CSCD01 Project Deliverable 3

Issue 1

Name: Feature request: Displaying a list of search results

Link: https://github.com/mozilla/pdf.js/issues/7605

Description:

pdf.js supports searching in the document for words and phrases, but in the same way that Chrome does it, where we can 'jump' between previous and next search results one at a time. The user who created this feature request has asked that a feature be implemented that allows users to see a list of the search results all at once, and not have to go through each one at a time. Moreover, the results should be clickable and scroll the user to the location of the found word whilst keeping current search functionality (highlight of found words) intact..

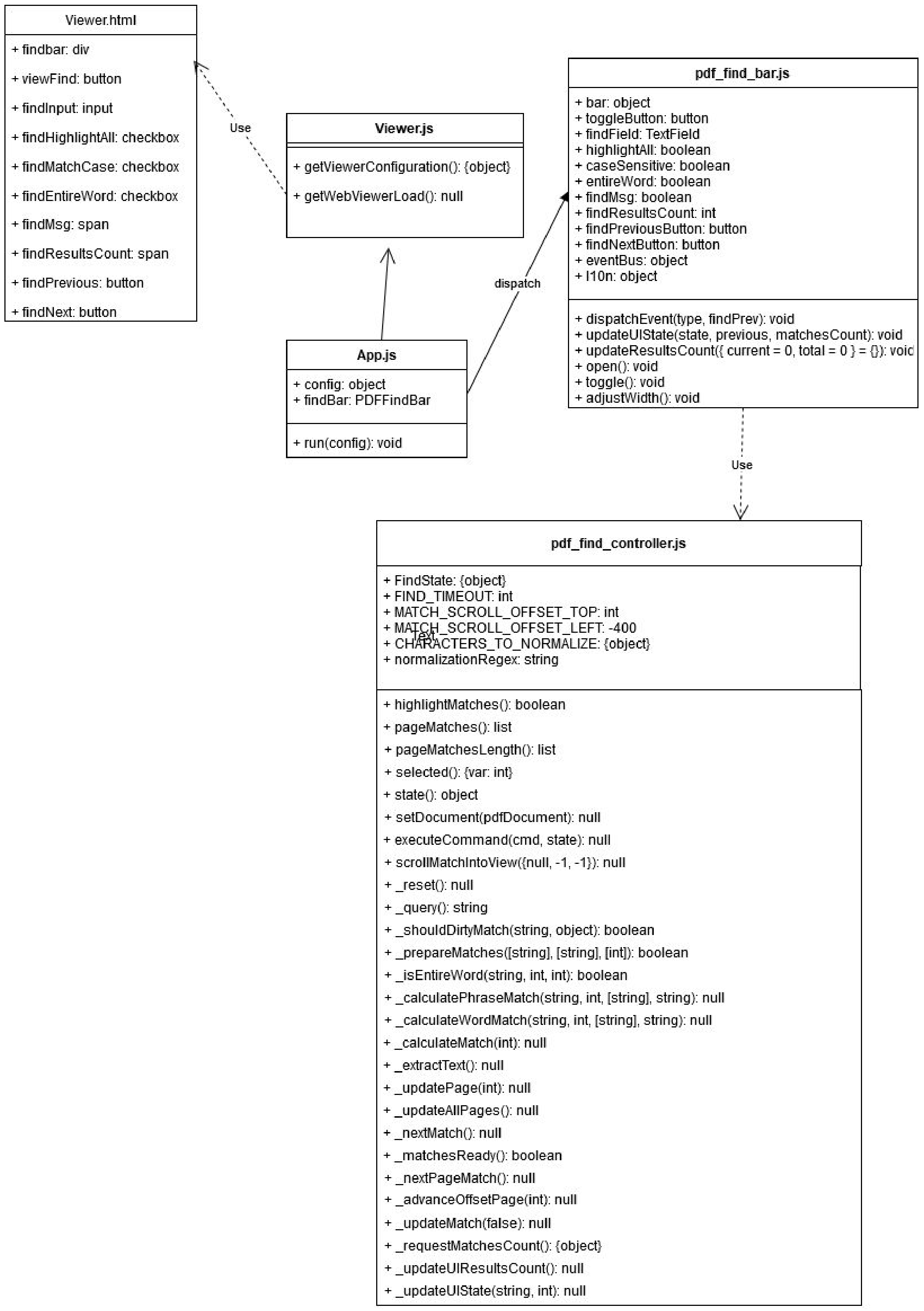
Potential Fixes:

