The dns module in Node.js provides functions for performing DNS (Domain Name System) lookups, allowing you to resolve domain names into IP addresses and other related information. It can also be used to reverse lookup IP addresses into domain names.

There are two key methods in the dns module:

* **dns.lookup()**: This uses the local operating system's DNS resolver and is synchronous.
* **dns.resolve()**: This bypasses the local DNS cache and performs a network DNS query.

**Using the dns module**

1. **Importing the dns module**
2. **Performing DNS lookups**
3. **Resolving different types of DNS records**

Let's create a simple DNS module program in Node.js.

**Step 1: Import the dns module**

You first need to import the built-in dns module in Node.js:

javascript

Copy code

const dns = require('dns');

**Step 2: DNS Lookup (IP Address Lookup)**

The dns.lookup() function resolves a domain name to an IP address using the operating system's DNS facilities.

javascript

Copy code

const dns = require('dns');

// Perform DNS lookup for a domain

dns.lookup('example.com', (err, address, family) => {

if (err) {

console.error('Error during DNS lookup:', err);

} else {

console.log(`Address: ${address}, Family: IPv${family}`);

}

});

In this example:

* dns.lookup('example.com', ...) resolves the domain name example.com into an IP address.
* The callback returns the address (the IP address) and family (the IP version: 4 for IPv4 or 6 for IPv6).

**Step 3: DNS Record Resolution (dns.resolve())**

The dns.resolve() function allows you to perform a full DNS resolution for specific record types like A, AAAA, MX, NS, CNAME, TXT, etc.

**Example: Resolve A Records (IPv4 Addresses)**

javascript

Copy code

const dns = require('dns');

// Resolve A records for a domain (IPv4 addresses)

dns.resolve('example.com', 'A', (err, addresses) => {

if (err) {

console.error('Error resolving A records:', err);

} else {

console.log(`IPv4 addresses: ${addresses}`);

}

});

In this example:

* We resolve the **A** record, which contains the IPv4 addresses for example.com.

**Example: Resolve MX Records (Mail Exchange Servers)**

javascript

Copy code

const dns = require('dns');

// Resolve MX records (Mail Exchange) for a domain

dns.resolve('example.com', 'MX', (err, addresses) => {

if (err) {

console.error('Error resolving MX records:', err);

} else {

console.log('MX records:', addresses);

}

});

In this example:

* We resolve the **MX** record for example.com. MX records specify the mail servers responsible for receiving email.

**Step 4: Reverse Lookup (IP Address to Domain Name)**

The dns.reverse() function is used to reverse resolve an IP address back to a domain name.

javascript

Copy code

const dns = require('dns');

// Perform a reverse DNS lookup on an IP address

dns.reverse('93.184.216.34', (err, hostnames) => {

if (err) {

console.error('Error during reverse lookup:', err);

} else {

console.log(`Reverse lookup results: ${hostnames}`);

}

});

In this example:

* We reverse resolve the IP address 93.184.216.34 to a domain name. The callback returns an array of hostnames associated with that IP.

**Step 5: Resolving Multiple DNS Record Types**

You can resolve various types of DNS records other than A and MX. Some of the common DNS record types you can resolve are:

* **A**: IPv4 addresses.
* **AAAA**: IPv6 addresses.
* **MX**: Mail exchange records.
* **NS**: Name server records.
* **TXT**: Text records.
* **CNAME**: Canonical name records (aliases).
* **SOA**: Start of Authority records.

Example: Resolve multiple DNS record types

javascript

Copy code

const dns = require('dns');

// Resolve multiple DNS records (NS, TXT, CNAME)

dns.resolve('example.com', 'NS', (err, addresses) => {

if (err) {

console.error('Error resolving NS records:', err);

} else {

console.log('Name servers:', addresses);

}

});

dns.resolve('example.com', 'TXT', (err, records) => {

if (err) {

console.error('Error resolving TXT records:', err);

} else {

console.log('TXT records:', records);

}

});

dns.resolve('example.com', 'CNAME', (err, records) => {

if (err) {

console.error('Error resolving CNAME records:', err);

} else {

console.log('CNAME records:', records);

}

});

**Example: Full DNS Module Program**

Here's a complete program that allows you to perform different DNS operations interactively:

javascript

Copy code

const dns = require('dns');

// Lookup IP address using dns.lookup()

dns.lookup('example.com', (err, address, family) => {

if (err) {

console.error('Error during DNS lookup:', err);

} else {

console.log(`DNS Lookup - Address: ${address}, Family: IPv${family}`);

}

});

// Resolve A record using dns.resolve()

dns.resolve('example.com', 'A', (err, addresses) => {

if (err) {

console.error('Error resolving A records:', err);

} else {

console.log(`DNS Resolve - A records: ${addresses}`);

}

});

// Resolve MX record using dns.resolve()

dns.resolve('example.com', 'MX', (err, addresses) => {

if (err) {

console.error('Error resolving MX records:', err);

} else {

console.log('DNS Resolve - MX records:', addresses);

}

});

// Reverse lookup using dns.reverse()

dns.reverse('93.184.216.34', (err, hostnames) => {

if (err) {

console.error('Error during reverse lookup:', err);

} else {

console.log(`Reverse Lookup - Hostnames: ${hostnames}`);

}

});