JS-REACTOR DEVELOPMENT UPDATE

JS-Reactor 0.1.2 Alpha: compatibility and VM bytecode progress

JS-Reactor is still in active alpha development. Version 0.1.2 focuses on compatibility, heavier protected output, and a cleaner path toward a public launch.

Version	Status	Period covered
0.1.2 Alpha	Not launched as a final product	April 11 to May 11, 2026

Compatibility coverage

This update adds compatibility checks for the JavaScript shapes that usually break when protection becomes more aggressive.

- ES5 syntax, prototypes, and automatic semicolon insertion cases.
- ES2015 classes, closures, Proxy, Symbol, Reflect, and strict mode.
- ES2020 optional chaining, nullish coalescing, BigInt, async/await, and unicode identifiers.
- ES2023 private fields, static blocks, reserved properties, ASI, and global object behavior.
- CommonJS, browser script output, IIFE output, and ESM with static import, dynamic import, and top-level await.
- Toolchain checks for Babel parse, Terser minify, esbuild, Rollup, Webpack, and Vite.

My current compatibility report shows passing results across these checks. That matters because protected code still has to run like the original code.

VM bytecode work

The VM layer now protects more than simple expressions. Version 0.1.2 expands the protected path for supported function shapes while keeping compatibility checks around it.

- Typed IR and a VM-owned constant pool were added.
- Safe-subset function compilation now lowers supported functions into bytecode.
- Function expressions and arrow functions are covered by the VM path when they match supported semantics.
- Bytecode is emitted as packed binary-like containers instead of readable source structures.
- Control-flow blocks can be split, reordered, and decoded lazily at runtime.
- Dispatcher variants, handler layout changes, register state layout changes, and v5 tamper-coupled bytecode were added.

The result is still JavaScript output, but the static shape is much heavier than the original source.

Current compatibility snapshot

Area	Committed result
Node.js latest and browser-script/CommonJS semantics	PASS
ES5, ES2015, ES2020, and ES2023 compatibility fixtures	PASS
ESM static import, dynamic import, and top-level await	PASS
Safe-subset full-function VM bytecode	PASS
Bytecode control-flow VM blocks and lazy block decode	PASS
Safe function expression and arrow VM bytecode	PASS
VM state layout, handler hardening, tamper-coupled keys, and mutated record layout	PASS
Register VM safe subset, localized variable pressure, and recursion fallback compatibility	PASS
CSP static scan for direct eval or Function constructor blockers	PASS

Protected-output reference

A public reference output is available here: https://end2end.space/pastes/aVShJXJTfaKZ/raw

The source program is intentionally small: a Node.js script with one print function. It was protected heavily with JS-Reactor so the current output shape is easy to inspect.

Use that paste as a snapshot of the current 0.1.2 Alpha protection output. It is meant to stay compatible with normal JavaScript environments while changing the static form of the program significantly.

> **Public status:** JS-Reactor 0.1.2 Alpha is not launched as a final product yet. I am keeping it visible as active development while the compatibility and protection work continues.

I will keep updating JS-Reactor while it stays in alpha. The focus for 0.1.2 is simple: stronger protected output, wider JavaScript compatibility, and proof that protected code still behaves like the original.