εχTEX

Under the Hood

Gerd Neugebauer, Michael Niedermair

EuroTEX 2005
March 2005
Pont-à-Mousson, France

History

- $\varepsilon_{\mathcal{X}}$TEX started as attempt to enhance $\mathcal{NTS}$
- Immediate performance improvements of $\mathcal{NTS}$
- $\mathcal{NTS}$ is not considered as a good base for extensions:
  - Too close to TEX: direct mapping of the internals
  - Not really designed modular or object-oriented
  - Not designed for extension
- 2003: Decision to start from scratch
- Intermediate use of some $\mathcal{NTS}$ classes.
  Reimplemented in the meantime
- Since 2004 public at `http://www.extex.org`
  with a CVS repository at Berlios

Developers

- Michael Niedermair
- Gerd Neugebauer
- Sebastian Waschik
- Rolf Niepraschk
- (Andre Wrobst)

Goals

- $\varepsilon_{\mathcal{X}}$TEX should provide a TEX-compatible typesetting system.
  - The result should look the same.
  - If configured differently the result may be "better".
  - The compatibility only holds for inputs which are processed without errors.
  - Compatibility of the log files is not guaranteed at all.
  - Compatibility does not mean identical output files (dvi, pdf,...)

Goals (2)

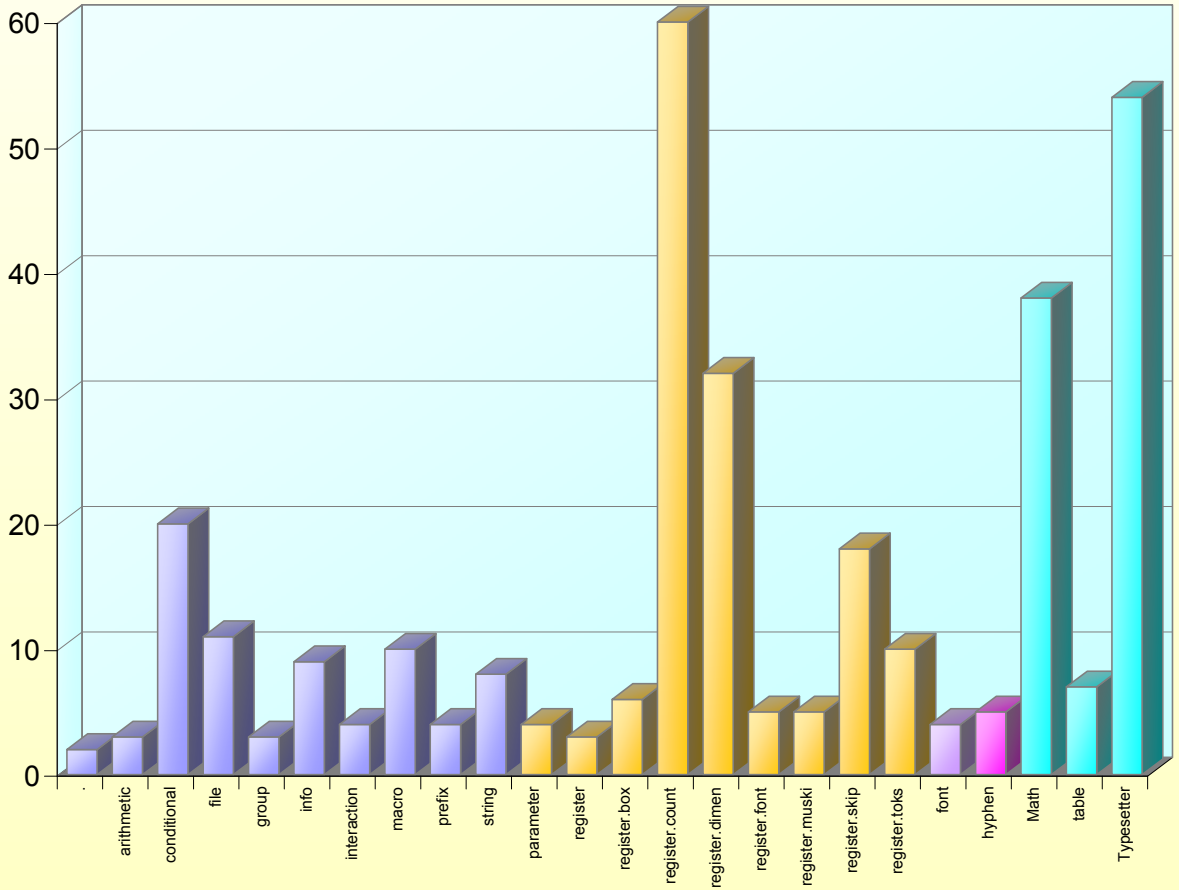
- εχTEX should be extensible and configurable.
  - Extension with new low-level primitives.
  - Extension of existing primitives.
  - Extension with additional document writers.
  - Extension with new font types.
  - Extension with new typesetters.
  - TEX-compatibility mode is a matter of configuration.

