GEMscript Advantages

Simplify Through GEMscript

What are the advantages of GEMscript and just how does GEMscript simplify the creation of user interfaces.

Informational Paper
Published: August 2019

Amulet

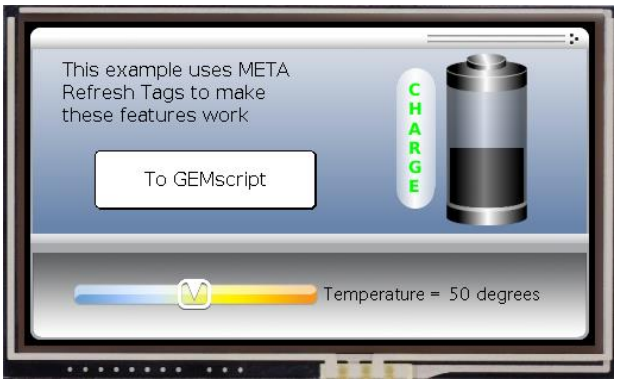
Simplify Through GEMscript

With the introduction of GEMscript, included with GEMstudio Pro, Amulet Technologies has taken the creation of HMI (Human Machine Interface) to a more powerful level. The GEMscript scripting language allows for an unprecedented level of control over the creation of visually stunning and intuitive controls for touch displays.

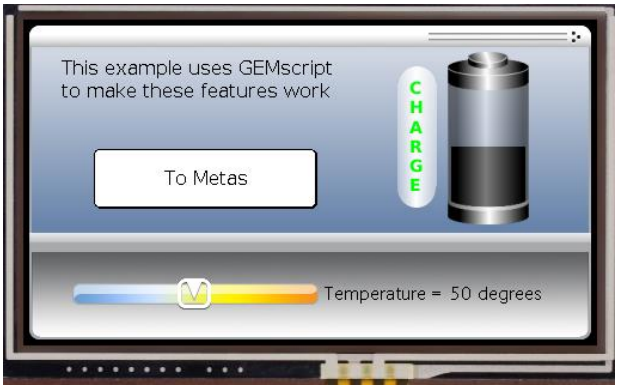
Not only does GEMscript add new functionality to the creation of a user interface, it also brings simplicity. This paper compares and contrasts the creation of a user interface with and without the use of GEMscript. The ease of creating and understanding GEMscript will become more and more evident as the comparisons are made.

This entire example is included with the demos shipped with GEMstudio Pro.

A simple two page project was developed. Each page is functionally identical, with one page created without the use of GEMscript and the other with GEMScript.



Page 1 (Meta page) is created without the use of GEMscript. In the absence of GEMscript, META Refresh Tags were utilized to create the features.

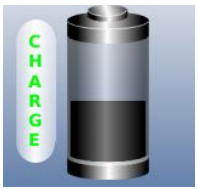


Page 2 (GEMscript page) is created with GEMscript. The single META statement which remains is used for a timer based function.

There are basically 2 elements on the page:

a. A slider that changes the displayed temperature, with a color change depending on temperature.
b. A bar graph which changes at the push of a button, also with color change depending on height of the graph.

The first most obvious difference between the use of METAs and use of GEMscript, is the amount of code that needs to be written. Let's start with the battery object and examine the code behind it's function.



Here's the code to implement the battery level changing with color change, initiated when the "CHARGE" button is pushed:

Without GEMscript

```
<meta http-equiv="refresh" content="0,0.01;
URL=Amulet:InternalRAM.byte(30).setValue(0)">

<meta http-equiv="refresh" content="0;
URL=Amulet:document.battery.setx(352),
    Amulet:document.battery.reappear();
name=batteryMeta">

<meta http-equiv="refresh" content="0;
if=Amulet:internalRAM.byte(30).value();
lt=255;
then=Amulet:document.ChargeMeta.forceUpdate();
else=Amulet:document.CheckFull.setUpdateRate(0);
```

