Pop Up Module

Contents

[Objective 2](#_Toc88591399)

[Prerequisite 2](#_Toc88591400)

[Module Implementation. 2](#_Toc88591401)

[Definition file explained. 3](#_Toc88591402)

[Resource strings. 3](#_Toc88591403)

[App level configurations. 3](#_Toc88591404)

[Business logic (pop-up.tsx) 4](#_Toc88591405)

[View (pop-up.view.tsx) 4](#_Toc88591406)

[Data action (actions/pop-up-state-data-actions.ts) 4](#_Toc88591407)

[Use case: Nested module alter the state of pop up 4](#_Toc88591408)

[CMS configuration 5](#_Toc88591409)

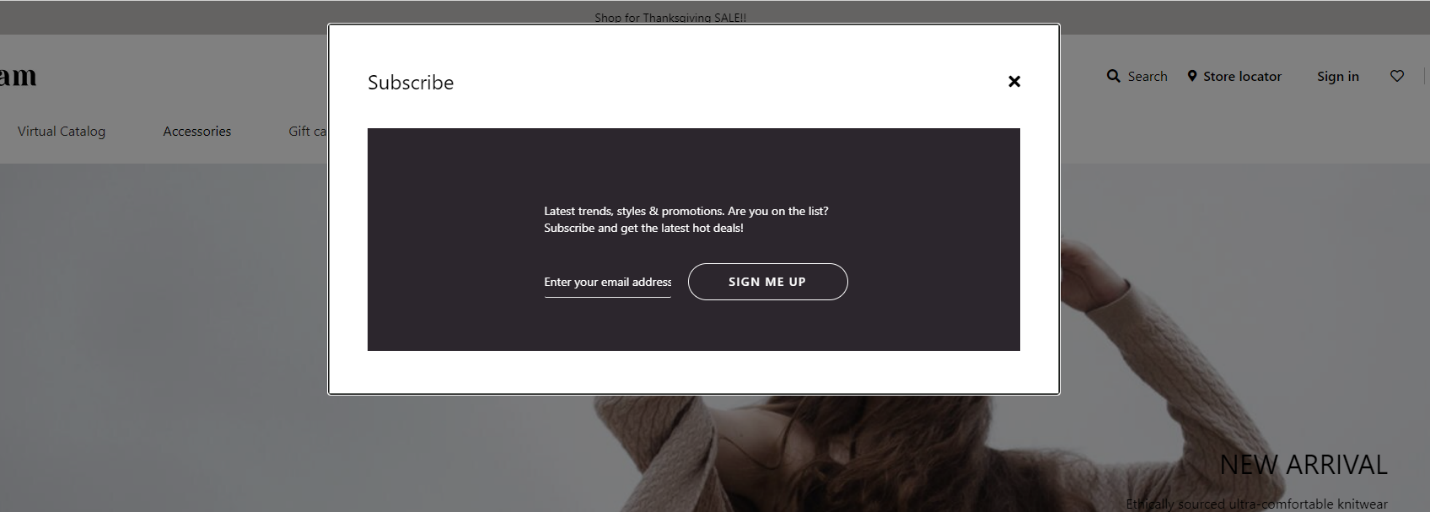
[Steps to test the sample module. 5](#_Toc88591410)

[References for module/theme extensibility and deployment. 7](#_Toc88591411)

# Objective

This is a generic sample code for implementing a pop-up module. Pop-up module can be added anywhere in the page, and it will create a modal dialog and attach it to the DOM. Other modules could be exposed inside this module to make them appear inside the pop-up. It can be set to pop up at the page load or anytime during the session. It can be associated with a cookie to determine whether certain action has taken place.

**Sample pop-up implementation with nested subscribe module snapshot:**

****

# Prerequisite

* SSK Module code need to be deployed and running.

Reference link: <https://docs.microsoft.com/en-us/dynamics365/commerce/e-commerce-extensibility/setup-dev-environment>

* Pop up configuration and its properties needs to be configured from authoring tool.

Reference link: <https://docs.microsoft.com/en-us/dynamics365/commerce/work-with-modules>

# Module Implementation.

Follow the process below to get started on pop-up module.

* 1. Unzip the pop-up source code.
  2. Copy the module code of pop-up module under “/src/module” as below:

Text

Description automatically generated

# Definition file explained.

Please refer to the table below for configurable settings for this module based on the need.

