Assignment: Issue resolution (refactoring)

Anneli Bogren, Carl Wang, Rikard Johansson, Paul Tissot-Daguette

March 4, 2024

Abstract

The goal of this project to realize the complexity of issue resolution in a real project.

1 Part 0: Project choice

To conduct this assignment, we chose to work with a project called Chart.js, an open-source javascript chart library. We identified a few issues that didn't have any assignees yet,

2 Overview of issues and work done

2.1 Part 1: Setup

Project setup and onboarding documentation

The project has a webpage dedicated to the onboarding of new contributors, describing the guidelines and steps.

Did you continue with the previous project or choose a new one?

We chose a new one and thought it would be interesting to work in a new language, as we have previously worked in Java during our past assignments.

If you chose a new project: How does the experience compare to the previous one?

The build and installation worked well, the only issues were with testing as the Firefox tests did not pass. We reached out to the dev team on Discord about it. The Chrome tests pass however so we can still start working. It should not affect us in the task we are trying to complete.

2.2 Issue #11684

2.2.1 Project Plan

1. Read the issue
2. Identify the requirements, and the current behavior
3. Understand the provided reproducible example in depth
4. Search where this part is handled in the code
5. Search for tests covering this requirement
6. Find a possible solution
7. Make a test
8. Create a Pull Request

2.2.2 Requirements for the new feature or requirements affected by functionality being refactored

Requirement 1.1: Bar chart indexAxis consistency

The expected behavior is that a bar chart configured with `indexAxis: 'y'` should show the same behavior as `indexAxis: 'x'`. This means that the chart should display properly even when `data.labels` are not explicitly defined. The current behavior deviates from this expectation: when `indexAxis` is set to `'y'` and `data.labels` are absent, the bar chart fails to render anything. To align with user expectations and ensure consistency across different indexAxis configurations, the system should display the bar chart correctly irrespective of whether data.labels are provided or not.

2.2.3 Existing test cases

It appeared that no test was explicitly testing this precise case, as all test pass with the code as-is. There might be a test that doesn't contain good enough assertions, but since it is very easy to create a new test for this behavior that is what we did. Also, the library uses what they call image based testing, where we provide a base configuration object, and then the output is compared with a goal image.

2.2.4 Code changes

It was noticed that when setting indexAxis x there were no labels or data required to be set over the usual key value pair of value and label. However, when setting the indexAxis to y it suddenly was required. In order to fix this we initialized a method in the configuration initialization that took the key value data array and remapped it to data and labels in the dataset. This made it possible to render the graph in an expected way and treated indexAxis y required setup to render as indexAxis x would do.

2.2.5 Patch

The PR included the new config initialization method and a suitable test case in order to test the solution.

```
 if (config.options.indexAxis === 'y' && config.data.labels.length === 0) {
    config.data.labels = Object.keys(config.data.datasets[0].data);
    config.data.datasets[0].data = Object.values(config.data.datasets[0].data);
}
```

Before:

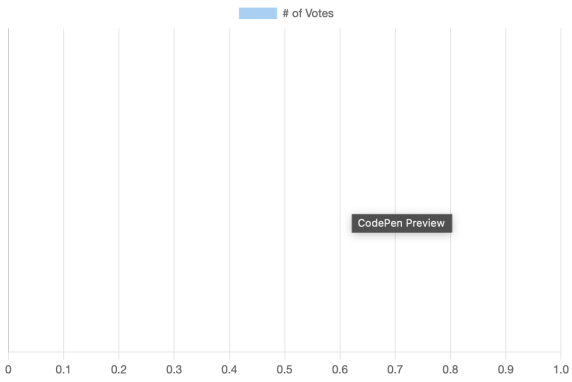

Figure 1: Before indexAxis y fix

After:

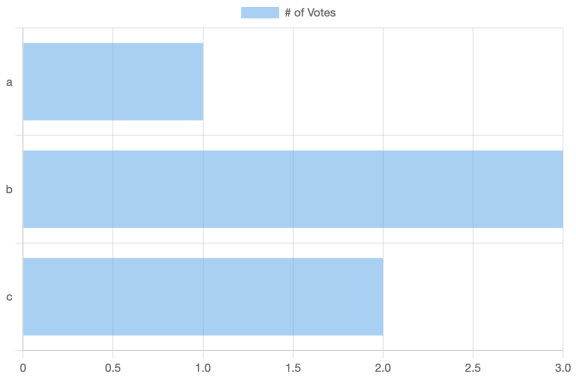

Figure 2: After indexAxis y fix

2.2.6 Test results

Tests still pass:

