



html academy

интерактивные
онлайн-курсы

Раздел 0×09: Будущее

ES-2016

ECMA-262

7th Edition / June 2016

ECMAScript® 2016 Language Specification

rainsoft.io/must-know-details-about-es2016-features



includes()

```
const countries = ['UK', 'USA', 'Ireland', 'France'];

console.log(countries.includes('UK'));           // true
console.log(countries.includes('Ireland', 1));  // true
console.log(countries.includes('USA', 3));      // false
console.log(countries.includes('France', -2));  // true
```



exponent operator

```
console.log(2 ** 3);    // 8  
console.log(5 ** 2);    // 25  
console.log(0.5 ** 1);  // 0.5  
console.log(0.5 ** 2);  // 0.25  
console.log(100 ** 0);  // 1
```



Исправления и улучшения

- Нельзя создать генератор через *new*
- Убирает оператор *enumerate* из *Proxy*
- Деструктуризация rest-параметров:

```
const sum = (...[a, b, c]) => a + b + c;
```

```
sum(1);           // NaN  
sum(1, 2, 3);     // 6  
sum(1, 2, 3, 4);  // 6
```



ES-2017

ECMA-262

8th Edition

ECMAScript® 2017 Language Specification

tc39.github.io/ecma262



Object.values(); Object.entries();

```
const obj = {a: 5, b: 7, c: 9};
```

```
for (const value of Object.values(obj)) {  
  console.log(value); // "5", "7", "9"  
}
```

```
for (const [key, value] of Object.entries(obj)) {  
  console.log(`${key} ${value}`); // "a 5", "b 7", "c 9"  
}
```



Async / await

```
const resolveAfter2Seconds = (x) => {  
  return new Promise((resolve) => setTimeout(() => resolve(x), 2000));  
};
```

```
const add1 = async (x) => {  
  const a = resolveAfter2Seconds(20);  
  const b = resolveAfter2Seconds(30);  
  return x + await a + await b;  
};
```

```
add1(10).then(v => console.log(v)); // 60 через 2 сек.
```

```
const add2 = async (x) => {  
  const a = await resolveAfter2Seconds(20);  
  const b = await resolveAfter2Seconds(30);  
  return x + a + b;  
};
```

```
add2(10).then(v => console.log(v)); // 60 через 4 сек.
```



Async / await

```
const resolveAfter2Seconds = (x) => {  
  return new Promise((resolve) => setTimeout(() => resolve(x), 2000));  
};
```

```
const iterate = async (num) => {  
  let sum = 0;  
  for (let i = num; i--;) {  
    sum += await resolveAfter2Seconds(10);  
  }  
  return sum;  
};
```

```
console.time(`iterate`);  
iterate(5).then(() => console.timeEnd(`iterate`)); // iterate: ???
```



Async / await

```
const resolveAfter2Seconds = (x) => {  
  return new Promise((resolve) => setTimeout(() => resolve(x), 2000));  
};
```

```
const iterate = async (num) => {  
  let sum = 0;  
  for (let i = num; i--;) {  
    sum += await resolveAfter2Seconds(10);  
  }  
  return sum;  
};
```

```
console.time(`iterate`);  
iterate(5).then(() => console.timeEnd(`iterate`)); // iterate: ~10s
```



String.prototype.padStart()/padEnd()

```
`abc`.padStart(10);           // "          abc"
`abc`.padStart(10, `foo`);    // "foofooabc"
`abc`.padStart(6, `123465`);  // "123abc"
`abc`.padStart(8, `0`);       // "00000abc"
`abc`.padStart(1);           // "abc"

`abc`.padEnd(10);            // "abc          "
`abc`.padEnd(10, `foo`);     // "abcfoofoo"
`abc`.padEnd(6, `123456`);   // "abc123"
`abc`.padEnd(1);            // "abc"
```



Trailing comma in functions

```
const fun = function f(p,) {};
```

```
const arrowFun = (p,) => {};
```

```
fun(5,);  
arrowFun(10,);
```

```
Math.max(10, 20,);
```



ES.Next

ES.Next

[developer.mozilla.org/en-US/docs/Web/JavaScript/New_in_JavaScript/
ECMAScript Next support in Mozilla](https://developer.mozilla.org/en-US/docs/Web/JavaScript/New_in_JavaScript/ECMAScript_Next_support_in_Mozilla)



