ФГБОУ ВО «Пермский национальный исследовательский государственный университет»

Физический факультет
Кафедра компьютерных систем и телекоммуникаций

Классы и наследование, транспиляция в Babel и TypeScript.

Работу выполнила студентка 2 курса

Направление «Радиофизика»
Профиль магистратуры «Информационные процессы и системы»
Дегтянникова Дарья Александровна

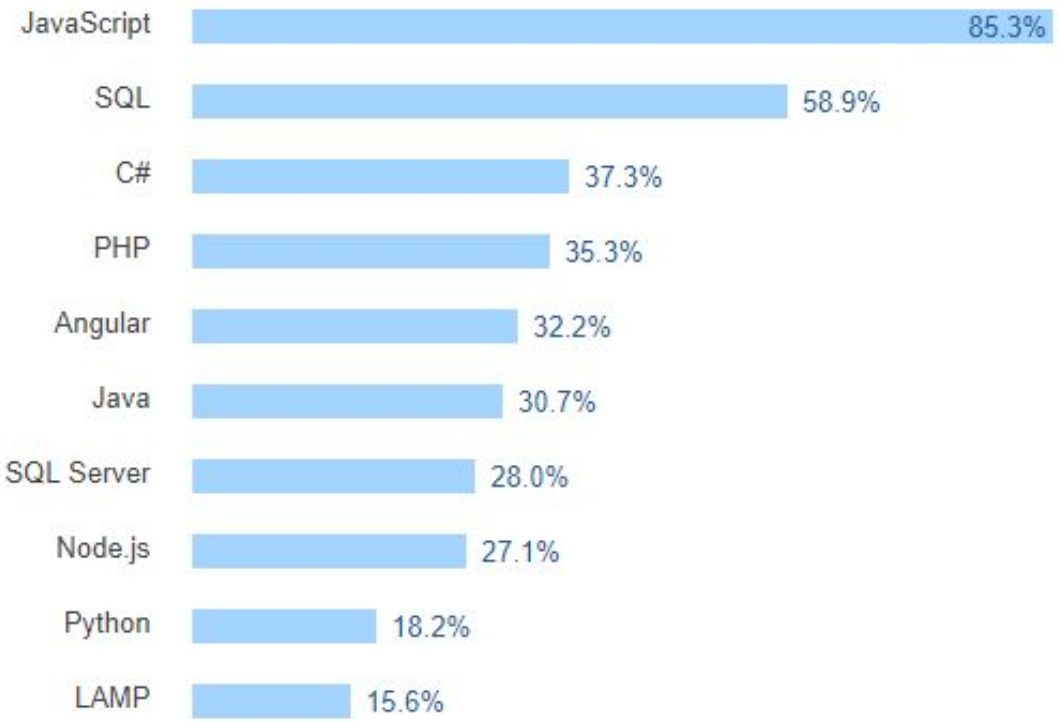
Пермь, 2019

Почему JavaScript?

- большое комьюнити;
- мало аналогов;
- много примеров и уже готовых решений;
- быстрый темп роста популярности вэба;
- низкий порог вхождения(справится даже не специалист);
- много библиотек;
- нет нужды в компиляторе(для старта достаточно блокнота и браузера).

Most Popular Technologies per Dev Type

Full-Stack | Front-End | Back-End | Mobile | Math & Data | Students



https://insights.stackoverflow.com/survey/2016

Понятие транспиляции

- Транспиляция - процесс преобразования кода, написанного с помощью следующих версий языка или на диалектах JavaScript в некий стандартный вариант, понимаемый всеми браузерами.
- Транспиляция - компиляция из одного языка в другой язык с таким же набором абстракций, либо в другую версию того же языка.
- **ECMAScript** — это встраиваемый расширяемый не имеющий средств ввода-вывода язык программирования, используемый в качестве основы для построения других скриптовых языков
- Зачем:
  - Использование ECMAScript6(ES6)
  - Желание писать для вэба не на JS

Babel(6to5)

Запуск кода на платформе, не поддерживающей новый синтаксис, приведет к синтаксической ошибке. Закономерным решением этой проблемы стало появление Babel — программы, которая берет указанный код и возвращает тот же код, но транслированный в старую версию JS.

