

Объявления

- В четверг последняя лекция по вёрстке
- В следующий вторник последняя лекция по JS
- Домашняя работа по JS будет в пятницу

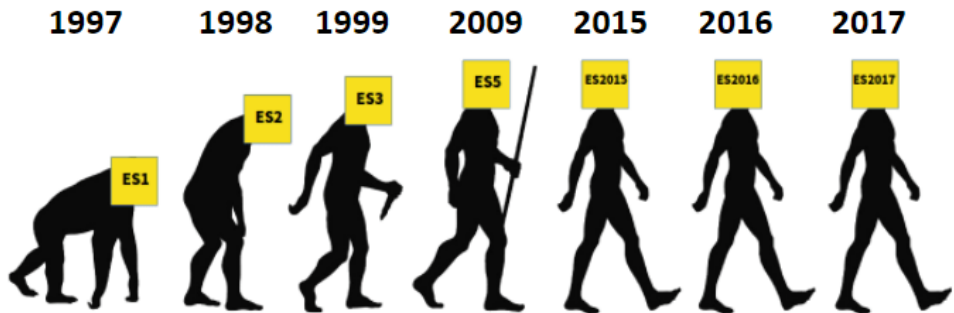
```
lecture.theme
```

ReferenceError: lecture is not defined

Mocha/LiveScript/JavaScript



ECMAScript



TC
39

Эволюция ECMAScript

- ES1, ES2 — первая редакция
- ES3 (1999) — RegExp, try/catch
- ES4 — ...
- ES5 (2009) — строгий режим, геттеры и сеттеры, нативная поддержка JSON, новые методы Array и Object

Пример кода на ES5

```
function Student(firstName, lastName) {  
    this.firstName = firstName;  
  
    this.lastName = lastName;  
};  
  
Object.defineProperty(Student.prototype, "name", {  
    get: function() {  
        return this.firstName + " " + this.lastName;  
    }  
});  
  
Student.prototype.setSkills = function(skills) {  
    this.skills = skills || [];  
};
```

ES2015 (ES6) — глобальное обновление языка

Классы

```
class Student {  
  constructor(firstName, lastName) {  
    this.firstName = firstName;  
    this.lastName = lastName;  
  }  
  
  get name() {  
    return `${this.firstName} ${this.lastName}`;  
  }  
  
  setSkills(skills = []) {  
    this.skills = skills;  
  }  
}
```

ES2015 (ES6) — глобальное обновление языка

let и const

```
function foo() {  
    var apples = 5;  
}  
  
{  
    let apples = 10;  
}  
  
{  
    const apples = 15;  
}
```


ES2015 (ES6) — глобальное обновление языка

Стрелочные функции

// краткий синтаксис

```
const func = x => x * x;
```

// блочный синтаксис

```
const func = (x, y) => { return x + y; };
```

ES2015 (ES6) — глобальное обновление языка

Шаблонные строки

``строка текста``

``строка текста 1
строка текста 2``

``строка текста ${выражение} строка текста``

ES2015 (ES6) — глобальное обновление языка

Деструктуризация массивов

```
var zero, one, two;  
var numbers = ['zero', 'one', 'two'];
```

```
zero = numbers[0];  
one = numbers[1];  
two = numbers[2];
```

```
const numbers = ['zero', 'one', 'two'];  
const [zero, one, two] = numbers;
```

ES2015 (ES6) — глобальное обновление языка

Деструктуризация объектов

```
var p, q;  
var obj = { p: 42, q: true };  
  
p = obj.p;  
q = obj.q;  
  
const obj = { p: 42, q: true };  
const { p, q } = obj;
```

ES2015 (ES6) — глобальное обновление языка

Промисы

```
const promise = new Promise((resolve, reject) => {  
  setTimeout(() => resolve("result"), 1000);  
  setTimeout(() => reject(new Error("ignored")), 2000);  
});  
  
promise.then(  
  result => alert("Fulfilled: " + result),  
  error => alert("Rejected: " + error)  
);
```

ECMAScript 2016

Оператор возведения в степень

```
2 ** 3 // 8  
3 ** 2 // 9  
3 ** 2.5 // 15.588457268119896  
10 ** -1 // 0.1
```

ECMAScript 2016

Метод includes у Array и String

```
const str = "Быть или не быть вот в чём вопрос.";

str.includes("Быть"); // true
str.includes("вопрос"); // true
str.includes("несуществующий"); // false
```

