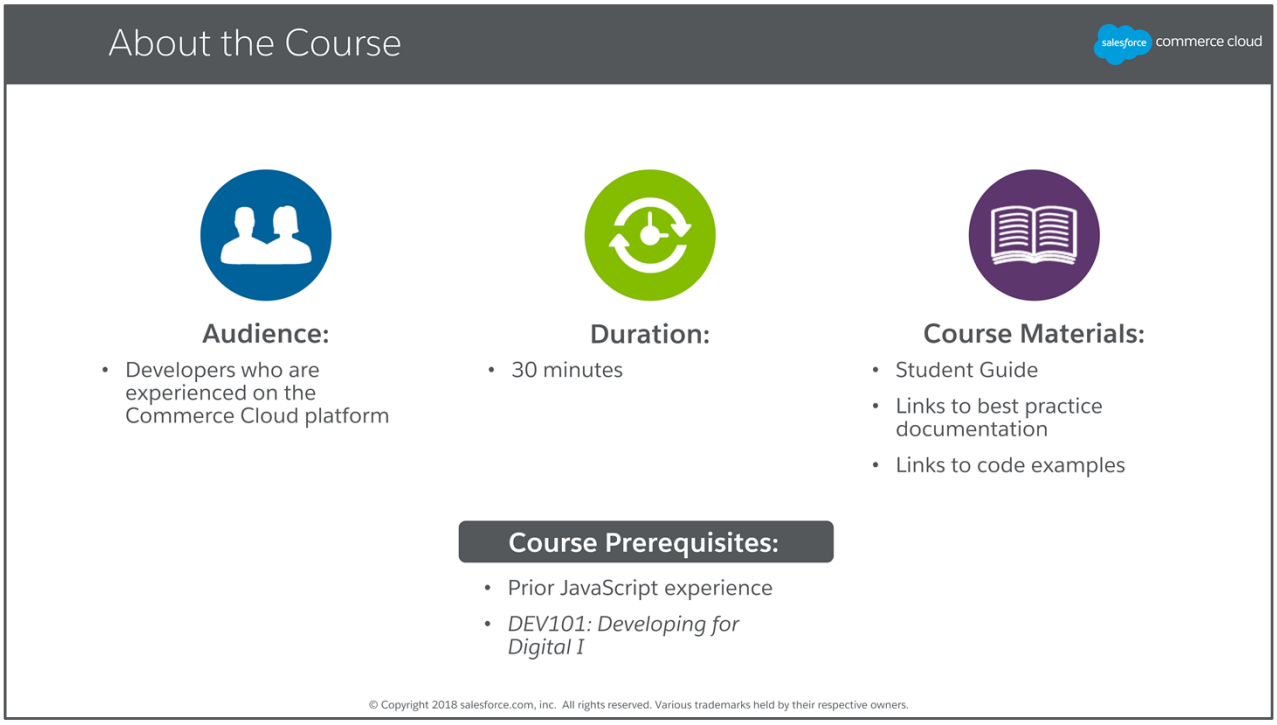
salesforce commerce cloud

Cartridges and Commerce Cloud Mobile-First Reference Architecture

Student Guide

The Mobile-First Reference Architecture is based on the Model View Controller (MVC) software architectural pattern. The MVC structure provides developers a modular separation of core reference features, merchant customization, and integration cartridges, enabling developers to add functionality without making changes to the underlying Mobile-First Reference Architecture codebase. Understanding this approach is important for SIs and LINK technology partners who integrate LINK technology cartridges and for merchant storefront developers who build customizations through overlay cartridges. This course covers how to build Mobile-First Reference Architecture cartridge extensions for storefront customizations and link technology integration.



This course is aimed at developers who are experienced on the Commerce Cloud platform, have prior JavaScript experience, and have completed *DEV101: Developing for Digital I*. Course materials, such as links and the student guide, are located in the Course Materials tab.

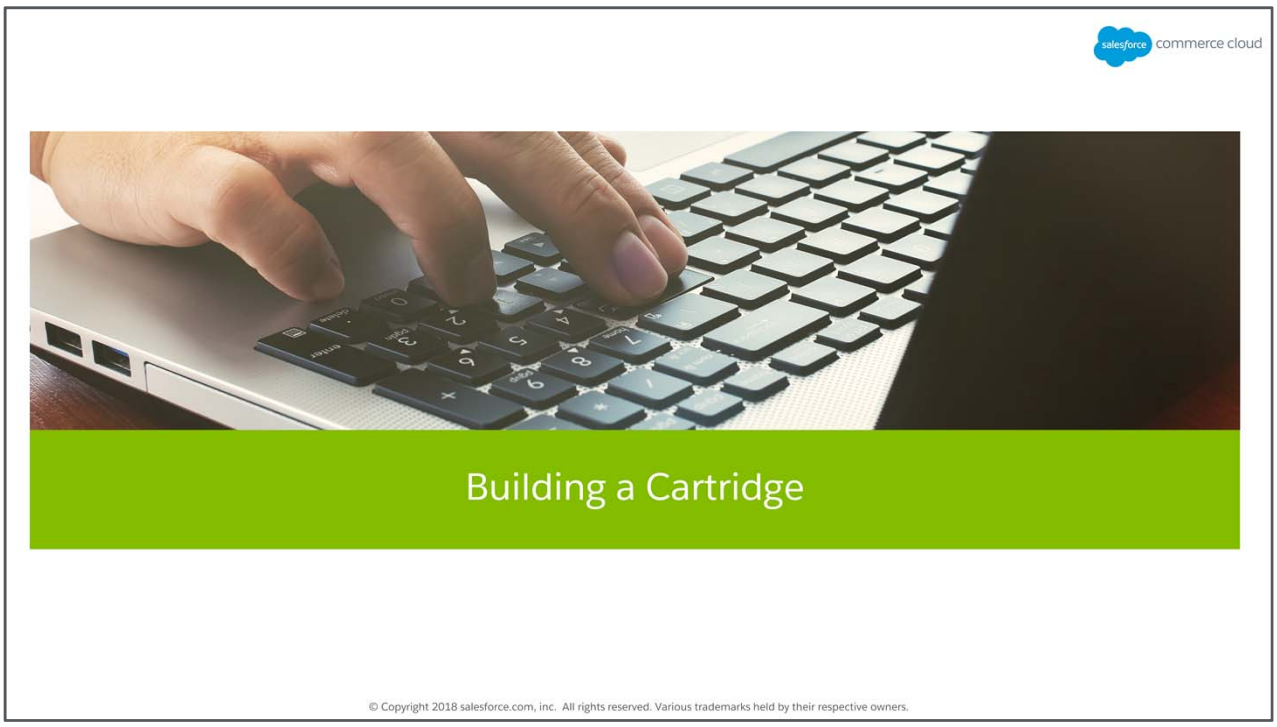
Copyright

Developing in Commerce Cloud Mobile-First Reference Architecture

Course Learning Objectives

Upon completion of this course you will be able to:

- Build a cartridge for a storefront on the Mobile-First Reference Architecture platform.



