

Benoit Prioux

benoit.prioux@gmail.com

Asciidoctor et Java

Me

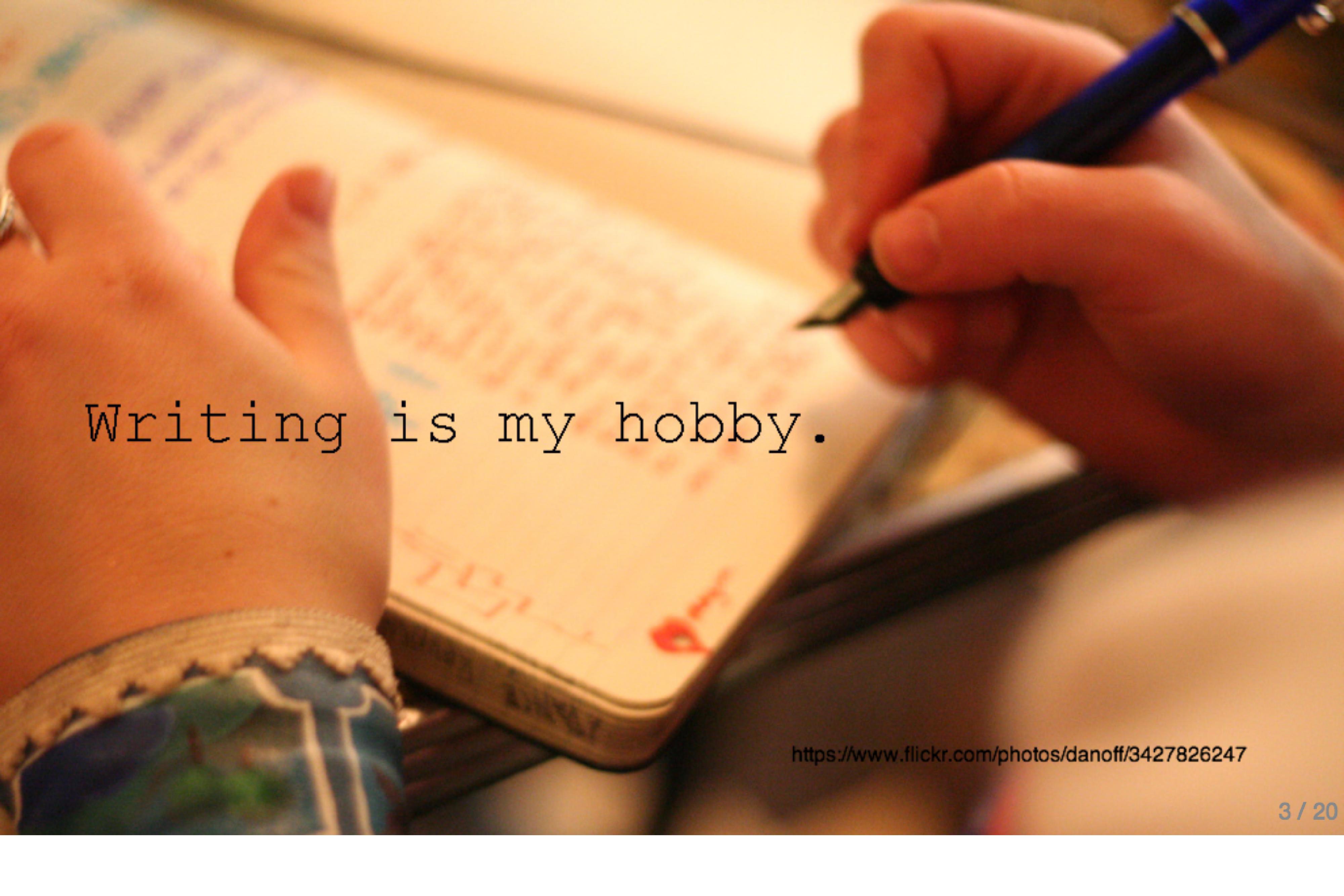
 @binout

 <https://github.com/binout>



- Développeur Java depuis presque 10 ans
- Technical Leader chez Lectra, numéro un mondial des solutions dédiées à l'industrie du textile (machines et logiciels)
- Membre du Bordeaux JUG
 - Conférence bdx.io le 17 octobre !





Writing is my hobby.

