Writing Sound Asynchronous Code with Coroutines

Tomas Kulich
ReactiveConf 2016

Old school: Callbacks

```
fetchData(url, (err, data) => {…})
```

```
fetchData(url1, (err, data1) => {
  if (err != null) {
    console.error(err)
  } else {
    fetchMoreData(url2, (err, data) => {
      if (err != null) {
        console.error(err)
      } else {
        fetchEvenMoreData(url3, (err, data) => {
          if (err != null) {
            console.error(err)
          } else {
            // do something with data
          }
        })
      }
    })
  }
})
```

Promises: Great Leap Forward

```
fetchData(url).then((data) => …)
```

```
let data1, data2, data3

fetchData(url1)
.then((result) => {
  data1 = result
  return fetchMoreData(url2)
})
.then((result) => {
  data2 = result
  return fetchEvenMoreData(url3)
})
.then((result) => {
  data3 = result
  /* do something with data */
}).catch((e) => {
  console.error(e)
})
```

Super Great Leap Forward: Promises + async / await

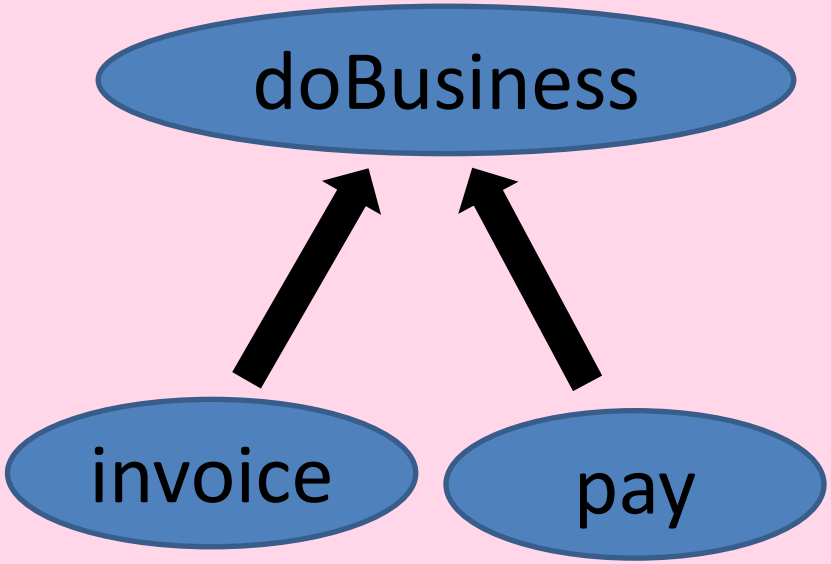
```
async function main() {
  try {
    let data1 = await fetchData(url1)
    let data2 = await fetchMoreData(url2)
    let data3 = await fetchEvenMoreData(url3)
  } catch (e) {
    console.error(e)
  }
}
```

