`pathsuris.sty`: Paths and URIs for sTeX[*]

Jinbo Zhang, Michael Kohlhase, Dennis Müller
FAU Erlangen-Nürnberg

November 3, 2020

Abstract

This package provides macros to deal with paths and base URIs for sTeX. In particular, it offers a path canonicalizer, which is used in package `modules`, in order to support modules specified with relative path.

Contents

[*]Version v2.1 (last revised 2020/09/30)

1 User Interface

1.1 Base URIs

\baseURI \baseURI[1]

EdN:1

1.2 Using Absolute Paths

\defpath

Finally, the separation of documents into multiple modules often profits from a symbolic management of file paths. To simplify this, the `modules` package supplies the `\defpath` macro: `\defpath[`⟨*baseURI*⟩`]{`⟨*cname*⟩`}{`⟨*path*⟩`}` defines a command, so that `\`⟨*csname*⟩`{`⟨*name*⟩`}` expands to ⟨*path*⟩`/`⟨*name*⟩. So we could have used

```
\defpath{OPaths}{../other}
\importmodule[load=\OPahts{bar}]{bar}
```

instead of the second line in Example **??**. The variant `\OPaths` has the big advantage that we can get around the fact that TEX/LATEX does not set the current directory in `\input`, so that we can use systematically deployed `\defpath`-defined path macros to make modules relocatable by defining the path macros locally. The optional parameter ⟨*baseURI*⟩ is for the LaTeXML transformation, which (if ⟨*baseURI*⟩ is specified) resolves ⟨*path*⟩ to an absolute URI according to [BFM05, section 5.2].

1.3 Path Canonicalization

\cpath

By calling `\@cpath{`⟨*path*⟩`}`, the canonicalized path will be stored in `\@CanPath`. To print a canonicalized path, simply use `\cpath{`⟨*path*⟩`}`. Here is a set of examples with their canonizalized paths for testing.

path	canonicalized path	expected
aaa	aaa	aaa
../../aaa	../../aaa	../../aaa
aaa/bbb	aaa/bbb	aaa/bbb
aaa/..		
../../aaa/bbb	../../aaa/bbb	../../aaa/bbb
../aaa/../bbb	../bbb	../bbb
../aaa/bbb	../aaa/bbb	../aaa/bbb
aaa/bbb/../ddd	aaa/ddd	aaa/ddd
aaa/bbb/./ddd	aaa/bbb/ddd	aaa/bbb/ddd
./		
aaa/bbb/../..		

[1] EdNote: document it

1.4 URIs

By calling `\seturi[\meta{macroname}]{`⟨*path*⟩`}`, the URI will be split into its components `\macronamescheme`, `\macronameauthority`, `\macronamepath`, `\macronamequery` and `\macronamefragment`, and the resolved URI itself is stored in `\macronameuri`, as in the following example. If the optional `macroname` is not provided, the default name is `pathsuris@curruri@`.

In order to differentiate between empty and missing components, a missing component will be equal to `\makeuri@empty`, whose *expansion* is `\relax`.

```
\seturi[myuri]{http://this.isatest/foo/bar/?query#fragment}
```

yields:

macro	value
`\myuriuri`	http://this.isatest/foo/bar?query#fragment
`\myurischeme`	http
`\myuriauthority`	this.isatest
`\myuripath`	foo/bar
`\myuriquery`	query
`\myurifragment`	fragment

`\makeuri{\meta{scheme}}{\meta{authority}}{\meta{path}}{\meta{query}}{\meta{fragment}}` constructs a URI from its individual components. The (expanded and resolved) individual components will be stored in `\makeuri@scheme`, `\makeuri@authority`, etc.; the resolved URI will be stored in `\makeuri@uri`.

`\asuri{\meta{macroname}}{\meta{uri}}`, similarly to setpath, defines a new macro `\macroname[\meta{newmacroname}]{\meta{command}}` that allows manipulating `uri` in various ways. `\asuri` calls `\seturi[\meta{macroname}]{\meta{uri}}`, so the individual components and full uri (as string) are subsequently stored in `\macronamescheme`, `\macronameauthority`, etc.

