EXPRESS AND MONGOOSE

CS498RK SUMMER 2017

UNIVERSITY OF ILLINOIS @ URBANA-CHAMPAIGN

EXPRESS JS

express

“A fast, unopinionated, minimalist web framework for Node.js ”

express

lightweight, extensible node module that lets you make web servers with very little code.

EXAMPLE

```
var http = require('http');
var fs = require('fs');
var path = require('path');

http.createServer(function (request, response) {
    console.log('request ', request.url);

    var filePath = '.' + request.url;
    if (filePath == './')
        filePath = './index.html';

    var extname = String(path.extname(filePath)).toLowerCase();
    var contentType = 'text/html';
    contentType = mimeTypes[extname] || 'application/octet-stream';

    fs.readFile(filePath, function(error, content) {
        if (error) {
            if(error.code == 'ENOENT'){
                fs.readFile('./404.html', function(error, content) {
                    response.writeHead(200, { 'Content-Type':
contentType });
                    response.end(content, 'utf-8');
                });
            }
            else {
                response.writeHead(500);
                response.end('Sorry, check with the site admin for
error: '+error.code+' ..\n');
                response.end();
            }
        }
        else {
            response.writeHead(200, { 'Content-Type': contentType });
            response.end(content, 'utf-8');
        }
    });
}).listen(8125);
console.log('Server running at http://127.0.0.1:8125/');
```

Example

```
express = require('express')
app = express()

app.use(express.static(__dirname + '/public'));

app.listen(3000, function () {
  console.log('Example app listening on port 
3000!')
})
```

WHAT DOES IT DO?

- Parses arguments and headers
- Routing
- Views
    - Partials
    - Layouts
- Configuration
- Sessions

FRONTEND

- Parses arguments and headers
- Routing
- Views
    - Partials
    - Layouts

Seen!

MP3 - STARTER

app.js

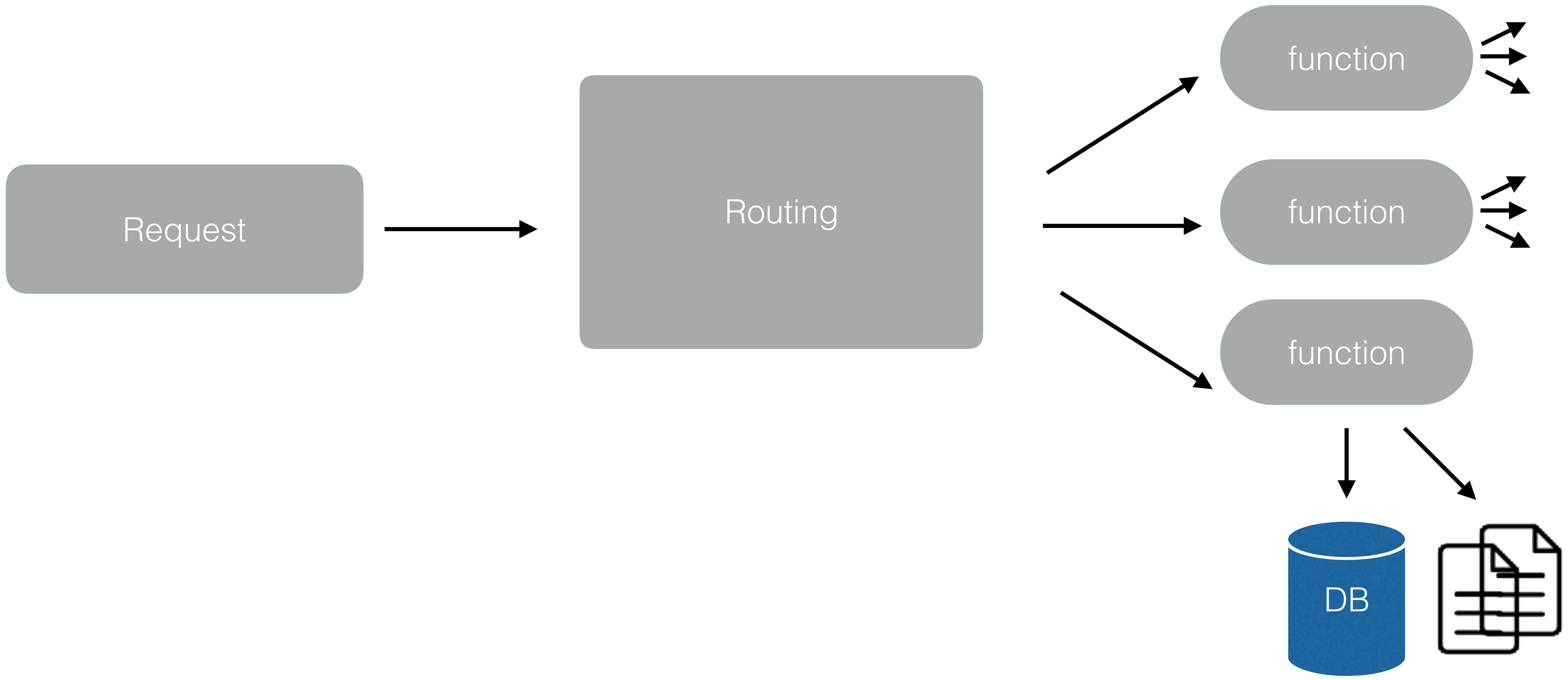
```
var express = require('express');
var app = express();

app.use(express.static(__dirname + '/public'));

var port = process.env.PORT || 3000;
console.log("Express server running on " + port);
app.listen(process.env.PORT || port);
)
```

