Divergence Parser Combinators

Spencer Tipping

August 18, 2010

Contents

1 Introduction

The Divergence parser combinator library is a port of the Scala or Haskell parser combinators into JavaScript. The notation benefits from using Rebase (thus enabling operator overloading), but it isn't necessary. Each parser method has both an identifier name and optionally an operator name.

Listing 1 `divergence.parser.js`

```
d.rebase (function () {
  var p = d.parser = '@matcher = $0'.ctor ({
      fn: '@bound("match")'.fn(),
    fail: '{failure: $0}'.fn(),
    pass: '{stream: $0, result: $1}'.fn(),
   match: '@matcher.apply(this, @_)'.fn()});

  d.rebase.alias_in (d.parser.prototype, {
    '%$%': 'fail',
      '*': 'match'});

  d.functional(p.prototype);
```

2 Streams

A stream is an immutable list of objects. In this case, a stream is an indexed proxy over a string. It must support the `head()` and `tail()` methods, where tail() should return a new stream minus the head. Technically this could all be done using strings directly, but the head/tail notation is convenient.

Listing 2 divergence.parser.js (continued)

```
p.indexed_stream = (index, empty) >$> '@xs = $0, @position = $1 || 0'.ctor ({
  empty: '$0.call(@xs, @position)'.fn(empty.fn()),
   head: '$0.call(@xs, @position)'.fn(index.fn()),
   tail: 'new @constructor (@xs, @position + 1)'.fn()});

p.string_stream = p.indexed_stream ('@charAt($0)', '$0 >= @length');
p.array_stream  = p.indexed_stream ('$_[$0]',      '$0 >= @length');
```