



Node and Express: Best Friends Forever!

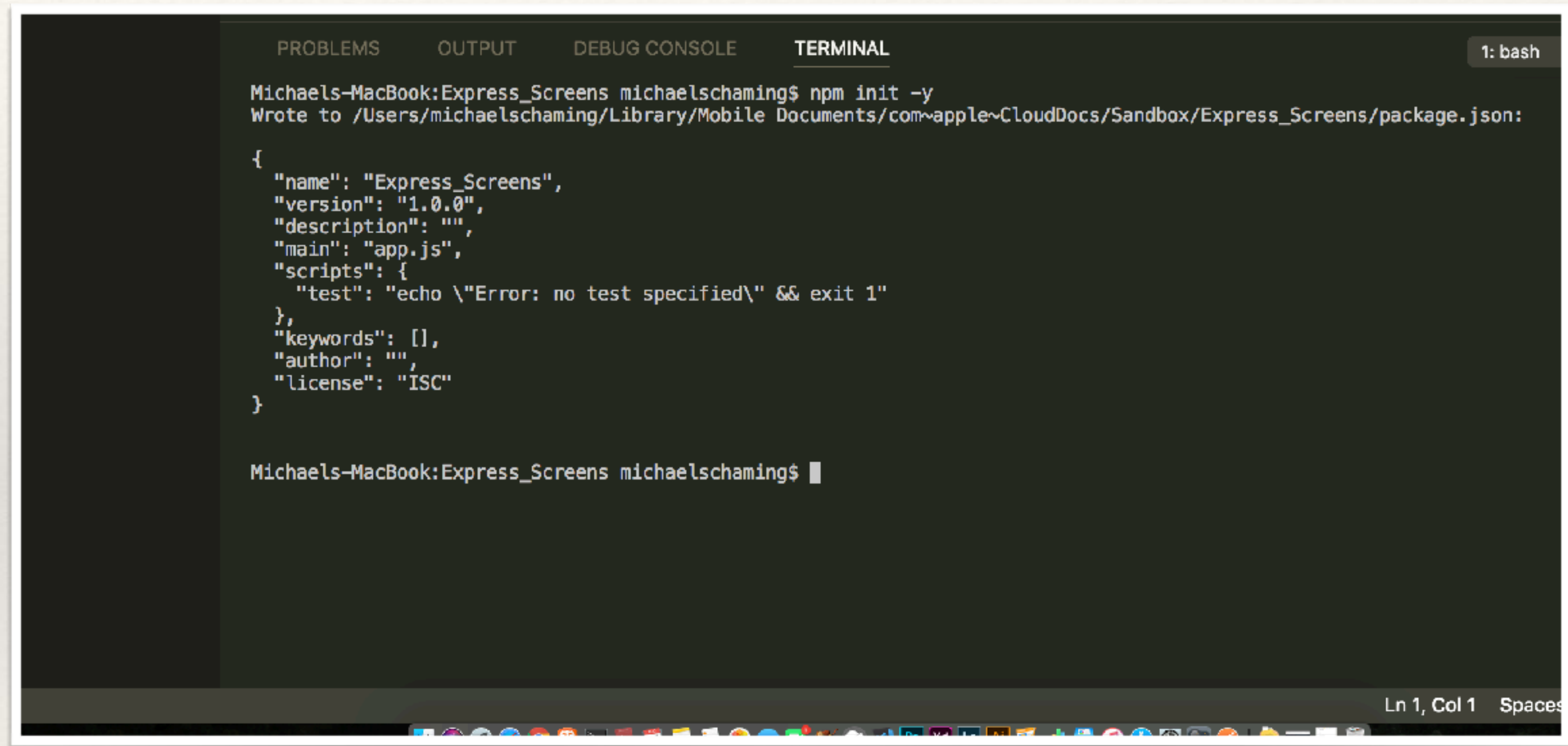
With a Special Appearance by Pug!

What is Express?

- ❖ A popular Node.js framework
- ❖ Build applications and get up running quickly
- ❖ Provides built in methods to work with HTTP requests at your specified route
- ❖ Use data in conjunction with a “view” rendering engine to send HTML back to the client
- ❖ Middleware provides flexibility to process additional requests at any point in your the route.

