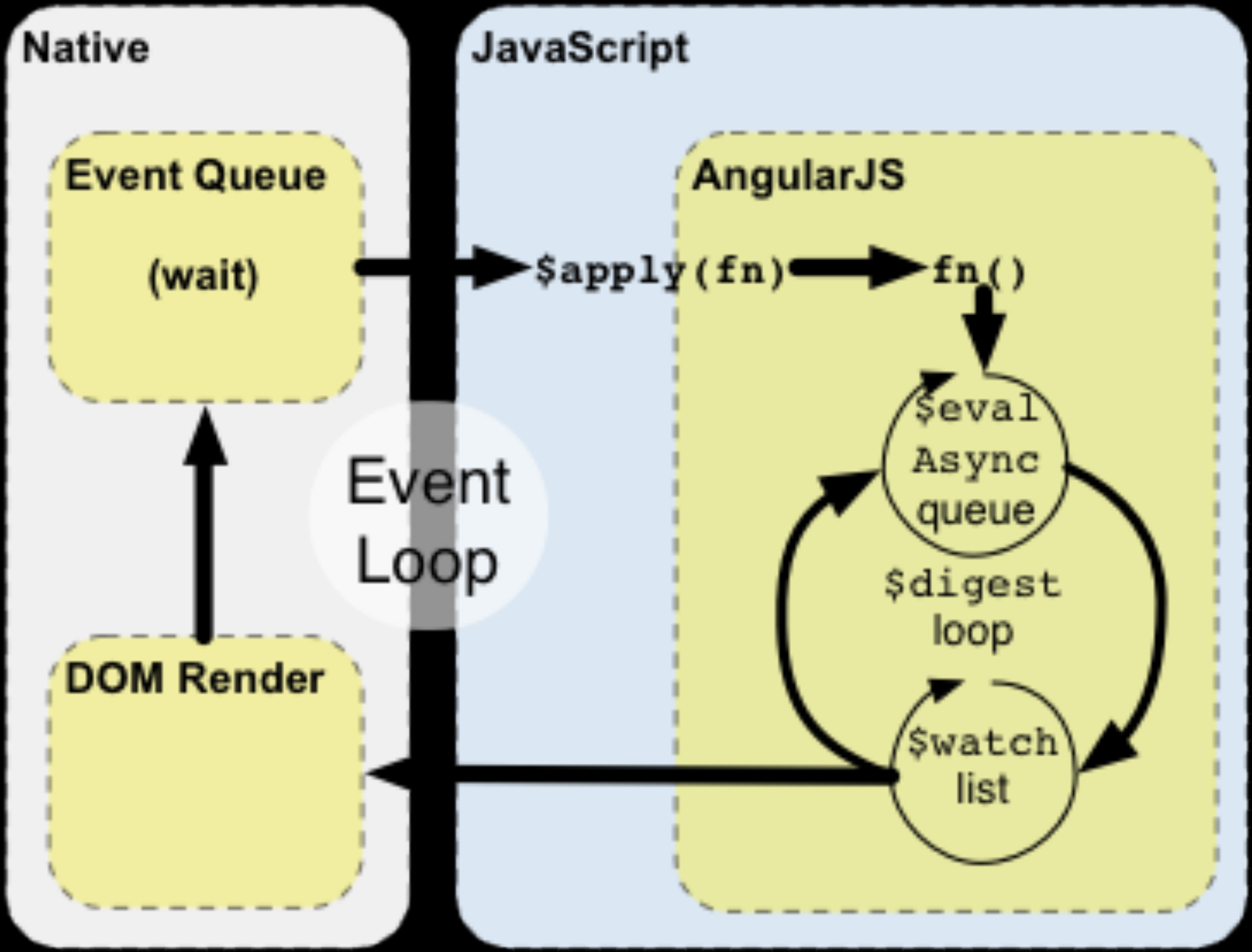
Promises

Why asynchronous?

