

JavaScript (Dr. SES on ES6) as a Language for Distributed Secure Algorithms

Mark S. Miller (Google Research)

Tom Van Cutsem (VUB)

Tyler Close (Google)



OCaps: Small step from pure objects

Memory safety and encapsulation

+ Effects ***only*** by using held references

+ No powerful references by default

OCaps: Small step from pure objects

Memory safety and encapsulation

+ Effects **only** by using held references

+ No powerful references by default

Reference graph \equiv Access graph

Only connectivity begets connectivity

OO expressiveness for security patterns

Dr. SES

Distributed Resilient Secure EcmaScript

Linguistic abstraction for safe messaging

Stretch reference graph between event loops & machines
Crypto analog of memory safety

SES + Promise library + infix "!" syntax

("Q" Library usable today without "!" syntax)

Unguessable URLs as Crypto-Caps

<https://www.example.com/app/#mhbqcmva5ja3>

How are secrets like object references?

Dr. SES

Distributed Resilient Secure EcmaScript

var result = bob.foo(carol);

Local-only immediate call

var resultP = bobP ! foo(carol);

Eventual send

Dr. SES

Distributed Resilient Secure EcmaScript

var result = bob.foo(carol);

Local-only immediate call

var resultP = bobP ! foo(carol);

Eventual send

var result = bob.foo;

Local-only immediate get

var resultP = bobP ! foo;

Eventual get

Dr. SES

Distributed Resilient Secure EcmaScript

var resultP = bobP ! foo(carol);

Eventual send

var resultP = bobP ! foo;

Eventual get

Dr. SES

Distributed Resilient Secure EcmaScript

var resultP = bobP ! foo(carol);

Eventual send

var resultP = bobP ! foo;

Eventual get

Dr. SES

Distributed Resilient Secure EcmaScript

```
var resultP = bobP ! foo(carol);
```

Eventual send

```
var resultP = bobP ! foo;
```

Eventual get

```
Q(resultP).when(function(result) {  
  ...result...  
}, function (ex) {  
  ...ex...  
});
```

Register for notification

Async object ops as JSON/REST ops

```
var resultP = bobP ! foo(carol);
```

```
POST https://...q=foo {...}
```

```
var resultP = bobP ! foo;
```

```
GET https://...q=foo
```

```
Q(resultP).when(function(result) {  
  ...result...  
}, function (ex) {  
  ...ex...  
});
```

```
xhr.onreadystatechange = ...
```

Q.race

Any done? (good or bad)

```
Q.race = function(...answerPs) {
  const result = Q.defer();

  answerPs.forEach(function(answerP) {
    Q(answerP).when(function(_) {
      result.resolve(answerP);
    }, function(reason) {
      result.resolve(Q.reject(reason));
    });
  });
  return result.promise;
};
```

Timeouts = Racing with a timeBomb

```
function timeBomb(deltaMillis) {  
  const result = Q.defer();  
  setTimeout(function() {  
    result.resolve(Q.reject(new Error("time expired")));  
  }, deltaMillis);  
  return result.promise;  
}
```

```
const resultP = Q.race(bobP ! foo(carol), timeBomb(3000));
```

Q.all

All good? Any bad?

```
Q.all = function(...answerPs) {
  let countDown = answerPs.length;
  if (countDown === 0) { return answerPs; }
  const result = Q.defer();

  answerPs.forEach(function(answerP) {
    Q(answerP).when(function(_) {
      if (--countDown === 0) { result.resolve(answerPs); }
    }, function(reason) {
      result.resolve(Q.reject(reason));
    });
  });
  return result.promise;
}
```

Q.join

Mutual acceptability

```
Q.join = function(xP, yP) {  
  return Q.all(xP, yP).when(function([x, y]) {  
    if (Object.is(x, y)) {  
      return x;  
    } else {  
      throw new Error("not the same");  
    }  
  });  
};
```

makeMint

The “factorial” of language security

```
function makeMint() {
  const amp = WeakMap();
  return function mint(balance) {
    Nat(balance);
    const purse = def({
      get balance()           { return balance; },
      makePurse: function()   { return mint(0) },
      deposit: function(amount, src) {
        Nat(balance + amount);
        amp.get(src)(Nat(amount));
        balance += amount;
      }
    });
    amp.set(purse, function decr(amount) { balance = Nat(balance - amount); });
    return purse;
  };
}
```

transfer # insecure

Participant in Two-Phase Commit

```
function transfer(decisionP, srcPurseP, dstPurseP, amount) {
  const escrowPurseP = srcPurseP ! makePurse();

  Q(decisionP).when(function(_) { # setup phase 2
    dstPurseP ! deposit(amount, escrowPurseP);
  }, function(reason) {
    srcPurseP ! deposit(amount, escrowPurseP);
  });
  return escrowPurseP ! deposit(amount, srcPurseP); # phase 1
}
```

