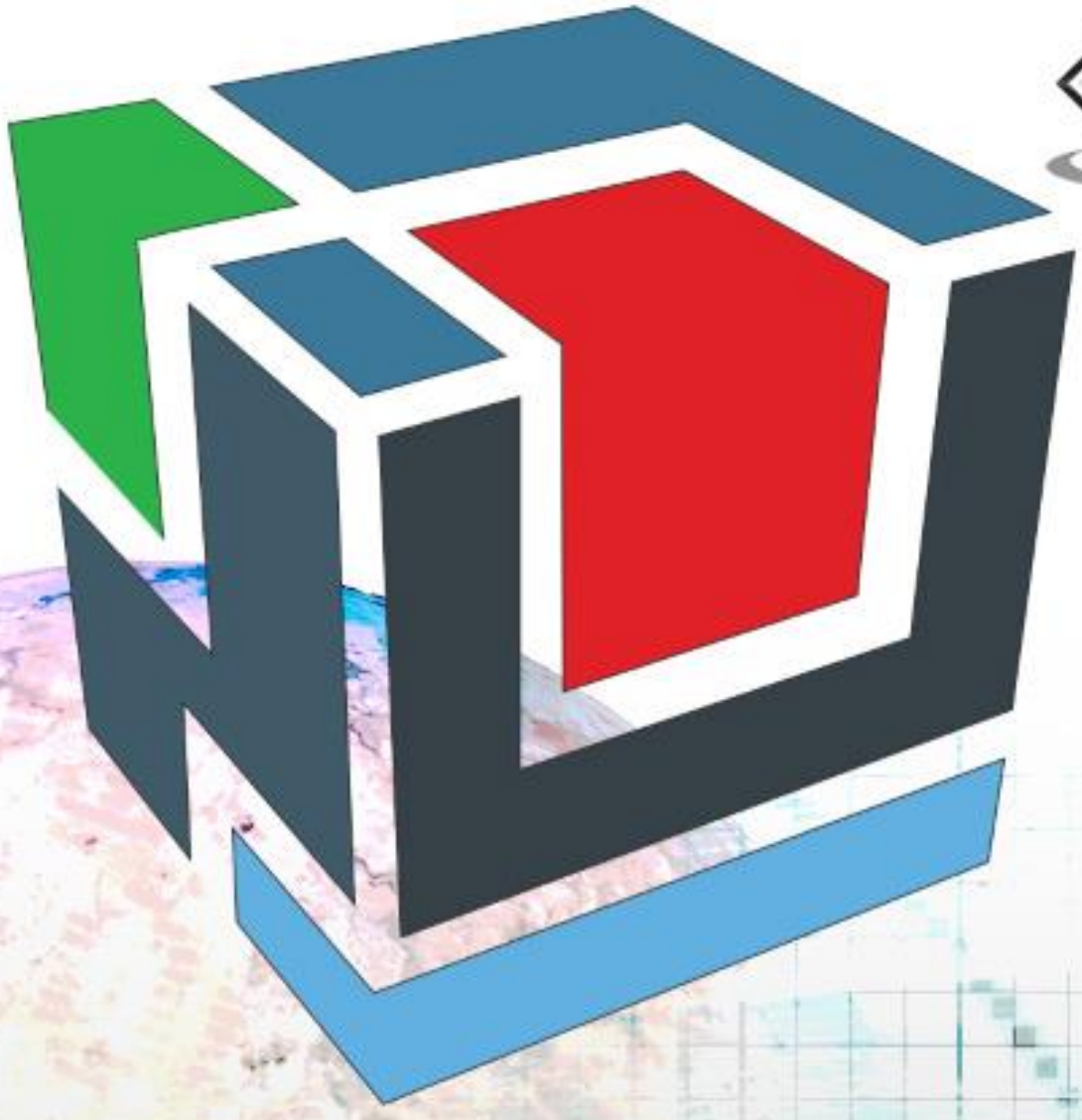


BabylonJS

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia https://twitter.com/tutorialspoint

About the Tutorial

BabylonJS is a javascript framework for building 3D games with HTML5 and WEBGL.It is anopen source framework and is hosted on github. The official website of BabylonJS is www.babylonJs.com.

Audience

This tutorial is designed for software programmers who want to learn the basics of BabylonJS and its programming concepts in simple and easy ways. This tutorial will give you enough understanding on various functionalities of BabylonJS with suitable examples.

Prerequisites

Before proceeding with this tutorial, you should have a basic understanding of HTML, CSS, JavaScript, and Document Object Model (DOM).

Copyright &Disclaimer

© Copyright 2019 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

tutorialspoint
SIMPLYEASYLEARNING

Table of Contents

tutorialspoint
SIMPLYEASYLEARNING

tutorialspoint
SIMPLYEASYLEARNING

tutorialspoint
SIMPLYEASYLEARNING

tutorialspoint
SIMPLYEASYLEARNING

tutorialspoint
SIMPLYEASYLEARNING

1. BabylonJS — Introduction

Babylon.js is a javascript open-source framework which is used to develop 3D applications/ video games for the web. The official website of BabylonJS is www.babylonJs.com.

Using Babylon.js framework is easy for the users. It contains all the required tools to create and manage 3D objects, special effects, and sounds, etc.

Babylon.js is one of the most popular 3D game engines and is widely used by developers. Being a 3D library, it provides built-in functions. These functions help you implement common 3D functionality with efficient and accurate ways.

It is developed using TypeScript language based on WebGL and javascript.

What is WebGL?

WebGL (Web Graphics Library) is the new standard for 3D graphics on the Web. It is designed for the purpose of rendering 2D graphics and interactive 3D graphics. It is derived from OpenGL's ES 2.0 library which is a low-level 3D API for phones and other mobile devices. WebGL provides similar functionality of ES 2.0 (Embedded Systems) and performs well on modern 3D graphics hardware.

The TypeScript

By definition, "TypeScript is JavaScript for application-scale development."