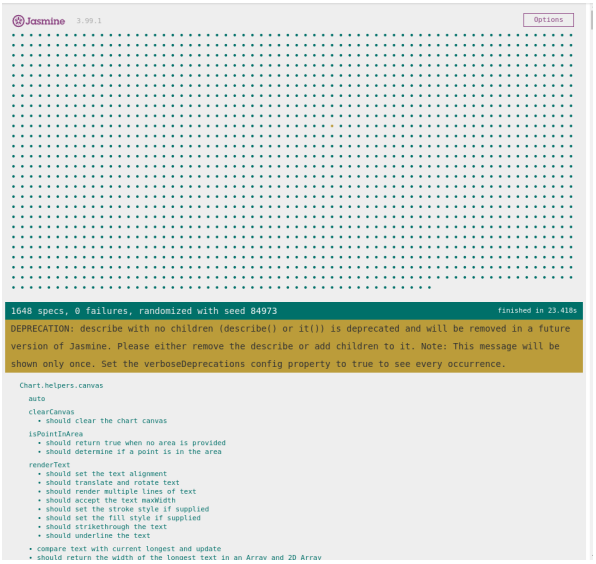

Figure 3: Run all tests

Our test:

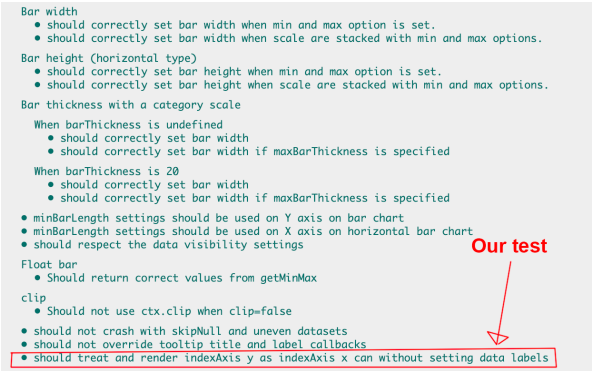

Figure 4: Enter Caption

2.2.7 UML class diagram and its description

See figure 4.

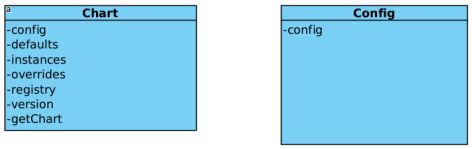

Figure 5: UML diagram

2.3 Issue #11679

2.3.1 Project Plan

1. Read the issue,
2. Identify the requirements, and the current behavior,
3. Understand the provided reproducible example in depth,
4. Search where this part is handled in the code,
5. Search for tests covering this requirement,
6. Find a possible solution,
7. Make a test,
8. Create a Pull Request

2.3.2 Requirements for the new feature or requirements affected by functionality being refactored

Requirement ID: 2.1

Title: Dataset property default value reset

Description:

The system should ensure that dataset properties, whether they are single values or arrays, reset to their default values after either deletion or setting to null. Currently, the behavior deviates from this expectation: when a dataset property, previously defined as an array, is deleted or set to null, it does not revert to its default value upon chart.update(). This behavior inconsistency needs correction to ensure consistent handling of dataset properties across different data manipulation scenarios.

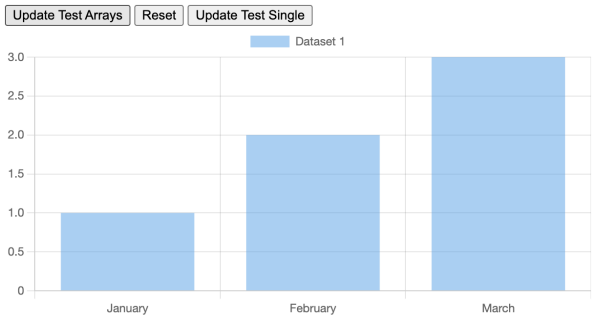

Figure 6: Reproducible sample "reset"

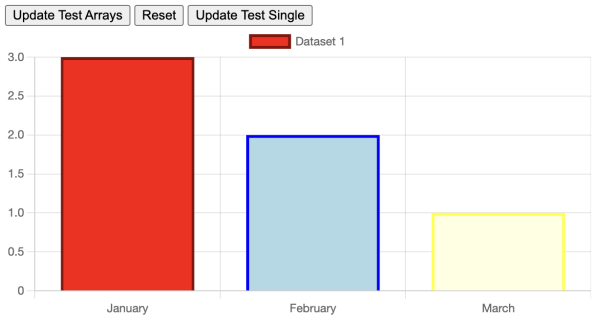

Figure 7: Reproducible sample "Update test arrays"

2.3.3 Discoveries

We put a lot of effort into understanding the project's architecture, and trying to trace down where the error was. Along the way, we made a series of observations that could potentially be useful for solving the issue. We shared them on the Github issue thread, but nobody has commented them yet. We have to mention that we found it was a difficult project to get into, as its functioning is quite complex and not very well documented. We think that if one of the main contributors would look at what we did and provide some guidance, we could probably fix the issue rather quickly but as of today, no one did.

The documentation for the end users is very good, but there very little for the contributors, so we almost had to reverse engineer the program's core. You can find below the observations we made.

Understanding the codepen example

The issue included a small codepen example, demonstrating the issue.

It shows a chart with some buttons that update the dataset with some values. With one of them, it sets some values like the background color as a simple string, and in the other one, it sets them as an array of values. We would expect that when we press the reset button, we get back to the first chart in both cases, but for some reason the color of bars in the "array" case doesn't get reset.

We noticed that hovering the bars with the mouse does reset the color correctly which is surprising, probably indicating that the state of the chart is correct, but the rendering does something unexpected, as it uses a stale state.

The code example contains a function called updateDatasets that implements updating a dataset by providing the new value for it. It will then assign all new values to existing properties, and delete properties that exist in the current dataset, but not in the new one. It does that differently depending on if the property value is a single value or an array. If it is a single value, it deletes it, and if it is an array, it sets it to null. We think that this is to simulate what the person's library is going to do.

We think that it shouldn't break the code if some attributes are set to null, default values should be used. But it could be interesting to see what the main contributors think about this, maybe it shouldn't be the case.

Looking into the hover callback

We noticed that when we hover the bars with our mouse, the colors get set back to the default color, which is what we want. So we decided to investigate what exactly is happening when we hover the bars with our mouse. We found a function in the Default class this is called hoverBackgroundColor and that gets called when we hover a bar. To understand in what context it is called, we threw an error from there to be able to see the trace. We spent quite some time trying to understand how the config object is actually updated in the project but it is still a bit unclear. We posted some comments in the issue thread with our observation. It seems that the default values are eventually set, but the render uses old values that get stuck in some sort of cache somewhere.

Resolving of options

In order to get the options values, the program has a quite complex system of resolving from a list of contexts, with decreasing priority. The highest priority would be what the user set, and the lowest is the default values. We are trying to understand how this resolving works, as it could be that a background color value is stuck somewhere and so get resolved as the actual value.

Scriptable options and needContext

Option values can be scriptable apparently, which in our understanding allows to set custom functions for computing the values of some options, which makes the library much more flexible for power users but also harder to understand for us. So the needContext function is used to determine if the context is going to be need to resolve the option value or not. For this, it determines if the option is scriptable, indexable and if it is an array. It needs context either if it is a function or contains a function or if it is indexable and is an array.

In order to learn more on how this whole mechanism works, we read the following threads on github

- https://github.com/chartjs/Chart.js/pull/8260
- https://github.com/chartjs/Chart.js/pull/8008
- https://github.com/chartjs/Chart.js/pull/8374

We found that, for arrays, by handling the props that don't exist in the new dataset but do exist in the existing dataset the same way as for single values (by deleting them) and then calling on chart.update('none) in the updateDatasets function, the bar chart resets but not in the same way as for the single value reset. Instead of smoothly transitioning to the default, it is a choppy and quick switch, showing that the issue is on a deeper level than this quick fix. Furthermore, when testing this issue on other chart types, such as line, we found that the same issue persists.

controller.bar draw function

It is in this function that the chart bars are actually drawn. So if we print the value of their options before drawing them, we see the wrong colors printed in the console. One thing I noticed is that when updating with simple values, when the bars are drawn the $shared attribute is set to true, whereas with the arrays it is set to false. This is probably because with the arrays the bars don't have the same color so they cannot share the same option object.

We then looked into the updateElements function, adding a print in the if (includeOptions) clause, putting the bars options in the console. Here we can see something interesting, when we do update array, they get printed meaning includeOptions is true, but when we reset, it is not so I think that is why the correct color isn't set back.

We can also see here how the options are chosen. If there is a sharedOption it is taken, otherwise it tries to resolve it, and lastly it uses the individual bars options.

Looking into how includeOptions is derived

The includeOptions is either gotten using this.includeOptions(mode, sharedOptions), or if (sharedOptions !== previouslySharedOptions). When we reset the arrays, the value of includeOptions is false, and I think it should be true. In the includeOptions method, we can see the following: So if either

sharedOptions is false, mode is set to direct or the animation are disabled, then the options object will be included in the updated properties. Observation: If we set the includeOptions to true at all times, then it works for the arrays, but the single value becomes buggy, you have to update twice for it to become red.

Animation

One final observation we made is that if we manually disable the animations in the code example, then everything works as it should and the requirement is met.

2.3.4 UML class diagram and its description

See figure 7.

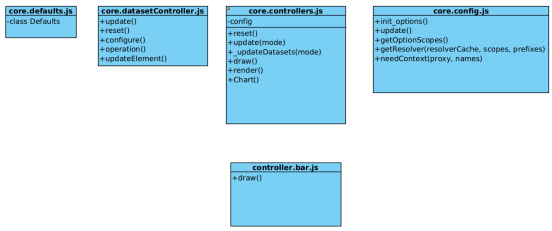

Figure 8: UML Diagram

2.3.5 Key changes/classes affected

The UML diagram in figure 7 outlines the different classes we explored in our attempt to solve the issue, as well as the classes we believe are most relevant in solving it.

2.3.6 Work plan

As we were not able to solve this issue within the allocated time, we have created a work plan. Our investigation highlights issues with option resolving, animation handling, and hover behavior that impact the reset behavior of datasets, specifically when using arrays. Further debugging and exploration of the library's source code is necessary to pinpoint and resolve these issues effectively.

We left the following message in the comment section of the issue:

"Here is a summary of what my team worked on and observed while trying to resolve this issue. The state of the chart seems to be updated correctly, but the rendering is using some outdated values. There might be a cache somewhere that keeps these values even though it shouldn't. We think the problem might be in the needContext function, as when we try to reset the chart in the example, it returns false and then it doesn't update the chart with the correct color. Moreover, when we reset the array example and hover on one of the bars, it gets updated to the correct value. So the hovering function is able to get the updated value but not the rendering one.
We looked into how the includeOptions function works, and we noticed that when updating with arrays, it return true (which seems logical since the options were updated), but when we hit reset, it returns false so it doesn't correctly update the chart to its initial state.
Lastly, we noticed that if we disable animations completely in the graph, everything works as expected. We just did chart.options.animation = false; disables all animations in each of the update function in the codepen. So there seem to be a problem with the way the animations get their option but we didn't have time to go in details of this."

2.4 Overall experience

During this project, we got to experience first-hand the challenges of issue resolution and refactoring when working on a repository that is new to all members.

2.4.1 Team Essence reflection

Comparing to assignment 1-3, that we work seprately on different things, the tasks in assignment 4 are much more indivisible. It is more difficult to ditribute the tasks, since we are working together to solve one single problem. There are many repeated works, such as understanding the related functions and documentation. However, this also allows us to learn how a real team works to solve an issue in a company.

About the project, it is much larger than assignment 1 and 2, perhaps as the same as karate. However, we did not have to understand how the functions are related to each other in Karate. When it comes to documentation, this project is much more structured, we did not need to "crack the code" like we did in assignemnt 3. One obstracle is that 2 of our team members never had experience with javascript before, they had to spend more time to get into the project. The debug phase is also affected, due to our lack of experience in debugging a javascript project.

Unlike karate, this project provides a fulfunctional tool framework for debugging, the failure unit tests can be easily traced, which hepled a lot in creating unit tests.