ECMAScript 2017

async/await

```
async function getAuthorAvatar(article) {  
  const userResponse = await fetch(`/articles/${article}`);  
  const { author } = await userResponse.json();  
  
  const avatarResponse = await fetch(`/avatars/${author.id}`);  
  const avatar = await avatarResponse.json();  
  
  return avatar.url;  
}
```


ECMAScript 2018

Операторы Rest и Spread для массивов

```
const numbers = [1, 2, 3, 4, 5];  
const [first, second, ...others] = numbers;  
console.log(others); // [3, 4, 5]
```

```
const others = [3, 4, 5];  
const numbers = [1, 2, ...others];
```

ECMAScript 2018

Операторы Rest и Spread для объектов

```
const numbers = {  
  zero: 1,  
  one: 2,  
  three: 3,  
  four: 4  
};  
  
const { zero, one, ...others } = numbers;  
console.log(others); // { three: 3, four: 4 }  
  
const others = { three: 3, four: 4 };  
const numbers = { zero: 1, one: 2, ...others };
```

ECMAScript 2018

Снова промисы

```
fetch('file.json')  
  .then(data => data.json())  
  .catch(error => console.error(error))  
  .finally(() => console.log('finished'));
```

Дальнейшее развитие ECMAScript



[tc39/ecma262](https://github.com/tc39/ecma262)

Зачем всё это знать?

Статистика использования браузеров










<input checked="" type="checkbox"/>	 Google Chrome	1 885 497 717	41,36 %
<input checked="" type="checkbox"/>	 Яндекс.Браузер	911 129 544	19,99 %
<input checked="" type="checkbox"/>	 Safari	479 313 861	10,52 %
<input checked="" type="checkbox"/>	 Opera	266 301 454	5,84 %
<input checked="" type="checkbox"/>	 Firefox	228 161 202	5,01 %
<input type="checkbox"/>	 Android Browser	137 164 299	3,01 %
<input type="checkbox"/>	 Internet Explorer	113 825 634	2,50 %
<input type="checkbox"/>	 Samsung Internet	92 911 563	2,04 %
<input checked="" type="checkbox"/>	 Edge	76 503 901	1,68 %

Таблица совместимости

<https://kangax.github.io/compat-table>

		Compilers/polyfills							Desktop browsers										Servers/runtimes							Mobile	
		78%	6%		56%	52%	55%	10%	0%	52%	58%	78%	78%	78%	100%	100%	83%	90%	2%	19%	77%	100%	8%	94%	83%	90%	
Feature name		Current browser	Tracur	Babel 6+ core-js	Closure 2018.10	Type-Script + core-js	es7-shim	IE 11	Edge 17	Edge 18	FF 60 ESR	FF 62	FF 63	CH 69, OP 56	CH 70, OP 57	SF 11.1	SF 12	PJS	Node >=6.5 <7 ^[2]	Node >=8.10 <9 ^[2]	Node >=10.13 <11 ^[2]	DUK 2.2	GraalVM 1.0 ^[3]	iOS 11.3	iOS 12		
2016 features																											
• exponentiation (**) operator ↗		▶	3/3	2/3	3/3	3/3	2/3	0/3	0/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	0/3	0/3	3/3	3/3	2/3	3/3	3/3	3/3	
• Array.prototype.includes ↗		▶	3/3	0/3	3/3	2/3	3/3	2/3	0/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	3/3	0/3	3/3	3/3	3/3	0/3	3/3	3/3	3/3	
2016 misc																											
• generator functions can't be used with "new" ↗ ^[7]		▶	Yes	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	
• generator.throw() caught by inner generator ↗ ^[8]		▶	Yes	No	No	Yes	Yes ^[9]	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	
• strict fn w/ non-strict non-simple params is error ^[10]		▶	Yes	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	
• nested rest destructuring, declarations ↗ ^[11]		▶	Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	
• nested rest destructuring, parameters ^[12]		▶	Yes	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	
• Proxy, "enumerate" handler removed ↗ ^[13]		▶	Yes	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
• Proxy internal calls, Array.prototype.includes		▶	Yes	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	No	Yes	Yes	Yes	
2017 features																											
• Object static methods		▶	4/4	0/4	4/4	3/4	4/4	3/4	0/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	0/4	0/4	4/4	4/4	0/4	4/4	4/4	4/4	
• String padding		▶	2/2	0/2	2/2	2/2	2/2	0/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	0/2	0/2	2/2	2/2	0/2	2/2	2/2	2/2	
• trailing commas in function syntax ↗		▶	2/2	0/2	2/2	2/2	2/2	0/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	0/2	0/2	2/2	2/2	0/2	2/2	2/2	2/2	
• async functions ↗		▶	15/15	3/15	3/15	9/15	8/15	0/15	0/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	0/15	0/15	15/15	15/15	0/15	13/15	15/15	15/15	
• shared memory and atomics		▶	0/17	0/17	0/17	0/17	0/17	0/17	0/17	0/17	0/17	0/17	0/17	17/17	17/17	0/17	0/17	0/17	0/17	0/17	17/17	17/17	0/17	17/17	0/17	0/17	
2017 misc																											
• Proxy "ownKeys" handler, duplicate keys for non-extensible targets (ES 2017 semantics) ↗ ^[22]		▶	No	No	No	No	No	No	Yes	Yes	No	No	No	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	
• RegExp "u" flag, case folding		▶	Yes	No	No	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	No	Yes	Yes	Yes	
• arguments.caller removed ↗		▶	Yes	No	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	Yes	Yes	Yes	
2017 annex b																											
• Object.prototype.getter/setter methods		▶	16/16	0/16	16/16	0/16	16/16	0/16	8/16	14/16	14/16	16/16	16/16	16/16	16/16	16/16	16/16	16/16	12/16	10/16	16/16	16/16	16/16	16/16	16/16	16/16	
• Proxy internal calls, getter/setter methods		▶	4/4	0/4	0/4	0/4	0/4	0/4	0/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	0/4	0/4	4/4	4/4	0/4	4/4	4/4	4/4	
• assignments allowed in for-in head in non-strict mode ↗		▶	Yes	Yes	No	No	No	No	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	

