Module API Specifiers

Exposing classes and functions to other modules.

The easiest way to think about these specifiers is that they are used to tag functions, classes or data as public to your module's DLL file. If you mark a function in the Engine module as `ENGINE_API`, then any module that imports Engine can access that function directly.

These are only used when compiling the engine in modular mode (DLL files on desktop platforms). The opposite is what we call monolithic mode, which puts all code together in a single executable file. The type of build is controlled by UnrealBuildTool settings and/or platforms and build configuration.

The actual API macro equates to one of the following depending on how the code is being compiled by UBT:

- `__declspec( dllexport )`, when compiling module code in modular mode.
- `__declspec( dllimport )`, when including public module headers for a module that you are importing.
- empty, when compiling in monolithic mode

API macros only make sense for modules that are statically imported from another module. The Core module is a great example -- almost every other module in UE4 specifies Core as a import dependency in their *.Build.cs file.

Many modules never need to be statically imported (e.g. SceneOutliner module.) We refer to those modules as dynamically loaded modules. Dynamically loaded modules are awesome because they can be discovered at startup (kind of like a plugin), and often can be reloaded on the fly.

The API macros are mostly used on older code to allow newer modules to access it from their DLL. In newer bits of code, the API macros are used far less, instead setting up nice interface layers to expose functionality across DLL boundaries.