If an optional new macro name is given in `\macroname`, then the result of the modification is stored in that new macro, as if defined via `\asuri`; otherwise, the macro is modified "in place".

- `\macroname{drop query}` drops the query component.
- `\macroname{drop fragment}` drops the fragment component.
- `\macroname{/other/path}` drops query and fragment, appends `other/path` to the path and resolves the URI.
- `\macroname{?newquery}` drops the fragment and either declares `newquery` as a new query component, or appends `?newquery` to the existing query component, if it is not `\makeuri@empty`. Note, that this behaviour diverges from the official URI specification, but it conforms to MMT URI's, which use ? as separator between DPaths, modules names and declaration names.
- `\macroname{#newfragment}` analogously to `{?newquery}`.

2 The Implementation

```
⟨*package⟩
\RequirePackage{stex-base}
\RequirePackage{xstring}
\RequirePackage{etoolbox}
```

2.1 Base URIs

\baseURI On the LaTeX side we do nothing (for the moment).

```
\newcommand\baseURI[2][]{}
```

2.2 Using Absolute Paths

\defpath `\defpath[optional argument]{macro name}{base path}` defines a new macro which can take another path to formal one integrated path. For example, `\MathHub` in every `localpaths.tex` is defined as:

```
\defpath{MathHub}{/path/to/localmh/MathHub}
```

then we can use `\MathHub` to form other paths, for example,

```
\MathHub{source/smglom/sets}
```

will generate `/path/to/localmh/MathHub/source/smglom/sets`.

```
\newrobustcmd\defpath[3][]{%
  \expandafter\newcommand\csname #2\endcsname[1]{#3/##1}%
}%
```

2.3 Path Canonicalization

We define two macros for changing the category codes of common characters in URIs, in particular #.

```
\def\pathsuris@setcatcodes{%
    \edef\pathsuris@oldcatcode@hash{\the\catcode`\#}%
    \catcode`\#=12\relax%
    \edef\pathsuris@oldcatcode@slash{\the\catcode`\/}%
    \catcode`\/=12\relax%
    \edef\pathsuris@oldcatcode@colon{\the\catcode`\:}%
    \catcode`\:=12\relax%
    \edef\pathsuris@oldcatcode@qm{\the\catcode`\?}%
    \catcode`\?=12\relax%
}
\def\pathsuris@resetcatcodes{%
    \catcode`\#\pathsuris@oldcatcode@hash\relax%
    \catcode`\/\pathsuris@oldcatcode@slash\relax%
    \catcode`\:\pathsuris@oldcatcode@colon\relax%
    \catcode`\?\pathsuris@oldcatcode@qm\relax%
}
```

We define some macros for later comparison.

```
\def\@ToTop{..}
\def\@Slash{/}
\def\@Colon{:}
\def\@QuestionMark{?}
\def\@Dot{.}

\pathsuris@setcatcodes
\def\@Fragment{#}
\pathsuris@resetcatcodes
```

Implement `\@cpath`.

