Node.js

Node.js is a **JavaScript runtime** that allows you to run JavaScript code outside of a web browser. It's used to build fast, scalable network applications, especially web servers.

Key Points:

- **Server-side**: Node.js is used to build the backend (server-side) of applications.
- **Non-blocking**: It uses a non-blocking, event-driven architecture, which makes it very efficient at handling multiple tasks at once (like serving many users at the same time).
- **Uses JavaScript**: If you're familiar with JavaScript for front-end web development, you can use the same language for backend tasks with Node.js.

Node. Modules

Node. provides a wide range of built-in modules that allow developers to build various applications, from servers to utility tools. Here are some key modules and their uses:

1. **`http`**
   - **Purpose**: To create and manage HTTP servers and handle HTTP requests and responses.
   - **Use Case**: Building web servers.
   - Example:

```
const http = require('http');
const server = http.createServer((req, res) => {
  res.writeHead(200, {'Content-Type': 'text/plain'});
  res.end('Hello World\n');
});
server.listen(3000, () => {
  console.log('Server running at http://localhost:3000/');
});
```

2. **`fs` (File System)**
   - **Purpose**: To interact with the file system, including reading, writing, and deleting files.
   - **Use Case**: Handling file operations like saving logs, reading configurations, etc.
   - Example:

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

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

3. **`path`**
   - **Purpose**: To handle file and directory paths.
   - **Use Case**: Joining, resolving, and normalizing file paths.
   - Example:

```
const path = require('path');
const filePath = path.join(__dirname, 'example.txt');
console.log(filePath);
```

4. **`events`**
   - **Purpose**: To handle event-driven programming, allowing you to create, listen, and trigger custom events.
   - **Use Case**: Implementing event listeners in server applications.
   - Example:

```
const EventEmitter = require('events');
const eventEmitter = new EventEmitter();
eventEmitter.on('start', () => {
  console.log('Start event triggered');
});
eventEmitter.emit('start');
```

5. **`os`**
   - **Purpose**: Provides information about the operating system.
   - **Use Case**: Fetching system-level data like CPU information, memory usage, etc.
   - Example:

```
const os = require('os');
console.log(`Free Memory: ${os.freemem()}`);
console.log(`Platform: ${os.platform()}`);
```

6. **`url`**
   - **Purpose**: To parse and work with URLs.
   - **Use Case**: Handling query parameters and URL manipulation.
   - Example:

```
const url = require('url');
const parsedUrl =
url.parse('https://example.com/profile?name=John');
```

```
console.log(parsedUrl.query); // Output: 'name=John'
```

7. **`crypto`**
   - **Purpose**: To provide cryptographic functionalities like hashing, encryption, and decryption.
   - **Use Case**: Password hashing, secure data transmission.
   - Example:

```
const crypto = require('crypto');
const hash =
crypto.createHash('sha256').update('password').digest('hex');
console.log(hash);
```

8. **`querystring`**
   - **Purpose**: To work with query strings, allowing you to parse and stringify them.
   - **Use Case**: Parsing URL query strings in GET requests.
   - Example:

```
const querystring = require('querystring');
const params = querystring.parse('name=John&age=30');
console.log(params.name); // Output: John
```

9. **`child_process`**
   - **Purpose**: To create and manage child processes, allowing you to run other programs from within Node..
   - **Use Case**: Executing shell commands or scripts from a Node. application.
   - Example:


```
const { exec } = require('child_process');
exec('ls', (err, stdout, stderr) => {
  if (err) throw err;
  console.log(stdout);
});
```

10. **`stream`**
    - **Purpose**: To handle streaming data, like reading large files or streaming content from a server.
    - **Use Case**: Working with real-time data streams, such as video or audio.
    - Example:

```
const fs = require('fs');
const stream = fs.createReadStream('largefile.txt');
stream.on('data', (chunk) => {
  console.log(chunk);});
```

Rundown

- **Node.**: The core runtime environment to run JavaScript on the server. It includes built-in modules like `http`, `fs`, `crypto`, etc., to build scalable applications.
- **Express.**: A web framework built on top of Node. to simplify web development tasks, such as routing and handling requests.