

Laboratório Prático: Implementando Fluxo de Controle e Instruções Condicionais



Tempo estimado necessário: 15 minutos

O que você vai aprender

Neste laboratório, você irá se aprofundar no conceito fundamental de fluxo de controle e instruções condicionais em JavaScript. Através da implementação prática, você compreenderá a essência das instruções if...else, instruções aninhadas e instruções switch, entendendo como essas estruturas permitem a execução de código com base em condições específicas. Você terá uma visão de como alterar esses elementos impacta o fluxo e a saída do programa.

Objetivos de aprendizagem

Após concluir este laboratório, você será capaz de:

- **Estruturas de tomada de decisão:** Aprender sobre instruções if para execução de uma única condição, instruções if else para executar diferentes blocos de código com base em condições, e instruções if else aninhadas para lidar com múltiplas condições de forma hierárquica.
- **Fluxo de controle e eficiência:** Explorar o controle lógico de fluxo para gerenciar o fluxo do programa com base em condições e otimização de código para melhorar a legibilidade e a eficiência.
- **Gerenciando múltiplos cenários:** Entender como gerenciar a complexidade para lidar efetivamente com múltiplas condições e instruções switch para simplificar o código para múltiplos cenários.
- **Aplicação no mundo real e resolução de problemas:** Aprender sobre resolução de problemas aplicada para utilizar instruções condicionais em cenários práticos e construção de lógica aprimorada para fortalecer habilidades de resolução de problemas através de estruturas lógicas.

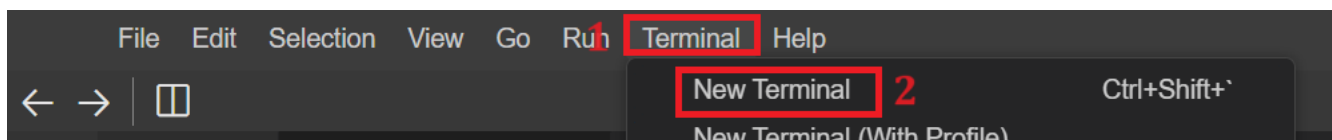
Pré-requisitos:

- Conhecimento básico de HTML e comandos Git.
- Compreensão básica do fluxo de controle e instruções condicionais em JavaScript, como if, if else e switch case.
- Navegador web com console (Chrome DevTools, Firefox Console, etc.).

Passo 1: Configurando o ambiente

1. Primeiro, você precisa clonar seu repositório principal no **Ambiente Skills Network**. Siga os passos dados:

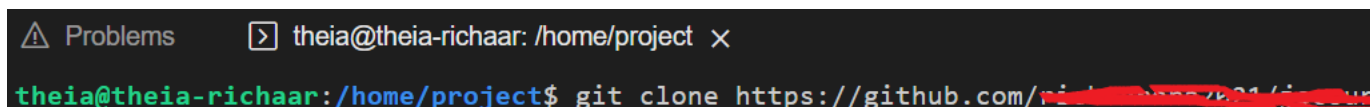
- Clique no terminal no canto superior direito e, em seguida, selecione **Novo Terminal**.



- Execute o comando `git clone` escrevendo o comando fornecido no terminal.

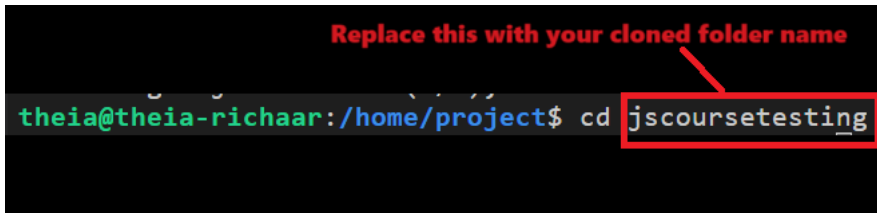
```
git clone <github-repository-url>
```

