118-Testing User Validation video

- over in users.test.js lets comment our everything for now
- over in users.js we are going to go a little re-structering in order for us to use the expect() function
- cut everything inside validateNewUser()
- make new const variable above it

```
import { Meteor } from 'meteor/meteor';
import SimpleSchema from 'simpl-schema';
import { Accounts } from 'meteor/accounts-base';

export const validateNewUser = (user)=>{
  const email = user.emails[0].address

  new SimpleSchema({
      email:{
        type: String,
        regEx: SimpleSchema.RegEx.Email
      }
    }).validate({ email });

  return true;
}
Accounts.validateNewUser();
```

- now we want dont want to break our code so all have to do is pass the new const we just created to Accounts.validateNewUser()

```
import { Meteor } from 'meteor/meteor';
import SimpleSchema from 'simpl-schema';
import { Accounts } from 'meteor/accounts-base';

export const validateNewUser = (user)=>{
  const email = user.emails[0].address

  new SimpleSchema({
      email:{
        type: String,
```

```
        regEx: SimpleSchema.RegEx.Email
      }
    }).validate({ email });


  return true;
}
Accounts.validateNewUser(validateNewUser);
```

- lets now go to users.test.js and import the validateNewUser

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

- **lets** start by creating a new describe block
- we are going to be adding to test cases
- in the first case we want this to pass

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


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 passes but it only shows in the server column, if we open the chrome tools in the console we see that** Uncaught TypeError: Accounts.validateNewUser is not a function and that is because this function only exists in the server

- to fix this lets go to users.js and make these changes

```
import { Meteor } from 'meteor/meteor';
import SimpleSchema from 'simpl-schema';
import { Accounts } from 'meteor/accounts-base';

export const validateNewUser = (user)=>{
  const email = user.emails[0].address

  new SimpleSchema({
      email:{
        type: String,
        regEx: SimpleSchema.RegEx.Email
      }
    }).validate({ email });

  return true;
}
if(Meteor.isServer){
  Accounts.validateNewUser(validateNewUser);
}
```

- lets also fix the users.test.js

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

- now lets create another test for throwing an error

```
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(()=>{

            }).toNotThrow();
        })
    });
}
```

- this will pass the test, but we throw an error , it will fail

```
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(()=>{
                throw new Error('Error message here');
            }).toNotThrow();
        })
    });
}
```

- in this we want it to throw an error so we need to change toNotThrow() to toThrow()
- and we are going to copy testUser to this it() function but make the email invalid

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

-