

Oni - a Structured Concurrency Framework for Javascript

FOSDEM 2009 talk

Alexander Fritze <alex@crozilla.com>

Oni - a Structured Concurrency Framework for Javascript

Warning: This talk is likely to be boring if you are not a Javascript person, web programmer, extension developer or something like that.

Oni - a Structured Concurrency Framework for Javascript

- Developed as part of Zap - the Mozilla-based JavaScript SIP stack
- Systematic way of trying to making complex concurrent systems more tractable and modular
- Two implementations:
 - 'straight' Javascript, using DOM setTimeout behind the scenes
 - Mozilla Javascript (*.jsm) module (uses some XPCOM)
- IMO most interesting for Firefox extensions that make use of webservice

How concurrency is forced upon us

Example: Get a webpage, translate it from English to German, display it

Sequential version:

```
function getWebpage(source) : access web, returns doc
function translate(doc) : access webservice, returns doc
function display(doc, elem)
```



```
function whenButtonPressed() {
    display(translate(getWebpage("http://cnn.com")),
            outputDiv)
}
```

Problem: Blocks the UI.

Possible Solution: Asynchronous Callbacks

```
function getWebpage(source, callback)
```

```
function translate(doc, callback)
```



```
function whenButtonPressed() {  
  function callback2(doc) {  
    display(doc, myDiv);  
  }  
  
  function callback1(doc) {  
    translate(doc, callback2);  
  }  
  
  getWebpage("http://cnn.com", callback1);  
}
```

Problem: Control flow not easy to follow.

Possible Solution: Asynchronous Callbacks

```
function getWebpage(source, callback)
function translate(doc, callback)
```

```
function whenButtonPressed() {
  function callback2(doc) {
    display(doc, myDiv);
  }
  function callback1(doc) {
    translate(doc, callback2);
  }
  getWebpage("http://cnn.com", callback1);
}
```

More problems: Composability

Asynchronous code doesn't compose well

- What if we want to try to get two websites at the same time and display the first one that returns?

What we would like to write:

```
function whenButtonPressed() {  
    display(translate(alt(getWebpage("http://cnn.com"),  
                          getWebpage("http://bbc.com")),  
            outputDiv)  
}
```

Asynchronous code doesn't compose well

What we have to write:

```
function whenButtonPressed() {
    var have_doc = false;

    function callback2(doc) {
        display(doc, myDiv);
    }

    function callback1(doc) {
        if (have_doc) return;
        have_doc = true;
        translate(doc, callback2);
    }

    getWebpage("http://cnn.com", callback1);
    getWebpage("http://bbc.com", callback1);
}
```

And this is still bad... e.g. what about request cancellation?

So how does Oni address this?

Tries to get as close as possible to the version we want to write,

- from within Javascript
- without requiring preprocessing or precompilation
- without introducing a new syntax

'Sequential' version:

```
function whenButtonPressed() {
    display(translate(alt(getWebpage("http://cnn.com"),
                           getWebpage("http://bbc.com")),
               outputDiv);
}
```

Oni version:

```
function whenButtonPressed() {
    Display(Translate(Alt(GetWebpage("http://cnn.com"),
                           GetWebpage("http://bbc.com")),
               outputDiv).run());
}
```

Dissecting Oni

Oni operators

```
function whenButtonPressed() {  
    Display(Translate(Alt(GetWebpage("http://cnn.com"),  
                          GetWebpage("http://bbc.com")),  
           outputDiv).run());  
}
```

Oni expression

- Oni expressions are 'programs' within 'programs'.
- You execute them in the background by calling run() on them.
- They are like threads if you will...
- ... but unlike threads they compose.
- They implement a functional 'language' on top of Javascript in which you can handle concurrency in 'sequential' style.

A simple Oni example

Celsius to Fahrenheit converter

```
fahrenheit(celsius) = celsius * 1.8 + 32
```

A simple Oni example

Celsius to Fahrenheit converter

```
fahrenheit(celsius) = celsius * 1.8 + 32
```

This has a similar structure as our earlier example:

```
display(translate(getWebpage("http://cnn.com")))
```



```
print ( add( mul( celsius, 1.8 ), 32 ) )
```

If add & mul were webservicees, this would be pretty much the same problem...

