

Introduction To Full-Stack Web Development

CS 386

Michael Kremer

Last updated: 11/12/2023 3:38:24 PM





- 22.1 Introduction to Express
- 22.2 NPM/Installing Express
- 22.3 Using Express

Class 22

22.1 Introduction to Express

- Time to pause, summary:
 - ❑ HTML5 (content, structure)
 - ❑ CSS (presentation)
 - ❑ Core JavaScript (language elements, syntax)
 - ❑ Client-Side JavaScript (behavioral, dynamic pages)
 - ❑ jQuery (client-side JS: do more with less)
 - ❑ Bootstrap (library for front-end user components)
 - ❑ Node.js (JavaScript server-side)
- What is next?
 - ❑ Express framework for node.js

22.1 Introduction to Express

- Express is node.js web application framework:
 - ❑ Provides broad features for building web and mobile applications
 - ❑ Used to build single page, multipage, and hybrid web applications
 - ❑ Layer built on top of Node.js ecosystem
 - ❑ Helps manage servers and routes
- Features of Express JS:
 - ❑ Fast Server-Side Development :
 - Features of node.js help express saving lot of time
 - ❑ Middleware:
 - Middleware is request handler that has access to application's request-response cycle
 - ❑ Routing:
 - Refers to how application's endpoint's URLs respond to client requests
 - ❑ Templating:
 - Provides templating engines to build dynamic content on web pages by creating HTML templates on server
 - ❑ Debugging:
 - Express makes it easier as it identifies exact part where bugs are

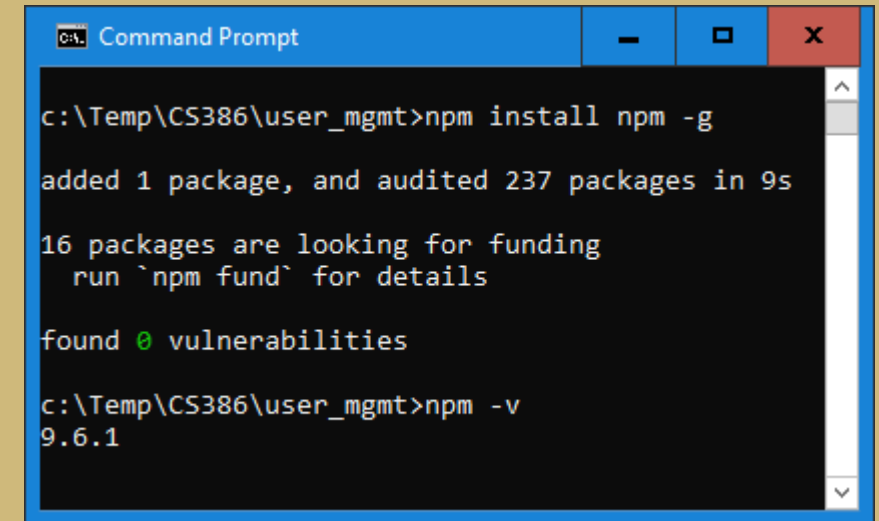
22.2 NPM/Installing Express

Node Package Manager (npm)

- What is npm?
 - ❑ Node Package Manager (NPM) is command line tool that installs, updates or uninstalls Node.js packages in applications
 - ❑ Online repository for open-source Node.js packages
- Node community around world creates useful modules and publishes them as packages in this repository
- NPM is included with Node.js installation
- Version npm:
- Update npm:

Syntax:
npm -v

Syntax:
npm install npm -g



```
C:\Temp\CS386\user_mgmt>npm install npm -g
added 1 package, and audited 237 packages in 9s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Temp\CS386\user_mgmt>npm -v
9.6.1
```

22.2 NPM/Installing Express

Node Package Manager (npm)

➤ Install package local or global

➤ Local:

- ☐ Installed only for current application
- ☐ Must be run under application folder
- ☐ Creates node_modules folder under application folder

➤ Global:

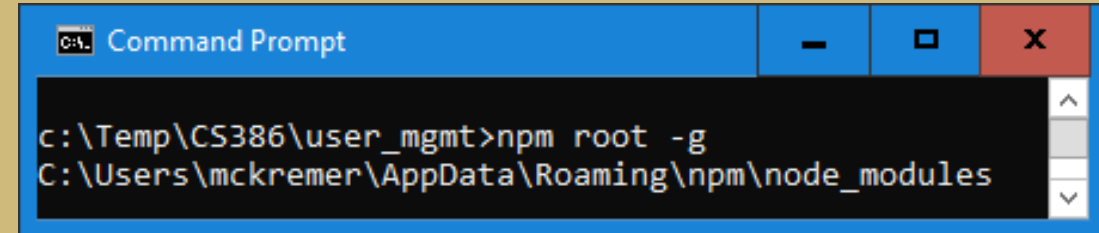
- ☐ Installed on computer for use by any application
- ☐ Check root folder:
 - o npm root -g
 - o Windows: %USERPROFILE%\AppData\Roaming\npm\node_modules
 - o Mac: /usr/local/lib/node_modules

Syntax:

npm install *package_name*

Syntax:

npm install -g *package_name*



```
Command Prompt
c:\Temp\CS386\user_mgmt>npm root -g
C:\Users\mckremer\AppData\Roaming\npm\node_modules
```

22.2 NPM/Installing Express

Node Package Manager (npm)

- Which package will be used: local or global?
 - ☐ Called module resolution
 - ☐ Starts at project/app folder looking for node_modules folder
 - ☐ If not found there, then it moves to the parent directory
 - ☐ And so on, until root of file system is reached
- Update package:
 - ☐ Navigate to app folder (local):
 - ☐ Update global:

Syntax:

npm update *package_name*

Syntax:

npm update -g *package_name*

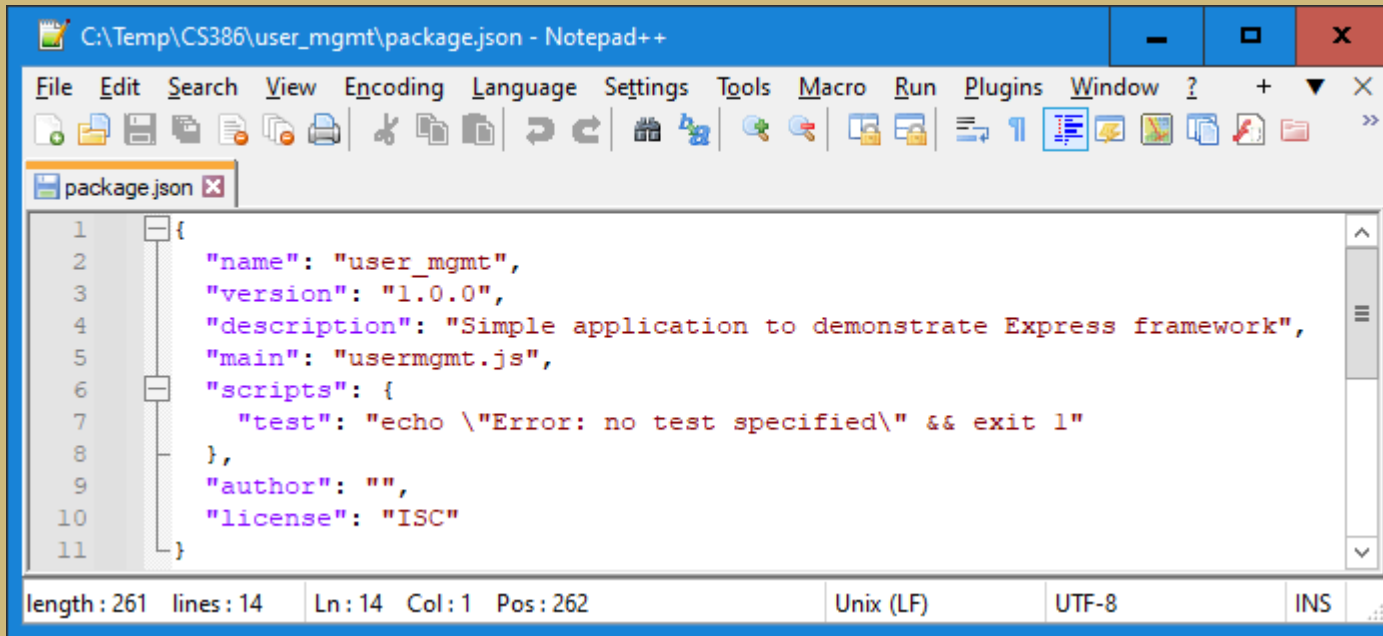
22.2 NPM/Installing Express

Node Package Manager (npm)