BACKEND

BASIC ARCHITECTURE



ROUTER

- A way to specify which function should handle requests for each URL

- Chain-able & abstract-able

- You can add route-specific middleware if you want.

ROUTER

server.js

```
var express = require('express');
var app = express();
var router = express.Router();
require('./routes')(app, router);
```

./routes/index.js

```
module.exports = function (app, router) {
  app.use('/api', require(‘./items.js’)(router));
};
```

ROUTER

items.js

```
module.exports = function(router) {
var itmesRoute= router.route('/items/:id');

itmesRoute.get(function(req, res) {
     …
     if (err || user === null)
     {
        res.status(404);
        res.json({"message": "Item not found"});
     }
});

return router

}
```

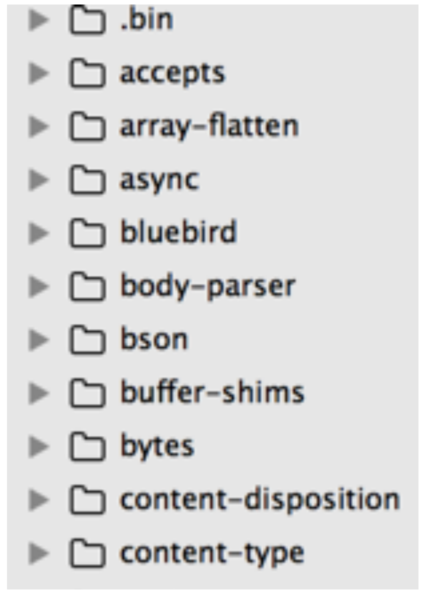
ROUTER

```
itmesRoute.get(function(req, res) {…});
itmesRoute.put(function(req, res) {…});
itmesRoute.post(function(req, res) {…});
itmesRoute.delete(function(req, res) {…});
```

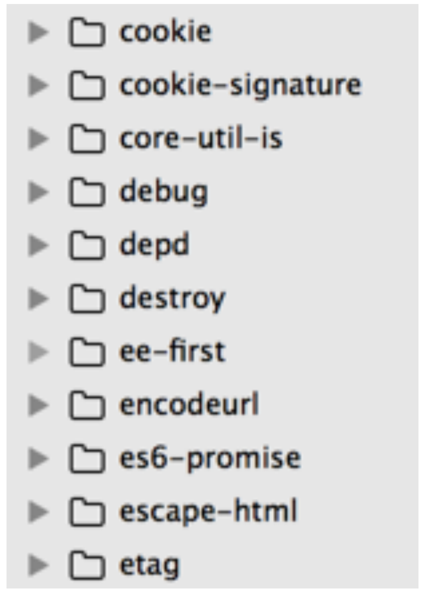
PACKAGES GALORE

- mongoose:
- body-parser:
- lodash:
- underscore: provides a useful functional programming helpers without extending any built-in objects.

