Forked Repository with changes:
https://github.com/iansnt/pdf.js

List of Issues

Issue 1: Set keyboard focus initially and in some user actions #9558

- Link: https://github.com/mozilla/pdf.js/issues/9558
- Issue description: An issue with the TabIndex on elements in the DOM. Screen reader users do not reach the selected and activated document locations by inner-document link activation or by opening pop-ups. So we need to remove the tab indexes and set focus once the document is loaded.
- Issue location: https://github.com/mozilla/pdf.js/blob/master/web/viewer.html
  Set the tabindex fixes here
  https://github.com/mozilla/pdf.js/blob/master/src/display/
  In the display folder, we need to find where the promise for the document file is being done and add in an extra line to tell it to set keyboard focus on the right element.
- Estimated amount of work: Around 1.5 hours to find the promise and make the changes required, then about 30 minutes to test.

Issue 2: Tab Order #9557

- Link: https://github.com/mozilla/pdf.js/issues/9557
- Issue description: When displayed on the DOM, the HTML elements have TabIndex values set to positive values. This is messing up navigation with the keyboard, as there are inconsistencies in how the keyboard focus moves around elements. Instead, we want to assign ARIA landmark roles to the elements to allow for better navigation with screen readers.
- Issue location: https://github.com/mozilla/pdf.js/blob/master/web/viewer.html
  Can see the tabindex values being set on elements like “body”, change those to have appropriate roles (“main” for parent elements of document content, “navigation” for nav content).
- Estimated amount of work: Not long (around 1 hour) to implement and maybe another hour for testing whether the fix worked

Issue 3: Set focus into the “Document Properties” dialogue #9560

- Link: https://github.com/mozilla/pdf.js/issues/9560
- Issue description: If the user is using keyboard controls and opens the document properties dialogue, the focus is not brought to the new pop up which makes it difficult to interact with (visually impaired users may find this unintuitive since the new information they just opened is not the next object they can tab to).
- Issue location:
  https://github.com/mozilla/pdf.js/blob/master/web/pdf_document_properties.js
  We can alter the open function in this file to focus the document properties as it opens as well as send a message to screen readers as it opens.
- Estimated amount of work: 1-2 hours of implementation with another hour or so for documentation and test case development.

Issue 4: Avoid prompting print dialogue on load of pdf viewer #11259

- Link: https://github.com/mozilla/pdf.js/issues/11259
- Issue description:
  If a pdf includes “print” anywhere in embedded javascript of a pdf it will immediately attempt to print the pdf when it loads, the requested feature is to include an option to prevent this from occurring..
- Issue location:
  https://github.com/mozilla/pdf.js/blob/master/web/app.js
  https://github.com/mozilla/pdf.js/blob/master/web/app_options.js
  Can add an option parameter to app_options.js to prevent running lines 1082-1089 of app.js to prevent the automatic print dialogue.
- Estimated amount of work: very little (< 1 hour) of work with another hour or so of documentation/testing.

Issue 5: Problem opening filename with hash (#) #10632

- Link: https://github.com/mozilla/pdf.js/issues/10632
- Issue description:
  PDF viewer fails to open PDF files that contain any symbols to be encoded in the title of the file. Possibly caused by the viewer attempting to interpret the characters in the title of the file as a fragment identifier of a URL.
- Issue location:
  https://github.com/mozilla/pdf.js/blob/master/web/viewer.js
  https://github.com/mozilla/pdf.js/blob/master/web/viewer.html
- Estimated amount of work:
  Under an hour to implement some code to make sure the URL is encoded, with up to another hour dedicated to testing and documentation

Chosen Bugs

Issue 2: Tab Order #9557

- Link: https://github.com/mozilla/pdf.js/issues/9557

Reasoning:
This problem involves changing the HTML code, not all of us have experience working on JavaScript projects, so figuring out how everything works will be too much in the short time we have. So working on something that is in HTML is the best idea since we all know how it works. This fix is also very approachable for first time contributors since you do not need to know how the entire project works. Finally, this fix seems very useful for people who have difficulty seeing, as it allows them to better read the file with file reader.

The bug should only take an hour or so at most, but there is a potential risk when it comes to regression and testing as we might have to consider everything else that is affected by this change (if there is any). We may even change the ordering of tabs in total.

Issue 3: Set focus into the “Document Properties” dialogue #9560

- Link: https://github.com/mozilla/pdf.js/issues/9560

Reasoning:
The issue of setting focus into the “Document Properties” dialogue is very important for the user experience of the visually impaired. Implementing a fix for this issue also allowed us to familiarize ourselves with how pop-ups are rendered using the overlayManager in pdf.js which we can use in our future feature. Considering the issue was also in javascript it also helped us use our past experience to better implement the fix.

