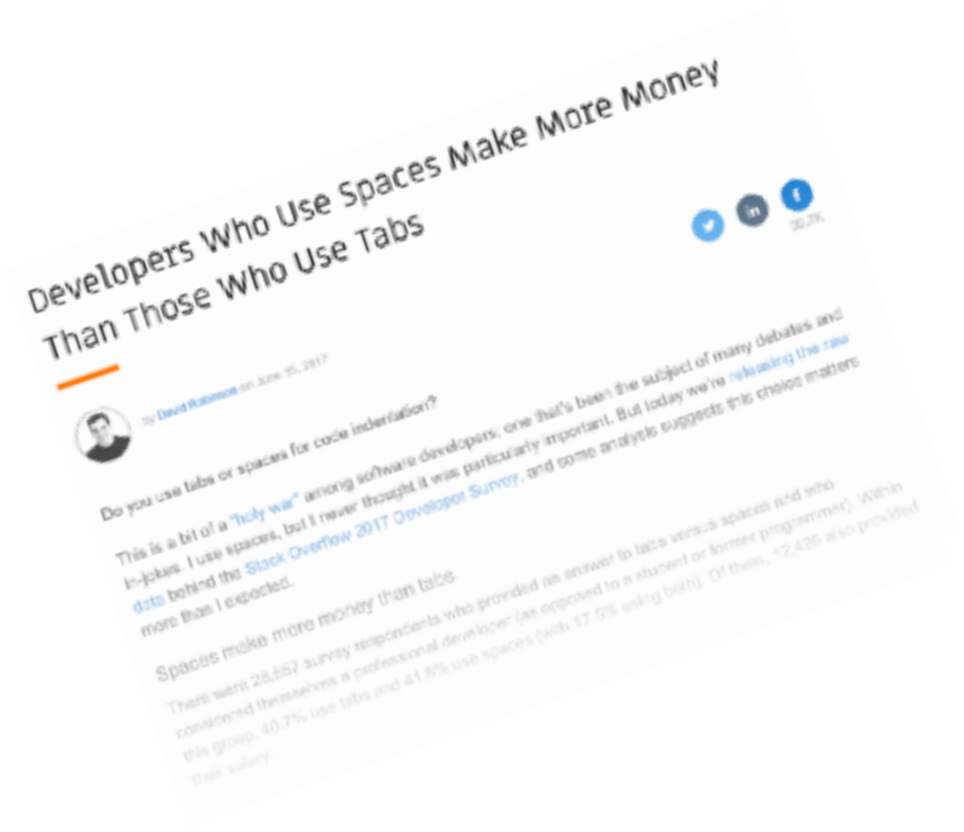


Developers Who Use Spaces Make More Money Than Those Who Use Tabs

by David Robinson on June 15, 2017

Do you use tabs or spaces for code indentation?

This is a bit of a "holy war" among software developers; one that's been the subject of many debates and in-jokes. I use spaces, but I never thought it was particularly important. But today we're releasing the raw data behind the Stack Overflow 2017 Developer Survey, and some analysis suggests this choice matters more than I expected.

Spaces make more money than tabs

There were 28,657 survey respondents who provided an answer to tabs versus spaces and who considered themselves a professional developer (as opposed to a student or former programmer). Within this group, 40.7% use tabs and 41.8% use spaces (with 17.5% using both). Of them, 12,426 also provided their salary.

Manual Formatting

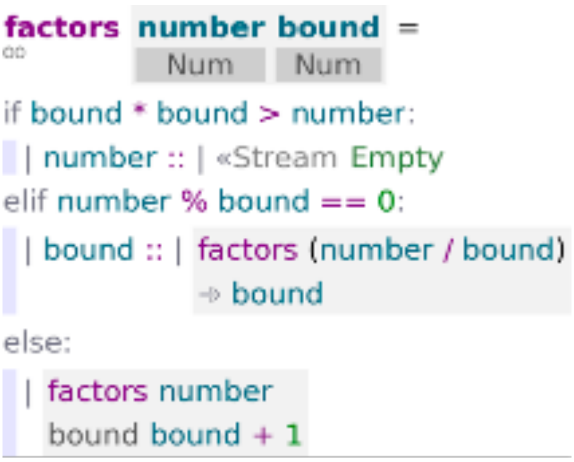
```diff
-     const Rectangle<int> scaled (area * Point<float> (peerBounds.getWidth()  / (float) getWidth(),
-                                                       peerBounds.getHeight() / (float) getHeight()));
+     auto scaled = area * Point<float> (peerBounds.getWidth()  / (float) getWidth(),
+                                        peerBounds.getHeight() / (float) getHeight());
```

A typical C++ code diff. The programmer maintains the spacing manually.

