How to deal with context

M. Anton Ertl
TU Wien

How to write **hex.**

```
: foo
  ... hex. ...
  ... .    ... ;

decimal foo
hex foo
```

How to write `hex.`

```
: hex.
  hex u. ;

: foo
  ... hex. ...
  ... .    ... ;

decimal foo
hex foo
```

How to write **hex.**

```
: hex.
  hex u. decimal ;

: foo
  ... hex. ...
  ... .    ... ;

decimal foo
hex foo
```

How to write `hex.`

```
: hex.
  base @ >r hex u. r> base ! ;

: foo
  ... hex. ...
  ... .    ... ;

decimal foo
hex foo
```

How to write hex.

```
: hex.-helper
  hex u. ;

: hex.
  base @ >r ['] hex.-helper catch r> base ! throw ;

: foo
  ... hex. ...
  ... .    ... ;

decimal foo
hex foo
```

How to write `hex.`

```
: hex.
  base @ { oldbase }
  TRY
    hex foo \ now the hex is placed correctly
    0       \ value for throw
  RESTORE
    oldbase base !
  ENDTRY
  throw ;

: foo
  ... hex. ...
  ... .    ... ;

decimal foo
hex foo
```

Get and Set?

`GET-CURRENT SET-CURRENT`

Has certain benefits, but not for context problem

Context wrapper

```
: hex.
  ['] u. $10 base-execute ;

: foo
  ... hex. ...
  ... .    ... ;

decimal foo
hex foo
```

Other contexts

```
outfile-execute
infile-execute
```

Other languages: Postscript

```
<< /base 16 >> begin ... end
```

Other languages: Lisp

Dynamically scoped variables

Other languages: Unix environment variables

```
env LANG=DE_at.utf-8 gforth
```