The following files would have to be edited in this feature request:

- **Viewer.html** is where all initial HTML is implemented. This is where the raw html for our search display would go.
- **Viewer.js** reads and gets all relevant elements in viewer.html and splits them into components in the config file. Our relevant one for the search would be the findbar element inside of config.
- Viewer.js then passes the config to **app.js**' run method, where the UI is initialized and passes the config of the findBar to the next file.
- **Pdf_find_bar.js** takes the config and uses **pdf_find_controller.js** to search the document and highlight/bring into focus search results.

Our potential implementation for this new feature would affect all of the above files to implement a UI for displaying a series of search results and adding functionality to get the list of all search results. Our implementation would require extensive function additions to pdf_find_controller to create functions specifically for our UI addition.

The outlined files can be illustrated with a UML to show their relationship in the following:


Viewer.html
+ findbar: div
+ viewFind: button
+ findInput: input
+ findHighlightAll: checkbox
+ findMatchCase: checkbox
+ findEntireWord: checkbox
+ findMsg: span
+ findResultsCount: span
+ findPrevious: button
+ findNext: button
Use
Viewer.js
+ getViewerConfiguration(): {object}
+ getWebViewerLoad(): null
App.js
+ config: object
+ findBar: PDFFindBar
+ run(config): void
dispatch
pdf_find_bar.js
+ bar: object
+ toggleButton: button
+ findField: TextField
+ highlightAll: boolean
+ caseSensitive: boolean
+ entireWord: boolean
+ findMsg: boolean
+ findResultsCount: int
+ findPreviousButton: button
+ findNextButton: button
+ eventBus: object
+ l10n: object
+ dispatchEvent(type, findPrev): void
+ updateUIState(state, previous, matchesCount): void
+ updateResultsCount({ current = 0, total = 0 } = {}): void
+ open(): void
+ toggle(): void
+ adjustWidth(): void
Use
pdf_find_controller.js
+ FindState: {object}
+ FIND_TIMEOUT: int
+ MATCH_SCROLL_OFFSET_TOP: int
+ MATCH_SCROLL_OFFSET_LEFT: -400
Text
+ CHARACTERS_TO_NORMALIZE: {object}
+ normalizationRegex: string
+ highlightMatches(): boolean
+ pageMatches(): list
+ pageMatchesLength(): list
+ selected(): {var: int}
+ state(): object
+ setDocument(pdfDocument): null
+ executeCommand(cmd, state): null
+ scrollMatchIntoView({null, -1, -1}): null
+ _reset(): null
+ _query(): string
+ _shouldDirtyMatch(string, object): boolean
+ _prepareMatches([string], [string], [int]): boolean
+ _isEntireWord(string, int, int): boolean
+ _calculatePhraseMatch(string, int, [string], string): null
+ _calculateWordMatch(string, int, [string], string): null
+ _calculateMatch(int): null
+ _extractText(): null
+ _updatePage(int): null
+ _updateAllPages(): null
+ _nextMatch(): null
+ _matchesReady(): boolean
+ _nextPageMatch(): null
+ _advanceOffsetPage(int): null
+ _updateMatch(false): null
+ _requestMatchesCount(): {object}
+ _updateUIResultsCount(): null
+ _updateUIState(string, int): null

