**Contents**

[NOTES: 2](#_Toc129722180)

[Developing Immersive Applications: Development Primer Live Stream on 17 Jan 2023 3](#_Toc129722181)

[Download WebXR chrome extension 3](#_Toc129722182)

[Downloading and installing npm: 4](#_Toc129722183)

[npm setup 5](#_Toc129722184)

[Configure webpack, typescript and other stuffs 8](#_Toc129722185)

[Run Build + Serve 17](#_Toc129722186)

[Render “HelloXR” text using TS [OPTIONAL] 19](#_Toc129722187)

[Create Scene 20](#_Toc129722188)

[Create Sphere Object 24](#_Toc129722189)

[Text rendering using BabyonJS GUI library 26](#_Toc129722190)

[Adding XR feature 29](#_Toc129722191)

[How to use WebXR API Emulator extension 31](#_Toc129722192)

[Async XR 32](#_Toc129722193)

# NOTES:

If you just want the final result, you can skip those labelled with **[OPTIONAL]**

# Developing Immersive Applications: Development Primer Live Stream on 17 Jan 2023

<https://www.youtube.com/watch?v=JruhVU8MkKQ>

## Download WebXR chrome extension

Graphical user interface, application, Teams

Description automatically generated

Google search and install the **WebXR API Emulator** extension.

Navigate to [How to use **WebXR API Emulator** extension](#_How_to_use) to learn how to use the extension.

## Downloading and installing npm:

<https://docs.npmjs.com/downloading-and-installing-node-js-and-npm>

Graphical user interface, text, application

Description automatically generated

Open cmd

Enter the following commands:

npm install -g npm

## npm setup

A screenshot of a computer

Description automatically generated with medium confidence

Create a folder called “Hello-XR”

Open wsl terminal in vscode

Enter the following commands:

npm init –yes

**package.json** file will be created (shown on the left panel of vscode)

A screenshot of a computer

Description automatically generated with medium confidence

Enter the following commands:

npm install --save babylonjs

**package-lock.json**, **node\_modules**, **babylonjs**, **.package-lock.json** files will be created.

A screenshot of a computer

Description automatically generated with medium confidence

Enter the following commands:

EITHER (the whole chunk of commands)

npm install --save-dev webpack webpack-cli webpack-dev-server html-webpack-plugin typescript ts-loader

OR (1 by 1)

npm install --save-dev webpack

npm install --save-dev webpack-cli

npm install --save-dev webpack-dev-server

npm install --save-dev html-webpack-plugin

npm install --save-dev typescript ts-loader

Lots of weird files will be created in node\_modules.

Navigate to [Text rendering using BabylonJS GUI library](#_Text_rendering_using) to setup the BabylonJS GUI.

## Configure webpack, typescript and other stuffs

A screenshot of a computer

Description automatically generated with medium confidence

Enter the following commands:

touch webpack.config.js tsconfig.json

mkdir src

touch src/index.html src/index.ts

**webpack.config.js**, **tsconfig.json**, **index.html**, **index.ts** files will be created.

**webpack.config.js**

A screenshot of a computer

Description automatically generated with medium confidence

Go to **webpack.config.js** and paste the following codes:

const path = require("path");

const HtmlWebpackPlugin = require('html-webpack-plugin')

module.exports = {

    entry: './src/index.ts',

    output: {

        filename: 'index.js',

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

    },

    resolve: {

        extensions: [".ts", ".js"]

    },

    module: {

        rules: [

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

        ]

    },

    mode: "development",

    devtool: 'inline-source-map',

    devServer: {

        static: false,

        port: 3000,

    },

    plugins: [

        new HtmlWebpackPlugin({

            template: path.resolve(\_\_dirname,'src/index.html')

        })

    ]

};

**tsconfig.json [OPTIONAL]**

A screenshot of a computer

Description automatically generated with medium confidence

Go to **tsconfig.json** and paste 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",

            "es5"

        ],

        "declaration": true,

        "outDir": "./src"

    },

    "files": [

        "./src/index.ts"

    ]

}

**index.html**

A screenshot of a computer

Description automatically generated with medium confidence

Go to **index.html** and paste the following codes:

<!DOCTYPE html>

<html>

    <head>

        <style>

            html,

            body {

                overflow: hidden;

                width: 100%;

                height: 100%;

                margin: 0;

                padding: 0;

                text-align: center;

            }

            #renderCanvas {

                width: 100%;

                height: 100%;

                touch-action: none;

            }

        </style>

        <title>Hello XR</title>

    </head>

    <body>

        <canvas id="renderCanvas"></canvas>

    </body>

</html>

**index.ts [OPTIONAL]**

**A screenshot of a computer

Description automatically generated**

Go to **index.ts** and paste the following codes:

console.log('hello xr')