TypeScript is a strongly typed, object oriented, compiled language. TypeScript is both a language and a set of tools. TypeScript is a typed superset of JavaScript compiled to JavaScript. In other words, TypeScript is JavaScript plus some additional features.

The goal of TypeScript language is to improve and secure the production of JavaScript code.Since BabylonJS is developed using TypScript, it is robust and secure.

tutorialspoint
SIMPLYEASYLEARNING

2. BabylonJS — Environment Setup

In this chapter, we will learn how to set up the environment for BabylonJS.

To start with the setup, visit the official website of Babylon.js- www.babylonJs.com. Go to the download section and choose the latest version of Babylon.js and store in your folder.

The screenshot for the same is as follows:

You can also go to GITHUB and clone the babylonjs project -

https://github.com/BabylonJS/Babylon.js.git

In your command line type -

```
git clone https://github.com/BabylonJS/Babylon.js.git
go to cd BabylonJS/
npm install
```

The required files will be available in the BabylonJS folder.

You can use the VSCode (Microsoft Visual Studio Code) for editing.The code comes with builtin functionalities like highlighting if any error, hightlighting the syntax, etc. You can use the editor of your choice and it is not mandatory to use only VSCode.

tutorialspoint
SIMPLYEASYLEARNING

3. BabylonJS — Overview

BabylonJS is an open sourceJavascript framework for building 3D games with HTML5 and WEBGL.It is hosted on github.The official web site of BabylonJS iswww.babylonjs.com.

In the world of 3D Animation,the shapes are drawn with triangles.With WebGL, the complexity increases with the deluge of coding that is involved in the process. BabylonJS is the easy solution that pitches in to mitigate the increased complexity. Here, the API for lights, cameras, engine are easy to handle and to create 3D objects.

The source code of babylonJS is coded in typescript.It is compiled to Javascript and made available tothe end user.

To start working with Babylonjs, download the babylonjs file, host it at your end and you are ready to get started to write your 3D code.

BabylonJS is developed by Microsoft employees in the year 2016.David Catuhe, a Principal Program Manager for the Window & Devices Group at Microsoft is the main person behind developing BabylonJs and making it a big success.

To run BabylonJS, we need modern browsers with WEBGL support. Latest browsers i.e Internet Explorer 11+, Firefox 4+, Google Chrome 9+, Opera 15+, etc. does have WEBGL support and the demos can be executed on same to see the output.

BabylonJs offers following features which help to create different types of 3D-scenes:

- Shapes like box, sphere, scylinder, cone, height ground
- Cameras, Lights
- Meshes, textures, Materials
- Sprites
- Morphing
- Mesh Intersection and collision detection
- Physics engine plug-in
- Action Manager
- SolidParticles
- Instances and Particles
- Support for Bones and Skeletons
- Adding music and sound to the scene

In addition to its own meshes, BabylonJS also allows the use of meshes created from third party 3D softwares like Blender, FBX and 3DS Max.

tutorialspoint
SIMPLYEASYLEARNING

Blender

Blender is anopen-source 3D computer graphics software product used to create animated scenes, 3D printed models, video games, etc. Blender gives .bablyon files which are to be used with Babylon to render meshes. How to convert files fom blender to .babylon is explained in subsequent chapters of this tutorial.

FBX

Also called thefilmbox, it helps with 3D animation and texture painting software. The FBX files are saved with the.fbx extension.

MAX

The MAX software helps you in creating massive world in games, stunning scenes for designs and engaging virtual reality experiences.

tutorialspoint
SIMPLYEASYLEARNING

4. BabylonJS — Basic Elements

Babylon.js is a popular framework to help build 3D games for developers. It has built-in functions to implement 3D functionalities. Let us build a simple demo using Babylon.js and understand the basic functionalities required to get started.

We will first create a demo which contains the basic elements of Babylon.js. In addition, we will also learn the various functionalities of Babylon.js.

