

СТАНДАРТЫ И ВЕРСИИ



ДЕНИС ЕЖКОВ



ДЕНИС ЕЖКОВ

Frontend-разработчик в «Ростелеком IT»



ПЛАН ЗАНЯТИЯ

1. Редакции и стандарты: основные изменения в версиях ES5-ES9
2. ES5: `var`, ES6: `let`, `const`
3. Синтаксис функций в привязке к стандартам ES5 / ES6
4. ES6: перебор итерируемых объектов с помощью `for...of`
5. ES6: тегированные шаблонные строки

РЕДАКЦИИ И СТАНДАРТЫ: ОСНОВНЫЕ ИЗМЕНЕНИЯ В ВЕРСИЯХ ES5-ES9



ЗАЧЕМ ЭТО ЗНАТЬ?

1. Точно знать какие именно возможности JavaScript можно использовать в конкретном проекте
2. Использовать новейшую версию языка, предоставляющую больше возможностей и содержащую меньше недостатков в дизайне языка

КАК ИСПОЛЬЗОВАТЬ?

Чтобы все самые крутые фишки новых стандартов были доступны, можно использовать, например, транспайлер. Самый популярный транспайлер на сегодняшний день — [Babel](#)*.

The word "BABEL" is written in a bold, yellow, hand-drawn style with black outlines and brushstroke textures. The letters are slanted and have a dynamic, energetic feel.

* Рассмотрим на следующей лекции.

РЕДАКЦИИ, СТАНДАРТЫ, ВЕРСИИ...?

1. Ecma International — организация, которая создает стандарты для технологий
2. **ЕСМА-262** — это **стандарт**, изданный Ecma International (в нём прописана спецификация скриптового языка общего назначения)
3. **ЕСМА-262** можно считать учётным номером **ECMAScript**
4. **ES1-ES9** — это **редакции (версии)** стандарта ЕСМА-262

ECMAScript — стандарт, а JavaScript — самая популярная реализация этого стандарта.

ВЕРСИИ ECMASCRIPT

- **ES1:** 1997
- **ES2:** Июнь 1998
- **ES3:** Декабрь 1999
- **ES4:** так и не была принята
- **ES5:** Декабрь 2009
- ES6 === ES2015
- ES7 === ES2016
- ES8 === ES2017
- ES9 === ES2018
- ES10 === ES2019 (ещё не принят)

ES5

- Strict mode – специальная директива `"use strict"` указывается для перевода кода в режим полного соответствия ES5 (с отсутствием полной обратной совместимости)
- объект `JSON` с методами `parse`, `stringify`
- новые методы `Array` (`indexOf`, `lastIndexOf`, `forEach`, `map`, `filter`, `reduce`)
- новые методы `Object` и др.

ES6 / ES2015

- `let`, `const`
- стрелочные функции
- параметры по умолчанию
- `spread` / `rest` оператор
- деструктуризация массивов и объектов
- тегированные шаблонные строки
- итераторы и генераторы
- `Promise`
- новый синтаксис для классов
- тип данных `Symbol`
- контейнерные типы: `Map`, `WeakMap`, `Set`, `WeakSet`

ES7 / ES2016

- Метод `includes` для класса `Array`
- оператор для возведения в степень `**` (вместо `Math.pow`)

ES8 / ES2017

- Конструкция `async / await`
- `Object.values()` - функция, которая возвращает все значения собственных свойств объекта, исключая любые значения в цепочке прототипов
- `Object.entries()` - метод, который возвращает ключи в виде массива в формате `[key, value]`
- дополнение строк до заданной длины:
`String.prototype.padStart()` / `String.prototype.padEnd()`



ES9 / ES2018

- Разделяемая память и атомарные операции
- `Promise.prototype.finally()`
- `for-await-of` для создания циклов, работающих с асинхронным кодом
- устранение некоторых ограничений тегированных шаблонных строк
- некоторые новые возможности работы с регулярными выражениями

ES10 / ES2019*

- Объявления полей в классах
- приватные поля и методы
- класс `BigInt`
- `Array.prototype.flat`, `Array.prototype.flatMap`
- некоторые новые возможности работы с регулярными выражениями

* Стандарт ещё не принят, возможны изменения.

