# Section 7 - Try & Catch

## Error Handling

In the previous section, we assumed that the value received in the setter is a valid string.



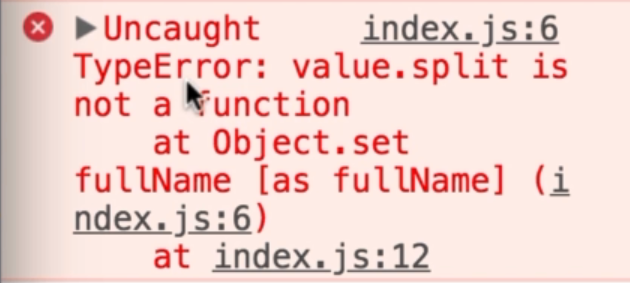
## Passing an invalid value

Here we attempt to pass a boolean value.



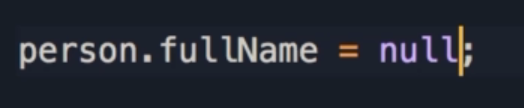
When we run this code, we get an error:

Uncaught TypeError: value.split is not a function



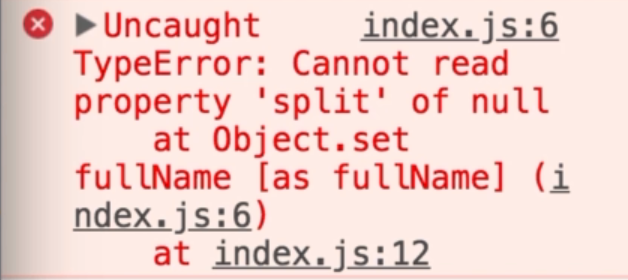
This is because split() is a method that belongs to strings, and booleans don’t have a split() method.

## Passing Null or Undefined



If we pass a value of null or undefined, we get a different error:

Uncaught TypeError: Cannot read property ‘split’ of null/undefined

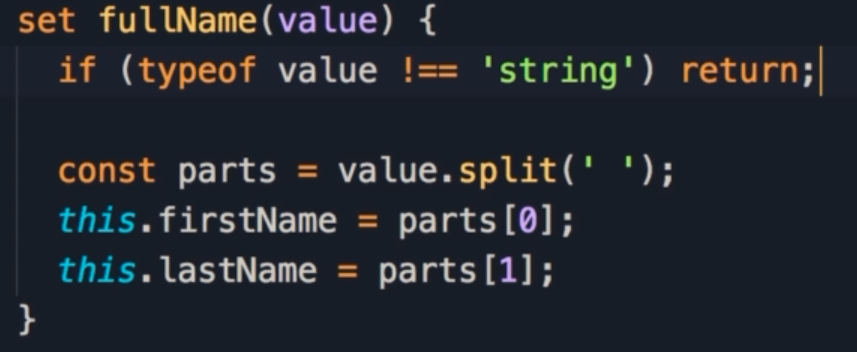


We cannot call the split method on null or undefined, as it's only available to strings.

In these cases, we would need to add error handling.

## Defensive Programming

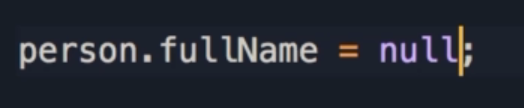
In situations like this, we should do error handling at the beginning of the function or method.



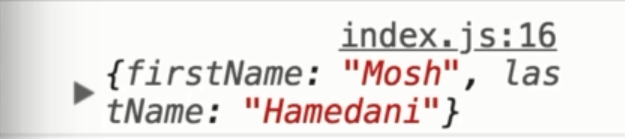
This is what we call defensive programming.

We want to make sure that the values coming in are valid, they're in the right shape, so we can execute our logic.

In the first line of this method, we use the typeof operator to check if the type of value is not equal to (!==) string.



When we pass a null object, it no longer displays a message, and the object does not change.



## Displaying Errors

Sometimes we want to report an error in our application.

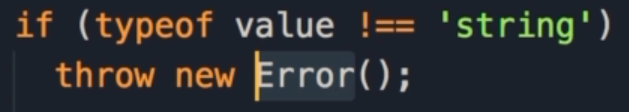
That’s when we need to **throw an exception**.

Throwing an exception is a technical jargon that you might hear in a lot of programming languages.

Let’s see how that works in JavaScript.

## Throwing an Exception

Instead of returning from this method, use the throw keyword and then create a new Error object.

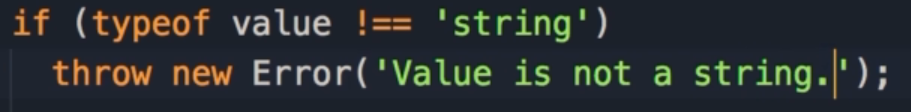


This Error object is a constructor function, as indicated by the use of Pascal Case.