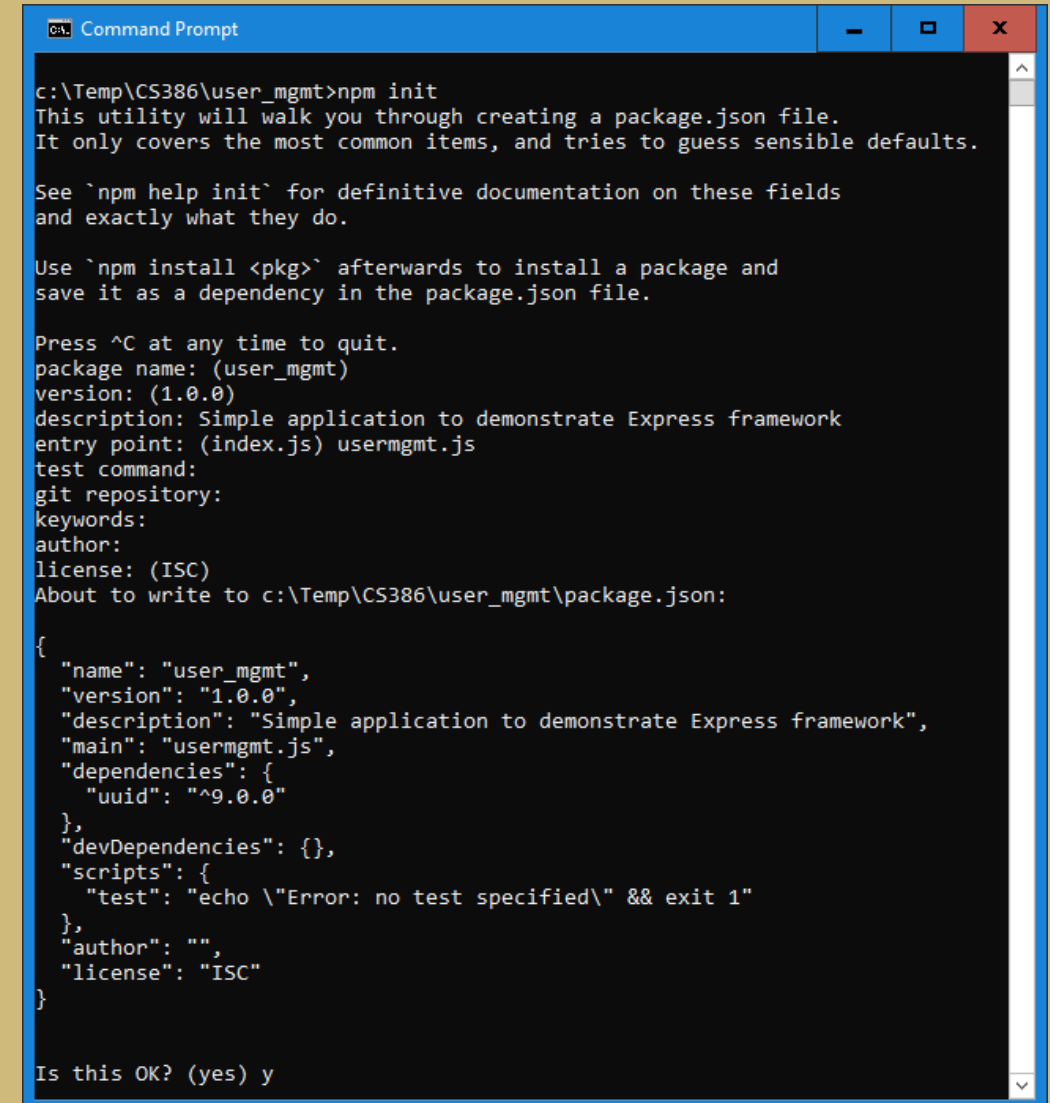
- To initialize project, use npm init
- npm init command is step-by-step tool to build out scaffolding for your project
- Prompt for input on few aspects in following order:
 - ☐ Project's name: Defaults to containing directory name
 - ☐ Project's initial version: 1.0.0 by default.
 - ☐ Project's description: Overview of project
 - ☐ Project's entry point: Meaning main file is to be executed when run
 - ☐ Project's test command: To trigger testing with something like Standard
 - ☐ Project's git repository: Where source code can be found
 - ☐ Project's keywords: Tags related to the project
 - ☐ Project's license: Defaults to ISC, most open-source Node.js projects are MIT

22.2 NPM/Installing Express

- **Example 22-1:**
- Use npm init for user management app
- Create folder user_mgmt under CS386 folder
- Notice package.json file in folder



```
1 {  
2   "name": "user_mgmt",  
3   "version": "1.0.0",  
4   "description": "Simple application to demonstrate Express framework",  
5   "main": "usermgmt.js",  
6   "scripts": {  
7     "test": "echo \"Error: no test specified\" && exit 1"  
8   },  
9   "author": "",  
10  "license": "ISC"  
11 }
```



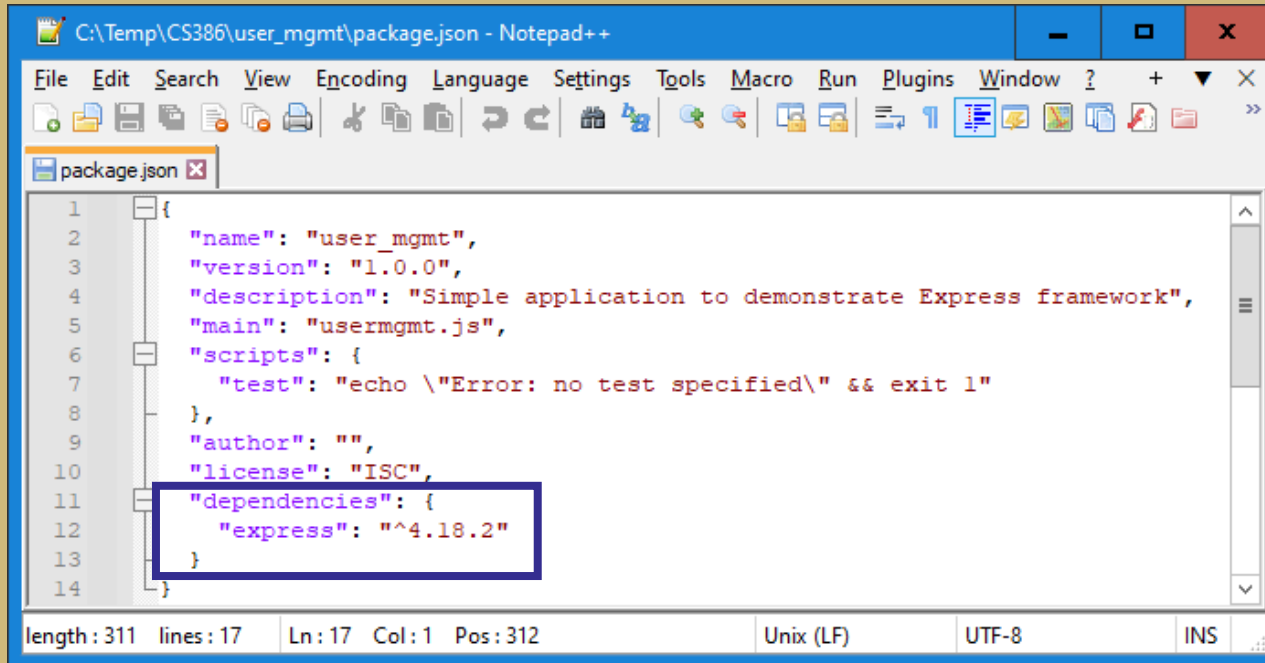
```
C:\Temp\CS386\user_mgmt>npm init  
This utility will walk you through creating a package.json file.  
It only covers the most common items, and tries to guess sensible defaults.  
  
See `npm help init` for definitive documentation on these fields  
and exactly what they do.  
  
Use `npm install <pkg>` afterwards to install a package and  
save it as a dependency in the package.json file.  
  
Press ^C at any time to quit.  
package name: (user_mgmt)  
version: (1.0.0)  
description: Simple application to demonstrate Express framework  
entry point: (index.js) usermgmt.js  
test command:  
git repository:  
keywords:  
author:  
license: (ISC)  
About to write to c:\Temp\CS386\user_mgmt\package.json:  
  
{  
  "name": "user_mgmt",  
  "version": "1.0.0",  
  "description": "Simple application to demonstrate Express framework",  
  "main": "usermgmt.js",  
  "dependencies": {  
    "uuid": "^9.0.0"  
  },  
  "devDependencies": {},  
  "scripts": {  
    "test": "echo \"Error: no test specified\" && exit 1"  
  },  
  "author": "",  
  "license": "ISC"  
}  
  
Is this OK? (yes) y
```