**package.json**

A screenshot of a computer

Description automatically generated with medium confidence

Go to **package.json** and paste the following codes:

{

  "name": "hello-xr",

  "version": "1.0.0",

  "description": "",

  "main": "index.js",

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1",

    "build": "webpack",

    "serve": "webpack serve",

    "start": "webpack serve --open"

  },

  "keywords": [],

  "author": "",

  "license": "ISC",

  "dependencies": {

    "babylonjs": "^5.44.0",

    "babylonjs-gui": "^5.44.0"

  },

  "devDependencies": {

    "html-webpack-plugin": "^5.5.0",

    "ts-loader": "^9.4.2",

    "typescript": "^4.9.4",

    "webpack": "^5.75.0",

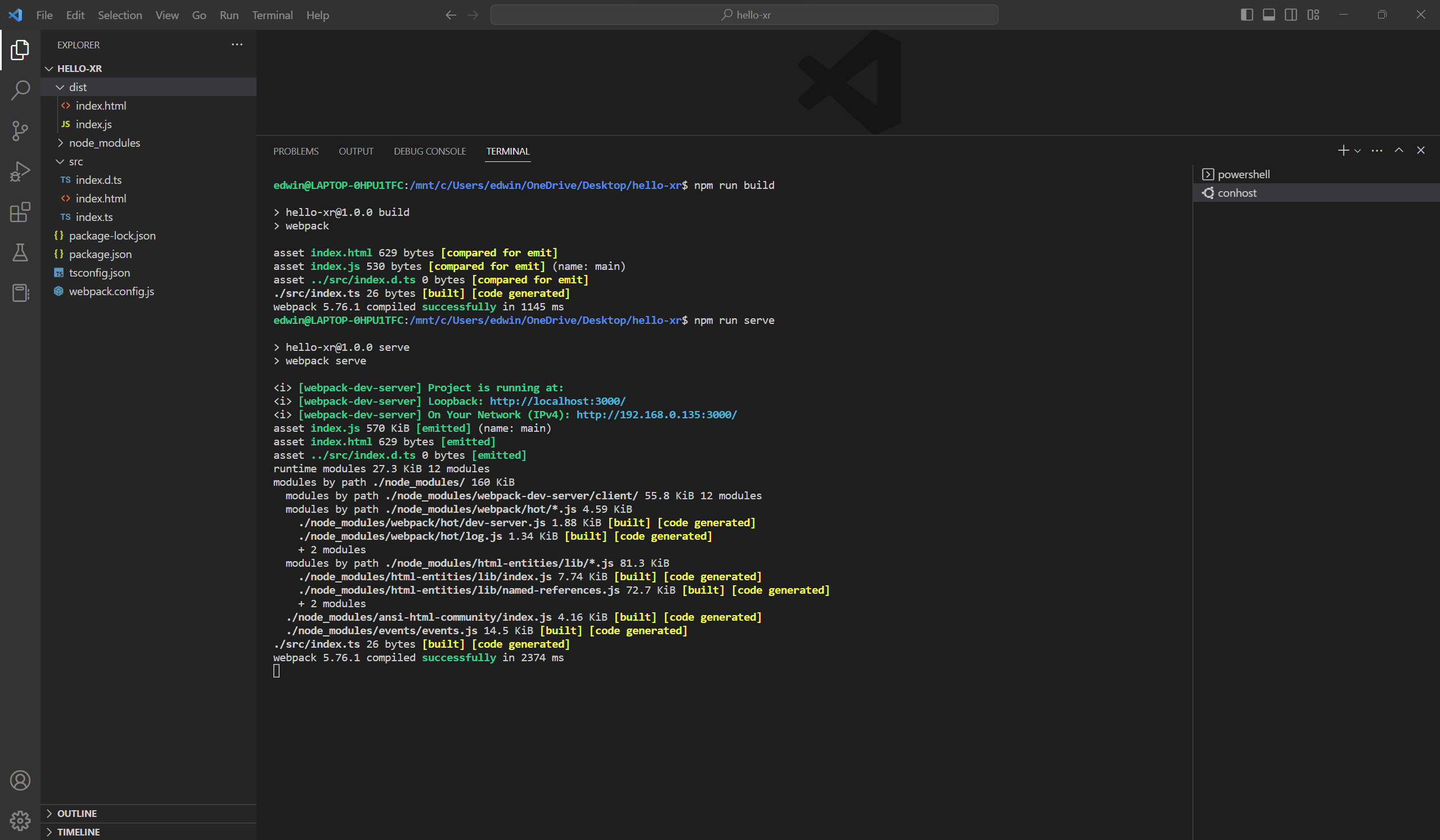
    "webpack-cli": "^5.0.1",

    "webpack-dev-server": "^4.11.1"

  }

}

## Run Build + Serve



Enter the following commands:

npm run build

**dist** folder will be created

npm run serve

**Checking setup [OPTIONAL]**

Graphical user interface, application, Word

Description automatically generated

Go to localhost:3000 in your chrome browser.

