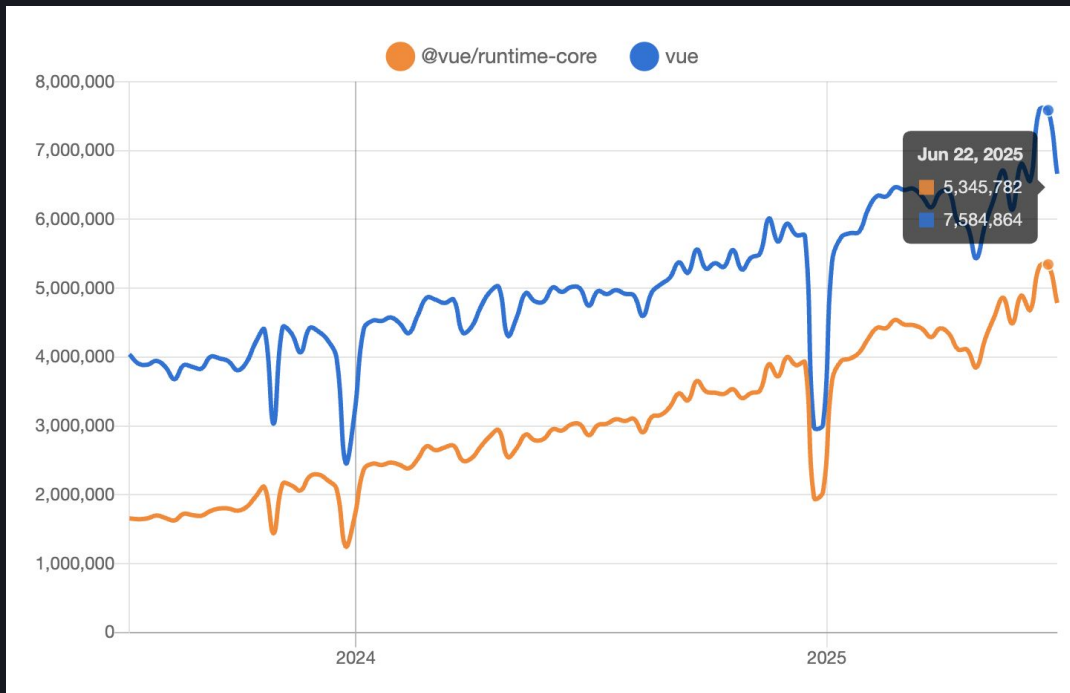




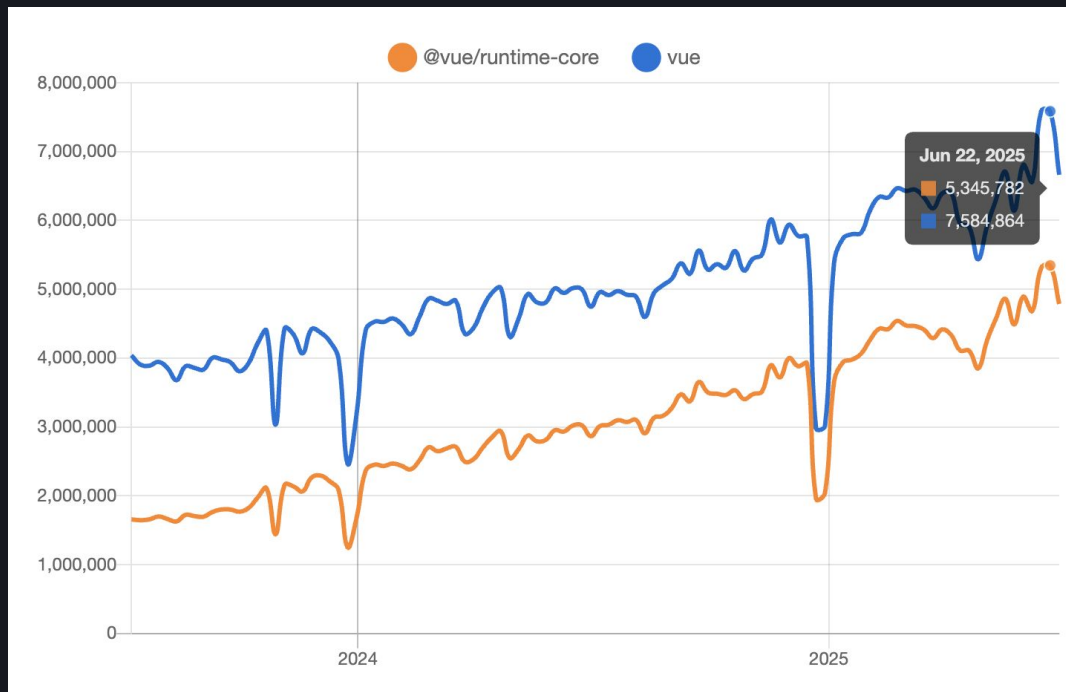
Vue & Vite 生态最新进展

VueConf China 2025
7.12.2025 深圳

~7.5m weekly npm downloads
(+50% YoY growth!)