Nota: Coloque o link do seu próprio repositório GitHub em vez de `<github-repository-url>` e ele deve ficar parecido com o mostrado abaixo:



- O passo acima irá clonar a pasta do seu repositório GitHub dentro da pasta do projeto no explorador. Você também verá várias pastas dentro da pasta clonada.
- Agora você precisa navegar para dentro da pasta clonada. Para isso, escreva o comando fornecido no terminal:

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



Nota: Escreva o nome da sua pasta clonada em vez de <repository-folder-name>. Execute `git clone` se você saiu do **Ambiente Skills Network** e não consegue ver nenhum arquivo ou pasta após fazer login novamente.

2. Agora, selecione a pasta **cloned Folder Name**, clique com o botão direito sobre ela e escolha **Nova Pasta**. Digite o nome da pasta como **controlFlow**. Isso criará a pasta para você. Em seguida, selecione a pasta **controlFlow**, clique com o botão direito e escolha **Novo Arquivo**. Digite o nome do arquivo como **control_flow.html** e clique em OK. Isso criará seu arquivo HTML.
3. Agora selecione a pasta **controlFlow** novamente, clique com o botão direito e selecione **Novo Arquivo**. Digite o nome do arquivo como **control_flow.js** e clique em OK. Isso criará seu arquivo JavaScript.
4. Crie uma estrutura básica de template para o arquivo **control_flow.html** adicionando o código fornecido abaixo.

```
<!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>
```

5. O código HTML acima possui uma tag <h1> no arquivo e uma tag <script> para incluir o arquivo js no **control_flow.html** usando o atributo **src**.

Passo 2: Definindo variáveis e declaração if else para userRole e accessLevel

1. Declare uma variável chamada **userRole** e inicialize-a com o valor de string "admin" no arquivo **control_flow.js**. Além disso, declare mais uma variável chamada **accessLevel**, mas ainda não atribua um valor a ela.

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

2. Agora, execute o bloco `if...else` atribuindo diferentes funções na condição `if...else` para verificar se **userRole** é igual a "admin" ou não. Inclua o seguinte código no arquivo **control_flow.js** após o código anterior:

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

3. Agora, o código acima verificará se **userRole** é "admin" ou algo diferente.
 - o Se **userRole** for "admin", o código atribuirá **accessLevel** como "Acesso total concedido".
 - o Se não, ele prosseguirá para verificar se **userRole** é "manager".
 - o Se **userRole** for "manager", ele atribuirá **accessLevel** como "Acesso limitado concedido".
 - o Se **userRole** não for nem "admin" nem "manager", o código atribuirá **accessLevel** como "Nenhum acesso concedido".

4. Com base no valor de **userRole**, a variável **accessLevel** será definida como um dos seguintes:

- o "Acesso total concedido" se `userRole === "admin"`
- o "Acesso limitado concedido" se `userRole === "manager"`
- o "Nenhum acesso concedido" para qualquer outro valor de `userRole`


Você poderá ver a saída usando este código:

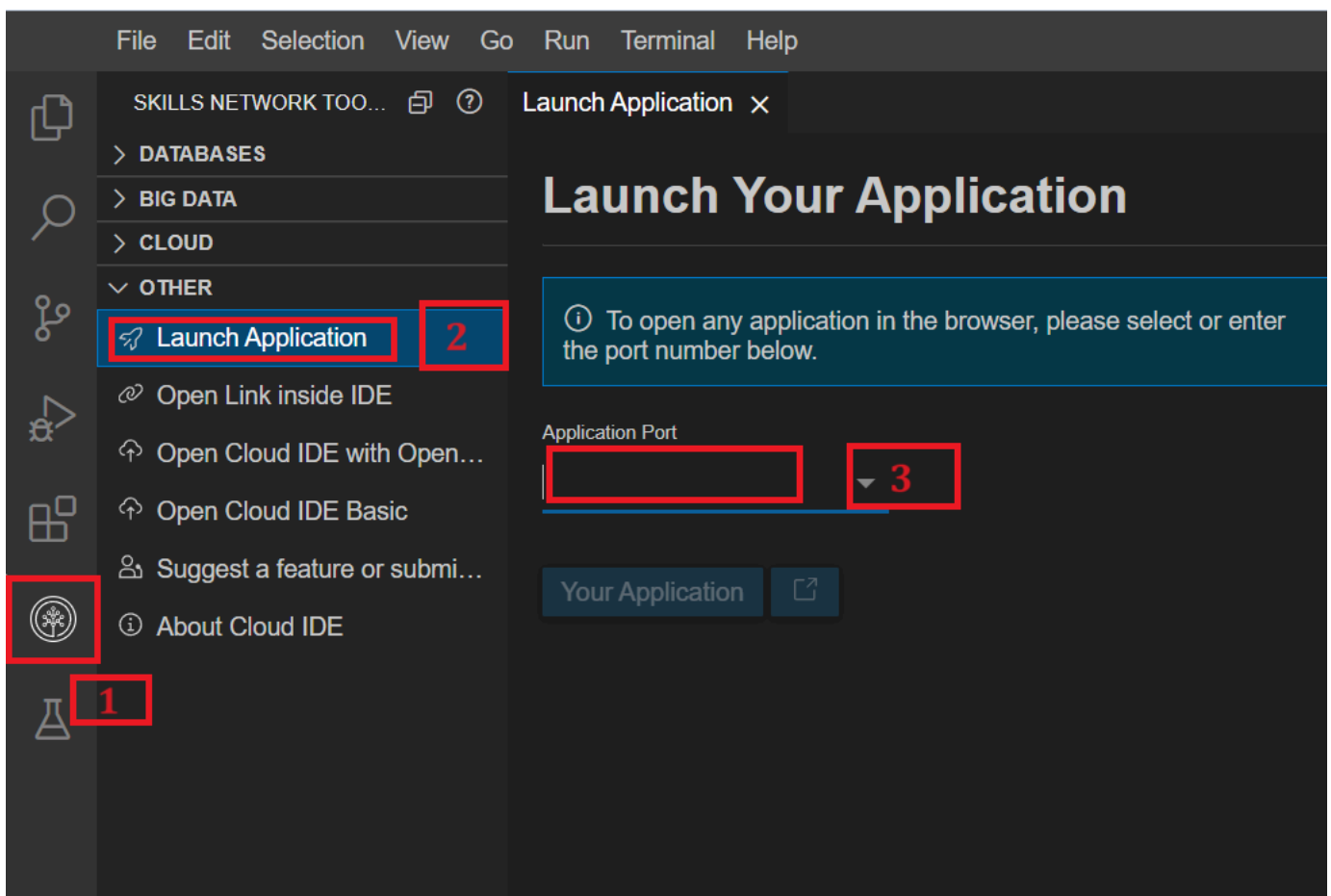
```
console.log("Access Level:", accessLevel);
```

Ver saída

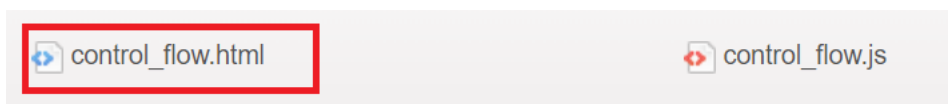
1. Para visualizar sua página HTML, clique com o botão direito no arquivo **control_flow.html** após selecioná-lo, e então selecione “Abrir com Live Server”.
2. O servidor deve iniciar na porta 5500, indicado por uma notificação no canto inferior direito.



