The Schema Language: A Tutorial from Zero

For the Comparator and SchemaFit Timer

Serega Photoskiff

2026

Contents

0. Why Schema exists **2**

1. The simplest atom: work / recovery **2**

2. Continuous work **2**

3. Repetition: `*N` **3**

4. Intensity: `@` **3**

5. Repetitions: `@rN` **3**

6. Tempo: `@tRPM` **3**

7. Combining intensity, repetitions and tempo **4**

8. Groups **4**

9. Nested groups **4**

10. Tail metadata **5**

11. Hard finish `#` and soft continuation `;` **5**

12. Tail-no-rest **5**

13. Local duration: `dur:N(...)` **5**

14. Deferred values: `~` and `~!N` **5**

15. Active and deferred Schemas **6**

16. Progressions **6**

17. Tempo ramps and adaptive atoms **6**

18. Random Schemas **7**

19. Global constants **7**

20. A practical reading algorithm **7**

21. Common errors **7**

22. Practice: simple to complex **8**

23. What makes a Schema powerful **8**

24. A learning ladder **9**

25. Active, deferred, materialised and realised **9**

26. Schema as an exchange language **9**

27. Visual vocabulary **9**

28. Short reference **10**

29. Final thought **10**

0. Why Schema exists

An ordinary interval timer can describe “work for 30 seconds, recover for 30 seconds, repeat ten times”. That is enough until a workout becomes more realistic: tempo changes, recovery disappears, some exercises use time and others use repetitions, and part of the session is known only when the athlete performs it.

Schema records all of this in a compact executable line:

```
20/10@8.5*8#
```

This is a complete Tabata-style session: 20 seconds of work, 10 seconds of recovery, work intensity 8.5, eight rounds, then a hard finish.

The central idea is that a Schema is not merely a note for a person. It is a compact workout program for a timer, graph and analysis system.

1. The simplest atom: work / recovery

The basic unit is an **atom**:

```
30/30
```

The first number is ON, or work. The second is OFF, or recovery. Add an end marker to make a complete workout:

```
30/30#
```

`#` means execute the Schema and finish the session.

2. Continuous work

Write zero explicitly when there is no recovery:

```
300/0#
```

This is five minutes of continuous work. Explicit `/0` distinguishes “no recovery” from “recovery not specified”.

3. Repetition: *N

```
30/30*5#
```

Repeat the `30/30` atom five times. Total duration is `(30 + 30) x 5 = 300` seconds.

4. Intensity: @

```
30/30@8*5#
```

Work for 30 seconds at intensity 8, recover for 30 seconds, repeat five times.

To give intensities for both sides:

```
30/30@8/2*5#
```

The work/recovery correspondence is preserved after `@`: work intensity 8, recovery intensity 2.

Intensity is not decoration. Combined with the athlete's parameters and session structure, it helps construct a plausible heart-rate and load model.

5. Repetitions: @rN

```
60/0@r20#
```

Work for 60 seconds with a target of 20 repetitions.

```
60/0@r20*10#title=10-minute set
```

Ten one-minute blocks, each with a target of 20 repetitions.

6. Tempo: @tRPM

```
60/0@t60#
```

Work for 60 seconds at a tempo of 60 counts or movements per minute.

`r20` and `t20` are related but different:

- `r20` - perform 20 repetitions during the block;
- `t20` - maintain a rate of 20 per minute.

7. Combining intensity, repetitions and tempo

`r` and `t` are mutually exclusive on one side of an atom. Intensity is an independent layer and may accompany either.

Schema	Prescribed values
`60/0@8#`	intensity only
`60/0@r20#`	target repetitions; intensity comes from defaults
`60/0@t60#`	tempo; intensity comes from defaults
`60/0@8r20#`	intensity + target repetitions
`60/0@8t60#`	intensity + tempo

A HIIT example:

```
15/15@8.5t40/2t20*10`#
```

Fifteen seconds of work at intensity 8.5 and tempo 40, then 15 seconds of active recovery at intensity 2 and tempo 20, repeated ten times without the final recovery.

A SIT example:

```
30/210@9.5t30*5`#
```

Five very hard 30-second efforts separated by 210 seconds of recovery, with the final recovery removed.

When intensity is omitted, the timer can take it from current settings. This allows a theoretical heart-rate curve to be built even before the value has been calibrated.