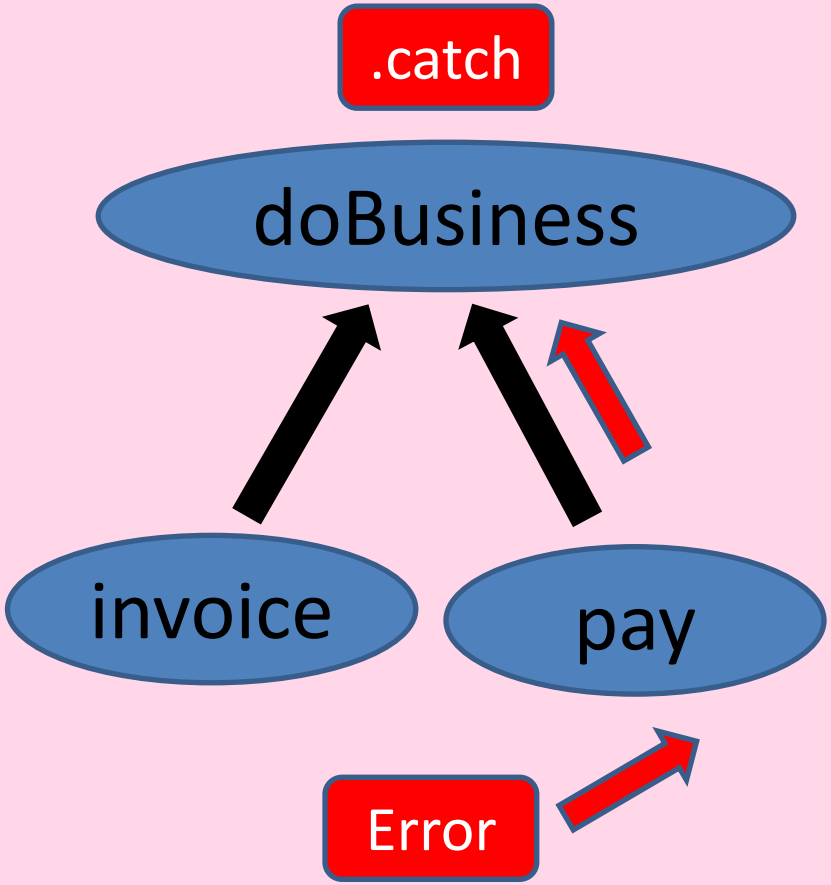
Alternative approaches

- Communicating Sequential Processes
  - aka CSP
  - js-csp, core.async in Clojure
- koa webserver
- co
- task.js
- redux-saga

yacol

- Yet Another COroutine Library
- https://github.com/vacuumlabs/yacol
- npm: yacol


doBusiness
invoice
pay


.catch
doBusiness
invoice
pay
Error

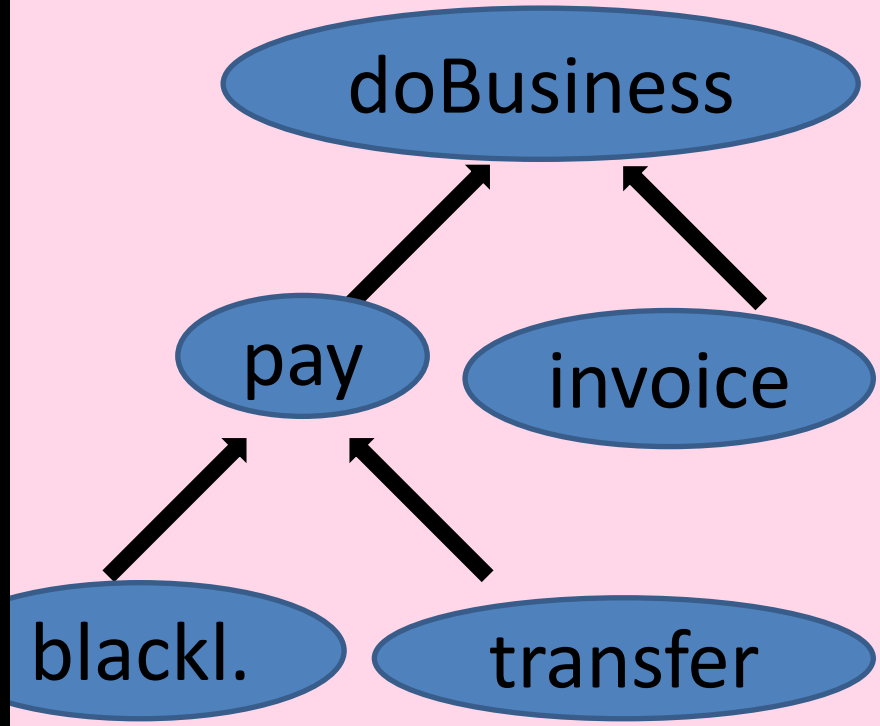
```
function* blacklistCheck() {
  console.log('do backlist check')
}

function* transfer() {
  throw new Error('catch me!')
}

function* pay() {
  const isOk = yield run(blacklistCheck)
  if (isOk) {
    run(transfer)
  }
}

function* invoice() {
  console.log('preparing invoice')
}

function* doBusiness() {
  run(pay)
  run(invoice)
}
```