- single thread
- high-latency executions
- users hate waiting

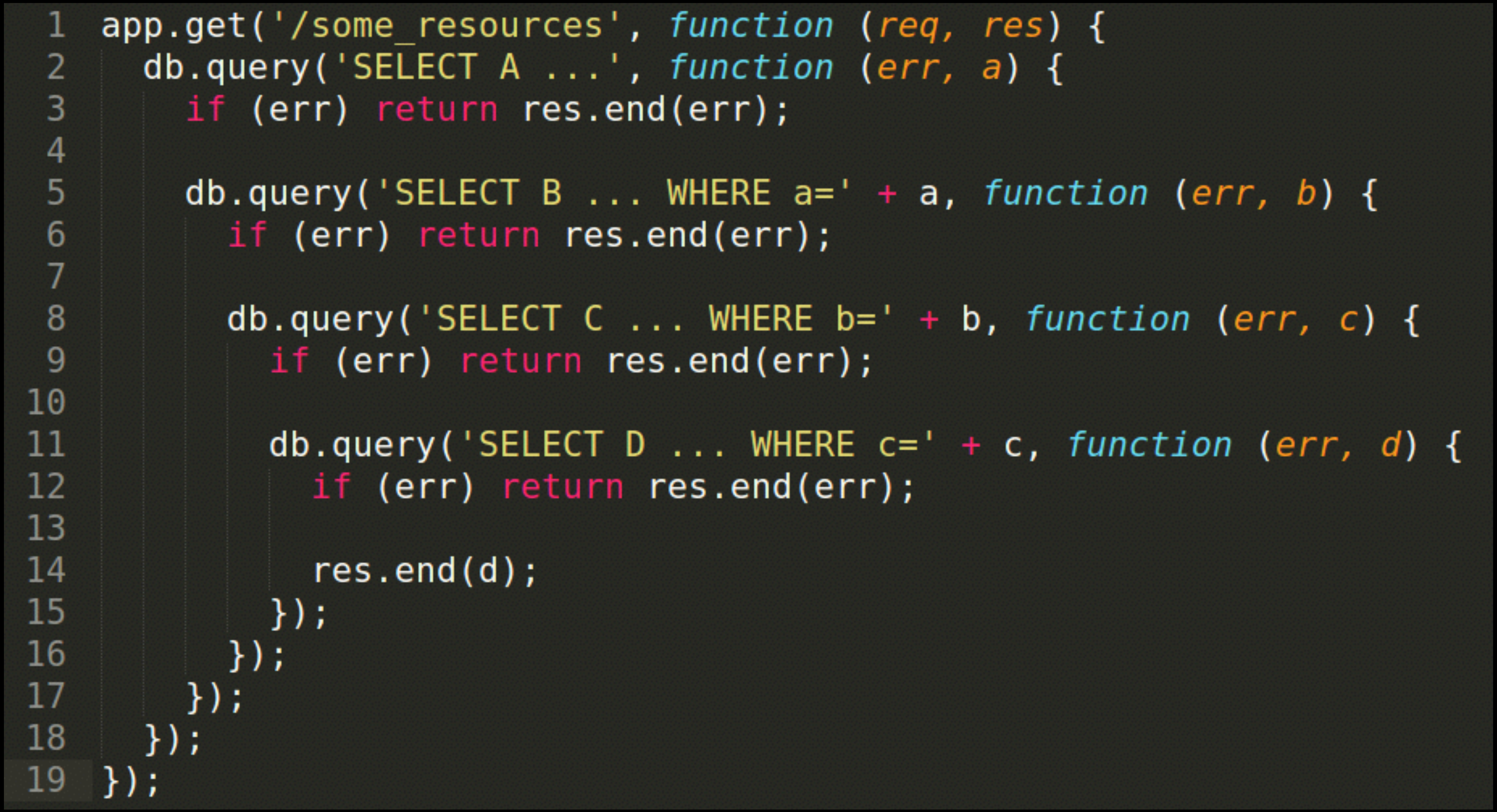
How?

Event Loop



Native
JavaScript
Event Queue
(wait)
DOM Render
Event Loop
AngularJS
$apply(fn)
fn()
$evalAsync queue
$digest loop
$watch list