exchange # insecure

All or nothing (or stuck)

```
function exchange(aliceMoneySrcP, aliceStockDstP, stockNeeded,  
                 bobStockSrcP, bobMoneyDstP, moneyNeeded) {  
  const decision = Q.defer();  
  
  decision.resolve(Q.all(  
    transfer(decision.promise, aliceMoneySrcP, bobMoneyDstP, moneyNeeded),  
    transfer(decision.promise, bobStockSrcP, aliceStockDstP, stockNeeded)));  
  return decision.promise;  
}
```

makePurseMaker

All customers get the same makePurse

```
function makePurseMaker(purse) {  
  return function makePurse() { return purse.makePurse(); };  
}
```

makeTransfer

Make mutually acceptable participant

```
function makeTransfer(makeEscrowPurseP) {  
  
  return function transfer(decisionP, srcPurseP, dstPurseP, amount) {  
    const escrowPurseP = makeEscrowPurseP ! ();  
  
    Q(decisionP).when(function(_) {                                     # setup phase 2  
      dstPurseP ! deposit(amount, escrowPurseP);  
    }, function(reason) {  
      srcPurseP ! deposit(amount, escrowPurseP);  
    });  
    return escrowPurseP ! deposit(amount, srcPurseP); # phase 1  
  };  
}
```

makeTransfer

Make mutually acceptable participant

```
function makeTransfer(makeEscrowPurseP) {  
  
  return function transfer(decisionP, srcPurseP, dstPurseP, amount) {  
    const escrowPurseP = makeEscrowPurseP ! ();  
  
    Q(decisionP).when(function(_) {                                     # setup phase 2  
      dstPurseP ! deposit(amount, escrowPurseP);  
    }, function(reason) {  
      srcPurseP ! deposit(amount, escrowPurseP);  
    });  
    return escrowPurseP ! deposit(amount, srcPurseP); # phase 1  
  }; }  
}; }
```

makeExchange

Secure Escrow Exchange Agent

```
function makeExchange(aliceMakeMoneyEscrowP, aliceMakeStockEscrowP,  
                      bobMakeStockEscrowP, bobMakeMoneyEscrowP) {  
  
  const transferMoney = makeTransfer(Q.join(aliceMakeMoneyEscrowP,  
                                             bobMakeMoneyEscrowP));  
  const transferStock  = makeTransfer(Q.join(bobMakeStockEscrowP,  
                                             aliceMakeStockEscrowP));  
  
  return function exchange(aliceMoneySrcP, aliceStockDstP, stockNeeded,  
                            bobStockSrcP, bobMoneyDstP, moneyNeeded) {  
  
    const d = Q.defer();  
    d.resolve(Q.all(  
      transferMoney(d.promise, aliceMoneySrcP, bobMoneyDstP, moneyNeeded),  
      transferStock (d.promise, bobStockSrcP, aliceStockDstP, stockNeeded)));  
    return d.promise;  
  
  }; };
```

makeExchange

Secure Escrow Exchange Agent

```
function makeExchange(aliceMakeMoneyEscrowP, aliceMakeStockEscrowP,  
                    bobMakeStockEscrowP, bobMakeMoneyEscrowP) {  
  const transferMoney = makeTransfer(Q.join(aliceMakeMoneyEscrowP,  
                                           bobMakeMoneyEscrowP));  
  const transferStock = makeTransfer(Q.join(bobMakeStockEscrowP,  
                                           aliceMakeStockEscrowP));  
  return function exchange(aliceMoneySrcP, aliceStockDstP, stockNeeded,  
                          bobStockSrcP, bobMoneyDstP, moneyNeeded) {  
    const d = Q.defer();  
    d.resolve(Q.all(  
      transferMoney(d.promise, aliceMoneySrcP, bobMoneyDstP, moneyNeeded),  
      transferStock (d.promise, bobStockSrcP, aliceStockDstP, stockNeeded)));  
    return d.promise;  
  }; }  
}; }
```

makeMembraningPurse

Non-fungible, exercisable, exclusive

```
function makeMembraningPurse(makeMembrane, preciousP) {
  const amp = WeakMap();
  function mint(optMembrane) {
    const membraningPurse = def({
      get balance()      { return optMembrane.wrapper; },
      makePurse: function() { return mint(null); },
      deposit: function(_, src) {
        amp.get(src);      optMembrane = makeMembrane(preciousP);
      } });
    amp.set(membraningPurse, function revoke() {
      optMembrane.revoke();  optMembrane = null;
    });
    return membraningPurse;
  }
  return mint(makeMembrane(preciousP));
}
```