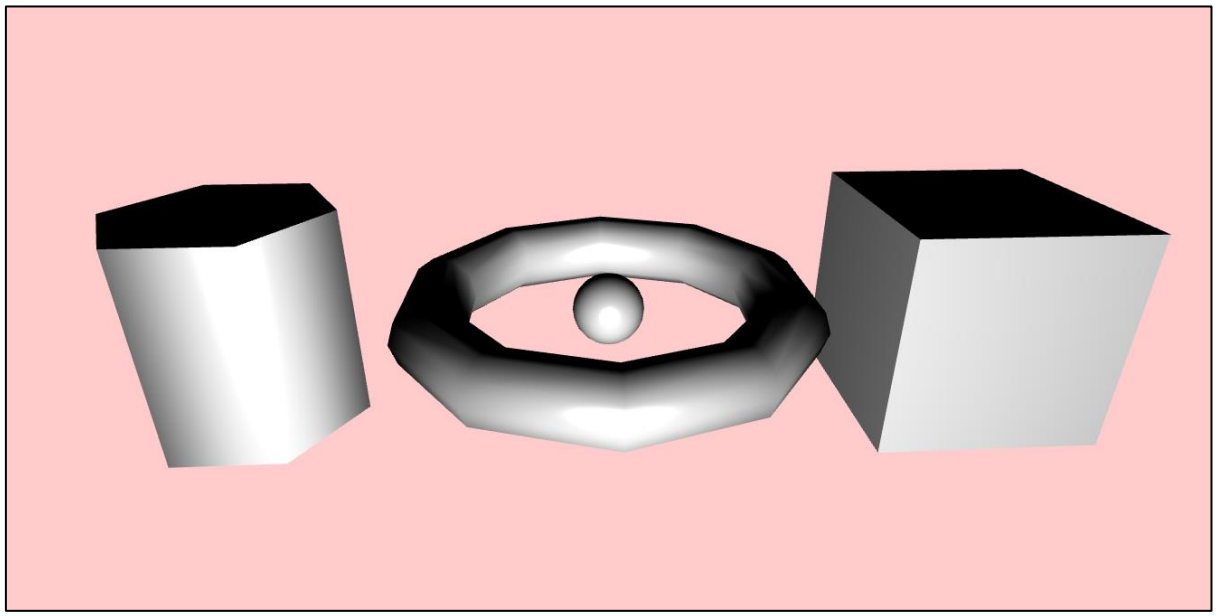
Sample Demo 1

In this section, we will learn how to create a demo containing the basic elements of BabylonJS.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title> Babylon.JS :  Demo2</title>
<script src="babylon.js"></script>
<style>
       canvas { width: 100%; height: 100%;}
</style>
</head>
<body>
<canvas id="renderCanvas"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("renderCanvas");
    var engine = new BABYLON.Engine(canvas, true);
     var createScene  = function() {
            var scene = new BABYLON.Scene(engine);
            scene.clearColor = new BABYLON.Color3(1, 0.8, 0.8);
            var camera = new BABYLON.ArcRotateCamera("Camera", 1, 0.8, 10, new
BABYLON.Vector3(0, 0, 0), scene);
            scene.activeCamera.attachControl(canvas);
            var light0 = new BABYLON.PointLight("Omni", new BABYLON.Vector3(0,
0, 10), scene);
            var origin = BABYLON.Mesh.CreateSphere("origin", 10, 1.0, scene);
```


tutorialspoint
SIMPLYEASYLEARNING

```
            var torus = BABYLON.Mesh.CreateTorus("torus", 5, 1, 10, scene,
false);

            var box = BABYLON.Mesh.CreateBox("box", 3.0, scene);

            box.position = new BABYLON.Vector3(-5, 0, 0);

            var cylinder = BABYLON.Mesh.CreateCylinder("cylinder", 3, 3, 3, 6,
1, scene, false);

            cylinder.position=new BABYLON.Vector3(5, 0, 0);

            return scene;

      };

      var scene = createScene();

      engine.runRenderLoop(function(){

            scene.render();

      });

</script>

</body>

</html>
```



To run BabylonJS, we need modern browsers with WEBGL support. The latest browsers - Internet Explorer 11+, Firefox 4+, Google Chrome 9+, Opera 15+, etc. does have WEBGL support and the demos can be executed on the same platformsto see the output. Create a directory to store the files for babylonjs. Fetch the latest BabylonJSscripts file from BabylonJS site. All the demo links in this tutorial are tested with babylonjs version 3.3.

tutorialspoint
SIMPLYEASYLEARNING

Step 1

- Create a simple html page and include the Babylon.js file.
- Create a canvas tag which is used to render contents by BabylonJSinside the body tag as shown below.
- Add css to the canvas to occupy the full width and height of the screen.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>MDN Games: Babylon.js demo - shapes</title>
     <script src="babylon.js"></script>
<style>
        canvas {width: 100%; height: 100%;}
</style>
</head>
<body>
<canvas id="renderCanvas"></canvas>
</body>
</html>
```

Step 2

Let us now start with the BabylonJScode for rendering contents on the canvas.

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>MDN Games: Babylon.js demo - shapes</title>
     <script src="babylon.js"></script>
<style>
        canvas {width: 100%; height: 100%;}
</style>
</head>
<body>
<canvas id="renderCanvas"></canvas>
<script type="text/javascript">
```


tutorialspoint
SIMPLYEASYLEARNING

```
   var canvas = document.getElementById("renderCanvas");
   var engine = new BABYLON.Engine(canvas, true);
</script>
</body>
</html>
```

Now, add the script tag to the html structure and store the canvas reference in variable canvas.

To get started with Babylon.js, create an engine instance and pass the canvas reference to render on it.

```
<script type="text/javascript">
  var canvas = document.getElementById("renderCanvas");
   var engine = new BABYLON.Engine(canvas, true);
</script>
```

The BABYLON global object contains all the Babylon.js functions available in the engine.

Step 3

In this step, we will first create a scene.

A scene is where all the contents will be displayed. We will create the different types of objects and add the same to the scene to make it visible on the screen. To create scene, add the following code to the already created html structure. At present, we will append to the already created code as a continuation to the above html structure.

```
var createScene  = function() {
      var scene = new BABYLON.Scene(engine);
      scene.clearColor = new BABYLON.Color3(1, 0.8, 0.8);
};
var scene = createScene();
```

The final html file will look as follows:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>MDN Games: Babylon.js demo - shapes</title>
      <script src="babylon.js"></script>
<style>
```


tutorialspoint
SIMPLYEASYLEARNING

```
        canvas {width: 100%; height: 100%;}
</style>
</head>
<body>
<canvas id="renderCanvas"></canvas>
<script type="text/javascript">
   var canvas = document.getElementById("renderCanvas");
   var engine = new BABYLON.Engine(canvas, true);
   var createScene  = function() {
      var scene = new BABYLON.Scene(engine);
      scene.clearColor = new BABYLON.Color3(0, 1, 0);
            return scene;
  };
   var scene = createScene();
</script>
</body>
</html>
```

In the above example, the CreateScene function is defined and the var scene = createScene () is calling the function.

The CreateScene function has the scene created inside it and the next line adds color to the scene, which is done using BABYLON.Color3(1, 0.8, 0.8) and the color over here is pink.

```
var scene = new BABYLON.Scene(engine);
scene.clearColor = new BABYLON.Color3(1, 0.8, 0.8);
```

Executing the above demo link in the browser will not display anything right now on the browser screen. There is one more step to be added to the code which is called the engine.runRenderLoop as in step 4.

tutorialspoint
SIMPLYEASYLEARNING

Step 4

To make the scene actually visible on the screen, we need to render it using engine.runRenderLoop call. Let us now see how this is done.

