**Practical Exercise 25 - Parsing Inputs and Running Shell Commands**

**Exercise Description**

**In this practical exercise, our goal is to explore how to parse inputs within a JavaScript custom action.**

Here are the instructions for the exercise:

1. Extend the file named action.yaml under the folder .github/actions/js-dependency-update by adding several necessary inputs:
   1. The base-branch input should:
      1. Have a description of The branch used as the base for the dependency update checks.
      2. Have a default of main.
      3. Not be required.
   2. The target-branch input should:
      1. Have a description of The branch from which the PR is created.
      2. Have a default of update-dependencies.
      3. Not be required.
   3. The working-directory input should:
      1. Have a description of The working directory of the project to check for dependency updates.
      2. Be required.
   4. The gh-token input should:
      1. Have a description of Authentication token with repository access. Must have write access to contents and pull-requests.
      2. Be required.
   5. The debug input should:
      1. Have a description of Whether the output debug messages to the console.
      2. Not be required.
2. Extend the file named 17-2-custom-actions-js.yaml by:
   1. Adding the necessary inputs to the workflow\_dispatch trigger. These are the base-branch, target-branch, working-dir, and debug. The gh-token input for the action can be retrieved from the workflow via the secrets.GITHUB\_TOKEN secret, and does not need to be provided as an input to the workflow.
   2. Pass these inputs as parameters to the js-dependency-update action.
   3. Update the run-name of the workflow to include information about the base branch, target branch, and working directory.
3. **[Optional - If you don't want to code in JavaScript, simply copy the code from the link in the resources of this lecture]** Update the index.js file to:
   1. Retrieve the inputs by using the getInput and getBooleanInput methods from the @actions/core package.
   2. Validating that the provided inputs follow the following constraints:
      1. Branch names should contain only letters, digits, underscores, hyphens, dots, and forward slashes.
      2. Directory paths should contain only letters, digits, underscores, hyphens, and forward slashes.
   3. If any validation fails, use the setFailed method from the @actions/core package to set an error message and fail the action execution.
   4. If all validations pass, print the following information on the screen:
      1. The value of the base branch
      2. The value of the target branch
      3. The value of the working directory
   5. Leverage the @actions/exec package to run shell scripts. For that, use the exec method the mentioned package, or the getExecOutput method whenever you need access to the stdout and stderr of the command.
      1. Run the npm update command within the provided working directory (check the documentation of the exec method for ways to provide the working directory for the command).
      2. Run the git status -s package\*.json to check for updates on package\*.json files. Use the getExecOutput and store the return value of the method in a variable for later usage.
   6. If the stdout of the git status command has any characters, print a message saying that there are updates available. Otherwise, print a message saying that there are no updates at this point in time.
4. Commit the changes and push the code. Trigger the workflow from the UI, passing both valid and invalid values to all the inputs, and take a few moments to inspect the output of the workflow run. How did the action handle different inputs?

Resources for this lecture: <https://github.com/lm-academy/github-actions-course/blob/d45edbdee41d2d65ba91d4fcf872e8e01acce9bf/.github/actions/js-dependency-update/index.js>