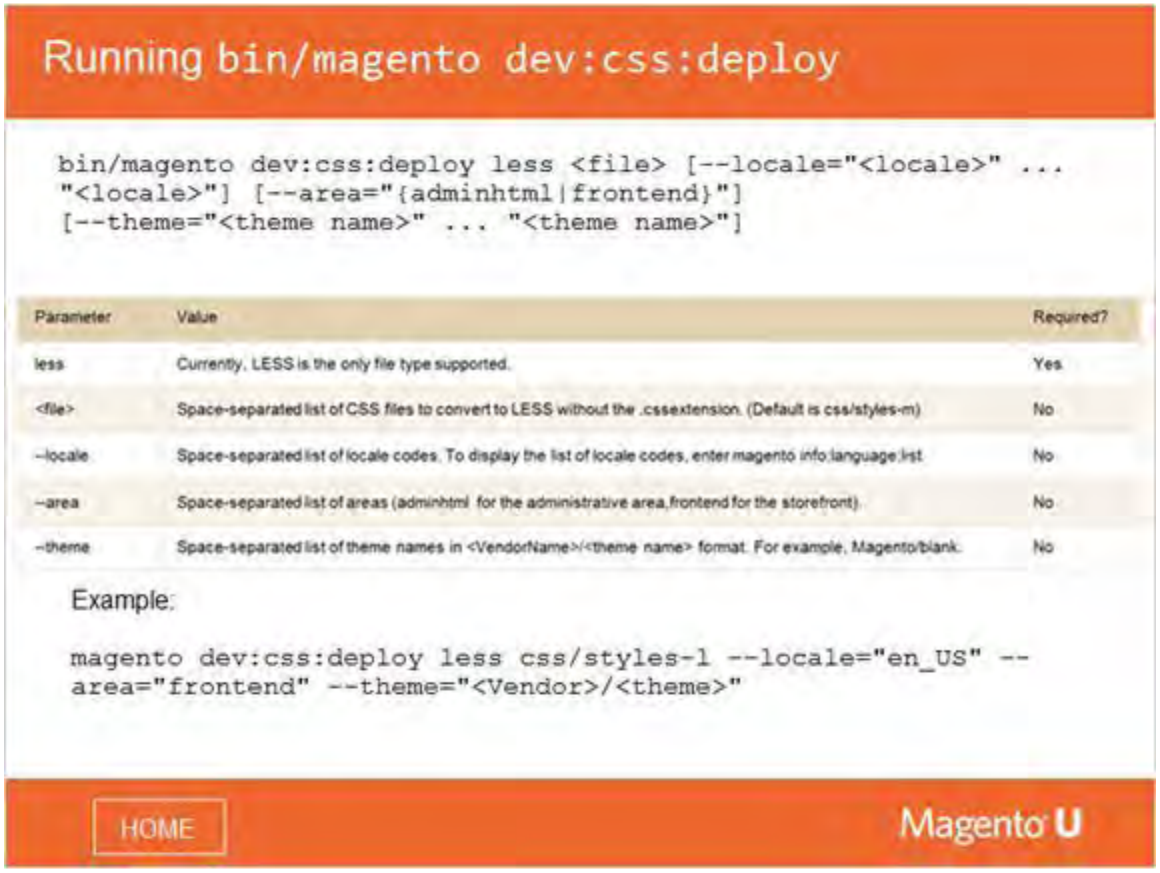
```
bin/magento setup:static-content:deploy en_US
```

Notes:

This example shows you how to deploy static content. You can simultaneously output files for various languages.

You can also do a dry run before an actual deployment.

4.9 Running bin/magento dev:css:deploy



Running bin/magento dev:css:deploy

```
bin/magento dev:css:deploy less <file> [--locale="<locale>" ...
"<locale>"] [--area="{adminhtml|frontend}"]
[--theme="<theme name>" ... "<theme name>"]
```

Parameter	Value	Required?
less	Currently, LESS is the only file type supported.	Yes
<file>	Space-separated list of CSS files to convert to LESS without the .cssextension. (Default is css/styles-m)	No
–locale	Space-separated list of locale codes. To display the list of locale codes, enter magento info:language:list	No
–area	Space-separated list of areas (adminhtml for the administrative area,frontend for the storefront).	No
–theme	Space-separated list of theme names in <VendorName>/<theme name> format. For example, Magento/blank.	No

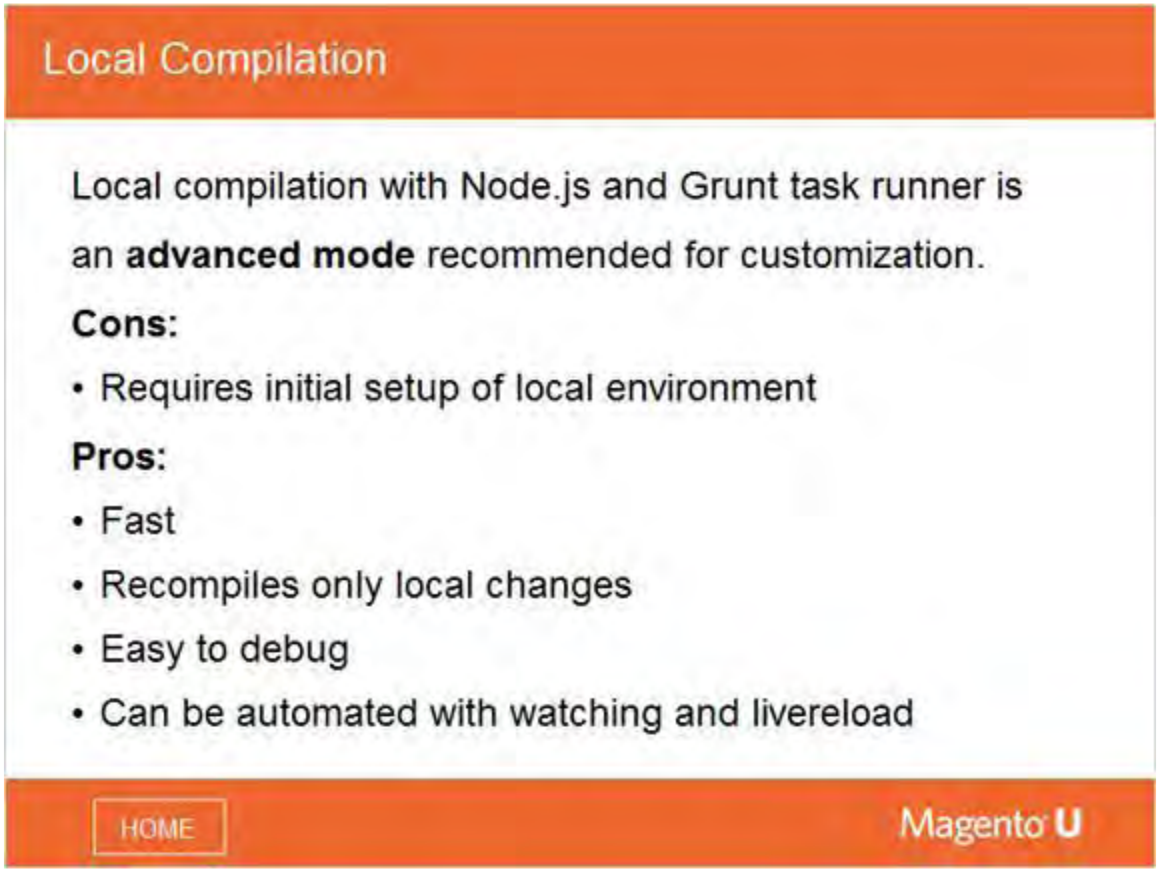
Example:

```
magento dev:css:deploy less css/styles-l --locale="en_US" --
area="frontend" --theme="<Vendor>/<theme>"
```

Notes:

This example shows how to deploy less files and shows the various options available.

4.10 Local Compilation



Local Compilation

Local compilation with Node.js and Grunt task runner is an **advanced mode** recommended for customization.

Cons:

- Requires initial setup of local environment

Pros:

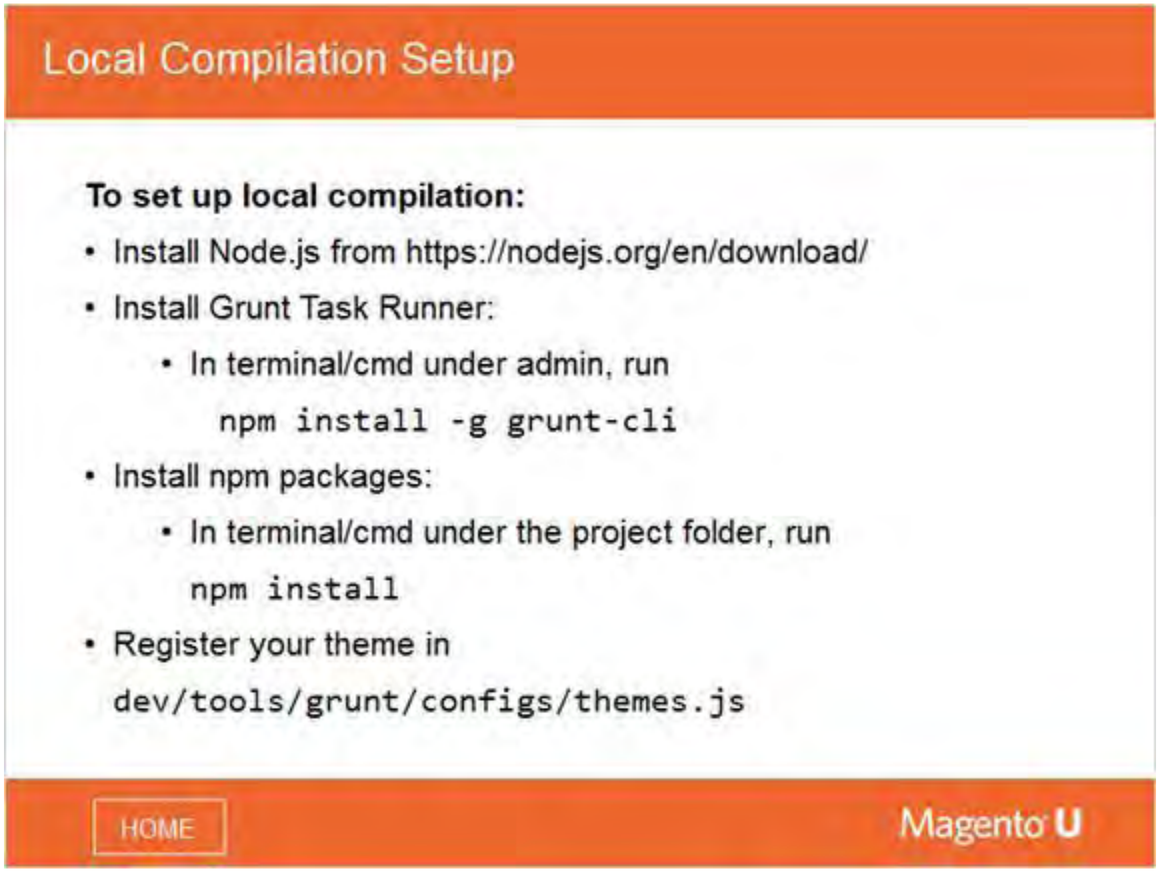
- Fast
- Recompiles only local changes
- Easy to debug
- Can be automated with watching and livereload

HOME
Magento U

Notes:

You can use local compilation with node.js and grunt to streamline the development process.

4.11 Local Compilation Setup



Local Compilation Setup

To set up local compilation:

- Install Node.js from https://nodejs.org/en/download/
- Install Grunt Task Runner:
  - In terminal/cmd under admin, run

    ```
    npm install -g grunt-cli
    ```
- Install npm packages:
  - In terminal/cmd under the project folder, run

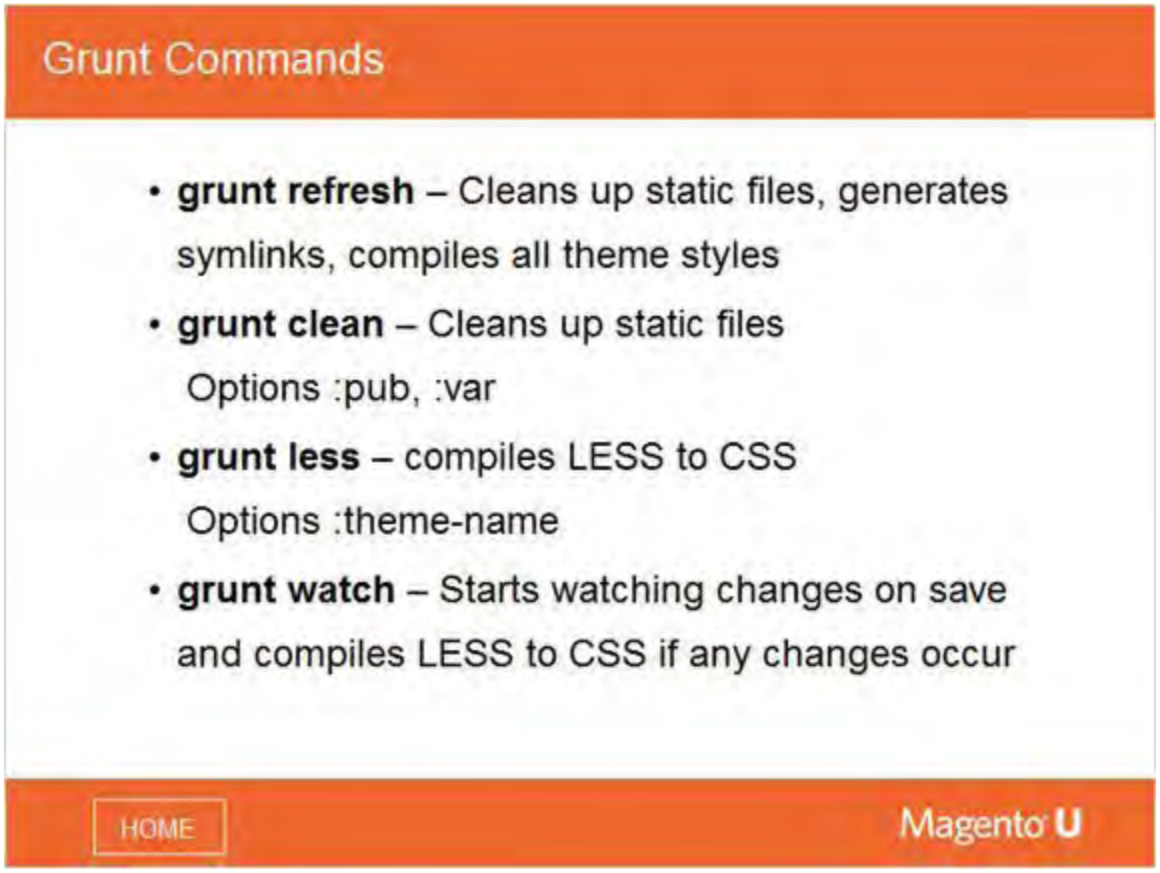
    ```
    npm install
    ```
- Register your theme in `dev/tools/grunt/configs/themes.js`

HOME
Magento U

Notes:

Here you can learn how to set up local compilation using Node.js and Grunt.

4.12 Grunt Commands



Grunt Commands

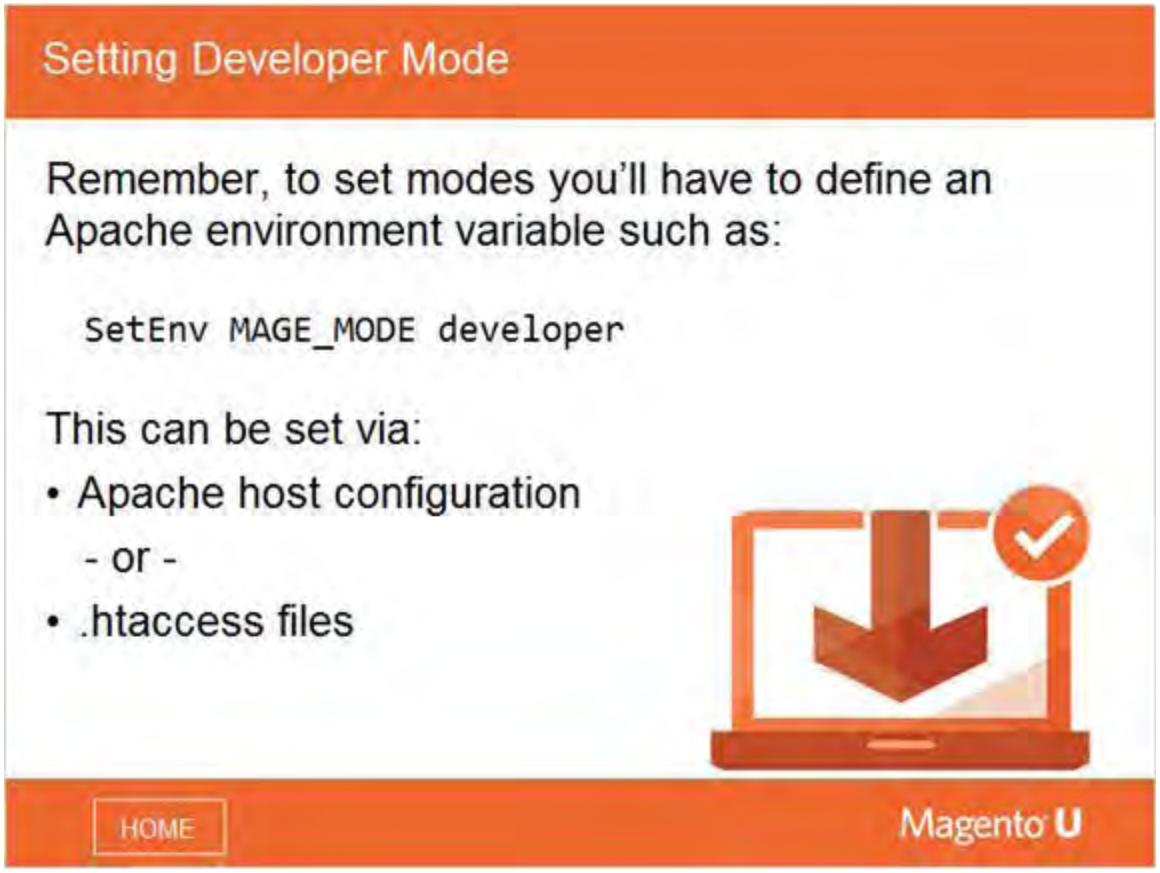
- **grunt refresh** – Cleans up static files, generates symlinks, compiles all theme styles
- **grunt clean** – Cleans up static files
  Options :pub, :var
- **grunt less** – compiles LESS to CSS
  Options :theme-name
- **grunt watch** – Starts watching changes on save and compiles LESS to CSS if any changes occur

HOME
Magento U

Notes:

Here are some of the common Grunt commands you'll use: refresh, clean, less, and watch.