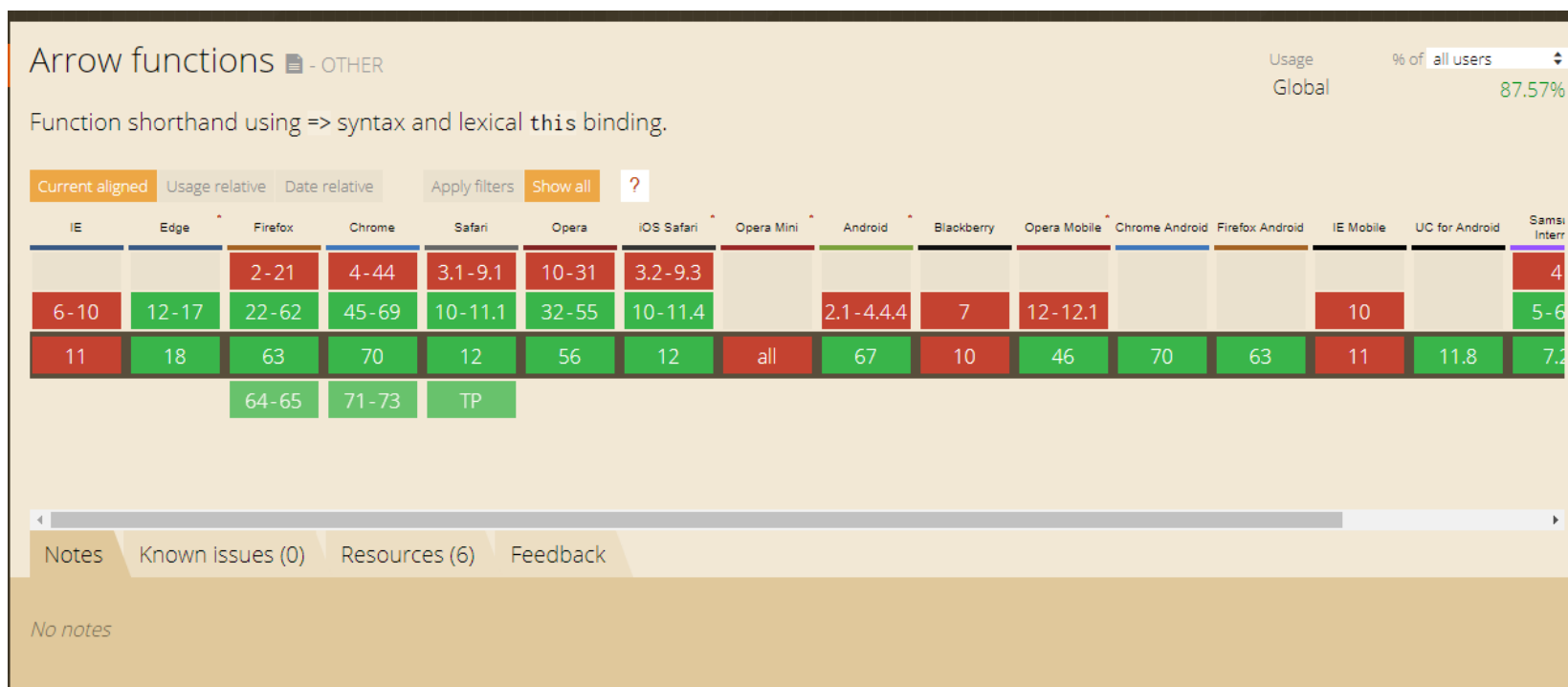


ES.NEXT

Динамический указатель на последнюю, еще находящуюся в разработке версию ECMAScript.

ГДЕ УЗНАТЬ ПРО ПОДДЕРЖКУ?

- <https://caniuse.com>
- <https://developer.mozilla.org>



Поддержка стрелочных функций в браузерах, сайт caniuse.com



ГДЕ УЗНАТЬ ПРО СТАНДАРТИЗАЦИЮ?

<https://github.com/tc39/ecma262>



НА КАКОЙ ВЕРСИИ ЯЗЫКА ПИСАТЬ?

ES6+

Возможности ES6 поддерживаются последними версиями практически всех браузеров (при необходимости использовать Babel).

[illegible]

ES5: `var`,

ES6: `let`, `const`

ОБЪЯВЛЕНИЕ ПЕРЕМЕННЫХ. ES5

В стандарте ES5 переменную можно было объявить только одним способом:

```
var a = 10;
```

ОБЪЯВЛЕНИЕ ПЕРЕМЕННЫХ. ES6

В стандарте ES6 объявить переменные можно как `const` или `let`:

```
const a = 10;
```

```
let a = 10;
```

ЧЕМ ОТЛИЧАЮТСЯ `const` ОТ `let` ?

