# WebGL Native File Browser

- ## Intro:

  WebGL Native File Browser a tool for Unity which provides functionality for:
  - Loading any files
  - Saving any files
  - Drag & Drop

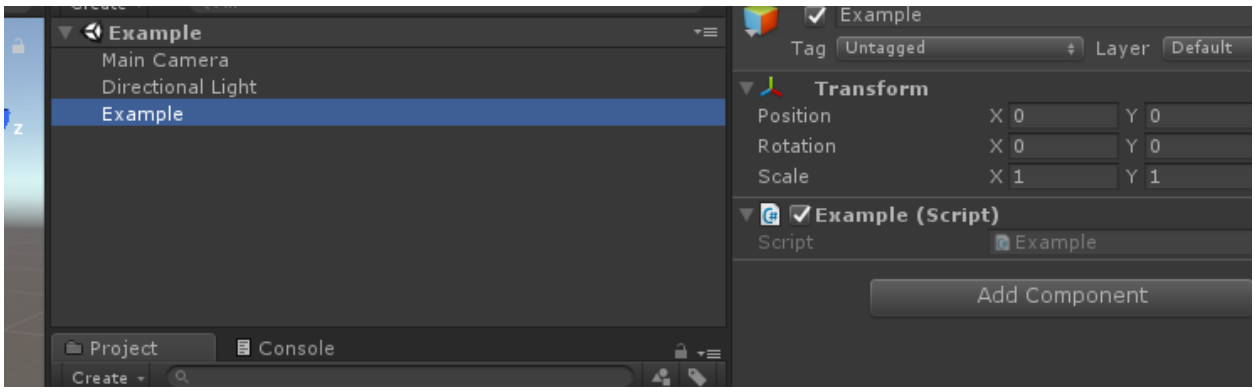
- ## How to use:

  Create you first an app example:

  Create the script with and name it ‘Example’:

```
Example.cs*
Assembly-CSharp                    Example
1   using UnityEngine;
2   using System;
3
    0 references
4   public class Example : MonoBehaviour
5   {
        0 references
6       private void Start()
7       {
8
9       }
10  }
```

  Attach it on Example object that was created in scene:




  Now we could write functional code in example script.

  We need to declare 3 parameters for Buttons, 1 for InputField and 2 for Text.

```
namespace FrostweepGames.Plugins.WebGLFileBrowser.Examples
{
    Скрипт Unity | Ссылок: 0
    public class Example : MonoBehaviour
    {
        public Button openFileDialogButton;

        public Button saveOpenedFileButton;

        public Button cleanupButton;

        public InputField filterOfTypesField;

        public Text fileNameText,
                    fileInfoText;
```

We will use them for API calls and showing result.

Lets make a *Start* function where we will make handlers for *onClick* events and will subscribe on plugins API events.

```
Сообщение Unity | Ссылок: 0
private void Start()
{
    openFileDialogButton.onClick.AddListener(OpenFileDialogButtonOnClickHandler);
    saveOpenedFileButton.onClick.AddListener(SaveOpenedFileButtonOnClickHandler);
    cleanupButton.onClick.AddListener(CleanupButtonOnClickHandler);
    filterOfTypesField.onValueChanged.AddListener(FilterOfTypesFieldOnValueChangedHandler);

    WebGLFileBrowser.FileWasOpenedEvent += FileWasOpenedEventHandler;
    WebGLFileBrowser.FilePopupWasClosedEvent += FilePopupWasClosedEventHandler;
    WebGLFileBrowser.FileOpenFailedEvent += FileOpenFailedEventHandler;
    WebGLFileBrowser.FileWasSavedEvent += FileWasSavedEventHandler;
    WebGLFileBrowser.FileSaveFailedEvent += FileSaveFailedEventHandler;

    // if you want to set custom localization for file browser popup -> use that function:
    // WebGLFileBrowser.SetLocalization(LocalizationKey.DESCRIPTION_TEXT, "Select file for loading:");
}
```

In this screenshot you can see that we subscribed on API events:

1. *FileWasOpenedEvent* - Will fire when file will successfully be loaded
2. *FilePopupWasClosedEvent* - Will fire when native file loading popup was closed
3. *FileOpenFailedEvent* - Will fire when error received during file loading
4. *FileWasSavedEvent* - Will fire when file was successfully saved
5. *FileSaveFailedEvent* - Will fire when error received during file saving

Also, we wrote handlers on UI elements events.

