# Promises

# Promise

A Promise is an object representing the eventual completion or failure of an asynchronous operation.

# Create Promise

```
new Promise( /* executor */
    function(resolve, reject) { ...
}
);
```

# Executor

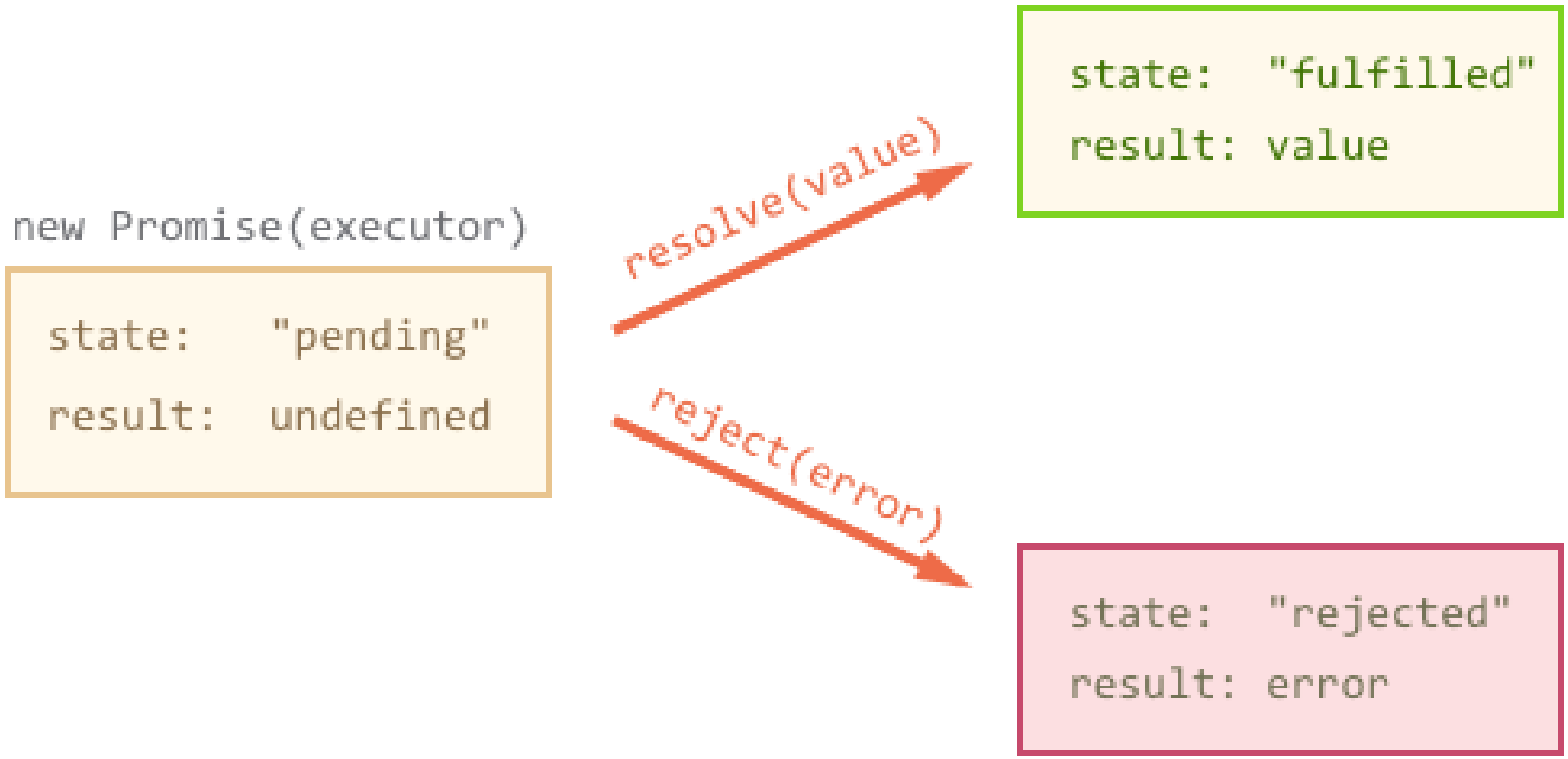
A function that

is passed with the arguments *resolve* and *reject*.
is executed immediately by the Promise implementation,
passing *resolve* and *reject* functions.

# Resolve/Reject

The *resolve* and reject *functions*, when called, resolve or reject the promise, respectively.

# Promise object



# Promise Tips

There can be only one result or an error. The executor should call only one *resolve* or *reject*.

The promise state change is final.

*resolve/reject* with more than one argument - only the first argument is used, the next ones are ignored.

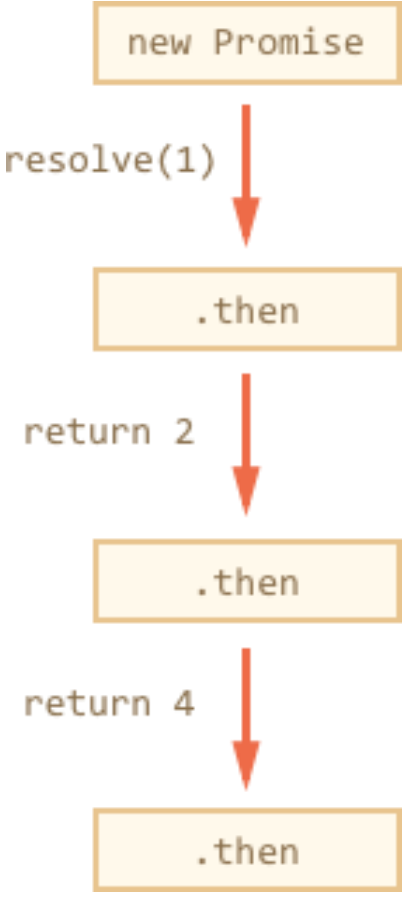
Use *Error* objects in *reject* (or inherit from them).

# Promise Tips

Properties state and result of a promise are internal. We can’t directly access them, but we can use methods *.then/catch* for that.

# .then and .catch

# Promises chaining



# Promises chaining