LaTeX_debugging

Eric Moyer

December 29, 2013

Part I

LaTeX Render Debugging

1 Rendering Behavior

Some background notes: The following operation prints the "WARNING:..." text, which might lead one to belive that the table being shown in the notebook is the LaTeX rendering, but it is actually the HTML implementation. As demonstrated later in this notebook, IP[y] executes both the _repr_html_() and the _repr_latex_() methods when rendering an object in the notebook, but (at least in all cases I've tested) displays the HTML rendering in the live notebook. The only way I've found to force IP[y] to render the LaTeX representaion is by running the 'nbconvert' utility on the notebook and rendering it to LaTeX.

In [1]:
```
from ipy_table import *
table = make_table([(1,2),(3,4)])
table
```

```
WARNING: include \usepackage{array} on latex template
```

Out [1]:

1	2
3	4

To test the rendering behavior, we'll create an object that renders a table with the data "1,2,3,4" in its HTML render method and "6,7,8,9" in its LaTeX render method.

In [2]:
```
table._repr_html_()
```

Out [2]:

```
'<table border="1" cellpadding="3" cellspacing="0"  style="border:1px
solid black;border-collapse:collapse;"><tr><td>1</td><td>2</td></tr><t
r><td>3</td><td>4</td></tr></table>'
```

In [3]:
```
table._repr_latex_()
```

Out [3]:

```
'\\begin{tabular}{|l|l|}\n\\cline{1-2}1 & 2 \\\\  \\cline{1-2}\n3 & 4
\\\\  \\cline{1-2}\n\\end{tabular}'
```

In [4]:

```
class render_test(object):
    def _repr_html_(self):
        print 'ENTRY: _repr_html_()'
        return r'''<table border="1" cellpadding="3" cellspacing="0"
            style="border:1px solid black;border-collapse:collapse;">
            <tr><td>1</td><td>2</td></tr>
            <tr><td>3</td><td>4</td></tr></table>'''
    def _repr_latex_(self):
        print 'ENTRY: _repr_latex_()'
        return r'''\begin{tabular}{|l|l|}
               \cline{1-2}6 & 7 \\
               \cline{1-2} 8 & 9 \\
               \cline{1-2}
               \end{tabular}'''
r=render_test()
```

When we render the object we can see that:

- Both rendering methods were executed by IP[y] (both "ENTRY:. . . " print logs were output)
- The HTML rendering is shown in the notebook (the data "1,2,3,4" appears on-screen)
- The LaTeX rendering is used when the notebook is converted to LaTeX format using nbconvert (the cell data "6,7,8,9" appears in the resulting LaTeX document)

In [5]:

```
r
```

```
ENTRY: _repr_html_()
ENTRY: _repr_latex_()
```

Out [5]:

6	7
8	9

From the reading I've done, IP[y] is incapable of rendering tabular LaTeX data in the browser (which I believe stems from IP[y]'s use of MathJax and the omission of tabular support from that engine). Even though the attempt below to force IP[y] to render a tabular structure fails (in the browser it renders the source code in a box), the LaTeX table renders properly when the notebook is converted using nbconvert.

In [6]:

```
from IPython.display import Latex
Latex(r'''\begin{tabular}{|l|l|}
      \cline{1-2}6 & 7 \\
      \cline{1-2} 8 & 9 \\
      \cline{1-2}
      \end{tabular}''')
```

Out [6]:

6	7
8	9

Conclusion: The only way to validate the LaTeX rendering method is to convert notebooks to LaTeX documents and check the output. I've been using the command:

```
ipython nbconvert --to latex --SphinxTransformer.author='Eric Moyer'
--post PDF LaTeX_debugging.ipynb
```

2 Implicit error suppression

IP[y] implicitly suppresses errors encountered when executing implicit rendering methods. That means that if you want to check whether render code is executing without a traceback then you need to call it explicitly. The following imlicit (and errant) code executes without a (reported) traceback:

In [7]:
```
table=make_table([(1,2),(3,4)])
table.set_cell_style(1, 1, color='ThisIsNotAValidColor')
table
```

Out [7]:
```
<ipy_table.IpyTable at 0x5409470>
```

When the same operation is called explicitly, the traceback is reported:

In [20]:
```
######################
# Uncomment the folowing line to generate a traceback.
# Note that if you leave it uncommented and try to "Run All" cells
# in the notebook, then executioin will stop at the traceback
# (subsequent cells will not execute)

#table._repr_latex_()
```

Part II

Issues with the current _repr_latex_() implementation.

