



Web-технологии

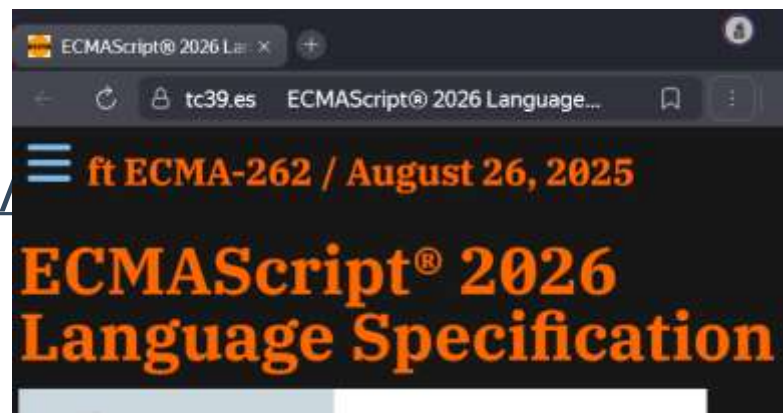
Основы JavaScript

- ECMAScript
- Установка и основы использования Node.js
- JavaScript
 - базовые конструкции языка
 - строки
 - функции
 - объекты, классы
 - массивы
 - JSON, промисы, генераторы, Map, Set, прокси
 - строгий режим

Что такое ECMAScript?

3

- JavaScript создавался как скриптовый язык для Netscape и не имеет никакого отношения к Java
- После чего был отправлен в ECMA International для стандартизации
- Это привело к появлению нового языкового стандарта, известного как ECMAScript
 - ECMAScript – стандарт
 - JavaScript – самая популярная реализация этого стандарта
- Известные реализации
 - SpiderMonkey (Netscape)
 - V8 (Node.JS)
 - JavaScriptCore (WebKit – Apple Safari)
 - Caracan (Opera)
 - Rhino (Java)
 - Chakra (IE)
- Стандарт <https://tc39.es/ecma262/>
 - ES1 ... ES5
 - ES2015 ... ES2026



Среда разработки

Node.JS

- <https://nodejs.org/>

JetBrains WebStorm

- <https://www.jetbrains.com/webstorm/>

Visual Studio Code

- <https://code.visualstudio.com/>

node.js

<https://nodejs.org/>

The image shows a composite of two screenshots. On the left is the Node.js website, and on the right is a Windows File Explorer window.

Node.js Website:

- Browser tab: Node.js — Run JavaScript Everywhere
- Address bar: nodejs.org
- Page title: Node.js — Run JavaScript Everywhere
- Logo: node.js
- Section: Run JavaScript Everywhere
- Description: Node.js® is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools and scripts.
- Button: Get Node.js®

Windows File Explorer:

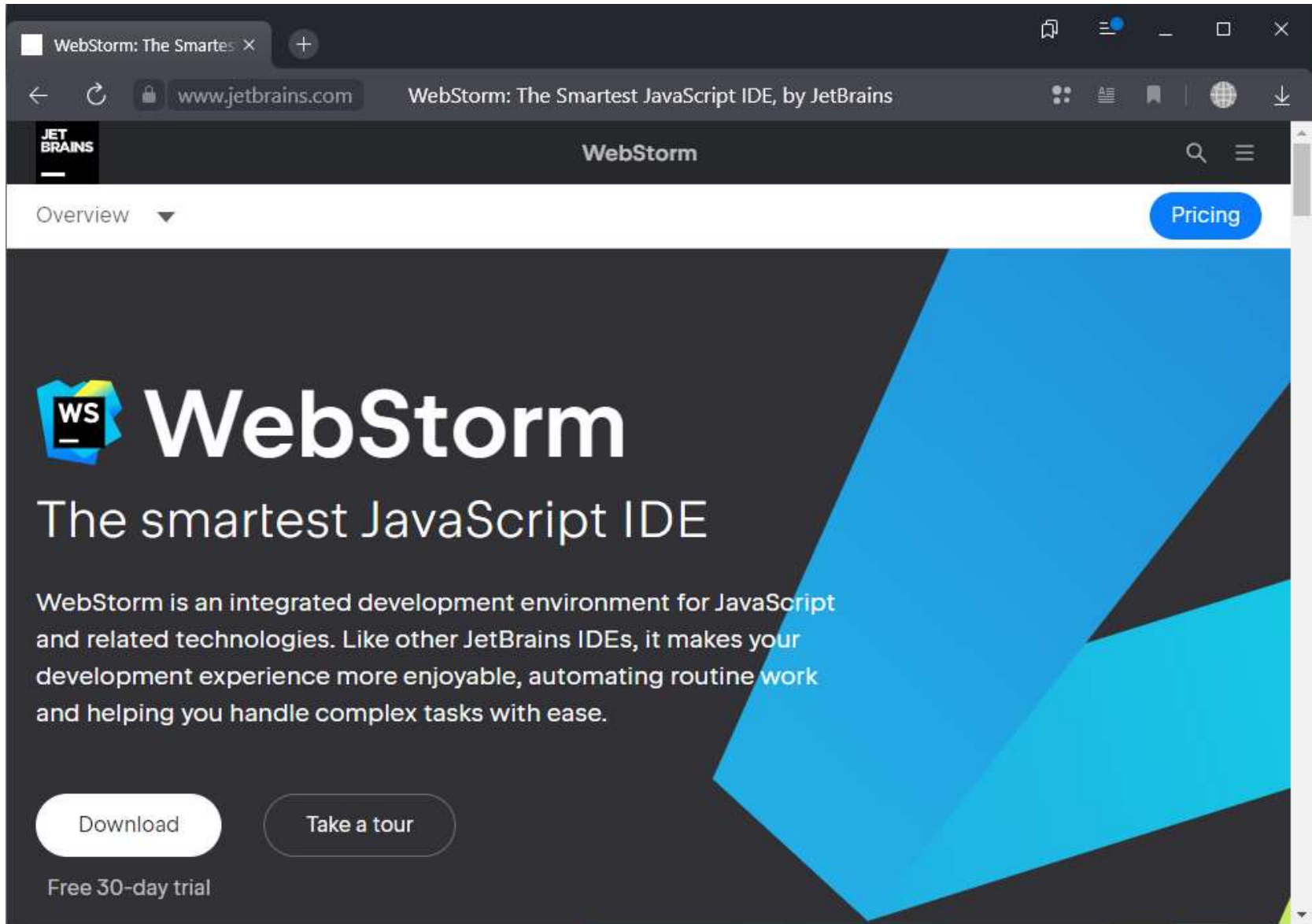
- Path: Program Files > nodejs
- Search: Поиск: nodejs
- Files and folders:
 - node_modules
 - node.exe
 - node_etw_provider.man
 - node_perfctr_provider.man
 - nodevars.bat
 - npm
 - npm.cmd
- Count: Элементов: 7

