ReactUI

What is it?

ReactUI allows you to design and manage user interfaces with a state-of-art, modern and powerful framework: React.
Not only: it allows you to use all tools and plugins you use in web development to implement a game's user interface.

Speaking about tooling, it comes with a pre-set environment composed by:

- webpack as bundling/building system
- Less as CSS super-set
- Babel as Javascript transpiler, so your Javascript code can run on old browsers too
- React as Virtual DOM rendering engine

It comes with a powerful and extensible **event system**, that allows you to easily raise and manage events from the game or the UI.
This is possible thanks to a special **Javascript/C# bridge**, integrated in the package.

Last, but not least, it comes with a little game sample that shows you simple interactions from game => UI or from UI => game.

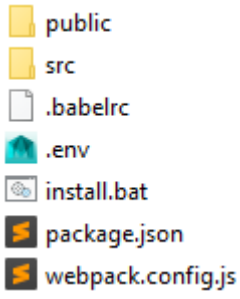
Prerequisites

You **need** Node.js and NPM installed on your computer before proceeding. Install them by downloading the binary on the official website and following their instructions: https://nodejs.org/

Installation

The first thing you have to do after importing the package is creating a folder **outside** of Unity project for your UI code. For the sake of this example, name it: "***MyReactUI***".

Extract the content of "***Assets/MHLab/ReactUI/React/React.zip***" in this "***MyReactUI***" folder. You should have something like this:



If you are on Windows, for your convenience, I added "***install.bat***" file: just run it.

If you are not on Windows or that file will not work for you for some reason, just open the command line in our "***MyReactUI***" folder and run "***npm install***" command.

Wait for this command to complete, it is installing required dependencies.

Development

You're ready to test the sample scene and to develop your own UI! Let's test it!

From the command line pointing to "***MyReactUI***" folder, run the command "***npm run start***".
After some building time, it will open your browser and will show you the example scene!

*N.B: if the browser does not automatically open, just open it manually and navigate to "**localhost:3000**" address.*

The previous command started a webpack dev-server. It has an **hot-reload** feature, so if you try to modify scripts/styles and you save, it will automatically reload them and show changes **realtime** in your browser.

If you like, it is time to hack: change the sample as you prefer, explore what you can do and test the workflow.
From here you can start building your own UI.

Production

When you are finally satisfied of your changes, you can build the whole thing for production. To do this, just run the command "***npm run build***".

This command generates an optimized build in a folder named "***dist***". The content of this folder is your complete game, the one you will deploy on your own host.

To test if it works properly, just open this "***dist***" folder. You will find "***index.html***": open it in your browser and play!

N.B: Chrome will not allow you to directly open this file, it requires a web server to serve it. Instead, you can directly open it with Firefox or Edge.

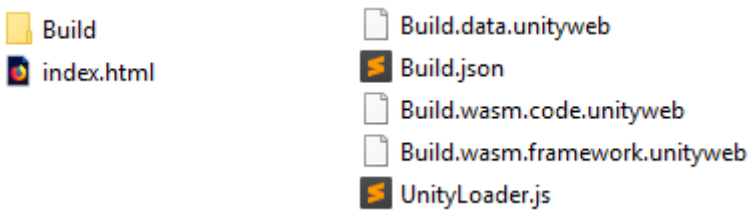
What now?

You built the sample, you tested it. Probably now you want to start with your own game and your own UI.

Developing your own UI is simple: just follow the app skeleton and hack it accordingly to your tastes.

Developing your own game is simple too: do it in Unity as usually. The only thing you have to take into consideration is the name of your build when you compile your game as WebGL application.

After you compiled your game, you should have something like this:



I named mine "***Build***" for convenience, you can name it as you prefer.

Now ignore that generated "***index.html***" and copy the "***Build***" folder (or whatever you named it) to your "***MyReactUI/public***" folder.

Open your "***MyReactUI/public/index.html***" file in your favorite text editor. You should see something similar:

```
47      <script>
48          var gameInstance = UnityLoader.instantiate("unity_player", "Build/Build.json");
49      </script>
```

If you named your compiled game differently than "***Build***", then just change that "***Build/Build.json***" accordingly.

API

You can learn about the whole API by reading the code: it is not so large. By the way, I want to simplify your life so I will put a little resume here!

Sending an event from the UI to the Game

If you have to send an event from your UI (Javascript) to the game (C#), you can call the following function from anywhere in your code:

```
window.FireUIEvent("your_named_event", {});
```

the first parameter is the name of the event (be sure it is unique), the second parameter is a simple Javascript's object that you can fill with the data you prefer.

Sending an event from the Game to the UI

If you have to send an event from your game (C#) to your UI (Javascript), you can call the following method:

```
var ev = new UIEvent<MyEventPayload>( eventName: "your_name_event", new MyEventPayload()
{
    value = "something"
});
UIEventDispatcher.Instance.Dispatch(ev);
```

where MyEventPayload is defined like:

```
private class MyEventPayload : UIEventPayload
{
    public string value;
}
```

Listening an event on the UI side

To allow one of your React components to listen for an event, you have to import the Dispatcher class:

```
import Dispatcher from './api/Dispatcher';
```

(it is in "MyReactUI/src/api/Dispatcher.js")

then you can simply register a listener:

```
Dispatcher.on("your_event_name", function (payload) {
    // Do something with the event's payload!
    console.log(payload);
});
```

Listening and event on the Game side

To allow your game to listen for an incoming event, you just have to call:

```
UIEventDispatcher.Instance.RegisterHandler<MyEventPayload>( eventName: "your_event_name", MyEventCallback);
```

where MyEventPayload is the same as before and MyEventCallback is like:

```
private void MyEventCallback(string eventName, UIEventPayload e)
{
    e = (MyEventPayload)e;
    // Do something with your event's payload!
}
```

Now you're ready to go! Have fun! :)