Rendering Loop

```
engine.runRenderLoop(function(){
     scene.render();
});
```

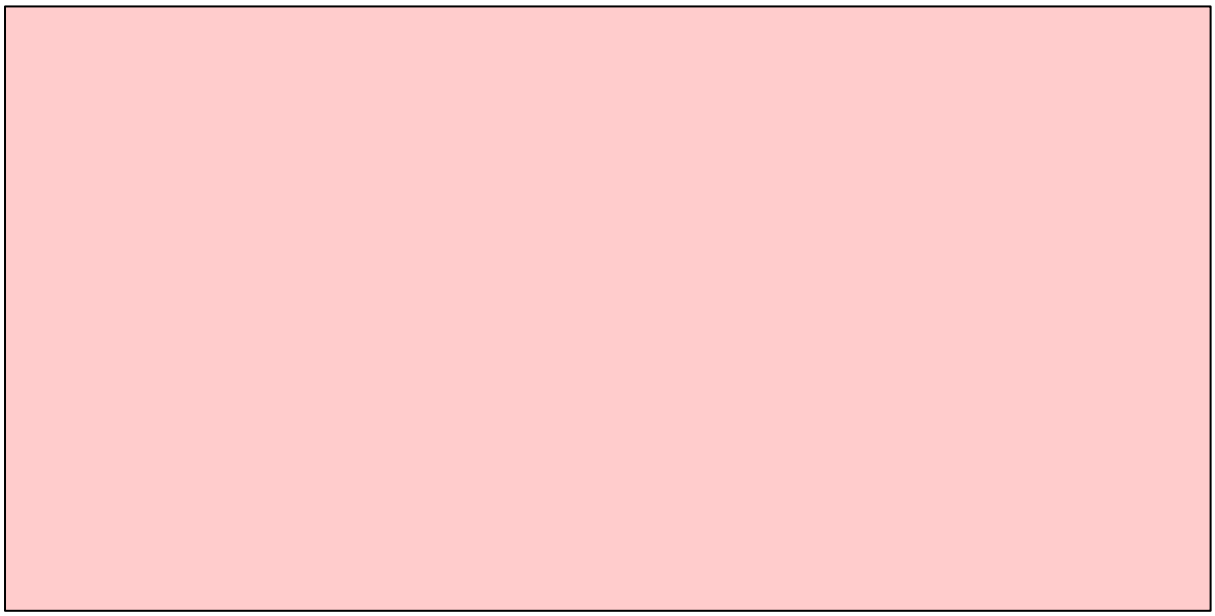
The Engine.runRenderLoop function calls scene.render, which will render the scene and make it visible to the user. The final .html will look as follows:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>BabylonJs - Basic Element-Creating Scene</title>
     <script src="babylon.js"></script>
<style>
        canvas {width: 100%; height: 100%;}
</style>
</head>
<body>
<canvas id="renderCanvas"></canvas>
<script type="text/javascript">
   var canvas = document.getElementById("renderCanvas");
   var engine = new BABYLON.Engine(canvas, true);
   var createScene  = function() {
     var scene = new BABYLON.Scene(engine);
     scene.clearColor = new BABYLON.Color3(1, 0.8, 0.8);
    return scene;
  };
   var scene = createScene();
   engine.runRenderLoop(function(){
            scene.render();
     });
</script>
</body>
```


tutorialspoint
SIMPLYEASYLEARNING

```
</html>
```

Save the above file as basicscene.html and check the output in the browser. The screen that is shown is in pink color as shown below:



Step 5

Now that we have the scene, we have to add camera to it.

Adding Camera and Light

The code given below adds camera to the scene. There are many types of camera that can be used on Babylon.

ArcRotateCamera is a camera that rotates around the target. It can be controlled with mouse, cursor or touch events. The parameters required are name, alpha, beta, radius, target, and scene. Let us discuss the details of the camera in a subsequent section.

```
var camera = new BABYLON.ArcRotateCamera("Camera", 1, 0.8, 10, new
BABYLON.Vector3(0, 0, 0), scene);
```

Now, we need to understand how to add light.

Lights are used to produce the diffuse and specular color received by each pixel. There are many types of lights. We will learn about the different types of lights in the lights section.

Here I am using the PointLight on the scene. The PointLight is emitted in every direction like theSun. The parameters are name, position and the scene to be used on.

To add light, execute the following code:

tutorialspoint
SIMPLYEASYLEARNING

```
var light0 = new BABYLON.PointLight("Omni", new BABYLON.Vector3(0, 0, 10),
scene);
```

Step 6

Let us now see how to add shapes.

Adding of shapes

The demo shared above has 4 shapes added to it.

- Sphere
- Torus
- Box
- Cylinder

To add sphere, execute the following code:

```
var origin = BABYLON.Mesh.CreateSphere("origin", 10, 1.0, scene);
```

Once the sphere is added, the code looks as follows:

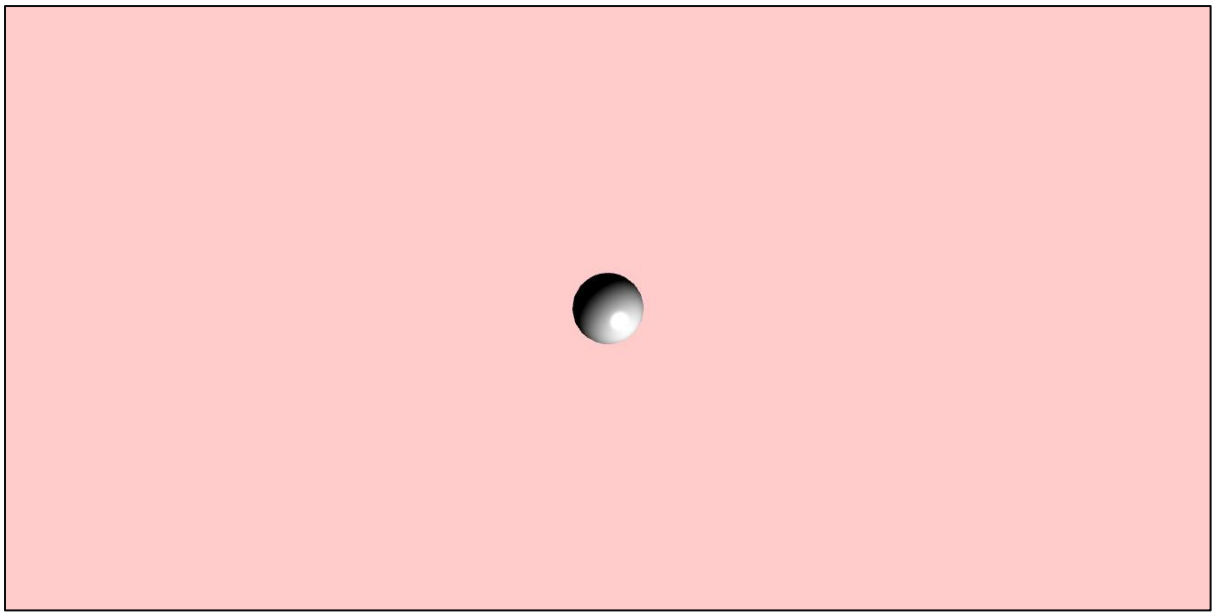
```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>MDN Games: Babylon.js demo - shapes</title>
     <script src="babylon.js"></script>
<style>
        html,body,canvas { margin: 0; padding: 0; width: 100%; height: 100%;
font-size: 0; }
</style>
</head>
<body>
<canvas id="renderCanvas"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("renderCanvas");
    var engine = new BABYLON.Engine(canvas, true);
     var createScene  = function() {
            var scene = new BABYLON.Scene(engine);
            scene.clearColor = new BABYLON.Color3(1, 0.8, 0.8);
            var camera = new BABYLON.ArcRotateCamera("Camera", 1, 0.8, 10, new
BABYLON.Vector3(0, 0, 0), scene);
```


tutorialspoint
SIMPLYEASYLEARNING

```
            var light0 = new BABYLON.PointLight("Omni", new BABYLON.Vector3(0,
0, 10), scene);
            var origin = BABYLON.Mesh.CreateSphere("origin", 10, 1.0, scene);
            scene.activeCamera.attachControl(canvas);
         return scene;
     };
     var scene = createScene();
     engine.runRenderLoop(function(){
            scene.render();
     });
</script>
</body>
</html>
```

Output

The above code generates the following output:



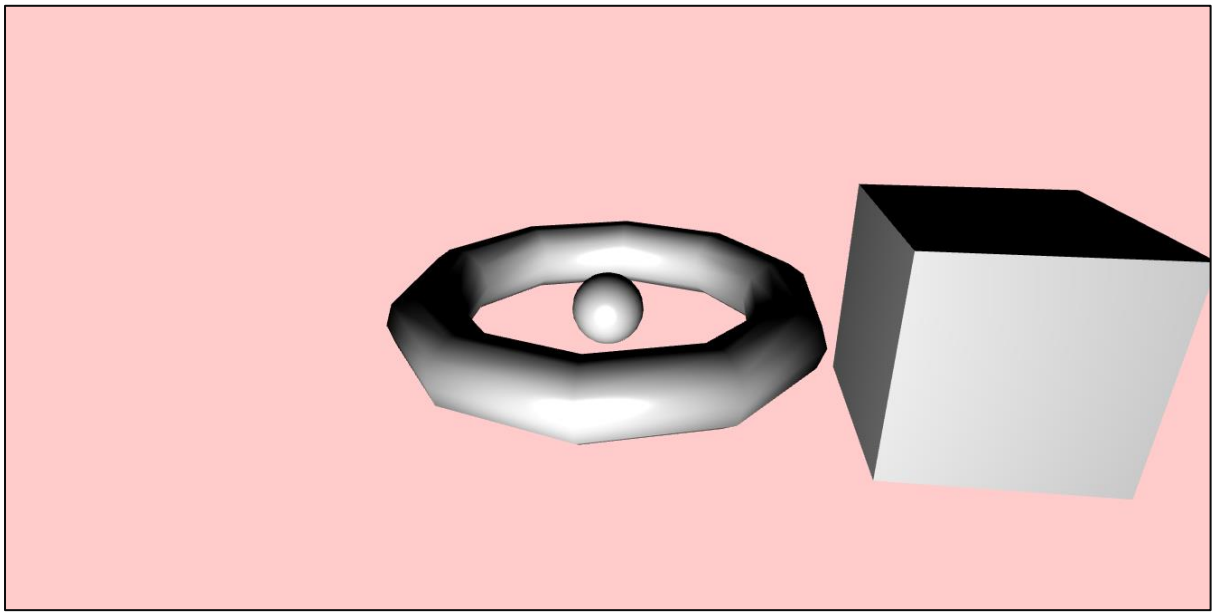
Let us now add the other shapes – the Torus and the Box. Execute the following code to add the Torus shape.

```
var torus = BABYLON.Mesh.CreateTorus("torus", 5, 1, 10, scene, false);
var box = BABYLON.Mesh.CreateBox("box", 3.0, scene);
box.position = new BABYLON.Vector3(-5, 0, 0);
```


tutorialspoint
SIMPLYEASYLEARNING

We will add a position to the box. BABYLON.Vector3(-5, 0, 0) takes the x,y and z direction.

Upon execution, the above code generates the following output:



Let us now add the final shape which is shown in the screenshot above - the cylinder.

