

Deliverable 2

Team 29: GoodbyeWorld!

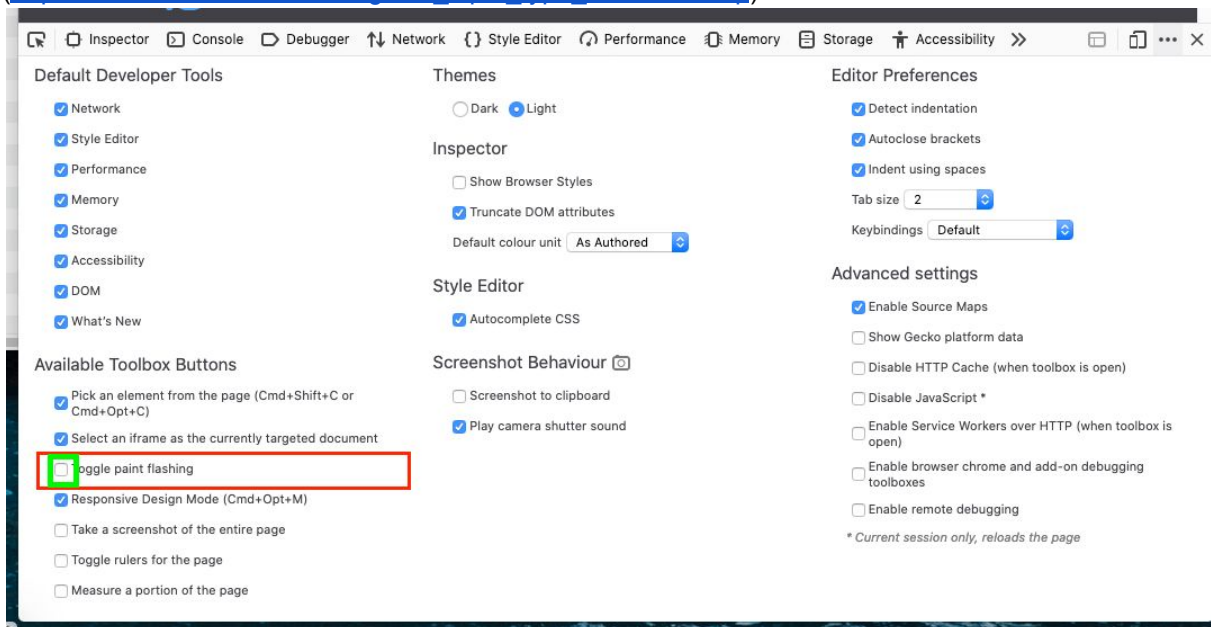
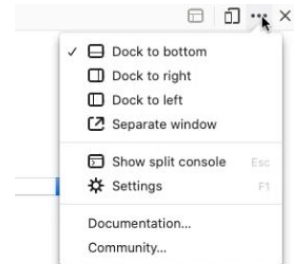
Thasitathan Sivakumaran | Ajay Rajendran | Jason Ku | Yathan Vidyananthan

Choosing 5 Bugs To Further Examine

Bug 1: https://bugzilla.mozilla.org/show_bug.cgi?id=1531370

Name: devTools settings - listed items expand over the full available width

Detailed Description: This bug is a problem with the settings menu (f1) for developer tools, you can get to this by clicking the three dotted arrow on the right side, and clicking on settings (image on the right). For all of the possible checkboxes below, you are able to click anywhere on its row, an example of this is showing for 'Toggle paint flashing', you can click anywhere in the red area and the checkbox will toggle, but you should actually only be able to click the actual checkbox highlighted in green. This problem exists with the entire settings page for all the possible checkboxes on this page, where input type="checkbox" (https://www.w3schools.com/tags/att_input_type_checkbox.asp).



