HTML To MSFS Gauges Tutorial

By LivToAir

Pages:

Additional Info:

Recommended additions for HTML gauges creation:

- **VS code:** recommended for any script editing.

- **MSFS 2020 WebUI DevKit:** https://github.com/dga711/msfs-webui-devkit
  I will call it **"debug.js"** and it will show you the console of your gauge in sim

- **MSFS SDK:**
  C:\MSFS SDK\Documentation\index.html
  or
  https://docs.flightsimulator.com/html/Introduction/Introduction.htm

- **MSFS SDK - STATUS OF SIMULATION VARIABLES:**
  C:\MSFS SDK\Documentation\html\Programming_Tools\SimConnect\Status\Status_Of_Simulation_Variables.htm
  or
  https://docs.flightsimulator.com/html/Programming_Tools/SimVars/Simulation_Variables.htm

  there you could find all the variables your gauge can read/write.

Note: you will need to know web development or at list basic web development knowledge to understand this tutorial.

MSFS and HTML introduction

MSFS gauges creation has 2 options,

- HTML, CSS, JS (act like web development)
- WASM, (don’t really know what it is XD)

When using HTML, it is just like web development mostly.

MSFS HTML Gauges is pretty much close to Unity Game engine for example.
Its using a Start Function and an Update function.

Note: I still don’t know everything about MSFS HTML so there will be stuff that I probably wont know, but I know enough to make this tutorial for all the community that was waiting for something like this.
Just like me in the past :)

Files Set Up

Firstly you want to go to your “PackageDefinitions” folder and open the XML script.

In the script you will need to add another “AssetGroup”,
(see in the tutorial folder for an example, PackageDefinitions/HTM_Turotial.xml)

you will need to add

```
<AssetGroup Name="html_ui">
    <Type>Copy</Type>
    <Flags>
        <FSXCompatibility>false</FSXCompatibility>
    </Flags>
    <AssetDir>PackageSources\html_ui\</AssetDir>
    <OutputDir>html_ui\</OutputDir>
</AssetGroup>
```

After you done this. You can create the “html_ui” folder in the PackageSources folder of your project:

In the main directory of the PackageSources folder create a new folder named “html_ui”.

After you added the AssetGroup and the html_ui folder, its now time to start to implement the html gauge!

Setting Up the html_ui folder

In html_ui create those folders:

Pages\VCockpit\Instruments

After those folders create a folder and name it your project name.
We do this to separate your html_ui folder from the default html_ui folders that MSFS takes the gauges for their default aircrafts.

So the folders will look like that in your project files:

PackageSources\html_ui\Pages\VCockpit\Instruments\YourProjectName

Now in the that folder we can put gauge code!

Setting Up the html, css, js scripts

In your project folder in the html_ui folder (Pages\VCockpit\Instruments\YourProjectName)

You can now create a folder and name it the gauge name

for this tutorial we will call the gauge “Gauge_Tamplate”

(html_ui\Pages\VCockpit\Instruments\Gauge_Tamplate)

In that folder we can now create our html css and js scripts!

In the tutorial folder go to:

PackageSources\html_ui\Pages\VCockpit\Instruments\YourProjectName\Gauge_Tamplate

And copy the html css and js files to your html_ui project folder.

After that you can change all the names (“Gauge_Tamplate”) to your gauge name.

Now you can open the 3 scripts in your file editor (VS Code Recommended)

Setting Up the html css js scripts

In the HTML:

Change the .css direction to your .css file name:

```
<link rel="stylesheet" type="text/css" href="Gauge_Tamplate.css" />
```

Change the script id to your gauge name:

```
<script type="text/html" id="Gauge_Tamplate">
```

Make sure that you load debug.js script in the html:

```
<script type="text/javascript" src="/JS/debug.js"></script>
```

Note: that line should be there already if you copied the scripts from the tutorial folder

Change the .js direction to your .js file name:

```
<script type="text/javascript" src="Gauge_Tamplate.js"></script>
```

Note: make sure to put your html code inside of the “Mainframe” DIV

```
<script type="text/html" id="Gauge_Tamplate">
    <div id="Mainframe">
        //HTML code here
        <div id="ExampleDiv"></div>
    </div>
</script>
```

In the CSS:

You don’t need to change anything in the css to make the script connect to your gauge.

In the JS:

Change the class name to your gauge name:

```
class Gauge_Tamplate extends BaseInstrument
```

Change templateID to your gauge name:

```
get templateID() { return "Gauge_Tamplate"; }
```

Change the is Interactive to fit your gauge:
if **true**, your gauge would be clickable and you could add buttons to it
If **false**, your gauge wont be clickable and you cant add buttons to it

```
get isInteractive() { return true; }
```

Change the “registration” of the gauge:

- The first text in the ‘ ’ should **not** be the same as your gauge name! name it something different like except of “Gauge_Tamplate” -> “gauge-tamplate”
- The second text should be like your class name, your gauge name.

```
registerInstrument('gauge-tamplate', Gauge_Tamplate);
```

Setting Up PANEL.CFG

In the PANEL.CFG:

```
[VCockpit01]
size_mm                 = 1920,1920
pixel_size              = 1920,1920
texture                 = $Gauge_Tamplate
htmlgauge00=YourProjectName/Gauge_Tamplate/Gauge_Tamplate.html,
        0, 0,1920,1920
```

MSFS and JS: Update & Start Function

MSFS JS is like Unity engine for example, you have an update function and a start function.

