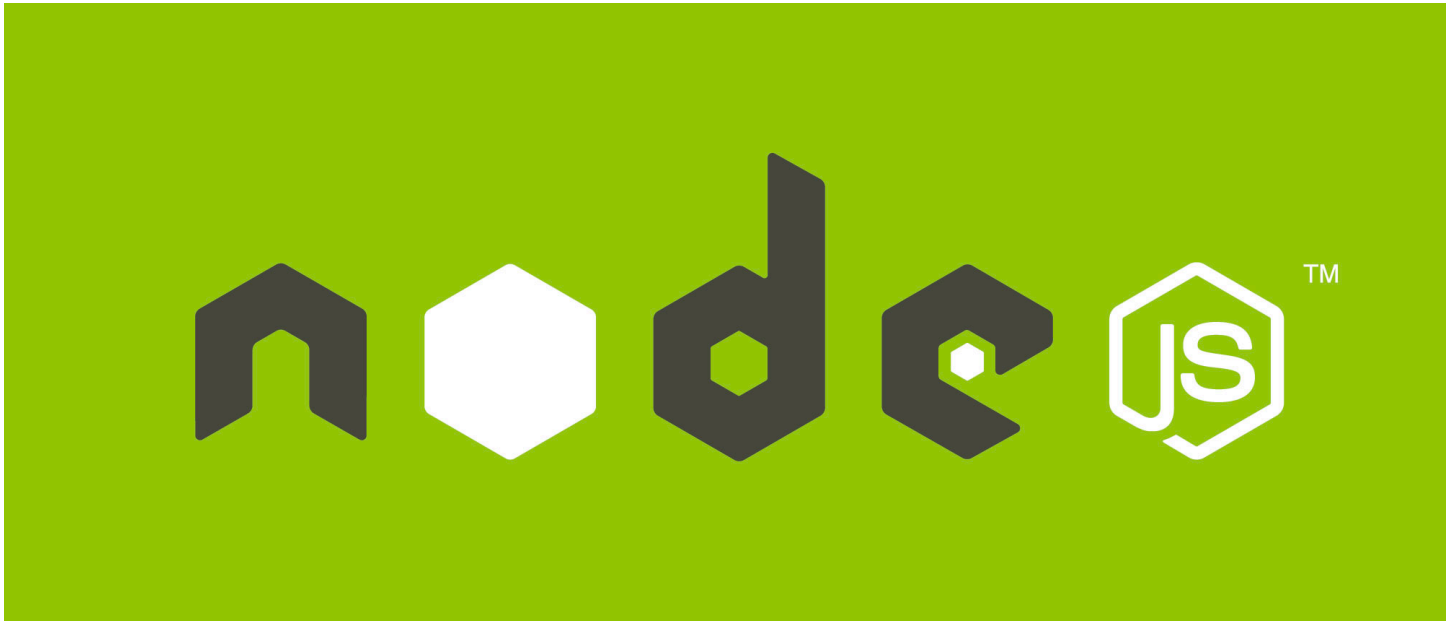
NodeJS

02

Presented by
Subhash EP

Subhash


node
JS
TM

What is Node.js

- in 2009
- by Ryan Dahl
- address concurrency issues
- dealing with web services.

V8 JavaScript engine

- Google created V8 JavaScript engine
- for the Chrome web browser
- highly optimized for web traffic.

Birth of Node.js

- Dahl created Node.js on top of V8
- as a server-side environment
  - that matches the client-side environment.

Node.js

- scalable server-side environment
- bridge the gap between client and server
- navigate back and forth between client and server code
- and can even reuse code between the two environments.

Node.js ecosystem

- new extensions are written all the time
- The Node.js environment is easy to install, configure, and deploy.
- In only a matter of an hour or two, you can have a Node.js webserver up and running

File-Based Module System

- Kevin Dongaoor created CommonJS in 2009 with the goal to specify an ecosystem for JavaScript modules on the server.
- Node.js follows the CommonJS module specification

File-Based Module System

- Each file is its own module.
- Each file has access to the current module definition using the module variable.
- The export of the current module is determined by the module.exports variable.
- To import a module, use the globally available require function.

Require Demo

Node.js require Function

- The Node.js require function is the main way of importing a module into the current file.
- There are three kinds of modules in Node.js:
  - core modules,
  - file modules, and
  - external node_modules,
- all of them use the require function

Node.js require Function

- When we make a require call with a relative path—for example, something like require('./filename') or require('../foldername/filename')
  - Node.js runs the destination JavaScript file in a new scope and returns whatever was the final value for module.exports in that file.
- This is the basis of file modules.

Node.js Is Safe

- Modules in many programming environments are not safe and pollute the global scope.
  - A simple example of this is PHP.
- But Node.Js is Safe
  - Conditionally Load a Module

Node.JS - Blocking

- The require function blocks further code execution until the module has been loaded.
- This means that the code following the require call is not executed until the module has been loaded and executed.
- This allows you to avoid providing an unnecessary callback like you need to do for all async I/O in Node.js

Other Features

- Cached
- Shared State
- Object Factories
- module.exports
- Exports Alias

Other Features Demo

That's all

End of Session

©

subhash.ep@gmail.com
linkedin.com/in/subhashep