
`http`

`http`

`if`

```
if (req.method === 'GET' && req.url ===
'/users')
```

`JSON.parse`

```
res.setHeader('Content-Type',
'application/json')
res.end(JSON.stringify(data))
```

`req.url` `URL`

```
app.get('/users', handler)
```

```
app.use(express.json())
req.body
```

```
res.json(data)
```

```
app.get('/users/:id', ...)
req.params.id
```

```
app.use(express.static('public'))
```

`req` `res`

`node server.js` `npx nodemon server.js`

`express()`

`http.RequestListener` `http.createServer(app)`
`app.listen()`

`app.get()` `app.post()` `app.put()` `app.patch()` `app.delete()` `app.all()`

`IncomingMessage` `ServerResponse`

```
req.params          :id  req.params.id
req.query            ?sort=asc
req.body
req.headers
req.method             'GET'
req.path               '/users/42'
```

```
res.send(body)
res.json(data)
res.status(code)
res.redirect(url)
res.set(field, value)
res.sendFile(path)
```

`app.js` `server.js` `app.listen()` `app`

```
                                                                  npm install express


next(err)                  express-async-errors
                                      express --version   npm list express
```

```
npm install express                                                    path-to-
regexp                  qs                      depd

                                        npm install
```

```
                                                                       GET
/                   { message: "Welcome to my API", version: "1.0" } GET /health       {
status: "ok", uptime: <seconds> }      process.uptime() GET /echo       ?message=
          { echo: "<value>" }                   { error: "message query param required" }
        GET /time
process.env.PORT || 3000
```

```
                                          src/app.js
                                                    listen() server.js       app
    listen()
   app.test.js      supertest            app
    version: "1.0"                                  uptime
                                      { echo: "hello" }
```

```
                                requestLogger(req, res, next)        [METHOD] /path →
STATUS_CODE (Xms)                    [GET] /health → 200 (3ms)
                                                'finish'        res
     Date.now()   process.hrtime.bigint()
                                console.log          jest.spyOn
```

```
express.Router()
```

`req.params`

`req.params.id` `/users/42` `===` `false` `parseInt(req.params.id, 10)` `Number(req.params.price)` `isNaN()`

`?` `qs` `req.query`

	`req.params.id`	`/users/42` `'42'`
	`req.query.sort`	`/items?sort=date` `'date'`
	`req.query.tag`	`['a', 'b']`
	`req.query.f.k`	`{ k: 'v' }`

```
GET /users/42
GET /posts/2025/06/my-slug
```

```
GET /users?role=admin&page=2
GET /search?q=express&limit=10
```

`app.js`

`express.Router()`

`router.route(path)`

app.use()

express.Router() { strict: true }

/users /users/ strict: true

{ caseSensitive: true } /Users /users

app.set('case sensitive routing', true)

router.param('id', callback) :id

(req, res, next, value) value

GET /products/:id :id

{ error: "id must be a positive integer" } { productId: <number> }

GET /products category 'all' minPrice maxPrice

sort

price|name|newest 'newest' GET /search q

tags [].concat()

---

express.Router() routes/products.js GET /

POST / GET /:id PUT /:id DELETE /:id

routes/categories.js GET / GET /:slug/products src/app.js productsRouter

/products categoriesRouter /categories router.route()

---

routes/articles.js router.param('articleId', ...) articleId { error: "Article not found" } req.article next()

GET /:articleId PUT /:articleId DELETE /:articleId req.article

`app.use()`

`app.get()`

`Content-Type: application/json` `req.body` `req.body` `undefined`

```
app.use(express.json({ limit: '1mb' }))
```

`limit` `100kb`

Content-Type: application/x-www-form-urlencoded
req.body
app.use(express.urlencoded({ extended: false }))
extended: false querystring true qs

/logo.png public/logo.png
app.use(express.static('public'))
Last-Modified 304 Not Modified

express.raw() req.body Buffer

express.text() req.body
limit express.json()

	'dev' 'combined' 'tiny'	npm i morgan
	X-Frame-Options Content-Security-Policy Strict-Transport-Security	npm i helmet
		npm i cors
		npm i express-rate-limit
		npm i compression

