Express Development

"Node Web Development Framework"

DotNetTricks

express

Agenda

- Http Module
- Express
- Express App
- Routing
- Router Object
- View Engine
- Middleware and Express Middleware
- Express Request Processing

Node Http Module

- Very low level
- No cookies handling or parsing
- No built-in session support
- No built-in routing support
- No static file serving

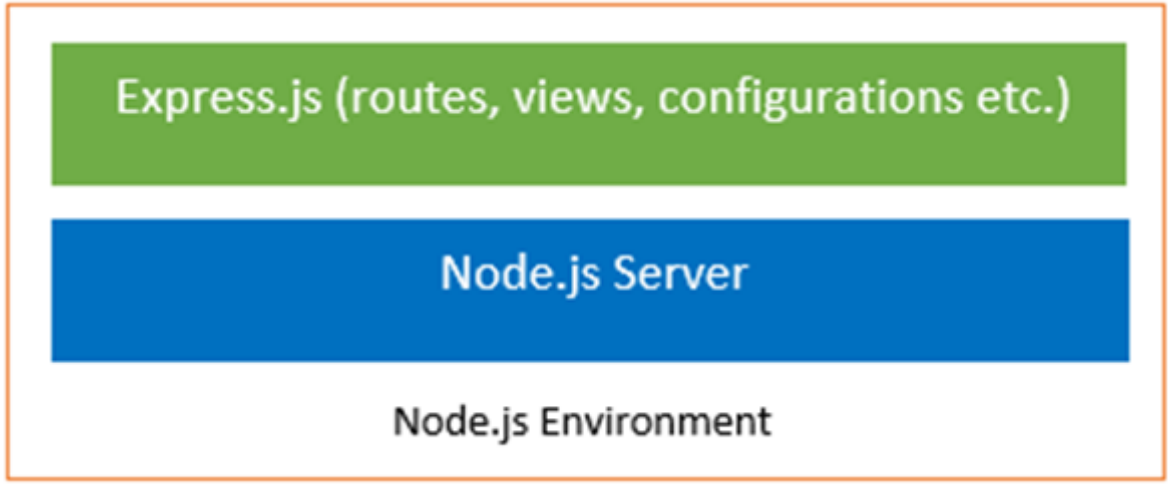
```
const http = require("http");

http.createServer( (req, res) => {
  res.writeHead(200, {'Content-Type': 'text/html'});
      res.end("Hello, Node!");
 }).listen(8081, () => {
   console.log('server is listening at 8081');
});
```

Express

Express

- Web development framework for Node and provides a thin layer for supporting web application fundamental features
- Used to build REST API with robust set of features for web and mobile apps
- Supports Routing
- Handle Configurations
- Supports Sessions
- Parses the arguments and headers



Express (Contd.)

- Supports Content Negotiation
- Supports Error Handling
- Supports multiple view engines like Pug, handlebars, JSHtml etc.
- Supports Multiple Databases
  - RDBMS – MySQL, MS SQL etc.
  - NoSQL – MongoDB, Firebase, Redis etc.

Express App

Express App

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

app.get("/", (req, res) => {
    res.send("This is Home Page");
});

var port= process.env.port || 1301;
app.listen(port);
```

server.js

Express App Architecture

Express App Architecture

- **root/**
  - **node_modules/**
    - express/
    - jade/
  - **public/**
    - css/
    - js/
    - img/
  - **routes/**
    - api.js
    - web.js
  - **views/**
    - layout.jade
    - index.jade
  - **models/**
    - user.js
  - app.js
  - package.json

Routing

Routing

- Routing helps to manage the application URIs.
- Helps us to respond for client requests.

```
const express = require('express');
const app = express();
app.get('/', (req, res) => { //Home Page Url
    res.send('Welcome to Home Page');
});
app.get('/about', (req, res) => { //About Us Page Url
    res.send('Welcome to About Us Page');
});
app.listen(1301);
```

server.js

Router Object

Router Object

- An isolated instance of middleware and routes
- Used to perform middleware and routing functions
- You can add middleware and HTTP method routes (such as get, put, post, and so on)
- Separate routes into their own file

Router Object Methods

Router Object Methods

- router.route() method is used to define a unique route
- router.METHOD() is used to provide routing functionality where METHOD is the HTTP method - GET, PUT, POST, DELETE and so on in lowercase.
- router.all() method is just like the router.METHOD(), except that it matches all HTTP verbs. Useful for adding "global" logic for specific path prefixes or arbitrary matches. For example, User authentication or authorization
- router.use() method is used to specify middleware function with optional path, that defaults to "/"

Router Object

