

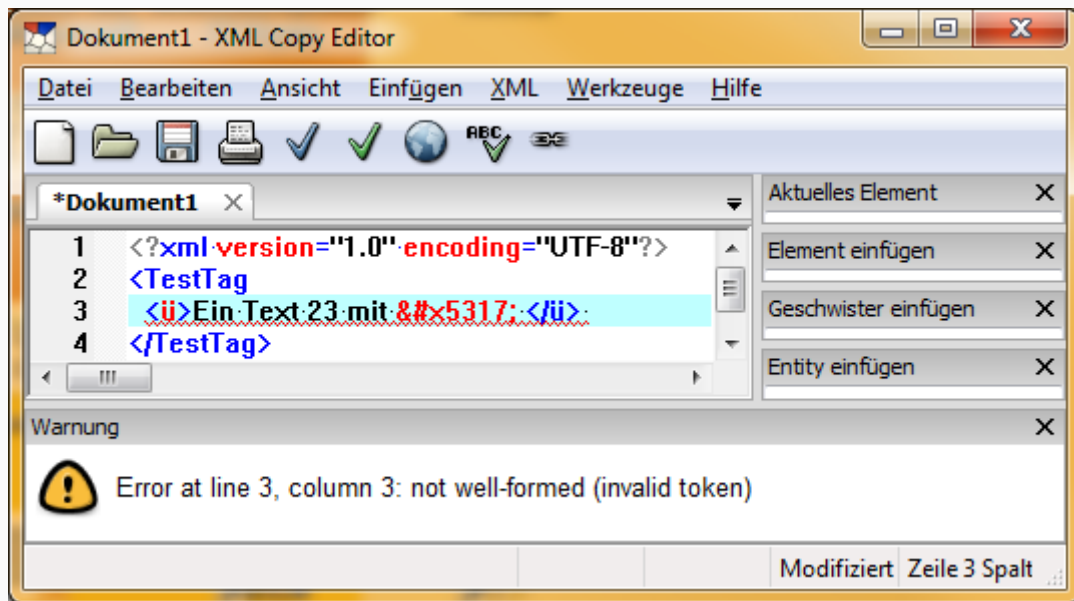
# Error-Recovery

---

Carsten Gips (HSBI)

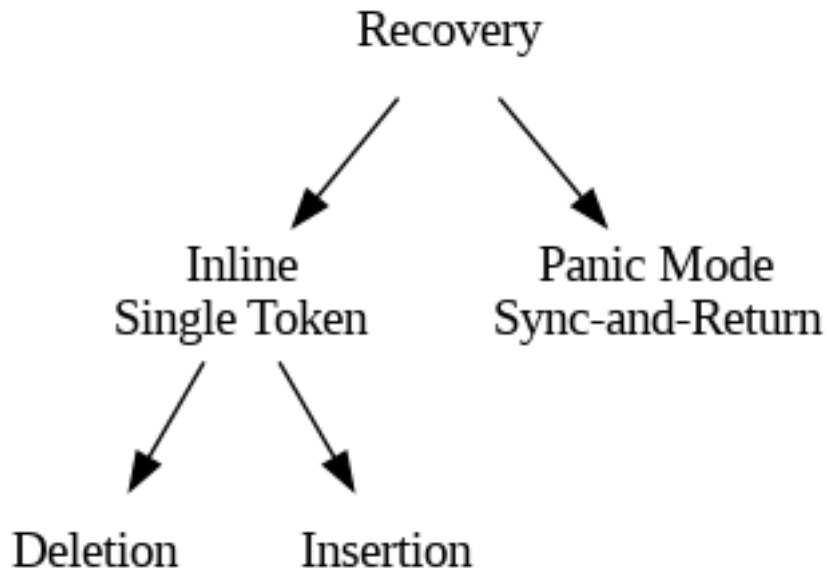
Unless otherwise noted, this work is licensed under CC BY-SA 4.0.

# Fehler beim Parsen



# Typische Fehler beim Parsing

```
grammar VarDef;  
  
alt    : stmt | stmt2 ;  
stmt   : 'int' ID ';' ;  
stmt2  : 'int' ID '=' ID ';' ;
```



## Skizze: Generierte Parser-Regeln (ANTLR)

```
stmt : 'int' ID ';' ;
```

```
def stmt():
    try: match("int"); match(ID); match(";")
    catch (RecognitionException re):
        _errHandler.reportError(self)           # let's report it
        _errHandler.recover(self)              # Panic-Mode

def match(x):
    if lookahead == x: consume()
    else: _errHandler.recoverInline(self)       # Inline-Mode
}
```

## Inline-Recovery bei Token-Mismatch (Skizze)

```
def recoverInline(parser):  
    # SINGLE TOKEN DELETION  
    if singleTokenDeletion(parser):  
        return getMatchedSymbol(parser)  
  
    # SINGLE TOKEN INSERTION  
    if singleTokenInsertion(parser):  
        return getMissingSymbol(parser)  
  
    # that didn't work, throw a new exception  
    throw new InputMismatchException(parser)  
}
```

## Panic Mode: Sync-and-Return (Skizze)

```
def rule():  
  try: ... rule-body ...  
  catch (RecognitionException re):  
    _errHandler.reportError(self)      # let's report it  
    _errHandler.recover(self)          # Panic-Mode  
}
```

=> Entferne solange Token, bis aktuelles Token im “Resynchronization Set”

# ANTLR: Einsatz des “*Resynchronization Set*”

```
stmt : 'if' expr ':' stmt           // Following Set für "expr": {' ':'}
      | 'while' '(' expr ')' stmt ; // Following Set für "expr": {'(' ')'}
expr : term '+' INT ;              // Following Set für "term": {'+' }
```

- Eingabe: `if :`
- Aufruf-Stack nach Bearbeitung von `if`: `[stmt, expr, term]`
- **Resynchronization Set:** `{'+', ':'}`

Hinweis: *FOLLOW*  $\neq$  *Following*



## Panic Mode in Bison (Error Recovery)

```
stmt : 'int' ID ';'      { printf("%s\n", $2); }  
    | error '\n'        { yyerror(); yyerrok; }  
    ;
```

# Fehlerproduktionen

```
stmt : 'int' ID ';'
      : 'int' ID          {notifyErrorListeners("Missing ';'");}
      : 'int' ID ';' ';' {notifyErrorListeners("Too many ';'");}
      ;
```

```
stmt : 'int' ID ';' { $$ = $2; }
      : 'int' ID    { yyerror("unterminated id");
                     $$ = $2; }

;

%%

void yyerror(char *s, ...) {
    va_list ap; va_start(ap, s);
    fprintf(stderr, "%d: error: ", yylineno);
    vfprintf(stderr, s, ap); fprintf(stderr, "\n");
}
```

- Fehler bei `match()`: *single token deletion* oder *single token insertion*
- Panic Mode: *sync-and-return* bis Token in *Resynchronization Set* (ANTLR) oder `error`-Token shiftbar (Bison)
  - ANTLR: Sonderbehandlung bei Start von Sub-Regeln und in Schleifen
  - ANTLR: Fail-Save zur Vermeidung von Endlosschleifen
- Fehler-Alternativen in Grammatik einbauen



Unless otherwise noted, this work is licensed under CC BY-SA 4.0.