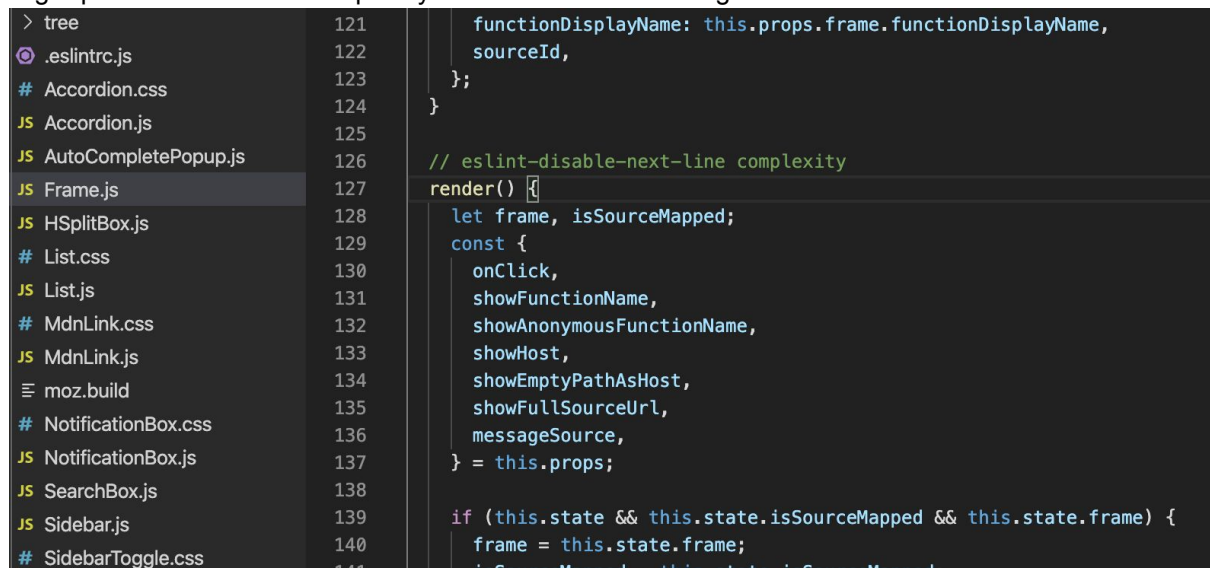
Estimated time of Work: The estimated time of developer work, would be about 2 hours to figure out the where the code is affected and why. Then to fix the generated elements would take about 4 hours, and to fix the hardcoded html elements would be about 2 hours. There is also work with our test driven development/acceptance tests which will take about 2 hours. The total effort for implementation and testing would be about 10 hours plus work for requirement definition 1 hour, component analysis 1hour, system design 1 hour, and system validation 1 hour. With the total effort adding up to 14 hours of work.

Where the bugs in the code are: The bug in the code after further inspection we found out this problem is under the client/framework folder under the toolbox options files. This was one of the bugs we tackled and found out that there are two sets of work being done on this page. The leftmost row, Default Develop[er] Tools and Available Toolbox buttons are generated elements, meaning that each element is generated when the page is loaded from other sources in the code. While the elements in the middle and farther right row, are all hard coded elements in an html file. So here we have to investigate why the problem occurs and what the fix will be.

Bug 2: https://bugzilla.mozilla.org/show_bug.cgi?id=1554887

Name: Simplify devtools/client/shared/components/Frame.js and remove eslint-disable complexity

Detailed Description: This bug is based on code that is failing one of the requirements that was set on eslint. Eslint is a style and formatting checker for javascript that the user can format to add rules or formatting parameters that all programmers have to follow. Once the project is set up with eslint, it will provide feedback on all the parameters that the programmer may fail to meet. One of the functions within the specified file failed to meet the requirement, but since they either did not have time or resources to amend the issue, they added a comment which made eslint ignore that function thus allowing the format error to persist. The error in this case was complexity. Eslint only allows a maximum of 20 levels of complexity (complexity refers to all the different paths a program can take, such as switch statements or if else statements) however the function render() had a level of 29. The bug report asks to fix the complexity issue so that the eslint ignore comment can be removed.



```
> tree 121
.eslintrc.js 122
# Accordion.css 123
JS Accordion.js 124
JS AutoCompletePopup.js 125
JS Frame.js 126
JS HSplitBox.js 127
# List.css 128
JS List.js 129
# MdnLink.css 130
JS MdnLink.js 131
# Moz.build 132
# NotificationBox.css 133
JS NotificationBox.js 134
JS SearchBox.js 135
JS Sidebar.js 136
# SidebarToggle.css 137

functionDisplayName: this.props.frame.functionDisplayName,
sourceId,
};
}

// eslint-disable-next-line complexity
render() {
  let frame, isSourceMapped;
  const {
    onClick,
    showFunctionName,
    showAnonymousFunctionName,
    showHost,
    showEmptyPathAsHost,
    showFullSourceUrl,
    messageSource,
  } = this.props;

  if (this.state && this.state.isSourceMapped && this.state.frame) {
    frame = this.state.frame;
    isSourceMapped = this.state.isSourceMapped;
```