```
var express = require('express');
// router object
var router = express.Router();
var users = [{ Id: 1, Name: "Shailendra"}];
// api routes
router.route('/user').get((req, res) =>{
        res.json(users);
});
router.route('/user/:id').get((req, res) =>{
    var id = req.params.id;
    res.json(Users[id-1]);
});
module.exports = router;
```

./routes/api.js

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

var apiRoutes = require('./routes/api');
// api routes configuration
app.use('/api', apiRoutes);

app.listen(1301);
```

server.js

View Engine

View Engine

- Tool to separate program-logic and UI.
- Makes the development easier.
- Improves flexibility.
- Easy to modify and maintain.
- Express supports multiple view engines like Pug, handlebars, JSHtml etc.

Handlebars View Engine

Handlebars View Engine

```
var express = require('express');
var exphbs = require('express-handlebars');
var app = express();

app.engine('handlebars', exphbs({defaultLayout:
'main'}));
app.set('view engine', 'handlebars');

app.get('/', function (req, res) {
res.render('home');
});

app.listen(3000);
```

server.js

```
<html>
<head>
   <title>Example App</title>
 </head>
<body>
   {{{body}}}
</body>
 </html>
```

views/layouts/main.handlebars

```
<h1>Example App</h1>
```

views/layouts/home.handlebars

Response Methods

Response Methods

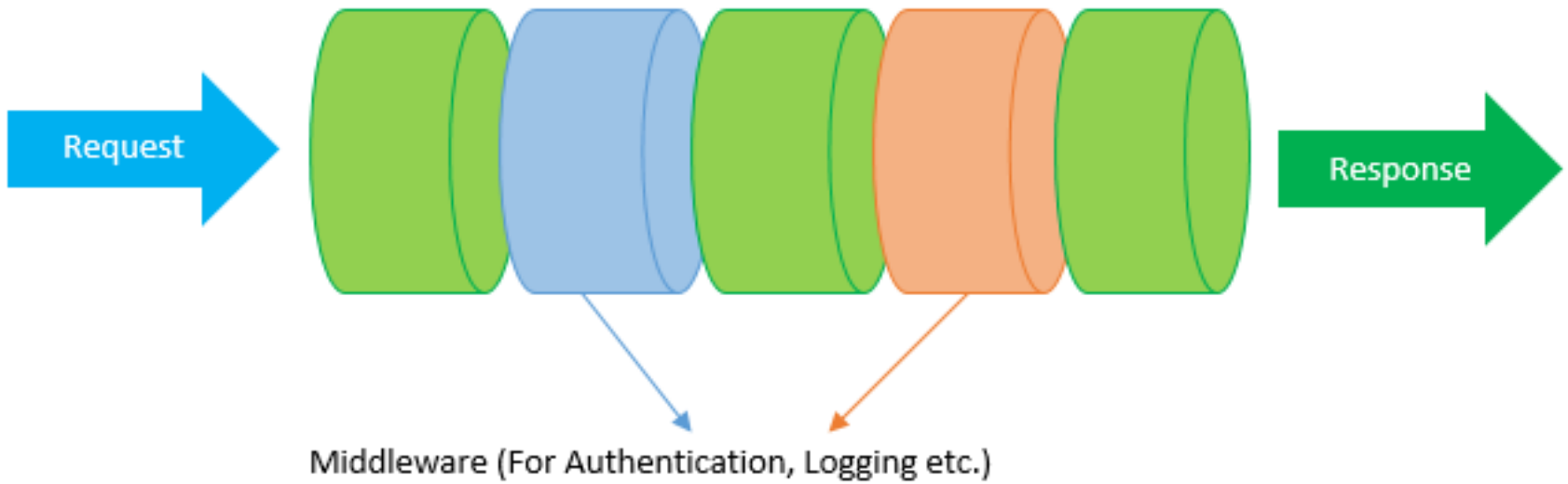
Here is a list of common methods that can be used to terminate the request/response cycle:

- res.json() - Send a JSON response
- res.jsonp() - Send a JSON response with JSONP support
- res.redirect() - Redirect a request
- res.render() - Render a view template
- res.send() - Send a response of various types
- res.sendFile - Send a file as an octet stream

Middleware

Middleware

- A middleware is a chain of functions which are executed before the request is being processed.
- Used for user authentication, logging, error handling etc.



