Practical-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.

Theoretical Background:

➔ URL Parsing: The "url" module in Node.js facilitates parsing a user-provided URL to extract its individual components like protocol, host, path, and query parameters, enabling easy access and manipulation of URL parts.

➔ URL Resolution: The "url" module allows resolving a relative path against a base URL to generate the absolute URL, ensuring correct navigation and proper location resolution on the web or within a file system.

Source Code:

```
const readline = require('readline');
const url = require('url');
// Function to parse the input URL and display its components
function parseURL(inputURL) {
 const parsedURL = url.parse(inputURL, true);
 console.log('Protocol:', parsedURL.protocol);
 console.log('Host:', parsedURL.host);
 console.log('Path:', parsedURL.pathname);
 console.log('Query Parameters:', parsedURL.query);
}
function resolveURL(baseURL, relativePath) {
 const resolvedURL = new URL(relativePath, baseURL);
 console.log('Resolved URL:', resolvedURL.href);
}
// Read input URL from the user
const rl = readline.createInterface({
 input: process.stdin,
 output: process.stdout
});
rl.question('Enter the URL to parse: ', (inputURL) => {
 parseURL(inputURL);
 rl.question('Enter the base URL: ', (baseURL) => {
   rl.question('Enter the relative path: ', (relativePath) => {
     resolveURL(baseURL, relativePath);
     rl.close();
   });
 });
});
```

Output:

```
mac@macs-MBP js cg % node pr4_1_1.js
Enter the URL to parse: http://localhost:3000?param1=first&param2=second
Protocol: http:
Host: localhost:3000
Path: /
Query Parameters: [Object: null prototype] { param1: 'first', param2: 'second' }
Enter the base URL: http://localhost:3000/add
Enter the relative path: /paramnew?param3=value3
Resolved URL: http://localhost:3000/paramnew?param3=value3
```

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.

Theoretical Background:

➔ The Node.js program takes a URL with a query string as input and utilizes the `querystring` module to extract the key-value pairs from the query string. The `querystring` module allows parsing and handling query strings in Node.js. The program then displays the extracted key-value pairs as output, making it easy to work with the data contained in the query string of the provided URL.

Source Code:

```
const readline = require('readline');
const querystring = require('querystring');
const rl = readline.createInterface({
 input: process.stdin,
 output: process.stdout
});
function extractQueryParams(url) {
 const parsedURL = new URL(url);
 const queryParams = parsedURL.searchParams;
 const extractedPairs = {};
 queryParams.forEach((value, key) => {
   extractedPairs[key] = value;
 });
 return extractedPairs;
}
rl.question('Enter the URL with a query string: ', (userInput) => {
 const extractedPairs = extractQueryParams(userInput);
 console.log('Extracted Key-Value Pairs:', extractedPairs);
 rl.close();
});
```

Output:

```
mac@macs-MBP js cg % node pr4_2.js
Enter the URL with a query string: http://localhost:3000?param1=first&param2=second&param3=third
Extracted Key-Value Pairs: { param1: 'first', param2: 'second', param3: 'third' }
```

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.

Theoretical Background:

➔ Checking if Two Paths Refer to the Same File:To determine if two file paths refer to the same file, we can use the `path` module in Node.js. The path module provides the `path.resolve()` method to normalize and resolve file paths, and then we can use standard comparison techniques to check if the resolved paths are equal, indicating they refer to the same file.

➔ Extracting File Extension:Extracting the file extension from a file path can be done using the `path` module as well. The `path.extname()` method extracts the file extension from a given file path, providing a simple and convenient way to get the extension separately.

Source Code:

```
const readline = require('readline');
const path = require('path');
const fs = require('fs');
const rl = readline.createInterface({
 input: process.stdin,
 output: process.stdout
});
function arePathsSame(path1, path2) {
 const absolutePath1 = path.resolve(path1);
 const absolutePath2 = path.resolve(path2);
 return absolutePath1 === absolutePath2;
}
function extractFileExtension(filePath) {
 return path.extname(filePath);
}
// Read input file paths from the user
rl.question('Enter the first file path: ', (filePath1) => {
 rl.question('Enter the second file path: ', (filePath2) => {
   const isSameFile = arePathsSame(filePath1, filePath2);
   console.log(`Are the paths referring to the same file? ${isSameFile}`);
   const fileExtension = extractFileExtension(filePath1);
   console.log('Extracted File Extension:', fileExtension);
   rl.close();
 });
});
```

Output:

```
mac@macs-MBP js cg % node pr4_3.js
Enter the first file path: ./folder1/file1.txt
Enter the second file path: ./folder1/file2.txt
Are the paths referring to the same file? false
Extracted File Extension: .txt
```

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 doesn't.

Theoretical Background:

➔ File Paths: File paths are strings that describe the location of a file or directory in a file system. They can be either relative (relative to the current working directory) or absolute (from the root of the file system). File paths can contain directories, subdirectories, and the file name itself, all separated by specific characters (e.g., `/` or `\`).

➔ Path Module: The path module is a built-in module in Node.js that provides utilities for working with file paths. It allows developers to manipulate, parse, and extract various components of file paths, such as the directory name, base name, extension, etc.

➔ fs Module: The fs (file system) module is another built-in module in Node.js that enables interacting with the file system. It provides methods for reading, writing, deleting, and performing other file-related operations.

➔ Checking File Existence: To check if a file exists, developers can use the `fs.stat()` or `fs.access()` methods from the fs module. The `fs.stat()` method returns file information, while the `fs.access()` method checks for file accessibility. A success message can be displayed if the file exists, or an error message if it doesn't.

Source Code:

```
const readline = require('readline');
const path = require('path');
const fs = require('fs');
const rl = readline.createInterface({
 input: process.stdin,
 output: process.stdout
});
function extractDirectoryAndBaseName(filePath) {
 const directoryName = path.dirname(filePath);
 const baseName = path.basename(filePath);
 return { directoryName, baseName };
}
function checkFileExists(filePath) {
 fs.access(filePath, fs.constants.F_OK, (err) => {
```

```
  if (err) {
    console.log(`File "${filePath}" does not exist.`);
  } else {
    console.log(`File "${filePath}" exists.`);
  }
 });
}
// Read input file path from the user
rl.question('Enter the file path: ', (filePath) => {
 const { directoryName, baseName } = extractDirectoryAndBaseName(filePath);
 console.log('Directory Name:', directoryName);
 console.log('Base Name:', baseName);
 checkFileExists(filePath);
 rl.close();
});
```

Output:

```
mac@macs-MBP js cg % node pr4_4.js
Enter the file path: ./folder1/file1.txt
Directory Name: ./folder1
Base Name: file1.txt
File "./folder1/file1.txt" exists.
```

Learning Outcome:

➔ Understanding more in javascript technologies(CO1).