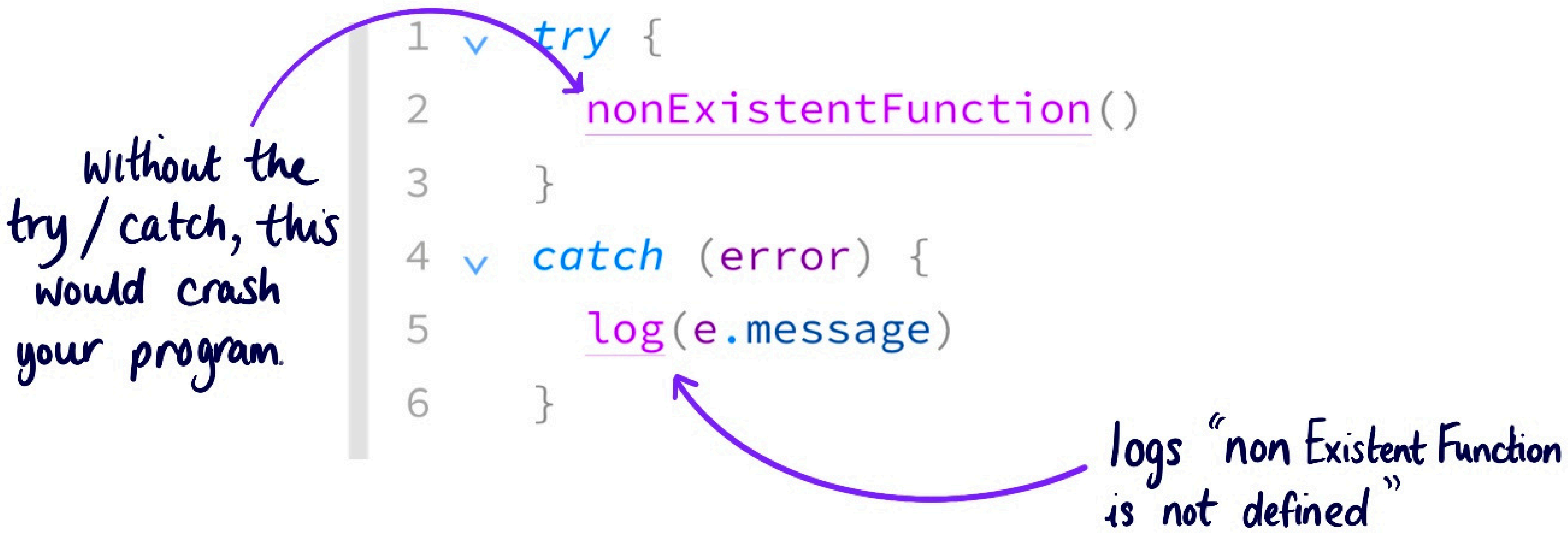
Exceptions

Exceptions are how we handle errors in programming. We can wrap code in a try/catch block to "catch" any errors that occur.



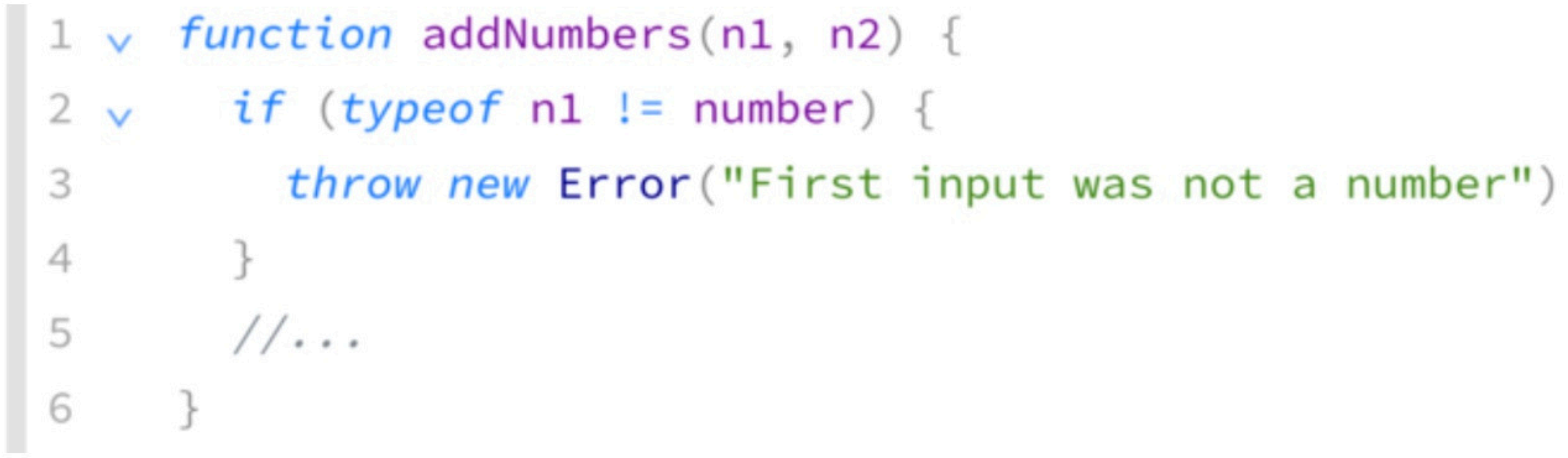
```
try {
    nonExistentFunction()
}
catch (error) {
    log(e.message)
}
```

