Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- ε_χTEX
- Integration into ε_χTEX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Namespaces for ε_χTEX

Gerd Neugebauer

EuroTEX 2005
March 2005
Pont-à-Mousson, France

Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- $\varepsilon\chi$TeX
- Integration into $\varepsilon\chi$TeX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Overview

- Requirements
- Concept
- Design
- Implementation

Overview

Requirements

Encapsulation

Backward Compatibility and Initialization

Concept

Definition of Namespaces

Communication between Namespaces

Namespaces and Groups

Namespaces and Expansion

Explicit Expansion without Import

Namespaces and the Basic Definitions

Design

$\varepsilon\chi$TEX

Integration into $\varepsilon\chi$TEX

Tokens

Context

Scanner

Implementation and Extensions

Extensions

Conclusion

Requirements

- Information hiding and privacy are basic principles in modern software engineering
- Module system/package system/namespaces provide privacy
- TEX has no real module system
- LATEX packages use naming conventions
  and a redefined catcode to protect internals
- An namespace extension of TEX is needed
- The existing code should not be affected

Overview

Requirements
Encapsulation
Backward Compatibility and Initialization

Concept
Definition of Namespaces
Communication between Namespaces
Namespaces and Groups
Namespaces and Expansion
Explicit Expansion without Import
Namespaces and the Basic Definitions

Design
ε_χTEX
Integration into ε_χTEX
Tokens
Context
Scanner

Implementation and Extensions
Extensions

Conclusion

Encapsulation

- Encapsulation: Hiding the current meaning
  - Macros
  - Active Characters
  - Registers
    (count, dimen, toks, ...)
  - Catcodes

- Focus here: Macros and Active Characters

Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- $\varepsilon\chi$TEX
- Integration into $\varepsilon\chi$TEX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Backward Compatibility and Initialization

- The extension should be backward compatible.
- The operation should be performed in the default namespace if not specified.
- Namespaces must be properly initialized without too much overhead.

- The attempt should work without syntactic sugar: KISS

Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- ε_χTeX
- Integration into ε_χTeX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Definition of Namespaces

- Special toks register for the current namespace.
- Assignment to this register changes the current namespace.

```
\namespace{tex.latex.dtk}
```

```
\namespace={tex.latex.dtk}
```

Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- $\varepsilon\chi$TEX
- Integration into $\varepsilon\chi$TEX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Default Namespace

- The default namespace has the empty toks register.

```
\namespace={}
```

Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- $\varepsilon\chi$TEX
- Integration into $\varepsilon\chi$TEX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Accessing the Current Namespace

- `\the` and `\showthe` can be used to get access to the current namespace.

```
\namespace{tex.latex.dtk}
\the\namespace
```

$\mapsto$ `tex.latex.dtk`

Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- ε_χTEX
- Integration into ε_χTEX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Communication between Namespaces: Export

- Namespaces provide encapsulation.
- Some entities need to be visible outside.
- A primitive `\export` should be used to specify potentially visible entities.

```
\export{\abc \xyz ~}
```

- `\export` acts like a special toks register.
- The tokens are stored locally for the namespace.

Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- ε_χTEX
- Integration into ε_χTEX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Communication between Namespaces: Import

- A primitive `\import` should be used to specify potentially visible entities in the target namespace.

```
\import{tex.latex.dtk}
```

- The import is performed into the current namespace.
- All entities exported frm the namespace are imported.
- The import works like `\let`
  The modification of the definition in both namespaces are independant

Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- εχTEX
- Integration into εχTEX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Namespaces and Groups

Namespace interact with the group in the expected way.

- Local definitions are discared at the end of the current group.

```
\begingroup
  \namespace{tex}
  \gdef\x{123}
\endgroup
\def\y{123}
```

- \x is undefined afterwards
- \y is defined in the outer namespace

Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- ε_χTEX
- Integration into ε_χTEX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Namespaces and Groups (2)

- `\import` defines the entities “group local”
- `\import` honors the prefix `\global`
- `\global \import` imports into the top group:

```
\begingroup
  \global\import{tex.latex.dtk}
\endgroup
```

- The imports are preserved past the end of the group

Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- $\varepsilon\chi$TEX
- Integration into $\varepsilon\chi$TEX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Namespaces and Groups (3)

```
\begingroup
  \namespace{one}
  \global\export{\x}
  \gdef\x#1{-#1-}
\endgroup
```

- The grouping restricts the effect of `namespace`
- The `\global\export` makes the export survive the end of the group
- the `\gdef` makes the maxro survive the end of the group

Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- $\varepsilon\chi$TEX
- Integration into $\varepsilon\chi$TEX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Namespaces and Expansion

- The same control sequence name can have different bindings

```
\namespace{two}
\begingroup
  \namespace{one}
  \global\export{\x}
  \gdef\x#1{-#1 \y-}
  \gdef\y{in one}
\endgroup
\import{one}
\def\y{two}
\x\y
```

```
↦ -two in one-
```

Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- ε_χTEX
- Integration into ε_χTEX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Explicit Expansion without Import

```
\begingroup
 \namespace{tex}\aftergroup\abc
\endgroup
```

- The namespace is attached to a token when it is created and not, when it is expanded
- The token `\abc` will carry the namespace `tex`
- The grouping restricts the namespace to the two tokens `\expandafter` and `\abc`.
- `\expandafter` delays the expansion until the group is closed

Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- ε_{χ}TEX
- Integration into ε_{χ}TEX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Namespaces and the Basic Definitions

- A new namespace should not start empty – like iniTEX
- `plain.tex`/LATEX/conTEXt provide many useful macros

Solution: Search Strategy for Definitions

- Search in the specified namespace first
- Search in the default namespace if needed

Overview

Requirements
Encapsulation
Backward Compatibility and Initialization

Concept
Definition of Namespaces
Communication between Namespaces
Namespaces and Groups
Namespaces and Expansion
Explicit Expansion without Import
Namespaces and the Basic Definitions

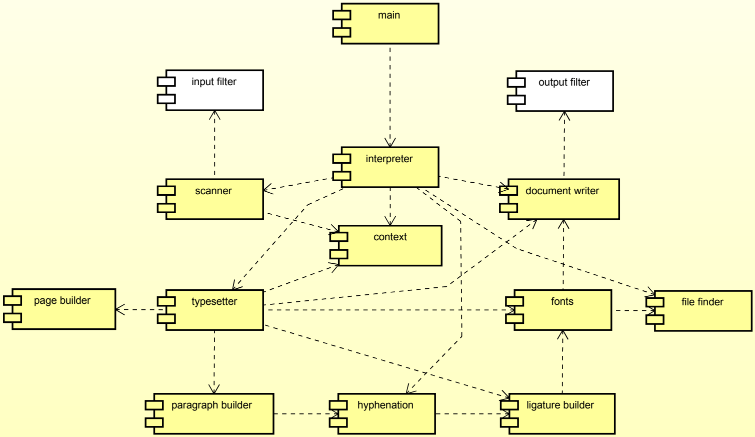
Design
εχTEX
Integration into εχTEX
Tokens
Context
Scanner

Implementation and Extensions
Extensions

Conclusion

εχTEX

- "Extensible" TEX
- Object-oriented reimplementation of TEX
- `http://www.extex.org`



Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- $\varepsilon\chi$TeX
- Integration into $\varepsilon\chi$TeX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Integration into $\varepsilon\chi$TeX

- Extend (some) tokens.
- Extend the Group and the Context.
- Extend The binding mechanisms for control sequences and active characters needs to be extended to take into account the fallback to the default namespace.
- Implement the primitive `\namespace`.
- Implement the primitive `\export`.
- Implement the primitive `\import`.

Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

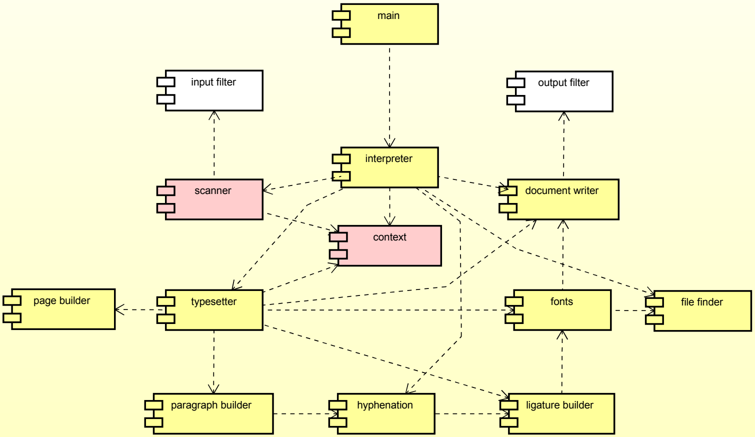
Design
- ε_χTEX
- Integration into ε_χTEX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Changes in ε_χTEX

- The changes are localizable at a few places.



Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

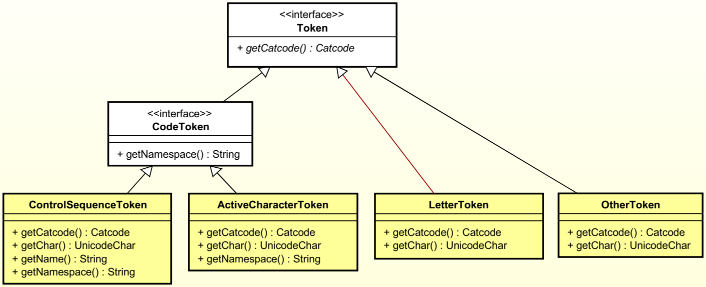
Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- ε_{χ}TEX
- Integration into ε_{χ}TEX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Tokens



- Extend the containers for control sequence tokens and active character tokens.
- Other tokens are not affected.

Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

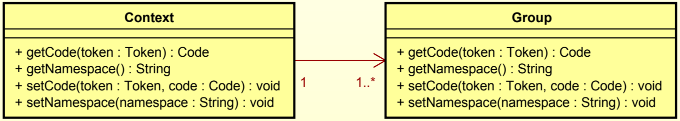
Design
- $\varepsilon\chi$TEX
- Integration into $\varepsilon\chi$TEX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Context

- The Context is the container for all data (like the eq table)
- The Context maintains a stack of Groups in its current implementation


Context
+ getCode(token : Token) : Code
+ getNamespace() : String
+ setCode(token : Token, code : Code) : void
+ setNamespace(namespace : String) : void
1
1..*
Group
+ getCode(token : Token) : Code
+ getNamespace() : String
+ setCode(token : Token, code : Code) : void
+ setNamespace(namespace : String) : void

- Extend the Context and the Group with getters and setters for the current namespace.
- Extend `getCode()` in the Context to contain the search strategy.

Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

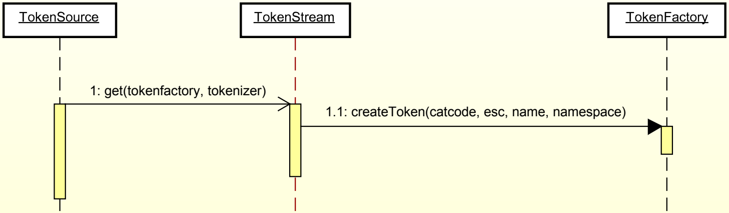
Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- $\varepsilon\chi$TEX
- Integration into $\varepsilon\chi$TEX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Scanner


TokenSource
TokenStream
TokenFactory
1: get(tokenfactory, tokenizer)
1.1: createToken(catcode, esc, name, namespace)

- The invocation of the token factory is extended to contain the namespace
- The token factory has to be extended accordingly

Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- $\varepsilon\chi$TEX
- Integration into $\varepsilon\chi$TEX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Implementation

```
/**...*/
public class ControlSequenceToken extends AbstractToken
    implements CodeToken {

    /**...*/
    private String name;

    /**...*/
    private String namespace;

    /**...*/
    protected ControlSequenceToken(final UnicodeChar esc,
            final String name,
            final String namespace) {

        super(esc);
        this.namespace = namespace;
        this.name = name;
    }
```

It works!

Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- $\varepsilon\chi$TEX
- Integration into $\varepsilon\chi$TEX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Extensions

- Extension of namespaces to registers.
  Partially implemented in $\varepsilon\chi$TEX
  (compile-time configuration)
  - Interferes with `plain.tex`
  - Experiments not convincing yet
- Selective import of dedicated tokens
- Renaming during the import
- Search strategy with intermediate levels of packages (decomposition of namespace identifier)
- Syntactic sugar

Overview

Requirements
- Encapsulation
- Backward Compatibility and Initialization

Concept
- Definition of Namespaces
- Communication between Namespaces
- Namespaces and Groups
- Namespaces and Expansion
- Explicit Expansion without Import
- Namespaces and the Basic Definitions

Design
- $\varepsilon\chi$TEX
- Integration into $\varepsilon\chi$TEX
- Tokens
- Context
- Scanner

Implementation and Extensions
- Extensions

Conclusion

Conclusion

- Namespaces can be provided with a few modifications of $\varepsilon\chi$TEX.
- Namespaces for control sequences and active characters are a good first step.
- The extension is performed minimalistically.
- Namespaces are an offer for macro writers.
- Make the best use of it.