# await vs return vs return await

When writing async functions, there are differences between await vs return vs return await, and picking the right one is important.

Let's start with this async function:

| async function waitAndMaybeReject() { await new Promise((r) => setTimeout(r, 1000)); const isHeads = Boolean(Math.round(Math.random())); if (isHeads) return 'yay'; throw Error('Boo!');} |
| --- |

#### 

This returns a promise that waits a second, then has a 50/50 chance of fulfilling with "yay" or rejecting with an error. Let's use it in a few subtlety different ways:

## Just calling

| async function foo() { try { waitAndMaybeReject(); } catch (e) { return 'caught'; }} |
| --- |

Here, if you call foo, the returned promise will always be fulfilled with undefined, without waiting.

Since we don't await or return the result of waitAndMaybeReject(), we don't react to it in any way. Code like this is usually a mistake.

## Awaiting

| async function foo() { try { await waitAndMaybeReject(); } catch (e) { return 'caught'; }} |
| --- |

Here, if you call foo, the returned promise will always wait one second, then either fulfill with undefined, or fulfill with "caught".

Because we await the result of waitAndMaybeReject(), its rejection will be turned into a throw, and our catch block will execute. However, if waitAndMaybeReject() fulfills, we don't do anything with the value.

## Returning

| async function foo() { try { return waitAndMaybeReject(); } catch (e) { return 'caught'; }} |
| --- |

Here, if you call foo, the returned promise will always wait one second, then either fulfill with "yay", or reject with Error('Boo!').

By returning waitAndMaybeReject(), we're deferring to its result, so our catch block never runs.

## Return-awaiting

The thing you want in try/catch blocks, is return await:

| async function foo() { try { return await waitAndMaybeReject(); } catch (e) { return 'caught'; }} |
| --- |

Here, if you call foo, the returned promise will always wait one second, then either fulfill with "yay", or fulfill with "caught".

Because we await the result of waitAndMaybeReject(), its rejection will be turned into a throw, and our catch block will execute. If waitAndMaybeReject() fulfills, we return its result.

If the above seems confusing, it might be easier to think of it as two separate steps:

| async function foo() { try { const fulfilledValue = await waitAndMaybeReject(); return fulfilledValue; } catch (e) { return 'caught'; }} |
| --- |

Note: Outside of try/catch blocks, return await is redundant because it doesn't catch the error. There's even an ESLint rule to detect it, but it allows it in try/catch.