Context-Dependent Parameters (`struct.dtx`)

Charles Duan

May 3, 2023

For some references, different parts need different parameters. For example, multivolume treatises often have different authors per volume, and some old multivolume works were published over multiple years. One possibility would be to cite each volume as a different reference, but that would mean that different volumes of the same work would not receive short or *id.* citations.

As a result, the better option is to allow these complex works to be defined as a single reference, but enable an option for changing parameter values depending on the part cited. The struct parameter enables this.

The value passed to struct is a key-value listing where the keys are the determinative document component (typically but not always the volume number), and the values are another key-value listing of replacement parameters. For example:

```
\defbook{treatise}{
  author=A A,
  title=The Treatise,
  year=1800,
  struct={1={author=B B},
          2={title=The Treatise Revised},
          3={year=1805}},
}
```

Here, the default citation would be "A A, The Treatise (1800)." But if volume 1 is cited then the author would be changed, the title changed if volume 2 is cited, and the year changed if volume 3 is cited.

Only certain reference types support parameter structures, and the particular parameters that can be replaced are determined by the reference type.

This macro should go at the beginning of any citation reference definition that might use a struct. It will be expanded to insert appropriate text to set up the struct. #1 is the (\noexpand'ed) macro for which the content is to be tested against the struct.

```
\def\hi@pstruct@initialize#1{%
    \hi@ifset\hi@kv@struct{%
        \noexpand\hi@pstruct@execute
            \expandafter\noexpand\hi@kv@struct % The macro containing the struct
            \noexpand#1% The testing macro
    }{}%
}
```

Upon execution, `\hi@pstruct@tmp` should be set to the selected struct data or `\relax`.

```
\DeclareRobustCommand\hi@pstruct@execute[2]{%
    \let\hi@pstruct@tmp\relax
    \expandafter\hi@pstruct@execute@\expandafter#2\expandafter{#1}%
}%
\make@find@in{,}
\make@find@in{=}
```

#1 is the macro of the key to test, #2 the struct contents being analyzed.

```
\def\hi@pstruct@execute@#1#2{%
    \find@in{=}{#2}{\hi@pstruct@execute@test{#1}}{}%
}%
```

#1 is the macro of the key to test, #2 the candidate key, and #3 the rest of the struct.

```
\def\hi@pstruct@execute@test#1#2#3{%
    \chop@space@then@run{#2}{\def\reserved@a}%
    \@test \ifx#1\reserved@a \fi{%
        \find@in{,}{#3}{\@tworun\hi@pstruct@execute@use\@gobble}{%
            \hi@pstruct@execute@use{#3}%
        }%
    }{%
        % If there's a comma, ignore everything before it and continue parsing
        % the struct thereafter.
        \find@in{,}{#3}{\@tworun\@gobble{\hi@pstruct@execute@{#1}}}{}%
    }%
}
\def\hi@pstruct@execute@use#1{\@unbrace\hi@pstruct@execute@use@#1\@stop}%
\def\hi@pstruct@execute@use@#1\@stop{\def\hi@pstruct@tmp{#1}}%
```

Macro to be used in reference definers. #1 is the name of the parameter desired. This macro will determine whether a struct is to be invoked or not, producing tokens useful to the reference definer. #2 is the callback; it must be able to withstand expansion in the citation definer.

```
\def\hi@pstruct@call#1#2{%
    \hi@ifset\hi@kv@struct{%
        \noexpand\hi@pstruct@try{#1}{\csname hi@kv@#1\endcsname}{#2}%
    }{#2{\csname hi@kv@#1\endcsname}}%
}
\def\hi@pstruct@use#1{\hi@pstruct@call{#1}{\noexpand\@iden}}%
\def\hi@pstruct@use@font#1#2{\hi@pstruct@call{#1}{\noexpand#2}}%
```

During execution of a citation macro, tries to use a struct's parameter. #1 is the parameter name, #2 the default value, and #3 the callback.

```
\DeclareRobustCommand\hi@pstruct@try[3]{%
    \@test \ifx\hi@pstruct@tmp\relax \fi{%
        #3{#2}%
    }{%
        \begingroup
        % Set parameters as usual
        \@expand{\setkeys{hi}}\hi@pstruct@tmp{i}%
        \expandafter\hi@ifset\csname hi@kv@#1\endcsname{%
            \@expand\hi@pstruct@try@use{\csname hi@kv@#1\endcsname}{ii}{#3}%
        }{%
            \endgroup
            #3{#2}%
        }%
    }%
}
\def\hi@pstruct@try@use#1#2{%
    \endgroup #2{#1}%
}
```