PHASER TUTORIALS

Create Native HTML5 Games with Phaser and Capacitor

BY JOSH MORONY | LAST UPDATED: JULY 10, 2018

BEGINNER CAPACITOR IONIC PHASER

Follow **Josh Morony** on

Building HTML5 games with Phaser has been somewhat of a hobby of mine over the past few years, and I've even written a few tutorials about Phaser with a focus on developing HTML5 games for mobile. However, despite having created a few mostly finished games most of them never actually see the light of day.

I do love the idea of creating games with web tech (I love the idea of creating everything with web tech, really). The recent release of Phaser 3, and of Capacitor (*and* a recent obsession with pixel art), has re-sparked my interest in building mobile games with web tech.

Phaser 3 is the next-generation of frameworks for building mobile applications with web tech, but what this article is going to be about is combining that with **Capacitor**. We are going to talk a little bit about what Capacitor is and why we might want to use it in just a second.

In this article, I am going to introduce you to a starter template I've put together for a **Phaser 3** project that uses **TypeScript**, is built using **Webpack**, and includes **Capacitor** for native builds. The starter template has some
set up, so we will end up with a little demo game that looks like this:

Help me show you more relevant content by answering a few questions: Which framework are you most interested in using with Ionic?
Click to answer

What is Capacitor?

Capacitor is a tool that the Ionic team recently released, and the easiest way to think about it is as a substitute for Cordova. It was built with Ionic in mind, but it is not in any way specific to Ionic (you can use it with any web-based code). I don't want to dive too much into the project and its goals here, because I have already covered that [in another blog post](). The main point is that Capacitor allows us to package up web code into a native application, and provides us with access to Native APIs.

The cool thing about Capacitor is that rather than just being a tool for getting your web code running on native devices, it is aiming to be a run time that allows your application to run just about *anywhere*. This means it would be possible to create a game using Phaser that you could run as:

- A standard website
- A Progressive Web Application (with offline support)
- A Native iOS Application
- A Native Android Application
- A Desktop Application (with Electron)

Designing a game that works seamlessly across multiple different device sizes/resolutions isn't an easy task, but this tech stack will make that possible and as easy as it possible *could* be. Aside from the lofty goal of building a game that works on every platform from a single codebase, there are other advantages to using Capacitor that haven't been fully addressed in the past:

- Issues with WKWebView and running Phaser games don't seem to be a problem now. Having to rely on UIWebView (a less performant mobile browser) previou ideal for games.
- It is very quick and easy to set up Capacitor in a Phaser project

- The local development workflow with Capacitor is easy to work with (but does require you to have Native SDKs configured on your machine)
- Using Capacitor for native builds is entirely free

Overall, it finally just feels like a solution where the tech stays out of your way and you can just focus on building the game.

Phaser/TypeScript/Webpack/Capacitor Starter Template

As well as Capacitor, I've also come to enjoy working in a TypeScript environment too much to not use it now. Phaser does not use TypeScript by default, and it is a bit tricky to set up, but once it's done it's done and you are free to work with TypeScript as you usually would.

I wanted to set something up that was configured with everything you need by default and handled the build process for you. So, I put together this starter template to do just that. I had help from a lot of sources in creating this template, including the official Phaser documentation and random threads on StackOverflow. These two repositories, in particular, were a great help in deciding on configuration and project structure:

- https://github.com/digitsensitive/phaser3-typescript
- https://github.com/TooManyCaptains/TooManyCaptains

Installing the Starter

In order to install the starter, you should run the following command to clone the repository:

```
git clone https://github.com/joshuamorony/phaser3-typescript-webpack-capacitor.git my
```

Then you will need to make it your working directory:

```
cd my-game
```

and install the dependencies:

```
npm install
```

Once the dependencies have been installed, you will need to initialise your Capacitor project:

```
npx cap init
```

Throughout development, you can use:

```
npm run dev
```

to run your game. When you are ready to build for production, you can run:

```
npm run build
```

This will handle generating the `www` folder that will contain the web code for Capacitor to natively package, and it will also copy the web code into the native projects if you have the appropriate platforms added. To add platforms to your project, just run:

```
npx cap add ios
```

```
npx cap add android
```

```
npx cap add electron
```

Native builds with Capacitor are handled locally using the native tools for the platform (XCode for iOS, Android Studio for Android). You can open your project in the appropriate environment with:

```
npx cap open ios
```

or

```
npx cap open android
```

From there, you can run or build the application as you would any other native mobile application project.