Issue 2

Name: No spaces between words when copying text #10640

Link: **https://github.com/mozilla/pdf.js/issues/10640**

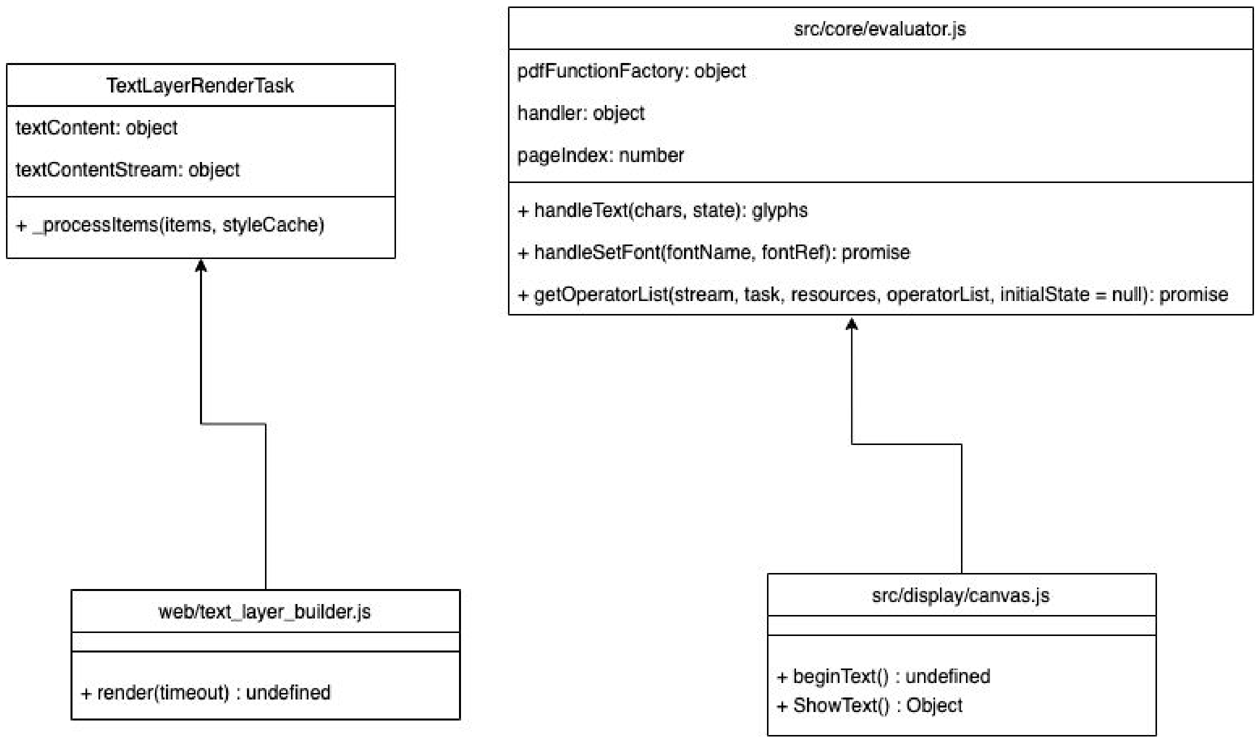
Description:

When copying text in certain PDF files, the copied text does not contain any whitespaces between words despite these spaces being clearly visible in the PDF. When these PDFs are viewed in other PDF viewers, the text is copied correctly with appropriate whitespace.

