

Express js

Why we need Express JS



Server logic is
Complex!

Your Application
code



Express  JS

Framework?



Completed Code with
Many Helpful functions



Using in our
Application

Alternatives to



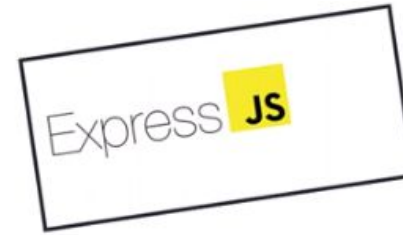
Node.js

Adonis.js (inspired from Larave

Koa

Sails.js

and more...



Body Parser

Installation

```
$ npm install body-parser
```

API

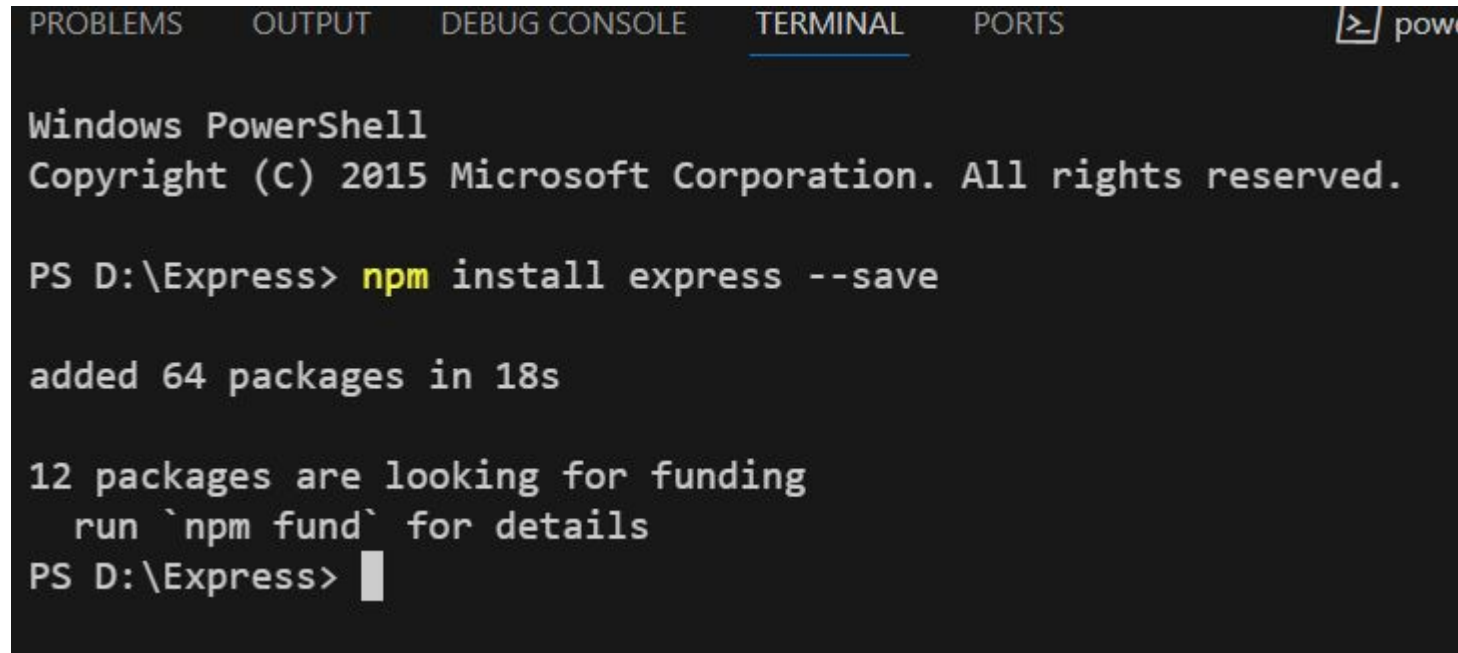
```
var bodyParser = require('body-parser')
```



Middleware module	Description	Replaces built-in function (Express 3)
body-parser	Parse HTTP request body. See also: body , co-body , and raw-body .	express.bodyParser
compression	Compress HTTP responses.	express.compress
connect-rid	Generate unique request ID.	NA
cookie-parser	Parse cookie header and populate req.cookies. See also cookies and keygrip .	express.cookieParser
cookie-session	Establish cookie-based sessions.	express.cookieSession
cors	Enable cross-origin resource sharing (CORS) with various options.	NA
errorhandler	Development error-handling/debugging.	express.errorHandler
method-override	Override HTTP methods using header.	express.methodOverride
morgan	HTTP request logger.	express.logger
multer	Handle multi-part form data.	express.bodyParser
response-time	Record HTTP response time.	express.responseTime
serve-favicon	Serve a favicon.	express.favicon
serve-index	Serve directory listing for a given path.	express.directory
serve-static	Serve static files.	express.static

Install express package

npm insatall express --save

A screenshot of a Windows PowerShell terminal window. The title bar at the top shows 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined), and 'PORTS'. On the far right of the title bar is a 'power' icon. The terminal content shows the following text:

```
Windows PowerShell  
Copyright (C) 2015 Microsoft Corporation. All rights reserved.  
  
PS D:\Express> npm install express --save  
  
added 64 packages in 18s  
  
12 packages are looking for funding  
  run `npm fund` for details  
PS D:\Express> 
```

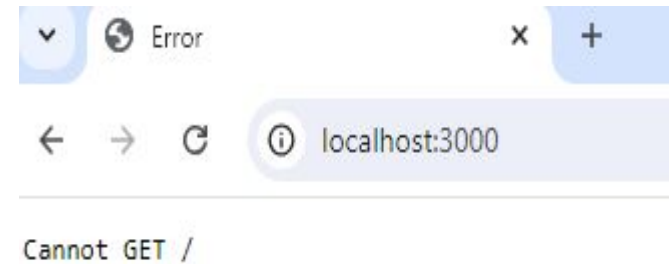

Creating http server

```
const http=require('http');  
const express=require('express');  
const app=express();  
const server = http.createServer(app);  
server.listen(3000);
```

Output

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\Express> node app.js
PS D:\Express> node app.js
█
```



Middleware

- Middleware functions can perform various tasks such as executing code, modifying the request and response objects, ending the request-response cycle, or calling the next middleware in the stack.
- Middleware in Express.js is a function that has access to the request object (req), the response object (res), and the next middleware function in the application's request-response cycle.

Node.js Approach

All URL Requests handled in this **Single** block.
But code looks **complex**!

Middleware1

Middleware2

REQUEST

Middleware3

Express

Creating middleware

```
const http=require('http');  
const express=require('express');  
const app=express();  
app.use(()=>{  
  console.log("express works");  
})  
const server = http.createServer(app);  
server.listen(3000);
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\Express> node app.js
PS D:\Express> node app.js
express works
express works
```

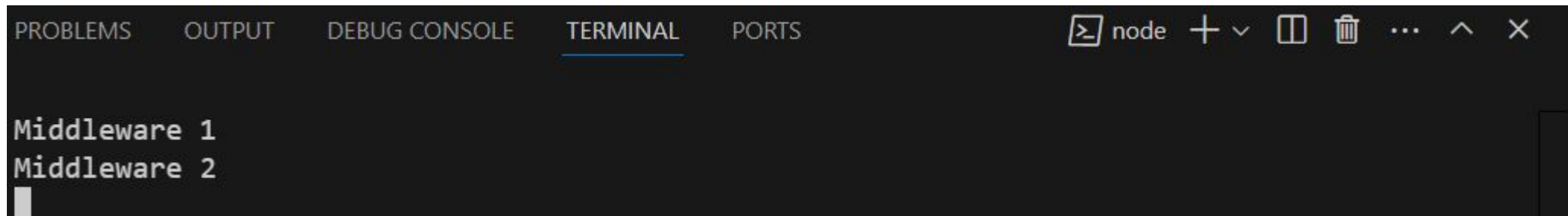
Ln 8, Col 21 Spaces: 4 UTF-8



Creating multiple Middleware

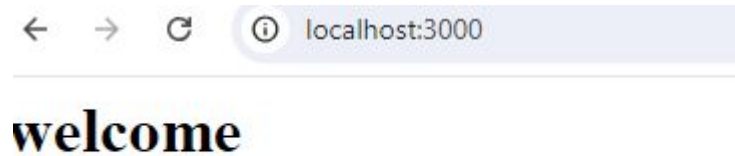
```
const http=require('http');
const express=require('express');
const app=express();
app.use((req,res,next)=>{
  console.log("Middleware 1");
  res.send('<h1>welcome</h1>');
  next();
})
app.use((req,res,next)=>{
  console.log("Middleware 2");
})
const server = http.createServer(app);
server.listen(3000);
```

Output



A screenshot of a Visual Studio Code terminal window. The terminal has a dark background and a light gray border. At the top, there is a tab bar with five tabs: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', 'TERMINAL' (which is selected and underlined), and 'PORTS'. To the right of the tabs, there is a toolbar with icons for running a command (a terminal icon), a dropdown menu (a plus sign), a window icon, a trash icon, a menu icon (three dots), a scroll bar icon (an upward arrow), and a close icon (an X). The terminal content shows two lines of text: 'Middleware 1' and 'Middleware 2'. A white cursor is visible at the end of the second line.

