

StruktJS

LangDev
CON2026

An AI-Driven, Web-Based Projectional Language Workbench

Alexander Pann · F1RE

The screenshot displays the StruktJS web-based projectional language workbench interface. The interface is divided into several panels:

- Left Panel (Scripting Demo):** Contains a breadcrumb trail: `nn > .struktjs > samples > maze-explorer > maze-explorer.model.py`. Below this, it shows the current script being edited: `world scripting demo`. The script content includes a comment: `maze Explorer (grid-world scripting demo):` and a description: `rts at (0, 0) facing north. This grid's internal walls edurally generated from the script's own name and size maze preview above) - a different name or size tes a different maze.` The script ends with `plorer (5 x 5):`. A small preview of the maze is shown at the bottom left of this panel.
- Right Panel (Preview):** Displays a 5x5 grid maze. A small robot icon is positioned at the top-left corner (0, 0), facing north. The maze is generated based on the script's content. The panel is titled `Preview` and includes a `explorer` sub-panel with the text `5 x 5 grid`. Below the maze, there are controls: `Step`, `Play`, and `Reset` buttons, along with a `Speed` slider.

The gaps

Projectional editors

Desktop only, custom VCS, models are not plain text

Grammar-based editors

Text-only output, no non-text projections, re-parsing on every keystroke

Web-based projectional editors

Browser-only — no native desktop app from the same codebase

No out-of-the-box collab

Real-time editing on a bespoke model store, not on plain, diffable source text

Closing the gaps

Web-native, not desktop-only

Browser IDE and native Electron app, built from the same extension codebase

Structural editing

Built-in non-text projections — diagrams, tables, MathML, custom HTML — no re-parsing

Real source files

Code round-trips to plain text via tree-sitter (Python/JS), not a bespoke model format

Real-time collaboration

Live Model Session: collaborative AST editing over Yjs, on that same plain-text-backed model

What is StruktJS?

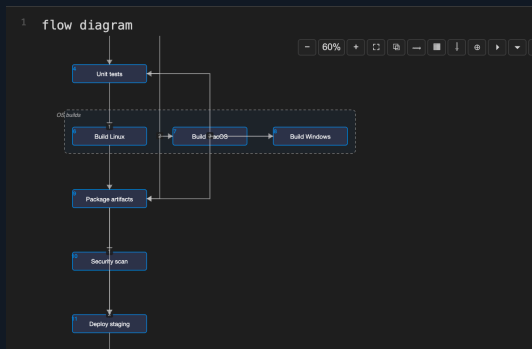
```
# --- Arithmetic & roots -----
# A fraction, stacked numerator over denominator.
print  $\frac{8}{4}$ 

# A power: base raised to a raised exponent.
print  $\frac{8^2}{4}$ 

# A square root.
print  $\sqrt{16}$ 

# A generalized nth root – the cube root of 27.
print  $\sqrt[3]{27}$ 

# Absolute value of a negative quantity.
print |-5|
```



```
Surface Full form
1 flowDiagram(
2   flowNode("Checkout code", "checkout", None, None, lint, compile, shape="ell
3   flowNode("Lint", "lint", None, None, unitTests),
4   flowNode("Compile", "compile", None, None, unitTests),
5   flowNode("Unit tests", "unitTests", None, None, buildLinux, buildMac, build
6   flowGroup("OS builds", "osBuilds", None, None, buildLinux, buildMac, build
7   flowNode("Build Linux", "buildLinux", None, None, package),
8   flowNode("Build macOS", "buildMac", None, None, package),
9   flowNode("Build Windows", "buildWin", None, None, package),
10  flowNode("Package artifacts", "package", None, None, securityScan, deploySt
11  flowNode("Security scan", "securityScan", None, None, deployStaging),
12  flowNode("Deploy staging", "deployStaging", None, None, smokeTests),
13  flowNode("Smoke tests", "smokeTests", None, None, deployProd, rollback),
14  flowNode("Deploy production", "deployProd", None, None, notifySuccess),
15  flowNode("Rollback staging", "rollback", None, None, notifyFailure),
16  flowNode("Notify success", "notifySuccess", None, None, shape="ellipse"),
17  flowNode("Notify failure", "notifyFailure", None, None, shape="ellipse"),
18 )
```