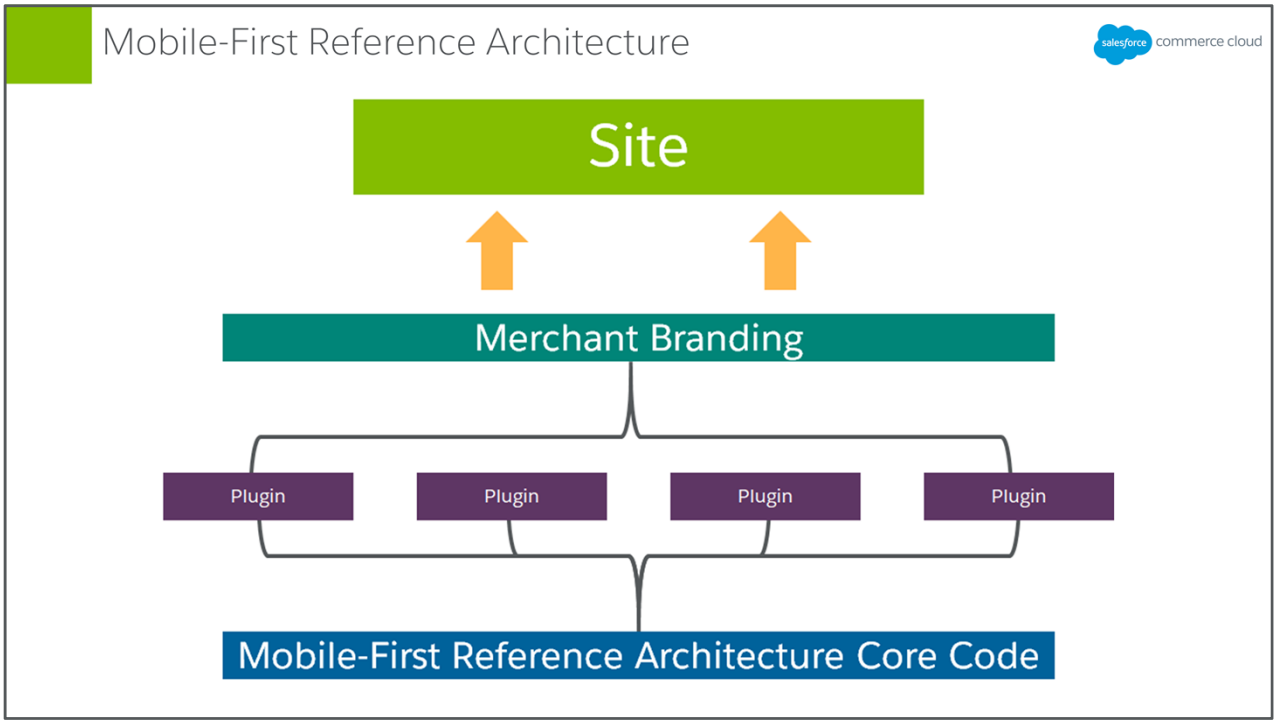
Building a Cartridge

salesforce commerce cloud

Module Learning Objectives

Upon completion of this module you will be able to:

- Explain how a cartridge extends functionality in the Mobile-First Reference Architecture platform.
- Set up the storefront's base cartridge and add overlay cartridges.
- Differentiate between major and minor releases and patches.
- Download and use sgmf-scripts on the platform.
- Apply Commerce Cloud's implementation of CommonJS standards to cartridge development.
- Follow Mobile-First Reference Architecture coding best practices and standards.
- Determine when it is best override or extend a function.
- Determine if a SiteGenesis cartridge needs to be altered to be compatible with the Mobile-First Reference Architecture platform.
- Explain the four major cartridge certification criteria that are specific to the Mobile-First Reference Architecture platform.



Mobile-First Reference Architecture was created using the MVC architectural pattern. Because of this, all business logic is removed from template. The Bootstrap 4 framework and jQuery are used for client-side scripts. Most actions after the initial load are done via Ajax. All scripts are automatically added to the bottom of the page and all CSS files are added into the <HEAD> element.

Storefronts built on the Mobile-First Reference Architecture platform are comprised of three major parts: the base cartridge, which contains the core Mobile-First Reference Architecture code, plugin cartridges that extend the functionality of a storefront, and a branding cartridge.

The Base Cartridge

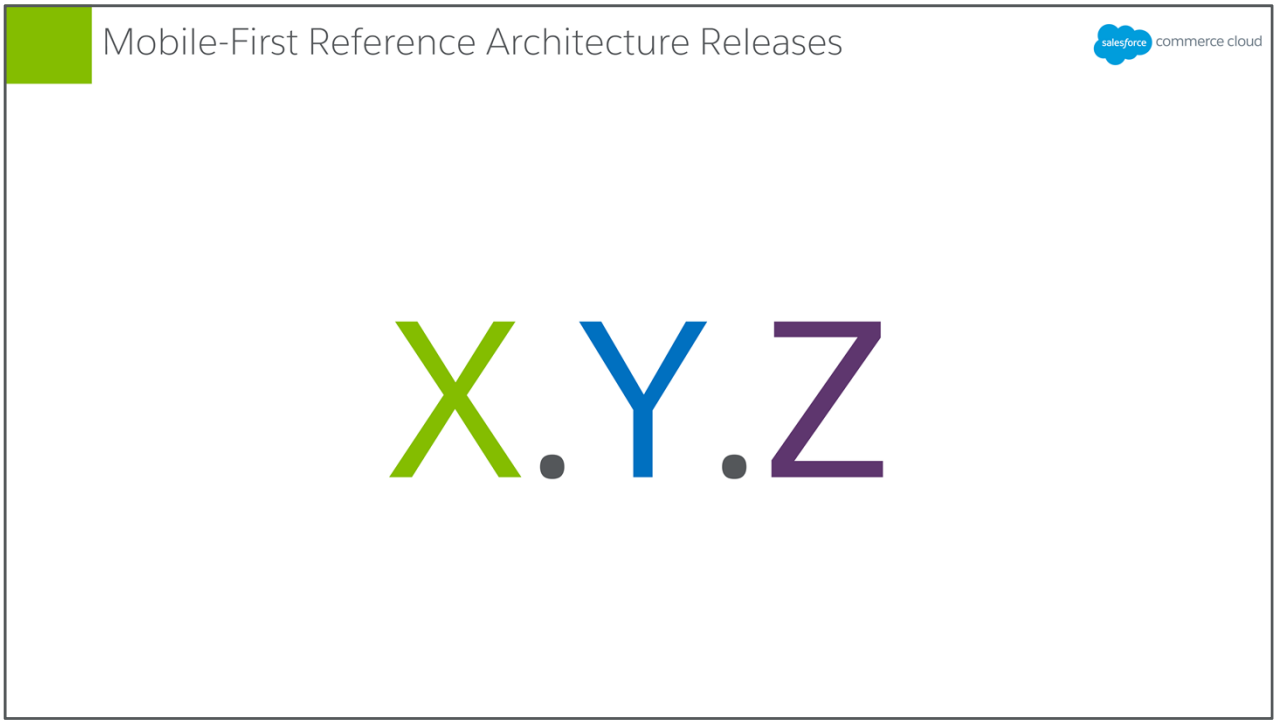
This base cartridge contains the `app_storefront_base` cartridge, which holds core storefront functionality and features that can be configured in the Business Manager. Changes to this cartridge should be made only by the Commerce Cloud team or through contributions to the project repositories on Bitbucket. Using an overlay, instead of altering the base code, enables you to upgrade to newer versions of the Mobile-First Reference Architecture without having to pick changes and perform merges manually. Depending upon the release, you may still need to make adjustments to your custom code, but the process should be quicker.

