mitrais | MEMBER OF CAC HOLDINGS GROUP

Bun: All-in-One JavaScript Runtime and Toolkit

Mitrais is a world-class technology company based in Indonesia and part of the global CAC Holdings Group. Founded in 1991, we have developed and implemented software for over 700 clients, and we are committed to building long-term and high-trust relationships.

www.mitrais.com

Table of Contents

Abstract

Bun is a powerful tool for developers to create JavaScript and TypeScript applications. Developed using the Zig programming language, Bun offers a wide range of functionalities such as running code, managing packages, testing, and bundling. This white paper provides an exploration of Bun's capabilities and highlights its value in app development.

One of the key features of Bun is its ability to run code efficiently. It utilizes the Zig programming language and JavaScriptCore, a JavaScript engine, to optimize the execution of JavaScript and TypeScript applications, resulting in faster and more efficient performance.

Additionally, Bun excels in package management, allowing developers to easily manage and organize their project dependencies. With Bun, developers can effortlessly install, update, and remove packages, streamlining the development process.

Another notable feature of Bun is its testing capabilities. It provides developers with a robust testing framework that enables them to thoroughly test their applications for bugs and errors. By automating the testing process, Bun helps ensure the reliability and stability of the developed apps.

Moreover, Bun offers seamless bundling functionality, allowing developers to bundle their code into a single file for deployment. This simplifies the distribution process and enhances the overall efficiency of app deployment.

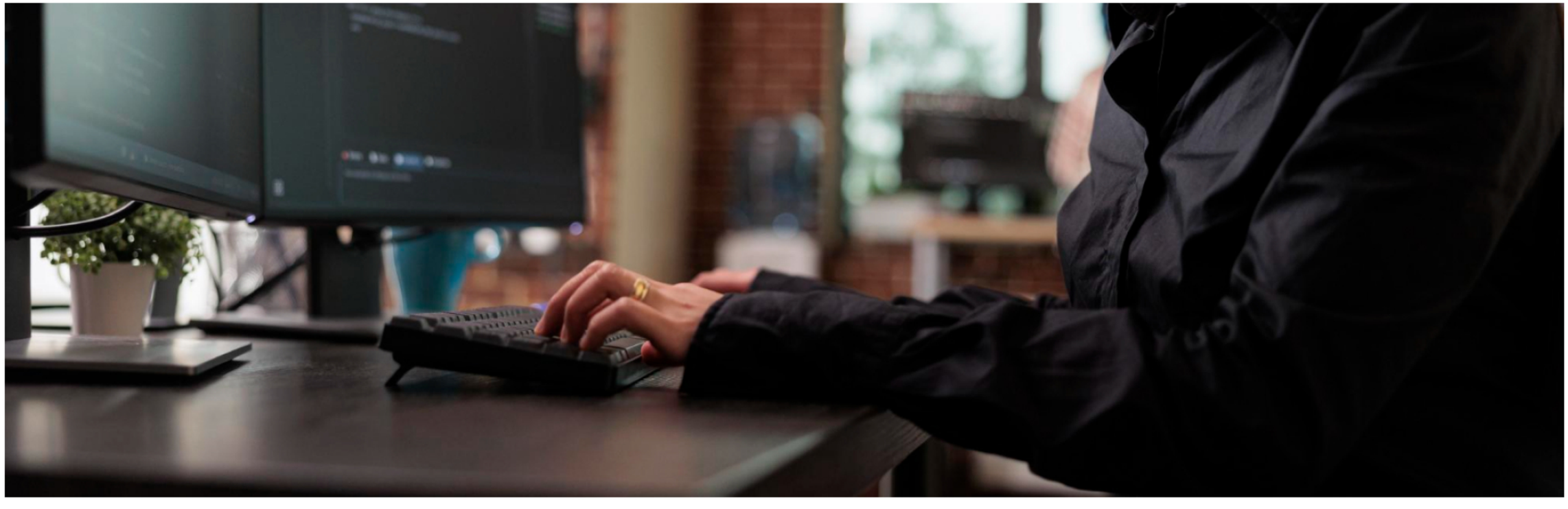
1. Introduction

Bun is an all-in-one JavaScript runtime and toolkit designed for speed, complete with a bundler, test runner, and Node.js-compatible package manager. It was created by Jarred Sumner as a drop-in replacement for Node.js. Some key features of Bun include:

