Course on Virtual Reality & Serious Games

Session 8

Terminate java intro
Paper reviews

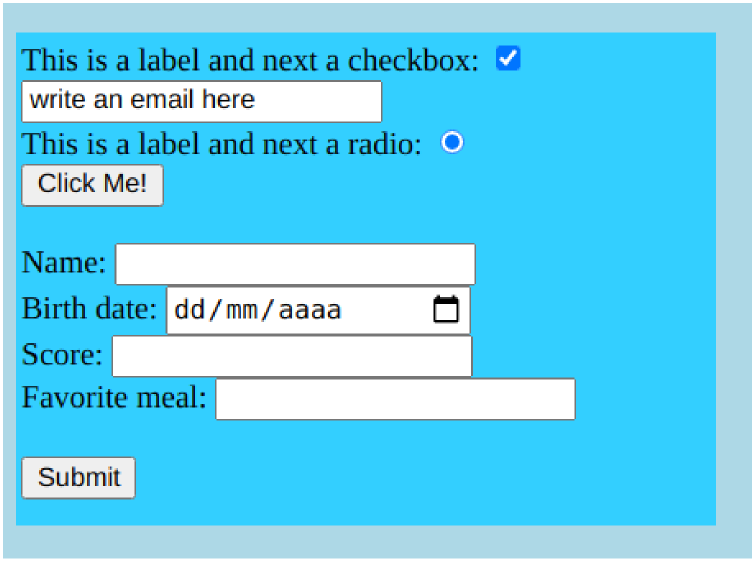
Dani Tost

Input values

In html, to input values, you can use the tag `<input>` that you use together with `<label>`. You can also use the tag `<button>` and the tag `<form>`.

Try to add a new cell with the following contents to your ex3.html example:

```
<div class="cell" style="background-color: #33cfff">
  <label id="input1"> This is a label and next a checkbox:</label>
  <input type="checkbox"> <br>
  <input name="input2" type="email" value= "write an email here"> <br>
  <label id="input3"> This is a label and next a radio:</label>
  <input type="radio" value="B" name="choice" checked> <br>
  <button type="button">Click Me!</button>  <br> <br>
  <form name="myForm">
    Name: <input type="text" name="fname"> <br>
    Birth date: <input type="date" name="fage"> <br>
    Score: <input type="number" name="fnumber"> <br>
    Favorite meal: <input type="text" name="fmeal"> <br><br>
    <input type="submit" value="Submit">
   </form>
</div>
```


This is a label and next a checkbox: ✓
write an email here
This is a label and next a radio: ●

https://www.w3schools.com/js/js_input_examples.asp

Input values and JS

A click on a button can be associated to a JS script. In the example before, modify the button “Click Me”:

```
<button type="button" onclick="changeSomething()">Click Me!</button>  <br> <br>
```

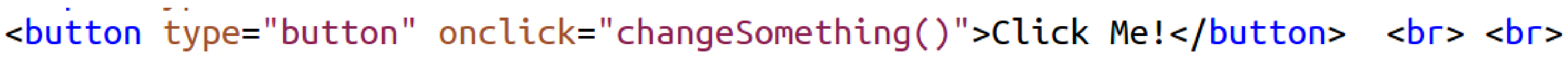
And add a JS script with the function `changeSomething()` after `</body>` and before `</html>`:

```
…..
</body>
 <script>
   function changeSomething(){
      document.getElementById("input1").innerHTML = “hola”;
    }
  </script>
</html>
```

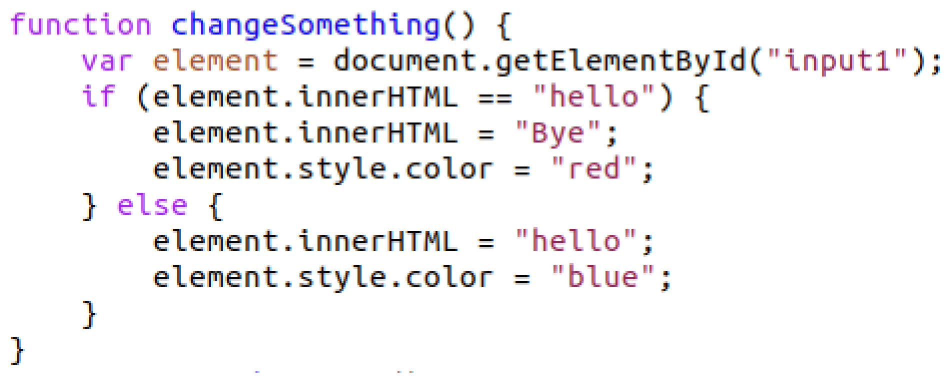
Observe the changes. Try to modify the code so that the new value will be alternatively “hola” or “adeu”.

