Deno против Node.js: обзор двух режимов выполнения

Описание

Режимы выполнения предоставляют разработчикам мощные инструменты для создания сложных веб-приложений. Выбор подходящего для вашего проекта может помочь вам быстрее выйти на финишную прямую развертывания. Из всех возможных вариантов Deno и Node.js являются главными претендентами на разработку на JavaScript и TypeScript. Появившись в 2009 году, Node.js может похвастаться хорошо развитой экосистемой документации и поддержки сообщества.

В то же время, Deno, выпущенный в 2018 году, находится на стадии относительного становления, но его знакомое происхождение делает его достойным внимания вариантом. В этой статье сравниваются эти два режима выполнения, выделяются их плюсы, минусы и примеры использования, чтобы помочь вам определить, какой из них лучше всего подходит для уникальных потребностей вашего проекта.

Понимание Deno и Node.js

Node.js – это кроссплатформенная среда выполнения JavaScript с открытым исходным кодом, написанная на C++ и построенная на движке V8. С тех пор как создатель Райан Даль представил его сообществу разработчиков в конце 2000-х годов, он стал одной из самых популярных в мире сред выполнения веб-разработки. Хотя Deno не является прямым потомком Node.js, это еще одно изобретение Даля.

Фактически, Даль разработал Deno специально для того, чтобы устранить то, что он

считал недостатками Node.js. Deno имеет безопасную модульную архитектуру, в которой каждый модуль работает в изолированной песочнице. Эта уникальная система загрузки модулей использует импорт на основе URL вместо менеджеров пакетов, таких как npm, кэшируя каждый модуль при импорте для обеспечения более быстрых последующих вызовов.

Сравнение Deno и Node.js

При выборе среды выполнения следует начать со сравнения требований к проекту с тем, что предлагает каждый из вариантов. В этом разделе будут описаны сходства и различия между Deno и Node.js, чтобы помочь вам принять более обоснованное решение.

Безопасность

Даль разработал Deno так, чтобы он был безопасным по умолчанию. Он требует явных флагов разрешения для доступа к системным ресурсам – таким как сеть, файловая система и переменные окружения – что делает его менее уязвимым для атак.

Между тем, Node.js имеет менее ограничительную модель безопасности, которая позволяет получить доступ к системным ресурсам по умолчанию, а не требует явных разрешений. Хотя эта модель обеспечивает большую гибкость и простоту разработки, она имеет больший потенциал для нарушения безопасности, если вы не реализуете все необходимые меры предосторожности (такие как управление доступом и проверка пользовательского ввода).

Поддержка TypeScript

Deno предлагает встроенную поддержку TypeScript, позволяющую писать и выполнять код TypeScript напрямую, без использования дополнительных инструментов или плагинов. Эта функция устраняет необходимость в настройке отдельного процесса сборки, что экономит время и усилия и позволяет сосредоточиться на написании кода.

Хотя вы можете использовать TypeScript в своих приложениях Node.js, это требует дополнительных инструментов и настройки. Установка компилятора и настройка конвейера сборки может быть сложной и отнимать много времени, особенно у

начинающих разработчиков. Кроме того, обновления компилятора могут потребовать изменений в конвейере сборки, что потенциально может вызвать проблемы с обслуживанием.