<https://www.flickr.com/photos/danoff/3427826247>

Une solution : Asciidoc

Langage de *balisage*, créé en 2002, processeur en Python



Une solution : Asciidoc

Langage de *balisage*, créé en 2002, processeur en Python



Parce que :

Une solution : Asciidoc

Langage de *balisage*, créé en 2002, processeur en Python



Parce que :

- on se concentre plus sur le fond que sur la forme

Une solution : Asciidoc

Langage de *balisage*, créé en 2002, processeur en Python



Parce que :

- on se concentre plus sur le fond que sur la forme
- c'est du texte, donc un éditeur classique suffit

Une solution : Asciidoc

Langage de *balisage*, créé en 2002, processeur en Python



Parce que :

- on se concentre plus sur le fond que sur la forme
- c'est du texte, donc un éditeur classique suffit
- on peut gérer l'historique avec un SCM

Une solution : Asciidoc

Langage de *balisage*, créé en 2002, processeur en Python



Parce que :

- on se concentre plus sur le fond que sur la forme
- c'est du texte, donc un éditeur classique suffit
- on peut gérer l'historique avec un SCM
- à partir d'une même source, on peut publier vers plusieurs formats

Une solution : Asciidoc

Langage de *balisage*, créé en 2002, processeur en Python



Parce que :

- on se concentre plus sur le fond que sur la forme
- c'est du texte, donc un éditeur classique suffit
- on peut gérer l'historique avec un SCM
- à partir d'une même source, on peut publier vers plusieurs formats
- on a un peu l'impression de hacker ;-)

Exemple

```
= Hello, Jug Summer Camp!  
Benoit Prioux <benoit.prioux@gmail.com>
```

```
Introduction à http://asciidoc.org[AsciiDoc].
```

```
== Première Section
```

- * foo
- * bar

```
== Deuxième Section
```

1. item 1
2. item 2

```
NOTE: C'est l'heure de la démo !
```

Asciidoctor

<https://github.com/asciidoctor>

- Implémentation open-source écrite en Ruby
- Permet de convertir des fichiers asciidoc vers différents backends : docbook, html5, epub, pdf et même deckjs !
- 12 août 2014 : sortie de la version 1.5.0
 - 1^{ère} version majeure après 2 ans de développement
 - 50 contributeurs, 1800 commits, 1500 tests



Dan Allen
mojavelinux



Ils utilisent déjà asciidoctor



O'REILLY®



Et si je ne connais pas Ruby, ce n'est pas pour moi ?



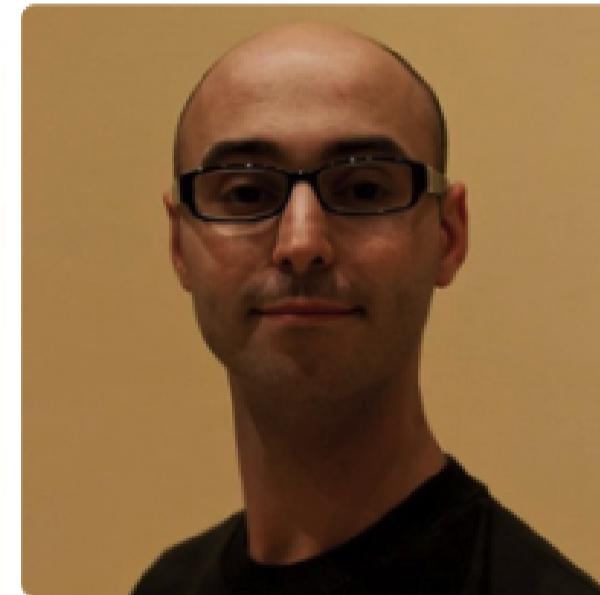
**Et si je ne connais pas Ruby, ce n'est
pas pour moi ?**



Et si je ne connais pas Ruby, ce n'est pas pour moi ?