Plugins

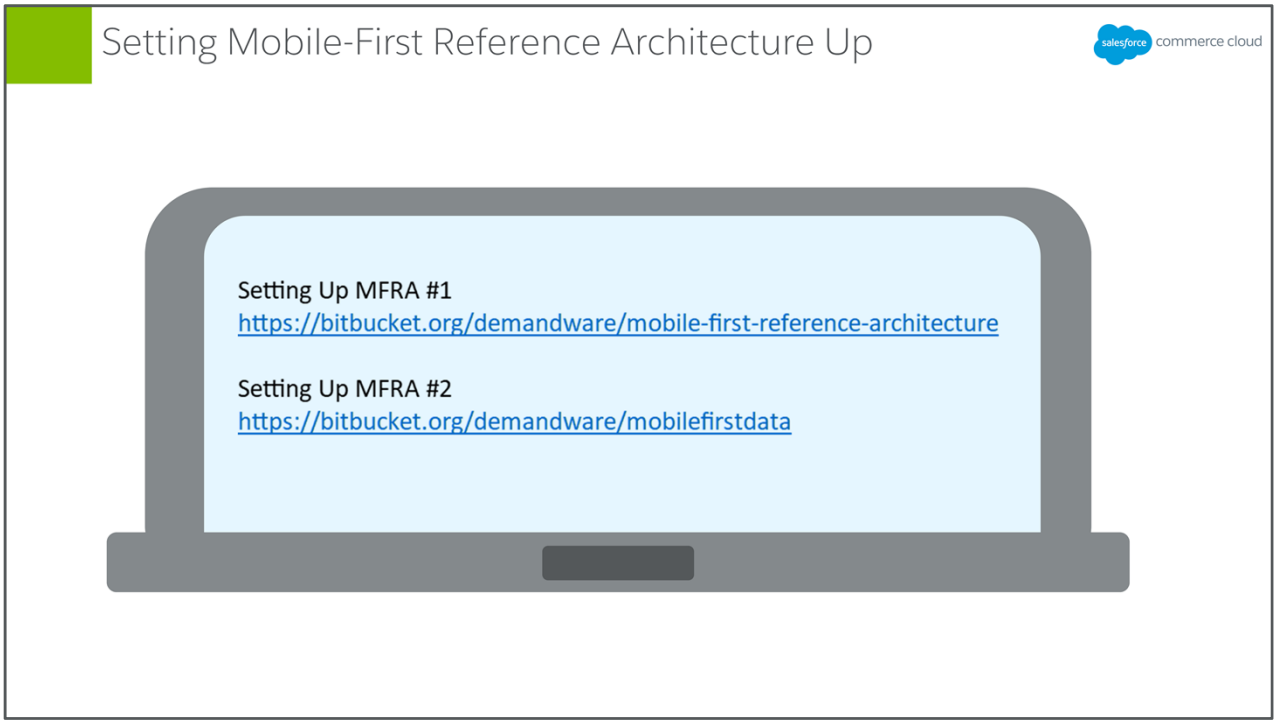
Plugin cartridges integrate external features, such as LINK Partner cartridges or optional cartridges from Commerce Cloud that enhance base functionality. Some examples are `plugin_appleypay` and plugins for tax services or product comparisons.

Merchant Branding

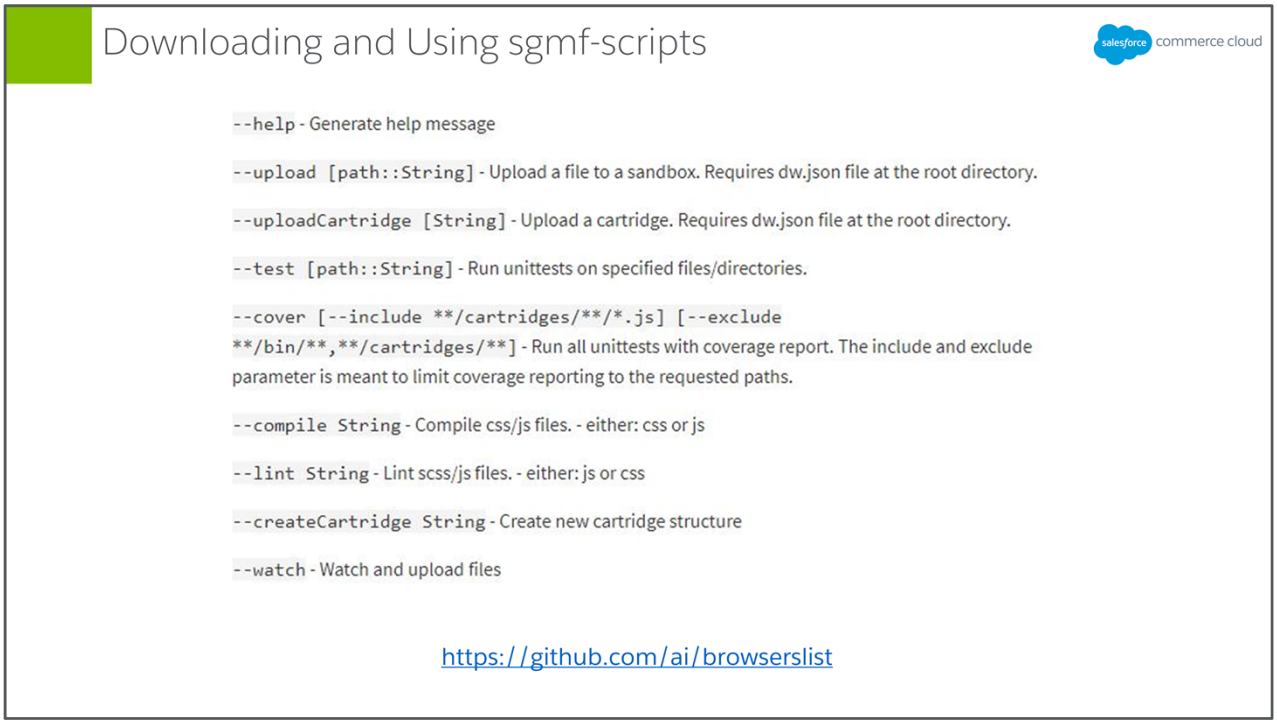
Branding cartridges contain branding and custom functionality that is specific to your storefront. All customizations to the base cartridge should be made on this layer. Branding cartridges may have names such as `app_custom_mybrand` or `app_customer_calculate`.



Commerce Cloud uses semantic versioning with three numbers to communicate major releases, minor releases, and patches to Mobile-First Reference Architecture. Each deployment should be evaluated before being pushed into production to see if it affects existing storefront functionality. Release changes will be communicated in the Commerce Cloud InfoCenter, on Xchange, and in Bitbucket change logs.



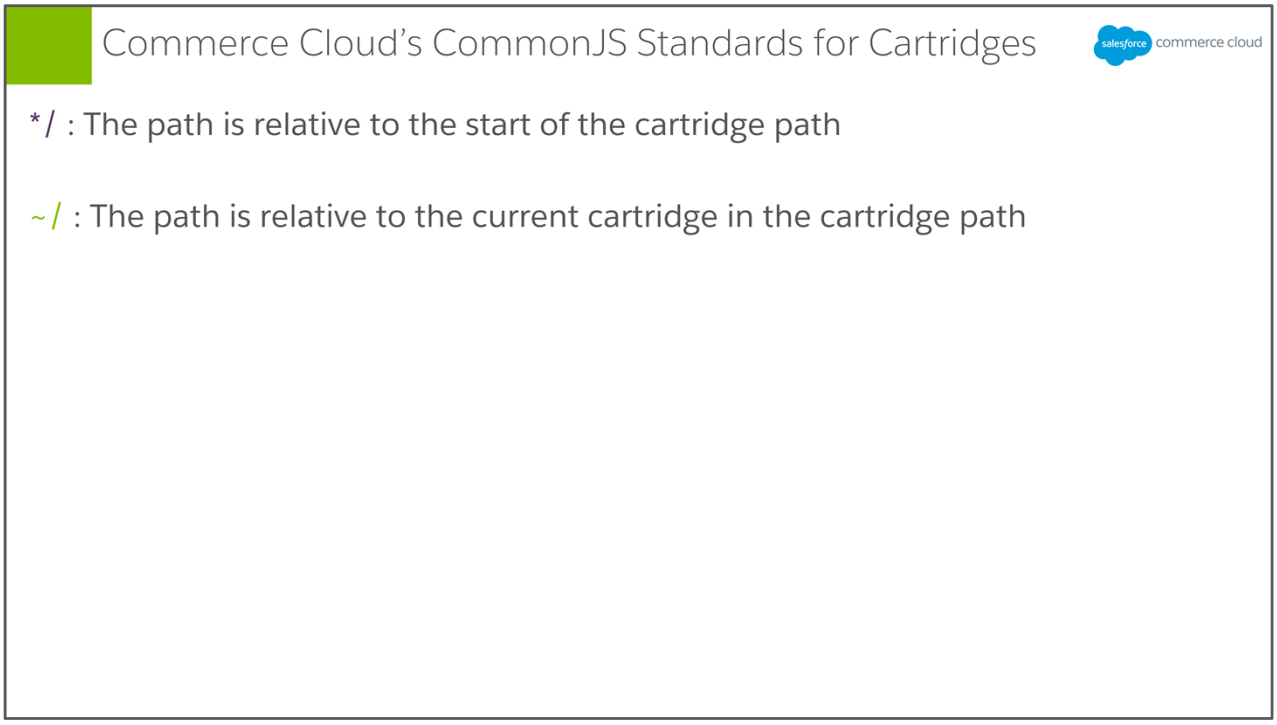
Instructions for setting up Mobile-First Reference architecture can be found on Bitbucket.



