הנדסת תוכנה

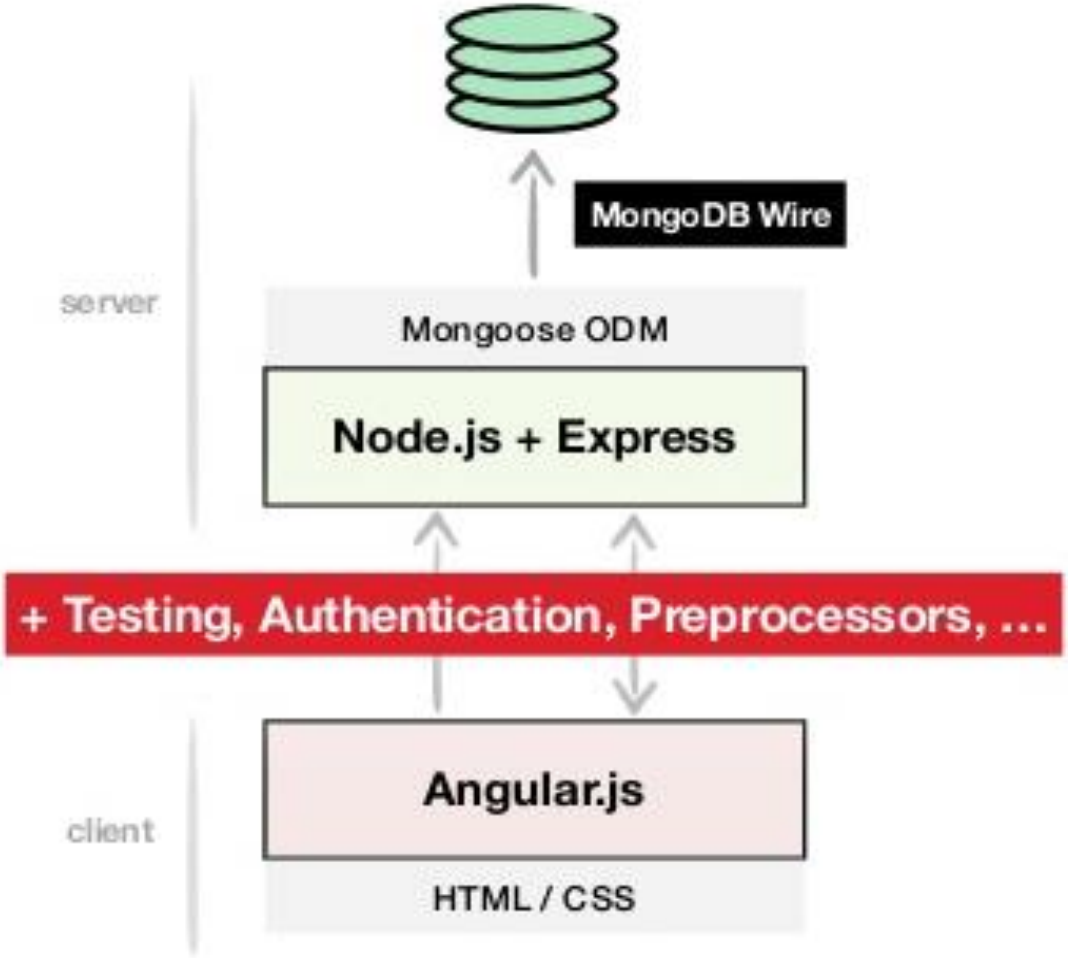
פיתוח צד שרת

הדר פיקאלי

תשע"ז - 2016

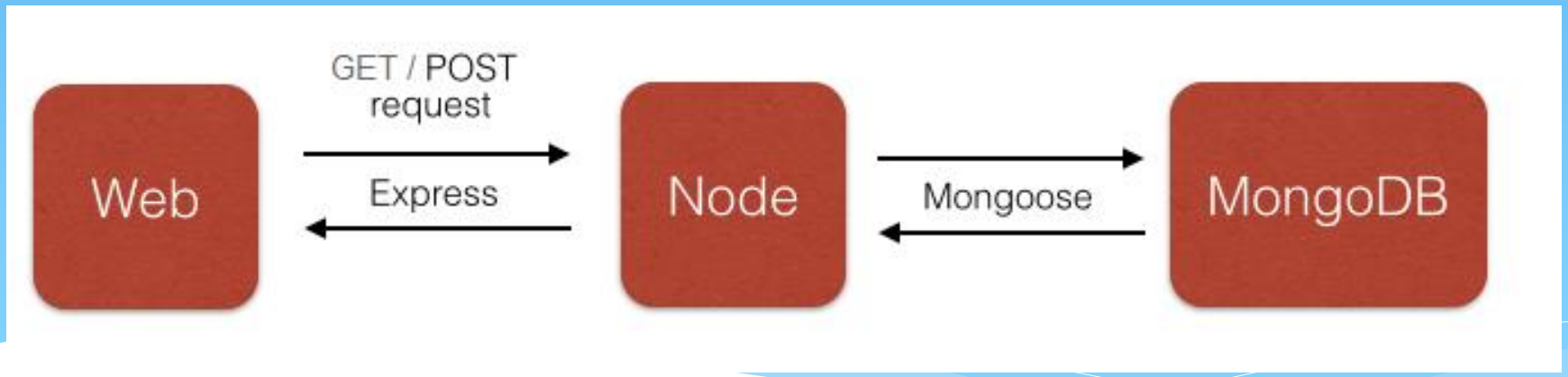
MEAN Stack


mongoDB
express
ANGULARJS
by Google
node
js


MongoDB Wire
server
Mongoose ODM
Node.js + Express
+ Testing, Authentication, Preprocessors, ...
client
Angular.js
HTML / CSS

Node and MongoDB

* יש מספר מודולים של Node.js שמטפלים בחיבור ל- MongoDB
* שניים מהם מאוד פופולריים:
  * הראשון הוא node-mongodb-native, הדרייבר הטבעי של Node.js ל-MongoDB
  * השני הוא Mongoose שיושב בעצם על גבי MongoDB Native ומבצע גם סוג של ORM



mongodb-native - תזכורת

```
var MongoClient = require('mongodb').MongoClient;

// Connect to the db
 MongoClient.connect("mongodb://localhost:27017/test", function(err, db) {

    if(err) { return console.log(err); } //handling errors
    console.log("connected!”);

    var collection = db.collection('users'); //selecting the collection

});
```

mongoDB

מבוא Mongoose

* Mongoose is a Mongo Based Object Document Modeling DB
* Its based on MongoDb
* We will be using it, as it is easier to develop in, and has better docs.
* Docs: http://mongoosejs.com/index.html

* Mongoose מתרגמם את הנתונים ב-DB לאובייקטי JavaScript לשימוש באפליקציה שלנו
* מסייע במידול הנתונים – מספק סכמה ועוד

mongoose

How to use it

* First we need to install it:

```
npm install mongoose --save
```

* Then we need to require it in the code:

```
var mongoose = require('mongoose');
```

* To connect to mongo backend –

```
