Education
@topdev_media

Express.js CHEATSHEET

With concrete examples...

INSTALLATION

Install Express globally or in your project.

```
# Install Express
npm install express --save
```

BASIC EXPRESS SERVER

```
const express = require('express');
const app = express(); // Create an Express application
const port = 3000; // Set the port

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

app.listen(port, () => {
  console.log(`Server is running on http://localhost:${port}`);
});
```

- **express()** creates an instance of an Express application.
- **app.get()** handles HTTP GET requests.
- **app.listen()** starts the server on the specified port.

MIDDLEWARE

Middleware functions are functions that have access to the request (req), response (res), and next middleware in the request-response cycle.

Custom Middleware

```
app.use((req, res, next) => {
  console.log(`${req.method} request for '${req.url}'`);
  next(); // Call the next middleware
});
```

Built-in Middleware

For serving static files like images, CSS, etc.

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

Built-in Middleware

To parse incoming JSON requests.

```
app.use(express.json());
```

ROUTING

Routing in Express is defining how the application responds to different HTTP methods and paths.

Basic Routing

```
app.get('/', (req, res) => {
  res.send('GET request received');
});

app.post('/submit', (req, res) => {
  res.send('POST request received');
});
```

Route and Query Parameters

```
app.get('/user/:id', (req, res) => {
  const userId = req.params.id;  // Route Parameter
  const searchQuery = req.query.q;  // Query Parameter

  res.send(`User ID: ${userId}, Search Query:
    ${searchQuery || 'No query provided'}`);
});
```

REQUEST AND RESPONSE OBJECTS

Request Object (req)

- **req.body**: Contains parsed data from POST requests.
- **req.params**: Access route parameters like /user/:id.
- **req.query**: Access query string parameters like /search?q=term.

Response Object (res)

- **res.send()**: Sends a response (text or data).
- **res.json()**: Sends a JSON response.
- **res.status()**: Sets the HTTP status code.
- **res.redirect()**: Redirects the request to another URL.

```
app.get('/user/:id', (req, res) => {
  res.status(200).json({ userId: req.params.id });
});
```

ERROR HANDLING

Basic Error Handling

```
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(500).send('Something broke!');
});
```

Use next(err) to pass errors to the middleware chain.

WORKING WITH FORMS

Parse form data

```
npm install body-parser --save
```

```
const bodyParser = require('body-parser');
app.use(bodyParser.urlencoded({ extended: true }));
```

Handling form submission

```
app.post('/form-submit', (req, res) => {
  const username = req.body.username;
  res.send(`Form submitted with username: ${username}`);
});
```

TEMPLATE ENGINES

Using EJS (Embedded JavaScript)

```
npm install ejs --save
```

```
app.set('view engine', 'ejs'); // Set the view engine to ejs

app.get('/user/:name', (req, res) => {
  res.render('profile', { username: req.params.name });
});
```

- **Views folder**: By default, Express looks for **templates in the views folder.**
- **Passing data**: { username: req.params.name } passes data to the template.

Example EJS template (views/profile.ejs):

```
<h1>Welcome, <%= username %></h1>
```

EXPRESS ROUTER

Modularize your routes by using the Router.

Creating a Router:

```
const express = require('express');
const router = express.Router();

router.get('/', (req, res) => {
  res.send('Welcome to the Home Page');
});

router.get('/about', (req, res) => {
  res.send('About Page');
});

module.exports = router;
```

Using Router in app.js:

```
const homeRouter = require('./routes/home');
app.use('/', homeRouter); // Mount the route
```

HANDLING FILE UPLOADS

Install the multer middleware to handle file uploads.

```
npm install multer --save
```

File upload setup:

```
const multer = require('multer');
const upload = multer({ dest: 'uploads/' });

app.post('/upload', upload.single('file'), (req, res) => {
  res.send('File uploaded successfully');
});
```