You Made Me Promises Promises

Jerry D’Antonio
@jerrydantonio
jerrydantonio.com

Concurrent Ruby

- Open source concurrency library
- Used by Rails, Sidekiq, Logstash,

  Microsoft Azure SDK for Ruby
- 0.0.1 - July 23, 2013
- 1.0.0 - November 13, 2015

Concurrent Ruby

- 50+ contributors from more than 20 countries on six continents
- 505,000+ downloads of 1.0.0
- http://concurrent-ruby.com

```
$.get( "test.php" ).then(
  function() {
    alert( "$.get succeeded" );
  }, function() {
    alert( "$.get failed!" );
  }
);
```

https://api.jquery.com/deferred.then/

The Plan

Promised Land

- Disambiguation
- What is a Promise?
- What's that really mean?
- States, fates, and values

The Plan

Show Me the Code!

- JavaScript
- Ruby
- JVM: Clojure, Scala & Java
- Tasks: C# & Elixir

Disambiguation

Synchronous & Asynchronous

"A synchronous operation **blocks** a process till the operation completes. An asynchronous operation is **non-blocking** and only initiates the operation."

http://www.cs.unc.edu/~dewan/242/s07/notes/ipc/node9.html

Synchronous

- Operations happen now
- Code positioned after the synchronous call occurs after the synchronous call
- Timing is **deterministic**

Asynchronous

- Operations happen whenever
- Code positioned after the synchronous may execute first
- Timing is **nondeterministic**

Concurrency & Parallelism

TL;DR Concurrency is **NOT** parallelism.

https://github.com/jdantonio/Everything-You-Know-About-the-GIL-is-Wrong-RubyConf-2015

Concurrency & Parallelism

Concurrency: "Programming as the **composition** of independently executing processes."

Parallelism: "Programming as the **simultaneous** execution of (possibly related) computations."

https://talks.golang.org/2012/waza.slide#1

Promise, Future, Delay & Defer

"The terms future, promise, and delay are often used interchangeably…"

https://en.wikipedia.org/wiki/Futures_and_promises

What is a Promise?

"In computer science, future, promise, and delay refer to **constructs** used for **synchronization** in some concurrent programming languages. They describe an object that acts as a **proxy** for a **result** that is **initially unknown**, usually because the computation of its value is yet incomplete."

https://en.wikipedia.org/wiki/Futures_and_promises

"A Promise is an object that is used as a **placeholder** for the **eventual results** of a deferred (and possibly asynchronous) **computation**."

https://github.com/domenic/promises-unwrapping

"A Promise is a container you get **immediately** for a value you get **eventually**."

What’s that really mean?

All The Things

- Do **this** thing then do **that** thing
- When each happens is irrelevant
- Synchronous, asynchronous, concurrent, and parallel are irrelevant
- So long as **order** is maintained

A Promise is a Contract

Promises allow us to **compose** our programs around **independent**, loosely coupled operations with **guarantees** that the order of operations will be maintained when it matters.

States, Fates, and Values

States

Three mutually exclusive states:

- **fulfilled** when complete and successful
- **rejected** when complete failed
- **pending** when not yet complete

Fates

Two mutually exclusive resolutions:

- **resolved** once the state becomes fulfilled or rejected
- **unresolved** while the state is still pending

Values

Promises may also have properties:

- When fulfilled the **value** represents the result of the operation
- When rejected the **reason** explains what went wrong

Show Me the Code

Characteristics

- Language or library?
- Concurrent or parallel?
- Callbacks or values?