Варианты ответа:

1. Функция не может быть объявлена как `const`
2. `const`, в отличие от `let`, создает неизменяемую переменную
3. Имеют разные области видимости
4. Переменные, объявленные как `const`, обязательно должны иметь быть названы верхнем регистре

const

`const` используют для объявления «констант», которые не будут в дальнейшем изменяться:

```
1  const numberOfDays = 31;  
2  // Uncaught TypeError: Assignment to constant variable  
3  numberOfDays = 28;
```


НЕЛЬЗЯ ПЕРЕОПРЕДЕЛИТЬ

Переопределение объекта запрещено:

```
1  const numberOfDaysInMonths = {  
2    november: 30,  
3    december: 31,  
4  };  
5  
6  //Uncaught TypeError: Assignment to constant variable  
7  numberOfDaysInMonths = {  
8    november: 30,  
9  };
```

МОЖЕМ ИЗМЕНИТЬ ИЛИ ДОБАВИТЬ СВОЙСТВО

Но можем изменить какое-то свойство или добавить новое:

```
numberOfDaysInMonths.january = 31;  
console.log(numberOfDaysInMonths.january); // 31
```

const C МАССИВАМИ

```
1  const numberOfDaysInMonths = [  
2      31, 28, 31, 30, 31, 30,  
3      31, 31, 30, 31, 30, 31,  
4  ];  
5  
6  // Uncaught TypeError: Assignment to constant variable  
7  numberOfDaysInMonths = [31, 28, 31];
```

ВОПРОС

Что будет выведено в стр. 3?

```
1 | const numbersArray = [1, 2, 3, 4, 5];  
2 | numbersArray[0] = 10;  
3 | console.log(numbersArray);
```

ОТЛИЧИЯ `var` ОТ `let` / `const`

Область видимости

- Переменная, объявленная как `var`, доступна в функции, в которой объявлена
- Переменная, объявленная как `let`, доступна только в рамках блока `{...}`, в котором объявлена. В качестве блока могут выступать: функции, `if`, `while` или `for` и др.

ВОПРОС

Что будет выведено (стр. 5, 7)?

```
1  var isExamPassed = false;
2  var isGoodStudent = true;
3  if (isGoodStudent) {
4      var isExamPassed = true;
5      console.log(isExamPassed); // ?
6  }
7  console.log(isExamPassed); // ?
```

OTBET

```
1  var isExamPassed = false;
2  var isGoodStudent = true;
3  if (isGoodStudent) {
4      var isExamPassed = true;
5      console.log(isExamPassed); // true
6  }
7  console.log(isExamPassed); // true
```

ВОПРОС

Что будет выведено в этом случае?

```
1  let isExamPassed = false;
2  let isGoodStudent = true;
3  if (isGoodStudent) {
4      let isExamPassed = true;
5      console.log(isExamPassed); // ?
6  }
7  console.log(isExamPassed); // ?
```


OTBET

```
1  let isExamPassed = false;
2  let isGoodStudent = true;
3  if (isGoodStudent) {
4      let isExamPassed = true;
5      console.log(isExamPassed); // true
6  }
7  console.log(isExamPassed); // false
```

let В ЦИКЛЕ

Для каждой итерации создаётся своя переменная. Сравним:

```
1 // выведет цифры от 0 до 9
2 for (let i = 0; i < 10; i++) {
3     setTimeout(() => console.log(i), 1000);
4 }
5
6 // выведет 10 раз цифру 10
7 for (var i = 0; i < 10; i++) {
8     setTimeout(() => console.log(i), 1000);
9 }
```



СИНТАКСИС ФУНКЦИЙ В ПРИВЯЗКЕ К СТАНДАРТАМ ES5 / ES6

СИНТАКСИС ФУНКЦИЙ В ES5

В ES5 функцию можно объявить следующим образом:

```
1 // объявление функции (Function Declaration)
2 function multiply(a, b) {
3     return a * b;
4 }
5
6 // функциональное выражение (Function Expression)
7 var multiply = function(a, b) {
8     return a * b;
9 }
```

СТРЕЛОЧНЫЕ ФУНКЦИИ В ES6

В ES6 появляются стрелочные функции:

```
1 // примерный аналог объявления функции выше
2 const multiply = (a, b) => {
3   return a * b;
4 };
5
6 // аналог предыдущей функции,
7 // но return опущен (т.к. подразумевается неявно):
8 const multiply = (a, b) => a * b;
9 // можно добавить скобки для читабельности: (a * b)
```



РЕШЕНИЕ ПРОБЛЕМ

Стрелочные функции позволяют использовать более короткий синтаксис при объявлении и, кроме того, решают проблему [потерянного this](#).

ES6: ПАРАМЕТРЫ ПО УМОЛЧАНИЮ

```
1 // ES5
2 var getTitle = (title) {
3     title = title || 'Безымянный';
4     console.log(title);
5 };
6
7 // ES6
8 const getTitle = (title = 'Безымянный') => {
9     // ...
10    console.log(title);
11 };
```

ES6: ДЕСТРУКТУРИЗАЦИЯ В ПАРАМЕТРАХ

Пользователь кликает на какой-то элемент и функции-обработчику событий `onClick` приходит объект `event`:

```
1 // ES5
2 var onClick2 = function(event) {
3     console.log(event.target, event.type);
4 }
5
6 // ES6
7
8 // event = { target: {...}, type: '...', ... }
9 const onClick = ({ target, type }) => {
10     console.log(target, type);
11 }
```


ES6: ОПЕРАТОР SPREAD (...REST)

Получение массива аргументов при помощи оператора spread (...rest):

```
1  const multiply = (multiplier, ...args) => args.map(  
2    element => multiplier * element,  
3  );  
4  
5  console.log(multiply(2, 1, 2, 3)); // [2, 4, 6]
```

В ЧЁМ РАЗНИЦА?

