



Node JS

The Complete Node.js Course [80 Hrs]

Instructor: Pondit Team

E-Mail:

Phone:

Overview

আইটি ইন্ডাস্ট্রির জন্য দক্ষ জনশক্তি তৈরিতে পন্ডিট স্পন্সরশীপ প্রদান করছে। যার পরিপ্রেক্ষিতে ইউনিভার্সিটির ছাত্রছাত্রীরা একটি পরীক্ষায় ভালো রেজাল্টের মাধ্যমে ১০০% পর্যন্ত ফ্রি কোর্স করার সুযোগ পাচ্ছেন।

পরীক্ষার মাধ্যমে আপনি ১০০% পর্যন্ত স্পন্সরশীপ পাবেন তবে আপনাকে অবশ্যই পরীক্ষায় ৮০% মার্ক্স পেতে হবে। ১০০% স্পন্সরশীপ না পেলে পরীক্ষার রেজাল্টের উপর ভিত্তি করে ৭৫% অথবা ৫০% ডিসকাউন্ট পাবেন।

নোড জে এস জাভাস্ক্রিপ্ট এর উপর ভিত্তি করে গড়ে ওঠা খুবই শক্তিশালী একটি ফ্রেমওয়ার্ক বা প্ল্যাটফর্ম যা গুগল ক্রোমের জাভাস্ক্রিপ্ট ভি এইট ইঞ্জিনে (V8 Engine) এ তৈরি করা হয়েছে। এই নোড জে এস ব্যবহার হয় বিভিন্ন ধরনের ওয়েব অ্যাপ্লিকেশন তৈরিতে যেমনঃ অনলাইনে ভিডিও দেখার সাইট তৈরিতে, এক পেইজের অ্যাপ্লিকেশন তৈরিতে বা অন্যান্য ওয়েব অ্যাপ্লিকেশন তৈরিতে। এটি একটি open source, সম্পূর্ণ ফ্রী এবং এটা সারা বিশ্বের হাজার হাজার ডেভেলপার এর ব্যবহারের একটি উৎস। সবচেয়ে পপুলার প্ল্যাটফর্মগুলোর মধ্যে এটি অন্যতম।

যারা নতুন তাদের মধ্যে অনেকেই ভাবে নোড জেএস হয়তো কোন প্রোগ্রামিং ল্যাঙ্গুয়েজ বা ফ্রেমওয়ার্ক! কিন্তু এটা সম্পূর্ণ ভুল। নোড জেএস কোন প্রোগ্রামিং ল্যাঙ্গুয়েজ বা ফ্রেমওয়ার্ক নয়। নোড জেএস হচ্ছে একটি জাভাস্ক্রিপ্ট Run-Time Environment। Run-Time Environment এর কাজ হচ্ছে নির্দিষ্ট একটি প্রোগ্রামিং ল্যাঙ্গুয়েজ এর কোড গুলোকে রান করা।

আমরা জানি জে জাভাস্ক্রিপ্ট তৈরী করা হয়েছিলো শুধুমাত্র ব্রাউজার এর জন্য। আগে শুধুমাত্র ওয়েবসাইট Interactive করার জন্য ব্যবহার করা হতো জাভাস্ক্রিপ্ট। একমাত্র ব্রাউজারই বুঝতে পারতো জাভাস্ক্রিপ্ট কোড। কিন্তু ২০০৯ সালে Ryan Dahl নামে একজন সফটওয়্যার ইঞ্জিনিয়ার বাজারে নিয়ে আসে নোড জেএস আর পাল্টে দেয় সবকিছু! নোড জেএস আসার পর এখন জাভাস্ক্রিপ্ট দিয়ে অনেক কিছু করা যায়! প্রত্যেকটা ব্রাউজার এর মধ্যে একটি করে জাভাস্ক্রিপ্ট ইঞ্জিন থাকে যার কাজ হচ্ছে জাভাস্ক্রিপ্ট কোড কে মেশিন কোডে রূপান্তর করা। Google Chrome এর আছে V8 Engine আর Mozilla Firefox এর আছে SpiderMonkey এবং প্রত্যেকটি ব্রাউজার এর জন্য রয়েছে আলাদা আলাদা জাভাস্ক্রিপ্ট ইঞ্জিন। সব থেকে ফাস্ট হচ্ছে V8 Engine এবং এটি ওপেন সোর্স। Ryan Dahl এর মাধ্যম চমৎকার একটি আইডিয়া আসলো! তিনি V8 engine কে সাথে নিয়ে C এবং C++ এর মিশ্রণে তৈরী করে ফেললেন নোড জেএস।