Try to modify the color of the text with:
`document.getElementById("input1").style.color = “blue”`

Input values and JS



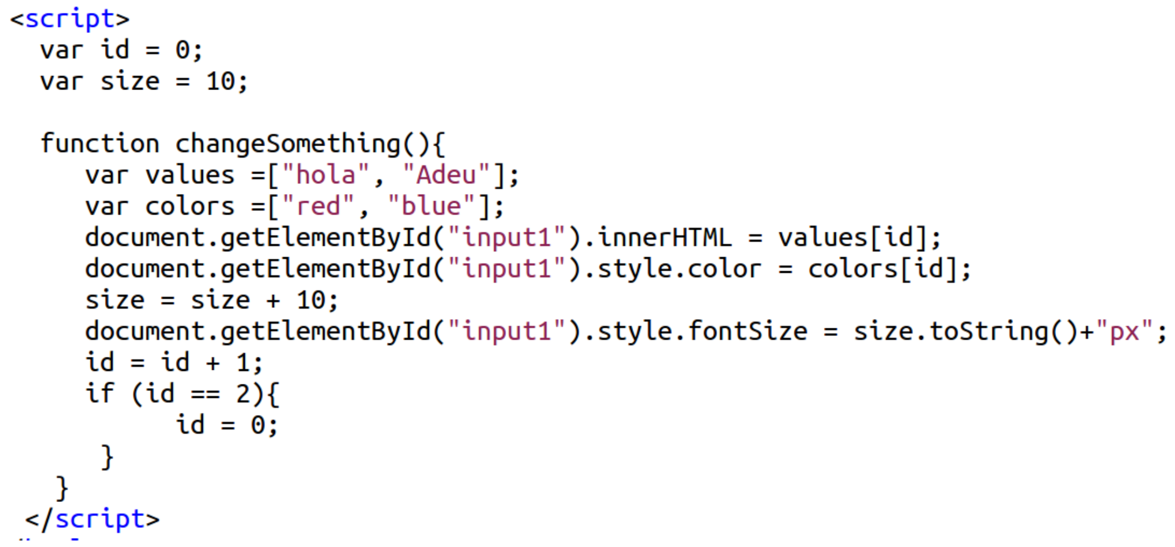
```
<button type="button" onclick="changeSomething()">Click Me!</button>  <br> <br>
```



```
function changeSomething() {
    var element = document.getElementById("input1");
    if (element.innerHTML == "hello") {
        element.innerHTML = "Bye";
        element.style.color = "red";
    } else {
        element.innerHTML = "hello";
        element.style.color = "blue";
    }
}
```

Two possible solutions as examples of use of JS

In the second we also increase the size of the font.



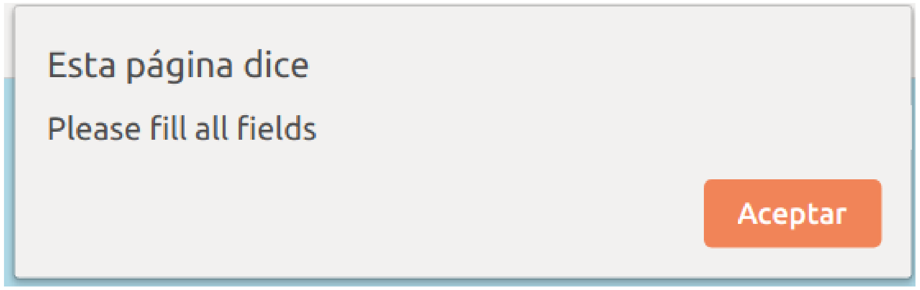
```
<script>
  var id = 0;
  var size = 10;

  function changeSomething(){
     var values =["hola", "Adeu"];
     var colors =["red", "blue"];
     document.getElementById("input1").innerHTML = values[id];
     document.getElementById("input1").style.color = colors[id];
     size = size + 10;
     document.getElementById("input1").style.fontSize = size.toString()+"px";
     id = id + 1;
     if (id == 2){
           id = 0;
      }
   }
 </script>
```

Input values and JS

Try to add a function that validates if all the form fields have been validated:

```
<script>
    function validateForm() {
      var x = document.forms["myForm"]["fname"].value;
      var y = document.forms["myForm"]["fage"].value;
      var z = document.forms["myForm"]["fnumber"].value;
      var u = document.forms["myForm"]["fmeal"].value;
      if (x == "" | y == "" | z == "" | u == "") {
         alert("Please fill all fields");
         return false;
      }
    }
  </script>
```


Esta página dice

Please fill all fields

Or better:

```
function validateForm() {
    var elements = document.forms[“myForm”].elements;
       for (x in elements)  {
              if (elements[x].value == “”) {
       alert (“Please fill all fields”);
       return ;
       }
      }
}
```

Input values and JS

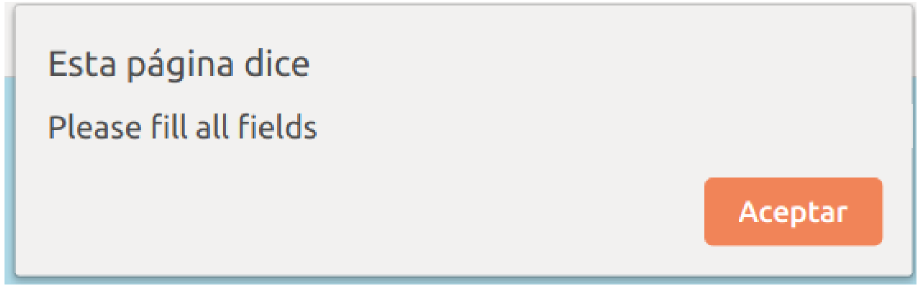
Try to add a function that validates if all the form fields have been validated:

```
<form name="myForm"  onsubmit="return validateForm()  >

    …

</form>


 <script>
     function validateForm() {
       var x = document.forms["myForm"]["fname"].value;
       var y = document.forms["myForm"]["fage"].value;
       var z = document.forms["myForm"]["fnumber"].value;
       var u = document.forms["myForm"]["fmeal"].value;
       if (x == "" | y == "" | z == "" | u == "") {
          alert("Please fill all fields");
          return false;
       }
     }
   </script>
```