`req` `res`

`next()`

`next(err)`

`next`

`app.use()`

`next()`

`next()`

`return` `res.json()` `res.send()`

`errorHandler` `(err, req, res, next)`

`next`

app.use() cors(options) rateLimit(options) helmet(options)

app.js requestId crypto.randomUUID() req.id X-Request-Id responseTimer 'finish' res [METHOD] /path → STATUS (Xms) bodyGuard Content-Type application/json GET /ping { pong: true } POST /echo Content-Type: text/plain

---

requireRole(...roles) req.headers['x-user-role'] next() req.userRole GET /public GET /user-area user admin GET /admin-only admin

---

createError(status, message)
GET /safe GET /client-error next(createError(400, 'Bad input')) GET /server-error next(createError(500, 'DB exploded'))
{ error: err.message }
{ error: "Internal server error" }
/safe
/client-error /server-error

req res

IncomingMessage ServerResponse

`req.method` `'GET'` `'POST'`

`req.path` `'/users/42'`

`req.url` `'/users/42?sort=asc'`

`req.originalUrl`

`req.hostname` `'api.example.com'`

`req.ip` `app.set('trust proxy', 1)` `X-Forwarded-For`

`req.protocol` `'http'` `'https'`

`req.secure` `req.protocol === 'https'`

`req.params` `:id`

`req.query`

`req.body` `express.json()`

`req.cookies` `cookie-parser`

`req.signedCookies` `cookie-parser`

`req.headers`

`req.body` `undefined`

	`application/json`		
	`application/x-www-form-urlencoded`		`<form method="POST">`
	`application/octet-stream`		
	`text/plain`		

	`JSON.stringify`	`application/json`
	`404 Not Found`	`text/plain`
	`Content-Disposition: attachment`	
	`res.redirect(301, url)`	
	`res.sendStatus(204)`	

`res.locals`

`req`

`req.cookies` `cookie-parser`
`res.cookie()`

`httpOnly: true`

`secure: true` `process.env.NODE_ENV === 'production'`

`sameSite: 'strict'` `'lax'` `'strict'`

`sameSite` `httpOnly`

`httpOnly`

`httpOnly`

```
req.path

req.url


req.originalUrl


                  /api               GET /api/users?sort=asc                        req.path
= '/users' req.url = '/users?sort=asc' req.originalUrl = '/api/users?sort=asc'
```

```
                                          req     GET /inspect
req.method req.path req.originalUrl req.hostname req.ip req.protocol
            POST /content-check       req.is()
            req.body                    req.body                    GET /negotiate
req.accepts(['json', 'html', 'text'])

                     Accept
```

```
                               res     GET /headers-demo                                    X-
App-Version Cache-Control: no-store X-Request-Id       crypto.randomUUID()
      GET /redirect-demo           ?to=
['/safe-a', '/safe-b']                            GET /safe-a    GET /safe-b
             GET /locals-demo                               res.locals.startTime
res.locals.requestId                              res.locals
                                                                    res.locals
```

cookie-parser POST /login { username, password } { alice: 'pass1', bob: 'pass2' } session httpOnly: true, maxAge: 600_000 { ok: true } GET /profile req.signedCookies.session false { username } POST /logout session { loggedOut: true } supertest agent() /profile

```
                                                                express.static()
```

`express.static(root, options)`

`root + req.path`

`next()`

`app.use()`

`node_modules`

	`'index.html'`	`false`
	`'ignore'`	`.env` `.htaccess` `'ignore'`

	`true`	`false`
	`true`	`Last-Modified`
		`max-age` `'1d'` `'30 days'` `0`
	`true`	
		`(res, path, stat)`
	`true`	`true` `next()` `false`

`express.static()`

`helmet()` `cors()` `express.static()`

`<% %>`

`include()`

`res.locals` `res.render()`

<%= value %> <script> <script> <%- value %> value <%- %> innerHTML

public/ maxAge: '1d' setHeaders Cache-Control: no-store .html dotfiles: 'ignore' etag: true /assets uploads/
GET / { status: 'ok' } /assets/test.txt /assets /assets/.env

