Literate sample

Literate sample

This file demonstrates how to write literate F# script files (`*.fsx`) that can be transformed into nice HTML using the `literate.fsx` script from the F# Formatting package.

As you can see, a comment starting with double asterisk is treated as part of the document and is transformed using Markdown, which means that you can use:

- Unordered or ordered lists

- Text formatting including **bold** and *emphasis*

And numerous other Markdown features.

Writing F# code

Code that is not inside comment will be formatted as a sample snippet (which also means that you can run it in Visual Studio or MonoDevelop).

```
1:  /// The Hello World of functional languages!
2:  let rec factorial x =
3:    if x = 0 then 1
4:    else x * (factorial (x - 1))
5:
6:  let f10 = factorial 10
```

Hiding code

If you want to include some code in the source code, but omit it from the output, you can use the `hide` command.

The value will be deffined in the F# code and so you can use it from other (visible) code and get correct tool tips:

```
1:  let answer = hidden
```

Moving code around

Sometimes, it is useful to first explain some code that has to be located at the end of the snippet (perhaps because it uses some definitions discussed in the middle). This can be done using `include` and `define` commands.

The following snippet gets correct tool tips, even though it uses `laterFunction`:

```
1:   let sample =
2:     laterFunction()
3:     |> printfn "Got: %s"
```

Then we can explain how `laterFunction` is defined:

```
1:   let laterFunction() =
2:     "Not very difficult, is it?"
```

This example covers pretty much all features that are currently implemented in `literate.fsx`, but feel free to fork the project on GitHub and add more features or report bugs!

Other features

The tool-tips also work for double-backtick identifiers. This might be useful to generate nice documents from tests:

```
1:   let ``1 + 1 should be equal to 2``() =
2:     1 + 1 = 2
```

Others examples follow here.