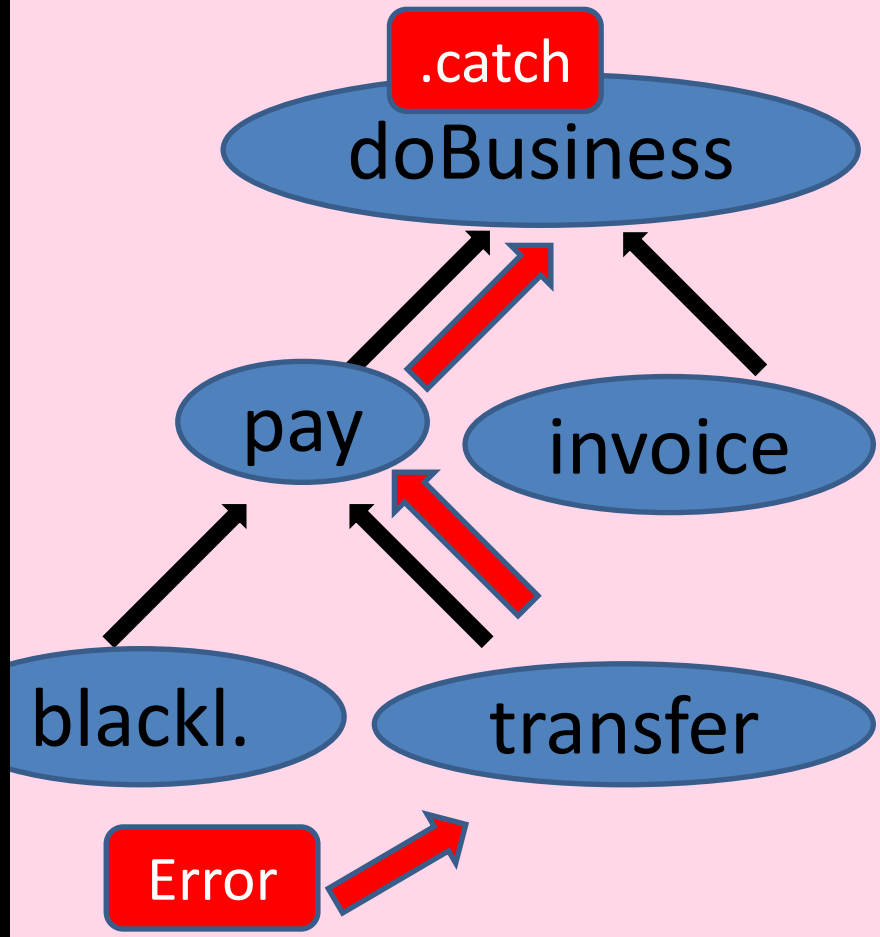
```
function* blacklistCheck() {
  console.log('do backlist check')
}

function* transfer() {
  throw new Error('catch me!')
}

function* pay() {
  const isOk = yield run(blacklistCheck)
  if (isOk) {
    run(transfer)
  }
}

function* invoice() {
  console.log('preparing invoice')
}

function* doBusiness() {
  run(pay)
  run(invoice)
}
```



Explicit parent-child relationship, yay!

- Context (Dart zones, thanks Dart)
- Messaging (very simple, thanks js-csp)
- Plays nice with promises