Callback Hell

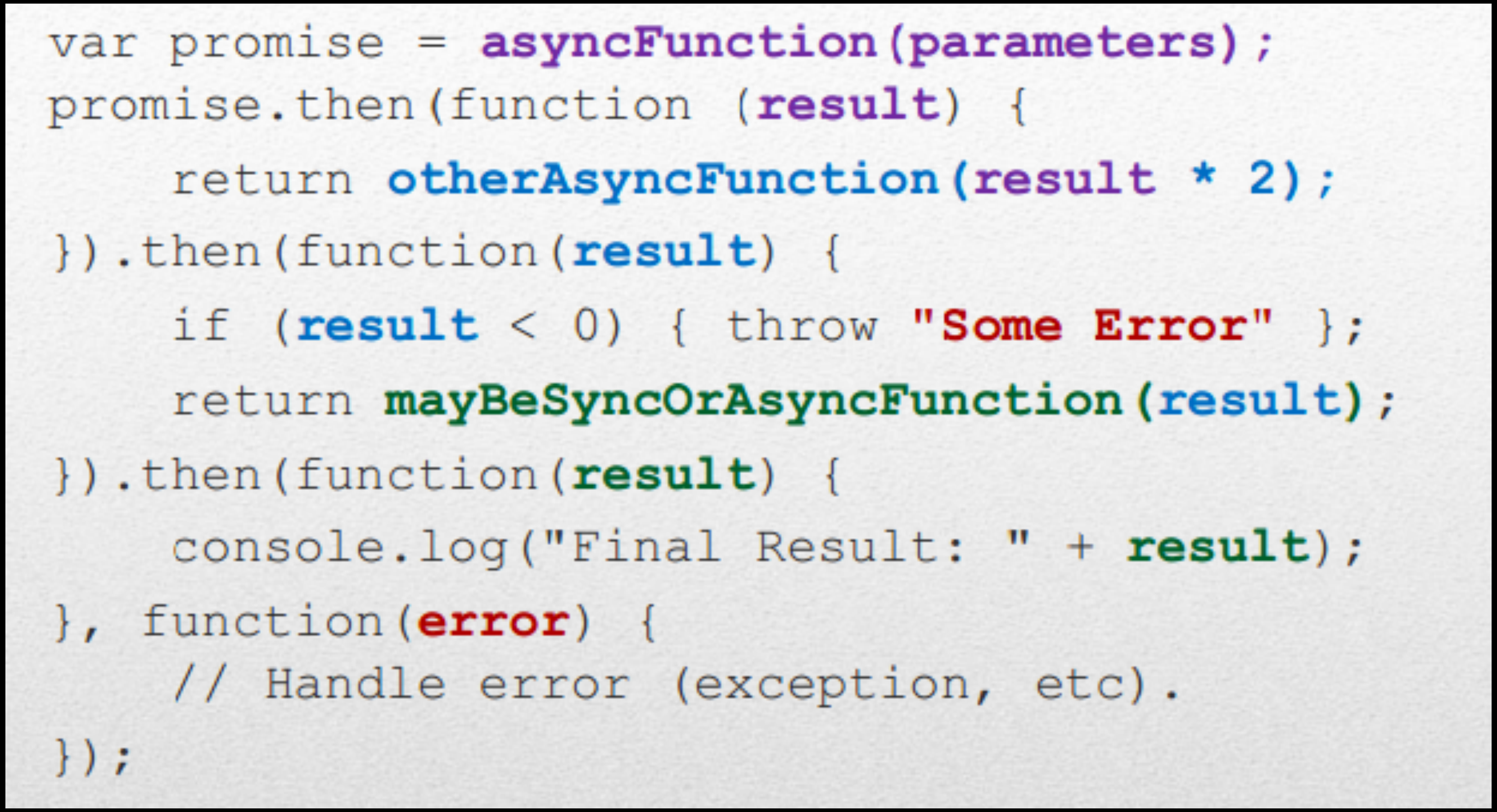


```
app.get('/some_resources', function (req, res) {
  db.query('SELECT A ...', function (err, a) {
    if (err) return res.end(err);

    db.query('SELECT B ... WHERE a=' + a, function (err, b) {
      if (err) return res.end(err);

      db.query('SELECT C ... WHERE b=' + b, function (err, c) {
        if (err) return res.end(err);

        db.query('SELECT D ... WHERE c=' + c, function (err, d) {
          if (err) return res.end(err);

          res.end(d);
        });
      });
    });
  });
});
```