mongoose.connect('mongodb://localhost/test');
```

http://mongoosejs.com/docs/index.html

mongoose

Mongoose Connecting to MongoDB

```
var mongoose = require('mongoose');

// Connect to the db
var db = mongoose.connection;
db.on('error', console.error);

db.once('open', function() {
        // we're connected! Create your schemas and models here
});

mongoose.connect('mongodb://localhost/test');
```

mongoDB

הגדרות בסיסיות

* Model - זהו האובייקט שמייצג את האוסף במונגו
* Schema - יש לנו יכולת להגדיר מבנה של מסמכים באוסף

```
var movieSchema = new mongoose.Schema ({
        title: { type: String } ,
        rating: String ,
        releaseYear: Number ,
        hasCreditCookie: Boolean
});

// Compile a 'Movie' model using the movieSchema as the structure.
// Mongoose also creates a MongoDB collection called 'Movies' for these documents.
var Movie = mongoose.model(' Movie', movieSchema);
```

mongoose

Create, Read, Update, and Delete (CRUD)

```
var thor = new  Movie ({
        title: 'Thor' ,
        rating: 'PG-13' ,
        releaseYear: '2011',
        // Notice the use of a String rather than a Number - Mongoose will
        automatically convert this for us.
        hasCreditCookie: true
});

thor.save(function(err, thor) {
        if (err) return console.error(err);
        console.log(thor);
});
```

```
{
  "title": "Thor",
  "rating": "PG-13",
  "releaseYear": 2011,
  "hasCreditCookie": true,
  "_id": "519a979e918d4d0000000001",
  "__v": 0
}
```

