TYPESCRIPT TINKER

A Beginner's Journey into Typed JavaScript

PRESENTED BY
Microsoft Learn Student Club

22nd - 23rd June, 2023

Asynchronous JavaScript

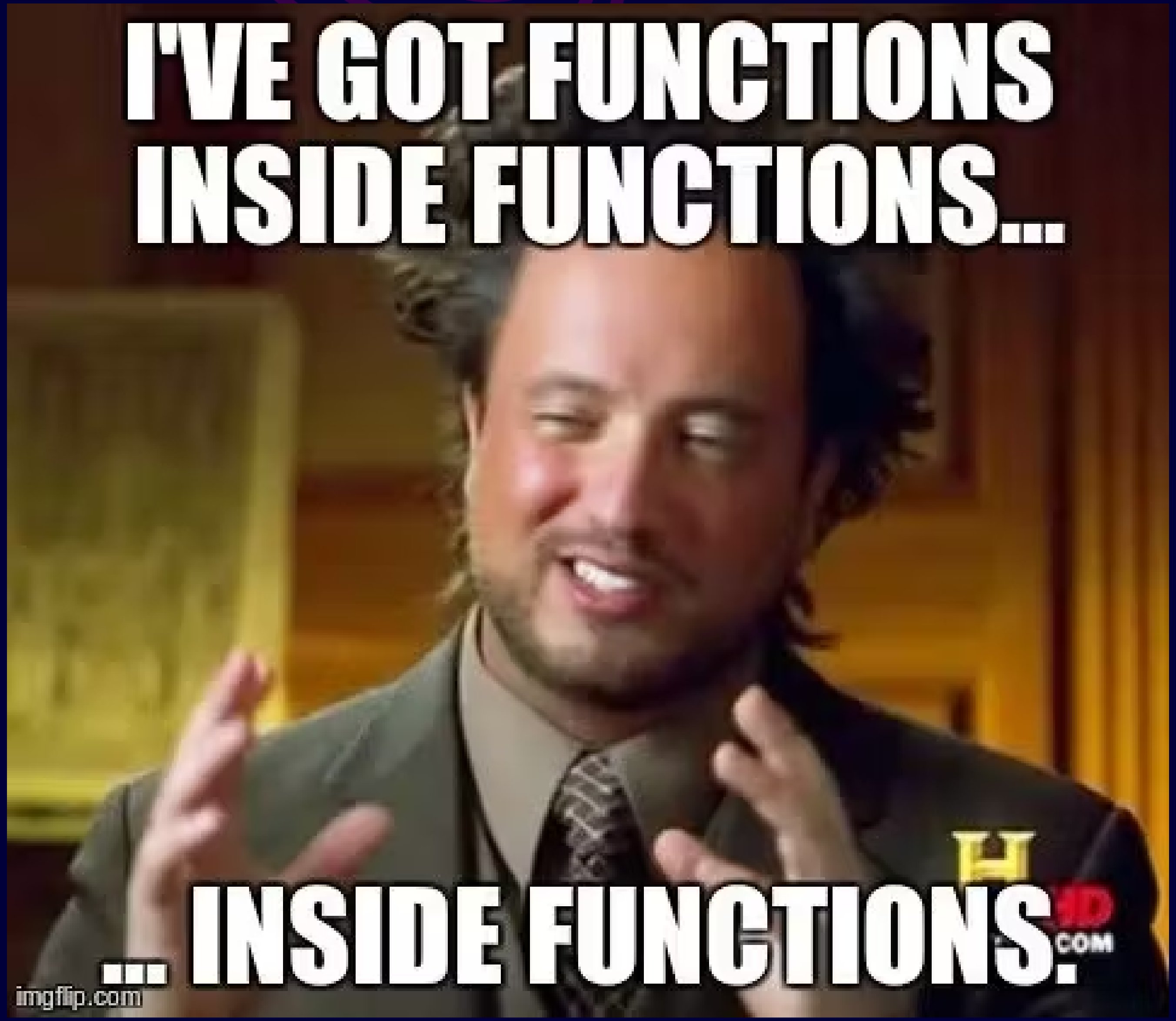
What are Synchronous and Asynchronous languages?

Synchronous languages execute tasks sequentially, one after another, while asynchronous languages allow concurrent execution and don't wait for a task to complete before moving on to the next one.

So to handle asynchronous code in JavaScript we use callbacks, promises and async/await functions.

Callbacks

Callbacks are functions that are passed as arguments to other functions and are executed at a later time or after a certain event occurs.



