Proposed Pre-Alpha API

Watson Editor Included Features

Syntax Highlighting
Click Event
Mouse Hover Event
Insertion Bar
Line Numbers
Adding Lines
Deleting Lines

Constructor

Constructor(divID, lineNumBool, syntaxHighlightingBool ,lineNumStart, cellWidth, insertBetweenRowsBool);

Functionality

- A div will need to be initialized and the constructor will find this div using the divID parameter. That div will be used to insert all the html of the editor.
- *lineNumBool* is a boolean that controls whether or not line numbers will be displayed.
- *syntaxHighlightingBool* is a boolean that controls whether keywords need to be highlighted or not.
- *lineNumStart* is an integer that specifies what line number to start with.
- *cellWidth* is an integer specifying the width of each cell. If cellWidth is dynamic, -1 should be supplied here.
- *insertBetweenRowsBool* is a boolean indicating if the user should be able to put the insertion pointer between two rows to insert lines between other lines. In the case for Database lab, this boolean would be false. Essentially, if this boolean is true, the last line will always be the line where insertion takes place.

Proposed Functions

rowToArr(index);

Functionality

- This function takes a row index and returns an array where each element of the array is the text from the cells of that row. Does not include line number cell. Several functions existed in the first version of this API that obtained data from the editor. This is the only one that remains. The idea is that a lab can build whatever it needs from this function. For example, if one was to want a 2D array of all the rows in the editor, he/she would simply call rowToArr() for all rows and build his/her own 2-D array (Assembly will be like this, I believe).

getRowCount();

Functionality

- This function returns the number of rows in the program definition of the editor.

addRow(index,arrayOfObjects);

Functionality

- This function takes a row index (Where you want to insert row) and an array of objects. Each object will contain a string containing desired cell text, and an array of class names.

deleteRow(index);

Functionality

- This function deletes a row of a given index. The idea behind this function is that when a user attempts to delete a row from the editor by clicking on the line's line number, the cell click listener function will be called within the lab. The lab can detect that this is an attempt at deletion due to the fact that the first cell in that row (index 0) will be the clicked cell (it is passed to the clickFunc() within the lab specified by the setCellClickListener below). The lab will then perform its check to ensure the line can be deleted. If it can be deleted, the lab will call the editor's deleteRow() function with that specified row index. In other words, the logic to ensuring a deletion is valid will be handled by each individual lab. Once it's declared valid, it calls the editor to perform the actual deletion. If it's not a valid deletion, the lab would probably call Watson's Alert Dialog saying that isn't a valid deletion.

selectRowByIndex(index);

Functionality

- Selects a row in the editor using a row index.

moveInsertionBarCursor(index);

Functionality

- This function moves the arrow within the insertion bar to the specified index.

getSelectedRowIndex();

Functionality

- This function returns the currently selected row index

setCellClickListener(clickFunc);

Functionality

- *clickFunc* will be a function inside the calling class that takes one parameter; the DOM cell object. This means when a user clicks any cell within the editor, clickFunc() will be called with the cell as a parameter. The lab can call cell.parent() to get the row. This is important because you must be able to look at the context around the clicked cell to be able to tell what it is the user is trying to do. For example, the lab will need to check if the line is a variable declaration, a document.write, etc to show the appropriate dialog.

setInsertionBarMouseEnterListener(mouseEnterFunc);

Functionality

- *mouseEnterFunc()* will be a function inside the calling class that takes one parameter; the row index of the line within the insertion bar that the mouse has entered. This means when a user's mouse enters the insert bar, mouseEnterFunc() will be called with the row index of the line within the insertion bar that the mouse has entered. mouseEnterFunc() will be responsible for determining if the cursor within the insertion bar can be shown at this index. If so, this function will need to call moveInsertionBarCursor() with this index.

CSS Class Names

Code: Every table cell will have this class name.
Keyword: blue
Literal: brown
Comment: green
Selected: red
openParen=> (
closeParen=>)
openBrack=> {
closeBrack=> }