Lecture 4: Interacting with the command line

As a coder and a developer, the command line will be your best friend along your journey in coding, why? Because it could potentially cut you a ton of time going through UIs and clicking on things, or just simply switching between your mouse and keyboard. Ans one other important reason why you should be familiar with the command line is that almost all the tools that you may use will have commands that could run through the terminal of your machine. Another use of the terminal is that it could be your only window to a remote machine that you don't have visual access to it, or you simply don't have any physical access to it.
So, to put it in short, the command line is the number one, and a must for any developer to learn to be able to make his work as efficient as it could be.

How do I use my command line?
Well to answer that question, we will first need to know what is the operating system that you will be using to know the type of command line you will need to learn, which means there are more than one command line to be known:

1. Shell-Based Command Lines

Command lines built into operating system shells allow users to interact with the system.

- **Unix/Linux Shells**:
    - **Bash (Bourne Again Shell)**: Standard shell for most Linux systems, offering scripting capabilities and rich tools.
    - **Zsh (Z Shell)**: An extended shell with more features than Bash, like better auto-completion.
    - **Dash (Debian Almquist Shell)**: A lightweight shell used for scripts and system tasks.
    - **Fish (Friendly Interactive Shell)**: User-friendly with advanced auto-suggestions.
- **Windows Command Shells**:
    - **Command Prompt (cmd.exe)**: Default Windows command-line interface for legacy operations.
    - **PowerShell**: An advanced shell with scripting capabilities and integration with .NET.

2. Application-Specific Command Lines

These are specialized for interacting with specific software or services.

- MySQL CLI (`mysql`)
- Git CLI (`git`)
- AWS CLI
- Azure CLI

3. Language-Specific Command Lines

Interpreters or shells for programming languages.

- **Python REPL**: Interactive environment for Python.
- **Node.js REPL**: Interactive JavaScript environment.

Power shell command	Linux command	Use
Get-Location	pwd	To know the directory you are currently in
Get-ChildItem	ls	To list all the files that are in your current directory
Set-Location	cd	To change your location from one directory to another
New-Item	mkdir / touch	To make new files and directories
Remove-Item	rm	To remove files and directories
Copy-Item	cp	To copy files and directories
Move-Item	mv	To move

Use Cases for PowerShell Commands

---

1. Get-Location

- **Use Case**: Retrieve the current working directory.
    - Example: When running scripts, you need to know your current location in the file system.

```
Get-Location
```

Output:

```
Path
----
C:\Users\YourUser
```

---

2. Set-Location

- **Use Case**: Change the current directory to another location.
    - Example: Navigate to a directory before performing file operations.

```
Set-Location -Path "C:\MyDocuments"
```

---

3. New-Item

- **Use Case**: Create a new file or directory.
  - Example 1: Create a text file for logging or notes.

    ```
    New-Item -Path "C:\Logs" -Name "logfile.txt" -ItemType File
    ```

  - Example 2: Create a folder for organizing documents.

    ```
    New-Item -Path "C:\" -Name "MyNewFolder" -ItemType Directory
    ```

---

4. Get-ChildItem

- **Use Case**: List files and directories in a specified location.
  - Example 1: List all files in the current directory.

    ```
    Get-ChildItem
    ```

  - Example 2: Search for files with specific extensions.

    ```
    Get-ChildItem -Path "C:\MyFiles" -Filter "*.txt"
    ```

---

5. Remove-Item

- **Use Case**: Delete files or directories.
  - Example 1: Delete a single file.

    ```
    Remove-Item -Path "C:\MyFiles\oldfile.txt"
    ```

  - Example 2: Delete a directory and all its contents.

    ```
    Remove-Item -Path "C:\OldDirectory" -Recurse -Force
    ```

---

6. Copy-Item

- **Use Case**: Copy files or directories to a new location.
  - Example 1: Backup a file.

    ```
    Copy-Item -Path "C:\MyFiles\file.txt" -Destination "C:\Backup"
    ```

  - Example 2: Copy a directory and its contents.

    ```
    Copy-Item -Path "C:\MyFiles" -Destination "C:\Backup" -Recurse
    ```

---

7. Move-Item

- **Use Case**: Move files or directories to a new location.

- Example 1: Organize files by moving them to a new folder.

  ```
  Move-Item -Path "C:\Downloads\file.txt" -Destination "C:\Documents"
  ```

- Example 2: Consolidate multiple folders into one.

  ```
  Move-Item -Path "C:\OldFolder" -Destination "C:\NewFolder"
  ```

Using Move-Item to Rename Files

Yes, **Move-Item** can be used to rename files or directories! This works because moving an item to the same directory with a new name effectively renames it.

Example: Renaming a File

```
Move-Item -Path "C:\MyFiles\oldname.txt" -Destination "C:\MyFiles\newname.txt"
```

Example: Renaming a Directory

```
Move-Item -Path "C:\OldFolder" -Destination "C:\NewFolder"
```

Would you like more examples or details on these commands?