DESCRIPTION

I know what you're thinking. There's already the Screen.fullScreen api, and it works on WebGL, even mobile devices.
Well, that's true. However, there are a couple of things that this plugin can do which the official solution can't:

1-You can configure the navigationUI parameter. The official solution has this parameter set to *auto*. Which means that in some browsers, like Chrome on Android, will show the navigation bar on fullscreen. By default, my plugin sets this parameter to *hide*.
2-You can set callbacks to do something when the app enters fullscreen and/or exits fullscreen. You can react to fullscreen changes. This is useful if you want to have a button to toggle fullscreen and change its appearance accordingly, for example. Note that this works even if the user exits fullscreen using the navigation bar or the ESC key instead of a button from your app.
3-You can detect if fullscreen is available on the device. Meaning you can show the toggle button only if the user can switch to fullscreen in the first place. On Safari for iPhone you can't for example.

TL;DR

Fullscreen on roids. If the official solution is enough for you, don't bother. But hey, it's free.

If you like my asset, please consider leaving a review! Constructive criticism is always welcome, and don't hesitate to contact me if you run into any issues!

Any Requirements?

Yes, your browser needs to implement the Fullscreen API.

HOW TO USE

There are only a couple of methods, if you check the example scene, the FullscreenWebGLExample.cs script attached to the canvas gameobject, it all becomes clear. I will explain them here though.

1- requestFullscreen

```
public static void requestFullscreen(Action<status> callback = null,
navigationUI navUI = navigationUI.hide)
```

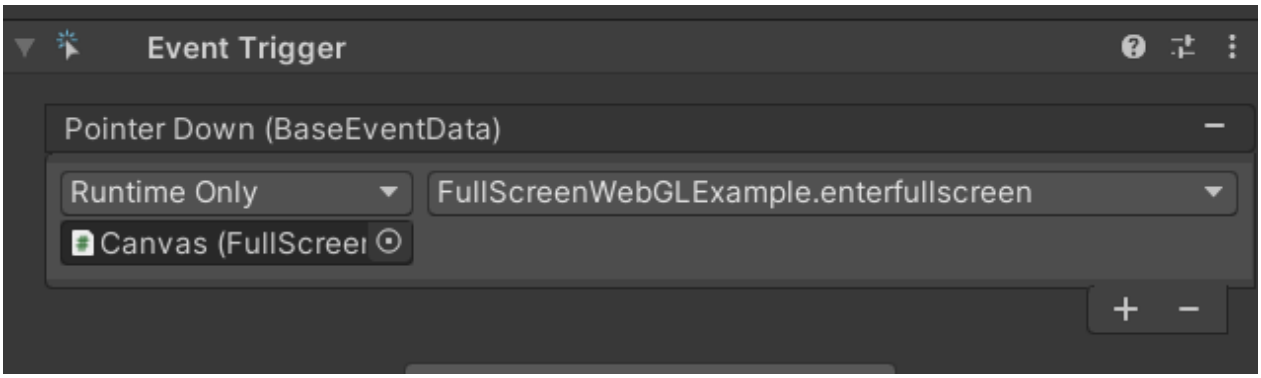
The callback takes a function with a status parameter, which is the following enum

```
public enum status {
  Success = 0,
  Error = 1
 };
```

The function will receive status Success if you enter fullscreen mode, Error if it fails. The second argument is just an enum value corresponding to the navigationUI parameter described here:

```
public enum navigationUI {
   hide = 0,
   show = 1,
   auto = 2
};
```

This method **must** be called with a pointerdown event. On the example scene, the enterfullscreen method calls requestFullscreen, so it's called on a pointerdown:



2- exitFullscreen

```
public static void exitFullscreen(Action<status> callback = null)
```

Exits fullscreen. The callback works the same as for the requestFullscreen.

This method **must** be called with a pointerdown event. Same logic as the requestFullscreen method.

3- isFullscreen

```
public static bool isFullscreen()
```

Returns true if the app is currently fullscreen, false if not

4-isFullscreenSupported

```
public static bool isFullscreenSupported()
```

Returns true if the app can go fullscreen, false if not.

5-subscribeToFullscreenchangedEvent

```
public static void subscribeToFullscreenchangedEvent()
```

If you need to listen to fullscreen changes, that is, if the app becomes fullscreen or exits fullscreen, either by a user action in the app(such as a button), or a user action outside the app(pressing esc or the hardware back button), then you need to call this method.

After calling this method, you can add callbacks that will be called when the fullscreenchange event is called using the public action

```
public static event Action onfullscreenchange;
```

. See this code snippet from the FullScreenWebGLExample.cs script.

```
FullscreenWebGL.subscribeToFullscreenchangedEvent();//I'm interested
in listening to fullscreen changes, so I subscribe to the event.
            FullscreenWebGL.onfullscreenchange += () => {//and then I
add a callback that will run once the user enters or exits fullscreen
                if (FullscreenWebGL.isFullscreen()) {//if it's
fullscreen
                    enterFullscreenBtn.SetActive(false);//deactivate
fullscreen button
                    exitFullscreenBtn.SetActive(true);//activate
```

```
exitfullscreen button
                } else {//otherwise do the opposite
                    enterFullscreenBtn.SetActive(true);
                    exitFullscreenBtn.SetActive(false);
                }
            };
```

6-unsubscribeToFullscreenchangedEvent

```
public static void unsubscribeToFullscreenchangedEvent()
```

If you don’t need to listen to the fullscreenchange event anymore, you can unsubscribe from it. Don’t forget to also remove the callbacks from the onfullscreenchange event that you added before.