22.2 NPM/Installing Express

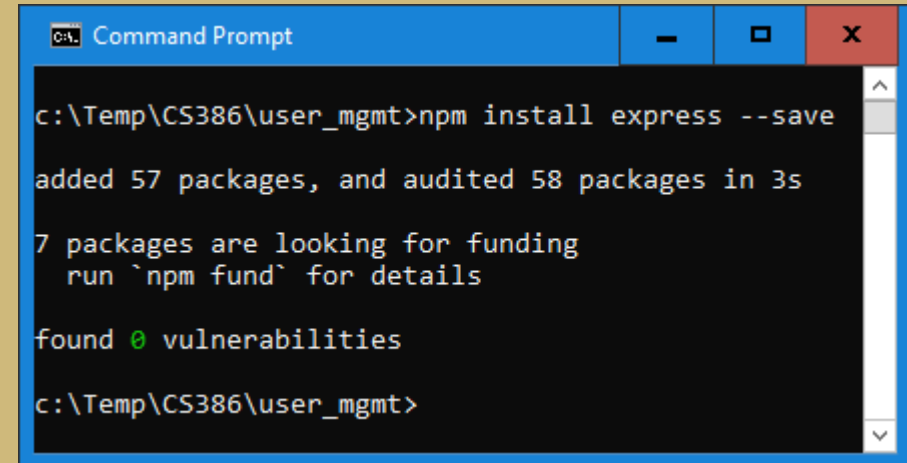
- Example 22-1:
- Next step is to install Express
- Use --save flag to add dependency to package.json file:
- Verify package.json file

Syntax:

npm install *package_name* **--save**



```
1 {  
2   "name": "user_mgmt",  
3   "version": "1.0.0",  
4   "description": "Simple application to demonstrate Express framework",  
5   "main": "usermgmt.js",  
6   "scripts": {  
7     "test": "echo \"Error: no test specified\" && exit 1"  
8   },  
9   "author": "",  
10  "license": "ISC",  
11  "dependencies": {  
12    "express": "^4.18.2"  
13  }  
14 }
```

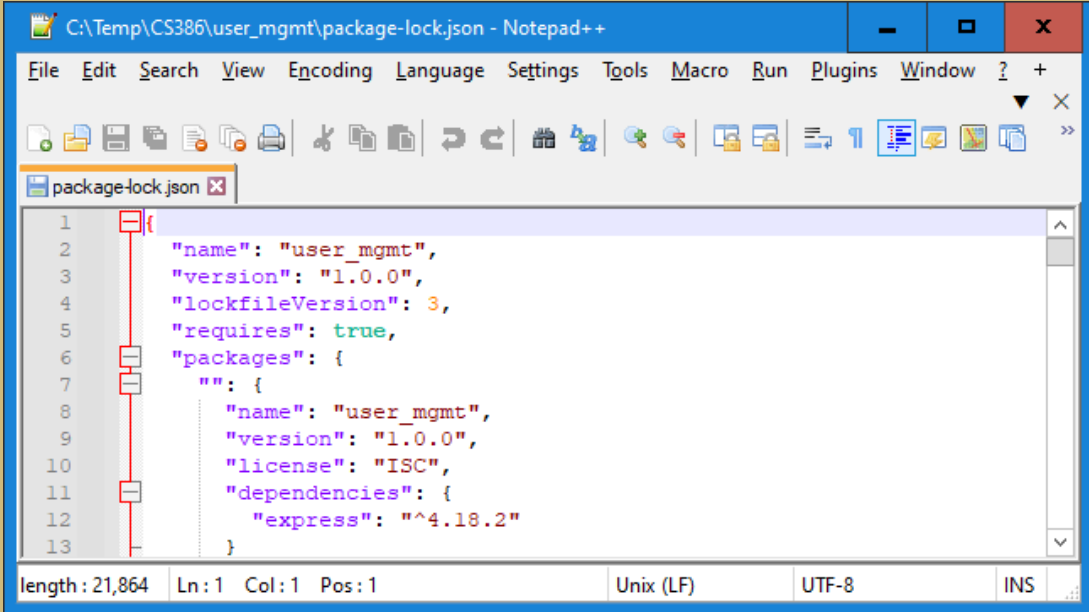


```
C:\Temp\CS386\user_mgmt>npm install express --save  
  
added 57 packages, and audited 58 packages in 3s  
  
7 packages are looking for funding  
  run `npm fund` for details  
  
found 0 vulnerabilities  
  
C:\Temp\CS386\user_mgmt>
```

22.2 NPM/Installing Express

➤ Example 22-1:

- Also notice new file: package-lock.json
- Goal of package-lock.json :
 - ❑ Keep track of exact version of every installed package
- Application is 100% reproducible in same way even if packages are updated by their maintainers
- package.json allows to set which versions to upgrade to (patch or minor), using semver notation (semantic versioning, semver.org):
 - ❑ ~3.13.0 means to only update patch releases: 3.13.1 is ok, but 3.14.0 is not
 - ❑ ^3.13.0 means to update patch and minor releases: 3.13.1, 3.14.0, but not 4.0.1
 - ❑ 3.13.0 means exact version that will be used, always



```
1 {
2   "name": "user_mgmt",
3   "version": "1.0.0",
4   "lockfileVersion": 3,
5   "requires": true,
6   "packages": {
7     "": {
8       "name": "user_mgmt",
9       "version": "1.0.0",
10      "license": "ISC",
11      "dependencies": {
12        "express": "^4.18.2"
13      }
14    }
15  }
```

22.3 Using Express

Basic architecture of Express application

- To use express module need to import it using require
- Create instance of express
- Create one or more routes using http verbs (get, post)
- Create server and listen for connections
- Routing is performed with app variable and methods

Syntax:

```
app.method( path, callback[, callback ...])
```

- Create server:

Syntax:

```
app.listen([port[, host[, backlog]]][, callback])
```

- backlog parameter to specify maximum length of queue of pending connections (default 511)

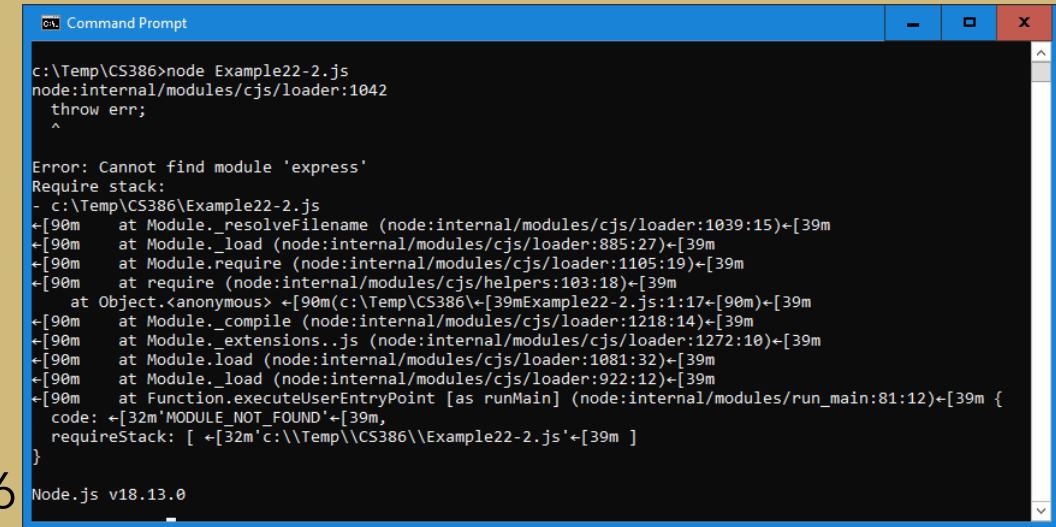
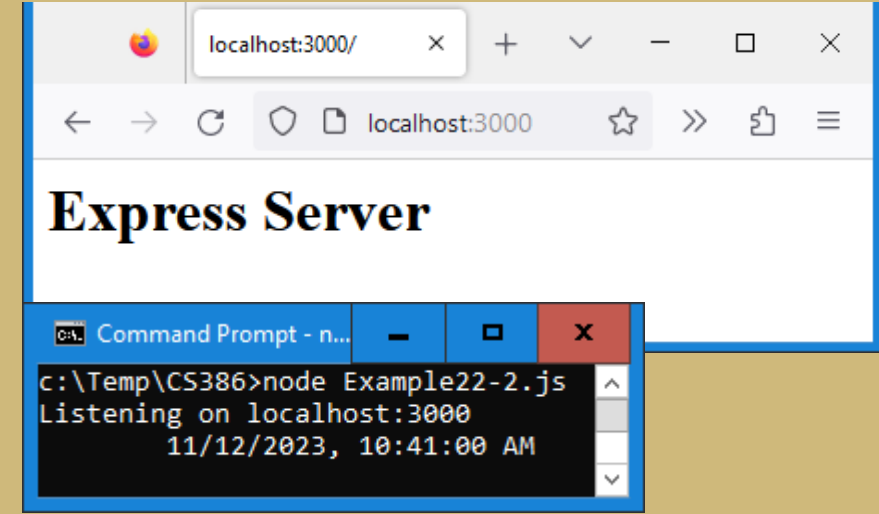
Syntax:

```
const express = require('express');  
const app = express();  
app.get('/', function(req, res) {  
    res.send(content);  
}  
app.listen(port, host , function() {  
    statements  
})
```