- **Bun runtime:** A fast JavaScript runtime written in Zig and powered by JavaScriptCore, which reduces startup times and memory usage.
- **Web-standard APIs:** Bun implements standard Web APIs like fetch, WebSocket, and ReadableStream, using Safari's implementation for some APIs like Headers and URL.
- **Node.js compatibility:** Bun supports Node-style module resolution and aims for full compatibility with built-in Node.js globals (process, Buffer) and modules (path, fs, http, etc.).
- **Built-in standard library:** Bun offers functionalities for diverse protocols and modules, including environment variables, HTTP, WebSocket, file system, and more.
- **Hot reloading:** Bun supports hot reloading, allowing developers to see changes in real-time without restarting the application.
- **Native TypeScript support:** Bun supports TypeScript out of the box, making it easier for developers to work with TypeScript projects.

Bun is an open-source project with a growing community of users actively contributing to its development and improvement. It is written in Zig and uses JavaScriptCore as the JavaScript engine, which is developed by Apple for Safari.

On September 8, 2023, Bun achieved a significant milestone by launching version 1.0, signifying the first stable release for Bun on macOS and Linux. However, the Windows version is still in the experimental phase and currently supports only the JavaScript runtime. In the Windows build, the package manager, test runner, and bundler are disabled, but there are plans to enable them once they become more stable and their performance is optimized.



2. Core Functionalities

Bun offers a range of core functionalities that make it an essential tool for JavaScript andTypeScript app development. These include:

2.1 JavaScript Runtime

JavaScript runtime is the environment where JavaScript code is executed, and it provides the necessary infrastructure for storing functions, variables, and managing memory. There are two main types of JavaScript runtimes: browser runtime environments and Node.js runtime environments, each with its own set of global objects and APIs.

- **Browser Runtime Environment:** The most common place where JavaScript code is executed is in a browser. JavaScript code in a browser has access to the Window object and the Document Object Model(DOM), which are critical for web developers because they provide the necessary APIs that allow them to create dynamic web applications.
- **Node.js Runtime Environment:** Node.js runtime environment is a cross-platform environment for executing JavaScript code, commonly used for server-side or desktop applications. It is built on Chrome's V8 JavaScript engine, which compiles JavaScript directly to native machine code before executing it. This approach allows Node.js to achieve low latency and high throughput by utilizing a single-threaded, non-blocking event loop and a low-level I/O API. Bun as a JavaScript runtime is written in Zig and powered by JavaScriptCore as the JavaScript engine, which is developed by Apple for Safari browser. Instead of relying on the V8 engine, as Node.js does, Bun utilizes JavaScriptCore, widely recognized for its superior performance.

2.2 Web-standard APIs

Web-standard APIs are a set of standardized web functionalities that are commonly available in web browsers. These APIs provide a consistent and familiar development experience for web developers. Web-standard APIs are used to transfer data and interact with the functionality between web-based systems, and they deliver requests from web applications and responses from servers using Hypertext Transfer Protocol (HTTP).

The Web-standardAPIs supported by Bun include standard web functionalities commonly available in browsers. These APIs are natively implemented in Bun, providing faster and more reliable support compared to traditional server-side environments like Node.js. Some of the Web-standard APIs supported by Bun include:

- **Fetch:** This API is used for making network requests. It allows you to make HTTP requests, including GET, POST, PUT, and DELETE, and is commonly used to fetch data from a server.
- **WebSocket:** This API provides full-duplex communication channels over a single TCP connection. It is commonly used in web development for creating real-time, bi-directional communication between clients and servers.
- **ReadableStream:** This API represents a readable stream of byte data. It is commonly used for handling streaming data in web applications.

Bun's native implementation of these Web APIs ensures they are faster and more reliable compared to traditional server-side environments like Node.js. This built-in support for Web-standard APIs simplifies development by eliminating the need for additional packages and ensuring consistent and reliable functionality across different environments.

2.3 Package Manager / Node.js compatibility

Bun supports Node.js-style module resolution and aims for full compatibility with built-in Node.js globals and modules. This means that Bun is designed to work seamlessly with existing Node.js code and packages, allowing developers to leverage their knowledge and resources from Node.js when using Bun. Some key aspects of this compatibility include:

- Support for built-in Node.js modules such as fs, path, and net.
- Recognition of global variables like __dirname and process.
- Adherence to the Node.js module resolution algorithm, including the familiar node_modules structure.

Bun's goal of achieving complete Node.js API compatibility means that most npm packages intended for Node.js environments will work with Bun out of the box. This compatibility is an ongoing effort, and the Bun team regularly updates the compatibility status to reflect the latest version of Bun.

