Troubleshooting

`npm start` doesn't detect changes

When you save a file while `npm start` is running, the browser should refresh with the updated code.

If this doesn't happen, try one of the following workarounds:

- Check that your file is imported by your entrypoint. TypeScript will show errors on any of your source files, but webpack only reloads your files if they are directly or indirectly imported by one of your entrypoints.
- If your project is in a Dropbox folder, try moving it out.
- If the watcher doesn't see a file called `index.js` and you're referencing it by the folder name, you need to restart the watcher due to a webpack bug.
- Some editors like Vim and IntelliJ have a "safe write" feature that currently breaks the watcher. You will need to disable it. Follow the instructions in "Adjusting Your Text Editor".
- If your project path contains parentheses, try moving the project to a path without them. This is caused by a webpack watcher bug.
- On Linux and macOS, you might need to tweak system settings to allow more watchers.
- If the project runs inside a virtual machine such as (a Vagrant provisioned) VirtualBox, create an `.env` file in your project directory if it doesn't exist, and add `CHOKIDAR_USEPOLLING=true` to it. This ensures that the next time you run `npm start`, the watcher uses the polling mode, as necessary inside a VM.

If none of these solutions help please leave a comment in this thread.

`npm start` fail due to watch error

If you are using a Linux operating system and see an error similar to: `ENOSPC: System limit for number of file watchers reached`, you can fix the issue by increasing the `fs.inotify.max_user_watches` setting of your operating system.

If you are running Debian, RedHat, or another similar Linux distribution, run the following in a terminal:

```
echo fs.inotify.max_user_watches=524288 | sudo tee -a /etc/sysctl.conf && sudo sysctl -p
```

If you are running ArchLinux, run the following command instead:

```
echo fs.inotify.max_user_watches=524288 | sudo tee /etc/sysctl.d/40-max-user-watches.conf &&
```

Then paste it in your terminal and press on enter to run it. You could find more information here.

`npm test` hangs or crashes on macOS Sierra

If you run `npm test` and the console gets stuck after printing `react-scripts test` to the console there might be a problem with your Watchman installation as described in facebook/create-react-app#713.

We recommend deleting `node_modules` in your project and running `npm install` (or `yarn` if you use it) first. If it doesn't help, you can try one of the numerous workarounds mentioned in these issues:

- facebook/jest#1767
- facebook/watchman#358
- ember-cli/ember-cli#6259

It is reported that installing Watchman 4.7.0 or newer fixes the issue. If you use Homebrew, you can run these commands to update it:

```
watchman shutdown-server
brew update
brew reinstall watchman
```

You can find other installation methods on the Watchman documentation page.

If this still doesn't help, try running `launchctl unload -F ~/Library/LaunchAgents/com.github.facebook.`

There are also reports that *uninstalling* Watchman fixes the issue. So if nothing else helps, remove it from your system and try again.

`npm run build` exits too early

It is reported that `npm run build` can fail on machines with limited memory and no swap space, which is common in cloud environments. Even with small projects this command can increase RAM usage in your system by hundreds of megabytes, so if you have less than 1 GB of available memory your build is likely to fail with the following message:

> The build failed because the process exited too early. This probably means the system ran out of memory or someone called `kill -9` on the process.

If you are completely sure that you didn't terminate the process, consider adding some swap space to the machine you're building on, or build the project locally.

`npm run build` fails on Heroku

This may be a problem with case sensitive filenames. Please refer to this section.

Moment.js locales are missing

If you use a Moment.js, you might notice that only the English locale is available by default. This is because the locale files are large, and you probably only need a subset of all the locales provided by Moment.js.

To add a specific Moment.js locale to your bundle, you need to import it explicitly.

For example:

```
import moment from 'moment';
import 'moment/locale/fr';
```

If you are importing multiple locales this way, you can later switch between them by calling `moment.locale()` with the locale name:

```
import moment from 'moment';
import 'moment/locale/fr';
import 'moment/locale/es';

// ...

moment.locale('fr');
```

This will only work for locales that have been explicitly imported before.

`npm run build` fails to minify

Before `react-scripts@2.0.0`, this problem was caused by third party `node_modules` using modern JavaScript features because the minifier couldn't handle them during the build. This has been solved by compiling standard modern JavaScript features inside `node_modules` in `react-scripts@2.0.0` and higher.

If you're seeing this error, you're likely using an old version of `react-scripts`. You can either fix it by avoiding a dependency that uses modern syntax, or by upgrading to `react-scripts@>=2.0.0` and following the migration instructions in the changelog.