22.3 Using Express

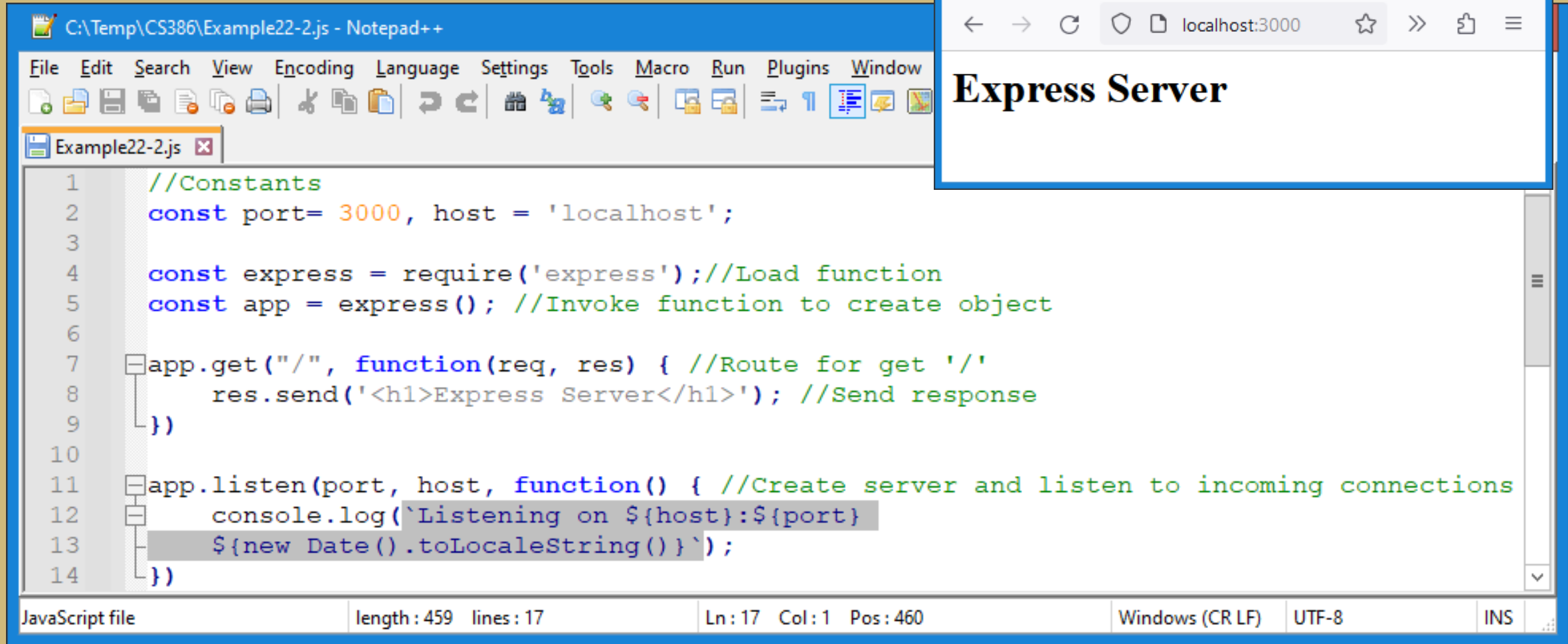
➤ **Example 22-2:**

- Create variable express to load express
- Create constants port (300) and host (localhost)
- Create variable app to instantiate object
- Create route app.get with '/'
- Return h1 header with Express Server
- Listen on port and host
- When run in CS386 folder:
 - ❑ Error message since express is not found!
- Options:
 - ❑ Install express in CS386 folder as well
 - ❑ Move file to user_mgmt folder
 - ❑ Move folder node_modules from user_mgmt to CS386



22.3 Using Express

➤ Example 22-2:



The image shows a web browser window and a code editor side-by-side. The browser window, titled 'localhost:3000/', displays the text 'Express Server'. The code editor, titled 'C:\Temp\CS386\Example22-2.js - Notepad++', contains the following JavaScript code:

```
1 //Constants
2 const port= 3000, host = 'localhost';
3
4 const express = require('express');//Load function
5 const app = express(); //Invoke function to create object
6
7 app.get("/", function(req, res) { //Route for get '/'
8     res.send('<h1>Express Server</h1>'); //Send response
9 })
10
11 app.listen(port, host, function() { //Create server and listen to incoming connections
12     console.log(`Listening on ${host}:${port}
13     ${new Date().toLocaleString()}`);
14 })
```

The status bar at the bottom of the code editor indicates: JavaScript file, length: 459 lines: 17, Ln: 17 Col: 1 Pos: 460, Windows (CR LF), UTF-8, INS.

22.3 Using Express

Express Middleware

- Middleware functions are functions that have access to:
 - ❑ request object (req)
 - ❑ response object (res)
 - ❑ next middleware function
- in application's request-response cycle
- next middleware function is commonly denoted by variable named next
- Middleware functions can perform the following tasks:
 - ❑ Execute any code
 - ❑ Make changes to request and response objects
 - ❑ End request-response cycle
 - ❑ Call next middleware function in stack

22.3 Using Express

Express Middleware

- If current middleware function does not end request-response cycle:
 - ❑ Must call next() to pass control to next middleware function
 - ❑ Otherwise, request will be left hanging
- Express applications can use following types of middleware:
 - ❑ Application-level middleware app.use
 - ❑ Router level middleware router.use
 - ❑ Built-in middleware express.static, express.json, express.urlencoded
 - ❑ Error handling middleware app.use(err, req, res, next)
 - ❑ Third-party middleware bodyparser, cookieparser
- app.use path is used for routing only
- Otherwise leave it out, just use callback

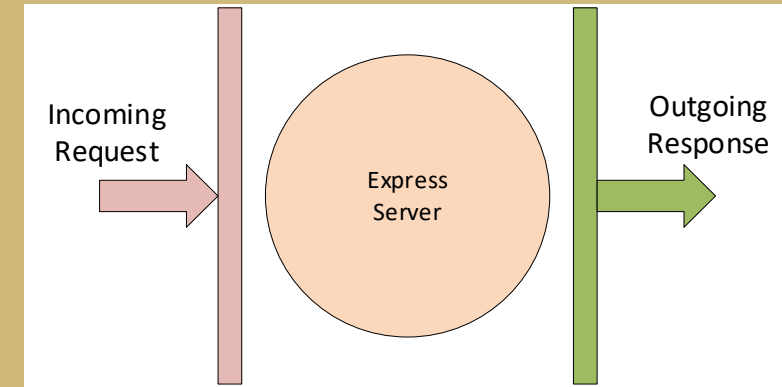
Syntax:

app.use([path], callback[, callback])

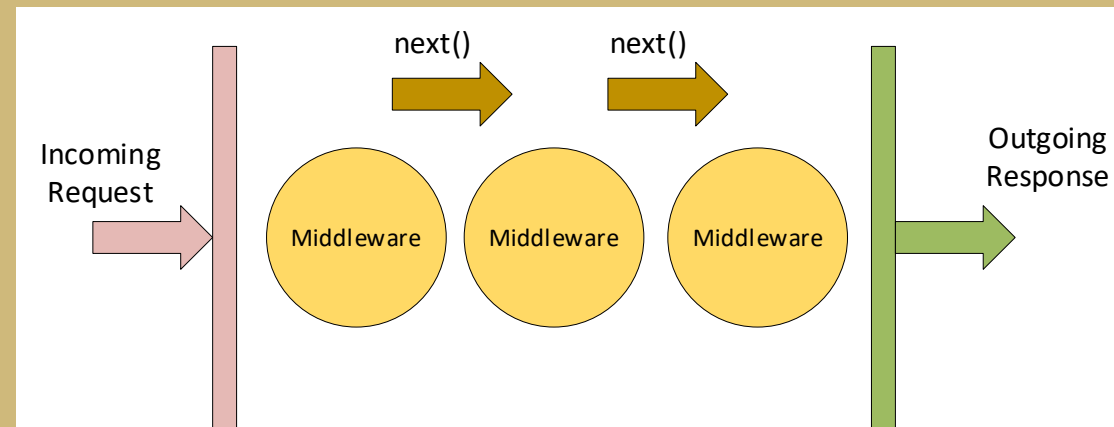
22.3 Using Express

Express Middleware

➤ Request .. Response Cycle



➤ Using Middleware



22.3 Using Express

Callback functions in Express

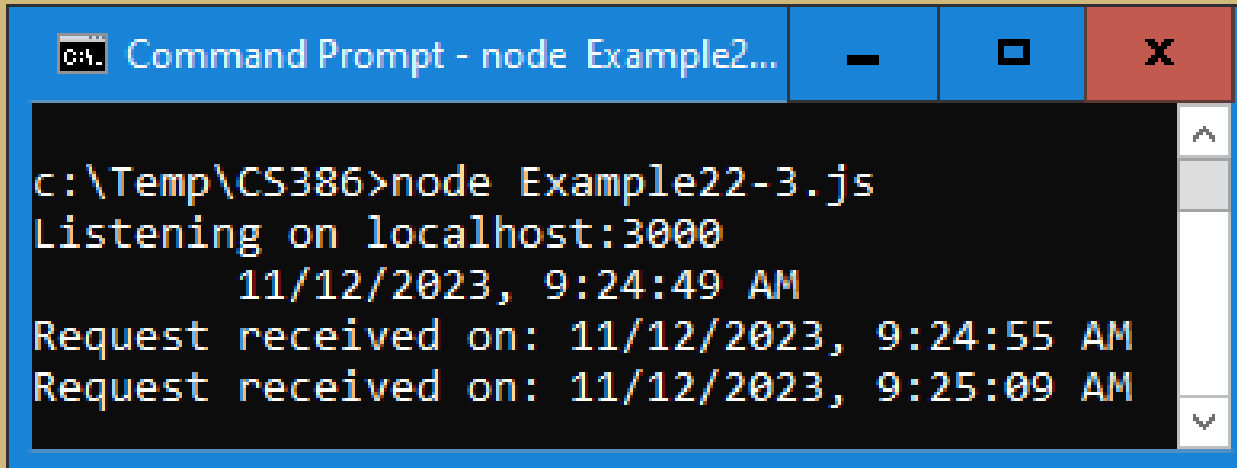
- Like in node itself, Express relies heavily on callback functions
- Callback functions are last parameter in function definition:
 - ❑ `app.listen`
 - ❑ `app.method` (method = get, post, ..)
 - ❑ `app.use`
- Callback function parameters always have error object as first parameter
- Followed by any number of other arguments