Разница между стрелочными и классическими функциями в том, что стрелочные функции не имеют своего `this` и `arguments` (получают из окружающего контекста):

```
1  const group = {  
2    groupNumber: 1,  
3    students: ['Иванов', 'Петров', 'Сидоров'],  
4    showList() {  
5      this.students.forEach(student =>  
6        console.log('Группа: ' + this.groupNumber + ', ' + student)  
7      );  
8    }  
9  }
```

ES6: ПЕРЕБОР ИТЕРИРУЕМЫХ ОБЪЕКТОВ С ПОМОЩЬЮ `for...of`



ИТЕРИРУЕМЫЕ ОБЪЕКТЫ

ES6 добавлены «итерируемые» (iterable) объекты, чьё содержимое можно перебрать в цикле (массивы, строки, Map, Set, DOM-коллекции и т.д.).

ES5: ПЕРЕБЕРЁМ МАССИВ

```
1  let numbersAr = [10, 20, 30];
2
3  var double = function() { // только для массивов
4      numbersAr.forEach(function(value) {
5          console.log(value);
6      });
7  };
8
9  var double2 = function() {
10     for (var i = 0; i < numbersAr.length; i++) {
11         console.log(numbersAr[i] * 2);
12     }
13  };
```

ES6: ПЕРЕБЕРЁМ МАССИВ

```
1  let numbersAr = [10, 20, 30];
2
3  const double = () => {
4    for (let value of numbersAr) {
5      console.log(value * 2);
6      // выведет построчно 20, 40, 60
7    }
8  };
```

ПЕРЕБЕРЁМ СТРОКУ

```
1  for (let char of 'лекция') {  
2    console.log(char.toUpperCase());  
3    // выведет Л, Е, К, Ц, И, Я  
4  }
```



ES6: ТЕГИРОВАННЫЕ ШАБЛОННЫЕ СТРОКИ

ШАБЛОННЫЕ СТРОКИ

Для начала вспомним что такое шаблонные строки (ES6):

```
1 //ES5
2 const a = 5;
3 const b = 10;
4 console.log('Сумма: ' + (a + b) + ', разность: ' + (a - b));
5
6 //ES6
7 console.log(`Сумма: ${a + b}, разность: ${a - b}`);
```

Помимо того, что это легче читается, уходит путаница с приведением типов с оператором `+`.



ЗАДАЧА

По количеству баллов, которые студент получил за тест, вывести его оценку в формате: Студент [Фамилия] получил оценку [N].

РЕШЕНИЕ: ФУНКЦИЯ

```
1  const formatMark = (strings, person, numberOfPoints) => {
2    const student = strings[0]; // Студент
3    const points = strings[1]; // получил оценку
4    let mark;
5    if (numberOfPoints <= 60) {
6      mark = '2';
7    } else if (numberOfPoints > 61 && numberOfPoints <= 75) {
8      mark = '3';
9    } else if (numberOfPoints > 76 && numberOfPoints <= 85) {
10     mark = '4';
11   } else if (numberOfPoints > 86) {
12     mark = '5';
13   }
14   return `${student}${person}${points}${mark}`;
15 }
```

РЕШЕНИЕ: ВЫВОД В КОНСОЛЬ

```
1  const person = 'В. Пупкин';
2  const numberOfPoints = 85;
3  // обратите внимание: ниже нет круглых скобок
4  const output = formatMark`Студент ${person} получил оценку ${numberOfPoints}`;
5  console.log(output); // Студент В. Пупкин получил оценку 4
```



ИТОГИ

ЧТО МЫ УЗНАЛИ

- Зачем нужны стандарты JavaScript, основные изменения в версиях ES5-ES9, как узнать поддерживает ли браузер ту или иную новую возможность JavaScript и как использовать свежие возможности
- `var`, `let`, `const`, в чём разница
- как объявлять функции в ES5 и стрелочные функции в ES6
- как перебирать итерируемые объекты с помощью `for...of` (ES6)
- как использовать тегированные шаблонные строки (ES6)



ПОЛЕЗНЫЕ ССЫЛКИ

- [Википедия: Консорциум Всемирной паутины](#)
- [MDN Web Docs: Функции](#)
- [MDN Web Docs: Шаблонные строки](#)



Задавайте вопросы и напишите отзыв о лекции!

ДЕНИС ЕЖКОВ

