# Writing Asynchronous Code

Brice Wilson

# Overview

Why asynchronous code is important

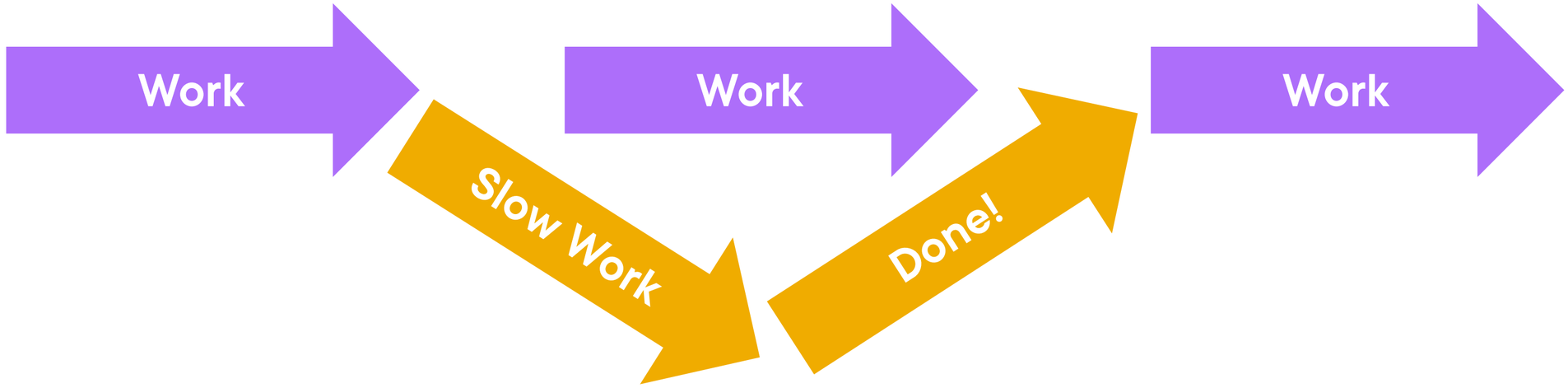
Promises

async/await

# Why Asynchronous Code Matters

## Synchronous Execution


## Asynchronous Execution



# Promise

**The Promise object represents the eventual completion (or failure) of an asynchronous operation and its resulting value.**

