In Node.js, the url module provides utilities for URL resolution, parsing, and formatting. It allows you to work with URLs and extract or modify different parts of a URL string. The url module can be imported with const url = require('url'); in CommonJS, or with import { URL } from 'url'; in ES Modules.

Here’s an overview of how to use the url module with examples:

### **1. Creating a URL Object**

In Node.js, you can create a URL object from a URL string using the URL constructor.

javascript

Copy code

const { URL } = require('url');

const myUrl = new URL('https://example.com:8080/pathname/index.html?name=John&age=30#section');

console.log(myUrl);

This will output a URL object with various properties, such as:

* **href**: Full URL as a string.
* **protocol**: URL protocol (e.g., https:).
* **host**: Hostname with port (e.g., example.com:8080).
* **hostname**: Hostname without port (e.g., example.com).
* **port**: Port number (e.g., 8080).
* **pathname**: Path after the host (e.g., /pathname/index.html).
* **search**: Query string (e.g., ?name=John&age=30).
* **hash**: Fragment identifier (e.g., #section).
* **origin**: The origin of the URL (e.g., https://example.com:8080).

### **2. Accessing URL Components**

Each part of the URL can be accessed as properties on the URL object:

javascript

Copy code

console.log(myUrl.href); // https://example.com:8080/pathname/index.html?name=John&age=30#section

console.log(myUrl.protocol); // https:

console.log(myUrl.host); // example.com:8080

console.log(myUrl.hostname); // example.com

console.log(myUrl.port); // 8080

console.log(myUrl.pathname); // /pathname/index.html

console.log(myUrl.search); // ?name=John&age=30

console.log(myUrl.hash); // #section

console.log(myUrl.origin); // https://example.com:8080

### **3. Working with Query Parameters**

The URLSearchParams object allows you to work with query parameters.

* **Accessing Query Parameters**

javascript

Copy code

console.log(myUrl.searchParams.get('name')); // John

console.log(myUrl.searchParams.get('age')); // 30

* **Adding, Deleting, and Modifying Parameters**

javascript

Copy code

myUrl.searchParams.append('city', 'New York');

console.log(myUrl.searchParams.toString()); // name=John&age=30&city=New+York

myUrl.searchParams.set('name', 'Doe');

console.log(myUrl.searchParams.toString()); // name=Doe&age=30&city=New+York

myUrl.searchParams.delete('age');

console.log(myUrl.searchParams.toString()); // name=Doe&city=New+York

### **4. Serializing a URL**

The URL object automatically serializes when accessed as a string.

javascript

Copy code

console.log(myUrl.toString()); // https://example.com:8080/pathname/index.html?name=Doe&city=New+York#section

console.log(myUrl.href); // Equivalent to myUrl.toString()

### **5. Relative and Absolute URLs**

The URL constructor can also resolve relative URLs based on a base URL.

javascript

Copy code

const baseUrl = new URL('https://example.com/path/');

const relativeUrl = new URL('/subpath', baseUrl);

console.log(relativeUrl.href); // https://example.com/subpath

### **6. Legacy URL Parsing (url.parse and url.format)**

The older url.parse() and url.format() methods are still available, but it’s recommended to use the URL constructor in new code.

* **url.parse()**: Parses a URL string into an object.

javascript

Copy code

const url = require('url');

const parsedUrl = url.parse('https://example.com:8080/pathname?name=John#section');

console.log(parsedUrl);

* **url.format()**: Serializes a URL object back into a string.

javascript

Copy code

const formattedUrl = url.format(parsedUrl);

console.log(formattedUrl); // https://example.com:8080/pathname?name=John#section

### **Example: Complete URL Parsing and Manipulation**

javascript

Copy code

const { URL } = require('url');

const myUrl = new URL('https://example.com:8080/pathname/index.html?name=John&age=30#section');

// Access URL parts

console.log('Protocol:', myUrl.protocol); // https:

console.log('Host:', myUrl.host); // example.com:8080

console.log('Pathname:', myUrl.pathname); // /pathname/index.html

console.log('Query:', myUrl.search); // ?name=John&age=30

console.log('Hash:', myUrl.hash); // #section

// Manipulate query parameters

myUrl.searchParams.append('city', 'New York');

console.log('New Query:', myUrl.searchParams.toString()); // name=John&age=30&city=New+York

// Serialize URL back to a string

console.log('Modified URL:', myUrl.toString()); // https://example.com:8080/pathname/index.html?name=John&age=30&city=New+York#section

### **Summary**

| **Method** | **Description** |
| --- | --- |
| new URL(url) | Creates a URL object from a URL string |
| .href | Full URL as a string |
| .protocol | Protocol, like http: or https: |
| .host | Hostname and port |
| .hostname | Hostname without port |
| .port | Port number |
| .pathname | Path following the host |
| .search | Query string, including ? |
| .searchParams | URLSearchParams object for queries |
| .hash | Fragment identifier |
| .origin | Protocol, hostname, and port |
| url.parse() (legacy) | Parses URL string into an object |
| url.format() (legacy) | Serializes URL object into a string |

The url module simplifies working with URLs, enabling easy parsing, modifying, and formatting URLs in Node.js applications.