d3Checker: static and runtime check tool for applying D3

Project for CPSC-547

Youssef Sherif: sharief@aucegypt.edu

Wei(William) Zheng: williamubc19@gmail.com

1.Project Background and Description

A wide range of users uses visualization libraries. Some of these users are experts in data visualization, and others are not. A lot of them would benefit immensely by checking whether they are adhering to visualization best practices or not. Even experts in data visualization might sometimes miss minor details while they are prototyping in a hurry. Displaying warnings and errors for suboptimal visualization practices can be useful for those users and would help improve the overall quality visualizations produced by the community.

Who doesn't like to understand whether the visualization they have built can have higher effectiveness? This is precisely what we would like to introduce to you. We are proposing a static analysis tool(linter) and library that helps D3.js users adhere to data visualization best practices.

But why two tools? Why not just a linter? Linters, which are static analysis tools, are not meant to check data-related issues. They are only meant to check for logical problems in the code. The reason linters can not check data-related matters is that data are dynamic and can not and AST trees should not analyze data. Data might not even exist in code and can be retrieved by an HTTP request from a backend server. We can use static analysis tools to check data-independent code, such as the usage of a specific kind of chart, for example.

Here comes the run time library to the rescue. Runtime libraries can detect issues with data, such as the color mismatch. This is because a runtime library has access to this data, even if the data is retrieved from a different server.

2.Programming Language and Framework

We have selected to work on D3.js for various reasons. First, web applications are currently on the rise. More and more desktop applications now migrate to web applications. So if we would like to pick a library to be used on the web, we are limited to javaScript libraries. Among all JavaScript libraries, D3.js is the most popular and resilient.

3.Previous Works

Andrew Mcnutt's Matplotlib linter and his paper “Linting for Visualization: Towards a Practical Automated Visualization Guidance System” is the only work we found that is similar to what we are trying to do[1]. Mcnutt proposed a long list of data visualization rules to be implemented but he implemented a few of them in his prototype(to the best of our understanding). While Mcnutt has named his solution Vislint and is referring to it as a linter, he is actually using a run-time library to check for errors and warnings. The term “linter” is usually associated with static analysis tools and this is the first time we find that “linter” is called for a run-time checker.

Mcnutt’s tool is used to check for warnings for Matplotlib. There is a huge difference, though, between D3.js and Matplotlib. Matplotlib is high level library that enables users to visualize without much hassle, but gives the user much less control. On the other hand, D3.js is a low-level library that gives the user a much more control.

We can say that Mcnuttl’s tool is the only runtime library checker for data visualization. We did not find any static analysis tool for d3.js or any other data visualization libraries. We have found javascript runtime checker libraries that are not involved with data visualization. These runtime error checkers are common in JavaScript since JavaScript is not statically types and runtime checking is in a lot of cases the only way to validate data and types.

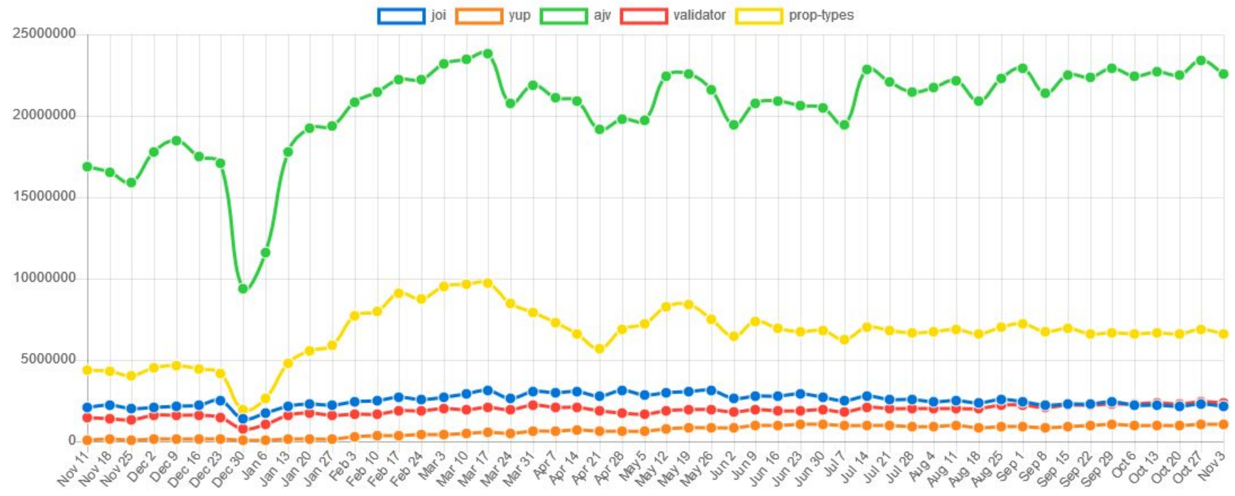


Figure 1.Number of installs versus time for the most popular JavaScript runtime error checkers

4.Personal Expertise

Youssef is professionally experienced in single page web applications with Javascript and has experience working with an array of JavaScript libraries. He did not work with D3 professionally we do not think would be a problem since he is planning to study D3 as part of the allocated hours for the project. Youssef is a first-year master’s student in computer engineering. One of the thesis ideas that he might continue to is about static analysis. He has also built before small (uncompleted) tools that analyze code statically. We believe that his professional experience in software engineering would help him structure the cleanVis library and setup the environment for the library besides setting standards on the library would be used.

Wei(William) is experienced in Data Analysis with Python and related data manipulating libraries such as Pandas and Numpy. He also has experience in visualization tools, including Matplotlib and Seaborn. Apart from programming experience based on Python, he has developed software using languages like Java, C++, and Pascal. He does not have expertise in D3; however, he has developed web applications using Javascript. William is a first-year master's student in Electrical and Computer Engineering. We believe that his experience in programming, especially web development experience would help him implement functions to check visualization rules for the run-time checker.

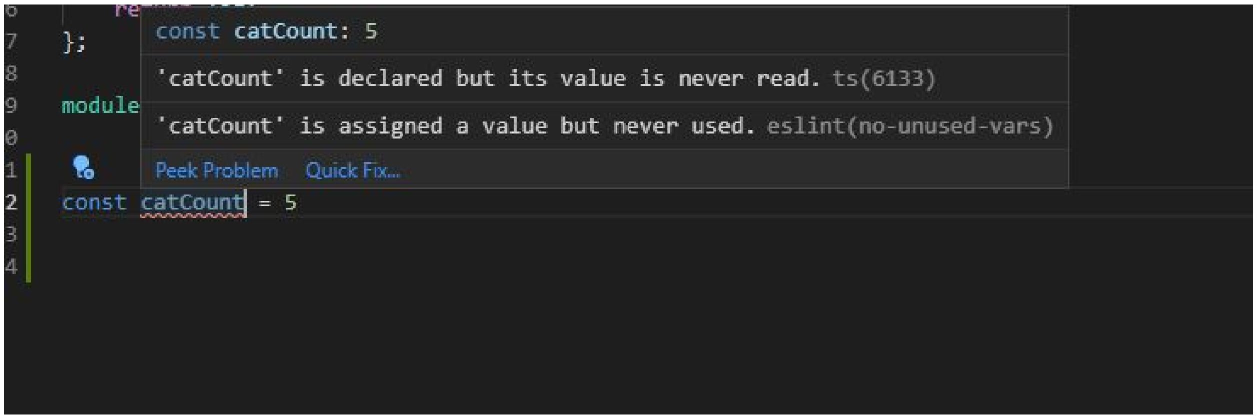
5.Rules for Data Visualization Best Practices

We are aiming to use rules from Tamara's book "Visualization Analysis and Design" 8, specifically order of effectiveness. Besides that, we plan to use "The Visualization Guidelines Repository" which includes a collection of guidelines hosted by the DBVIS group at the University of Konstanz4. In addition to that, we found that Yan Holtz's online guidelines to have useful rules of thumb that we think at least some of them can be implemented [6].

6.Scenario of use and Future Advancements

6.1 Static analysis tool Linter

The static analysis tool is intended to identify warnings and errors before the code is run. A usually good compliment to a static analysis tool is usually an editor's plugin to highlight areas in code that need to be changed. This way, developers can see errors and warnings while they are coding. An example of a very simple error for a static analysis tool is shown below.



```
};
module
const catCount = 5
```

```
const catCount: 5
'catCount' is declared but its value is never read. ts(6133)
'catCount' is assigned a value but never used. eslint(no-unused-vars)
```