নোড জেএস ব্যবহার করে event-driven, non-blocking I/O model যা নোড জেএস কে পরিণত করে লাইটওয়েট এবং খুবই শক্তিশালী! Asynchronous কাজে নোড এর বিকল্প টাফ। তাই নোড খুবই দ্রুত পপুলার হওয়া শুরু করে। বাজারে বর্তমানে তুমুল জনপ্রিয়। নোড ডেভেলপারদের ডিম্যান্ডও মার্কেটে বেশ।

Prerequisites

প্রোগ্রামিং নলেজ থাকলে ভালো

Course Information

- Duration : 80 Hours
- Day : রবিবার এবং মঙ্গলবার
- Time : রাত ৯.০০টা থেকে ১১.০০টা পর্যন্ত
- Reg. Start : Jan 26, 2022
- Reg. End : May 14, 2022
- Class Start : May 15, 2022

Audience

- শিক্ষার্থী (যারা সফটওয়্যার ডেভেলপমেন্টে ক্যারিয়ার গড়তে চান)
- চাকুরীজীবী

Evaluation

Curriculum


1. Basic JavaScript

- 1.1. ## Introduction:
- 1.2. Javascript introduction
- 1.3. Job field
- 1.4. Tools and technologies in javascript
- 1.5. Installing necessary tools
- 1.6. Variables
- 1.7. Constants
- 1.8. Primitive types
- 1.9. Objects
- 1.10. Arrays
- 1.11. Functions
- 1.12. Loop
- 1.13. Operators
- 1.14. Control flow
- 1.15. Factory functions
- 1.16. Constructor functions
- 1.17. Value vs reference types
- 1.18. Cloning an object
- 1.19. Garbage collections
- 1.20. Math
- 1.21. String
- 1.22. Template literals
- 1.23. Map
- 1.24. Reduce
- 1.25. forEach
- 1.26. Filter
- 1.27. Elements or objects finding
- 1.28. Prototype chaining
- 1.29. ## Advanced Topics:
- 1.30. Execution context
- 1.31. -- Memory component
- 1.32. -- Code component
- 1.33. Execution context phases
- 1.34. -- Creation phase
- 1.35. -- Execution phase
- 1.36. Call stack
- 1.37. Hoisting
- 1.38. Scope
- 1.39. Lexical environment
- 1.40. Scope chain
- 1.41. Temporal dead zone

- 1.42. Block
- 1.43. Shadowing
- 1.44. Closure
- 1.45. Browser
- 1.46. Callback queue or task queue
- 1.47. Event loop
- 1.48. Microtask queue
- 1.49. Microtasks
- 1.50. Starvation of callback functions
- 1.51. ## ES6 features:
- 1.52. Let vs const
- 1.53. Objects
- 1.54. This keyword
- 1.55. Binding this
- 1.56. Arrow function
- 1.57. Arrow function and this
- 1.58. Spread operator
- 1.59. Classes
- 1.60. Inheritance
- 1.61. Modules
- 1.62. Named and defaults exports
- 1.63. ## Extra topics:
- 1.64. Promise
- 1.65. Promise chaining
- 1.66. Async/await
- 1.67. Currying
- 1.68. Modules export import

