

## 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.

**Source Code:**

```
const readline = require('readline');
const urlModule = require('url');

// Create an interface to read user input
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout
});

// Function to parse and display URL components
function parseURL(url) {
  const parsedURL = urlModule.parse(url, true);

  console.log('\nURL Components:');
  console.log('Protocol:', parsedURL.protocol);
  console.log('Host:', parsedURL.host);
  console.log('Path:', parsedURL.pathname);

  console.log('\nQuery Parameters:');
  const queryParameters = parsedURL.query;
  for (const key in queryParameters) {
    console.log(key + ':', queryParameters[key]);
  }
}

// Prompt user for URL input
rl.question('Enter a URL: ', (url) => {
  parseURL(url);
  rl.close();
});
```

**Output:**

```
C:\Users\Mohit\FSWD\Practical_4>node 1.js
Enter a URL: http://www.mohitwebsite.com/path/to/resource?mahil=value1&mahi2=value2

URL Components:
Protocol: http:
Host: www.mohitwebsite.com
Path: /path/to/resource

Query Parameters:
mahil: value1
mahi2: value2
```

**Aim:**

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.

**Source Code:**

```
const readline = require('readline');
const urlModule = require('url');
const rl = readline.createInterface({
  input: process.stdin,
  output: process.stdout,
});
function resolveAbsoluteURL(baseURL, relativePath) {
  try {
    const absoluteURL = new URL(relativePath, baseURL);
    return absoluteURL.href;
  } catch (err) {
    return null; // Return null if there's an error in resolving the URL
  }
}
rl.question('Please enter the base URL: ', (baseURL) => {
  rl.question('Please enter the relative path: ', (relativePath) => {
    const absoluteURL = resolveAbsoluteURL(baseURL, relativePath);
    if (absoluteURL) {
      console.log('Absolute URL:', absoluteURL);
    } else {
      console.error('Invalid input. Unable to resolve the absolute URL.');
```

**Output:**

```
C:\Users\Mohit\FSWD\Practical_4>node 1_1.js
Please enter the base URL: https://www.example.com/path/to/
Please enter the relative path: ../resource?param1=value1&param2=value2
Absolute URL: https://www.example.com/path//resource?param1=value1&param2=value2
```

**Theoretical Background:**

URL parsing and manipulation are fundamental concepts in web development and network programming. They enable developers to interact with resources on the web, build dynamic URLs for APIs, and navigate through web applications efficiently. Libraries and built-in tools simplify these tasks, ensuring proper handling of URLs and avoiding common pitfalls like URL injection and security vulnerabilities.

## 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.

### Source Code:

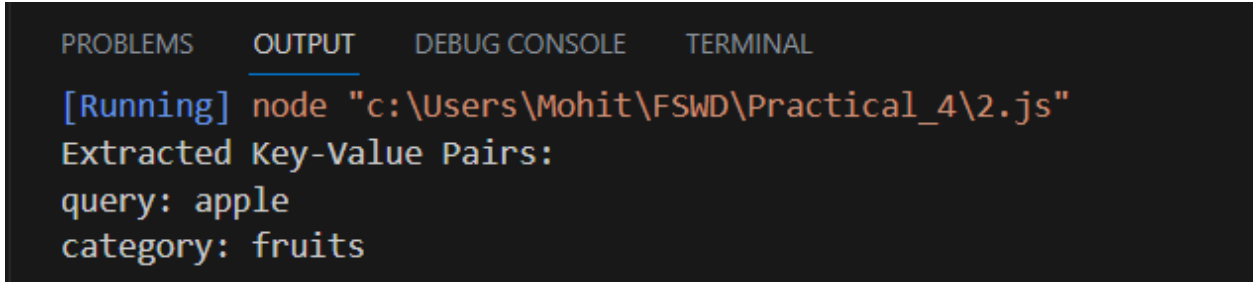
```
const url = require('url');
const querystring = require('querystring');

function extractKeyValuePairsFromURL(urlString) {
  const parsedURL = url.parse(urlString);
  const queryObject = querystring.parse(parsedURL.query);

  console.log('Extracted Key-Value Pairs:');
  for (const key in queryObject) {
    console.log(`${key}: ${queryObject[key]}`);
  }
  return queryObject;
}

const inputURL = 'https://www.example.com/search?query=apple&category=fruits';
extractKeyValuePairsFromURL(inputURL);
```

### Output:



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
[Running] node "c:\Users\Mohit\FSWD\Practical_4\2.js"
Extracted Key-Value Pairs:
query: apple
category: fruits
```

### Theoretical Background:

In the context of web development, a "query string operation" typically refers to the manipulation or extraction of parameters from a URL's query string. The query string is the part of the URL that follows the question mark (?) and contains key-value pairs separated by ampersands (&).

## 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.

### Source Code:

```
const fs = require('fs');
const path = require('path');

function areFilesSame(filePath1, filePath2) {
  // Resolve the absolute paths of the input files
  const absolutePath1 = path.resolve(filePath1);
  const absolutePath2 = path.resolve(filePath2);

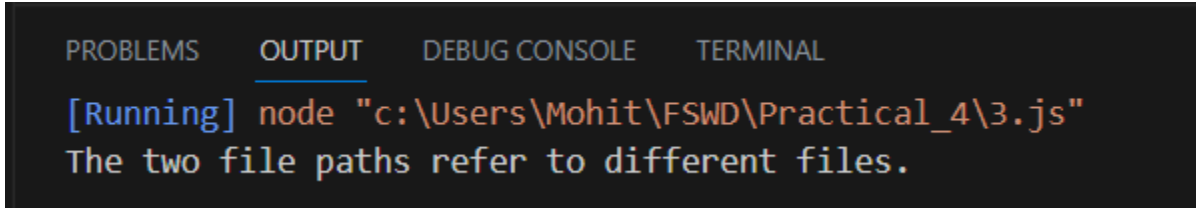
  // Check if the files exist
  if (!fs.existsSync(absolutePath1) || !fs.existsSync(absolutePath2)) {
    return false;
  }