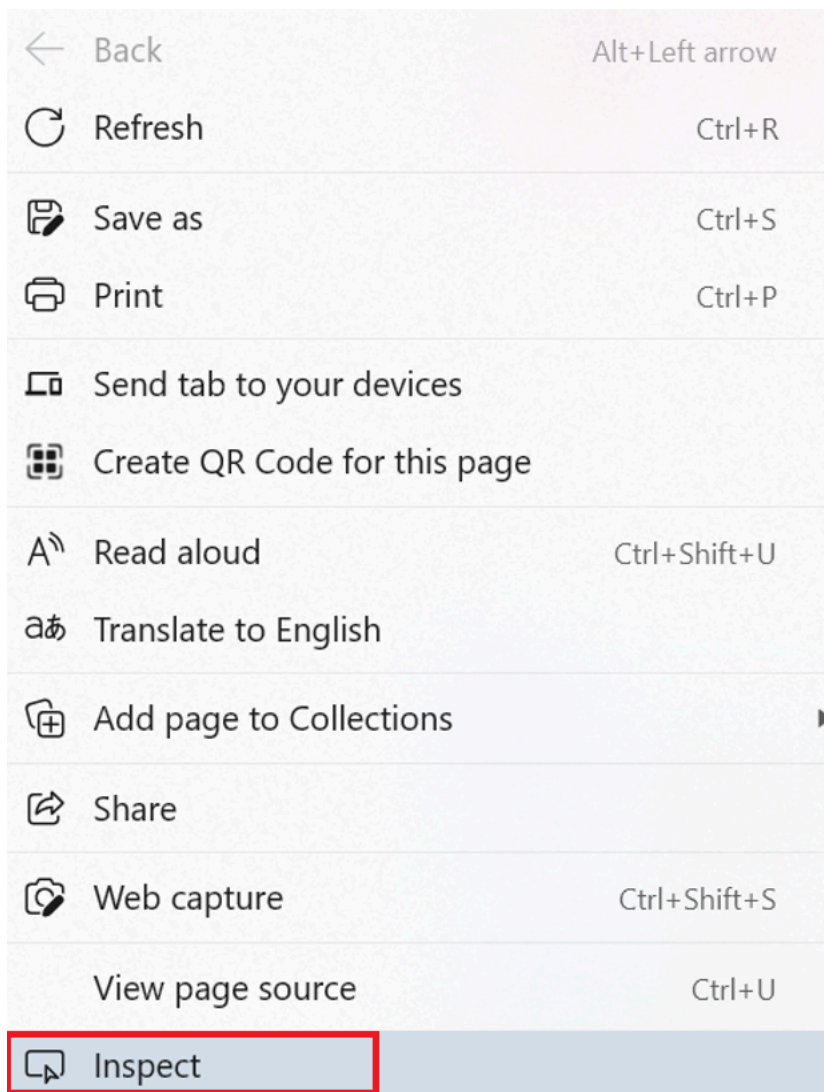
3. Clique no botão Skills Network à esquerda (consulte o número 1). Isso abrirá a “Caixa de Ferramentas do Skills Network”. Em seguida, clique em Iniciar Aplicação (consulte o número 2). A partir daí, você insere a porta 5500 no número 3 e clica neste botão .



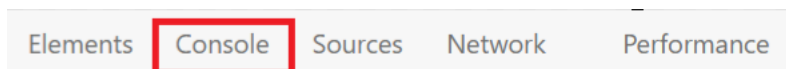
4. Isso abrirá seu navegador padrão, onde você verá primeiro o nome da sua pasta clonada. Clique nessa pasta e, dentro dela, entre outras pastas, você encontrará o nome da pasta **controlFlow**. Clique na pasta **controlFlow** e, em seguida, selecione o arquivo HTML, conforme mostrado abaixo.



5. Para visualizar a saída no navegador, clique com o botão direito na janela que se abre após selecionar o arquivo “control_flow.html” e, em seguida, escolha a opção “inspecionar”.



6. Em seguida, vá para a aba console.



7. Você verá a saída **Nível de Acesso: Acesso total concedido** porque o valor padrão de `userRole` é `admin`.

Access Level: Full access granted

Passo 3: Definindo variáveis e instruções `if...else` aninhadas para `isLoggedIn` e `userMessage`

1. Declare uma variável chamada `isLoggedIn` e inicialize-a com o valor booleano `"true"` no arquivo `control_flow.js`. Declare mais uma variável chamada `userMessage`, mas ainda não atribua um valor a ela. Insira o código fornecido após o código anterior.

