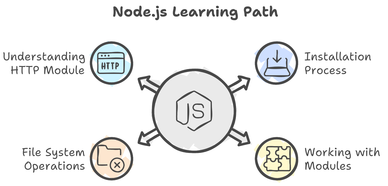
Introduction to Node.js Basics

This document provides an overview of the fundamental concepts of Node.js, a powerful JavaScript runtime built on Chrome's V8 engine. It covers the installation process of Node.js and npm (Node Package Manager), working with modules, performing file system operations, and understanding the HTTP module. This guide is designed for beginners who want to get started with Node.js and build server-side applications.



Introduction to Node.js

Node.js is an open-source, cross-platform JavaScript runtime environment that allows developers to execute JavaScript code on the server side. It uses an event-driven, non-blocking I/O model, making it efficient and suitable for building scalable network applications. With Node.js, developers can use JavaScript for both client-side and server-side programming, streamlining the development process.

Installing Node.js and npm

To get started with Node.js, you need to install it along with npm, which is included with the Node.js installation. Follow these steps:

1. **Download Node.js**: Visit the [official Node.js website](https://nodejs.org/) and download the installer for your operating system (Windows, macOS, or Linux).
2. **Run the Installer**: Follow the installation instructions. Make sure to check the box that says "Install npm package manager" during the installation process.
3. **Verify Installation**: Open your terminal or command prompt and run the following commands to verify that Node.js and npm are installed correctly:

```
node -v
npm -v
```

Working with Modules

Node.js has a built-in module system that allows you to organize your code into reusable components. You can create your own modules or use built-in modules. Here's how to work with modules:

1. **Creating a Module**: Create a new JavaScript file (e.g., **myModule.js**) and export a function:

```
// myModule.js
function greet(name) {
    return `Hello, ${name}!`;
}
module.exports = greet;
```

2. **Importing a Module**: In another file (e.g., **app.js**), import and use the module:

```
// app.js
const greet = require('./myModule');
console.log(greet('World')); // Output: Hello, World!
```

File System Operations

Node.js provides a **fs** module that allows you to interact with the file system. You can read, write, and manipulate files easily. Here are some basic file system operations:

1. **Reading a File**:

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

fs.readFile('example.txt', 'utf8', [err, data] => {

```
    if (err) {
```

```
        console.error(err);
```

```
        return;
```

```
    }
```

```
    console.log(data);
```

});

2. **Writing to a File**:

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

fs.writeFile('output.txt', 'Hello, Node.js!', [err] => {

```
    if (err) {
```

```
        console.error(err);
```

```
        return;
```

```
    }
```

```
    console.log('File has been written.');
```

});

Understanding HTTP Module

The HTTP module in Node.js allows you to create web servers and handle HTTP requests and responses. Here's a simple example of creating a basic HTTP 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');
});

const PORT = 3000;
server.listen(PORT, () => {
    console.log(`Server running at http://localhost:${PORT}/`);
});
```

In this example, the server listens on port 3000 and responds with "Hello, World!" to any incoming request.

Conclusion

This document has introduced you to the basics of Node.js, including installation, module usage, file system operations, and the HTTP module. With this foundational knowledge, you can start building your own server-side applications using Node.js. As you continue to explore Node.js, you'll discover its vast ecosystem and the many libraries available to enhance your development experience.