Vue 3 percentage: 61.5% → 71.7%
Vue 3 YoY growth: +76.7%



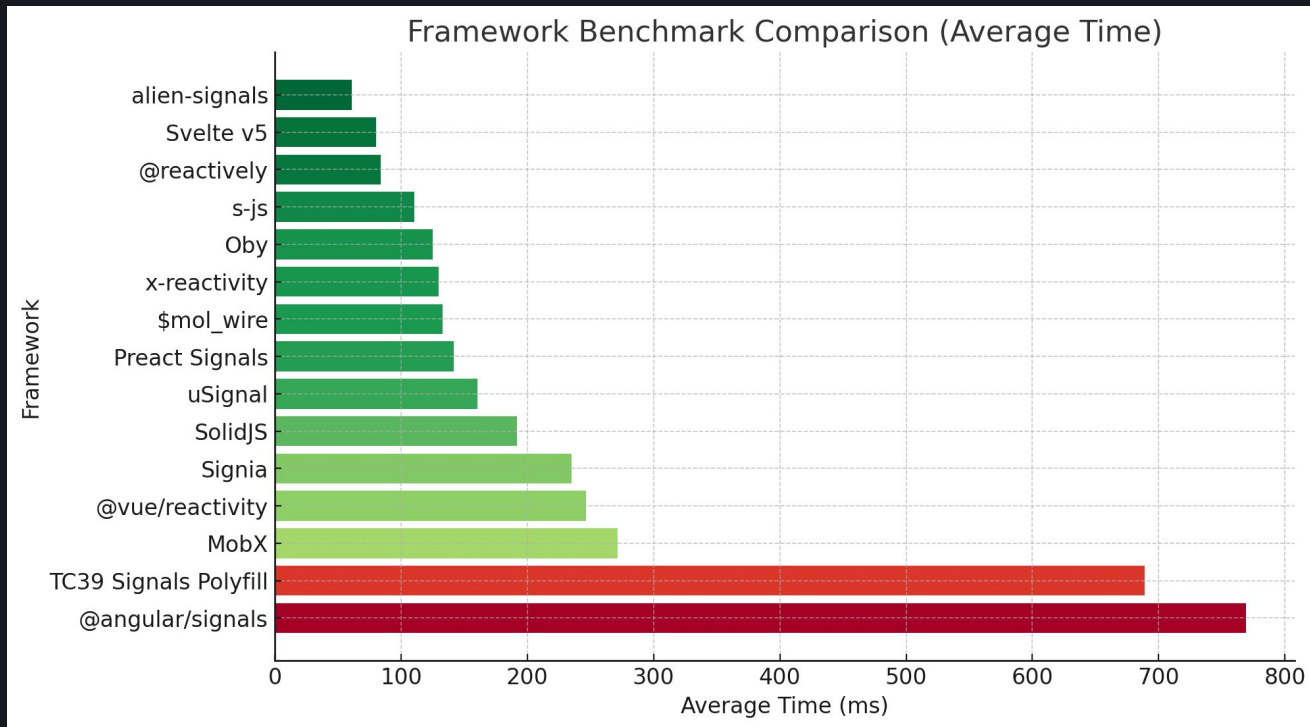
Vue Language Tools 3.0

- 默认 Hybrid Mode
- 大幅改进稳定性
- 感谢 Johnson 和御守的努力！

3.6 (alpha)