- A projectional language workbench built on Eclipse Theia and standard web technologies
- Code is edited/stored as an Abstract Syntax Tree while still round-tripping to source files
- Support for different projections: diagrams, tables, MathML, custom HTML etc.
- Full internationalization and customization support
- Demo: Different notations + Maze demo

Validation & Analysis

- Checker: severity-tagged diagnostics per node
- Constraint engine: structural constraint violations
- Static type system: optional type inference/checking per concept
- Scope analysis: in-scope variable computation
- Linting: backend Ruff integration for the Python-grammar model

```
1 import math
2 from typing import List
3 def stats(nums: List[int]) -> int:
4     total = 0
5     scratch = 0
6     biggest = nums[0]
7     for i in 0..len(nums):
8         total = total + nums[i]
9         if biggest < nums[i]:
10             biggest = nums[i]
11         + else
12     return total
13 def doubled(x: int) -> int:
14     result = x * 2
15     return result
16 def sign(x: int) -> int:
```

Problems 29 x M4README.md Console Messages zsh

29

- △ Incomplete — this slot hasn't been filled in yet. StruktJS [Ln 1, Col 1]
- △ `math` imported but unused (F401) StruktJS [Ln 1, Col 8]
- △ `typing.List` is deprecated, use `list` instead (UP035) StruktJS [Ln 2, Col 1]
- △ Use `list` instead of `List` for type annotation (UP006) StruktJS [Ln 3, Col 17]
- △ Local variable `scratch` is assigned to but never used (F841) StruktJS [Ln 5, Col 3]
- △ Unnecessary `start` argument in `range` (PIE808) StruktJS [Ln 7, Col 18]
- △ Replace `if` statement with `biggest = max(biggest, nums[i])` (PLR1730) StruktJS [Ln 9, Col 5]
- △ Unnecessary assignment to `result` before `return` statement (RET504) StruktJS [Ln 14, Col 10]
- △ Missing explicit `return` at the end of function able to return non-`None` value (RET503) StruktJS [Ln 15, Col 1]
- △ Do not use mutable data structures for argument defaults (B006) StruktJS [Ln 20, Col 26]
- △ Unused function argument: `label` (ARG001) StruktJS [Ln 20, Col 30]
- △ PEP 484 prohibits implicit `Optional` (RUF013) StruktJS [Ln 20, Col 37]
- △ Unused function argument: `unhappy` (ARG001) StruktJS [Ln 20, Col 40]

Code Generation

- Text generator: template-driven source output
- Model-to-model generator
- Live HTML/JS preview



explorer
5 × 5 grid

⏮ Step ▶ Play ⏹ Reset

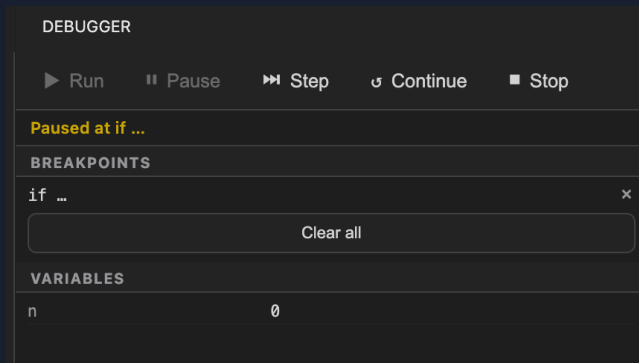
Speed

- Item placed at (0, 1).
- Byte turns right, now facing E.
- Byte turns right, now facing S.

