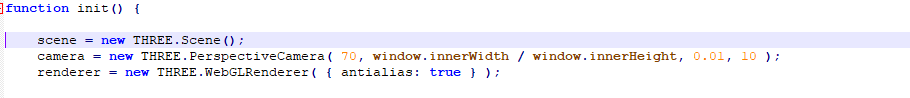
Hey Guys. Welcome to my blog. Today I want to show you guys one of the JavaScript libraries that I think is pretty darn cool, it’s called three.js. What three does is allows you to create 3D objects in a browser and have beautiful visuals directly on the webpage, without any additional software being installed.

It can be used for games, animations and I saw an example of a website that used it to showcase its furniture. It can do some pretty powerful things. So without further ado let me show you how to use it.

Before you jump straight into doing stuff there are a couple things that need to be initialized first, you need:

1. A Scene: In this context a scene is in shorthand what is on screen.
2. A Camera: A camera would be the position that the user will start at, which can be modified to make them look like they are moving or looking around.
3. A Renderer: which makes the objects, once they have a mesh and a shape (more on this later).

Which can be done by the following code:



So what these do is they create you a camera, so that will be where you are “looking” at the graphics from, that can be changed dynamically depending on where you need it. The Scene is the sort of like a backdrop so you add everything to the scene, you can have more than 1 depending on what the plan for it is, although for now 1 is good. Finally you need a renderer which allows the mesh objects to be rendered or displayed on the screen. Ok so you have that out of the way, we want to create our first object:

All objects need to be defined through geometry in the form of



Where the blank is your type of geometry. There are many types but to start the basic ones are:

1. Square which can be made with THREE.BoxGeometry(x,y,z) the parameters are the dimensions of the square.
2. Not to be confused with a Plane type which is a 2 dimensional square which can be made with THREE.PlaneGeometry(x,y).
3. For Circles you have to use the THREE.SphereGeometry(x,y,z), the parameters for this one are radius, height segments and width segments, which determine how round the sphere is, the higher the more round it is.

There are many different types all available on the documentation so since those are the basic ones we will stick with those for now.

Now that we have our geometric item we need to make a material for it, so material is a mix of the color and the texture of the shape. There are again many of these available on the official documentation for the library (link provided at the bottom of the page). For now though we will identify our mesh with the simple MeshBasicMaterial,



This will give you a very plain, very white item, with no shading which is a result of the basic material. So just to help demonstrate the power of Three.js replace that line and put in MeshNormalMaterial, like this:



Now that we have an object and a material we need to combine them, the THREE.Mesh() method works for that. It takes 2 parameters, one of type geometry, and one of type material. Plug in your items into the mesh and your code should look like this:



Almost done, we have to do 2 more things, so we need to add our new item to the scene. This is an easy one there is one line of code that looks like this:



Almost all set up, the last thing we need is to use that renderer that we made earlier actually display the objects on the screen. First thing to do is set the renderer size so it knows how big the area will be. So call the setSize() method for the renderer and give it a window size. A little trick is to give it window.innerWidth and window.outerWidth and it will fit it to the size of the screen being used, that would look like this:



Great… well sort of, 3D components in Three make everything start at the exact same spot so that means that your object is at one point and the camera is inside the object, so the object is not super visible. So quickly just add in a line to offset your camera. So you can see your object being rendered. That would look something along the lines of this:



The number would be a point on the graph, so the larger it is the smaller your item will be because when we set up the camera we used the THREE.PerspectiveCamera. This fact is a nice segue into the way items are positioned in the 3D space.

All of the items in the scene are dropped right at (0,0,0), which is the origin of the graph, and the reason you can’t see it is because your camera was also at that same point. All items have specified coordinates which can be manipulated with $variable.position.coordinate = someInt, where coordinate is x,y, or z, and $variable is any object.

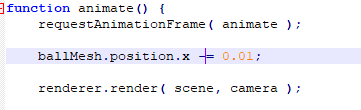
The important ones to know are:

1. Position – which is where the object is located.
2. Rotate – This rotates the object.
3. Size – This determines the size of the object.

There are an abundance of other ways to manipulate them but that’s the gist of it.

Now that you have a simple object the main draw of using Three.js is it has an method called requestAnimationFrame() which takes a function (itself) as a parameter, and it will call that method every time the monitor refreshes

Inside that method you want to put anything you want your animation to do using whichever loops or statements you need to get it done. So to make your object move to the left put in a the function a line like this:



This will call itself every time the monitor refreshes, your object will move .01 pixels to the left every time which is roughly 60 times per second on a standard monitor. The renderer is being updated at the end of the function which will make the object move smoothly.

.

So that is pretty much the basics to Three.js. It’s a pretty powerful library for anything that has to do with animation moving parts, because it handles it much more gracefully then you could without it. All of the code in this how to can be found here: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The official documentation which contains all the important information about this library here:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

If you want to check out something to get some ideas flowing about how you would like to use it, take a look at this: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

All of the code for it is available here: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

So that’s all for now, have fun with Three.js and until next time; keep your stick on the ice.