

```

--: {place: [], (😊, moves: [], (😊, steps: [], (😊) } <- Places Functions Library ->
muse/docs/lib/places.md
--😊 places: Naming places at MUSE coordinates, moving there, stepping there for operations. ->
places, place, moves, steps
--:+ place: Name places (points, trails, ranges); serialize and load serializations for disk and
network operations.
--:+ moves: Move turtles to named places or along named trails.
--:+ steps: Iterator to move block by block to named places or along named trails.

--:# Type definitions that will be serialized for network transport and disk storage

--:> place: A point, trail, or range -> {name: ":", label: ":", :situations:, :features:}

--:> features: Dictionary of string key, any value pairs -> [key: ""]: any

--:# Utilities for places (points, trails, and ranges)

--:: place.reset() -> Resets places to the empty table. -> nil

--:: place.count() -> Returns number of places. -> #:

--:: place.site(value: "?:") -> Set or return local site (isolates global). -> ":"

--:: place.qualify(name: "😊 -> Return already sited name, otherwise prepend site to name ->
sitedName: ":"

--:: place.distance(a: xyzf, b: xyzf) -> Manhattan: abs(delta x) + abs(delta y) + abs(delta z). ->
distance: #:

--:: place.match(name: "😊 -> Lookup place qualified by site, return nil if not found. -> index:
#?:, place?

--:: place.xyzf(name: "?:", number: #?:) -> Looks up name [defaults to current situation]. -> xyzf?,
index: #?:

--:: place.name(name: ":", label: ":", supplied: situation?, :features:??) -> Make or update place. ->
":", #:

--:+ Include current situation or optionally supplied situation in places. Optionally update features
with key = value.
--:+ Return index of situation in global places and the serialized situation including its features.

--:: place.add(name: ":", :situation:) -> Add situation to situations of an existing place. ->
serialized: ":", index: #:

--:: place.erase(name: "😊 -> Removes named place from array of places. -> #:, index: #:
--:+ Return new length of places table and the (previous) index of the removed place.

--:# Answering "where?"

```

--:: place.near(span: #:?, reference?: ":"|position) -> __ -> (): name: ":", label: ":", xyz, distance: #:, situations, serial: ":"

--:+ If both span and name (or a position) are specified, return places within a span of blocks of the named place (or position).

--:+ If only the span is specified, return places within a span of blocks of the current situation or player position.

--:+ If neither is specified return each of the named places. In any case, iterator returns include serialized places.

--:: place.nearby(:xyzf:?, :cardinals:) -> Sorted -> `[distance: #:, name: ":", label: ":", cardinal: ":", :xyzf:]

--:> cardinals: Function to get one of the eight cardinal points of the compass -> (dx: #:, dz: #😊: cardinal: ":"

--:+ Nearest places to specified xyzf coordinates or current position (as default).

--:+ Returned table is sorted by distances and includes the name, label, and xyzf position of each place.

--:+ If a *cardinals* function is supplied, the eight point cardinal direction is also included.

--:# Support for trails (names and labels for sequences of situations)

--:: place.fix(:xyzf:, track: ^:?) -> Sets situation position, can start tracking for trail. -> xyzf

--:: place.trail(headName: ":", tailName: ":", label: "😊 -> Makes two places. -> headSerial: ":", tailSerial: ":"

--:+ Trail places share a label and represent trails from head to tail and tail to head; head set by *place.fix*.

--:: place.track(name: "😊 -> Returns trail -> name: ":"?", label: ":"?", situations?

--:# Moving and stepping for known places: to points or along trails

--:: moves.along(name: "😊 -> Move from first to second situation of place. -> code: ":", remaining: #:, xyzf: ":" &! recovery

--:+ If the named place is the head of a trail, go from there to its tail. If it's a tail of a trail, go to its head.

--:: steps.along(name: "😊 -> Iterator: first to next situation of place. -> (): code: ":", remaining: #:, xyzf: ":" &! recovery

--:+ If the named place is the head of a trail, step from there to its tail. If it's a tail of a trail, step to its head.

--:: moves.to(target: ":", first: "😊 -> Move to target, first along direction. -> code: ":", remaining: #:, xyzf: ":" &! recovery

--:: steps.to(target: "😊 -> Step (iterator) to target place. -> (): code: ":", remaining: #:, xyzf: ":" &! recovery