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

2. Agora, implemente e execute a declaração `if...else` aninhada para verificar se o usuário está logado ou não:

```
if (isLoggedIn) {
  if (userRole === "admin") {
    userMessage = "Welcome, Admin!";
  } else {
    userMessage = "Welcome, User!";
  }
} else {
  userMessage = "Please log in to access the system.";
}
```

3. Se o usuário estiver logado `isLoggedIn === true`, o código verifica o papel do usuário (`userRole`).

- o Se `userRole` for "admin", ele define `userMessage` como "Bem-vindo, Admin!".
- o Se `userRole` não for "admin", ele define `userMessage` como "Bem-vindo, Usuário!".

4. Se o usuário não estiver logado `isLoggedIn === false`, então:

- o A mensagem é definida como "Por favor, faça login para acessar o sistema."

5. Você pode usar o seguinte método de console para ver a saída:

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

6. Você verá a saída como **User Message: Welcome, Admin!** porque o valor padrão de `isLoggedIn` é verdadeiro.

Access Level: Full access granted

User Message: Welcome, Admin!

Passo 4: Definindo variáveis e declaração switch para `userType` e `userCategory`

1. Declare uma variável chamada **`userType`** e inicialize-a com o valor de string "subscriber" no arquivo **`control_flow.js`**. Declare mais uma variável chamada **`userCategory`**, mas não atribua um valor a ela ainda. Insira o código fornecido após o código anterior.

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

2. Agora, você precisa implementar e executar a instrução switch para avaliar o valor de **`userType`** fornecendo diferentes valores de caso:

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

3. A saída para **casos** depende do seu valor, como:

- Caso "admin":
 - o Se `userType` for "admin", `userCategory` é atribuído como "Administrador".
 - o `break`; sai da declaração switch após a atribuição.
- Caso "manager":
 - o Se `userType` for "manager", `userCategory` é atribuído como "Gerente".
 - o `break`; sai da declaração switch após a atribuição.

- Caso "subscriber":
 - Se userType for "subscriber", userCategory é atribuído como "Assinante".
 - break; sai da declaração switch após a atribuição.
- Caso Padrão:
 - Se userType não corresponder a nenhum dos casos definidos ("admin", "manager" ou "subscriber"), userCategory é atribuído como "Desconhecido".

4. Você pode usar o seguinte método do console para ver a saída:

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

5. Você verá a saída como **Categoria do Usuário: Assinante** porque o valor padrão de userType é assinante.

Access Level: Full access granted

User Message: Welcome, Admin!

User Category: Subscriber

6. Execute os comandos `git add`, `git commit` e `git push` para atualizar as alterações da sua pasta **controlFlow** no repositório do GitHub para uma gestão adequada do código.

Passo 5: Use o operador ternário para isAuthenticated e authenticationStatus

1. Declare uma variável chamada **isAuthenticated** e inicialize-a com o valor booleano `true` no arquivo **control_flow.js**.

```
let isAuthenticated = true;
```

2. Declare mais uma variável chamada **authenticationStatus** e use um operador ternário (`? :`) para verificar se o usuário está autenticado ou não.

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

3. Agora a condição será verificada.

- Se **isAuthenticated** for verdadeiro, a expressão antes de `:` (neste caso, "Authenticated") é atribuída a **authenticationStatus**.
- Se **isAuthenticated** for falso, a expressão depois de `:` (neste caso, "Not authenticated") é atribuída a **authenticationStatus**.