4.13 Setting Developer Mode



Setting Developer Mode

Remember, to set modes you'll have to define an Apache environment variable such as:

```
SetEnv MAGE_MODE developer
```

This can be set via:

- Apache host configuration

  - or -

- .htaccess files

HOME
Magento U

Notes:

We saw this earlier but just in case you need a reminder….

Remember, to set modes, you need to define an Apache environment variable such as:

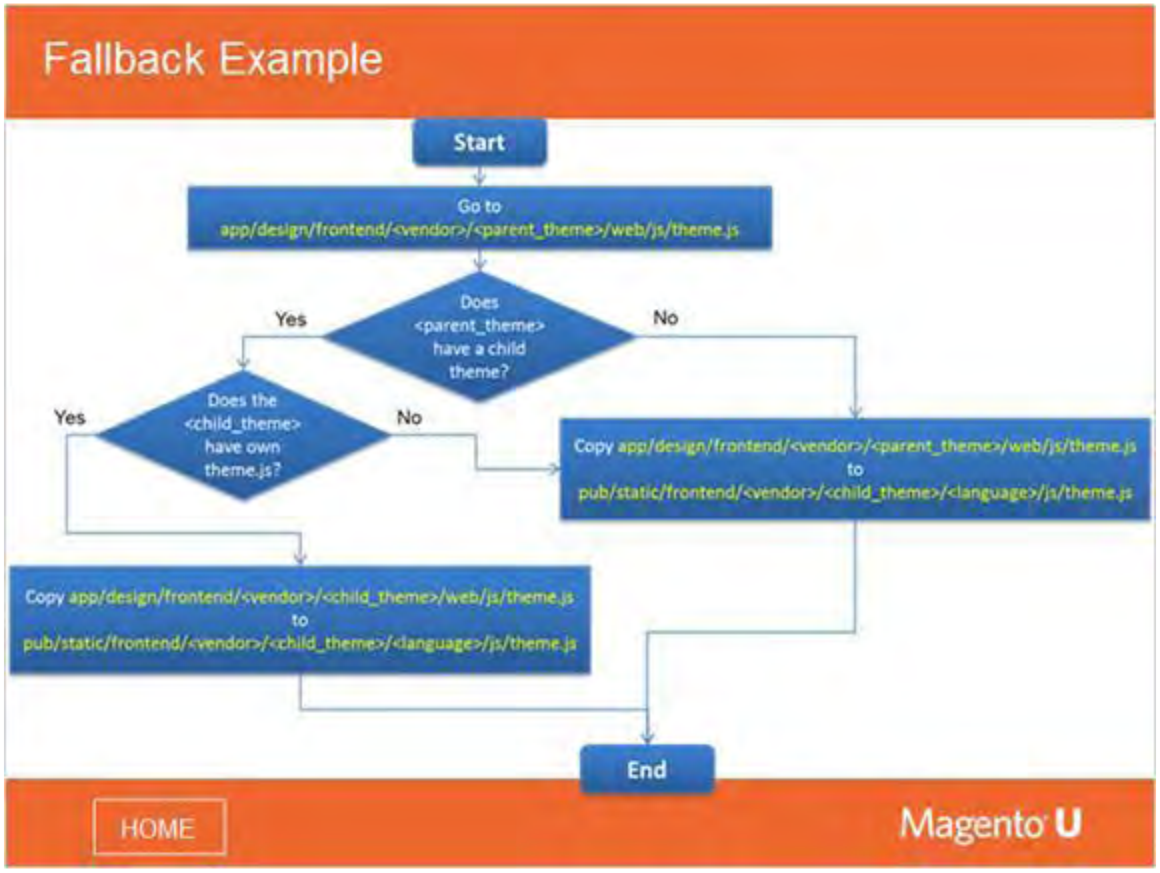
```
        SetEnv MAGE_MODE developer
```

In developer mode, when you install or enable a new module, it might load new JavaScript, CSS, layouts, and so on. To avoid issues with static files, you must flush the old files to make sure you get all the changes for the new module.

You can clear static files in any of the following ways:

- Manually by clearing the pub/static and var/view_preprocessed directories and subdirectories.
- Using the Magento command line. Several commands support an optional parameter --clear-static-content, which clears the appropriate directories. For example, see enable or disable modules.

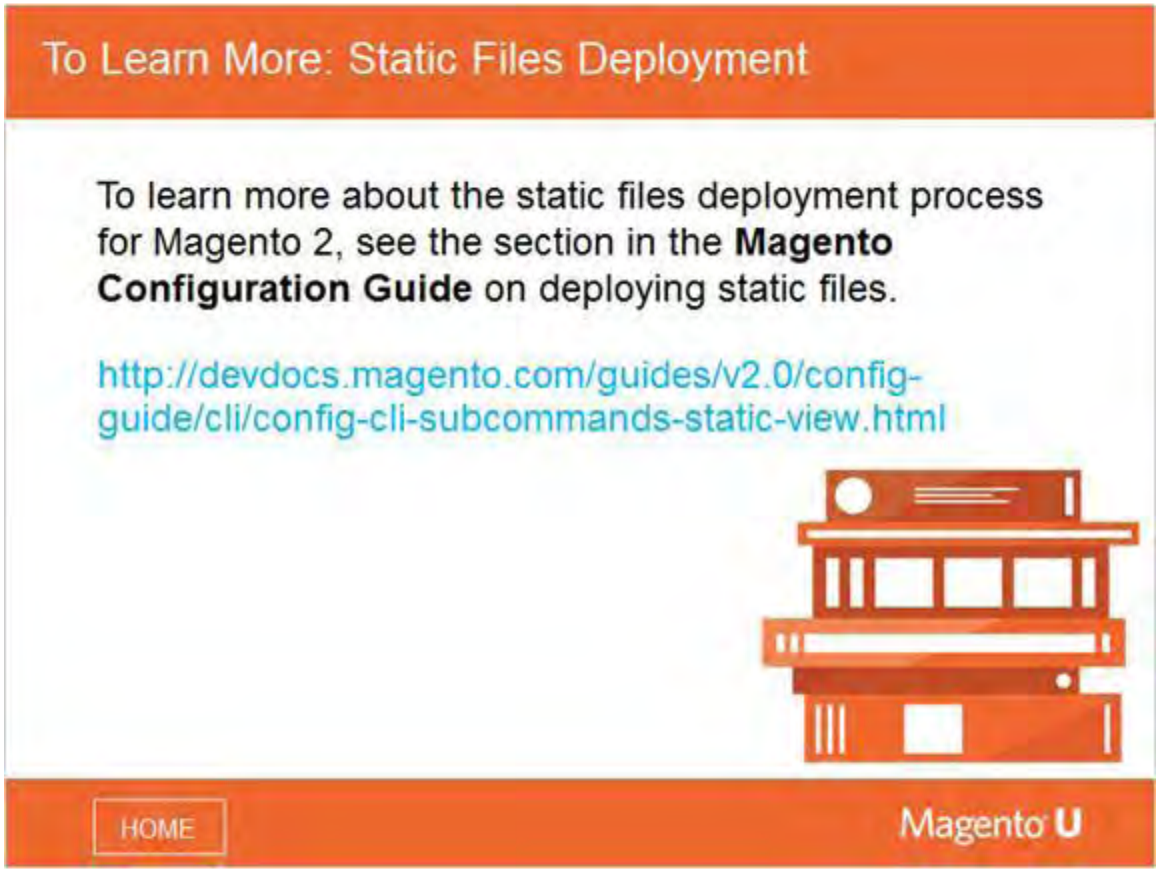
4.14 Fallback Example


Fallback Example
Start
Go to app/design/frontend/<vendor>/<parent_theme>/web/js/theme.js
Does <parent_theme> have a child theme?
Yes
No
Does the <child_theme> have own theme.js?
Yes
No
Copy app/design/frontend/<vendor>/<parent_theme>/web/js/theme.js to pub/static/frontend/<vendor>/<child_theme>/<language>/js/theme.js
Copy app/design/frontend/<vendor>/<child_theme>/web/js/theme.js to pub/static/frontend/<vendor>/<child_theme>/<language>/js/theme.js
End
HOME
Magento U

Notes:

Here is a fallback process example for the theme.js of the frontend <child_theme> theme that is inherited from <parent_theme> theme.

4.15 To Learn More: Static Files Deployment



To Learn More: Static Files Deployment

To learn more about the static files deployment process for Magento 2, see the section in the **Magento Configuration Guide** on deploying static files.

http://devdocs.magento.com/guides/v2.0/config-guide/cli/config-cli-subcommands-static-view.html

HOME
Magento U

Notes:

To learn more about the static files deployment process for Magento 2, see the section in the Magento Configuration Guide on deploying static files.

