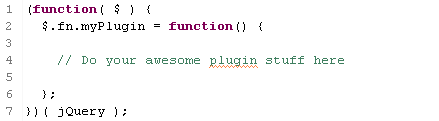
## Introduction

XXX is a code recommendation tool for JavaScript, and we implement a prototype as an Eclipse plugin. JavaScript is a dynamic language, the type of object could be change at runtime. In addition, JavaScript Library also should be execute before the user’s code. So XXX use the method combine dynamic and static analysis, can give user runtime recommendation.

## Motivating example

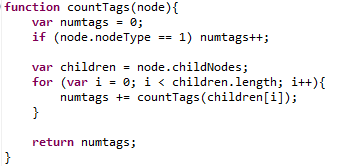
**Framework**



Some frameworks have dependencies on other library. Fig.1 shows our example, a simple program of initialization for jQuery Plugin (more than 1000 plugins exists). It add a new property to the object of jQuery at runtime. So we need to execute the library and build the model. We can get the

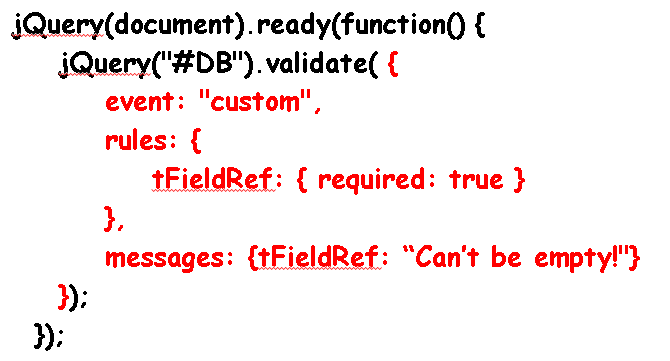
**Variable may have several different types.**

Sometimes variable in JavaScript do not have declared type and may have several different types, as parameter and return type. We want to give proposals in such kind of situation.



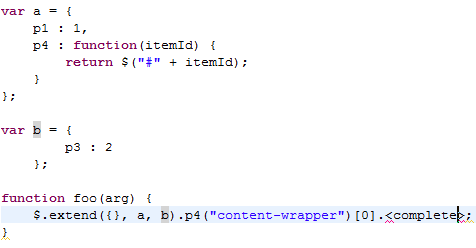
**Completion on argument**

For some arguments in JS library which could be literal object. Most of them are options, so it will have amount of fields and the object without explicit type structure, which could be confusion to user. In general, user need to read the library document or find some examples for the usage of complex arguments.



**Complex expression in JS**

As there is no return type in the function declaration, so it could be difficult for user to get recommendations.

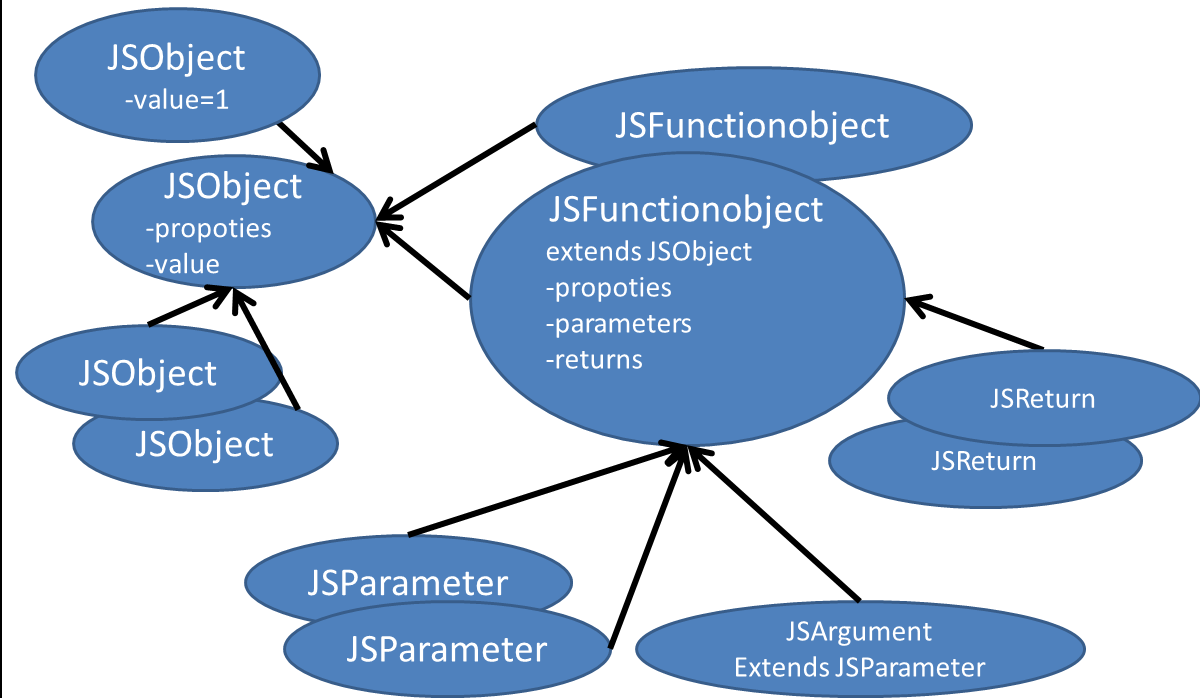


## Generating framework model

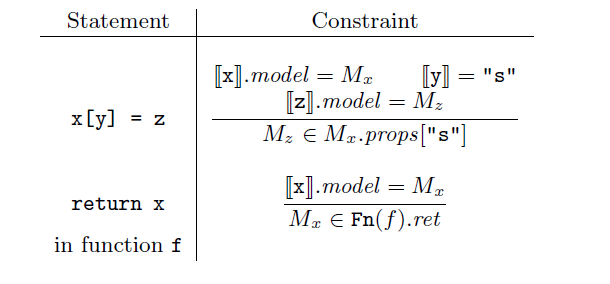
We choose the method combine the static analysis and dynamic analysis to build the structure model for user’s library. The process can be divided into 2 phases, in the first phase, we simulate a browser and execute the JS library, and then we record all of the class structure and variable into our model, also get the information of parameter and return type information in the source code. In the second phase, XXX need example code of library usages provided by programmer, we learn from example code to improve the completion proposals.

**Build model from execution result**

As the problem shows at 1.1, XXX generates structure model from execution result instead of static analysis. When generating the structure model, XXX first simulate browser environment using Env.js. Then we execute all of library files, as the result we have many JS Value Objects. Actually there are two kinds of JS Object, one is common object and another is function object which is a special object. In the JS Object, there are two fields: property set and value. So we can also use this class to present both primitive variable and complex Object. For function object, there are another two fields one is a parameter set and the other the return object. These two fields used for completion of arguments and call sequence. At this time these two fields are empty, and we will fill it in the following steps. Fig \*\*\* shows the structure of our JS Object model.

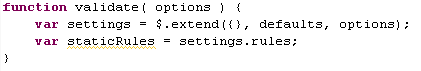


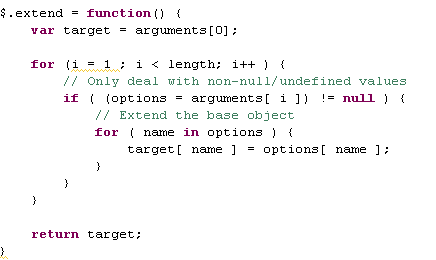
Then we need to collect the information of parameter and return type. XXX first scans the function declaration code to find the usage of parameter and return object. Now we collect the methods invocation and field access happened on the parameter or return object as the usage. And we can get the type of object and its fields from analysis of assignment or condition statements. The technique report of IBM do the related work.



**Build model from example code**

We learn from example code to improve model from execution result, specially focus on the parameters. Consider the following example, we can’t get the structure of the parameter only from function declaration. In the declaration of function validate, default value of options would be merged to options, and in the remaining part of the function, it will use settings instead of options. So it would be hard to relate options with settings. Therefore, we decide to learn parameter usage from example code.





