Javascript Report for Honours Compilers2

Clayton Sibanda

Department of Computer Science
University of Cape Town
South Africa

Abstract

CCS Concepts • **Compilers** → Compiler Design

Keywords Lexer, Parser, Code Generation

1. History of JavaScript

The story of JavaScript begins in 1995 when Netscape owned over 90% browser market share after taking over from Mozaic. Netscape hired Brandon Eich to write Scheme in the browser. Scheme is a Lisp family language that was famous in the 90s[12]. After attempting to do a deal with Sun systems, it was decided that Brandon creates a language similar to Java but different. on top of being similar to Java, JavaScript was suppose to tolerate minor errors, have simplified event handling and make it easy to copy and paste code snippets.

The new language was called to be called Mocha which was later changed to LiveScript. By the end of 1995, Netscape and Sun systems had struck a deal and LiveScript was changed to JavaScript[6]. This was mainly done for marketing purposes, not meant to imply that the new language is a derivative of Java[6].

These were the days when Java was the dominating and most promising programming language. As a result the new language for the web was suppose to look like Java [6]. For this reason JavaScript adopted a syntax similar to that of Java. It also inherited date objects that were similar to Java together with the Java y2k date bugs[11]. However the programming paradigm was never meant to be the same. This similarity in syntax can deceive for most new developers trying out JavaScript. As the two languages differ especially when it comes to their object oriented programming paradigm.

JavaScript was meant to be an 'Object-based/ object oriented scripting' language [11]. For this, Eich looked into an object oriented language called Self. Self's programming paradigm employs the concepts of prototypes [13]. So JavaScript adopted the use of prototypes in objects and prototypical inheritance which is still present today. This means that all functions in JavaScript have a prototype field [11]. This allows all functions in JavaScript to be able to construct objects. Unlike most languages that use access modifiers for information hiding, JavaScript employs closures to achieve this.

Another feature which was discovered later by most developers in JavaScript was its functional programming capabilities[6]. Functional nature of JavaScript was inspired by Scheme, a Lisp family functional programming language. It meant that functions in JavaScript are to be treated as first class citizens. The functional programming paradigm in JavaScript has been persisted ever since and its still utilised today[6].

The feature of tolerating minor errors made JavaScript to be permissive in most cases. For example a function can be invoked with too few or too many arguments. Types also convert freely e.g '1.0'==1 and '1'==1 both evaluate to true[11]. This allows bugs to hide in most programming projects. Most of the bugs in JavaScript also come from the fact that the entire language was created in ten days.

2. JavaScript evolution and role in web development

The late 90s were characterised by brutal browser wars in the software industry[3]. These were mainly between Netscape's navigator and Microsoft's Internet explorer(IE).

Microsoft had developed their own JavaScript variants, VBScript and JScript. JScript was said to be a reverse engineering of JavaScript which was given a different name for legal reasons.

Having many languages for developing websites posed a lot of challenges for web developers. It meant that developers needed different codebases for different browsers. It lead to situations where browsers would have logos written "best viewed on IE" or "best viewed on Netscape navigator". A standard was needed so as to ease the process of developing websites.

In 1997 European Computer Manufacturers Association(ECMA) standardised JavaScript. The ECMA standards body could not get anyone to donate a trade mark, so they gave it the number ECMA-262. As a result JavaScript adopted the standards name ECMAScript. This led to major players like Microsoft adopting the use of JavaScript in their IE browsers.

The coming of IE4 extended JavaScript with XML first class object subtype literal syntax[11]. Later Asynchronous JavaScript and XML(AJAX) was later added to JavaScript which allowed browsers to interact with the server without reloading. With adoption from Microsoft, JavaScript eventually became the de facto language for web development.

Initially JavaScript was used to manipulate document object model(DOM) components so as to achieve popups and animations. The coming of AJAX allowed developers to expand its usage to a wide range of applications.