You can use the link shown on the screen to get to that section, or the link in the Notes to get to the specific section of that guide.

http://devdocs.magento.com/guides/v2.0/config-guide/cli/config-cli-subcommands-static-view.html#config-cli-static-overview

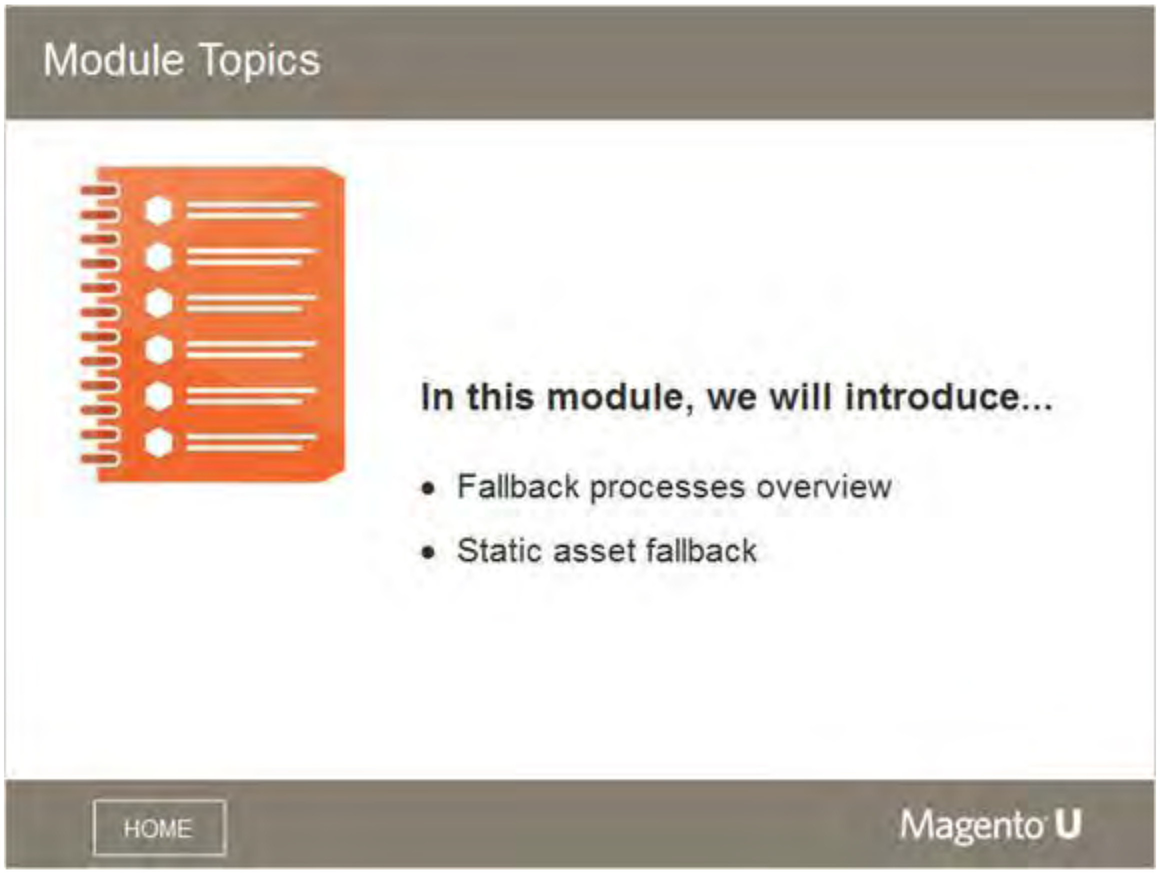
5. Module 5 Fallback

5.1 Module 5 Fallback

Notes:

We will now discuss fallback and learn how Magento determines which files to use for a theme.

5.2 Module Topics



Module Topics

In this module, we will introduce...

- Fallback processes overview
- Static asset fallback

HOME
Magento U

Notes:

In this module, we present the fallback processes for dynamic and static assets.

5.3 Fallback

Fallback

Fallback...

Fallback is a contingency option to be taken if the preferred choice is unavailable.

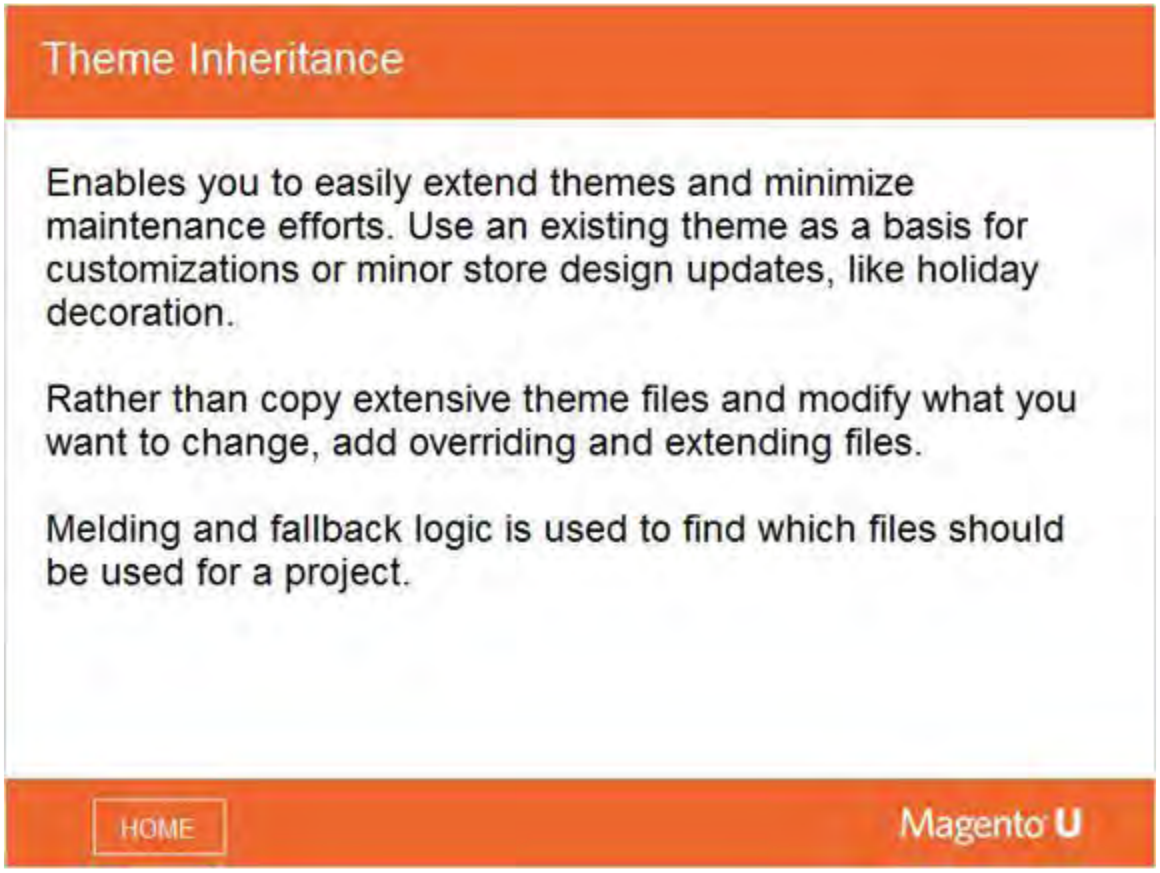
HOME
Magento U

Notes:

Fallback is a contingency option to be taken if the preferred choice is unavailable.

Melding and fallback logic is used to find which files should be used for a project.

5.4 Theme Inheritance



Theme Inheritance

Enables you to easily extend themes and minimize maintenance efforts. Use an existing theme as a basis for customizations or minor store design updates, like holiday decoration.

Rather than copy extensive theme files and modify what you want to change, add overriding and extending files.

Melding and fallback logic is used to find which files should be used for a project.

Notes:

Theme inheritance allows you to extend themes and minimize maintenance. You can modify only what you want to change and then override or extend files.

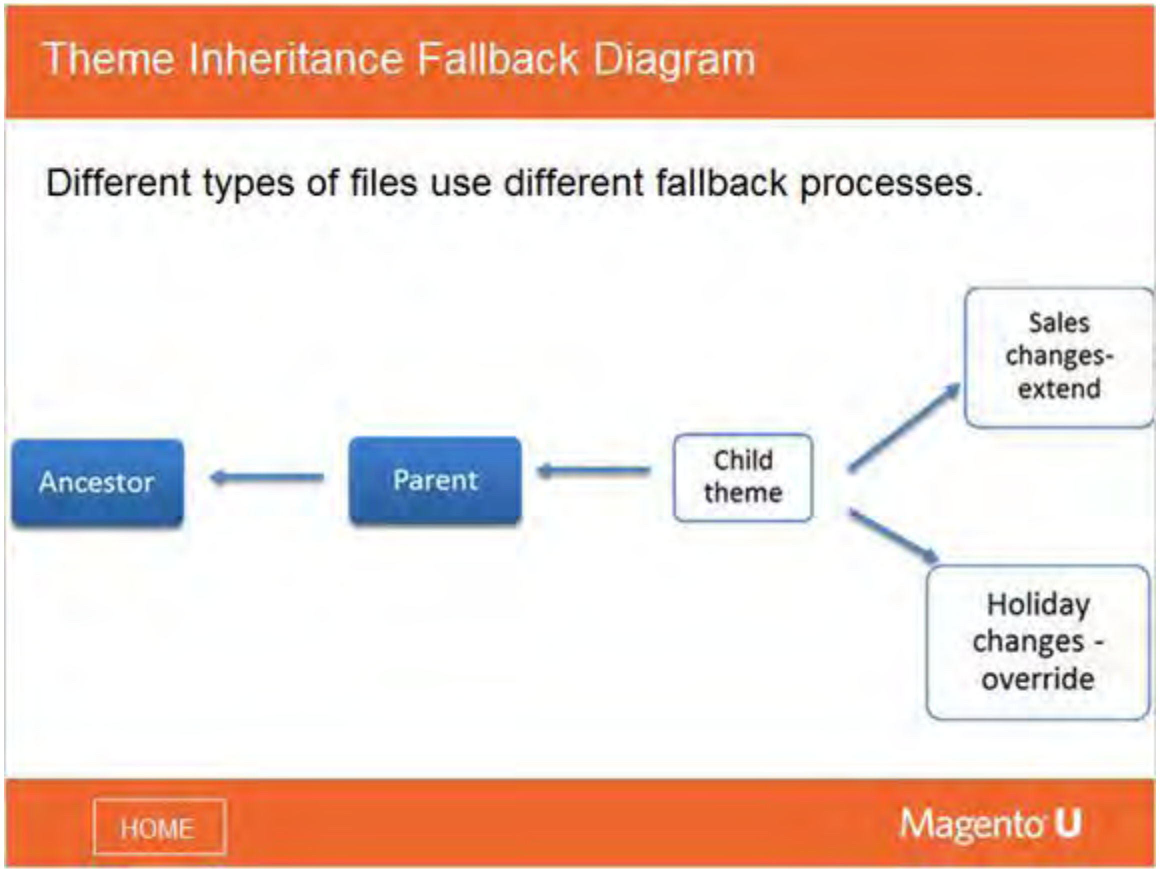
Fallback is the process of determining which option should be used.

