Node.js Cheat Sheet

1) Event - Driven Architecture:

- Node.js utilizes an event - driven architecture, where actions or events trigger callbacks.
- This approach enables Node.js to handle multiple connections simultaneously without blocking the execution thread.

2) Asynchronous Programming:-

- Asynchronous programming is fundamental in Node.js, allowing non- blocking I/O operations.
- It uses callbacks, promises & async/await to manage asynchronous tasks effectively.

) npm (Node Package manager) :-

- npm is the default package manager for node.js, providing access to a vast ecosystem of reusable libraries & tools.

) Modules:-

- Node.js follows commonJS module system, allowing code modularization & reusability.
- Modules encapsulate code within files & 'require()' function is used to import modules into other files.

CommonJS module System:-

- defines standardized way to structure code into reusable modules.

6) Streams :-

- are key feature in Node.js for handling data flow.
- allow reading from & writing to data sources in chunks, improving efficiency & scalibility for I/O operations

7) Error Handling :-

- crucial due to its asynchronous nature.
- Techniques such as try-catch blocks, error-first callbacks, & error events are used.

8) Event Emitters :-

- Node.js includes an EventEmitter class to implement publish-subscribe pattern.
- enables communication b/w different parts of an application by emitting & listening to named events.

9) HTTP module :-

- facilitates creating web servers & handling HTTP requests & responses.

10) Middleware :-

- Middleware functions are essential in Node.js frameworks like Express.js for handling requests & responses in a modular way.

11) Security Considerations :-

- Node.js applications should follow best practices for security, including input validation / sanitization & secure coding practices.
- Modules such as Helmet.js can be used to enhance security by setting various HTTP headers.

12) Performance optimization :-

- crucial especially in high-traffic scenarios.
- using techniques like caching, load balancing, optimizing I/O operations

13) Debugging & Testing :-

- Node.js provides tools & frameworks for debugging & testing, including built-in debugging support via Node.js debugger & frameworks like Mocha & Jest for unit & integration testing.

14) Scalability :-

- Node.js applications can achieve scalability through techniques such as clustering, horizontal scaling & microservices architecture.
- Load balancing & asynchronous event-driven architecture contribute to it.

15) Deployment :-

- Node.js applications can be deployed on various platforms including traditional servers, cloud services like AWS & Azure, & containerized environments like Docker & Kubernetes.