Состоит из двух частей:

1. Транспайлер, который переписывает код.

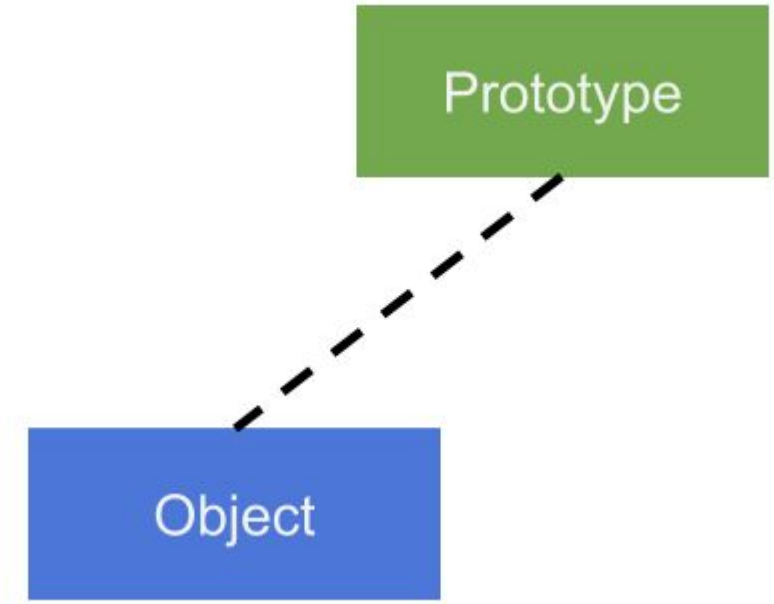
2. Полифилл, который добавляет методы.

«Полифилл» (англ. polyfill) – это библиотека, которая добавляет в старые браузеры поддержку возможностей, которые в современных браузерах являются встроенными. Работает она через модификацию стандартных объектов и их прототипов.

Прототипы объектов

В JavaScript не используется традиционная система наследования, основанная на классах. Для решения схожих задач используются прототипы.

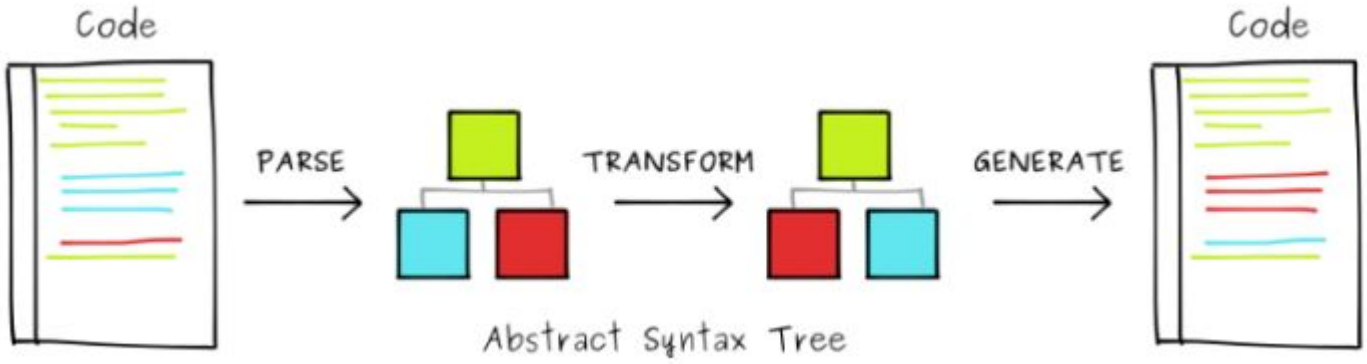
Каждый объект связан с ещё одним объектом — со своим прототипом. Когда вы пытаетесь обратиться к свойству или методу объекта, поиск того, что вам нужно, сначала выполняется в самом объекте. Если поиск не увенчался успехом, он продолжается в прототипе объекта.