The fallback system is used only for template and ./web files.

Static files also use a type of fallback but **only** when you compile the ./pub directory with the command-line tool. During runtime it's never looking for static files such as images, CSS, and JavaScript.

Layout fallback is discussed with Layout.

5.5 Theme Inheritance Fallback Diagram



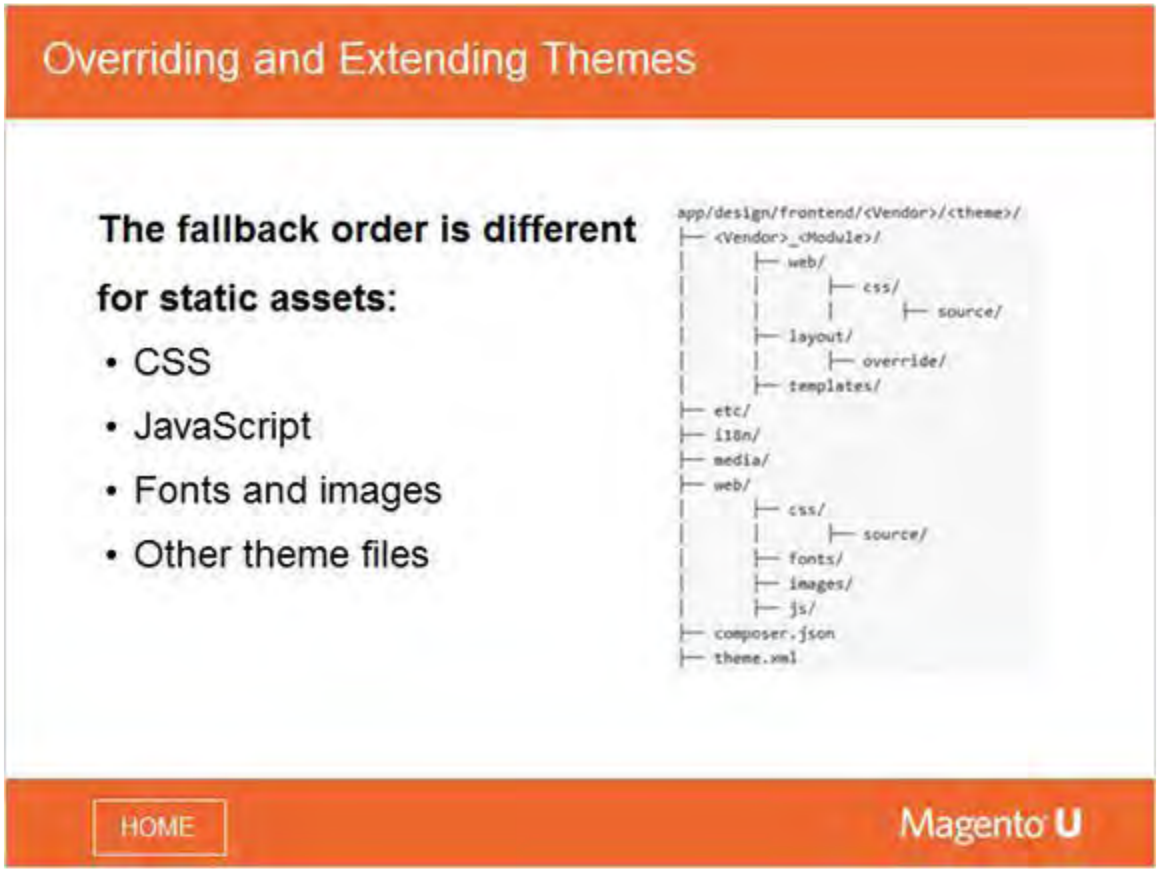
Theme Inheritance Fallback Diagram

Different types of files use different fallback processes.

Notes:

The different types of files use different fallback processes. We'll discuss some of them now.

5.6 Overriding and Extending Themes



Overriding and Extending Themes

The fallback order is different for static assets:

- CSS
- JavaScript
- Fonts and images
- Other theme files

```
app/design/frontend/<Vendor>/<theme>/
├─ <Vendor>_<Module>/
|        ├─ web/
|        |        ├─ css/
|        |        |        ├─ source/
|        ├─ layout/
|        |        ├─ override/
|        ├─ templates/
├─ etc/
├─ i18n/
├─ media/
├─ web/
|        ├─ css/
|        |        ├─ source/
|        ├─ fonts/
|        ├─ images/
|        ├─ js/
├─ composer.json
├─ theme.xml
```

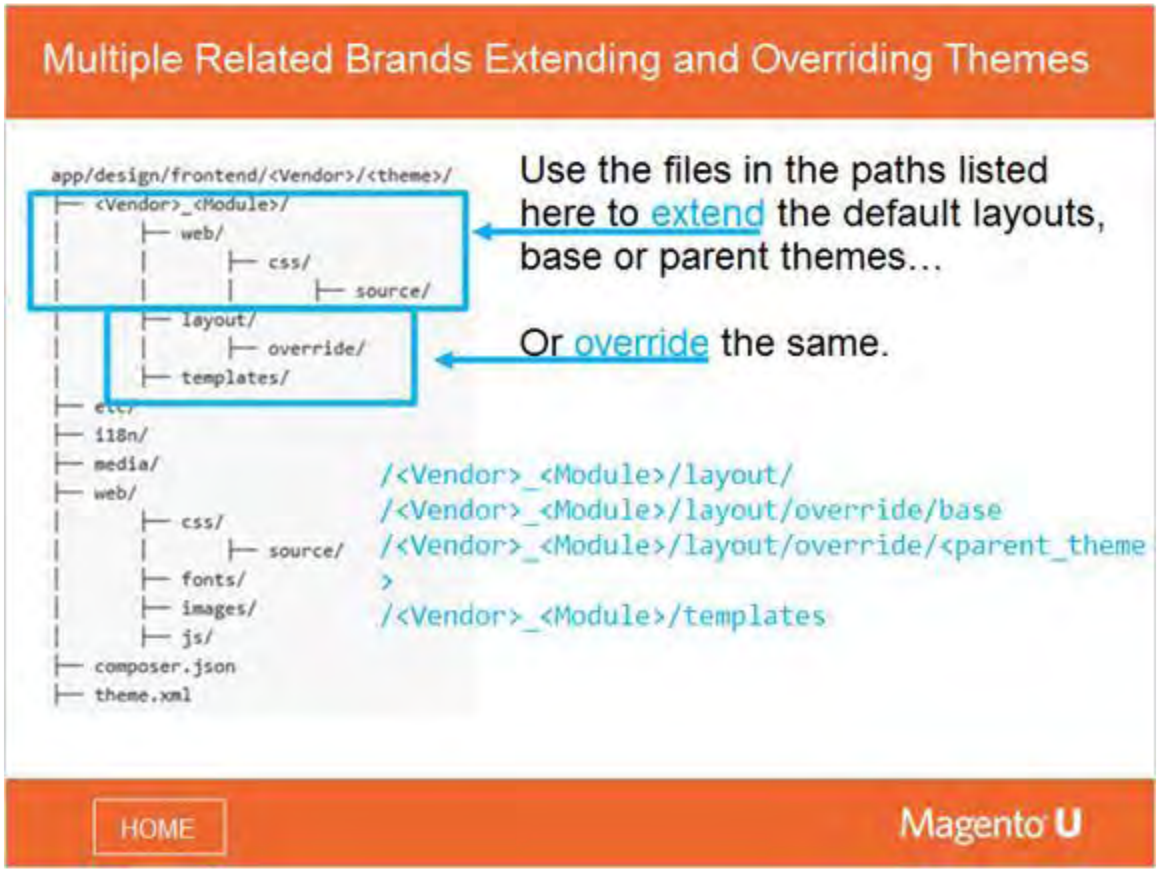
HOME

Magento U

Notes:

The next few slides describe the fallback for each type of theme files, and provide an overview of how to override ancestor themes and module designs.