without the try/catch, this would crash your program.

logs "non Existent Function is not defined"

Throwing Errors

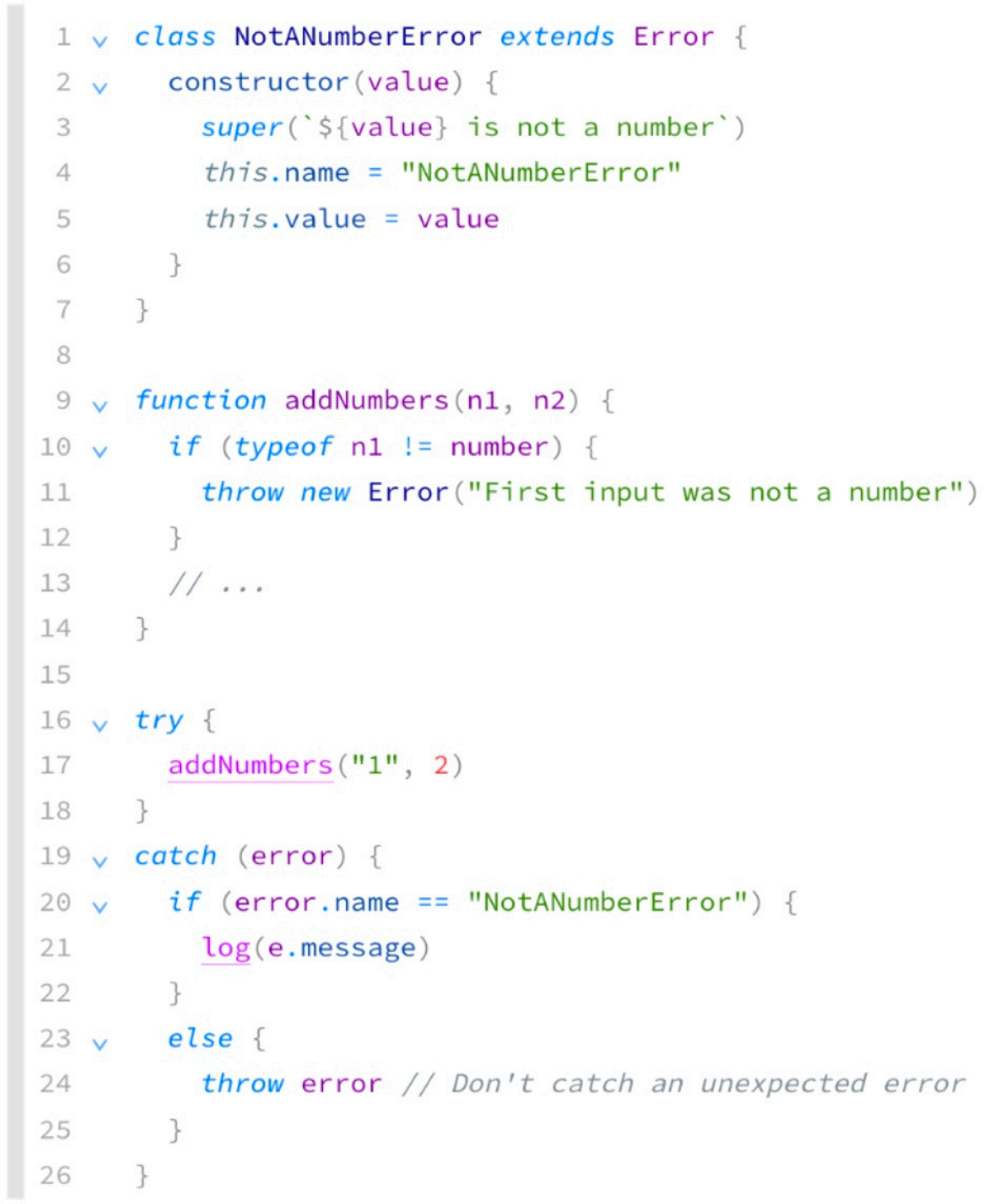
Sometimes we want to throw an error. For example, if an input to a function is null or we have a string when we expect a number. This means our program will have predicatable errors!



```
function addNumbers(n1, n2) {
  if (typeof n1 != number) {
    throw new Error("First input was not a number")
  }
  //...
}
```

