DESCRIPTION

This asset is used to call the OS sharing functionality on the web. It uses the web share api. It can be used to share text, images, videos, etc. In addition, this asset also provides functionality to download files.

This asset can be used as follows:

1. C#->JS. When you have a byte array that you generated/fetched from C#, and you want to share/download.
2. C#->JS. You have the file in the JS side, but you call the share/save code from C#, using the path of the file stored in JS. This is very fast and the preferred method when you already have data in the JS context, because you avoid having to transfer and decode to C# first before sharing/saving.
3. JS->JS. You can have your own jsplugin and call my js functions in it, for example. Similar to number 2, you already have the data in the JS context, but you don't want to invoke the share/download functionality from C#, but use directly from JS instead.

Please note that this asset, as all of my other jsplugin assets, don't work in the editor. Only in the actual build.

METHODS AVAILABLE

```
public static void Share(Action<status> callback, byte[] file, string mimeType, string fileName = null, string url = null, string title = null, string text = null)
```

```
public static void Share(Action<status> callback, string blobPropertyPath, string fileName = null, string url = null, string title = null, string text = null)
```

```
public static void Save(byte[] file, string mimeType, string fileName = null)
```

```
public static void Save(string blobPropertyPath, string fileName = null)
```

```
public static status CanShare(byte[] file, string mimeType, string fileName = null, string url = null, string title = null, string text =
```

```
null)
```

```
public static status CanShare(string blobPropertyPath, string fileName =
null, string url = null, string title = null, string text = null)
```

In addition to all these methods, there is also a public enum:

```
public enum status {Success, NotAllowedError, TypeError, AbortError,
DataError, InvalidStateError, CantShareError, NotABlobError,
WebShareUnsupportedError, WebShare2UnsupportedError}
```

All of these status represent what happened when you tried to share. Below is what each status means:

1. Success: The user successfully shared the file and/or text
2. NotAllowedError, TypeError, AbortError, DataError: See here.
3. InvalidStateError: This is a generic DOMException error, but in the context of the web share api, it is fired when the user tries to share something in the middle of a previous share operation.
4. CantShareError: The CanShare method returned false.
5. NotABlobError: The user tried to share a blob but the object wasn't actually a blob, or the path that leads to a property that should hold a blob, doesn't hold a blob.
6. WebShareUnsupportedError: The Web Share API, used to share text only, is not available in the target browser.
7. WebShare2UnsupportedError: The Web Share API Level 2, used to share files in addition to text, is not available in the target browser.

HOW TO SHARE

If you want to share, the first thing that you need to do is define the callback that will be called if the share method fails or succeeds. The method just takes one argument, a status enum, and doesn't return anything.

Here's an example of such a callback:

```
public void shareCallback(status stat) {
      Debug.Log( "status: " + stat.ToString());
}
```

You can share a byte array, or a blob.

If you want to share a byte array, use the following method:

```
public static void Share(Action<status> callback, byte[] file, string mimeType, string fileName = null, string url = null, string title = null, string text = null)
```

Parameters:

1. callback: Takes the callback that you defined in the first step.
2. file: Takes the byte array.
3. mimeType: Takes the mime type of the file.
4. fileName: Takes a file's name of your choosing. If the fileName's extension is not provided, the method tries to guess based on the mimeType. The way it guesses is simply by looking at what's on the other side of the /. For example, a mimeType of "video/webm" has a file extension of webm. When the mime type includes the file extension, having the file extension on the fileName is not needed. Please note that if you provide the wrong file extension or if the file extension is guessed wrong, you will get a NotAllowedError. If you don't provide a fileName at all, it will default to "file." + the guessed file extension.
5. url: Takes a url of your choosing.
6. title: Takes a title of your choosing.
7. text: Takes a text of your choosing.

Here's an example

```
ShareNSaveWebGL.Share(shareCallback, file, "image/png", "doesntmatter.png")
```

This method was designed primarily to be used when you generate or fetch the byte array from the C# side.

However, if you want to share a file that you generate on the javascript side, it's best to use the second form of share, the one that shares a blob. The reason for this is performance: if you generate a file in the javascript side, to use the share that takes a byte array you will need to convert the file to a string, send it to C#, and in C# convert it

to a byte array. This process is very, **very** slow, it can take several seconds depending on the size of the file(it took 4 seconds to share a 2MB image on my phone). Instead of sending the data to C#, you can leave it in the javascript context and just use the second method(that takes a blob) to take the file without the conversion step. This is very fast in comparison, almost instant(about ~100ms to share a 2MB image on my phone).