`\@cpath`

```
\def\@cpath#1{%
    \edef\pathsuris@cpath@temp{#1}%
    \def\@CanPath{}%
    \IfBeginWith\pathsuris@cpath@temp\@Slash{%
      \@cpath@loop%
      \edef\@CanPath{\@Slash\@CanPath}%
    }{%
        \IfBeginWith\pathsuris@cpath@temp{\@Dot\@Slash}{%
            \StrGobbleLeft\pathsuris@cpath@temp2[\pathsuris@cpath@temp]%
            \@cpath@loop%
        }{%
            \ifx\pathsuris@cpath@temp\@Dot\else%
            \@cpath@loop\fi%
        }%
    }%
    \IfEndWith\@CanPath\@Slash{%
      \ifx\@CanPath\@Slash\else%
        \StrGobbleRight\@CanPath1[\@CanPath]%
      \fi%
    }{}%
}

\def\@cpath@loop{%
    \IfSubStr\pathsuris@cpath@temp\@Slash{%
        \StrCut\pathsuris@cpath@temp\@Slash\pathsuris@cpath@temp@a\pathsuris@cpath@temp%
        \ifx\pathsuris@cpath@temp@a\@ToTop%
            \ifx\@CanPath\@empty%
                \edef\@CanPath{\@ToTop}%
            \else%
                \edef\@CanPath{\@CanPath\@Slash\@ToTop}%
            \fi%
            \@cpath@loop%
        \else%
        \ifx\pathsuris@cpath@temp@a\@Dot%
            \@cpath@loop%
        \else%
        \IfBeginWith\pathsuris@cpath@temp\@ToTop{%
            \StrBehind{\pathsuris@cpath@temp}{\@ToTop}[\pathsuris@cpath@temp]%
            \IfBeginWith\pathsuris@cpath@temp\@Slash{%
                \edef\pathsuris@cpath@temp{\@CanPath\pathsuris@cpath@temp}%
            }{%
                \ifx\@CanPath\@empty\else%
                    \edef\pathsuris@cpath@temp{\@CanPath\@Slash\pathsuris@cpath@temp}
                \fi%
            }%
            \def\@CanPath{}%
            \@cpath@loop%
        }{%
            \ifx\@CanPath\@empty%
                \edef\@CanPath{\pathsuris@cpath@temp@a}%
            \else%
                \edef\@CanPath{\@CanPath\@Slash\pathsuris@cpath@temp@a}%
            \fi%
            \@cpath@loop
        }%
        \fi\fi%
    }{
        \ifx\@CanPath\@empty%
            \edef\@CanPath{\pathsuris@cpath@temp}%
        \else%
            \edef\@CanPath{\@CanPath\@Slash\pathsuris@cpath@temp}%
        \fi%
    }%
}
```

Implement `\cpath` to print the canonicalized path.

`\cpath`

```
\newcommand\cpath[1]{%
    \@cpath{#1}%
    \@CanPath%
}
```

2.4 URIs

Various macros for dealing with URIs. To deal with empty URI components (scheme, authority, etc.), we use `{\relax}` to signify an non-existent component as oppsed to an empty one.

```
\def\makeuri@setempty#1{\def#1{\relax}}
\def\makeuri@empty{\relax}
\def\makeuri@test#1{%
    \ifx#1\makeuri@empty\else#1\fi%
}
```

`\makeuri` `\makeuri` constructs a URI from scheme, authority, path, query and fragment separately.

```
\def\makeuri@uri{}
\def\makeuri#1#2#3#4#5{
    \edef\makeuri@scheme{#1}
    \edef\makeuri@authority{#2}
    \edef\makeuri@path{#3}
    \ifx\makeuri@path\makeuri@empty\else
        \@cpath{#3}
        \edef\makeuri@path{\@CanPath}
    \fi
    \edef\makeuri@query{#4}
    \edef\makeuri@fragment{#5}
    \ifx\makeuri@scheme\makeuri@empty\else
        \edef\makeuri@scheme{\makeuri@scheme\@Colon}
    \fi
    \ifx\makeuri@authority\makeuri@empty\else
        \edef\makeuri@authority{\@Slash\@Slash\makeuri@authority}
        \ifx\makeuri@path\makeuri@empty\else
            \IfBeginWith\makeuri@path\@Slash{}{
                \edef\makeuri@path{\@Slash\makeuri@path}
            }
        \fi
    \fi
    \ifx\makeuri@query\makeuri@empty\else
        \edef\makeuri@query{\@QuestionMark\makeuri@query}
    \fi
    \ifx\makeuri@fragment\makeuri@empty\else
        \edef\makeuri@fragment{\@Fragment\makeuri@fragment}
    \fi
    \edef\makeuri@uri{%
        \makeuri@test\makeuri@scheme%
        \makeuri@test\makeuri@authority%
        \makeuri@test\makeuri@path%
        \makeuri@test\makeuri@query%
        \makeuri@test\makeuri@fragment%
    }
}
```