Syntax:

`callback_function(err, par1, par2, parn)`

22.3 Using Express

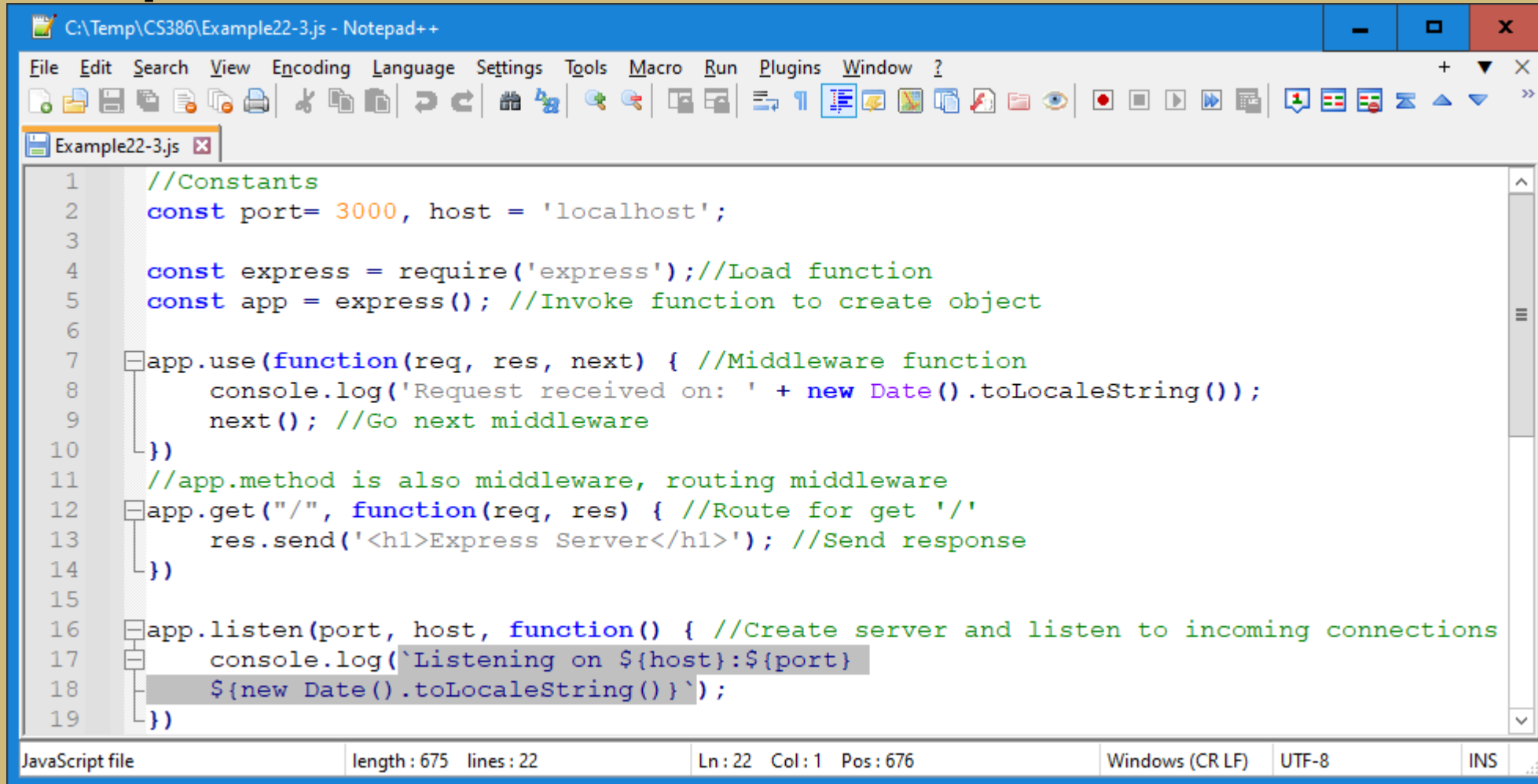
- **Example 22-3:**
- Using simple application middleware to log current date and time
- Clone Example22-2.js to Example22-3.js
- Create middleware before home route using app.use
- In callback function issue current date/time in console
- Call next function to go to next middleware



```
Command Prompt - node Example2...  
c:\Temp\CS386>node Example22-3.js  
Listening on localhost:3000  
11/12/2023, 9:24:49 AM  
Request received on: 11/12/2023, 9:24:55 AM  
Request received on: 11/12/2023, 9:25:09 AM
```

22.3 Using Express

➤ Example 22-3:

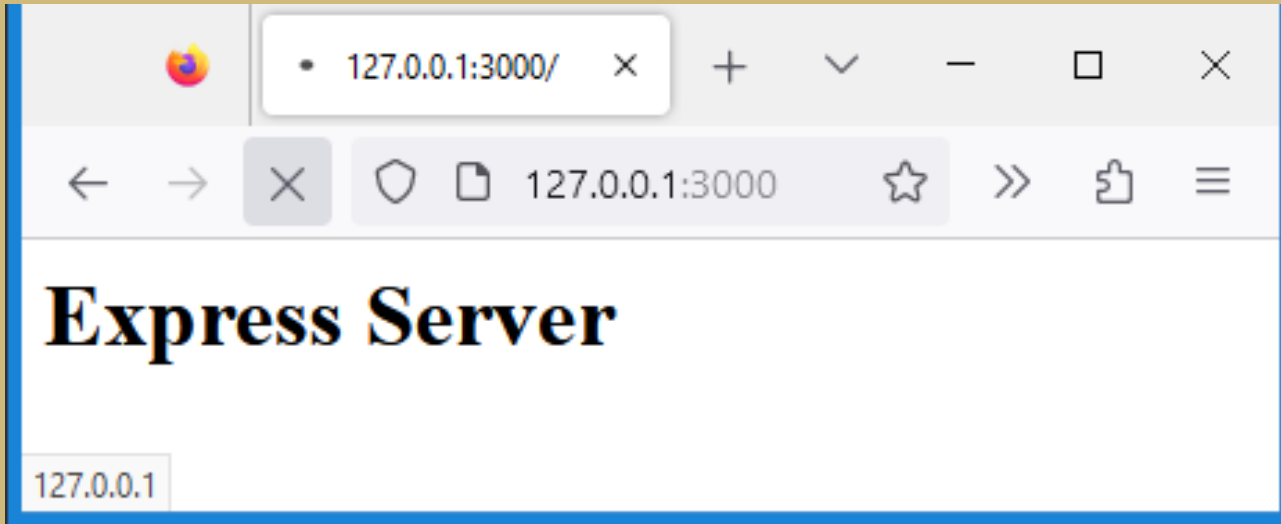


```
1 //Constants
2 const port= 3000, host = 'localhost';
3
4 const express = require('express');//Load function
5 const app = express(); //Invoke function to create object
6
7 app.use(function(req, res, next) { //Middleware function
8     console.log('Request received on: ' + new Date().toLocaleString());
9     next(); //Go next middleware
10 })
11 //app.method is also middleware, routing middleware
12 app.get("/", function(req, res) { //Route for get '/'
13     res.send('<h1>Express Server</h1>'); //Send response
14 })
15
16 app.listen(port, host, function() { //Create server and listen to incoming connections
17     console.log(`Listening on ${host}:${port}
18     ${new Date().toLocaleString()}`);
19 })
```

JavaScript file length : 675 lines : 22 Ln: 22 Col: 1 Pos: 676 Windows (CR LF) UTF-8 INS

22.3 Using Express

- Example 22-3 (continued):
- Comment out next() statement
- Restart server
- Notice application hangs
- Page is not finished loading