Express Middleware

Express Middleware

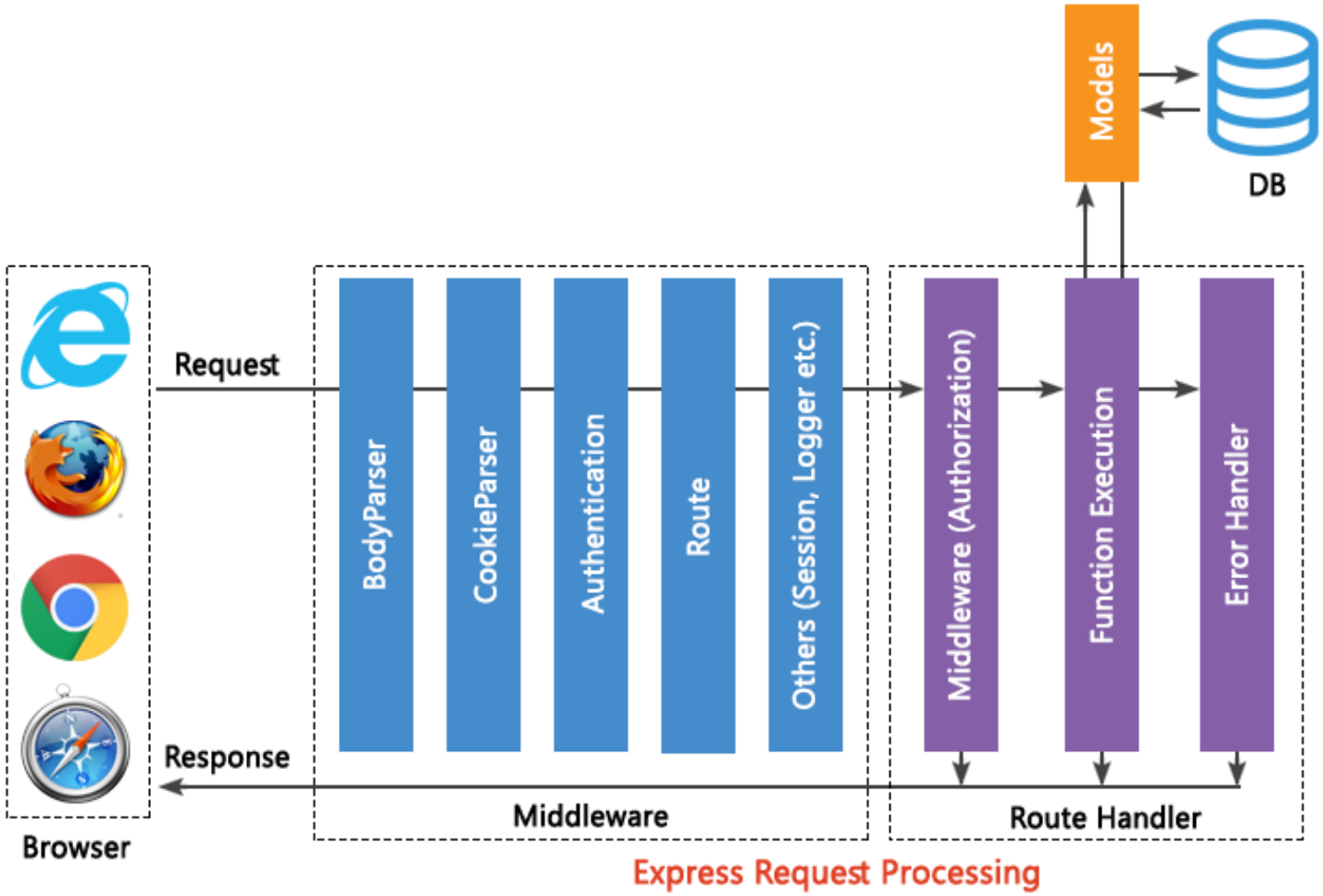
- **Application-level middleware** - Bind to an instance of the **app** object by using the app.use() and app.METHOD() functions, where METHOD is the HTTP method - GET, PUT, POST, DELETE in lowercase
- **Router-level middleware -** Bind to an instance of express.Router(). Load router-level middleware by using the router.use() and router.METHOD() functions
- **Built-in middleware -** From Express 4.x, the only built-in middleware function in Express is express.static to serve static assets such as HTML files, images, and so on.

