

Multi-platform setup

How to integrate React Native for Web into an existing React Native codebase.

Please read the [setup](#) guide first. If you have an existing React Native application, there are more areas that require attention and customization before most web bundlers can consume the non-standard JavaScript in packages produced by the React Native ecosystem. Additionally, 3rd party React Native packages with web support are listed in the [React Native Directory](#).

If you are interested in making a multi-platform app it is *strongly recommended* that you use [Expo](#) (or learn from the source code for the Web integration). Expo includes [web support](#) and takes care of all the configuration work required.

Web-specific code

Minor platform differences can use the **Platform** module.

```
import { Platform } from 'react-native';

const styles = StyleSheet.create({
  height: (Platform.OS === 'web') ? 200 : 100,
});
```

More significant platform differences should use platform-specific files (see the webpack configuration below for resolving `*.web.js` files):

For example, with the following files in your project:

```
MyComponent.android.js
MyComponent.ios.js
MyComponent.web.js
```

And the following import:

```
import MyComponent from './MyComponent';
```


React Native will automatically import the correct variant for each specific target platform.

Compiling and Bundling

What follows is only an *example* of a basic way to package a Web app using [webpack](#) and [Babel](#). ([Metro](#) is the React Native bundler with [undocumented web support](#).)

Install webpack-related dependencies, for example:

```
npm install --save-dev babel-loader url-loader webpack webpack-cli webpack-dev-se
```



React Native's Babel preset rewrites ES modules to CommonJS modules, preventing bundlers from automatically performing "tree-shaking" to remove unused modules from your web app build. To help with this, you can install the following Babel plugin:

```
npm install --save-dev babel-plugin-react-native-web
```

Create a `web/webpack.config.js` file:

```
// web/webpack.config.js

const path = require('path');
const webpack = require('webpack');

const appDirectory = path.resolve(__dirname, '../');

// This is needed for webpack to compile JavaScript.
// Many OSS React Native packages are not compiled to ES5 before being
// published. If you depend on uncompiled packages they may cause webpack build
```

```
// errors. To fix this webpack can be configured to compile to the necessary
// `node_module`.
const babelLoaderConfiguration = {
  test: /\.js$/,
  // Add every directory that needs to be compiled by Babel during the build.
  include: [
    path.resolve(appDirectory, 'index.web.js'),
    path.resolve(appDirectory, 'src'),
    path.resolve(appDirectory, 'node_modules/react-native-uncompiled')
  ],
  use: {
    loader: 'babel-loader',
    options: {
      cacheDirectory: true,
      // The 'metro-react-native-babel-preset' preset is recommended to match React
      presets: ['module:metro-react-native-babel-preset'],
      // Re-write paths to import only the modules needed by the app
      plugins: ['react-native-web']
    }
  }
};

// This is needed for webpack to import static images in JavaScript files.
const imageLoaderConfiguration = {
  test: /\.(gif|jpe?g|png|svg)$/,
  use: {
    loader: 'url-loader',
    options: {
      name: '[name].[ext]',
      esModule: false,
    }
  }
};

module.exports = {
  entry: [
    // load any web API polyfills
    // path.resolve(appDirectory, 'polyfills-web.js'),
    // your web-specific entry file
    path.resolve(appDirectory, 'index.web.js')
  ],
```

```

// configures where the build ends up
output: {
  filename: 'bundle.web.js',
  path: path.resolve(appDirectory, 'dist')
},

// ...the rest of your config

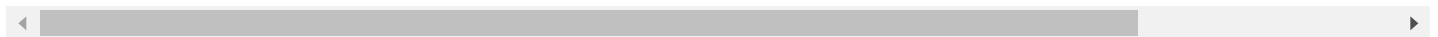
module: {
  rules: [
    babelLoaderConfiguration,
    imageLoaderConfiguration
  ]
},

resolve: {
  // This will only alias the exact import "react-native"
  alias: {
    'react-native$': 'react-native-web'
  },
  // If you're working on a multi-platform React Native app, web-specific
  // module implementations should be written in files using the extension
  // `.web.js`.
  extensions: [ '.web.js', '.js' ]
}
}

```

To run in development from the root of your application:

```
./node_modules/.bin/webpack-dev-server -d --config ./web/webpack.config.js --mode development
```



To build for production:

```
./node_modules/.bin/webpack -p --config ./web/webpack.config.js
```

Please refer to the Webpack documentation for more information on configuration.

Updated July 20, 2023 Edit



[React Native for Web](#) – Copyright © Nicolas Gallagher and Meta Platforms, Inc.