--:! {remote: []: (😃 } <- **Functions Library for Remote Procedure Calls** -> muse/docs/lib/remote.md
--😐 remote: *Client and server side support for RPCs and client (player) side support for* `come` *and* `tail`. -> remote, _remote
--:+ **Test functions are provided for out-of-game, no network operation.**

--:# **Server Side Remote Call Operations: Protocols to Receive Muse Calls (MC), Send Muse Responses (MR)**

--:: remote.wait() -> *Setup turtle to repeatedly wait for MC network requests, send MR results.* -> `nil`

--:# **Client Side Remote Call Operations: Protocols to Send Muse Calls (MC), Receive Muse Responses (MR)**

--:: remote.call(server: ":", command: ":", arguments: any[], callback: (😃?) -> *RPC:* -> `any &: &!`
--:+ *Form serialized request table from command string and arguments. Get server ID from server name.*
--:+ *Send request to server, wait for result, return call (default* `remote.return`*) callback function to result.*

--:: remote.returns(results: any[]) -> *Default client side handling of server response: just print results as string.* -> `nil`

--:# **Turtle fetch functions:** `come` **(once) and** `tail` **(repeatedly) to player**
--:+ *Prepare remote call to server turtle by getting player xyz position and forming argument table.*

--:: remote.come(turtle: "😊 -> *Towards GPS player position.* -> `report: ":"`

--:- come -> *Turtle towards GPS player position.*

--:: remote.tail(turtle: ":", __ : "tail", rates: ":"?) -> _Repeatedly towards player position, default rate _G.Muse.rates.tail seconds -> `nil`

--:- tail rate? -> *Turtle every rate seconds towards player.*