Most Social media platforms started using it for single page applications. In single page applications, developers would leverage JavaScript features to achieve infinite scroll. Infinite scroll is used to simulate content delivery on demand for web applications as users scroll down [5]. JavaScript also allowed to websites to dynamically renders advertisements after the page had loaded.

3. Variants of JavaScript

The dramatic history of JavaScript resulted in an erroneous language that is non-conforming when compared to most of its peers. A lot of developers experience problems when delving deep into JavaScript due to its unusual features. Some of these features include dynamic typing, tolerance to errors and asynchronous call back functions. These problems lead to unnecessary bugs and performance issues on websites.

In the quest to make developing JavaScript applications easier and efficient, seasoned developers began to develop JavaScript frameworks. Some developed languages that compile to JavaScript that have a better typing system. On the other hand, others like jQuery were created to simplify and speed up DOM manipulation as vanilla JavaScript is deemed long and verbose. One could also argue and say that developers created frameworks to obey the principle of "Don't Repeat Yourself"(DRY) since most of the programming done in the web is repetitive.

DOM manipulation is the most recurring task when using JavaScript. In order to make it simple and easy, jQuery was created. For the past ten years, jQuery has been the most used JavaScript framework. Its advantage is that you write few lines of code and achieve more. It therefore speeds up development and lowers development costs in the long run.

Being used for DOM manipulation and browser based applications, JavaScript was always thought of as a browser based language. However, the coming of Node.js in 2009 changed everything. Node.js is a JavaScript run-time environment that executes JavaScript code outside of a browser. Node.js allowed developers to developer server side applications using JavaScript. This made JavaScript even more popular in the software industry. It lead to the development of a package system for JavaScript and gave it the ability to manipulate binary files.Its non-blocking event loop was very attractive to most developers and companies. Node.js was built on top of the google V8 engine, which is a JavaScript engine for the Google chrome browser. V8 was chosen over Mozilla firefox SpiderMonkey and IE Chakra due to its speed of execution. Node.js excelled in easing the process of writing web apis and real-time applications compared to other famous web applications.

Even though developers were excited about the new developments in JavaScript especially the coming of Node.js, some were still not happy about the dynamic type system in JavaScript. This led to the creation of TypeScript in 2012 by Microsoft. TypeScript is a strict syntactical superset of JavaScript, and adds optional static typing to the language. TypeScript allows developers to write code that has less bugs compared to JavaScript. It is also less error tolerant and types don't just change.

Another language that has come up which is similar TypeScript is Dart. Dart includes both optional and static typing. It is mainly used for developing client side applications both for the web and mobile. Dart is still fairly new and so its advantages haven't been explored much.

Other variants include CoffeeScript, React, Angular and Vue. CoffeeScript's main goal is to expose the "good parts" of JavaScript. It is basically a little language that compiles to JavaScript [4]. CoffeeScript allows developers to write few lines of code to produces code that produces that performs the same or better than code written JavaScript.

React, Angular and Vue are the most trending JavaScript frameworks today. These allow developers to develop websites in components. This is great because it makes the code modular and easier to maintain. They also make state management for complex applications easy [8]. These component based libraries leverage a build tool called webpack. Webpack is a frontend tool used to bundle static assets in web development [10].

4. compilation/translation model for JavaScript and each of its variants

4.1 compilation/translation model for JavaScript

JavaScript is both a dynamically typed and interpreted language. This means that it has no compilation step like Java, C++ and other similar languages. To run JavaScript, an interpreter in the browser reads over the code and interprets each line and runs it. Some modern browsers use Just-In-Time compilers which compiles JavaScript to bytecode just as it is about to run.

The compilation process for JavaScript involves Lexing, Parsing and Code generation. Lexing takes in the code characters/strings to produce token objects. These tokens are taken by the parser which generates an abstract syntax tree. During the code generation step, the abstract syntax tree is traversed to produce the needed code[7].