```
Middleware 1
Middleware 2
```



Express js repository link

[Github.com/expressjs/express](https://github.com/expressjs/express)



lib

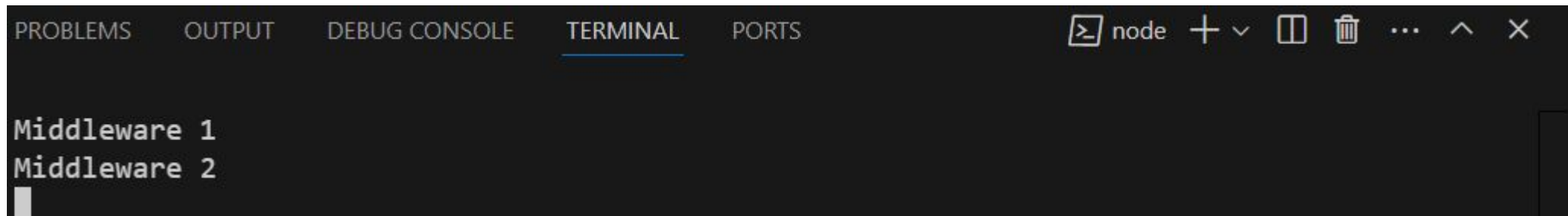


response.js

Without using http module, create a server using express js

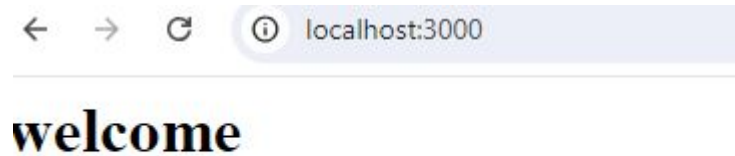
```
const express=require('express');
const app=express();
app.use((req,res,next)=>{
  console.log("Middleware 1");
  res.send('<h1>welcome</h1>');
  next();
})
app.use((req,res,next)=>{
  console.log("Middleware 2");
})
app.listen(3000);
```

Output



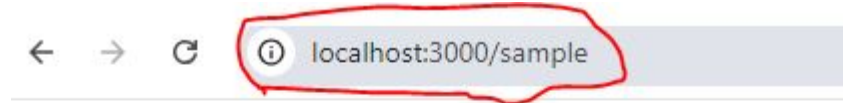
A screenshot of a Visual Studio Code terminal window. The terminal has a dark background and a light gray border. At the top, there is a tab bar with the following tabs: PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL, and PORTS. To the right of the tabs, there is a toolbar with icons for running a command (a terminal icon), a dropdown menu (a plus sign), a window icon, a trash icon, a menu icon (three dots), a scroll bar icon (an upward arrow), and a close icon (an X). The terminal content shows two lines of text: "Middleware 1" and "Middleware 2". A white cursor is visible at the end of the second line.

```
Middleware 1
Middleware 2
```



How the route URL works??