- Vapor Mode!
- Reactivity system refactor based on alien-signals

Alien Signals 性能



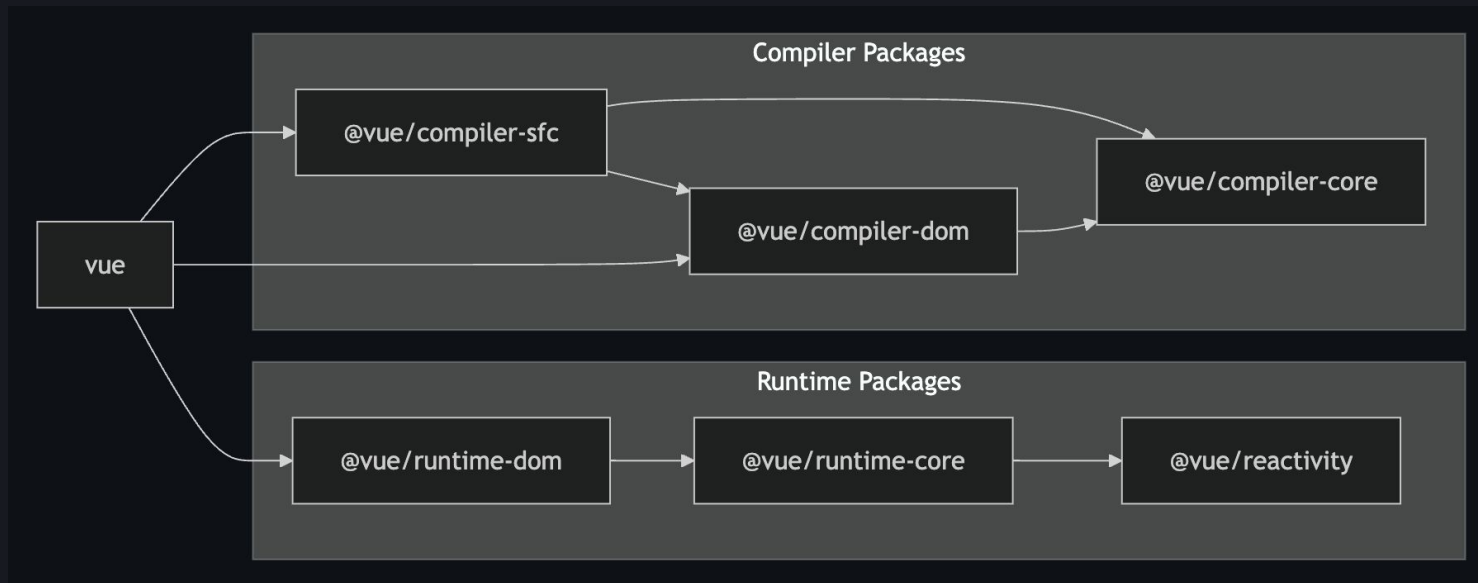
Vapor Mode 到底是什么？

一个为了极致性能而存在的
全新的编译和渲染模式

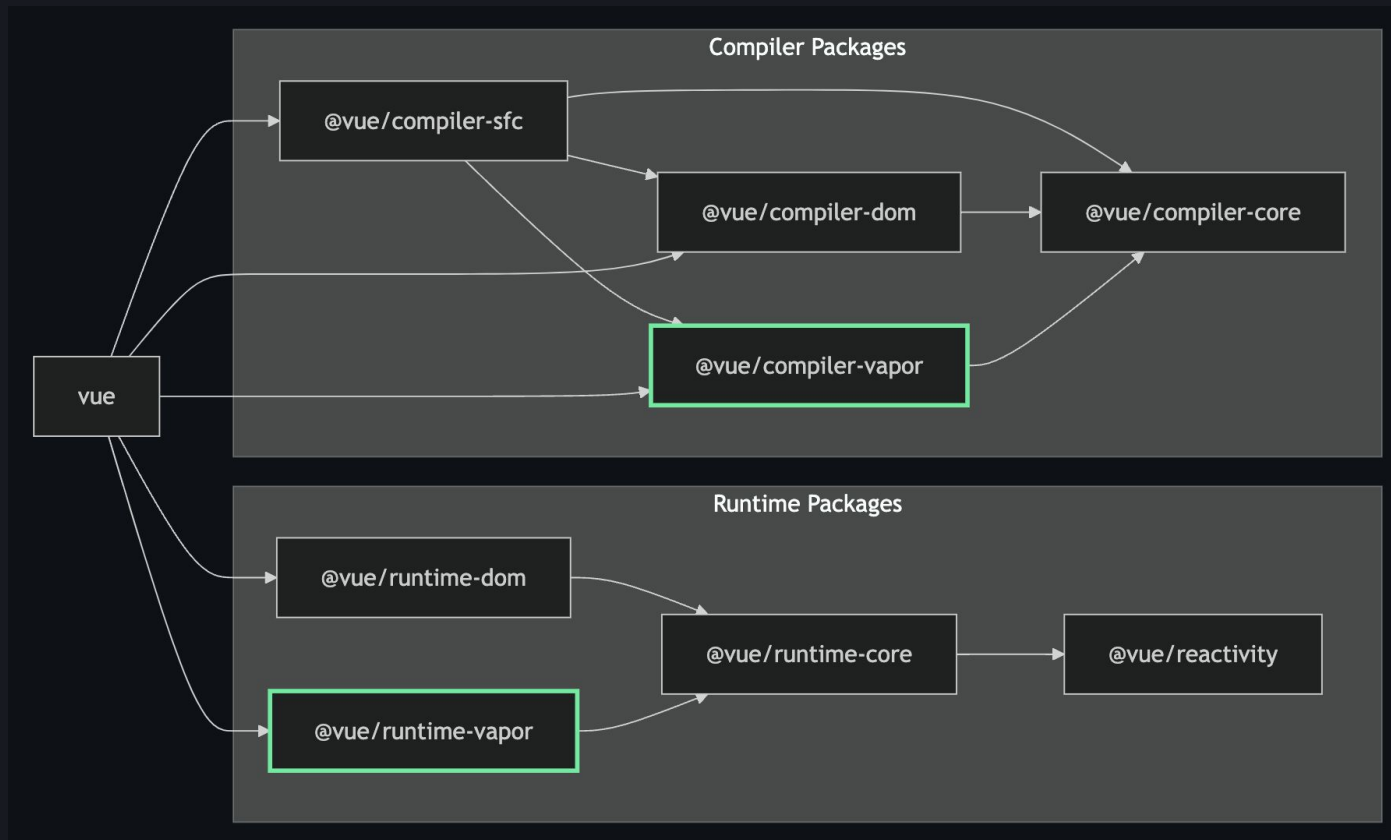
Vapor Mode 重点

- 一样的 API(子集)
- 一样的行为
- 不同的编译输出
- 大幅提升性能
- 细粒度启用
- 只支持 SFC

Vapor 之前的 Vue 内部依赖结构



Vapor 之后的 Vue 内部依赖结构



Same API

Different output

SFC source

```
1 <script setup>
2 import { ref } from 'vue'
3
4 const msg = ref('Hello World!')
5 </script>
6
7 <template>
8   <h1>{{ msg }}</h1>
9   <input v-model="msg" />
10 </template>
11
```




