Tarea 05

Problema I

A continuación se muestra una manera desordenada y poco eficiente de crear usuarios.

```
//Tratemos de crear usuarios de la siguiente manera

//usuario0
let nombre = 'Paola';
let apellido = 'Ortiz';
let email = 'paola@company.ru'
let direccion = {
municipio: 'Jocotenango',
calle: 'Calle ancha',
numero: 25,
};

//usuario1
let nombre1 = 'Paolo';
let apellido1 = 'Ortega';
let email1= 'paolo@company.ru'
let direccion1 = {
municipio: 'Jocotenango',
calle: 'Calle ancha',
numero: 25,
};

//usuario2...
```

Además de estos usuarios, crea al usuario2, al usuario3, al usuario4, al usuario5, con las mismas características que se muestran en la figura inmediatamente anterior. ¿Puedes mostrar estas características de los usuarios de manera ordenada y coherente en consola? ¿Tienen los usuarios alguna funcionalidad, pueden hacer algo?

Solucion:

```
// Problema I: Creación de usuarios de manera ordenada

let usuario0 = {
    nombre: 'Paola',
    apellido: 'Ortiz',
    email: 'paola@company.ru',
    direccion: {
        municipio: 'Jocotenango',
        calle: 'Calle ancha',
        numero: 25
```

```
    }
};

let usuario1 = {
    nombre: 'Paolo',
    apellido: 'Ortega',
    email: 'paolo@company.ru',
    direccion: {
        municipio: 'Jocotenango',
        calle: 'Calle ancha',
        numero: 25
    }
};

let usuario2 = {
    nombre: 'Pedro',
    apellido: 'Pérez',
    email: 'pedro@company.ru',
    direccion: {
        municipio: 'Jocotenango',
        calle: 'Calle ancha',
        numero: 25
    }
};

let usuario3 = {
    nombre: 'Pablo',
    apellido: 'Ramírez',
    email: 'pablo@company.ru',
    direccion: {
        municipio: 'Jocotenango',
        calle: 'Calle ancha',
        numero: 25
    }
};

let usuario4 = {
    nombre: 'Patricia',
    apellido: 'Rodríguez',
    email: 'patricia@company.ru',
    direccion: {
        municipio: 'Jocotenango',
        calle: 'Calle ancha',
        numero: 25
    }
};

let usuario5 = {
```

```
    nombre: 'Pamela',
    apellido: 'Reyes',
    email: 'pamela@company.ru',
    direccion: {
        municipio: 'Jocotenango',
        calle: 'Calle ancha',
        numero: 25
    }
};

// Mostrar las características de los usuarios en consola
console.log(usuario0);
console.log(usuario1);
console.log(usuario2);
console.log(usuario3);
console.log(usuario4);
console.log(usuario5);
```

Explicación

Hemos creado seis usuarios (usuario0, usuario1, usuario2, usuario3, usuario4, usuario5) con las mismas características.

Cada usuario es un objeto que contiene las propiedades nombre, apellido, email, y direccion.

La dirección es otro objeto que contiene las propiedades municipio, calle, y numero. Finalmente, mostramos las características de cada usuario en la consola utilizando console.log.

Problema II

Crea otro código utilizando las características de los usuarios del Problema I, pero en esta ocasión agrega una funcionalidad, método recuperarClave, de la manera que se observa en la siguiente figura.

```
let user = {                          //objeto user
nombre: 'Paola',
apellido: 'Ortiz',
email: 'paola@company.ru',
direccion: {                          // objeto direccion dentro del ojeto
municipio: 'Jocotenango',
calle: 'Calle ancha',
numero: 25,
},
estado: true,                         //podemos asignar valores boolean
recuperarClave: function () {   //podemos asignar funciones, en este
console.log('Recuperar clave...')
}
```

Despliega en consola a los 6 usuarios creados, observa que ahora si hay un orden y existe coherencia entre las propiedades y métodos. El resultado será la creación de 6 usuarios, pero en esta ocasión diremos que son 6 objetos.

¿Qué diferencias conceptuales observas entre el Problema I y el Problema II? ¿Para crear usuarios es más fácil y coherente la manera del Problema I o la manera en que se crean en el Problema II?

Solución:

```
// Problema II: Usuarios con funcionalidad recuperarClave

let usuario0 = {
    nombre: 'Paola',
    apellido: 'Ortiz',
    email: 'paola@company.ru',
    direccion: {
        municipio: 'Jocotenango',
        calle: 'Calle ancha',
        numero: 25
    },
    estado: true,
    recuperarClave: function() {
        console.log(`Recuperar clave para ${this.nombre} ${this.apellido}`);
    }
};
```

```
let usuario1 = {
    nombre: 'Paolo',
    apellido: 'Ortega',
    email: 'paolo@company.ru',
    direccion: {
        municipio: 'Jocotenango',
        calle: 'Calle ancha',
        numero: 25
    },
    estado: true,
    recuperarClave: function() {
        console.log(`Recuperar clave para ${this.nombre} ${this.apellido}`);
    }
};

let usuario2 = {
    nombre: 'Pedro',
    apellido: 'Pérez',
    email: 'pedro@company.ru',
    direccion: {
        municipio: 'Jocotenango',
        calle: 'Calle ancha',
        numero: 25
    },
    estado: true,
    recuperarClave: function() {
        console.log(`Recuperar clave para ${this.nombre} ${this.apellido}`);
    }
};

let usuario3 = {
    nombre: 'Pablo',
    apellido: 'Ramírez',
    email: 'pablo@company.ru',
    direccion: {
        municipio: 'Jocotenango',
        calle: 'Calle ancha',
        numero: 25
    },
    estado: true,
    recuperarClave: function() {
        console.log(`Recuperar clave para ${this.nombre} ${this.apellido}`);
    }
};

let usuario4 = {
    nombre: 'Patricia',
    apellido: 'Rodríguez',
```

```
    email: 'patricia@company.ru',
    direccion: {
        municipio: 'Jocotenango',
        calle: 'Calle ancha',
        numero: 25
    },
    estado: true,
    recuperarClave: function() {
        console.log(`Recuperar clave para ${this.nombre} ${this.apellido}`);
    }
};

let usuario5 = {
    nombre: 'Pamela',
    apellido: 'Reyes',
    email: 'pamela@company.ru',
    direccion: {
        municipio: 'Jocotenango',
        calle: 'Calle ancha',
        numero: 25
    },
    estado: true,
    recuperarClave: function() {
        console.log(`Recuperar clave para ${this.nombre} ${this.apellido}`);
    }
};

// Mostrar las características de los usuarios y ejecutar el método recuperarClave
console.log(usuario0);
usuario0.recuperarClave();

console.log(usuario1);
usuario1.recuperarClave();

console.log(usuario2);
usuario2.recuperarClave();

console.log(usuario3);
usuario3.recuperarClave();

console.log(usuario4);
usuario4.recuperarClave();

console.log(usuario5);
usuario5.recuperarClave();
```

Diferencias conceptuales entre Problema I y Problema II

Estructura y Organización:

Problema I: Los usuarios están representados únicamente como objetos con propiedades, sin métodos. Es una manera sencilla y directa de almacenar datos.

Problema II: Los usuarios no solo tienen propiedades, sino también un método (recuperarClave). Esto hace que los objetos sean más completos y puedan realizar acciones.

Funcionalidad:

Problema I: Los usuarios solo contienen datos estáticos. No hay ninguna funcionalidad o comportamiento asociado a ellos.

Problema II: Los usuarios pueden realizar acciones a través de métodos. En este caso, el método recuperarClave permite a cada usuario mostrar un mensaje en la consola, lo que añade comportamiento a los datos.

Reusabilidad y Mantenimiento:

Problema I: Si necesitamos cambiar o agregar una nueva funcionalidad a los usuarios, tenemos que modificar cada objeto individualmente.

Problema II: Al usar métodos, podemos fácilmente modificar la funcionalidad o agregar nuevas capacidades sin tener que cambiar la estructura de datos básica de cada objeto.

¿Cuál manera es más fácil y coherente?
La manera del Problema II es más coherente y escalable para la creación de usuarios. Al agregar métodos a los objetos, no solo almacenamos datos, sino que también definimos comportamientos asociados a esos datos, lo cual es un enfoque más completo y alineado con los principios de la programación orientada a objetos. Esto facilita el mantenimiento y la extensión del código, ya que las funcionalidades pueden ser modificadas o extendidas sin necesidad de cambios drásticos en la estructura de los datos.

Problema III

Crea otro código utilizando el Problema II, pero ahora agrega a
cada usuario (objeto) la propiedad dpi y el método cambiarDireccion.
Utiliza el ejemplo 05-Objetos/02-dinamico.js como guía para
solucionar este problema.
Muestra a cada usuario en la consola.

```
const user = { id: 1};  // a las variables declaradas como const no es posible asignarle otro valor, pero...

user.name = 'Paola';

console.log('Soy el objeto user, con la nueva propiedad', user);

user.leerPropiedad_name = function () { // recuerda que esta función es llamda anónima y hace referencia a .leerPropiedad_name
   console.log('Leyendo la  propiedad name: ', user.name);
}

user.leerPropiedad_name();
console.log(user);

console.log('Ahora cambiaremos cosas del objeto user');
```

Solución:

```
// Problema III: Usuarios con propiedad dpi y método cambiarDireccion

class Usuario {
   constructor(nombre, apellido, email, password, dpi) {
      this.nombre = nombre;
      this.apellido = apellido;
      this.email = email;
      this.password = password;
      this.dpi = dpi;
      this.direccion = {
         municipio: 'Jocotenango',
         calle: 'Calle ancha',
         numero: 25
      };
      this.estado = true;
   }

   recuperarClave() {
      console.log(`Recuperar clave para ${this.nombre} ${this.apellido}`);
```

```
    }

    cambiarDireccion(nuevaDireccion) {
        this.direccion = nuevaDireccion;
        console.log(`La dirección de ${this.nombre} ${this.apellido} ha sido cambiada a
${JSON.stringify(this.direccion)}`);
    }
}

// Crear los usuarios con la nueva propiedad y método
let usuario0 = new Usuario('Paola', 'Ortiz', 'paola@company.ru', '1234', '1234567890');
let usuario1 = new Usuario('Paolo', 'Ortega', 'paolo@company.ru', '5678', '0987654321');
let usuario2 = new Usuario('Pedro', 'Pérez', 'pedro@company.ru', 'abcd', '1122334455');
let usuario3 = new Usuario('Pablo', 'Ramírez', 'pablo@company.ru', 'efgh', '5566778899');
let usuario4 = new Usuario('Patricia', 'Rodríguez', 'patricia@company.ru', 'ijkl', '6677889900');
let usuario5 = new Usuario('Pamela', 'Reyes', 'pamela@company.ru', 'mnop', '7788990011');

// Mostrar las características de los usuarios y ejecutar los métodos
console.log(usuario0);
usuario0.recuperarClave();
usuario0.cambiarDireccion({ municipio: 'Antigua', calle: 'Calle nueva', numero: 10 });

console.log(usuario1);
usuario1.recuperarClave();
usuario1.cambiarDireccion({ municipio: 'Antigua', calle: 'Calle nueva', numero: 11 });

console.log(usuario2);
usuario2.recuperarClave();
usuario2.cambiarDireccion({ municipio: 'Antigua', calle: 'Calle nueva', numero: 12 });

console.log(usuario3);
usuario3.recuperarClave();
usuario3.cambiarDireccion({ municipio: 'Antigua', calle: 'Calle nueva', numero: 13 });

console.log(usuario4);
usuario4.recuperarClave();
usuario4.cambiarDireccion({ municipio: 'Antigua', calle: 'Calle nueva', numero: 14 });

console.log(usuario5);
usuario5.recuperarClave();
usuario5.cambiarDireccion({ municipio: 'Antigua', calle: 'Calle nueva', numero: 15 });
```

Explicación

Clase Usuario:

Extendemos la clase Usuario desde el Problema II, añadiendo la propiedad dpi y el método cambiarDireccion.

El método recuperarClave sigue igual.

El método cambiarDireccion toma una nueva dirección como parámetro y actualiza la propiedad direccion.

Creación de Usuarios:

Creamos seis instancias de la clase Usuario, añadiendo el dpi como nueva propiedad.

Mostrar y Actualizar:

Mostramos las características de cada usuario en la consola usando console.log.
Ejecutamos el método recuperarClave para cada usuario.
Cambiamos la dirección de cada usuario usando el método cambiarDireccion y mostramos el resultado en la consola.

Problema IV
Utilizando las propiedades y métodos agrupados en cada usuario creado en el Problema III, vuelve a crear esos usuarios (objetos) pero en esta ocasión crea los usuarios (objetos) utilizando una factory function.
Puedes guiarte en el ejemplo 05-Objetos/03-factoryFunction.js.
Muestra a los usuarios en la consola.

Solución:

```
// Factory function para crear usuarios
function createUser(nombre, apellido, email, password, dpi) {
    return {
        nombre: nombre,
        apellido: apellido,
        email: email,
        password: password,
        dpi: dpi,
        direccion: {
            municipio: 'Jocotenango',
            calle: 'Calle ancha',
            numero: 25
        },
        estado: true,
        recuperarClave: function() {
            console.log(`Recuperar clave para ${this.nombre} ${this.apellido}`);
        },
        cambiarDireccion: function(nuevaDireccion) {
            this.direccion = nuevaDireccion;
            console.log(`La dirección de ${this.nombre} ${this.apellido} ha sido cambiada a
${JSON.stringify(this.direccion)}`);
        }
    };
}

// Crear los usuarios usando la factory function
let usuario0 = createUser('Paola', 'Ortiz', 'paola@company.ru', '1234', '1234567890');
let usuario1 = createUser('Paolo', 'Ortega', 'paolo@company.ru', '5678', '0987654321');
let usuario2 = createUser('Pedro', 'Pérez', 'pedro@company.ru', 'abcd', '1122334455');
let usuario3 = createUser('Pablo', 'Ramírez', 'pablo@company.ru', 'efgh', '5566778899');
let usuario4 = createUser('Patricia', 'Rodríguez', 'patricia@company.ru', 'ijkl', '6677889900');
let usuario5 = createUser('Pamela', 'Reyes', 'pamela@company.ru', 'mnop', '7788990011');

// Mostrar las características de los usuarios y ejecutar los métodos
console.log(usuario0);
usuario0.recuperarClave();
usuario0.cambiarDireccion({ municipio: 'Antigua', calle: 'Calle nueva', numero: 10 });
```

```
console.log(usuario1);
usuario1.recuperarClave();
usuario1.cambiarDireccion({ municipio: 'Antigua', calle: 'Calle nueva', numero: 11 });

console.log(usuario2);
usuario2.recuperarClave();
usuario2.cambiarDireccion({ municipio: 'Antigua', calle: 'Calle nueva', numero: 12 });

console.log(usuario3);
usuario3.recuperarClave();
usuario3.cambiarDireccion({ municipio: 'Antigua', calle: 'Calle nueva', numero: 13 });

console.log(usuario4);
usuario4.recuperarClave();
usuario4.cambiarDireccion({ municipio: 'Antigua', calle: 'Calle nueva', numero: 14 });

console.log(usuario5);
usuario5.recuperarClave();
usuario5.cambiarDireccion({ municipio: 'Antigua', calle: 'Calle nueva', numero: 15 });
```

Explicación

Factory Function:

La función createUser toma los parámetros necesarios (nombre, apellido, email, password, dpi) y devuelve un objeto con las propiedades y métodos definidos.
Este objeto incluye el método recuperarClave y el método cambiarDireccion, similar a los objetos creados en Problema III.

Creación de Usuarios:

Utilizamos la factory function createUser para crear seis instancias de usuarios con las propiedades y métodos requeridos.

Mostrar y Actualizar:

Mostramos las características de cada usuario en la consola usando console.log.
Ejecutamos el método recuperarClave para cada usuario.
Cambiamos la dirección de cada usuario usando el método cambiarDireccion y mostramos el resultado en la consola.