Estimated time of Work: The biggest component of this project will be understanding the interconnections of the components of frame and everything it interacts with. Since we are changing up established code, we need to understand what we can and cannot alter so that it does not affect other parts. This process will most likely take about 3 hours to go through the code and the tests. Next we need to understand how eslint is implemented. Since each directory has their own config file and they are all stacked on top of each other, plus the base initial config file, understanding that will probably take us around 2 hours. Then creating a design for how we are going to implement the fix and the tests will take about 2 hours. Then we need to create tests for the changes we are about to make and for eslint, which is different from the existing tests, as we have to create an automated process, so that will take about 4 hours. Implementing the changes to fix the bug will take about 5 hours, and then creating and running the acceptance test will be about 1 hour. In total, this bug will take around 17 hours to fully implement and validate.

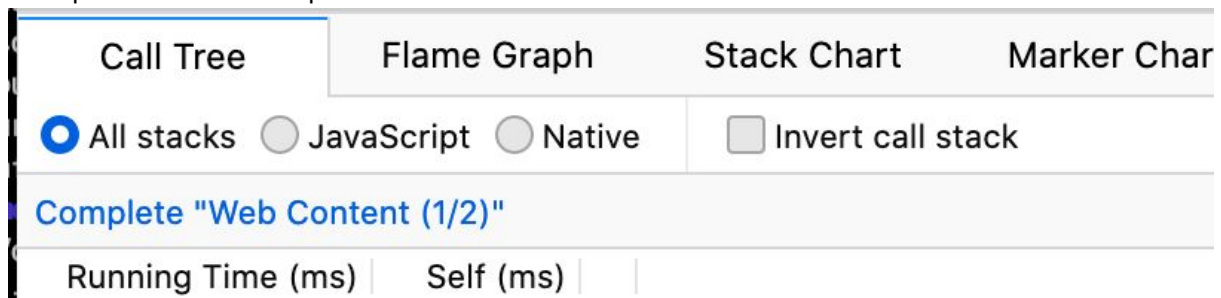
Where the bugs in the code are: The code for this bug is in the file that they specified: devtools/client/shared/components/Frame.js. Specifically, it is in the render() function of the file, under the comment regarding eslint-ignore. We should have removed the comment after the fix line 126 in the above screenshot will be removed, and eslint tests should pass, and there should be no errors after the code refactoring.

Bug 3: <https://github.com/firefox-devtools/profiler/issues/2088>

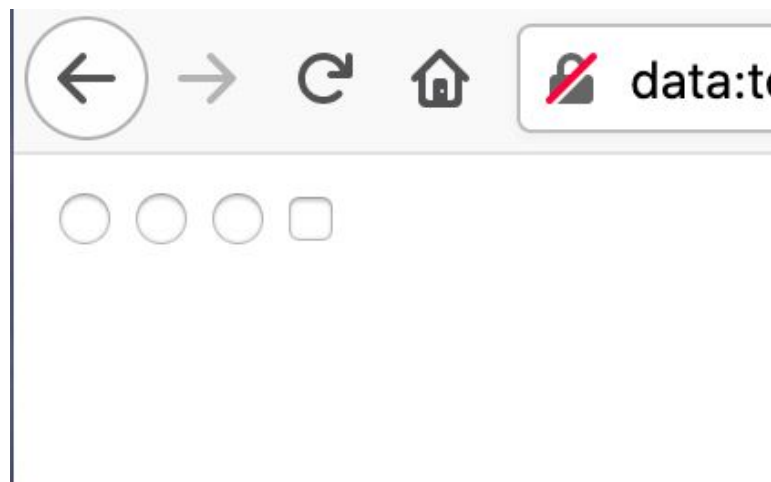
Name: Radio buttons for callstacks (implementation filters) look like they are grayed out and disabled