The start function is the “connectedCallback” function:
every time your gauge will load into the sim everything in this function will run once

```
connectedCallback() {
    super.connectedCallback();
    //Global Vars & On Start

}
```

And the Update function just called Update:

Update function will run in loops as long as your aircraft is loaded

```
Update() {

}
```

Note: there is “browserUpdate” and “msfsUpdate” functions.
Both of them I cant say clearly what they do and what are the differents between those and the Update function.

I'm using only the Update function and everything works perfectly fine.

MSFS and JS: Creating Vars

Normal JS vars like “var”, “const”, “let” etc… will work as long as you use the var only on the same function that you created them in.

Because of how MSFS is creating functions, when you cant write code outside a function. So you cant make a global var with the default JS vars.

So how to create a global var in MSFS JS?

Global vars are just -> `this.varname = value`

To declare a global var you just need to add “this.” before the var name.

Also! You don’t need to create a var before making it global with “this.”, if you wrote:

```
this.example_var = "this is an example";
```

It will create a string var and will make it global.

So now each time you will write for example:

```
console.log(this.example_var);
```

You will get “this is an example” in the console.

you can also put numbers in the same way: `this.num_var = 1;`

Note: you can still use: var, const, let etc… **but** only if you not using them outside of the function that you created them.

MSFS and JS: Creating Vars Example

Example for using vars:

```
connectedCallback() {
    super.connectedCallback();
    //Global Vars & On Start ->
    this.string_var = "this is a string";
    this.number_var = 10;

    console.log("String Var Value is: " + this.string_var);
    console.log("Number Var Value is: " + this.number_var);
}
```

This “connectedCallback” is the on start function.

And in this function I created 2 vars. A string one and a number one. And then I console.log them once.

Note: when creating a var that you want to be change in the future by the code.
Create it in the start function and then play with it in the update.

Don’t create vars in the update functions! You wont be able to change them because in each frame they will go back to their default value.

MSFS and JS: Creating Functions

In normal JS you will just do this:

```
function name(params) {

}
```

But! In MSFS you just need to do this:

```
 ThisIsAFunction(params){

 }
```

And to call a function add “this.” like with calling vars:

```
this.ThisIsAFunction();
```

MSFS and JS: Reading and Writing sim vars

To read a SimVar Value we will use:

```
SimVar.GetSimVarValue("L:LocalVar", "number");
```

in the (), first value is the var, second value is the unit.

To Write a SimVar Value we will use:

```
SimVar.SetSimVarValue("L:LocalVar", "number", 0);
```

The “0” at the end is the value to set of the SimVar.

The SDK has a page especially for all the SimVars that can be readable,

Recommended to use the SDK that is downloaded on your pc: C:\MSFS SDK\Documentation\html\Programming_Tools\SimConnect\Status\Status_Of_Simulation_Variables.htm

Or if you want the web SDK also have that page: https://docs.flightsimulator.com/html/Programming_Tools/SimVars/Simulation_Variables.htm

JS with HTML:

In normal JS you edit an html object with:

```
document.getElementById('name');
```

Actually in MSFS JS the “document” commend dose work, but not recommended!

In MSFS JS you will use “this.getChildById”:

```
this.getChildById("name");
```

So to change the style of an html object in MSFS JS:

```
this.getChildById("name").style.
```

Some examples:

```
// sets the visibility of the object, hidden / visible
this.getChildById("name").style.visibility = "hidden";
// sets the height of the object, number + unit in " ".
this.getChildById("name").style.height = 10 + "px";
// sets the top of the object, number + unit in " ".
this.getChildById("name").style.top = 50 + "%";
// sets the opacity of the object, 0 to 1.
this.getChildById("name").style.opacity = 0.5;
//examples from my XA42:

//makes a "progress bar" based on the brightness value.
this.getChildById("BrightnessProgKnob").style.background = "linear-
gradient(90deg, transparent " + this.BrightnessLevel + "% , black 0)"
//sets an background Image based on URL that I took from an API.
this.getChildById("news1Logo").style.backgroundImage = "url("+
data.news1.logo+")";
```

In MSFS JS to change the text of an object:

```
this.getChildById("name").textContent = "text";
```

To create a button:

In MSFS JS to create a button we still use the Event Listener, but in MSFS JS it looks a bit different:

```
connectedCallback() {
    this.button = this.getChildById("buttonObj");
    this.button.addEventListener(
        "mousedown",
        this.buttonOnClick.bind(this)
    );
}

buttonOnClick(){
    //everything in this function will run once when
    button is pressed.
}
```

You can also change the “mousedown” to any Mouse Events that you want, for example:“mouseup”,“click”, “dblclick” etc…

see: https://www.javascripttutorial.net/javascript-dom/javascript-mouse-events/ for more info about Mouse Events

HTML To MSFS Gauges Tutorial Summary

I think I wrote here pretty much everything I know.

There will be probably things that I forgot to mention.
I will keep working on this tutorial and add what I missed
(if I missed any) and add new things that I learned.

For any help / questions feel free to DM me on Discord:
YANIV#1257

I hope this tutorial helped you to get started in the world HTML gauges to MSFS!!!
And if not feel free to say me why.