Goals (3)

- εχTEX should be different right from the start.
  - Avoiding (some of) the design flaws of TEX
  - 32-bit Unicode characters as internal representation
  - Long (at least 32-bit) count and dimen registers
  - Characters carry a typographic context
  - Integration of the best of ε-TEX, pdfTEX, and Omega
  - LR and RL typesettng build in
  - No restrictions on the number of registers
  - No distinction like TEX and iniTEX

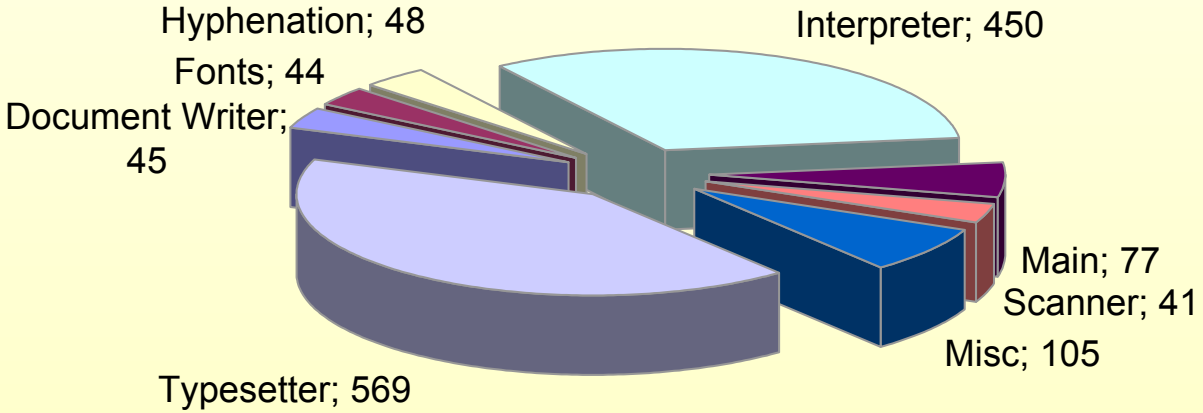
