¡Por supuesto! Vamos a desglosar paso por paso cómo funciona la función deepClone en JavaScript para copiar objetos de manera recursiva.

Paso 1: Comprender el objetivo

Queremos copiar un objeto sin que los cambios en el objeto copiado afecten al original. Si un objeto contiene otros objetos dentro (es decir, tiene una estructura anidada), debemos asegurarnos de copiar cada nivel de manera independiente.

Paso 2: Crear la función deepClone

La función deepClone(obj) recibe un objeto y lo copia completamente, incluso si tiene propiedades que son también objetos.

function deepClone(obj) {

Aquí declaramos la función que aceptará un parámetro obj.

Paso 3: Verificar si obj es un objeto

Antes de empezar a copiar, debemos verificar si obj es realmente un objeto. Si es un tipo de dato primitivo (como un número o cadena), lo devolvemos directamente.

```
  if (obj === null || typeof obj !== 'object') {
    return obj;
  }
```

Si obj es null o no es un objeto, simplemente devolvemos obj sin cambios.

Paso 4: Crear una copia inicial

Si obj es un objeto, necesitamos crear un nuevo objeto vacío ({}) o, si es un array, un nuevo array ([]).

```
  const copy = Array.isArray(obj) ? [] : {};
```

Aquí usamos Array.isArray(obj) para comprobar si obj es un array y, según el resultado, creamos un array vacío o un objeto vacío.

Paso 5: Recorrer las propiedades del objeto

Ahora recorremos cada propiedad dentro de obj y copiamos su valor recursivamente:

```
  for (const key in obj) {
    if (obj.hasOwnProperty(key)) {
      copy[key] = deepClone(obj[key]);
    }
  }
```

- Usamos un bucle for...in para recorrer las propiedades del objeto.
- obj.hasOwnProperty(key) verifica que la propiedad pertenece al objeto y no es heredada de otro prototipo.
- deepClone(obj[key]) llama nuevamente a la función para copiar la propiedad recursivamente.

Paso 6: Devolver el objeto copiado

Al final de la función, devolvemos copy, que contiene la copia completa del objeto.

```
  return copy;
}
```

Paso 7: Probar la función

Ahora podemos probar nuestra función con un objeto anidado:

```
const original = {
  a: 1,
  b: { c: 2, d: { e: 3 } }
};

const cloned = deepClone(original);
console.log(cloned);
```

Esto imprimirá una copia del objeto original, que se ha duplicado sin estar referenciado al original.

¿Te ha quedado claro hasta aquí? 😊 Si tienes dudas en algún paso, dime y te ayudo a entenderlo mejor. 🚀