Getting Started

- ❖ Node.js
- ❖ npm init : to initialize your package.json*
- ❖ npm install express - - save
- ❖ nodemon (optional)



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: bash
Michaels-MacBook:Express_Screens michaelschaming$ npm init -y
Wrote to /Users/michaelschaming/Library/Mobile Documents/com~apple~CloudDocs/Sandbox/Express_Screens/package.json:

{
  "name": "Express_Screens",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}

Michaels-MacBook:Express_Screens michaelschaming$
```


JS app.js



```
1  const express = require('express');
2
3  const app = express();
4
5
6
7  app.listen(3000, () =>{
8    console.log('Welcome! Server up and running at port 3000!')
9  });
10
11
```

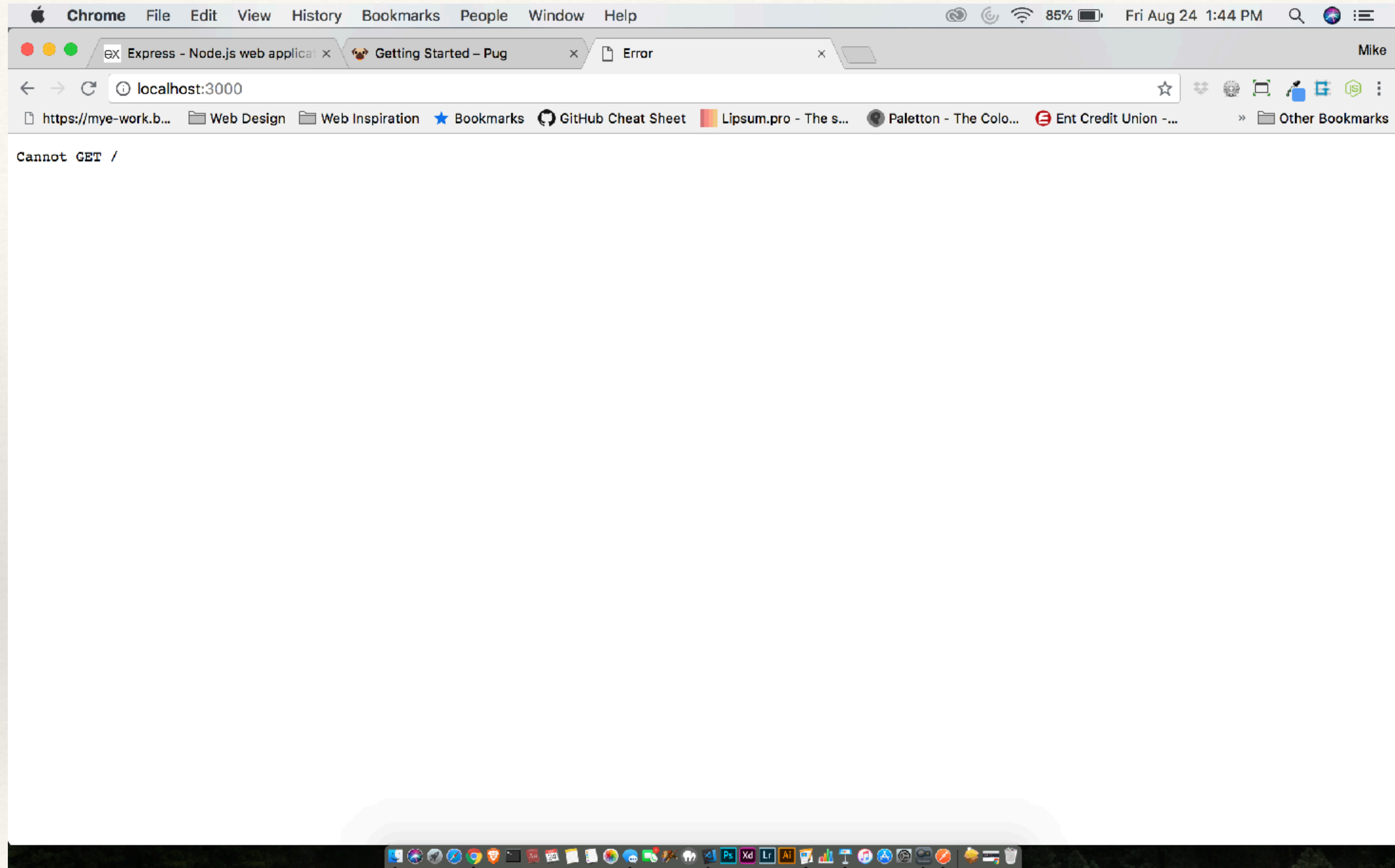
PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

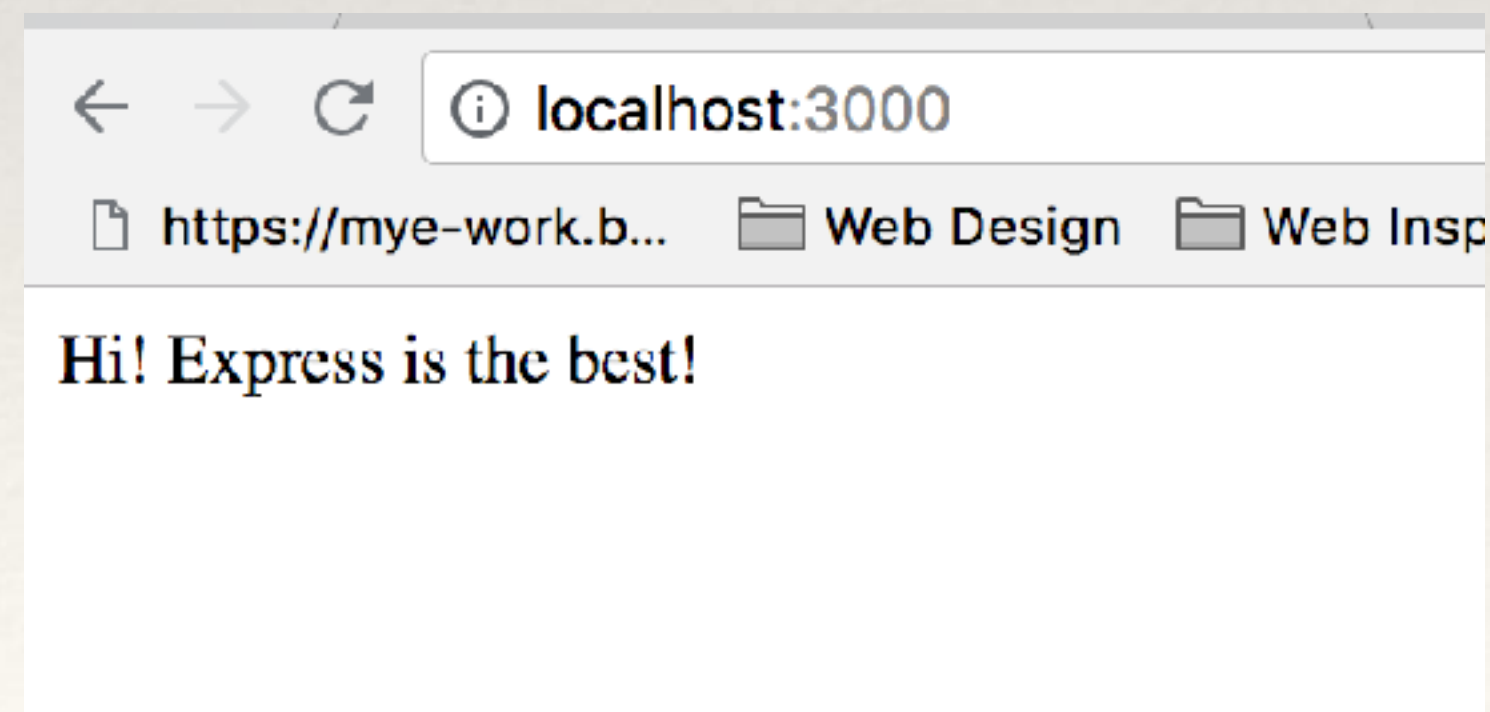
```
Michael's-MacBook:Express_Screens michaelshaming$ nodemon
[nodemon] 1.18.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
Welcome! Server up and running at port 3000!
█
```



Routing Basics

- ❖ Control how the app responds to a client's request at the route you specify
- ❖ Syntax: `app.HTTPmethod('/myRoute', (req, res) => { *code goes here});`

