Core Concepts in Express.js

1. Routing

Handle different endpoints:

```
app.get('/about', (req, res) => {
  res.send('About Page');
});
app.post('/submit', (req, res) => {
  res.send('Form submitted');
});
```

2. Middleware

Functions that run between receiving a request and sending a response.

```
app.use(express.json()); // built-in middleware to parse JSON
```

Custom middleware:

```
app.use((req, res, next) => {
  console.log('Request received at', new Date());
  next();
});
```

3. Serving Static Files

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

Place images, CSS, or JS files in the public/ folder.

4. Handling POST Data

```
app.use(express.urlencoded({ extended: true }));
app.post('/form', (req, res) => {
  res.send(`Name: ${req.body.name}`);
});
```

5. Sending JSON

```
app.get('/data', (req, res) => {
  res.json({ name: 'Express', type: 'Framework' });
});
```

6. Error Handling

```
app.use((err, req, res, next) => {
```

```
  console.error(err.stack);
  res.status(500).send('Something broke!');
});
```

Directory Structure (Typical Express Project)

```
myapp/
├── public/
├── routes/
│   └── users.js
├── views/
│   └── index.ejs
├── app.js
├── package.json
```

Use Cases of Express.js

- Building REST APIs
- Serving web pages
- Handling user authentication
- Integrating with databases (MongoDB, MySQL)
- Backend for mobile apps and SPAs (like React, Angular)