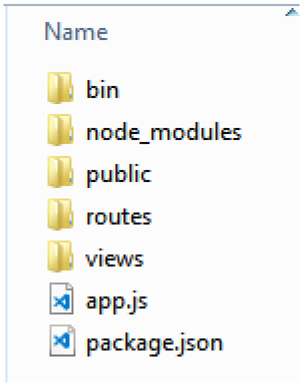
التعليمة	الشرح
Install Express	
npm install express@4.1.2 –g npm i express-generator@4.0.0 -g	تنصيب express تنصيب express-generator للcli
Initialize Express project	
In cmd: express –c styl project-name then: cd project-name && npm install DEBUG=my-application ./bin/www	إنشاء مشروع جديد تنصيب موديولز المشروع تشغيل المشروع على السيرفر

This command will create Express Project with these folders:



```
Name
bin
node_modules
public
routes
views
app.js
package.json
```

app.js: Where is all the middleware and error handler and functions.

package.json: Where is all modules used in this project.

node_modules: Where our modules files.

views folder: Where is the view files related to our 'view engine' like (jade, ejs).

routes folder: Where is all the routes defined and it contains Node.js modules that contain request handlers.

public folder: Where is the static (frontend) files, like (html, css, js).

bin folder: The bash that we used to fire the server.

Created by: Wael Zoaiter - Github

App.js

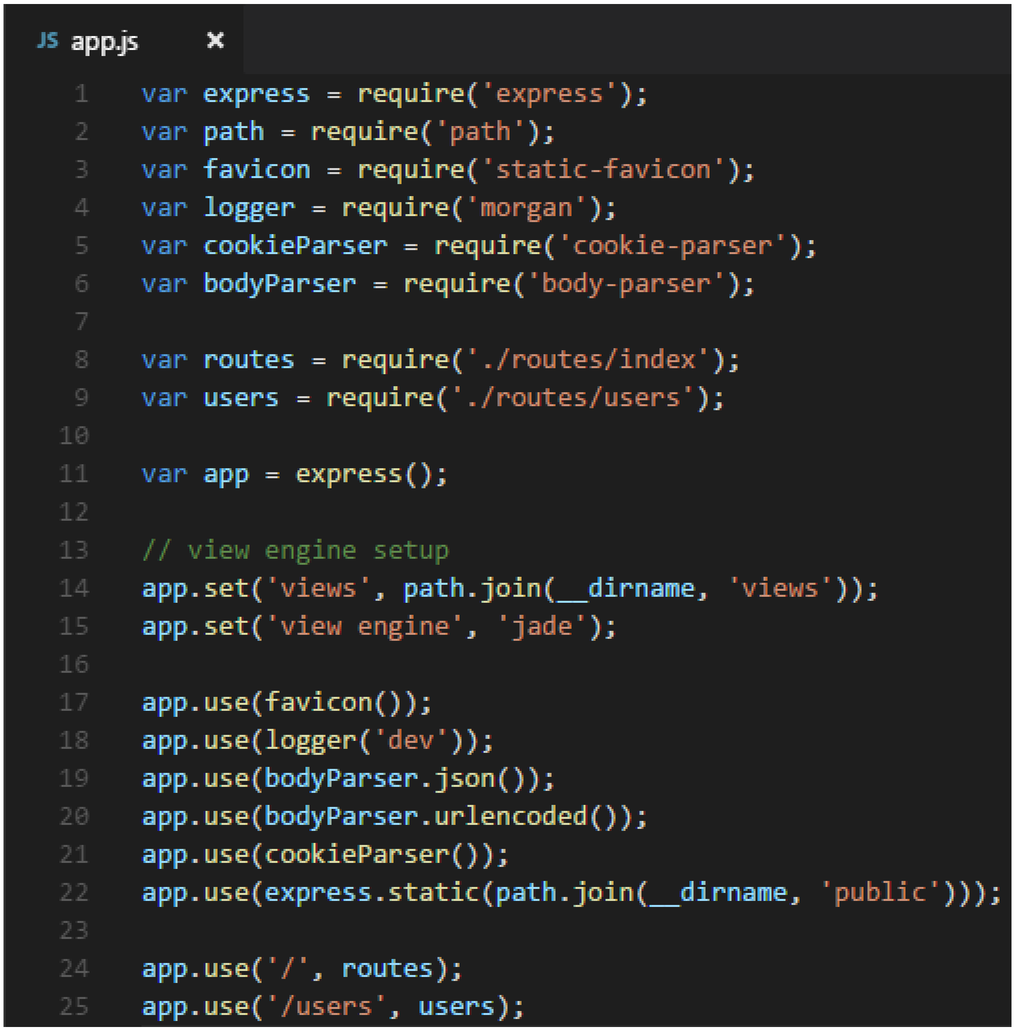
A typical structure of the main Express.js file consists of the following areas:

1. Require dependencies	#1-9
2. Configure settings	#11-15
3. Connect to database (optional)	#---
4. Define middleware	#17-22
5. Define routes	#24-25
6. Start the server	#---
7. Start workers with clusters (optional)	#---

The order here is important, because requests travel from top to bottom in the chain of middleware.

