

Hands-on Lab: Implementing Control Flow and Conditional Statements



Estimated time needed: 15 minutes

What you will learn

In this lab, you will delve into the fundamental concept of control flow and conditional statements in JavaScript. Through hands-on implementation, you will grasp the essence of if...else statements, nested statements, and switch statements, understanding how these structures enable code execution based on specific conditions. You will gain insight into how altering these elements impacts the program's flow and output.

Learning objectives

After completing this lab, you will be able to:

- **Decision-making constructs:** Learn about if statements for single-condition execution, if else statements to execute different code blocks based on conditions, and nested if else statements to handle multiple conditions hierarchically.
- **Control flow and efficiency:** Explore logical flow control to manage program flow based on conditions and code optimization to enhance readability and efficiency.
- **Handling multiple scenarios:** Understand how to manage complexity to deal with multiple conditions effectively and switch statements to streamline code for multiple scenarios.
- **Real-world application and problem-solving:** Learn about applied problem solving to utilize conditional statements in practical scenarios and enhanced logic building to strengthen problem-solving skills through logic constructs.

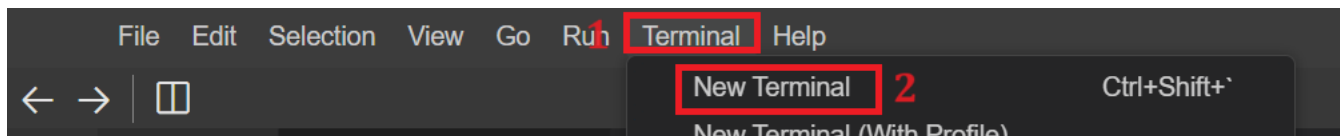
Prerequisites:

- Basic knowledge of HTML and Git commands.
- Basic understanding of JavaScript control flow and conditional statements such as if, if else, and switch case.
- Web browser with a console (Chrome DevTools, Firefox Console, and so on).

Step 1: Setting up the environment

1. First, you need to clone your main repository in the **Skills Network Environment**. Follow the given steps:

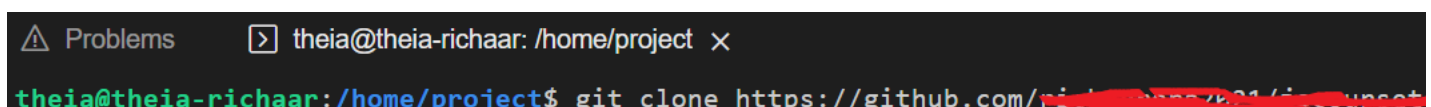
- Click on the terminal in the top-right window pane and then select **New Terminal**.



- Perform `git clone` command by writing given command in the terminal.

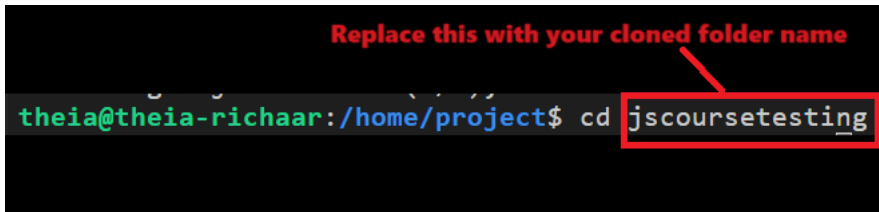
```
git clone <github-repository-url>
```

Note: Put your own GitHub repository link instead of `<github-repository-url>` and it should look like as given below:



- Above step will clone folder for your GitHub repository under project folder in explorer. You will also see multiple folders inside cloned folder.
- Now you need to navigate inside the cloned folder. For this write given command in the terminal:

```
cd <repository-folder-name>
```



Note: Write your cloned folder name instead of <repository-folder-name>. Perform `git clone` if you have logged out of **Skills Network Environment** and you cannot see any files or folder after you logged in.