Potential Fixes:

- **src/display/text_layer.js**: Manually adding extra spaces at the end of text fragments to create spaces in between individual spans in the html
- Because the text rendered to the canvas is correct, route the data that is piped to the canvas rendering section to **text_layer.js** and use that data instead

Implementing these changes will most likely involve the **text_layer.js** files as this file creates the spans with invisible text, or the *text layer* of the pdf, which is how text on the pdf is able to be highlighted. It is also possible that the spaces are not present in the text because of the symbolicFont property of the font of the PDF. Because symbolic fonts are interpreted differently than regular fonts, spaces in the original pdf may be misinterpreted or they are not being recognized at all, thus not appearing in the text layer. This may explain why the font is rendered to the canvas fine, but appears differently in the text layer. Fixes here would have to be made earlier in the text processing pipeline, possibly in **src/shared/message_handler**


TextLayerRenderTask
textContent: object
textContentStream: object
+ _processItems(items, styleCache)
web/text_layer_builder.js
+ render(timeout) : undefined
src/core/evaluator.js
pdfFunctionFactory: object
handler: object
pageIndex: number
+ handleText(chars, state): glyphs
+ handleSetFont(fontName, fontRef): promise
+ getOperatorList(stream, task, resources, operatorList, initialState = null): promise
src/display/canvas.js
+ beginText() : undefined
+ ShowText() : Object

Issue 3

Name: Extra white spaces between words #11670

Link: https://github.com/mozilla/pdf.js/issues/11670

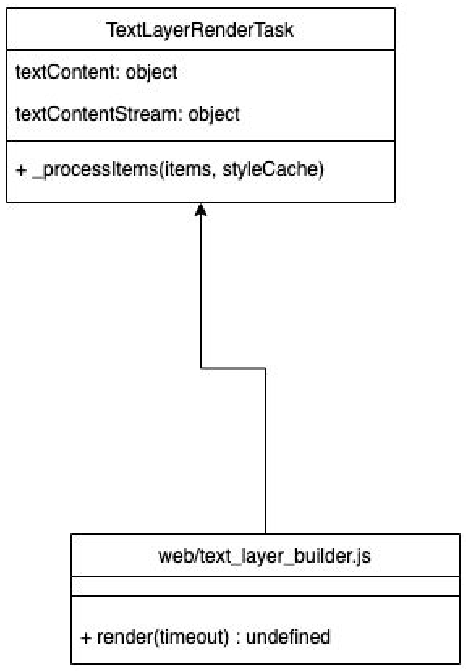
Description:

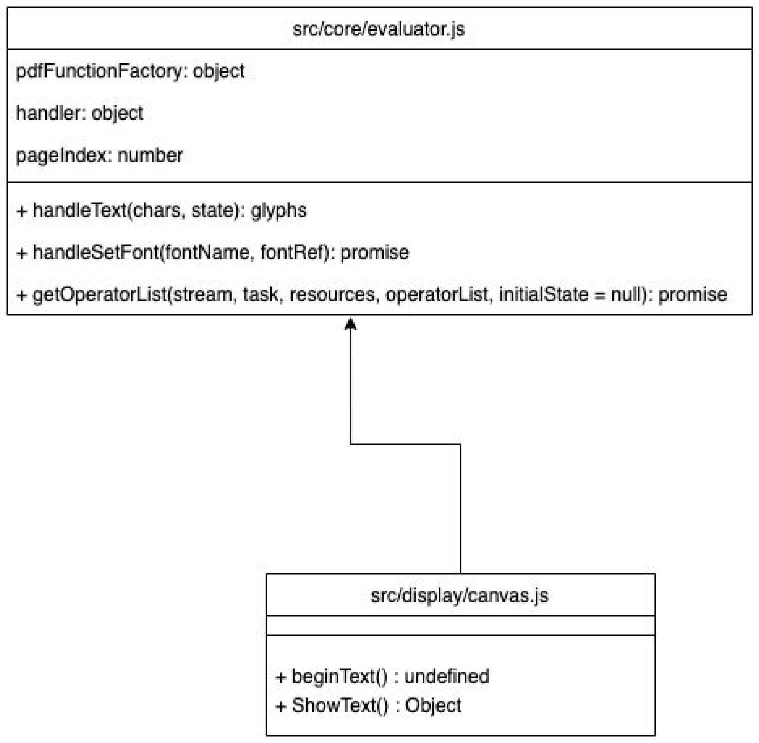
In some parts of certain PDFs, the text layer (for copying) doesn't match up with its corresponding drawn section on the canvas (the visible text) due to extra whitespace surrounding words as well as incorrect font size.