4.2 compilation/translation model for Babeljs

In 2015 the ECMAScript 6(ECMAScript 2015) specification was released for JavaScript. This new specification would change how developers write JavaScript. It introduced new keywords like 'const', 'let' and brought about classes to JavaScript. Some of the life changing features were arrow functions. As with all new standards in programming languages, it was loved by a few and hated by some. Those who hated the new features were part of the people with large code bases written in the old standards.

To solve this, Babeljs was born. According to the documentation on the official website, Babeljs is defined as a toolchain that is used to convert ECMAScript 2015+(ES6) code into backwards compatible version of JavaScript[1]. Babeljs transforms most of the new syntactic sugar in ES6 into the old backward compatible syntax. As a compiler, Babel has three components which are: Parser, Transformer and the Generator. The parser takes in a stream of characters and turns them into a token object. The transformer takes the token object and converts it to an abstract syntax tree. The abstract syntax tree(AST) is fed into the Generator which then generates code[2].

4.3 compilation/translation model for TypeScript

TypeScript, a superset of a stricter version of JavaScript is used to achieve static typing in JavaScript. The compilation process for TypeScript begins with preprocessing. The preprocessor has to figure out what files should be included in the compilation process first. The parser then generates AST nodes from the code. This then followed by a binding process where the binder traverse the AST nodes, generates and binds symbols. The binder also ensures that generated symbols are created in the correct enclosing scope.

After the binding process, a sourceFile is generated by invoking the createSourceFile API. From the sourceFile, a program is created by calling the createProgram API which builds source files into a program. The program is the used to create an instance of the TypeChecker. The TypeChecker is the core of the TypeScript system since it is used to decipher the relationship between symbols from different files. An emitter can also be generated from the program. The emitter is used to generate .js and .jsx files [9].

4.4 compilation/translation model for CoffeeScript

CoffeScript is a JavaScript variant that allows users to write simple and clean code which is then converted to its JavaScript equivalent. The first part of translation involves a lexer which takes in a stream of characters and converts it to token objects. From the token objects an AST is generated which is later converted to JavaScript code in the code generation stage.

References

[1] Babeljs javascript. `https://babeljs.io/docs/en/`. Accessed: 2019-10-20.

[2] Babeljs javascript. `https://github.com/hzoo/so-how-does-babel-even-work/blob/master/React%20Rally.pdf`. Accessed: 2019-10-20.

[3] Browser wars - 1990s javascript. `https://crossbrowsertesting.com/blog/test-automation/history-of-web-browsers/`. Accessed: 2019-10-20.

[4] Coffee Script javascript. `https://coffeescript.org/`. Accessed: 2019-10-20.

[5] *Development of Web UX Pattern System for Wordpress Service, and Design Suggestion for Wordpress Theme Filtering Service.*

[6] JavaScript: how it all began javascript. `https://2ality.com/2011/03/javascript-how-it-all-began.html`. Accessed: 2019-10-20.

[7] JavaScript javascript. `http://web.stanford.edu/class/cs98si/`. Accessed: 2019-10-20.

[8] React.js javascript. `https://reactjs.org/`. Accessed: 2019-10-20.

[9] TypeScruptjs javascript. `https://github.com/microsoft/TypeScript/wiki/Architectural-Overview`. Accessed: 2019-10-20.

[10] Webpack.js javascript. `https://webpack.js.org/`. Accessed: 2019-10-20.

[11] Eich, B. Javascript at ten years. *SIGPLAN Not. 40*, 9 (Sept. 2005), 129–129.

[12] FINDLER, R. B., CLEMENTS, J., FLANAGAN, C., FLATT, M., KRISHNAMURTHI, S., STECKLER, P., AND FELLEISEN, M. Drscheme: a programming environment for scheme. *Journal of Functional Programming 12*, 2 (2002), 159–182.

[13] UNGAR, D., AND SMITH, R. B. Self: The power of simplicity. *SIGPLAN Not. 22*, 12 (Dec. 1987), 227–242.