Laboratorio Práctico: Implementación de Flujo de Control y Declaraciones Condicionales

Tiempo estimado necesario: 15 minutos

Lo que aprenderás

En este laboratorio, profundizarás en el concepto fundamental de flujo de control y declaraciones condicionales en JavaScript. A través de la implementación práctica, comprenderás la esencia de las declaraciones if…else, las declaraciones anidadas y las declaraciones switch, entendiendo cómo estas estructuras permiten la ejecución del código en función de condiciones específicas. Obtendrás información sobre cómo alterar estos elementos impacta el flujo y la salida del programa.

Objetivos de aprendizaje

Después de completar este laboratorio, podrás:

- **Construcciones de toma de decisiones:** Aprender sobre las declaraciones if para la ejecución de una sola condición, las declaraciones if else para ejecutar diferentes bloques de código según las condiciones, y las declaraciones anidadas if else para manejar múltiples condiciones de manera jerárquica.
- **Flujo de control y eficiencia:** Explorar el control de flujo lógico para gestionar el flujo del programa según las condiciones y la optimización del código para mejorar la legibilidad y eficiencia.
- **Manejo de múltiples escenarios:** Entender cómo gestionar la complejidad para tratar múltiples condiciones de manera efectiva y las declaraciones switch para simplificar el código para múltiples escenarios.
- **Aplicación en el mundo real y resolución de problemas:** Aprender sobre la resolución de problemas aplicada para utilizar declaraciones condicionales en escenarios prácticos y el fortalecimiento de la construcción lógica para mejorar las habilidades de resolución de problemas a través de construcciones lógicas.

Requisitos previos:

- Conocimientos básicos de HTML y comandos de Git.
- Comprensión básica del flujo de control y declaraciones condicionales de JavaScript como if, if else y switch case.
- Navegador web con consola (Chrome DevTools, Firefox Console, etc.).

Paso 1: Configuración del entorno

1. Primero, necesitas clonar tu repositorio principal en el **Entorno de Skills Network**. Sigue los pasos dados:

- Haz clic en la terminal en el panel de la ventana superior derecha y luego selecciona **Nueva Terminal**.

- Realiza el comando `git clone` escribiendo el comando dado en la terminal.

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

  Nota: Pon tu propio enlace de repositorio de GitHub en lugar de `<github-repository-url>` y debería verse como se indica a continuación:

  - El paso anterior clonará la carpeta de tu repositorio de GitHub en la carpeta del proyecto en el explorador. También verás múltiples carpetas dentro de la carpeta clonada.
  - Ahora necesitas navegar dentro de la carpeta clonada. Para esto, escribe el comando dado en la terminal:

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

    Nota: Escribe el nombre de tu carpeta clonada en lugar de `<repository-folder-name>`. Realiza `git clone` si has cerrado sesión en el **Entorno de Skills Network** y no puedes ver ningún archivo o carpeta después de iniciar sesión.

2. Ahora, selecciona la carpeta **cloned Folder Name**, haz clic derecho sobre ella y elige **Nueva Carpeta**. Ingresa el nombre de la carpeta como **controlFlow**. Esto creará la carpeta para ti. Luego, selecciona la carpeta **controlFlow**, haz clic derecho y elige **Nuevo Archivo**. Ingresa el nombre del archivo como **control_flow.html** y haz clic en Aceptar. Esto creará tu archivo HTML.

3. Ahora selecciona la carpeta **controlFlow** nuevamente, haz clic derecho y selecciona **Nuevo Archivo**. Ingresa el nombre del archivo como **control_flow.js** y haz clic en Aceptar. Esto creará tu archivo de JavaScript.