Custom Errors

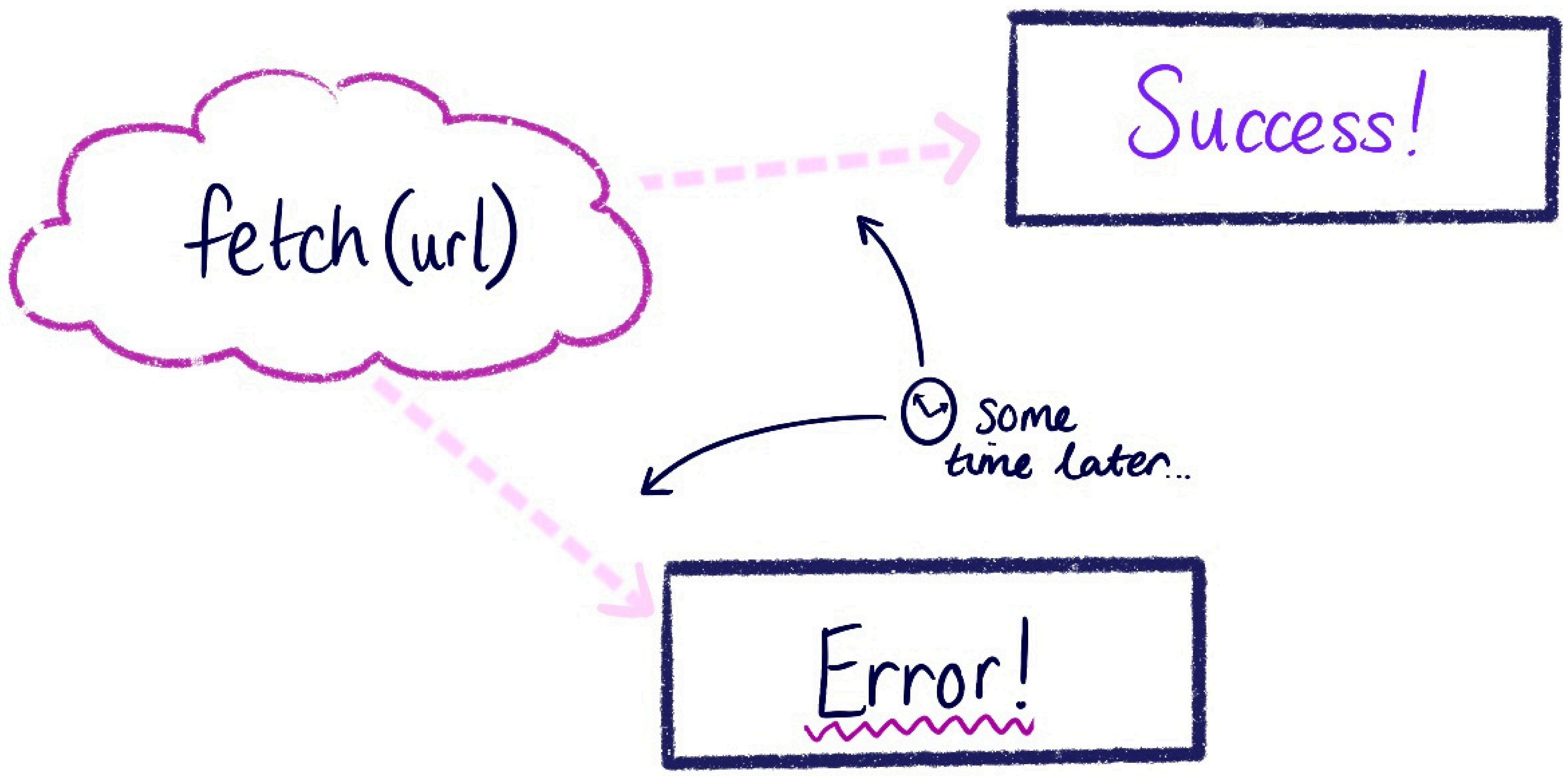
We can also create our own error types to capture more data and improve our ability to catch specific errors.



```javascript
class NotANumberError extends Error {
  constructor(value) {
    super(`${value} is not a number`)
    this.name = "NotANumberError"
    this.value = value
  }
}

function addNumbers(n1, n2) {
  if (typeof n1 != number) {
    throw new Error("First input was not a number")
  }
  // ...
}

try {
  addNumbers("1", 2)
}
catch (error) {
  if (error.name == "NotANumberError") {
    log(e.message)
  }
  else {
    throw error // Don't catch an unexpected error
  }
}
```

