

```
App {  
  
static void main(String[] args) throws Exc  
em.out.println("Hello, World!");
```

Introduction to Node.js

A comprehensive overview of Node.js, its features, and applications

규태 김

What is Node.js? An Overview

Understanding the Core Features of Node.js for Efficient Development

1

Asynchronous and Event-Driven

Node.js utilizes non-blocking I/O operations, enabling it to handle multiple tasks simultaneously, making it both efficient and lightweight.



2

Single-Threaded Architecture

Despite being single-threaded, Node.js effectively manages numerous connections at once through its event loop mechanism.



3

Robust Package Ecosystem

The Node Package Manager (npm) provides a rich library of open-source packages, significantly boosting development efficiency and capabilities.



History and Evolution of Node.js

A detailed look at the key milestones in Node.js development.

2009

Initial release of Node.js.

Node.js was launched, introducing a new way to build scalable network applications using JavaScript on the server side.

Introduction of npm.

The Node Package Manager (npm) was introduced, facilitating package management and making it easier for developers to share and reuse code.

2010

2011

Widespread adoption.

Node.js gained significant traction, with many companies adopting it for production environments, enhancing its credibility and demand.

Formation of the OpenJS Foundation.

The Node.js Foundation merged with the JS Foundation to form the OpenJS Foundation, aiming to support the growth and adoption of JavaScript and Node.js.

2018

Core Concepts of Node.js

Understanding the Key Mechanisms and Tools in Node.js Development

Event Loop

1 The mechanism that allows Node.js to perform non-blocking operations, making it efficient for I/O-bound tasks.

Modules

2 Node.js uses the CommonJS module system, enabling developers to organize code into reusable modules.

NPM (Node Package Manager)

3 A critical tool for sharing and managing dependencies, facilitating easier project setup and maintenance.

The Architecture of Node.js

An Overview of Key Components in Node.js Framework



V8 Engine

The V8 Engine converts JavaScript code into machine code, enhancing execution speed and performance.



Libuv

Libuv is responsible for handling asynchronous I/O operations, which allows Node.js to perform non-blocking operations efficiently.



Event Loop

The Event Loop manages the execution of multiple requests simultaneously without blocking, ensuring high concurrency.

Setting Up a Node.js Environment

A step-by-step guide for beginners to get started with Node.js



Install Node.js

Download from the official Node.js website to get the latest version.



Verify Installation

Use the command `node -v` in your command line to check the installed version.



Use npm

Ensure npm is installed by running `npm -v` to check the version of npm.



Next Steps

Create a simple 'Hello World' project to familiarize yourself with the environment.

Building Your First Node.js Application

A step-by-step guide to creating a basic Node.js server

1

Create a basic HTTP server with Node.js

Utilize Node.js to set up a simple HTTP server that can handle requests.

2

Server listens on port 3000

Configure the server to listen for incoming connections on port 3000, a common practice in Node.js applications.

Exploring NPM and Package Management

Essential Commands for Managing Node.js Packages



npm init

Initializes a new Node.js project, creating a package.json file.



npm install

Installs a specific package from the NPM registry into your project.



npm update

Updates the installed packages to their latest versions.



npm
763K Packages



PyPI
135K Packages



CocoaPods
44.3K Packages



Meteor
13.4K Packages



Hex
6.4K Packages



Emacs
4.23K Packages



Julia
2.27K Packages



Racket
1.24K Packages



Alcatraz
465 Packages



Packagist
215K Packages



NuGet
119K Packages



CPAN
35.6K Packages



CRAN
13.3K Packages



Puppet
5.69K Packages



SwiftPM
4.15K Packages



Sublime
1.97K Packages



Haxelib
1.19K Packages



PureScript
237 Packages

Node.js Application Development Best Practices

Essential techniques for reliable
and efficient Node.js applications

Error Handling

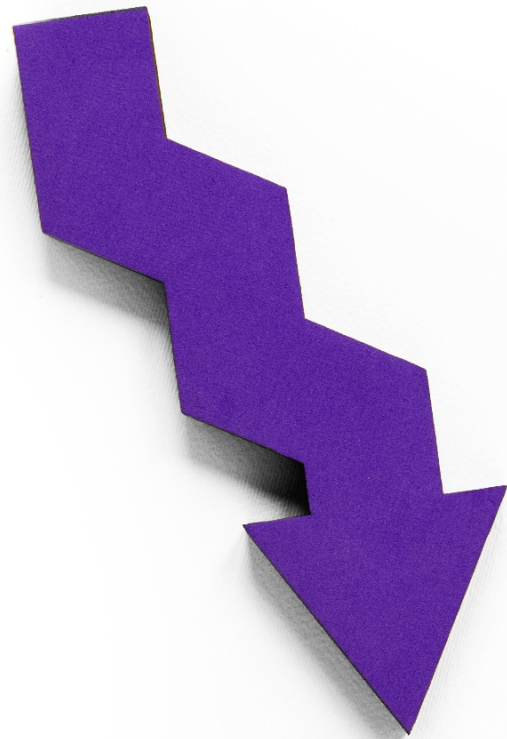
Always handle errors to prevent application crashes, ensuring stability.

Use Environment Variables

Store sensitive information outside your codebase to enhance security.

Optimize Performance

Utilize asynchronous programming techniques to improve application performance.



Real-World Applications of Node.js

Exploring the versatility of Node.js in various domains

1

Web Development

Frameworks like Express.js facilitate rapid web app development.

2

APIs

Node.js is ideal for building RESTful APIs due to its non-blocking nature.

3

Streaming Services

Applications like Netflix utilize Node.js for its performance.

The Future of Node.js

Exploring advancements and community growth in Node.js technology

- **Performance Enhancements**

Ongoing updates to the V8 engine improve speed and efficiency, making Node.js applications faster.

- **Community Growth**

Increasing adoption among developers fosters a vibrant ecosystem, leading to greater collaboration and innovation.

Unlock the Power of Node.js

Join our session to master Node.js for scalable applications.