```
app.get('/', (req, res) => {  
  res.send('Hi! Express is the best!');  
});
```



The Response Object

Common Response Object properties :

`res.send();`

`res.sendFile();`

`res.json();`

`res.redirect();`

`res.locals();`

`res.render();`

Pug

- ❖ Pug is a popular HTML template engines
- ❖ It creates HTML, or “views”, based on data you inject into it
- ❖ Able to render HTML with conditionals and loops
- ❖ Uses a short and simple format
- ❖ Other popular template engines include EJS, Mustache, Handlebars.



Getting Started with Pug

- ❖ `npm install pug - - save`

```
"dependencies": {  
  "express": "^4.16.3",  
  "pug": "^2.0.3"  
}
```

- ❖ use `app.set()` to set your view engine

```
app.set('view engine', 'pug');
```

- ❖ Create a folder named “views” to store your .pug files.

▶ node_modules

◀ views

🐶 layout.pug

JS app.js

Pug Format and Features

Format:

- ❖ use `res.render();`
- ❖ no closing tags needed
- ❖ add attributes with `()`
- ❖ `# {}` or `=` to add variables
- ❖ auto divs
- ❖ format with spaces or tabs
***be consistent!**

Pug is Modular:

- ❖ Important Keywords:
 - ❖ `block`
 - ❖ `extends`
 - ❖ `include`

Pug Demo

Modular Routes with Router()

- ❖ Router() object lets you modularize your routes
- ❖ acts as a mini-app where you can store routes and middleware for better organization and maintenance.

Router Setup

Files

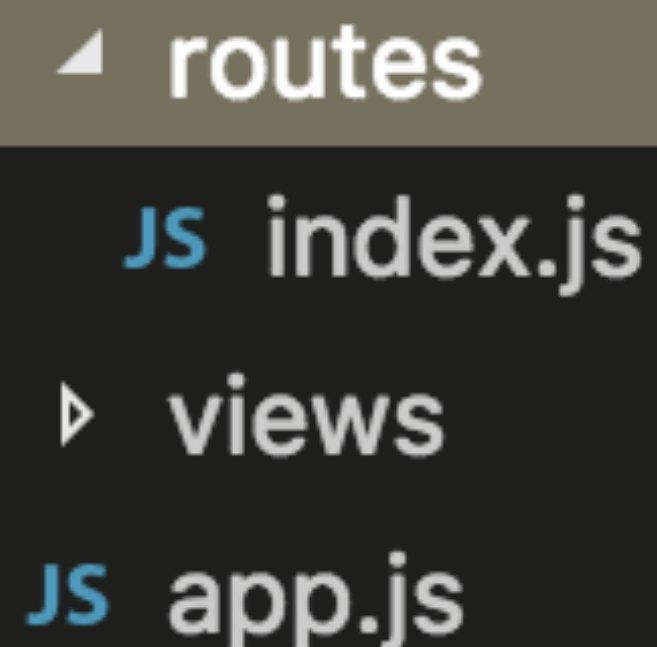
- ❖ Create new folder “routes” with file “index.js”
- ❖ Express automatically looks for files named “index”

app.js

- ❖ `const routes = require('./routes')`
- ❖ `app.use(routes);`

index.js

- ❖ `require('express');`
- ❖ `const router = express.Router();`
- ❖ change all “app.” to “router.”
- ❖ `module.exports = router`



```
└─ routes
   └─ JS index.js
└─ views
   └─ JS app.js
```

```
const routes = require('./routes');
app.use(routes);
```

```
const express = require('express');
const router = express.Router();

router.get('/', (req, res) => {
  res.render('welcome');
});

module.exports = router;
```

Express & The Request Object

- ❖ Works with the incoming HTTP request from the client.
- ❖ Common methods:
 - ❖ `req.body`
 - ❖ `req.params`
- ❖ Will need to use body-parser and cookie-parser to use incoming information.

Installing cookie and body-parser

- ❖ `npm install body-parser - - save`
- ❖ `npm install cookie-parser - - save`
- ❖ use in app.js with:

```
const bodyParser = require('body-parser');  
const cookieParser = require('cookie-parser');  
  
app.use(bodyParser.urlencoded({extended: false}));  
app.use(cookieParser());
```


body-parser

- ❖ body-parser lets you easily work with incoming body content
- ❖ you can attach info to the req.body object and use in your app

without body-parser

```
10 router.post('/', (req, res) => {  
11   console.dir(req.body);  
12   res.render('welcome');
```

PROBLEMS OUTPUT DEBUG CONSOLE

Michaels-MacBook:recipe_app michaelschaming
[nodemon] 1.18.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
Hi! Server up and running at port 3000
undefined

with body-parser