Babel(6to5)

Принцип действия:

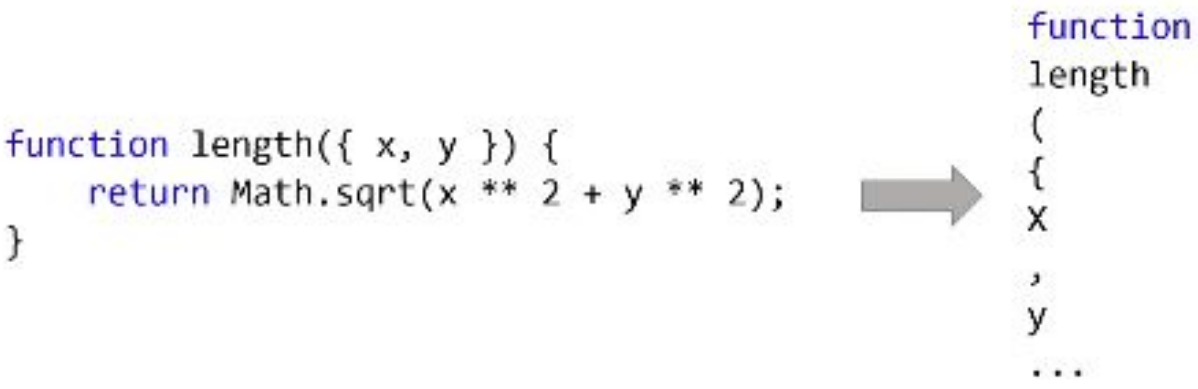
1. Загрузка JS на новом стандарте или на старом стандарте со странными спецификациями;
2. Вызов транспилятора (передача параметров);
3. Получение JS другого стандарта(чаще ES5).



Лексический анализ

На шаге лексического анализа происходит разбор текста на логические сущности – токены:

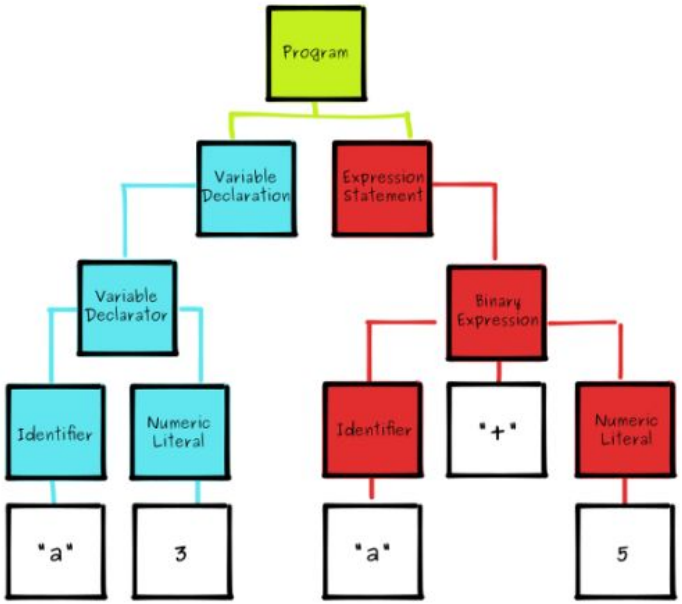
- числа;
- строк;
- идентификаторы;
- строки и т.д.



Синтаксический анализ

На шаге синтаксического анализа происходит создание абстрактного синтаксического дерева(AST) из потока токенов.

```
var a = 3;
a + 5
```



https://astexplorer.net/

Синтаксический анализ

```
Math
.
sqrt
(
x
**
2
+
y
**
2
)
```

```
- argument: CallExpression {
   - callee: MemberExpression {
      - object: Identifier {
           name: "Math"
        }
      - property: Identifier {
           name: "sqrt"
        }
        computed: false
     }
   - arguments: [
      - BinaryExpression = $node {
         + left: BinaryExpression {left, operator, right}
           operator: "+"
         + right: BinaryExpression {left, operator, right}
        }
     ]
```

Синтаксический анализ

```
x
**
2
```

```
- left: BinaryExpression {
   - left: Identifier = $node {
        name: "x"
     }
     operator: "**"
   - right: NumericLiteral {
      + extra: {rawValue, raw}
        value: 2
     }
  }
```