+



Alex Soto
lordofthejars

= AsciidoctorJ

Asciidoctorj en action !

```
Asciidoctor asciidoctor = Asciidoctor.Factory.create();  
  
Options options = options().backend("html5").get();  
String rendered = asciidoctor.convert("*Gras* ou  
_italique_?", options);  
  
System.out.println(rendered);
```



Asciidoctorj en action !

```
Asciidoctor asciidoctor = Asciidoctor.Factory.create();  
  
Options options = options().backend("html5").get();  
String rendered = asciidoctor.convert("*Gras* ou  
_italique_?", options);  
  
System.out.println(rendered);
```

Console

```
<div class="paragraph">  
<p><strong>Gras</strong> ou <em>italique</em> ?</p>  
</div>
```

Asciidoctor et Maven

pom.xml

```
<plugin>
  <groupId>org.asciidoctor</groupId>
  <artifactId>asciidoctor-maven-plugin</artifactId>
  <version>1.5.0</version>
  <executions>
    <execution>
      <id>output-html</id>
      <phase>generate-resources</phase>
      <goals>
        <goal>process-asciidoc</goal>
      </goals>
    </execution>
  </executions>
</plugin>
```

Asciidoctor et Gradle

build.gradle

```
buildscript {  
    repositories {  
        jcenter()  
    }  
  
    dependencies {  
        classpath 'org.asciidoctor:asciidoctor-gradle-  
plugin:1.5.0'  
    }  
}  
  
apply plugin: 'org.asciidoctor.gradle.asciidoctor'
```



Asciidoctor et Ant (*unofficial*)

<https://github.com/binout/asciidoctor-ant>

```
<target name="doc">
  <taskdef
    resource="net/jtools/classloadertask/antlib.xml"
    classpath="lib/ant-classloadertask.jar"/>
    <classloader loader="thread"
    classpath="lib/asciidoctor-ant.jar"/>

    <taskdef name="asciidoctor"
    classname="org.asciidoctor.ant.AsciidoctorAntTask"/>
      <asciidoctor sourceDirectory="src/asciidoc"
    outputDirectory="build/docs"/>
  </target>
```

Asciidoctor et Javadoc

Javadoc avec Asciidoclet

```
/**  
 * This class has the following  
 features:  
 *  
 * - Support for *foo*  
 * - Support for bar  
 */
```

```
public class Thing implements  
Something { ... }
```

The screenshot shows a Java documentation page for the class `Thing`. The top navigation bar includes links for PACKAGE, CLASS (which is highlighted in orange), USE, TREE, DEPRECATED, INDEX, and HELP. Below the navigation bar are links for PREV CLASS, NEXT CLASS, FRAMES, and NO FRAMES. Further down are links for SUMMARY, NESTED, FIELD, CONSTR, and METHOD, and DETAIL, FIELD, CONSTR, and METHOD.

The main content area starts with the package name `net.binout.asciidoc`. The class is identified as **Class Thing**, which extends `java.lang.Object` and implements `Something`. A section titled "All Implemented Interfaces:" lists `Something`.

Below this, the class definition is shown in code:

```
public class Thing  
extends Object  
implements Something
```

A note states: "This class has the following features:" followed by a bulleted list:

- Support for foo
- Support for bar

There are three tabs at the bottom: "Constructor Summary" (highlighted in orange), "Constructors" (grayed out), and "Constructor and Description". The constructor `Thing()` is listed under the "Constructor and Description" tab.

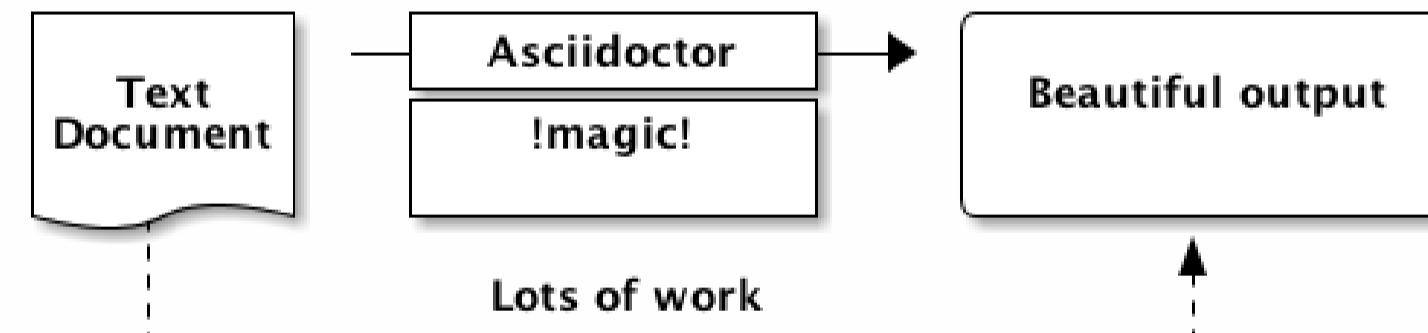
At the very bottom is a "Method Summary" tab.