Summary

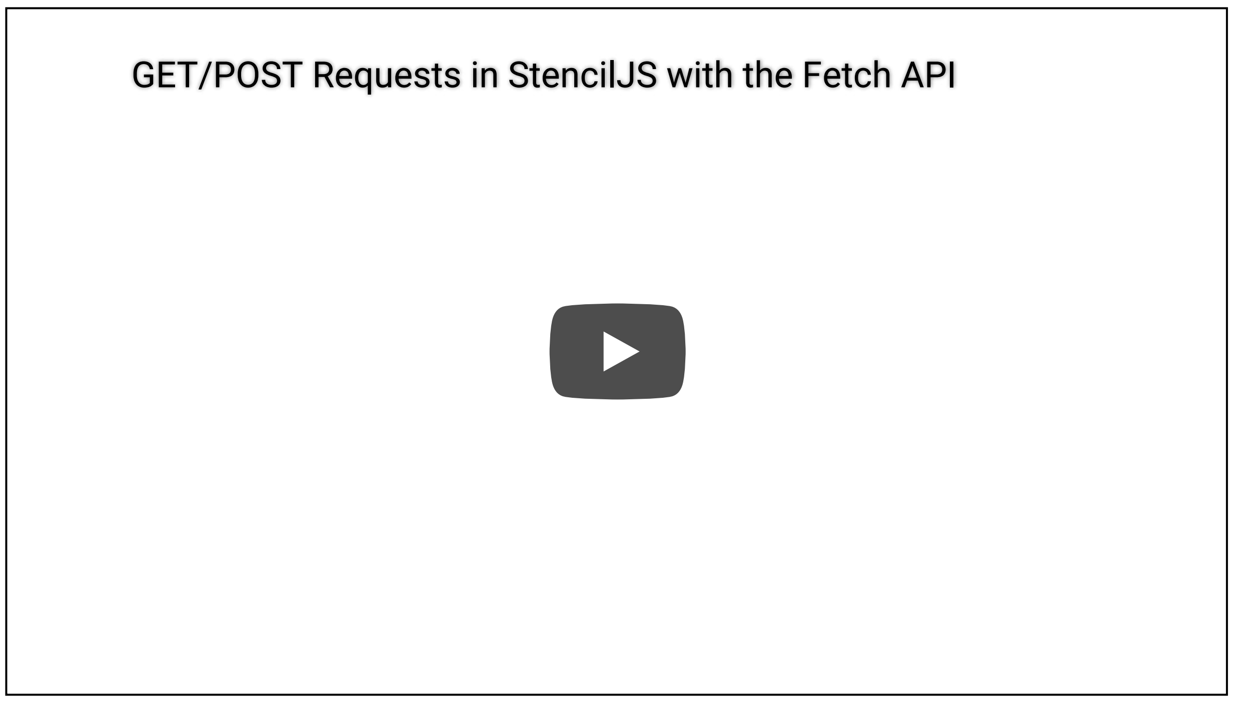
The starter project has some default code supplied that enforces sc
on how you should go about building the application, but these are
preferences. You do not need to follow the supplied structure and a

change it up however you want. I find it useful to separate entities like that "Player" out into their own classes, but that's entirely up to you. Just keep in mind that both Phaser 3 and Capacitor are not officially complete yet, so you may run into some bugs or missing documentation if you use this for a project.

← Stack Based Navigation (Push/Pop) with Ionic Web Components

Using Angular Routing with Ionic 4 →

Check out my latest videos:



GET/POST Requests in StencilJS with the Fetch API

Help me show you more relevant content by answering a few questions: Which framework are you most interested in using with Ionic?

Click to answer