Detailed Description: This bug is related to the UI of the DevTools. The radio buttons are grey and it makes it look like they are disabled. This is misleading to users and could result in users not using the features and settings at all since they assume they are deactivated/not available to use. An example of the bug is shown below where the “JavaScript”, “Native”, and “Invert call stack” radios are all grey, making it look like they are unavailable to be used. These are not the only affected radios, as there are several others in other locations within DevTools.

Example of the current implementation:



How the radios should look:



Estimated time of Work: The most work here would be to locate the code first. This may surprisingly take longer than usual as the DevTools of Mozilla do not allow for the use of the browser’s “Inspect Element” feature. This means that we will not know which file renders the radios by simply using the browser and checking. Instead, it would be necessary to look through the code base and understand the process as well as how the code is being rendered. Once we locate the related lines and files, we can then tackle the problem. The problem seems to be a css one, so it should not take too long once it is found. Finding all the instances of radio buttons and understanding what it affects will probably take 2 hours. Fixing should be easy, and should be done in 1 hour. Verification can be done by running existing so that will be under 1 hour as well. In total, it would take around 4 hours in total.

Where the bugs in the code are: It seems like the code is fine itself within DevTools but requires some coordination with “photon”, another area of Mozilla. Since DevTools follows Photon guidelines, it would not be possible to modify a lot of code without first consulting them and seeing if they have any fixes themselves. Thus, there are no bugs in the DevTools’ related code for this bug.

Bug 4: https://bugzilla.mozilla.org/show_bug.cgi?id=1620932

Name: Fix React warning in devtools/client/responsive/components/Browser.js allowFullScreen: "true"

Detailed Description: When building the browser code, the React part of the build specifies a warning. This warning is not critical and does not impact the code itself but should be addressed in order to maintain clean code. The browser still allows for rendering in full screen with this warning, although when allowFullScreen flag is switched to "false", unexpected behaviour may occur. However, the "false" flag is usually not used at all. Still, it is important to address this issue even if it does not break things in most cases, for both usability in the future as well as the rare cases that "false" is used instead of "true"

Screenshot of the bug:

```
console.error: "Warning: Received the string `s` for the boolean attribute `s`. %s
Did you mean %s={s}?%s" "true" "allowFullScreen"
"Although this works, it will not work as expected if you pass the string `false`."
"allowFullScreen" "true"
in iframe (created by Browser)
in Browser (created by ResizableViewport)
in div (created by ResizableViewport)
in div (created by ResizableViewport)
in div (created by ResizableViewport)
in ResizableViewport (created by Viewports)
in div (created by Viewports)
in div (created by Viewports)
in Viewports (created by Connect(Viewports))
in Connect(Viewports) (created by App)
in div (created by App)
in App (created by Connect(App))
in Connect(App)
in Provider"
```

Estimated time of Work: The amount of work should not be too much as it is a simple type error. Instead of having a string "true" and "false", change it to the boolean type: true and false. This will fix the error and in both cases when true or false are used, there will be no warnings or errors. This will take around 0.5 - 1 hour to change the code.

Where the bugs in the code are: The bug in the code is located under: devtools/client/responsive/components/Browser.js. There is a suspected line of code where the allowFullScreen option is set to True as a string, rather than a boolean. This can be seen in the below screenshot in the Browser.js file.