Трансформация

Преобразование одного синтаксического дерева в другое

```
x ** 2
```

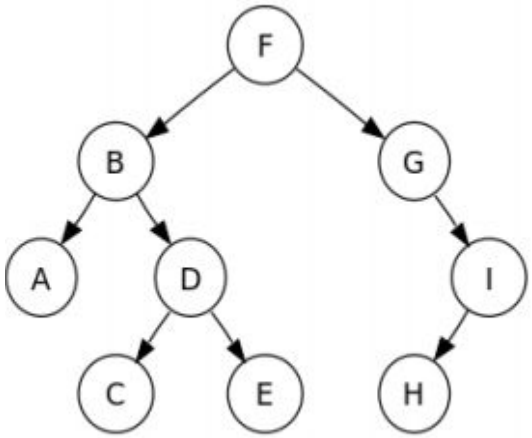
```
Math.pow(x, 2)
```

```
- left: BinaryExpression {
   - left: Identifier = $node {
        name: "x"
     }
     operator: "**"
   - right: NumericLiteral {
      + extra: {rawValue, raw}
        value: 2
     }
 }
```

```
- left: CallExpression {
   - callee: MemberExpression {
      - object: Identifier {
           name: "Math"
        }
      - property: Identifier {
           name: "pow"
        }
        computed: false
     }
   - arguments: [
      - Identifier {
           name: "x"
        }
      - NumericLiteral {
         + extra: {rawValue, raw}
           value: 2
```

Генерация

Перевод готовой AST обратно в текст




```
"use strict";

function length(_ref) {
    var x = _ref.x,
        y = _ref.y;

    return Math.sqrt(Math.pow(x, 2) +
                     Math.pow(y, 2));
}
```

TypeScript

TypeScript — язык программирования, позиционируемый как средство разработки веб-приложений, расширяющее возможности JavaScript.

TypeScript отличается от JavaScript:

- возможностью явного статического назначения типов;
- поддержкой использования полноценных классов (как в традиционных объектно-ориентированных языках);
- поддержкой подключения модулей.

TypeScript это расширение языка ECMAScript 5.

TypeScript

TypeScript предлагает новый синтаксис для написания JS-приложений. Он использует аннотации типов, чтобы избавиться от некоторых проблем с типизацией в JavaScript.

```
function greeter(person: string) {
    return "Hello, " + person;
}

var user = "Jane User";

document.body.innerHTML = greeter(user);
```

TypeScript

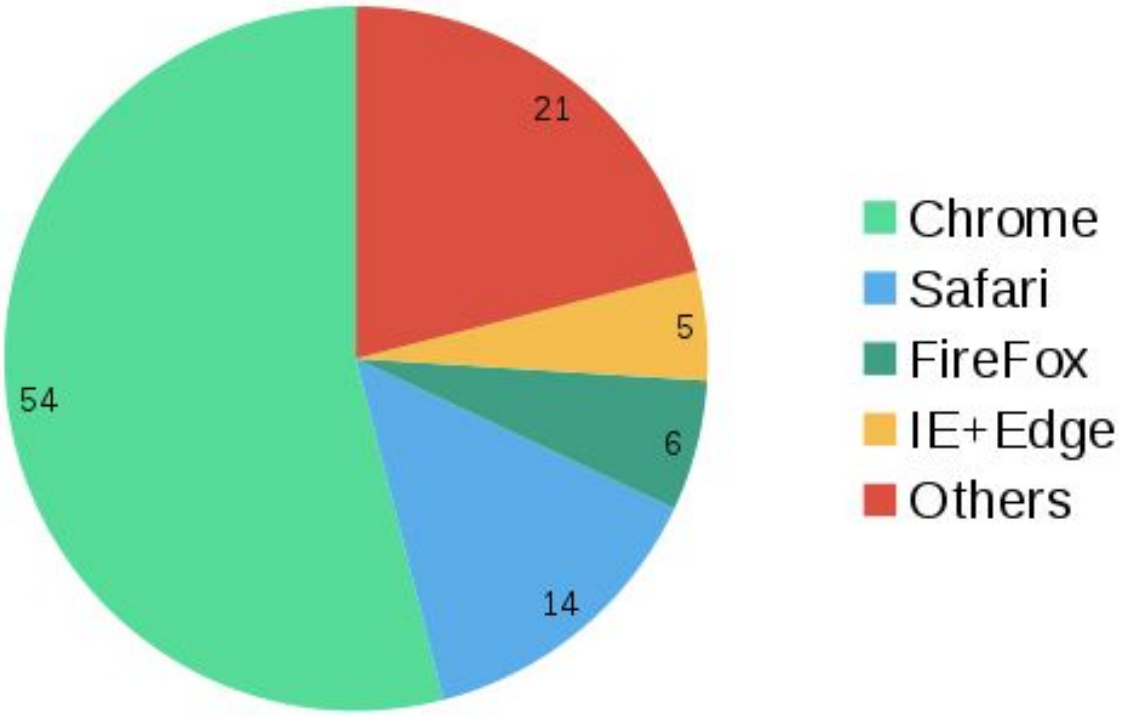
```
function greeter(person) {
    return "Hello, " + person;
}

var user = "Jane User";

document.body.innerHTML = greeter(user);
```