The sgmf-script repository contains a collection of scripts that are useful for creating new overlay cartridges. The sgmf-scripts repository in Bitbucket is at https://www.npmjs.com/package/sgmf-scripts. All of the scripts are executable through CLI. Commerce Cloud publishes the command-line tools in this repository as a node that can be installed by using the following NPM command: npm install sgmf-scripts --save-dev.

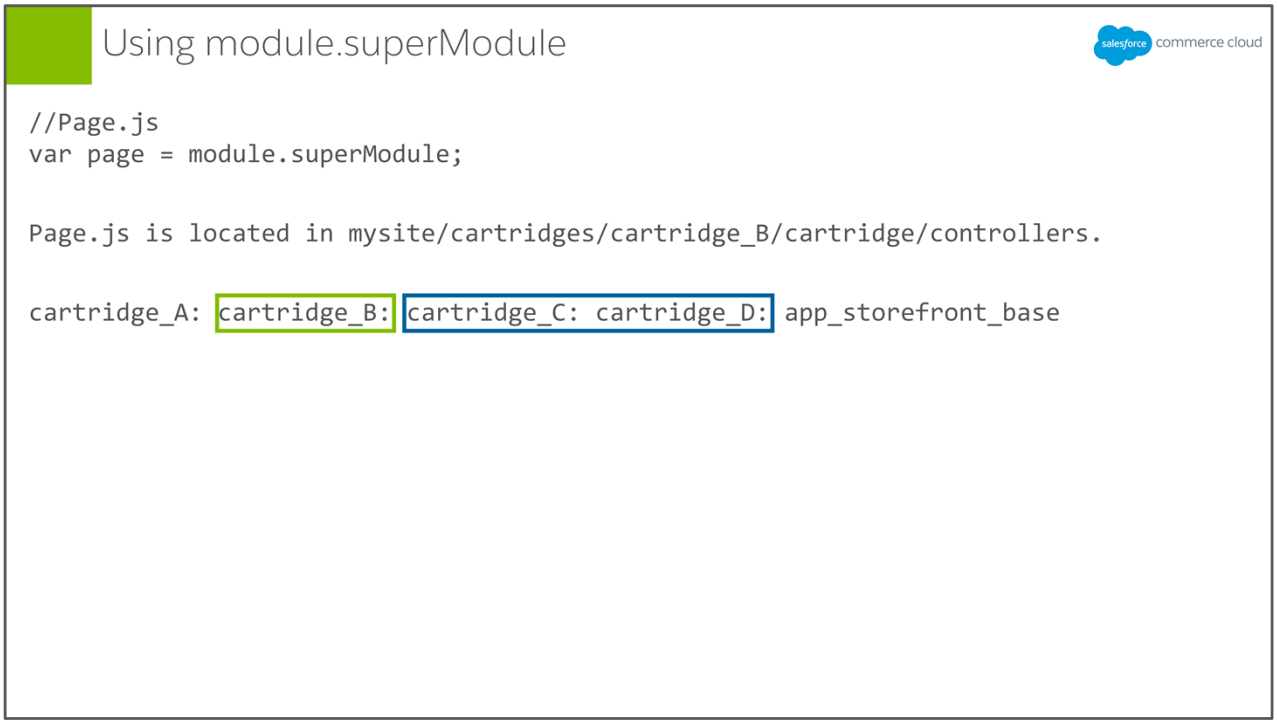
In order for all commands to work, this script makes the following assumptions:

- There's a dw.json file at the root of your repository that contains information with the path to your sandbox as well as username and password.
- There's a cartridges top level folder that contains your cartridge.
- The name property in package.json matches the name of your cartridge, or if it doesn't, there's a packageName property with the name of the cartridge.
- If this an overlay cartridge, package.json contains paths property, that's of type Object and contains key/value pairs with the name/path to all cartridges that will come below yours. For example, if you are creating a cartridge that will overlay the app_storefront_base paths property, it will look something like this: "paths": { "base": "../sgmf/cartridges/app_storefront_base" }.
- ESLint and Stylelint are dev-dependencies of your cartridge and you will have all required plugins and configs installed as well.
- There's a webpack.config.js at the top of your project that specifies how to compile client-side JavaScript files.
- Your package.json file contains browserslist key that specifies which browsers you are targeting, to compile SCSS files with correct prefixes. For more details, visit https://github.com/ai/browserslist.



Commerce Cloud is using CommonJS standards for Mobile-First Reference Architecture cartridge development with a few additions.

If a path argument starts with "*/" the path is relative to the start of the cartridge path.
If a path argument starts with "~/" the path is relative to the current cartridge in the cartridge path.

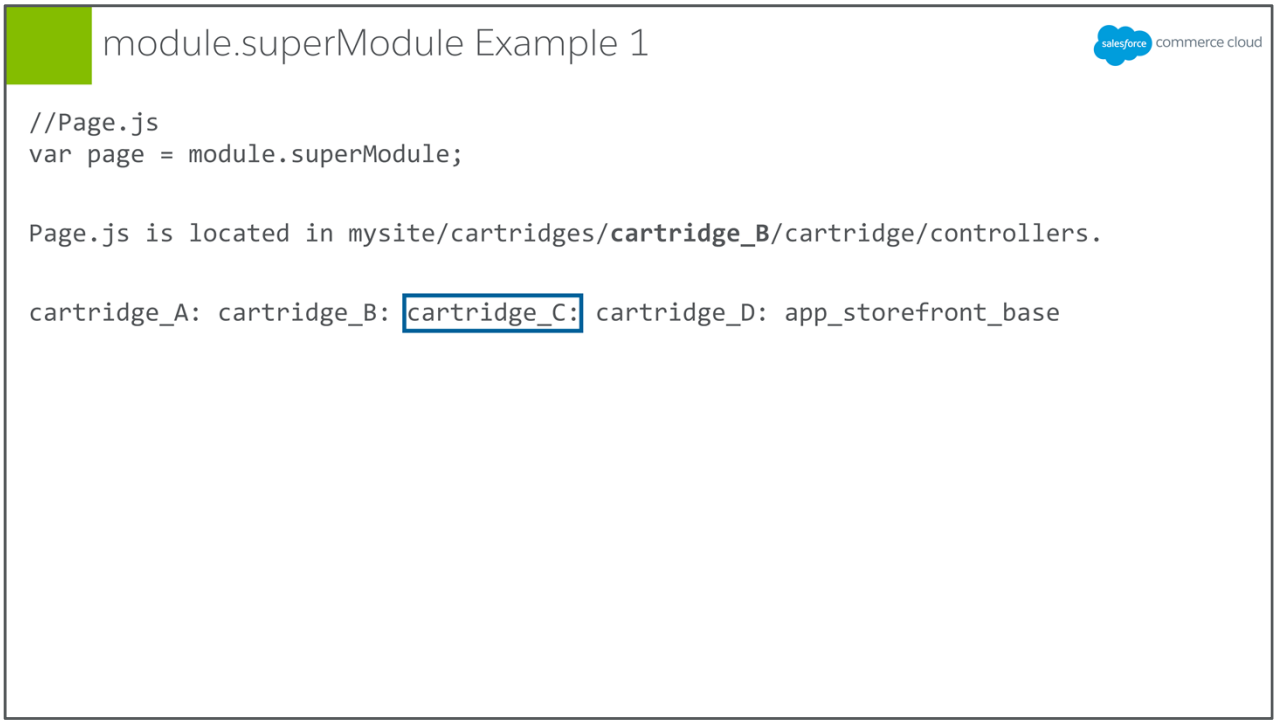


There may be instances where there are several layers of cartridges that overlay one another. When this happens, each cartridge can import properties from previous cartridges and overlay them. To make this easier, Commerce Cloud provides a chaining mechanism in module.superModule enables you to access modules that you intend to override.