Now lets create events handlers functions:

```
ссылка: 1
private void SaveOpenedFileButtonOnClickHandler()
{
}

ссылка: 1
private void OpenFileDialogButtonOnClickHandler()
{
}

ссылка: 1
private void CleanupButtonOnClickHandler()
{
}

Ссылок: 2
private void FileWasOpenedEventHandler(File file)
{
}

Ссылок: 2
private void FilePopupWasClosedEventHandler()
{
}

Ссылок: 2
private void FileWasSavedEventHandler()
{
}

Ссылок: 2
private void FileSaveFailedEventHandler(string error)
{
}

Ссылок: 2
private void FileOpenFailedEventHandler(string error)
{
}

ссылка: 1
private void FilterOfTypesFieldOnValueChangedHandler(string value)
{
}
```

In these functions we will write logic for handling loaded files, saving file, cleaning loaded data and errors logging.

To save file you have to write that logic:

```
private void SaveOpenedFileButtonOnClickHandler()
{
    //if you want to save custom file use this flow:
    File file = new File()
    {
        fileInfo = new FileInfo()
        {
            fullName = "Myfile.txt"
        },
        data = System.Text.Encoding.UTF8.GetBytes("my text content!")
    };
    WebGLFileBrowser.SaveFile(file);
}
```

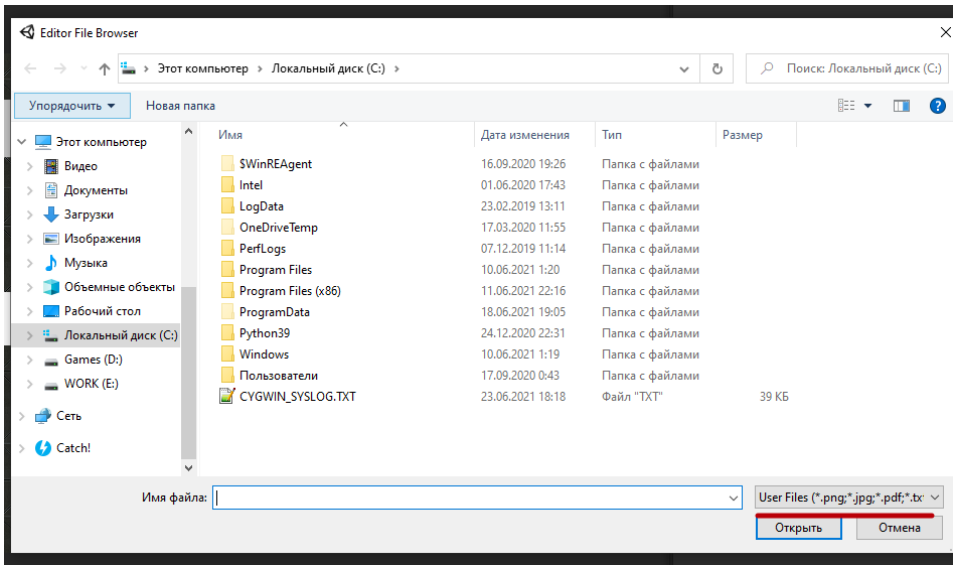
Our API provides ability for saving file as byte array. So you have to convert your file to byte[] before saving.

In this screenshot we used **SaveFile** function which saves object with type **File**. If saving was successful or failed you could handle that in events handlers which we wrote above.

To load file, you have to call **OpenFilePanelWithFilters** API function. Lets add it in button *onClick* handler and add some extensions filter:

```
ссылка: 1
private void OpenFileDialogButtonOnClickHandler()
{
    WebGLFileBrowser.OpenFilePanelWithFilters(".png,.jpg,.pdf,.txt,.json");
}
```

When you will click on button during play mode you will see file browser popup with filter by extensions.




Now you're ready to select file for loading.

When file will be loaded the ***FileWasOpenedEvent*** will be thrown, lets add handling of file in event handler:

```
Ссылок: 2
private void FileWasOpenedEventHandler(File file)
{
    fileNameText.text = file.fileInfo.name;
    fileInfoText.text = $"File name: {file.fileInfo.name}\nFile extension: {file.fileInfo.extension}\nFile size: {file.fileInfo.SizeToString()}";
}
```

In this screenshot you can see that we taking info from loaded file and fill text components by file information.

In this function we could also handle file type and load it as text or image:

```
Ссылок: 2
private void FileWasOpenedEventHandler(File file)
{
    _loadedFile = file;

    fileNameText.text = file.fileInfo.name;
    fileInfoText.text = $"File name: {file.fileInfo.name}\nFile extension: {file.fileInfo.extension}\nFile size: {file.fileInfo.SizeToString()}";

    if (file.IsImage())
    {
        contentImage.sprite = file.ToSprite(); // dont forget to delete unused objects to free memory!

        WebGLFileBrowser.RegisterFileObject(contentImage.sprite); // add sprite with texture to cache list. should be used with  WebGLFileBrowser.FreeMemory() when its no need anymore
    }

    if (file.IsText())
    {
        string content = file.ToStringContent();
        fileInfoText.text += $"\nFile content: {content.Substring(0, Mathf.Min(30, content.Length))}";
    }
}
```

We added additional component named as *contentImage* with type ***Image*** for showing result of loaded image file.

