# Formatting F# code

```fsharp
module FantomasTools.Client.ASTViewer.Model

open ASTViewer.Shared

module Graph =
    type Layout =
        | HierarchicalUpDown
        | HierarchicalLeftRight
        | Free

    type Options = {
        MaxNodes: int
        MaxNodesInRow: int
        Layout: Layout
    }

    type Model =
        { RootsPath: Node list
          Options: Options }

    // The Msg type defines what events/actions can occur while the application is running
// the state of the application changes *only* in reaction to these events
type GraphMsg =
    | SetRoot of Node
    | RootBack
    | SetOptions of Graph.Options

type Msg =
    | VersionFound of string
        | SetSourceText of string
            | DoParse
                | DoTypeCheck
                    | ASTParsed of Dto
                        | Error of string
                            | ShowJsonViewer
                                | ShowEditor
                                    | ShowRaw
                                        | ShowGraph
                                            | Graph of GraphMsg
                                                | DefinesUpdated of string
                                                    | SetFsiFile of bool
                                                        | HighLight of FantomasTools.Client.Editor.HighLightRange

type EditorState =
    | Loading
    | Loaded

type View =
    | Editor
    | JsonViewer
    | Raw
    | Graph

type Model =
    { Source: string
      Defines: string
      IsFsi: bool
```

1. Fantomas high-level
2. A real-world case
3. Other integrations

fsprojects / fantomas

Unwatch ▾ | 19    ★ Unstar | 371    Fork | 82

<> Code      Issues 66      Pull requests 3      Actions      Projects 0      Security 0      Insights      Settings

FSharp source code formatter   https://jindraivanek.gitlab.io/fantom...                    Edit

formatter    ast    fsharp    Manage topics

1,589 commits        2 branches        0 packages        23 releases        44 contributors        View license

Branch: master ▾    New pull request                    Create new file   Upload files   Find file   Clone or download ▾

nojaf Escape entire curl command as string (#771)                    ✓ Latest commit d600273 yesterday

.config                           Initial changes for FCS 34.0.1 (#671)                 2 months ago

fantomas_logo.png                 Use VS 2013 image                                    5 years ago

paket.dependencies                Fix for FCS API changes (#739)                       9 days ago

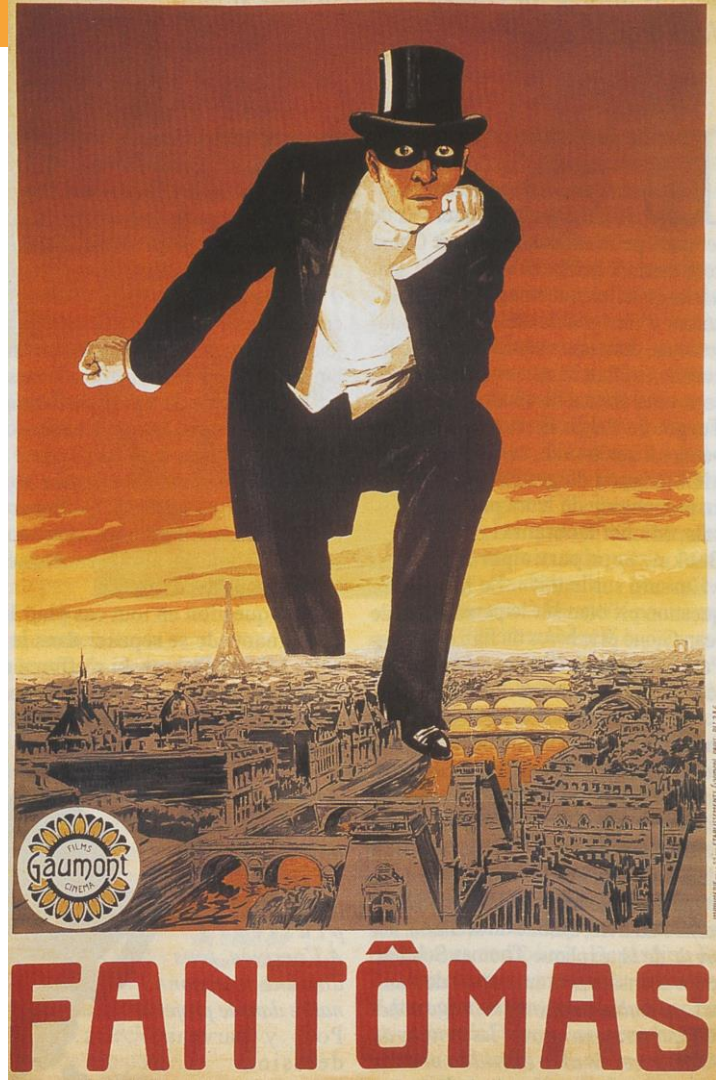paket.lock                        Fix for FCS API changes (#739)                       9 days ago

README.md                                                                                         ✎

# Fantomas



F# source code formatter, inspired by scalariform for Scala, ocp-indent for OCaml and PythonTidy for Python.

