Aysnc Architecture in js

Async thread: larger problem divided into Process and then divided into chunks and chunks are divided into threads.

Asyncio vs threading: **Async runs one block of code at a time while threading just one line of code at a time**. With async, we have better control of when the execution is given to other block of code but we have to release the execution ourselves.

Asynchronous Behavior

It means that **it doesn't wait for the response of an API call, I/O events, etc., and can continue the code execution**. It is possible thanks to the JS engines which use (under the hood) real multi-threading languages, like C++ (Chrome) or Rust (Firefox)

What is Thread-Based Asynchronous Programming? The thread-based asynchronous programming approach, **allows one thread pool to hand over a task to another thread pool i.e a worker thread pool, and then get notified to handle the result when the worker thread pool is done with the task**.

**async function**

The **async function** declaration declares an async function where the await keyword is permitted within the function body. The async and await keywords enable asynchronous, promise-based behavior to be written in a cleaner style, avoiding the need to explicitly configure promise chains.

**await**

The **await** operator is used to wait for a [Promise](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise) and get its fulfillment value. It can only be used inside an [async function](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function) or at the top level of a [module](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules).

await is usually used to unwrap promises by passing a [Promise](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise) as the expression. Using await pauses the execution of its surrounding async function until the promise is settled (that is, fulfilled or rejected). When execution resumes, the value of the await expression becomes that of the fulfilled promise.

If the promise is rejected, the await expression throws the rejected value. The function containing the await expression will [appear in the stack trace](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/await#improving_stack_trace) of the error. Otherwise, if the rejected promise is not awaited or is immediately returned, the caller function will not appear in the stack trace.

What is **Promise-based** in JavaScript?

A promise is **an object that may produce a single value sometime in the future** : either a resolved value, or a reason that it's not resolved (e.g., a network error occurred). A promise may be in one of 3 possible states: fulfilled, rejected, or pending.

function resolveAfter2Seconds() {

return new Promise(resolve => {

setTimeout(() => {

resolve('resolved');

}, 2000);

});

}

async function asyncCall() {

console.log('calling');

const result = await resolveAfter2Seconds();

console.log(result);

// Expected output: "resolved"

}

asyncCall();

//calling//after2 sec //resolved

Lifo :

Fifo :

Protocol: set of rule for networking communication. A communication protocol is a system of rules that allows two or more entities of a communications system to transmit information via any kind of variation of a physical quantity. The protocol defines the rules, syntax, semantics and synchronization of communication and possible error recovery methods.

1. Pooling
2. Callback
3. Pub-sub

Cookies:

Async code:

By asynce and await keywords or by already async libraries/functions such as filesyestem lib in js to read a file asynchronously.

Filesystem in js (npm install fs and const file\_system = require(“fs”);

url(uniform resource locater – unique ident) vs uri vs ip address

callback chaining: calling multiple nested ascyn functions with nested callbacks

promise.js(.then) : to get

Postman

API: (API is Set of Protocol or Way to communicate or access functionality for cross platforms) Application Programing Interface that make us able to communicate, data retrieval, functionality add in our website or to communicate with server securely and conveniently without breaking down the security or restricts bare metal access. It also creates a bridge between frontend and backend. We will Code API in Express.js.

fetch('http://example.com/movies.json')

.then((response) => response.json())

.then((data) => console.log(data));

JSON: JavaScript Object Notation to share data over the browser or internet in the meaningful way.

Promises vs Callbacks

Use any Rapid API