JS

```
var promise = new Promise(function(resolve, reject) {
  // do a thing, possibly async, then…

  if (/* everything turned out fine */) {
    resolve("Stuff worked!");
  }
  else {
    reject(Error("It broke"));
  }
});
```

http://www.html5rocks.com/en/tutorials/es6/promises/

```
promise.then(function(result) {
   console.log(result);
}, function(err) {
   console.log(err);
});
```

http://www.html5rocks.com/en/tutorials/es6/promises/

```
var promise =
  new Promise(function(resolve, reject) {
  resolve(1);
});

promise.then(function(val) {
  console.log(val); // 1
  return val + 2;
}).then(function(val) {
  console.log(val); // 3
});
```

http://www.html5rocks.com/en/tutorials/es6/promises/

```
Promise.resolve("Success").then(function(value) {
  console.log(value); // "Success"
}, function(value) {
  // not called
});
```

http://www.html5rocks.com/en/tutorials/es6/promises/

JavaScript Promise

Language or library?	- ES6 has promises in the language - Prior to ES6 promises were provided by several libraries
Concurrent or parallel?	- The JavaScript event loop prevents true parallelism - But asynchronous I/O can provide pseudo parallelism
Callbacks or values?	- Callbacks only - Idiomatic usage is to pass high-order functions for success and failure


Ruby

```ruby
p = Concurrent::Promise.execute do
  "Hello, world!”
end
sleep(1)
```

```ruby
p.state          #=> :fulfilled
p.fulfilled?     #=> true
p.value          #=> "Hello, world!"
```

http://ruby-concurrency.github.io/concurrent-ruby/Concurrent/Promise.html

```
p = Concurrent::Promise.execute do
  raise StandardError
end
sleep(1)
```

```
p.state      #=> :rejected
p.rejected?  #=> true
p.reason     #=> StandardError
```

http://ruby-concurrency.github.io/concurrent-ruby/Concurrent/Promise.html

```
Concurrent::Promise.fulfill(20).
    then{|result| result - 10 }.
    then{|result| result * 3 }.
    then{|result| result %
5 }.execute
```

http://ruby-concurrency.github.io/concurrent-ruby/Concurrent/Promise.html

```ruby
promise = Concurrent::Promise.new do
  connection.post do |request|
    request.headers = request_headers
    request.body = request_content
    @client.credentials.sign_request(request) unless @client.credentials.nil?
  end
end

promise = promise.then do |http_response|
  status_code = http_response.status
  response_content = http_response.body
  unless (status_code == 200)
    error_model = JSON.load(response_content)
    fail MsRestAzure::AzureOperationError.new(connection, http_response, error_model)
  end
  # Create Result
  result = MsRestAzure::AzureOperationResponse.new(connection, http_response)
  result.request_id = http_response['x-ms-request-id'] unless ...
  # Deserialize Response
  if status_code == 200
    # important stuff ...
  end

  result
end

promise.execute
```

https://github.com/Azure/azure-sdk-for-ruby/blob/arm_netw-v0.1.0/resource_management/azure_mgmt_storage/lib/azure_mgmt_storage/storage_accounts.rb#L895

Ruby Promise

Language or library?	- Libraries - concurrent-ruby is most popular - http://concurrent-ruby.com
Concurrent or parallel?	- Concurrent on MRI, parallel on JRuby and Rubinius - Asynchronous I/O can provide pseudo parallelism on MRI
Callbacks or values?	- Both - Also supports chaining, fanning, error handler, and observation

```
;; Create a promise
user> (def p (promise))
#'user/p ; p is our promise

;; Check if was delivered/realized
user> (realized? p)
false ; No yet

;; Delivering the promise
user> (deliver p 42)
#<core$promise$reify__5727@47122d: 42>

;; Check again if it was delivered
user> (realized? p)
true ; Yes!

;; Deref to see what has been delivered
user> (deref p)
42
```

https://clojuredocs.org/clojure.core/promise

```
;; A future's calculation is started here
;; and it runs in another thread
user=> (def f
  (future
    (Thread/sleep 10000)
    (println “done")
    100))
#’user/f

;; if you wait 10 seconds before
;; dereferencing it you'll see "done"

;; When you dereference it you will block
;; until the result is available.
user=> (deref f)
done
100
```

https://clojuredocs.org/clojure.core/future

Clojure Promise & Future

Language or library?	- Language since 1.1
Concurrent or parallel?	- Parallel - Promises are synchronous, futures are asynchronous
Callbacks or values?	- Values only - No chaining or fanning