Potential Fixes:

- **src/display/text_layer.js**: Re-evaluate the width of text that is rendered to the canvas and approximate the scaleX style attribute of the associated span
- **src/display/text_layer.js**: Trim spans for unique whitespace characters when generating text layer

Implementation of this change will mostly likely involve **text_layer.js** as the issues stem from the way the text appears in the text layer. The issues regarding the text highlighting not matching the text rendered to the canvas are the cause of the text in the text layer being scaled incorrectly, such that the words in the text layer do not occupy the same horizontal space as the word in the canvas that the user sees. Potential fixes would most likely have to look into improper text scaling specifically onlines where the font changes to something irregular.


TextLayerRenderTask
textContent: object
textContentStream: object
+ _processItems(items, styleCache)
web/text_layer_builder.js
+ render(timeout) : undefined


src/core/evaluator.js
pdfFunctionFactory: object
handler: object
pageIndex: number
+ handleText(chars, state): glyphs
+ handleSetFont(fontName, fontRef): promise
+ getOperatorList(stream, task, resources, operatorList, initialState = null): promise
src/display/canvas.js
+ beginText() : undefined
+ ShowText() : Object

Selected Issues

For deliverable 4, since the bugs we have here were not big enough individually, with the approval of the TA we will be doing both of them. These issues were chosen because they seem reasonable to complete in the two weeks that we have for deliverable 4, and they are both areas that we have some familiarity with, having looked into them a bit for deliverable 2. Additionally, these features and bugs are something we would personally like to see fixed in the viewer when we use it.

Implementation Plan

As Outlined in the issue descriptions, we plan to follow the code to figure out exactly how the system works and interacts, to see where the issue needs to be implemented

Issue 1

1. Add option in existing search bar to “display all”
2. Create the container for displaying the results
3. Create search functions to get all instances of a search query
4. Display the query

Issue 2

1. In the getOperatorList function in evaluator.js, when it handles text store the strings (i.e. chars input to handleText()) as well as the spaces
2. Send them to the constructor for TextLayerRenderTask instead of the normal text stream in text_layer_builder.js

Acceptance Tests

Issue 1:

Test 1:

1. Open pdf.js and view the default pdf file available.
2. Use CTRL+F to open up the “Find in document” menu.
3. Enter the word “provide” in the search bar and hit the “Enter” key.

4. There should be a series of search results with context displayed in the UI (five results)
5. Click the 4th result.

Test succeeds if the user is brought down to the 4th search result in the general document.

Test 2:

1. Open pdf.js and view the default pdf file available.
2. Use CTRL+F to open up the “Find in document” menu.
3. Enter the word “provide” in the search bar and hit the “Enter” key.
4. There should be a series of search results with context displayed in the UI (five results).
5. Click the 3rd result.
6. Use the next key to navigate to the 4th result in the document.

Test succeeds if the user is brought down to the 4th search result in the general document.

Issue 2:

Test 1:

1. Open pdf.js and view the default pdf file
2. Use CTRL+F to find the second occurence of “identifies”
3. Highlight “identifies *hot* (frequently” and copy
4. Paste into a text document

Test succeeds if it pastes “identifies *hot* (frequently” or “identifies hot (frequently”. Test fails if it pastes “identifies*hot*(frequently” or “identifieshot(frequently”.

Test 2:

1. Open pdf.js and view the file https://github.com/mozilla/pdf.js/files/2955999/1.pdf
2. Highlight and copy the text “Open Sans is a humanist sans” under the “Introduction header”
3. Paste into a text document

Test succeeds if it pastes “Open Sans is a humanist sans”, test fails if it pastes “OpenSansisahumanistsans”.