# Reading Test Data from External Files in TypeScript/JavaScript

## Why Read Data from External Files?

Reading data from external files is a common task in testing and automation. It allows us to:

- Store test data separately from code (e.g., login credentials).
- Keep code clean and maintainable.
- Easily update or scale data without changing the logic.

## Reading Data from a JSON File

**What is JSON?**

**JSON (JavaScript Object Notation)** is a lightweight, easy-to-read data format, ideal for storing structured data.

**Example:**

**data.json:**

```
[
  {
    "email": "laura.taylor1234@example.com",
    "password": "test123",
    "validity": "valid"
  },
  {
    "email": "invaliduser@example.com",
    "password": "wrongpassword",
    "validity": "invalid"
  }
]
```

**Readjsondata.ts**

```
import fs from 'fs';

const jsonPath = 'testdata/data.json';
const loginData: any = JSON.parse(fs.readFileSync(jsonPath, 'utf-8'));

// Destructured loop
for (const { email, password, validity } of loginData) {
  console.log(email, password, validity);
}

// Or using a single variable
for (const data of loginData) {
  console.log(data.email, data.password, data.validity);
}
```

**Code Concepts:**

- fs.readFileSync(path, encoding) – Reads file content synchronously.
- JSON.parse() – Parses a JSON string into a JavaScript object.

## Reading Data from a CSV File

**What is CSV?**

**CSV (Comma-Separated Values)** is a simple text format where each line represents a row and columns are separated by commas. It's ideal for lightweight tabular data.

**Pre-requisite:**

Install the csv-parse module to read CSV files:

```
npm install csv-parse
```

**Example:**

**data.csv**

```
email,password,validity
user1@example.com,pass123,valid
invaliduser@example.com,wrongpassword,invalid
```

**Readcsvdata.ts**

```
import fs from 'fs';
import { parse } from 'csv-parse/sync';

const csvPath = 'testdata/data.csv';
const fileContent = fs.readFileSync(csvPath, 'utf-8');

const loginData = parse(fileContent, {columns: true, skip_empty_lines: true,});


// Destructured loop
for (const { email, password, validity } of loginData) {
  console.log(email, password, validity);
}

// Or using a single variable
for (const data of loginData) {
  console.log(data.email, data.password, data.validity);
}
```

**Code Concepts:**

- Use csv-parse/sync for parsing.
- { columns: true } – Treats the first row as headers.
- { skip_empty_lines: true } – Skips any blank rows.

## Reading Data from an Excel File (.xlsx)

**What is Excel?**

**Excel (.xlsx)** is a file format used by Microsoft Excel to store tabular data. It supports multiple sheets and is widely used in enterprises for managing datasets.

**Pre-requisite:**

Install the **xlsx** module to read Excel files:

```
npm install xlsx
```

**Example:**

**data.xlsx**

| | A | B | C |
|---|---|---|---|
| 1 | **email** | **password** | **validity** |
| 2 | laura.taylor1234@example.com | test123 | valid |
| 3 | invaliduser@example.com | test321 | invalid |
| 4 | validuser@example.com | test222 | invalid |
| 5 | | | invalid |

**readExceldata.ts**

```
import * as XLSX from 'xlsx';

const excelPath = 'testdata/data.xlsx';
const workbook = XLSX.readFile(excelPath);
const sheetName = workbook.SheetNames[0];
const worksheet = workbook.Sheets[sheetName];

const loginData: any = XLSX.utils.sheet_to_json(worksheet);

// Destructured loop
for (const { email, password, validity } of loginData) {
  console.log(email, password, validity);
}

// Or using a single variable
for (const data of loginData) {
  console.log(data.email, data.password, data.validity);
}
```

**Code Concepts:**

- Use the xlsx npm package.
- XLSX.readFile(path) – Reads the Excel file.
- SheetNames[0] – Gets the name of the first sheet.
- XLSX.utils.sheet_to_json() – Converts a sheet to an array of objects.

## Summary:

| File Type | Format Example | Parser Package | Key Methods Used |
|---|---|---|---|
| **JSON** | data.json | fs (built-in) | fs.readFileSync(), JSON.parse() |
| **CSV** | data.csv | csv-parse/sync | parse() with columns, skip_empty_lines |
| **Excel** | data.xlsx | xlsx | readFile(), sheet_to_json() |