Scala

```
val p = promise[T]
val f = p.future

val producer = Future {
  val r = produceSomething()
  p success r
  continueDoingSomethingUnrelated()
}
```

```
val consumer = Future {
  startDoingSomething()
  f onSuccess {
    case r => doSomethingWithResult()
  }
}
```

http://docs.scala-lang.org/overviews/core/futures.html

```
val f: Future[List[String]] = Future {
  session.getRecentPosts
}

f onComplete {
  case Success(posts) =>
    for (post <- posts) println(post)
  case Failure(t) =>
    println("An error has occured: “
    + t.getMessage)
}
```

http://docs.scala-lang.org/overviews/core/futures.html

Scala Promise & Future

Language or library?	- Language
Concurrent or parallel?	- Parallel - Promises are synchronous, futures are asynchronous
Callbacks or values?	- Callbacks, no chaining or fanning - Blocking on a future’s value is possible but highly discouraged


Java™

```
CompletableFuture completableFutureToBeCompleted =
  CompletableFuture.supplyAsync( ( ) -> {
    for( int i = 0; i < 10; i-- )
    {
      System.out.println( "i " + i );
    }
    return 10;
} );

CompletableFuture completor =
  CompletableFuture.supplyAsync( ( ) -> {
    System.out.println( "completing the other" );
    completableFutureToBeCompleted.complete( 222 );
    return 10;
  } );

System.out.println( completor.get() );
System.out.println( completableFutureToBeCompleted.get() );
```

http://examples.javacodegeeks.com/core-java/util/concurrent/java-8-concurrency-tutorial/

```
final Future<String> contentsFuture =
  startDownloading(new URL(“http://www.example.com"));

while (!contentsFuture.isDone()) {
  askUserToWait();
  doSomeComputationInTheMeantime();
}

contentsFuture.get();
```

https://dzone.com/articles/javautilconcurrentfuture

Java Promise & Future

Language or library?	- CompletableFuture (Promise) in Java 8 - Future in Java 7
Concurrent or parallel?	- Parallel - Promises are synchronous, futures are asynchronous
Callbacks or values?	- Callbacks and values

Visual C#

```
Thread.CurrentThread.Name = "Main";

Task taskA = new Task( () =>
   Console.WriteLine("Hello from taskA."));
taskA.Start();

Console.WriteLine("Hello from thread '{0}'.",
                  Thread.CurrentThread.Name);
taskA.Wait();
```