- Now, select the **cloned Folder Name** folder, right-click on it, and choose **New Folder**. Enter the folder name as **controlFlow**. This will create the folder for you. Then, select the **controlFlow** folder, right-click, and choose **New File**. Enter the file name as **control_flow.html** and click OK. This will create your HTML file.
- Now select the **controlFlow** folder again, right-click and select **New File**. Enter the file name as **control_flow.js** and click OK. This will create your JavaScript file.
- Create a basic template structure for **control_flow.html** file by adding the code provided below.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Control Flow</title>
</head>
<body>
  <h1>Control Flow and Conditional Statements</h1>
  <script src="./control_flow.js"></script>
</body>
</html>
```

- Above HTML code has one `<h1>` tag in file and one `<script>` tag to include js file in **control_flow.html** file using **src** attribute.

Step 2: Defining variables and if else statement for userRole and accessLevel

- Declare variable named **userRole** and initialize it with the string value "admin" in **control_flow.js** file. Also, declare one more variable named **accessLevel** but do not assign a value to it yet.

```
let userRole = "admin";
let accessLevel;
```

- Now, execute the if...else block by assigning different roles in the if...else condition to check if **userRole** is equal to "admin" or not. Include the following code in the **control_flow.js** file after the previous code:

```
if (userRole === "admin") {
  accessLevel = "Full access granted";
} else if (userRole === "manager") {
  accessLevel = "Limited access granted";
} else {
  accessLevel = "No access granted";
}
```

3. Now, the above code will check if **userRole** is "admin" or something else.

- If **userRole** is "admin", the code will assign **accessLevel** as "Full access granted".
- If not, it will proceed to check if **userRole** is "manager".
- If **userRole** is "manager", it will assign **accessLevel** as "Limited access granted".
