ASSETMANAGER SCRIPT INTERFACE

View online: https://www.construct.net/en/make-games/manuals/construct-3/scripting/scripting-reference/interfaces/iassetmanager

The `IAssetManager` interface provides access to the project's assets, such as audio files and other project files added to your project. It is typically accessed via `runtime.assets`.

On most modern platforms, assets can be directly retrieved as if over the network with standard APIs like `fetch()` or `XMLHttpRequest`, even in non-web export platforms like mobile or desktop apps. However in some circumstances, generally with unusual export options like Playable Ads or with legacy projects, it may be necessary to use Construct's IAssetManager interface to fetch resources, as it implements workarounds that prevent standard fetches working in those environments.

AssetManager APIs

async fetchText(url)

async fetchJson(url)

async fetchBlob(url)

async fetchArrayBuffer(url)

Retrieve the contents of a given URL as a string, JSON object, Blob or ArrayBuffer. Returns a promise that resolves when the resource has been loaded.

async getProjectFileUrl(url)

Retrieve a URL that can be fetched directly for a given resource. Returns a promise that resolves to a string with a URL that may be the same as the original URL, or a different URL (e.g. `blob:` URL) if direct fetching is not supported. This is intended for using with local files where the other fetch methods are not appropriate, such as assigning the `src` attribute of a video.

async getMediaFileUrl(url)

As with `getProjectFileUrl` but for sound and music files, which are exported to a *media* subfolder.

mediaFolder

A string of the subfolder media files are in, including sound and music files. In preview this is an empty string, and after export it is the media subfolder followed by a forward slash, e.g. `"media/"`.

---

projectFileList

A read-only array with the list of project files in this project at the time of preview or export. Files in the Sounds, Music, Videos, Fonts, Icons & Screenshots and Files folders are included. Each entry is a an object with the properties `name` being the relative path to the file, e.g. `subfolder/mydata.json`, and `size` which is the file size in bytes.

> *This file list only reflects the state of the project at the time it was previewed or exported. If you export the project and then change some of the files, this list will not update to reflect the change. If you need a list that updates to post-export changes, consider another approach, such as using File System to list folder contents on desktop, or a separate data file to list files in which can also be updated after export.*

async loadScripts(...urls)
Fetch and run the JavaScript files at the given URLs. This can load scripts in the *Files* folder of the Project Bar, none of which are automatically loaded by Construct. When loading multiple scripts, they will run in the order they are provided, e.g. `loadScripts("script1.js", "script2.js")` will always run script1.js first and script2.js second. For best efficiency, try to load all the scripts you need in a single call, rather than repeated calls.

---

async compileWebAssembly(url)

Fetch and compile a WebAssembly.Module from the given URL, which is typically a .wasm file. This uses streaming compilation where supported. Note this does not instantiate the module, which needs to be done before any calls can be made. Pass the module resulting from this call to WebAssembly.instantiate() to get a WebAssembly.Instance from the module.

---

async loadStyleSheet(url)

Fetch a stylesheet at the given URL and attach it to the current document, applying its styles. Returns a Promise that resolves when the stylesheet has been applied to the document.