The module.superModule global property provides access to files with the same path as a given file in the underlying cartridge. The system will search for the file path of the current file in every cartridge below the current cartridge until it finds a match. If no matching files are found, module.superModule returns null.

The power of the module.superModule mechanism is that it allows you to chain overlay cartridges on top of one another. For example, if cartridge_C and cartridge_D both use the module.superModule function and both add functionality, all of that functionality is available to cartridge_B.

Because chaining depends upon an expected order of cartridges in the site cartridge path, changing the order of cartridges in the cartridge path may impact your code. Therefore, it is important to think about the order of your cartridge layers in advance, to minimize changes to code.



For example, assume that you are in `Page.js` and it defines a page variable using the `module.superModule` property and the cartridge path given in this slide. The system searches the path starting with `cartridge_C`. for a file with the path `/cartridge/controllers/Page.js`.

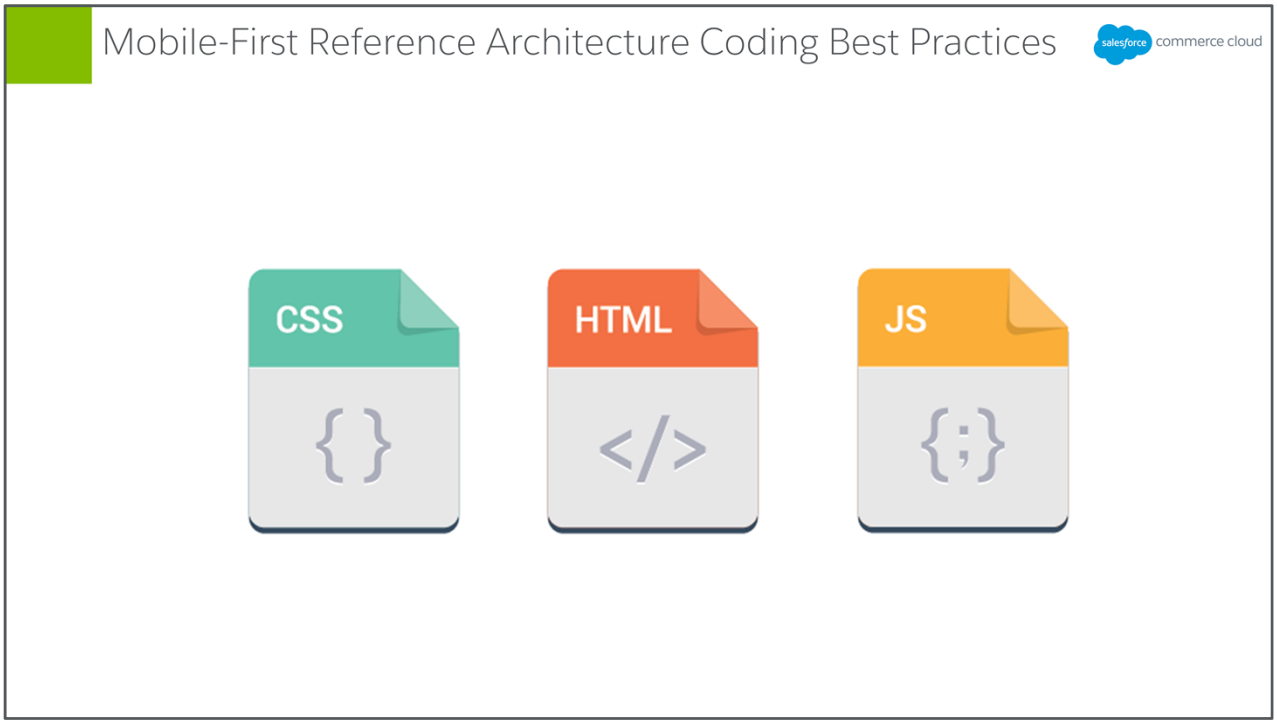
If:	The platform returns:
All cartridges contain a `Page.js` file in the same location as `cartridge_B`	The `Page.js` file in `cartridge_C` because it is the first cartridge after `cartridge_B` in the cartridge path
`cartridge_A` and `cartridge_D` contain a `Page.js` file in the same location as `cartridge_B`	The `Page.js` file in `cartridge_D` because it is after `cartridge_B` in the cartridge path.
Only `cartridge_A` contains `Page.js` file in the same location as `cartridge_B`	null because `cartridge_A` is not after `cartridge_B` in the path
All cartridges contain a `Page.js` file in a different location than `cartridge_B`	null because none of the files match the location of the file in the current cartridge.
`cartridge_C` contains a `Page.js` file in a different location than `cartridge_B` and `app_storefront_base` contains a `Page.js` file in the same location as `cartridge_B`	The `Page.js` file in `app_storefront_base` because the files must match both name and location.

module.superModule Example 2

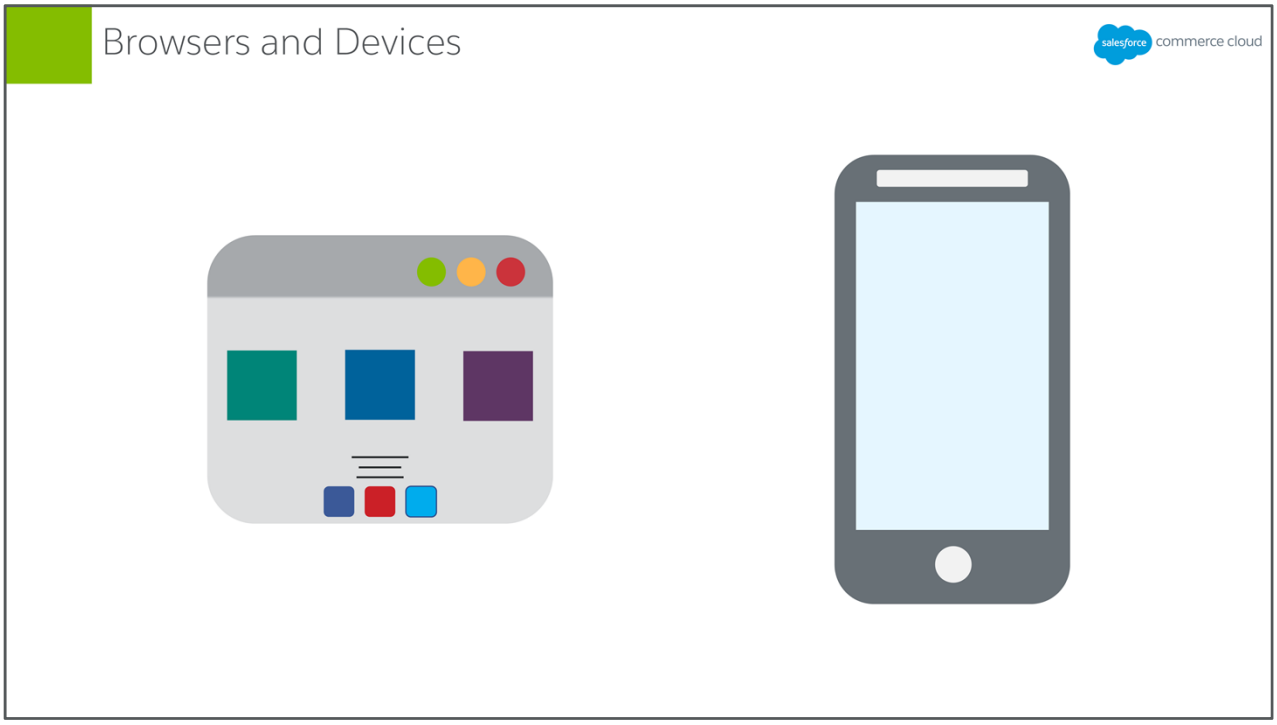
```
//Page.js
var page = module.superModule; // require functionality from last
controller in the chain with this name
var server = require('server');

server.extend(page);

server.append('Show', function(req, res, next) {
res.setViewData({ value: 'Hello Commerce Cloud' });
next();
});

module.exports = server.exports();
```

`module.superModule` gives you access to an object that results from executing code in an underlying cartridge. The resulting object can be modified. For example, it can be overridden, extended, or inherited.

