Bucle Canciones JavaScript

A continuación, explicamos la implementación y creación de un código simple en JavaScript, que permite utilizar las listas circulares en un programa que simula la reproducción en bucle de 5 canciones en una playlist (lista circular). Se utilizan dos clases: **ListaCircular** y **Nodo.**

Clase `Nodo`

Definición de la Clase

```
class Nodo {
```

- **class Nodo**: Aquí estamos definiendo una clase llamada `Nodo`. En JavaScript, una clase es una plantilla para crear objetos con propiedades y métodos.

Constructor

```
constructor(cancion) {
    this.cancion = cancion;
    this.siguiente = null;
}
```

- **constructor(cancion)**: El constructor es un método especial para crear e inicializar un objeto creado a partir de una clase. En este caso, el constructor toma un parámetro `cancion`.
- **this.cancion = cancion**: Aquí estamos asignando el valor del parámetro `cancion` a una propiedad del objeto `Nodo` llamado `cancion`. Esto significa que cada instancia de `Nodo` tendrá una propiedad `cancion` que almacena el valor pasado al constructor.
- **this.siguiente = null**: Esta línea inicializa la propiedad `siguiente` del nodo a `null`. La propiedad `siguiente` se utilizará para apuntar al siguiente nodo en

la lista circular. Inicialmente, se establece en `null` porque cuando se crea un nodo, no está conectado a ningún otro nodo.

Exportación del Módulo

```
module.exports = Nodo;
```

- **module.exports = Nodo**: Esta línea exporta la clase `Nodo` para que pueda ser utilizada en otros archivos. En Node.js, `module.exports` se utiliza para exportar funciones, objetos o valores primitivos desde un archivo para que puedan ser utilizados en otro archivo con `require`.

Resumen

- **Clase `Nodo`**: Define una estructura básica para un nodo en una lista circular.
- **Constructor**: Inicializa las propiedades `cancion` y `siguiente`.
- **Exportación**: Permite que la clase `Nodo` sea utilizada en otros archivos.

Clase `ListaCircular`

Importación del Módulo `Nodo`

```
const Nodo = require('./Nodo');
```

- **const Nodo = require('./Nodo')**: Esta línea importa la clase `Nodo` desde el archivo `Nodo.js`. Esto permite que la clase `ListaCircular` cree instancias de `Nodo`.

Definición de la Clase

```
class ListaCircular {
```

- **class ListaCircular**: Aquí estamos definiendo una clase llamada `ListaCircular`. En JavaScript, una clase es una plantilla para crear objetos

con propiedades y métodos.

Constructor

```
constructor() {
    this.inicio = null;
    this.ultimo = null;
}
```

- **constructor()**: El constructor es un método especial para crear e inicializar un objeto creado a partir de una clase. En este caso, el constructor no toma parámetros.
- **this.inicio = null**: Inicializa la propiedad `inicio` a `null`. Esta propiedad apuntará al primer nodo de la lista circular.
- **this.ultimo = null**: Inicializa la propiedad `ultimo` a `null`. Esta propiedad apuntará al último nodo de la lista circular.

Método `agregarCancion`

```
agregarCancion(cancion) {
    const nuevoNodo = new Nodo(cancion);
    if (this.inicio == null) {
        this.inicio = nuevoNodo;
        this.ultimo = nuevoNodo;
        this.ultimo.siguiente = this.inicio;
    } else {
        this.ultimo.siguiente = nuevoNodo;
        this.ultimo = nuevoNodo;
        this.ultimo.siguiente = this.inicio;
    }
}
```

- **agregarCancion(cancion)**: Este método toma un parámetro `cancion` y agrega un nuevo nodo con esa canción a la lista circular.
- **const nuevoNodo = new Nodo(cancion)**: Crea una nueva instancia de `Nodo` con la canción proporcionada.
- **if (this.inicio == null)**: Verifica si la lista está vacía.

- **this.inicio = nuevoNodo**: Si la lista está vacía, el nuevo nodo se convierte en el primer nodo (`inicio`).
- **this.ultimo = nuevoNodo**: El nuevo nodo también se convierte en el último nodo (`ultimo`).
- **this.ultimo.siguiente = this.inicio**: El `siguiente` del último nodo apunta al primer nodo, cerrando el ciclo.

- **else**: Si la lista no está vacía:
  - **this.ultimo.siguiente = nuevoNodo**: El `siguiente` del último nodo actual apunta al nuevo nodo.
  - **this.ultimo = nuevoNodo**: El nuevo nodo se convierte en el último nodo.
  - **this.ultimo.siguiente = this.inicio**: El `siguiente` del nuevo último nodo apunta al primer nodo, manteniendo el ciclo.