```
var cylinder = BABYLON.Mesh.CreateCylinder("cylinder", 3, 3, 3, 6, 1, scene,
false);

cylinder.position=new BABYLON.Vector3(5, 0, 0);
```

The position is added to the cylinder which is x direction 5. The final code is as shown below:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title> Babylon.JS :  Demo2</title>
<script src="babylon.js"></script>
<style>
       canvas { width: 100%; height: 100%;}
</style>
</head>
<body>
<canvas id="renderCanvas"></canvas>
<script type="text/javascript">
```

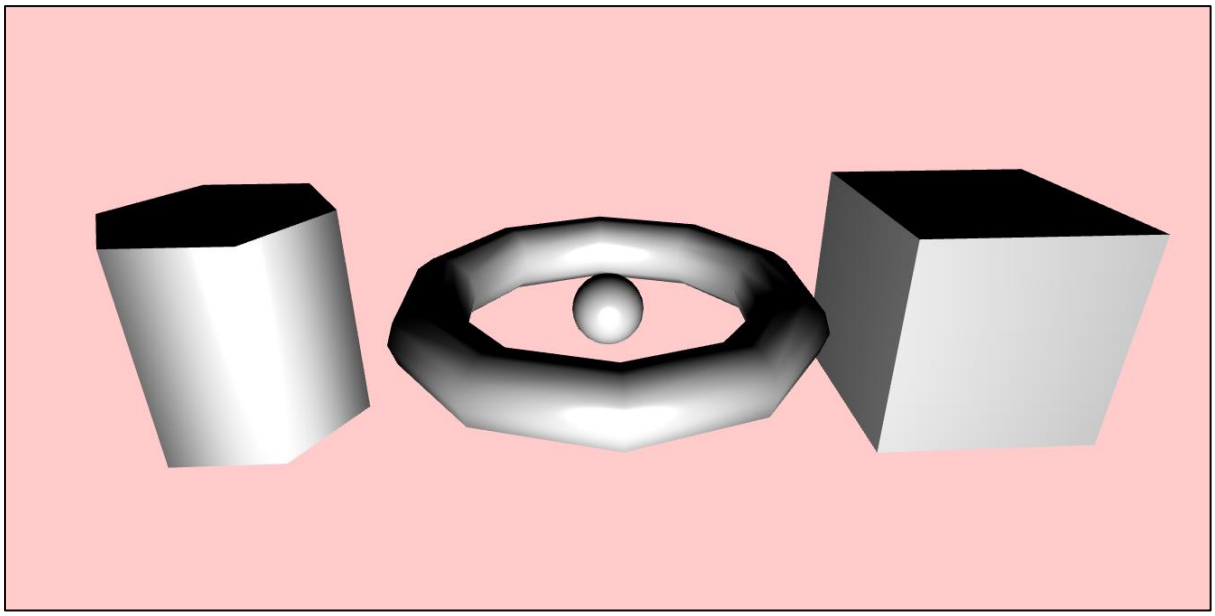

tutorialspoint
SIMPLYEASYLEARNING

```
    var canvas = document.getElementById("renderCanvas");
    var engine = new BABYLON.Engine(canvas, true);
     var createScene  = function() {
            var scene = new BABYLON.Scene(engine);
            scene.clearColor = new BABYLON.Color3(1, 0.8, 0.8);
            var camera = new BABYLON.ArcRotateCamera("Camera", 1, 0.8, 10, new
BABYLON.Vector3(0, 0, 0), scene);
            scene.activeCamera.attachControl(canvas);
            var light0 = new BABYLON.PointLight("Omni", new BABYLON.Vector3(0,
0, 10), scene);
            var origin = BABYLON.Mesh.CreateSphere("origin", 10, 1.0, scene);
            var torus = BABYLON.Mesh.CreateTorus("torus", 5, 1, 10, scene,
false);
            var box = BABYLON.Mesh.CreateBox("box", 3.0, scene);
            box.position = new BABYLON.Vector3(-5, 0, 0);
            var cylinder = BABYLON.Mesh.CreateCylinder("cylinder", 3, 3, 3, 6,
1, scene, false);
            cylinder.position=new BABYLON.Vector3(5, 0, 0);
            return scene;
     };
     var scene = createScene();
     engine.runRenderLoop(function(){
            scene.render();
     });
</script>
</body>
</html>
```


tutorialspoint
SIMPLYEASYLEARNING

Output

Upon execution, the above code will generate the following output:



The shapes will move as per the direction you move the cursor; the same is done using the attachcontrol of the camera to the scene.

```
scene.activeCamera.attachControl(canvas);
```

Let us now discuss each shape in detail.

Here is the summary of all the shapes and the syntax:

S.No	Shape	Syntax
1.	**Box**	var box = BABYLON.Mesh.CreateBox("box", 6.0, scene, false, BABYLON.Mesh.DEFAULTSIDE);
2.	**Sphere**	var sphere = BABYLON.Mesh.CreateSphere("sphere", 10.0, 10.0, scene, false, BABYLON.Mesh.DEFAULTSIDE);
3.	**Plane**	var plane = BABYLON.Mesh.CreatePlane("plane", 10.0, scene, false, BABYLON.Mesh.DEFAULTSIDE);
4.	**Disc**	var disc = BABYLON.Mesh.CreateDisc("disc", 5, 30, scene, false, BABYLON.Mesh.DEFAULTSIDE);
5.	**Cylinder**	var cylinder = BABYLON.Mesh.CreateCylinder("cylinder", 3, 3, 3, 6, 1, scene, false, BABYLON.Mesh.DEFAULTSIDE);
6.	**Torus**	var torus = BABYLON.Mesh.CreateTorus("torus", 5, 1, 10, scene, false, BABYLON.Mesh.DEFAULTSIDE);

tutorialspoint
SIMPLYEASYLEARNING

7.	**Knot**	var knot = BABYLON.Mesh.CreateTorusKnot("knot", 2, 0.5, 128, 64, 2, 3, scene, false, BABYLON.Mesh.DEFAULTSIDE);
8.	**Line Mesh**	var lines = BABYLON.Mesh.CreateLines("lines", [new BABYLON.Vector3(-10, 0, 0), new BABYLON.Vector3(10, 0, 0), new BABYLON.Vector3(0, 0, -10), new BABYLON.Vector3(0, 0, 10)], scene);
9.	**Dashes Lines**	var dashedlines = BABYLON.Mesh.CreateDashedLines("dashedLines", [v1, v2, ... vn], dashSize, gapSize, dashNb, scene);
10.	**Ribbon**	var ribbon = BABYLON.Mesh.CreateRibbon("ribbon", [path1, path2, ..., pathn], false, false, 0, scene, false, BABYLON.Mesh.DEFAULTSIDE);
11.	**Tube**	var tube = BABYLON.Mesh.CreateTube("tube", [V1, V2, ..., Vn], radius, tesselation, radiusFunction, cap, scene, false, BABYLON.Mesh.DEFAULTSIDE);
12.	**Ground**	var ground = BABYLON.Mesh.CreateGround("ground", 6, 6, 2, scene);
13.	**GroundFromHeight Map**	var ground = BABYLON.Mesh.CreateGroundFromHeightMap("ground", "heightmap.jpg", 200, 200, 250, 0, 10, scene, false, successCallback);

