Expressions in GitHub Actions

GitHub Actions provides a powerful feature called expressions that allows you to dynamically evaluate and manipulate data within your workflows. Expressions enable you to create more flexible and intelligent automation. This README provides an overview of how to use expressions effectively in GitHub Actions.

Expressions in GitHub Actions are enclosed within `${{ }}` and can be used to access and transform data from GitHub contexts, environment variables, inputs, and more. They are evaluated at runtime and can be used in various workflow scenarios, such as conditional statements, setting variables, or generating dynamic content.

Common Use Cases

Here are some common use cases for using expressions in GitHub Actions:

Dynamic Variable Assignment

You can use expressions to assign dynamic values to variables within your workflow:

```
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Set Variable Using an Expression
        run: |
          # Define a dynamic variable using an expression
          MY_VAR="Hello, ${{ github.actor }}!"

          # Use the variable in an echo statement
          echo $MY_VAR
```

Conditional Steps

Expressions allow you to create conditional steps based on GitHub context or other data:

```
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Conditional Step Using an Expression
        run: |
          if [[ "${{ github.event_name }}" == "push" && "${{ github.ref }}" ==
```

```
"refs/heads/master" ]]; then
            echo "This is a push to the master branch."
          fi
```

We can also use expressions in `if` statements when deciding whether to execute a step or a job. Under the `if` key, we don't need to add the special `${{ <expression> }}` syntax.

```
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Conditional Step Using an Expression
        if: github.event_name == "push" && github.ref == "refs/heads/master"
        run: echo "This is a push to the master branch."
```

Testing Expressions in Workflow

1. Create Workflow File:

- Create a new file named `09-01-Using-Expressions.yml` in the `.github/workflows` directory of your repository.

```
name: 09-01-Using Expressions
run-name: 09-01-Using Expressions | DEBUG - ${{ inputs.debug && 'ON' ||
'OFF' }}

on:
  push:
  workflow_dispatch:
    inputs:
      debug:
        type: boolean
        default: false

jobs:
  echo:
    runs-on: ubuntu-latest
    steps:
      - name: '[debug] Print start-up data'
        if: inputs.debug
        run: |
          echo "Triggered by: ${{ github.event_name }}"
          echo "Branch: ${{ github.ref }}"
          echo "Commit SHA: ${{ github.sha }}"
          echo "Runner OS: ${{ runner.os }}"
      - name: '[debug] Print when triggered from master'
        if: inputs.debug && github.ref == 'refs/heads/master'
        run: echo "I was triggered from master"
```

```
      - name: Greeting
        run: echo "Hello, world"
```

- Copy and paste the provided YAML configuration into this file.

2. Understanding the Workflow:

- This workflow is triggered by both the `push` and `workflow_dispatch` events.
- It accepts a boolean input named `debug`, which defaults to `false`.
- The `run-name` specifies a dynamic name for the workflow run based on the value of the `debug` input.
- The `echo` job runs on the `ubuntu-latest` environment and consists of three steps.
- The first step (`[debug] Print start-up data`) prints startup data if the `debug` input is set to `true`.
- The second step (`[debug] Print when triggered from master`) prints a message if the workflow is triggered from the master branch and the `debug` input is set to `true`.
- The third step (`Greeting`) always prints a greeting message.

3. Testing the Workflow:

- Commit and push the workflow file (`09-01-Using-Expressions.yml`) to your repository.
- Navigate to the "Actions" tab in your GitHub repository.
- Manually trigger the workflow by clicking on the "Run workflow" button for the `09-01-Using-Expressions` workflow.
- If prompted, provide a value for the `debug` input. You can set it to `true` or `false` based on your testing requirements.
- Once the workflow run completes, click on the job to view its details.
- Review the logs of each step to ensure that the conditional steps are executed according to the provided input value.

4. Observing Dynamic Run Name:

- After triggering the workflow, observe the name of the workflow run. It should reflect whether debugging is enabled (`ON`) or disabled (`OFF`) based on the input value provided during manual triggering.