Asciidoctor est extensible ...



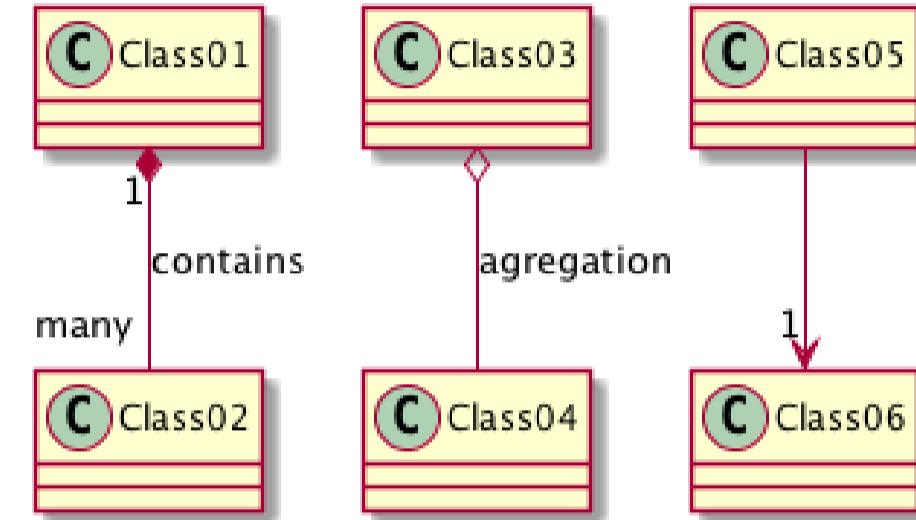
asciidocor-diagram

```
[ditaa]
-----
+-----+ +-----+----+ /-----\
|     | --- Asciidocor ---> | |
| Text | +-----+ |Beautiful output|
| Document| !magic! | |
| {d}| | |
+-----+ +-----+ \-----/
:
| Lots of work |
+-----+
-----
```



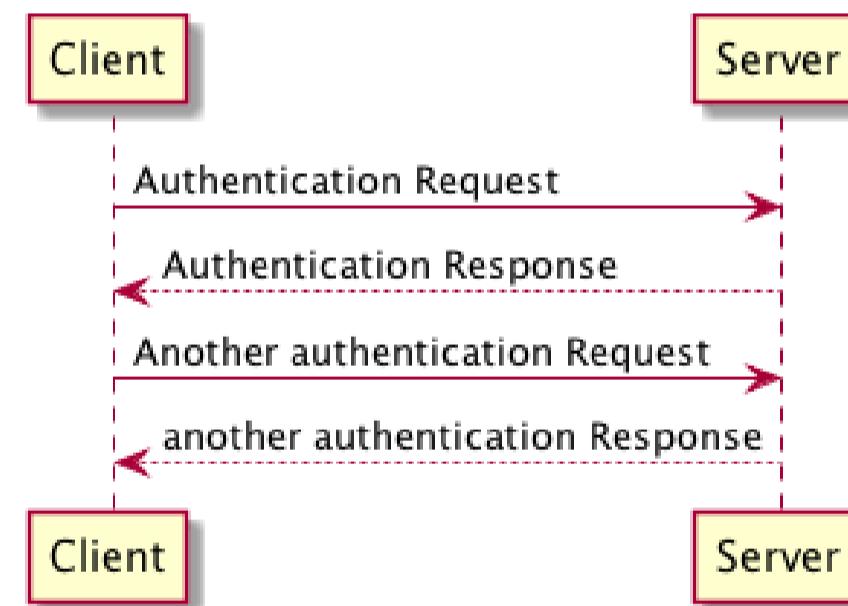
asciidocdoctor-diagram et UML

```
[plantuml]
-----
Class01 "1" *-- "many" Class02 : contains
Class03 o-- Class04 : aggregation
Class05 --> "1" Class06
-----
```



```
[plantuml]
-----
Client -> Server: Authentication Request
Server --> Client: Authentication Response

Client -> Server: Another authentication Request
Client <-- Server: another authentication Response
-----
```