**Configuration (pop-up.definition.json)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Default** |
| heading | String | Heading for the pop-up | - |
| className | String | A custom class name to target the CSS for this module. | **-** |
| isBackDropStatic | Boolean | Option to choose a static backdrop. Static backdrop doesn’t close modal on click. | false |
| isCookieEnabled | Boolean | Option to associate pop-up view based on the presence or value of cookie. | false |
| shouldPopUpOnLoad | Boolean | Option to choose if the pop-up should pop-up on initial load. | false |

# Resource strings.

|  |  |  |
| --- | --- | --- |
| **Key** | **Value** | **Description** |
| popUpAriaLabel | Pop Up | Aria label for Pop Up |

# App level configurations.

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Name** | **Type** | **Description** | **Default** |
| popUpCookieName | String | Optional cookie name for pop up module. It is a persistent cookie and pop-up module checks for its existence or its value before rendering the pop up. Writing this cookie after certain event should be taken care in nested module. | \_msdyn365\_\_popUp\_ |

# Business logic (pop-up.tsx)

This module will be attached to the DOM as a modal dialog. It will contain other modules as per the need, configurable through CMS (i.e., Content Management System). Its state is determined by the configuration and data action. If cookie is enabled, presence of cookie and it’s truthy will be checked before displaying the modal dialog.

# View (pop-up.view.tsx)

There is a single view for this module, which is as a modal dialog.

# Data action (actions/pop-up-state-data-actions.ts)

This data action could be used from any module to alter the state of pop-up module like below:



# Use case: Nested module alter the state of pop up

This is only applicable if we want nested module to alter the state of pop up and/or write cookie. This can be implemented with any nested module. In this sample case, we are working with ‘subscribe’ module.

1. **Eject ‘subscribe’ module.**

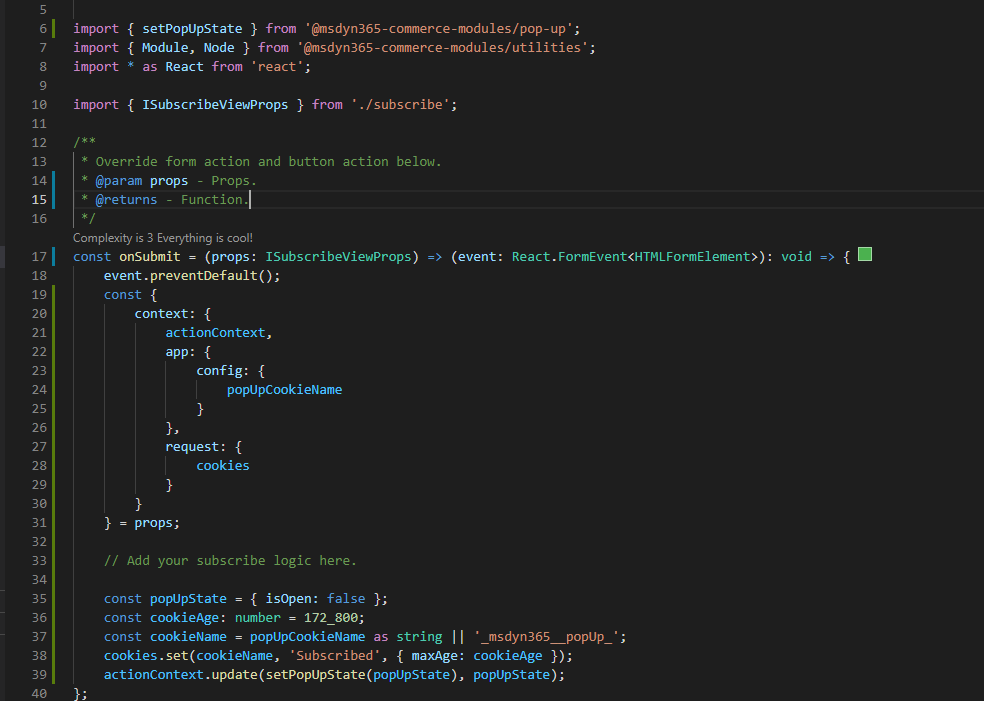
Reference link: <https://docs.microsoft.com/en-us/dynamics365/commerce/e-commerce-extensibility/clone-starter-module>

1. **Assuming popup module is deployed. In package.json file, under dependencies, add dependency for pop-up module.**

Text

Description automatically generated

1. **Update onSubmit click handler, function in subscribe.view.tsx file and import ‘setPopUpState’**

****

**Text

Description automatically generated**

# CMS configuration

1. Open site authoring tool, click on any container, click three dots to add pop-up module.
2. Add nested modules inside the pop-up module similarly.
3. Edit module configuration as per the need.
4. Save and publish the changes.
5. Browse your site to test.

# Steps to test the sample module.

1. **Build the repository**:

Execute yarn build from command line interface on your repository.

1. **Module Styling**:

CSS for this module(pop-up.scss) file should be placed in styles folder and the reference should be made in index.scss.