Code Snippet:

```
10 server.listen(3000, '127.0.0.1', () => {
11   console.log('Listening on 127.0.0.1');
12 });
13
14 // run with `node server.mjs`
```

JetBrains WebStorm

<https://www.jetbrains.com/webstorm/>



The screenshot shows the JetBrains WebStorm website. The browser's address bar displays "www.jetbrains.com" and the page title is "WebStorm: The Smartest JavaScript IDE, by JetBrains". The website header includes the "JETBRAINS" logo and the "WebStorm" product name. A navigation bar features a dropdown menu with "Overview" selected and a blue "Pricing" button. The main content area has a dark background with a large blue geometric shape on the right. It features the WebStorm logo (a blue cube with "WS" on it) and the text "WebStorm" in large white font, followed by "The smartest JavaScript IDE". Below this, a paragraph describes WebStorm as an integrated development environment for JavaScript and related technologies, highlighting its ability to make development more enjoyable by automating routine work and simplifying complex tasks. At the bottom, there are two buttons: "Download" and "Take a tour", and a text line stating "Free 30-day trial".

WebStorm: The Smartest JavaScript IDE, by JetBrains

Overview Pricing

WebStorm

The smartest JavaScript IDE

WebStorm is an integrated development environment for JavaScript and related technologies. Like other JetBrains IDEs, it makes your development experience more enjoyable, automating routine work and helping you handle complex tasks with ease.

Download Take a tour

Free 30-day trial

Настройка WebStorm

7

The screenshot displays the WebStorm 2017.2 IDE interface. On the left, the 'New Project' dialog is open, showing a list of project templates including 'Empty Project', 'HTML5 Boilerplate', 'Web Starter Kit', 'React App', 'Twitter Bootstrap', 'Foundation', 'AngularJS', 'Angular CLI', 'React Native', and 'Node.js Express App'. The 'Settings' dialog is also open, showing the 'Languages & Frameworks' section. Under 'Languages & Frameworks', 'Node.js and NPM' is selected. The 'Node interpreter' is set to 'C:\Program Files\nodejs\node.exe' (version 6.11.1). The 'Coding Assistance' section shows 'Node.js Core library is enabled'. The 'Packages' section lists installed packages:

Package	Version	Latest
express	4.15.3	4.15.4
express-generator	4.15.0	4.15.0

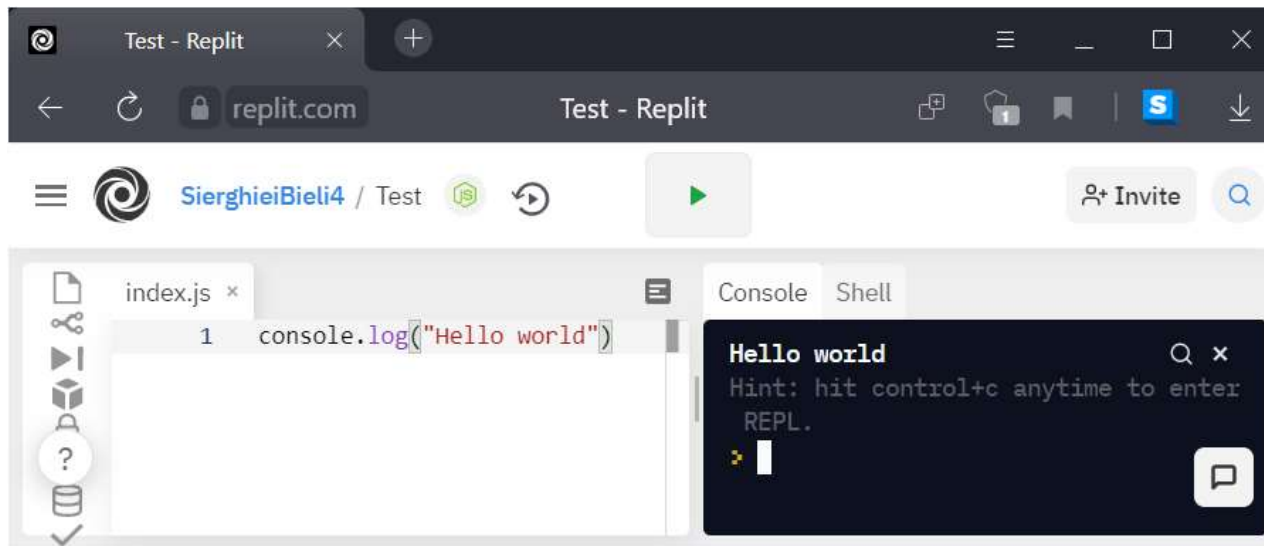
At the top right, the 'js1' project is open, showing the 'test.js' file with the code: `console.log("Проверка");`. The 'Run' button is highlighted. Below the code editor, the 'Run' output shows the command: `"C:\Program Files\JetBrains\WebStorm 2017.2\bin\runnerw.exe" "C:\Program Files\nodejs\node.exe" "C:\Users\User\WebstormProjects\js1\test.js"` and the output: `Проверка`. The status bar indicates 'Process finished with exit code 0'.

At the bottom right, a terminal window titled 'Select Command ...' is open, showing the command: `C:\Users\serge>node`. The terminal output shows: `Welcome to Node.js v14.16.1. Type ".help" for more information. > console.log("test")`. The terminal also shows the output: `test` and `undefined`.

<https://jsconsole.com/>



<https://replit.com/>



ОСНОВЫ JAVASCRIPT

Переменные и комментарии

```
var x = 1;  
let y = 2;  
{  
  let z = 3;  
}  
const b = 4;
```



БЛОК

// Комментарий, занимающий одну строку, кстати, здесь z не существует

/ Комментарий,
занимающий несколько строк.
/

/ Нельзя вкладывать /* комментарий в комментарий */ **SyntaxError** */*

Ключевые слова для объявления переменных: **var, let, const**

Область видимости переменных

10

```
if (true) {  
  var a = 2;  
}  
console.log(a); // 2
```

```
if (true) {  
  let b = 2;  
}  
console.log(b); // ReferenceError
```

```
window.foo = 'bar'; // Глобальная переменная
```

Присваивание значений

```
var x;  
console.log("The value of x is " + x); //Значение переменной x undefined  
console.log("The value of y is " + y); //Uncaught ReferenceError: y не определена  
console.log("The value of z is " + z); //Значение переменной z undefined  
var z;  
console.log("The value of a is " + a); //Uncaught ReferenceError: a не определена  
let a;
```

undefined

12

```
var i;  
if (i === undefined) {  
    correct(); // Пойдёт сюда  
} else {  
    none();  
}
```

```
var ar = [];  
if (!ar[0]) {  
    correct(); // Пойдёт сюда  
}
```

```
var x;  
x + 3; // NaN
```

```
console.log(null * 2) // 0
```

- **Примитивы**
 - Boolean
 - true и false.
 - Null
 - «пустое» значение
 - Undefined
 - переменная, не имеющая присвоенного значения
 - Number
 - String
 - Symbol
- Object

Переменные

```
let x1 = "строка#1" // "строка#1"  
let x2 = 'строка#2'; // "строка#2"  
let x3 = `строка#3`; // "строка#3"  
let x4 = true; // true  
let x5 = null; // null  
let x6; // undefined  
let x = 1 + 1; // 2  
sum = x + x * 2; // 6  
2 + x // 4  
2 + "test" // "2test"  
2 + x6 // NaN
```

Преобразование типов данных

15

```
var abc = 314;  
abc = "Thanks for all the fish...";  
a = "The abc is " + 314 // "The abc is 314"  
b = 314 + " is the abc" // "314 is the abc"  
"27" - 7 // 20  
"27" + 7 // "277"  
"1.1" + "2.1" // "1.12.1"  
(+"1.1") + (+"2.1") // 3.2  
// Скобки не обязательны  
parseFloat("3.14") // 3.14  
parseInt("42") // 42  
parseInt("101",2) // 5
```

«Продвинутый» JS

```
null + 1 // 1  
undefined + null // NaN  
[] + 1 // "1"  
[] - 1 // -1  
undefined + [] // "undefined"  
undefined - [] // NaN
```

Строки

```
let karl = "Карл";  
let text = `${karl}а у Клара ukrала кораллы,  
а ${karl} у Клары ukrал кларнет`;  
let small = "Тест ";  
text // "Карла у Клара ukrала кораллы,  
//а Карл у Клары ukrал кларнет"  
text.includes(karl) // true  
text.endsWith("кларнет") // true  
text.startsWith("кларнет") // false  
small.repeat(5) // "Tecm Tecm Tecm Tecm Tecm"  
t1 = "One row \nAnother" // "One row  
//Another"  
t2 = "One row \  
same row" // "One row same row"
```


Литералы

17

```
let a = ["1", 2, 3.14] // ["1", 2, 3.14]
a = ["1", , "2", ] // ["1", undefined, "2"]
a = ["1", , "2", ,] // ["1", undefined, "2", undefined]
a = [, "1", , "2"] // [undefined, "1", undefined, "2"]
b = true // true
b = false // false
x = 11 // 11
x = 011 // 9
x = 0x11 // 17
x = 0b11 // 3
y = 3.14 // 3.14
y = 3.14e+0 // 3.14
y = -.314 // -0.314

n = new Number()
n // [Number: 0]
n++ // 1
```

Литералы объектов

```
x = 11 // 11
```

```
function f(n) { console.log(n) }
```

```
o = { p1: "s", p2: f(1), p3: x, 4: "d" }
```

```
o.p1 // "s"
```

```
o.p3 // 11
```

```
o[4] // "d"
```

```
o.4 // SyntaxError: Unexpected number
```

```
o[p1] // ReferenceError: p1 is not defined
```

```
o["p1"] // "s"
```

```
o["4"] // "d"
```

```
// Аналогично числам - "" и "!"
```

УСЛОВИЯ (1)

```
let x = 1 - 1;
if(x < 5)
    console.log("x < 5, x = ", x);
else
    console.log("x >= 5, x = ", x);
let y;
if(!y)
    console.log('Не задан y');
```

