* Asynchronous (async) code is what makes it possible for a JavaScript engine to do 2 things at the same time... or "asynchronously"
* We can use async code to allow the browser to keep executing code while something else is happening - usually a call to an external API for data
* When we use this technique, we create a helper object, called a "Promise", to inform the browser that the second, async task has finished

**Promises**

* A Promise is simply an object with a few properties
* When we want to run some async code, we create a new Promise, and use that Promise to inform the JavaScript engine that the async function has finished
* When we instantiate a new Promise with the "new" keyword, we pass in a callback function that receives a "resolve" function and a "reject" function
* If the async function finishes and was successful, we call the resolve() function. If it was unsuccessful, we call the reject() function
* When a Promise is resolved or rejected we use the Promise object's methods .then() or .catch() to tell the JavaScript engine what to do next

Here's the text content from the image about .then() and .catch():

**.then() & .catch()**

* .then() and .catch() are both methods on the Promise object that receive a callback function as an argument
* When our async function finishes running, that callback function is executed

**Let's see this all in code!**

Okay, here's the text content from the image showing a code example of Promises:

**Promises**

* The asyncFunction here creates and returns a new Promise;

JavaScript

const asyncFunction = () => {

return new Promise((resolve, reject) => {

// perform some async action

});

};

* This means that wherever we call this function, we can use .then() or .catch()

JavaScript

asyncFunction()

.then(() => {

console.log("async stuff finished");

})

.catch(() => {

console.log("async stuff rejected");

});