119-Testing Meteor Methods video

- create two new files in imports/api called notes.js and its companion called notes.test.js
- there are the only two files we are going to be needing for methods and publications to test them
- in notes.js we are going to setup the basic collection, we will define a single method for inserting a note the we'll test it
- so in notes.js we will need a collection, so we need to import Meteor mongo, and Meteor

```
import { Mongo } from 'meteor/mongo';
import { Meteor } from 'meteor/meteor';
```

- And now we need to create a new Mongo collection, and we know it takes one argument, the name of our collection

```
import { Mongo } from 'meteor/mongo';
import { Meteor } from 'meteor/meteor';

export const Notes = new Mongo.Collection('notes');
```

- **create a** that is going to insert a new note

```
import { Mongo } from 'meteor/mongo';
import { Meteor } from 'meteor/meteor';

export const Notes = new Mongo.Collection('notes');

Meteor.methods({
    'notes.insert'(){

    }
});
```

- we then are going to some user authentication, we don't want an anonymous user
- if the user is not defined we will throw an error

```
import { Mongo } from 'meteor/mongo';
import { Meteor } from 'meteor/meteor';

export const Notes = new Mongo.Collection('notes');
```

```
Meteor.methods({
    'notes.insert'(){
        if(!this.userId){
            throw new Meteor.Error('not-authorized');
        }
    }
});
```

- if the user is authorized we just one to return a new note

```
import { Mongo } from 'meteor/mongo';
import { Meteor } from 'meteor/meteor';

export const Notes = new Mongo.Collection('notes');

Meteor.methods({
    'notes.insert'(){
        if(!this.userId){
            throw new Meteor.Error('not-authorized');
        }

        return Notes.insert({

        })
    }
});
```

- we will four object to return

```
import { Mongo } from 'meteor/mongo';
import { Meteor } from 'meteor/meteor';

export const Notes = new Mongo.Collection('notes');

Meteor.methods({
    'notes.insert'(){
        if(!this.userId){
```

```
            throw new Meteor.Error('not-authorized');
        }

        return Notes.insert({
            title: '',
            body: '',
            userId: this.userId,
            updatedAt: new Date().getTime();
        })
    }
});
```

- instead of new Date lets use moment, so lets import it first and then add it our code

```
import { Mongo } from 'meteor/mongo';
import { Meteor } from 'meteor/meteor';
import moment from 'moment';

export const Notes = new Mongo.Collection('notes');

Meteor.methods({
    'notes.insert'(){
        if(!this.userId){
            throw new Meteor.Error('not-authorized');
        }

        return Notes.insert({
            title: '',
            body: '',
            userId: this.userId,
            updatedAt: moment().valueOf()
        })
    }
});
```

- now lets go to notes.test.js
- import Meteor and expect

```
import { Meteor } from 'meteor/meteor';
import expect from 'expect';
```

- lets now check if we are in the server, if we are lets call describe block, takes two arguments, the name of our block and a function

```
import { Meteor } from 'meteor/meteor';
import expect from 'expect';

if(Meteor.isServer){
    describe('notes', function(){

    });
}
```

- **we then** want to run the succesful case

```
import { Meteor } from 'meteor/meteor';
import expect from 'expect';

if(Meteor.isServer){
    describe('notes', function(){
        it('should insert new note', function(){

        })
    });
}
```

- now we need to figure out how to access the notes.insert function to test it, we can use this Meteor method

```
import { Meteor } from 'meteor/meteor';
import expect from 'expect';

if(Meteor.isServer){
    describe('notes', function(){
        it('should insert new note', function(){
            Meteor.server.method_handlers['notes.insert']
```

```
        })
    });
}
```

- in our case we want to call notes.insert with a userId, how do we the this keyword, and we are going to be doing this by using apply, apply is javascript feature

```
import { Meteor } from 'meteor/meteor';
import expect from 'expect';

if(Meteor.isServer){
    describe('notes', function(){
        it('should insert new note', function(){
            Meteor.server.method_handlers['notes.insert'].apply({userId: 'testid'});
        })
    });
}
```