1. **Testing**:

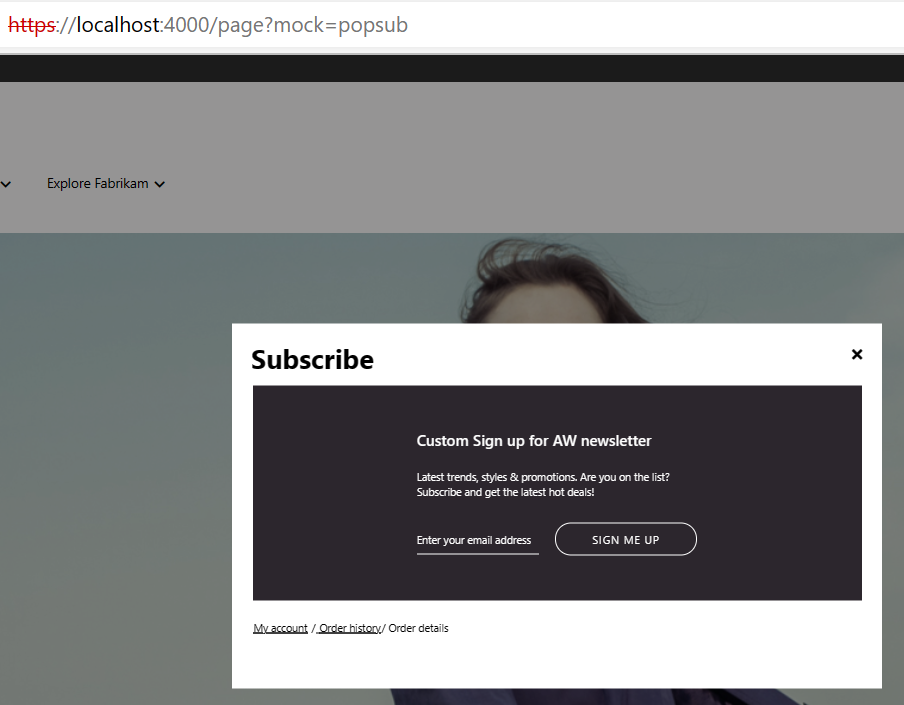
Testing can be done by adding mock to the sample-file.json mock in pageMocks and hitting your localhost like this: <https://localhost:4000/?mock=sample-file>

**Note**: The sample code is tested on below package version:

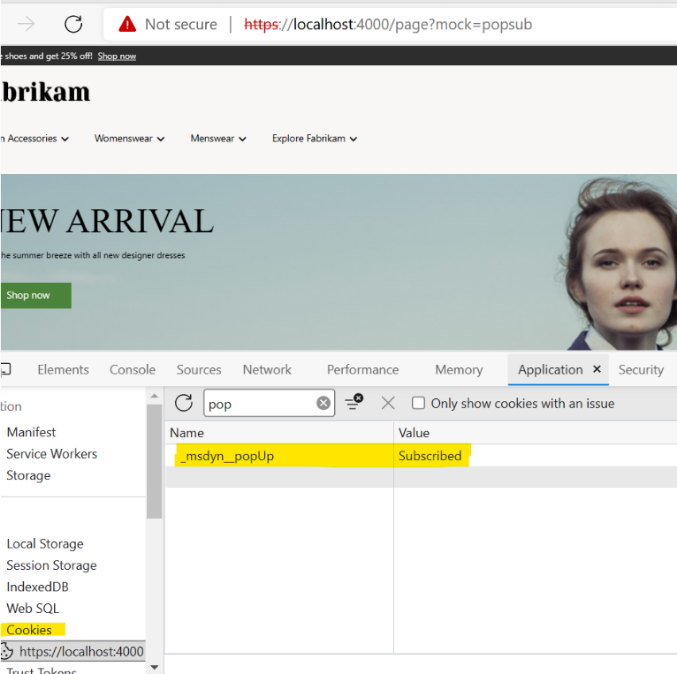
**Graphical user interface, text

Description automatically generated**

**Refer screenshots for PopUp module with subscribe module as nested module:**



**Cookie updation:**



# References for module/theme extensibility and deployment.

* **Extend a theme to add module extensions.**

<https://docs.microsoft.com/en-us/dynamics365/commerce/e-commerce-extensibility/theme-module-extensions>

* **Extend a theme from a base theme.**   
  <https://docs.microsoft.com/en-us/dynamics365/commerce/e-commerce-extensibility/extend-theme>
* **Clone a module library module.**

<https://docs.microsoft.com/en-us/dynamics365/commerce/e-commerce-extensibility/clone-starter-module>

* **package creation and deployment**

<https://docs.microsoft.com/en-us/dynamics365/commerce/e-commerce-extensibility/package-deploy>