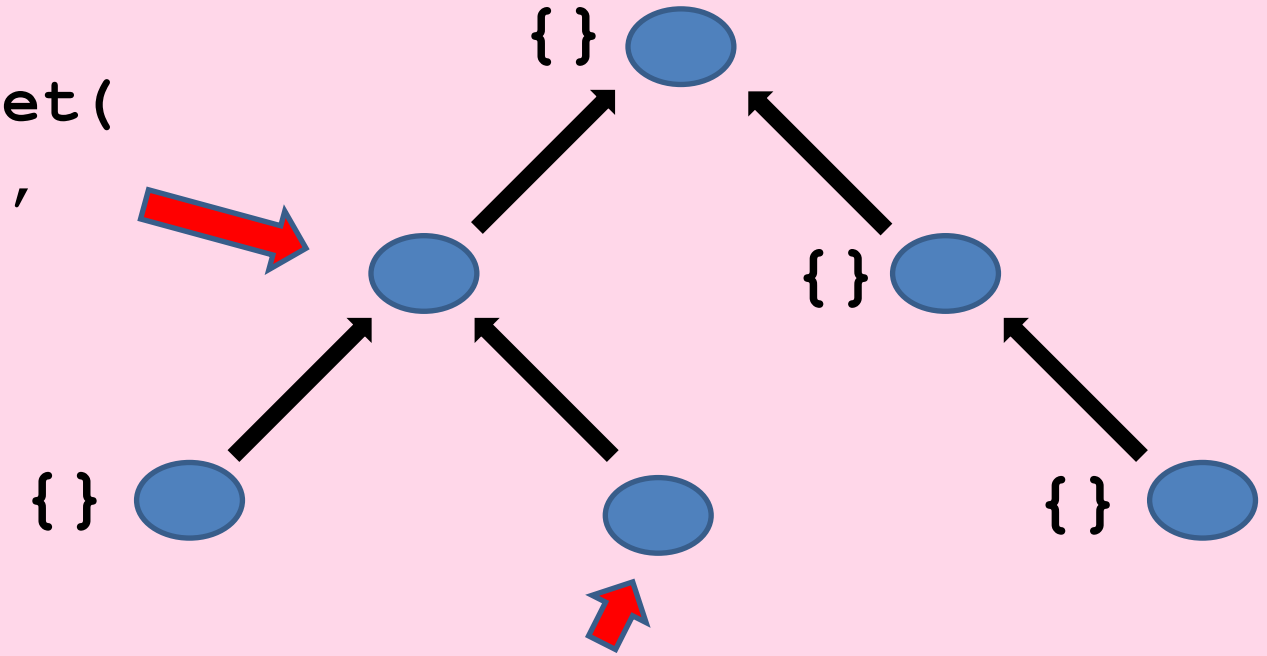
--- TBI ---

- Terminate
- Awesome stacktraces
- Inspect

Explicit parent-child relationship, yay!

- Context (Dart zones, thanks Dart)

```
context.set(
   'hello',
   'world'
)
```




```
context.get('hello') === 'world'
```

Explicit parent-child relationship, yay!

- Context (Dart zones, thanks Dart)
- Messaging (very simple, thanks js-csp)

Explicit parent-child relationship, yay!

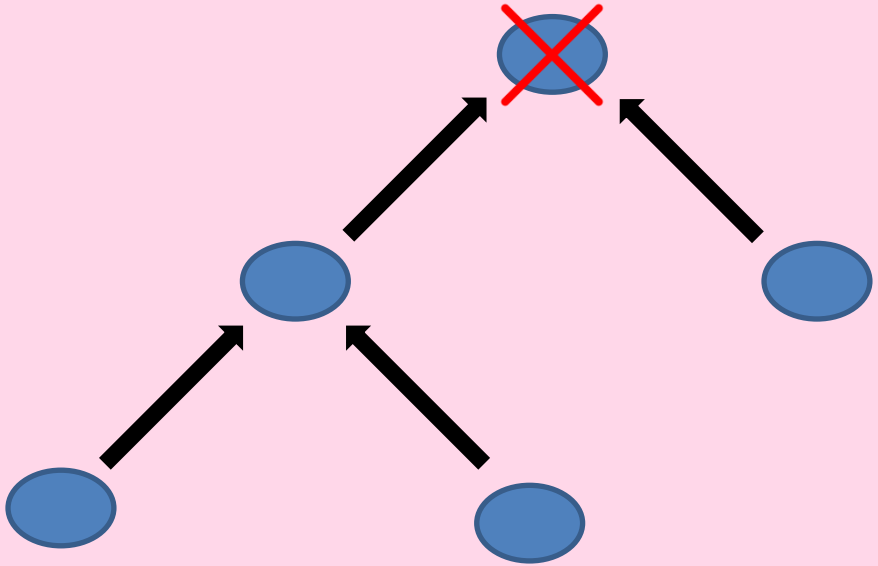
- Context (Dart zones, thanks Dart)
- Messaging (very simple, thanks js-csp)
- Plays nice with promises

