Week- 2

Backend Engineering Launchpad

Pawan Panjwani

Airtribe

Backend Development Node JS Asynchronous programming

Airtribe

Agenda

- Paradigm of asynchronous programming
- Callbacks, Examples and problems with callbacks
- Promises, Examples, Promise chaining
- Error handling with promises
- Introduction to async await
- Error Handling with async await

Airtribe

“Node Js is a runtime environment for javascript that allows developers to write server sided applications using javascript

What is an Asynchronous programming?

- Asynchronous programming is a programming paradigm that allows code to execute non-blocking operations in parallel, without waiting for each operation to complete before moving on to the next.
- Asynchronous execution is possible in node Js with the help of event loop
- Different ways to write asynchronous code in Javascript - Callbacks, promises and async await

What are callbacks ?

- A callback is a function that is passed as an argument to another function and is executed when that function completes its task.

- Example 1: Reading a file asynchronously using callbacks

  Example 2: Making an HTTP request using callbacks

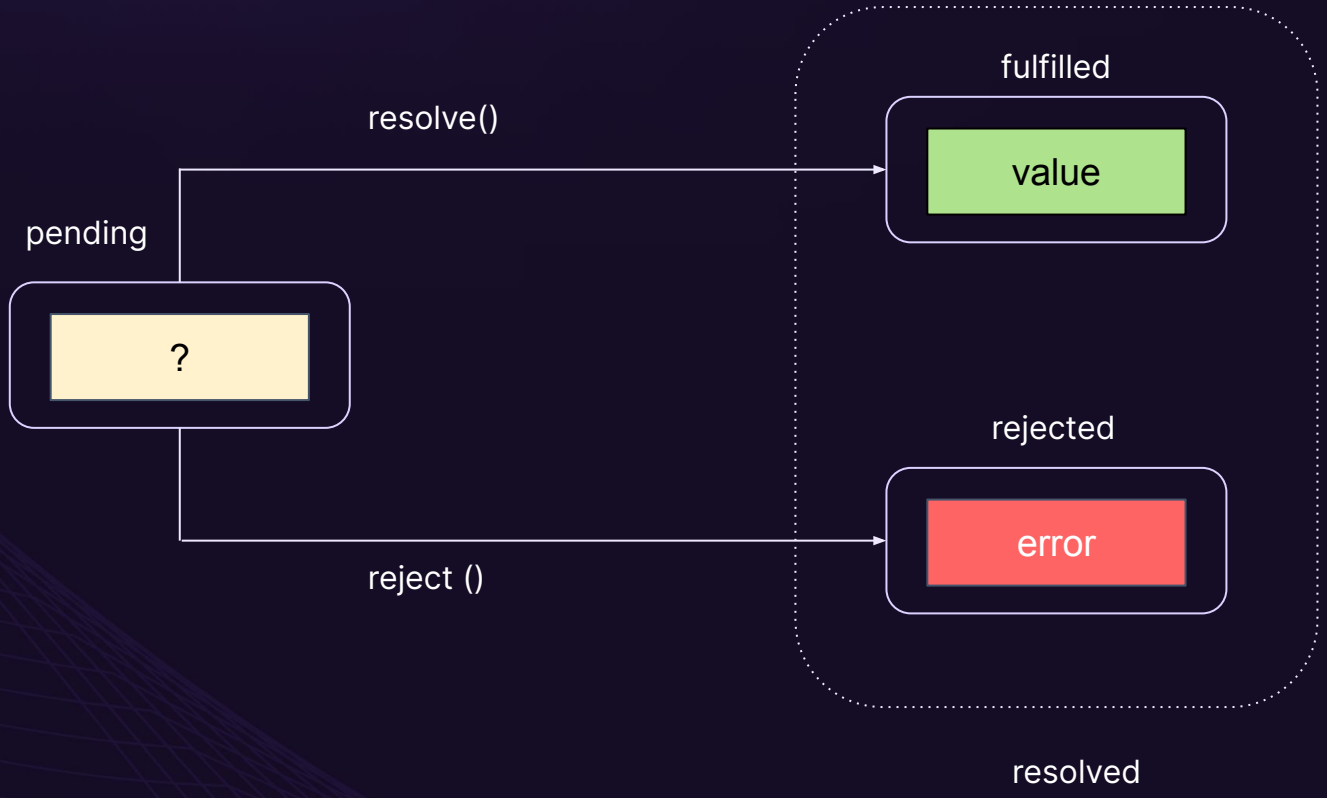
What is callback hell ?

- Callback hell: a situation where callbacks are nested inside one another, making the code difficult to read and maintain.
- It can occur when there are multiple asynchronous operations that need to be executed in sequence.
- Example code that reads a file, processes the data, and then makes an HTTP request, all using callbacks.

What are promises ?

- Promises are a way to handle asynchronous operations in JavaScript that provides a cleaner, more readable syntax than callbacks.
- They allow us to write cleaner, more readable code by avoiding deeply nested callbacks.
- A promise is created using the Promise constructor, which takes a function with two parameters: resolve and reject.
- Resolve is called when the promise is fulfilled, and reject is called when the promise is rejected.

Visualizing the state of promises !



Airtribe

What is async await?

- Async/await is a syntax for working with promises that allows us to write asynchronous code that looks like synchronous code.
- The async keyword is used to define a function that returns a promise, and the await keyword is used to wait for a promise to resolve before continuing execution.
- Async/Await is a newer way of handling asynchronous code in JavaScript introduced in ES6.It uses the async keyword to mark a function as asynchronous and the await keyword to wait for a Promise to resolve before continuing execution.
- Async/Await makes code more readable by avoiding nested callbacks and provides a more synchronous style of coding.

Visualizing callback hell!

```
1  // Callback Hell
2
3
4  a(function (resultsFromA) {
5      b(resultsFromA, function (resultsFromB) {
6          c(resultsFromB, function (resultsFromC) {
7              d(resultsFromC, function (resultsFromD) {
8                  e(resultsFromD, function (resultsFromE) {
9                      f(resultsFromE, function (resultsFromF) {
10                         console.log(resultsFromF);
11                     })
12                 })
13             })
14         })
15     })
16 });
17
```

Airtribe

Using promises as solution to callback hell !

```
1   // promises
2
3
4   a.then(resultsFromA => {
5     b(resultsFromA).then(resultsFromB => {
6       c(resultsFromB).then(resultsFromC => {
7         d(resultsFromC).then(resultsFromD => {
8           e(resultsFromD).then(resultsFromE => {
9             f(resultsFromE).then(resultsFromF => {
10                console.log(resultsFromF);
11            });
12          });
13        });
14      });
15    });
16  });
```

Airtribe

Simplifying the code using async await

```
1   // async await
2
3   async function getResults() {
4     let resultsFromA = await a();
5     let resultsFromB = await b(resultsFromA);
6     let resultsFromC = await c(resultsFromB);
7     let resultsFromD = await d(resultsFromC);
8     let resultsFromE = await e(resultsFromD);
9     let resultsFromF = await f(resultsFromE);
10    console.log(resultsFromF);
11  }
12
13  getResults()
```

Airtribe

Time for some action

Let’s look at callbacks, callback hell, promises, promise chaining, async await and error handling.

Thank you

Airtribe