Graphical Illustration of Code Structure

Wenhao Su* wenhaos3
Yichen Yang† yy18
Zhen Yu‡ zheny2

1 Abstract

In this project, we have implemented an information visualization application using D3.js. This application have two functions: 1. visualize the structure of html with tree structure, 2. show the function relationship of java with hierarchical edge bundling. This project can help the developers to visualize the structure of html and easily understand how the java program works. In addition, in java visualization, one can click the specific parts to learn more. In this way, the application can help the users to quickly understand the program.

2 Introduction

In people's daily life, it is hard for one to quickly understand the structure of a html file or the function relationship in a java file, especially when the file contains thousands lines of code. To solve this problem, we developed this application. It can provide a easy and simply visualization of the html and java file. One can choose to either upload a file to the browser or copy and paste the code to the text area.

For html, since html is tree structure, we choose to use Collapsible Tree in d3. The html parser will parse out all the html tags and their relationships. Then d3 will generate the tree out. This application will provide the users with a clear and direct visualization of the html. When one want to fetch some specific parts of the html, it will really help to get the path of the tag.

For java, since the function relationship in java may occur circle, we choose to use Hierarchical Edge Bundling in d3. The java parser will parse out all the function names and the function the call in java. Then d3 will generate the edges between each function with call and call by. One can see the red and blue edge which shows call and call by on the graph.

3 Works

3.1 Java Parser

For Java Parse, we use the npm library parser-java to convert the java program to an Abstract Syntax Tree(AST), represented by JavaScript Object. After that, we recursively traverse the AST to find all names of the methods in a class, and then figure out all function calls within each methods. The information is stored in JSON format and then feed into D3.js program to construct edges in different colors.

3.2 HTML Parser

For HTML Parser, we simply use the npm library html-to-json to transform the html text into json format, which provides the information of the DOM tree that can be passed to D3.js SVG sketch program.