Using GEMscript

```
public chargeUp()
{
        new int charge = internalRAM.byte(0)
        if (charge < 0x7F)
        {
                InternalRAM.color(0).setRed(0xFF)
                InternalRAM.color(0).setGreen(2 * charge)
        }
        else
        {
                InternalRAM.color(0).setRed(0x1FE - (2 * charge))
                InternalRAM.color(0).setGreen(0xFF)
```

```
name=CheckFull">

<meta http-equiv="refresh" content="0;
URL=Amulet:InternalRAM.byte(30).add(5),
      Amulet:document.battery.disappear(),
      Amulet:document.BarGraph1.forceUpdate(),
      Amulet:document.batteryMeta.forceUpdate(),
      Amulet:document.Meta0.ForceUpdate();
name=ChargeMeta">

<meta http-equiv="refresh" content="0;
URL=Amulet:InternalRAM.byte(30).copyToRAMByte(31),
      Amulet:InternalRAM.byte(30).copyToRAMByte(32),
      Amulet:InternalRAM.byte(33).SetValue(0xFF),
      Amulet:Document.Meta1.forceUpdate();
name=Meta0">

<meta http-equiv="refresh" content="0;
if=Amulet:InternalRAM.byte(30).value();
lt=0x80;
then=Amulet:InternalRAM.byte(31).mul(2),
       Amulet:InternalRAM.byte(32).setValue(0xFF),
       Amulet:Document.Meta2.forceUpdate();
gt=0x7F;
then=Amulet:InternalRAM.byte(32).mul(2),
       Amulet:InternalRAM.byte(31).setValue(0xFF),
       Amulet:Document.Meta3.forceUpdate();
```

```
            }
            document.battery.disappear()
            document.battery.setX(352)
            document.BarGraph1 = charge
            document.battery.reappear()
}
</script>

<meta http-equiv="refresh" content="0;
if=Amulet:internalRAM.byte(0).value();
lt=255;
then=GEMscript.chargeUp(),Amulet:InternalRAM.byte(0).add(5);
else=Amulet:document.CheckFull.setUpdateRate(0);
name=CheckFull">
```

```
name=Meta1">


<meta http-equiv="refresh" content="0;

URL=Amulet:internalRAM.color(10).setGreen(InternalRAM.byte(31)
),

      Amulet:internalRAM.color(10).setRed(InternalRAM.byte(32));

name=Meta2">


<meta http-equiv="refresh" content="0;

URL=Amulet:InternalRAM.byte(33).sub(InternalRAM.byte(32)),

      Amulet:document.Meta4.forceUpdate;

name=Meta3">


<meta http-equiv="refresh" content="0;

URL=Amulet:internalRAM.color(10).setGreen(InternalRAM.byte(31)
),

     Amulet:internalRAM.color(10).setRed(InternalRAM.byte(33));

name=Meta4">
```

Looking closely at the GEMscript, it's evident that the entire function for the battery is written with a single IF-THEN-ELSE statement. The sheer reduction in the number of lines of code and the simplicity of the code allows easy maintainability.

Examine the href property for the "CHARGE" button .

Without GEMscript

```
Amulet:document.CheckFull.setUpdateRate(0.07);

Amulet:internalRAM.byte(30).setValue(0),

Amulet:document.BarGraph1.forceUpdate(),

Amulet:document.batteryMeta.forceUpdate()
```

Using GEMscript

```
Amulet:document.CheckFull.setUpdateRate(0.07),

Amulet:internalRAM.byte(0).setValue(0)
```

Examining closely, the first two lines of href are the same. First line calls the CheckFull function at a rate of 7ms. The function CheckFull looks like this:

```
<meta http-equiv="refresh" content="0;

if=Amulet:internalRAM.byte(30).value();

lt=255;

then=Amulet:document.ChargeMeta.forceUpdate();

else=Amulet:document.CheckFull.setUpdateRate(0);
```

```
<meta http-equiv="refresh" content="0;

if=Amulet:internalRAM.byte(0).value();

lt=255;

then=GEMscript.chargeUp(),Amulet:InternalRAM.byte(0).add(5);

else=Amulet:document.CheckFull.setUpdateRate(0);
```

This META Refresh Object command is the same, but this is where the similarity ends. Using GEMscript, the META command is used only once. Without GEMscript, the entire project needed 13 META statements.