(ini)TEX in Numbers

Control sequences:	325
Register primitives:	143



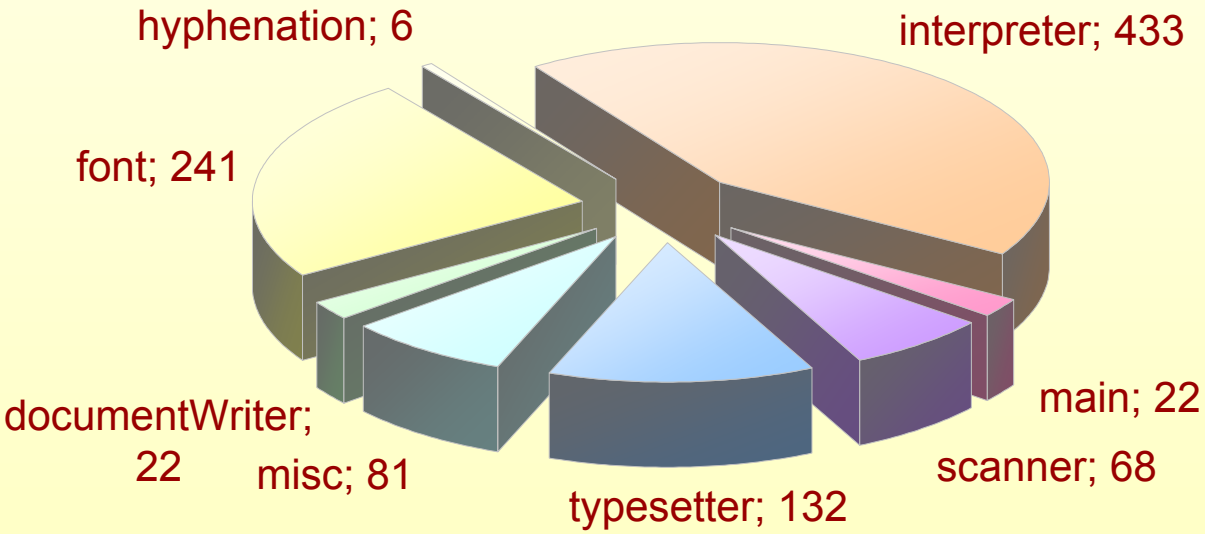
"TEX the Program" in Numbers

Sections: 1380

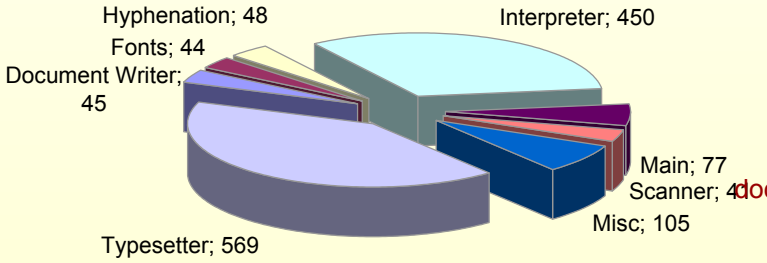


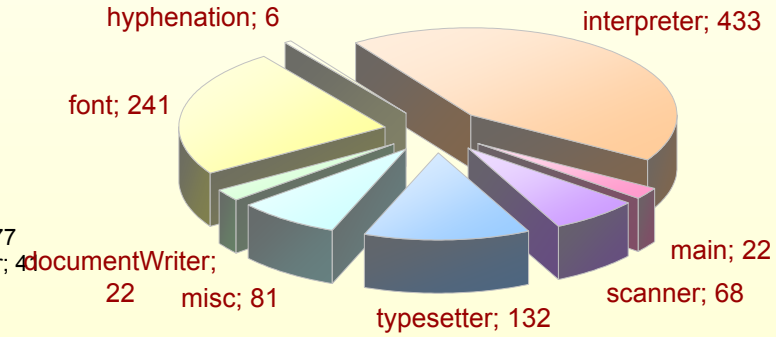
εχTEX in Numbers

Java-Interfaces:	109
Java-Classes:	896
Lines of code:	38370
Properties:	81
Configurations:	13

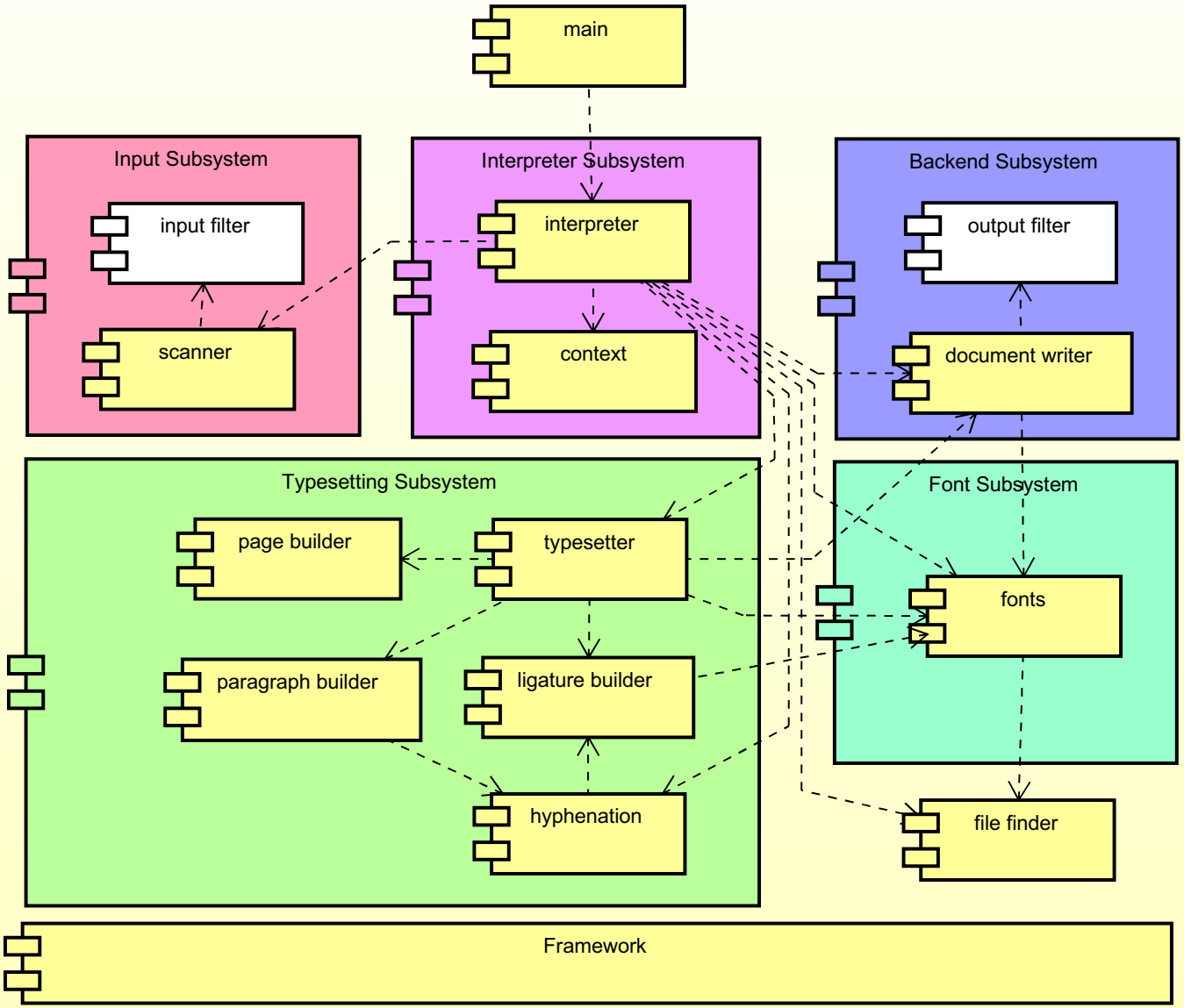


Numbers, Side by Side





Overview εχTEX(2)



