**Week - 4**

**Task-1**

**Aim: URL Parsing and Manipulation:**

* Write a program that accepts a URL as user input and uses the url module to parse it.Display the protocol, host, path, and query parameters separately.
* Implement a function that takes a base URL and a relative path as input, and uses the url module to resolve and display the absolute URL.

**Description:**

* URL parsing and manipulation in Node.js is facilitated by the built-in `url` module. This module provides several functions to work with URLs, allowing developers to parse, construct, and manipulate URLs easily.

1. **Parsing URLs:**

The `url.parse()` function is used to parse a URL string and extract its different components such as the protocol, host, path, query parameters, etc. It returns an object containing these components.

1. **Constructing URLs:**

The `url.format()` function is used to construct a URL string from its components. It takes an object containing the components and returns a formatted URL string.

1. **URL Class:**

Node.js also provides a `URL` class (introduced in Node.js v7.5.0) for working with URLs. It offers an improved and more modern API compared to the legacy `url.parse()` and `url.format()` functions.

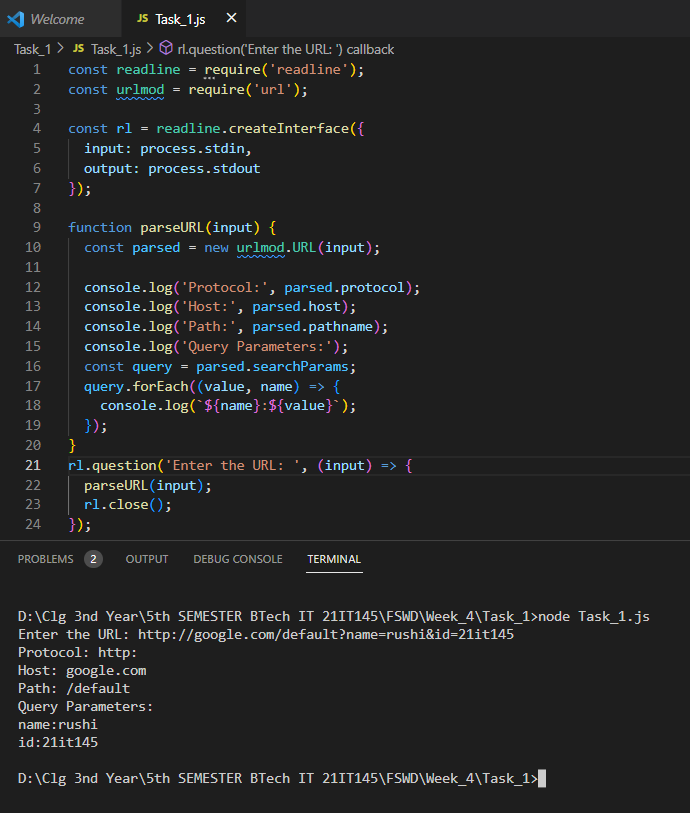
1. **Resolving URLs:**

The `url.resolve()` function is used to resolve a relative URL against a base URL and construct the absolute URL. This is useful when you want to navigate from one URL to another.

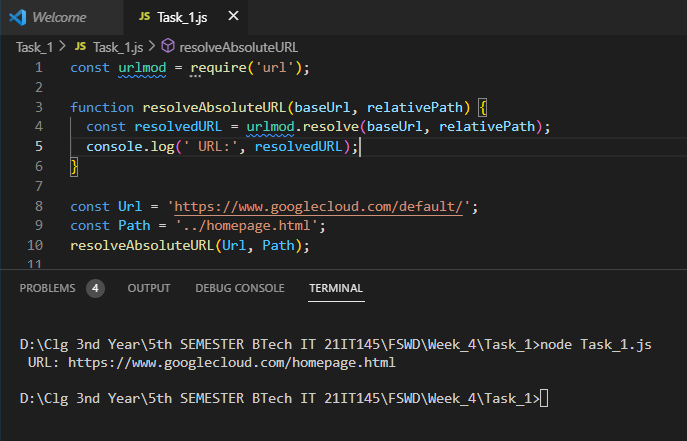
With these capabilities provided by the `url` module, developers can easily work with URLs in Node.js, parse them into their components, construct new URLs, and resolve relative URLs to their absolute forms.