- lets store that value by creating a const

```
import { Meteor } from 'meteor/meteor';
import expect from 'expect';

if(Meteor.isServer){
    describe('notes', function(){
        it('should insert new note', function(){
            const _id = Meteor.server.method_handlers['notes.insert'].apply({userId: 'testid'});
        })
    });
}
```

- import Notes from notes.js

```
import { Meteor } from 'meteor/meteor';
import expect from 'expect';
import { Notes } from './notes'
```

- **next** we are going to find one single document

```
import { Meteor } from 'meteor/meteor';
import expect from 'expect';
import { Notes } from './notes'

if(Meteor.isServer){
    describe('notes', function(){
        it('should insert new note', function(){
            const _id = Meteor.server.method_handlers['notes.insert'].apply({userId: 'testid'});

            Notes.findOne()
        })
    });
}
```

- we do want to provide a query, we want to find the _id and their userId, lets create a const for testId up above, because we are going to be referencing it more than once

```
import { Meteor } from 'meteor/meteor';
import expect from 'expect';
import { Notes } from './notes'

if(Meteor.isServer){
    describe('notes', function(){
        it('should insert new note', function(){
            const userId = 'testid';
            const _id = Meteor.server.method_handlers['notes.insert'].apply({ userId });

            Notes.findOne({ _id, userId })
        })
    });
}
```

- the last thing to do to assert a note was indeed found, we will do that by using expect, we are expecting the return from Notes.findOne, so we need to copy that into the expect argument

```
import { Meteor } from 'meteor/meteor';
import expect from 'expect';
import { Notes } from './notes'
```

```
if(Meteor.isServer){
    describe('notes', function(){
        it('should insert new note', function(){
            const userId = 'testid';
            const _id = Meteor.server.method_handlers['notes.insert'].apply({ userId });

            expect(Notes.findOne({ _id, userId }));

        })
    });
}
```

- then we attach toExist, and this will pass if their is a value and fail if there isnt

```
import { Meteor } from 'meteor/meteor';
import expect from 'expect';
import { Notes } from './notes'

if(Meteor.isServer){
    describe('notes', function(){
        it('should insert new note', function(){
            const userId = 'testid';
            const _id = Meteor.server.method_handlers['notes.insert'].apply({ userId });

            expect(Notes.findOne({ _id, userId })).toExist();

        })
    });
}
```

- now over in the browser, we should see it passing
- now we also want to test our method if there is no userId, we want to make sure we do get an error and that no note gets inserted
- lets go back to notes.test.js, lets add a second test case

```
import { Meteor } from 'meteor/meteor';
import expect from 'expect';
```

```
import { Notes } from './notes'

if(Meteor.isServer){
    describe('notes', function(){
        it('should insert new note', function(){
            const userId = 'testid';
            const _id = Meteor.server.method_handlers['notes.insert'].apply({ userId });

            expect(Notes.findOne({ _id, userId })).toExist();

        });

        it('should not insert note if not authenticated', function(){

        });
    });
}
```

- we then are going to be calling the Meteor.server.method_handlers['notes.insert'].apply({ userId }); handler without the this content
- to this we will be using expect with an arrow that is going to get called, and toss a toThrow()

```
import { Meteor } from 'meteor/meteor';
import expect from 'expect';
import { Notes } from './notes'

if(Meteor.isServer){
    describe('notes', function(){
        it('should insert new note', function(){
            const userId = 'testid';
            const _id = Meteor.server.method_handlers['notes.insert'].apply({ userId });

            expect(Notes.findOne({ _id, userId })).toExist();

        });

        it('should not insert note if not authenticated', function(){
            expect(()=>{
```

```
                Meteor.server.method_handlers['notes.insert']();
            }).toThrow();
        });
    });
}
```

- over in the browser we should get a pass because new should not get inserted