When we are calling it, we are using the new keyword to create a new Error object.

As an argument to the Error constructor, we can pass a string as the error message:

‘Value is not a string’

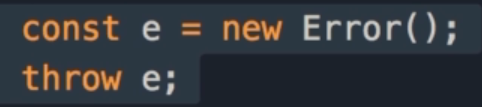


This is how we throw exceptions

## Errors & Throwing Exceptions

Some people confuse errors with exceptions, but there is a slight difference between the two.

First we can create an error object and assign it to a variable called e.



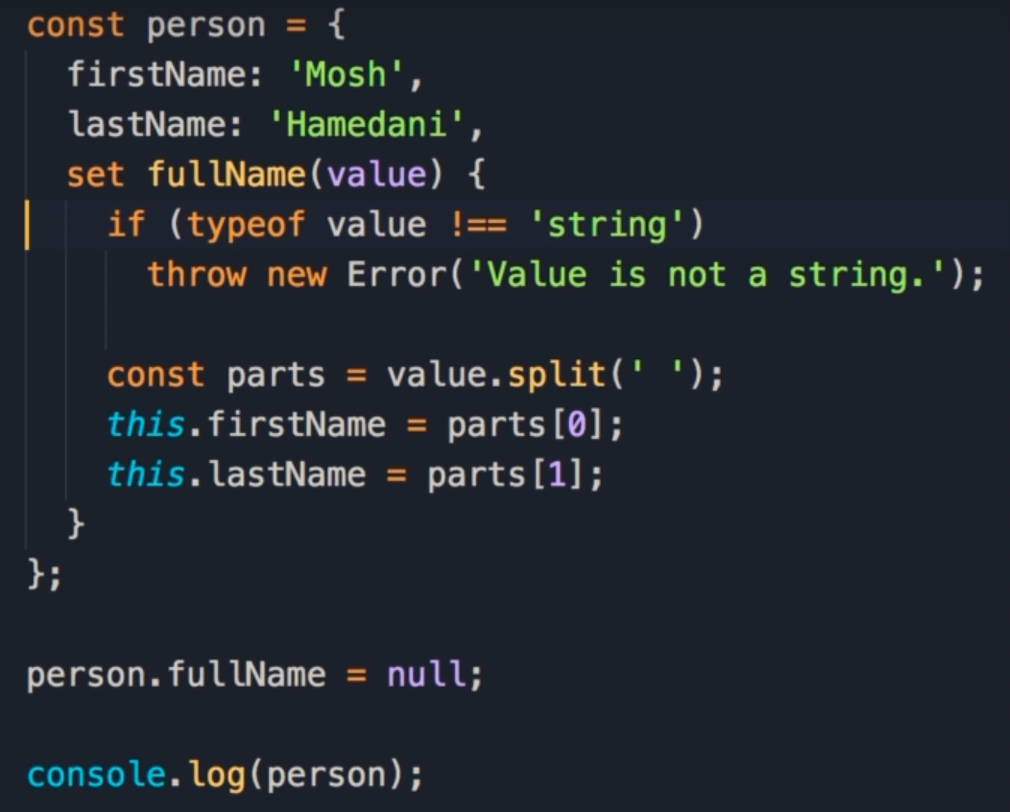
This is just a plain JavaScript object, there's nothing special about this.

But the moment you throw this error, we refer to that as an exception.

Because this is an exceptional situation that should not have happened.

## Error Handling

Now we have some basic error handling in this method, by throwing the exception.

