Node.js Cheat

Structure

```
example-node-project/
.gitignore
API DOC.md
app.js
node_modules/
  ...
package.json
public/
  img/
  ...
  index.html
  index.js
  index.css
```

Example Express app

```
"use strict"
/* file comment */
const express = require("express");
const app = express();
app.use(express.static("public"));
app.use(express.urlencoded({
  extended: true}));
app.use(express.json());
app.use(multer().none());
app.use(express.json());
const port = process.env.PORT||8080;
app.listen(PORT);
```

NPM Commands

npm init → initializes a new 'package.json' file interactively.

npm install → installs dependencies listed in the 'package.json' file.

npm install <package-name> used to install a specific package.

npm update → updates the dependencies listed in the 'package.json'.

npm list → Displays a tree of all installed packages in the current project.

npm config → manages npm configuration settings, allowing users to set, get or delete configuration values.

npm outdated → check for outdated dependencies in the current project, displaying a list of packages that have newer versions available.

Term	Definition
API	Application programming Interface.
Web Service	supports HTTPs requests, returning data such as JSON or plain text
Npm	Node Package Manager, used to initialize package.json files.
Package	Any object with a Json file (package.json).
Server	publically accessible machine for exchange information with clients
Client	private/public machine which requests information from server
Module	standalone package used to extend the functionality.

Core Modules.

Module	Description
fs	filesystem module with function to process data in file system.
path	provides functions to process path strings
util	provides utility functions, like util.promisify.

☆ Express Route Functions

	Description
app.get("path", middlewarefn(s));	
app.get("/", (req, res) => { });	Defines a server endpoint which accepts a valid GET request, and response objects are passed as req and res. path parameters can be specified in path with ":varname" and accessed via req.params, Query parameters by req.query.
app.get("/:city", (req, res) => { let city = req.params.city; });	
app.get("/cityData", (req, res) => { let city = req.query.city; });	
app.post("path", middlewarefn(s));	Defines server endpoint which accepts POST request. req and res for request & response. POST body via req.body. Requires middleware and matter for form Data POST req.
app.post("/addItem", (req, res) => { let itemName = req.body.name; });	

✳ Request and Response Object syntax

Request	Response
req.params (Directory of path parameters)	res.set(headername, value); used to set different response headers commonly the content type.
req.query (Captures dictionary of query parameters)	res.send(data); sets data object to the client.
req.body (Hold dictionary of POST parameters as key/value pairs)	res.end(); res.json(data);
req.cookies returns all cookies sent in the request Requires cookie-parser module	res.status(statusCode);

✳ Useful fs module functions

Functions

```
Fs.readFile(filename, "utf8", callback);
Fs.writeFile(filename, data, "utf8", callback);
Fs.appendFile(filename, data, "utf8", callback);
Fs.existsSync(filename);
Fs.readdir(path, callback);
```

path module functions
path.basename(pathStr);
path.extname(pathStr);
path.dirname(pathStr);

The glob module functions

```
glob(pattern, callback);
glob("img/*", (err, paths) => { ... });
```

The promise-mysql module

Function	Description
mysql.createConnection({ host: hostname, port: port, user: username, password: pw, database: dbname});	Returns a connected database object using config variables. If any error occurs during connection, does not return a defined database object.