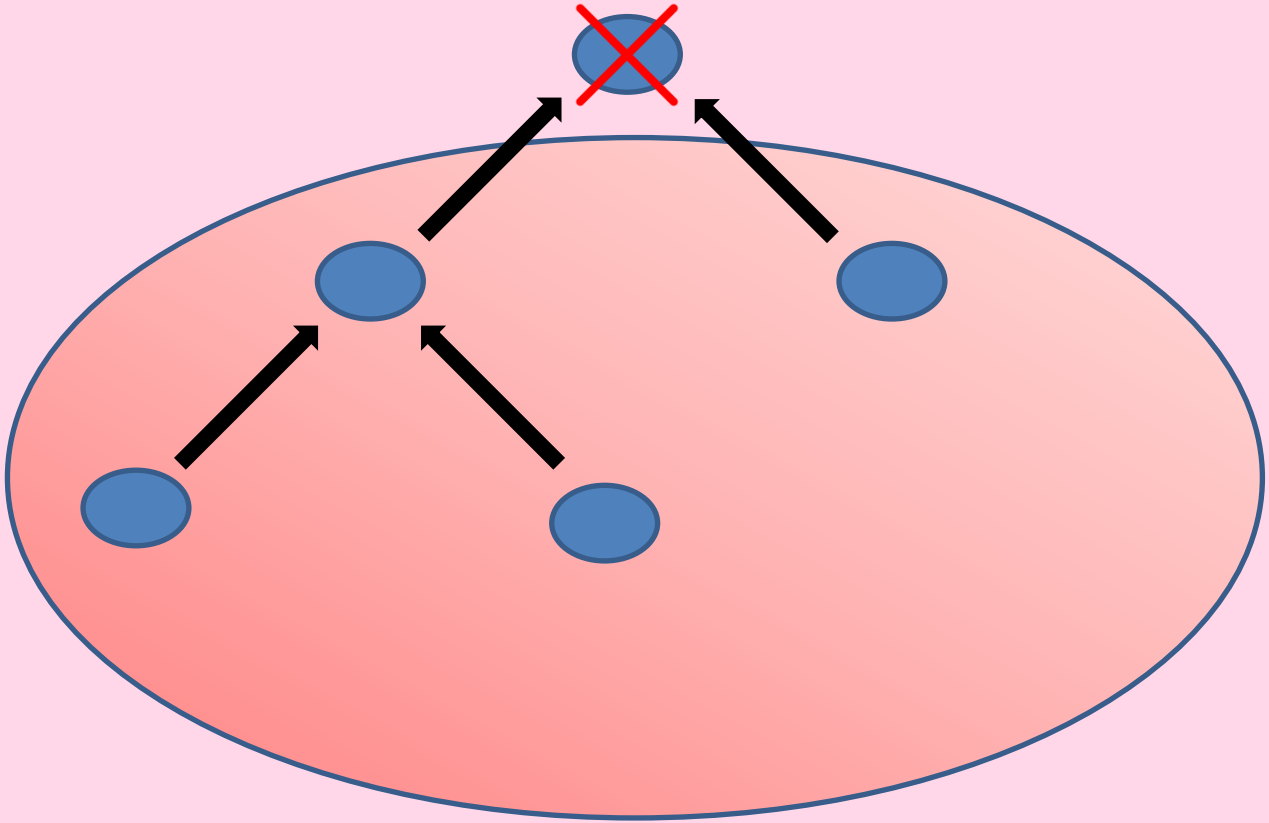
Explicit parent-child relationship, yay!

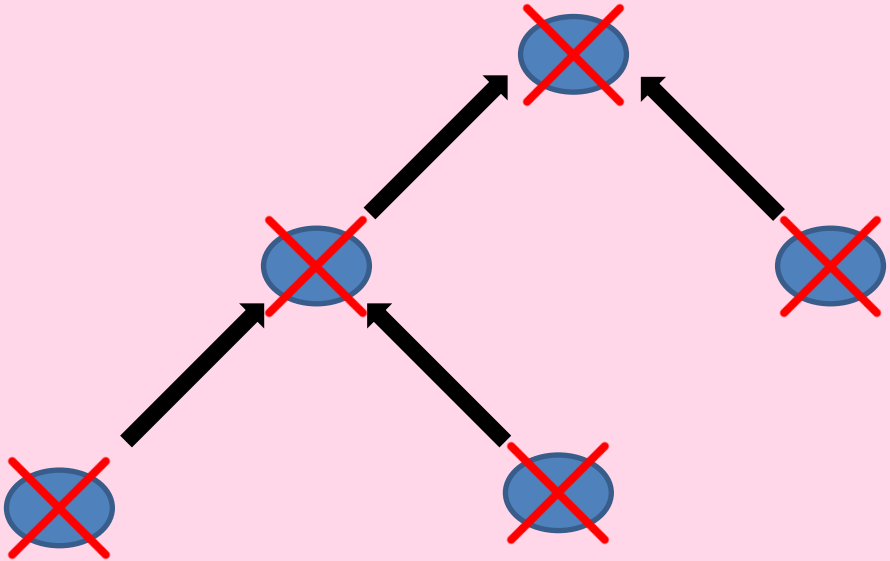
- Context (Dart zones, thanks Dart)
- Messaging (very simple, thanks js-csp)
- Plays nice with promises

--- TBI ---

- Terminate
- Awesome stacktraces
- Inspect







- https://github.com/vacuumlabs/yacol
- npm: yacol

Questions please?