Overview $\varepsilon\chi$TEX

- Composition of several components
- Components defined via interfaces
- Sometimes several implementations
- Component framework based on the ideas of Apache Avalon
- Infrastructure functionalty provided by the framework
  - Initialization
  - Logging
  - Configuration

Interpreter Context

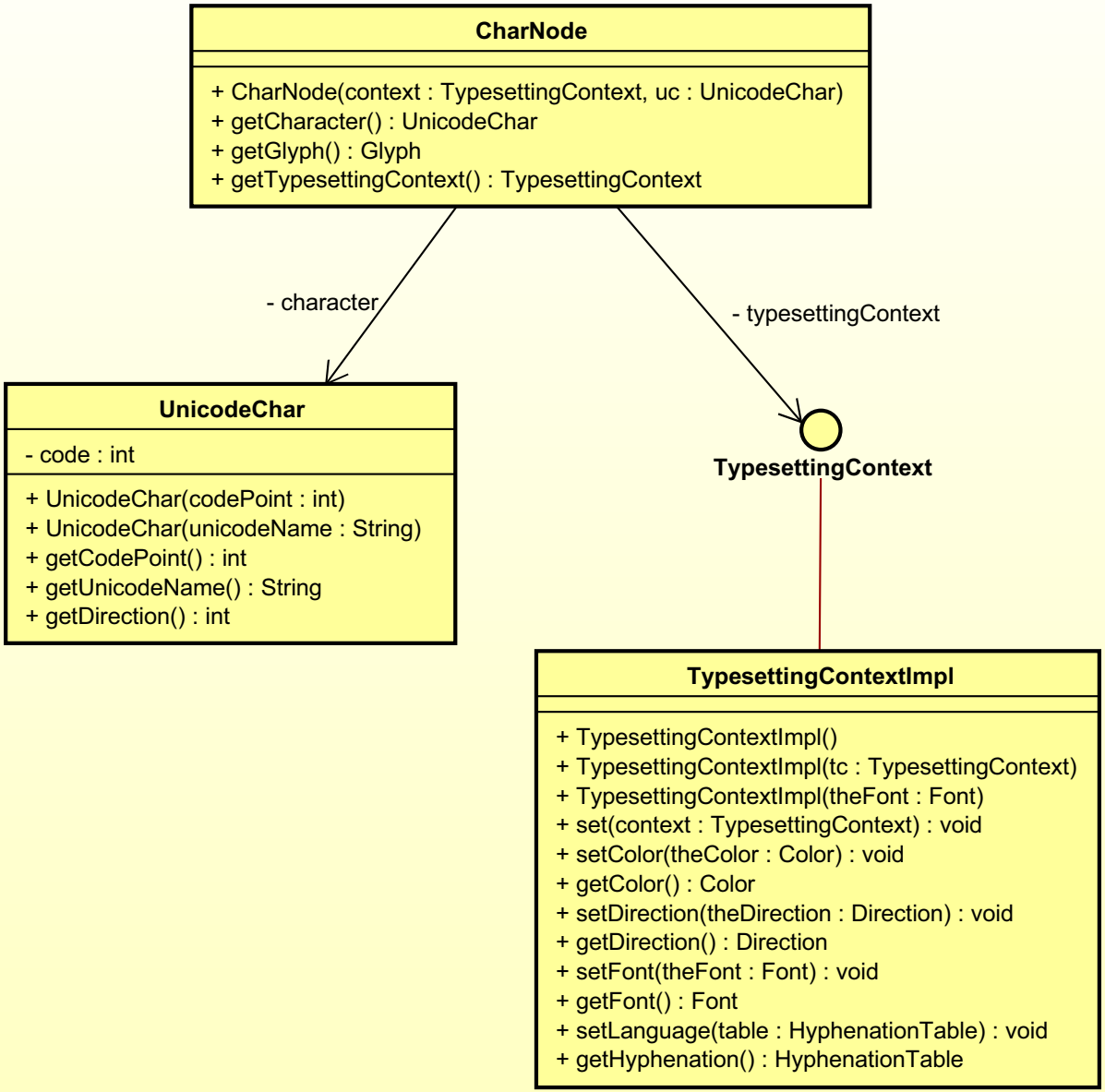
<<interface>>
Context

```
+ esc(name : String) : String
+ esc(token : Token) : String
+ escapechar() : char
+ expand(tokens : Tokens, typesetter : Typesetter) : Tokens
+ getAfterassignment() : Token
+ getBox(name : String) : Box
+ getCode(t : CodeToken) : Code
+ getCount(name : String) : Count
+ getDelcode(c : UnicodeChar) : Count
+ getGlue(name : String) : Glue
+ getHyphenationTable(language : int) : HyphenationTable
+ getId() : String
+ getInteraction() : Interaction
+ getLccode(uc : UnicodeChar) : UnicodeChar
+ getMagnification() : long
+ getMathcode(uc : UnicodeChar) : Count
+ getMuskip(name : String) : Muskip
+ getNamespace() : String
+ getParshape() : ParagraphShape
+ getSfcode(uc : UnicodeChar) : Count
+ getTokenFactory() : TokenFactory
+ getTokenizer() : Tokenizer
+ getToks(name : String) : Tokens
+ getTypesettingContext() : TypesettingContext
+ getUccode(lc : UnicodeChar) : UnicodeChar
+ popConditional() : Conditional
+ pushConditional(locator : Locator, value : boolean) : void
+ registerCodeChangeObserver(observer : CodeChangeObserver, name : Token) : void
+ setAfterassignment(token : Token) : void
+ setBox(name : String, value : Box, global : boolean) : void
+ setCatcode(c : UnicodeChar, cc : Catcode, global : boolean) : void
+ setCode(t : CodeToken, code : Code, global : boolean) : void
+ setCount(name : String, value : long, global : boolean) : void
+ setDelcode(c : UnicodeChar, code : Count, global : boolean) : void
+ setGlue(name : String, value : Glue, global : boolean) : void
+ setId(id : String) : void
+ setInteraction(interaction : Interaction, global : boolean) : void
+ setLccode(uc : UnicodeChar, lc : UnicodeChar) : void
+ setMagnification(mag : long) : void
+ setMathcode(uc : UnicodeChar, code : Count, global : boolean) : void
+ setMuskip(name : String, value : Muskip, global : boolean) : void
+ setNamespace(namespace : String, global : boolean) : void
+ setParshape(shape : ParagraphShape) : void
+ setSfcode(uc : UnicodeChar, code : Count, global : boolean) : void
+ setStandardTokenStream(standardTokenStream : TokenStream) : void
+ setTokenFactory(factory : TokenFactory) : void
+ setToks(name : String, toks : Tokens, global : boolean) : void
+ setTypesettingContext(color : Color) : void
+ setTypesettingContext(direction : Direction) : void
+ setTypesettingContext(font : Font) : void
+ setTypesettingContext(context : TypesettingContext) : void
+ setTypesettingContext(context : TypesettingContext, global : boolean) : void
+ setUccode(lc : UnicodeChar, uc : UnicodeChar) : void
+ unregisterCodeChangeObserver(observer : CodeChangeObserver, name : Token) : void
```