- If **userRole** is neither "admin" nor "manager", the code will assign **accessLevel** as "No access granted".

4. Based on the value of **userRole**, the **accessLevel** variable will be set to one of the following:

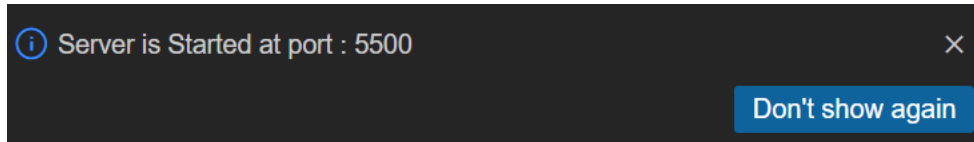
- "Full access granted" if `userRole === "admin"`
- "Limited access granted" if `userRole === "manager"`
- "No access granted" for any other value of `userRole`


You will be able to see the output using this code:

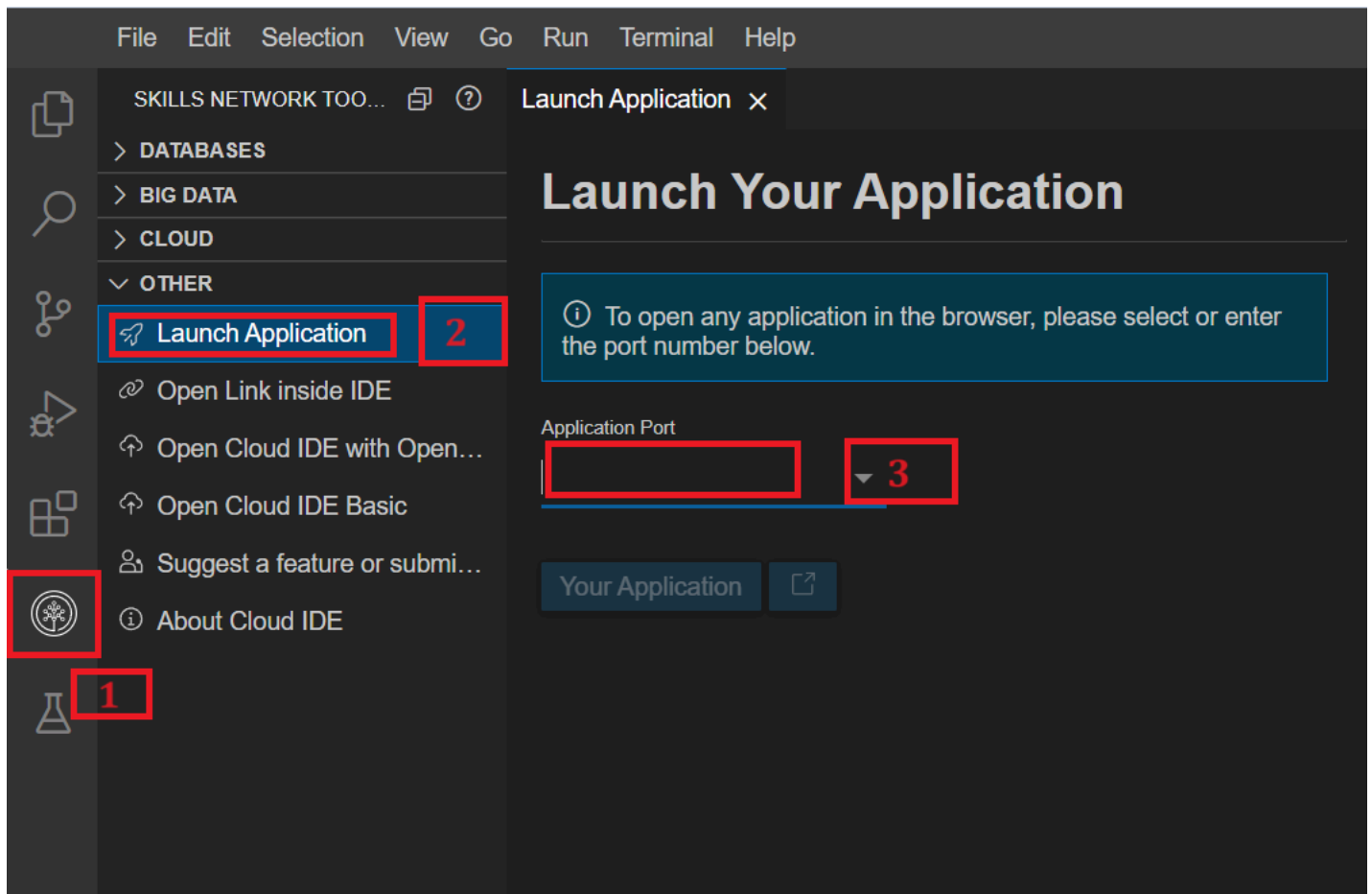
```
console.log("Access Level:", accessLevel);
```

Check output

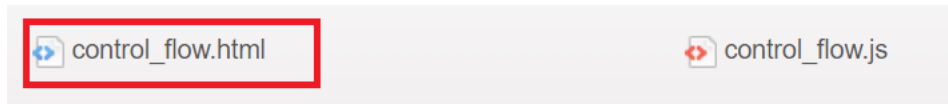
1. To view your HTML page, right-click the **control_flow.html** file after selecting this file, then select "Open with Live Server".
2. The server should start on port 5500, indicated by a notification on the bottom-right side.



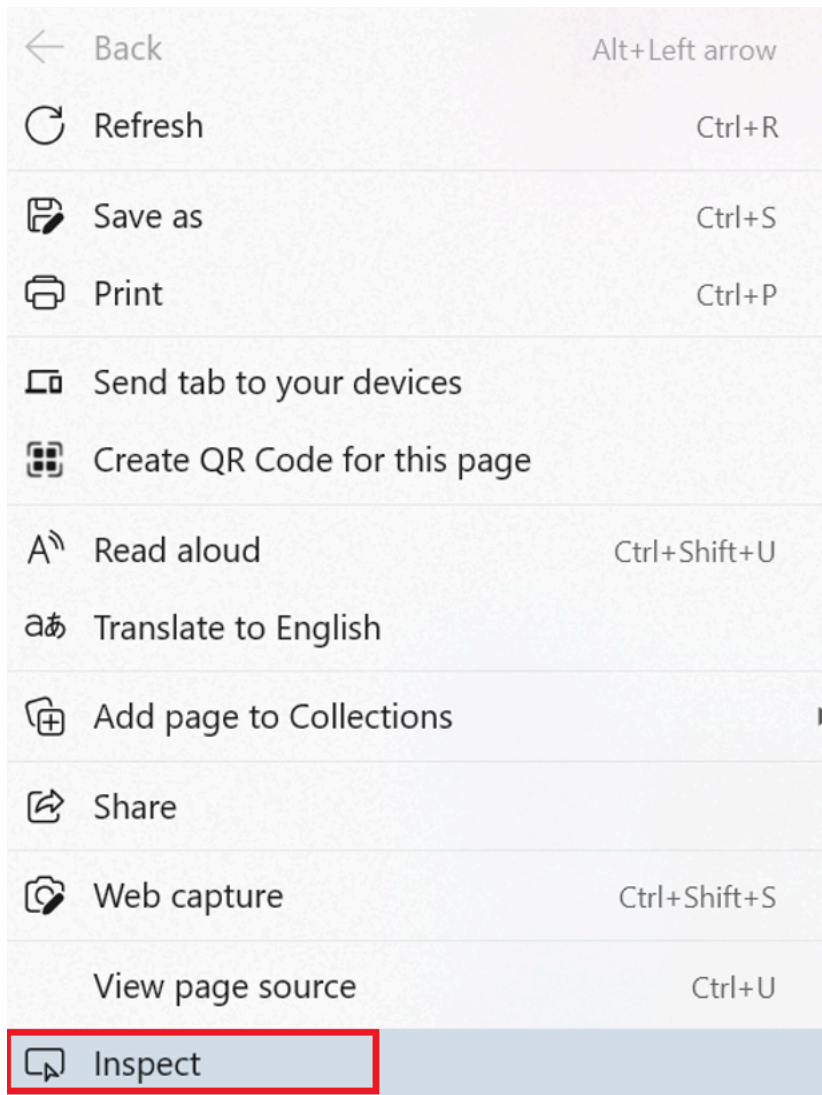
3. Click the Skills Network button on the left (refer to number 1). It will open the "Skills Network Toolbox". Then click Launch Application (refer to number 2). From there, you enter port 5500 at number 3 and click this button .



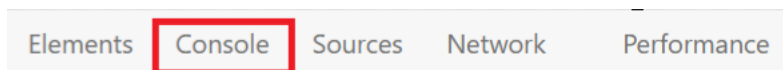
4. It will open your default browser, where you will first see the name of your cloned folder. Click on that folder, and inside it, among other folders, you will find the **controlFlow** folder name. Click on the **controlFlow** folder, and then select the HTML file, as shown below.



5. To view the output in the browser, right-click on the window that opens after selecting the "control_flow.html" file, and then choose the "inspect" option.



6. Then, go to console tab.



7. You will see the output **Access Level:Full access granted** because default value of userRole is admin.

Access Level: Full access granted

Step 3: Defining variables and nested if...else statementd for isLoggedIn and userMessage

1. Declare a variable named **isLoggedIn** and initialize it with the boolean value "true" in **control_flow.js** file. Declare one more variable named **userMessage** but do not assign a value to it yet. Insert the provided code after the previous code.