5.7 Multiple Related Brands Extending and Overriding Themes



Multiple Related Brands Extending and Overriding Themes

```
app/design/frontend/<Vendor>/<theme>/
├── <Vendor>_<Module>/
|       ├── web/
|       |       ├── css/
|       |       |       ├── source/
|       ├── layout/
|       |       ├── override/
|       ├── templates/
├── etc/
├── i18n/
├── media/
├── web/
|       ├── css/
|       |       ├── source/
|       ├── fonts/
|       ├── images/
|       ├── js/
├── composer.json
├── theme.xml
```

Use the files in the paths listed here to extend the default layouts, base or parent themes…

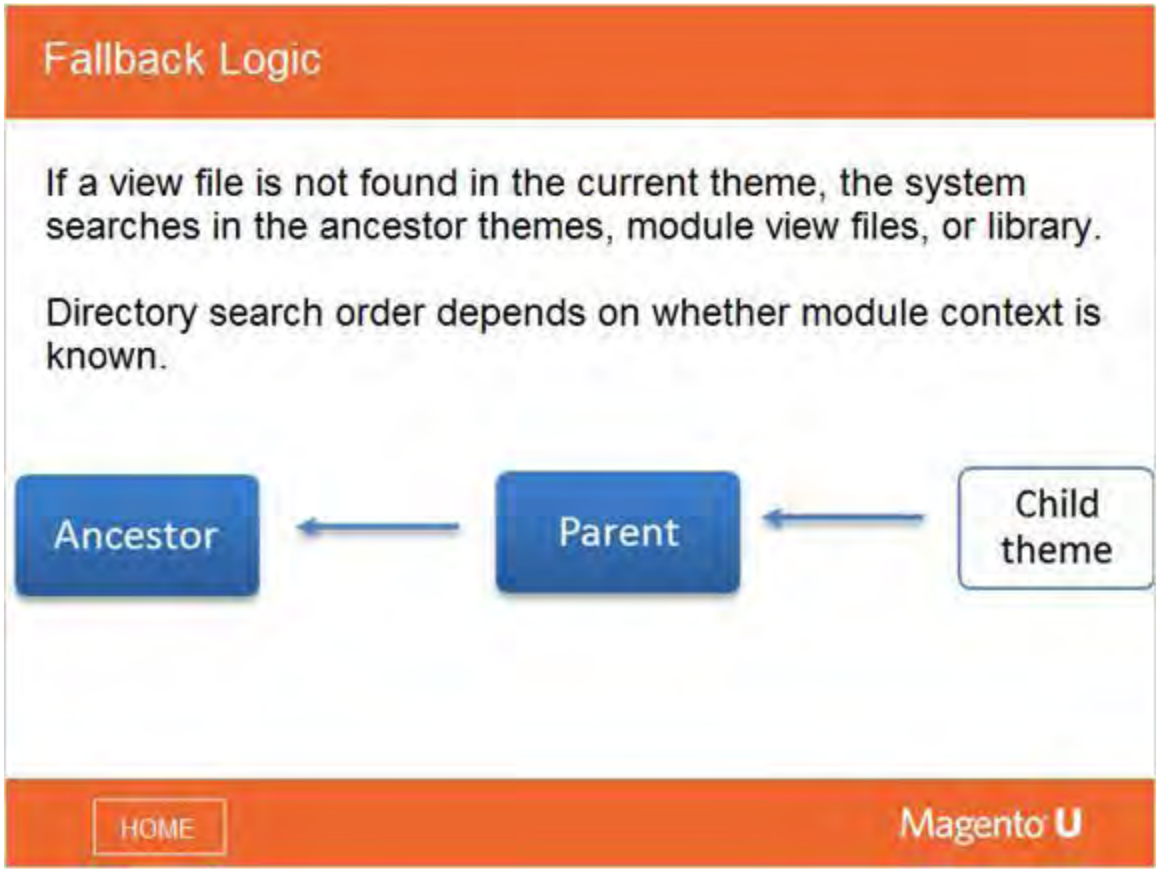
Or override the same.

```
/<Vendor>_<Module>/layout/
/<Vendor>_<Module>/layout/override/base
/<Vendor>_<Module>/layout/override/<parent_theme
>
/<Vendor>_<Module>/templates
```

Notes:

The details about extending and overriding themes will be covered in later slides.

5.8 Fallback Logic


Fallback Logic

If a view file is not found in the current theme, the system searches in the ancestor themes, module view files, or library.

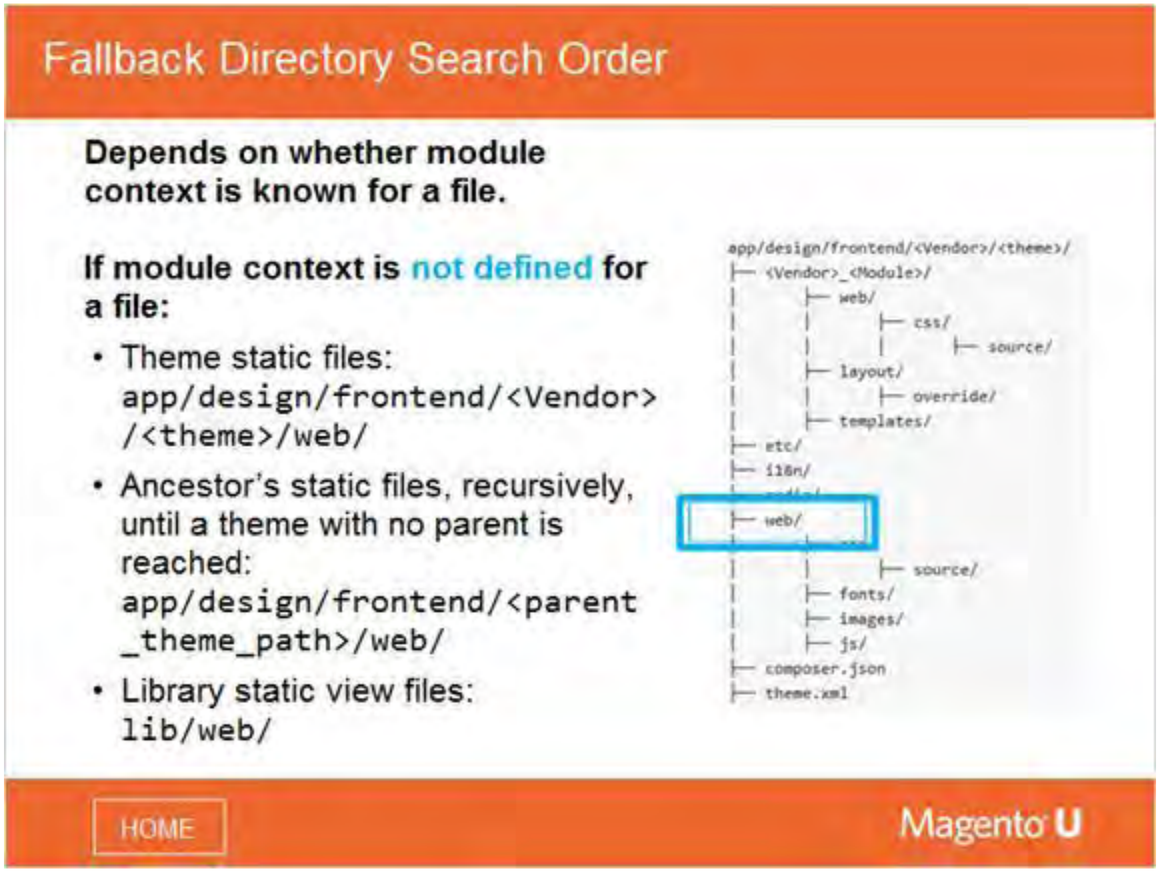
Directory search order depends on whether module context is known.

Notes:

If a view file is not found in the current theme, the system searches in the ancestor themes, module view files, or library.

Directory search order depends on whether module context is known.

5.9 Fallback Directory Search Order



Fallback Directory Search Order

Depends on whether module context is known for a file.

If module context is not defined for a file:

- Theme static files: `app/design/frontend/<Vendor>/<theme>/web/`
- Ancestor's static files, recursively, until a theme with no parent is reached: `app/design/frontend/<parent_theme_path>/web/`
- Library static view files: `lib/web/`