A simple Oni example

... so for demonstration purposes, I made some maths webservices:

`http://testbed.croczilla.com:8080/fosdem2009/add?arg1=x&arg2=y`

`http://testbed.croczilla.com:8080/fosdem2009/sub?arg1=x&arg2=y`

`http://testbed.croczilla.com:8080/fosdem2009/mul?arg1=x&arg2=y`

`http://testbed.croczilla.com:8080/fosdem2009/div?arg1=x&arg2=y`

E.g.:

`http://testbed.croczilla.com:8080/fosdem2009/add?arg1=40&arg2=2`



`<result>42</result>`

Try it ->

To make this a bit more realistic, the server delays results by a random delay of between 0 to 5 seconds.

A simple Oni example

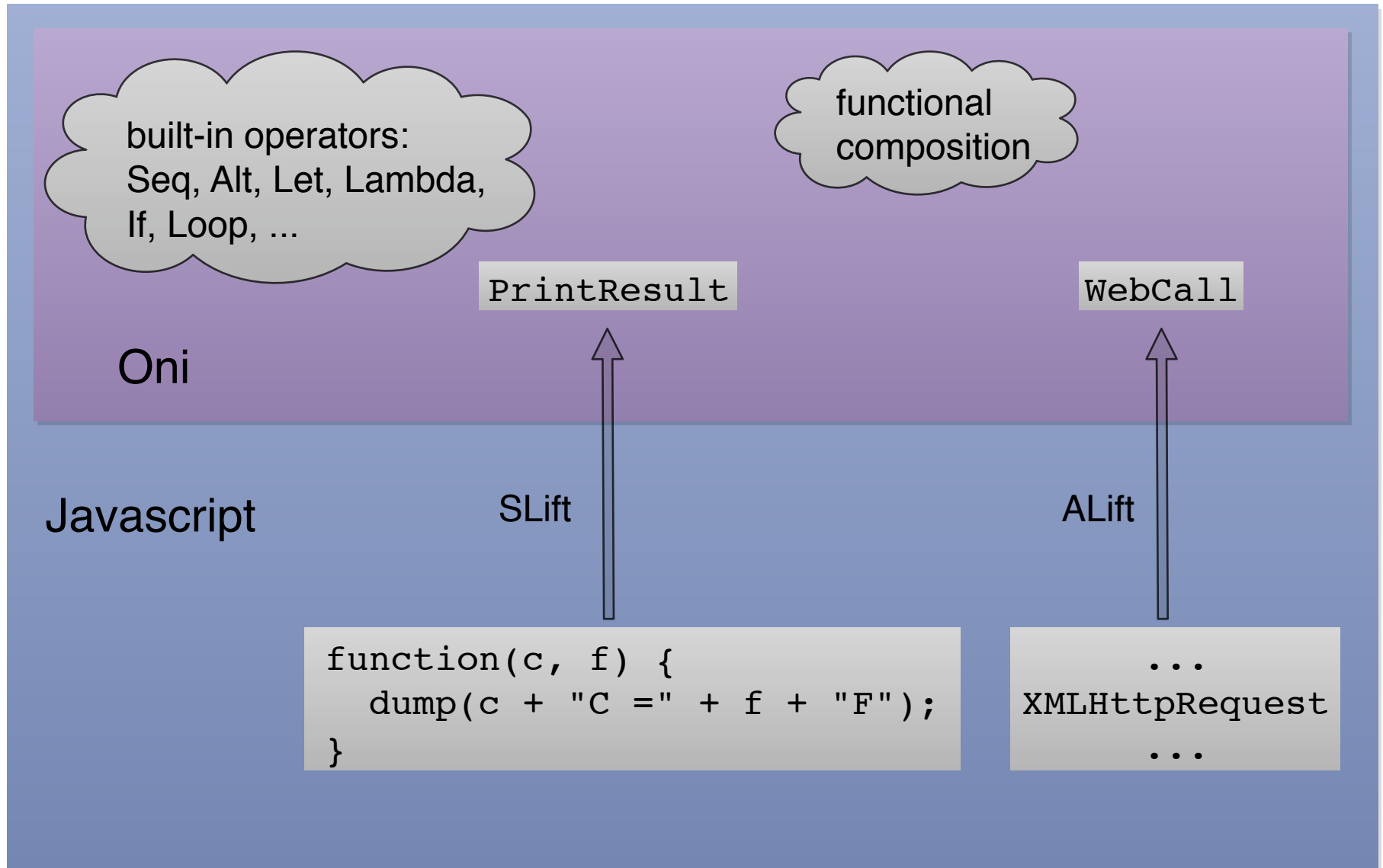
Temperature converter using web service - 'traditional' version:

