**Contents**

[NOTES: 2](#_Toc130004709)

[Developing Immersive Applications: Quick run-through on extending XRAuthor 3](#_Toc130004710)

[Updating XRAuthor 3](#_Toc130004711)

[Cleaning up project files 4](#_Toc130004712)

[Extending XRAuthor from the Standalone Web App 5](#_Toc130004713)

[App 5](#_Toc130004714)

[Possible Errors 10](#_Toc130004715)

[Extension 11](#_Toc130004716)

[Load Assets 15](#_Toc130004717)

# NOTES:

- If you haven’t done Part 1 yet, here’s the link to [Part 1](IPAsetup_part1.docx)

- If you haven’t done Part 2 yet, here’s the link to [Part 2](IPAsetup_part2.docx)

- If you haven’t done Part 3 yet, here’s the link to [Part 3](IPAsetup_part3.docx)

**TOPICS COVERED:**

* Component Classes
* Creating a ground plane
* Moving sphere around
* Observables
* Mouse/Keyboard inputs
* Collision
* Coroutines
* Locomotion (Teleportation/Controller/Walking)
* HandTracking (Using Oculus Quest 2 HMD)
* Bimanual interaction (Using Oculus Quest 2 controllers)
* Gizmo (Sadly, it is not working on my side ☹)
* Natural Grabbing

**FINAL CODE CHANGES:**

Skip to the final code changes:

[Final Code Changes](#_Final_Code_Changes)

Skip to the individual file changes:

[webpack.config.js](#_webpack.json)

[tsconfig.json](#_tsconfig.json)

[app.ts](#_app.ts)

[components\meshes\index.ts](#_componentsmeshesindex.ts)

[components\meshes\text-plane.ts](#_componentsmeshestext-plane.ts)

[components\meshes\hello-mesh.ts](#_componentsmesheshello-mesh.ts)

# Developing Immersive Applications: Quick run-through on extending XRAuthor

<https://www.youtube.com/watch?v=oze68OIyZVM>

## Updating XRAuthor

Note: if you did not setup XRAuthor previously, you can follow this link to [setup XRAuthor](https://hub.docker.com/r/immersification/xrauthor).

If you already have XRAuthor in your device, follow the steps below to update XRAuthor:

Open powershell and cd to the path of your xrauthor-uploads folder

Enter the following commands:

docker stop xrauthor

docker rm xrauthor

docker pull immersification/xrauthor

docker run -dp 3000:3000 --mount type=bind,source="$(pwd)",target=/app/uploads --name xrauthor immersification/xrauthor

Text

Description automatically generated

## Cleaning up project files

Enter the following commands:

cd src

rm \*.d.ts

A picture containing text

Description automatically generated

You should see the following files removed from your project:

Before:

Graphical user interface, text, application

Description automatically generated

After:

A picture containing text

Description automatically generated

## Extending XRAuthor from the Standalone Web App

### App

Enter the following commands:

cp index.ts init.ts

You should also see a “init.ts” file created.

A picture containing text

Description automatically generated

**init.ts**

Replace with the following codes:

import { Engine } from "babylonjs";

import { App } from "./app";

export function createXRScene(

  canvasID: string,

  // authoringData: {

  //   [dataType: string]: { [key: string]: any };

  // }

  authoringData: any,

) {

  const canvas = <HTMLCanvasElement>document.getElementById(canvasID);

  const engine = new Engine(canvas, true);

  const app = new App(engine, canvas, authoringData);

  const scene = app.createScene();

  const scenePromise = app.createScene();

  scenePromise.then((scene) => {

    engine.runRenderLoop(() => {

      scene.render();

    });

  });

  window.addEventListener("resize", () => {

    engine.resize();

  });

}

**index.ts**

Replace with the following codes:

import { createXRScene } from './init'

createXRScene('renderCanvas', 'hello from app!!! :D')

**webpack.config.js**

Change the port to 9000 instead of 3000, as XRAuthor is using port 3000.



**app.ts**

Add the following codes:

private data: any;

Replace with the following codes:

constructor(engine: Engine, canvas: HTMLCanvasElement, data: any) {

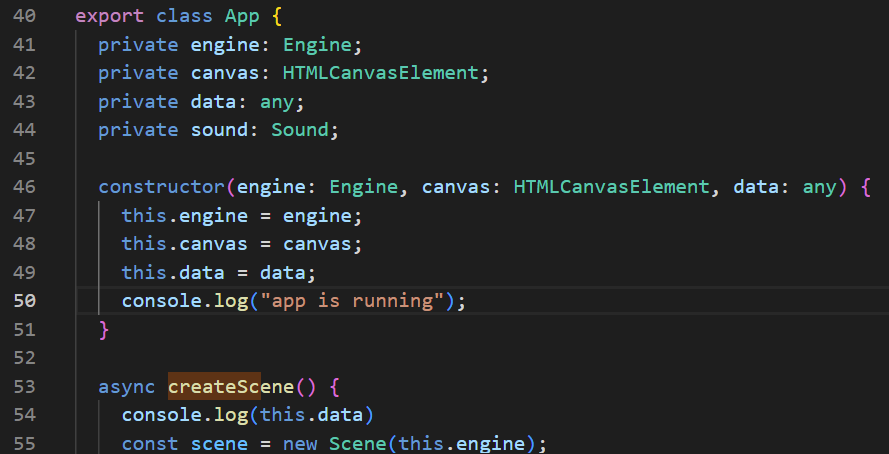
Add the following codes:

this.data = data;

Add the following codes:

console.log(this.data)

Here:



**Check**

You should be able to see the text “hello from app!!! :D” printed on the Console tab.

