



### Core Concepts & Setup

#### Basic Setup

```

Installation:

npm install express --save

Basic App Structure:

const express = require('express');
const app = express();
const port = 3000;

app.get('/', (req, res) => {
  res.send('Hello World!');
});

app.listen(port, () => {
  console.log(`Example app listening at http://localhost:${port}`);
});

Running the App:

node <your_app_file>.js
  
```

#### Middleware Basics

```

app.use(middleware) Applies middleware to all routes.

Middleware functions have access to the request object ( req ), the response object ( res ), and the next() middleware function in the application's request-response cycle.

next() A function to pass control to the next middleware function. Crucial for chaining middleware.

Example:
app.use((req, res, next) => {
  console.log('Time: ', Date.now());
  next();
});
  
```

#### Request and Response Objects

<code>req.params</code>	Access route parameters (e.g., <code>/users/:userId</code> ).
<code>req.query</code>	Access query string parameters (e.g., <code>/search?q=term</code> ).
<code>req.body</code>	Access the request body (requires body-parsing middleware).
<code>res.send()</code>	Sends the HTTP response. The <code>body</code> can be a string, an object, or an array.
<code>res.json()</code>	Sends a JSON response.
<code>res.status(code)</code>	Sets the HTTP status code.

### Routing

#### Basic Routing

```

Route Methods:

app.get('/', (req, res) => { ... }); // GET route
app.post('/', (req, res) => { ... }); // POST route
app.put('/:id', (req, res) => { ... }); // PUT route
app.delete('/:id', (req, res) => { ... }); // DELETE route

Route Parameters:

app.get('/users/:userId', (req, res) => {
  res.send(`User ID: ${req.params.userId}`);
});

Chaining Route Handlers:

app.route('/book')
  .get((req, res) => { ... })
  .post((req, res) => { ... })
  .put((req, res) => { ... });
  
```

#### Route Paths

<code>/</code>	Matches only the path <code>/</code> .
<code>/*</code>	Matches any single character except <code>/</code> .
<code>/users/:userId?</code>	The <code>?</code> after <code>userId</code> makes the route parameter optional.
<code>/:username([a-zA-Z]+)</code>	Matches only if username contains letters.

#### Using `next()` in Routes

```

app.get('/example', (req, res, next) => {
  console.log('First handler');
  next();
}, (req, res) => {
  console.log('Second handler');
  res.send('Hello from example!');
});

This allows you to define multiple handlers for a single route, performing different operations in sequence.
  
```

### Middleware

## Built-in Middleware

<code>express.static</code>	Serves static files (e.g., images, CSS, JavaScript) from a directory.
<code>root</code>	<code>root</code> specifies the root directory from which to serve static assets.
<code>options</code>	
	<pre>app.use(express.static('public'));</pre>
<code>express.json</code>	Parses incoming requests with JSON payloads and is based on <code>body-parser</code> .
<code>options</code>	
	<pre>app.use(express.json());</pre>
<code>express.urlencoded</code>	Parses incoming requests with URL-encoded payloads and is based on <code>body-parser</code> .
<code>options</code>	
	<pre>app.use(express.urlencoded({ extended: true }));</pre>

## Advanced Topics

### Error Handling

Express comes with a built-in error handler. To use it, you simply pass an <code>Error</code> object to <code>next()</code> :
<pre>app.get('/error', (req, res, next) =&gt; {   const err = new Error('This is an error!');   next(err); });  app.use((err, req, res, next) =&gt; {   console.error(err.stack);   res.status(500).send('Something broke!'); });</pre>
Remember to place the error-handling middleware after all other middleware and route handlers.

## Third-Party Middleware

Examples:
<ul style="list-style-type: none"><li><code>morgan</code>: HTTP request logger middleware.</li><li><code>cors</code>: Enable Cross-Origin Resource Sharing.</li><li><code>helmet</code>: Helps secure Express apps by setting various HTTP headers.</li><li><code>cookie-parser</code>: Parse Cookie header and populate <code>req.cookies</code>.</li></ul>
Example with Morgan:
<pre>npm install morgan  const morgan = require('morgan'); app.use(morgan('dev')); // 'dev' is a predefined format</pre>

## Custom Middleware

You can create your own middleware functions to handle specific tasks.
<pre>const myLogger = (req, res, next) =&gt; {   console.log('LOGGED');   next(); };  app.use(myLogger);</pre>
Middleware can also be applied to specific routes:
<pre>app.get('/profile', myLogger, (req, res) =&gt; {   res.send('Profile Page'); });</pre>

### Template Engines

Setting up EJS	<pre>npm install ejs  app.set('view engine', 'ejs');  app.get('/template', (req, res) =&gt; {   res.render('index', { name: 'World' }); // renders views/index.ejs });</pre>
Popular Engines	Besides EJS, other template engines include Pug (formerly Jade), Handlebars, and Mustache.

### Best Practices

<ul style="list-style-type: none"><li><b>Security:</b> Use Helmet to secure HTTP headers, and sanitize user inputs to prevent XSS and SQL injection attacks.</li><li><b>Configuration:</b> Use environment variables for configuration.</li><li><b>Logging:</b> Use a logging library like Winston or Bunyan for structured logging.</li><li><b>API Versioning:</b> Version your APIs to maintain backward compatibility.</li></ul>
---