Método `reproducir`

```
reproducir(veces) {
    if (this.inicio !== null) {
        let actual = this.inicio;
        for (let i = 0; i < veces; i++) {
            do {
                console.log(`Reproduciendo: ${actual.cancio
n}`);
                actual = actual.siguiente;
            } while (actual !== this.inicio);
        }
    }
}
```

- **reproducir(veces)**: Este método toma un parámetro `veces` y reproduce la lista circular el número de veces especificado.
- **if (this.inicio !== null)**: Verifica si la lista no está vacía.
- **let actual = this.inicio**: Inicializa una variable `actual` que apunta al primer nodo.
- **for (let i = 0; i < veces; i++)**: Un bucle que se ejecuta el número de veces especificado.

- **do { ... } while (actual !== this.inicio)**: Un bucle `do-while` que recorre la lista circular.
  - **console.log(`Reproduciendo: ${actual.cancion}`)**: Imprime la canción del nodo actual.
  - **actual = actual.siguiente**: Avanza al siguiente nodo en la lista.

Exportación del Módulo

```
module.exports = ListaCircular;
```

- **module.exports = ListaCircular**: Esta línea exporta la clase `ListaCircular` para que pueda ser utilizada en otros archivos. En Node.js, `module.exports` se utiliza para exportar funciones, objetos o valores primitivos desde un archivo para que puedan ser utilizados en otro archivo con `require`.

Resumen

- **Clase** `ListaCircular`: Define una estructura para una lista circular de canciones.
- **Constructor**: Inicializa las propiedades `inicio` y `ultimo`.
- **Método** `agregarCancion`: Agrega un nuevo nodo a la lista circular.
- **Método** `reproducir`: Reproduce la lista circular el número de veces especificado.
- **Exportación**: Permite que la clase `ListaCircular` sea utilizada en otros archivos.

Archivo `main.js`

Importaciones

```
require('./Nodo');
const ListaCircular = require('./ListaCircular');
```

- **require('./Nodo')**: Aunque esta línea no es necesaria en `main.js` porque `Nodo` se utiliza dentro de `ListaCircular`, no causa problemas. `require` se utiliza para importar módulos en Node.js.
- **const ListaCircular = require('./ListaCircular')**: Importa la clase `ListaCircular` desde el archivo `ListaCircular.js`. Esto permite crear una instancia de `ListaCircular` y utilizar sus métodos.

Creación de una Instancia de `ListaCircular`

```
const listaCanciones = new ListaCircular();
```

- **const listaCanciones = new ListaCircular()**: Crea una nueva instancia de `ListaCircular` llamada `listaCanciones`. Esta instancia se utilizará para almacenar y manipular la lista circular de canciones.

Agregar Canciones a la Lista Circular

```
listaCanciones.agregarCancion("Nice Guy - BOYNEXTDOOR");
listaCanciones.agregarCancion("0X1=LOVESONG (I Know I Love
You) Feat. Seori - TOMORROW X TOGETHER");
listaCanciones.agregarCancion("Style (Taylor´s Version) - T
aylor Swift");
listaCanciones.agregarCancion("Rock with you - SEVENTEEN");
listaCanciones.agregarCancion("Espresso - Sabrina Carpente
r");
```

- **listaCanciones.agregarCancion("...")**: Llama al método `agregarCancion` de la instancia `listaCanciones` para agregar varias canciones a la lista circular. Cada llamada crea un nuevo nodo con la canción especificada y lo agrega a la lista.

Simular la Reproducción en Bucle

```
console.log("Iniciando reproducción:");
listaCanciones.reproducir(5);
```

- **console.log("Iniciando reproducción:")**: Imprime un mensaje en la consola indicando que la reproducción está comenzando.

- **listaCanciones.reproducir(5)**: Llama al método `reproducir` de la instancia `listaCanciones` para simular la reproducción de la lista circular 5 veces. Este método recorre la lista circular y imprime cada canción el número de veces especificado.

Resumen

- **Importaciones**: Importa los módulos necesarios (`Nodo` y `ListaCircular`).
- **Instancia de `ListaCircular`**: Crea una instancia de la lista circular.
- **Agregar Canciones**: Agrega varias canciones a la lista circular.
- **Reproducción en Bucle**: Simula la reproducción de la lista circular 5 veces, imprimiendo cada canción en la consola.

Este archivo `main.js` actúa como el punto de entrada para la aplicación, configurando y ejecutando la lógica de la lista circular de canciones.