Created by: Wael Zoaiter - Github

Middleware



```
 1  var express = require('express');
 2  var path = require('path');
 3  var favicon = require('static-favicon');
 4  var logger = require('morgan');
 5  var cookieParser = require('cookie-parser');
 6  var bodyParser = require('body-parser');
 7
 8  var routes = require('./routes/index');
 9  var users = require('./routes/users');
10
11  var app = express();
12
13  // view engine setup
14  app.set('views', path.join(__dirname, 'views'));
15  app.set('view engine', 'jade');
16
17  app.use(favicon());
18  app.use(logger('dev'));
19  app.use(bodyParser.json());
20  app.use(bodyParser.urlencoded());
21  app.use(cookieParser());
22  app.use(express.static(path.join(__dirname, 'public')));
23
24  app.use('/', routes);
25  app.use('/users', users);
```

Line #N:
#1-9: require third-party modules that we are going to use in this project.

#11: putting the express functionality in app variable.
the Express.js object is instantiated (Express.js uses a functional pattern)

#14-15: configure Express.js settings is to use app.set(), with the name of the setting and the value.
#14: views: name and path to the folder with template.
#15: view engine: file extension for the template files, like (jade, ejs, html).

#17-25: **The Middlewares**:

#17: favicon():
the icon for our website

#18: logger('dev'):
is tirelessly printing in the terminal pretty logs for each request.
dev:

Concise output colored by response status for development use.
The :status token will be colored red for server error codes, yellow for client error codes, cyan for redirection codes, and uncolored for all other codes.
https://www.npmjs.com/package/morgan

#19-20: bodyParser :
Parse incoming request bodies in a middleware before your handlers, available under the req.body property.
https://www.npmjs.com/package/body-parser

#19: bodyParser.json()
Returns middleware that only parses json and only looks at requests where the Content-Type header matches the type option. This parser accepts any Unicode encoding of the body.

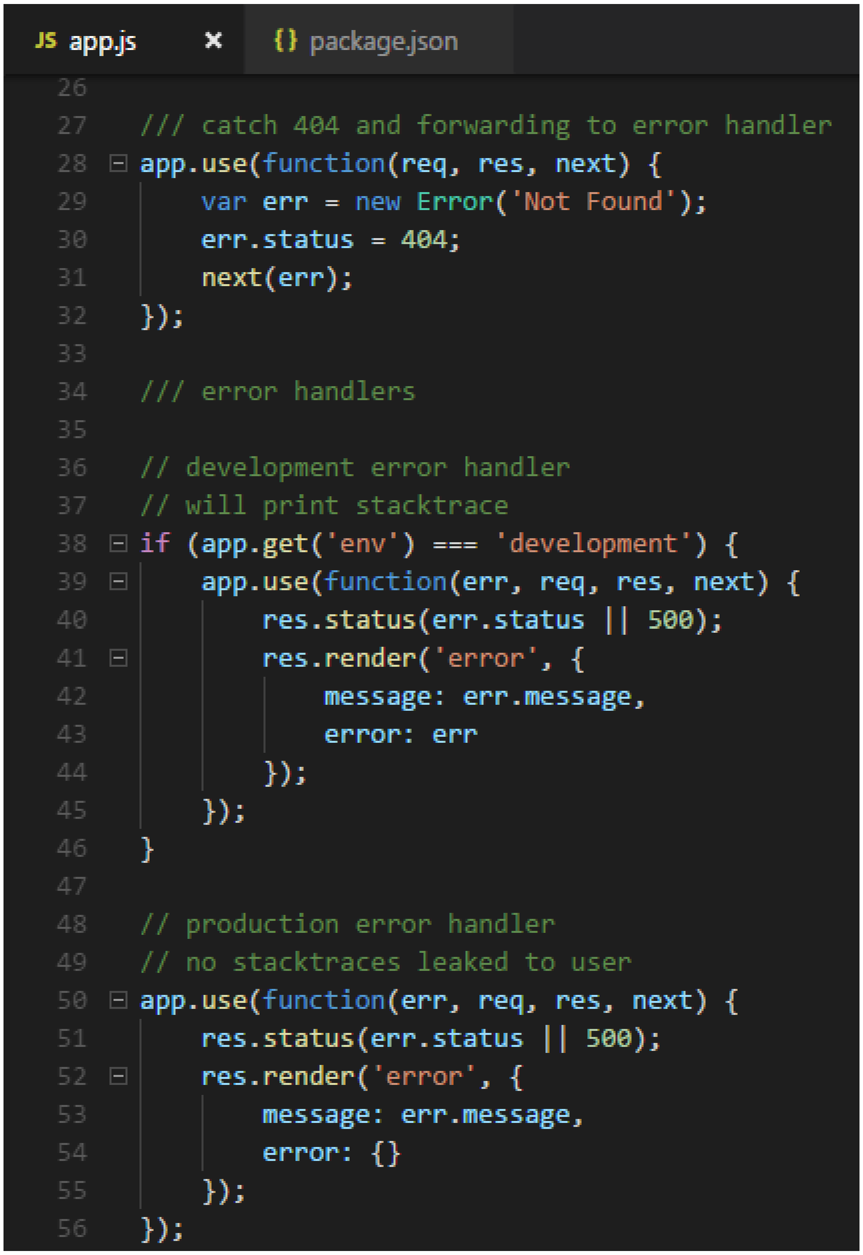
#20: bodyParser.urlencoded()
Returns middleware that only parses urlencoded bodies and only looks at requests where the Content-Type header matches the type option. This parser accepts only UTF-8 encoding of the body