Suspected line:

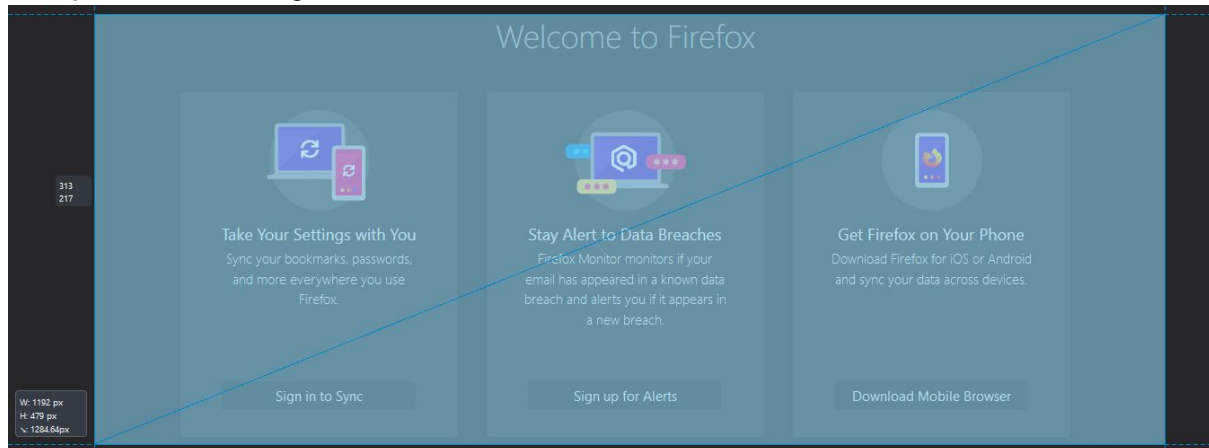
```
allowFullScreen: "true",
```

Bug 5: https://bugzilla.mozilla.org/show_bug.cgi?id=1262782

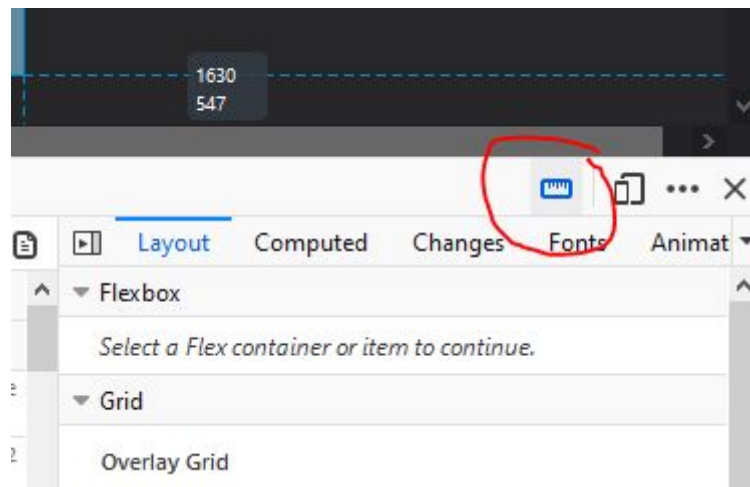
Name: Allow resizing the area selected by measuring tool via keyboard

Detailed Description: This bug is to address the resizing of the selected area when using the measuring tool. It should allow the user to change the size of the selected area using the keyboard and not only limiting them to use the mouse. Additionally, the developers would prefer if there were some sort of “modifier” or “multiplier” when changing the size of the area using the keyboard. They prefer if a key such as “CTRL” was used when using the keyboard to change size to make the area be enlarged or smaller faster.

Example of the tool being used:



Measuring tool:



Estimated time of Work: This bug fix will involve a few things. Analyzing the code and understanding all the factors will take around 4 to 5 hours since this is a major bug. Creating the initial tests will take around 3 hours. For implementation, first, a listener should be added to listen for the input of the keyboard when the measuring tool is being used. Then, the event listener should also be able to process the keys being pressed and manipulate the size of the selected area based on how much the keyboard is pressed. This implementation would probably take around 6 hours. Validation and verification will take another 2 hours. In total, this will take around 16 hours to fully implement.

Where the bugs in the code are: The associated file is `devtools/server/actors/highlighters/measuring-tool.js`. This file is where the listener should be added as well as the functionality of the keys expanding the selection area. Additionally, if the multiplier feature is to be implemented, this is where the code for that functionality should reside as well.