Interpreter Context (2)

- The Context contains the state of the interpreter
  - Catcodes
  - Count register
  - Dimen register
  - Box registers
  - ...
- Group handling is encapsulated in the Context
- The format contains mainly the Context
- The context is provided to several components under different Interfaces

Character Nodes



Character Nodes (2)

- Each character node carries a typographic context
- The typographic context carries
  - font
  - language
  - color
  - direction
- Any switching problems across pages can not be reproduced with this scheme

Configuring εχTEX

Java - extex.xml - Eclipse Platform

File Edit Navigate Search Project Run Window Help

extex.xml

```
<!--
  -  The section "Interpreter" contains the specification for the core
  -  interpreter.
  -->
<Interpreter class="de.dante.extex.interpreter.max.Max">

  <!-- The maximum number of errors before an abort is forced.
    -->
  <maxErrors>100</maxErrors>

  <!-- Tokens to be inserted at every job.
    -->
  <everyjob></everyjob>

  <!-- The default font if nothing else is done.
    -  The attribute "name" can be used to specify the font name. Default: nullfont
    -  The attribute "size" can be used to specify the size. Default: design size.
    -->
  <Font name=""/>

  <!-- The configuration of the token factory.
    -->
  <TokenFactory class="de.dante.extex.scanner.type.TokenFactoryImpl"/>

  <!-- The configuration of the context implementation to use.
    -->
  <Context default="ExTeX">
    <ExTeX class="de.dante.extex.interpreter.context.impl.ContextImpl">
      <!-- The implementation of the group to use.
        -->
      <Group class="de.dante.extex.interpreter.context.impl.GroupImpl"/>

      <!-- The implementation of the typesetting context to use.
        -->
      <TypesettingContext class="de.dante.extex.interpreter.context.TypesettingContext

      <Language></Language>
    </ExTeX>

  </Context>

  <!-- The implementation of the error handler to use.
    -->
  <ErrorHandler default="ExTeX">
```

ExTeX/src/java/config/extex.xml | Writable | Smart Insert | 103 : 20 | Xml

Configuring $\varepsilon_{\mathcal{X}}$TEX (2)