22.3 Using Express

Error Handling

- Remember, http error code categories!
 - ❑ Informational responses (100 - 199)
 - ❑ Successful responses (200 - 299)
 - ❑ Redirection messages (300 - 399)
 - ❑ Client error responses (400 - 499)
 - ❑ Server error responses (500 - 599)
- https://en.wikipedia.org/wiki/List_of_HTTP_status_codes
- Implement error handling for:
 - ❑ Invalid routes (Error 404 - not found)
 - ❑ Server error (Error 500 - Internal Server Error)

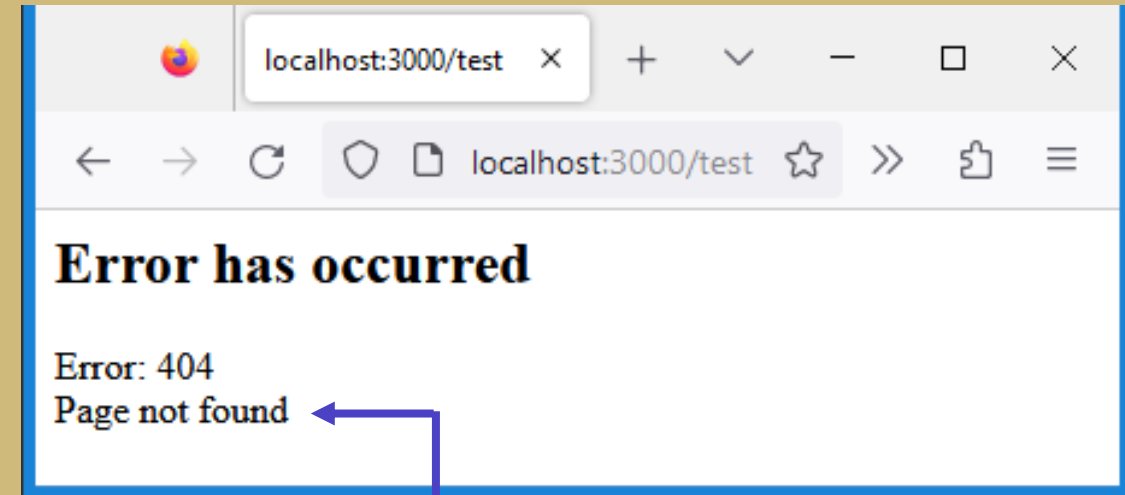
22.3 Using Express

Error Handling

- Add middleware (remember routing is middleware) using `app.use`
- Must come after routing because routing was not matched
- 404 error must come first, path/resource not found
- Followed by 500 error handler

22.3 Using Express

- **Example 22-4:**
- Clone Example22-3.js to Example22-4.js
- 404 Error:
 - ❑ Use app.use middleware using callback
 - ❑ Callback function parameters: req, res, next
 - ❑ Create const error and assign new instance of Error object:
 - Instantiate error object with message (new error(message))
 - Set error code to 404
 - ❑ Use next passing error object into next middleware handler

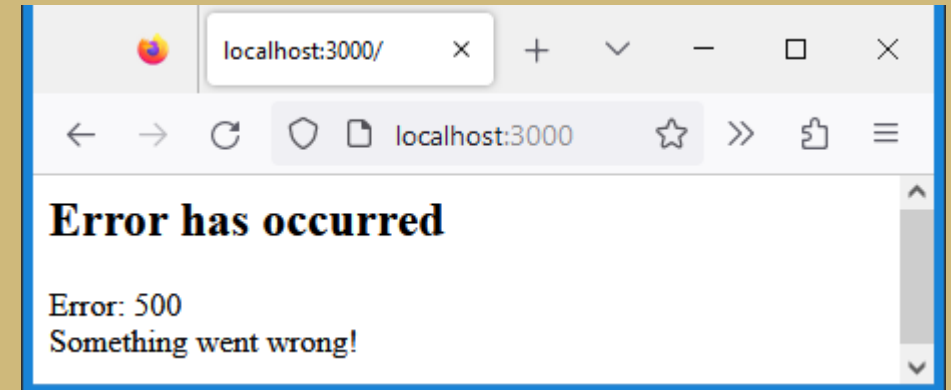


22.3 Using Express

➤ Example 22-4:

➤ 500 Error:

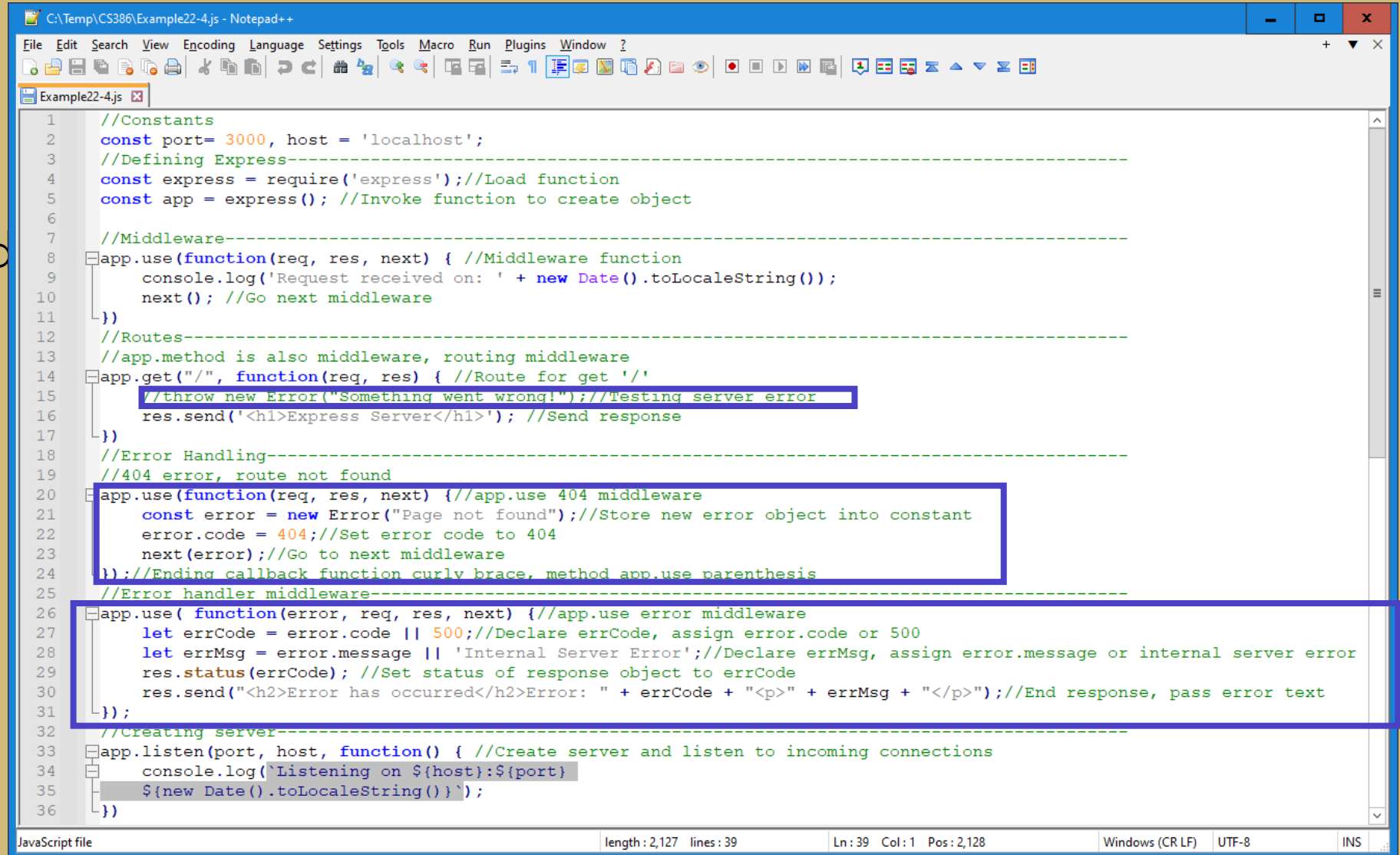
- ❑ Use app.use with callback:
 - First argument is error object, then req, res, next
- ❑ Create variable `errCode`, assign it either `error.code` or 500 (OR short-circuiting behavior)
- ❑ Create variable `errMsg`, assign it either `error.message` or 'Internal Server Error'
- ❑ Set response status passing `errCode` as argument
- ❑ Send response back ending request – response cycle:
 - `h2` header with 'Error has occurred'
 - 'Error: ' concatenated with `errCode`,
 - `errMsg` wrapped in `p` element



22.3 Using Express

➤ Example 22-4:

- Express can distinguish between 404 and 500 handlers by number of arguments of their callback functions



```
1 //Constants
2 const port= 3000, host = 'localhost';
3 //Defining Express-----
4 const express = require('express');//Load function
5 const app = express(); //Invoke function to create object
6
7 //Middleware-----
8 app.use(function(req, res, next) { //Middleware function
9   console.log('Request received on: ' + new Date().toLocaleString());
10  next(); //Go next middleware
11 })
12 //Routes-----
13 //app.method is also middleware, routing middleware
14 app.get("/", function(req, res) { //Route for get '/'
15   //throw new Error("Something went wrong!");//Testing server error
16   res.send('<h1>Express Server</h1>');//Send response
17 })
18 //Error Handling-----
19 //404 error, route not found
20 app.use(function(req, res, next) { //app.use 404 middleware
21   const error = new Error("Page not found");//Store new error object into constant
22   error.code = 404;//Set error code to 404
23   next(error);//Go to next middleware
24 }); //Ending callback function curly brace, method app.use parenthesis
25 //Error handler middleware-----
26 app.use( function(error, req, res, next) { //app.use error middleware
27   let errCode = error.code || 500;//Declare errCode, assign error.code or 500
28   let errMsg = error.message || 'Internal Server Error';//Declare errMsg, assign error.message or internal server error
29   res.status(errCode); //Set status of response object to errCode
30   res.send("<h2>Error has occurred</h2>Error: " + errCode + "<p>" + errMsg + "</p>");//End response, pass error text
31 });
32 //Creating server-----
33 app.listen(port, host, function() { //Create server and listen to incoming connections
34   console.log(`Listening on ${host}:${port}
35   ${new Date().toLocaleString()}`);
36 })
```

22.3 Using Express

Serving Static Files

- Rather than serving HTML string responses serve static HTML files
- Static files are files that clients download as they are from server
- Example:
 - ❑ Use file name as route in url (localhost:3000/myPage.html)
 - ❑ If html page contains <script>, <link>, or other resources → need to also download
 - ❑ Based on get request of initial page
- Need to add route for each of these files and return files
- Too cumbersome and not dynamic
- Therefore, use static middleware to define paths of static files
- Acts like router to retrieve those resources (get requests)

22.3 Using Express

Serving static files

- root:
 - ❑ Root directory from which to serve static assets
- Options:
 - ❑ Specify options in object syntax { prop: value }

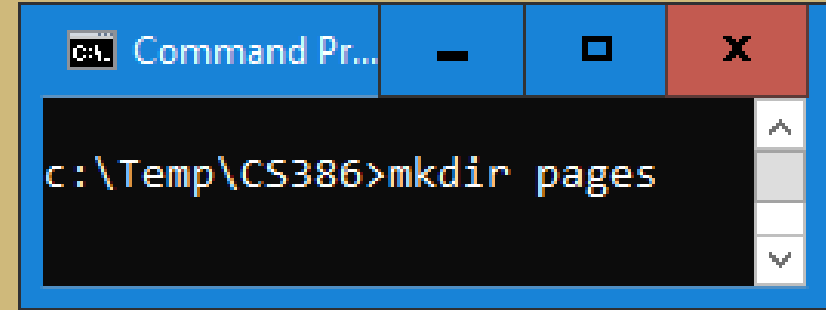
Syntax:

`express.static(root, [options])`

Property	Description	Default
dotfiles	Determines how dotfiles (files or directories that begin with a dot ".") are treated.	"ignore"
etag	Enable or disable etag generation	true
extensions	Sets file extension fallbacks: If a file is not found, search for files with the specified extensions and serve the first one found. Example: ['html', 'htm'].	false
fallthrough	Let client errors fall-through as unhandled requests, otherwise forward a client error.	
	Boolean	
immutable	Enable or disable the immutable directive in the Cache-Control response header. If enabled, the maxAge option should also be specified to enable caching. The immutable directive will prevent supported clients from making conditional requests during the life of the maxAge option to check if the file has changed.	false
index	Sends the specified directory index file. Set to false to disable directory indexing.	"index.html"
lastModified	Set the Last-Modified header to the last modified date of the file on the OS.	true
maxAge	Set the max-age property of the Cache-Control header in milliseconds or a string in ms format.	0
redirect	Redirect to trailing "/" when the pathname is a directory.	true
setHeaders	Function for setting HTTP headers to serve with the file.	

22.3 Using Express

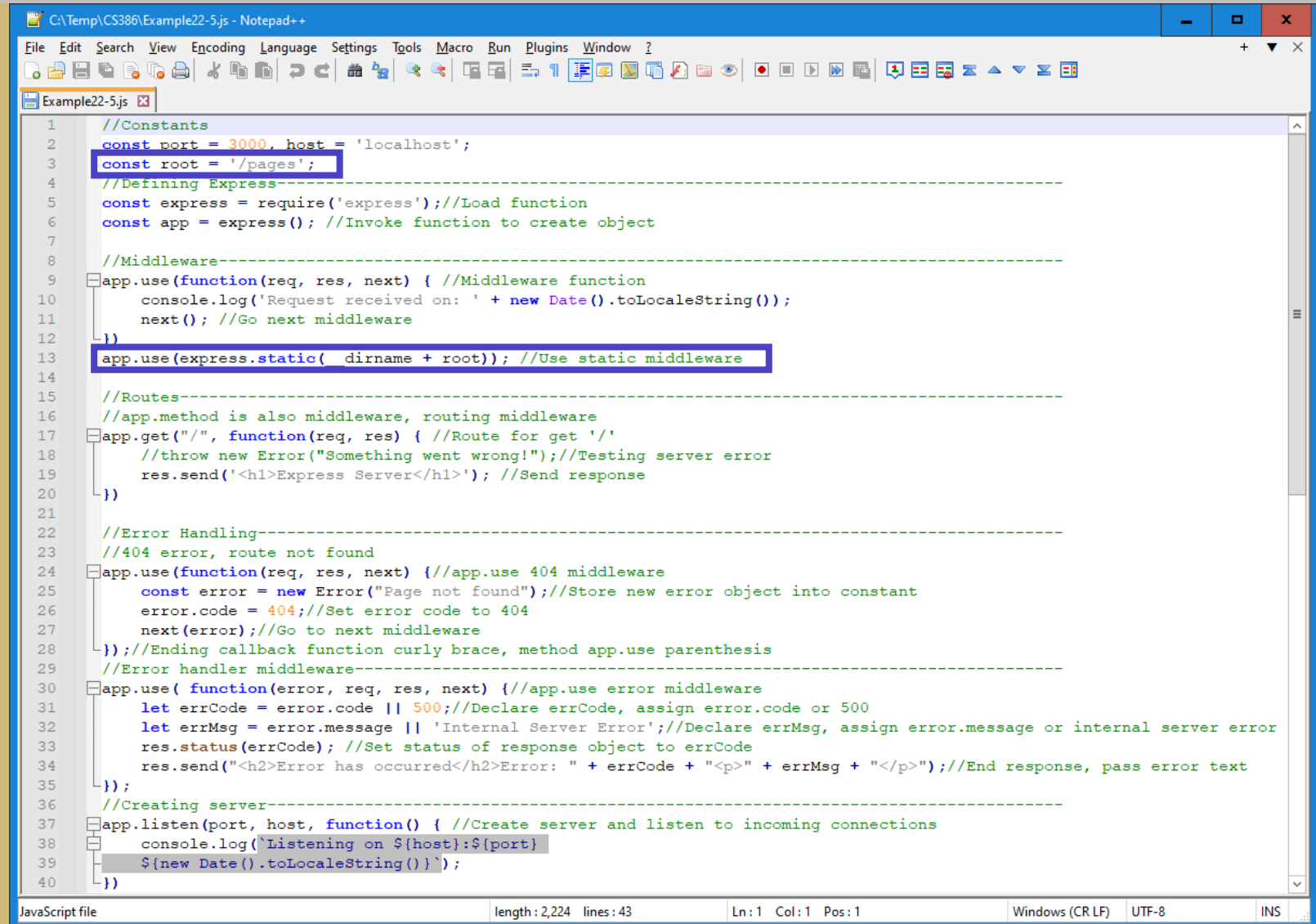
- **Example 22-5:**
- Clone Example22-4.js to Example22-5.js
- Create folder pages to store static html pages
- Create const root and set it to '/pages'
- Then add static middleware before routing:
 - ☐ Use app.use:
 - Use express.static using __dirname and root constant
- Place some html files into folder pages



```
Command Pr...  
c:\Temp\CS386>mkdir pages
```


22.3 Using Express

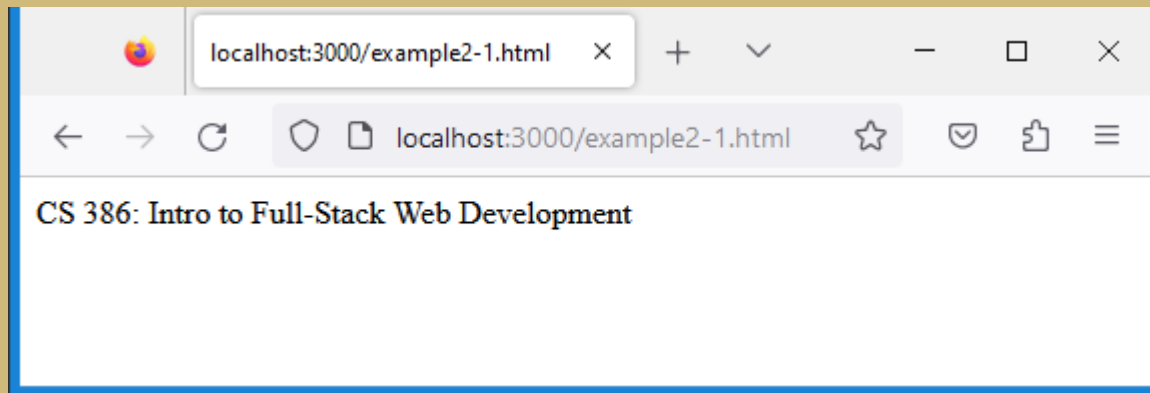
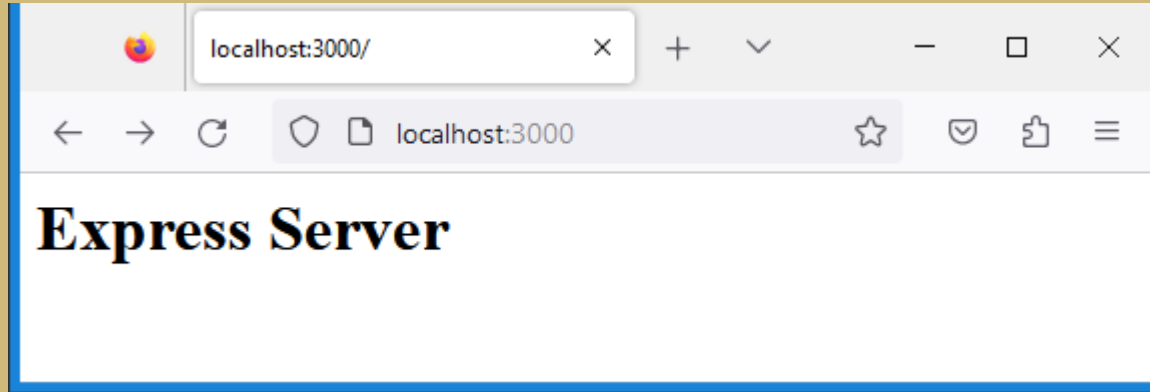
- **Example 22-5:**
- Will only serve files without routing



```
1 //Constants
2 const port = 3000, host = 'localhost';
3 const root = '/pages';
4 //Defining Express-----
5 const express = require('express');//Load function
6 const app = express(); //Invoke function to create object
7
8 //Middleware-----
9 app.use(function(req, res, next) { //Middleware function
10   console.log('Request received on: ' + new Date().toLocaleString());
11   next(); //Go next middleware
12 })
13 app.use(express.static( dirname + root)); //Use static middleware
14
15 //Routes-----
16 //app.method is also middleware, routing middleware
17 app.get("/", function(req, res) { //Route for get '/'
18   //throw new Error("Something went wrong!"); //Testing server error
19   res.send('<h1>Express Server</h1>'); //Send response
20 })
21
22 //Error Handling-----
23 //404 error, route not found
24 app.use(function(req, res, next) { //app.use 404 middleware
25   const error = new Error("Page not found");//Store new error object into constant
26   error.code = 404; //Set error code to 404
27   next(error); //Go to next middleware
28 }); //Ending callback function curly brace, method app.use parenthesis
29 //Error handler middleware-----
30 app.use( function(error, req, res, next) { //app.use error middleware
31   let errCode = error.code || 500; //Declare errCode, assign error.code or 500
32   let errMsg = error.message || 'Internal Server Error'; //Declare errMsg, assign error.message or internal server error
33   res.status(errCode); //Set status of response object to errCode
34   res.send("<h2>Error has occurred</h2>Error: " + errCode + "<p>" + errMsg + "</p>"); //End response, pass error text
35 });
36 //Creating server-----
37 app.listen(port, host, function() { //Create server and listen to incoming connections
38   console.log(`Listening on ${host}:${port}
39   ${new Date().toLocaleString()}`);
40 })
```

22.3 Using Express

➤ Example 22-5:



22.3 Using Express

Serving Static Files

- Previous example:
 - ❑ User typed filename into URL address bar in browser
 - ❑ Not good idea to show file names in url:
 - localhost:3000/myPage.html
- Rather create own route to map to file server side
- Use sendFile method to load file and return to client
- **Note:**
 - ❑ Later will use templating engines to create dynamic html

22.3 Using Express

Serving Static Files

Syntax:

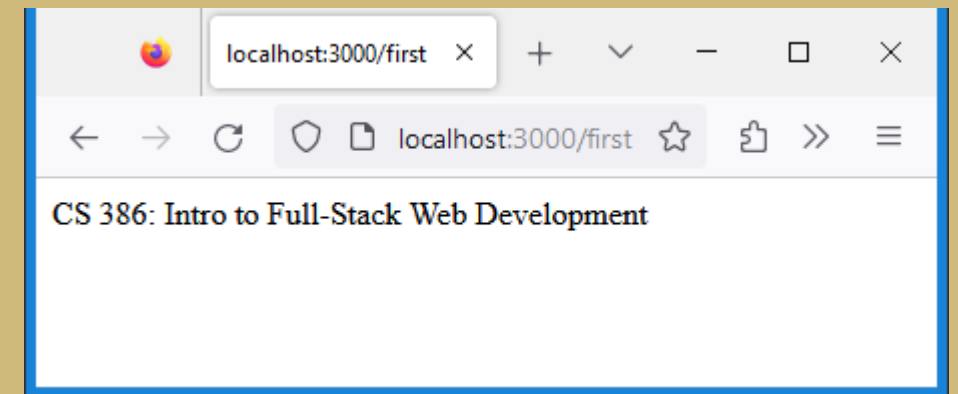
```
res.sendFile(path [, options] [, callback])
```

- sendFile method in Express:
- Remember how elaborate sending files in plain Node.js was (streaming)
- path:
 - ❑ Either must be absolute file path
 - ❑ Or just file name and root specified in options object
- options:
 - ❑ Specify options in object syntax { prop: value }
- Callback
 - ❑ Callback function

Property	Description	Default
maxAge	Sets the max-age property of the Cache-Control header in milliseconds or a string in ms format	0
root	Root directory for relative filenames.	
lastModified	Sets the Last-Modified header to the last modified date of the file on the OS. Set false to disable it.	Enabled
headers	Object containing HTTP headers to serve with the file.	
dotfiles	Option for serving dotfiles. Possible values are "allow", "deny", "ignore".	"ignore"
acceptRanges	Enable or disable accepting ranged requests.	true
cacheControl	Enable or disable setting Cache-Control response header.	true
immutable	Enable or disable the immutable directive in the Cache-Control response header. If enabled, the maxAge option should also be specified to enable caching. The immutable directive will prevent supported clients from making conditional requests during the life of the maxAge option to check if the file has changed.	false

22.3 Using Express

- **Example 22-6:**
- Add routing to map files (hide file name in address bar)
- Clone Example22-5.js to Example22-6.js
- Add route '/first'
- In callback, use sendFile method and specify html file
- Option object with property root and value of __dirname and constant root



22.3 Using Express



Example 22-6:

```
C:\Temp\CS386\Example22-6.js - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Example22-6.js
1 //Constants
2 const port = 3000, host = 'localhost';
3 const root = '/pages';
4 //Defining Express-----
5 const express = require('express');//Load function
6 const app = express();//Invoke function to create object
7
8 //Middleware-----
9 app.use(function(req, res, next) { //Middleware function
10   console.log('Request received on: ' + new Date().toLocaleString());
11   next();//Go next middleware
12 })
13 app.use(express.static(__dirname + root)); //Use static middleware
14
15 //Routes-----
16 //app.method is also middleware, routing middleware
17 app.get("/", function(req, res) { //Route for get '/'
18   //throw new Error("Something went wrong!"); //Testing server error
19   res.send('<h1>Express Server</h1>');//Send response
20 })
21 app.get('/first', function(req, res) {
22   res.sendFile('Example2-1.html', {root: __dirname + root}); //Send specified file
23 })
24
25 //Error Handling-----
26 //404 error, route not found
27 app.use(function(req, res, next) { //app.use 404 middleware
28   const error = new Error("Page not found");//Store new error object into constant
29   error.code = 404;//Set error code to 404
30   next(error);//Go to next middleware
31 });//Ending callback function curly brace, method app.use parenthesis
32 //Error handler middleware-----
33 app.use(function(error, req, res, next) { //app.use error middleware
34   let errCode = error.code || 500;//Declare errCode, assign error.code or 500
35   let errMsg = error.message || 'Internal Server Error';//Declare errMsg, assign error.message or internal server error
36   res.status(errCode); //Set status of response object to errCode
37   res.send("<h2>Error has occurred</h2>Error: " + errCode + "<p>" + errMsg + "</p>");//End response, pass error text
38 });
39 //Creating server-----
40 app.listen(port, host, function() { //Create server and listen to incoming connections
41   console.log('Listening on ${host}:${port}
42   ${new Date().toLocaleString()}');
43 })

JavaScript file length: 2,351 lines: 46 Ln: 46 Col: 1 Pos: 2,352 Windows (CR LF) UTF-8 INS
```