120-Testing Note Removal video

- now we are going to be using Notes.remove, and it will have an argument, the id
- so lets go to notes.js
- to remove a note, we first need to have a note

```
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()
        })
    },
    'notes.remove'(_id){

    }

});
```

- now we need to setup the database before our test cases run, go to notes.test.js
- we are going to be a mocha lify cycle method, we are going to be using beforeEach(), this function runs before every test case, this takes only one arguement, a function,

```
import { Meteor } from 'meteor/meteor';
import expect from 'expect';
import { validateNewUser } from './users';
```

```
if(Meteor.isServer){
    describe('users', function(){

        beforeEach(function(){


        });

        it('should allow valid email address', function(){
            const testUser = {
                emails: [
                    {
                        address: 'Test@email.com'
                    }
                ]
            };
            const res = validateNewUser(testUser);

            expect(res).toBe(true);
        });

        it('should reject invalid email', function(){
            expect(()=>{
                const testUser = {
                    emails: [
                        {
                            address: 'Testemailcom'
                        }
                    ]
                };
                validateNewUser(testUser);
            }).toThrow();
        })
    });
}
```

- first we want to remove everything from the notes collection
- then we are going to insert a new note, and we are going to be adding the same properties as the one in notes.js

```
import { Meteor } from 'meteor/meteor';
import expect from 'expect';
import { validateNewUser } from './users';



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

        beforeEach(function(){
            Notes.remove({});
            Notes.insert({
                _id: 'testNoteId1',
                title: 'My Title',
                body: 'My body for note',
                updateAt: 0,
                userId: 'testUserId1'
            });
        })

        it('should allow valid email address', function(){
            const testUser = {
                emails: [
                    {
                        address: 'Test@email.com'
                    }
                ]
            };
            const res = validateNewUser(testUser);

            expect(res).toBe(true);
        });

        it('should reject invalid email', function(){
            expect(()=>{
                const testUser = {
                    emails: [
                        {
                            address: 'Testemailcom'
                        }
                    ]
                };
                validateNewUser(testUser);
            }).toThrow();
        })
    });
}
```

- now lets do another test case below that should remove a note, inside our method we a need a method that removes a note

```
        it('should reject invalid email', function(){
            expect(()=>{
                const testUser = {
                    emails: [
                        {
                            address: 'Testemailcom'
                        }
                    ]
                };
                validateNewUser(testUser);
            }).toThrow();
        });
        it('should remove note', function(){
            Meteor.server.method_handlers['notes.remove'].apply({userId: 'testUserId1'});
        })
    });
}
```

- we are now going to add a second argument to apply(), an array, and its going to be the _id we want to remove

```
        it('should reject invalid email', function(){
            expect(()=>{
                const testUser = {
                    emails: [
                        {
                            address: 'Testemailcom'
                        }
                    ]
                };
                validateNewUser(testUser);
            }).toThrow();
        });
        it('should remove note', function(){
            Meteor.server.method_handlers['notes.remove'].apply({userId: 'testUserId1'}, ['testN
oteId1']);
        })
    });
}
```

- we are now going to call expect and inside we are going to call Notes.findOne

```
        it('should reject invalid email', function(){
            expect(()=>{
                const testUser = {
                    emails: [
                        {
                            address: 'Testemailcom'
                        }
                    ]
                };
                validateNewUser(testUser);
            }).toThrow();
        });
        it('should remove note', function(){
            Meteor.server.method_handlers['notes.remove'].apply({userId: 'testUserId1'}, ['testN
oteId1']);


            expect(Notes.findOne({_id: 'testNoteId1'})).toNotExist();
        })
    });
}
```

- there was a mistake of mine that I put code in wrong file, I put in users.test.js instead of notes.test.js
- this is what users.test.js should look like

```
import { Meteor } from 'meteor/meteor';
import expect from 'expect';
import { validateNewUser } from './users';



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

        it('should allow valid email address', function(){
            const testUser = {
                emails: [
                    {
                        address: 'Test@email.com'
                    }
                ]
            };
            const res = validateNewUser(testUser);

            expect(res).toBe(true);
        });

        it('should reject invalid email', function(){
            expect(()=>{
                const testUser = {
                    emails: [
                        {
```

```
                        address: 'Testemailcom'
                    }
                ]
            };
            validateNewUser(testUser);
        }).toThrow();
    });
  });
}
```

