# Determine how to locate different types of files in Magento.

One of the first things you can do to get started with component development is to understand and set up the file system. Each type of component has a different file structure, although all components require certain files. In addition, you can choose the component root directory to start development. The following sections have more information. - Magento DevDocs - About component file structure

# Where are the files containing JavaScript, HTML, and PHP located?

- view/frontend/web/js
- view/frontend/requirejs-config.js
- · view/frontend/layout
- view/frontend/templates

# **Root directory location**

A component's root directory is the top-level directory for that component under which its folders and files are located. Depending on how your Magento development environment was installed, your component's root directory can be located in two places: - Magento DevDocs - Create your component file structure

#### <Magento install directory>/app:

- Modules app/code.
- Storefront themes app/design/frontend.
- · Admin themes app/design/adminhtml.
- Language packages use app/i18n.

# <Magento install directory>/vendor

This location is found in the alternative setups where the composer create-project command was used to get a Magento 2 metapackage (which downloads the CE or EE code), or a compressed Magento 2 archive was extracted in order to install Magento.

Any third party components (and the Magento application itself) are downloaded and stored under the vendor directory. If you are using Git to manage project, this directory is typically added to the .gitignore file. Therefore, we recommend you do your customization work in app/code, not vendor.

- Magento DevDocs - Create your component file structure

## Required files

registration.php: Among other things, this file specifies the directory in which the component is installed by vendors in production environments. By default, composer automatically installs components in the /vendor directory. For more information, see Component registration.

etc/module.xml: This file specifies basic information about the component such as the components dependencies and its version number. This version number is used to determine schema and data updates when bin/magento setup:upgrade is run.

composer.json: Specifies component dependencies and other metadata. For more

information, see Composer integration.

#### - Magento DevDocs - About component file structure

Class Magento\Framework\Module\ModuleList\Loader load etc/module.xml files and sort modules by sequence. The sequence use for sorting events, plugins, preferences and layouts.

#### **Common directories**

- Api Any PHP classes exposed to the API.
- Block PHP view classes as part of Model View Controller(MVC) vertical implementation of module logic.
- Console Console commands
- Controller PHP controller classes as part of MVC vertical implementation of module logic.
- Controller/Adminhtml Admin controllers
- Cron Cron job classes
- etc Configuration files; in particular, module.xml, which is required.
- Helper Helpers
- i18n Localization files in CSV format
- Model PHP model classes as part of MVC vertical implementation of module logic.
- Model/ResourceModel Database interactions
- Observer Event listeners
- Plugin Contains any needed plug-ins.
- Setup Classes for module database structure and data setup which are invoked when installing or upgrading.
- Test Unit tests
- Ui UI component classes
- view View files, including static view files, design templates, email templates, and layout files.
  - o view/{area}/email
  - o view/{area}/layout
  - view/{area}/templates
  - o view/{area}/ui\_component
  - o view/{area}/ui\_component/templates
  - o view/{area}/web
  - view/{area}/web/template
  - view/{area}/requirejs-config.js

see Magento DevDocs - Create your component file structure

#### Service contract

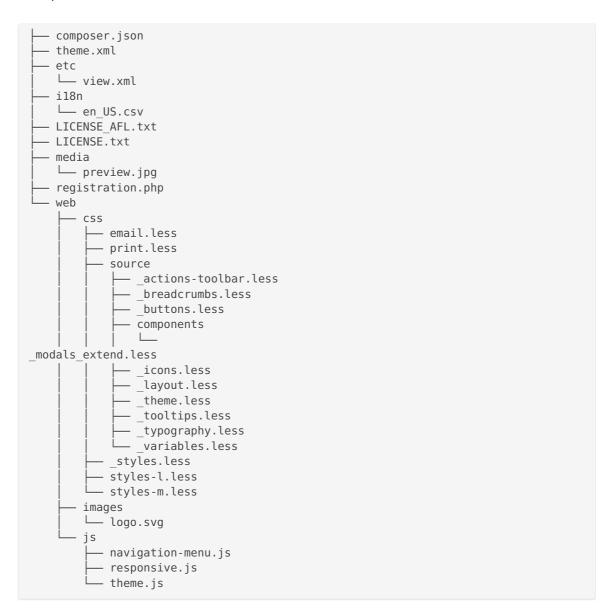
The service contract of a module is defined by the set of interfaces in the module's /Api directory.

This directory contains:

- Service interfaces in the Api namespace of the module (Catalog API).
- Data (or entity) interfaces in the Api/Data directory (Catalog API/Data). Data
  entities\* are data structures passed to and returned from service interfaces. Files in
  the data directory contain get() and set() methods for entries in the entity table and
  extension attributes.

## Theme file structure

Example #1



# Language package file structure

Example #2

Examples #1, #2 - - Magento DevDocs - Create your component file structure