4. Você pode usar o seguinte método do console para ver a saída:

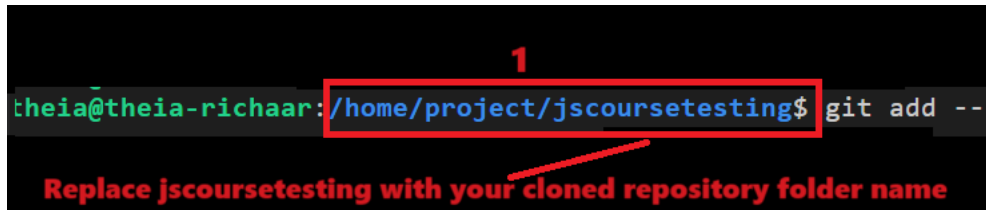
```
console.log("Authentication Status:", authenticationStatus);
```

Passo 6: Executar comandos Git

1. Execute `git add` para adicionar os arquivos e pastas mais recentes, escrevendo o comando fornecido no terminal no ambiente git.

```
git add --a
```

Certifique-se de que o terminal tenha o caminho conforme mostrado:



2. Em seguida, execute `git commit` no terminal. Ao executar `git commit`, o terminal pode mostrar uma mensagem para configurar seu `git config --global` para `user.name` e `user.email`. Se sim, você precisará executar o comando `git config` também para `user.name` e `user.email`, conforme mostrado.

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

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

Nota: Substitua os dados entre aspas pelos seus próprios detalhes.

Em seguida, execute o comando de commit conforme mostrado:

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

3. Depois, execute `git push` apenas escrevendo o comando fornecido no terminal.

```
git push origin
```

- Após o comando de push, o sistema solicitará que você insira seu nome de usuário e senha. Insira o nome de usuário da sua conta GitHub e a senha que você criou no primeiro laboratório. Após inserir as credenciais, todas as suas pastas e arquivos mais recentes serão enviados para o seu repositório GitHub.

Tarefa de prática

- Suponha que uma organização organize um programa de "Serviços Dietéticos" para fornecer dietas a seus funcionários e clientes, com base no peso e na rotina diária de uma pessoa. Você precisa criar um código baseado em autorização para fornecer acesso às pessoas com base em seus papéis na organização, como funcionários, membros inscritos nos "Serviços Dietéticos" e assinantes.
 - Se a pessoa for um **Funcionário**, ela está autorizada a ter acesso aos "Serviços Dietéticos".
 - Se a pessoa for um **Membro Inscrito**, ela está autorizada a ter acesso aos "Serviços Dietéticos" e interação individual com um nutricionista.
 - Se a pessoa for um **Assinante**, ela está autorizada a ter acesso parcial para facilitar os "Serviços Dietéticos" apenas.
 - Se a pessoa for um **Não-Assinante**, ela precisa se inscrever ou pelo menos assinar primeiro para usufruir dessa facilidade.
- Você precisa se comunicar com o usuário imprimindo uma mensagem indicando se essa pessoa é elegível para usufruir de qual tipo de serviços.

Resumo

- Usando declarações condicionais e controle de fluxo, você pode direcionar como um programa se comporta com base em diferentes situações ou critérios, permitindo a tomada de decisões e definindo caminhos específicos dentro do código.

1. Declaração de variáveis:

- Configure um arquivo HTML vinculado a um arquivo JavaScript em uma pasta chamada "controlFlow."
- Crie variáveis para userRole, accessLevel, isLoggedIn, userMessage, userType, userCategory, isAuthenticated e authenticationStatus.

2. Implementando controle de fluxo:

- Use declarações if...else para atribuir níveis de acesso com base em funções de usuário.
- Implemente declarações if...else aninhadas para personalizar mensagens com base no status de login e nas funções de usuário.
- Utilize uma declaração switch para categorizar usuários com base em seu tipo.

3. Operador ternário para autenticação:

- Use um operador ternário para determinar o status de autenticação.
- Dependendo do valor de isAuthenticated, defina authenticationStatus como "Autenticado" ou "Não autenticado."

© IBM Corporation. Todos os direitos reservados.