mongoose

Create, Read, Update, and Delete (CRUD)

```
// Find a single movie by name.
Movie.findOne({ title: 'Thor' }, function(err, thor) {
        if (err) return console.log(err);
        console.log(thor);
});

// Find all movies.
Movie.find(function(err, movies) {
        if (err) return console.log(err);
        console.log(movies);
});

// Find all movies that have a credit cookie.
Movie.find({ hasCreditCookie: true }, function(err, movies) {
        if (err) return console.log(err);
        console.log(movies);
});
```

mongoose

Create, Read, Update, and Delete (CRUD)

Mongoose גם מאפשר לנו ליצור בקלות פונקציות עזר סטטיות למציאת נתונים

```
movieSchema.statics.findAllWithCreditCookies = function(callback) {
        return this.find({ hasCreditCookie: true }, callback);
};

// Use the helper as a static function of the compiled Movie model.
Movie.findAllWithCreditCookies(function(err, movies) {
        if (err) return console.error(err);
        console.dir(movies);
});
```

mongoose

Model.find(conditions, [projection], [options], [callback])

* המקבילה לחיפוש במונגו שראינו.
* מקבלת אובייקט תנאים – conditions
* מקבלת אובייקט החזרה - projection
* מקבלת אובייקט אפשרויות נוספות
* מקבלת פונקציית callback להתמודדות עם הצלחה או כשלון

```
MyModel.find({ name: 'john', age: { $gte: 18 }}); // named john and at least 18

// executes immediately, passing results to callback
MyModel.find({ name: 'john', age: { $gte: 18 }}, function (err, docs) {});

// name LIKE john and only selecting the "name" and "friends" fields, executing immediately
MyModel.find({ name: /john/i }, 'name friends', function (err, docs) { })

// passing options
MyModel.find({ name: /john/i }, null, { skip: 10 })
```

mongoose

Find methods - http://mongoosejs.com/docs/api.html#model_Model.find

שמירה

```
//create model
var Tank = mongoose.model('Tank', yourSchema);

//init the model
var small = new Tank({ size: 'small' });

//save to db
small.save(function (err) {
        if (err) return handleError(err);    // saved!
})
```

mongoose

שמירה על אוסף ממוין

* אנחנו יכולים להגדיר סדר הופעה בתוך אוסף כדי להקל על החיפוש
* את ההגדרה אפשר לעשות ברמת הנתון או ברמת הסכימה

```
var animalSchema = new Schema({
  name: String,
  type: String,
  tags: { type: [String], index: true }     // field level
});

animalSchema.index({ name: 1, type: -1 });     // schema level
```

mongoose

הוספת משתנים וירטואליים

* לפעמים אנחנו רוצים להגדיר משתנים דינאמיים
* המשתנים אינם נשמרים במסמך בפועל
* נרצה להגדיר אותם כדי להקל על עצמנו בעבודה
* למשל נרצה להגדיר משתנה המחזיר שם מלא של בנאדם כאשר במסמך יש משתנים של שם פרטי ושם משפחה

mongoose

הוספת משתנים וירטואליים

```
// define a schema
var personSchema = new Schema({
  name: {
    first: String,
    last: String
  }
});
var Person = mongoose.model('Person', personSchema);

// create a document
var bad = new Person({
    name: { first: 'Walter', last: 'White' }
});
```

~~console.log(bad.name.first + ' ' + bad.name.last); // Walter White~~

```
personSchema.virtual('name.full')
.get(function () {
  return this.name.first + ' ' + this.name.last;
});

console.log('%s is insane', bad.name.full); // Walter White is insane
```

mongoose

מחיקה

* כמובן שניתן למחוק מאוסף
* מבנה:

model.remove([condition:Object], Callback);

```
var Tank = mongoose.model('Tank', yourSchema);

Tank.remove({ size: 'large' }, function (err) {
  if (err) return handleError(err);
  // removed!
});
```

* במקרה הזה מחקנו מTank את כל מה שמכיל size: 'large'

mongoose