

--{:! {map: []} (:)} <- **Map Command Line Library** -> muse/docs/lib/map.md

--(:) map: Orientation and position reporting, broadcast and persistence of places -> map

--:# File and Broadcast Operations for points, trails, and ranges (including features)

--:: map.place(placeString: "(:)" -> Instantiate string as named place, include in named places. -> serial: ":" , index: #: &!

--:: map.read(thisMap: "(:)" -> Reinstantiate places from map file. -> serial: ":" , index: #: &!

--:: map.write(thisMap: ":"?) -> Delete old, write new locally. Default current. -> nil &!

--:- site name? -> Remote operation to report or change site (persistently) after, e.g., porting rover.

--:- join site role -> Set site and join landed turtle to it with specified role.

--:: map.update(serial: "(:)" -> Append received instantiated MU to local map file. -> nil &!

--:# Map File Operations

--:- sync -> Muse Update (MU) broadcast local map to (MQ) registered units.

--:- erase name -> Remove named place, broadcast Muse eXcise (MX).

--:# Referenced through map.op for CLI dispatch

--:: map.erase(name: "(:)" -> Remove named place, overwrite local map file -> remaining: #:

--:: map.get(name: ":" , key: "(:)" -> Get named place local feature value for key. -> value: any? &!

--:: map.gets(name: ":" , key: "(:)" -> Less generic retrieval interface: gets string feature value. -> ":"?

--:: map.put(name: ":" , key: ":" , value: any?) -> Set named place feature, send MU. -> key: ":"? , value: any|true|nil &!

--:: map.puts(name: ":" , key: ":" , value: ":"?) -> Set string feature value, send MU. -> key: ":" , value: ":"|true &!

--:# Report direction turtle is facing (requires GPS in game)

--:: map.testFacing(dx: #; dz: #(:) -> Find orientation using position changes for non-zero movement. -> facing: ":" & !

--:- fix trail? -> Set and report GPS turtle position for dead reckoning. Optionally begin named trailhead.

--:< Places - Points, Locations, Trails, and Ranges of Maps

--:- point name label trail? -> Add named labeled point, can start trail, MU updated map. (Player situation needs GPS.)

--:+ Optional trail starts turtle movement track ended by call to trail limited by Muse.tracking.limit.

--:: map.set(name: ":", label: ":", x: #, y: #, z: #, f: "😊" -> _Set turtle at created point -> ":")

--:: map.point(name: ":", label: ":", :xyzf:) -> Create, send point update. -> **nil & !**

--:: map.locations(template: :[name: ":", offset: xyz], base: ":", label: ":", top: #😊 -> Add points offset from base. -> **nil**

--:+ Add labelled points using template names and offsets from named base point or top for y-axis.

--:- trail name label -> Include named point at head and (current situation) tail of a new trail, update map.

--:+ Call to **trail** establishes a trail of tracked turtle movements from the head of the trail named and started by **point**.

--:+ It also establishes a trail from the tail of the trail named by **trail** back to the head of the trail.

--:+ Both trails (from the head to the tail of the **trail** and back) share a **Label** as specified in the call to **trail**.

--:+ Turtles can move along trails with calls to **roam.trace**.

--:- range name label point point key? value?? -> Volume by named points, optional key and value for feature.

--:- chart filename ... -> Loads and runs named file in **charts** directory to create named point and associated ranges.

--:+ **While there are conventions (indicated), there's no restriction in what loading and running the file actually does!**

--:+ The function generated by loading the file is applied to the ... parameters following the chart file name.

--:+ This chart file function is expected to create ranges establishing the **chart** and a way to reference those ranges.

--:+ There is nothing to enforce this expectation. The chart file could do (oh, my) pretty much anything.

--:: map.borders(range: place) -> Get range elements -> **borders, features, position, position &!**

--:> borders: Range boundaries -> {east: #:, west: #:, north: #:, south: #:, top: #:, bottom: #:}

--:< **Navigation in Maps: Where Are We, What's Nearby, and Where Are We Heading?**

--:- at -> Report current (dead reckoning) turtle position and facing or player GPS position.

--:- test name, label, x, y, z, facing, key?, value??} -> Force mapped position, optionally feature and value for **point**.

--:- where place? count?? -> Report movement direction, distance to named place (or all) three (or count) closest places.

--:- headings rate? place? count?? -> _Repeated movement report at specified rate (or every G.Muse.rates.headings) seconds).

2025-11-25

--:- near place? span?? -> *Report points within span blocks (or all) of named place (or current player or turtle position).*

--:- view place -> *Report place details including name, label (if any), features and all situations.*

--:# **Command Line Interface**

--:: map.op(commands: ":"[]}) -> *Command Line Interface* -> **report:** ":" &: