Promesas, async y await en Javascript

Promesas

Las promesas son una forma de manejar tareas asíncronas en JavaScript. Una promesa es un objeto que representa el resultado de una tarea que puede tardar algún tiempo en completarse.

Las promesas tienen tres estados posibles:

- Pendiente: La tarea aún no se ha completado.
- Cumplida: La tarea se ha completado satisfactoriamente.
- Rechazada: La tarea ha fallado y no se ha completado satisfactoriamente.

Las promesas se utilizan para realizar tareas asíncronas y devolver un resultado cuando se completa la tarea. En lugar de bloquear el hilo de ejecución mientras espera a que se complete la tarea, la promesa se devolverá inmediatamente y se ejecutará una vez que se complete la tarea.

Ejemplo de cómo se puede utilizar una promesa en Javascript:

```javascript
const promesa = new Promise((resolve, reject) => {
  // Hacer alguna tarea asíncrona, como una petición HTTP
  // Cuando se complete la tarea, llamar a resolve() si es exitoso o a reject() si falla.
});

// Luego se puede utilizar el método then() para manejar la promesa cumplida
promesa.then((resultado) => {
  // Hacer algo con el resultado
}).catch((error) => {
  // Manejar el error
});
```

En el ejemplo anterior, se crea una promesa utilizando la función constructora Promise. Dentro de la función constructora, se realiza una tarea asíncrona y se llama a resolve() si la tarea es exitosa o a reject() si falla.

Después de crear la promesa, se puede utilizar el método then() para manejar el resultado si la promesa se cumple. También se puede utilizar el método catch() para manejar los errores si la promesa es rechazada.

Async y Await en JavaScript

Async y Await son características de ES-2017 que permiten escribir código asíncrono en una forma más limpia y fácil de leer.

La palabra clave async se utiliza para definir una función asíncrona en JavaScript. Cuando se llama a una función asíncrona, devuelve una promesa. Dentro de una función asíncrona, se pueden utilizar las palabras clave await para esperar a que se complete una tarea asíncrona.

Ejemplo de cómo se puede utilizar async y await en JavaScript:

JavaScript

```
 //Definición de funciones

async function procesarPagosAsync() {
  const datosSonValidos = await validarDatosTarjetaAsync();
  // en este ejemplo solo utilizamos la variable datosSonValidos
para mostrar dos llamadas a funciones async pero podriamos usar
su resultado para hacer una validación.

  const nuevoSaldo = await actualizarSaldosAsync();

  return nuevoSaldo;
}

async function validarDatosTarjetaAsync() {
  console.log("validando datos...");
```

```
  //tarjeta OK
  return true;
}

async function actualizarSaldosAsync() {
  //Devolvemos el nuevo saldo (no importa el valor)
  return 100;
}

//Invocación
procesarPagosAsync()
  .then((saldo) => {
    //Usamos el nuevo saldo y lo imprimimos por consola. Notese
que "saldo" es una variable local
    console.log("el nuevo saldo es: $" + saldo);
  })
  .catch((error) => {
    // Manejar el error
  });
```

En el ejemplo anterior, se define una función asíncrona utilizando la palabra clave async. Dentro de la función, se utiliza la palabra clave await para esperar a que se complete una tarea asíncrona. La función devuelve una promesa que se puede manejar utilizando los métodos then() y catch().

Podemos reescribir la función usando async y await de la siguiente manera:

JavaScript

```
//Definición de funciones
```

```
async function procesarPagosAsync() {

  try {
    const datosSonValidos = await validarDatosTarjetaAsync();
    // en este ejemplo solo utilizamos la variable
datosSonValidos  para mostrar dos llamadas a funciones async pero
podriamos usar su resultado para hacer una validación.

    const nuevoSaldo = await actualizarSaldosAsync();

    return nuevoSaldo;
  } catch (error) {
     // Manejar el error
  }

}

async function validarDatosTarjetaAsync() {
  console.log("validando datos...");
  //tarjeta OK
  return true;
}

async function actualizarSaldosAsync() {
  //Devolvemos el nuevo saldo (no importa el valor)
  return 100;
}

//Invocación con async y await
const saldo = await procesarPagosAsync();
console.log("el nuevo saldo es: $" + saldo);
```

Lectura adicional:

https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Using_promises