Cheat Sheet for comprehensive Node.js

Basic Setup

- **Install Node.js**:

- Download from nodejs.org
- Verify installation: `node -v` and `npm -v`

- **Create a Project**:

- Initialize a new project: `npm init -y`
- Install dependencies: `npm install <package-name>`
- Install dev dependencies: `npm install <package-name> --save-dev`

- **Run a Script**:

- Run a JS file: `node <filename>.js`
- Run a script from `package.json`: `npm run <script-name>`

Core Modules

- **File System (fs)**:

- Read a file:

```
const fs = require('fs');
fs.readFile('file.txt', 'utf8', (err, data) => {
  if (err) throw err;
  console.log(data);
});
```

- Write to a file:

```
fs.writeFile('file.txt', 'Hello, Node.js', (err) => {
  if (err) throw err;
  console.log('File written');
});
```

- **HTTP**:

- Create a server:

```
const http = require('http');
const server = http.createServer((req, res) => {
```

```
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello, World\n');
});
server.listen(3000, '127.0.0.1', () => {
  console.log('Server running');
});
```

- **Path**:

- Join paths:

```
const path = require('path');
const fullPath = path.join(__dirname, 'folder', 'file.txt');
```

NPM (Node Package Manager)

- **Commands**:

- Install a package: `npm install <package-name>`
- Install globally: `npm install -g <package-name>`
- Uninstall a package: `npm uninstall <package-name>`
- Update a package: `npm update <package-name>`
- List installed packages: `npm list`

- **package.json**:

- Scripts:

```
"scripts": {
  "start": "node index.js",
  "test": "jest"
}
```

- Dependencies:

```
"dependencies": {
  "express": "^4.17.1"
}
```

Express.js

- **Basic Setup**:

```
const express = require('express');
const app = express();
app.get('/', (req, res) => {
  res.send('Hello, World');
});
app.listen(3000, () => {
  console.log('Server running on port 3000');
});
```

- **Middleware**:

- Use middleware:

```
app.use(express.json());
app.use((req, res, next) => {
  console.log('Time:', Date.now());
  next();
});
```

- **Routing**:

- Basic routes:

```
app.get('/', (req, res) => {
  res.send('GET request');
});
app.post('/', (req, res) => {
  res.send('POST request');
});
```

- Route parameters:

```
app.get('/user/:id', (req, res) => {
  res.send(`User ID: ${req.params.id}`);
});
```

Asynchronous Programming

- **Callbacks**:

```
function fetchData(callback) {
  setTimeout(() => {
    callback('Data fetched');
  }, 1000);
```

```
}
fetchData((data) => {
  console.log(data);
});
```

- **Promises**:

```
function fetchData() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve('Data fetched');
    }, 1000);
  });
}
fetchData().then(data => console.log(data));
```

- **Async/Await**:

```
async function fetchData() {
  return new Promise((resolve) => {
    setTimeout(() => {
      resolve('Data fetched');
    }, 1000);
  });
}
async function main() {
  const data = await fetchData();
  console.log(data);
}
main();
```

Error Handling

- **Try/Catch**:

```
try {
  throw new Error('Something went wrong');
} catch (error) {
  console.error(error.message);
}
```

- **Error-First Callbacks**:

```
function fetchData(callback) {
  setTimeout(() => {
    callback(new Error('Failed to fetch'), null);
  }, 1000);
}
fetchData((err, data) => {
  if (err) {
    console.error(err.message);
    return;
  }
  console.log(data);
});
```

Debugging

- **Node.js Debugger**:

- Start with debugger: `node inspect <filename>.js`
- Set breakpoints: `debugger;`
- Step through code: `n` (next), `c` (continue)

- **VS Code Debugging**:

- Launch configuration:

```
{
  "type": "node",
  "request": "launch",
  "name": "Launch Program",
  "program": "${workspaceFolder}/index.js"
}
```

Performance Optimization

- **Cluster Module**:

- Use multiple CPU cores:

```
const cluster = require('cluster');
const numCPUs = require('os').cpus().length;
if (cluster.isMaster) {
  for (let i = 0; i < numCPUs; i++) {
    cluster.fork();
  }
} else {
  // Worker code
}
```

- **Caching**:

- Use `node-cache` or `lru-cache` for in-memory caching.

Security

- **Helmet**:

- Secure HTTP headers:

```
const helmet = require('helmet');
app.use(helmet());
```

- **Sanitization**:

- Use `express-validator` to sanitize input data.

Testing

- **Jest**:

- Basic test:

```
test('adds 1 + 2 to equal 3', () => {
  expect(1 + 2).toBe(3);
});
```

- **Mocha & Chai**:

- Basic test:

```
const assert = require('chai').assert;
describe('Array', function() {
  it('should return -1 when the value is not present', function() {
    assert.equal([1, 2, 3].indexOf(4), -1);
  });
});
```

Environment Variables

- **Dotenv**:

- Load environment variables:

```
require('dotenv').config();
console.log(process.env.API_KEY);
```

Event Loop

- **Phases**:

- Timers: `setTimeout`, `setInterval`
- I/O callbacks
- Idle, prepare
- Poll: new I/O events
- Check: `setImmediate`
- Close callbacks

Best Practices

- **Code Structure**:

- Modularize code using modules and exports.
- Use environment variables for configuration.

- **Error Handling**:

- Always handle errors in asynchronous code.
- Use `try/catch` for synchronous code.

- **Security**:

- Validate and sanitize input data.
- Use HTTPS for secure communication.

- **Performance**:

- Optimize database queries.
- Use caching for frequently accessed data.

Additional Resources

- **Documentation**:

- [Node.js Documentation](https://nodejs.org/en/docs/)
- [Express.js Documentation](https://expressjs.com/)

- **Community**:

- [Stack Overflow](https://stackoverflow.com/questions/tagged/node.js)
- [GitHub](https://github.com/nodejs)

- **Learning**:

- [NodeSchool](https://nodeschool.io/)
- [MDN Web Docs](https://developer.mozilla.org/en-US/docs/Learn/Server-side/Express_Nodejs)

This cheat sheet provides a comprehensive overview of essential Node.js features, setup, and best practices. Use it as a quick reference for developing robust and efficient Node.js applications.

By Ahmed Baheeg Khorshid

ver 1.0