VDOM compiler output

```
return (_ctx, _cache) => {
  return (_openBlock(), _createElementBlock(_Fragment, null, [
    _createElementVNode("h1", null, _toDisplayString(msg.value), 1 /* TEXT */),
    _withDirectives(_createElementVNode("input", {
      "onUpdate:modelValue": _cache[0] || (_cache[0] = $event => ((msg).value = $event))
    }, null, 512 /* NEED_PATCH */), [
      [_vModelText, msg.value]
    ])
  ]), 64 /* STABLE_FRAGMENT */))
}
```

Vapor compiler output

```
const n0 = t0()
const n1 = t1()
const x0 = _child(n0)
_applyTextModel(n1, () => (msg.value), _value => (msg.value = _value))
_renderEffect(() => _setText(x0, _toDisplayString(msg.value)))
return [n0, n1]
```



Vapor Codegen

```
<script setup vapor>  
import { ref } from 'vue'
```

```
const msg = ref('Hello World!')  
</script>
```

```
<template>  
  <h1>{{ msg }}</h1>  
</template>
```

1. 分析模版中连续的静态结构部分并生成创建 DOM 片段的工厂函数

```
const t0 = _template("<h1> </h1>")
```

```
// in component setup  
const msg = ref('Hello World!')  
const n0 = t0()  
const x0 = _child(n0)  
  
_renderEffect(() => {  
  _setText(x0, _toDisplayString(msg.value))  
})
```

Vapor Codegen

```
<script setup vapor>  
import { ref } from 'vue'
```

```
const msg = ref('Hello World!')  
</script>
```

```
<template>  
  <h1>{{ msg }}</h1>  
</template>
```

```
const t0 = _template("<h1> </h1>")
```

2. Script setup code

```
// in component setup
```

```
const msg = ref('Hello World!')
```

```
const n0 = t0()
```

```
const x0 = _child(n0)
```

```
_renderEffect(() => {  
  _setText(x0, _toDisplayString(msg.value))  
})
```

Vapor Codegen

```
<script setup vapor>  
import { ref } from 'vue'
```

```
const msg = ref('Hello World!')  
</script>
```

```
<template>  
  <h1>{{ msg }}</h1>  
</template>
```

```
const t0 = _template("<h1> </h1>")
```

```
// in component setup
```

```
const msg = ref('Hello World!')
```

```
const n0 = t0()  
const x0 = _child(n0)
```

3. 创建 DOM 片段, 然后精确
定位其内部的动态节点

```
_renderEffect(() => {  
  _setText(x0, _toDisplayString(msg.value))  
})
```

Vapor Codegen

```
<script setup vapor>  
import { ref } from 'vue'
```

```
const msg = ref('Hello World!')  
</script>
```

```
<template>  
  <h1>{{ msg }}</h1>  
</template>
```

```
const t0 = _template("<h1> </h1>")
```

```
// in component setup
```

```
const msg = ref('Hello World!')
```

```
const n0 = t0()
```

```
const x0 = _child(n0)
```

```
_renderEffect(() => {  
  _setText(x0, _toDisplayString(msg.value))  
})
```