#21: cookieParser()
Parse Cookie header and populate req.cookies with an object keyed by the cookie names.
https://www.npmjs.com/package/cookie-parser

#22: express.static()
Express middleware for Serve static files like (html/css and js).

#24-25: app.use('/', routes) and app.use('/users', users)
Use 'routes' file for serve the main route.
and 'users' file for serve '/users' route.

Error Handler



```
26
27  /// catch 404 and forwarding to error handler
28  app.use(function(req, res, next) {
29      var err = new Error('Not Found');
30      err.status = 404;
31      next(err);
32  });
33
34  /// error handlers
35
36  // development error handler
37  // will print stacktrace
38  if (app.get('env') === 'development') {
39      app.use(function(err, req, res, next) {
40          res.status(err.status || 500);
41          res.render('error', {
42              message: err.message,
43              error: err
44          });
45      });
46  }
47
48  // production error handler
49  // no stacktraces leaked to user
50  app.use(function(err, req, res, next) {
51      res.status(err.status || 500);
52      res.render('error', {
53          message: err.message,
54          error: {}
55      });
56  });
```

#28: Catch the error and pass it to error handler
#38: Error handler for the error in the development Environment, it prints:
#42: the error message
#43: stacktrace
#50: Production error handler, it prints:
#53: the error message
#54: didn't print stacktraces to the user

General Note

process.evn.PORT:

the port number provided in the environmental variables (env vars).

Middleware types:

- Defined in external module, like(app.use(bodyParser.json()).
- Defined in the app, like (app.use(function(req, res, next){…}).

Catch Requests:

a single route is used to catch requests of all methods on all URLs (* wildcard):

e.g. app.all('*', function(req, res) {...});

res.render():

res.render(viewName, data, callback(error, html)) where parameters mean following:

- viewName: a template name with filename extension or if view engine is set without the extension, e.g. 'index'
- data: an optional object that is passed as locals; for example, to use msg in Jade, we need to have {msg: "..."}
- callback: an optional function that is called with an error and HTML when the compilation is complete

e.g. res.render('index', {msg: 'Welcome to the Practical Node.js!'});

if res.render() invoked it calls res.end() which end the response.

Create Server:

```
http.createServer(app)
.listen(app.get('port'),function(){
        console.log('Express.js server listening on port ' + app.get('port'));
});
```

Jade:

Is template engine that allows developers to type less code and execute almost all JavaScript functions.

Example:

```
h1 hello
p Welcome to the Practical Node.js!

p= msg
```

where 'msg' is a variable we pass it from the app.

scaffolding is a (command-line tool)

Functions

```
app.get(‘/’,function(req,res){
        res.send(‘<h1>Hello</h1>’);
        res.sendFile(__dirname + ‘/index.html’);


});
```

Created by: Wael Zoaiter - Github