4. Crea una estructura de plantilla básica para el archivo **control_flow.html** agregando el código proporcionado a continuación.

```
<!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. El código HTML anterior tiene una etiqueta `<h1>` en el archivo y una etiqueta `<script>` para incluir el archivo js en **control_flow.html** utilizando el atributo **src**.

Paso 2: Definición de variables y declaración if else para userRole y accessLevel

1. Declara una variable llamada **userRole** e inicialízala con el valor de cadena "admin" en el archivo **control_flow.js**. Además, declara una variable más llamada **accessLevel** pero aún no le asignes un valor.

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

2. Ahora, ejecuta el bloque if…else asignando diferentes roles en la condición if…else para verificar si **userRole** es igual a “admin” o no. Incluye el siguiente código en el archivo **control_flow.js** después del código anterior:

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

3. Ahora, el código anterior verificará si **userRole** es “admin” o algo más.

    - Si **userRole** es “admin”, el código asignará **accessLevel** como “Acceso completo concedido”.
    - Si no, procederá a verificar si **userRole** es “manager”.
    - Si **userRole** es “manager”, asignará **accessLevel** como “Acceso limitado concedido”.

- Si **userRole** no es ni “admin” ni “manager”, el código asignará **accessLevel** como “Sin acceso concedido”.

4. Basado en el valor de **userRole**, la variable **accessLevel** se establecerá en uno de los siguientes:

   - “Acceso completo concedido” si userRole === “admin”
   - “Acceso limitado concedido” si userRole === “manager”
   - “Sin acceso concedido” para cualquier otro valor de userRole

   Podrás ver la salida utilizando este código:

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

Ver salida

1. Para ver tu página HTML, haz clic derecho en el archivo **control_flow.html** después de seleccionarlo, luego selecciona “Abrir con Live Server”.

2. El servidor debería iniciarse en el puerto 5500, indicado por una notificación en la parte inferior derecha.

3. Haz clic en el botón de Skills Network a la izquierda (consulta el número 1). Se abrirá la “Caja de herramientas de Skills Network”. Luego haz clic en Lanzar Aplicación (consulta el número 2). Desde allí, ingresa el puerto 5500 en el número 3 y haz clic en este botón .

4. Se abrirá tu navegador predeterminado, donde primero verás el nombre de tu carpeta clonada. Haz clic en esa carpeta, y dentro de ella, entre otras carpetas, encontrarás el nombre de la carpeta **controlFlow**. Haz clic en la carpeta **controlFlow**, y luego selecciona el archivo HTML, como se muestra a continuación.

5. Para ver la salida en el navegador, haz clic derecho en la ventana que se abre después de seleccionar el archivo “control_flow.html”, y luego elige la opción “inspeccionar”.

6. Luego, ve a la pestaña de consola.

7. Verás la salida **Nivel de Acceso: Acceso completo concedido** porque el valor predeterminado de userRole es admin.

Paso 3: Definiendo variables y declaración de if… else anidados para isLoggedIn y userMessage

1. Declara una variable llamada **isLoggedIn** e inicialízala con el valor booleano "true" en el archivo **control_flow.js**. Declara una variable más llamada **userMessage** pero aún no le asignes un valor. Inserta el código proporcionado después del código anterior.

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

2. Ahora, implementa y ejecuta la declaración anidada if…else para verificar si el usuario ha iniciado sesión o no:

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

3. Si el usuario está conectado `isLoggedIn === true`, el código verifica el rol del usuario (userRole).
    - Si userRole es “admin”, establece userMessage en “¡Bienvenido, Admin!”.
    - Si userRole no es “admin”, establece userMessage en “¡Bienvenido, Usuario!”.
4. Si el usuario no está conectado `isLoggedIn === false`, entonces:
    - El mensaje se establece en “Por favor, inicie sesión para acceder al sistema.”
5. Puede usar el siguiente método de consola para ver la salida:

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

6. Verás la salida como **Mensaje del Usuario: ¡Bienvenido, Admin!** porque el valor predeterminado de **isLoggedIn** es verdadero.

Paso 4: Definición de variables y declaración switch para userType y userCategory

1. Declara una variable llamada **userType** e inicialízala con el valor de cadena "subscriber" en el archivo **control_flow.js**. Declara una variable más llamada **userCategory** pero no le asignes un valor todavía. Inserta el código proporcionado después del código anterior.

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

2. Ahora, necesitas implementar y ejecutar la declaración switch para evaluar el valor de **userType** proporcionando 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. La salida para **casos** depende de su valor, como por ejemplo:

- Caso "admin":
    - Si userType es "admin", userCategory se asigna como "Administrador".
    - break; sale de la declaración switch después de la asignación.
- Caso "manager":
    - Si userType es "manager", userCategory se asigna como "Gerente".

    - break; sale de la declaración switch después de la asignación.

- Caso "subscriber":

    - Si userType es "subscriber", userCategory se asigna como "Suscriptor".
    - break; sale de la declaración switch después de la asignación.

- Caso por Defecto:

    - Si userType no coincide con ninguno de los casos definidos ("admin", "manager" o "subscriber"), userCategory se asigna como "Desconocido".

4. Puedes usar el siguiente método de consola para ver la salida:

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

5. Verás la salida como **Categoría de Usuario: Suscriptor** porque el valor predeterminado de userType es suscriptor.

6. Realiza los comandos `git add`, `git commit` y `git push` para actualizar los cambios de tu carpeta **controlFlow** en el repositorio de GitHub para una adecuada gestión del código.

Paso 5: Usar el operador ternario para isAuthenticated y authenticationStatus

1. Declara una variable llamada **isAuthenticated** e inicialízala con el valor booleano `true` en el archivo **control_flow.js**.