Press F12 to go to the Developer tools.

Check that “hello xr” text is there under the Console tab.

***Setup completed!***

## Render “HelloXR” text using TS [OPTIONAL]

Text

Description automatically generated

Go to **index.ts** and replace with the following codes:

const canvas = document.getElementById('renderCanvas') as HTMLCanvasElement

const ctx = canvas.getContext('2d')

ctx.font = '50px Arial'

ctx.fillText('Hello XR', 50, 50)

Graphical user interface, application, Word

Description automatically generated

The “Hello XR” text should appear in localhost:3000

## Create Scene

A screenshot of a computer

Description automatically generated with medium confidence

Enter the following commands:

touch src/app.ts

**app.ts** file will be created

**index.ts [OPTIONAL]**

Text

Description automatically generated

Go to **index.ts** and replace with the following codes:

import { Engine } from "babylonjs"

import { App } from './app'

const canvas = document.getElementById('renderCanvas') as HTMLCanvasElement

const engine = new Engine(canvas, true)

const app =  new App(engine, canvas)

const scene = app.createScene()

engine.runRenderLoop(() =>

    {

        scene.render()

    }

)

**app.ts [OPTIONAL]**

A screenshot of a computer

Description automatically generated

Go to **app.ts** and paste the following codes:

import {Engine, MeshBuilder, Scene} from "babylonjs";

export class App

{

    private engine: Engine

    private canvas: HTMLCanvasElement

    constructor(engine: Engine, canvas: HTMLCanvasElement)

    {

        this.engine = engine

        this.canvas = canvas

        console.log('app is running')

    }

    createScene()

    {

        const scene = new Scene(this.engine)

        scene.createDefaultCameraOrLight()

        return scene

    }

}

**Check [OPTIONAL]**

**Text

Description automatically generated**

Start the web app using the command:

npm run serve

Go to localhost:3000 in your chrome browser.

Graphical user interface, application, Word

Description automatically generated

You should see a purple background with “app is running” under the Console tab.

## Create Sphere Object

**app.ts [OPTIONAL]**

A screenshot of a computer

Description automatically generated

Go to **app.ts** and replace with the following codes:

import {Engine, MeshBuilder, Scene} from "babylonjs";

export class App

{

    private engine: Engine

    private canvas: HTMLCanvasElement

    constructor(engine: Engine, canvas: HTMLCanvasElement)

    {

        this.engine = engine

        this.canvas = canvas

        console.log('app is running')

    }

    createScene()

    {

        const scene = new Scene(this.engine)

        scene.createDefaultCameraOrLight()

        const sphere = MeshBuilder.CreateSphere('sphere', {diameter: 1.3})

        sphere.position.y = 1

        sphere.position.z = 5

        return scene

    }

}

**Check [OPTIONAL]**

Start the web app using the command:

npm run serve

Go to localhost:3000 in your chrome browser.

A picture containing graphical user interface

Description automatically generated

You should see a **Sphere** rendered in the purple background.

## Text rendering using BabylonJS GUI library

A screenshot of a computer

Description automatically generated with medium confidence

Enter the following commands:

npm install --save babylonjs-gui

Go to **tsconfig.json** and replace 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",

            "es5"

        ],

        "declaration": true,

        "outDir": "./src",

        "types": [

            "babylonjs",

            "babylonjs-gui"

        ]

    },

    "files": [

        "./src/index.ts"

    ]

}

Go to **app.ts** and replace with the following codes: **[OPTIONAL]**

import {Engine, MeshBuilder, Scene} from "babylonjs";

import { AdvancedDynamicTexture, TextBlock } from "babylonjs-gui";

export class App

{

    private engine: Engine

    private canvas: HTMLCanvasElement

    constructor(engine: Engine, canvas: HTMLCanvasElement)

    {

        this.engine = engine

        this.canvas = canvas

        console.log('app is running')

    }

    createScene()

    {

        const scene = new Scene(this.engine)

        scene.createDefaultCameraOrLight()

        const sphere = MeshBuilder.CreateSphere('sphere', {diameter: 1.3})

        sphere.position.y = 1

        sphere.position.z = 5

        const helloPlane = MeshBuilder.CreatePlane('hello plane', {size:15})

        helloPlane.position.y = 0

        helloPlane.position.z = 5

        const helloTexture = AdvancedDynamicTexture.CreateForMesh(helloPlane)

        const helloText = new TextBlock('hello')

        helloText.text = 'Hello XR'

        helloText.color = 'purple'

        helloText.fontSize = 50

        helloTexture.addControl(helloText)

        return scene

    }

}

**Check [OPTIONAL]**

Start the web app using the command:

npm run serve

Go to localhost:3000 in your chrome browser.

