Using the Engine Standalone

It is possible to build applications on the PlayCanvas Engine without using the Editor. Some examples of applications built directly against the Engine are:

- glTF Viewer [GitHub]
- SuperSplat [GitHub]
- ...and, of course, the PlayCanvas Editor itself!

This page guides you in how to get started.

> **NOTE**
>
> Before you begin, ensure you have Node.js installed.

When setting up your project, there are two main options to consider.

Option 1: Build Tool and NPM

This is the recommended set up that should suit most developers.

A build tool can bundle your application into an optimized package that can run on a wide range of browsers. There are many build tools such as webpack, Rollup and esbuild, and PlayCanvas will work with all of them. Here, we will use Vite, a popular build tool that aims to provide a faster and leaner development experience for modern web projects.

First, select whether you prefer to develop in JavaScript or TypeScript:

JavaScript TypeScript

1. Open a Terminal/Command Prompt, create a folder for your app and `cd` inside it.
2. Install `playcanvas` and `vite`:

```
npm install playcanvas vite --save-dev
```

3. Create an `index.html` and paste this:

index.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <style>
            body { margin: 0; overflow: hidden; }
        </style>
    </head>
    <body>
        <script type="module" src="main.js"></script>
        <canvas id='application'></canvas>
    </body>
</html>
```

4. Create a `main.js` and paste this:

main.js

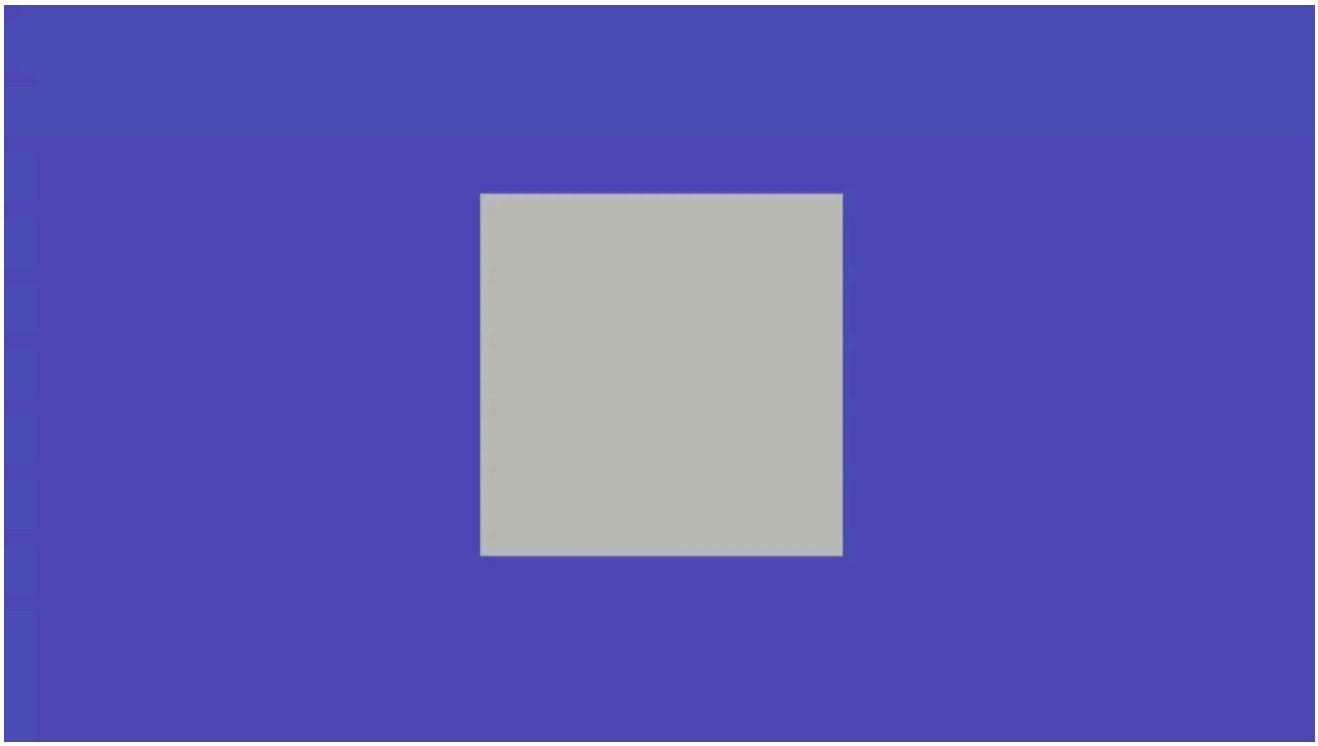
```
import * as pc from 'playcanvas';