8. Groups

```
(30/30 20/40)*3#
```

Perform `30/30`, then `20/40`, and repeat the whole group three times. Parameters after `@` belong to atoms, not to a group as a whole:

```
(30/30@8 20/40@8)*3#
```

9. Nested groups

```
((30/30)*2 (20/40)*3)*4#
```

Within each outer cycle, repeat `30/30` twice and `20/40` three times; repeat the whole structure four times. Read from the outer parentheses inward.

10. Tail metadata

Metadata follows # or ;:

```
60/0@t60*5#title=Five-minute challenge
```

The executable core is `60/0@t60*5#`. The title helps the interface but does not change workout physics.

Numbered labels connect segments to metadata:

```
30!1/0@t60 30!2/0@t60#title=Ring support 1=position one 2=position two
```

11. Hard finish # and soft continuation ;

```
20/10*8#
```

Execute once and finish. With ;, the execution environment may continue the remainder of the session using its standard mechanism. For most ordinary workouts, use #.

12. Tail-no-rest

```
40/30*5`#
```

The backtick after the repeat removes recovery from the final repetition only.

13. Local duration: dur:N(...)

```
dur:30((~!1/0@r5 ~!2/0@r10 ~!3/0@r15)*~)#
```

Repeat a Cindy-style circuit - five pull-ups, ten push-ups, fifteen squats - within a 30-second local time budget, recording actual segment durations during execution.

The duration belongs in the executable core:

```
dur:30((...)*~)#
```

not in tail metadata.

14. Deferred values: ~ and ~!N

```
~/0@r~*~#title=Recorder
```

The work duration, repetition count and number of sets will all be known only during execution.

```
~!1/0@r5# 1=pull-ups
```

`~!1` is a deferred duration linked to metadata label 1.

15. Active and deferred Schemas

An active Schema knows its structure in advance:

```
20/10@8.5*8#
```

The timer can build the complete theoretical graph before the start.

A deferred Schema contains `~`:

```
~/~@r~*~#title=Recorder with recovery
```

The graph emerges during execution. Active Schemas suit intervals, tempo sessions and pre-session modelling. Deferred Schemas suit AMRAP, manual logging and workouts where actual execution determines the structure.

For an active Schema, the timer may display three layers together:

1. theoretical heart rate;
2. measured heart rate from a monitor;
3. a previous performance of the same Schema.

16. Progressions

```
30/30->[for:3,delta:-5/+5]#
```

Start with `30/30`; produce three steps; on each step reduce work by five seconds and increase recovery by five:

```
30/30
25/35
20/40
```

A two-way pull-up ladder can be expressed as:

```
(2/2@r1->[to:auto/auto@r10,delta:+2/+2@r+1]
->[to:auto/auto@r1,delta:-2/-2@r-1])*1`#
```

17. Tempo ramps and adaptive atoms

```
60/10@t[15->25]#
```

Tempo rises from 15 to 25 inside one 60-second work atom. A step may be supplied:

```
60/10@t[15->25 delta:2]#
```

A practical example:

```
58/2@t[18->26]*10`#title=accelerating high swing
```

An adaptive active atom is marked with ^:

```
300/0@t180^#
```

The session has an initial theoretical plan, but the athlete may alter tempo during execution. It remains an active Schema; it does not become a free-form Recorder.

18. Random Schemas

```
random(30/0@9t220 30/0@6t140 30/0@7t180 turb:7 dur:580 min:8)
20/0@t[200->280]#title=broken-pace run
```

The runtime selects among permitted atoms. `dur` sets the random block's duration, `turb` its turbulence, and `min` the minimum hold time. Randomness is constrained, not arbitrary. Add `seed` to reproduce the same materialised pattern.

19. Global constants

```
[low=12 high=24]
(150/0@t[low->high] 145/0@t[high->low] 0/10)*2#
```

Constants make a long Schema easier to adjust and explain.

20. A practical reading algorithm

When a Schema looks unfamiliar:

1. Find the end of the executable core: `#` or `;`.
2. Find the largest groups in parentheses.
3. Find repeats: `*5`, `*~`, `*5`` `.
4. Read each atom: ON/OFF, `@`, `r`, `t`.
5. Read metadata: `title=`, numbered labels, voice and metronome settings.