Esta página dice

Please fill all fields

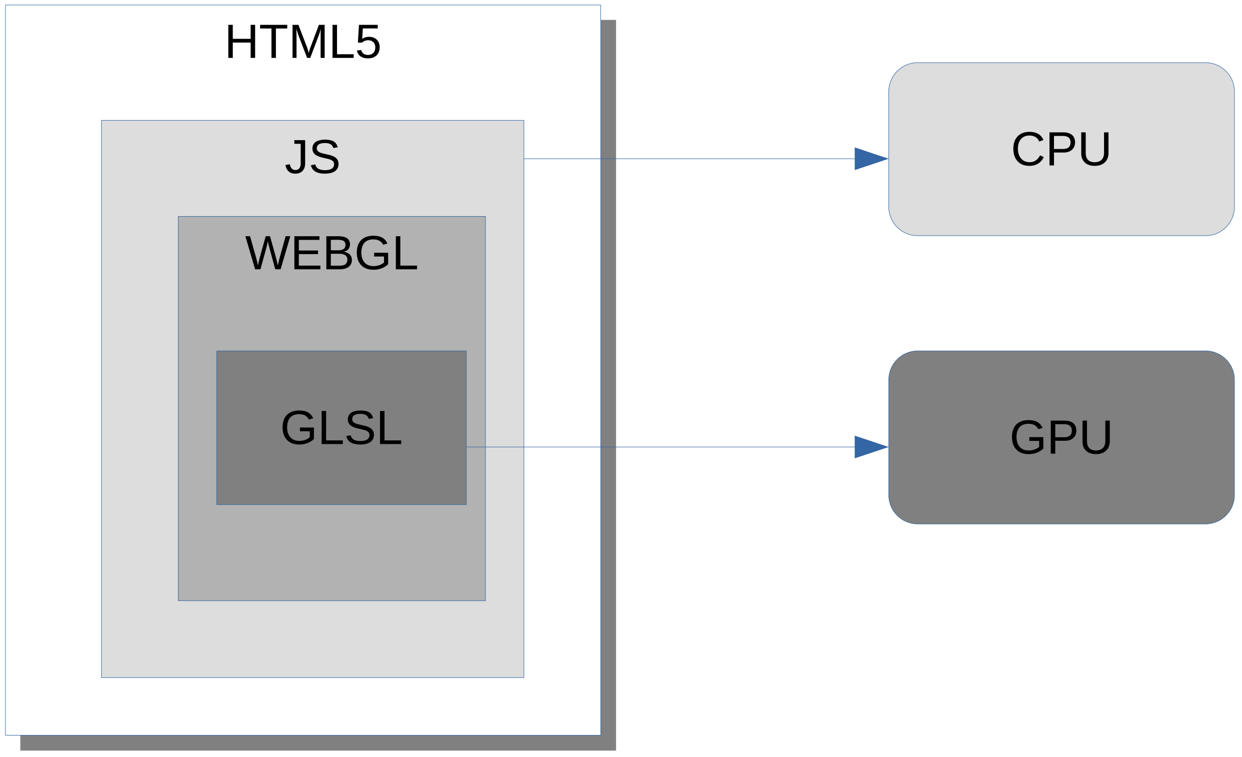
WEBGL

OpenGL is the premier environment for developing portable, interactive 2D and 3D graphics applications. Created in 1992, it is the industry's most widely used and supported 2D and 3D graphics application programming interface (API).

WebGL is OpenGL ES 2.0 for the Web. It's a JavaScript API providing tools to generate and render dynamic 3D hardware accelerated graphics.

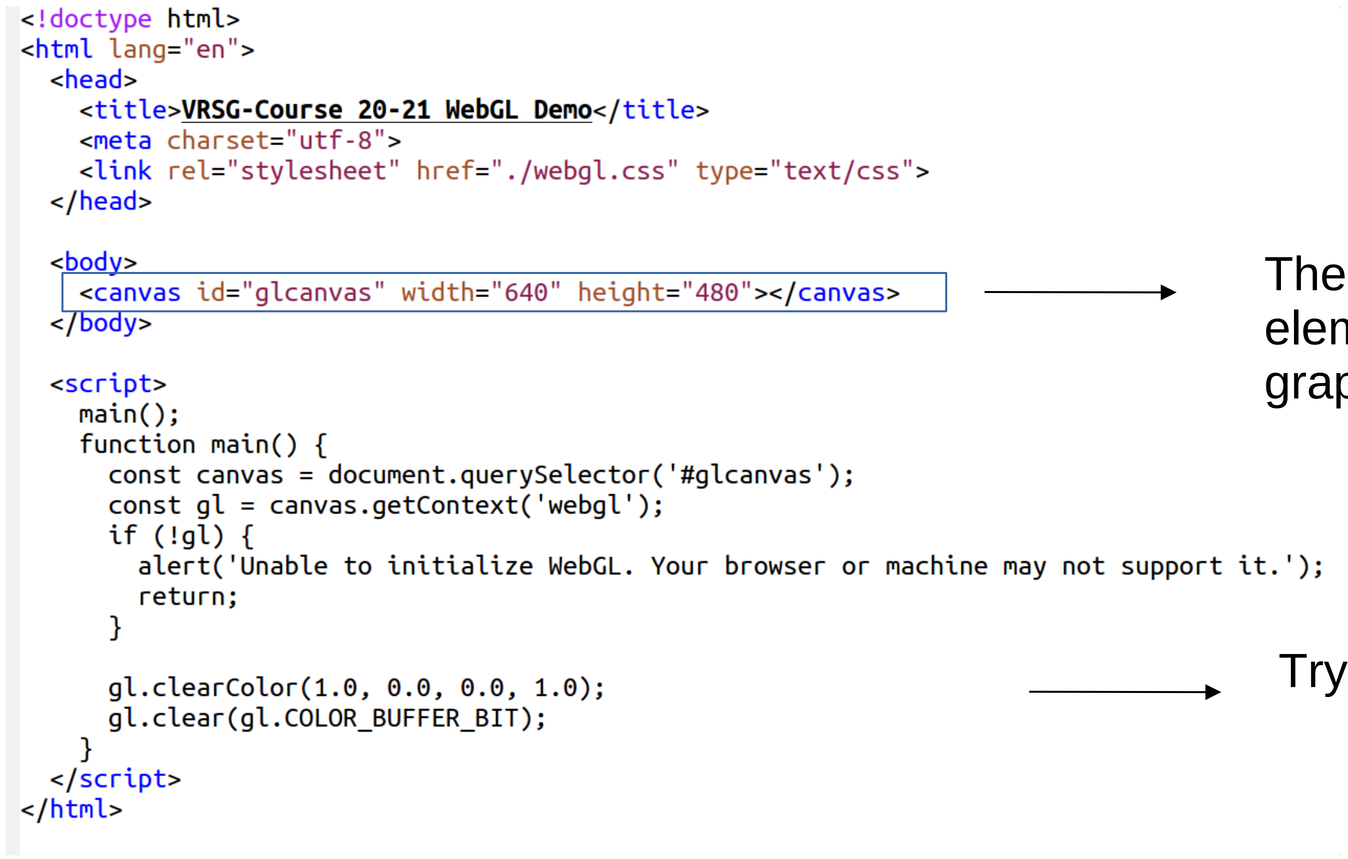
OpenGL has the GLSL (GL shading language) that allow programmers defining how they want their objects to be rendered using the GPU.

