УДК 004

Преимущества использования стандарта ES6 для разработки на языке JavaScript

Круглик Роман Игоревич
Приамурский государственный университет им. Шолом-Алейхема
Студент

Аннотация
В статье рассматриваются преимущества использования стандарта ES6 на языке JavaScript. Для примера было проведено сравнение 2 способов.
Ключевые слова: ES6, программирование, JavaScript, сравнение.

Benefits of using the ES6 standard for JavaScript development

Kruglik Roman Igorevich
Sholom-Aleichem Priamursky State University
Student

Abstract
In article discusses the benefits of using the ES6 standard in JavaScript. For example, a comparison was made of 2 methods.
Keywords: ES6, programming, JavaScript, comparison.

Интернет представляет собой массу возможностей, за которые ухватываются веб-разработчики. Веб-разработка имеет свои инструменты для воплощения идей в жизнь. Один из них - язык программирования JavaScript.

JavaScript - мультипарадигменный язык программирования. Поддерживает объектно-ориентированный, императивный и функциональный стили.

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты:

1. динамическая типизация;
2. автоматическое управление памятью;
3. прототипное программирование;
4. функции как объекты первого класса.

Так же, как и все языки программирование он имеет свои стандарты. Стандартизация проходит через компанию ECMA, а сам стандарт называется ECMAScript (ES).

Многие разработчики даже не знают о таких стандартах, но они помогают программист более аккуратно и быстро писать код.

Так же все современные браузеры и среды исполнения уже поддерживают ES6. Chrome, MS Edge, Firefox, Safari, Node и многие другие системы имеют встроенную поддержку большинства возможностей JavaScript ES6 (см. рис.1).

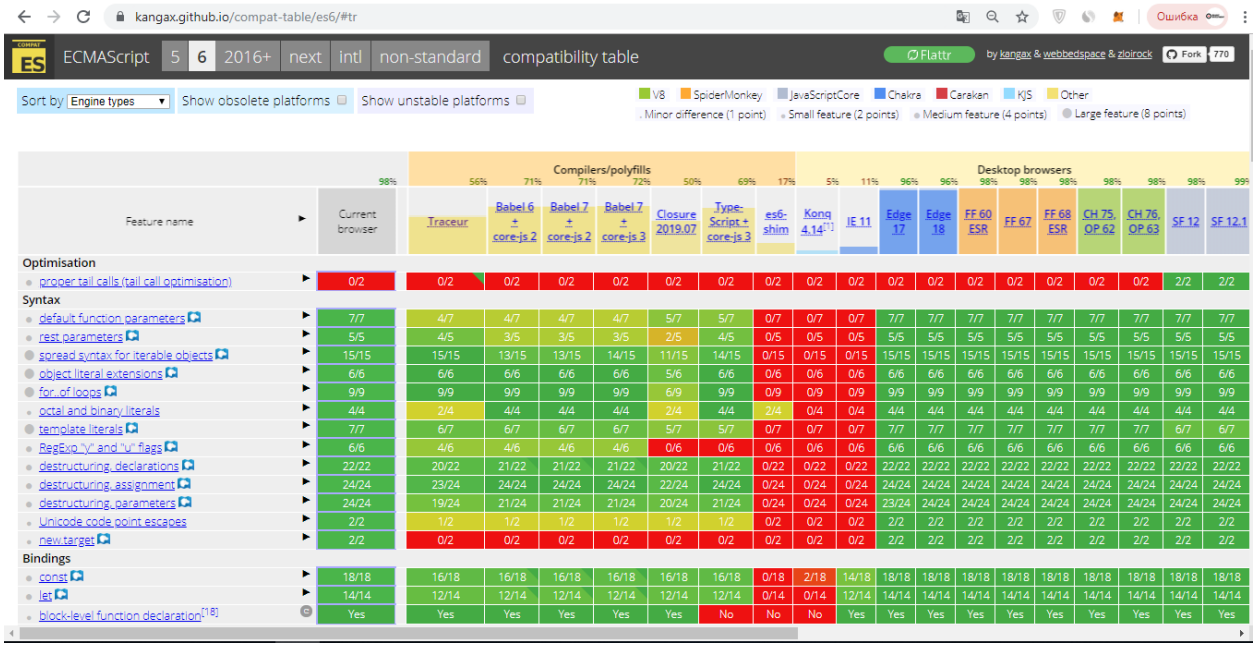

Рисунок 1. Поддержка ES6 браузерами

Областью обновления и стандартизации языка JavaScript интересуются многие. В статье П.А. Васильев [1] рассказывается об обновление языка программирования JavaScript (ES5). В статье А. Майоров [2] приводятся примеры шаблонных строк, которые используются в ES6. В работе К. Сухов [3] рассказывает о том, какое будущее ждёт Javascript.

В данной статье представлено сравнение 2 стандартов. Показаны основные преимущества использования стандарта ES6 для разработки на языке JavaScript.

