Testing admons

hpl

Feb 17, 2014

1 Introduction

First some ordinary text to compare font sizes in admonitions and the surrounding text.

1.1 Code

Need some code outside admons for color and font comparisons:

```
    def some_code(x):
        return sin(x)*exp(1-x)
```

And some plain text verbatim:

```
x=1.0 y=0.9 z=0.4
x=1.1 y=0.3 z=0.1
```

1.2 Quotes and boxes

Here is a plain quote environment.

> Sayre's law states that "in any dispute the intensity of feeling is inversely proportional to the value of the issues at stake."
> By way of corollary, it adds:
> "That is why academic politics are so bitter."
> *Source*: wikipedia

Does quotes with title also work? No...cannot work in LaTeX and HTML and then it does not make sense to support it.

A plain *box* is sometimes useful. Let's show it here for comparison with admons (especially the block admon has much in common with a box). The box is more aimed at framing a law or an equation.

First a simple block with text, an equation, and a list:

A generic equation

$$f(x) = 0$$

must be solved by a numerical method, such as

- Newton's method
- The Bisection method
- Fixed-point (Picard) iteration by rewriting $f(x) = x - g(x)$
- The Secant method

Now test a box with equation only (note that this line continues the box, it is not a new paragraph):

$$f(x) = \sin(x)e^{1-x} \tag{1}$$

Let's begin a new paragraph and show a box with code only:

```
    def some_code(x):
        return sin(x)*exp(1-x)
```

1.3 Admonitions

Let us start with a plain warning environment.

Warning. And here is a warning about something to pay attention to. We test how the heading behave and add quite some extra texts in comparison with the other admons.

- and a list
- with items

We continue with more text to see how that affects the layout. And more and more text. And more and more text. And more and more text. And more and more text. And more and more text. And more and more text.

Test warning with title:

Title ending with math $\sqrt{2} \approx 1.4$. And here comes some text with bad news in larger font.

Also some code:

```
def f(x):
    return x
```

And a complete program

```
print "Hello, World!"
```

Test warning with large title with math:

Watch out for $\nabla \cdot \boldsymbol{u} = 0$ equations. Divergence freedom is often problematic from a numerical point of view.

Then we test a block, which is guaranteed to never have any admon icon.

Block with title. Here is a block of text with title. It is typeset *without any icon* and is useful when you want some admons with icon and some without. With the small font size, as used here, one can have more comment-style text or text that really goes deeper or talks about fun facts that are not strictly necessary for the main flow of understanding.

Here is a block of text with no title. As above, it is typeset without any icon and is useful when you want some admons with icon and some without.

The next admonition features a title "Note, eventually!".

Note, eventually! Ah, we are soon close to the end (with illegal font size specification!). But first a bit of math where we define θ and $\boldsymbol{r}$:

$$\theta = q^2,$$
$$\boldsymbol{r} = \varrho \boldsymbol{i}$$

Point1. Ah, we are soon close to the end.

Question. So, how many admonition environments does Doconce support?

Question.

1. Once more, how many admonition environments does Doconce support?

Tip. It is of outmost important to

1. stay cool
2. read hints and tips carefully

Because here the thing is to do

```
import urllib

def grab(url, filename):
    urllib.urlretrieve(url, filename=filename)
```

Next is a warning without a title ("none" implies no title).

And here comes some text with bad news.

1.4 Going deeper environments

Here is a long notice environment with a custom title and much text, math and code.

Going deeper. We have some equations that should be preceded by much text, so the task is to write and write. The number of words, and not the meaning, is what counts here. We need desperately to fill up the page in the hope that some admonitions will experience a page break, which the LaTeX environment should handle with ease.

Let us start with some equations:

$$\frac{Du}{dt} = 0$$
$$\frac{1}{2} = 1/2$$
$$\frac{1}{2}\boldsymbol{x} = \boldsymbol{n}$$

The implementation of such complicated equations in computer code is task that this "Going deeper" environment targets.

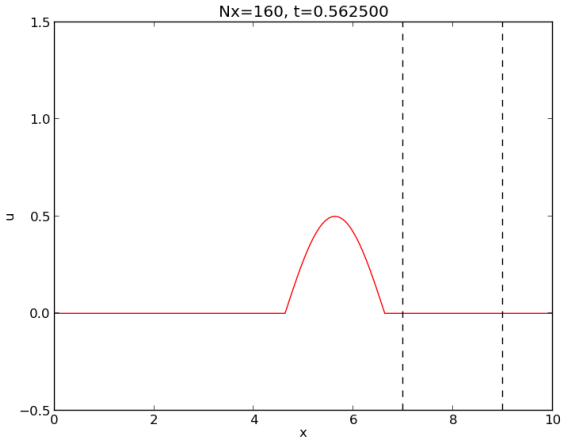
```
def Dudt(u):
    r = diff(u, t) + u*grad(u)
    return r

half = 0.5
x = 2*n
```

And some more text that can help going into the next page. Longer computer code requires vertical space:

```
class Diff:
    def __init__(self, f, h=1E-5):
        self.f = f
        self.h = float(h)

class Forward1(Diff):
    def __call__(self, x):
        f, h = self.f, self.h
        return (f(x+h) - f(x))/h

class Backward1(Diff):
    def __call__(self, x):
        f, h = self.f, self.h
        return (f(x) - f(x-h))/h

class Central2(Diff):
    def __call__(self, x):
        f, h = self.f, self.h
        return (f(x+h) - f(x-h))/(2*h)

class Central4(Diff):
    def __call__(self, x):
```

```
            f, h = self.f, self.h
            return (4./3)*(f(x+h)   - f(x-h))  /(2*h) - \
                   (1./3)*(f(x+2*h) - f(x-2*h))/(4*h)

    class Central6(Diff):
        def __call__(self, x):
            f, h = self.f, self.h
            return (3./2) *(f(x+h)   - f(x-h))  /(2*h) - \
                   (3./5) *(f(x+2*h) - f(x-2*h))/(4*h) + \
                   (1./10)*(f(x+3*h) - f(x-3*h))/(6*h)

    class Forward3(Diff):
        def __call__(self, x):
            f, h = self.f, self.h
            return (-(1./6)*f(x+2*h) + f(x+h) - 0.5*f(x) - \
                    (1./3)*f(x-h))/h
```

And then we add a figure too.



1.5 The end

A bit of text before the summary, which we now call "Concluding remarks, for the novice", just because we can.

Concluding remarks, for the novice. We can summarize the most important things with admons: they have a different typesetting, and they may have a symbol. Titles should be optional.

Remark. The `remarks` and `hint` environments are not allowed outside exercises (and problems and projects too).