Ложные значения

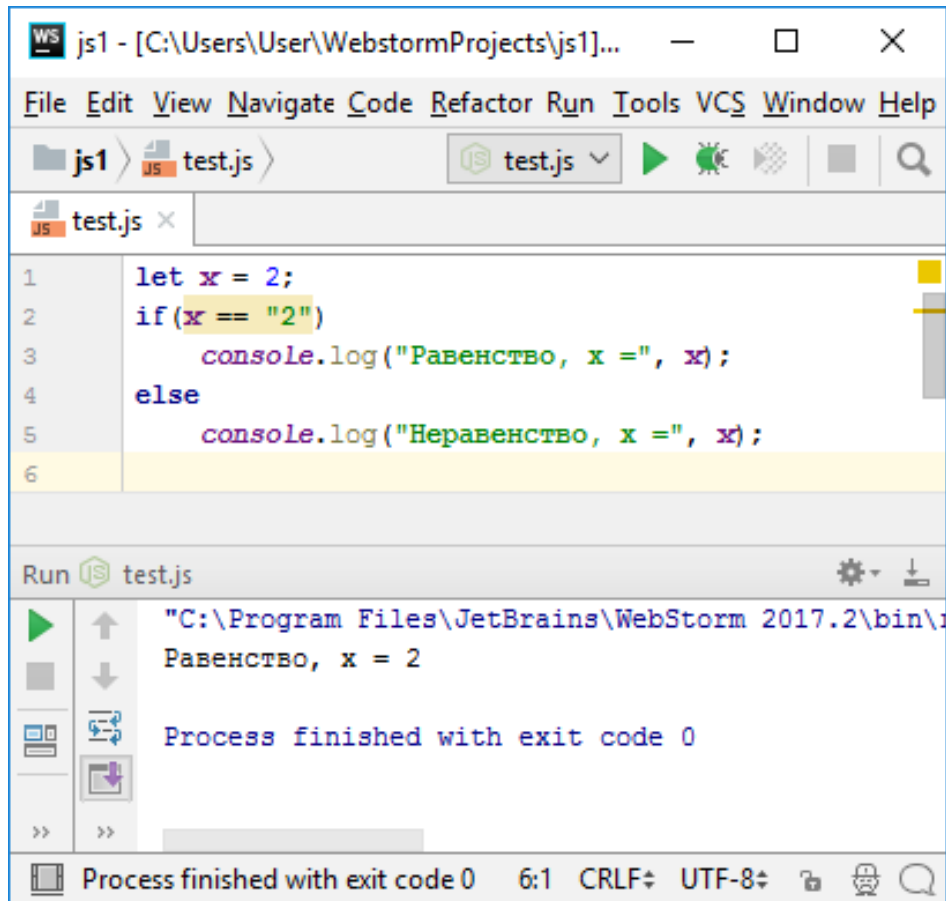
- false
- undefined
- null
- 0
- NaN
- ""

```
js1 - [C:\Users\User\WebstormProjects\js1] - ...test.js - ...
File Edit View Navigate Code Refactor Run Tools VCS Window Help
js1 test.js test.js
test.js x
1 let x = 1 - 1;
2 if(x < 5)
3     console.log("x < 5, x = ", x);
4 else
5     console.log("x >= 5, x = ", x);
6 let y;
7 if(!y)
8     console.log('Не задан y');
Run test.js
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\runnerw.exe"
x < 5, x = 0
Не задан y
Process finished with exit code 0
IDE and Plugin Updates: We... (9 minutes ago) 6:7 CRLF UTF-8
```

УСЛОВИЯ (2)

20

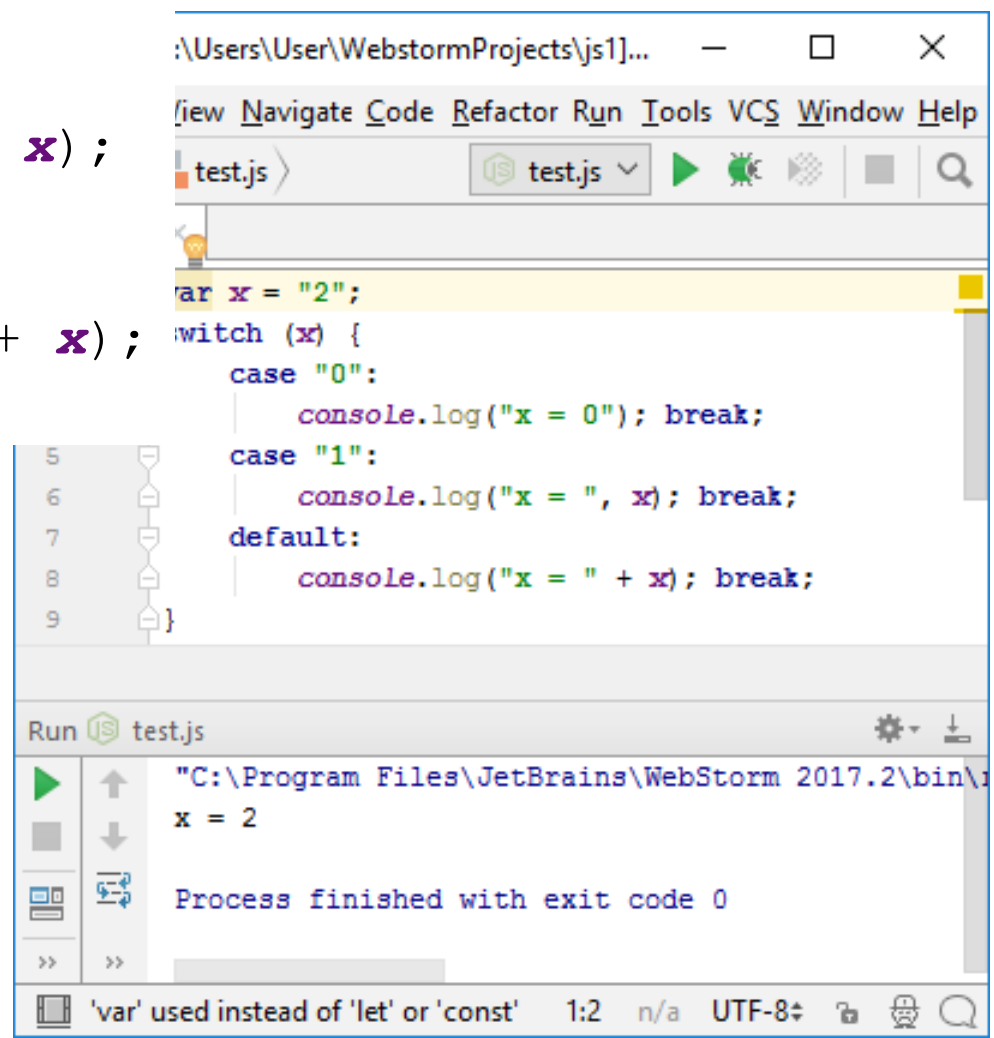
```
let x = 2;  
if (x == "2")  
    console.log("Равенство, x =", x);  
else  
    console.log("Неравенство, x =", x);
```



```
let b = new Boolean(false);  
if (b) // это условие true  
if (b == true) // это условие false
```