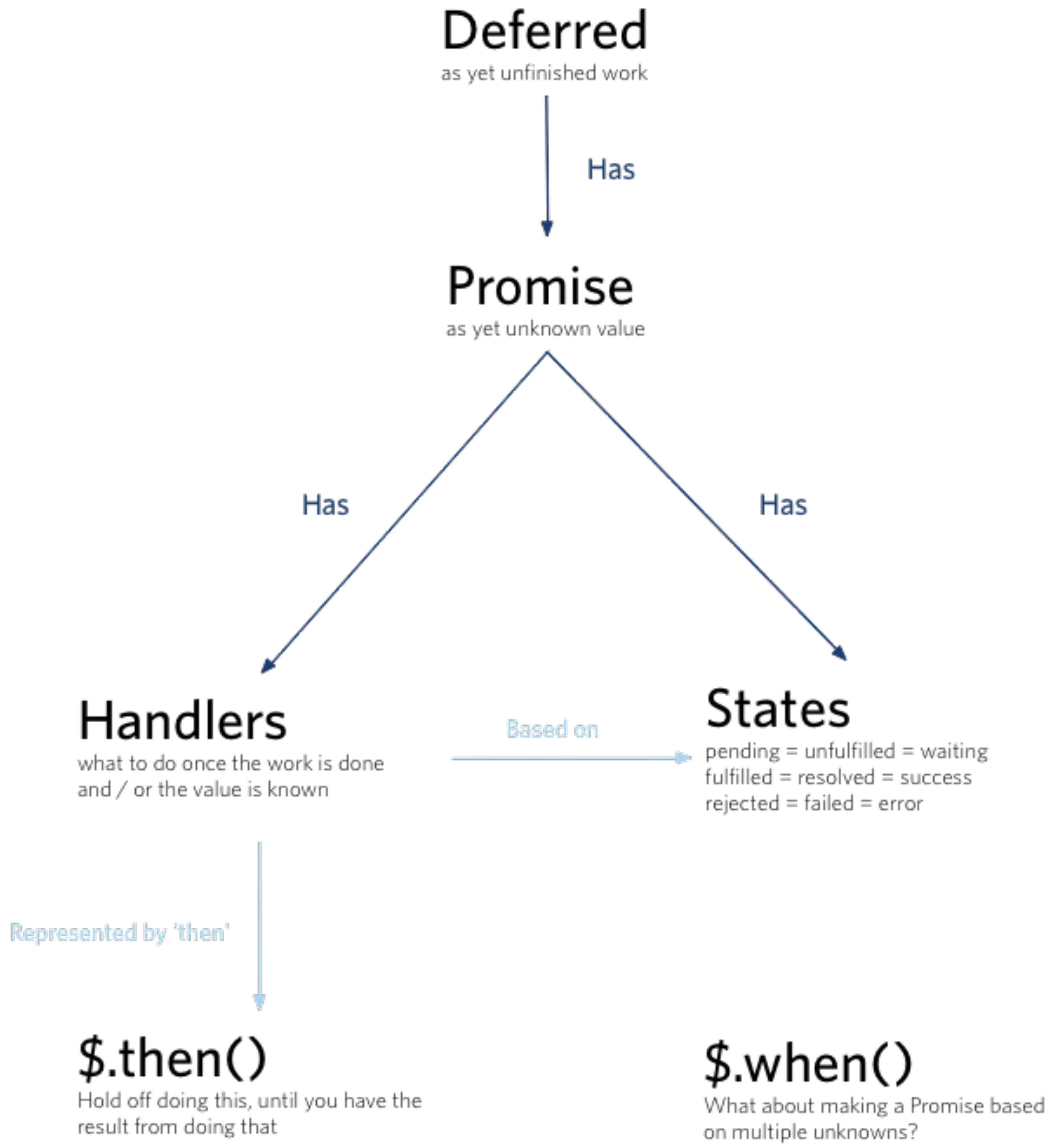
What’s a promise?

- promise/future/thenable (Promise A+)
- container for future value
- monadic
- closures

Promise Chaining



```
var promise = asyncFunction(parameters);
promise.then(function (result) {
    return otherAsyncFunction(result * 2);
}).then(function(result) {
    if (result < 0) { throw "Some Error" };
    return mayBeSyncOrAsyncFunction(result);
}).then(function(result) {
    console.log("Final Result: " + result);
}, function(error) {
    // Handle error (exception, etc).
});
```


Deferred
as yet unfinished work
Has
Promise
as yet unknown value
Has
Has
Handlers
what to do once the work is done and / or the value is known
Based on
States
pending = unfulfilled = waiting
fulfilled = resolved = success
rejected = failed = error
Represented by 'then'
$.then()
Hold off doing this, until you have the result from doing that
$.when()
What about making a Promise based on multiple unknowns?

Nomenclature

- states of a promise
  - pending
  - settled
    - resolved/fulfilled
    - rejected
- then
- when (resolve) / promisification
- reject

Code

- with some anti-patterns