Extensions ruby en java

Maven proxy : <http://rubygems-proxy.torquebox.org/>

gem asciidoctor-diagram

```
<dependency>
  <groupId>rubygems</groupId>
  <artifactId>asciidoctor-diagram</artifactId>
  <version>1.2.0</version>
  <type>gem</type>
  <scope>provided</scope>
<dependency>
```



AsciidoctorJ et extensions ruby

asciidoc-maven-plugin

```
<plugin>
  <groupId>org.asciidoctor</groupId>
  <artifactId>asciidoctor-maven-plugin</artifactId>
  <version>${asciidoctor.version}</version>
  <configuration>
    <gemPath>${project.build.directory}/gems-
provided</gemPath>
    <requires>
      <require>asciidoctor-diagram</require>
    </requires>
  </configuration>
  ...
</plugin>
```

Et si je veux faire une extension, dois-je coder en ruby ?



Et si je veux faire une extension, dois-je coder en ruby ?

Nouveau : on peut coder des extensions directement en Java, Groovy, Scala



Et si je veux faire une extension, dois-je coder en ruby ?

Nouveau : on peut coder des extensions directement en Java, Groovy, Scala

<http://mrhaki.blogspot.fr/2014/08/awesome-asciidoc-write-extensions-using.html>



Et si je veux faire une extension, dois-je coder en ruby ?

Nouveau : on peut coder des extensions directement en Java, Groovy, Scala

<http://mrhaki.blogspot.fr/2014/08/awesome-asciidoc-write-extensions-using.html>

twitter:binout[]

Et si je veux faire une extension, dois-je coder en ruby ?

Nouveau : on peut coder des extensions directement en Java, Groovy, Scala

<http://mrhaki.blogspot.fr/2014/08/awesome-asciidoc-write-extensions-using.html>

twitter:binout[]

devient :

Et si je veux faire une extension, dois-je coder en ruby ?

Nouveau : on peut coder des extensions directement en Java, Groovy, Scala

<http://mrhaki.blogspot.fr/2014/08/awesome-asciidoc-write-extensions-using.html>

twitter:binout[]

devient :

```
<a href="http://www.twitter.com/binout">@binout</a>
```

Merci

<https://github.com/binout/asciidoc-quickie/>

Asciidoctor User Manual

Sarah White – [@carbonfray](#) · Dan Allen – [@mojavelinux](#)



This document is under active development and discussion!

If you find errors or omissions in this document, please don't hesitate to [submit an issue or open a pull request](#) with a fix. We also encourage you to ask questions and discuss any aspects of the project on the [mailing list](#) or [IRC](#). New contributors are always welcome!

This manual assumes you are using Asciidoctor to produce and render your document. Asciidoctor implements more syntax, attributes and functions than the legacy AsciiDoc.py processor. [Appendix A](#) lists which features are available to the Asciidoctor and AsciiDoc processors.

Introduction to Asciidoctor



Section Pending

1. What is Asciidoctor?

Asciidoctor is a *fast* text processor and publishing toolchain for converting AsciiDoc content to HTML5, EPUB3, PDF, DocBook 5 (or 4.5) slidedecks and other formats. Asciidoctor is written in Ruby, packaged as a RubyGem and published to [RubyGems.org](#). The gem is also packaged in several Linux distributions, including Fedora, Debian and Ubuntu. Asciidoctor is open source, [hosted on GitHub](#), and released under the MIT license.

Table of Contents

Introduction to Asciidoctor

1. What is Asciidoctor?
 - 1.1. The Big Picture
 - 1.2. Asciidoctor on the JVM
 - 1.3. Asciidoctor.js
 - 1.4. Asciidoctor's most notable benefits
 - 1.5. Compared to AsciiDoc
 - 1.6. Compared to MarkDown

Quick Starts

2. Installation Quick Start
3. Usage Quick Start
 - 3.1. Using the Command Line Interface
 - 3.2. Using the Ruby API
4. Syntax Quick Start
5. Custom Output Quick Start

Getting Started

6. System Requirements
7. Installing the Asciidoctor Ruby Gem
 - 7.1. Install with Bundler
 - 7.2. Install with yum
 - 7.3. Install with apt-get
8. Upgrading the Asciidoctor Ruby Gem
9. Extensions and Integrations