122-Testing Note Updating - part 2 video

- lets add a new test case in notes.test.js on the bottom

```
        it('should throw error if extra updates provided', function(){
            const name = 'jose'
            expect(()=>{
                Meteor.server.method_handlers['notes.update'].apply({
                    userId: noteOne.userId
                }, [
                    noteOne._id,
                    { title: 'new title', name: 'jose' }
                ]);
            }).toThrow();
        })

        it('should not update note if the user is not creator', function(){

        })
    });
}
```

- now copy everything from the test case 'should update note' and paste it on our new test case

```
        it('should not update note if the user is not creator', function(){
            const title = "This is an updated title"
            Meteor.server.method_handlers['notes.update'].apply({
                userId: noteOne.userId
            }, [
                noteOne._id,
                { title }
            ]);

            const note = Notes.findOne(noteOne._id);

            expect(note.updatedAt).toBeGreaterThan(0);
            expect(note).toInclude({
```

```
                title,
                body: noteOne.body
            });
        })
    });
}
```

- lets change the value of userId

```
 it('should not update note if the user is not creator', function(){
            const title = "This is an updated title"
            Meteor.server.method_handlers['notes.update'].apply({
                userId: 'testid'
            }, [
                noteOne._id,
                { title }
            ]);
```

- lets now change the assertion and delete expect(note.updatedAt) line
- and lets change .toInclude to .toInclude(noteOne)

```


        it('should not update note if the user is not creator', function(){
            const title = "This is an updated title"
            Meteor.server.method_handlers['notes.update'].apply({
                userId: 'testid'
            }, [
                noteOne._id,
                { title }
            ]);

            const note = Notes.findOne(noteOne._id);
            expect(note).toInclude(noteOne);
        })
    });
}
```

- this test should fail

- in order for this to pass we need to go to notes.js and tweek our code a little bit
- on Notes.update we want to check two things the _id and userId is equal to this.userid

```
        }).validate({
            _id,
            ...updates
        });

        Notes.update({
            _id,
            userId : this.userId,
        }, {
            $set: {
                updatedAt : moment().valueOf(),
                ...updates
            }
        })
    }

});
```

- this will pass our test in the browser
- lets go back to notes.test.js to do more test cases

```
            Meteor.server.method_handlers['notes.update'].apply({
                userId: 'testid'
            }, [
                noteOne._id,
                { title }
            ]);

            const note = Notes.findOne(noteOne._id);
            expect(note).toInclude(noteOne);
        });

        it('should not update note if unauthenticated', function(){
            expect(()=>{
                Meteor.server.method_handlers['notes.update'].apply({},[noteOne._id]);
```

```
            }).toThrow();
        });

        it('should not update note if invalid _id', function() {
            expect(()=>{
                Meteor.server.method_handlers['notes.update'].apply({_id: noteOne.userId});
            }).toThrow();
        });

    });
}
```

-
-