// create an application
const canvas = document.getElementById('application');
const app = new pc.Application(canvas);
app.setCanvasResolution(pc.RESOLUTION_AUTO);
app.setCanvasFillMode(pc.FILLMODE_FILL_WINDOW);
app.start();

// create a camera
const camera = new pc.Entity();
camera.addComponent('camera', {
    clearColor: new pc.Color(0.3, 0.3, 0.7)
});
camera.setPosition(0, 0, 3);
app.root.addChild(camera);

// create a light
const light = new pc.Entity();
```

```
light.addComponent('light');
light.setEulerAngles(45, 45, 0);
app.root.addChild(light);

// create a box
const box = new pc.Entity();
box.addComponent('model', {
    type: 'box'
});
app.root.addChild(box);

// rotate the box
app.on('update', (dt) => box.rotate(10 * dt, 20 * dt, 30 * dt));
```

5. Run the Vite development server:

```
npx vite
```

This starts a server at `http://localhost:5173`. Visit this URL in your browser and you will see the following:



> **TIP**
>
> Every time you save your source files, the tab will reload automatically.

Option 2: Import Map and CDN

An import map can resolve module specifiers in JavaScript modules. Consider this import statement:

```
import * as pc from 'playcanvas';
```

An import map can resolve `playcanvas` to a CDN-hosted build of the engine that can be dynamically loaded by the browser. This means that we can skip the build step described in Option 1.

First, select whether you prefer to develop in JavaScript or TypeScript:

JavaScript **TypeScript**

1. Open a Terminal/Command Prompt, create a folder for your app and `cd` inside it.
2. Create an `index.html` and paste this:

index.html

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="utf-8">
        <style>
            body { margin: 0; overflow: hidden; }
        </style>
        <script type="importmap">
        {
            "imports": {
                "playcanvas":
"https://cdn.jsdelivr.net/npm/playcanvas"
            }
        }
        </script>
    </head>
    <body>
        <script type="module" src="main.js"></script>
        <canvas id='application'></canvas>
```

```
    </body>
</html>
```

3. Create a `main.js` and paste this:

main.js

```
import * as pc from 'playcanvas';

// create an application
const canvas = document.getElementById('application');
const app = new pc.Application(canvas);
app.setCanvasResolution(pc.RESOLUTION_AUTO);
app.setCanvasFillMode(pc.FILLMODE_FILL_WINDOW);
app.start();

// create a camera
const camera = new pc.Entity();
camera.addComponent('camera', {
    clearColor: new pc.Color(0.3, 0.3, 0.7)
});
camera.setPosition(0, 0, 3);
app.root.addChild(camera);

// create a light
const light = new pc.Entity();
light.addComponent('light');
light.setEulerAngles(45, 45, 0);
app.root.addChild(light);

// create a box
const box = new pc.Entity();
box.addComponent('model', {
    type: 'box'
});
app.root.addChild(box);

// rotate the box
app.on('update', (dt) => box.rotate(10 * dt, 20 * dt, 30 * dt));
```

4. Run `serve`:

```
npx serve
```

This starts a server at `http://localhost:3000`. Visit this URL in your browser and you will see the following:

Edit this page