After this assignment, we learned how to quickly get our hands on an unfamilar project and experience of quickly getting started with new programming language. In the discussion before the presentation, we think we are still in the collaborating stage as we were in assignment 2. In order to achieve the next stage, we have to find a better way to distribute the tasks, so it could be more effient.

2.4.2 Community reflection

During this project, we reached out to the Chart.js community in 2 main ways. First, we commented on the issues we were interested in and asked if we could be assigned that issue, though we did not receive any response regarding this request. We continued with issue-related discussions on the respective issue pages and received some responses from the person who created the issue. We also joined the Chart.js discord community in hopes of being able to ask more general questions and receive help when facing failing tests. However, this did not seem to be the case, as we did not receive any responses on almost all of our messages. When we reported the Firefox test issue we faced in the very beginning, we received one response with a suggestion that ended up not working. That being said, there seemed to be others receiving proper help with smaller, user-related issues, which was nice.

3 Effort Spent

Anneli:

- **1h**: Reading project requirements and looking into possible repos to work on
- **2h30**: Setup meeting, learning the basic tools and commands used in the project, reading documentation
- **30m**: Configuration and setup
- **3h10**: Looked for a suitable issue, started working on issue #11684 with the entire group, Rikard made a PR
- **2h30**: Analyzing code and output for issue #11679, trying to find tests relating to the issue or similar previous issues
- **30**: Setting up a project plan for the issue, with Paul
- **1h30**: Analyzing code and output, looking into the hover function and trying to understand how it gets the correct color value, manual debugging
- **2h**: Reading documentation for Chart.js and js functions in general
- **3h30**: Analyzing code and output, looking into the reset function in core.controller js and how its use differs from the wanted outcome, manual debugging
- **5h**: Working on the report

- **Total time: 22h10**

Paul:

- **1h**: Reading the project requirements
- **1h**: Learning to use pnpm and basic commands to build and run tests
- **2h**: Reading the documentation to understand what the project does, how it works in general
- **2h**: Getting into issue #11684, trying to trace back what exactly happens when the axis is set to y. Eventually, Rikard found a fix and made a PR with his patch
- **2h**: Understanding issue #11679, creation of a local file with content of the codepen to be able to experiment, reading in detail of the codepen code
- **30m**: Collaboration with Anneli to make a proper working plan for trying solving the issue
- **1h**: Trying to understand the mechanism used to render the chart when dataset is updated
- **30m**: Trying to understand why the correct color is rendered when we hover the bars but not when using update
- **1h30**: Trying to understand the resolving system for attributes in the project, looking into how needContext works
- **1h45**: Looking into the draw function
- **30m**: Looking into animations and why would something be wrong
- **30m**: Discovering the DevTools javascript debugger.
- **3h**: Working on the report.
- **Total time: 17h15**

Carl:

- **1h**: Reading the assignment requirement, and choosing a project
- **1h**: Basic set up and general build tests of the project
- **2h**: Reading the documentation to understand what the project does, how it works in general.
- **2h**: Getting into issue #11684, trying to trace back what exactly happens when the axis is set to y.
- **2h**:Using the debugger built in the program to find the problem, searching for similar issues at the same time. Rikard found a solution after his individual work.
- **2h**: Getting into issue #11679, creating a testfile using the code provided by the issue author. Trying to find where the in code the error occurs, realized when hovering, the bars show correct color.
- **45m**: Trying to understand why the correct color is rendered when we hover the bars but not when using update.
- **3h**: A total trace back of the problem, realizing the needContext function determines whether the new context should be considered during a chart update.
- **1h30m**: A deep look into needContext, multiple manual debugs using console.log to find out the reason why the context is considered not needed when needed.
- **1h30m**: Trying to dig more into the problem.

- **1h30m**: Trying to set up a unit test, and realized the problem is actually that the author is not corretly using the functions and directly deleting variables. However, the programme implements a cache and doesn't realize any change of the context when varaiables are deleted in that way, so the changes are not implemented.
- **1h45m**: Writing the report.
- **Total time: 22h**

Rikard:

- **1h**: Reading the assignment requirement, and choosing a project
- **1h**: Basic set up and general build tests of the project
- **2h**: Reading the documentation to understand what the project does, how it works in general.
- **1h**: Simple setup for testing solution.
- **6h**: Issue #11532
- **6h**: Issue #11684
- **2h**: Writing the report.
- **Total time: 19h**