Execution, Debugging & Version Control

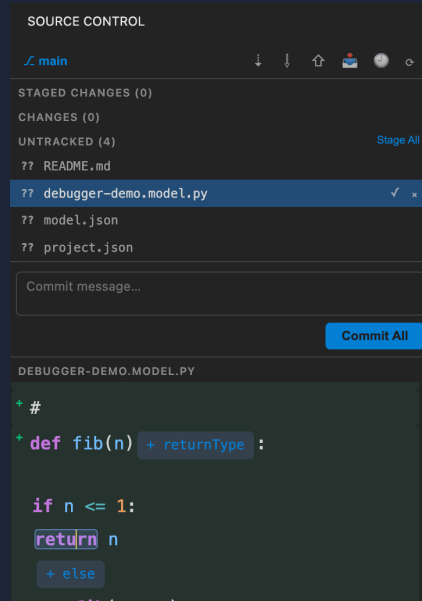
Execution & Debugging

- Built-in evaluator
- Debugger pane and Type Trace pane for stepping through and inspecting evaluation



Diffing & Version Control

- AST-aware diff view
- Git pane integrated directly into the IDE



AI Support & Collaboration

- AI compatibility layer: tested against local LLMs for AI-assisted language authoring
- Live Model Session: real-time collaborative AST editing over Yjs

condition	result
Filter...	Filter...
score >= 90	"A"
score >= 80	"B"
score >= 70	"C"
score >= 65	"D"
score >= 60	"D"
otherwise	"E"

```

11 print("Grade: ")
12 print(grade)
  
```

Participants (choose one to make host):

You (you) — host

You [Completing...]

Cancel

StruktJS

Guess the number x

```

1 def is_prime(n) > returnType :
2   if n <= 1:
3     return false
4   + else
  
```

Problems

Console

Messages x

README.m

☒ Output
 ☐ System
 ☐ Info
 ☐ Warn
 ☐ Error

Copy All

Clear

```

2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
  
```

AI CHAT

You

Create me a model where the prime numbers from 1 to 100 are outputted and run the program.

StruktJS

ollama/llama3.1:8B-64k

Ran struktjs_apply_code(import math def is_prime(n): ...)

Ran struktjs_get_editor_context()

Ran struktjs_apply_code(import math def is_prime(n): ...)

Ran struktjs_get_model()

Ran struktjs_run_program()

The prime numbers from 1 to 100 are outputted and the program ran successfully.

.INE

TzgEQONM2QfLiumuX6BpKmNt · 2 · Host

AI Workflow with Claude Code 🌟

- Claude Code with Sonnet + Opus as advisor, auto mode enabled

Phase 1: find ideas & prototyping

- Short feature descriptions; Claude asks clarifying questions if needed
- Multiple prototyping iterations
- Questioning of existing approaches

Phase 2: ongoing development

- Code maintenance
- dead code/technical debt elimination
- Parallel agents for similar fixes

```

Read 1 file

• All 416 test files / 9,732 tests pass, zero failures.

Summary

Explorer/Outline "where did they go" - both are still there, not removed:
- Stubs: nested by an earlier commit into Explorer > Project > Languages > core > stubs (collapsed 3 levels deep by default).
- Outline: embedded in the same Explorer view, but intentionally hides whenever a project editor tab isn't focused (struktjs-shell-control

Stub-informed type inference - implemented for both languages, return type + argument checking as you asked:
- src/typesystem.ts: import <name> now binds the module in scope (moduleBindingType/moduleNameFromBindingType), so a receiver can be recognized
- Python (plugins/lang-core/languages/concepts-modules.tsx): AttrCall/AttrAccess consult stub-registry.ts's real typeshed data - math.floor(10.5)
  (unwrapping Final[float]). A _stubType() guard keeps typeshed's internal TypeVar/Protocol noise (_T, SupportsIndex, ...) from causing false-p
- TypeScript (plugins/lang-ts/languages/concepts-ts-expressions.tsx): TsCallExpression/IsMemberExpression do the same against js-stub-registry.
  Math.PI infers number, mismatches are flagged. Real import * as ns from "..." bindings aren't tracked yet for TS - documented as a known, nar
- New test tests/test-stub-informed-typesystem.mjs (6 tests) plus a fix to tests/test-ts-concept-hook-coverage.mjs's recorded hook inventory for

Changed files: src/typesystem.ts, plugins/lang-core/languages/concepts-modules.tsx, plugins/lang-ts/languages/concepts-ts-expressions.tsx, tests
committed - let me know if you'd like these committed.

* Churned for 16s · done 8:10 AM

* recap: You asked about missing Explorer/Outline items (both fine, just relocated) and wanted math.floor-style type inference wired to real stubs
  to commit it if you want. (disable recaps in /config)
  