`\seturi@`

```
\newif\if@pathsuris@done@
\def\seturi@[#1]#2{%
    \@pathsuris@done@false%
    \def\pathsuris@prefix@temp{#1}
    \edef\pathsuris@curruri{#2}%
    \edef\pathsuris@curruri{\expandafter\detokenize\expandafter{\pathsuris@curruri}}
    \let\pathsuris@temp\pathsuris@curruri%
    \makeuri@setempty\pathsuris@curruri@scheme%
    \makeuri@setempty\pathsuris@curruri@authority%
    \makeuri@setempty\pathsuris@curruri@path%
    \makeuri@setempty\pathsuris@curruri@query%
    \makeuri@setempty\pathsuris@curruri@fragment%
```

```
% scheme
\IfSubStr{\pathsuris@temp}{\@Colon}{%
    % TODO check for valid scheme
    \StrBefore{\pathsuris@temp}{\@Colon}[\pathsuris@curruri@scheme]%
    \StrBehind{\pathsuris@temp}{\@Colon}[\pathsuris@temp]%
}{}%
% authority
\IfBeginWith{\pathsuris@temp}{\@Slash\@Slash}{%
    \StrBehind{\pathsuris@temp}{\@Slash\@Slash}[\pathsuris@temp]%
    \IfSubStr{\pathsuris@temp}{\@Slash}{%
        \StrBefore{\pathsuris@temp}{\@Slash}[\pathsuris@curruri@authority]%
        \StrBehind{\pathsuris@temp}{\@Slash}[\pathsuris@temp]%
        % TODO userinfo,host,port
    }{%
        \IfSubStr\pathsuris@temp\@QuestionMark{
            \StrBefore{\pathsuris@temp}{\@QuestionMark}[\pathsuris@curruri@authority]%
            \StrBehind{\pathsuris@temp}{\@QuestionMark}[\pathsuris@temp]%
            \edef\pathsuris@temp{\@QuestionMark\pathsuris@temp}%
        }{
            \IfSubStr\pathsuris@temp\@Fragment{
                \StrBefore{\pathsuris@temp}{\@Fragment}[\pathsuris@curruri@authority]%
                \StrBehind{\pathsuris@temp}{\@Fragment}[\pathsuris@temp]%
                \edef\pathsuris@temp{\@Fragment\pathsuris@temp}%
            }{
                \edef\pathsuris@curruri@authority{\pathsuris@temp}%
                \@pathsuris@done@true%
            }
        }
    }%
}{}%
% path, query, fragment
\if@pathsuris@done@\else%
    \IfSubStr{\pathsuris@temp}{\@QuestionMark}{%
        % path
        \StrBefore{\pathsuris@temp}{\@QuestionMark}[\pathsuris@curruri@path]%
        \@cpath\pathsuris@curruri@path%
        \edef\pathsuris@curruri@path{\@CanPath}%
        \StrBehind{\pathsuris@temp}{\@QuestionMark}[\pathsuris@temp]%
        % query,fragment
        \IfSubStr{\pathsuris@temp}{\@Fragment}{%
            \StrBefore{\pathsuris@temp}{\@Fragment}[\pathsuris@curruri@query]%
            \StrBehind{\pathsuris@temp}{\@Fragment}[\pathsuris@curruri@fragment]%
        }{%
            \edef\pathsuris@curruri@query{\pathsuris@temp}%
        }%
    }{%
        % path,fragment
        \IfSubStr{\pathsuris@temp}{\@Fragment}{%
            \StrBefore{\pathsuris@temp}{\@Fragment}[\pathsuris@curruri@path]%
            \@cpath\pathsuris@curruri@path%
```

```
                \edef\pathsuris@curruri@path{\@CanPath}%
                \StrBehind{\pathsuris@temp}{\@Fragment}[\pathsuris@curruri@fragment]%
            }{%
                \edef\pathsuris@curruri@path{\pathsuris@temp}%
            }%
        }%
    \fi%
    \makeuri\pathsuris@curruri@scheme\pathsuris@curruri@authority\pathsuris@curruri@path\pathsu
    \let\pathsuris@curruri@uri\makeuri@uri
    %drop trailing slash of path
    %\IfEndWith{\pathsuris@curruri@path}{\@Slash}{%
    %    \StrGobbleRight{\pathsuris@curruri@path}{1}[\pathsuris@curruri@path]
    %}{}%
    %
    %\edef\pathsuris@curruri@path{\cpath{\pathsuris@curruri@path}}%
    \ifx\pathsuris@prefix@temp\@empty\else%
        \expandafter\let\csname \pathsuris@prefix@temp scheme\endcsname\pathsuris@curruri@schem
        \expandafter\let\csname \pathsuris@prefix@temp authority\endcsname\pathsuris@curruri@au
        \expandafter\let\csname \pathsuris@prefix@temp path\endcsname\pathsuris@curruri@path%
        \expandafter\let\csname \pathsuris@prefix@temp query\endcsname\pathsuris@curruri@query%
        \expandafter\let\csname \pathsuris@prefix@temp fragment\endcsname\pathsuris@curruri@fra
        \expandafter\let\csname \pathsuris@prefix@temp uri\endcsname\pathsuris@curruri@uri%
    \fi%
}
```

