**ASYNCHRONOUS FUNCTION**

JavaScript calls the constructor() method every time it creates a new instance of a class.

The new keyword calls the constructor(), runs the code inside of it, and then returns the new instance.

async functions we should still take advantage of concurrency, the ability to perform asynchronous actions at the same time.

Note: if we have multiple truly independent promises that we would like to execute fully in parallel, we must use individual .then() functions and avoid halting our execution with await.

pass an array of promises as the argument to Promise.all(), and it will return a single promise. This promise will resolve when all of the promises in the argument array have resolved. This promise’s resolve value will be an array containing the resolved values of each promise from the argument array.

* async...await is syntactic sugar built on native JavaScript promises and generators.
* We declare an async function with the keyword async.
* Inside an async function we use the await operator to pause execution of our function until an asynchronous action completes and the awaited promise is no longer pending .
* await returns the resolved value of the awaited promise.
* We can write multiple await statements to produce code that reads like synchronous code.
* We use try...catch statements within our async functions for error handling.
* We should still take advantage of concurrency by writing async functions that allow asynchronous actions to happen in concurrently whenever possible.

Promise.all() is a good choice if multiple asynchronous tasks are all required, but none must wait for any other before executing.