Department of Computer Engineering

GovernmentPolytechnicfor Girls, Surat

March–2021 Vol.-18

TechTrends

E-Newsletter

Vision:

To empower girls of diploma computer engineering to excel in IT Industries and serve the society.

Mission:

- To strive for academic excellence and professional competence among students and staff.
- To encourage innovative ideas among students to enhance their entrepreneurship skills.
- To provide high tech educational resources and supportive infrastructure.

Follow us on

gpgdceenewsletter@gmail.com

gpgdceenewsletter@gmail.com

Introduction

Node.js isan open-source, cross-platform back-end Java Script runtime environment that runs on the Chrome V8 engine and executes JavaScript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting—running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, Node.js represents a "JavaScript everywhere" paradigm, unifying web-application development around a single programming language, rather than different languages for server-side and client-side scripts.

Mr. S. I. Patel
Lecturer,
Department of
Computer Engineering

Though .js is the standard filename extension for JavaScript code, the name "Node.js" doesn't refer to a particular file in this context and is merely the name of the product. Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games)

The Node.js distributed development project was previously governed by the Node.js Foundation, and has now merged with the JS Foundation to form the OpenJS Foundation, which is facilitated by the Linux Foundation's Collaborative Projects program.

History

Node.js was written initially by Ryan Dahl in 2009,about thirteen years after the introduction of the first server-side JavaScript environment, Netscape's LiveWire Pro Web.[The initial release supported only Linux and Mac OS X. Its development and maintenance was led by Dahl and later sponsored by Joyent.

Dahl criticized the limited possibilities of the most popular web server in 2009, Apache HTTP Server, to handle a lot of concurrent connections (up to 10,000 and more) and the most common way of creating code (sequential programming), when code either blocked the entire process or implied multiple execution stacks in the case of simultaneous connections.

Dahl demonstrated the project at the inaugural European JSConf on 8 November 2009. Node.js combined Google's V8 JavaScript engine, an event loop, and a low-level I/O API.

In January 2010, a package manager was introduced for the Node.js environment called npm The package manager makes it easier for programmers to publish and share source code of Node.js packages and is designed to simplify installation, updating, and uninstallation of packages.

In June 2011, Microsoft and Joyent implemented a native Windows version of Node.js. The first Node.js build supporting Windows was released in July 2011.

In January 2012, Dahl stepped aside, promoting coworker and npm creator Isaac Schlueter to manage the project. In January 2014, Schlueter announced that Timothy J. Fontaine would lead the project.

In December 2014, Fedor Indutny started io.js, a fork of Node.js. Due to the internal conflict over Joyent's governance, io.js was created as an open governance alternative with a separate technical committee.Unlike Node.js,the authors planned to keep io.js up-to-date with the latest releases of the Google V8 JavaScript engine.

In February 2015, the intent to form a neutral Node.js Foundation was announced. By June 2015, the Node.js and io.js communities voted to work together under the Node.js Foundation.

In September 2015, Node.js v0.12 and io.js v3.3 were merged back together into Node v4.0.[This merge brought V8 ES6 features into Node.js and a long-term support release cycle.As of 2016, the io.js website recommends that developers switch back to Node.js and that no further releases of io.js are planned due to the merge.

In 2019, the JS Foundation and Node.js Foundation merged to form the OpenJS Foundation.

What is Node.js?

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js = Runtime Environment + JavaScript Library

Features of Node.js

Asynchronous and Event Driven – All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.

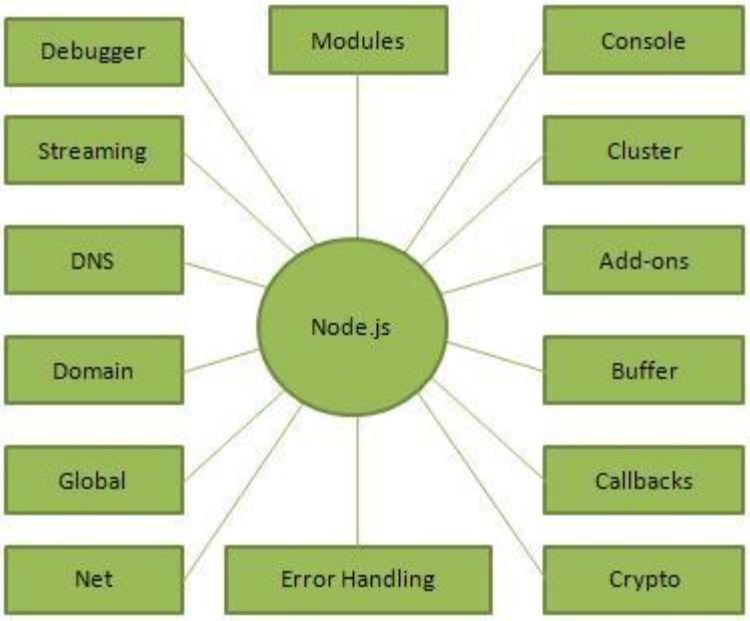
Very Fast – Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution

Single Threaded but Highly Scalable – Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.

No Buffering – Node.js applications never buffer any data. These applications simply output the data in chunks.

Concepts

The following diagram depicts some important parts of Node.js which we will discuss in detail in the subsequent chapters.



Where to Use Node.js?

Following are the areas where Node.js is proving itself as a perfect technology partner.

- I/O bound Applications
- Data Streaming Applications
- Data Intensive Real-time Applications (DIRT)
- JSON APIs based Applications
- Single Page Applications

Technical details

Internals

Node.js uses libuv underhood to handle asynchronous events. Libuv is an abstraction layer for network and file system functionality on both Windows and POSIX-based systems such as Linux, macOS, OSS on NonStop, and Unix.

Threading

Node.js operates on a single-thread event loop, using non-blocking I/O calls, allowing it to support tens of thousands of concurrent connections without incurring the cost of thread context switching. The design of sharing a single thread among all the requests that use the observer pattern is intended for building highly concurrent applications, where any function performing I/O must use a callback. To accommodate the single-threaded event loop, Node.js uses the libuv library—which, in turn, uses a fixed-sized thread pool that handles some of the non-blocking asynchronous I/O operations.

A thread pool handles the execution of parallel tasks in Node.js. The main thread function call posts tasks to the shared task queue, which threads in the thread pool pull and execute. Inherently non-blocking system functions such as networking translate to kernel-side non-blocking sockets, while inherently blocking system functions such as file I/O run in a blocking way on their own threads. When a thread in the thread pool completes a task, it informs the main thread of this, which in turn, wakes up and executes the registered call-back.

A downside of this single-threaded approach is that Node.js doesn't allow vertical scaling by increasing the number of CPU cores of the machine it is running on without using an additional module, such as cluster, Strong Loop Process Manager. However, developers can increase the default number of threads in the libuv thread pool. The server operating system (OS) is likely to distribute these threads across multiple cores. Another problem is that long-lasting computations and other CPU-bound tasks freeze the entire event-loop until completion.

V8

V8 is the JavaScript execution engine which was initially built for Google Chrome. It was then open-sourced by Google in 2008. Written in C++, V8 compiles JavaScript source code to native machine code at runtime. As of 2016, it also includes Ignition, a bytecode interpreter.

Package management

npm is the pre-installed package manager for the Node.js server platform. It installs Node.js programs from the npm registry, organizing the installation and management of third-party Node.js programs. Packages in the npm registry can range from simple helper libraries such as Lodash to task runners such as Grunt.

Unified API

Node.js can be combined with a browser, a database that supports JSON data (such as Postgres, MongoDB, or CouchDB) and JSON for a unified JavaScript development stack. With the adaptation of what were essentially server-side development patterns such as MVC, MVP, MVVM, etc., Node.js allows the reuse of the same model and service interface between client side and server side.

Event loop

Node.js registers with the operating system so the OS notifies it of connections and issues a callback. Within the Node.js runtime, each connection is a small heap allocation. Traditionally, relatively heavyweight OS processes or threads handled each connection. Node.js uses an event loop for scalability, instead of processes or threads.[75] In contrast to other event-driven servers, Node.js's event loop does not need to be called explicitly. Instead, callbacks are defined, and the server automatically enters the event loop at the end of the callback definition. Node.js exits the event loop when there are no further callbacks to be performed.

Web Assembly

Node.js supports WebAssembly and as of Node 14 has experimental support of WASI, the WebAssembly System Interface.