4. 创建作用于动态节点的响应式 effect

Vapor Mode in 3.6 alpha

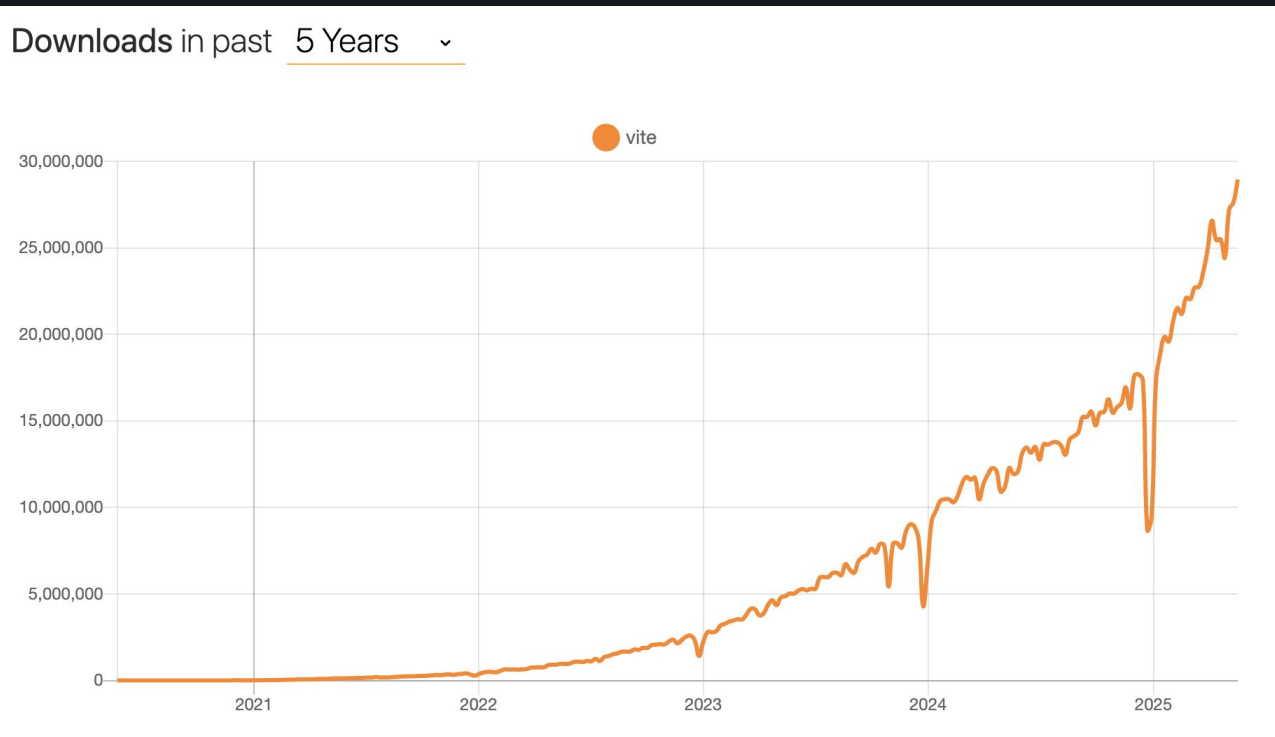
- Experimental status
- 通过 ``<script setup vapor>`` 在组件级别启用
- 通过 ``createVaporApp`` 创建纯 Vapor 应用
- 以下功能 PR 会在 beta 前 merge
 - Async Component
 - Transition
 - KeepAlive
 - SSR hydration