We estimate this bug to take us roughly an hour of research into the code structure and development, with low risk involved considering there is not much changing to overall structure or core.

Documentation and Test Cases

Issue 2: Tab Order #9557:
Of the two problems brought up in the issues (tab order and missing aria landmarks) the tab order inconsistencies were not reproducible. While it is generally recommended to avoid using positive tabindex values to avoid inconsistent tabbing at this point removing them would require rearranging most of the elements in the html to match the current order.

Aria landmarks main and navigation were added to relevant html elements, main for the container that holds the pdf pages and navigation for the containers that hole page thumbnails and the container holding the next/previous page buttons.

Test 1:
1. Enable screen reading software (Used: Windows Narrator)
2. Open a PDF file in pdf.js
3. Use screen reading software to jump “main” ARIA landmark
4. Use screen reading software to begin reading from

Test passes if the used screen reading software begins reading from the start of the PDF file

Test 2:
1. Enable screen reading software (Used: Windows Narrator)
2. Open a PDF file in pdf.js
3. Navigate to the first page in the PDF file
4. Use screen reading software to jump “navigation” ARIA landmark

Test passes if the used screen reading software selects the “Next Page” button in the page navigation menu

Issue 3: Set focus into the “Document Properties” dialogue #9560:
This issue was fixed by calling the focus() method to focus on the new dialog as soon as the document properties button is clicked and the overlay is opened. As seen in web/pdf_document_properties.js it was added as the promise is resolved in the open function.

Issue is resolved if after opening the document properties, webpage is now focused onto the “close” button of the document properties dialogue.

Test 1:

1. Open pdf.js
2. Using the keyboard, use the tab key to navigate to the rightmost button on the navigation bar
3. Press the enter key on the keyboard
4. Use the tab key to navigate down to the “Document Properties” button in that dropdown menu
5. Press the enter key on the keyboard

Test passes if the web page is now focused on the close button of the document properties pop-up dialogue.

Test 2:

1. Open pdf.js
2. Use the mouse to navigate to the rightmost button on the navigation bar
3. Click on document properties at the bottom of the dropdown menu.

Test passes if the web page is now focused on the close button of the document properties pop-up dialogue.

Technical Commentary

Issue 2: Tab Order #9557:
The fix for this issue didn't affect the architecture or design of the code of the project. It only added a few tags to pre-existing elements in the html to allow screen readers to jump to specific landmarks.

Added role tags (and aria-labels for navigation roles) to div elements in lines 82, 216, 304 of https://github.com/iansnt/pdf.js/tree/aria-labels/web/viewer.html

Issue 3: Set focus into the "Document Properties" dialogue #9560:
In implementing a fix for this issue, we weren't required to change any of the architecture, having a minimal effect overall on the design and code of the project. We simply used the javascript focus() method to focus on the desired element on the web page as outlined in the issue.

Added line 118 to focus on the new dialogue in https://github.com/iansnt/pdf.js/tree/documentProperties/web/pdf_document_properties.js

Usage of Kanbanized Agile:

Kanbanized Agile was used to moderate the development process of deliverable 2. Over the course of development, various cards were added to the Kanban to for different parts of the development process. During the first section of the process, finding possible issues on the pdf.js GitHub, a new card was created and added to the "Available" column of the Kanban for each issue the team decided to document. Each team member would assign one or two of these cards to themself and document the issue, moving the card from "Available" to "Working" to "Pending Review" until the documentation of the issue was deemed complete by the team and moved to the "Done" column. At the same time, a card was created and assigned to the entire team to decide which two issues would be implemented. After deciding on these two issues, two new cards were added to the board for these issues, and team members chose which issue they wanted to work on, with a limit of two members to each issue. Cards were also created for creating the documentation of the tests done for each issue implemented. Over the course of the implementation of the issue fixes, all team members worked on the documentation of these fixes. All members were assigned to a card for documenting implementation, which, once all fixes were complete and fully documented, was reviewed by the team and moved to the "done" column.

The usage of a Kanbanized Agile approach was very effective during the development process of deliverable 2. On account of the teams volatile schedules, having short daily standups online, and using a Kanban to document progress allowed the team to manage time spent on the deliverable when time was available to work. The individuality offered by the Kanban approach was also quite beneficial to development, giving each team member the ability to do work that was best suited to them, particularly when deciding which issue each team member would work on. The Kanban itself was also quite useful when communicating with other team members during online standups, as it allowed the team to easier visualize the flow of work on the deliverable. We were also able to avoid conflicts in the work assigned because everyone had an equal level of leadership, so people were not forced into anything they did not like.