Promises

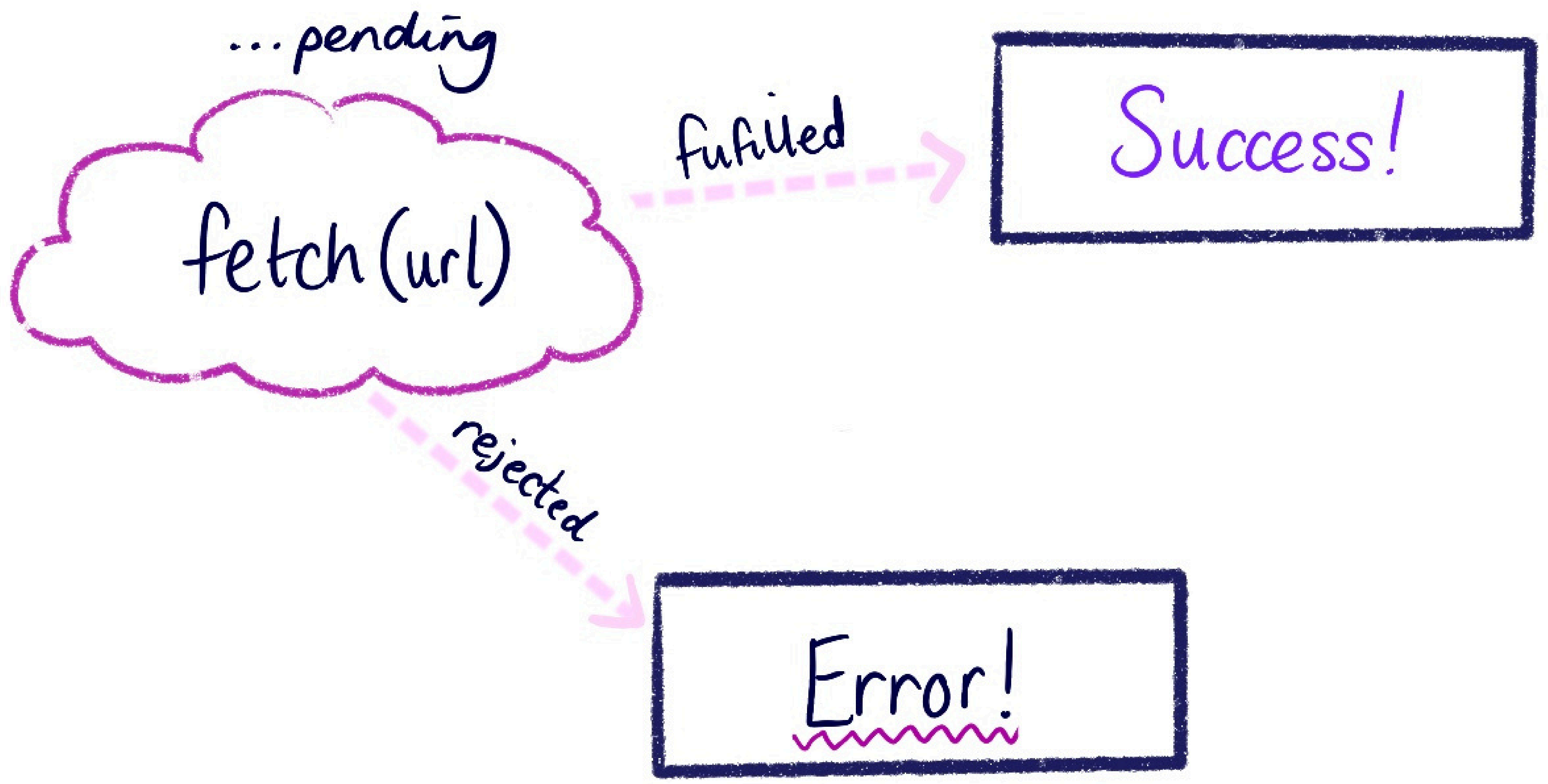
Promises are a way of dealing with asyncronous code, for example when fetching data from an API.



