express.js

express.js is the most frequently used node library for back-end development.

Prerequisites

Install the following packages using `npm` or `bun`:

- `express`
- `nodemon` (for refreshing the server when changes are made)
- `ejs` (rendering engine for express)

Create a local working directory and run `npm init -y`. Create a file called `server.js`. Additionally, under `scripts` in `package.json`, add the line `"dev": "nodemon server.js"`.

Setting up

Add the following lines to `server.js`

```
const express = require('express')
const app = express()
```

`app` is the back-end server we'll be using.

Set `ejs` as the view engine.

```
app.set("view engine", "ejs")
```

To start rendering a page, use `app.get()`. This function takes two parameters.

```
app.get('/', (req, res) => {
        // do something
})
```

We can tinker around with `res` to display pages in the front-end.

```
app.get('/',(req, res) => {
        res.render('index')
        res.sendStatus(404) // any code can be sent
})
```

Placing HTML files

All HTML files must be placed under a sub-directory called `views`, and saved with the extension `.ejs` instead of `.html`.

While rendering pages, additional information can be sent in the form of objects:

```
res.render('index', {text: "Hi!!!" })
```

To access this information/run code with the `.ejs` page, enclose the code like this: `<%= text >`

GET requests

GET requests are served using the `app.get()` function.

```
app.get('/path/to/url', (req, res) => {
        res.send(data)
})
```

Router

Routers are mini applications that live within the app. They can be nested inside a parent route.

```
const router = express.Router()
```

A router behaves just like a normal app and supports the same functions as `app`.

Routers can be imported from elsewhere, especially when repeating naming conventions for URLs.

> users.js

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

router.get('/', (req, res) => {
        res.send("User list")
})

router.get('/new', (req, res) => {
        res.send("User new form")
})
```

```
module.exports = router
```

> server.js

```
const express = require('express')
const app = express()

app.set("view engine", "ejs")

const userRouter = require('/routes/users')

app.use('/users', userRouter)
```

In the above code, `userRouter` cleanly handles all URLs starting with `localhost:3000/users/*`.

`app.use()` has a multitude of functions, one of which is setting the URL for the router.

Dynamic URL parameters

Passing unknown values into the URL might be tricky. In express, parameters are sent by prefixing the param with a colon.

```
router.get('/:id', (req, res) => {
        res.send(`User info with ID ${req.params.id}`) // NOT
res.params.id !!
})
```