Самые «вкусные» новинки

- rest/spread-операторы для объектов:

```
const {x, y, ...z} = {x: 1, y: 2, a: 3, b: 4};
```

```
const myObject = {x, y, ...z};
```

```
const objectCopy = {...myObject};
```

- Class and Property Decorators
- *Observable*



Транспайлер (Transpiler)

Преобразует исходный код более новой версии языка ECMAScript в более старый



Преимущества



Преимущества

- Улучшает поддержку браузеров



Преимущества

- Улучшает поддержку браузеров
- Можно использовать возможности не доступные прямо сейчас



Недостатки



Недостатки

- Синтаксис может меняться



Недостатки

- Синтаксис может меняться
- Не всемогущ (некоторые вещи просто невозможно сделать без поддержки в браузере)



Недостатки

- Синтаксис может меняться
- Не всемогущ (некоторые вещи просто невозможно сделать без поддержки в браузере)
- Увеличивает размер кода



Недостатки

- Синтаксис может меняться
- Не всемогущ (некоторые вещи просто невозможно сделать без поддержки в браузере)
- Увеличивает размер кода
- Замедляет код и сборку



Недостатки

- Синтаксис может меняться
- Не всесилен (некоторые вещи просто невозможно сделать без поддержки в браузере)
- Увеличивает размер кода
- Замедляет код и сборку
- Дополнительный инструмент при сборке проекта



Транспайлеры



Транспайлеры

- Babel
<http://babeljs.io>



Транспайлеры

- Babel
<http://babeljs.io>
- Google Closure Compiler
<https://developers.google.com/closure/compiler>



Транспайлеры

- Babel
<http://babeljs.io>
- Google Closure Compiler
<https://developers.google.com/closure/compiler>
- TypeScript
<https://www.typescriptlang.org>



Транспайлеры

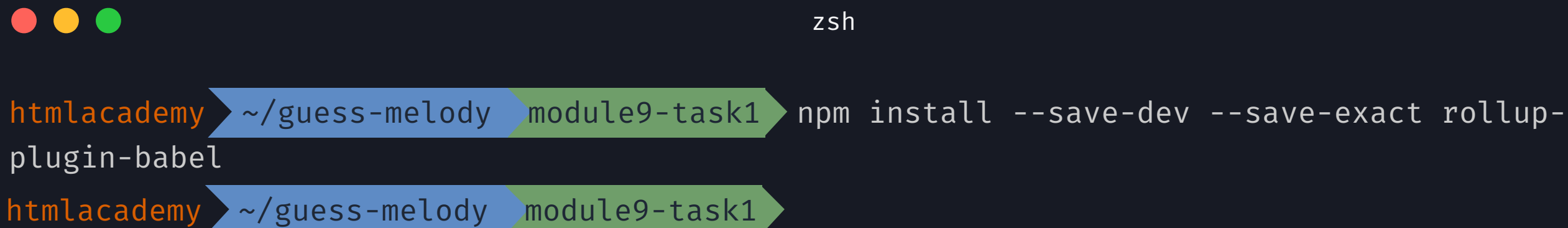
- Babel
<http://babeljs.io>
- Google Closure Compiler
<https://developers.google.com/closure/compiler>
- TypeScript
<https://www.typescriptlang.org>
- Traceur
<https://github.com/google/traceur-compiler>



Добавим в проект babel

Установим плагин для Rollup – BabelJS

npm install -DE rollup-plugin-babel



```
htmlacademy ~/guess-melody module9-task1 npm install --save-dev --save-exact rollup-  
plugin-babel  
htmlacademy ~/guess-melody module9-task1
```



Добавим нужные нам пресеты

Добавим дополнительные пакеты настройки BabelJS

npm install -DE babel-plugin-external-helpers babel-preset-env



zsh

htmlacademy > ~/guess-melody module9-task1 npm install --save-dev --save-exact babel-plugin-external-helpers babel-preset-env

htmlacademy > ~/guess-melody module9-task1



Добавим нужные нам пресеты

Добавим полифиллы BabelJS и whatwg-fetch

npm install -SE babel-polyfill whatwg-fetch



zsh

htmlacademy > ~/guess-melody module9-task1 npm install --save --save-exact babel-polyfill
whatwg-fetch

htmlacademy > ~/guess-melody module9-task1



Добавим babel в сборку


```
const babel = require('rollup-plugin-babel');
gulp.task('scripts', function () {
  return gulp.src('js/main.js')
    .pipe(plumber())
    .pipe(sourcemaps.init())
    // note that UMD and IIFE format requires
    .pipe(rollup({
      plugins: [
        babel({
          babelrc: false,
          exclude: 'node_modules/**',
          presets: [
            ['env', {modules: false}]
          ],
          plugins: [
            'external-helpers',
          ]
        })
      ]
    }, 'iife'))
    .pipe(sourcemaps.write(''))
    .pipe(gulp.dest('build/js'));
});
```