Our 2 Selected Bugs

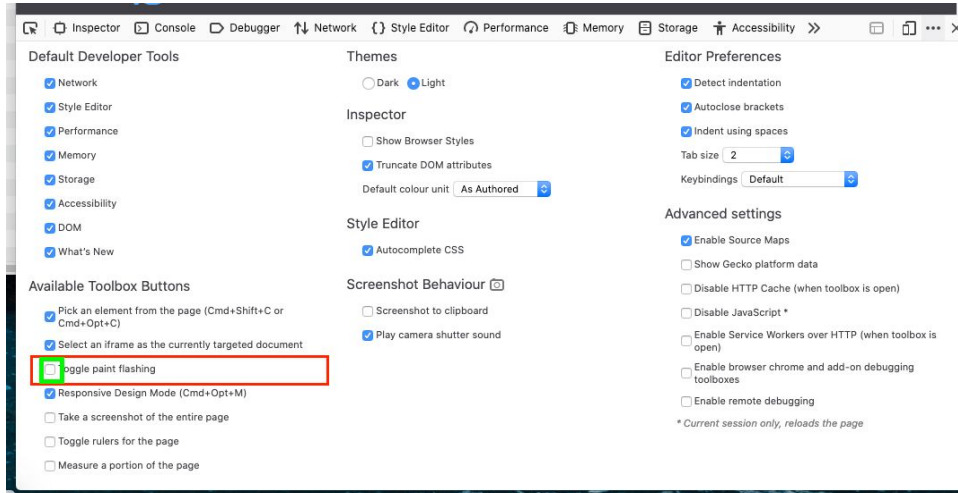
Explain why they are the best items: Bug 1 was associated with fixing the devTools settings panel, there was a bug with the checkboxes, which allowed you to click the entire row of the checkbox instead of the actual box itself. This bug from our original idea we thought would have some CSS, HTML, and JS changes which was interesting, as it would also allow us to work on multiple different aspects of the code base. We felt bug 1 was a good item to work on as our first bug because it was working with the visuals in the firefox devtools. So our change would be noticeable and something that we visually see after we fix it. Bug 2 we worked on reducing the complexity of one of the files in the project so that it will pass the ESLint check. Currently, the file just omits the ESLint complexity check on the render function. Bug 2 gives us a good understanding of the code structure of the front end of devtools. We are manipulating the render code of a front-end component, which means that it interacts with other components as well. Working on this bug gives us a good idea of how the code for the front end is interconnected, and gives us a good understanding of the structure and layout of the code as well. We are also working on refactoring the code, which allows us to practice working on established code and implementing better fixes and improvements even if we are not adding any new feature. This helps us exercise our skills as well as help us understand how our work impacts other parts of the project.

Estimated effort required to implement each change: For bug 1; the estimated time of developer work would be about 2 hours to figure out where the code is affected and why. Then to fix the generated elements would take about 4 hours, and to fix the hardcoded html elements would be about 2 hours. There is also work with our test driven development/acceptance tests which will take about 2 hours. The total effort for implementation and testing would be about 10 hours plus work for requirement definition (1 hour), component analysis (1 hour), system design (1 hour), and system validation (1 hour). With the total effort adding up to 14 hours of work. For bug 2; The biggest component of this project will be understanding the interconnections of the components of frame and everything it interacts with. Since we are changing up established code, we need to understand what we can and cannot alter so that it does not affect other parts. This process will most likely take about 3 hours to go through the code and the tests. Next we need to understand how eslint is implemented. Since each directory has their own config file and they are all stacked on top of each other, plus the base initial config file, understanding that will probably take us around 2 hours. Then creating a design for how we are going to implement the fixes and the tests will take about 2 hours. Then we need to create tests for the changes we are about to make and for eslint, which is different from the existing tests, as we have to create an automated process, so that will take about 4 hours. Implementing the changes to fix the bug will take about 5 hours, and then creating and running the acceptance test will be about 1 hour. In total, this bug will take around 17 hours to fully implement and validate.

Identify anticipated risks: For bug 1; some of the risks may include breaking the settings panel of the browser with these changes. We might also affect the UI design by changing the elements or we might introduce new bugs into the code. For bug 2; some of the risks might be to break the frame component itself, as it is a big refactoring effort. This may also break some of the other components which are interconnected with it. A lot of tests will fail if we break the eslint check. We may also introduce more eslint errors if the fix is not applied properly.