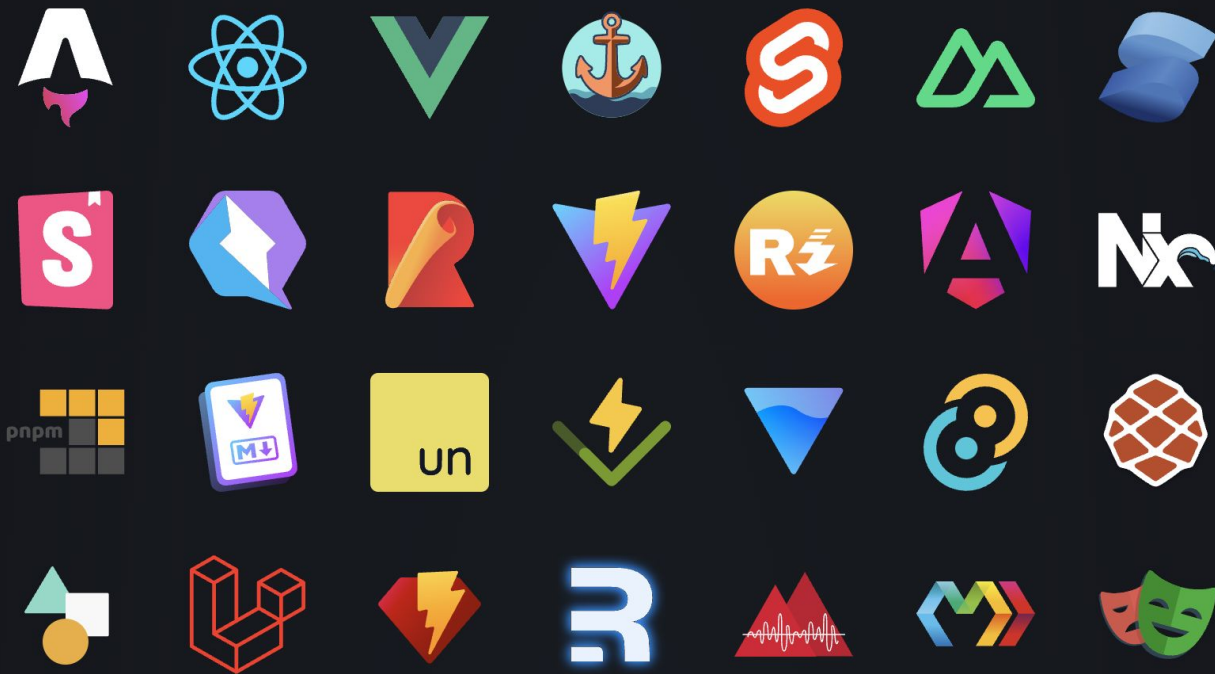
现在来聊聊 Vite 生态



~32 million weekly npm downloads
+138% YoY 🚀



A Vast Ecosystem



void(0)



Vite



Vitest



Rollup



OXC

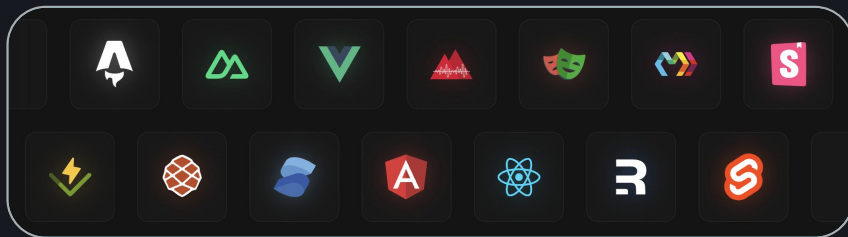
JS 生态的碎片化

- **Parser:** babel / acorn / esbuild / espree / flow / TS / swc / uglify-js
- **Transformer:** babel / esbuild / swc / sucrose / TS
- **Test runner:** mocha / jasmine / tape / ava / jest
- **Linters:** jshint / eslint / biome
- **Formatter:** prettier / eslint
- **Bundlers:** webpack / rollup / esbuild / parcel
- **Monorepo tools:** nx / turborepo / rush / lerna / lage / wireit

JS 生态的庞大带来的丰富的选择，
但也带来了不可忽视的**复杂税**。

随着最佳实践的沉淀，我们认为构建
一体化工具链的时机已经到来。

Frameworks & Tools (ecosystem partners)



void(0)
Open Source

Vite+

End-to-end toolchain that integrates all features w/ additional enterprise support (customization, support, security SLAs)

WIP



Vite

Dev server / HMR
Application support
Framework support

production



Vitest

Vite-native test runner
Node.js + Browser

production



Rolldown

Dev + Prod Bundler

beta



Oxc

Language toolchain

Composable NPM Packages & Rust Crates

Linters ✓

Formatter

WIP

Transformer ✓

Resolver ✓

Minifier

alpha

Parser ✓

Semantic Analysis ✓

目前进度



OXC - Progress



Parser

100% ECMA 262 spec compliance + TypeScript, JSX
Semantic Analysis + Control Flow Graph