Научим rollup загружать node модули

Добавим плагины для загрузки модулей в Rollup

npm install -DE rollup-plugin-node-resolve rollup-plugin-commonjs



zsh

```
htmlacademy ~/guess-melody module9-task1 npm install --save-dev --save-exact rollup-  
plugin-node-resolve rollup-plugin-commonjs  
htmlacademy ~/guess-melody module9-task1
```



Добавим загрузчики в сборку

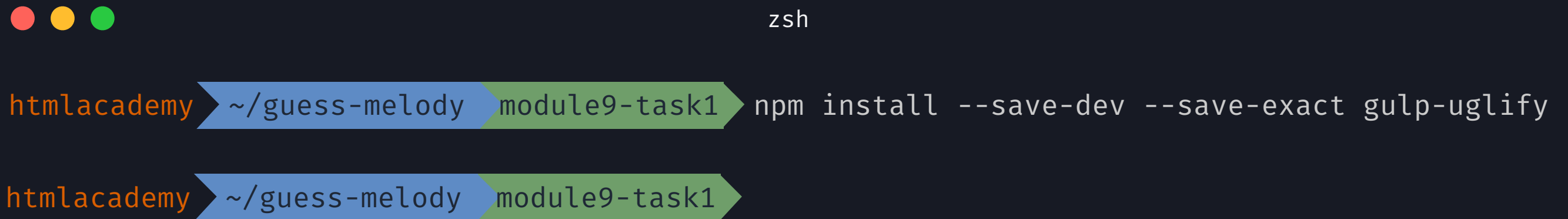
```
const resolve = require('rollup-plugin-node-resolve');
const commonjs = require('rollup-plugin-commonjs');
const babel = require('rollup-plugin-babel');
gulp.task('scripts', function () {
  return gulp.src('js/main.js')
    ...
    .pipe(rollup({
      plugins: [
        // resolve node_modules
        resolve({browser: true}),
        // resolve commonjs imports
        commonjs(),
        // use babel to transpile into ES5
        babel({
          babelrc: false,
          exclude: 'node_modules/**',
          presets: [
            ['env', {modules: false}]
          ],
          plugins: [
            'external-helpers',
          ]
        })
      ]
    }), 'iife'))
```



UglifyJS

Дополнительно можно подключить UglifyJS, чтобы минифицировать конечный файл скриптов:

npm install -DE gulp-uglify



```
htmlacademy ~/guess-melody module9-task1 npm install --save-dev --save-exact gulp-uglify
```

```
htmlacademy ~/guess-melody module9-task1
```



Добавим UglifyJS в сборку

```
const uglify = require('gulp-uglify');
const resolve = require('rollup-plugin-node-resolve');
const commonjs = require('rollup-plugin-commonjs');
const babel = require('rollup-plugin-babel');
gulp.task('scripts', function () {
  return gulp.src('js/main.js')
    .pipe(plumber())
    .pipe(sourcemaps.init())
    // note that UMD and IIFE format requires
    .pipe(rollup({
      plugins: [
        // resolve node_modules
        resolve({browser: true}),
        // resolve commonjs imports
        commonjs(),
        // use babel to transpile into ES5
        babel({
          babelrc: false,
          exclude: 'node_modules/**',
          presets: [
            ['env', {modules: false}]
          ],
          plugins: [
            'external-helpers',
          ]
        })
      ]
    }), 'iife'))
    // Uglify
    .pipe(uglify())
    // save sourcemap as separate file (in the same folder)
    .pipe(sourcemaps.write(''))
    .pipe(gulp.dest('build/js'));
```



Подключим полифиллы в main.js

```
import 'babel-polyfill';  
import 'whatwg-fetch';
```



К полету готовы



Async/Await (Demo)



React vs AngularJS (Demo)