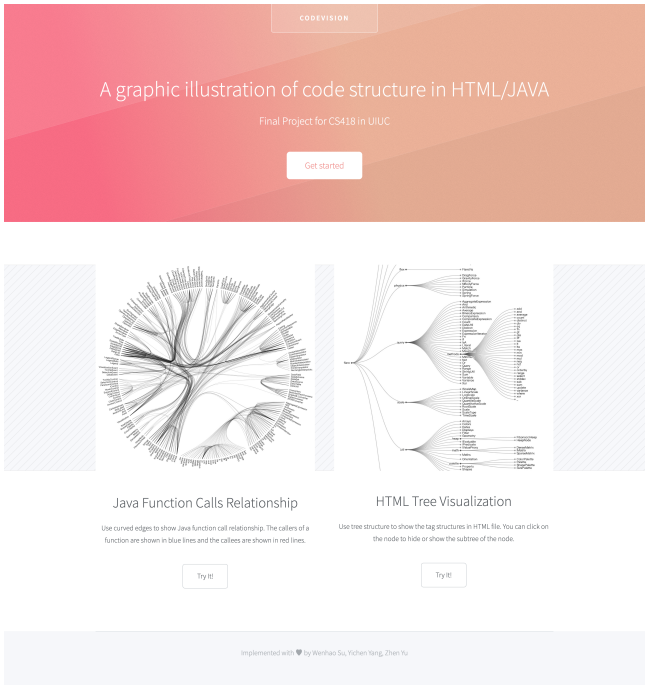

Figure 1: Welcome Page

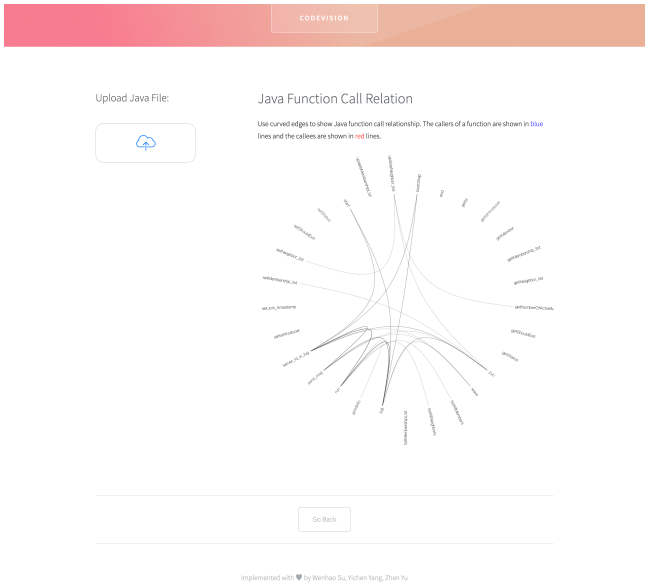

Figure 2: Java Parser Page

*e-mail: wenhaos3@illinois.edu
†e-mail: yy18@illinois.edu
‡e-mail: zheny2@illinois.edu

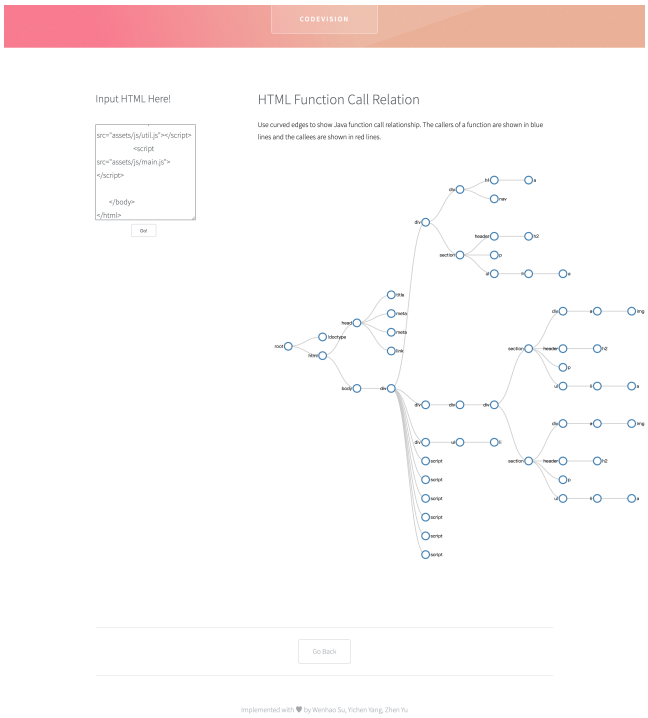

Figure 3: HTML Parser Page

3.3 Visualization via D3.js

Besides the parser, in order to give a clear and interactive visualization, we chose to use D3.js to show the parsed code structure.

For the HTML data, based on its internal DOM tree data structure, we could directly implement an interactive Collapsible Tree for data visualization. The implementation is based on the tree structure defined in the d3-hierarchy package. Upon the DOM tree of the HTML is parsed into a JSON data, we recursively traverse the JSON and create a d3 tree node for each object. The result is displayed as an SVG to the front-end, whose height and width are subject to the scale of the generated tree. Since the DOM tree of input HTML maybe complex, the sketched tree might not fix into the screen. Hence, we added an interactive function that, once user clicks on a node, all the child nodes will be collapsed, resulting in a more condensed graph for better display effect.

For the Java data, we choose to use the Hierarchical Edge Bundling structure to represent the call relationships between different functions. After getting the parsed JSON, we still extract all the functions, and bind each of them with a node of the d3 tree structure. However, we use the bilink function to support two different edges between nodes: the in-coming edge and the out-coming edge, shown on the front-end respectively as red and blue curves. Once the user hovers on the function name, all curves will be highlighted in different colors according to their property, for a better insight to the detailed relationship between different functions.

3.4 User Interface

We implemented a text-area and a file upload button on the front-end to allow users upload their code or paste their onto the area.

4 Results

Here is the github source link:
`https://github.com/WenhaoSu/code-structure-illustrator`

Here is the demo web link:
`https://wenhaosu.github.io/code-structure-illustrator/`

5 Conclusion

In this project, we have successfully implemented the graphical illustration of code structure. It provides the user will a easy way to understand the code structure of the html and java. With the application, users can see the tree structure of html and relationship between functions in java clearly. Therefore, we think this project is a successful project.

6 Reference

`https://observablehq.com/@d3/hierarchical-edge-bundling`
`https://bl.ocks.org/d3noob/8375092`
`https://html5up.net/telephasic`