`\seturi`

```
\newrobustcmd\seturi[1][]{%
    \pathsuris@setcatcodes%
    \expandafter\pathsuris@resetcatcodes\seturi@[#1]%
}
```

`\asuri` `\asuri{macroname}{uri}` generates `\macroname[optional new macro name]{action}`, that allows for modifying `uri` in various ways.

```

\def\asuri#1{%
    \pathsuris@setcatcodes%
    \expandafter\pathsuris@resetcatcodes\@asuri[#1]%
}

\def\@asuri[#1]#2{
    \@cpath{#2}
    \expandafter\def\csname #1\endcsname{}
    \expandafter\edef\csname #1uri\endcsname{\@CanPath}
    \seturi[#1]{\@CanPath}
    \expandafter\renewcommand\csname #1\endcsname[1][]{%
        \pathsuris@setcatcodes%
        \@@asuri@[##1]{#1}%
    }%
}
```

```

\protected\def\@@asuri@[#1]#2#3{
    \pathsuris@resetcatcodes
    \@@asuri[#1]{#2}{#3}
}

\newif\if@asuri@changed@
\protected\def\@@asuri[#1]#2#3{
    \@asuri@changed@false
    \edef\@@asuri@command{#3}
    \trimstring\@@asuri@command
    \IfBeginWith\@@asuri@command{drop}{
        \StrBehind{\@@asuri@command}{drop}[\@@asuri@command]
        \trimstring\@@asuri@command
        \IfStrEq\@@asuri@command{query}{
            \makeuri{\csname #2scheme\endcsname}%
                {\csname #2authority\endcsname}%
                {\csname #2path\endcsname}%
                \makeuri@empty%
                {\csname #2fragment\endcsname}%
            \@asuri@changed@true
        }{
        \IfStrEq\@@asuri@command{fragment}{
            \makeuri{\csname #2scheme\endcsname}%
                {\csname #2authority\endcsname}%
                {\csname #2path\endcsname}%
                {\csname #2query\endcsname}%
                \makeuri@empty%
            \@asuri@changed@true
        }{
        \IfStrEq\@@asuri@command{extension}{
            \edef\@asuri@oldpath{\csname #2path\endcsname}
            \StrCount\@asuri@oldpath.[\@asuri@lastdot]
            \ifnum\@asuri@lastdot>0
                \StrBehind[\@asuri@lastdot]\@asuri@oldpath.[\@asuri@extension]
                \IfSubStr\@asuri@extension\@Slash{}{
                    \StrBefore[\@asuri@lastdot]\@asuri@oldpath.[\@asuri@oldpath]
                }
            \fi
            \makeuri{\csname #2scheme\endcsname}%
                {\csname #2authority\endcsname}%
                \@asuri@oldpath%
                \makeuri@empty%
                \makeuri@empty%
            \@asuri@changed@true
        }{}}}
    }{
    \IfBeginWith\@@asuri@command{\@Slash}{
        \@cpath{\csname #2path\endcsname\@@asuri@command}
        \makeuri{\csname #2scheme\endcsname}%
```

```
            {\csname #2authority\endcsname}%
            {\@CanPath}%
            \makeuri@empty%
            \makeuri@empty%
        \@asuri@changed@true
    }{
    \IfBeginWith\@@asuri@command{\@QuestionMark}{
        \expandafter\ifx\csname #2query\endcsname\makeuri@empty
            \StrBehind\@@asuri@command\@QuestionMark[\@@asuri@command]
            \edef\@@asuri@nquery{\@@asuri@command}
        \else
            \edef\@@asuri@nquery{\csname #2query\endcsname\@@asuri@command}
        \fi
        \makeuri{\csname #2scheme\endcsname}%
            {\csname #2authority\endcsname}%
            {\csname #2path\endcsname}%
            {\@@asuri@nquery}%
            \makeuri@empty%
        \@asuri@changed@true
    }{
    \IfBeginWith\@@asuri@command{\@Fragment}{
        \expandafter\ifx\csname #2fragment\endcsname\makeuri@empty
            \StrBehind\@@asuri@command\@Fragment[\@@asuri@command]
            \edef\@@asuri@nfrag{\@@asuri@command}
        \else
            \edef\@@asuri@nfrag{\csname #2fragment\endcsname\@@asuri@command}
        \fi
        \makeuri{\csname #2scheme\endcsname}%
            {\csname #2authority\endcsname}%
            {\csname #2path\endcsname}%
            {\csname #2query\endcsname}%
            {\@@asuri@nfrag}%
        \@asuri@changed@true
    }{}
    }}}
    \edef\@@asuri@ncs{#1}
    \if@asuri@changed@
        \ifx\@@asuri@ncs\@empty
            \asuri{#2}\makeuri@uri
        \else
            \asuri\@@asuri@ncs\makeuri@uri
        \fi
    \fi
}