```
const express=require('express');
const app=express();
app.use('/',(req,res,next)=>{
  console.log("Middleware 2");
  res.send('<h1>Thank you</h1>');
})
app.use('/sample',(req,res,next)=>{
  console.log("Middleware 1");
  res.send('<h1>welcome</h1>');
  next();
})
app.listen(3000);
```



Thank you

```
PS D:\Express> node routeurlworks.js
Middleware 2
```

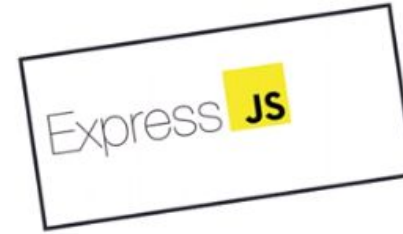
How to give the right path??

```
const express=require('express');
const app=express();
app.use('/test',(req,res,next)=>{
  console.log("Middleware 2");
  res.send('<h1>Thank you</h1>');
})
app.use('/sample',(req,res,next)=>{
  console.log("Middleware 1");
  res.send('<h1>welcome</h1>');
  next();
})
app.listen(3000);
```



welcome

Parse the request data



Body Parser

Installation

```
$ npm install body-parser
```

API

```
var bodyParser = require('body-parser')
```

Parse the request data

```
const express=require('express');
const app=express();
const parse=require('body-parser');
app.use(parse.urlencoded());
app.use('/add',(req,res,next)=>{
  res.send('<h1>Add product</h1><form action="/store" method="post"><input type="text" name="title"><input
type="submit" value="submit"></form>');
})
app.use('/store',(req,res,next)=>{
  console.log(req.body);
  res.send('<h1>product submitted</h1>');

})
app.listen(3000);
```

Output

← → ↻ ⓘ localhost:3000/add

Add product

← → ↻ ⓘ localhost:3000/store

product submitted

```
PS D:\Express> node parse.js
body-parser deprecated undefined extended: provide extended option parse.js:4:15
{ title: 'soap' }
```


Handler function

```
const express=require('express');
const app=express();
const parse=require('body-parser');
app.use(parse.urlencoded());
app.get('/add',(req,res,next)=>{
res.send('<h1>Add product</h1><form action="/store" method="post"><input type="text" name="title"><input type="submit"
value="submit"></form>');
})
app.post('/store',(req,res,next)=>{
  console.log(req.body);
res.send('<h1>product submitted</h1>');

})

app.listen(3000);
```

Output

← → ↻ ⓘ localhost:3000/add

Add product

← → ↻ ⓘ localhost:3000/store

product submitted

```
PS D:\Express> node parse.js
body-parser deprecated undefined extended: provide extended option parse.js:4:15
{ title: 'soap' }
```

Manage the routes in different file

Create a new file with name admin.js

```
const express = require('express');
const route=express.Router();
route.get('/add',(req,res,next)=>{
  res.send('<h1>Add product</h1><form action="/store"
  method="post"><input type="text" name="title"><input type="submit"
  value="submit"></form>');
})
route.post('/store',(req,res,next)=>{
  console.log(req.body);
  res.send('<h1>product submitted</h1>'));
};
module.exports=route;
```

Import the admin.js file

```
const express=require('express');  
const app=express();  
const parse=require('body-parser');  
const adminroutes=require('./Routes/admin.js');  
app.use(parse.urlencoded());  
app.use(adminroutes);  
app.listen(3000);
```

Output

← → ↻ ⓘ localhost:3000/add

Add product

← → ↻ ⓘ localhost:3000/store

product submitted

```
PS D:\Express> node parse.js
body-parser deprecated undefined extended: provide extended option parse.js:4:15
{ title: 'soap' }
```

404 Page not found