```

Issues: What went wrong

Claude

- Output too verbose
- AI code is hard to read, modify, and maintain manually
- Different test paths than the user goes through
- High token usage (parallel agents)
- Anthropic API issues: Claude overloaded, agents becoming dumb

Development

- Bugs fixed in only one layer (e.g. web view only)
- Stale cache issues (incognito mode)
- Race conditions and wrong initialization order

What I Learned & Optimizations

Lessons

- Vague instructions → misinterpretations and wrong features
- A few bad initial decisions → multi-day effort to fix
- Getting influenced by existing solutions
- No TDD at start → breaking changes; now 11.3k tests
- Test types: Vitest (unit), Playwright E2E, AI agent eval, static type checks

Optimizations

- No external MCP servers
- Split long files into smaller ones (less token usage)
- "Come up with a plan": give examples to Claude first
- Outsource simple tasks to other LLMs to save tokens
- Recurring failure patterns captured as named, reusable "bug classes"

Result and Takeaway

362K

total LOC

4

months

35/21

plugins/samples

0.86

test/code ratio

The first workbench to combine real-time collaborative AST editing, non-text projections, and plain-text round-tripping — in a browser and desktop app from one codebase — built solo in 4 months using an AI-native workflow."

Questions?

Comparison with other LWBs

	JetBrains MPS	Eclipse Xtext	Langium	StruktJS
Editing paradigm	Projectional	Textual (grammar-parsed)	Textual (grammar-parsed)	Projectional
Git-diffable source	No — custom VCS	Yes (native)	Yes (native)	Yes — real source round-trips
Tech stack	Java / Swing	Java / EMF	TypeScript / Node	TypeScript / web
Deployment	Desktop only	Eclipse or LSP client	Any LSP client	Browser IDE + native desktop
Non-text projections	Possible, high effort	Not applicable	Not applicable	Built-in: Diagram / Table / MathML / HTML
Real-time collaboration	Not built in	Not built in	Not built in	Built in — Live Model Session
AI-assisted authoring	Not built in	Not built in	Not built in	Built in

Comparison with other LWBs

	Modelix	Freon	StruktJS
Editing paradigm	Projectional (MPS editors forwarded to browser)	Projectional	Projectional
Git-diffable source	No — models live in a shared model server, not plain source files	No — own model/JSON format, not a real host-language grammar	Yes — real source round-trips
Tech stack	Kotlin/MPS backend + web frontend	TypeScript / Svelte	TypeScript / web
Deployment	Browser only (extends an MPS server)	Eclipse or LSP client	Browser IDE + native desktop
Non-text projections	Inherits MPS's editor set (no dedicated diagram/table/MathML kit)	Not a documented focus	Built-in: Diagram / Table / MathML / HTML
Real-time collaboration	Yes — built in (its core feature)	Not built in	Built in — Live Model Session
AI-assisted authoring	Not built in	Not built in	Built in — tested against local LLMs

Tech Stack

Eclipse Theia



Browser IDE + native Electron app, same extension codebase

TypeScript



Full codebase except tests

SolidJS



Reactive UI for the editor

web-tree-sitter



Parses Python / JS surface syntax into the AST

KDL



Canonical AST text serialization format

Yjs



Real-time collaborative live-model editing

Live Model Session: Under the Hood

- **Changes sync instantly**

Every edit (adding, moving, or deleting) is shared with everyone in the session and merged into their view automatically, no reload needed

- **Typing together, live**

If two people are typing in the same spot at the same time, both sets of keystrokes are woven together — instead of one person's edit silently overwriting the other's

- **Conflicts resolve fairly**

Edits to different parts of the model never clash. Edits to the exact same spot merge together if both people are actively there, or the most recent one wins if not

- **Nothing gets lost**

In-progress work is saved before a teammate's change arrives, undo won't erase someone else's edit, and a quick highlight shows you exactly what just changed and who changed it