  // Compare the file paths
  return absolutePath1 === absolutePath2;
}

// Example usage
const file1 = 'D:\file1.txt';
const file2 = 'D:\file2.txt';

if (areFilesSame(file1, file2)) {
  console.log('The two file paths refer to the same file.');
```

### Output :



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
[Running] node "c:\Users\Mohit\FSWD\Practical_4\3.js"
The two file paths refer to different files.
```

**Aim:** 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.

**Source Code:**

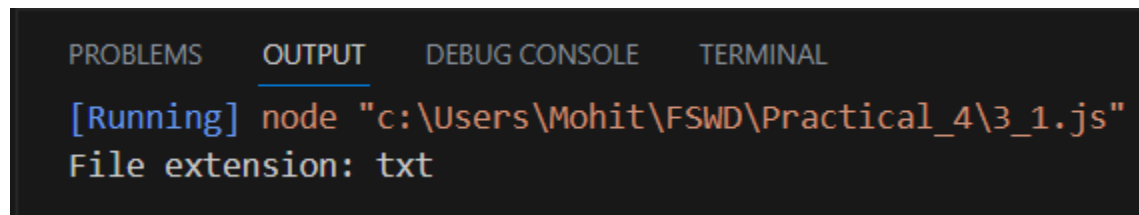
```
const path = require('path');

function getFileExtension(filePath) {
  // Get the file extension using the path module
  const extension = path.extname(filePath);

  // Remove the dot from the extension
  return extension.slice(1);
}

// Example usage
const filePath = '/path/to/file.txt';
const extension = getFileExtension(filePath);
console.log('File extension:', extension);
```

**Output :**



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
[Running] node "c:\Users\Mohit\FSWD\Practical_4\3_1.js"
File extension: txt
```

**Theoretical Background :**

Path operations in the context of web development and file systems involve manipulating file paths to perform various tasks. 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.

## 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.

### Source Code:

```
const path = require('path');

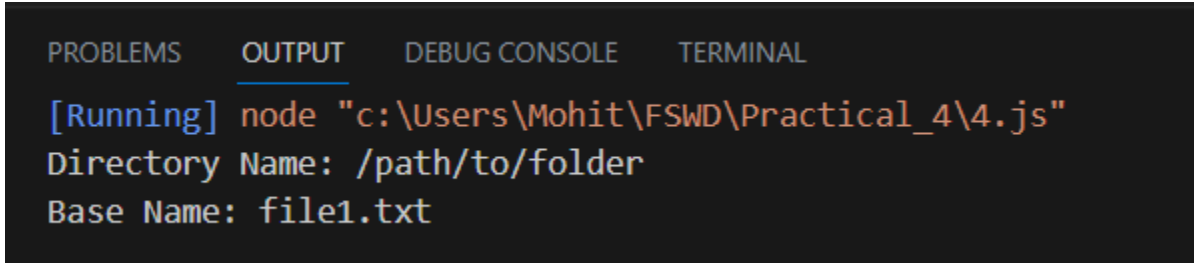
function extractDirectoryAndBaseName(filePath) {
  // Get the directory name and base name using the path module
  const directoryName = path.dirname(filePath);
  const baseName = path.basename(filePath);

  return {
    directoryName,
    baseName,
  };
}

// Example usage
const filePath = '/path/to/folder/file1.txt';
const { directoryName, baseName } = extractDirectoryAndBaseName(filePath);

console.log('Directory Name:', directoryName);
console.log('Base Name:', baseName);
```

### Output :



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
[Running] node "c:\Users\Mohit\FSWD\Practical_4\4.js"
Directory Name: /path/to/folder
Base Name: file1.txt
```



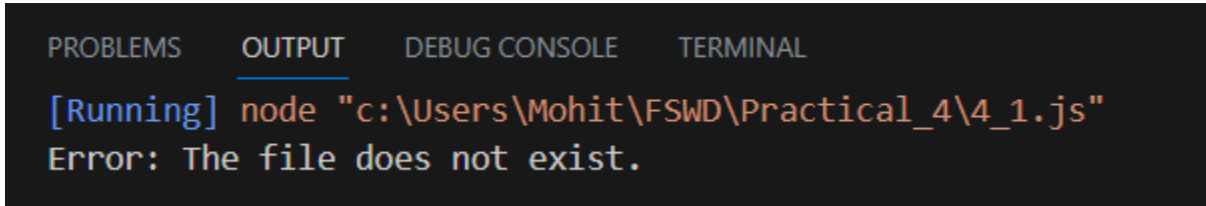
**Aim:** Write a function that uses the fs module to check if a given file path exists. Display a success message if the file exists, or an error message if it doesn't.

**Source Code:**

```
const fs = require('fs');

function checkFileExistsAsync(filePath) {
  // Check if the file exists asynchronously using fs.access()
  fs.access(filePath, fs.constants.F_OK, (err) => {
    if (err) {
      console.log('Error: The file does not exist.');
```

**Output :**



```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
[Running] node "c:\Users\Mohit\FSWD\Practical_4\4_1.js"
Error: The file does not exist.
```

**Theoretical Background :**

File paths and file operations are fundamental concepts in computer programming and web development. Understanding how to work with file paths and perform file operations is essential for reading, writing, and managing files in various programming languages and platforms.

**Learning Outcome :**

CO1 : Understand various technologies and trends impacting single page web applications.

CO4 : Demonstrate the use of JavaScript to fulfill the essentials of front-end development To back-end development