Some drawbacks of using callbacks in JavaScript

- Callback hell or Pyramid of Doom
- Lack of clarity and readability
- Error handling can be cumbersome
- Lack of control flow management

"CALLBACK HELL"
Promise & Async/Await

Features of ES6

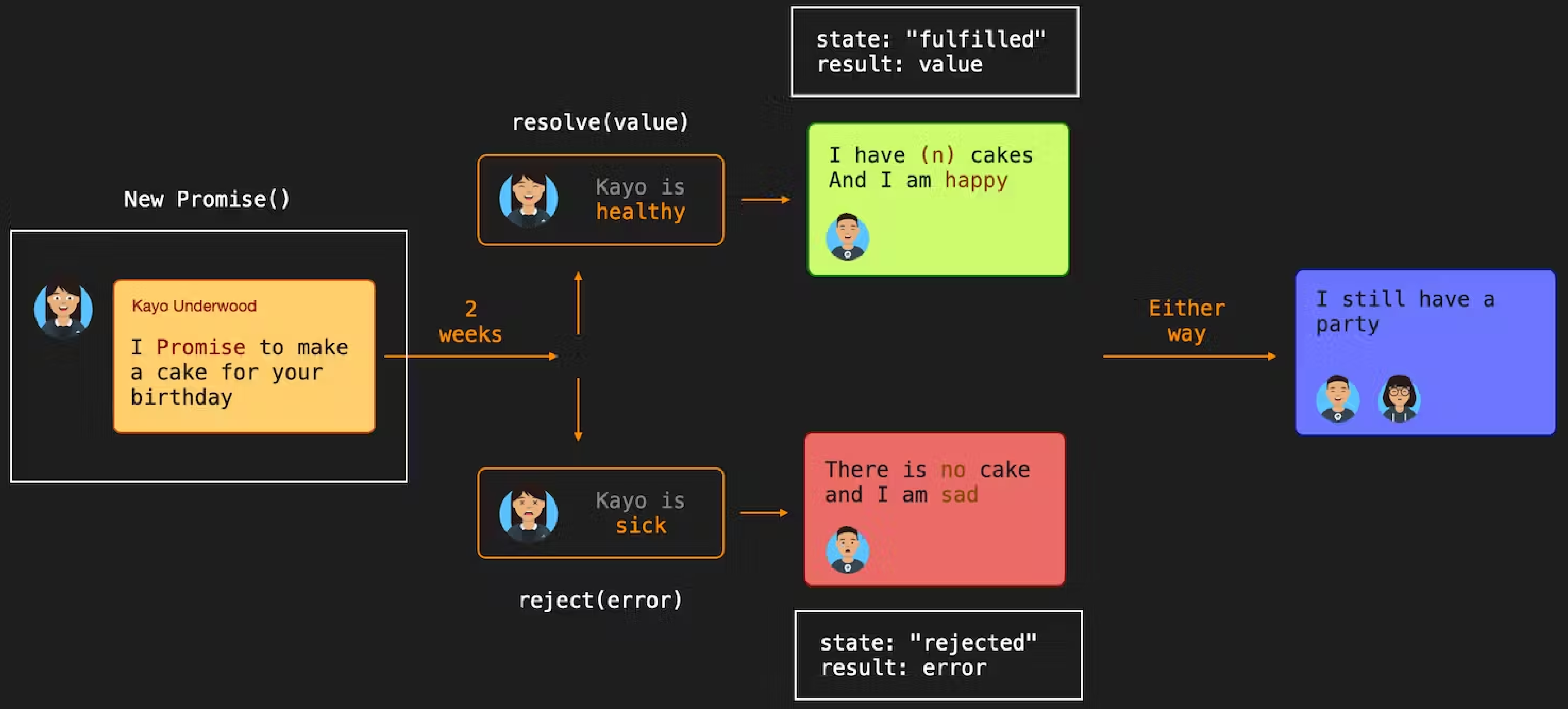
- Block-Scoped Variables
- Arrow Functions
- Classes and Modules
- Iterators and Generators
- Promises

Let's Learn about Promises

Promises

- Promises are objects used to handle asynchronous operations and represent the eventual completion (or failure) of an asynchronous task.

- Promises offer a cleaner, more organized method to create asynchronous code, making it simpler to manage and think through asynchronous activities.



Synchronous = happens at the same time. **Asynchronous** = doesn't happen at the time

Based on real-life scenario

I USED TO MAKE PROMISES
NOW I JUST AWAIT MY ASYNC FUNCTIONS

Async/Await Functions

- Async-await is syntactic sugar for Promises.
- Async/await is a newer syntax introduced in ECMAScript 2017 (ES8) that provides a more elegant way to write asynchronous code.
- Async - It ensures that the function returns a promise, and wraps non-promises in it.
- Await - It makes JavaScript wait until that promise settles and returns its result.

Despite these advantages, it's important to note that async/await is not a replacement for promises.

Async/await is built on top of promises and offers a more elegant syntax for working with promises. Promises still serve as the foundation for handling asynchronous operations in JavaScript and provide more fine-grained control in certain scenarios.