**Using Immutable.js Records**

[Immutable.js records](https://facebook.github.io/immutable-js/docs/#Record) are an awesome addition to your app. Why?

1. **They're self-documenting**. Any developer will be able to see all the fields within a resource/record
2. **They enforce strict code quality**. You can't add undocumented fields to a record.
3. **They combine the benefits of ImmutableJS' Map() with normal JS objects**. They have standard accessors (post.name vs post.get('name')), you can use destructuring, and they have Immutable's strict equality.
4. You can use getIn within maps and records. If a map contains records you can read record's values using getIn.

**Setting up Records**

Immutable records are easy to set up and define. For each resource in your app you define a new record as follows:

'use strict';

import { Record, Map } from 'immutable';

export const Post = new Record({

id: undefined,

title: '',

content: '',

author: undefined

comments: new Map()

});

So you're passing in an object to define a record. The keys of this object are the *only* fields you can set on the new record; the values are used as defaults. Which means you can't set post.foo - it'll fail.

All said, because you need to explicitly state each field as you define a record it becomes self-documenting. Any developer joining your team can look at the record definitions to see which data to use in their selectors. This saves them from having to dig around in your API looking for responses.

**Using Records**

Records are immutable. That means that once set you can't change any of the values. Instead, you can do this:

const post = new Post({ title: 'foo', content: 'misc' });

const edited = post.set('title', 'bar');

Records inherit all [Immutable.Map functions](https://facebook.github.io/immutable-js/docs/#/Map) such as set, setIn, merge etc, and each of these return a *new* instance of the record.

This also means that you can use setIn from a parent Map to access record values:

const data = new Map({

somePost: new Post({ title: 'some post' })

});

console.log(data.getIn(['somePost', 'title'])); // === 'some post';