В JavaScript используется переменная var. Рассмотрим первое сравнение (см. рис. 2).

```
var x = 'outer';
function test(inner) {
    if (inner) {
        var x = 'inner';
        return x;
    }
    return x;
}

test(false);
test(true);

let x = 'outer';
function test(inner) {
    if (inner) {
        let x = 'inner';
        return x;
    }
    return x;
}

test(false);
test(true);
```

Рисунок 2. Сравнение двух способов

В первом варианте мы хотели получить ответ «girl», но получаем «undefined», потому что var проникает через блоки и перед выводом переименовывает переменную x. Для этого и существует стандарт ES6, который включает в себя переменную let. Теперь если у нас условие не выполнено, браузер пойдёт дальше и будет именно тот результат, который нам нужен (см. рис. 3).

```
undefined                    (index):19
outer                        (index):34
>
```

Рисунок 3. Результат сравнения

Так же существует const, но её необходимо использовать если переменная будет определена всего 1 раз.

Все стандарты с каждый обновлением, стараются упростить и ускорить разработку, так и в следующем примере (см. рис. 4).

```
<script>
    var firstname1 = 'Иван';
    var lastname2 = 'Иванов';
    console.log('Тебя зовут ' + firstname1 + ' ' + lastname2 + '.');

    const firstname = 'Иван';
    const lastname = 'Иванов';
    console.log(`Тебя зовут ${firstname} ${lastname}.`);
</script>
```

Рисунок 4. Сравнение двух способов

Оба примера выполняют одно и тоже, но второй вариант выполнен с помощью стандарта ES6 и является более универсальней (см. рис. 5).

```
Тебя зовут Иван Иванов.          (index):15
Тебя зовут Иван Иванов.          (index):19
>
```

Рисунок 5. Результат сравнения

Так же были введены бэктики, которые увеличили скорость программирования (см. рис. 6).

```
var template = '<li>\n' +
    '  <div class="view">\n' +
    '    <input class="toggle" type="checkbox">\n' +
    '    <label></label>\n' +
    '    <button class="destroy"></button>\n' +
    '  </div>\n' +
    '  <input class="edit" value="">\n' +
    '</li>';
console.log(template);

const template = `<li>
  <div class="view">
    <input class="toggle" type="checkbox">
    <label></label>
    <button class="destroy"></button>
  </div>
  <input class="edit" value="">
</li>`;
console.log(template);
```

Рисунок 6. Сравнение двух способов

Теперь не нужно обрабатывать каждую строку отдельно, а всего лишь обернуть HTML код в бэктики и результаты будут одинаковы (см. рис. 7).

```
<li>                                                    (index):21
  <div class="view">
    <input class="toggle" type="checkbox">
    <label></label>
    <button class="destroy"></button>
  </div>
  <input class="edit" value="">
</li>
<li>                                                    (index):31
      <div class="view">
        <input class="toggle" type="checkbox">
        <label></label>
        <button class="destroy"></button>
      </div>
      <input class="edit" value="">
</li>
>
```

Рисунок 7. Результат сравнения

Работа с массивами так же была упрощена (см. рис. 8).

```
var array = [1, 2, 3, 4];

var first = array[0];
var third = array[2];

console.log(first, third); // 1 3

const array2 = [1, 2, 3, 4];

const [first2, ,third2] = array2;

console.log(first2, third2); // 1 3
```

Рисунок 8. Сравнение двух способов

Теперь можно таким способом определять массивы. Результат так же будет одинаков (см. рис. 9).

```
1 3                                                     (index):18
1 3                                                     (index):24
>
```

Рисунок 9. Результат сравнения

Так же в ES6 перешли от «функций» к «классам» (см. рис. 10).

```
var Animal = (function () {
    function MyConstructor(name) {
        this.name = name;
    }
    MyConstructor.prototype.speak = function speak() {
        console.log(this.name + ' прыгает');
    };
    return MyConstructor;
})();

var animal = new Animal('Зайчик');
animal.speak();

class Animall {
    constructor(name) {
        this.name = name;
    }
    speak() {
        console.log(this.name + ' прыгает');
    }
}

const animall = new Animall('Зайчик');
animall.speak();
```

Рисунок 10. Сравнение двух способов

Намного удобнее использовать конструкторы, нежели определять функции. Результат будет одинаковый (см. рис. 11).

```
Зайчик прыгает                    (index):18
Зайчик прыгает                    (index):31
>
```

Рисунок 11. Результат сравнения

В результате были рассмотрены те изменения, которые используются почти в каждом приложении или системе написанной при помощи языка JavaScript. Данный работа станет основой для более глубокого изучения стандарта ES6.

Библиографический список

1. Васильев П.А. Обновление языка программирования JavaScript (ES5) - ECMAScript 2015 (ES6) // Современные инновации. 2016. № 12 (14). С. 45-46.
2. Майоров А. Шаблонные строки в ES6. больше, чем строки // Системный администратор. 2016. № 4 (161). С. 72-74.
3. Сухов К. Javascript - есть будущее // Системный администратор. 2014. № 11 (144). С. 65-73.