---

views/ partials/header.ejs /css/styles.css partials/footer.ejs index.ejs items/list.ejs items res.locals.appName = 'My Store' GET / index { title: 'Home', itemCount: 3 } GET /items items/list GET /items/empty items/list

---

```
                                                res.locals.year
res.locals.currentUser      null     { name: 'Alice' }       x-user: alice
             views/error.ejs                                        res.locals.year
                                                 res.render('error', { title: 'Error',
status, message })                                req.accepts('html')
                GET /crash        next(createError(500, 'Something broke'))
/crash     Accept: text/html                                        /crash     Accept:
application/json                          x-user: alice
res.locals.currentUser.name       'Alice'                    GET /whoami
```

```
                                                    res.status(500).send()
```

`next(err)`

`(err, req, res)` `(err, req, res, next)`

`next`

`await`

```
```

```
```

`next()`

```
```

`next(err)`

`express-async-errors`

`next(err)`

```
express@5
```

`.status` `.statusCode`

`http-errors`

`next(err)`

asyncHandler(fn) AppError Error status isOperational

GET /users/:id

AppError(404) POST /users

req.body.name AppError(422) new Error('DB write failed')

'Internal server error'

/api/products router.route() /api/categories next(createError(404, ...)) name === 'ValidationError' createError(422, err.message) code === 'ENOENT' createError(404, 'File not found') GET /api/trigger-validation name: 'ValidationError' GET /api/trigger-enoent code: 'ENOENT' /api/trigger-validation /api/trigger-enoent

express-async-errors GET /tasks GET /tasks/:id POST /tasks title await Promise.resolve() PATCH /tasks/:id done

```
GET /users          GET /users/5          POST /users          DELETE /users/5
```

```
/users    /user
/products    /product
```

```
/users/:id/orders
/posts/:id/comments
```

```
/user-profiles    /userProfiles    /user_profiles
```

```
/users    /users/
```

	`/users`			
	`/users/:id`			
	`/users`			
	`/users/:id`			
	`/users/:id`			
	`/users/:id`			

		`Location`
		`Allow`
		`Retry-After`
		`Retry-After`

`body.error`

`body.message` `body.errors[0].msg`

	`/api/v1/users`	
	`Accept: application/vnd.myapp.v2+json`	
	`/users?version=1`	

```
        ORDER BY ${req.query.sort}                                  Set

order           'asc'   'desc'
```

/api/v1/products / / name price
details Location /:id
/:id name price
/:id
/api/v1/products express-async-errors Location
details

/api/v1/products ?category=
?minPrice= ?maxPrice= ?
sort=name|price|id id ?order=asc|desc asc ?page=
?limit= { data: [...], meta: { total, page,
perPage, pages } }
minPrice maxPrice
meta.pages
minPrice maxPrice

/api/v1/articles /api/v2/articles
{ data: ... }
createArticlesRouter({
wrap }) wrap { data: ... } {
data: [...], meta: { total } } / / /
/:id /:id { error, status }
{ data: [...], meta }
{ data: ... }

---

app.js

`app.js`

METHOD /path
src/routes/v1/products.js

req res
src/controllers/productsController.js

req res
src/services/productsService.js

src/models/product.js

`listen()` `app.js`

`supertest` `listen()`

`server.js` `app.js`

`process.env.X`

`config/index.js`

app.js

app.js server.js

app.js

GET /api/v1/notes POST /api/v1/notes DELETE /api/v1/notes/:id

app.js server.js src/config/index.js src/routes/v1/notes.js src/controllers/notesController.js src/services/notesService.js src/middleware/errorHandler.js

app.js

src/config/index.js PORT APP_SECRET NODE_ENV process.env src/middleware/index.js requestId req.id notFound next(err) errorHandler src/routes/v1/index.js /health { status: 'ok', uptime: process.uptime() } /echo app.js /api/v1/health /api/v1/echo x-request-id APP_SECRET

---

src/models/book.js findAll findById insert update remove src/services/booksService.js /^\d{13}$/ src/controllers/booksController.js src/routes/v1/books.js app.js server.js

---

pdfs/