**Source Code & Output:**

**Task1.1**

****

**Task 1.2**

****

**Task-2**

**Aim:**  **Query String Operation:**

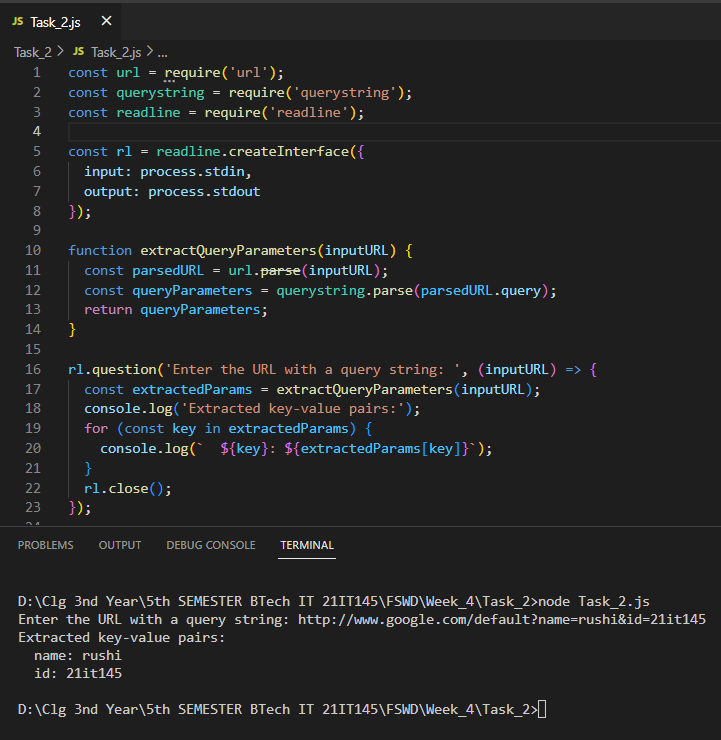
* Write a Node.js program that takes a URL with a query string as input and extracts the key-value pairs from the query string using the querystring module. The program should display the extracted key-value pairs as output.

**Description:**

* In Node.js, the "querystring" module provides utilities for working with query strings. Query strings are commonly used in URLs to pass data as key-value pairs, typically for parameters in HTTP requests. The "querystring" module allows you to parse and stringify query strings, making it easy to work with the data they contain.

1. **Parsing a Query String:** The `querystring.parse()` function is used to parse a query string and convert it into a JavaScript object.
2. **Stringifying an Object to a Query String:** The `querystring.stringify()` function is used to convert a JavaScript object into a query string.
3. **Encoding and Decoding:** The `querystring` module also provides the `querystring.escape()` and `querystring.unescape()` functions for encoding and decoding special characters in a query string.

**Source Code & Output:**

****

**Task-3**

**Aim:** **Path Operations:**

* Create a program that accepts two file paths as input and uses the path module to determine if they refer to the same file.
* Implement a function that accepts a file path as input and uses the path module to extract the file extension. Display the extracted extension to the user.

**Description:**

* In Node.js, the `path` module provides utilities for working with file and directory paths. It is built-in and does not require any installation.
* Here are some of the common operations provided by the `path` module in Node.js:

1. **Joining Paths:**

The `path.join()` function is used to join multiple path segments into a single normalized path.

1. **Normalizing Paths:**

The `path.normalize()` function is used to normalize a given path by resolving '..' and '.' segments.

1. **Getting the Directory Name:**

The `path.dirname()` function is used to get the directory name from a file path.

1. **Getting the File Extension:**

The `path.extname()` function is used to get the file extension from a file path.

1. **Getting the Base Name:**

The `path.basename()` function is used to get the base name (last portion) of a file path.

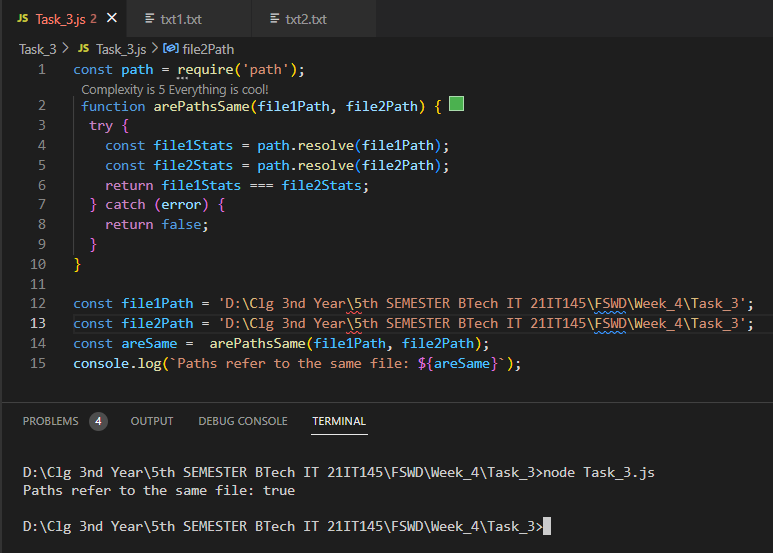
1. **Getting the File Name without Extension:**