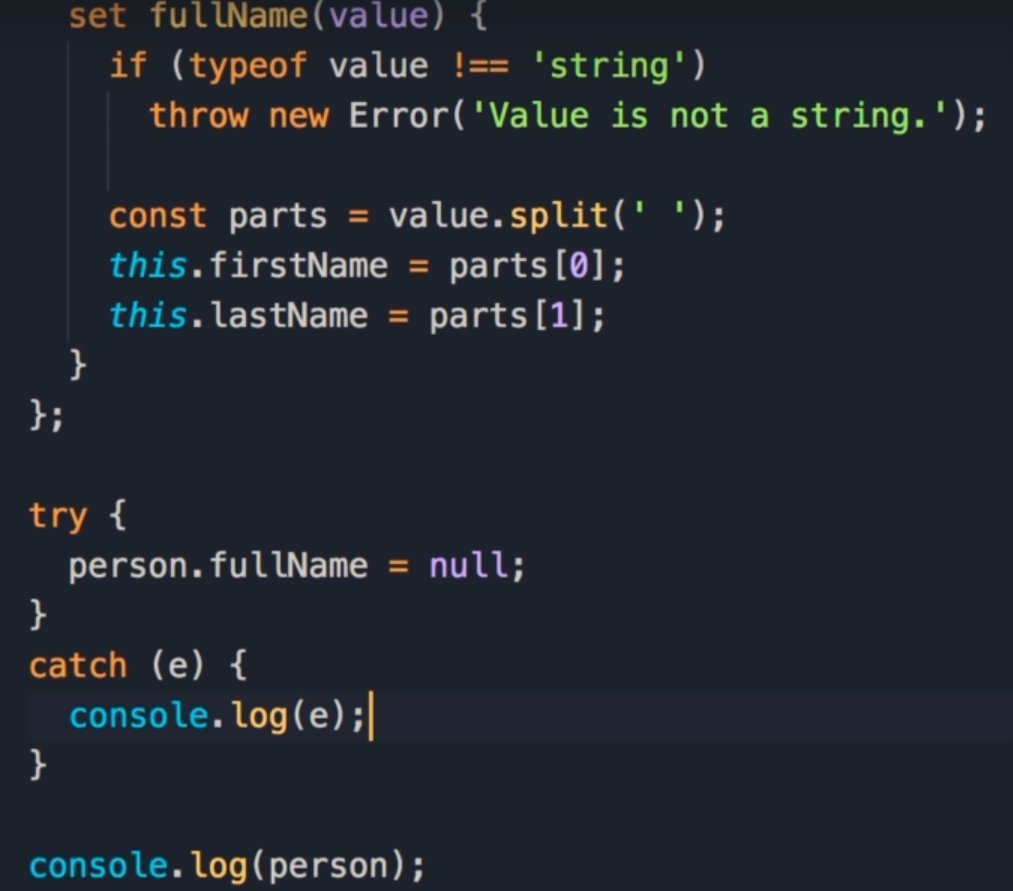


Now somewhere else we will need to catch that exception, and display the error.

## 

## Try & Catch

We get the error when assigning person.fullName, so we start by wrapping that in a try block.



A try block can accept 1 or more statements, with at least 1 being able to throw an exception.

Next we add the catch block, and within parentheses, we pass an identifier.

The identifier is the error object that is thrown in the setter.

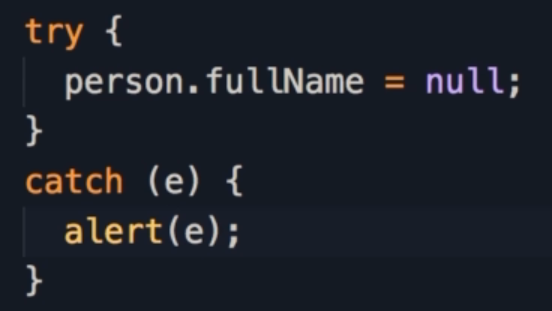
In the catch block, we get the error object and do something with it.

We can display it on the console, which is only visible to developers, so an end user will not see this error.

The proper way is to display a label, perhaps a red label on the user interface.

## Throwing the Exception

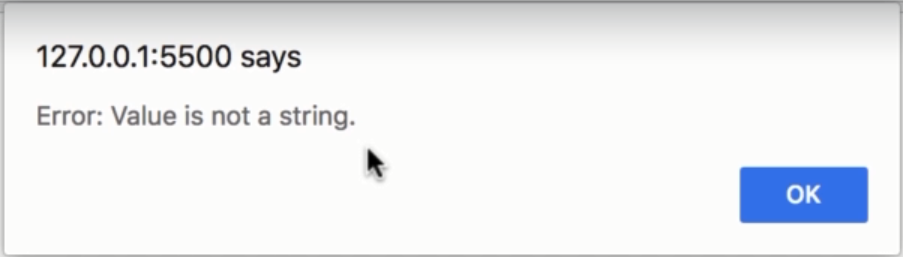
For now let's just use the built in alert function.



This is not something I'd recommend you to do, because that's a very old and poor way of reporting errors to users.

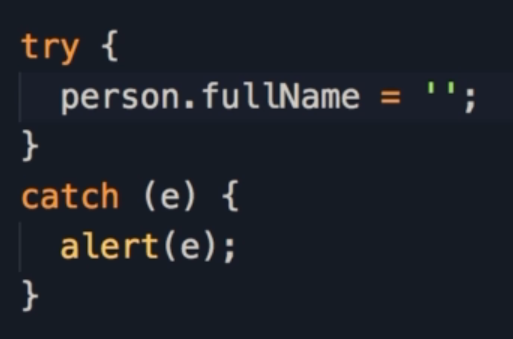
When you go to the browser, you should see an alert like this:

Error: Value is not a string



## More Error Handling

Instead of null, let’s pass in an empty string.