Build master passing    BUILDING...    chat on gitter

# Fantomas

```
string -> FormatConfig -> string
```
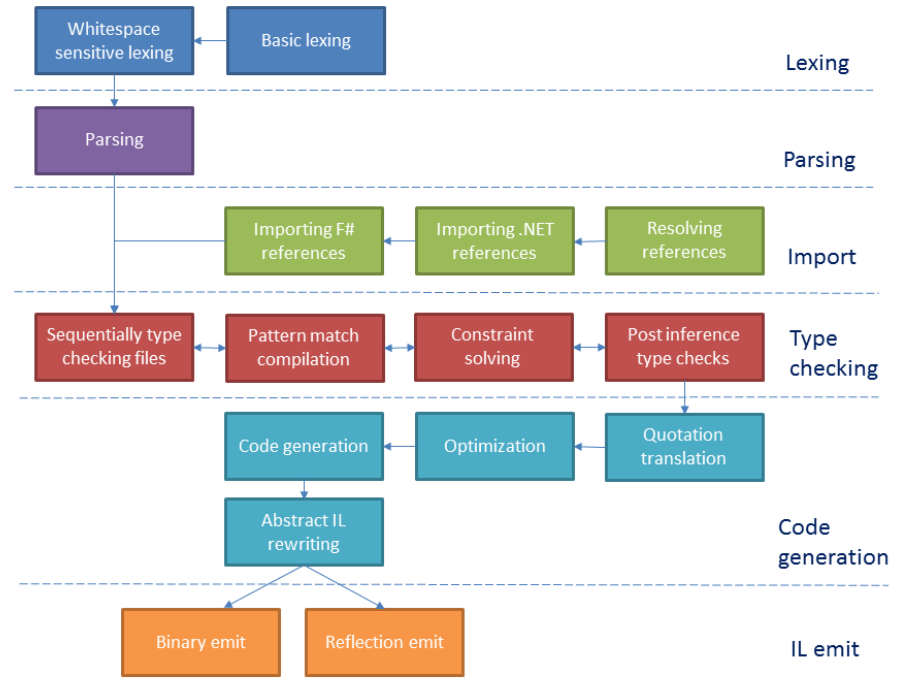
# Fantomas internal

- FSharp.Compiler.Service
  - Tokenizer
  - Untyped AST

## Key Compiler Phases

Below, you can see a diagram of how different phases of F# compiler work:

| | |
|---|---|
| Whitespace sensitive lexing | Basic lexing | **Lexing** |

Parsing — **Parsing**

Importing F# references ← Importing .NET references ← Resolving references — **Import**

Sequentially type checking files → Pattern match compilation → Constraint solving → Post inference type checks — **Type checking**

Code generation ← Optimization ← Quotation translation — **Code generation**

Abstract IL rewriting

Binary emit | Reflection emit — **IL emit**

# Fantomas internal

1. AST
2. Trivia
3. Writer Events
4. Merge result

```fsharp
/// Formatting F# code
/// A case study

type FormatConfig = {
    PageWidth: int
    Indent: int } // The number of spaces

let binaryUIntValue = 0x4000u

type ILogger =
    member this.LogDebug message = printfn (* TODO: implement something *) ()

let run (req: HttpRequest) (log: ILogger) =
    Http.main getFantomasVersion format FormatConfig.FormatConfig.Default log req
    |> Async.StartAsTask
```

```fsharp
1   /// Formatting F# code
2   /// A case study
3
4   type FormatConfig = {
5       PageWidth: int
6       Indent: int } // The number of spaces
7
8   let binaryUIntValue = 0x4000u
9
10  type ILogger =
11      member this.LogDebug message = printfn (* TODO: implement something *) ()
12
13  let run (req: HttpRequest) (log: ILogger) =
14      Http.main getFantomasVersion format FormatConfig.FormatConfig.Default log req
15      |> Async.StartAsTask
```

RUN    .NET Core Launch (console)

launch.json   tasks.json   ≡ CodeFormatterImpl.fs ✕   ⓘ fsharp.json

src › Fantomas › ≡ CodeFormatterImpl.fs › {} Fantomas.CodeFormatterImpl › ⬡ formatWith