ES5

Почему это не понадобится в ближайшее время?

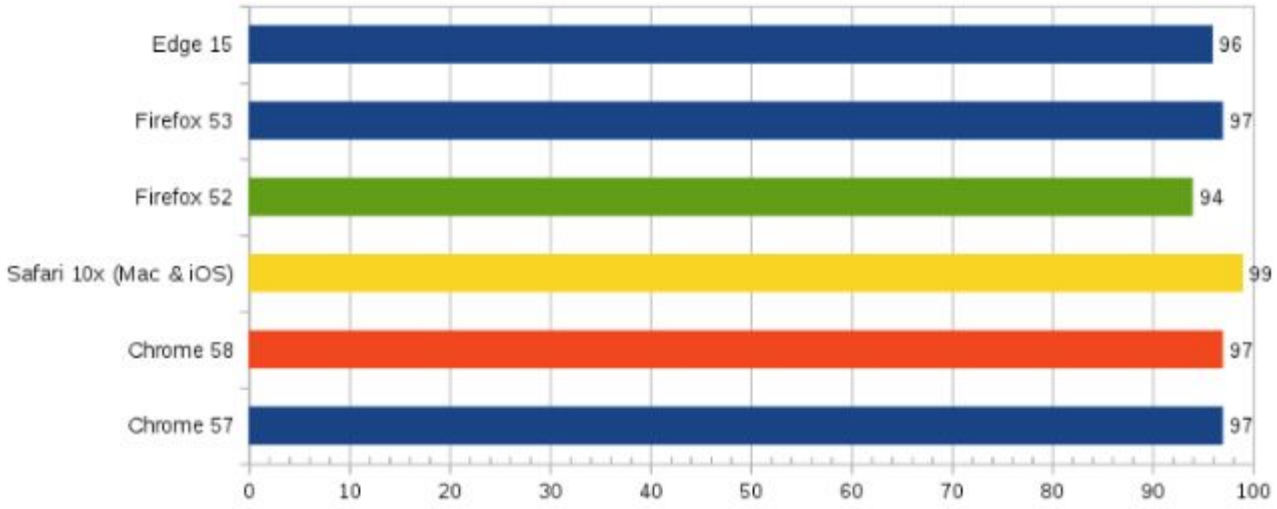


Доля рынка браузеров по состоянию на май 2017 года в%

Почему это не понадобится в ближайшее время?

Поддержка классов появилась в браузере Chrome в 2014 году. Это позволяет браузеру работать с объявлениями классов без применения транспиляции или каких-либо вспомогательных библиотек.

На самом деле, поддержка этих механизмов браузерами — не более чем «синтаксический сахар». Эти конструкции преобразуются в те же базовые структуры, которые уже поддерживаются языком. В результате, даже если пользоваться новым синтаксисом, на более низком уровне всё будет выглядеть как создание конструкторов и манипуляции с прототипами объектов.



Все основные браузеры имеют очень хорошую поддержку ES6

Почему это не понадобится в ближайшее время?

https://kangax.github.io/compat-table/es6/