

A compiler providing incremental scalability for web application

Etienne Brodu, Stéphane Frénrot, Fabien Cellier, Frédéric Oblé

etienne.brodu@insa-lyon.fr, stephane.frenot@insa-lyon.fr, fabien.cellier@worldline.com, frederic.oble@worldline.com

To develop a **web application**, one have to **choose**

features

scalability

Monolithic models

threading

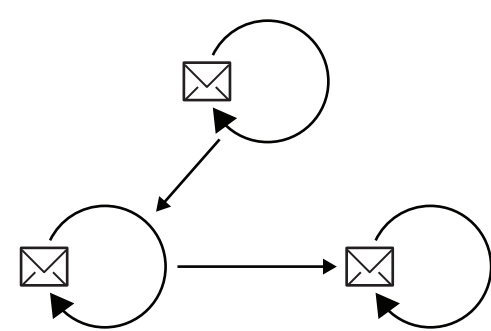
event-loop



performance

scalability

Distributed models



SEDA, System S, Storm

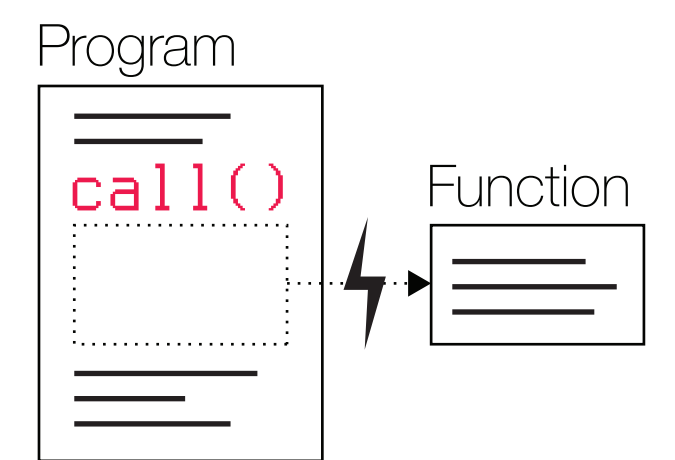
We want to **compile**  into 

The compiler extracts **autonomous parts** by searching for **rupture points** marking them out

A **Rupture point** is a call of a loosely coupled function

In **node.js**, an **asynchronous call** is a **rupture point**.

I/O Operation like **fs** and **express**.



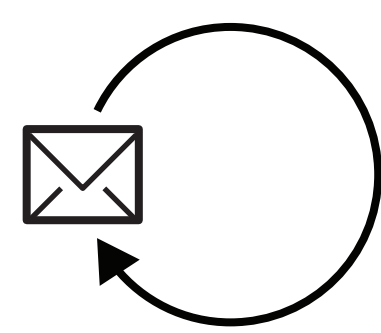
```
var app = require('express')();
app.get('/', function handler(req, res){
  //...
});
```

Application parts are enveloped into **fluxions**.

A **fluxion** is an autonomous  with :

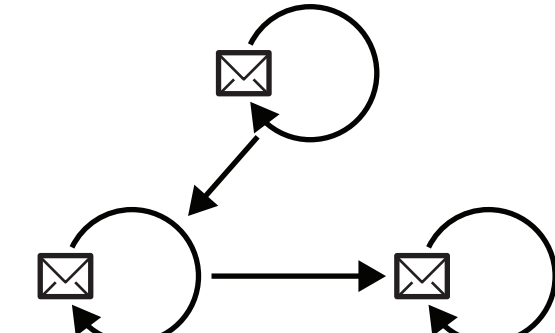
```
flx handler-1000 {fs}
-> null <-
  function handler(req, res) {
    //...
  }
```

- + a unique **name**,
- + a persisted **memory**,
- + and a **function**
- + which outputs **streams**



source

..... **compilation** ➔



target

```
var app = require('express')(),
    fs = require('fs'),
    count = 0;

app.get('/', function handler(req, res){
  fs.readFile(__filename, function reply(err, data){
    count += 1;
    var code = ('' + data)
      .replace(/\n/g, '<br>')
      .replace(/ /g, '&nbsp;');

    res.send(err
      || 'downloaded ' + count +
        ' times<br><br><code>' +
          code + '</code>');
  });
});

app.listen(8080);
console.log('>> listening 8080');
```

```
flx source.js {}
>> handler-1000 [res]
  var app = require('express')(),
      fs = require('fs'),
      count = 0;
  app.get('/', >> handler-1000);
  app.listen(8080);
  console.log('>> listening 8080');
```



```
flx handler-1000 {fs}
-> reply-1001 [res]
  function handler(req, res) {
    fs.readFile(__filename, -> reply-1001);
  }
```



```
flx reply-1001 {count}
-> null
  function reply(err, data) {
    count += 1;
    var code = ('' + data)
      .replace(/\n/g, '<br>')
      .replace(/ /g, '&nbsp;');

    res.send(err
      || 'downloaded ' + count +
        ' times<br><br><code>' +
          code + '</code>');
  }
```