Metro

React Native uses Metro to build your JavaScript code and assets.

Configuring Metro

Configuration options for Metro can be customized in your project's `metro.config.js` file. This can export either:

- **An object (recommended)** that will be merged on top of Metro's internal config defaults.
- **A function** that will be called with Metro's internal config defaults and should return a final config object.

TIP

Please see **Configuring Metro** on the Metro website for documentation on all available config options.

In React Native, your Metro config should extend either `@react-native/metro-config` or `@expo/metro-config`. These packages contain essential defaults necessary to build and run React Native apps.

Below is the default `metro.config.js` file in a React Native template project:

```
const {getDefaultConfig, mergeConfig} = require('@react-native/metro-config');

/**
 * Metro configuration
 * https://facebook.github.io/metro/docs/configuration
 *
 * @type {import('metro-config').MetroConfig}
 */
const config = {};
```

```
module.exports = mergeConfig(getDefaultConfig(__dirname), config);
```

Metro options you wish to customize can be done so within the `config` object.

Advanced: Using a config function

Exporting a config function is an opt-in to managing the final config yourself — **Metro will not apply any internal defaults**. This pattern can be useful when needing to read the base default config object from Metro or to set options dynamically.

> INFO
>
> **From `@react-native/metro-config` 0.72.1**, it is no longer necessary to use a config function to access the complete default config. See the **Tip** section below.

```
const {getDefaultConfig, mergeConfig} = require('@react-native/metro-config');

module.exports = function (baseConfig) {
  const defaultConfig = mergeConfig(baseConfig, getDefaultConfig(__dirname));
  const {resolver: {assetExts, sourceExts}} = defaultConfig;

  return mergeConfig(
    defaultConfig,
    {
      resolver: {
        assetExts: assetExts.filter(ext => ext !== 'svg'),
        sourceExts: [...sourceExts, 'svg'],
      },
    },
  );
};
```

> TIP
>
> Using a config function is for advanced use cases. A simpler method than the above, e.g. for customising `sourceExts`, would be to read these defaults from `@react-native/metro-config`.

Alternative

```
const defaultConfig = getDefaultConfig(__dirname);

const config = {
  resolver: {
    sourceExts: [...defaultConfig.resolver.sourceExts, 'svg'],
  },
};

module.exports = mergeConfig(defaultConfig, config);
```

However!, we recommend copying and editing when overriding these config values — placing the source of truth in your config file.

✅ **Recommended**

```
const config = {
  resolver: {
    sourceExts: ['js', 'ts', 'tsx', 'svg'],
  },
};
```

Learn more about Metro

- Metro website
- Video: "Metro & React Native DevX" talk at App.js 2023

Is this page useful?

Edit this page

*Last updated on **Jul 3, 2023***