- XML-based configuration files
- Provides mapping from logical names to the implementation
- Allows the selection of alternatives
- Key-value pairs for user settings (`.extex`)

Configuring Primitives: config/tex.xml

```
<?xml version="1.0"?>

<primitives>

  <define name="␣"
   class="de.dante.extex.interpreter.primitives.typesetter.spacing.Space"/
  <define name="/"
   class="de.dante.extex.interpreter.primitives.typesetter.spacing.ItalicC
  <define name="\\"
   class="de.dante.extex.interpreter.primitives.typesetter.paragraph.NewLi
  <define name="above"
   class="de.dante.extex.interpreter.primitives.math.fraction.Above"/>
  <define name="abovedisplayshortskip"
   class="de.dante.extex.interpreter.primitives.register.skip.SkipParamete
  <define name="abovedisplayskip"
   class="de.dante.extex.interpreter.primitives.register.skip.SkipParamete
  <define name="abovewithdelims"
   class="de.dante.extex.interpreter.primitives.math.fraction.Abovewithdel
  <define name="accent"
   class="de.dante.extex.interpreter.primitives.typesetter.Accent"/>
  <define name="adjdemerits"
   class="de.dante.extex.interpreter.primitives.register.count.IntegerPara
  <define name="advance"
   class="de.dante.extex.interpreter.primitives.arithmetic.Advance"/>
  <define name="afterassignment"
   class="de.dante.extex.interpreter.primitives.register.Afterassignment"/
```

Extending $\varepsilon_{\mathcal{X}}$TEX

Extending $\varepsilon_{\mathcal{X}}$TEX can mean

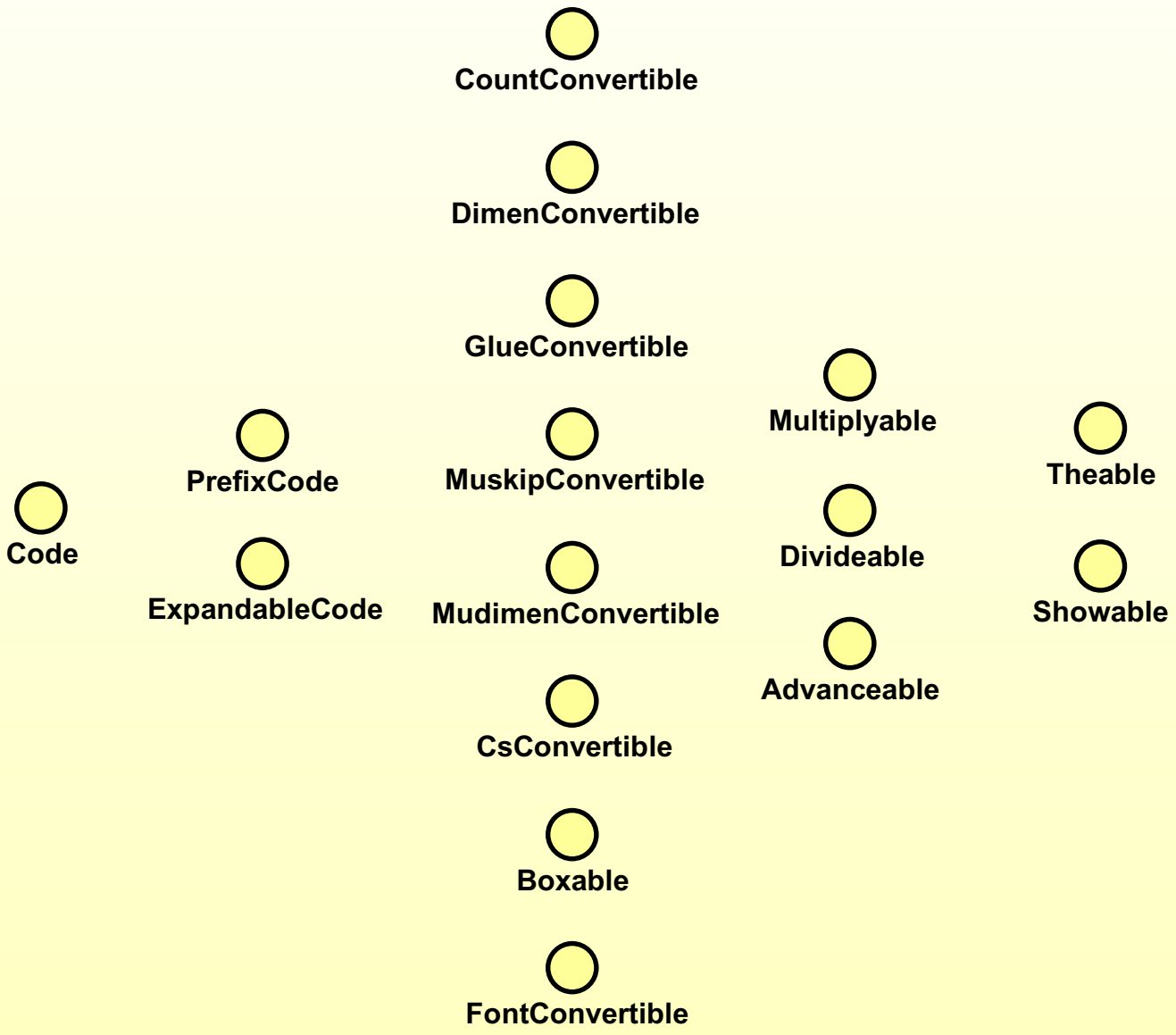
- Providing a new primitive
  - Write a new primitive (in Java)
  - Register the new primitive in a (copy of a) configuration
  - Use the new configuration when running $\varepsilon_{\mathcal{X}}$TEX
- Providing an alternative implementation for some component
  - Write a new implementation (in Java)
  - Register the new implementation in a configuration
  - Use configuration when running $\varepsilon_{\mathcal{X}}$TEX

