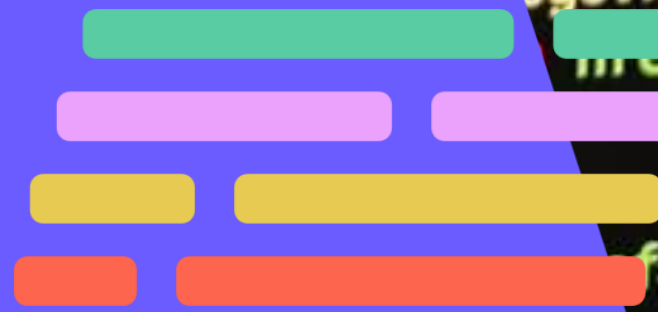
Somos un ecosistema de desarrolladores de software

</Riwi>

Java Script ASYNC- AWAIT



ASYNC- AWAIT

Promesas en Grupo

Mediante la API Promise nativa de Javascript se puede realizar operaciones con grupos de promesas, tanto independientes como dependientes entre sí.

El tiempo que requiere una tarea asíncrona no es fijo por lo que muchas veces el orden en que se resuelven no será el mismo. Cuando las tareas tienen una dependencia interna deben tenerse en cuenta en grupo, y no de forma individual.

Las promesas grupales pueden tener algún condicional como por ejemplo que se cumplan todas o al menos una. A continuación veremos las diferentes posibilidades.

ASYNC- AWAIT

Promesas en Grupo

Métodos	Definición
`Promise.all(list)`	Acepta sólo si **todas** las promesas del se cumplen.
`Promise.allSettled(list)`	Acepta sólo si **todas** las promesas del se cumplen o rechazan.
`Promise.any(list)`	Acepta con el valor de la primera promesa del que se cumpla.
`Promise.race(list)`	Acepta o rechaza según la primera promesa del que se procese.
`Promise.resolve(value)`	Devuelve una promesa cumplida directamente con el dado.
`Promise.reject(value)`	Devuelve una promesa rechazada directamente con el dado.

ASYNC- AWAIT

Promesas en Grupo

Para los siguientes métodos, se trabajará con 3 promesas que usarán setTimeout.

```
function operation1() {
    return new Promise(resolve => {
        setTimeout(() => {
            console.log("Operation 1 completed");resolve("Result of Operation 1");
        }, 1000);
    });
}
```

```
function operation2() {
    return new Promise(resolve => {
        setTimeout(() => {
            console.log("Operation 2 completed");resolve("Result of Operation 2");
        }, 1500);
    });
}
```

```
function operation3() {
    return new Promise(resolve => {
        setTimeout(() => {
            console.log("Operation 3 completed");resolve("Result of Operation 3");
        }, 2000);
    });
}
```

ASYNC- AWAIT

Promesas en Grupo - Promise.all()

A Promise.all() le pasamos un Array con las promesas individuales. Cuando todas y cada una de esas promesas se cumplan favorablemente, entonces se ejecutará la función callback de su .then(). En el caso de que alguna se rechace, no se llegará a ejecutar.

También se podría realizar utilizando .then() en lugar de async/await.

```
function performMultipleOperations() {
    console.log("Start multiple operations");

    const promise1 = operation1();
    const promise2 = operation2();
    const promise3 = operation3();

    Promise.all([promise1, promise2, promise3])
        .then(results => {
            console.log("All operations completed successfully");
            console.log("Results:", results);
        })
        .catch(error => {
            console.error("An error occurred:", error);
        });
}

performMultipleOperations();
```

```
async function performMultipleOperations() {
    console.log("Start multiple operations");

    try {
        const [result1, result2, result3] = await Promise.all([
            operation1(),
            operation2(),
            operation3()
        ]);

        console.log("All operations completed successfully");
        console.log("Results:", result1, result2, result3);
    } catch (error) {
        console.error("An error occurred:", error);
    }
}

performMultipleOperations();
```

ASYNC- AWAIT

Promesas en Grupo - Promise.allSettled()

Promise.allSettled() devuelve una promesa que se cumple cuando todas las promesas del Array se hayan procesado, independientemente de que se hayan cumplido o rechazado. Esta operación devuelve un Array de objetos (uno por cada promesa) donde cada objeto tiene dos propiedades:

La propiedad status, donde nos indica si cada promesa individual ha sido cumplida o rechazada.