Управление зависимостями

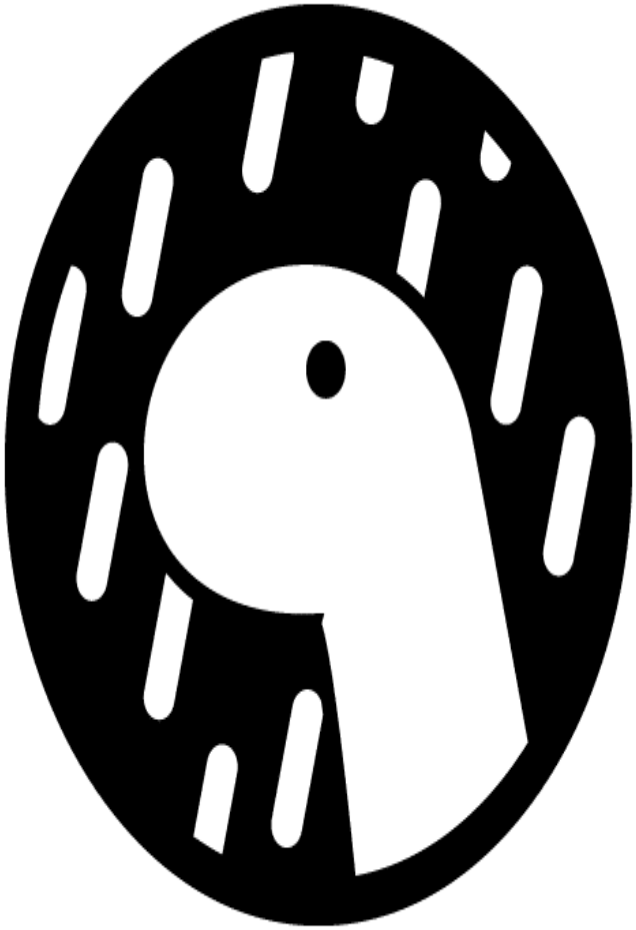
Deno использует импорт на основе URL, что устраняет необходимость в отдельном менеджере пакетов, таком как npm. С Deno вы можете импортировать модули непосредственно из URL-адресов, не скачивая и не устанавливая их отдельно.

В отличие от этого, Node.js полагается на npm и файл package.json для управления зависимостями. Этот файл содержит полный список зависимостей проекта, что облегчает другим разработчикам установку и запуск проекта на своих системах. Однако этот процесс может привести к конфликтам зависимостей и проблемам с версиями, особенно по мере развития пакетов.

Стандартная библиотека и API

Deno включает встроенную стандартную библиотеку, которая предоставляет необходимые инструменты и функции для решения общих задач. В результате вы можете настроить базовую функциональность вашего приложения без установки и настройки библиотек сторонних разработчиков, что обеспечивает более согласованный API. Кроме того, он включает в себя линтер, форматер и библиотеку тестирования, которые помогают поддерживать качество и согласованность кода.

Однако, будучи более новым инструментом, экосистема Deno все еще несколько ограничена. Хотя он предоставляет все базовые возможности, вам, возможно, придется создавать более продвинутые функции с нуля или полагаться на менее проверенные сторонние библиотеки. Встроенная стандартная библиотека Node.js более ограничена, а ее API постоянно развиваются. Эти ограничения вынуждают использовать сторонние инструменты для выполнения некоторых базовых задач, что может привести к функциональным и дизайнерским несоответствиям.



Плюсы и минусы Deno

Если вы подумываете об использовании Deno в своем следующем проекте, подумайте, что отличает его от других. Вот некоторые основные плюсы и минусы движка выполнения Deno.

Плюсы

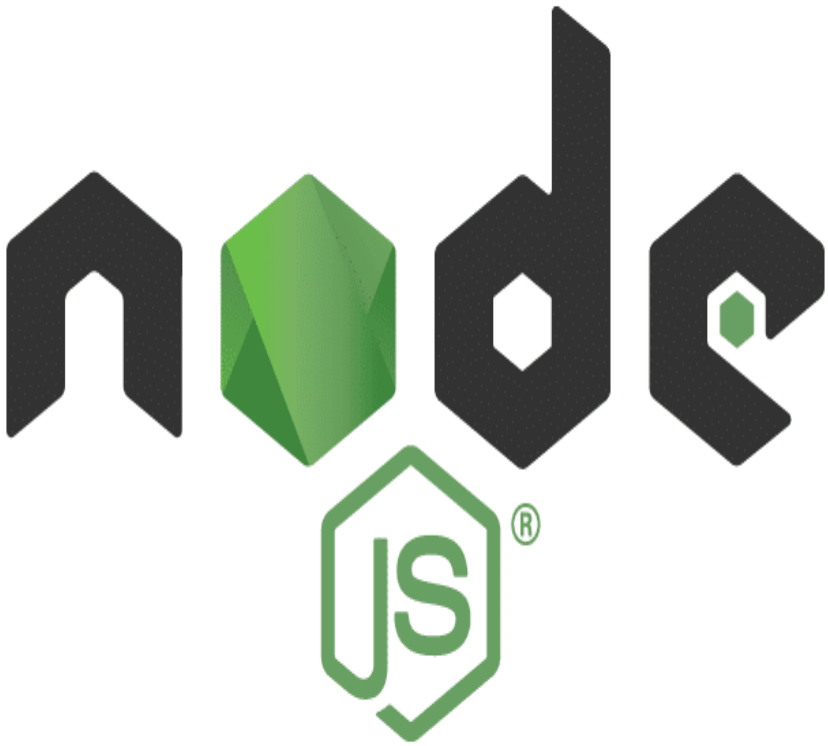
- Расширенные функции безопасности
- Встроенная поддержка TypeScript
- Упрощенное управление зависимостями
- Встроенная стандартная библиотека