New Primitives

- New primitives can be integrated in $\varepsilon\chi$TEX
- Implementation language can be Java
- Minor restrictions have to be honoured
- Some infrastructure is provided by $\varepsilon\chi$TEX

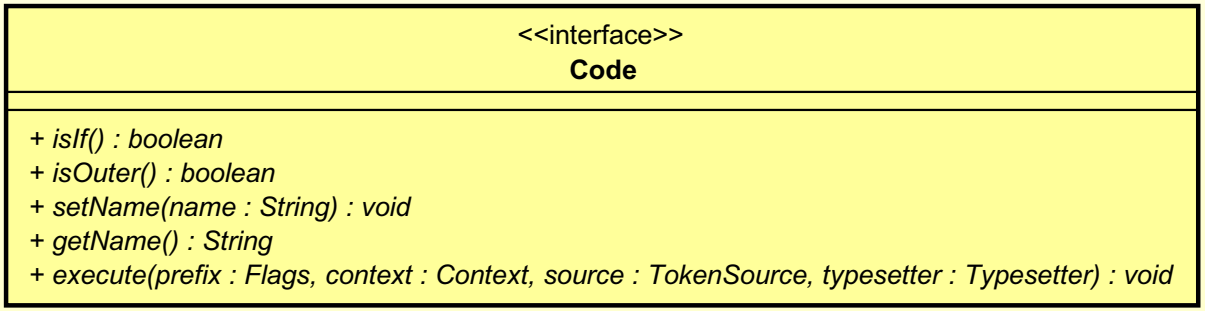
The Code Interfaces

- Interfaces describe the possible features of a primitive



The Code Interface

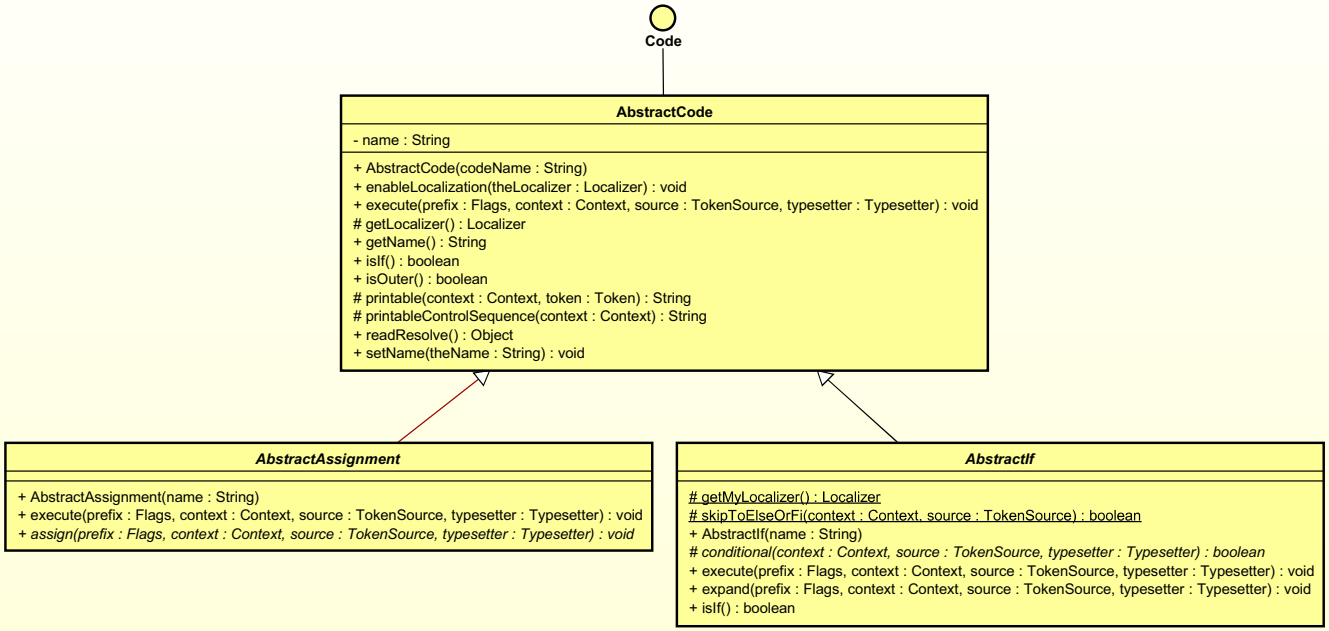
- The minimal requirement for a primitive is to implement the interface Code