We record the argument structure and property type from example code, and store these information into files. For every usage, we would create a JSArgument Object, which would deep copy all of the properties and analysis the type of the property. As programmer maybe interested in the original code of the arguments, we also record it as string in JSArgument.

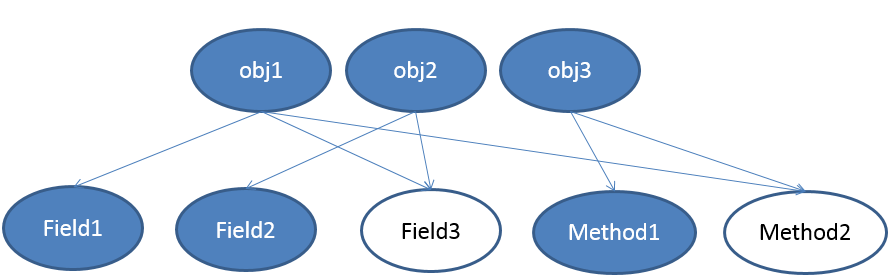
## Get possible candidate from model

For the variables which could be any types, so we have to find the possible candidate from model. First we extract feature of it, then look into the model we build before, use Bayes network to get the most possible candidates. XXX use static analysis for library source code and learn from the example code to get the parameter information, give proposals of parameter structure and example usages to programmer.

Variable have several features could show us which type in the situation (like 1.2). First one is the fields and methods invocated on this variable. This feature could show the most valuable information of the type structure, but when there are no invocations this feature would be useless. So we add another feature: the variable name. Generally, the name of variable consists of several words and separated by capital letters. So we split the word name into tokens, then find the similar variable in the model.

**Feature 1:**

Firstly, we extract the methods and fields invocated by this variable. And search in the model find all possible types. Build a Bayes network to rank the candidates. The top node has the outputs of all possible candidates. And there're some nodes could be observed, presents the fields or methods could be invoked. We can use this method to rank the proposals. Here's an example.



**Feature 2:**

If we couldn’t get information from Feature 1, we use the variable name as Feature 2. Most of variable name could be separated by capital letters, and every token is meaningful. So we use these token as an additional feature and also could be build a Bayes network. The top node contains all objects which contain tokens, then we can give some possible proposals. We use this feature only there’s no feature1.

## Arguments recommendation

We want to give recommendations for arguments, especially focus on the structure of arguments. Eclipse and Visual Studio use annotation to give the type of arguments, we try to get the type structure from library source code and example code.

The argument in example code is most possible, we make summarize and give the structure we learned at the first place. Then XXX scans the function declaration, get features of the parameter. Use the method in chapter 4 get the possible candidate. When we get the request from programmer, XXX will show type structure at this place, if programmer use object literal and fill in some fields, these are all features of argument so use these information to find candidate from model.

## Give Runtime Recommendation

XXX not only execute the code in library, but also the code is writing. Runtime recommendation is useful for complex usage, such as call sequence or array access. Programmer can get proposals as when they are debugging.

Some requests might in function declaration or anonymous class definition. Therefore we run code in a special sequence. XXX separate the execution into several scopes, the previous scope provides the context of the next scope. Here’s an example: Run JS code with unknown environment would have many problems. For example: related html file, parameter, field in the class. All of above issues can be summarized as there’s undefined variable. The main idea is find the most possible variable replace the undefined variable at the runtime. We use the method in chapter 4 to find the replace variable. First extract the features of the undefined variable then find in the model. The major workflow as shown below:



Related HTML file is important to JS, XXX allow programmer define the dependency html file. Then we can give proposals focus on this special kind html file. If there’s no html file XXX provide a default HTML file.

## Future work

**Ranking**

In some special JS object like “jQuery” it is the major object for jQuery, actually it almost contains every method of jQuery (more than 240). And other jQuery plugin also add method in this Object. XXX gives more than 240 proposals, so ranking is very important for this situation.

We have two ideas of ranking, one is Bayes Network, we can improve our result from learning example code. The other is compare the trace of execution with test case to find the most possible proposal.

**Execute example code not only static analysis.**

Now we only use static analysis on example code, and we want to get more info from it. So we plan to run example code try to get the trace of it, and other …