This is an example of overriding and extending an existing controller. This example inherits the `Page.js` module from the last cartridge in the stack and overrides the `Show` function. Instead of displaying the response from the Show function, the page displays the `setViewData` value "Hello Commerce Cloud.“



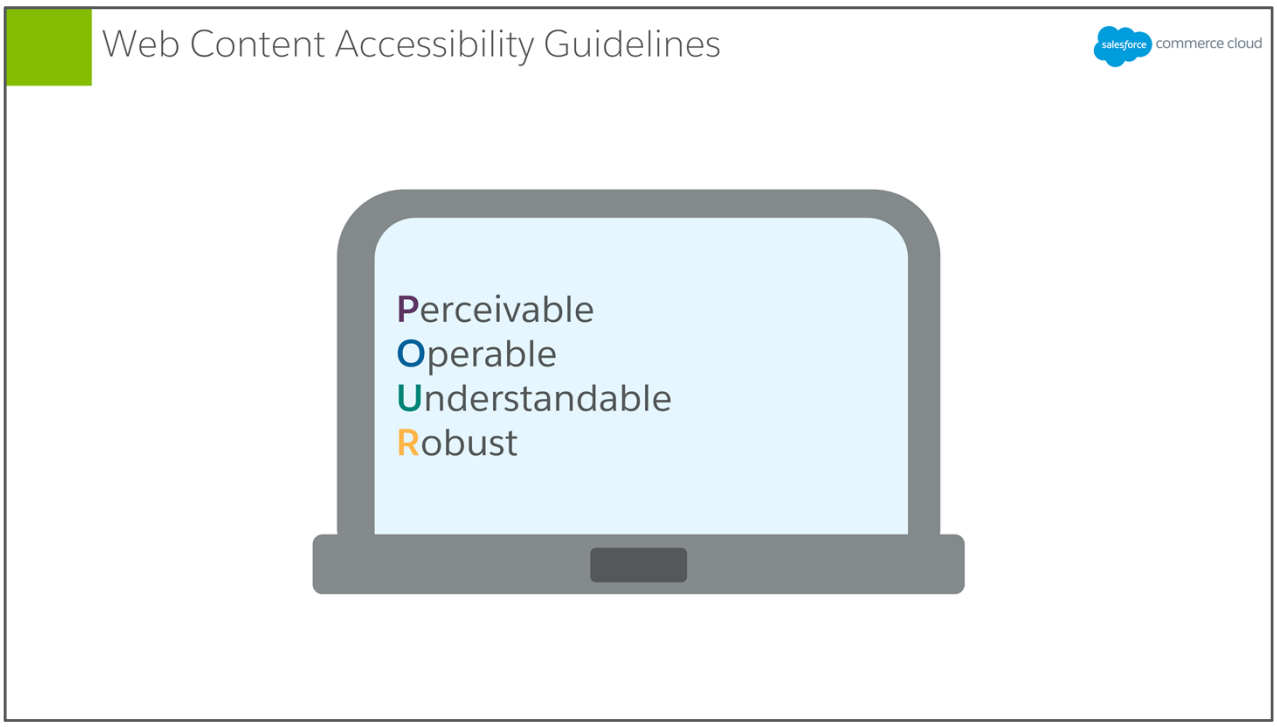
Coding best practices for Mobile-First Reference Architecture can be found in the Commerce Cloud InfoCenter and in the LINK Integration Guide, which you can access in the Best Practices section on XChange.



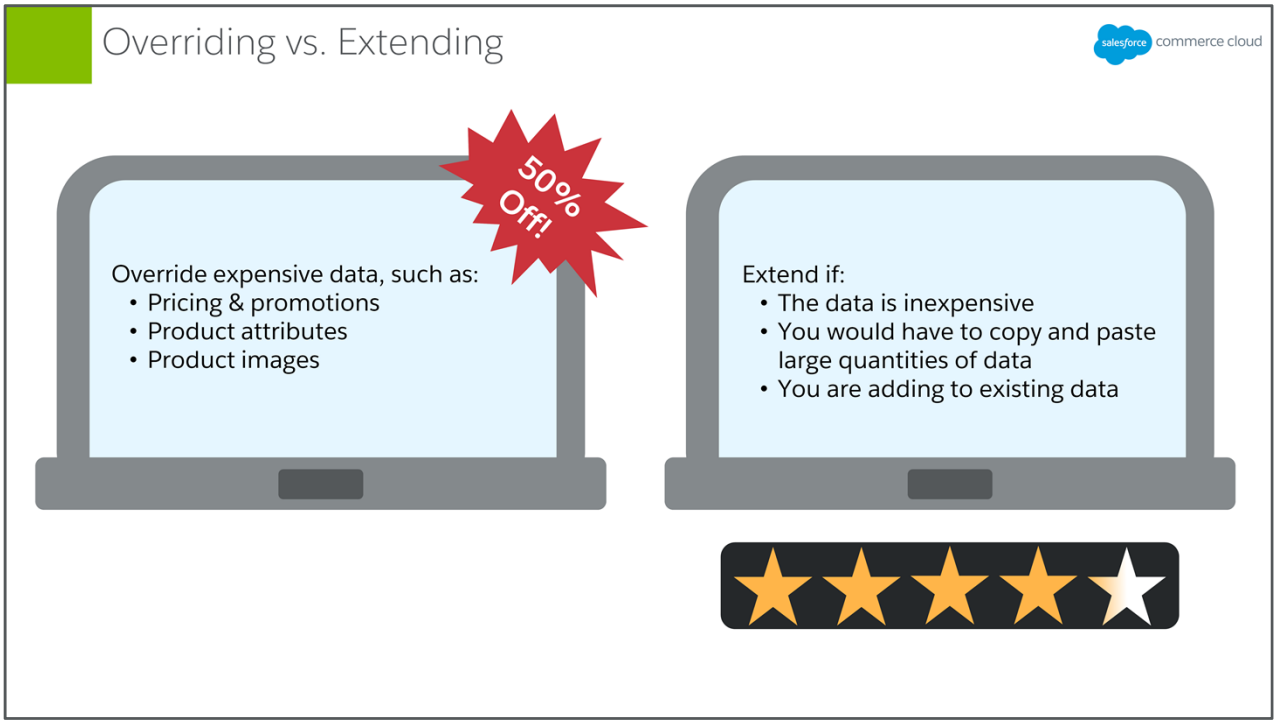
Mobile-First Reference Architecture is built to be compatible with the two latest versions of the following browsers:

- Internet Explorer
- Edge
- Safari
- Firefox
- Chrome

In addition to working on desktops and laptops, Mobile-First Reference Architecture is built to be compatible with Android and iOS devices. For the most up to date compatibility list, visit the Commerce Cloud InfoCenter.



The Mobile-First Reference Architecture's user interface was tested against the WCAG 2.0 Level A and Level AA accessibility standards. For example, the corporate colorsheet aligns with accessible contrast standards, and tab-based navigation has been improved for screen readers. Although Commerce Cloud attempts to conform to Level AA test standards, compliance with WCAG is the sole responsibility of the merchant.

