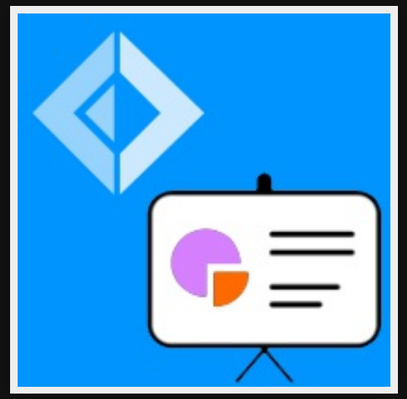
What is FsReveal?

- Generates reveal.js presentation from markdown
- Utilizes FSharp.Formatting for markdown parsing
- Get it from http://fsprojects.github.io/FsReveal/



Reveal.js

- A framework for easily creating beautiful presentations using HTML.

> ***Atwood's Law**: any application that can be written in JavaScript, will eventually be written in JavaScript.*

FSharp.Formatting

- F# tools for generating documentation (Markdown processor and F# code formatter).
- It parses markdown and F# script file and generates HTML or PDF.
- Code syntax highlighting support.
- It also evaluates your F# code and produce tooltips.

Syntax Highlighting

F# (with tooltips)

```
1: let a = 5
2: let factorial x = [1..x] |> List.reduce (*)
3: let c = factorial a
```

C#

```
1: using System;
2:
3: class Program
4: {
5:     static void Main()
6:     {
7:         Console.WriteLine("Hello, world!");
8:     }
9: }
```

JavaScript

```
 1: function copyWithEvaluation(iElem, elem) {
 2:     return function (obj) {
 3:         var newObj = {};
 4:         for (var p in obj) {
 5:             var v = obj[p];
 6:             if (typeof v === "function") {
 7:                 v = v(iElem, elem);
 8:             }
 9:             newObj[p] = v;
10:         }
11:         if (!newObj.exactTiming) {
12:             newObj.delay += exports._libraryDelay;
13:         }
14:         return newObj;
15:     };
16: }
```

Haskell

```haskell
recur_count k = 1 : 1 :
    zipWith recurAdd (recur_count k) (tail (recur_count k))
        where recurAdd x y = k * x + y

main = do
  argv <- getArgs
  inputFile <- openFile (head argv) ReadMode
  line <- hGetLine inputFile
  let [n,k] = map read (words line)
  printf "%d\n" ((recur_count k) !! (n-1))
```

code from NashFP/rosalind

SQL

```
1: select *
2: from
3: (select 1 as Id union all select 2 union all select 3) as X
4: where Id in (@Ids1, @Ids2, @Ids3)
```

sql from Dapper

Paket

```
1: source https://nuget.org/api/v2
2:
3: nuget Castle.Windsor-log4net >= 3.2
4: nuget NUnit
5:
6: github forki/FsUnit FsUnit.fs
```

C/AL

```
 1: PROCEDURE FizzBuzz(n : Integer) r_Text : Text[1024];
 2: VAR
 3:   l_Text : Text[1024];
 4: BEGIN
 5:   r_Text := '';
 6:   l_Text := FORMAT(n);
 7:
 8:   IF (n MOD 3 = 0) OR (STRPOS(l_Text,'3') > 0) THEN
 9:     r_Text := 'Fizz';
10:   IF (n MOD 5 = 0) OR (STRPOS(l_Text,'5') > 0) THEN
11:     r_Text := r_Text + 'Buzz';
12:   IF r_Text = '' THEN
13:     r_Text := l_Text;
14: END;
```

Bayes' Rule in LaTeX

$$\Pr(A|B) = \frac{\Pr(B|A)\Pr(A)}{\Pr(B|A)\Pr(A) + \Pr(B|\neg A)\Pr(\neg A)}$$

The Reality of a Developer's Life

When I show my boss that I've fixed a bug:

When your regular expression returns what you expect:

from The Reality of a Developer's Life - in GIFs, Of Course