I haven't yet tested all possible style options, but I've run some cell color tests and the LaTeX rendering is not working properly.

3 Single Cell color

In the following example a single cell is set to Cyan. The color is defined but it doesn't appear to be applied to any cell. I've never worked with the LaTeX syntax before so I'll have to go educate myself. Nevertheless, LaTeX renders the table but without color (see PDF generated from LaTeX doc. The color appears in the browser/nbviewer but that is the HTML representation).

In [9]:
```
table=make_table([(1,2),(3,4)])
table.set_cell_style(1, 1, color='Cyan')
print table._repr_latex_()
```

```
WARNING: include \usepackage{array} on latex template

% Some macros could be added the first time table is created
\newlength{\Oldarrayrulewidth}
\newcommand{\thickline}[2]{%
  \noalign{\global\setlength{\Oldarrayrulewidth}{\arrayrulewidth}}%
  \noalign{\global\setlength{\arrayrulewidth}{#1}}\cline{#2}%
```

```
  \noalign{\global\setlength{\arrayrulewidth}{\Oldarrayrulewidth}}}

\definecolor{Cyan}{rgb}{0,1,1}

\begin{tabular}{|l|l|}
\cline{1-2}1 & 2 \\  \cline{1-2}
3 & 4 \\  \cline{1-2}
\end{tabular}
```

In [10]:
```
table
```

Out [10]:

1	2
3	4

4 Applying style after render clobbers macros

If a table has been rendered once already,then the subsequent application of a style operation clears the macros (color definitions etc) normally appearing at the beginning of the LaTeX code.

In [24]:
```
table=make_table([(1,2),(3,4)])
table.set_cell_style(1, 1, color='Cyan')
print table._repr_latex_()
```

```
WARNING: include \usepackage{array} on latex template

% Some macros could be added the first time table is created
\newlength{\Oldarrayrulewidth}
\newcommand{\thickline}[2]{%
  \noalign{\global\setlength{\Oldarrayrulewidth}{\arrayrulewidth}}%
  \noalign{\global\setlength{\arrayrulewidth}{#1}}\cline{#2}%
  \noalign{\global\setlength{\arrayrulewidth}{\Oldarrayrulewidth}}}

\definecolor{Cyan}{rgb}{0,1,1}

\begin{tabular}{|l|l|}
\cline{1-2}1 & 2 \\  \cline{1-2}
3 & 4 \\  \cline{1-2}
\end{tabular}
```

In [26]:
```
table.set_cell_style(0, 1, color='Cyan')
print table._repr_latex_()
```

```
\begin{tabular}{|l|>{\columncolor{Cyan}}l|}
\cline{1-2}1 & 2 \\  \cline{1-2}
3 & 4 \\  \cline{1-2}
\end{tabular}
```

5 Applicaiton of basic style causes rendering errors

In [29]:
```
table=make_table([(1,2),(3,4)])
table.apply_theme('basic')
print table._repr_latex_()
```

```
WARNING: include \usepackage{array} on latex template

% Some macros could be added the first time table is created
\newlength{\Oldarrayrulewidth}
\newcommand{\thickline}[2]{%
  \noalign{\global\setlength{\Oldarrayrulewidth}{\arrayrulewidth}}%
  \noalign{\global\setlength{\arrayrulewidth}{#1}}\cline{#2}%
  \noalign{\global\setlength{\arrayrulewidth}{\Oldarrayrulewidth}}}

\definecolor{Ivory}{rgb}{1,1,0}
\definecolor{LightGray}{rgb}{0.75,0.75,0.75}

\begin{tabular}{|>{\columncolor{Ivory}}l|>{\columncolor{Ivory}}l|}
\cline{1-2}\textbf{1} & \textbf{2} \\  \cline{1-2}
3 & 4 \\  \cline{1-2}
\end{tabular}
```

In [31]:
```
######################
# Uncomment the folowing line to render the table.  The table
# will render propery in the notebook (HTML) but will fail
# the nbconvert conversion to LaTeX

#table
```

Part III

Success Criteria

At the very least, a successful LaTeX render implementation will need to be capable of rendering the ipy_table-Reference notebook Without LaTex conversion errors (since that notebook contains examples of all ipy_table styles). That means that the command...

```
ipython nbconvert --to latex --post PDF ipy_table-Reference.ipynb
```

...should execute without error. Today it fails.

At best, a successful implementation should render as many aspects of the HTML implementation as are possible in LaTeX's syntax. I'm not familiar with the details of the LaTeX syntax so I don't know how many of ipy_table's current capabilities may be unrenderable in LaTeX.