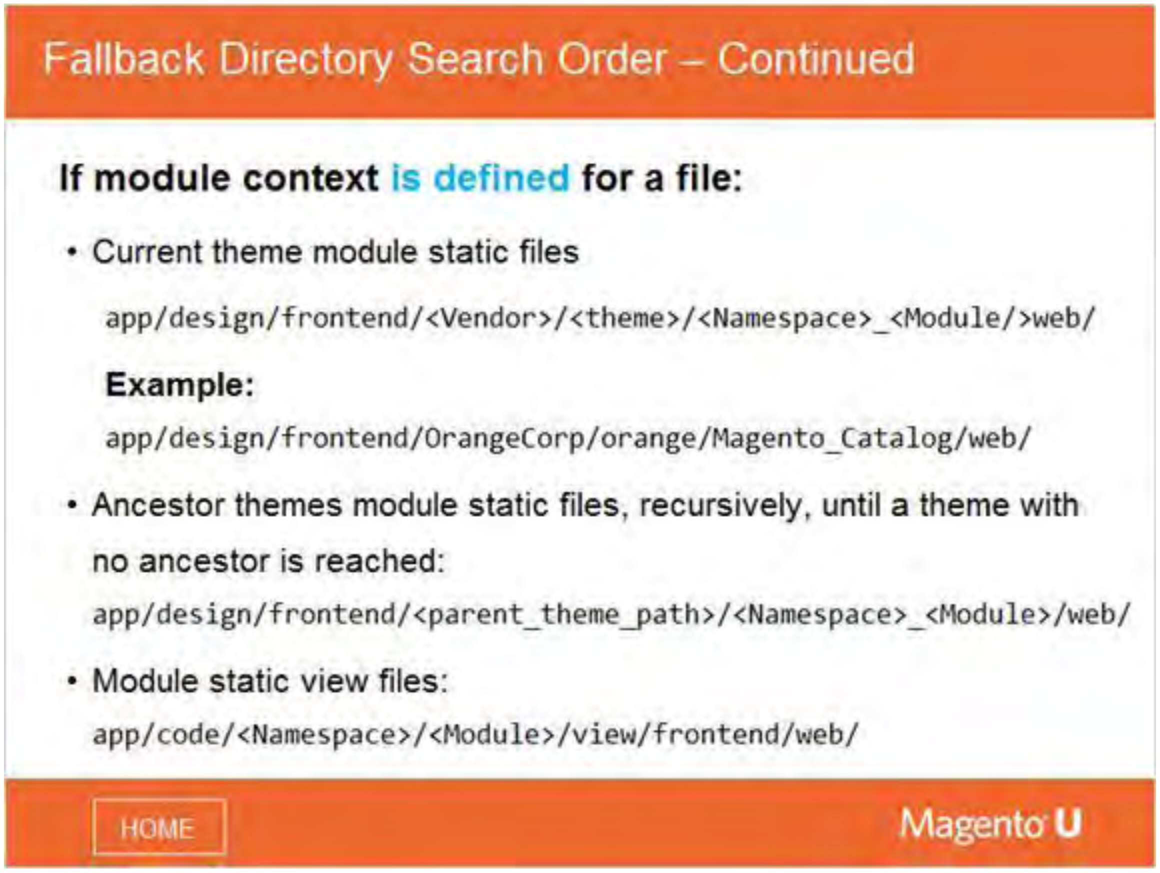
```
app/design/frontend/<Vendor>/<theme>/
├── <Vendor>_<Module>/
|       ├── web/
|       |       ├── css/
|       |       |       ├── source/
|       ├── layout/
|       |       ├── override/
|       ├── templates/
├── etc/
├── i18n/
├── media/
├── web/
|       ├── ...
|       |       ├── source/
|       ├── fonts/
|       ├── images/
|       ├── js/
├── composer.json
├── theme.xml
```

Notes:

The search order shown here is used when the module context is not defined.

To customize static view files defined in the parent theme, module view, or library files, you can override them by adding a file with the same name in the relevant location according to the fallback schemes. This also refers to the .less files, which technically are not static assets.

5.10 Fallback Directory Search Order - Continued



Fallback Directory Search Order – Continued

If module context is defined for a file:

- Current theme module static files

  `app/design/frontend/<Vendor>/<theme>/<Namespace>_<Module/>web/`

  Example:

  `app/design/frontend/OrangeCorp/orange/Magento_Catalog/web/`

- Ancestor themes module static files, recursively, until a theme with no ancestor is reached:

  `app/design/frontend/<parent_theme_path>/<Namespace>_<Module>/web/`

- Module static view files:

  `app/code/<Namespace>/<Module>/view/frontend/web/`

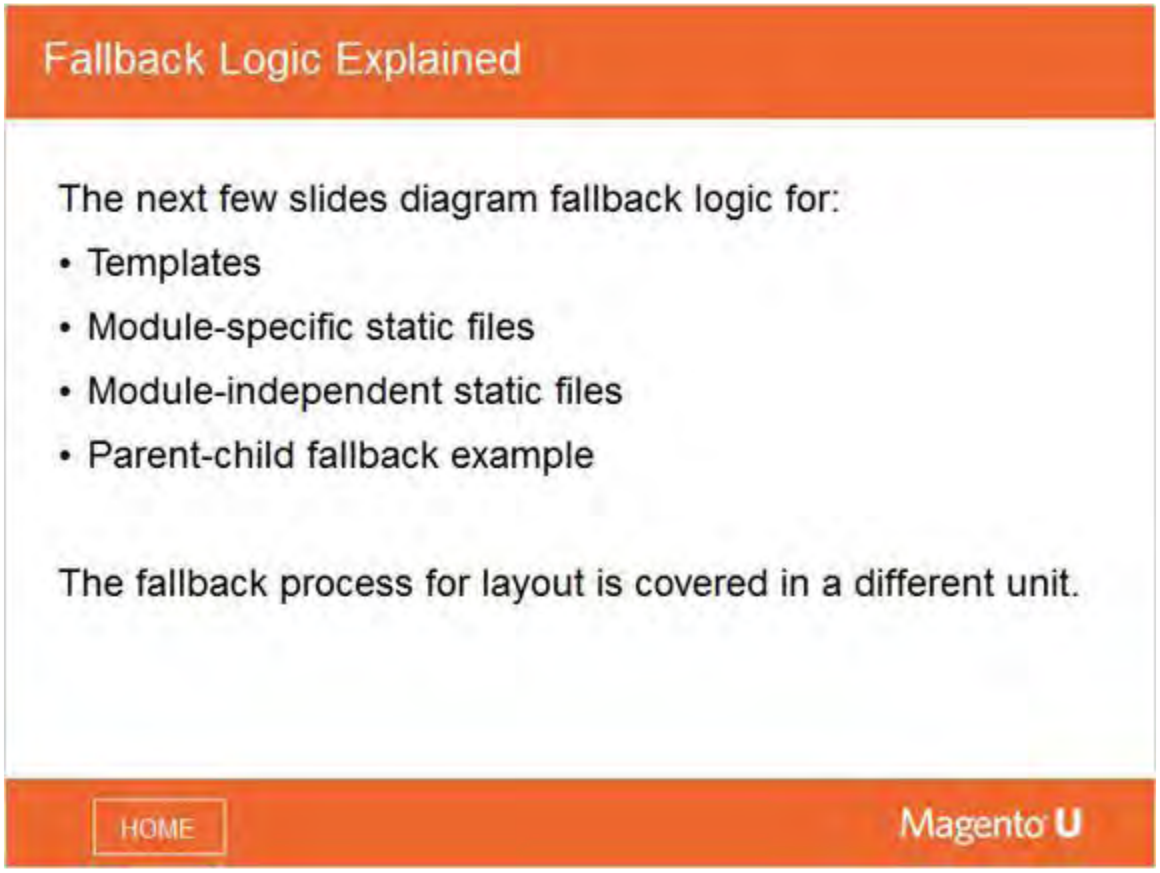
HOME

Magento U

Notes:

The search order shown here is used when the module context is defined.

5.11 Fallback Logic Explained



Fallback Logic Explained

The next few slides diagram fallback logic for:

- Templates
- Module-specific static files
- Module-independent static files
- Parent-child fallback example

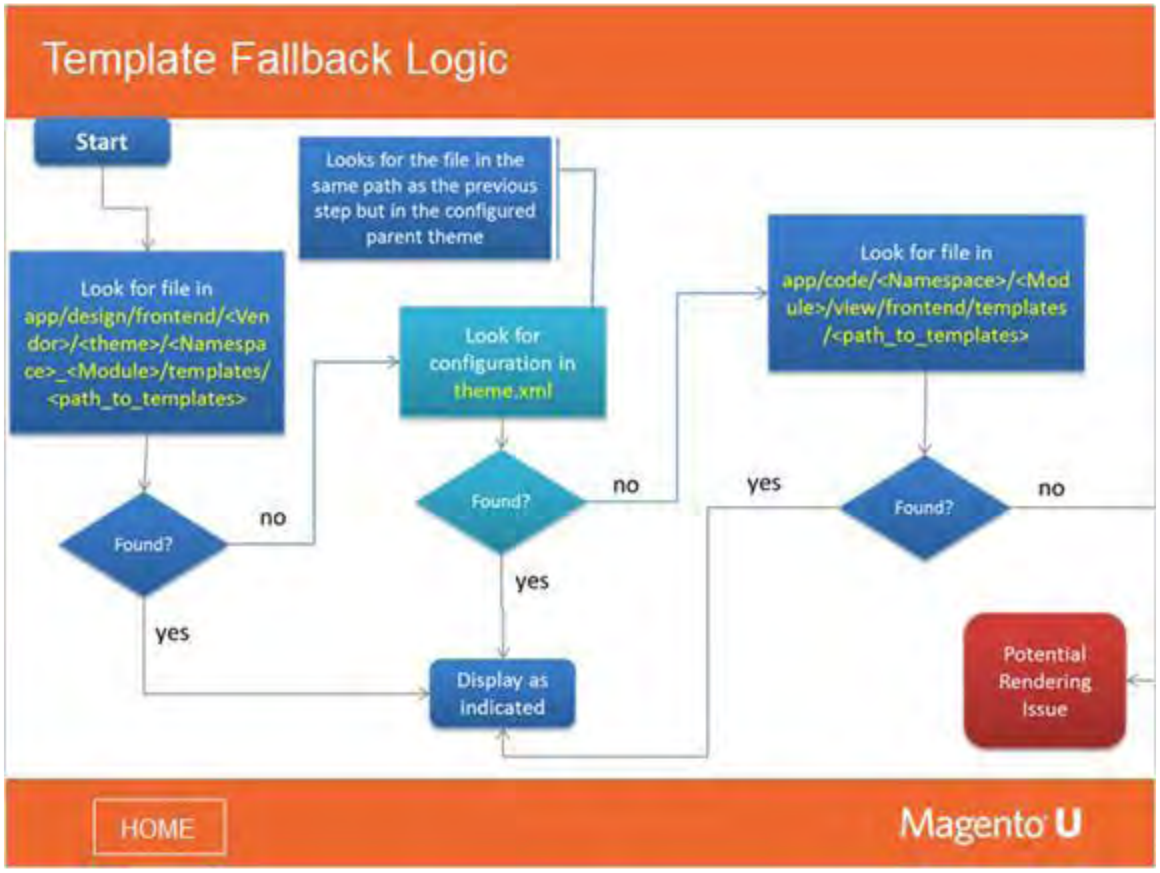
The fallback process for layout is covered in a different unit.