Статическая типизация

Приём в языке программирования, при котором тип переменной указывается во время объявления переменной и не может быть изменён позже



Преимущества

medium.freecodecamp.com/why-use-static-types-in-javascript-part-1-8382da1e0adb



Преимущества

- Помогает находить ошибки в программе до её запуска



Преимущества

- Помогает находить ошибки в программе до её запуска
- Улучшает читаемость кода



Преимущества

- Помогает находить ошибки в программе до её запуска
- Улучшает читаемость кода
- Фиксирует поведение методов и функций



Преимущества

- Помогает находить ошибки в программе до её запуска
- Улучшает читаемость кода
- Фиксирует поведение методов и функций
- Упрощает изменение и рефакторинг кода



Преимущества

- Помогает находить ошибки в программе до её запуска
- Улучшает читаемость кода
- Фиксирует поведение методов и функций
- Упрощает изменение и рефакторинг кода
- Улучшает работу «умных» редакторов кода



Недостатки



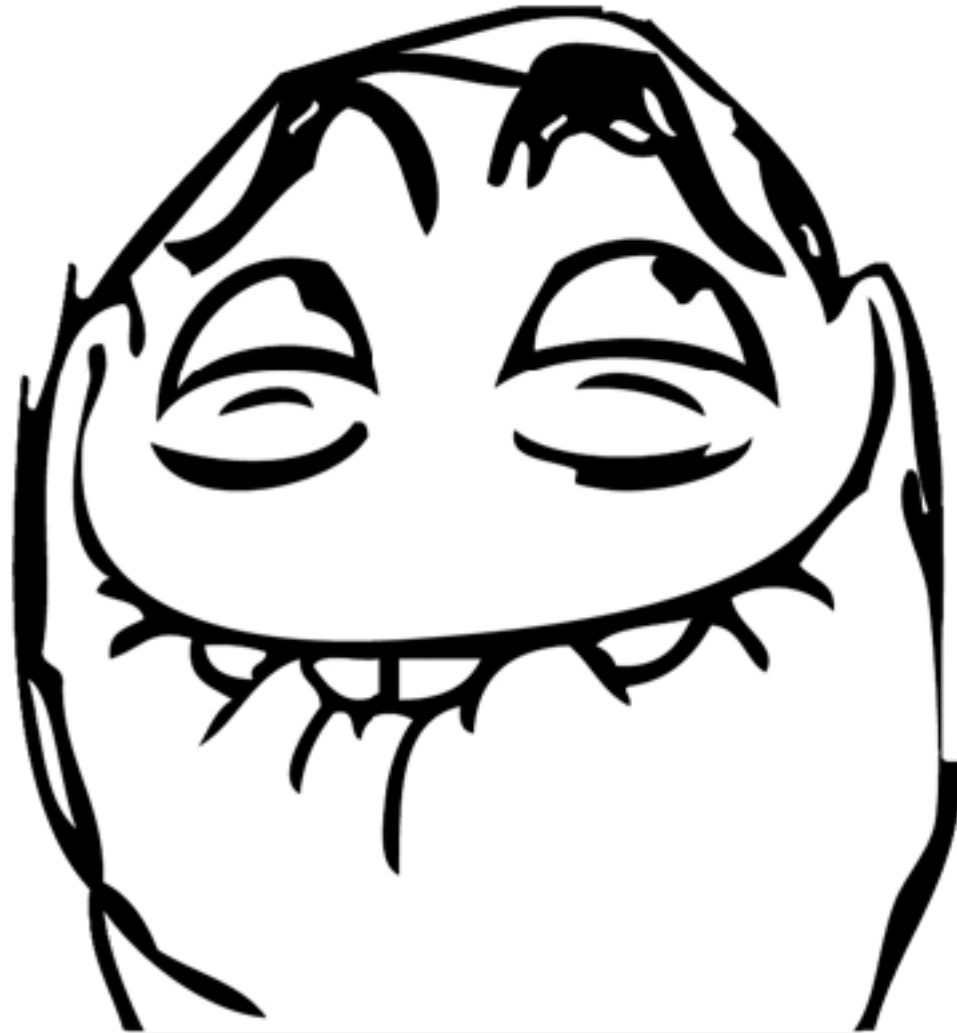
Недостатки

- Увеличивает количество кода



Недостатки

- Увеличивает количество кода
- Нужно понимать в общих чертах как работает система ТИПОВ



