```
// Grab the first name textbox, last name textbox, name material drop down, and name animation drop down
const fNameTextbox = document.getElementById("textbox-fname");
const lNameTextbox = document.getElementById("textbox-lname");
const nameMaterialSelect = document.getElementById("name-material-select");
const nameAnimationSelect = document.getElementById("name-animation-select");

// Grab the slogan textbox, slogan material drop down, and slogan animation drop down
const sloganTextbox = document.getElementById("textbox-slogan");
const sloganMaterialSelect = document.getElementById("slogan-material-select");
const sloganAnimationSelect = document.getElementById("slogan-animation-select");

// Grab the background dropdown
const backgroundSelect = document.getElementById("background-select");

// Grab the render button
const renderButton = document.getElementById("render-btn");

// Grab the document body
const body = document.body;


// Adds an event listener to the render button, that when clicked, will grab the values from all the text boxes and drop down menus and send them to the initializer function for the 3d scene.
renderButton.addEventListener("click", (ev) => {

#
    // Grab the values from the first name textbox, the last name textbox, and the name material drop down
    let fName = fNameTextbox.value;
    let lName = lNameTextbox.value;
    let nameMat = nameMaterialSelect.value;
    let nameAnim = nameAnimationSelect.value;

    // Grab the values of the slogan textbox and the slogan material drop down
    let slogan = sloganTextbox.value;
    let sloganMat = sloganMaterialSelect.value;
    let sloganAnim = sloganAnimationSelect.value;

    // Grab the value of the background drop down
    let bg = backgroundSelect.value;


    // This is the full, capitalized version of the first and last name combined. Typically you should abstract this into a function, or import a library to handle it, but we only need to do it twice here, which is simple enough.
    let fullName = "";
    // Capitalize the first letter of the first name and add it to fullName
    fullName += fName.charAt(0).toUpperCase() + fName.slice(1);
    // Add a space to fullName
    fullName += " ";
    // Capitalize the first letter of the last name and add it to fullName
    fullName += lName.charAt(0).toUpperCase() + lName.slice(1);
```

```
    //  Call the function to initialize and render the 3d scene,
    init(fullName, nameMat, nameAnim, slogan, sloganMat, sloganAnim, bg);
})

function init(fullName, nameMatChoice, nameAnimChoice, slogan, sloganMatChoice,
sloganAnimChoice, bg) {


    //  Log the parameters into the console for clarity
    console.log("Full Name: ", fullName, "| Name Material Choice: ", nameMatChoice, "| Name
Animation Choice: ", nameAnimChoice, "| Slogan Animation Choice: ", sloganAnimChoice, "|
Slogan: ", slogan, "| Slogan Material Choice: ", sloganMatChoice, "| Background Choice: ", bg);


    //  1) The scene (think of it as a container) that... contains all the meshs, lights, particles, or
whatever you want to display
    //  2) The camera that views the scene
    //  3) The object that renders what the camera sees and puts it into a format the DOM can
handle
    let scene, camera, renderer;

    //  1) The orbital controls that allow us to zoom in and out and rotate the view. Note that it's
not that text that is rotating but the camera rotating around the text.
    //  2)  This object lets us import fonts from a default list Three.js has to give to the text
geometry. The text geometry must have a font to work
    let controls, fontLoader;

    //  1) A directional light to..give light to the scene, there are several different types of lights, a
directional light allows us to send light in a specefic direction. Unlike a point light, that is a
point of light that sends lights in all directions around it. A directional light is like a flash light, or
led panel. A point light is like a bare lightbulb.
    let dirLight;

    //  The mesh (actual renderable object) of the text for the full name and slogan
    let fullNameMesh, sloganMesh;



    //  Creates the scene. The scene contains all the meshes, particles, lights, etc.. You can think
of it as a container in a way. Containing everything you want rendered/displayed
    scene = new THREE.Scene();
```

```
    //  Creates the camera. There are several different types of camera's, this one being a perspective camera, such as a stereo camera that using two perspective camera to create 3d anaglyphs. You'll probably be using perpsective cameras for most your projects
    //  Parameters:
    //  1)  This is the Field of View (FoV) of the camera in degrees
    //  2)  This is the aspect ratio of the camera, "window.innerWidth / window.innerHeight" is a pretty common ratio
    //  3)  Anything closer to the camera than this many units will not be rendered
    //  4)  Anything further away from the camera than this many units will not be rendered
    camera = new THREE.PerspectiveCamera(60, window.innerWidth / window.innerHeight, 1, 10000);
    //  Sets the camera's positon to 500 units on the Z axis
    camera.position.set(0, 0, 500);
    //  Tells the camera to look at the center of the scene
    camera.lookAt(scene.position);


    //  Creates a new rendering engine, this takes the scene and camera data, renders everything and converts that into something the DOM can use.
    renderer = new THREE.WebGLRenderer();
    //  Sets the size of the render to the width and height of the window containing it.
    renderer.setSize(window.innerWidth, window.innerHeight);
    //  Adds the DOM version of the render to the body of the webpage;
    body.appendChild(renderer.domElement);


    //  These are the controls that let us zoom and orbit around the scene. Take not that it's not the meshs that are rotating  but the camera rotating around the meshes.