A New Primitive

```
package my.extex;

import de.dante.extex.interpreter.contect.Context;
import de.dante.extex.interpreter.primitives.dynamic.java.Loadable;
import de.dante.extex.typesetter.Typesetter;
import de.dante.util.GeneralException;

class MyPrimitive extends AbstractCode {

  public MyPrimitive(final String name) {
    super(name);
  }

  public void execute(final Flags prefix, final Context context,
          final TokenSource source, final Typesetter typesetter)
          throws InterpreterException {
    // implement the primitive here
  }
}
```

The Code Interfaces



Some abstract base classes are present which provide a good starting point

- `AbstractCode` can serve as base class for all primitives
- `AbstractIf` can serve as base class for all conditional primitives (ifs)
- `AbstractAssignment` can serve as base class for all assignments

Java Extension Point

- The primitive `\javadef` acts like `def`
- The primitive `\javadef` takes a class implementing `Code`.
- The class is sought on the Java classpath
- Java provides dynamic loading upon demand
- The primitive `\javadef` is in the configuration `extex-jx`

```
\javadef\abc{my.extex.Primitive}
```

```
\global\javadef\abc{my.extex.Primitive}
```

Loading Extensions

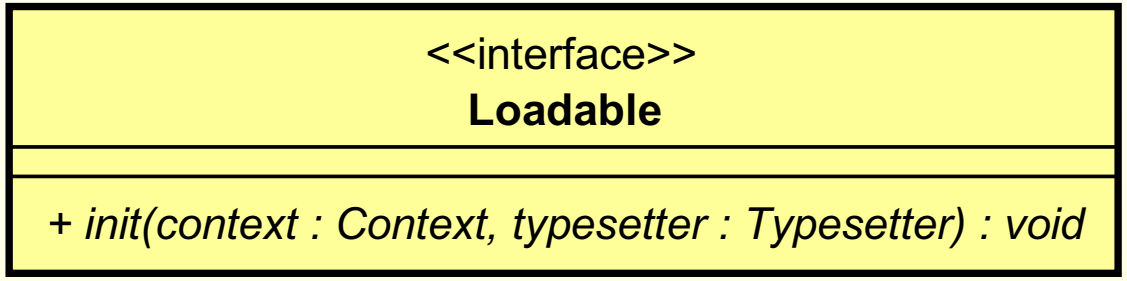
- Instead of defining a single primitive several can be loaded
- The primitive `\javaload` can be used to invoke a certain method at startup time
- The class is sought on the Java classpath
- Java provides dynamic loading upon demand
- The primitive `\javadef` is in the configuration `extex-jx`

```
\javaload{my.extex.Extension}
```

Loading Extensions (2)

```
package my.extex;

import de.dante.extex.interpreter.contect.Context;
import de.dante.extex.interpreter.primitives.dynamic.java.Loadable;
import de.dante.extex.typesetter.Typesetter;
import de.dante.util.GeneralException;

class Extension implements Loadable {

  public Extension() {
    super();
    // initialization code -- if required
  }

  public void init(final Context context,
                   final Typesetter typesetter
                  ) throws GeneralException {
    // implement the initialization code here
  }
}
```

Loading Extensions (3)



- The interface `Loadable` provides full access to the `Context`
  - Primitives can be defined
  - Registers can be changed
  - ...
- The current state can be inspected

Current State

- Interpreter nearly complete
  - All primitives of TEX are present
  - Some primitives of ε-TEX are present
  - Some extensions are provided
- Typesetter sketched
- Math typesetting, table typesetting roughly implemented
- Font engine can read tfm, type1; truetype, opentype in progress
- Document writer for dvi, pdf, svg in progress

Future

- $\varepsilon_{\mathcal{X}}$TEX has to be completed
- $\varepsilon_{\mathcal{X}}$TEX has to be tested
- $\varepsilon_{\mathcal{X}}$TEX has to be documented
- $\varepsilon_{\mathcal{X}}$TEX has to be released and used

http://www.extex.org

Production Notes

- These slides have been made with beamer 3.01
- with the (private) theme Agadir
- the verbatim text is typeset with listings
- The class diagrams have been produced with Jude 1.4.3
- and converted to PDF with ghostscript (as printer)
- Charts have been made with Excel