```

auxiliary code:

```
\def\@Space{ }
\def\trimstring#1{
    \edef\pathsuris@trim@temp{#1}
    \IfBeginWith\pathsuris@trim@temp\@Space{
        \StrGobbleLeft\pathsuris@trim@temp1[#1]
        \trimstring{#1}
    }{
        \IfEndWith\pathsuris@trim@temp\@Space{
            \StrGobbleRight\pathsuris@trim@temp1[#1]
            \trimstring{#1}
        }{
            \edef#1{\pathsuris@trim@temp}
        }
    }
}

% windows paths

\catcode`\.=0
.catcode`.\=12
.let.@BackSlash\
.catcode`.\=0
\catcode`\.=12

\newif\if@windowstopath@inpath@
\def\windows@to@path#1{
    \@windowstopath@inpath@false
    \def\windows@temp{}
    \edef\windows@path{#1}
    \ifx\windows@path\@empty\else
        \expandafter\windows@path@loop\windows@path\windows@path@end
    \fi
    \let#1\windows@temp
}
\def\windows@path@loop#1#2\windows@path@end{
    \def\windows@temp@b{#2}
    \ifx\windows@temp@b\@empty
        \def\windows@continue{}
    \else
        \def\windows@continue{\windows@path@loop#2\windows@path@end}
    \fi
    \if@windowstopath@inpath@
        \ifx#1\@BackSlash
            \edef\windows@temp{\windows@temp\@Slash}
        \else
            \edef\windows@temp{\windows@temp#1}
        \fi
    \else
        \ifx#1:
            \edef\windows@temp{\@Slash\windows@temp:}
            \@windowstopath@inpath@true
        \else
            \edef\windows@temp{\windows@temp#1}
        \fi
    \fi
    \windows@continue
}

\def\path@to@windows#1{
    \@windowstopath@inpath@false
    \def\windows@temp{}
    \edef\windows@path{#1}
    \edef\windows@path{\expandafter\@gobble\windows@path}
    \ifx\windows@path\@empty\else
        \expandafter\path@windows@loop\windows@path\windows@path@end
    \fi
    \let#1\windows@temp
}
\def\path@windows@loop#1#2\windows@path@end{
    \def\windows@temp@b{#2}
    \ifx\windows@temp@b\@empty
        \def\windows@continue{}
    \else
        \def\windows@continue{\path@windows@loop#2\windows@path@end}
    \fi
    \if@windowstopath@inpath@
        \ifx#1/
            \edef\windows@temp{\windows@temp\@BackSlash}
        \else
            \edef\windows@temp{\windows@temp#1}