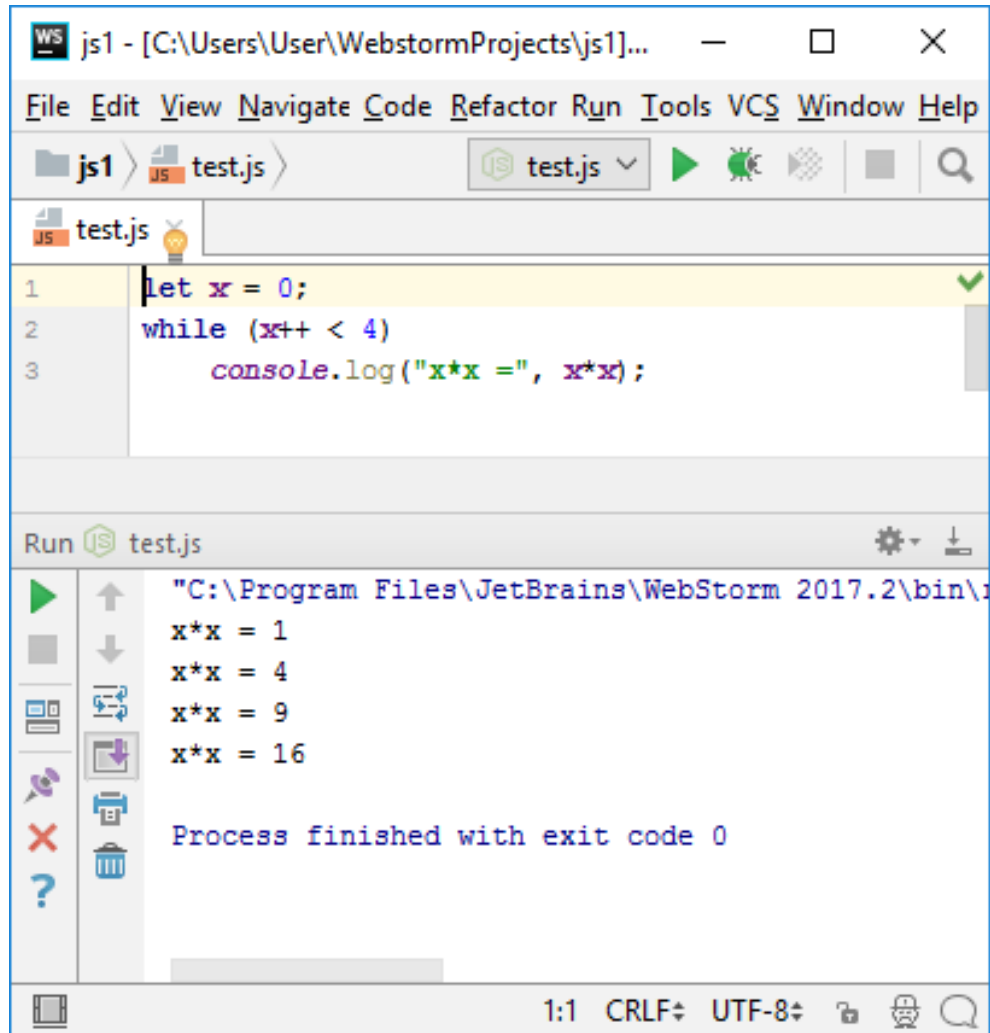
switch

```
var x = "2";  
switch (x) {  
  case "0":  
    console.log("x = 0");  
    break;  
  case "1":  
    console.log("x = ", x);  
    break;  
  default:  
    console.log("x = " + x);  
}
```



while

```
let x = 0;  
while (x++ < 4)  
    console.log("x*x =", x*x);
```



The screenshot shows an IDE window titled 'js1 - [C:\Users\User\WebstormProjects\js1]...'. The menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The breadcrumb navigation shows 'js1 > test.js'. The editor displays the following code:

```
1 let x = 0;  
2 while (x++ < 4)  
3     console.log("x*x =", x*x);
```

The code is executed, and the Run tool bar shows 'Run test.js'. The output console displays the following log messages:

```
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\...  
x*x = 1  
x*x = 4  
x*x = 9  
x*x = 16  
  
Process finished with exit code 0
```

The status bar at the bottom shows '1:1 CRLF UTF-8'.