- and this is how notes.test.js should look like

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

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

        beforeEach(function(){
            Notes.remove({});
            Notes.insert({
                _id: 'testNoteId1',
                title: 'My Title',
                body: 'My body for note',
                updateAt: 0,
                userId: 'testUserId1'
            });
        })

        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();
        });

        it('should remove note', function(){
            Meteor.server.method_handlers['notes.remove'].apply({userId: 'testUserId1'}, ['testN
oteId1']);

            expect(Notes.findOne({_id: 'testNoteId1'})).toNotExist();
        })
    });
}
```

- and now lets go to notes.js and we are going to user notes.remove to a note, only if there is a userId, if there is we will new SimpleSchema to validate the _id to make sure it is a String, it has to be in length more than 1 character, and then we call Notes.remove and we pass _id as well as userId which will equal to this.userId

```
import { Mongo } from 'meteor/mongo';
import { Meteor } from 'meteor/meteor';
import SimpleSchema from 'simpl-schema';

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()

        })

    },

    'notes.remove'(_id){

        if(!this.userId){

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

        }


        new SimpleSchema({

            _id: {

                type: String,

                min: 1

            }

        }).validate({ _id });


         Notes.remove({ _id, userId: this.userId });

    }


});
```

- lets go back to notes.test.js to make quick twick, lets change the testNoteId1 to testNoteId4 which doesnt exist, but we want to check this to see if we get an error

```
        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();

        });
        it('should remove note', function(){
            Meteor.server.method_handlers['notes.remove'].apply({userId: 'testUserId1'}, ['testN
oteId4']);

            expect(Notes.findOne({_id: 'testNoteId1'})).toNotExist();
        })
    });
}
```

- and we do get an error, so everything is working as expected, now change it back to the way it was
- we are going to be adding two more test cases

```
       it('should not insert note if not authenticated', function(){
            expect(()=>{
                Meteor.server.method_handlers['notes.insert']();
            }).toThrow();
        });

        it('should remove note', function(){
            Meteor.server.method_handlers['notes.remove'].apply({userId: 'testUserId1'}, ['testN
oteId1']);

            expect(Notes.findOne({_id: 'testNoteId1'})).toNotExist();
        });

        it('should not remove note if unauthenticated', function(){

        })
```

- **in this case we want to call 'notes.remove' but without the this context, and we are going to be using expect**

```

        it('should not insert note if not authenticated', function(){
            expect(()=>{
                Meteor.server.method_handlers['notes.insert']();
            }).toThrow();
        });

        it('should remove note', function(){
            Meteor.server.method_handlers['notes.remove'].apply({userId: 'testUserId1'}, ['testN
oteId1']);

            expect(Notes.findOne({_id: 'testNoteId1'})).toNotExist();
        });

        it('should not remove note if unauthenticated', function(){
            expect(()=>{
                Meteor.server.method_handlers['meteor.remove'].apply({},['testNoteId1']);
            }).toThrow();
        })
    });
}
```

- now we are going to test _id

```


        it('should remove note', function(){
            Meteor.server.method_handlers['notes.remove'].apply({userId: 'testUserId1'}, ['testN
oteId1']);

            expect(Notes.findOne({_id: 'testNoteId1'})).toNotExist();
        });

        it('should not remove note if unauthenticated', function(){
            expect(()=>{
                Meteor.server.method_handlers['meteor.remove'].apply({},['testNoteId1']);
            }).toThrow();
        });

        it('should not remove note if invalid _id', function() {

        })

```

```
    });
}
```

- then lets call expect() and toThrow()

```


        it('should remove note', function(){
            Meteor.server.method_handlers['notes.remove'].apply({userId: 'testUserId1'}, ['testN
oteId1']);

            expect(Notes.findOne({_id: 'testNoteId1'})).toNotExist();
        });

        it('should not remove note if unauthenticated', function(){
            expect(()=>{
                Meteor.server.method_handlers['meteor.remove'].apply({},['testNoteId1']);
            }).toThrow();
        });

        it('should not remove note if invalid _id', function() {
            expect(()=>{
                Meteor.server.method_handlers['meteor.remove'].apply({_id: 'testUserId1'});
            }).toThrow();
        })
    });
}
```

-
-