Jennifer for coders



You already program in another language. This is what is different here, and why.

by @mplx

The language in one slide

- Interpreted, one self-contained binary, no runtime to install
- Types are written out and checked; no implicit conversions
- No truthiness: a condition is a `bool` or an error
- Value semantics: assignment and arguments copy
- Errors are values you `throw` and `catch`
- Batteries included: a standard library and decks for the rest

Three kinds of names

```
use io;

def const LIMIT as int init 3;
def count as int init 0;

func bump(n as int) {
    return $n + 1;
}

$count = bump($count);
io.printf("%d of %d\n", $count, LIMIT);
```

Strict by default

```
$ jennifer run sum.j
sum.j: runtime error at /home/ada/sum.j:2:19: operator + requires
numeric operands, got string and int
```

```
use io;
use convert;

def n as int init convert.toInt("42");
io.printf("%d\n", $n + 1);
```

Numbers that do not lie

Expression	Result
7 / 2	3.5: / is always float division
7 // 2	3: floor division
-7 % 3	2: floored, to match //
9223372036854775807 + 1	error: integer overflow
1e308 * 10.0	error: no infinity, no NaN

Strings: cooked, raw, and sliced

```
use io;

def word as string init "Jennifer";
io.printf("%s\n", $word[0..3]);
def pattern as string init '\d+\.\d+';
io.printf("%d chars, {$word}\n", len($pattern));
```

```
Jen
8 chars, Jennifer
```

Assignment copies

```
use io;

def a as list of int init [1, 2, 3];
def b as list of int init $a;
$b[] = 4;
io.printf("%d %d\n", len($a), len($b));
```

```
3 4
```

Structs, enums and match

```
use io;

def enum Shape { Circle { r as float }, Rect { w as float, h as float
} };

func area(s as Shape) {
    match ($s) {
        when Circle(c) { return 3.14159 * $c.r * $c.r; }
        when Rect(r) { return $r.w * $r.h; }
    }
}

io.printf("%f|prec=2\n", area(Shape.Rect{w: 2.0, h: 1.5}));
```

Errors are values

```
use io;

func work() {
    defer io.printf("cleanup\n");
    throw Error{kind: "io", message: "disk full"};
}

try {
    work();
} catch (e) {
    io.printf("caught %s: %s\n", $e.kind, $e.message);
}
```

Functions are values

```
use io;
use lists;

func isEven(n as int) { return $n % 2 == 0; }
func square(n as int) { return $n * $n; }

def xs as list of int init 1..7;
def out as list of int init lists.map(lists.filter($xs, isEven),
square);
io.printf("{$out}\n");
```

```
[4, 16, 36]
```

Libraries, modules and decks

```
export def const EARTH_KM as float init 6371.0;

export func circumference(r as float) {
    return 2.0 * 3.14159 * $r;
}

use io;                              # a built-in library
import "./geo.j" as geo;             # a module of your own
import "@jennifer/telegram/";        # a deck, vendored by jvc
```

Concurrency with spawn

```
use io;
use task;

func sumTo(n as int) {
    def total as int init 0;
    for (def i in 0..$n) {
        $total = $total + $i;
    }
    return $total;
}
def a as task of int init spawn { return sumTo(1000); };
def b as task of int init spawn { return sumTo(2000); };
io.printf("%d\n", task.wait($a) + task.wait($b));
```

Tests are built in

```
use testing;
import "./geo.j" as geo;

func testCircumference() {
    testing.assertEqual(geo.circumference(1.0), 6.28318);
}
```

```
$ jennifer test geo_test.j
PASS testCircumference (0 ms)

1 passed, 0 failed, 1 total
```

The toolbox in the binary

Command	Does
jennifer run, repl	run a file, try a line
jennifer fmt -w	format the code the one canonical way
jennifer lint	catch what compiles but is suspect
jennifer test	run the test functions
jennifer profile	time and allocations per line
jennifer tokens, ast	show what the parser sees

Packages with jvc

```
jvc add "@jennifer/telegram" "^0.1.0"   # a requirement in deck.toml
jvc install                             # vendor what camcorder.lock
pins
jvc update                              # move to the newest allowed
```

Decks live at registry.jennifer-lang.dev.

The traps, once more

Elsewhere	In Jennifer
x = 5	def x as int init 5; then $x = 6;
x += 1, i++	$x = $x + 1;
&&, ||, !	and, or, not
// comment	# comment (// is floor division)
s[0]	$s[0..1]
my_var	myVar (MY_CONST for constants)

Where to go next

- The user guide and every library: docs.jennifer-lang.dev
- Decks to build on: registry.jennifer-lang.dev
- More talks: the Jennifer Academy

Based on Jennifer 0.26.0, November 2026.