net Module

Constants	net.
net.createConnection()	Creates a client.
net.createServer()	Creates a server.
net.server:close()	Closes the server.
net.server:listen()	Listen on port from IP address.
net.socket:close()	Closes socket.
net.socket:connect()	Connect to a remote server.
net.socket:dns()	Provides DNS resolution for a hostname.
net.socket:on()	Register callback functions for specific events.
net.socket:send()	Sends data to server.
net.dns.getdnsserver()	Gets the IP address of the DNS server used to resolve hostnames.
net.dns.resolve()	Resolve a hostname to an IP address.
net.dns.setdnsserver()	Sets the IP of the DNS server used to resolve hostnames.

Constants

`net.TCP`, `net.UDP`

net.createConnection()

Creates a client.

Syntax

`net.createConnection(type, secure)`

Parameters

- `type` `net.TCP` or `net.UDP`
- `secure` 1 for encrypted, 0 for plain

Returns

net.socket sub module

Example

```
net.createConnection(net.UDP, 0)
```

See also

`net.createServer()`

net.createServer()

Creates a server.

Syntax

`net.createServer(type, timeout)`

Parameters

- `type` `net.TCP` or `net.UDP`
- `timeout` for a TCP server timeout is 1~28'800 seconds (for an inactive client to be disconnected)

Returns

net.server sub module

Example

```
net.createServer(net.TCP, 30) -- 30s timeout
```

See also

`net.createConnection()`

net.server Module

net.server:close()

Closes the server.

Syntax

`net.server.close()`

Parameters

none

Returns

`nil`

Example

```lua
-- creates a server
sv = net.createServer(net.TCP, 30)
-- closes the server
sv:close()
```

See also

`net.createServer()`

net.server:listen()

Listen on port from IP address.

Syntax

`net.server.listen(port,[ip],function(net.socket))`

Parameters

- `port` port number
- `ip` IP address string, can be omitted
- `function(net.socket)` callback function, pass to caller function as param if a connection is created successfully

Returns

`nil`

Example

```lua
 -- 30s time out for a inactive client
sv = net.createServer(net.TCP, 30)
-- server listens on 80, if data received, print data to console and send '
sv:listen(80, function(c)
    c:on("receive", function(c, pl)
        print(pl)
    end)
    c:send("hello world")
end)
```

See also

`net.createServer()`

net.socket Module

net.socket:close()

Closes socket.

Syntax

`close()`

Parameters

none

Returns

`nil`

See also

`net.createServer()`

net.socket:connect()

Connect to a remote server.

Syntax

`connect(port, ip|domain)`

Parameters

- `port` port number
- `ip` IP address or domain name string

Returns

`nil`

See also

`net.socket:on()`

net.socket:dns()

Provides DNS resolution for a hostname.

Syntax

`dns(domain, function(net.socket, ip))`

Parameters

- `domain` domain name
- `function(net.socket, ip)` callback function. The first parameter is the socket, the second parameter is the IP address as a string.

Returns

`nil`

Example

```
sk = net.createConnection(net.TCP, 0)
sk:dns("www.nodemcu.com", function(conn, ip) print(ip) end)
sk = nil
```

See also

`net.createServer()`

net.socket:on()

Register callback functions for specific events.

Syntax

`on(event, function())`

Parameters

- `event` string, which can be "connection", "reconnection", "disconnection", "receive" or "sent"
- `function(net.socket[, string])` callback function. The first parameter is the socket. If event is "receive", the second parameter is the received data as string.

Returns

`nil`

Example

```
sk = net.createConnection(net.TCP, 0)
sk:on("receive", function(sck, c) print(c) end )
sk:connect(80,"192.168.0.66")
sk:on("connection", function(sck,c)
    -- Wait for connection before sending.
    sk:send("GET / HTTP/1.1\r\nHost: 192.168.0.66\r\nConnection: keep-alive
end)
```

See also

`net.createServer()`

net.socket:send()

Sends data to server.

Syntax

`send(string, function(sent))`

Parameters

- `string` data in string which will be sent to server
- `function(sent)` callback function for sending string

Returns

`nil`

Note

Multiple consecutive `send()` calls aren't guaranteed to work (and often don't) as network requests are treated as separate tasks by the SDK. Instead, subscribe to the "sent" event on the socket and send additional data (or close) in that callback. See #730 for an example and explanation.

See also

`net.socket:on()`

net.dns Module

net.dns.getdnsserver()

Gets the IP address of the DNS server used to resolve hostnames.

Syntax

`net.dns.getdnsserver(dns_index)`

Parameters

dns_index which DNS server to get (range 0~1)

Returns

IP address (string) of DNS server

Example

```
print(net.dns.getdnsserver(0)) -- 208.67.222.222
print(net.dns.getdnsserver(1)) -- nil

net.dns.setdnsserver("8.8.8.8", 0)
net.dns.setdnsserver("192.168.1.252", 1)

print(net.dns.getdnsserver(0)) -- 8.8.8.8
print(net.dns.getdnsserver(1)) -- 192.168.1.252
```

See also

`net.dns:setdnsserver()`

net.dns.resolve()

Resolve a hostname to an IP address. Doesn't require a socket like `net.socket.dns()`.

Syntax

`net.dns.resolve(host, function(ip))`

Parameters

- `host` hostname to resolve
- `function(sk, ip)` callback called when the name was resolved. Don't use `sk`, it's a socket used internally to resolve the hostname.

Returns

`nil`

Example

```
net.dns.resolve("www.google.com", function(sk, ip)
    if (ip == nil) then print("DNS fail!") else print(ip) end
end)
```

See also

`net.socket:dns()`

net.dns.setdnsserver()

Sets the IP of the DNS server used to resolve hostnames. Default: resolver1.opendns.com (208.67.222.222). You can specify up to 2 DNS servers.

Syntax

`net.dns.setdnsserver(dns_ip_addr, dns_index)`

Parameters

- `dns_ip_addr` IP address of a DNS server
- `dns_index` which DNS server to set (range 0~1). Hence, it supports max. 2 servers.

Returns

`nil`

See also

`net.dns:getdnsserver()`