Readability and familiarity of the scripting code are clear advantages when using GEMscript. This reduces the number of bugs and makes the debugging process easier.

Let's look at the temperature slider that's been implemented.

The numeric field changes with the slider. The font color of the numeric field changes colors. This color value is stored in the following internal RAM address: Amulet:internalRAM.color(1).

Using GEMscript, this is the entire code for this slider operation is:

```
new int Temperature = document.CustomSlider2
        if (Temperature < 0x7F)
        {
                InternalRAM.color(1).setRed(2*Temperature)
                InternalRAM.color(1).setGreen(2*Temperature)
                InternalRAM.color(1).setBlue(0xFF)
        }
        else
        {
                InternalRAM.color(1).setRed(0xFF)
                InternalRAM.color(1).setGreen(0x1FE - (2* Temperature))
                InternalRAM.color(1).setBlue(0x1FE - (2* Temperature))
        }
        document.CustomSliderField2 = Temperature
```

Anybody with some reasonable programming experience can upon first glance understand the GEMscript code.

```
new int Temperature = document.CustomSlider2
```

Set Temperature to the value of CustomSlider

```
if (Temperature < 0x7F)
{
        InternalRAM.color(1).setRed(2*Temperature)
        InternalRAM.color(1).setGreen(2*Temperature)
        InternalRAM.color(1).setBlue(0xFF)
}
else
{
        InternalRAM.color(1).setRed(0xFF)
        InternalRAM.color(1).setGreen(0x1FE - (2* Temperature))
        InternalRAM.color(1).setBlue(0x1FE - (2* Temperature))
}
```

If Temperature is less than 7F then set RGB value of InternalRAM.color(1) with various alpha values.

Else set InternalRAM.color(1) with a different alpha values for RGB

```
document.CustomSliderField2 = Temperature
```

Update the numeric field

Without the ability to use GEMscript, this same function would be written with the following META statements:

```
<meta http-equiv="refresh" content="0;
URL=Amulet:InternalRAM.byte(12).copyToRAMByte(13),
        Amulet:InternalRAM.byte(14).SetValue(0xFF),
        Amulet:Document.TempMeta2.forceUpdate();
name=TempMeta">

<meta http-equiv="refresh" content="0;
URL=Amulet:InternalRAM.byte(13).mul(2),
        Amulet:Document.TempMeta3.forceUpdate();
name=TempMeta2">

<meta http-equiv="refresh" content="0;
URL=Amulet:InternalRAM.byte(14).sub(InternalRAM.byte(13)),
        Amulet:Document.TempMeta4.forceUpdate();
name=TempMeta3">

<meta http-equiv="refresh" content="0;
if=Amulet:InternalRAM.byte(12).value();
lt=0x80;
then=Amulet:InternalRAM.color(1).setRed(InternalRAM.byte(13)),
        Amulet:InternalRAM.color(1).setGreen(InternalRAM.byte(13)),
        Amulet:InternalRAM.color(1).setBlue(0xFF),
        Amulet:document.TempMeta5.forceUpdate();
```

```
gt=0x7F;
then=Amulet:InternalRAM.color(1).setRed(0xFF),
        Amulet:InternalRAM.color(1).setGreen(InternalRAM.byte(14)),
        Amulet:InternalRAM.color(1).setBlue(InternalRAM.byte(14)),
        Amulet:document.TempMeta5.forceUpdate();
name=TempMeta4">

<meta http-equiv="refresh" content="0;
URL=Amulet:document.CustomSliderField2.forceUpdate();
name=TempMeta5">
```

Trying to understand the structure and meaning of all the META statements is a daunting task without a good understanding of the function of METAs. Also the META statements don't need to adhere to any logical order. This makes the readability and understanding of the code extremely difficult.

This paper has shown that GEMscript is not an absolute necessity when designing interactive and highly intuitive user interfaces. However with the power of GEMscript, this task becomes much simpler, while reducing the steep learning curve which comes with tackling any new programming language.