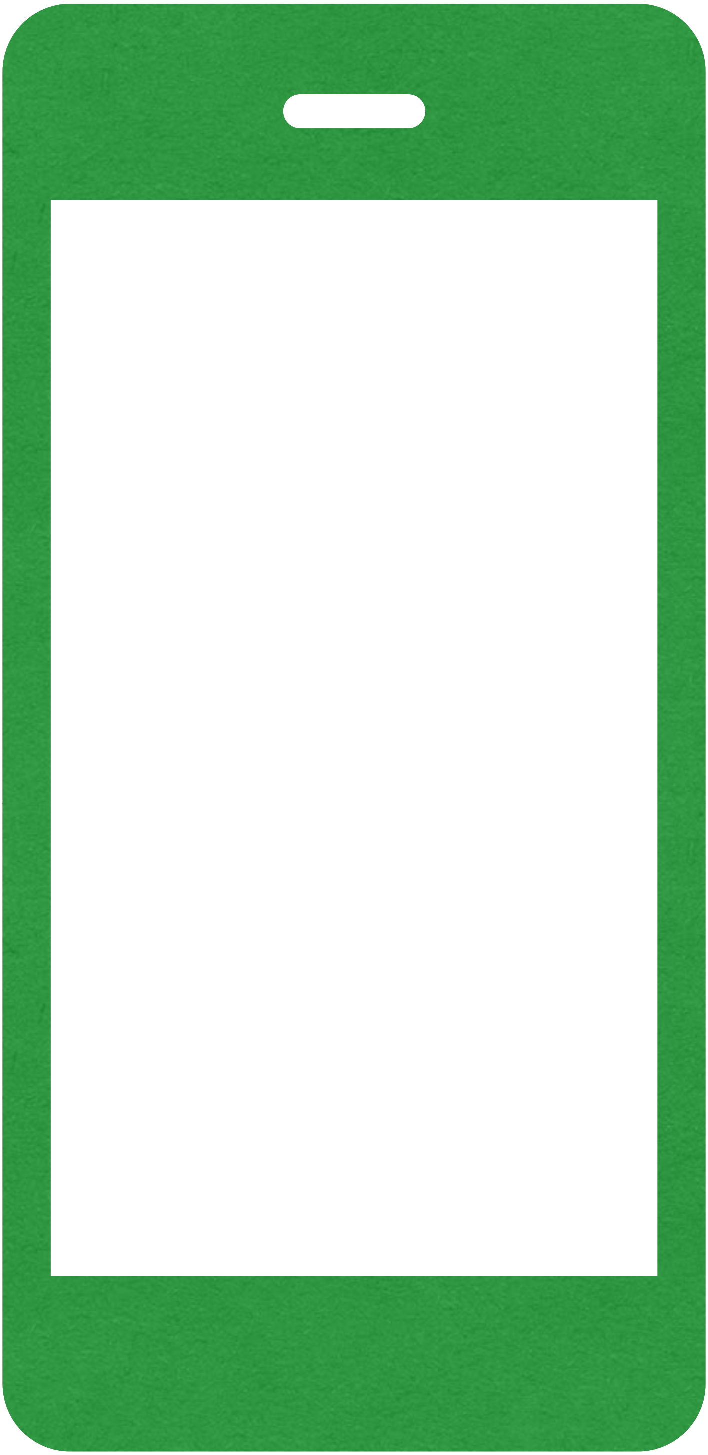
Programmers maintain whitespace, deciding how to indent their code, split their lines and align function arguments, to make the code readable while fitting the screen width.

Automatic Layout

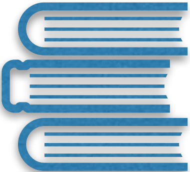
```
factors number bound = if   bound * bound > number: | number :: | «Stream Empty
        Num    Num     elif number % bound == 0:    | bound :: | factors (number / bound)
                                                    |          | ⇛ bound
                       else:                        | factors number
                                                    | bound bound + 1
```



```
factors number bound =
        Num    Num
if bound * bound > number:
  | number :: | «Stream Empty
elif number % bound == 0:
  | bound :: | factors (number / bound)
               ⇾ bound
else:
  | factors number
    bound bound + 1
```



- Convenient
- Responsive



- Consistent
- No conflicts

go fmt

Lamdu (Summary)

- No syntax errors
- No name errors
- Type mismatches with better blame assignment
- Live debugging
- Projectional syntactic sugar
- Automatic responsive layout
- Structural source control - eliminating spurious merge conflicts
- Localisation support (i18n)
- Rich custom visualisations for evaluation results of different types
- Integrated unit tests