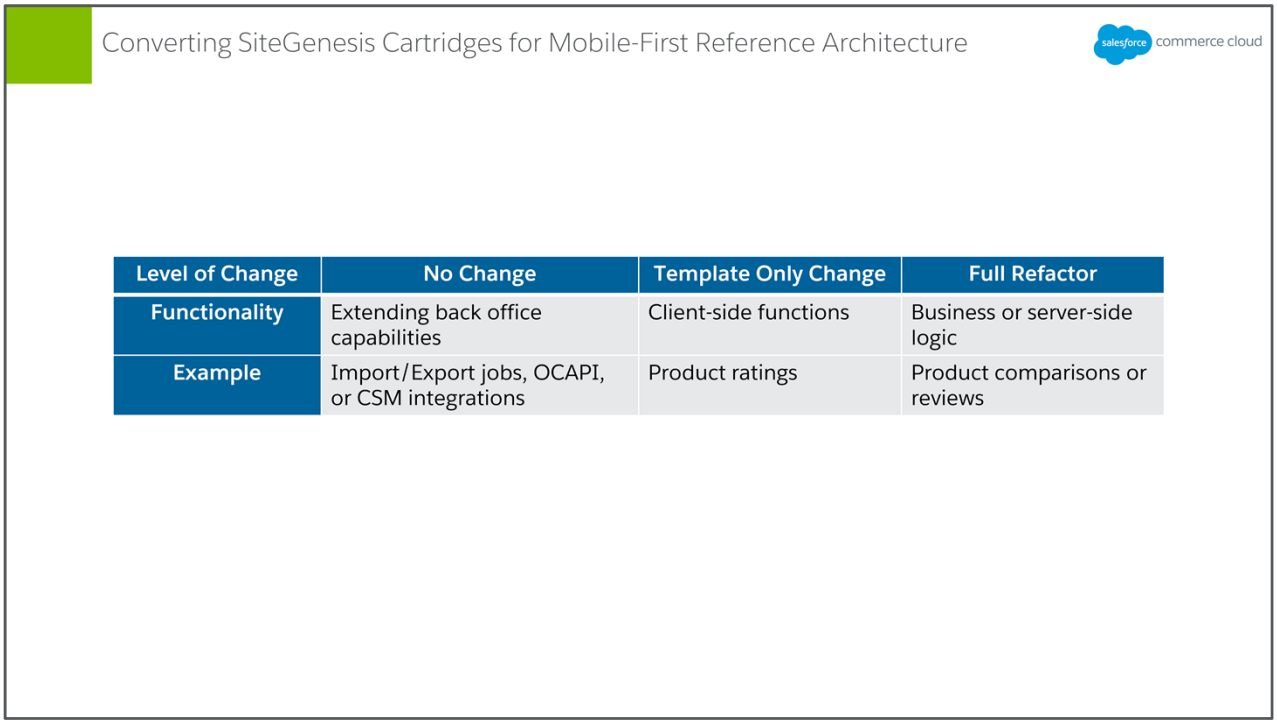


Although cartridges are used to extend the functionality, sometimes it is better to completely override a function, and in some cases it is required.

It is better to override if the data being replaced was expensive to create in the base implementation. Examples of costly data are anything to do with pricing and promotions, product attributes, and product images.

It is better to extend if the data was inexpensive to create, if you would be required to copy and paste large amounts of code, or if you are adding to existing data. A good example of appropriate extension is adding a ratings function for products.

Functionality defined in the template cannot be extended. It must be overridden.



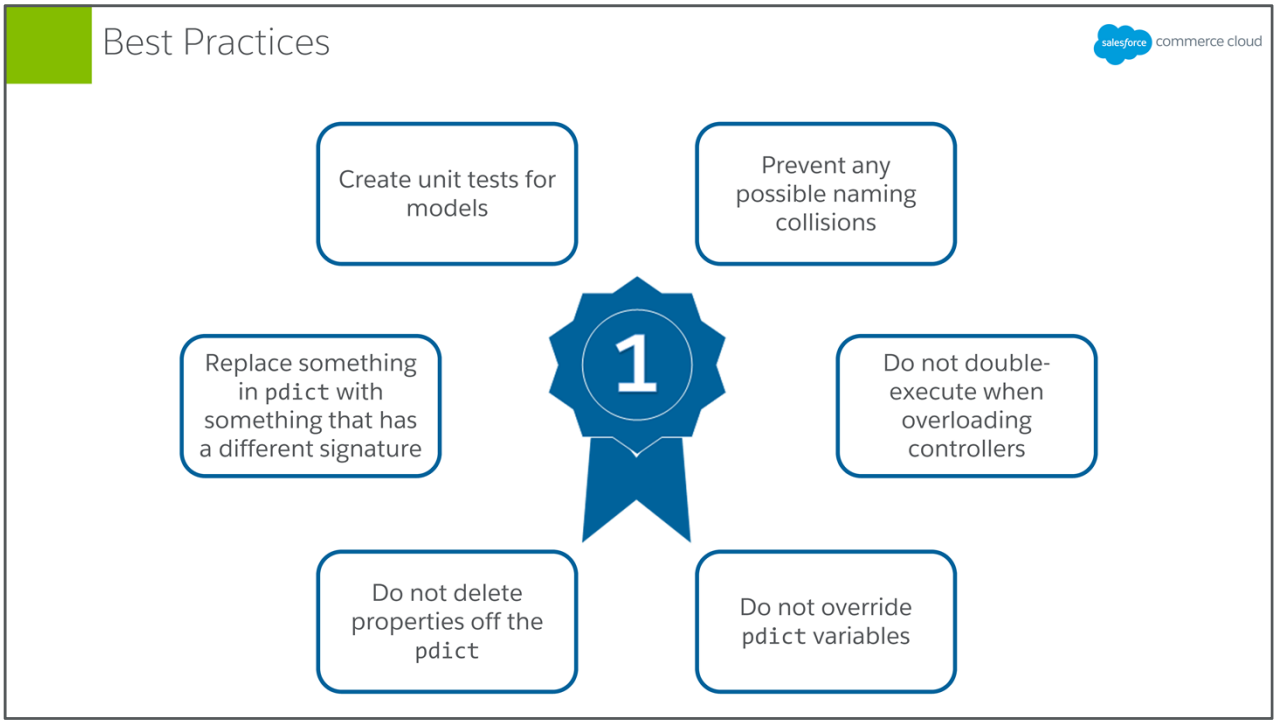
You may currently have cartridges created for storefronts built on Commerce Cloud's SiteGenesis reference application. Depending upon the functionality of the cartridge, you may need to make changes to the cartridge to make it compatible with the Mobile-First Reference Architecture platform. You will need a Bitbucket account to access the code examples mentioned in this slide.

If a cartridge extends only back office capabilities such as import and export jobs, OCAPI, or CSM integrations, no changes are needed to make the cartridge Mobile-First Reference Architecture compliant.

If a cartridge changes things only on the client side, then you will need to make changes to the template. An example of this kind of change can be found on the Course Materials tab of this eLearning and is called plugin_ratings.

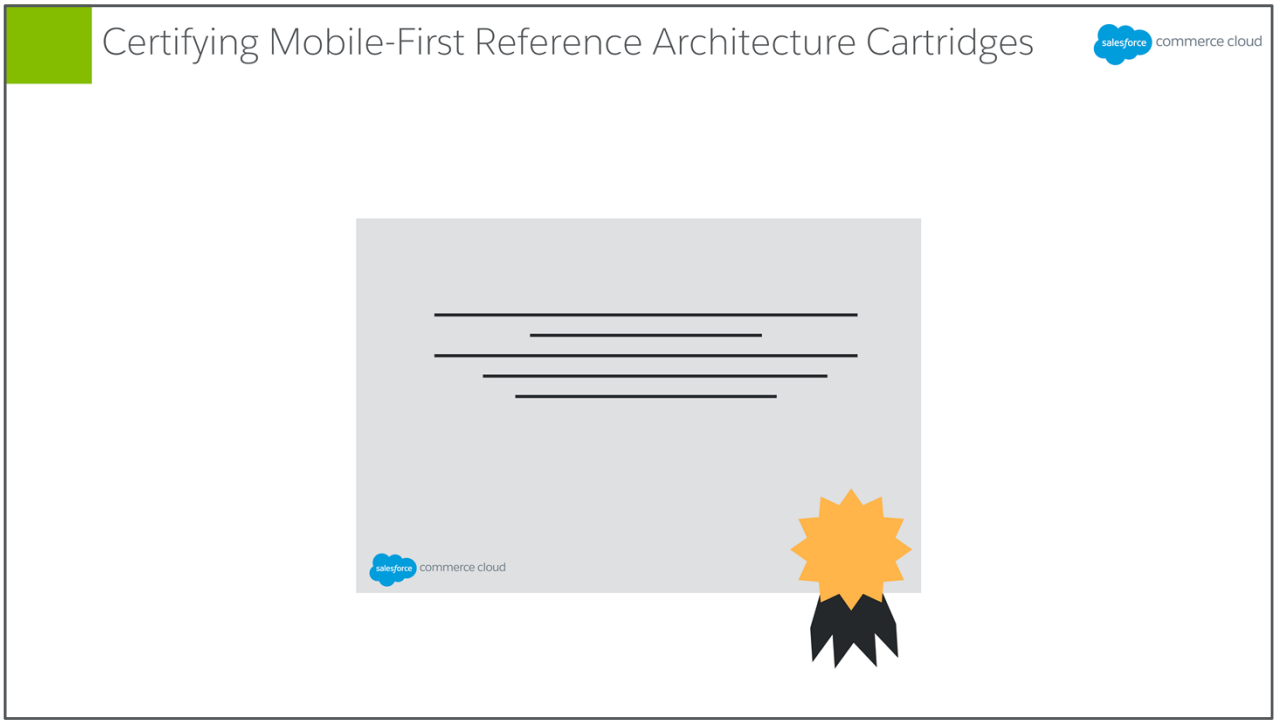
If a cartridge involves business or server-side logic such as product reviews or product comparisons, they will need to be refactored to fit inside the MVC model. Three examples of these kinds of code changes can be found in the Course Materials tab of this eLearning. They are:

- `plugin_productcompare` which modifies controllers, models, and views.
- `plugin_reviews` which builds on the ratings plugin and adds controllers and styling.
- `plugin-applepay` which is an example of adding a new payment method and changing business logic.



When creating a custom cartridge we recommend the following as a set of best practices.

1. Create unit tests for models.
2. Be aware of and prevent any possible naming collisions between cartridges.
3. Do not double-execute when overloading controllers.
4. Do not override `pdict` variables.
5. Do not delete properties off the `pdict.`
6. Replace something in `pdict` with something that has a different signature.



To be certified, a cartridge must adhere to these Commerce Cloud standards:

1. The cartridge has a complete set of unit tests for any models implemented. These unit tests should be based on the unit test framework that comes bundled with Mobile-First Reference Architecture.
2. The cartridge is completely plug-and-play. Commerce Cloud will not allow modification to the base cartridge. Doing this means that testing the cartridge should be as easy as:
    - Running the unit tests and verifying that they run without errors.
    - Loading the cartridge onto the server.
    - Adding it to the cartridge path.
    - Configuring it in the Business Manager.
    - Testing it and the tests should all run correctly without any code modifications.
3. Code is developed using the ESlint code style checking supplied with Mobile-First Reference Architecture. Please adopt the code styles supplied in the built-in eslint files
4. Newly added visual elements are built as `<isincludes>` so that they can be overridden by the customer's customization layer.
5. Documentation reflects the new integration model.

When a cartridge is submitted, the Commerce Cloud cartridge certification team installs the cartridge according to the installation guide to verify that it is coherent and factually accurate. The team also runs the supplied test cases and checks the error logs, deprecated API logs, and quota logs. The submitted cartridge is tested on controller-based sites to ensure that it works. Additionally, the cartridge will be checked for adherence to best practices for custom object usage, services, frameworks, and job frameworks.

The Commerce Cloud cartridge certification team does not conduct load testing, perform extreme functional testing, or compare a company's marketing materials to the features implemented in the cartridge.

Knowledge Check

salesforce commerce cloud

You would like to display customer ratings and feedback for your company's products. This feature is added through:

- ○ A plugin cartridge.
- ○ The branding cartridge.
- ○ The base cartridge.

Knowledge Check

salesforce commerce cloud

You would like to display customer ratings and feedback for your company's products. This feature is added through:

- ● A plugin cartridge.
- ○ The branding cartridge.
- ○ The base cartridge.

Knowledge Check

salesforce commerce cloud

Salesforce Commerce Cloud released a patch update to version 1.2.0. Which version number would represent this release?

- ○ 1.2.1
- ○ 2.2.0
- ○ 1.3.0

Knowledge Check

salesforce commerce cloud

Salesforce Commerce Cloud released a patch update to version 2.1.0. Which version number would represent this release?

- ● 1.2.1
- ○ 2.2.0
- ○ 1.3.0

Knowledge Check

salesforce commerce cloud

For sgmf-scripts to work, the `dw.json` file at the root of your repository must contain:

- ○ A path to all cartridges and your username and password.
- ○ The path to your sandbox and to all cartridges.
- ○ The path to your sandbox and your username and password.

Knowledge Check

salesforce commerce cloud

For sgmf-scripts to work, the `dw.json` file at the root of your repository must contain:

- ○ A path to all cartridges and your username and password.
- ○ The path to your sandbox and to all cartridges.
- ● The path to your sandbox and your username and password.

Knowledge Check

salesforce commerce cloud

What comes at the beginning of a path that is relative to the start of the cartridge path?

- ○ !/
- ○ */
- ○ ~/

Knowledge Check

salesforce commerce cloud

What comes at the beginning of a path that is relative to the start of the cartridge path?

- ○ !/
- ● */
- ○ ~/

Knowledge Check

salesforce commerce cloud

Best practices for developing cartridges are contained in:

- ○ The Best Practices & Guidelines page on XChange
- ○ Cartridge Development Best Practices
- ○ The LINK Integration Guide

Knowledge Check

salesforce commerce cloud

Best practices for developing cartridges are contained in:

- ○ The Best Practices & Guidelines page on XChange
- ○ Cartridge Development Best Practices
- ● The LINK Integration Guide

Knowledge Check

salesforce commerce cloud

You are creating a cartridge that would implement a pricing change through promotions. What is the best way to achieve this change?

- ○ Override the pricing data on the site.
- ○ Extend the pricing data on the site.
- ○ Override or extend the pricing data on the site. These methods work equally well.

Knowledge Check

salesforce commerce cloud

You are creating a cartridge that would implement a pricing change through promotions. What is the best way to achieve this change?

- ● Override the pricing data on the site.
- ○ Extend the pricing data on the site.
- ○ Override or extend the pricing data on the site. These methods work equally well.

Knowledge Check

salesforce commerce cloud

Which part of a storefront must be overridden?

- ○ The view
- ○ The template
- ○ The model

Knowledge Check

salesforce commerce cloud

Which part of a storefront must be overridden?

- ○ The view
- ● The template
- ○ The model

Knowledge Check

salesforce commerce cloud

A cartridge built for SiteGenesis affects an import/export job. What changes are needed to make the cartridge compatible with the Mobile-First Reference Architecture?

- ○ No changes are needed.
- ○ Only template changes are needed.
- ○ The cartridge must be fully refactored.

Knowledge Check

salesforce commerce cloud

A cartridge built for SiteGenesis affects an import/export job. What changes are needed to make the cartridge compatible with the Mobile-First Reference Architecture?

- ● No changes are needed.
- ○ Only template changes are needed.
- ○ The cartridge must be fully refactored.

Knowledge Check

When you submit a cartridge to Commerce Cloud for certification, the cartridge certification team:

- ○ Performs load testing.
- ○ Verifies that the cartridge functionality matches marketing materials.
- ○ Verifies that the installation guide is accurate.

Knowledge Check

salesforce commerce cloud

When you submit a cartridge to Commerce Cloud for certification, the cartridge certification team:

- ○ Performs load testing.
- ○ Verifies that the cartridge functionality matches marketing materials.
- ● Verifies that the installation guide is accurate.

Course Summary

salesforce commerce cloud

Having completed this course, you should now be able to:

- Build a cartridge for a storefront on the Mobile-First Reference Architecture.

salesforce commerce cloud

Congratulations!

You have completed *Cartridges and Commerce Cloud Mobile-First Reference Architecture.*

Your satisfaction is very important to us. Click on the **Course Survey** link emailed to you at the end of this course.