**# Project 0**

**Prerequisite: HTML, CSS, BASICS OF JAVASCRIPT**

The goal of this project is to practice JavaScript and its paradigms by creating a TODO app. This app should be able to add TODOs and track the number of total TODOs as well as the number of unchecked TODOs.

**Instructions**

Inside of **[index.html](/index.html)**, you'll find some starter HTML. You shouldn't need to edit this file at all. Open this file on your computer into any browser to run the project. Make sure that **[script.js](/script.js)** and **[styles.css](/styles.css)** are in the same local directory. With the file open in your browser, you should see a `New TODO` button, which `alert`s when clicked. Your goal will be to get this button to create new TODOs.

Inside **[styles.css](/styles.css)**, you'll find some pre-written CSS for your convenience. You shouldn't need to edit this file at all, but feel free to if desired.

**[script.js](/script.js)** is where most of your work will be done. There is some starter code for you in the file. The `classNames` variable can be used to link any elements you create in JS with the associated CSS class names. The next 3 lines of code are the HTML elements that you'll need to update when creating new TODOs. Lastly, you'll see the `addTodo()` function. This gets executed when creating a new TODO. You should replace the `alert()` call with logic to create new TODOs.

Good luck!

**Challenge!** (Not Required)

If you finish early and are up for a challenge, try adding delete functionality. This should be in the form of a button within each TODO that removes that TODO when clicked.

**# Project 1 - Pomodoro Timer**

**Prerequisite:**

For this project, you'll be implementing a Pomodoro timer. This timer will help people trying to use the **[Pomodoro Technique](**[**https://en.wikipedia.org/wiki/Pomodoro\_Technique**](https://en.wikipedia.org/wiki/Pomodoro_Technique)**)**. It will vibrate to tell you when to take breaks or resume working, based on some determined values. Check out the **[staff solution](#staff-solution)** for a working version.

**Requirements**

- You may not import libraries other than the below:

-`expo`

-`react`

-`react-native`

-`prop-types`

- Timer should display minutes and seconds in text

- Timer should count down seconds until it reaches 00:00

- Phone should buzz when timer reaches 0

- Timers should switch between 25 and 5 minutes

- Timer should be able to start, stop, and reset

The aesthetics of the app is up to you!

**Challenge** (Not Required)

- Allow the user to input any arbitrary time for the timers (e.g. 5 mins of work time and 5 mins of break)

**Getting Started**

First, head to **[this link](<https://docs.expo.io/versions/latest/introduction/installation.html>)** to install Expo. You'll need the XDE for your computer and the mobile client (Expo app) on your phone. If you prefer, you can also install the iOS simulator (Macs only) and/or the Android emulator.

You'll also need Node.js and NPM installed. You can check if you already have them installed by opening a terminal and running `node --version` and `npm --version`. If numbers are printed, you're good to go. If not, **[install them](**[**https://nodejs.org/en/**](https://nodejs.org/en/)**)**. You'll probably want the LTS version (v8.x.x). NPM will be installed automatically when you install node.

After installing those software dependencies, you'll need to install your app's "dependencies" (libraries that are required to run the app, such as `react`, `react-native`, etc.). Fortunately, it's very easy to do! From a terminal, `cd` into this directory and run the command `npm install`. NPM will look at the **[`package.json`](/package.json)** file's `dependencies` key and install those libraries, as well as all of those libraries' dependencies (and the dependencies' dependencies and so on).

Now you have everything installed that you need to run the app! Open the Expo XDE app and click the `Open existing project...` button. Select the folder that contains this file (make sure you have the parent folder and not this file) and press `Open`.

You should now see two panels with logs. The left will output some messages, hopefully including `Dependency graph loaded.`. If you see this message, then your app is running (well technically the bundler that serves your app is running).

You can now open the app on your phone or simulator by clicking one of the buttons in the top right. To open on your phone, click the `Share` button and scan the QR code from the Expo app on your phone. To open in a simulator, click the `Device` button and select the simulator into which you want to open your app.

When you have the app open in your phone or simulator, try opening **[`App.js`](/App.js)** and changing a line. You should see it update on your phone!

You can now begin to work on your app. You may find the vibrate function in **[`/utils`](/utils)** helpful. Feel free to import and use it in your app like this:

```javascript import {vibrate} from './utils'

// causes phone to vibrate vibrate()```

Good luck!

**Staff Solution:** If you want to play with the staff implementation, you can view it usingSnack at **[@jhhayashi/project1-solution](<https://snack.expo.io/@jhhayashi/project1-solution>)**.

**# Project 2 - Movie Browser**

**Prerequisite:**

For this project, you'll be implementing a movie browser. It will allow users to search for movies included in the [Open Movie Database](<http://www.omdbapi.com/>) and view additional information about any movies they select. Check out the [staff solution](#staff-solution) for a working version.

**Requirements**

- You may not import libraries other than the below:

- `expo`

- `react`

- `react-native`

- `prop-types`

- `react-navigation`

- Any library for icons

- There should be at least one `Stack Navigator`

- There should be a search screen that allows users to search for movies

- You should show more than 10 results if more than 10 results exist

- There should be a screen that shows additional information about a selected movie

The aesthetics of the app are up to you!

**Getting Started**

First, head to [this link](<https://docs.expo.io/versions/latest/introduction/installation.html>) to install Expo. You'll need the XDE for your computer and the mobile client (Expo app) on your phone. If you prefer, you can also install the iOS simulator (Macs only) and/or the Android emulator.

You'll also need Node.js and NPM installed. You can check if you already have them installed by opening a terminal and running `node --version` and `npm --version`. If numbers are printed, you're good to go. If not, **[install them](https://nodejs.org/en/)**.

You'll probably want the LTS version (v8.x.x). NPM will be installed automatically when you install node.

After installing those software dependencies, you'll need to install your app's "dependencies" (libraries that are required to run the app, such as `react`, `react-native`, etc.). Fortunately, it's very easy to do! From a terminal, `cd` into this directory and run the command `npm install`. NPM will look at the **[`package.json`](/package.json)** file's `dependencies` key and install those libraries, as well as all of those libraries' dependencies (and the dependencies' dependencies and so on).

Now you have everything installed that you need to run the app! Open the Expo XDE app and click the `Open existing project...` button. Select the folder that contains this file (make sure you have the parent folder and not this file) and press `Open`.

You should now see two panels with logs. The left will output some messages, hopefully including `Dependency graph loaded.`. If you see this message, then your app is running (well technically the bundler that serves your app is running).

You can now open the app on your phone or simulator by clicking one of the buttons in the top right. To open on your phone, click the `Share` button and scan the QR code from the Expo app on your phone. To open in a simulator, click the `Device` button and select the simulator into which you want to open your app.

When you have the app open in your phone or simulator, try opening **[`App.js`](/App.js)** and changing a line. You should see it update on your phone!

If you want to get started before we talk about data fetching in the next lecture, you can use the mock data defined in **[`mockData.js`](./mockData.js)**.

Good luck!

**Staff Solution**

If you want to play with the staff implementation, you can view it using

Snack at **[@jhhayashi/project2-solution](**[**https://snack.expo.io/@jhhayashi/project2-solution**](https://snack.expo.io/@jhhayashi/project2-solution)**).**