We've tried to make upgrading your app or package as painless as possible, but in some cases, you will have to make small modifications to your Meteor 1.1 code to get it to run in Meteor 1.2. This document outlines the breaking changes and potential compatibility issues in Meteor 1.2.

This document is meant to be as short as possible so that you can read it all. If you want to learn about all of the new features in Meteor 1.2, read here: XXX to be posted later

Build tool and package improvements

1. **standard-minifiers package (automatic upgrade):** Starting in Meteor 1.2, your app must include the `standard-minifiers` package to get the default CSS and JS handling that was hard-coded in Meteor 1.1. When you update your app to 1.2 or a prerelease of 1.2, Meteor will add the `standard-minifiers` package to your app automatically. Besides minifying your code in production, the biggest effect of `standard-minifiers` is in how URLs in CSS are interpreted, such as for fonts and images. In a Meteor 1.1 app or an app with `standard-minifiers`, the URLs are interpreted relative to the top level, because all CSS is concatenated in development, and concatenated and minified in production. If you remove `standard-minifiers` in Meteor 1.2, URLs in CSS will be interpreted as relative to the CSS file, as per web standards. This change was made to give developers more control over minification, with less hard-coded functionality.
2. **PackageAPI#addAssets (change required to package.js file):** In packages, files added with `api.addFiles` that have no recognizable extension are no longer added as static assets. You must now use `api.addAssets` to add a static asset. Package authors must make this change to their `package.js` files before they can publish from Meteor 1.2, but existing published packages will continue to work. The hidden `isAsset: true` flag to `api.addFiles` is now deprecated and throws an error suggesting that you should use `addAssets` instead.
3. **LESS, CoffeeScript, Stylus incompatible (change to package dependencies and republish required):** We've rewritten the core build plugin packages - LESS, CoffeeScript, and Stylus - to use a new, faster build system. Packages that use these packages will need to update their dependencies to the new versions and be republished before they can be used with Meteor 1.2.

Splitting up meteor-platform

1. **New apps don't include meteor-platform (packages need to declare all of their dependencies):** In the past, if a package author forgot to include a dependency on `jquery`, `mongo`, or a similarly common package in your `package.js` file, the package would usually still work because most apps had these packages by default. We are now trying to make these parts of Meteor more modular, so package authors should take care to declare all of their package dependencies in `package.js`.
2. **test-packages no longer includes any packages globally (no changes needed):** To help with the above transition, we have update the `test-packages` command to avoid making any packages available globally. If your tests passed before but fail in Meteor 1.2 with a message like `$ is undefined`, that means that your package had an undeclared dependency on jQuery and the tests only passed because `test-packages` itself added jQuery to the global namespace. The fix is to make sure your dependencies in `package.js` are correct.

Ecmascript

1. **New language keywords (changes required to some variable names):** According to the ES2015 standard, some names that used to be valid have become keywords, for example `package` and `let`. You should get a nice error that will notify you that some names need to be changed.

Case insensitive usernames and emails

1. **Meteor will attempt to keep usernames and emails unique (change required to code that reads and writes user data):** In Meteor 1.2, the accounts-password package tries to make sure that users don't need to remember the capitalization of their usernames and email addresses. Some code in your app might be using `Meteor.users.findOne({username: x})` or similar to search for users, and this will be a case-sensitive lookup. Instead, the new API method should be used like so: `Accounts.findUserByUsername(x)` This way, the username will be matched in a case-insensitive fashion. The same thing applies for email addresses and the new API method `Accounts.findUserByEmail`.

React

1. **New official React package (update package dependencies if you are using a different React package):** There are several popular React packages for Meteor, and we hope to centralize them all into one package, simply called `react`. If your package depends on `reactjs:react` or another package that loads React, switch to the official one so that users can have one global version of the React library. If you are using the `getMeteorData` mixin for reactive data, you should depend on `react`; if you just need the React rendering library itself, then you should depend on `react-runtime`, which is a subset of the `react` package.

Mobile

1. **Bring your own Android tools**: The bundled Android tools have been removed and a system-wide install of the Android SDK is now required. This should make it easier to keep the development toolchain up to date and helps avoid some difficult to diagnose failures. If you don't have your own Android tools installed already, you can find more information about installing the Android SDK for [Mac] (https://github.com/meteor/meteor/wiki/Mobile-Development-Install:-Android-on-Mac) or [Linux] (https://github.com/meteor/meteor/wiki/Mobile-Development-Install:-Android-on-Linux).
2. **Cordova has been updated from 4.2.0 to 5.2.0**: tools, platforms and plugins have been updated. This may require you to make changes to your app. For details, see the Cordova release notes for the releases up to 5.2.0.
3. **New names for Cordova plugins**: We've upgraded to a new version of Cordova that uses npm to manage its plugins instead of the old Cordova registry. As part of this move, many Cordova plugins have been renamed. Meteor should perform conversions automatically, but you may want to be aware of this to avoid surprises. See here for more information.
4. **Output logging**: As a known issue, we do not show logging output when running on the iOS Simulator. As a workaround, you can use `meteor run ios-device` to open the project in Xcode and watch the console output in the debug area.