HOME
Magento U

Notes:

Now we’ll look at how fallback works. Note that the fallback process for layout is covered in a different unit.

5.12 Template Fallback Logic


Template Fallback Logic
Start
Look for file in app/design/frontend/<Vendor>/<theme>/<Namespace>_<Module>/templates/<path_to_templates>
Found?
no
yes
Looks for the file in the same path as the previous step but in the configured parent theme
Look for configuration in theme.xml
Found?
no
yes
Display as indicated
Look for file in app/code/<Namespace>/<Module>/view/frontend/templates/<path_to_templates>
Found?
yes
no
Potential Rendering Issue
HOME
Magento U

Notes:

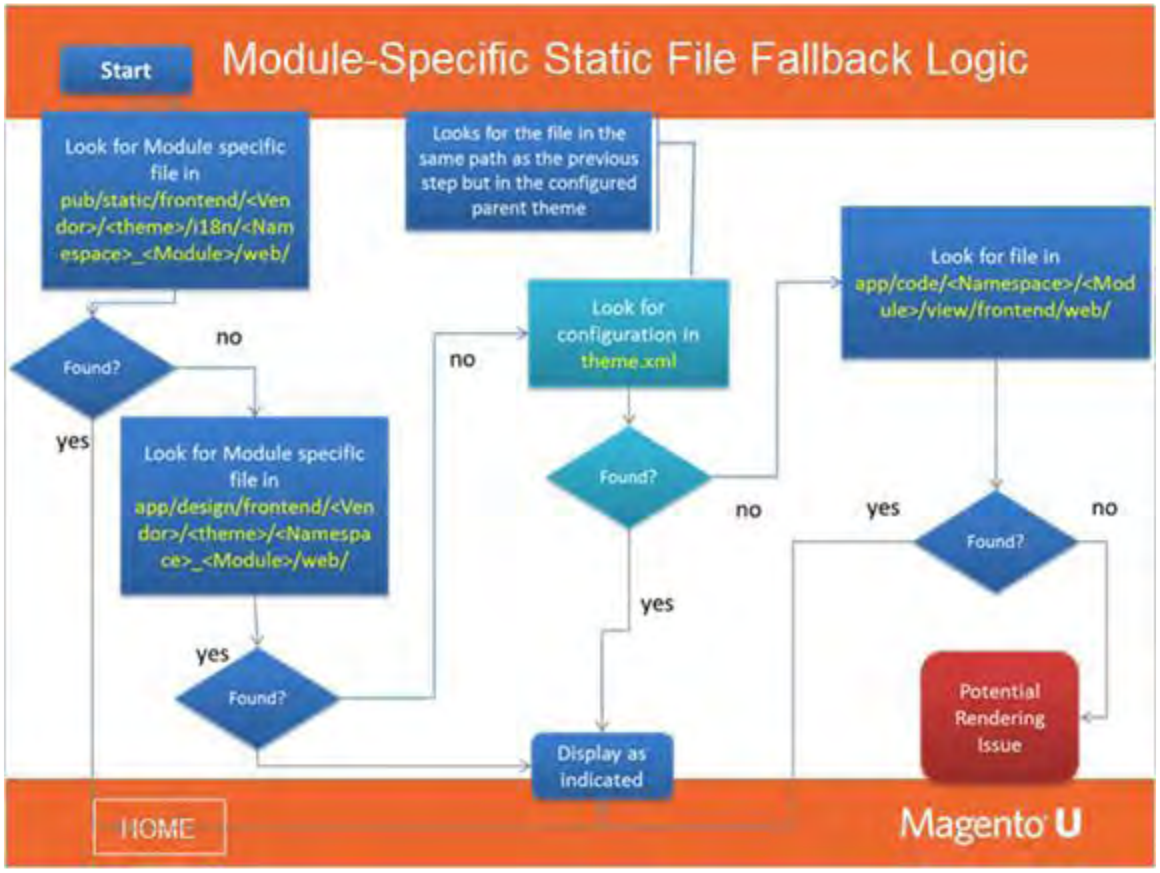
There a few things to note in Magento 2 template fallback logic:

- **Only files that are customized should be present in custom themes.**
- All other files will be pulled from lower in the fallback scheme.
- This significantly enhances maintainability and upgradeability. Because of this fallback model, even a highly customized theme will contain only a subset of the total files.

In Magento there's a special template which serves as the root template for all pages in the application:
`app/code/Magento/Theme/view/base/templates/root.phtml`

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

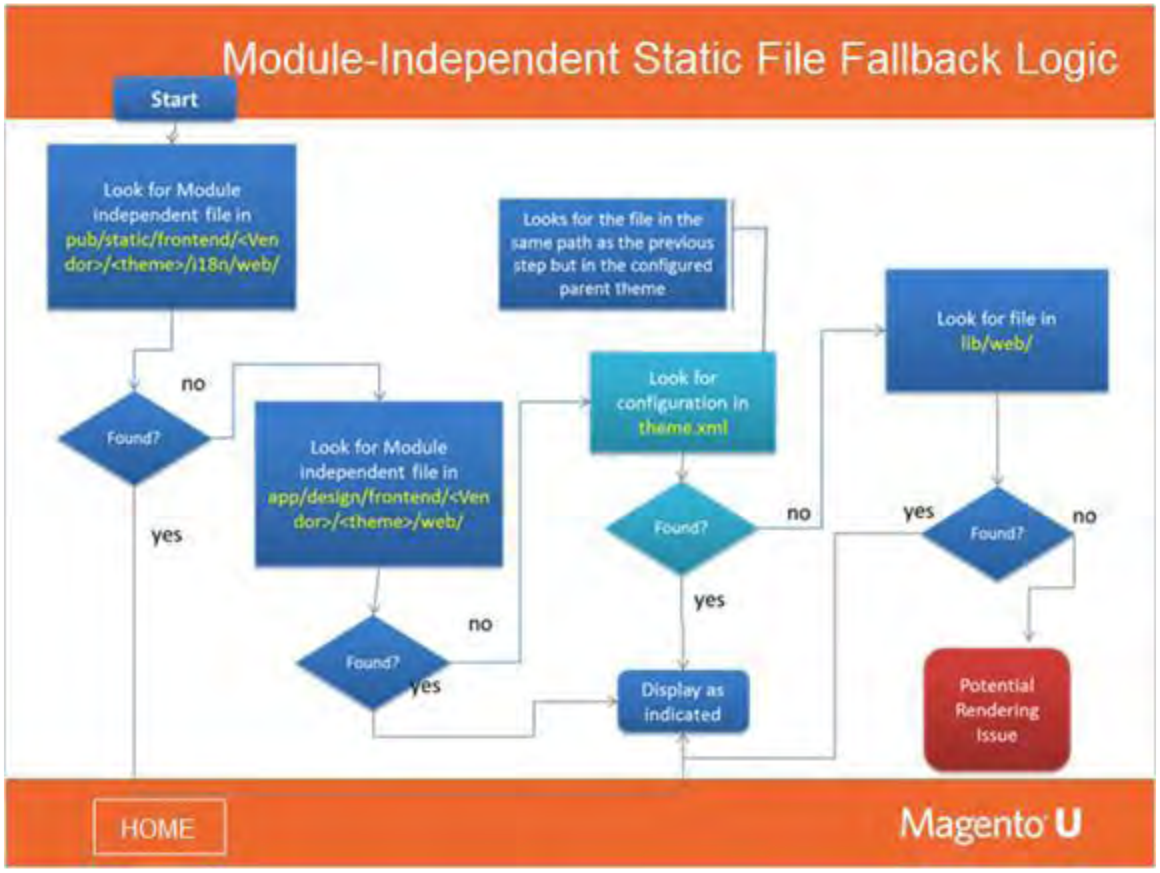
5.13 Module-Specific Static File Fallback Logic


Start
Module-Specific Static File Fallback Logic
Look for Module specific file in pub/static/frontend/<Vendor>/<theme>/i18n/<Namespace>_<Module>/web/
Found?
no
yes
Look for Module specific file in app/design/frontend/<Vendor>/<theme>/<Namespace>_<Module>/web/
yes
Found?
no
Looks for the file in the same path as the previous step but in the configured parent theme
Look for configuration in theme.xml
Found?
no
yes
Display as indicated
Look for file in app/code/<Namespace>/<Module>/view/frontend/web/
yes
Found?
no
Potential Rendering Issue
HOME
Magento U

Notes:

Static view files fallback also supports lookup through a list of additional extensions for a specific extension. For example, search for a LESS file, if a CSS file was not found.

5.14 Module-Independent Static File Fallback Logic


Module-Independent Static File Fallback Logic
Start
Look for Module independent file in pub/static/frontend/<Vendor>/<theme>/i18n/web/
Found?
no
yes
Look for Module independent file in app/design/frontend/<Vendor>/<theme>/web/
Found?
no
yes
Looks for the file in the same path as the previous step but in the configured parent theme
Look for configuration in theme.xml
Found?
no
yes
Display as indicated
Look for file in lib/web/
Found?
yes
no
Potential Rendering Issue
HOME
Magento U

Notes:

You can follow the diagram to see how module-independent fallback logic works.