A picture containing graphical user interface

Description automatically generated

You should see the text **“Hello XR”** rendered below the **Sphere**.

## Adding XR feature

Go to **app.ts** and replace with the following codes: **[OPTIONAL]**

import {Engine, MeshBuilder, Scene} from "babylonjs";

import { AdvancedDynamicTexture, TextBlock } from "babylonjs-gui";

export class App

{

    private engine: Engine

    private canvas: HTMLCanvasElement

    constructor(engine: Engine, canvas: HTMLCanvasElement)

    {

        this.engine = engine

        this.canvas = canvas

        console.log('app is running')

    }

    createScene()

    {

        const scene = new Scene(this.engine)

        scene.createDefaultCameraOrLight()

        const sphere = MeshBuilder.CreateSphere('sphere', {diameter: 1.3})

        sphere.position.y = 1

        sphere.position.z = 5

        const helloPlane = MeshBuilder.CreatePlane('hello plane', {size:15})

        helloPlane.position.y = 0

        helloPlane.position.z = 5

        const helloTexture = AdvancedDynamicTexture.CreateForMesh(helloPlane)

        const helloText = new TextBlock('hello')

        helloText.text = 'Hello XR'

        helloText.color = 'purple'

        helloText.fontSize = 50

        helloTexture.addControl(helloText)

        const xr = scene.createDefaultXRExperienceAsync(

            {

                uiOptions:

                {

                    sessionMode: 'immersive-vr'

//sessionMode: 'immersive-ar'

                }

            }

        );

        return scene

    }

}

**Check [OPTIONAL]**

Start the web app using the command:

npm run serve

Go to localhost:3000 in your chrome browser.

A picture containing graphical user interface

Description automatically generated

You should see the **VR Logo** at the bottom of the screen.

## How to use WebXR API Emulator extension

A picture containing graphical user interface

Description automatically generated

Click on the **>>** in the Developer tools and select **WebXR**

Click on the **VR Goggles Logo** at the bottom right of the screen.

**Graphical user interface, application

Description automatically generated**

You should see the scene splitting in 2.

Play around with the **WebXR** extension.

## Async XR

Go to **app.ts** and replace with the following codes:

import {Engine, MeshBuilder, Scene} from "babylonjs";

import { AdvancedDynamicTexture, TextBlock } from "babylonjs-gui";

export class App

{

    private engine: Engine

    private canvas: HTMLCanvasElement

    constructor(engine: Engine, canvas: HTMLCanvasElement)

    {

        this.engine = engine

        this.canvas = canvas

        console.log('app is running')

    }

    async createScene()

    {

        const scene = new Scene(this.engine)

        scene.createDefaultCameraOrLight()

        const sphere = MeshBuilder.CreateSphere('sphere', {diameter: 1.3})

        sphere.position.y = 1

        sphere.position.z = 5

        const helloPlane = MeshBuilder.CreatePlane('hello plane', {size:15})

        helloPlane.position.y = 0

        helloPlane.position.z = 5

        const helloTexture = AdvancedDynamicTexture.CreateForMesh(helloPlane)

        const helloText = new TextBlock('hello')

        helloText.text = 'Hello XR'

        helloText.color = 'purple'

        helloText.fontSize = 50

        helloTexture.addControl(helloText)

        const xr = await scene.createDefaultXRExperienceAsync(

            {

                uiOptions:

                {

                    sessionMode: 'immersive-vr'

//sessionMode: 'immersive-ar'

                }

            }

        );

        //only for debugging

        //(window as any).xr = xr

        return scene

    }

}

Go to **index.ts** and replace with the following codes:

import { Engine } from "babylonjs"

import { App } from './app'

// console.log('hello xr')

const canvas = document.getElementById('renderCanvas') as HTMLCanvasElement

// const ctx = canvas.getContext('2d')

// ctx.font = '50px Arial'

// ctx.fillText('Hello XR', 50, 50)

const engine = new Engine(canvas, true)

const app =  new App(engine, canvas)

const scene = app.createScene()

const scenePromise = app.createScene()

// engine.runRenderLoop(() =>

//     {

//         scene.render()

//     }

// )

scenePromise.then(scene =>

    {

        engine.runRenderLoop(() =>

            {

                scene.render()

            })

    })

**Check [OPTIONAL]**

Start the web app using the command:

npm run serve

Go to localhost:3000 in your chrome browser.

Graphical user interface

Description automatically generated with low confidence

Everything should be rendered the same as previous steps.