```
let isAuthenticated = true;
```

2. Declara una variable más llamada **authenticationStatus** y utiliza un operador ternario `(? :)` para verificar si el usuario está autenticado o no.

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

3. Ahora se verificará la condición.
    - Si **isAuthenticated** es verdadero, la expresión antes de : (en este caso, "Authenticated") se asigna a **authenticationStatus**.
    - Si **isAuthenticated** es falso, la expresión después de : (en este caso, "Not authenticated") se asigna a **authenticationStatus**.
4. Puedes usar el siguiente método de consola para ver la salida:

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

Paso 6: Realizar comandos de Git

1. Realiza `git add` para agregar los últimos archivos y carpetas escribiendo el comando dado en la terminal en el entorno de git.

```
git add --a
```

Asegúrate de que la terminal tenga la ruta como sigue:

2. Luego realiza `git commit` en la terminal. Al realizar `git commit`, la terminal puede mostrar un mensaje para configurar tu `git config --global` para user.name y user.email. Si es así, entonces necesitas ejecutar el comando `git config` también para `user.name` y `user.email` como se indica.

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

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

Nota: Reemplaza los datos entre comillas con tus propios detalles.

Luego realiza el comando de commit como se indica:

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

3. Luego realiza `git push` simplemente escribiendo el comando dado en la terminal.

```
git push origin
```

- Después del comando de push, el sistema te pedirá que ingreses tu nombre de usuario y contraseña. Ingresa el nombre de usuario de tu cuenta de GitHub y la contraseña que creaste en el primer laboratorio. Después de ingresar las credenciales, todas tus últimas carpetas y archivos se enviarán a tu repositorio de GitHub.

Tarea de práctica

1. Suponga que una organización organiza un programa de "Servicios Dietéticos" para proporcionar dietas a sus empleados y clientes, basado en el peso de una persona y su rutina diaria. Necesita crear un código basado en autorización para proporcionar acceso a las personas según sus roles en la organización, como empleados, miembros inscritos para los "Servicios Dietéticos" y suscriptores.
    - Si la persona es un **Empleado**, está autorizado a tener acceso a los "Servicios Dietéticos".
    - Si la persona es un **Miembro Inscrito**, está autorizado a tener acceso a los "Servicios Dietéticos" y a interacción uno a uno con un dietista.
    - Si la persona es un **Suscriptor**, está autorizado a tener acceso parcial para facilitar los "Servicios Dietéticos" únicamente.

- Si la persona es un **No Suscriptor**, necesita inscribirse o al menos suscribirse primero para poder acceder a esta instalación.

2. Necesita comunicarse con el usuario imprimiendo un mensaje que indique si esa persona es elegible para acceder a qué tipo de servicios.

Resumen

- Usando declaraciones condicionales y flujo de control, puedes dirigir cómo se comporta un programa según diferentes situaciones o criterios, lo que permite la toma de decisiones y la definición de caminos específicos dentro del código.

1. Declaración de variables:

    - Configura un archivo HTML vinculado a un archivo JavaScript en una carpeta llamada "controlFlow."
    - Crea variables para userRole, accessLevel, isLoggedIn, userMessage, userType, userCategory, isAuthenticated y authenticationStatus.

2. Implementación del flujo de control:

    - Usa declaraciones if…else para asignar niveles de acceso según los roles de usuario.
    - Implementa declaraciones if…else anidadas para personalizar mensajes según el estado de inicio de sesión y los roles de usuario.
    - Utiliza una declaración switch para categorizar a los usuarios según su tipo.

3. Operador ternario para autenticación:

    - Usa un operador ternario para determinar el estado de autenticación.
    - Dependiendo del valor de isAuthenticated, establece authenticationStatus como "Autenticado" o "No autenticado."

© IBM Corporation. Todos los derechos reservados.