Here's the signature for the second method:

```
public static void Share(Action<status> callback, string blobPropertyPath, string fileName = null, string url = null, string title = null, string text = null)
```

It's very similar to the first, but instead of having the file parameter, you have the blobPropertyPath parameter, and no mimeType parameter since this information is in the blob already. It will attempt to guess the file's extension in the same way if you don't provide it or if you don't provide the file name. The rest works the same. Here's an example using my ScreenshotWebGL asset:

```
ShareNSaveWebGL.Share(shareCallback, "Module.ScreenshotWebGL.screenShotBlob", "doesntmatter.png");
```

"Module.ScreenshotWebGL.screenShotBlob" is a path that leads to a property that holds a blob. After taking a screenshot, my ScreenshotWebGL asset(not yet approved by the asset store at the time of writing) stores the blob at Module.ScreenshotWebGL.screenShotBlob(the method variant that stores a blob). Any path will work, as long as it starts with "Module", "window", or "document". Don't worry, I am not using eval or Function to do this.

In addition to taking a path that leads to a property that holds a blob, you can just pass a blob directly. This is useful if you want to call the share method in the javascript context, in another jslib for example. On Unity 2020, you can call `Module.asmLibraryArg._ShareNSaveWebGL_ShareBlob(shareCallback, blob, fileName, url, title, text)`
See this forum post.

shareCallback here is a callback defined in the javascript side, as all the other parameters. **All the blob versions of the other methods can be used in the javascript context as well.**

One thing worth mentioning is that you can share only text if you want to. Just call the

file parameters with null:

```
ShareNSaveWebGL.Share(shareCallback, null, null, "MyURL", "MyTitle", "MyText");
```

HOW TO CHECK IF YOU CAN SHARE

There are 2 methods available:

```
public static status CanShare(byte[] file, string mimeType, string fileName = null, string url = null, string title = null, string text = null)
```

and

```
public static status CanShare(string blobPropertyPath, string fileName = null, string url = null, string title = null, string text = null)
```

They take the exact same parameters as the share methods, the byte array version and blob version, respectively, and work in the same way. They can return one of the following status

Success,CantShareError, WebShareUnsupportedError, WebShare2UnsupportedError
The blob version can also return the NotABlobError status

HOW TO DOWNLOAD

There are 2 methods to do this, one that takes a byte array and one that takes a blob. Just like for sharing. The parameters also work the same.

```
public static void Save(byte[] file, string mimeType, string fileName = null)
```

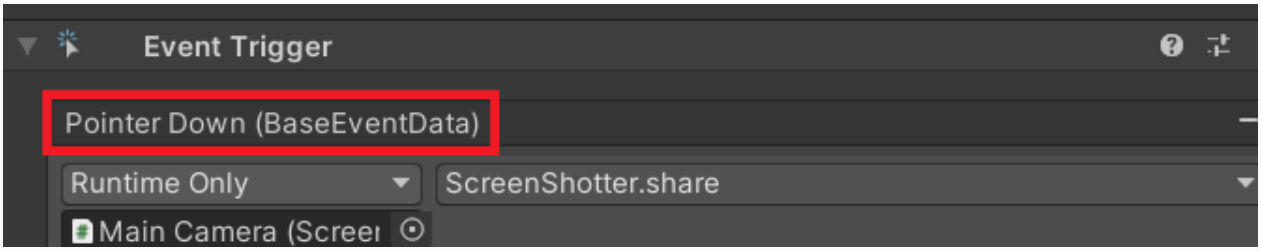
```
public static void Save(string blobPropertyPath, string fileName = null)
```

Please note that there are no callbacks for saving, because the api being used to download files doesn't provide any callbacks. There is a new download api that provides this functionality, but it has no support from most browsers sadly.

Here's how you can call all blob versions in the javascript side, in Unity 2020:
Module.asmLibraryArg._ShareNSaveWebGL_ShareBlob
Module.asmLibraryArg._ShareNSaveWebGL_SaveBlob
Module.asmLibraryArg._ShareNSaveWebGL_CanShareBlob

IMPORTANT NOTES:

1. **TO IGNORE PARAMETERS IN C#, USE THE NULL VALUE. BUT IF YOU ARE CALLING THE METHODS IN JAVASCRIPT, USE THE EMPTY STRING "" TO IGNORE PARAMETERS INSTEAD.**
2. **YOU MUST USE HTTPS. HTTP WON'T WORK!**
3. **ALL SHARE METHODS MUST BE CALLED WITH A POINTERDOWN. FAILING TO DO SO WILL RESULT IN A 'NOTALLOWEDERROR'.**



KNOWN ISSUES

Sharing files doesn't work the same everywhere, unfortunately. Depending on the OS + app combo, you can have different results. Nothing I can do about this, just wait as time goes by more and more apps will behave the way you expect following the web standard. Here's a list of specific issues I know of:

1. If you try to share a file to whatsapp on iOS15, it will only work if you don't share it with any text. If you try to share an image + text, only the text will be shared. This issue doesn't happen on whatsapp if you use Chrome on Android.
2. On some social media, text, and URL don´t appear on the post (Facebook, Instagram, Messenger, Linkedin).
3. It doesn't work on firefox. Check the compatibility table.
4. You can't share files on iOS 14 or lower, only on iOS15 onwards. This is because the Web Share API level 2 only became available on iOS v15, it has nothing to do with my asset.
5. The UI can become unresponsive after you share. Please check the last reply from this post and make sure to have Application.runInBackground set to true.