    controls = new THREE.OrbitControls(camera, renderer.domElement);



    //  Adds a light to... light the scene! This type of light is directional, there are other types of lights like point lights. Directional lights send light in a specefic direction and point lights send light in all directions. Think of a directional light as a led panel or a photographer's softbox and a point light as a bare lightbulb or the sun.
    dirLight = new THREE.DirectionalLight(0xffffff);
    //  Sets the lights position and orientation.
    dirLight.position.setScalar(10);
    //  Adds the light to the scene.
    scene.add(dirLight);


    //  Creates a fontLoader object that can load fonts from a json file
    fontLoader = new THREE.FontLoader();

    //  This function is interesting, the first parameter is the link to the font to use, the second parameter is the function to send the loaded font object to, I highly recomend making helper functions to automate as many tasks as you can to keep this function as clean as possible.

    //  This example from the Three.js documentation really helped me understand this

    // var loader = new THREE.FontLoader();
    // var font = loader.load(
```

```
//     // resource URL
//     'fonts/helvetiker_bold.typeface.json',

//     // onLoad callback
//     function ( font ) {
//         // do something with the font
//         scene.add( font );
//     },

//     // onProgress callback
//     function ( xhr ) {
//         console.log( (xhr.loaded / xhr.total * 100) + '% loaded' );
//     },

//     // onError callback
//     function ( err ) {
//         console.log( 'An error happened' );
//     }
// );



    fontLoader.load('https://threejs.org/examples/fonts/helvetiker_bold.typeface.json', function
(font) {
        //  Creates the name mesh
        fullNameMesh = createTextMesh(fullName, font, 80, materialSelector(nameMatChoice));
        //  Creates the slogan mesh.
        sloganMesh = createTextMesh(slogan, font, 50, materialSelector(sloganMatChoice));
        //  Sets the position of the slogan mesh 100 units below the y axis.
        sloganMesh.position.y = -100;

        //  Adds the name mesh and the slogan mesh to the scene
        scene.add(fullNameMesh);
        scene.add(sloganMesh);
    });


    // We are going to make some helper functions to make some tasks more abstracted and
harder to mess up. JavaScript files, especially three.js ones can get complex very fast, so it's
really important to have good formatting and abstract everything you can. Comments can
really help to

    //  This is the function that actually renders the scene. This is where we put our animation
functions.
    function render() {
        //  RequestAnimationFrame(render) starts a loop that refreshes the scene everytime the
screen refreshes (typically 60hz or 60 times per second).
        requestAnimationFrame(render);

        //Animate our meshes
        animationSelector(nameAnimChoice)(fullNameMesh);
        animationSelector(sloganAnimChoice)(sloganMesh);
```

```
    //  Tells the render engine to render the scene with our scene object and camera object.
    renderer.render(scene, camera);
}

function animationSelector(choice) {
    switch (choice) {
        case "Stay Still":
            var fn = function (mesh) {
            }
            return fn;
            break;
        case "Spin":
            var fn = function (mesh) {
                mesh.rotation.y += 0.05;
            }
            return fn;
            break;
        case "Grow and Shrink":
            var fn = function (mesh) {
                if (mesh.scale.x < 2.0) {
                    mesh.scale.x += 0.01;
                    mesh.scale.y += 0.01;
                    mesh.scale.z += 0.01;
                } else if (mesh.scale.x > 2.0) {
                    mesh.scale.x = 1;
                    mesh.scale.y = 1;
                    mesh.scale.z = 1;
                }
            }
            return fn;
            break;
        case "Rotate":
            var fn = function (mesh) {
                mesh.rotation.z += 0.05;
            }
            return fn;
            break;
    }
}

//  This function automates making a mesh object, and combing a geometry object with a material to make a mesh
function createTextMesh(text, font, size, mat) {
    var geo = new THREE.TextGeometry(text, {
        font: font,
        size: size,
        height: 5,
        curveSegments: 12,
        bevelEnabled: true,
        material: 0,
        extrudeMaterial: 1
    });
```

```
        geo.center();
        geo.computeBoundingBox();

        return new THREE.Mesh(geo, mat);
    }

    //  This function takes in the value from one of the two material drop downs and returns a corresponding material.
    function materialSelector(choice) {
        switch (choice) {
            case "Red":
                return new THREE.MeshPhongMaterial({
                    color: 0x9e0031,
                    specular: 0x555555,
                    shininess: 30
                });
                break;
            case "Blue":
                return new THREE.MeshPhongMaterial({
                    color: 0x40476d,
                    specular: 0x555555,
                    shininess: 30
                });
                break;
            case "Green":
                return new THREE.MeshPhongMaterial({
                    color: 0x436436,
                    specular: 0x555555,
                    shininess: 30
                });
                break;
            case "Polka Dots":
                return new THREE.MeshPhongMaterial({
                    color: 0xc8bfc7,
                    specular: 0x555555,
                    shininess: 30
                });
                break;
        }
    }


    //  Calls the render function we described above and starts the render loop.
    render();

}
```