Exception

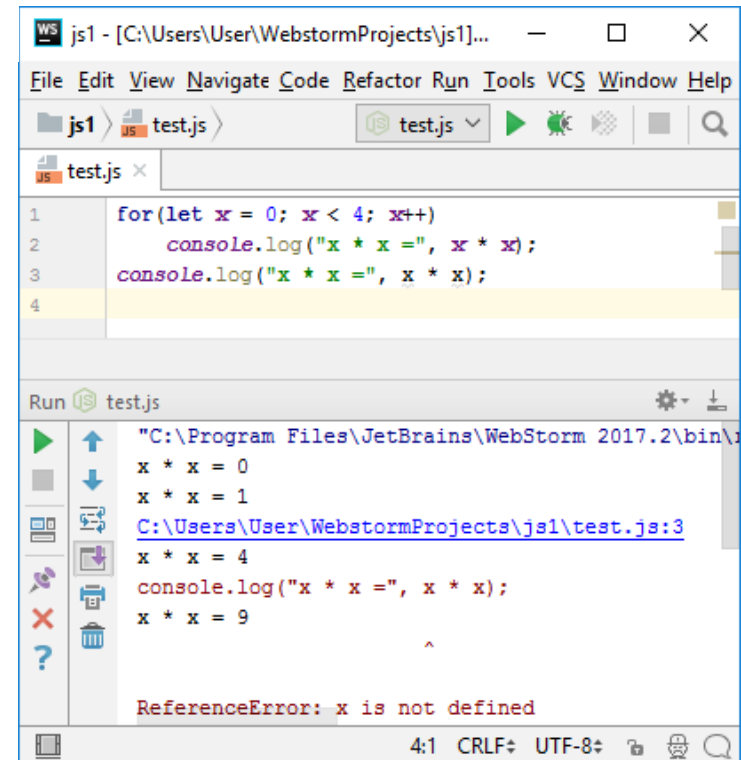
```
throw "Text"; // string
throw 42; // number
throw true; // boolean
throw { answer: "42" }; // object
```

```
try {
  throw "exception"
} catch (e) {
  console.error(e);
}
```

for

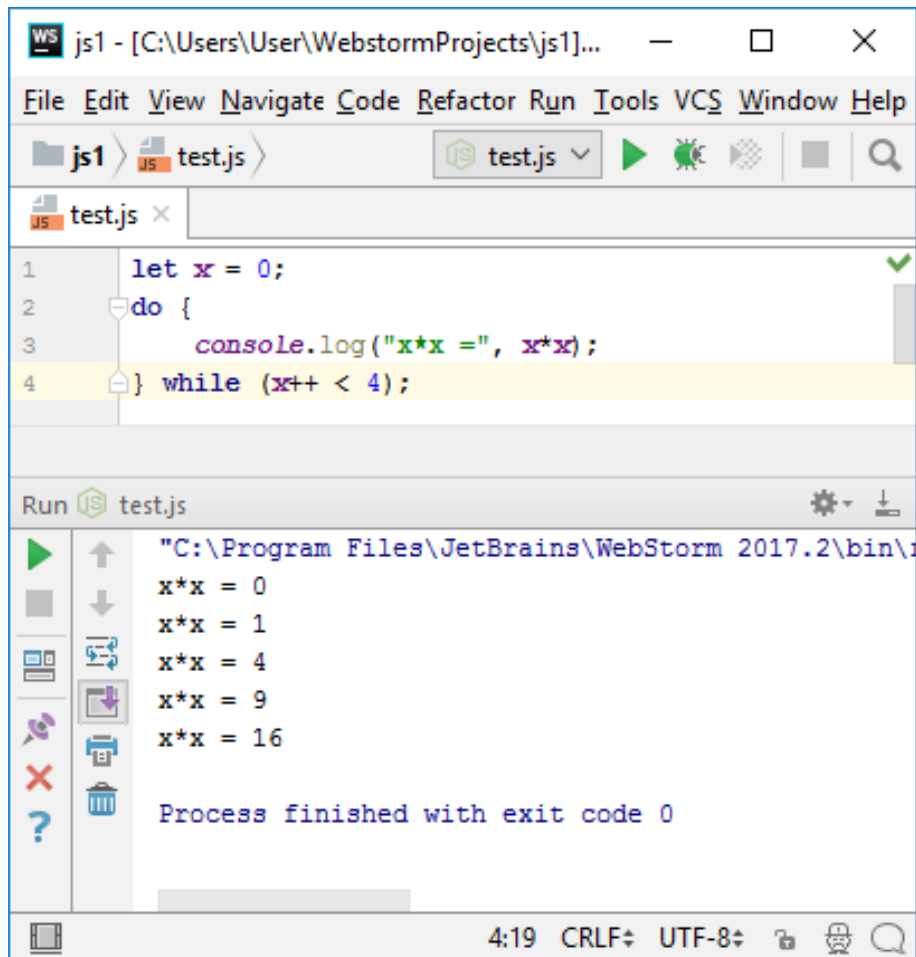
```
for(let x = 0; x < 4; x++)  
    console.log("x * x =", x * x);  
console.log("x * x =", x * x);
```

```
for(var x = 0; x < 4; x++)  
    console.log("x * x =", x * x);  
console.log("x * x =", x * x);
```



do-while

```
let x = 0;  
do {  
    console.log("x*x =", x*x);  
} while (x++ < 4);
```



The screenshot shows an IDE window titled 'js1 - [C:\Users\User\WebstormProjects\js1]...'. The editor displays the following JavaScript code:

```
1 let x = 0;  
2 do {  
3     console.log("x*x =", x*x);  
4 } while (x++ < 4);
```

The code is executed, and the output is shown in the 'Run test.js' panel:

```
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\...  
x*x = 0  
x*x = 1  
x*x = 4  
x*x = 9  
x*x = 16  
  
Process finished with exit code 0
```

The status bar at the bottom indicates the time is 4:19, the encoding is CRLF, and the character set is UTF-8.