Lint*

400+ ESLint compatible rules
IDE extensions / WIP linter plugin design in collab w/ Deno team



Resolver

Node.js compatible CJS + ESM path resolution
Fully customizable



Transformer

TypeScript, React JSX, React Fast Refresh, isolatedDeclarations DTS,
Syntax-lowering



Minifier

Alpha, integrated in Rolldown as default minifier



Formatter

WIP (50%)



OXC - Performance



The fastest parser

3x faster than SWC / 5x faster than Biome
[Benchmark](#)



The fastest linter

50~100x faster than ESLint
[Benchmark](#)



The fastest resolver

28x faster than webpack/enhanced-resolve
[Benchmark](#)



The fastest transformer

TS / TSX: 4x faster than SWC / 40x faster than Babel
React Refresh: 6x faster than SWC / 70x faster than Babel
Isolated Declarations DTS Emit: 20~45x faster than tsc
[Benchmark](#)



The minifier with the best speed + compression ratio combo

8x faster than SWC
50% faster than esbuild
[Benchmark](#)



Rolldown - Progress



核心打包

CJS + ESM interop / Source maps / Output formats
Basic code splitting / CLI / Config file support



Oxc 整合

Built-in transforms / Node resolver



高级功能

Tree-shaking / define / inject
Advanced chunk split options



兼容性对齐

80%+ esbuild / Rollup test coverage
90% Rollup plugin compatibility



1.0 Beta

Performance tuning + code quality polish
Documentation



Vite 整合

Rolldown-vite

```
{  
  "overrides": {  
    "vite": "npm:rolldown-vite@latest"  
  }  
}
```



Rolldown-Vite 性能案例

<https://github.com/vitejs/rolldown-vite-perf-wins>

- Excalidraw: 16x faster build
- PLAID, Inc.: 8~16x faster build
- Particl: 9.7x faster build
- Payfit: 4.7x faster build
- Appwrite: 3.7x faster build, 4x less memory
- Linear: 3.5x faster build
- Mercedes Benz: 3x faster build
- GitLab: 2.6x faster build, 100x less memory

Rolldown 产物优化

- Advanced chunks 高级分包
- strictExecutionOrder 严格模块执行顺序
- CJS treeshaking
- WIP: 跨模块 DCE
- WIP: import map chunking

WIP: Full Bundle Mode

- 针对超大型应用, 优化 dev 下页面 load 性能
- 基于 ESM 的新 HMR 格式
- 增量构建
- Lazy 构建

Oxlint 1.0

- 500+ ESLint rules
- 比 ESLint 快 50~100 倍
 - Used by Shopify, Airbnb, Mercedes Benz
- 开发中: JS 插件 / 自定义 rule 支持
 - 黑科技: Raw AST transfer
- 计划中:
 - Type-aware integration with tsslint
 - Vue SFC 支持

tsdown

- [tsdown](#) - tsup 的继承者, 基于 Rolldown
- TypeScript dts generation + bundling
 - isolatedDeclarations
 - Rolldown-plugin-dts

Vite+

“cargo for JavaScript”

- Vite 的严格超集
- vite dev / build (Vite / Rolldown)
- vite test / bench (Vitest)
- vite lint / format (Oxc)
- vite lib (tsdown)
- vite new - 项目创建、代码生成
- vite task - Monorepo 任务编排 / 缓存系统
- GUI devtools
- AI agent tools

Vite+

As “framework for frameworks”

- “Framework as Vite plugin”
 - Remix / React Router 7
 - TanStack Start
 - Redwood SDK
- “Framework as Vite+ plugin”
 - Inject lint rules
 - Inject custom test matchers / envs
 - Inject devtools panels

Thank You!