La propiedad value, con los valores devueltos por la promesa si se cumple.
La propiedad reason, con la razón del rechazo de la promesa si no se cumple.

ASYNC- AWAIT

Promesas en Grupo - Promise.allSettled()

```
function performMultipleOperationsWithThen() {
    console.log("Start multiple operations with .then()");

    Promise.allSettled([operation1(), operation2(), operation3()])
        .then(results => {
            console.log("All operations settled");
            results.forEach(result => {
                if (result.status === "fulfilled") {
                    console.log("Operation fulfilled with result:", result.value);
                } else {
                    console.log("Operation rejected with reason:", result.reason.message);
                }
            });
        });
}

performMultipleOperationsWithThen();
```

```
async function performMultipleOperationsWithAsyncAwait() {
    console.log("Start multiple operations with async/await");

    try {
        const results = await Promise.allSettled([operation1(), operation2(), operation3()]);

        console.log("All operations settled");
        results.forEach(result => {
            if (result.status === "fulfilled") {
                console.log("Operation fulfilled with result:", result.value);
            } else {
                console.log("Operation rejected with reason:", result.reason.message);
            }
        });
    } catch (error) {
        console.error("An error occurred:", error);
    }
}

performMultipleOperationsWithAsyncAwait();
```

ASYNC- AWAIT

Promesas en Grupo - Promise.any()

El método Promise.any() devuelve una promesa con el valor de la primera promesa individual del Array que se cumpla. Si todas las promesas se rechazan, entonces devuelve una promesa rechazada.

```
function performAnyOperationWithThen() {
    console.log("Start any operation with .then()");

    Promise.any([operation1(), operation2(), operation3()])
        .then(result => {
            console.log("One operation fulfilled with result:", result);
        })
        .catch(error => {
            console.error("All operations rejected with reason:", error.message);
        });
}

performAnyOperationWithThen();
```

```
async function performAnyOperationWithAsyncAwait() {
    console.log("Start any operation with async/await");

    try {
        const result = await Promise.any([operation1(), operation2(),
operation3()]);
        console.log("One operation fulfilled with result:", result);
    } catch (error) {
        console.error("All operations rejected with reason:", error.message);
    }
}

performAnyOperationWithAsyncAwait();
```

ASYNC- AWAIT

Promesas en Grupo - Promise.race()

La primera promesa del Array que sea procesada, independientemente de que se haya cumplido o rechazado, determinará la devolución de la promesa del Promise.race(). Si se cumple, devuelve una promesa cumplida, en caso negativo, devuelve una rechazada. De forma muy similar a la anterior, Promise.race() devolverá la promesa que se resuelva primero, ya sea cumpliéndose o rechazándose.

```
function performRaceOperationWithThen() {
    console.log("Start race operation with .then()");

    Promise.race([operation1(), operation2(), operation3()])
        .then(result => {
            console.log("One operation fulfilled with result:", result);
        })
        .catch(error => {
            console.error("One operation rejected with reason:", error.message);
        });
}

performRaceOperationWithThen();
```

```
async function performRaceOperationWithAsyncAwait() {
    console.log("Start race operation with async/await");

    try {
        const result = await Promise.race([operation1(), operation2(),
operation3()]);
        console.log("One operation fulfilled with result:", result);
    } catch (error) {
        console.error("One operation rejected with reason:", error.message);
    }
}

performRaceOperationWithAsyncAwait();
```

</Be a coder>