Promise States

A function returns a Promise when it wants to give you a way of dealing with its return value later.

A promise starts as "pending" and later becomes "fufilled" or "rejected".

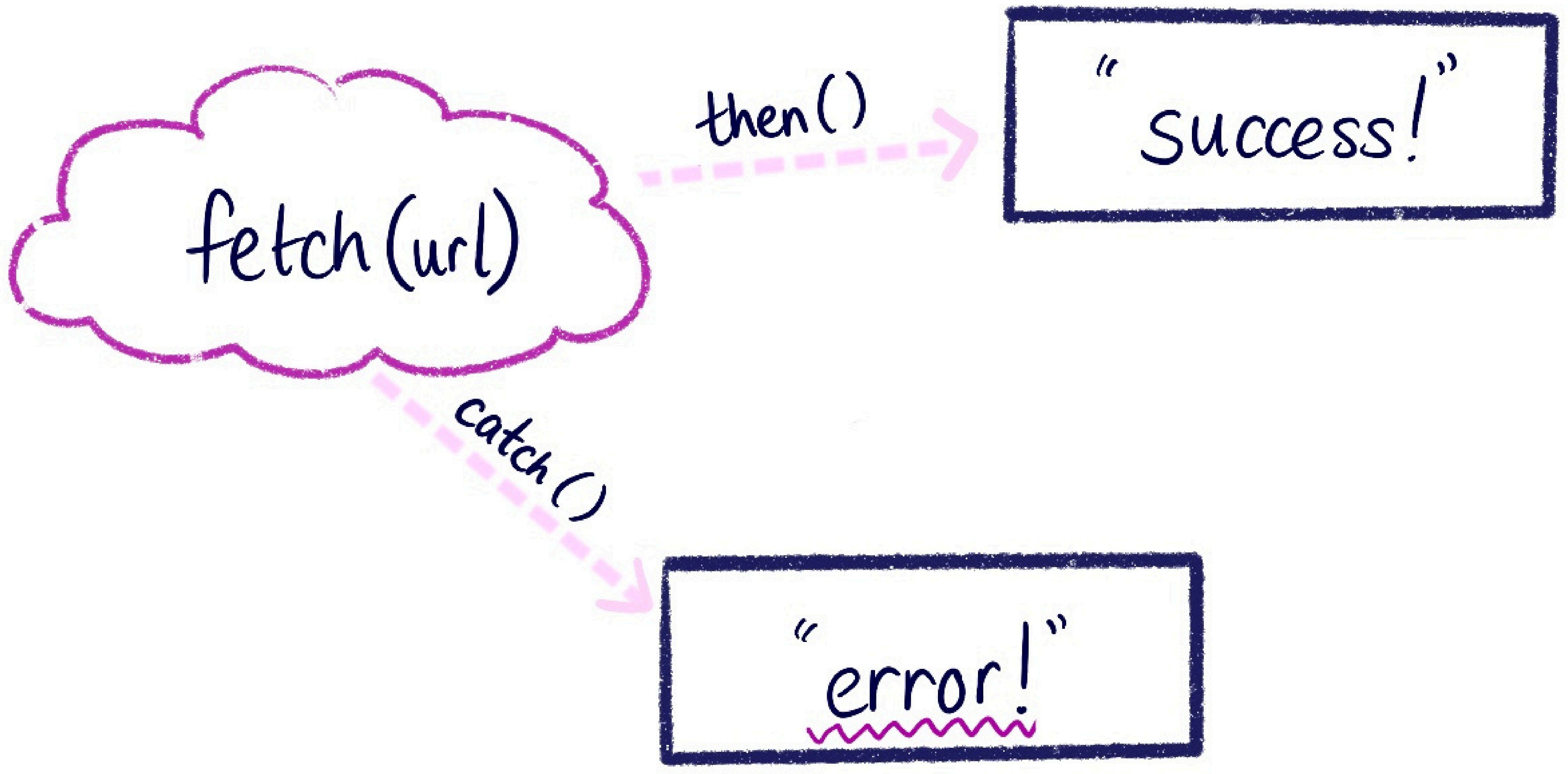


Once a promise changes to fufilled or rejected, it will never change state again.

then() and catch()

We can use promises with the then() and catch() methods.

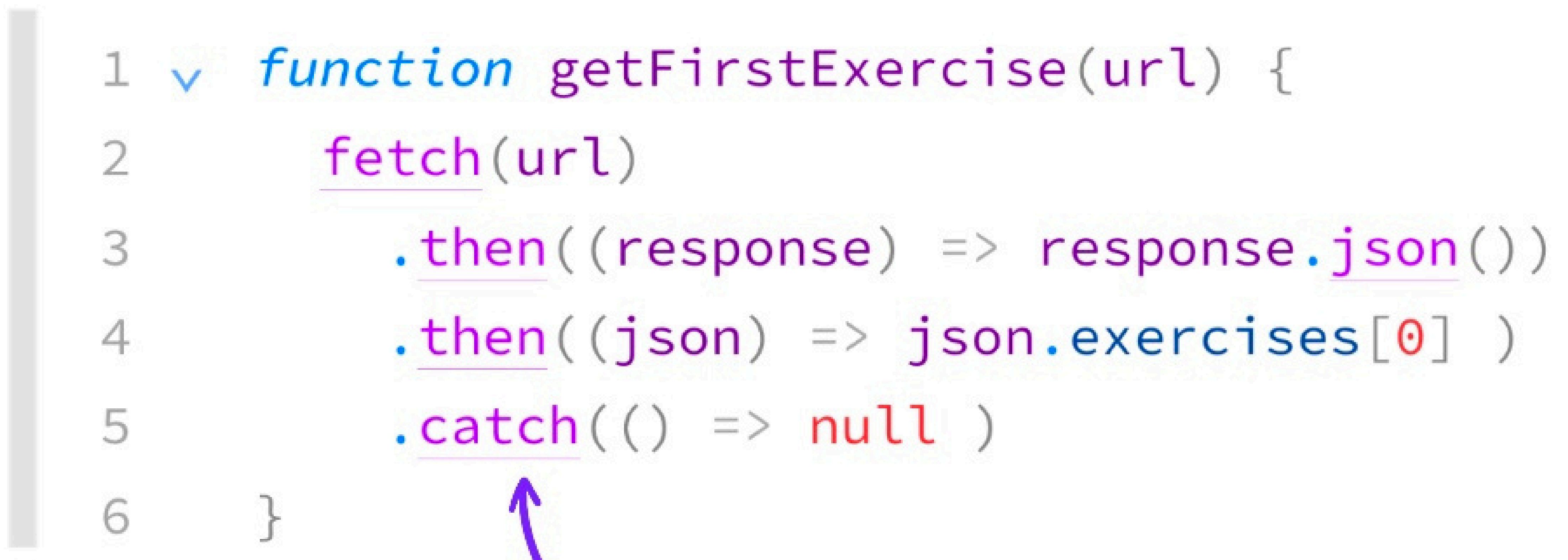
```
const promise = fetch(url)

promise.then(() => log("Success!"))
       .catch(() => log("Error!"))
```



Chaining "then()"

We can chain multiple promises together.

For example, if we want to get some data from a URL, then convert it to JS objects, we can do.



```
function getFirstExercise(url) {
  fetch(url)
    .then((response) => response.json())
    .then((json) => json.exercises[0] )
    .catch(() => null )
}
```

This catch will handle errors in the fetch() or the json() promises.