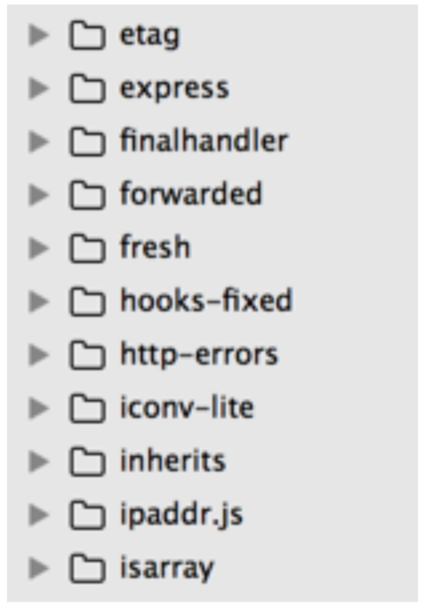
PACKAGES GALORE



```
▶ 🗀 .bin
▶ 🗀 accepts
▶ 🗀 array-flatten
▶ 🗀 async
▶ 🗀 bluebird
▶ 🗀 body-parser
▶ 🗀 bson
▶ 🗀 buffer-shims
▶ 🗀 bytes
▶ 🗀 content-disposition
▶ 🗀 content-type
```



```
▶ 🗀 cookie
▶ 🗀 cookie-signature
▶ 🗀 core-util-is
▶ 🗀 debug
▶ 🗀 depd
▶ 🗀 destroy
▶ 🗀 ee-first
▶ 🗀 encodeurl
▶ 🗀 es6-promise
▶ 🗀 escape-html
▶ 🗀 etag
```



```
▶ 🗀 etag
▶ 🗀 express
▶ 🗀 finalhandler
▶ 🗀 forwarded
▶ 🗀 fresh
▶ 🗀 hooks-fixed
▶ 🗀 http-errors
▶ 🗀 iconv-lite
▶ 🗀 inherits
▶ 🗀 ipaddr.js
▶ 🗀 isarray
```

MONGOOSE

mongoose

Neat wrapper for interacting with MongoDB

Makes it easy(*ish*) to create schemas, validate data, and run DB queries mongoose

“Writing MongoDB validation, casting and business logic boilerplate is a drag. That's why we wrote Mongoose.”

SETUP

```
var mongoose = require('mongoose');
mongoose.connect('mongodb://localhost/test');
var db = mongoose.connection;
db.on('error', console.error.bind(console, 'connection error:’));
//pass the error to the console

db.once('open', function() {
  // connected!
  console.log("Opened");
});
```

SCHEMAS

- set of formulas called integrity constraints imposed on a database

- constraints ensure compatibility between parts of the schema

SCHEMAS

```
var ItemSchema   = new mongoose.Schema({
  name: {
   type: String,
   required: true
  },
  price: {
   type: Number,
   required: true
  }
});
```

next step is compiling our schema into a Model

MODELS

```
var Item = mongoose.model('Item', itemSchema);
```

- a class with which we construct documents.
- each document will be a **Item** with properties, behaviors, and methods as declared in our schema.

MONGOOSE

CRUD

CREATE

```
var item = new Item({name:’iPhone’, price:’600’});

item.save(function(err){
        if(err)
        {…}
        else
        {…}
});
```

CREATE

create object

```
var item = new Item({name:'iPhone', price:'600'});

item.save(function(err){
        if(err)
        {…}
        else
        {…}
});
```

write to db

QUERY

```
Item.findOne({ condition}, {projection}, {options}, function (err, item) {
  if (err) return handleError(err);
  else {…}
 })
```

- chain-able
- order matters

READ

```
Item.findOne({ 'name': 'iPhone' }, 'name price', function (err,
item) {
  if (err) return handleError(err);
  else {…}
 })
```

READ

condition

selection

```
Item.findOne({ 'name': 'iPhone' }, 'name price', function (err,
item) {
  if (err) return handleError(err);
  else {…}
 })
```

callback

UPDATE

```
Item.findByIdAndUpdate(id, { name: 'jason borne' },
options, callback)

// is sent as
Item.findByIdAndUpdate(id, { $set: { name: 'jason
borne' }}, options, callback)
```

DELETE

```
Items.findByIdAndRemove(req.params.id, req.body,
function(err, item)
     {
       if (err)
       {…}

       else{
         res.json({"message": "Deleted item", "data":
item });
       }
});
```

RESOURCES & TIPS

- Read the Mongoose Quick Start guide.
- Learn how Promises work and use them to write your DB code.
- Use the node debugger.
- Use Postman to test your HTTP calls.