Unity Addressable Asset system

The Addressable Asset system provides an easy way to load assets by “address”. It handles asset management overhead by simplifying content pack creation and deployment.

The Addressable Asset system uses asynchronous loading to support loading from any location with any collection of dependencies. Whether you use direct references, traditional [AssetBundles](https://docs.unity3d.com/Manual/AssetBundlesIntro.html), or Resource folders for asset management, Addressable Assets provide a simpler way to make your game more dynamic.

What is an asset?

An asset is content that you use to create your game or app. Common examples of assets include Prefabs, textures, materials, audio clips, and animations.

What is an Addressable Asset?

Making an asset "Addressable" allows you to use that asset's unique address to call it from anywhere. Whether that asset resides in the local application or on a content delivery network, the Addressable Asset System locates and returns it. You can load a single Addressable Asset via its address, or load many Addressable Assets using a custom group label that you define.

Why use Addressable Assets?

Traditional means of structuring game assets make it challenging to efficiently load content. Addressables shorten your iteration cycles, allowing you to devote more time to designing, coding, and testing your application.

* **Iteration time**: Referring to content by its address is extremely efficient. Optimizations to the content no longer require changes to your code.
* **Dependency management**: The system returns all dependencies of the requested content, so that all meshes, shaders, animations, and so forth load before returning the content to you.
* **Memory management**: The system unloads assets as well as loading them, counting references automatically and providing a robust profiler to help you spot potential memory problems.
* **Content packing**: Because the system maps and understands complex dependency chains, it allows for efficient packing of bundles, even when moving or renaming assets. You can easily prepare assets for both local and remote deployment, to support downloadable content and reduced application size.
* **Profiles**: The system allows you to create a set of string variables that more easily enables you to change how your content is built into bundles without modifying settings in multiple places.

What about my existing game?

The Addressable Asset System provides a [migration path](https://docs.unity3d.com/Packages/com.unity.addressables@1.16/manual/AddressableAssetsMigrationGuide.html) for upgrading, whether you use direct references, Resource folders, or AssetBundles.