В целом, Deno облегчает написание безопасного и сопровождаемого кода, не полагаясь на сторонние библиотеки или инструменты.

Минусы

- Менее зрелая экосистема и сообщество
- Ограниченная поддержка сторонних библиотек
- Может потребоваться повторное изучение некоторых концепций и практик

Язык и базовая архитектура Deno могут потребовать дополнительного времени для понимания, что делает кривую обучения более сложной.



Плюсы и минусы Node.js

Как и любая технология, Node.js также имеет свою долю достоинств и недостатков. Давайте рассмотрим их подробнее.

Плюсы

- Зрелая и устоявшаяся экосистема
- Большое сообщество и обширная поддержка библиотек
- Доказанный послужной список в различных отраслях и проектах

Node.js существует уже более десяти лет, и многие компании используют его в

качестве основной платформы для разработки. Благодаря большому сообществу разработчиков он имеет надежную экосистему библиотек, инструментов и фреймворков. Сайт npm может похвастаться более чем миллионом сторонних библиотек для Node.js, что позволяет легко найти нужные инструменты для конкретных проектов.

Минусы

- Потенциальные проблемы с безопасностью
- Отсутствие встроенной поддержки TypeScript
- Несогласованность API и отсутствие встроенной стандартной библиотеки

Хотя Node.js имеет свои преимущества, его менее строгая модель безопасности и зависимость от библиотек сторонних разработчиков могут быть рискованными.

Случаи использования Deno и Node.js

Не существует универсальной среды выполнения. Правильный выбор для вашего проекта зависит от самого проекта. Давайте рассмотрим некоторые наиболее распространенные случаи использования Node.js и Deno, чтобы определить, что лучше всего соответствует требованиям вашего проекта.

Случаи использования Deno

Deno имеет улучшенную модель безопасности, которая не требует установки дополнительных пакетов. Эти особенности делают его идеальным для малых и средних проектов, для которых безопасность является приоритетом. Встроенная поддержка TypeScript также делает его отличным вариантом для разработчиков, предпочитающих работать в среде, ориентированной на TypeScript.

Наконец, Deno хорошо подходит для проектов, которым требуется небольшое количество зависимостей и выгодно использовать встроенную стандартную библиотеку. Импорт на основе URL в Deno устраняет необходимость в менеджере пакетов, что делает его исключительным вариантом для разработчиков, которые хотят более упрощенного процесса разработки.

Примеры использования Node.js

Как популярная и давно существующая среда исполнения, Node.js имеет

проверенную репутацию и обширную поддержку библиотек. Его устоявшееся сообщество гарантирует вам доступ к ресурсам, необходимым для создания стабильных приложений в течение длительного времени.

Этот основной инструмент веб-разработки лучше всего использовать для следующих целей:

- Крупномасштабные проекты и проекты уровня предприятия
- приложения, использующие широкий спектр инструментов сторонних разработчиков
- Проекты, в которых зрелость и стабильность экосистемы являются критическими факторами.

Заключение

При выборе среды выполнения для JavaScript или TypeScript необходимо учитывать цели, требования, приоритеты и ограничения вашего проекта. Deno и Node.js – мощные и популярные варианты. Однако их сильные и слабые стороны делают их подходящими для разных случаев использования. Deno устраняет многие недостатки Node.js, предлагая встроенную поддержку TypeScript, более строгую безопасность и встроенную стандартную библиотеку.

Однако его библиотека сторонних разработчиков все еще растет. Напротив, Node.js представляет собой более привычный вариант, которому доверяют многие разработчики и организации. Даже если вы пока придерживаетесь Node.js, вы, вероятно, увидите, как растет популярность Deno по мере расширения его экосистемы.

Дата Создания
05.07.2023