Разный уровень поддержки языка



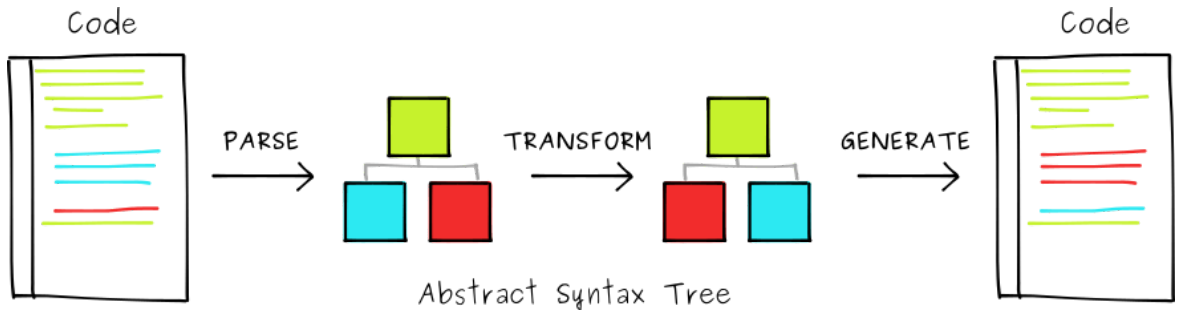
Как решать проблему?

Транспиляция — это процесс перевода исходного кода одного языка в другой



BABEL

Babel Playground



Код на ES2015

```
class Student {  
  constructor(firstName, lastName) {  
    this.firstName = firstName;  
    this.lastName = lastName;  
  }  
  
  setSkills(skills = []) {  
    this.skills = skills;  
  }  
}
```

Код после транспиляции

```
var Student = function () {  
  function Student(firstName, lastName) {  
    _classCallCheck(this, Student);  
  
    this.firstName = firstName;  
    this.lastName = lastName;  
  }  
  
  _createClass(Student, [{  
    key: "setSkills",  
    value: function setSkills() {  
      var skills =  
        arguments.length > 0 &&  
        arguments[0] !== undefined ? arguments[0] : [];  
  
      this.skills = skills;  
    }  
  ]]);  
  
  return Student;  
}();
```

Система плагинов

Plugins

- arrow-functions
- classes
- destructuring
- duplicate-keys
- for-of
- ...