21. Common errors

Wrong - spaces inside an atom:

```
30/30 @8 t20 #
```

Right:

```
30/30@8t20#
```

Wrong - fractional repeat:

```
30/30*3.5#
```

Wrong - applying `@` to a group:

```
(30/30 20/40)@8*3#
```

Right:

```
(30/30@8 20/40@8)*3#
```

Wrong - deferred tempo:

```
30/0@t~#
```

Wrong - a numbered label without matching metadata:

```
30!1/0#
```

Right:

```
30!1/0# 1=pull-ups
```

22. Practice: simple to complex

```
20/40*10#                          simple intervals
20/40@8/2*10#                     work 8, recovery 2
20/10@8.5*8#                      Tabata-style session
60/0@t60*5#                       five minutes with a metronome
60/0@r20*10#                      ten one-minute repetition targets
(30!1/0@r5 30!2/0@r10
 30!3/0@r15)*5#                   three-exercise complex
```

A manually recorded AMRAP:

```
dur:1200((~!1/0@r5 ~!2/0@r10 ~!3/0@r15)*~)#
manual=1 metro=0 title=Cindy
1=pull-ups 2=push-ups 3=squats
```

23. What makes a Schema powerful

The value is not merely brevity. One record can be used to:

1. model and compare a session;
2. execute it in SchemaFit Timer;
3. edit it and immediately inspect the resulting graph;
4. store it as a preset;
5. analyse intended and actual load;
6. compare modelled, measured and previous heart-rate curves;
7. exchange an exact executable workout with another person.

24. A learning ladder

Level 1 - read simple Schemas: ON/OFF, `#`, `*N`, `@8`, `@r20`, `@t60`.

Level 2 - write interval workouts: work/recovery intensity, groups, backtick and titles.

Level 3 - write complexes: numbered labels, metadata binding, nested groups and manual mode.

Level 4 - write AMRAP and deferred Schemas: `~`, `~!N`, `dur:N(...)`, `*~`.

Level 5 - write variable sessions: progressions, tempo ramps, constants and random blocks.

25. Active, deferred, materialised and realised

- **Active:** the structure is known before execution.
- **Deferred:** one or more values are discovered during execution.
- **Materialised:** a concrete active form has been produced from deferred values.
- **Realised:** the athlete's actual execution of the plan, including any permitted changes.

These forms allow analysis to distinguish intention, executable structure and reality.

26. Schema as an exchange language

Verbose instruction:

> Perform ten HIIT rounds: 15 seconds active at intensity 8.5 and tempo 40, followed by 15 seconds of light recovery at intensity 2 and tempo 20. Skip recovery after the final round.

Schema:

```
15/15@8.5t40/2t20*10`#
```

The first is readable prose. The second is readable by a person, timer, graph, analyser and workout history. A Schema can be stored, published, sent to an athlete or transferred by QR code.

27. Visual vocabulary

Term	Meaning	Example
Atom	smallest executable unit	`30/30@8`
Repeat	repeats an atom or group	`30/30*5#`
Group	combines elements	`(30/30 20/40)*3#`
Scope	local time or repetition budget	`dur:1200((...)*~)#`
Tail	metadata after # or ;	`#title=Five minutes`
Progression	changes values between steps	`->[for:3,delta:-5/+5]`
Ramp	changes tempo inside an atom	`@t[15->25]`
Deferred value	determined during execution	`~`
Adaptive atom	active atom adjustable in execution	`@t180^`

28. Short reference

Schema	Meaning
`30/30#`	work 30, recover 30, finish
`30/30*5#`	repeat five times
`30/30@8/2#`	work intensity 8, recovery intensity 2
`60/0@r20#`	60 seconds, target 20 repetitions
`60/0@t60#`	60 seconds, tempo 60
`(30/30 20/40)*3#`	repeat a two-atom group three times
`40/30*5`#`	remove final recovery
`~/0@r~*~#`	free-form Recorder
`dur:1200((...)*~)#`	execute within a 20-minute scope
`60/10@t[15->25]#`	tempo ramp
`300/0@t180^#`	adaptive active tempo atom

29. Final thought

Schema is compact, but compactness is not the goal. The goal is a workout that can be described precisely, executed reproducibly, compared with a model, and understood through repeated real sessions.