Brief Technical Documentation(Explanation/Source Files)

Bug 1 was changing the type of element used from an html 'label' element to a 'div' element. This had to be done in two different places. One is the static html file which is under devtools/client/framework/toolbox-options.html. This is the static parts of the settings panel, which is the columns in the middle and the right side (as shown in the diagram below). The second file is the generated elements which is under devtools/client/framework/toolbox-options.js which generates the left column (as shown in the diagram below). These elements are generated when the page is rendered dependent on some other information in the code. With these two files, we will fix the problems with all the checkboxes as shown below. We also followed a test driven approach and created acceptance tests under devtools/test/bug-1531370-acceptance-tests.md

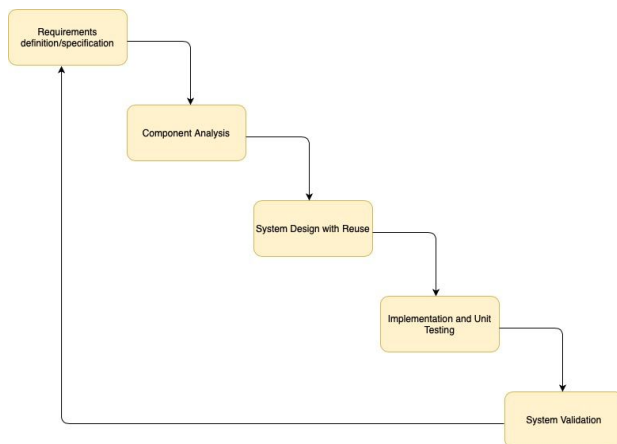


Bug 2 was mainly refactoring, meaning we not only altered the code so that it passes the eslint check, but also made it more readable since the render function is not as big, and components of the code are now in helper functions with better names so people can understand the structure better. We made three big changes in the code. We took three components within the render function which handled assigning certain components and we moved them out into their own method. We gave them a descriptive name and called them in the render functions. This allows the code to be less complex and readable. These changes were done in devtools/client/shared/components/Frame.js and we also added test_eslint.sh in the gecko_dev directory which is a test file to verify eslint and validate the changes. We also added a directory in gecko_dev called test_results which contains a file where the output of the tests are. The sh script reads that file and tells the user if the code changes passed or failed all the checks including an eslint check. So as shown below the comment in line 126 that turns off the eslint complexity check will be removed in the fix as we have refactored the function to pass the eslint check without ignoring it.

```
121     functionDisplayName: this.props.frame.functionDisplayName,
122     sourceId,
123   };
124 }
125
126 // eslint-disable-next-line complexity
127 render() {
128   let frame, isSourceMapped;
129   const {
130     onClick,
131     showFunctionName,
132     showAnonymousFunctionName,
133     showHost,
134     showEmptyPathAsHost,
135     showFullSourceUrl,
136     messageSource,
137   } = this.props;
138
139   if (this.state && this.state.isSourceMapped && this.state.frame) {
140     frame = this.state.frame;
141     isSourceMapped = this.state.isSourceMapped;
```

```
167   frameAndSourceMapped(state) {
168     if (state && state.isSourceMapped && state.frame) {
169       return true;
170     }
171     return false;
172   }
173
174   render() {
175     let frame, isSourceMapped;
176     const { onClick, showHost, showFullSourceUrl, messageSource
177
178     const frameMapping = this.frameAndSourceMapped(this.state);
179     if (frameMapping) {
180       frame = this.state.frame;
181       isSourceMapped = this.state.isSourceMapped;
182     } else {
183       frame = this.props.frame;
184     }
185   }
```

Software Development Process



The software development process we followed was what we had planned for in deliverable 1 which was a combination of a waterfall and reuse oriented plan. We were going to repeat this process for each bug that we decided to fix.

Bug 1:

Requirements/definition specification: this was done by reading the bug documentation on the firefox website (https://bugzilla.mozilla.org/show_bug.cgi?id=1531370), and understanding the code, we also did this step by analyzing the bug and writing what we understood as required which included a detailed description, estimated time of work, and where the bugs are in the code.

