**Getting Started - Chapter 1 - Working with Models**

**@@启程-第一章-操作模型@@**

**Table Of Contents 内容列表**

[Importing a Scene or Model 导入一个场景或模型](https://doc.babylonjs.com/start/chap1/first_import" \l "importing-a-scene-or-model)

[Moving On 更进一步](https://doc.babylonjs.com/start/chap1/first_import" \l "moving-on)

[Warning 警告](https://doc.babylonjs.com/start/chap1/first_import" \l "warning)

The playgrounds on this page contain, for example, houses which are be positioned and rotated. Once we have introduced you to the basics of importing and putting your project on a web page we will give you details on how to do this with Babylon.js code.

@@这个页面中的训练场示例中的房子模型被进行了位置和姿态调整。在我们向你介绍完有关向网页中导入3D内容的基础知识后，我们将告诉你如何通过Babylon.js的代码来进行这些定位和旋转操作。@@

**Importing a Scene or Model**

**@@导入一个场景或模型@@**

When you put a model into a scene you are, in fact, loading it into a browser. As you will already know when you load anything into a website it is asynchronous. Before you can do anything with your models you need to know they have loaded. You can do this using the *ImportMeshAsync* method of the *SceneLoader*, which has the form

@@当你向一个场景中添加模型时，你事实上正在通过网络把它加载到浏览器。正如你所知，从一个网站加载任何资源的过程都是异步的。在你可以对你的模型做任何操作之前，你必须先确保它们已经加载完毕。你可以使用“*SceneLoader*”的“*ImportMeshAsync* ”方法，以如下格式：@@

BABYLON.SceneLoader.ImportMeshAsync(model name, folder path, file name, scene);

The scene parameter is optional and will default to the current scene. The first parameter can be of three types depending whether you want to load all the models, just one model or a list of models.

@@其中scene参数是可选的，并且将默认为当前场景。第一个参数可能是三种类型，这取决于你想加载文件中所有的模型，还是只加载一个模型，还是加载一个模型列表。（译者注：不论加载文件中的几个模型，整个文件都会被下载到浏览器中）@@

BABYLON.SceneLoader.ImportMeshAsync("", "/relative path/", "myFile"); //empty string all meshes 空字符串表示导入所有网格

BABYLON.SceneLoader.ImportMeshAsync("model1", "/relative path/", "myFile"); //Name of model for one model 以模型的名字做参数表示导入一个模型

BABYLON.SceneLoader.ImportMeshAsync(["model1", "model2"], "/relative path/", "myFile"); //Array of model names 模型名称数组

All of the above will only load the models and you will not be able to manipulate them in any way. You have set up a Promise object but not given any way to act after the Promise is fulfilled and it provides a result. Examples of this are in the following two playground which **only** import the named models.

@@上面所有的方法都只是单纯的加载了模型而已 ，你没法对这些模型做任何操作（没有回调）。前面的代码其实是设置了一个没有响应的Promise对象，而这个Promise对象的result就是加载的模型。下面的两个训练场演示了如何使用这个result，它们都只导入了提名的模型：@@

Loading Your First ModelLoading Multiple Models at Once

加载你的第一个模型/一次性加载多个模型

[https://playground.babylonjs.com/#YNEAUL#11](https://playground.babylonjs.com/#YNEAUL)

https://playground.babylonjs.com/#YNEAUL#12

To then act on the result we follow the Promise with the *then* method to call a function with the *result* of the *Promise*. The *result* is an object containing, among others, the property *meshes* which contains all the loaded models. We can use this array, or their names, to manipulate each mesh.

@@接下来处理result，我们用then方法延续Promise，然后使用Promise的result作为参数调用一个回调方法。这里的result对象是一个容器对象，它的“meshes”属性包含着所有加载的模型的网格。我们可以使用这个数组，或使用它们的名字（在scene中查找，其前提是模型使用者知道文件里的网格名字且网格名不重复）来操作每个网格。@@

BABYLON.SceneLoader.ImportMeshAsync("", "/relative path/", "myFile").then((result) => {

result.meshes[1].position.x = 20;

const myMesh\_1 = scene.getMeshByName("myMesh\_1");

myMesh1.rotation.y = Math.PI / 2;

});

This playground imports all the models and their positions are changed.

@@这个训练场导入了所有的模型并且修改了它们的位置。@@

Modifying Models After Load

加载后修改模型

[https://playground.babylonjs.com/#YNEAUL#13](https://playground.babylonjs.com/#YNEAUL)

译者注：Promise是ES6处理异步的新语法，“加载后修改模型”的ES5写法为：

BABYLON.SceneLoader.ImportMesh("", "https://assets.babylonjs.com/meshes/", "both\_houses\_scene.babylon"

    ,scene,function(newMeshes, particleSystems, skeletons){

        const house1 = scene.getMeshByName("detached\_house");

        house1.position.y = 2;

        const house2 = newMeshes[2];

        house2.position.y = 1;

    });

（https://playground.babylonjs.com/#YNEAUL#344）

事实上，使用WebGL技术就意味着要使用较新的浏览器，所以一般可以放心使用新API。

**Moving On**

**@@进一步@@**

Having a working scene in the playground is one thing but you will want your game or app to work on your own website. No we will give you an HTML template to do just this.

@@在训练场中建立一个可工作的场景是一回事，但你可能更想让你的游戏或者应用运行在你自己的网站上。我们将为你提供一个HTML模板来做这件事。@@

**Warning**

**@@警告@@**

An obvious statement - different file types export models differently.

@@一个明显的事实——不同的模型文件类型导出模型的方式不同。@@

A less obvious statement - different file types may be changed when importing into Babylon.js.

@@一个没那么明显的事实——不同的文件类型在导入到Babylon.js时可能被施加不同的修改。@@

You need to be aware of how the type you are using affects the outcome. It is not appropriate at this stage to go into detail but the following examples indicate why this is important.

@@你需要清楚你选用的文件类型将如何影响其输出。在这段教程里深入研究细节并不合适，但下面的例子将展示为什么这一点很重要。@@

1. Some software saves all meshes with a rotationQuaternion set and you cannot then use the *rotation* method unless you first add

@@1、一些软件使用旋转四元数保存所有网格的姿态，因此在这种情况下你无法使用旋转方法，除非你先添加：@@

myMesh.rotationQuaternion = null; //Any version of Babylon.js 任何Babylon.js版本

myMesh.rotation = BABYLON.Vector3(); //babylon.js versions > 4.00 大于4.00的版本

1. The following two types were exported from exactly the same scene and imported into Babylon.js.

@@2、以下两种类型的文件都是从同一个场景导出的，然后再导入到Babylon.js中。@@

**.babylon** A model is stored as one mesh, i.e. each house body and roof forms one house.

@@.babylon类型中每个模型都被保存为一个网格，比如每个“房体”和房顶组成一个房子。（这其实是导出方法决定的，事实上Babylon.js也可以用submesh的组织形式导出导入.babylon文件）@@

detached\_house

ground

semi\_house

**.glb** A \_root\_ node is added to hold all the models and model parts are stored as sub-meshes.

@@.glb中一个根节点被用来总领所有模型，然后所有模型组件被以“子网格”的形式存储@@

\_root\_

detached\_house

detached\_house primitive0

detached\_house primitive1

ground

semi\_house

semi\_house primitive0

semi\_house primitive1

（译者注：另外需要注意的是，.babylon格式导出的只是场景的文本描述，模型的二进制内容比如纹理则需要另外下载并放置在合适的目录中，而.glb格式则是文本和二进制内容混合文件，一个文件即可包含模型的所有数据。）