WEBGL



WEBGL

Download and run `exwebgl1.html.`



```
<!doctype html>
<html lang="en">
  <head>
    <title>VRSG-Course 20-21 WebGL Demo</title>
    <meta charset="utf-8">
    <link rel="stylesheet" href="./webgl.css" type="text/css">
  </head>

  <body>
    <canvas id="glcanvas" width="640" height="480"></canvas>
  </body>

  <script>
    main();
    function main() {
      const canvas = document.querySelector('#glcanvas');
      const gl = canvas.getContext('webgl');
      if (!gl) {
        alert('Unable to initialize WebGL. Your browser or machine may not support it.');
        return;
      }

      gl.clearColor(1.0, 0.0, 0.0, 1.0);
      gl.clear(gl.COLOR_BUFFER_BIT);
    }
  </script>
</html>
```

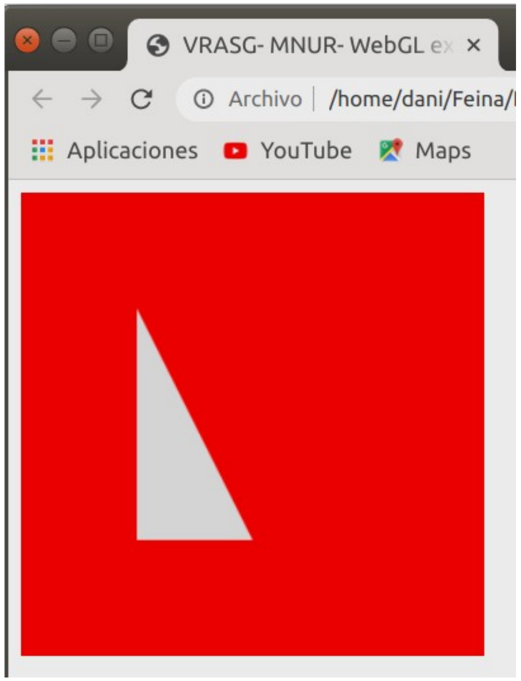
The HTML <canvas> element is used to draw graphics on a web page.

Try to change the color

WebGL

Programming with WebGL is not easy. You need to understand the concept of shader (vertex shader and fragment shader) (see ex_webgl_2.html).

It consists of differents steps:
- define the geometry you want to draw: (define the vertices and pass them to a vertex buffer)
- create, compile and link the shader programs
- associate the shaders with the buffer
- draw the geometry



WebGL

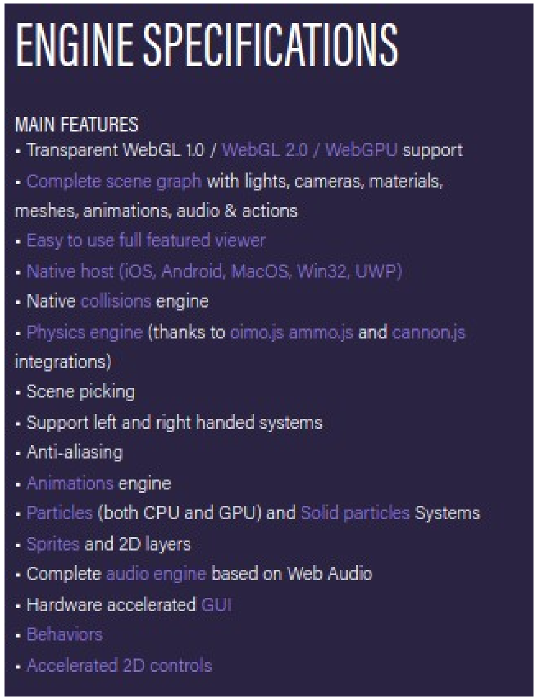
Several frameworks have been created on top of webGL to ease programmers tasks. They provide geometric primitives and geometry readers along with shaders.