```
10 router.post('/', (req, res) => {  
11   console.dir(req.body);  
12   res.render('welcome');  
13 });
```

PROBLEMS OUTPUT DEBUG CONSOLE

Michaels-MacBook:recipe_app michaelschamin
[nodemon] 1.18.3
[nodemon] to restart at any time, enter `rs`
[nodemon] watching: *.*
[nodemon] starting `node app.js`
Hi! Server up and running at port 3000
{ user: 'Mike' }

cookie-parser

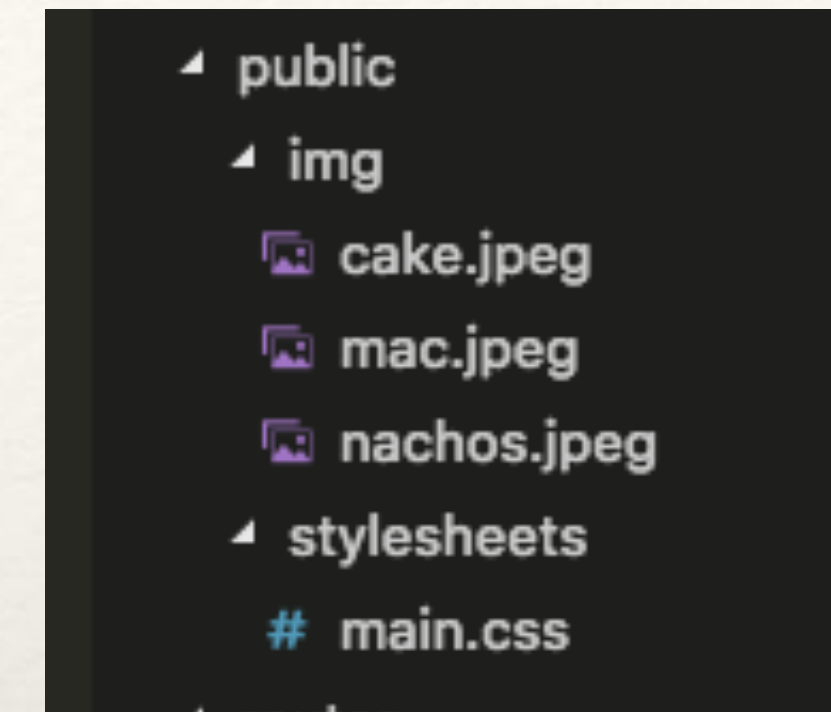
- ❖ like body-parser, we need cookie-parser to work with the Request and Response Objects.
- ❖ properties:
 - ❖ res.cookie('cookie-name', value)
 - ❖ req.cookies.cookieName
 - ❖ res.clearCookie()

```
router.get('/', (req, res) => {  
  res.render('welcome', {name : req.cookies.username});  
});  
  
router.post('/', (req, res) => {  
  res.cookie('username', req.body.user);  
  res.redirect('/');  
});
```

```
router.get('/exit', (req, res) => {  
  res.clearCookie('username');  
  res.redirect('/');  
});
```


Static Files

- ❖ static files are anything that is not processed by your app
 - ❖ CSS
 - ❖ images
 - ❖ client side scripts
- ❖ convention is to store them in a folder named “public”
- ❖ `express.static()` - built in middleware function to serve your static files



```
app.use('/static', express.static('public'));
```

```
html
  head
    link(rel="stylesheet", href="/static/stylesheets/main.css")
  body
    include includes/header.ejs
```


Using Data and Variables In Our Routes

- ❖ inject data into your Express
 - ❖ require your data path
- ❖ req.params is a property that lets you store url specific variables in your routes
- ❖ /: to define a variable in your route

```
data
  {} data.json
```

```
const { data } = require('../data/data.json');
```

```
const { recipes } = data;
```

```
router.get('/recipes/:page', (req, res) => {
  res.render('recipes', {
    name: recipes[req.params.page].name,
    instructions: recipes[req.params.page].instructions,
    ingredients: recipes[req.params.page].ingredients,
    image: recipes[req.params.page].image
  });
});
```


Pug Conditionals and Loops

```
extends layout

block main
  if name
    h2 Welcome, #{name}!
    span
      a(href="/recipes/0") View Recipes
  else
    h2 Welcome!
    form(action="/" method="post")
      label(for="name") Please Enter Your Name
      input(type="text" id="name" name="user" placeholder="Enter Name")
```

```
extends layout

block main
  h2= name
  ul
    each item in ingredients
      li= item
  p= instructions
  img(src=`/static/img/${image}`, alt="food")
```

[GitHub.com/MSchaming](https://github.com/MSchaming)