NODE.JS - DNS MODULE

Node.js **dns** module is used to do actual DNS lookup as well as to use underlying operating system name resolution functionalities. This module provides an asynchronous network wrapper and can be imported using following syntax.

```
var dns = require("dns")
```

Methods

S.N.	Method & Description
1	**dns.lookup***hostname*[, *options*], *callback* Resolves a hostname *e. g.*'*google. com*' into the first found A *IPv*4 or AAAA *IPv*6 record. options can be an object or integer. If options is not provided, then IP v4 and v6 addresses are both valid. If options is an integer, then it must be 4 or 6.
2	**dns.lookupService***address, port, callback* Resolves the given address and port into a hostname and service using getnameinfo.
3	**dns.resolve***hostname*[, *rrtype*], *callback* Resolves a hostname *e. g.*'*google. com*' into an array of the record types specified by rrtype.
4	**dns.resolve4***hostname, callback* The same as dns.resolve, but only for IPv4 queries *Arecords*. addresses is an array of IPv4 addresses *e. g.* ['74.125.79.104','74.125.79.105','74.125.79.106'].
5	**dns.resolve6***hostname, callback* The same as dns.resolve4 except for IPv6 queries *anAAAAquery*.
6	**dns.resolveMx***hostname, callback* The same as dns.resolve, but only for mail exchange queries *MXrecords*.
7	**dns.resolveTxt***hostname, callback* The same as dns.resolve, but only for text queries *TXTrecords*. addresses is an 2-d array of the text records available for hostname *e. g.* , [['*v* = *spf*1*ip*4: 0.0.0.0',' *all*']]. Each sub-array contains TXT chunks of one record. Depending on the use case, the could be either joined together or treated separately.
8	**dns.resolveSrv***hostname, callback* The same as dns.resolve, but only for service records *SRVrecords*. addresses is an array of the SRV records available for hostname. Properties of SRV records are priority, weight, port, and name *e. g.* , ['*priority*': 10,'*weight*': 5,'*port*': 21223,'*name*':'*service. example. com*', . . .].
9	**dns.resolveSoa***hostname, callback* The same as dns.resolve, but only for start of authority record queries *SOArecord*.
10	**dns.resolveNs***hostname, callback* The same as dns.resolve, but only for name server records *NSrecords*. addresses is an array of the name server records available for hostname *e. g.* , ['*ns*1.*example. com*','*ns*2.*example. com*'].
11	**dns.resolveCname***hostname, callback* The same as dns.resolve, but only for canonical name records *CNAMErecords*. addresses is an array of the canonical name records available for hostname *e. g.* , ['*bar. example. com*'].
12	**dns.reverse***ip, callback* Reverse resolves an ip address to an array of hostnames.
13	**dns.getServers**

	Returns an array of IP addresses as strings that are currently being used for resolution.
14	**dns.setServers***servers* Given an array of IP addresses as strings, set them as the servers to use for resolving.

rrtypes

Following is the list of valid rrtypes used by dns.resolve method

- **A** - IPV4 addresses, default
- **AAAA** - IPV6 addresses
- **MX** - mail exchange records
- **TXT** - text records
- **SRV** - SRV records
- **PTR** - used for reverse IP lookups
- **NS** - name server records
- **CNAME** - canonical name records
- **SOA** - start of authority record

Error Codes

Each DNS query can return one of the following error codes:

- **dns.NODATA** - DNS server returned answer with no data.
- **dns.FORMERR** - DNS server claims query was misformatted.
- **dns.SERVFAIL** - DNS server returned general failure.
- **dns.NOTFOUND** - Domain name not found.
- **dns.NOTIMP** - DNS server does not implement requested operation.
- **dns.REFUSED** - DNS server refused query.
- **dns.BADQUERY** - Misformatted DNS query.
- **dns.BADNAME** - Misformatted hostname.
- **dns.BADFAMILY** - Unsupported address family.
- **dns.BADRESP** - Misformatted DNS reply.
- **dns.CONNREFUSED** - Could not contact DNS servers.
- **dns.TIMEOUT** - Timeout while contacting DNS servers.
- **dns.EOF** - End of file.
- **dns.FILE** - Error reading file.
- **dns.NOMEM** - Out of memory.
- **dns.DESTRUCTION** - Channel is being destroyed.
- **dns.BADSTR** - Misformatted string.
- **dns.BADFLAGS** - Illegal flags specified.

- **dns.NONAME** - Given hostname is not numeric.
- **dns.BADHINTS** - Illegal hints flags specified.
- **dns.NOTINITIALIZED** - c-ares library initialization not yet performed.
- **dns.LOADIPHLPAPI** - Error loading iphlpapi.dll.
- **dns.ADDRGETNETWORKPARAMS** - Could not find GetNetworkParams function.
- **dns.CANCELLED** - DNS query cancelled.

Example

Create a js file named main.js having the following code:

```
var dns = require('dns');

dns.lookup('www.google.com', function onLookup(err, address, family) {
   console.log('address:', address);
   dns.reverse(address, function (err, hostnames) {
   if (err) {
      console.log(err.stack);
   }

   console.log('reverse for ' + address + ': ' + JSON.stringify(hostnames));
});
});
```

Now run the main.js to see the result:

```
$ node main.js
```

Verify the Output.

```
address: 173.194.46.83
reverse for 173.194.46.83: ["ord08s11-in-f19.1e100.net"]
```