Among others, two of the most populars are three.js and babylon.js.

Starting next week we'll use **babylon**.

Babylon

https://www.babylonjs.com/


ENGINE SPECIFICATIONS

MAIN FEATURES
- Transparent WebGL 1.0 / WebGL 2.0 / WebGPU support
- Complete scene graph with lights, cameras, materials, meshes, animations, audio & actions
- Easy to use full featured viewer
- Native host (iOS, Android, MacOS, Win32, UWP)
- Native collisions engine
- Physics engine (thanks to oimo.js ammo.js and cannon.js integrations)
- Scene picking
- Support left and right handed systems
- Anti-aliasing
- Animations engine
- Particles (both CPU and GPU) and Solid particles Systems
- Sprites and 2D layers
- Complete audio engine based on Web Audio
- Hardware accelerated GUI
- Behaviors
- Accelerated 2D controls

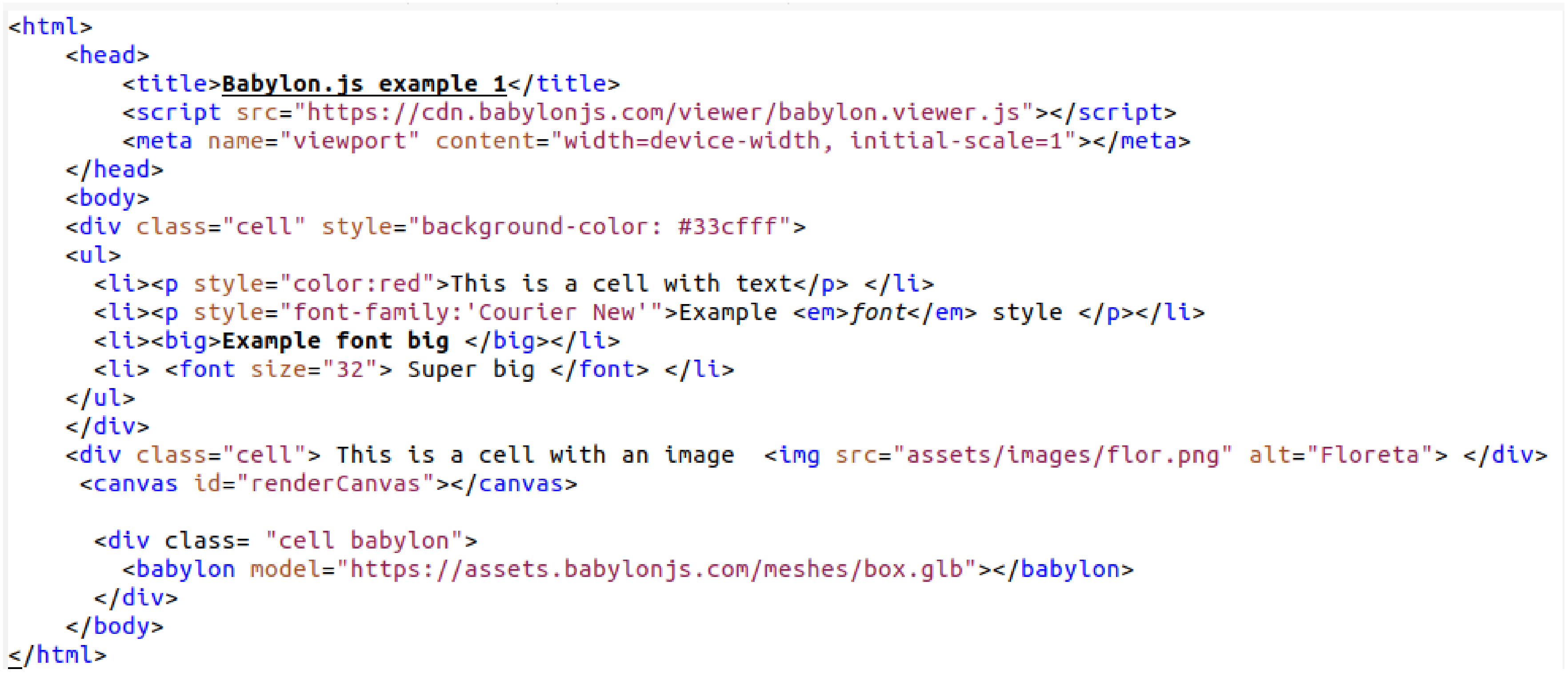
A Web rendering and game engine. It provides a layer onto webgl making it easier to program 2D and 3D graphics, animations, special effects and realistic rendering. It includes several tools such as the **playground** to test the applications and the **sandbox** to tests models.