14.	**Tiled Ground**	var precision = {
		"w" : 2,
		"h" : 2
		};
		var subdivisions = {
		'h' : 8,
		'w' : 8
		};
		var tiledGround = BABYLON.Mesh.CreateTiledGround("Tiled Ground", -3, -3, 3, 3, subdivisions, precision, scene, false);

tutorialspoint
SIMPLYEASYLEARNING

Basic Element - Box

In this section, we will learn how to add box to the scene we created.

To create box, following is the syntax.

Syntax

```
var box = BABYLON.Mesh.CreateBox("box", 6.0, scene, false,
BABYLON.Mesh.DEFAULTSIDE);
```

Parameters

The following are the different parameters to add box:

- **Name:** The Name to be given to the box; for example, "box".
- **Size of the box:** The size of the box.
- **Scene:** The scene is where the box will be attached.
- **Boolean value:** False is the default value.
- **BABYLON.Mesh.DEFAULTSIDE:** This is used for orientation.

The last 2 parameters are optional.

Demo - Box

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>BabylonJs - Basic Element-Creating Scene</title>
     <script src="babylon.js"></script>
<style>
        canvas {width: 100%; height: 100%;}
</style>
</head>
<body>
<canvas id="renderCanvas"></canvas>
<script type="text/javascript">
     var canvas = document.getElementById("renderCanvas");
     var engine = new BABYLON.Engine(canvas, true);
     var createScene  = function() {
            var scene = new BABYLON.Scene(engine);
            scene.clearColor = new BABYLON.Color3(0, 1, 0);
```

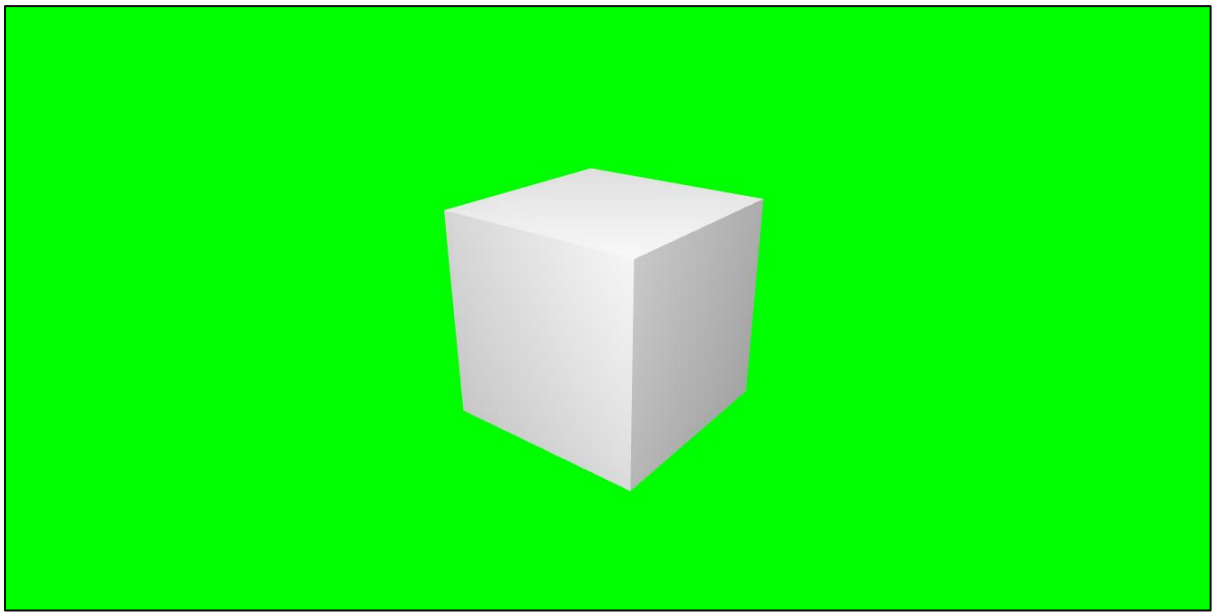

tutorialspoint
SIMPLYEASYLEARNING

```
            var camera = new BABYLON.ArcRotateCamera("Camera", 1, 0.8, 10, new
BABYLON.Vector3(0, 0, 0), scene);
            camera.attachControl(canvas, true);
            var light = new BABYLON.HemisphericLight("light1", new
BABYLON.Vector3(0, 1, 0), scene);
            light.intensity = 0.7;
            var pl = new BABYLON.PointLight("pl", BABYLON.Vector3.Zero(),
scene);
            pl.diffuse = new BABYLON.Color3(1, 1, 1);
            pl.specular = new BABYLON.Color3(1, 1, 1);
            pl.intensity = 0.8;
                  var box = BABYLON.Mesh.CreateBox("box", '3', scene);
            scene.registerBeforeRender(function() {
                  pl.position = camera.position;
            });

            return scene;
      };
      var scene = createScene();
      engine.runRenderLoop(function(){
            scene.render();
      });
</script>
</body>
</html>
```


tutorialspoint
SIMPLYEASYLEARNING

Output

Upon execution, the above code will generate the following output:



Size refers to the height of the box on all sides. A size of 100 will basically be a box occupying full screen. The color given to the background scene is green. We are using the camera and light effect to move the screen onmouse cursor. This helps in the light effect too.

Basic Element - Sphere

In this section, we will learn how to add Sphere to the scene we created.

Syntax