```
let isLoggedIn = true;
let userMessage;
```

2. Now, implement and execute the Nested if...else statement to check if user is logged in or not:

```
if (isLoggedIn) {
  if (userRole === "admin") {
    userMessage = "Welcome, Admin!";
  } else {
```

```

        userMessage = "Welcome, User!";
    }
} else {
    userMessage = "Please log in to access the system.";
}

```

3. If user is logged in `isLoggedIn === true`, the code checks the user's role (`userRole`).

- If `userRole` is "admin", it sets `userMessage` to "Welcome, Admin!".
- If `userRole` is not "admin", it sets `userMessage` to "Welcome, User!".

4. If User is not logged in `isLoggedIn === false`, then:

- The message is set to "Please log in to access the system."

5. You can use the following console method to view the output:

```
console.log("User Message:", userMessage);
```

6. You will see the output as **User Message: Welcome, Admin!** because the default value of `isLoggedIn` is true.

Access Level: Full access granted

User Message: Welcome, Admin!

Step 4: Defining variables and switch statement for `userType` and `userCategory`

1. Declare a variable named **`userType`** and initialize it with the string value "subscriber" in **`control_flow.js`** file. Declare one more variable named **`userCategory`** but do not assign a value to it yet. Insert the provided code after the previous code.

```
let userType = "subscriber";
let userCategory;
```

2. Now, you need to implement and execute the switch statement to evaluate the value of **`userType`** by providing different case values:

```

switch (userType) {
    case "admin":
        userCategory = "Administrator";
        break;
    case "manager":
        userCategory = "Manager";
        break;
    case "subscriber":
        userCategory = "Subscriber";
        break;
    default:
        userCategory = "Unknown";
}

```

3. The output for **cases** depend upon its value, such as:

- Case "admin":
 - If userType is "admin", userCategory is assigned as "Administrator".
 - break; exits the switch statement after the assignment.
- Case "manager":
 - If userType is "manager", userCategory is assigned as "Manager".
 - break; exits the switch statement after the assignment.
- Case "subscriber":
 - If userType is "subscriber", userCategory is assigned as "Subscriber".
 - break; exits the switch statement after the assignment.
- Default Case:
 - If userType doesn't match any defined cases ("admin", "manager", or "subscriber"), userCategory is assigned as "Unknown".

4. You can use the following console method to view the output:

```
console.log("User Category:", userCategory);
```

5. You will see the output as **User Category: Subscriber** because the default value of userType is subscriber.

Access Level: Full access granted

User Message: Welcome, Admin!

User Category: Subscriber

6. Perform git add, git commit, and git push commands to update changes of your **controlFlow** folder into GitHub repository for proper code management.

Step 5: Use ternary operator for isAuthenticated and authenticationStatus

1. Declare a variable named **isAuthenticated** and initialize it with the boolean value true in **control_flow.js** file.