https://msdn.microsoft.com/en-us/library/hh195051(v=vs.110).aspx

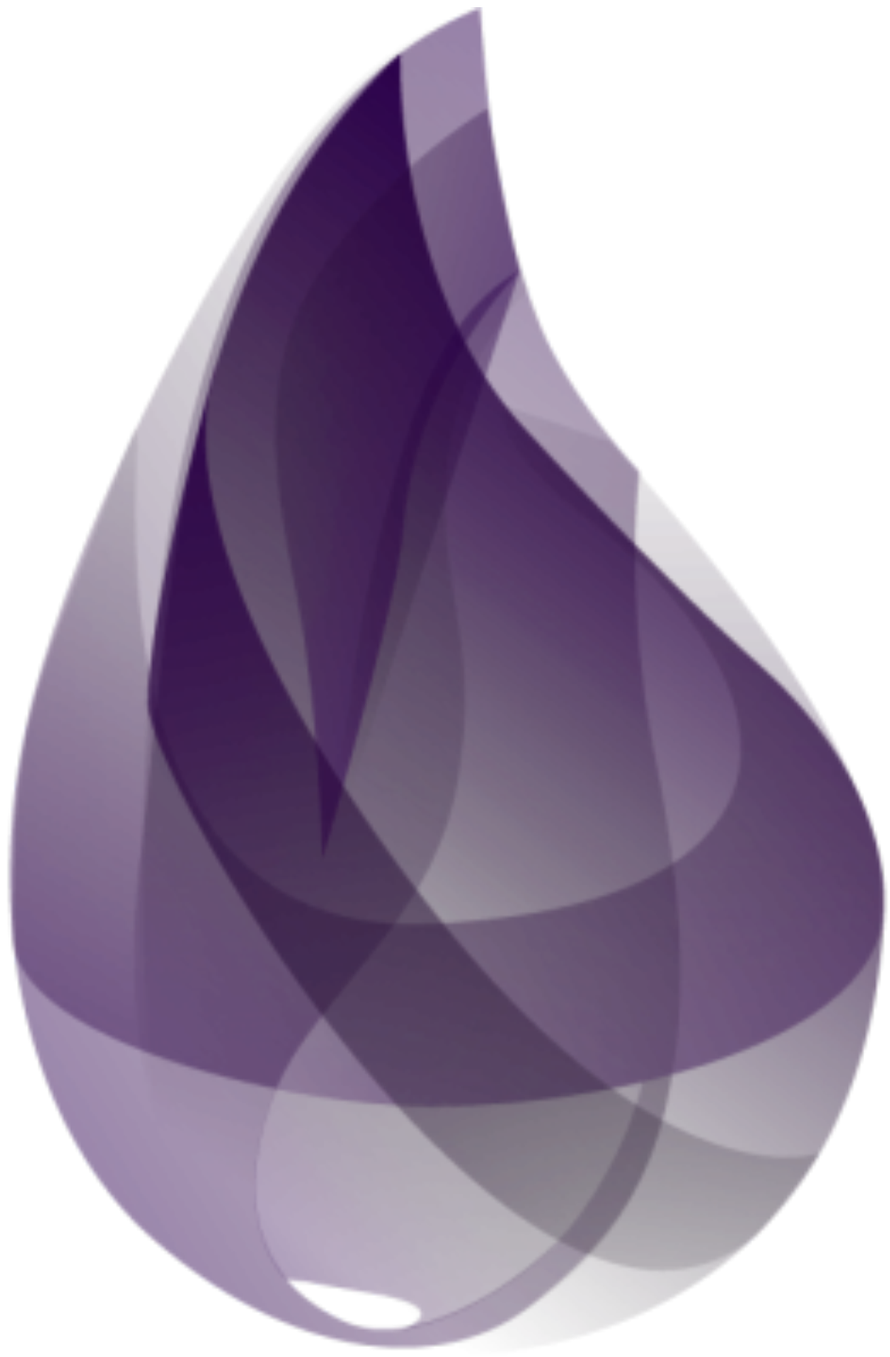
```
Task<Test> task =
  Task<Test>.Factory.StartNew(() =>
  {
    string s = ".NET";
    double d = 4.0;
    return new Test { Name = s, Number = d };
});

Test test = task.Result;
```

https://msdn.microsoft.com/en-us/library/dd537613(v=vs.110).aspx

C# Task

Language or library?	- .NET Framework 4.6 and 4.5
Concurrent or parallel?	- Parallel
Callbacks or values?	- Values only - Chaining



```
task = Task.async(
   fn -> do_some_work() end)

res  = do_some_other_work()

res + Task.await(task)
```

http://elixir-lang.org/docs/v1.1/elixir/Task.html

Elixir Task

Language or library?	- Since 1.0.5
Concurrent or parallel?	- Parallel
Callbacks or values?	- Values (Erlang message passing)

Fulfilling My Promise

And then() we're done()

Promises allow us to:

- compose our programs into
- independent operations
- with guarantees to
- preserve the order of operations

Shutouts

- Akron Code Club

  http://www.meetup.com/AkronCodeClub/

- Akron Women In Tech

  http://akronwit.org/

- Cleveland Tech on Slack

  http://cleveland-tech.herokuapp.com/

My name is Jerry D’Antonio

Tweet me @jerrydantonio

Say hello@testdouble.com