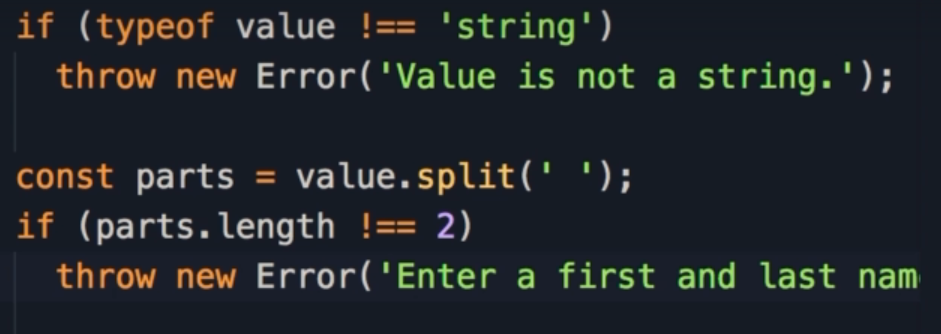


We don’t get an error, however firstName is set to an empty string, and lastName is undefined, which is not desirable.



Ideally we want to make sure our users enter a first name and a last name.

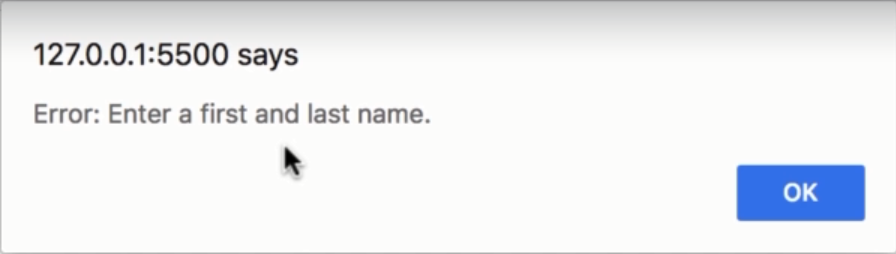
When splitting the string, we can check the length of the parts array created.



Here we check if parts.length is not equal to 2, then something must be missing.

Next we throw another exception, so we throw a new Error with the message:

Enter a first and last name.



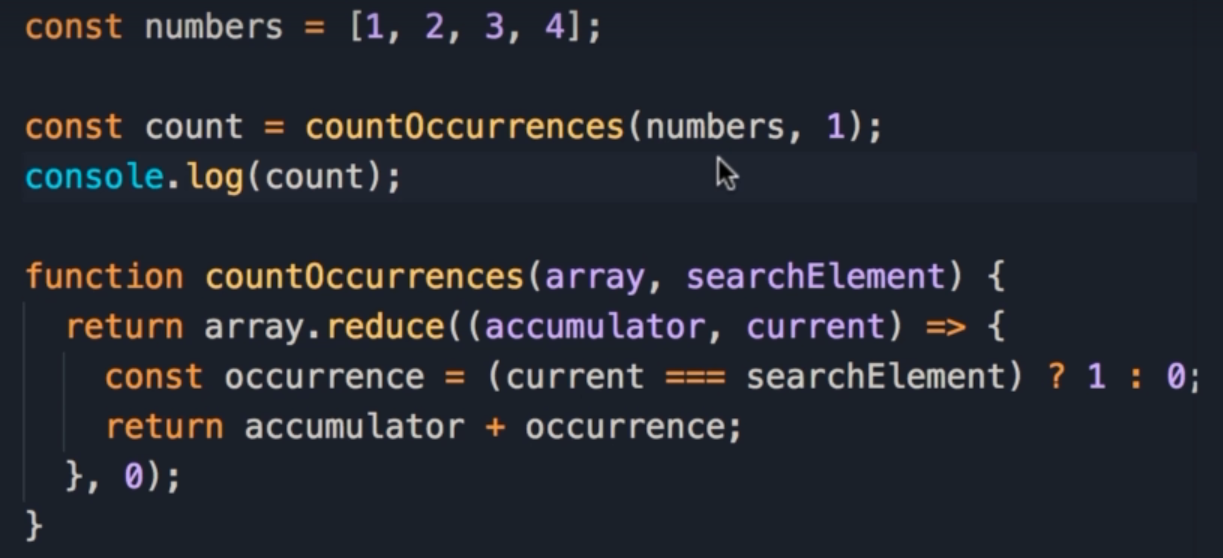
## Recap

When we throw an exception, the lines after the throw statement are not executed.

The throw keyword will jump out of the method, and control flow will move to the catch block, where we catch the exception and do something with it.

This is basic error handling in JavaScript.

# TASKS

1. Error Handling  
   In an earlier section, we did an exercise to count occurences of a number within an array  
     
     
   This function has a problem, we are assuming this first argument is a valid array. If we pass another value, like a boolean, we will get an UncaughtTypeError, because we are using the array.reduce() method, which is not available to booleans.  
     
   The purpose of this exercise is to add error handling to this function.
   1. If the first argument is not an array, throw an exception