**Getting Started - Chapter 1 - First Scene**

**@@启程-第一章-第一个场景@@**

**Table Of Contents**

**@@内容列表@@**

[First Scene and Model 第一个场景与模型](https://doc.babylonjs.com/start/chap1/first_scene" \l "first-scene-and-model)

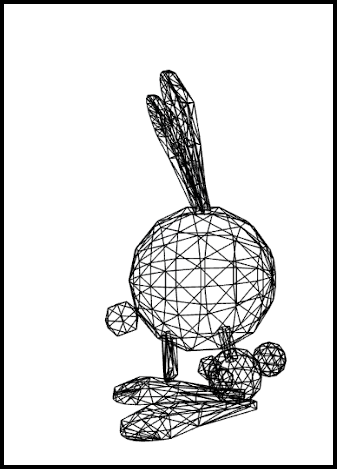
[Say Hello to Your First World 对你的第一个世界说你好](https://doc.babylonjs.com/start/chap1/first_scene" \l "say-hello-to-your-first-world)

**First Scene and Model**

**@@第一个场景与模型@@**

Whether you are creating a whole world or just placing one model into a web page you need a [scene](https://doc.babylonjs.com/divingDeeper/scene) to contain the world or model, a [camera](https://doc.babylonjs.com/divingDeeper/cameras) to view it, a [light](https://doc.babylonjs.com/divingDeeper/lights) to illuminate it and, of course, at least one viewable object as a model. All models, whether just a box or a complex character, are made from a [mesh](https://doc.babylonjs.com/divingDeeper/mesh) of triangles or facets.

@@无论你是想建立一个完整的世界还是只是想在网页中放置一个模型，你都需要建立一个“场景”来容纳这个世界或者模型，建立一个“相机”来观察它，建立一个“光源”来照亮它，以及，当然，至少要有一个可见的物体来作为模型。所有的模型，无论只是一个简单的盒子或者是一个复杂的角色，都是由以三角形或小平面组成的网格来构成的。@@



Wireframe View Showing Mesh Triangles

@@显示网格三角形的线框视图@@

A large number of meshes can be created directly within Babylon.js using code, or, as you will shortly see, imported as models from meshes created with other software. Let us start simply with a box.

@@在Babylon.js中可以直接通过代码建立很多种网格，或者，如你将看到的，以模型的方式导入其他软件创造的网格。让我们从一个简单的盒子网格开始。@@

**Say Hello to Your First World**

**@@向你的第一个世界说你好@@**

All projects using the Babylon.js Engine need a scene with a camera and a light added. Then we can create our box.

@@所有使用Babylon.js引擎的工程都需要一个场景，场景中要添加一个相机和一个光源。然后我们就可以建立我们的盒子。@@

const scene = new BABYLON.Scene(engine);

const camera = new BABYLON.ArcRotateCamera("camera", -Math.PI / 2, Math.PI / 2.5, 3, new BABYLON.Vector3(0, 0, 0), scene);

camera.attachControl(canvas, true);

const light = new BABYLON.HemisphericLight("light", new BABYLON.Vector3(0, 1, 0), scene);

const box = BABYLON.MeshBuilder.CreateBox("box", {}, scene);

Like most meshes created with MeshBuilder the box is created positioned with its center at the origin and needs three parameters. These are a name, *a string*, options, *a JavaScript object*, and a scene. By leaving the options as an empty object *{}* with no properties the box defaults to one of unit size for its width, height and depth.

@@与绝大部分通过“网格建造器”建立的网格一样，盒子网格建立时的中心位于原点，另外构造函数需要三个参数，它们分别是名字-一个字符串、选项-一个JavaScript对象、场景对象。如果把选项参数设为没有属性的空对象{}，则这个盒子将默认为1单位尺寸的宽度、高度、厚度@@

To be usable in a playground we need to place these within a function called **createScene** which has to return a scene. The playground app takes care of the rest.

@@要在训练场里使用这些代码，我们需要把它们放在一个叫做“**创建场景**”的方法中，这个方法将返回一个场景对象，而我们的训练场应用将处理其余的工作。@@

const createScene = () => {

const scene = new BABYLON.Scene(engine);

const camera = new BABYLON.ArcRotateCamera("camera", -Math.PI / 2, Math.PI / 2.5, 3, new BABYLON.Vector3(0, 0, 0));

camera.attachControl(canvas, true);

const light = new BABYLON.HemisphericLight("light", new BABYLON.Vector3(0, 1, 0));

const box = BABYLON.MeshBuilder.CreateBox("box", {});

return scene;

}

Since at this point there is only one scene you may notice that this parameter can be dropped from the camera, light and box as the default is for them to be placed in the current scene.

@@因为现在只有一个场景对象，你可以省略掉相机、光源和盒子网格构造函数中的场景参数，它们将默认放置在当前的场景中。@@

Getting Started First Scene Playground

@@开始查看第一个训练场场景@@

https://playground.babylonjs.com/#KBS9I5



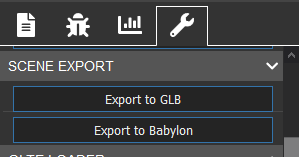
Having created our box we can save, or export, the scene from within the playground by selecting the *Inspector*

@@建立完毕我们的盒子网格之后，我们可以在训练场中保存或者导出这个场景：首先点击“检查”按钮@@



followed by *Tools* and choose which type to export, the *.babylon* format or the *GLB* format.

@@向下拉“工具”选项卡，然后选择导出类型——.babylon格式或者GLB格式@@



Now we have a file we can use it to demonstrate how to view it in a web page.

@@现在我们拥有了一个模型文件，我们可以用它来展示如何在网页中观察模型。@@

（译者注：在自己的项目中开启调试工具的方法：在建立场景对象后执行scene.debugLayer.show();）