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise

# Promises

**Native support in ES2015**

- Requires TypeScript *--target* compiler option set to ES2015 or greater

**Small API**

- then
- catch

**Similar to Tasks in C#**

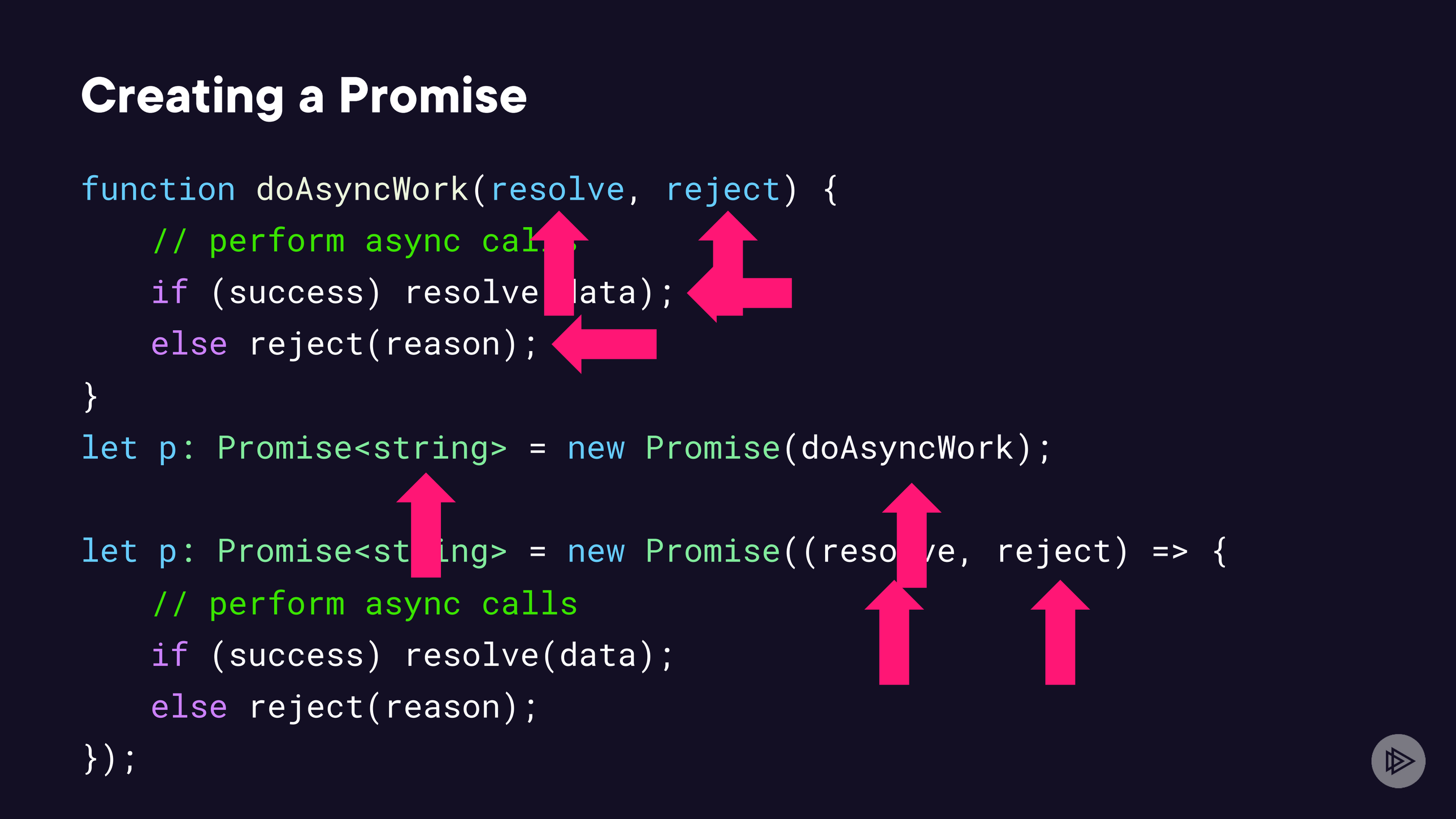
**May be chained together**

**Created by passing a function to the Promise constructor**

# Creating a Promise

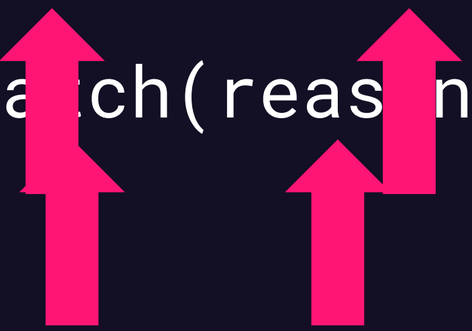
```
function doAsyncWork(resolve, reject) {
    // perform async calls
    if (success) resolve(data);
    else reject(reason);
}
let p: Promise<string> = new Promise(doAsyncWork);

let p: Promise<string> = new Promise((resolve, reject) => {
    // perform async calls
    if (success) resolve(data);
    else reject(reason);
});
```



# Handling Promise Results

```
let p: Promise<string> = MethodThatReturnsPromise();
p.then(stringData => console.log(stringData))
  .catch(reason => console.log(reason));
```



# Demo

## Creating and using promises

# async/await

Allows code to be written more linearly

Very similar to async/await in C#

Works with promises

# async/await Syntax

```
async function doAsyncWork() {
    let results = await GetDataFromServer();
    console.log(results);
}
console.log('Calling server to retrieve data...');
doAsyncWork();
console.log('Results will be displayed when ready...');
```

# Demo

## Writing asynchronous code with async/await

# Summary

Asynchronous code keeps your application responsive

Promises

async/await

Up Next:

# Creating and Using Generics