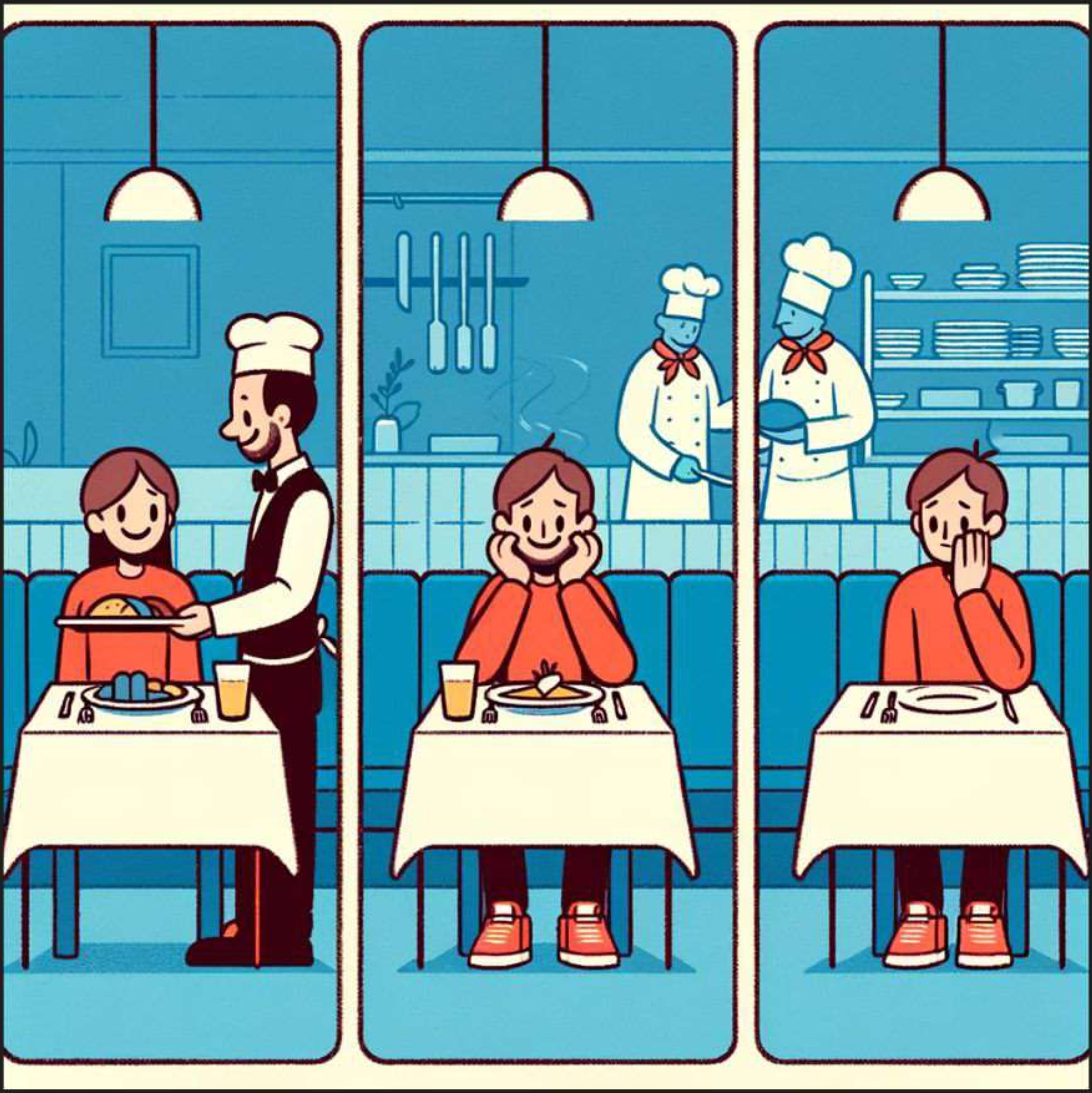
Promise

Explained to 5 year old

Chetan Mahajan
@_chetanmahajan



Fulfilled Rejected Pending

Imagine you're at a restaurant and you order a meal.

The waiter promises to bring your food after it's cooked.

This promise can end in three ways:

1. Fulfilled: You get your meal as ordered.
2. Rejected: The kitchen can't make your meal, so the waiter apologizes.
3. Pending: The kitchen is still preparing your meal.

In JavaScript, a Promise works similarly. It's a way to handle asynchronous operations, like requesting data from a server, where you don't get a response immediately.

Chetan Mahajan
@_chetanmahajan

Why Promises?

Before Promises, JavaScript used callbacks for asynchronous tasks.

However, callbacks could lead to nested, hard-to-read code, known as "callback hell".

Promises offer a cleaner, more manageable way to handle asynchronous code.

Chetan Mahajan
@_chetanmahajan

Where Used:

- Fetching data from APIs.
- Timers or delay functions.
- Reading files in Node.js.
- Any task that doesn't complete immediately.

Chetan Mahajan
@_chetanmahajan

Creating Promise

```
let promise = new Promise(function (resolve, reject) {
    // Asynchronous task here
    if (/* task successful */) {
        resolve(result);
    } else {
        reject(error);
    }
});
```

Handling Promise

```
promise
    .then(function (result) {
        // Handle the successful result
    })
    .catch(function (error) {
        // Handle the error or rejection
    });
```

Chetan Mahajan
@_chetanmahajan