Типизированный JavaScript



Типизированный JavaScript

- Flow
<https://flow.org>



Типизированный JavaScript

- Flow
<https://flow.org>
- TypeScript
<https://www.typescriptlang.org>



Типизированный JavaScript

- Flow
<https://flow.org>
- TypeScript
<https://www.typescriptlang.org>
- Kotlin JavaScript Compiler
<https://kotlinlang.org/docs/tutorials/javascript/kotlin-to-javascript/kotlin-to-javascript.html>



Типизированный JavaScript

- Flow
<https://flow.org>
- TypeScript
<https://www.typescriptlang.org>
- Kotlin JavaScript Compiler
<https://kotlinlang.org/docs/tutorials/javascript/kotlin-to-javascript/kotlin-to-javascript.html>
- Scala.JS
<https://www.scala-js.org/>



Статическая типизация

На примере языка Flow



Объявление

```
// @flow
```

```
const myNum: number = 12;
```

```
let name: string = `Строка`;
```

```
var isTrue: boolean = true;
```



Вывод типа

```
// @flow
```

```
const myNum = 12;
```

```
let name = `Строка`;
```

```
var isTrue = true;
```



Специальные типы

```
// @flow
```

```
const nullData: null = null;
```

```
const undefinedData: void = undefined;
```

```
const iCanBeAnything: any = 'LALA' + 2;
```



Массивы и объекты

```
// @flow
```

```
const messages: Array<string> = ['hello', 'world', '!'];
```

```
const aboutMe: { name: string, age: number } = {  
  name: 'Preethi',  
  age: 26,  
};
```

```
const namesAndCities: { [name: string]: string } = {  
  Preethi: 'San Francisco',  
  Vivian: 'Palo Alto',  
};
```



Функции

```
// @flow
```

```
const calculateArea = (radius: number): number => {  
  return 3.14 * radius * radius;  
};
```

```
const print = (number: number,  
               calculator: (number) => number = calculateArea) => {  
  console.log(calculator(number));  
};
```

```
print(12);
```



Вывод типов в функциях

```
// @flow
```

```
const sum = (left, right) => left + right;
```

```
sum(`12`, `22`);
```

```
sum(12, `22`);
```

```
sum(12, 100); // undefined (too few arguments, expected  
default/rest parameters)
```

```
sum(true, false); // boolean. This type cannot be added to  
string
```



Именованные типы

```
// @flow
```

```
type PaymentMethod = {  
  id: number,  
  name: string,  
  limit: number,  
};
```

```
const myPaypal: PaymentMethod = {  
  id: 123456,  
  name: 'Preethi Paypal',  
  limit: 10000,  
};
```

```
type Email = string;
```

```
const academy: Email = 'mail@htmlacademy.ru';
```



Дженерики (Generics)

```
// @flow
```

```
type KeyObject<T> = { key: T };
```

```
const numberT: KeyObject<number> = {key: 123};
```

```
const stringT: KeyObject<string> = {key: `Preethi`};
```

```
const arrayT: KeyObject<Array<number>> = {key: [1, 2, 3]};
```



Объединения (*Join-type*)

// @flow

const even: 2|4|6|8|10 = 11; // number. This type is incompatible with number enum

const stringOrNumber: string|number = `Hello!`;

const stringOrUndefined: string|void = undefined;



Возможно (*Maybe*)

```
// @flow
```

```
type Maybe<T> = T | void | null;
```

```
let something: Maybe<string> = null;
```

```
let maybeNumber: Maybe<number> = 100;
```



Возможно (*Maybe*)

```
// @flow
```

```
let message: ?string = null;
```

```
let maybeArray: Array<?number> = [100, , 5];
```

```
let maybeObject: ?any = {};
```



Когда стоит использовать статическую типизацию



Когда стоит использовать статическую типизацию

- Программа очень важна для бизнеса



Когда стоит использовать статическую типизацию

- Программа очень важна для бизнеса
- Код скорее всего придётся многократно рефакторить



Когда стоит использовать статическую типизацию

- Программа очень важна для бизнеса
- Код скорее всего придётся многократно рефакторить
- Программу будет поддерживать большая команда или её нужно передавать заказчику