**VARIABLES**

- WriterEvents: {Fantomas.Queue<Fantomas.Context.WriterE...
  - › Front [FSharpList]: Length = 1
  - › Head [WriterEvent]: Write "/// Formatting F# code"
    - IsEmpty [bool]: false
    - Length [int]: 119
  - › RBack [FSharpList]: Length = 118
  - › Tail: {Fantomas.Queue<Fantomas.Context.WriterEvent>}
  - › TryHead [FSharpOption]: error CS1546: Property, index...
  - › TryTail [FSharpOption]: error CS1546: Property, index...
  - › TryUncons [FSharpOption]: error CS1546: Property, ind...
  - › Uncons [Tuple]: {(Write "/// Formatting F# code", Fan...
  - › front [FSharpList]: Length = 1
    - hashCode [FSharpOption]: null
  - › rBack [FSharpList]: Length = 118
  - ⌄ Results View: Expanding the Results View will enumera...
    - › [0] [object]: Write "/// Formatting F# code"
    - [1] [object]: WriteLine
    - › [2] [object]: Write ""
    - › [3] [object]: Write "/// A case study"
    - [4] [object]: WriteLine
    - › [5] [object]: Write ""
    - [6] [object]: WriteLine
    - › [7] [object]: Write ""
    - › [8] [object]: Write "type "
    - › [9] [object]: Write "FormatConfig"

**WATCH**

```
378  ----}
379
380  let formatWith ast defines formatContext config = // ParsedInput -> string
381  ----let moduleName = Path.GetFileNameWithoutExtension formatContext.FileNa
382  ----let input =
383  --------if String.IsNullOrWhiteSpace formatContext.Source then None
384  --------else Some formatContext.Source
385  --------
386  ----// Use '\n' as the new line delimiter consistently
387  ----// It would be easier for F# parser
388  ----let sourceCode = defaultArg input String.Empty
389  ----let normalizedSourceCode = String.normalizeNewLine sourceCode
390  ----let formattedSourceCode =
391  --------let context = Fantomas.Context.Context.Create config defines norma
392  --------context |> genParsedInput { ASTContext.Default with TopLevelModuleN
393  --------|> Dbg.tee (fun ctx -> printfn "%A" ctx.WriterEvents)
394  --------|> Context.dump
395  //--------|> if config.StrictMode then id
396  //--------else integrateComments config formatContext.ProjectOptions.Co
397
398  ----// Sometimes F# parser gives a partial AST for incorrect input
399  ----if input.IsSome && String.IsNullOrWhiteSpace normalizedSourceCode <> S
400  --------raise <| FormatException "Incomplete code fragment which is most l
401  ----else formattedSourceCode
402  ----
403  ----|> String.removeTrailingSpaces
404
405  let format (checker: FSharpChecker) (parsingOptions: FSharpParsingOptions)
406  ----async {
407  --------let! asts = parse checker parsingOptions formatContext
408  --------let results =
```

Formatting in action

Multiple defines

Settings

How to use?

Fantomas DIY

Fantomas as library

What's next?

What's next?

Search

Home

Trending

Subscriptions

Library

History

Your videos

Watch later

Liked videos

Show more

▶ PLAY ALL
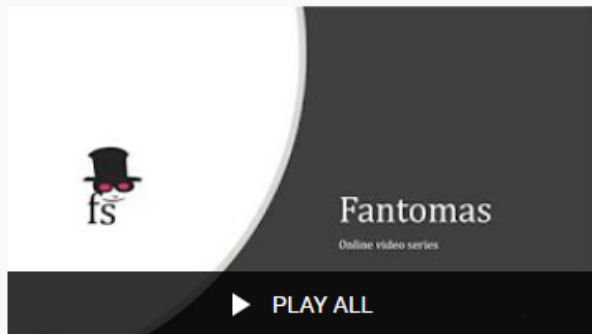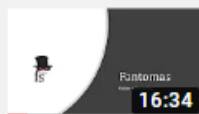
## Fantomas ✏

7 videos • 72 views • Updated 7 days ago

Public ▾

⤬  ↱  ⋯

SORT BY
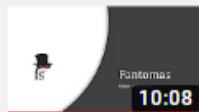
**Fantomas - Untyped AST**
nojaf
16:34

**Fantomas - Context & WriterEvents**
nojaf
7:46

**Fantomas - Trivia**
nojaf
10:08

**Fantomas - Conditional Compiler Directives**

# Florian Verdonck

.NET consultant @ Axxes

@verdonckflorian

@nojaf

# The end

IT IS ABOUT PEOPLE