```
let isAuthenticated = true;
```

2. Declare one more variable named **authenticationStatus** and use a ternary operator (? :) to see if the user is authenticated or not.

```
let authenticationStatus = isAuthenticated ? "Authenticated" : "Not authenticated";
```

3. Now the condition will be checked.

- If **isAuthenticated** is true, the expression before : (in this case, "Authenticated") is assigned to **authenticationStatus**.
- If **isAuthenticated** is false, the expression after : (in this case, "Not authenticated") is assigned to **authenticationStatus**.

4. You can use the following console method to view the output:

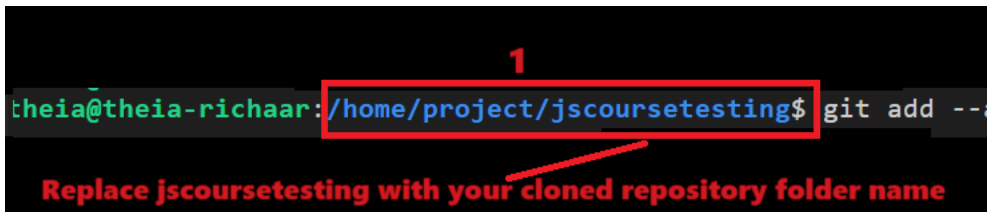
```
console.log("Authentication Status:", authenticationStatus);
```

Step 6: Perform Git commands

1. Perform `git add` to add latest files and folder by writing given command in terminal in git environment.

```
git add --a
```

Make sure terminal should have path as follows:



2. Then perform `git commit` in the terminal. While performing `git commit`, terminal can show message to set up your `git config --global` for user.name and user.email. If yes, then you need to perform `git config` command as well for user.name and user.email as given.

```
git config --global user.email "you@example.com"
```

```
git config --global user.name "Your Name"
```

Note: Replace data within quotes with your own details.

Then perform commit command as given:

```
git commit -m "message"
```

3. Then perform `git push` just by writing given command in terminal.

```
git push origin
```


- After the push command, the system will prompt you to enter your username and password. Enter the username for your GitHub account and the password that you created in the first lab. After entering the credentials, all of your latest folders and files will be pushed to your GitHub repository.

Practice task

1. Suppose an organization arranges a "Dietary Services" program to provide diets to its employees and customers, based on a person's weight and day-to-day routine. You need to create an authorization-based code to provide access to people based on their roles in organization, such as employees, enrolled members for "Dietary Services," and subscribers.
 - If the person is an **Employee**, they are authorized to have access to "Dietary Services".
 - If the person is an **Enrolled Member**, they are authorized to have access to "Dietary Services" and one-on-one interaction with a dietician.
 - If the person is a **Subscriber**, they are authorized to have partial access to facilitate "Dietary Services" only.
 - If the person is a **Non-Subscriber**, they need to enroll or at least subscribe first to avail this facility.
2. You need to communicate with the user by printing a message indicating whether that person is eligible to avail which type of services.

Summary

- Using conditional statements and control flow, you can direct how a program behaves based on different situations or criteria, allowing for decision-making and defining specific pathways within the code.
1. Variables declaration:
 - Set up a HTML file linked to a JavaScript file in a folder named "controlFlow."
 - Create variables for userRole, accessLevel, isLoggedIn, userMessage, userType, userCategory, isAuthenticated, and authenticationStatus.
 2. Implementing control flow:
 - Use if...else statements to assign access levels based on user roles.
 - Implement nested if...else statements to customize messages based on login status and user roles.
 - Utilize a switch statement to categorize users based on their type.
 3. Ternary operator for authentication:
 - Use a ternary operator to determine the authentication status.
 - Depending on the value of isAuthenticated, set the authenticationStatus as "Authenticated" or "Not authenticated."

© IBM Corporation. All rights reserved.