Planned features

go fmt

Manual Formatting

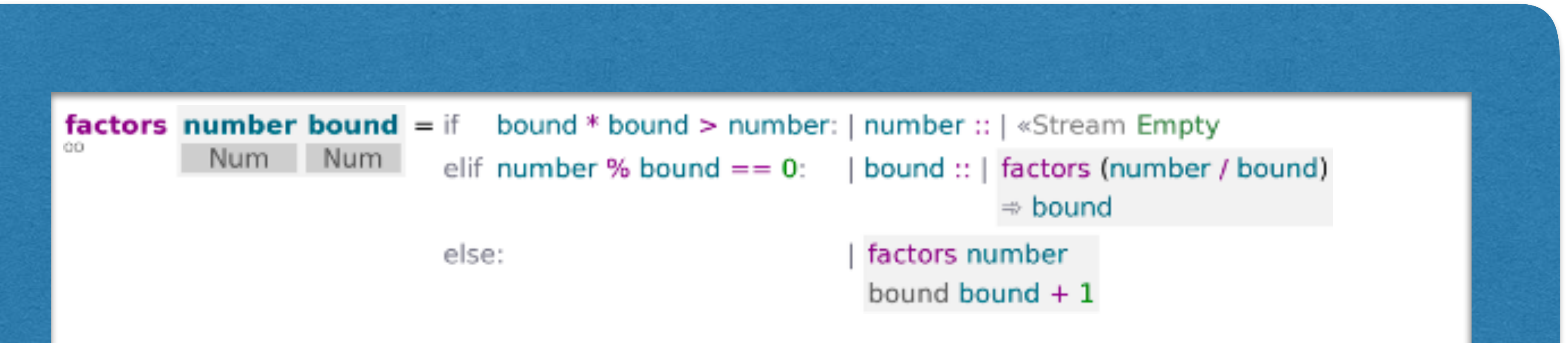
Programmers maintain whitespace, deciding how to indent their code, split their lines and align function arguments, to make the code readable while fitting the screen width.

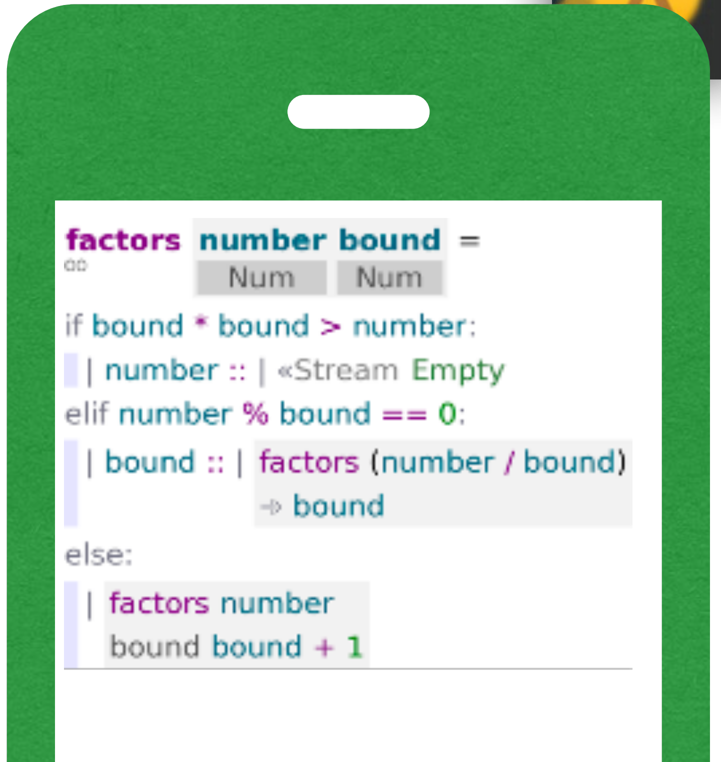
Developers Who Use Spaces Make More Money Than Those Who Use Tabs

```
-     const Rectangle<int> scaled (area * Point<float> (peerBounds.getWidth()  / (float) getWidth(),
-                                                      peerBounds.getHeight() / (float) getHeight()));
+     auto scaled = area * Point<float> (peerBounds.getWidth()  / (float) getWidth(),
+                                        peerBounds.getHeight() / (float) getHeight());
```
A typical C++ code diff. The programmer maintains the spacing manually.

Automatic Layout

- Convenient
- Responsive
- Consistent
- No conflicts


```
factors number bound = if    bound * bound > number:  | number :: | «Stream Empty
        Num    Num     elif  number % bound == 0:     | bound :: | factors (number / bound)
                                                                    ⇒ bound
                       else:                          | factors number
                                                        bound bound + 1
```


```
factors number bound =
        Num    Num
if bound * bound > number:
  | number :: | «Stream Empty
elif number % bound == 0:
  | bound :: | factors (number / bound)
                ⇒ bound
else:
  | factors number
    bound bound + 1
```