### Eliminate implicit magic in Proc.new and Kernel#proc

**11/12/2014 02:18 PM - headius (Charles Nutter)**

<table>
<thead>
<tr>
<th>Status:</th>
<th>Open</th>
</tr>
</thead>
<tbody>
<tr>
<td>Priority:</td>
<td>Normal</td>
</tr>
<tr>
<td>Assignee:</td>
<td>matz (Yukihiro Matsumoto)</td>
</tr>
<tr>
<td>Target version:</td>
<td>3.0</td>
</tr>
</tbody>
</table>

**Description**

Proc.new and Kernel#proc have a little known feature: if called without a block, they capture whatever block was passed to the current method.

I propose that this feature should be removed, finally, since it:

- Doesn't enhance readability (where is this block coming from?)
- Doesn't reflect any other behavior in Ruby
- Can lead to bugs (call either without a block accidentally and you aren't sure what you'll get)

I believe this was an implementation artifact in MRI, since the most recently-pushed block would still be on global stacks, which is where the logic for proc and Proc.new looked for it.

All argument syntaxes now support &block, which I believe is the correct way to clearly, explicitly capture the incoming block into an object.

**Thoughts?**

**Related issues:**
Copied to Ruby master - Feature #15554: warn/error passing a block to a method... | Open

**History**

**#1 - 11/12/2014 03:23 PM - marcandre (Marc-Andre Lafortune)**

- Category set to core
- Assignee set to matz (Yukihiro Matsumoto)

I agree.

Deprecate first (2.2?), remove afterwards.

This would also simplify things if and when we want to warn/raise on unused blocks when calling user methods.

**#2 - 12/08/2014 08:44 PM - headius (Charles Nutter)**

Adding a deprecation warning would be easy if we can get buy-in from matz.

matz: ball's in your court, I think!

**#3 - 05/28/2019 08:41 PM - k0kubun (Takashi Kokubun)**

- Copied to Feature #15554: warn/error passing a block to a method which never use a block added