```
const express=require('express');
const app=express();
const parse=require('body-parser');
app.use(parse.urlencoded());
app.get('/add',(req,res,next)=>{
  res.send('<h1>Add product</h1><form action="/store" method="post"><input type="text" name="title"><input type="submit"
value="submit"></form>');
})
app.post('/store',(req,res,next)=>{
  console.log(req.body);
  res.send('<h1>product submitted</h1>');
})
app.use((req,res,next)=>
{
  res.send('<h1>404 page not found</h1>');
})
app.listen(3000);
```

Output

404 page not found

The screenshot shows a web browser at `localhost:3000`. The main content area displays "404 page not found". The Network tab in the developer tools is open, showing a list of requests. The first request, `localhost`, has a status of `304` and a type of `document`. The second request, `favicon.ico`, also has a status of `304` and a type of `text/html`. The status codes `304` are underlined in red in the original image.

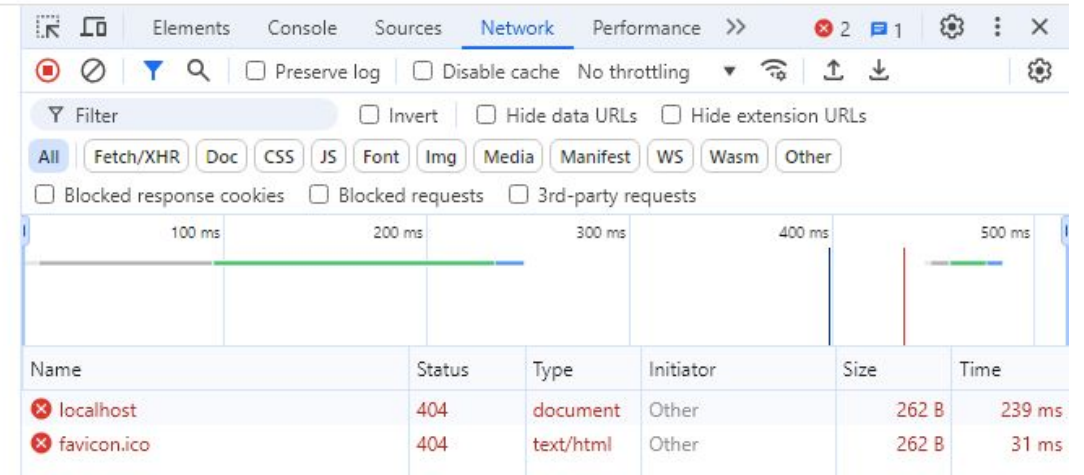
Name	Status	Type	Initiator	Size	Time
localhost	304	document	Other	178 B	52 ms
favicon.ico	304	text/html	Other	178 B	20 ms

Right way of sending 404 error

```
const express=require('express');
const app=express();
const parse=require('body-parser');
app.use(parse.urlencoded());
app.get('/add',(req,res,next)=>{
res.send('<h1>Add product</h1><form action="/store" method="post"><input type="text" name="title"><input type="submit" value="submit"></form>');
})
app.post('/store',(req,res,next)=>{
  console.log(req.body);
res.send('<h1>product submitted</h1>');
})
app.status(404).use((req,res,next)=>
{
res.send('<h1>404 page not found</h1>');
})
app.listen(3000);
```


Output

404 page not found



The screenshot shows the Network tab of a web browser's developer tools. The address bar at the top indicates the page is loaded from localhost:3000. The Network tab is active, displaying a list of requests. Two requests are visible, both with a status of 404 (Not Found). The first request is for 'localhost' and the second is for 'favicon.ico'. Both are document and text/html types respectively, initiated by 'Other'. The size for both is 262 B. The time taken for the first request is 239 ms and for the second is 31 ms. A timeline at the top of the Network tab shows the duration of these requests.

Name	Status	Type	Initiator	Size	Time
localhost	404	document	Other	262 B	239 ms
favicon.ico	404	text/html	Other	262 B	31 ms

Filtering Paths/ URL Prefix

Create a new file with name admin.js

```
const express = require('express');
const route=express.Router();
route.get('/add',(req,res,next)=>{
  res.send('<h1>Add product</h1><form action="/admin/store"
method="post"><input type="text" name="title"><input type="submit"
value="submit"></form>');
})
route.post('/store',(req,res,next)=>{
  console.log(req.body);
  res.send('<h1>product submitted</h1>'));
module.exports=route;
```

Import the admin.js file

```
const express=require('express');  
const app=express();  
const parse=require('body-parser');  
const adminroutes=require('./Routes/admin.js');  
app.use(parse.urlencoded());  
app.use('/admin',adminroutes);  
app.listen(3000);
```

Output

← → ↻ ⓘ localhost:3000/add

Add product

← → ↻ ⓘ localhost:3000/store

product submitted

```
PS D:\Express> node parse.js
body-parser deprecated undefined extended: provide extended option parse.js:4:15
{ title: 'soap' }
```