        \fi
    \else
        \ifx#1/
            \edef\windows@temp{\windows@temp:\@BackSlash}
            \@windowstopath@inpath@true
        \else
            \edef\windows@temp{\windows@temp#1}
        \fi
    \fi
    \windows@continue
}

% kpsewhich

\newif\if@iswindows@\@iswindows@false
\IfFileExists{nul:}{\IfFileExists{/dev/null}{}{\@iswindows@true}}{}

\def\kpsewhich#1#2{\begingroup
  \def\@Space{ }
  \edef\kpsewhich@cmd{"|kpsewhich #2"}
  \everyeof{\noexpand}
  \catcode`\\=12
  \def\par{}
  \edef#1{\@@input\kpsewhich@cmd\@Space}
  \trimstring#1
  \global\let#1#1
\endgroup}

% main directory

\edef\oldpercentcatcode{\the\catcode`\%}
\catcode`\%=12
\let\percent%
\catcode`\%=\oldpercentcatcode

\edef\pwd@cmd{\if@iswindows@ -expand-var \percent CD\percent\else -var-value PWD\fi}
\kpsewhich\stex@maindir\pwd@cmd
\if@iswindows@\windows@to@path\stex@maindir\fi

\def\path@filename#1#2{
    \edef\filename@oldpath{#1}
    \StrCount\filename@oldpath\@Slash[\filename@lastslash]
    \ifnum\filename@lastslash>0
        \StrBehind[\filename@lastslash]\filename@oldpath\@Slash[\filename@oldpath]
        \csedef{#2}{\filename@oldpath}
    \else
        \csedef{#2}{\filename@oldpath}
    \fi
}

\def\path@dropextension#1#2{
    \path@filename{#1}{dropextension@temp}
    \StrCount\dropextension@temp\@Dot[\dropextension@lastdot]
    \ifnum\dropextension@lastdot>0
        \StrBehind[\dropextension@lastdot]\dropextension@temp\@Dot[\dropextension@ext]
        \StrLen\dropextension@ext[\dropextension@lastdot]
        \StrGobbleRight{#1}{\the\numexpr\dropextension@lastdot+1\@Space}[\dropextension@temp]
        \trimstring\dropextension@temp
        \csedef{#2}{\dropextension@temp}
    \else
        \csedef{#2}{#1}
    \fi
}

⟨/package⟩
```

Change History

v1.0

General: First Version with Documentation 1

v1.1

General: adding \baseURI from omdoc.sty and \defpath from modules.sty 1

References

[BFM05] Tim Berners-Lee, Roy T. Fielding, and Larry Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. RFC 3986. Internet Engineering Task Force (IETF), 2005. URL: `http://www.ietf.org/rfc/rfc3986.txt`.