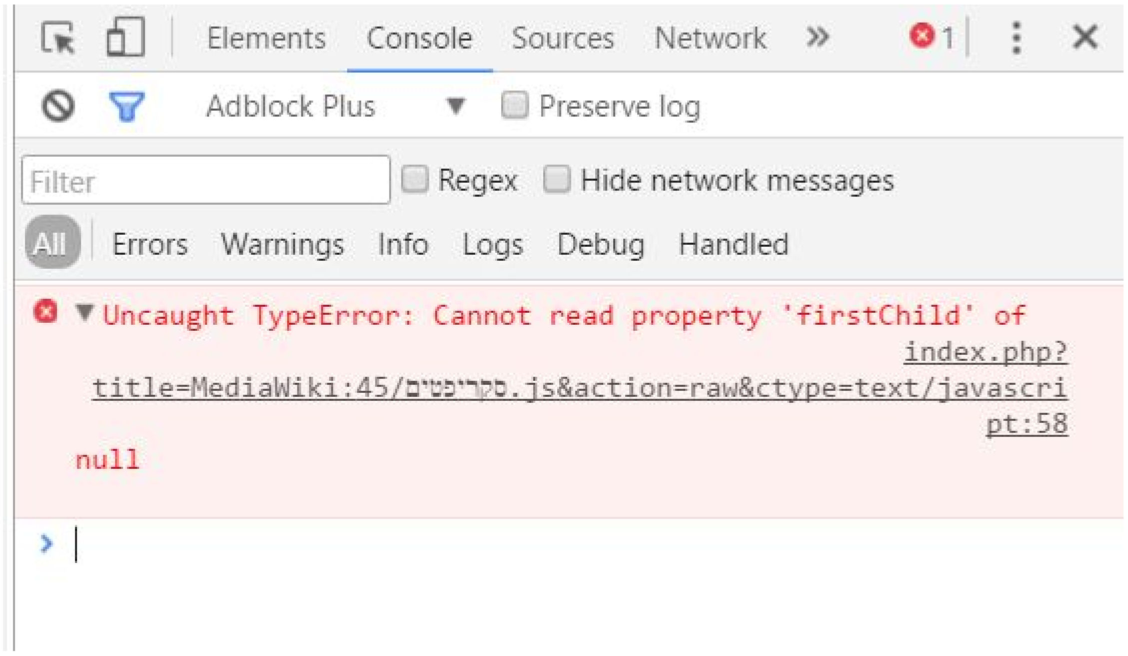
Figure 2. An example for Static Analysis Tool

Although we are planning to build a prototype for our static analysis tool(linters) in this project, we are not planning to integrate it with a specific IDE or editor's plugin. We believe that working on these extensions is outside the scope of this project.

For the scenario of use, let’s assume Ahmed is a web developer who has no domain experience in data visualization. Ahmad started using D3.js and thought that a pie chart is a cool visualization to display the distribution of grades for a classroom. The linter can put a yellow underline below the line where the pie chart method is called. A warning message that displays “A bar chart is usually a better visualization than a pie chart” would pop up when Ahmed hovers over that part of the line.

6.2 Run-time checker

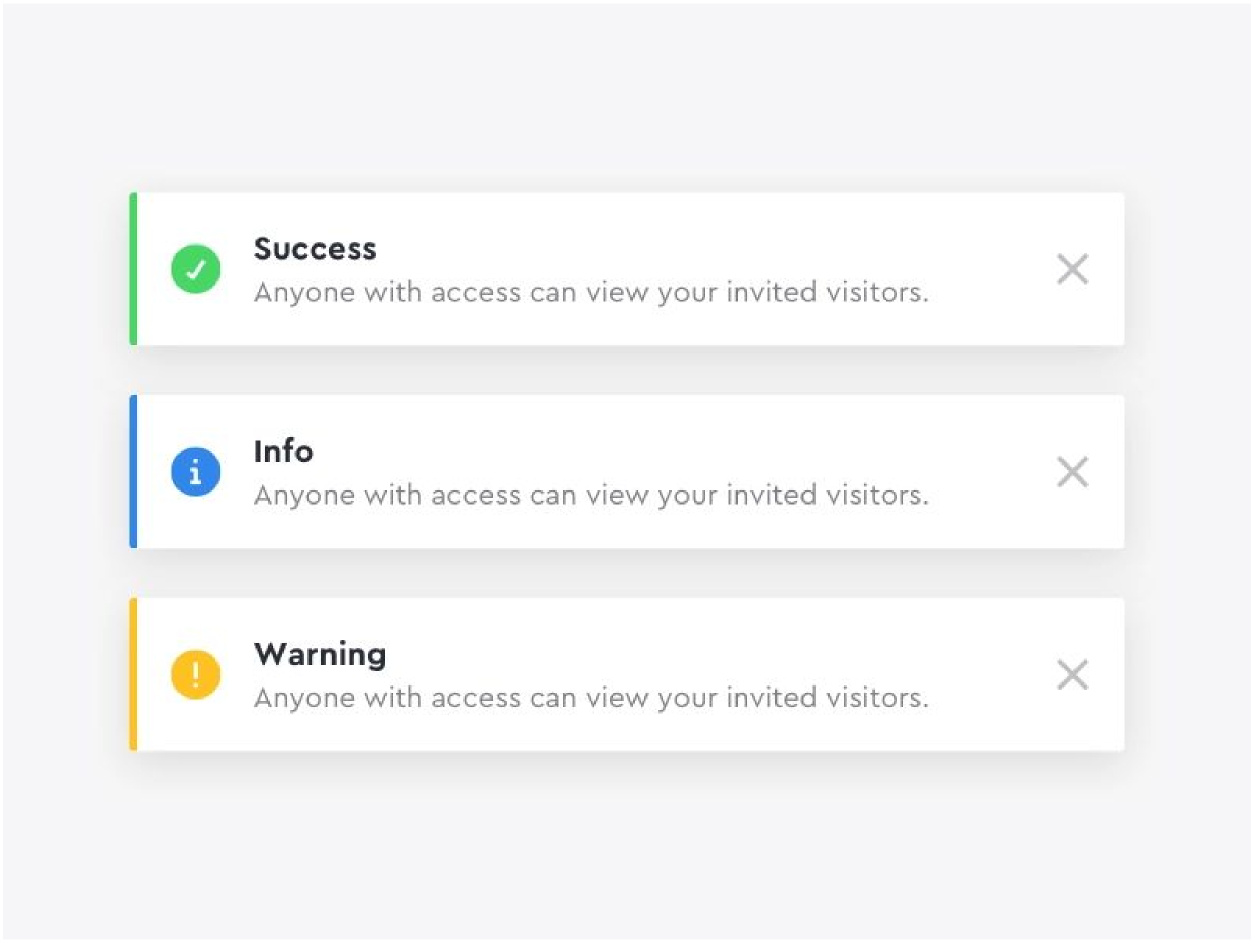
Run time errors and warnings appear in the console log for JavaScript web applications. This means that the user would need to open the dev-tools of a browser. We have thought of having alerts instead of console log warnings, but we found that alerts are quite intrusive, and it is impractical to display more than one warning via an alert. An example of console log error is shown below.



```
Uncaught TypeError: Cannot read property 'firstChild' of
                                                  index.php?
title=MediaWiki:45/סקריפטים.js&action=raw&ctype=text/javascri
                                                       pt:58
null
```

Figure 3. An example for Run-time Check

A better idea might be using unobtrusive toasts. For sure, there should be a switch to turn off or turn on these toasts so that they are displayed only during development and not in production. We are not planning to implement this in this project. We believe that this is out of the scope of the course since it depends mainly on UI development and UX design. An example of unobtrusive toasts that can be displayed in run time is shown below.



Anyone with access can view your invited visitors.

Anyone with access can view your invited visitors.

Anyone with access can view your invited visitors.

Figure 4. An example of unobtrusive toasts that can be displayed in run time

For the scenario of use, let's assume that the same developer, Ahmed, is using the bar chart which its labels are vertical. A toast then appears stating "Use horizontal labels for bar charts"

7.Proposed implementation approach

7.1 Linter

We plan to build the linter(static analysis tool) as a plugin in ESLint, which is the most popular JavaScript linter. Eslint is a pluggable linter. Plugins have been built before for JQuery, React and other libraries.

7.2 Runtime Checker

We plan to build the run-time checker library in JavaScript. We plan to use Webpack to be able to have different modules in code.

8.Milestones and Schedule

Table 1. Milestones and schedule

Task	Hours	Deadline	Description
Team Meeting 1	3	October 16	Brainstorm ideas, meet with the professor
Team Meeting 2	4	October 28	Discuss the goal and scope of this project
Proposal	13	November 4	Plan our work and write the proposal
Understand Mcnutt's research and tool	5	November 6	Check Mcnutt's research and tool
Understand how D3 works	10	November 8	Read online tutorials and attend online courses
Static analysis implementation	35	November 30	Implementing static analysis part
Run-time checker implementation	40	November 30	Implementing run-time checker
Test for visualization rules and debug	10	December 5	Making the tool as full-featured and refined as possible, adding any of the interaction features that were missed in the first sprint. Also evaluation of the tool's performance and utility
Final presentation	10	December 10	Make slides and practice
Final paper	10	December 13	Write final paper

Reference

[1] Andrew McNutt, Gordon Kindlmann. “Linting for Visualization: Towards a Practical Automated Visualization Guidance System”.

[2] Jonathan Harper, Maneesh Agrawala. ”Deconstructing and Restyling D3 Visualizations”. UIST '14 Proceedings of the 27th annual ACM symposium on User interface software and technology, pages 253-262.

[3] Extracting Data and Structure from Charts and Graphs for Analysis, Reuse and Indexing. http://graphics.stanford.edu/projects/dataExtract/.

[4] Visualization Guidelines Repository. http://visguides.repo.dbvis.de/guidelines.php.

[5] Stephen Ingram, Interactive Visualization of Small World Graphs in Prefuse.

https://www.cs.ubc.ca/~tmm/courses/cpsc533c-05-fall/projects.html#sfingram.

[6] A collection of dataviz caveats. https://www.data-to-viz.com/caveats.html.

[7] D3 Gallery. https://github.com/d3/d3/wiki/Gallery

[8] Tamara Munzner. ”Visualization Analysis and Design”. CRC Press, 2014.