

Practica

Patrones de diseño

Código	Descripción
<pre>1 class Database { 2 constructor() { 3 if (!Database.instance) { 4 Database.instance = this; 5 } 6 return Database.instance; 7 } 8 9 query(sql) { 10 console.log("Ejecutando consulta:", sql); 11 } 12 } 13 14 const db1 = new Database(); 15 const db2 = new Database(); 16 17 console.log(db1 === db2); // Output: true 18 db1.query("SELECT * FROM users");</pre>	<p>Se crea una clase llamada <i>Database</i> contiene un constructor donde mediante el <code>if</code> checa si es que ya existe una instancia de <i>Database</i> y si no le da un valor, en caso de que si haya regresa su valor.</p> <p>Tengo entendido que en <i>query</i> lo único que hace es que imprime el mensaje "<i>Ejecutando consulta:</i>", pero aún no termino de comprender que función hace el <code>sql</code>.</p> <p>Fuera de la clase <i>Database</i> se crean dos instancias de la clase <i>Database</i> las cuales después se usan para compararse e indicar si son la misma instancia.</p> <p>Yo creo que se esta usando el patrón Singleton porque este nos habla de que se puede limitar una clase para que solo se instancié una vez y es lo que se hace en el <code>if</code> del constructor. Algo que también ayuda a verlo es que al comparar <code>db1</code> y <code>db2</code> la salida es <code>true</code>, entonces es la misma instancia de la clase.</p>

Código	Descripción
<pre> 1 class Logger { 2 constructor() { 3 this.logs = []; 4 } 5 6 log(message) { 7 this.logs.push(message); 8 console.log("Log registrado:", message); 9 } 10 11 static getInstance() { 12 if (!Logger.instance) { 13 Logger.instance = new Logger(); 14 } 15 return Logger.instance; 16 } 17 } 18 19 const logger1 = Logger.getInstance(); 20 const logger2 = Logger.getInstance(); 21 22 console.log(logger1 === logger2); // Output: true 23 logger1.log("Error: No se puede conectar al servidor"); </pre>	<p>Se crea una clase llamada <i>Logger</i> que contiene su constructor donde se declara un arreglo llamado <i>logs</i> que esta vacío.</p> <p>Existe una subclase llamada <i>log</i> donde entran mensajes, aquí lo que se hace es que mediante el <u>push</u> el mensaje que entra se agrega al arreglo <i>logs</i> y al hacerlo imprime "Log registrado:" .</p> <p>En <i>getInstance</i> lo que sucede es que de nuevo revisa si es que hay una instancia en <i>Logger</i> y en caso de que no se instancia, en caso de que sí se devuelve el valor.</p> <p>Fuera de la clase se declaran <i>logger1</i> y <i>logger2</i> donde para ambos se llama a la función <i>getInstance</i>.</p> <p>Se comparan para verificar si son el mismo valor y después se llama al método <i>log</i> en <i>logger1</i> que registra un mensaje en el array <i>logs</i>.</p> <p>Creo que también usa el patrón Singleton en el método <i>getInstance</i>.</p>

Código	Descripción
<pre> 1 class User { 2 constructor(name) { 3 this.name = name; 4 } 5 6 greet() { 7 console.log("Hola, soy", this.name); 8 } 9 } 10 11 class UserFactory { 12 createUser(name) { 13 return new User(name); 14 } 15 } 16 17 const factory = new UserFactory(); 18 const user1 = factory.createUser("Juan"); 19 const user2 = factory.createUser("Maria"); 20 21 user1.greet(); // Output: Hola, soy Juan 22 user2.greet(); // Output: Hola, soy Maria </pre>	<p>Se crea una clase llamada <i>User</i> y en su constructor se guarda el nombre de usuario, en el método <i>greet</i> lo imprime junto con el mensaje "Hola, soy".</p> <p>Hay una segunda clase <i>UserFactory</i> con un método (<i>createUser</i>) para crear instancias de <i>User</i> con el nombre que se da. Aquí es donde se crean los objetos.</p> <p>Se agrega una instancia en la línea 17 de <i>UserFactory</i> bajo el nombre <i>factory</i>.</p> <p>Se crean los usuarios usando el método <i>createUser</i>.</p> <p>Por último se llama al método <i>greet</i> con cada usuario, lo que imprime el saludo.</p> <p>En este código no pude identificar ningún patrón, pero creo que lo que sí contiene es el principio de Single Responsibility ya que <i>User</i> solo tiene una responsabilidad que es guarda el nombre del usuario y su método para saludar, así como <i>UserFactory</i> solo crea instancias de <i>User</i>.</p>