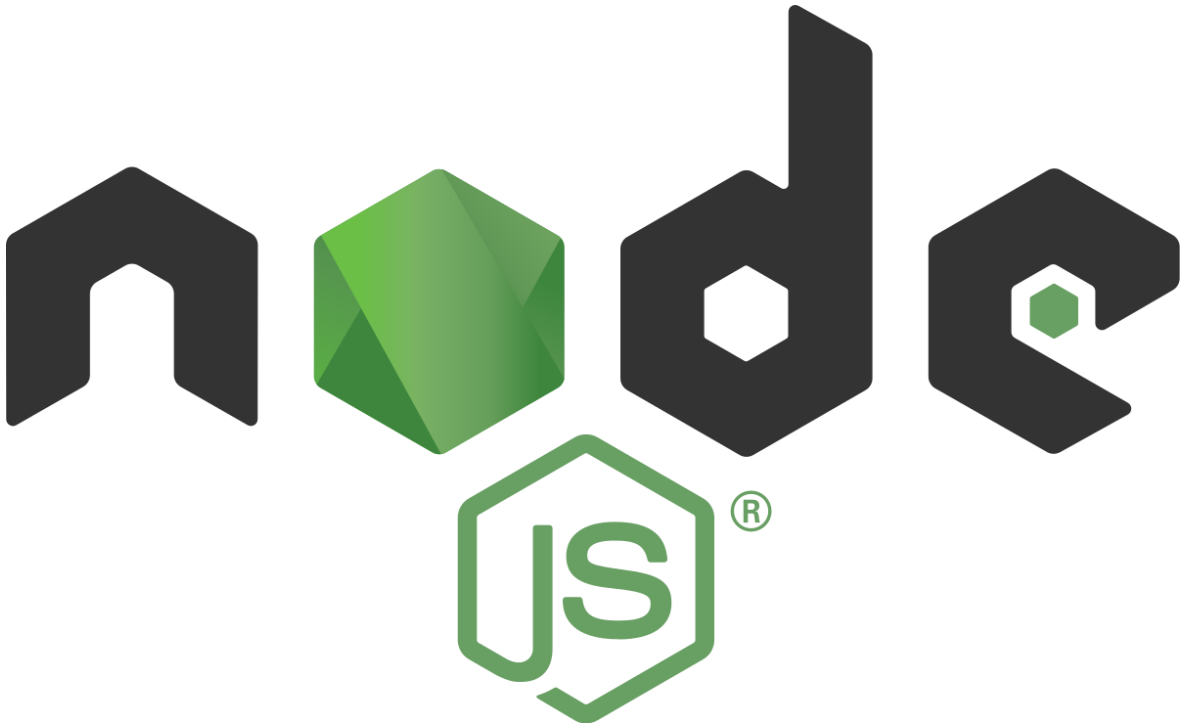
API Development using NodeJS & Express

AHMAD ROSID

Today Topic

- NodeJS
- Express Simple Server
- Authentication
- DB BREAD
- Testing
- Q&A

NodeJS



Express Installation

Express Simple Server

```
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 on port ${port}!`))
```

Express Route

```
const UserController = require('./app/controllers/UserController');

const router = express.Router()

router.post('/users', async (req, res) => {
  // logic here
})

router.post('/users/login', UserController.login)

router.get('/users/me', auth, UserController.me)

router.post('/users/me/logout', auth, UserController.logout)
```

Authentication

```
const auth = async(req, res, next) => {
    const token = req.header('Authorization').replace('Bearer ', '')
    const data = jwt.verify(token, process.env.JWT_KEY)
    try {
        const user = await User.findOne({ _id: data._id, 'tokens.token': token })
        if (!user) {
            throw new Error()
        }
        req.user = user
        req.token = token
        next()
    } catch (error) {
        res.status(401).send({ error: 'Not authorized to access this resource' })
    }
}

app.use(auth)
```

DB BREAD

Environment Configuration

```
MONGODB_URL=mongodb://127.0.0.1:27017/api_service

JWT_KEY=WinterIsComingGOT2019

PORT=3000
```

Database Model

```
const userSchema = mongoose.Schema({
    name: {
        type: String,
        required: true,
        trim: true
    },
    email: {
        type: String,
        required: true,
        unique: true,
        lowercase: true,
        validate: value => {
            if (!validator.isEmail(value)) {
                throw new Error({error: 'Invalid Email address'})
            }
        }
    },
    password: {
        type: String,
        required: true,
        minLength: 7
    },
    tokens: [{
        token: {
            type: String,
            required: true
        }
    }]
})
```

Database Browse

```
async (req, res) => {
    // Browse all users
    try {
        const users = await User.find()

        res.status(200).send({ users })
    } catch (error) {
        res.status(400).send(error)
    }
}

// https://mongoosejs.com/docs/queries.html
```

Database Create

```
// Create a new user
try {
    const user = new User(req.body)
    await user.save()
    const token = await user.generateAuthToken()
    res.status(201).send({ user, token })
} catch (error) {
    res.status(400).send(error)
}
```

Database Update

```
async (req, res) => {
    // Update user
    try {
        const user = req.user
        await user.update(req.body)
        res.status(201).send({ user })
    } catch (error) {
        res.status(400).send(error)
    }
}
```

Database Delete

```
async (req, res) => {
    // Update user
    try {
        await User.deleteOne({ _id: req.params.id });

        res.status(204).send()
    } catch (error) {
        res.status(400).send(error)
    }
}
```

Testing

Testing

```
it('POST /users create new user', async () => {
    const user = {
        name: 'Ahmad rosid',
        email: 'alahmadrosid@gmail.com',
        password: '123456'
    };
    const response = await request(server).post('/users').send(user);
    expect(response.statusCode).to.equal(201)
    expect(response.body).to.be.an.instanceof(Object);
    expect(response.body).to.include.keys(['token', 'user']);
    expect(response.body.user).to.include({
        name: 'Ahmad rosid',
        email: 'alahmadrosid@gmail.com',
    });
})
```

Q & A

Thanks