```
\starttitle[title={Using \CONTEXT}]

The \CONTEXT\ macro package is more than just a \TEX\ processor,
various input is possible, some we show here. An example of a
method not shown here is fetching data from a database. The
various input methods can be combined, so depending on what you
need you can mix \TEX\ (for typesetting text), \METAPOST\ (for
producing graphics) or \LUA\ (as language for manipulating
data.

All these methods are quite efficient and always have access to
the full power of the typesetting engine.

When you use \CONTEXT\ with \LUAMETATEX, you get a reasonable
small self contained component that can be used in workflows
that need quality rendering. The ecosystem is rather future
proof too.

The \CONTEXT\ macro package has been around for decades and
evolved from \MKII, to \MKIV\ and now \LMTX. The development
team has always been involved in the development of engines
like \PDFTEX, \LUATEX\ and \LUAMETATEX. There is an active
mailing list and there are \CONTEXT\ meetings.

\stoptitle
```

```
\starttext

    \starttitle[title={Some \TEX}]

        Just an example.

        \starttabulate[|c|c|]
            \NC first 1 \NC last 1 \NC \NR
            \NC first 2 \NC last 2 \NC \NR
        \stoptabulate

    \stoptitle

\stoptext
```

```
\startbuffer[demo]
<?xml version="1.0"?>
<document>
    <title>Some XML</title>
    <p>Just an example.</p>
    <table>
        <r> <c>first 1</c> <c>last 1</c> </r>
        <r> <c>first 2</c> <c>last 2</c> </r>
    </table>
</document>
\stopbuffer

\startxmlsetups xml:basics
    \xmlsetsetup{#1}{title|p|table}{xml:*}
\stopxmlsetups
\startxmlsetups xml:title
    \title{\xmltext{#1}{.}}
\stopxmlsetups
\startxmlsetups xml:p
    \xmlflush{#1}\par
\stopxmlsetups
\startxmlsetups xml:table
    \starttabulate[|c|c|]
        \xmlfilter{#1}{/r/command(xml:r)}
    \stoptabulate
\stopxmlsetups
\startxmlsetups xml:r
    \NC \xmlfilter{#1}{/c/command(xml:c)} \NR
\stopxmlsetups
\startxmlsetups xml:c
    \xmlflush{#1} \NC
\stopxmlsetups

\xmlregistersetup{xml:basics}

\starttext
    \xmlprocessbuffer{demo}{demo}{}
\stoptext
```

```
\startluacode
    local tmp = {
        { a = "first 1", b = "last 1" },
        { b = "last 2", a = "first 2" },
    }

    -- local tmp = table.load("somefile.lua")

    context.starttext()

        context.starttitle { title = "Some Lua" }

            context("Just an example.") context.par()

            context.starttabulate { "|c|c|" }
                for i=1,#tmp do
                    local t = tmp[i]
                    context.NC()
                        context(t.a) context.NC()
                        context(t.b) context.NC()
                    context.NR()
                end
            context.stoptabulate()

        context.stoptitle()

    context.stoptext()
\stopluacode
```

```
\startMPpage
    draw textext("\bf Some \MetaPost")
        xsized 4cm
        rotated(45)
        withcolor "white" ;

    draw textext("\bs\strut in \ConTeXt")
        xsized 5cm
        shifted (0,-40mm)
        withcolor "white" ;

    draw fullcircle
        scaled 6cm
        dashed evenly
        withcolor "gray" ;
\stopMPpage
```

```
\startluacode
    local tmp = [[
        first,second
        first 1,last 1
        first 2,last 2
    ]]

    -- local tmp = io.loaddata("somefile.csv")

    local mycsvsplitter = utilities.parsers.rfc4180splitter()
    local list, names = mycsvsplitter(tmp,true)

    context.starttext()

        context.starttitle { title = "Some CSV" }

            context("Just an example.") context.par()

            context.starttabulate { "|c|c|" }
                for i=1,#list do
                    local l = list[i]
                    context.NC()
                        context(l[1]) context.NC()
                        context(l[2]) context.NC()
                    context.NR()
                end
            context.stoptabulate()

        context.stoptitle()

    context.stoptext()
\stopluacode
```

```
\startluacode
    require("util-jsn")

    -- local str = io.loaddata("somefile.json")

    local str = [[ {
        "title": "Some JSON",
        "text" : "Just an example.",
        "data" : [
            { "a" : "first 1", "b" : "last 1" },
            { "b" : "last 2", "a" : "first 2" }
        ]
    } ]]

    local tmp = utilities.json.tolua(str)

    context.starttext()

        context.starttitle { title = tmp.title }

            context(tmp.text) context.par()

            context.starttabulate { "|c|c|" }
                for i=1,#tmp.data do
                    local d = tmp.data[i]
                    context.NC()
                        context(d.a) context.NC()
                        context(d.b) context.NC()
                    context.NR()
                end
            context.stoptabulate()

        context.stoptitle()

    context.stoptext()
\stopluacode
```

```
% normally there is already a file:

\startbuffer[demo]
\starttext
  \starttitle[title={Some template}]

  Just an example. \blank

  \startlinecorrection
    \bTABLE
      <?lua for i=1,20 do ?>
        \bTR
          <?lua for j=1,5 do ?>
            \bTD
              cell (<?lua inject(i) ?>,<?lua inject(j) ?>)
              is <?lua inject(variables.text or "unset") ?>
            \eTD
          <?lua end ?>
        \eTR
      <?lua end ?>
    \eTABLE
  \stoplinecorrection

  \stoptitle
\stoptext

\stopbuffer

\savebuffer[file=demo.mkxi,prefix=no,list=demo]

% the action:

\startluacode
    document.variables.text = "set"
\stopluacode

\input{demo.mkxi}
```

