After reading this guide, you'll know:

1. What React is, and why you would consider using it with Meteor.
2. How to install React in your Meteor application, and how to use it correctly.
3. How to integrate React with Meteor's realtime data layer.
4. How to route in a React/Meteor application.

Introduction

React is a JavaScript library for building reactive user interfaces developed and distributed by the Facebook team.

React has a vibrant and growing ecosystem and is used widely in production in a variety of combinations with different frameworks.

To learn more about using React in general and coming up to speed with the library, you should check out the React documentation.

To get started with React in Meteor, you can follow along the React tutorial.

Installing and using React

To install React in Meteor should add it as a npm dependency:

```
meteor npm install --save react react-dom
```

This will install `react` into your project and allow you to access it within your files with `import React from 'react'`.

```
import React from 'react';

export const HelloWorld = () => <h1>Hello World</h1>;
```

You can render a component hierarchy to the DOM using the `react-dom` package:

```
import { Meteor } from 'meteor/meteor';
import React from 'react';
import { render } from 'react-dom';
import { HelloWorld } from './HelloWorld.js';

Meteor.startup(() => {
  render(<HelloWorld />, document.getElementById('app'));
});
```

You need to include a `<div id="app"></div>` in your body's HTML somewhere of course.

By default Meteor already uses React when you create a new app using `meteor create my-app` then this basic set up will be already ready for you.

Using 3rd party packages

Meteor does not require any different configuration as Meteor is 100% compatible with NPM, so you can use any React component library.

Using Meteor's data system

React is a front-end rendering library and as such doesn't concern itself with how data gets into and out of components.

On the other hand, Meteor offers in the core packages publications and methods, used to subscribe to and modify the data in your application.

To integrate the two systems, we've developed a `react-meteor-data` package which allows React components to respond to data changes via Meteor's Tracker reactivity system.

Using `useTracker`

> *The `useTracker` function follows latest best practices of React. Choosing hooks instead of HOCs.*

To use data from a Meteor collection inside a React component, install `react-meteor-data`:

```
meteor add react-meteor-data
```

Once installed, you'll be able to import the `useTracker` function and others.

You can learn more about them here

Routing

Although there are many solutions for routing with Meteor and React, react-router is the most popular package right now.

As always Meteor does not require anything different when using React Router so you can follow their quick-start guide to set up React Router in your Meteor project.

Meteor Packages and Blaze

Using React in Atmosphere Packages

If you are writing an Atmosphere package and want to depend on React or an npm package that itself depends on React, you can't use `Npm.depends()` and `Npm.require()`, as this will result in *2* copies of React being installed into the application (and besides `Npm.require()` only works on the server).

Instead, you need to ask your users to install the correct npm packages in the application itself. This will ensure that only one copy of React is shipped to the client and there are no version conflicts.

In order to check that a user has installed the correct versions of npm packages, you can use the `tmeasday:check-npm-versions` package to check dependency versions at runtime.

React Components in Blaze

If you are not using Blaze with React you can skip this.

If you'd like to use React within a larger app built with Blaze (which is a good strategy if you'd like to incrementally migrate an app from Blaze to React), you can use the `react-template-helper` component which renders a react component inside a Blaze template. First run `meteor add react-template-helper`, then use the `React` helper in your template:

```
<template name="userDisplay">
  <div>Hello, {{username}}</div>
  <div>{{> React component=UserAvatar userId=_id}}</div>
</template>
```

You will need to pass in the component class with a helper:

```
import { Template } from 'meteor/templating';

import './userDisplay.html';
import UserAvatar from './UserAvatar.js';

Template.userDisplay.helpers({
  UserAvatar() {
    return UserAvatar;
  }
})
```

The `component` argument is the React component to include, which should be passed in with a helper.

Every other argument is passed as a prop to the component when it is rendered.

Note that there a few caveats:

- React components must be the only thing in the wrapper element. Due to a limitation of React (see facebook/react #1970, #2484), a React component must be rendered as the only child of its parent node, meaning it cannot have any siblings.
- This means a React component also can't be the only thing in a Blaze template, because it's impossible to tell where the template will be used.

Passing callbacks to a React component

To pass a callback to a React component that you are including with this helper, make a template helper that returns a function, and pass it in as a prop, like so:

```
Template.userDisplay.helpers({
  onClick() {
    const instance = Template.instance();

    // Return a function from this helper, where the template instance is in
    // a closure
    return () => {
      instance.hasBeenClicked.set(true)
    }
  }
});
```

To use it in Blaze:

```
<template name="userDisplay">
  <div>
    {{> React component=UserAvatar userId=_id onClick=onClick}}
  </div>
</template>
```

Blaze Templates in React

We can also use Blaze templates in React components. This is similarly useful for a gradual transition strategy; but more importantly, it allows us to continue to use the multitude of Atmosphere packages built for Blaze in our React projects, as well as core packages like `accounts-ui`.

One way to do this is with the `gadicc:blaze-react-component` package. First run `meteor add gadicc:blaze-react-component`, then import and use it in your components as follows:

```
import React from 'react';
import Blaze from 'meteor/gadicc:blaze-react-component';

const App = () => (
  <div>
    <Blaze template="itemsList" items={items} />
  </div>
);
```

The `<Blaze template="itemsList" items={items} />` line is the same as if you had written `{% raw %}{{> itemsList items=items}}{% endraw %}` inside of a Blaze template. For other options and further information, see the package's project page.