```
var sphere = BABYLON.Mesh.CreateSphere("sphere", 10.0, 10.0, scene, false,
BABYLON.Mesh.DEFAULTSIDE);
```

Parameters

following parameters to add Sphere:

- **Name**: This is the Name of the sphere.
- **Segment**s: This shows the number of segments.
- **Size**: This is the size of the sphere.
- **Scene**: This is the scene to be attached.
- **Boolean**: This is updatable if the mesh needs to be modified later.
- **BABYLON.Mesh.DEFAULTSIDE**:This is the optional side orientation.

tutorialspoint
SIMPLYEASYLEARNING

The last two parameters are optional.

We have already seen an example of sphere while creating the scene. Let us now glance through the creation of sphere again.

Demo - Sphere

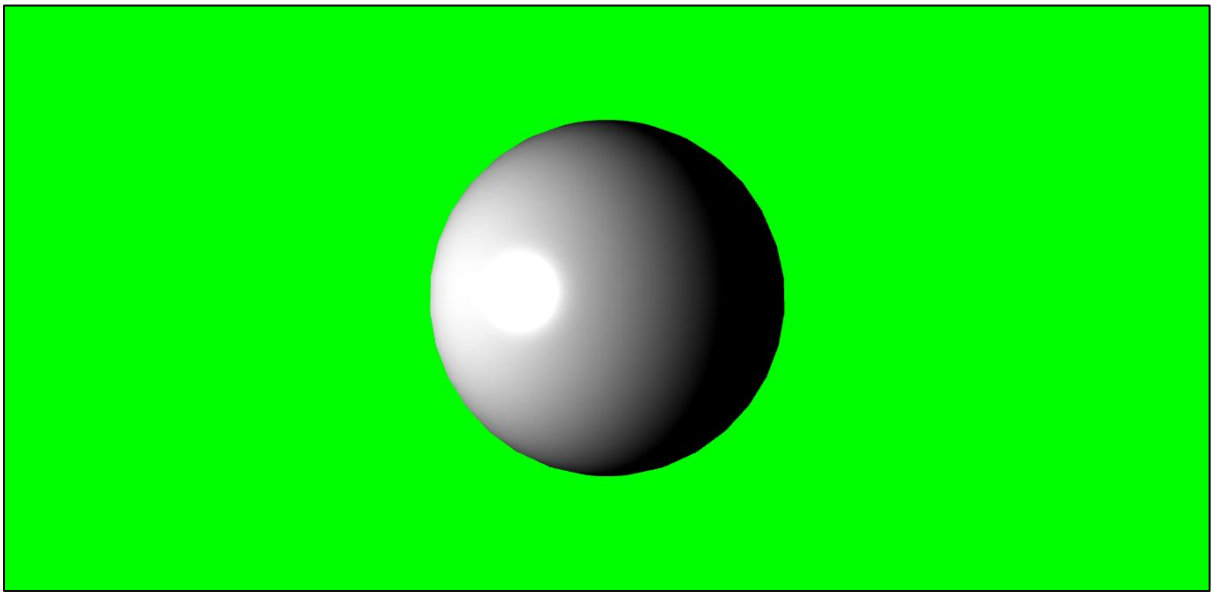
```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>MDN Games: Babylon.js demo - shapes</title>
     <script src="babylon.js"></script>
<style>
        html,body,canvas { margin: 0; padding: 0; width: 100%; height: 100%;
font-size: 0; }
</style>
</head>
<body>
<canvas id="renderCanvas"></canvas>
<script type="text/javascript">
    var canvas = document.getElementById("renderCanvas");
    var engine = new BABYLON.Engine(canvas, true);
     var createScene  = function() {
            var scene = new BABYLON.Scene(engine);
            scene.clearColor = new BABYLON.Color3(0, 1, 0);
            var camera = new BABYLON.ArcRotateCamera("Camera", 1, 0.8, 10, new
BABYLON.Vector3(0, 0, 0), scene);
            var light0 = new BABYLON.PointLight("Omni", new BABYLON.Vector3(0,
0, 10), scene);
            var origin = BABYLON.Mesh.CreateSphere("origin", 15, 5.0, scene);
            scene.activeCamera.attachControl(canvas);
          return scene;
     };
     var scene = createScene();
     engine.runRenderLoop(function(){
            scene.render();
     });
</script>
</body>
```


tutorialspoint
SIMPLYEASYLEARNING

```
</html>
```

Output

Upon execution we get following output:



Sphere can also be created using the meshbuilder.

Syntax

Following is the syntax to create a sphere using the meshbuilder:

```
var sphere = BABYLON.MeshBuilder.CreateSphere("sphere", {diameter: 2,
diameterX: 3}, scene);
```

The Properties for a Sphere

Consider the following properties for a sphere. These properties are optional.

- **Segments:**The default value is 32.It is for the horizontal segments.
- **Diameter:**This is the diameter of the sphere, the default value of which is 1.
- **DiameterX:**The diameter on X-axis overwrites the diameter property. If not specified, it uses the diameter property.
- **DiameterY:** The diameter on Y-axis overwrites the diameter property. If not specified, it uses the diameter property.
- **DiameterZ:**The diameter of Z-axis, overwrites the diameter property. If not specified, it uses the diameter property.
- **Arc:**This is the ratio of the circumference (latitude) between 0 and 1.
- **Slice:**This is the ratio of the height (longitude) between 0 and 1.

tutorialspoint
SIMPLYEASYLEARNING

- **Boolean:**The Boolean value is true if the mesh is updatable. By default, it is false.

- **SideOrientation:**This refers to the side orientation.

tutorialspoint
SIMPLYEASYLEARNING

End of ebook preview

If you liked what you saw…

Buy it from our store @ **https://store.tutorialspoint.com**

tutorialspoint
SIMPLYEASYLEARNING