While some discussions express reservations about Bun's compatibility with Node.js, the focus on achieving this compatibility is aimed at easing the transition for Node.js developers and leveraging the strengths of both environments

2.4 Built-in standard library

Bun's built-in standard library refers to the functionalities and modules that are included in the Bun runtime environment. These functionalities and modules are designed to provide developers with a comprehensive set of tools for building JavaScript and TypeScript applications. Some of the functionalities and modules included in the Bun standard library are:

- **Environment variables:** Bun provides access to environment variables, which are used to store configuration information for an application.
- **HTTP:** Bun includes a built-in HTTP server that can be used to serve web pages and handle HTTP requests.
- **WebSocket:** Bun includes a built-in WebSocket server that can be used to create real-time, bi-directional communication between clients and servers.
- **Filesystem:** Bun includes a built-in file system module that can be used to read and write files.
- **SQLite:** Bun includes a built-in SQLite module that can be used to interact with SQLite databases.

Bun distinguishes itself from Node.js by not relying on npm or external dependencies for its core operations. Instead, Bun features a built-in standard library that provides functionalities for various protocols and modules, including environment variables, HTTP, WebSocket, file system, and more.

2.5 Hot Reloading

Bun supports hot reloading to automatically detect and apply code changes in real-time without the need to manually restart the application. When hot reloading is enabled, Bun monitors the source code for any modifications, and upon detecting a change, it automatically updates the running application with the new code, allowing developers to see the effects of their changes instantaneously.

This feature significantly enhances the development experience by eliminating the need for manual restarts, thereby saving time, and increasing productivity. It is important to note that Bun's hot reloading is a server-side feature and should not be confused with the hot reloading experience provided by many front-end frameworks, where changes to the frontend code are reflected in the browser without requiring a full-page refresh.

2.6 Native TypeScript support

Bun can directly compile and run TypeScript code without the need for an additional transpiler. This built-in support for TypeScript makes it easier for developers to work with TypeScript projects, as they can write and execute TypeScript code without the extra step of transpiling it to JavaScript.

Bun treats TypeScript as a first-class citizen, allowing developers to directly execute .ts and .tsx files just like vanilla JavaScript, with no extra configuration. It's important to note that, similar to other build tools, Bun does not type check the files, so developers should use the official TypeScript CLI (tsc) if they are looking to catch static type errors.

This native TypeScript support is a key feature of Bun, designed to streamline the development process and enhance the efficiency of working with TypeScript projects.

3. Advantages

Bun offers several advantages over traditional JavaScript runtimes and toolkits, including:

3.1 Speed

The Zig-based and JavaScriptCore implementation in Bun significantly impacts its performance, contributing to its speed, efficiency, and memory usage. The use of Zig, a high-performance, low-level programming language, allows Bun to achieve faster startup times and reduced memory usage. Additionally, Bun's gzip implementation, which is optimized and written in Zig, further enhances its performance.

The integration of JavaScriptCore, the engine powering Apple’s Safari, under the hood of Bun, also plays a crucial role in reducing startup times and memory usage. This combination of Zig and JavaScriptCore enables Bun to outperform other platforms in various benchmarks, making it a promising choice for modern JavaScript development. The emphasis on performance, elegant APIs, and developer experience makes Bun a compelling option for those seeking a more efficient and faster JavaScript runtime.

According to the benchmarks on the Bun website, Bun's serve() function outperforms Node.js and Deno by 377% and 102% respectively.

Additionally, various articles and blog posts have conducted performance tests comparing Bunto Node.js, with results indicating that Bun is faster in certain scenarios. For example, a benchmark conducted by TSH.io showed that Bun serves 4 times more requests per second and that packages are installed 30 times faster compared to Node.js.

Another article on ByteOfDev.com also presented benchmarks where Bun outperformed Node.js and Deno in various tests, such as I/O operations and SQLite queries.

These benchmarks and tests suggest that Bun demonstrates superior performance in certain scenarios compared to Node.js and Deno.

3.2 Compatibility

While Bun aims for full compatibility with built-in Node.js globals and modules, the extent of this compatibility is still an ongoing effort and remains incomplete.

According to a review on Delicious Brains, while Bun uses the same module resolution algorithm as Node.js, many built-in and global modules are only partially implemented, and some are not implemented at all. Developers should check the compatibility page on Bun's website for the latest updates before deciding if Bun will meet their needs as Bun's team regularly updates the compatibility page as support is improved.