Graphical user interface, text, email

Description automatically generated

### **Possible Errors**

If you have met the following error: Cannot find global type 'IterableIterator'.ts  
You can try to replace “tsconfig.json” with the following codes:

{

    "compilerOptions": {

        "target": "es5",

        "module": "commonjs",

        "noResolve": false,

        "noImplicitAny": false,

        "removeComments": true,

        "preserveConstEnums": true,

        "sourceMap": true,

        "experimentalDecorators": true,

        "isolatedModules": false,

        "lib": [

            "dom",

            "es2015.promise",

            "es2015.iterable",

            "es5"

        ],

        //"declaration": true,

        "outDir": "./src",

        "types": [

            "babylonjs",

            "babylonjs-gui",

            "babylonjs-materials"

        ]

    },

    "files": [

        "./src/index.ts"

    ]

}

### Extension

Enter the following commands:

touch index-ext.ts

touch webpack.ext.js

You should be able to see “index-ext.ts” and “webpack.ext.js” file created.

**index-ext.ts**

Replace with the following codes:

import { createXRScene } from './init'

window['extension'] = { createXRScene: createXRScene }

**webpack.ext.js**

Replace with the following codes:

const path = require("path");

module.exports =

{

    entry: path.resolve(\_\_dirname, 'src/index-ext.ts'),

    output: {

        filename: 'index.js',

        path: path.resolve(\_\_dirname, 'dist/ext')

    },

    resolve: {

        extensions: ['.ts']

    },

    module: {

        rules: [

            { test: /\.tsx?$/, loader: "ts-loader" }

        ]

    },

    mode: 'production',

}

Enter the following commands:

npm run build-ext

You should be able to see a “ext” folder created under the “dist” folder.

Text

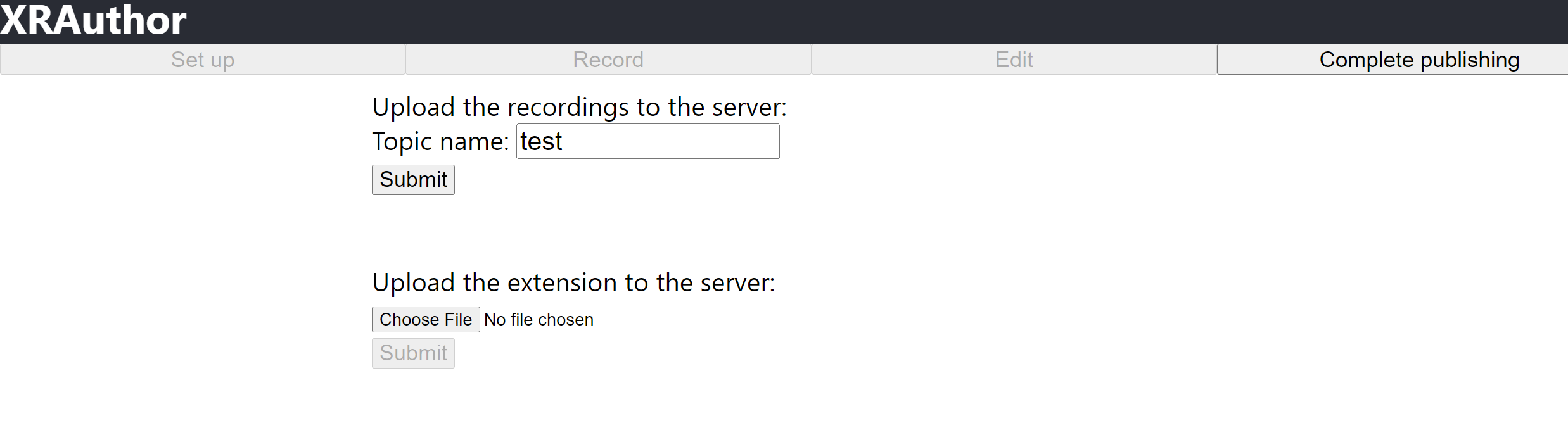
Description automatically generated

**Check**

Go to <http://localhost:3000/author> and record your video.

After you have finish recording your video, click on Complete editing then Publish.

Under “Upload the extension to the server” click on Choose File



Navigate to hello-xr/dist/ext and click upload.

Graphical user interface, text

Description automatically generated

You should see 2 files uploaded.

Graphical user interface, text, application

Description automatically generated

Click on Complete Publishing and click Edit. You should see a new XR Format tab added below your video. Click on the XR Format tab and your VR Web App should be shown.

Graphical user interface

Description automatically generated

## Load Assets

**Webpack.config.js**

Comment out the following codes:

//static: false,

Here:

A screenshot of a computer

Description automatically generated with medium confidence

Navigate to xrauthor-uploads/assets and copy the test folder.

Graphical user interface, text, application

Description automatically generated

Paste the test folder in hello-xr-xrauthor/hello-xr/public/assets

Graphical user interface, text, application, email

Description automatically generated

Enter the following commands:

npm i xrauthor-loader -D

A screenshot of a computer

Description automatically generated with medium confidence

**app.ts**

Replace with the following codes:

private data: AuthoringData;

Replace with the following codes:

constructor(engine: Engine, canvas: HTMLCanvasElement, data: AuthoringData) {

Here:

Text

Description automatically generated

**index.ts**

Replace with the following codes:

import {AuthoringData, loadAuthoringData} from 'xrauthor-loader'

import {createXRScene} from './init'

loadAuthoringData('assets/test').then((data: AuthoringData) => {

    createXRScene('renderCanvas', data)

})

**init.ts**

Replace with the following codes:

authoringData: AuthoringData,

Here:

**Text

Description automatically generated**

**Check**

You should see the AuthoringData shown in your Console tab.

**Graphical user interface, text, application

Description automatically generated**