Node.js

Master in Electrical and Electronic Engineering

luis.marcelino@ipleiria.pt
2020/11/10

POLITÉCNICO DE LEIRIA
ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Overview

- Node.js
  - Event loop
  - Asynchronous programming
  - HTTP
- Module system
- NPM
- Express

Node.js

Node.js

- JavaScript runtime
  - V8 is the name of the JavaScript engine that powers Google Chrome
- Asynchronous and event-driven
- Several connections can be handled concurrently
  - Upon each connection, the callback is fired
  - If there is no work to be done, Node.js will sleep
- Event loop as a runtime construct

Event Loop

- Allows Node.js to perform non-blocking I/O operations
  - Operations are offloaded to the system kernel
  - When one of these operations completes, the kernel tells Node.js to add the callback to the poll queue
- Node.js simply enters the event loop after executing the input script
- Node.js exits the event loop when there are no more callbacks to perform

Event Loop Phases

- Timers
- Pending callbacks
- idle , prepare: only used internally.
- Poll
  - Calculates how long it should block and poll for I/O
  - Processing events in the poll queue.
- Check
- Close callbacks

(Non-) Blocking

Blocking vs. non-blocking

- Blocking methods execute synchronously

```
const fs = require('fs');
const data = fs.readFileSync('/file.md');
// blocks here until file is read
console.log(data);
moreWork(); // will run after console.log
```

- Non-blocking methods execute asynchronously

```
const fs = require('fs');
fs.readFile('/file.md', (err, data) => {
  if (err) throw err;
  console.log(data);
});
moreWork(); // will run before console.log
```

HTTP

HTTP

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
  });
```

Node.js Frameworks

- Node.js is a low-level platform
- Libraries are built upon Node.js
- Node.js can be installed in different ways

```
node script.js
```

Browser <> Node.js

Browser

- Browser provides a document and window objects
- Interacting with the DOM
- Using other Web Platform APIs like Cookies
- ES Modules standard
    - import

Node.js

- Several APIs available, like the filesystem access
- Environment control (node version)
- Uses the CommonJS module system
    - require()

process

Process

- The process object provides information about, and control over, the current process
- It's automatically available
  - process does not require a "require"
- The **process.exit()** method terminates the process synchronously

```
process.env.NODE_ENV // "development"
```

exports

Module System

- Built-in module system
  - A file can import functionality exposed by other files
  - Import the functionality exposed in a file
    - require
  - Functionality must be exposed before it can be imported by other files
  - Assign an object to **module.exports**
    - The file exports just that object
    - Exposes the object it points to
  - Add the exported object as a property of **exports**
    - Possible to export multiple objects, functions or data
    - Exposes the properties of the object it points to

Exports

```
const car = {
  brand: 'Ford',
  model: 'Fiesta'
}

module.exports = car

//..in the other file
const car = require('./car')
```

```
const car = {
  brand: 'Ford',
  model: 'Fiesta'
}

exports.car = car

//..in the other file
const items = require('./cars')

items.car
```

npm

Node package manager

- The standard package manager for Node.js.
  - It started as a way to download and manage dependencies of Node.js packages
- It consists of a command line client, also called npm
- It is an online database of public and paid-for private packages, called the npm registry
- For a project with a **package.json** file, npm installs everything the project needs
  - in the node_modules folder
- Allows to install and save a package to the **package.json** file
- Has an init option to create a **package.json** file with supplied values

express

Express

- A minimal and flexible Node.js web application framework
- Robust set of features for web and mobile applications
- HTTP utility methods and middleware

```
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}`)
})
```

Routing

- Routing refers to determining how an application responds to a client request to a particular endpoint
  - which is a URI (or path) and a specific HTTP request method (GET, POST, ...).
- Each route can have one or more handler functions
  - which are executed when the route is matched

```
app.METHOD(PATH, HANDLER)
```

```
const app = express()
// ...
app.post('/', function (req, res) {
  res.send('Got a POST request')
  })
app.put('/user', function (req, res) {
  res.send('Got a PUT request at /user')
  })
// ...
app.listen(port, () => {
 console.log(`Example app listening at
http://localhost:${port}`)
})
```

Static files

- To serve static files such as images, CSS files, and JavaScript files
- Built-in middleware function in Express: express.static

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

References

https://nodejs.dev/learn/introduction-to-nodejs

https://nodejs.org/en/docs/guides/getting-started-guide/

https://nodejs.org/en/docs/guides/event-loop-timers-and-nexttick/

https://expressjs.com/

https://expressjs.com/en/starter/examples.html