-> http://testbed.croczilla.com:8080/fosdem2009/fahrenheit_xmlhttpreq.html

Temperature converter using web service - 'Oni' version:

-> http://testbed.croczilla.com:8080/fosdem2009/fahrenheit_oni.html

Lifting: Making new Oni operators



SLift: Lifting a (synchronous) function into Oni

```
var PrintResult = SLift(  
  function(c, f) {  
    dump(c + "C =" + f + "F");  
  });
```



```
PrintResult(..., ...).run();
```

```
var IsLess = SLift(  
  function(x, y) {  
    return x < y;  
  });
```



```
If(IsLess(Add(5, 10), 11),  
  Print("Yes, it's less"),  
  Print("No, it's not less"))
```

ALift() is similar but it lifts 'asynchronous constructs' like e.g. XMLHttpRequest or setTimeout

A slightly more complex Oni example

Fibonacci numbers:

```
F(0)=0; F(1)=1; F(n) = F(n-1) + F(n-2) for n > 1
```

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

In Javascript:

```
function fib(n) {  
  if (n < 2)  
    return n;  
  else  
    return fib(n-1) + fib(n-2);  
}
```

A slightly more complex Oni example

In Javascript:

```
function fib(n) {  
  if (n < 2)  
    return n;  
  else  
    return fib(n-1) + fib(n-2);  
}
```

In Oni:

```
var Fib = Defun(['n'],  
  If(IsLess(Get('n'), 2),  
    Get('n'),  
    Add(Recurse(Sub(Get('n'), 1)),  
        Recurse(Sub(Get('n'), 2)))));
```

Ok, so this looks a bit more complicated, but it works with asynchronous Add, Sub.

A slightly more complex Oni example

```
var Fib = Defun(['n'],  
               If(IsLess(Get('n'), 2),  
                 Get('n'),  
                 Add(Recurse(Sub(Get('n'), 1)),  
                    Recurse(Sub(Get('n'), 2))));
```

Functional composition in Oni:

```
Fun(Exp1, Exp2, ...)
```

Executes Exp1, Exp2, ... **concurrently** and calls Fun once all results are available.

-> http://testbed.croczilla.com:8080/fosdem2009/fibonacci_oni.html

Composability: Adding cancellation support

```
PrintResult(n, Fib(n))
```



```
Alt( Seq( WaitForDOMEvent("click", cancel_button),  
        PrintResult(n, "cancelled")  
      ),  
      PrintResult(n, Fib(n))  
    )
```

Seq(exp1, exp2, ...) : executes exp1, exp2, ... sequentially

Alt(exp1, exp2, ...) : executes exp1, exp2, ... concurrently, returns the value of the first expression that completes, and **Cancels all pending expressions.**

Composability: Adding timeouts / fault-tolerance

```
CallWebserviceA()
```



```
Alt( CallWebserviceA(),  
      Seq( Timeout(t), CallWebserviceB() )  
    )
```

For our toy Add, Sub webservises, let's fallback to local calculations when the webservice call doesn't reply within 3s:

```
var AddLocal = SLift(function(x, y) x+y);  
  
var AddWithFallback =  
  Defun(['x', 'y'],  
        Alt(Add(Get('x'), Get('y')),  
            Seq(Timeout(3000), AddLocal(Get('x'), Get('y')))));
```

-> http://testbed.croczilla.com:8080/fosdem2009/fibonacci_oni_fault_tolerance.html

That's it!

- Is it worth it? IMHO yes, **but**:
 - maybe only for 'large' programs with lots of concurrency
 - or for systems where you want 'pluggable' concurrency... think Ubiquity.
- Future plans:
 - new operators (Recurse, Atomic)
 - better sugaring
 - port to ActionScript (yeah, sorry)
 - port to Java
- More info on [-> http://www.croczilla.com/oni](http://www.croczilla.com/oni)