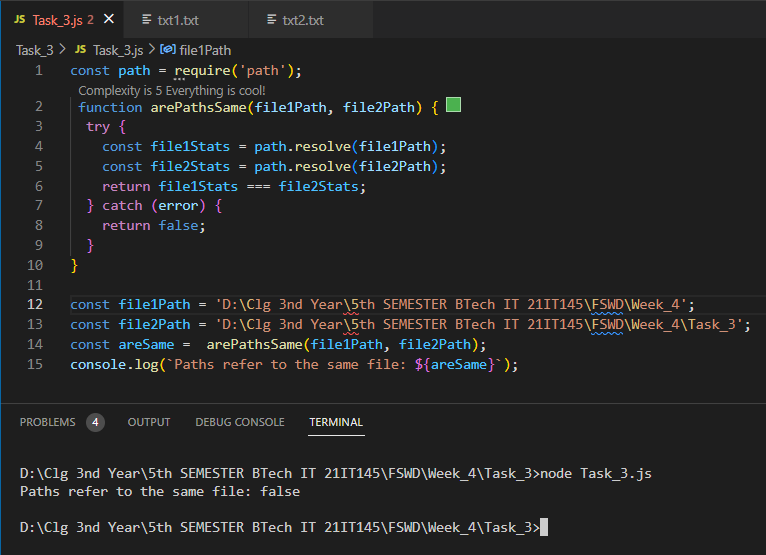
The `path.parse()` function is used to parse a file path and return an object containing its different components.

These are some of the useful operations provided by the `path` module in Node.js, making it easier to work with file and directory paths in a platform-independent manner.

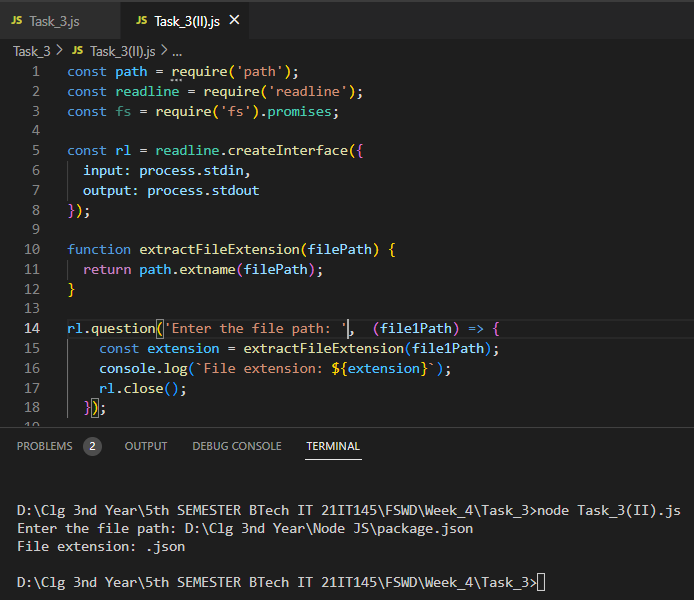
**Source Code & Output:**

**Task 3.1js**

****

****

**Task 3.2js**

****

**Task-4**

**Aim:** **File Paths and Operations:**

* Implement a program that accepts a file path as input and uses the path module to extract the directory name and base name. Display the extracted values separately.
* Write a function that uses the fs module to check if a given file path exists. Display a success message if the file exists, or an error message if it does not.

**Description:**

* File path and operations in Node.js involve working with file and directory paths, reading and writing files, checking file existence, and performing various file-related operations.
* Below are some of the common file path and operations in Node.js:

1. **File Path Operations:**

- `path.join()`: Joins multiple path segments into a single normalized path.

- `path.normalize()`: Normalizes a given path by resolving '..' and '.' segments.

- `path.dirname()`: Gets the directory name from a file path.

- `path.basename()`: Gets the base name (last portion) of a file path.

- `path.extname()`: Gets the file extension from a file path.

- `path.parse()`: Parses a file path and returns an object containing different components.

1. **Checking File Existence:**

- `fs.access()`: Checks if a file or directory exists and has the specified permissions.

1. **Reading and Writing Files:**

- `fs.readFile()`: Reads the content of a file asynchronously.

- `fs.writeFile()`: Writes data to file asynchronously, overwriting the existing file content.

- `fs.appendFile()`: Appends data to file asynchronously, preserving existing file content.

1. **Creating and Removing Directories:**

- `fs.mkdir()`: Creates a directory asynchronously.