Express Middleware

- **Third-party middleware -** Used to add functionality to your Express app. Installed by using NPM and can be added at the application level or at the router level like as : cookie-parser, body-parser, express-session etc.
- **Error-handling middleware –** Used to handle errors and Error-handling middleware always takes 4 arguments

Express Request Processing

Express Request Processing



Error Handling

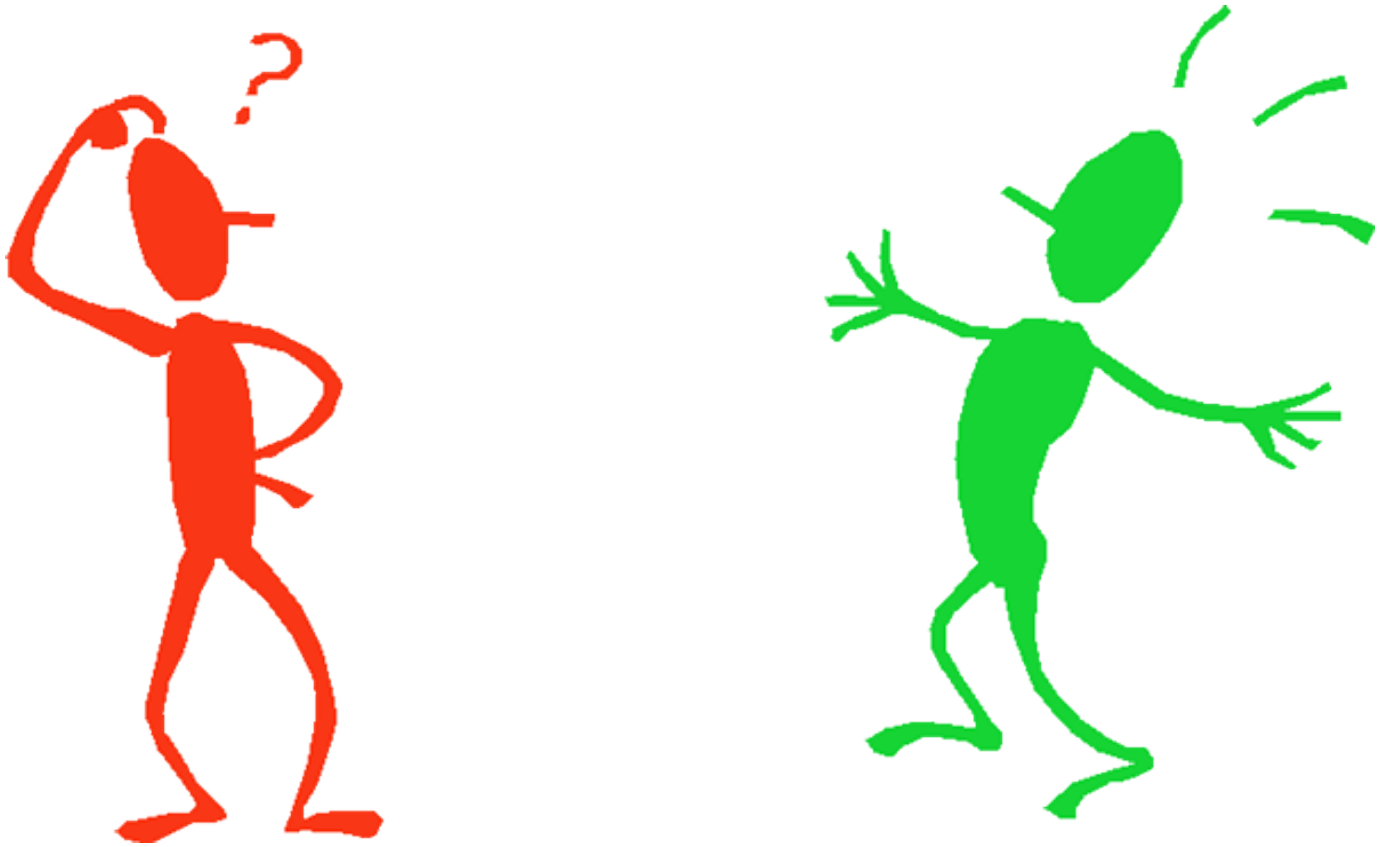
Error Handling

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

if (process.env.NODE_ENV === 'development') {
  // only use in development
  app.use(errorhandler())
}
```

server.js

Q&A



It’s the beginning…