Эксперименты

```
const nestedNumbers = {  
  one: {  
    two: {  
      three: 42  
    }  
  }  
};
```

```
const answer = nestedNumbers?.one?.two?.three; // 42
```

```
const safe = nestedNumbers?.four?.five?.baz; // undefined
```



Что не так?

```
function addNumbers(a, b) {  
  return a + b;  
}
```

```
addNumbers(2, 2); // 4
```

```
addNumbers(2, '2'); // '22'
```

```
addNumbers([], []); // ''
```

```
addNumbers([], {}); // [object Object]
```

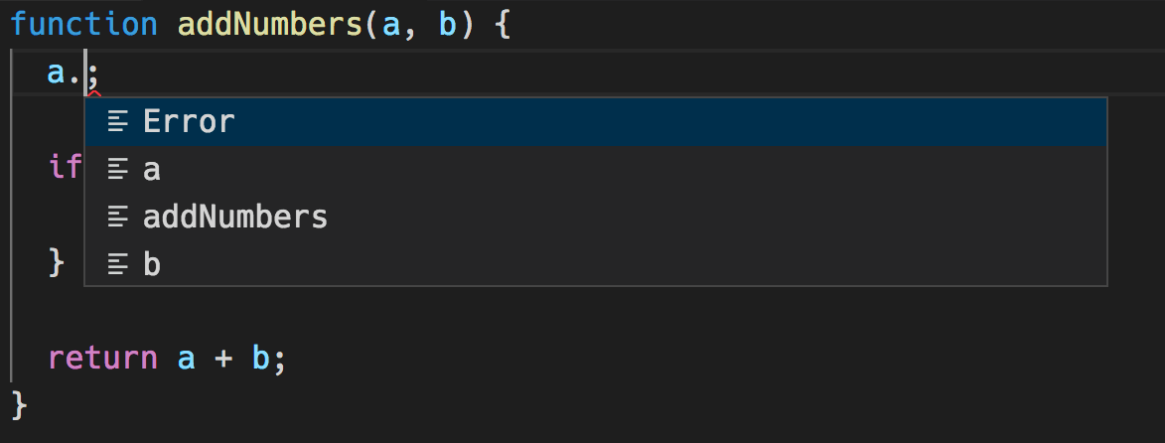
```
addNumbers({}, []); // 0
```


Как исправить?

```
function addNumbers(a, b) {  
  if (typeof a !== 'number' || typeof b !== 'number') {  
    throw new Error();  
  }  
  
  return a + b;  
}
```

Что в IDE?

```
function addNumbers(a, b) {  
  a.;  
  if  
}  
  
return a + b;  
}
```

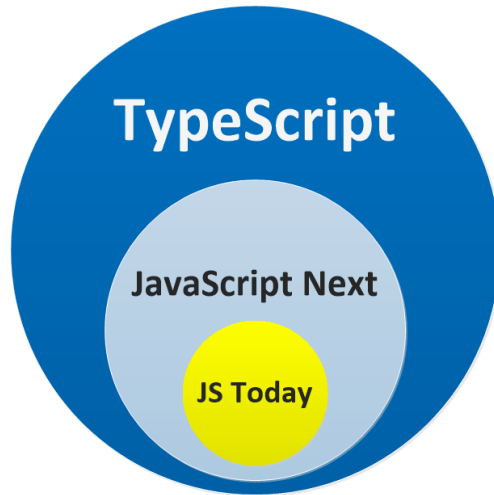


Слабая динамическая типизация

- Неявные преобразования
- Типы становятся известны на этапе выполнения

TypeScript

TypeScript Playground



Автор языка



Андерс Хейлсберг

Известен также по Turbo Pascal, Delphi, C#

Возможности

- ES2015-ESNext
- Транспиляция в ES3, ES5, ES2015, ...
- Аннотации типов и проверка
- Вывод типов
- Интерфейсы
- Обобщённые типы
- Кортежи

Примитивные типы данных

Boolean

```
let isDone: boolean = false;
```

Примитивные типы данных

Number

```
const decimal: number = 6;  
const hex: number = 0xf00d;  
const binary: number = 0b1010;  
const octal: number = 0o744;
```


Примитивные типы данных

String

```
const color1: string = "blue";
```

```
const color2: string = 'red';
```

```
const fullName: string = `Arkady`;
```

```
const sentence: string = `Hello, my name is ${fullName}.`;
```

Примитивные типы данных

null, undefined

```
const u: undefined = undefined;  
const n: null = null;
```

Array

```
const list1: number[] = [1, 2, 3];  
const list2: Array<number> = [1, 2, 3];
```

Tuple

```
// Объявление кортежа
let point: [number, number];

// Некорректная инициализация
point = [10, 'hello']; // Error

// Инициализация
point = [20, 10]; // OK

// Обращение
point[1]; // 10

// Деструктуризация
const [first, second] = point;
```

Enum

```
enum Color {  
    Red,  
    Green,  
    Blue  
}
```

```
const c: Color = Color.Green;
```

