В. С. Белошедов (ГГУ имени Ф. Скорины, Гомель)
Науч. рук. **Е. А. Дей,** канд. физ.-мат. наук, доцент

ИСПОЛЬЗОВАНИЕ ABSTRACT SYNTAX TREE В КОМПИЛЯТОРАХ И ИНТЕРПРЕТАТОРАХ JAVASCRIPT

Современные тенденции разработки программного обеспечения показывают, что веб-браузер является самым популярным способом взаимодействия человека и электронных устройств. Нередко можно встретить случаи, когда разработчики разрабатывают приложение доступное в браузере для любого современного устройства. Современные банки предлагают услуги интернет-банкинга, которые доступны через браузер компьютера, телефона, планшета и даже телевизора.

Возникает задача: как можно разработать приложение, которое должно работать одинаково на всех устройствах с разными версиями браузеров? Что, если мы хотим не только использовать современный синтаксис Javascript, но и поддерживать доступность приложения в старых браузерах Internet Explorer 11 или старых версиях Google Chrome, в которых отсутствует реализация последних стандартов ES?

Для этого необходимо использовать компиляторы (или как еще называют транспайлеры) для Javascript.

Одним из таких популярных транспайлеров является Babel. Одна из главных задач Babel состоит в том, чтобы дать разработчику возможность использовать новейший синтаксис Javascript, который даже может являться proposal (предложением) и не волноваться, что код не запустится в старом браузере. Сам Babel состоит из большого числа пакетов, которые можно использовать для различных целей.

Но как Babel способен понимать и переводить код? Для этого он использует абстрактное синтаксическое дерево (Abstract Syntax Tree, AST). Иногда эту структуру данных называют просто синтаксическое дерево. Особенностью такого дерева является то, что его листья являются операндами языка, а внутренние вершины – операторами.

На рисунке 1 отображена простая программа, в которой реализована функция square, возвращающая квадрат от числа и вызов функции square с n = 2. Листья дерева представляют собой операнды – в случае функции square операнд n. В случае вызова функции операн-

дом является число 2. Внутренние вершины – операнды. Программа начинается с вершины Program.

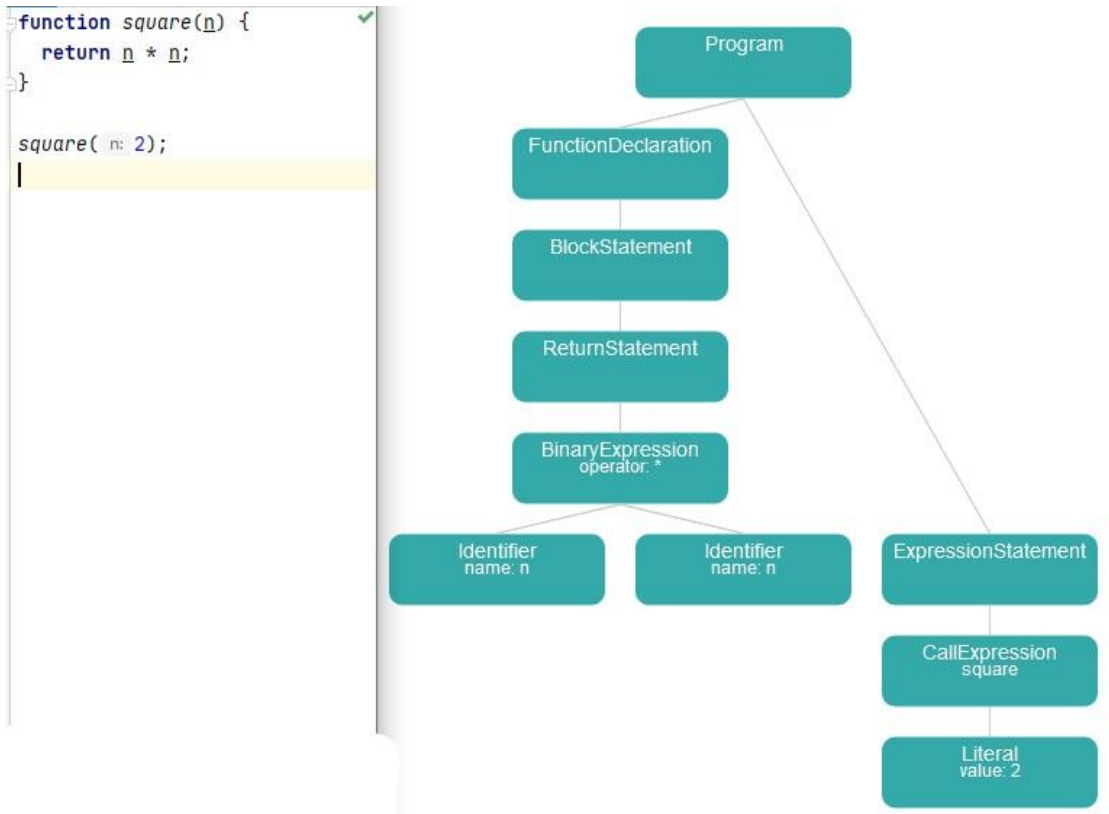

Рисунок 1 – Пример исходного кода и визуализация AST дерева

Следует отметить, что это лишь визуализация, само дерево мы можем получить как JSON-объект при парсинге программы.

Существует несколько реализаций парсера Javascript. Они могут отличаться какими-то внутренними реализациями, количеством информации, которое они выдают как результат, принципами и скоростью работы. Из популярных можно отметить такие как: Acorn, Esprima, Recast, Tenko и другие.

Babel использует свой собственный парсер, раньше он назывался Babylon, но теперь его можно найти под названием @babel/parser. Он находится в открытом доступе.

Точно такие же AST деревья используются и в интерпретаторе Javascript. Как известно, во многих движках используется Just-In-Time компилятор для исходного кода.

Таким образом, абстрактные синтаксические деревья используются широко для оптимизации и транспайлеринга в Javascript.

Рисунок 2 – Пример AST дерева

Литература

1. Дакетт, Дж. JavaScript и jQuery. Интерактивная веб-разработка / Дж. Дакетт. – 1-е издание. – Эксмо, 2016. – 640 с.

2. Роббинс, Дж. HTML5, CSS3 и JavaScript. Исчерпывающее руководство / Дженнифер Роббинс; [пер. с англ. М. А. Райтман]. – 4-е издание. – М.: Эксмо, 2014. — 528 с. + DVD. – (Мировой компьютерный бестселлер).