- `fs.rmdir()`: Removes a directory asynchronously.

- `fs.readdir()`: Reads the contents of a directory asynchronously.

1. **File Stats:**

- `fs.stat()`: Gets the file stats asynchronously, including file size, creation time, etc.

1. **Renaming and Deleting Files:**

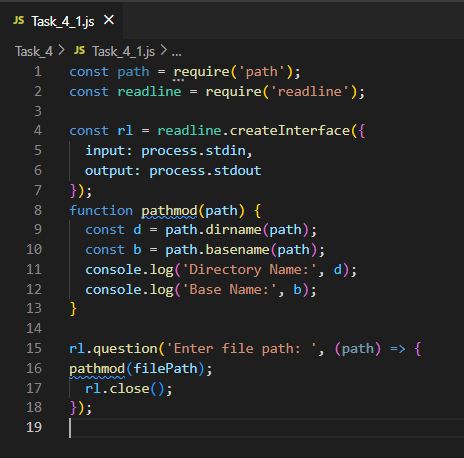
- `fs.rename()`: Renames a file or moves it to another location asynchronously.

- `fs.unlink()`: Deletes a file asynchronously.

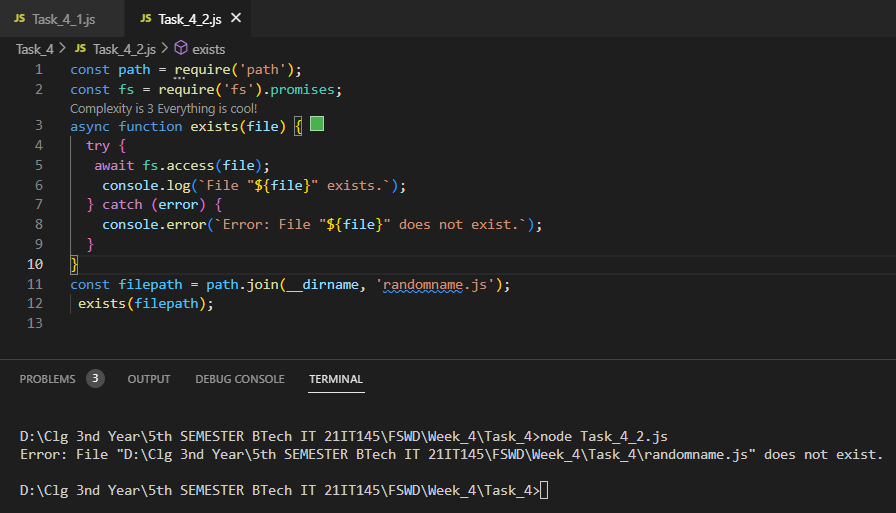
These are some of the basic file path and operations available in Node.js through the built-in `path` and `fs` modules.

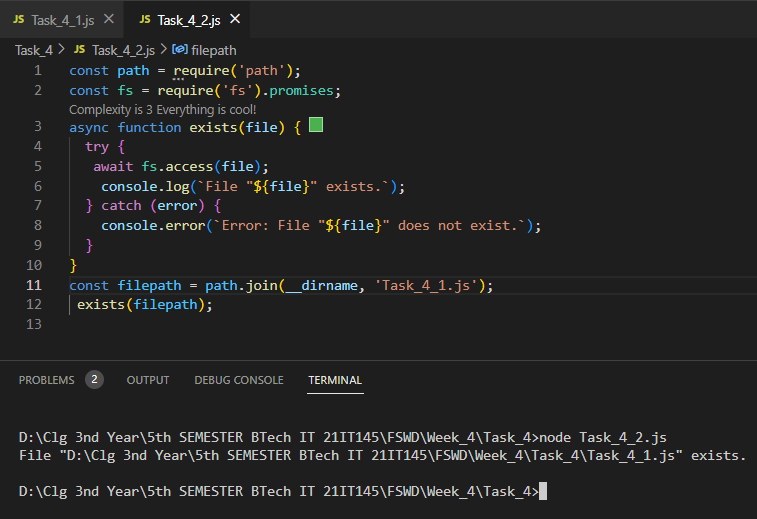
**Source Code & Output:**

**Task 4.1js**

****

**Task 4.2js**

****

****

**Learning Outcome:**

From this practical, I learnt about various concepts of NodeJS which helped me to understand the basics of NodeJS and also practically performed it.

* Demonstrate the use of JavaScript to fulfill the essentials of front-end development To back-end development.
* Apply a deep knowledge of MVC(ModelViewController) architecture,making the development process easier and faster using open-source technologies.