Когда не стоит использовать статическую типизацию



Когда не стоит использовать статическую типизацию

- Некритичная подсистема



Когда не стоит использовать статическую типизацию

- Некритичная подсистема
- Прототип, который скорее всего выкинут



Когда не стоит использовать статическую типизацию

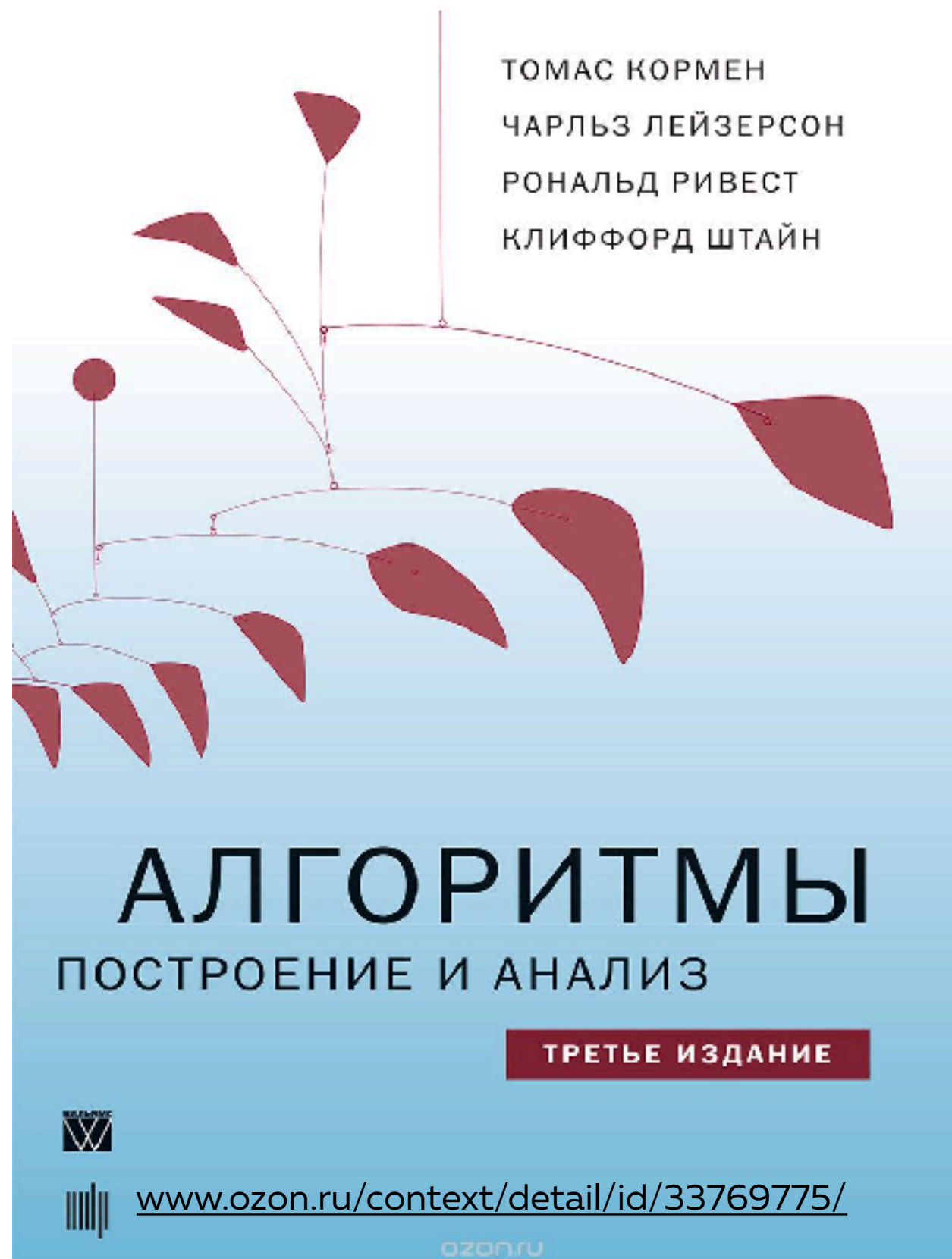
- Некритичная подсистема
- Прототип, который скорее всего выкинут
- Это ваш домашний проект



Что дальше?



Алгоритмы построение и анализ



Новые языки

- TypeScript
- Kotlin



Новые платформы

- node.js
- mobile (ionic, React.Native)





```
PROGRAM BYEBYE  
WRITE(UNIT=*, FMT=*) 'Näkemiin!'  
END
```

