Code that needs to be executed once:

* the usual code for creating a scene, camera, light
* load textures and create 3 materials
* initialize empty array to hold cube meshes
* initialize empty parallel array to hold information about each cube: rotation rate, direction from the cube’s starting position to the camera
* Define event handlers:
  + window.onresize
  + document.onmousedown = function (event) {  
    switch (event.button) {  
     case 0: button = "left"; break;  
     case 1: button = "middle"; break;  
     case 2: button = "right"; break;  
     }
  + window.oncontextmenu = function() { return false; }

Code that needs to be executed repeatedly (so in the render function):

* Increment the variable that keeps track of how long it’s been since the last cube was spawned:  
  totalDelta += clock.getDelta(); // getDelta() adds # seconds since last call to getDelta()  
  // clock is a THREE.Clock(), which should also have been created outside of the render function.

// totalDelta should have been created and initialized to 0 outside of the render function.

* If the value in totalDelta is greater than the delay you want between cube spawnings (in seconds), do the following:
  + get a random starting position for a new cube
  + based on the random position and the camera position, calculate the direction the cube needs to move to approach the camera. This will be the camera position minus the starting position of the cube.  
    The camera position is a THREE.Vector3, and if you make the starting position a THREE.Vector3 too, you can subtract them using camera.position.sub(cubeStartPosition), but this changes the camera.position! So use new THREE.Vector3().copy(camera.position).sub(startPosition).normalize().
  + Randomly decide on material for new cube
  + Create new cube.
  + Push cube on array of meshes
  + Push direction, random rotation info onto cubeInfo array