CoCoA-5 - Bug #1002

Emacs UI: cocoa5-send-line skips too many chars

24 Jan 2017 19:19 - John Abbott

Status:	Closed	**Start date:**	24 Jan 2017
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	EmacsUI	**Estimated time:**	2.60 hours
Target version:	CoCoA-5.2.0 spring 2017	**Spent time:**	2.60 hours

Description

While trying to understand how cocoa5-send-line works, I discovered that it can skip too far forwards sometimes.

This is probably not serious since no one has reported any such problems, but it should be rectified.

Related issues:

Related to CoCoA-5 - Design #998: Emacs UI: sending empty input from cocoa5 b...	**Closed**	**18 Jan 2017**

History

#1 - 24 Jan 2017 19:21 - John Abbott

Here is an example:

```
-- Line 1
(
1+2);
```

Doing "send line" on the first line then moves the cursor to the **third** line; the open bracket on the second line is skipped over!

#2 - 24 Jan 2017 19:21 - John Abbott

- Related to Design #998: Emacs UI: sending empty input from cocoa5 buffer added

#3 - 24 Jan 2017 19:25 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

The problem lies in the "mysterious" regexp given to skip-chars-forward.
Currently the regexp is **"-*\n-*"** but I have no idea where that came from.
I believe it means skip over all **- *** and any char between **n** and ***** (in the ASCII encoding). To be honest I do not know how it manages to skip over the whitespace that it does skip over.

Anyway, I suggest replacing the regexp with something more comprehensible (or at least add an explanation for what the current regexp means).

#4 - 24 Jan 2017 19:31 - John Abbott

This leads naturally to the question: which chars do we want to skip over?

At least: Newline, space and tab.

Anything else?
Emacs offers char classes in regexps: for instance

- **[:blank:]** matches space and tab
- **[:space:]** matches any char with whitespace syntax -- I don't know what that means!

#5 - 24 Jan 2017 19:55 - John Abbott

- % Done changed from 10 to 20

I think Anna might like to skip over minus signs since she sometimes uses a line of minus signs to separate sections in a demo file... I don't know whether it makes sense to skip over them.

If we do skip minus signs then a file like the following would cause trouble:

```
3
-
1;
```

Read as a whole file this is correct input, and produces the result 2 (of course!).
Sent one line at a time, the file is incorrect if the line containing just the minus sign is skipped over!

Similar arguments apply to (forward) slash (which can also be used to introduce comments).

#6 - 24 Jan 2017 21:12 - John Abbott

The following file will give different results (but no errors) if you read the whole file in one go, or send the file line by line:

```
len
("()")
;
```

The point being that with the weird regexp the line containing ("()") is skipped completely.

JAA is feeling ever more convinced that the old regexp is **bad**.

#7 - 17 Mar 2017 16:12 - John Abbott

- Status changed from In Progress to Feedback

- Assignee set to John Abbott

- Target version changed from CoCoA-5.?.? to CoCoA-5.2.0 spring 2017

- % Done changed from 20 to 90

- Estimated time set to 2.60 h

I made a change to the regexp a while ago. All the examples in the comments above seem to work in a reasonable way.

#8 - 27 Apr 2017 11:31 - John Abbott

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

Closing after 3 months in feedback, and no problems have come up.