2. Node JS

- 2.1. ## Introduction:
- 2.2. What is node?
- 2.3. Node architecture
- 2.4. How node works
- 2.5. Installing the environments
- 2.6. ## Node modules system:
- 2.7. Global objects
- 2.8. Modules
- 2.9. Module creation
- 2.10. Loading a module
- 2.11. Module wrapper function
- 2.12. Path module
- 2.13. OS module
- 2.14. File system module
- 2.15. Events module
- 2.16. Event arguments
- 2.17. Extending EventEmitter
- 2.18. HTTP module
- 2.19. ## Node Package Manager(NPM)
- 2.20. what is package.josn
- 2.21. how to install a node package
- 2.22. Semantic versioning



- 2.23. DevDependencies
- 2.24. Uninstalling a package
- 2.25. Global Package
- 2.26. ## Restful apis:
- 2.27. Restful services
- 2.28. What is Express.js
- 2.29. Basic server
- 2.30. Nodemon
- 2.31. Environment variables
- 2.32. Route parameters
- 2.33. Handling HTTP GET Requests
- 2.34. Handling HTTP POST Requests
- 2.35. Handling HTTP PUT Requests
- 2.36. Handling HTTP PATCH Requests
- 2.37. Handling HTTP DELETE Requests
- 2.38. Calling endpoints using postman
- 2.39. Input validation
- 2.40. A CRUD simple project
- 2.41. ## Express Advanced Topics:
- 2.42. Middleware
- 2.43. Custom middleware
- 2.44. Built in middleware
- 2.45. Third Party middleware
- 2.46. Environments
- 2.47. Configuration
- 2.48. Debugging
- 2.49. Template Engine
- 2.50. Database Integration
- 2.51. Authentication
- 2.52. Structuring Express application
- 2.53. Restructure the previous project
- 2.54. ## Asynchronous Javascript:
- 2.55. Synchronous vs Asynchronous
- 2.56. Callback
- 2.57. Callback Hell
- 2.58. Promises
- 2.59. Async and await
- 2.60. ## CRUD operations using mongoose:
- 2.61. Introduction to Mongodb
- 2.62. Installing mongodb
- 2.63. Connecting to mongodb
- 2.64. Schemas
- 2.65. Models
- 2.66. Saving a document
- 2.67. Query documents
- 2.68. Comparison query operators
- 2.69. Logical Query operators
- 2.70. Counting
- 2.71. Pagination
- 2.72. Updating query
- 2.73. Deleting query

2.74. Project restructured with mongoose
2.75. ## Mongo Data Validation:
2.76. Validation
2.77. Built in validators
2.78. Custom Validators
2.79. Async validators
2.80. Validation errors
2.81. SchemaType options
2.82. Restructured previous Project with validations
2.83. Another Project
2.84. ## Mongoose- Modeling Relationships between
Connected Data:
2.85. Modeling Relationships
2.86. Referencing documents
2.87. Embedded documents
2.88. Transaction
2.89. Validating objectId
2.90. Project – Building a project
2.91. ## Authentication and Authorization:
2.92. Creating the user model
2.93. Registering a user
2.94. Password Hashing
2.95. Authenticating users
2.96. JSON web tokens (jwt)
2.97. Generating authentication tokens
2.98. Storing secrets in environment variables
2.99. Setting response Headers
2.100. Encapsulating logic in mongoose models
2.101. Authorization middleware
2.102. Protecting routes
2.103. Getting the current user
2.104. Logging out
2.105. Role based authorization
2.106. ## Handling and logging errors:
2.107. Handling rejected promises
2.108. Express error middleware
2.109. Removing try catch block
2.110. Express Async errors
2.111. Logging errors
2.112. Uncaught exceptions
2.113. Unhandled promise rejection
2.114. ## Unit testing and Integration testing:
2.115. What is automated testing
2.116. Unit testing vs integration testing vs end to end
testing
2.117. Configuring the app to be tested
2.118. Writing testing
2.119. Deployment:
2.120. Preparing the app for Production
2.121. Introduction to Heroku
2.122. Preparing the app for Heroku

- 2.123. Adding the code to a git repository
- 2.124. Deploying to the Heroku
- 2.125. Setting Environments variables