Very good documentation at: https://doc.babylonjs.com/

Babylon

Let test the first example. Run `ex_babylon_1.html` by clicking on it or opening a browser from the command line with the html file (e.g `x-www-browser example_bab_2.html` in linux or, in windows, `start chrome ex_babylon_1.html`).

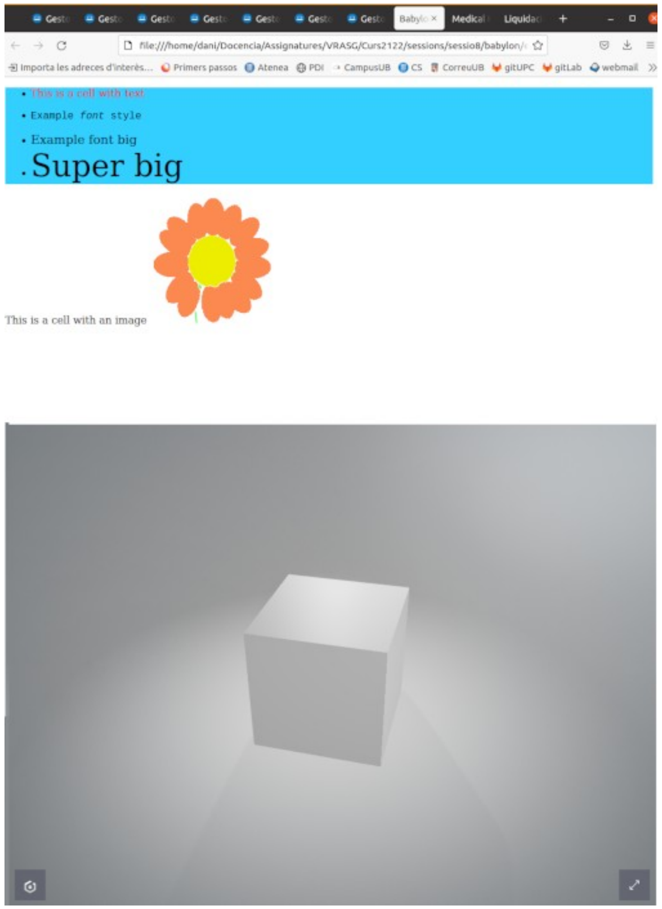
It uses a scene model (`box.glb`) stored in babylon's server and a script (`babylon.viewer.js`) also stored in babylon's server.



```html
<html>
    <head>
        <title>Babylon.js example 1</title>
        <script src="https://cdn.babylonjs.com/viewer/babylon.viewer.js"></script>
        <meta name="viewport" content="width=device-width, initial-scale=1"></meta>
    </head>
    <body>
    <div class="cell" style="background-color: #33cfff">
    <ul>
      <li><p style="color:red">This is a cell with text</p> </li>
      <li><p style="font-family:'Courier New'">Example <em>font</em> style </p></li>
      <li><big>Example font big </big></li>
      <li> <font size="32"> Super big </font> </li>
    </ul>
    </div>
    <div class="cell"> This is a cell with an image  <img src="assets/images/flor.png" alt="Floreta"> </div>
     <canvas id="renderCanvas"></canvas>

      <div class= "cell babylon">
        <babylon model="https://assets.babylonjs.com/meshes/box.glb"></babylon>
      </div>
    </body>
</html>
```

Babylon example 1



- This is a cell with text
- Example font style
- Example font big
- Super big

This is a cell with an image

You can see below our html paragraphs a canvas showing a cube, lighted and rendered from a specific viewpoint. The cube rotates around axis y. These elements: camera, object and light have been created and stored as the scene `box.glb` that is now rendered.

The model is on the babylon server:

```
<babylon model="
https://assets.babylonjs.com/meshes/box.glb
">
</babylon>
```

Next week
more
Thank you