Native bindings

Node.js provides a way to make "addons" via a C-based API called N-API which can be used to produce loadable (importable) .node modules from source code written in C/C++. The modules can be directly loaded into memory and executed from within JS environment as simple CommonJS modules. The implementation of the N-API relies on internal C/C++ Node.js and V8 objects requiring users to import (#include) Node.js specific headers into their native source code. As Node.js platform constantly evolves the API compatibility is subject to changes and may get broken sometimes by a new version (as consequence modules have to be built against specific Node.js versions to work correctly). To address the issue third parties have introduced open-sourced C/C++ wrappers on top of the API that partially alleviate the problem. They simplify interfaces but as side effect they may also introduce complexity which maintainers have to deal with. Even though the core functionality of Node.js resides in a JavaScript built-in library, modules written in C++ can be used to enhance capabilities and to improve performance of applications.

In order to produce such modules one needs to have an appropriate C++ compiler and necessary headers (the latter are typically shipped with Node.js itself): gcc, clang or MSVC++.

The N-API is similar to Java Native Interface.

References

[1] https://www.tutorialspoint.com

[2] http://en.wikipedia.org

QUIZ (18)

Quiz : 1 Look at this series: 2, 1, (1/2), (1/4), ... What number should come next?

A. (1/3) B. (1/8) C. (2/8) D. (1/16)

Quiz : 2 Look at this series: 7, 10, 8, 11, 9, 12, ... What number should come next?

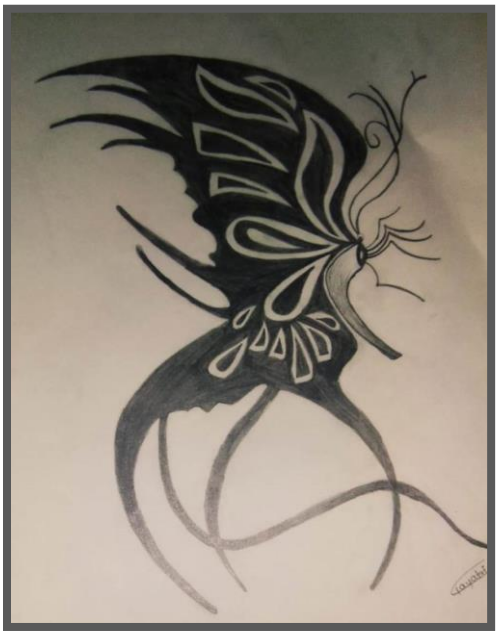
A. 7 B. 10 C. 12 D. 13

Answer of Last Quiz (17)

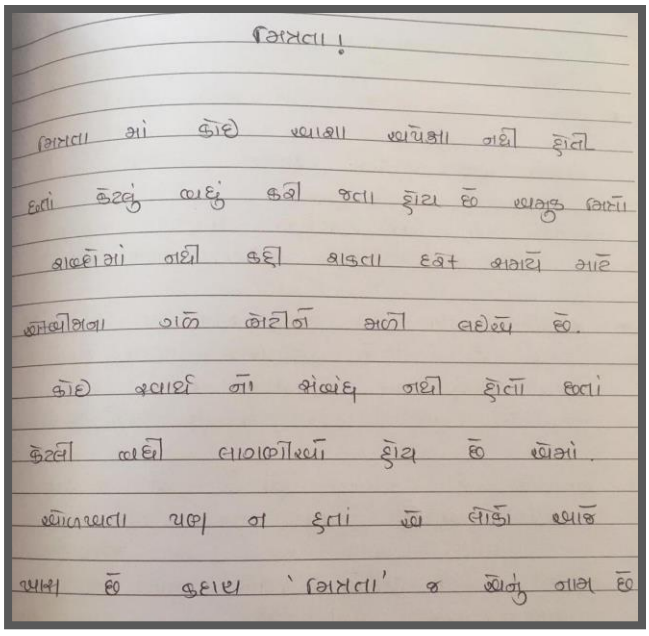
Q. 1 Answer: D Explanation: This is an alternating number of subtraction series. First, 1 is subtracted, then 2 is added.

Q. 2 Answer: A Explanation: This is an alternating number of subtraction series. First, 2 is subtracted, then 4, then 2, and so on.

Student Corner:



Kum. Gaytri Deshmukh
Div.: 4C
Enrollment No.: 196150307138
Department of Computer Engineering



Kum. Urvisha Kachhadiya
Div.: 4C
Enrollment No.: 196150307522
Department of Computer Engineering

1. Re Opening After Covid (offline Study) :

2. Fire Extinguisher Training :