Using ConTeXt

The ConTeXt macro package is more than just a TeX processor, various input is possible, some we show here. An example of a method not shown here is fetching data from a database. The various input methods can be combined, so depending on what you need you can mix TeX (for typesetting text), MetaPost (for producing graphics) or Lua (as language for manipulating data.

All these methods are quite efficient and always have access to the full power of the typesetting engine.

When you use ConTeXt with LuaMetaTeX, you get a reasonable small self contained component that can be used in workflows that need quality rendering. The ecosystem is rather future proof too.

The ConTeXt macro package has been around for decades and evolved from MkII, to MkIV and now lmtx. The development team has always been involved in the development of engines like pdfTeX, LuaTeX and LuaMetaTeX. There is an active mailing list and there are ConTeXt meetings.

Some TeX

Just an example.

first 1	last 1
first 2	last 2

Some XML

Just an example.

first 1	last 1
first 2	last 2

Some Lua

Just an example.

first 1	last 1
first 2	last 2

Some CSV

Just an example.

first 1	last 1
first 2	last 2

Some JSON

Just an example.

first 1	last 1
first 2	last 2

Some template

Just an example.

cell (1,1) is set	cell (1,2) is set	cell (1,3) is set	cell (1,4) is set	cell (1,5) is set
cell (2,1) is set	cell (2,2) is set	cell (2,3) is set	cell (2,4) is set	cell (2,5) is set
cell (3,1) is set	cell (3,2) is set	cell (3,3) is set	cell (3,4) is set	cell (3,5) is set
cell (4,1) is set	cell (4,2) is set	cell (4,3) is set	cell (4,4) is set	cell (4,5) is set
cell (5,1) is set	cell (5,2) is set	cell (5,3) is set	cell (5,4) is set	cell (5,5) is set
cell (6,1) is set	cell (6,2) is set	cell (6,3) is set	cell (6,4) is set	cell (6,5) is set
cell (7,1) is set	cell (7,2) is set	cell (7,3) is set	cell (7,4) is set	cell (7,5) is set
cell (8,1) is set	cell (8,2) is set	cell (8,3) is set	cell (8,4) is set	cell (8,5) is set
cell (9,1) is set	cell (9,2) is set	cell (9,3) is set	cell (9,4) is set	cell (9,5) is set
cell (10,1) is set	cell (10,2) is set	cell (10,3) is set	cell (10,4) is set	cell (10,5) is set
cell (11,1) is set	cell (11,2) is set	cell (11,3) is set	cell (11,4) is set	cell (11,5) is set
cell (12,1) is set	cell (12,2) is set	cell (12,3) is set	cell (12,4) is set	cell (12,5) is set
cell (13,1) is set	cell (13,2) is set	cell (13,3) is set	cell (13,4) is set	cell (13,5) is set
cell (14,1) is set	cell (14,2) is set	cell (14,3) is set	cell (14,4) is set	cell (14,5) is set
cell (15,1) is set	cell (15,2) is set	cell (15,3) is set	cell (15,4) is set	cell (15,5) is set
cell (16,1) is set	cell (16,2) is set	cell (16,3) is set	cell (16,4) is set	cell (16,5) is set
cell (17,1) is set	cell (17,2) is set	cell (17,3) is set	cell (17,4) is set	cell (17,5) is set
cell (18,1) is set	cell (18,2) is set	cell (18,3) is set	cell (18,4) is set	cell (18,5) is set
cell (19,1) is set	cell (19,2) is set	cell (19,3) is set	cell (19,4) is set	cell (19,5) is set
cell (20,1) is set	cell (20,2) is set	cell (20,3) is set	cell (20,4) is set	cell (20,5) is set