3.3 All-in-one toolkit

By providing a comprehensive set of tools within a single environment, Bun enhances the user experience, reduces the need for multiple external tools, and offers a more seamless and integrated development workflow.

This all-in-one approach is designed to speed up the development cycle, eliminate the need for separate transpilers, bundlers, and test runners, and provide a package manager with unmatched speed.

4. Use-Cases

Bun is suitable for a wide range of use cases, including:

- **Web development:** Bun can be used to develop web applications, including server-side and client-side JavaScript/TypeScript projects.
- **Desktop development:** Bun can be used to develop desktop applications, including cross-platform JavaScript/TypeScript projects.
- **Command-line tools:** Bun can be used to develop command-line tools, including JavaScript/TypeScript projects that can be run from the command line.

Conclusion

Bun is fast, all-in-one JavaScript runtime that aims to provide a complete toolkit for JavaScript and TypeScript apps.

It includes a built-in bundler, test runner, and package manager, and implements standard Web APIs like fetch, WebSocket, and ReadableStream.

Bun is powered by the JavaScriptCore engine, and is written in Zig, which contributes to its speed and efficiency. The performance of Bun is a key focus, with significant efforts spent on profiling, benchmarking, and optimizing to achieve its speed.

Whilst Bun is still considered experimental, its performance, speed, and potential for enhancing developer productivity are highlighted as key advantages.

The built-in tools, such as the bundler, test runner, and package manager, are considered advantageous due to their speed, efficiency, and seamless integration with the Bun runtime. If considering Bunas a replacement, we need to carefully checks for any errors or issues and mitigate any risks before fully integrating it into our development workflow.

Despite being a relatively new project, Bun is gaining ground as a competitor to widely used runtime environments like Node.js and Deno, and it is seen as a promising choice for developing fast, performant, error-free applications.

References

1. https://bun.sh/
2. https://kinsta.com/blog/bun-sh
3. https://dev.to/snickdx/what-is-the-javascript-runtime-4n09
4. https://www.codecademy.com/article/introduction-to-javascript-runtime-environments
5. https://www.freecodecamp.org/news/what-is-node-js-explained/
6. https://www.infoworld.com/article/3210589/what-is-nodejs-javascript-runtime-explained.html
7. https://en.wikipedia.org/wiki/WebKit
8. https://en.wikipedia.org/wiki/Bun_(software)
9. https://en.wikipedia.org/wiki/Web_API
10. https://refine.dev/blog/bun-js-vs-node/
11. https://www.builder.io/blog/bun-vs-node-js
12. https://github.com/oven-sh/bun/discussions/5518
13. https://www.linkedin.com/pulse/bun-10-new-era-javascript-development-taqui-imam
14. https://www.linkedin.com/pulse/arrival-bun-10-game-changer-javascript-development-dennis-hundertmark-x6zee
15. https://news.ycombinator.com/item?id=37244012
16. https://byteofdev.com/posts/what-is-bun/
17. https://tsh.io/blog/bun-benchmark/
18. https://blog.devgenius.io/bun-pros-and-cons-and-main-usage-scenarios-2e8402a91e5f?gi=ad7961e5ff78
19. https://www.newdev.io/blog/bun-vs-nodejs
20. https://deliciousbrains.com/a-short-guide-to-using-bun/
21. https://blog.logrocket.com/bun-adoption-guide/
22. https://www.infoq.com/news/2023/07/bun-native-bundler-macros/
23. https://www.thegreenreport.blog/articles/buns-test-runner-the-future-of-javascript-testing/buns-test-runner-the-future-of-javascript-testing.html
24. https://www.wearecapicua.com/blog/bun-javascript

About Mitrais

Mitrais is a world-class technology company based in Indonesia and a part of the global CAC Holdings Group. We have been recognized as Indonesia's leading provider of offshore development services by Forrester Research, and our goal is to help your business meet and exceed your expectations. Combining Western innovation with Eastern productivity, Mitrais maintains its preeminent position in the Asia Pacific region. As a member of the Microsoft Partner Network with a Gold Application Development competency, we demonstrate the highest level of competence and expertise with Microsoft technologies. Our close working relationship with Microsoft enables us to deliver exceptional software development services. Through collaboration with trusted partners and our team of talented software engineers, we are committed to providing outstanding solutions to our valued clients.

MITRAIS | MEMBER OF CAC HOLDINGS GROUP

Terima Kasih　　**Thank** You　　**ありがとう**ございました