Enum

```
enum Color {  
    Red = 5,  
    Green = 7,  
    Blue = 9  
}
```

```
enum Color {  
    Red = '#F00',  
    Green = '#0F0',  
    Blue = '#00F'  
}
```

Object

```
const colors: object = {  
  
  red: '#F00',  
  green: '#0F0',  
  blue: '#00F'  
}
```

```
const settings: {  
  
  color: string;  
  delay: number;  
  retry: boolean;  
} = {  
  color: '#F00',  
  delay: 2000,  
  retry: false  
};
```

Any

```
let notSure: any = 4;  
notSure = "maybe a string instead";  
notSure = false;
```

```
let notSure: any = 4;  
notSure.ifItExists(); // Ошибка в рантайме  
notSure.toFixed(); // Всё ок, toFixed есть в рантайме
```

```
let prettySure: object = 4;  
prettySure.toFixed(); // toFixed нет у object
```


Void и Never

```
function warnUser(): void {  
    console.log("This is my warning message");  
}
```

```
function error(message: string): never {  
    throw new Error(message);  
}
```

```
function infiniteLoop(): never {  
    while (true) {}  
}
```

Приведение типов

```
const someValue: any = 'this is a string';
```

```
const strLength1: number = (<string>someValue).length;
```

```
const strLength2: number = (someValue as string).length;
```

Функции

```
function add(x: number, y: number): number {  
    return x + y;  
}
```

```
const add = (x: number, y: number): number => x + y;
```

Интерфейсы

```
function printLogin(user: { login: string }) {  
    console.log(user.login);  
  
}
```

```
const user = {  
    login: 'savichev'  
};
```

```
printLogin(user);
```

Интерфейсы

```
interface IUser {  
    login: string;  
  
}  
  
function printLogin(user: IUser) {  
    console.log(user.login);  
}  
  
const user = {  
    username: 'savichev'  
};  
  
printLogin(user);
```

Опциональные свойства

```
interface ISquareConfig {  
  color?: string;  
}
```

```
function createSquare(config: SquareConfig) {  
  const newSquare = { color: "white", area: 100 };  
  
  if (config.color) {  
    newSquare.color = config.color;  
  }  
  
  return newSquare;  
}
```

```
let mySquare = createSquare({ color: "black" });
```

Классы

```
interface IMakesSound {  
    makeSound(): void;  
}
```

```
class Python implements IMakesSound {  
    private readonly _length: number;  
  
    public get length() {  
        return this._length / 100;  
    }  
  
    constructor(length: number) {  
        this._length = length;  
    }  
  
    protected makeSound() {  
        console.log('Ssssss!');  
    }  
}
```

Классы

```
abstract class Snake {  
    private readonly _length: number;  
  
    public get length() {  
        return this._length / 100;  
    }  
  
    constructor(length: number) {  
        this._length = length;  
    }  
  
    protected abstract makeSound(): void;  
}
```


Классы

```
class Python extends Snake {  
    private static population = 10000;  
  
    public static incrementPopulation() {  
        Python.population++;  
    }  
  
    constructor(length: number) {  
        super(length);  
        Python.incrementPopulation();  
    }  
  
    protected makeSound() {  
        console.log('Ssssss!');  
    }  
}
```

Классы

```
var Snake = (function() {  
    function Snake(length) {  
        this._length = length;  
    }  
  
    Object.defineProperty(  
        Snake.prototype,  
        "length",  
        {  
            get: function() {  
                return this._length / 100;  
            },  
            enumerable: true,  
            configurable: true  
        }  
    );  
  
    return Snake;  
})();
```

Классы

```
var Python = (function(_super) {
    __extends(Python, _super);

    function Python(length) {
        var _this = _super.call(this, length) || this;
        Python.incrementPopulation();
        return _this;
    }

    Python.incrementPopulation = function() {
        Python.population++;
    };

    Python.prototype.makeSound = function() {
        console.log("Ssssss!");
    };

    Python.population = 10000;

    return Python;
})(Snake);
```

Плагины для IDE


Visual Studio Code

- TSLint




WebStorm

- Всё есть "из коробки"

Почитать

-  TypeScript Handbook
-  TypeScript Deep Dive

Посмотреть

-  Андрей Морозов (Яндекс) — Типизация
-  Андрей Старовойт (JetBrains) — Эволюция TypeScript
-  Константин Кичинский (Microsoft) — Разработка на TypeScript

Спасибо!

Вопросы?