Also you could see that we used special files management API for storing cache(objects):

```
WebGLFileBrowser.RegisterFileObject(contentImage.sprite);
```

This API caches Unity Object and could be cleaned up by using that function

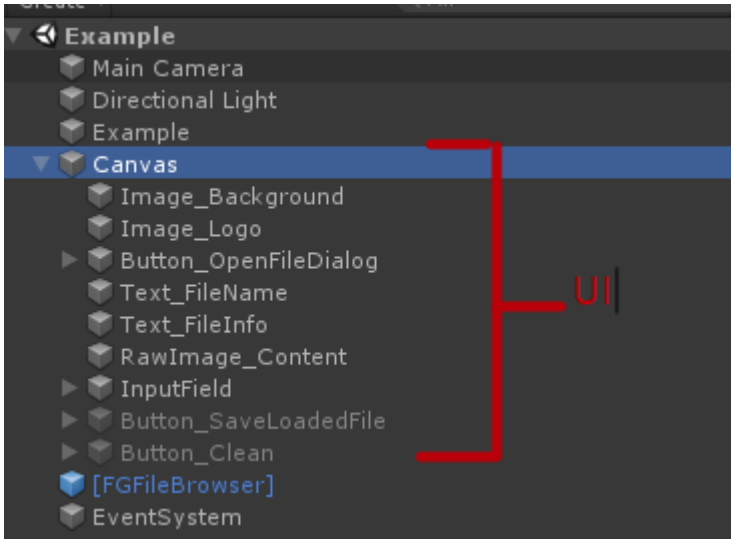
```
WebGLFileBrowser.FreeMemory(); // free used memory and destroy created content
```

It helps to manage dynamic memory usage.

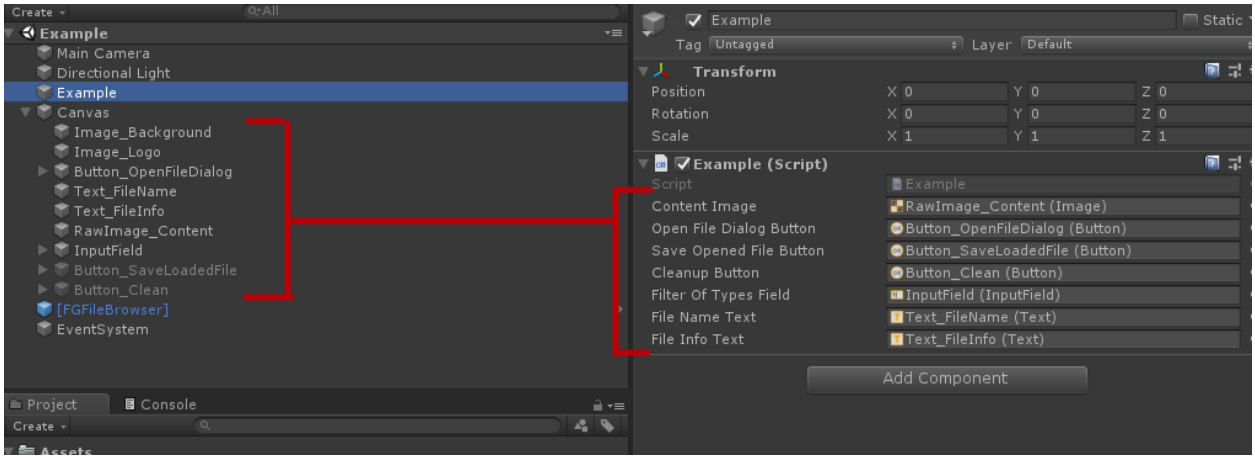
Do not use ***FreeMemory*** frequently as it uses high CPU usage functions such as ***GC.Collect***.

Best way to use it when you will not use large files anymore to free memory or before Scene unload.

Now we ready to create UI elements in scene. Lets create a Canvas and add needed UI elements:




Then connect UI object with Example script variables fields:




And now you're ready to use it.

Full source code of Example script with scene you could find in asset project by path:
***Assets\Plugins\FrostweepGames\WebGLFileBrowser\Examples***

*Thanks for using our products!*

*Best Regards*

*Frostweep Games Team*