Component Analysis: In this step we understood how the components worked, this included understanding the settings panel, and the checkboxes, and how the error was being caused. We recreated the error here in a mozilla browser and noticed that it was still failing. We looked under devtools/client/framework and understood the code that was being affected. This included understanding files such as toolbox-options.js, toolbox-options.html.

System Design with Reuse: Since we were fixing a bug there was not much planning associated however we did decide to use the existing component. We planned out how we will try fixing the bug by making changes to the existing component as the bug was a visual error. We realized that it was an issue with the html tags because they were incorrectly utilizing label tags. We decided to change the wrapper from a label tag to a div tag which should format the checkboxes correctly.

Implementation and Unit Testing: We assigned this bug 2 two people Jason/Yathan to work on, and then followed a pair programming approach. We created a branch for the bug called bug-1531370. We followed a test driven development approach where we created a set of manual test cases that would initial fail and would later pass after the implementation is complete (this commit and tdd can be found

here:<https://github.com/thasitathan/gecko-dev/commit/6394710dca74d8690492083deae5b5625ed9782b>). We then proceeded to modify the existing code in order to fix the problem with the wrappers. All the commits for this can be found here:

<https://github.com/thasitathan/gecko-dev/commits/bug-1531370>.

System Validation: We then run the test cases we initially produced to ensure that our changes fixed the problem and to verify that nothing else was negatively affected. This can be seen in this commit: <https://github.com/thasitathan/gecko-dev/commit/344650f21d2db51d9e8d96f90af9c1942e52dc8f> After verifying the changes we then created a pull request to merge our branch with the master branch. We assigned the pull request to our other two group members, Thasi and Ajay, so that they can verify the new changes going into master. This can be seen here:

<https://github.com/thasitathan/gecko-dev/commit/bb20c304db04c56189791277eb2623907436144f>.

Bug 2:

Requirements/definition specification: this was done by reading the bug documentation on the firefox website (https://bugzilla.mozilla.org/show_bug.cgi?id=1554887), and understanding the code, we also did this step by analyzing the bug and writing what we understood as required which included a detailed description, estimated time of work, and where the bugs are in the code.

Component Analysis: In this step we understood how the components worked, this included understanding the frame.js component and how the error was being caused. We removed the comment above the render function that voided the ESLint complexity check so that we can see details on the error. The issue was that the render functions complexity was 29 which was above ESLints function complexity threshold of 20.

System Design with Reuse: Since we were fixing a bug there was not much planning associated however we did decide to use the existing component. We planned how we will refactor the render function so that we maintain its functionality while reducing its function complexity. We found 3 sections in the render function that could be moved out into their own methods which would drop the render functions complexity to 20.

Implementation and Unit Testing: We assigned this bug 2 two people Thasi/Ajay to work on, and then followed a pair programming approach. We created a branch for the bug called bug-1554887. We followed a test driven development approach where we created a bash script that would test ESLint and also test the frames functionality (this commit and tdd can be found here: <https://github.com/thasitathan/gecko-dev/commit/9f66fa94ed26029fb389e869793ad5b74e262312>

). This test script was initially failing because our render function complexity was above 20. We also had the test script run tests to test the frames functionality to ensure that we do not break the frame. We found these existing tests when we were analyzing the component in the component analysis stage. We then proceeded to modify the existing code in order to fix the problem with the eslint failing. All the commits for this can be found here: <https://github.com/thasitathan/gecko-dev/commits/bug-1554887>.

System Validation: We then run the test cases we initially produced to ensure that our changes fixed the problem and to verify that nothing else was negatively affected. This can be seen in this commit: <https://github.com/thasitathan/gecko-dev/commit/827119120e9d750a06c3add234d0150cac11694c>. After verifying the changes we then created a pull request to merge our branch with the master branch. We assigned the pull request to our other two group members, Yathan and Jason, so that they can verify the new changes going into master. This can be seen here: <https://github.com/thasitathan/gecko-dev/commit/7d6f741526aca11be083a5d01039439af8819ef4>.

We tracked all our work using trello:

