JavaScript API Misuse Detection by Using TypeScript

Jihyeok Park
KAIST
jhpark0223@kaist.ac.kr

Abstract

Static analysis of JavaScript programs to detect errors in them is a challenging task. Especially when the program imports massive JavaScript libraries such as jQuery and ExtJS, analyzing the whole program and the libraries is expensive and extremely degrades the analysis efficiency. In this paper, we introduce a novel approach to solve the problem by modularizing the analysis. We separate the analysis of JavaScript libraries by using types extracted from their corresponding specifications in TypeScript and the analysis of JavaScript applications by a static analysis framework. We use DefinitelyTyped, an open-source repository that provides TypeScript declaration files of over 300 popular JavaScript libraries, and we extend SAFE, an open-source analysis framework for JavaScript.

JavaScript [2]

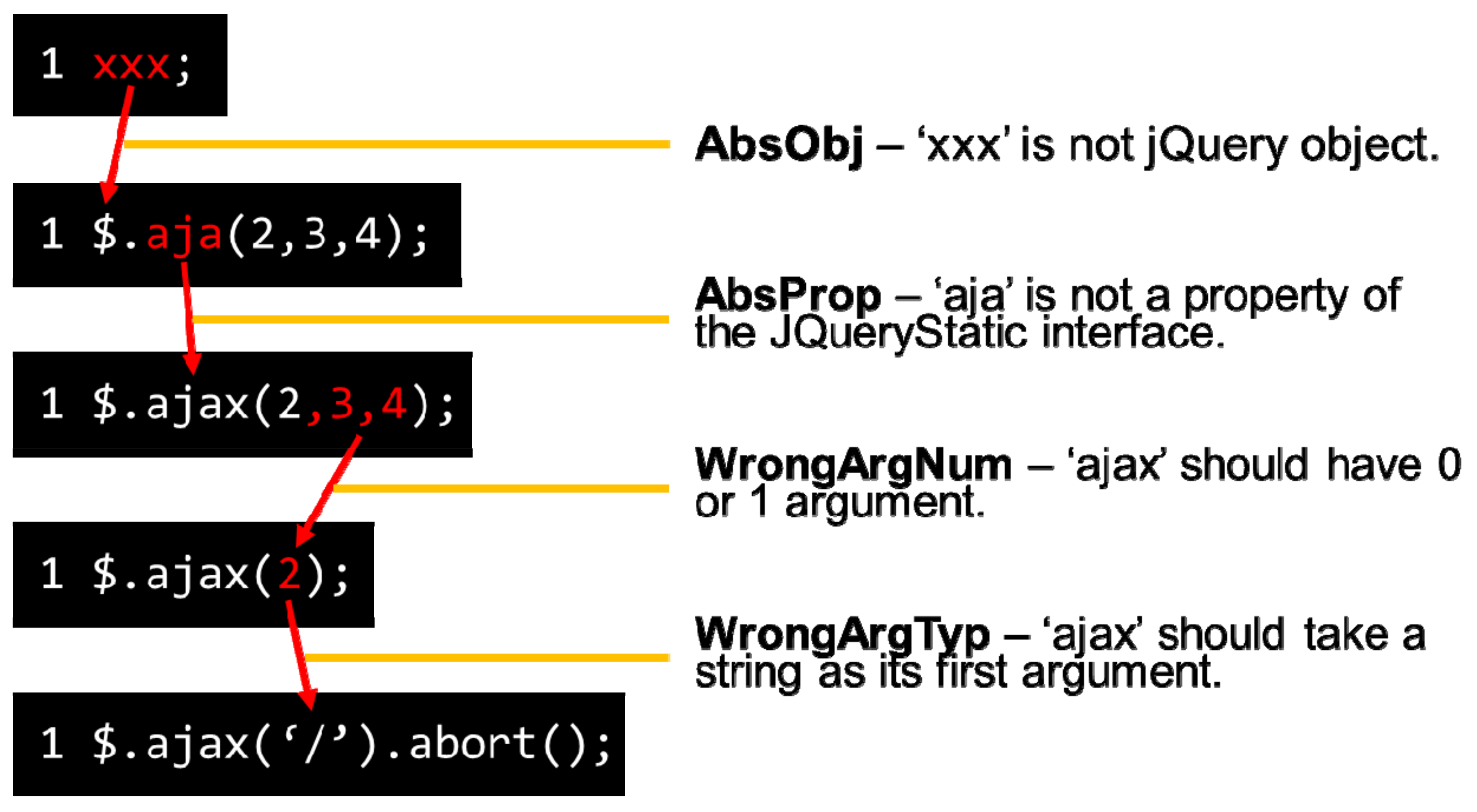
- Scripting language
- Dynamic language
- Prototype-based object-oriented language
- Implicit type conversion

TypeScript [1]

- Strict superset of JavaScript
- Static typing
- Class-based object-oriented language
- TypeScript declaration file

JavaScript API Misuses

AbsObj - Access to absent objects/functions in APIs
AbsProp - Access to absent properties of API objects
WrongArgNum - Wrong number of arguments
WrongArgTyp - Wrong type of arguments



```
1 xxx;
```

AbsObj – 'xxx' is not jQuery object.

```
1 $.aja(2,3,4);
```

AbsProp – 'aja' is not a property of the JQueryStatic interface.

```
1 $.ajax(2,3,4);
```

WrongArgNum – 'ajax' should have 0 or 1 argument.

```
1 $.ajax(2);
```

WrongArgTyp – 'ajax' should take a string as its first argument.

```
1 $.ajax('/').abort();
```

DefinitelyTyped [3]

- Open-source repository that provides TypeScript declaration files of over 300 popular JavaScript libraries.
- TypeScript Declaration files assign types to API objects with newly defined types as *interfaces*, *classes*, and *enums*

- Example :

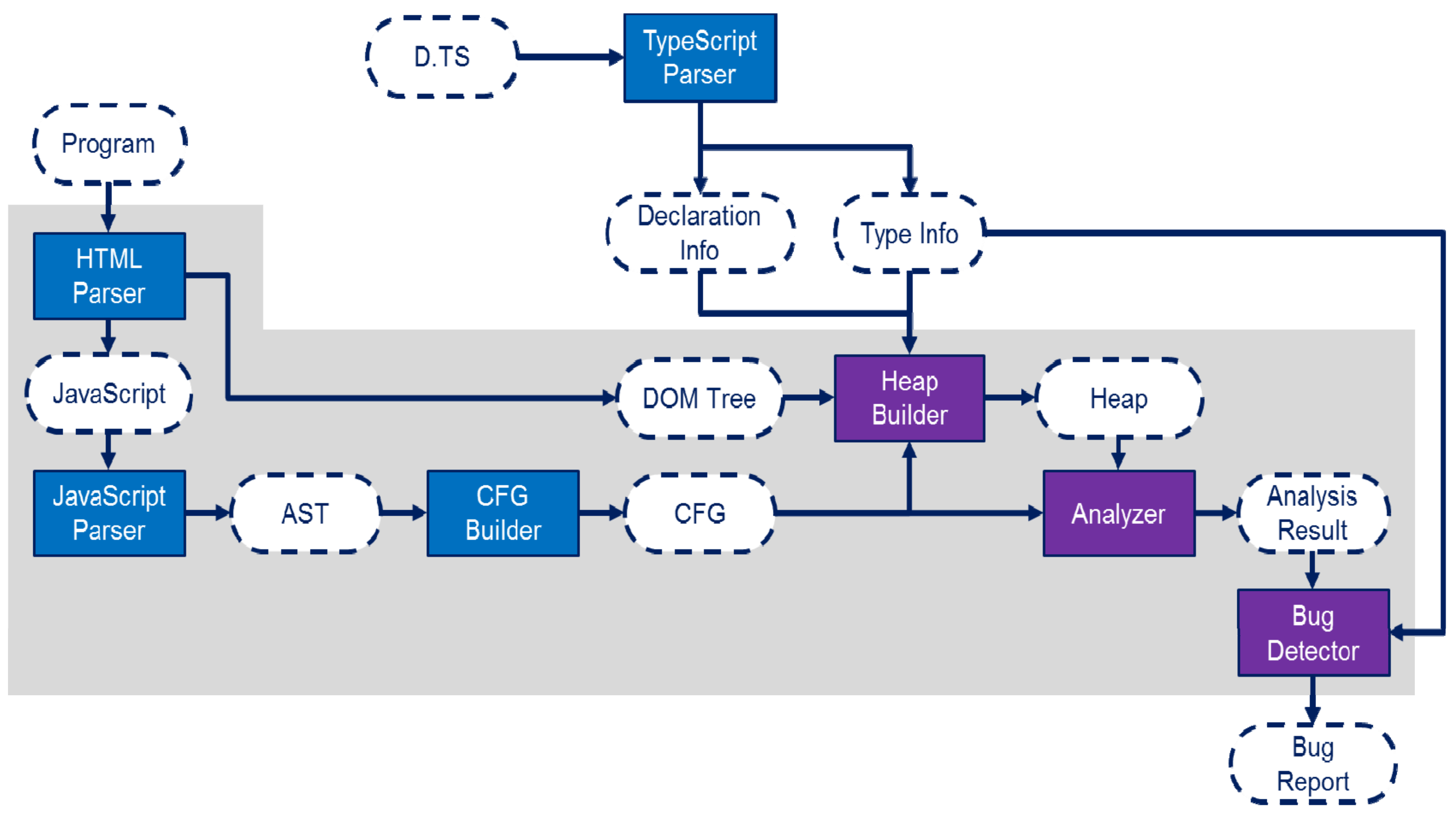
– jquery.d.ts

```
...
interface JQueryXHR extends XMLHttpRequest, JQueryPromise<any> {
    overrideMimeType(mimeType: string): any;
    abort(statusText?: string): void;
}
...
interface JQueryStatic {
    ...
    ajax(url: string, settings?: JQueryAjaxSettings): JQueryXHR;
    ...
}
...
declare var $: JQueryStatic;
```

Methodology

Extend SAFE (Scalable Analysis Framework for ECMAScript) [4, 5]

- **DefinitelyTyped** [3] to get TypeScript declaration files of over 300 popular JavaScript libraries
- **TypeScript parser** to parse TypeScript declaration files
- **Mock-up objects** to model every interface, class, and enum defined in TypeScript declaration files
- **Misuse detection** by using mock-up objects and the analysis results from SAFE



Reference

[1] *Microsoft. TypeScript.* http://www.typescriptlang.org.
[2] *ECMAScript® Language Specification*. Edition 5.1. http://www.ecma-interfnational.org/publications/standards/Ecma-262.htm.
[3] *DefinitelyTyped*. https://github.com/borisyankov/DefinitelyTyped.
[4] Hongki Lee, et al., *SAFE: Formal Specification and Implementation of a Scalable Analysis Framework for ECMAScript, In FOOL*, 2012.
[5] SAFE, http://safe.kaist.ac.kr.

This work is supported in part by Korea Ministry of Education, Science and Technology(MEST) / National Research Foundation of Korea(NRF)
(Grants NRF-2011-0016139 and NRF-2008-0062609).