break, break + метка

again:

```
for(let i = 0; i < 100; i++) {  
  for(let j = 0; j < 2; j++) {  
    console.log(`${i}.${j}`)  
    if (i == 2)  
      break again;  
  }  
}  
  
// 0.0  
// 0.1  
// 1.0  
// 1.1  
// 2.0
```

continue, continue + метка

27

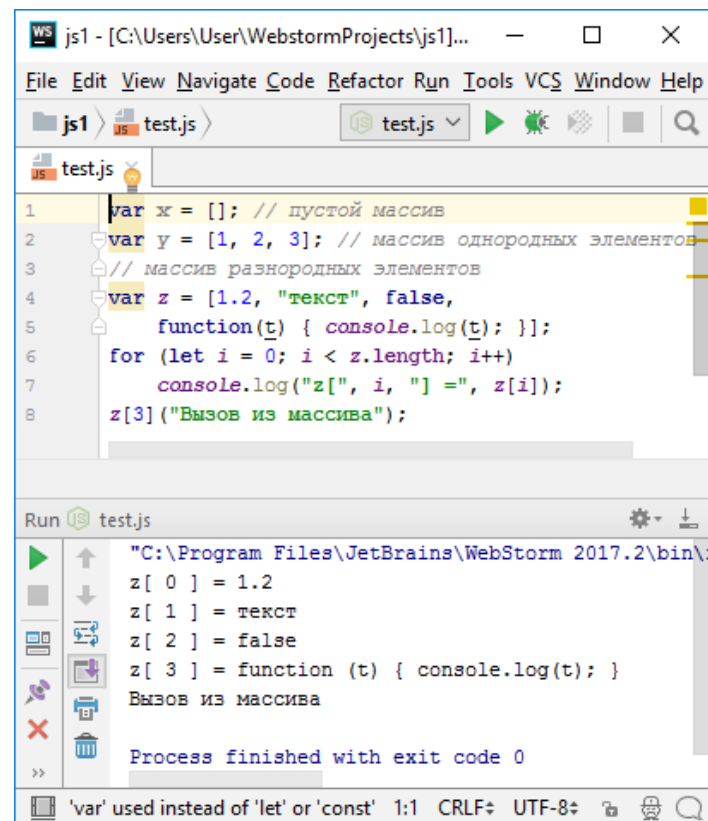
again:

```
for(let i = 0; i < 2; i++) {  
  for(let j = 0; j < 4; j++) {  
    if (j == 2)  
      continue again;  
    console.log(`${i}.${j}`)  
  }  
}  
// 0.0  
// 0.1  
// 1.0  
// 1.1
```

```
for(let i = 0; i < 2; i++) {  
  again:  
  for(let j = 0; j < 4; j++) {  
    if (j == 2)  
      continue again;  
    console.log(`${i}.${j}`)  
  }  
}  
// 0.0  
// 0.1  
// 0.3  
// 1.0  
// 1.1  
// 1.3
```

Массивы (1)

```
var x = []; // пустой массив
var y = [1, 2, 3]; // массив однородных элементов
// массив разнородных элементов
var z = [1.2, "текст", false,
        function(t) { console.log(t); }];
for (let i = 0; i < z.length; i++)
    console.log("z[" + i + "] =", z[i]);
z[3] ("Вызов из массива");
```



The screenshot shows a web browser window titled "js1 - [C:\Users\User\WebstormProjects\js1]...". The address bar shows "test.js". The page content displays the following JavaScript code:

```
1 var x = []; // пустой массив
2 var y = [1, 2, 3]; // массив однородных элементов
3 // массив разнородных элементов
4 var z = [1.2, "текст", false,
5         function(t) { console.log(t); }];
6 for (let i = 0; i < z.length; i++)
7     console.log("z[" + i + "] =", z[i]);
8 z[3] ("Вызов из массива");
```

The browser's console shows the output of the script:

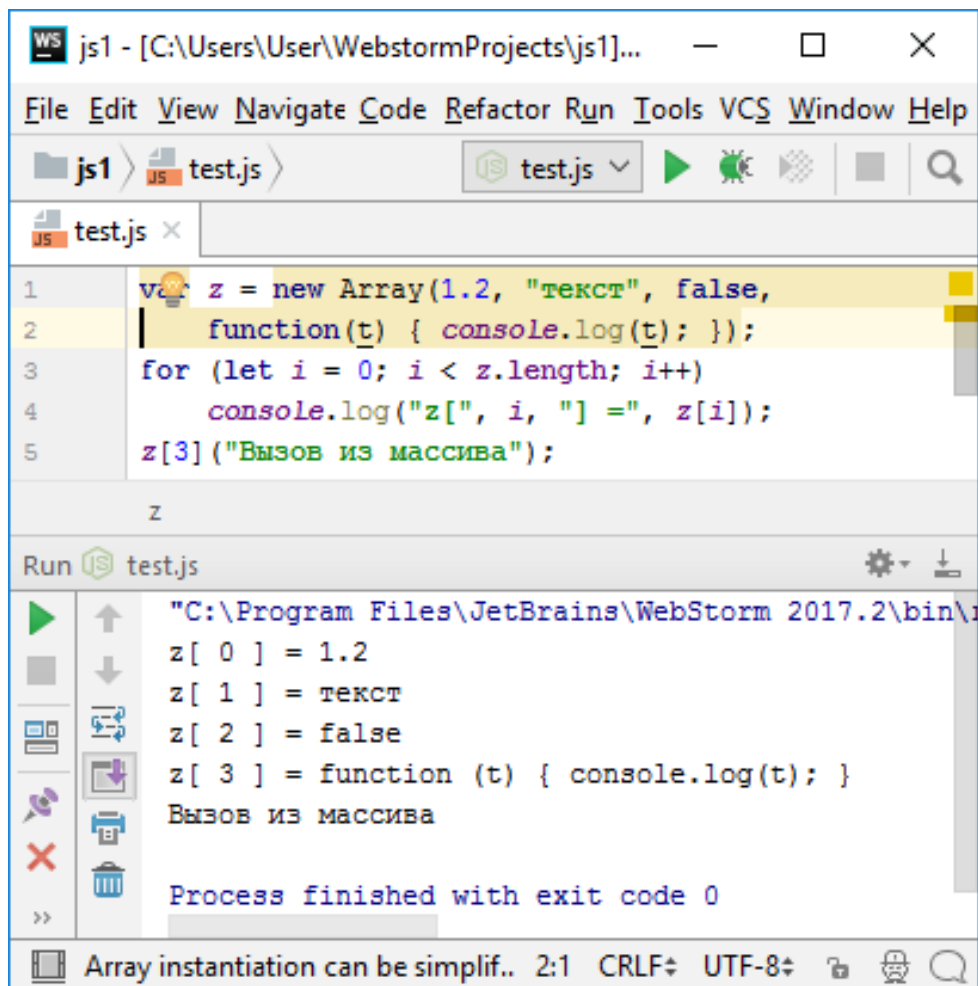
```
z[ 0 ] = 1.2
z[ 1 ] = текст
z[ 2 ] = false
z[ 3 ] = function (t) { console.log(t); }
Вызов из массива
```

At the bottom of the console, it states "Process finished with exit code 0". A status bar at the very bottom indicates a warning: "'var' used instead of 'let' or 'const' 1:1 CRLF UTF-8".

Массивы (2)

29

```
var z = new Array(1.2, "текст", false,  
    function(t) { console.log(t); });  
for (let i = 0; i < z.length; i++)  
    console.log("z[" + i + "] =", z[i]);  
z[3] ("Вызов из массива");
```

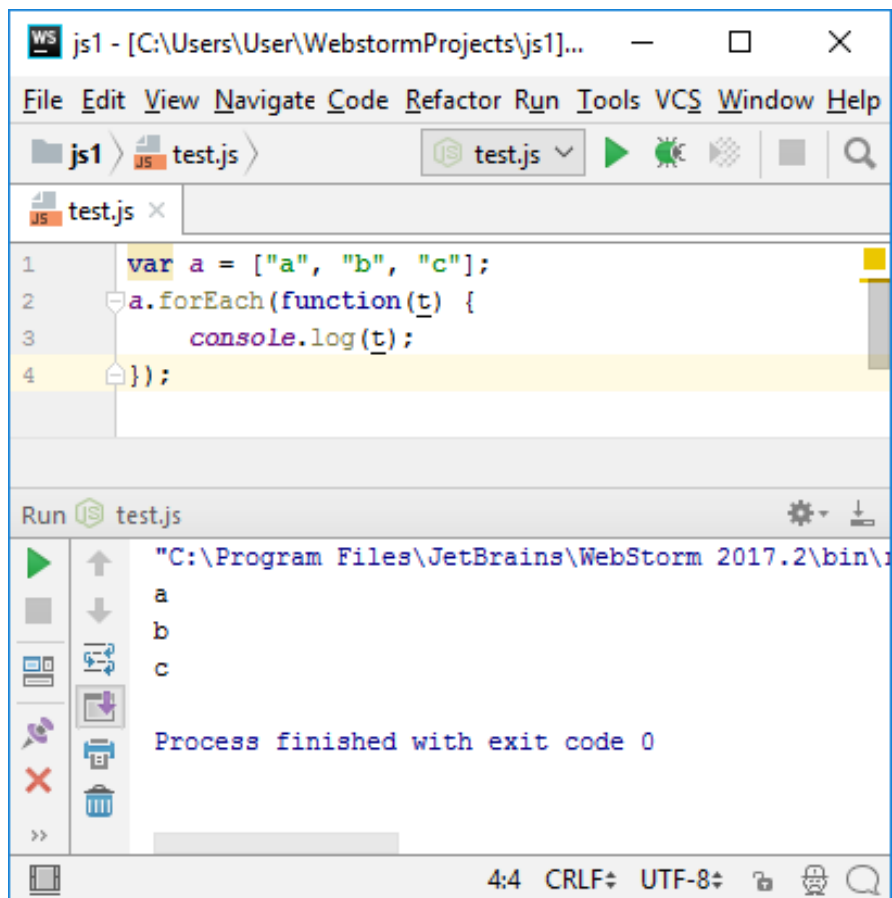


The screenshot shows an IDE window titled 'js1 - [C:\Users\User\WebstormProjects\js1]...'. The editor displays the same JavaScript code as the previous block. Below the editor, the 'Run' tab is active, showing the command 'Run test.js'. The console output displays the execution results: 'z[0] = 1.2', 'z[1] = текст', 'z[2] = false', 'z[3] = function (t) { console.log(t); }', and 'Вызов из массива'. The process finished with exit code 0. The status bar at the bottom indicates 'Array instantiation can be simplif.. 2:1 CRLF UTF-8'.

```
js1 - [C:\Users\User\WebstormProjects\js1]...  
File Edit View Navigate Code Refactor Run Tools VCS Window Help  
js1 test.js test.js  
test.js  
1 var z = new Array(1.2, "текст", false,  
2   function(t) { console.log(t); });  
3 for (let i = 0; i < z.length; i++)  
4   console.log("z[" + i + "] =", z[i]);  
5 z[3] ("Вызов из массива");  
z  
Run test.js  
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\  
z[ 0 ] = 1.2  
z[ 1 ] = текст  
z[ 2 ] = false  
z[ 3 ] = function (t) { console.log(t); }  
Вызов из массива  
Process finished with exit code 0  
Array instantiation can be simplif.. 2:1 CRLF UTF-8
```

Массивы (3), forEach

```
var a = ["a", "b", "c"];  
a.forEach(function(t) {  
    console.log(t);  
});
```



Массивы (4), for in/of

```
var a = ["a"];
a[10] = "b";
a[100] = "c";
a.forEach(function(t) {
    console.log(t);
});
for (var key in a)
    console.log(a[key]);
```

```
let arr = ["a", "b", "c"];
arr.x = "Who?"
for (let key in arr)
    console.log(key);
for (let value of arr)
    console.log(value);
for (let char of "Да")
    console.log(char);
```

```
// 0
```

```
// 1
```

```
// 2
```

```
// x
```

```
// a
```

```
// b
```

```
// c
```

```
// Д
```

```
// а
```

The screenshot shows an IDE window titled 'js1 - [C:\Users\User\WebstormProjects\js1]...'. The editor displays the following JavaScript code in 'test.js':

```
1 var a = ["a"];
2 a[10] = "b";
3 a[100] = "c";
4 a.forEach(function(t) {
5     console.log(t);
6 });
7 for (var key in a)
8     console.log(a[key]);
```

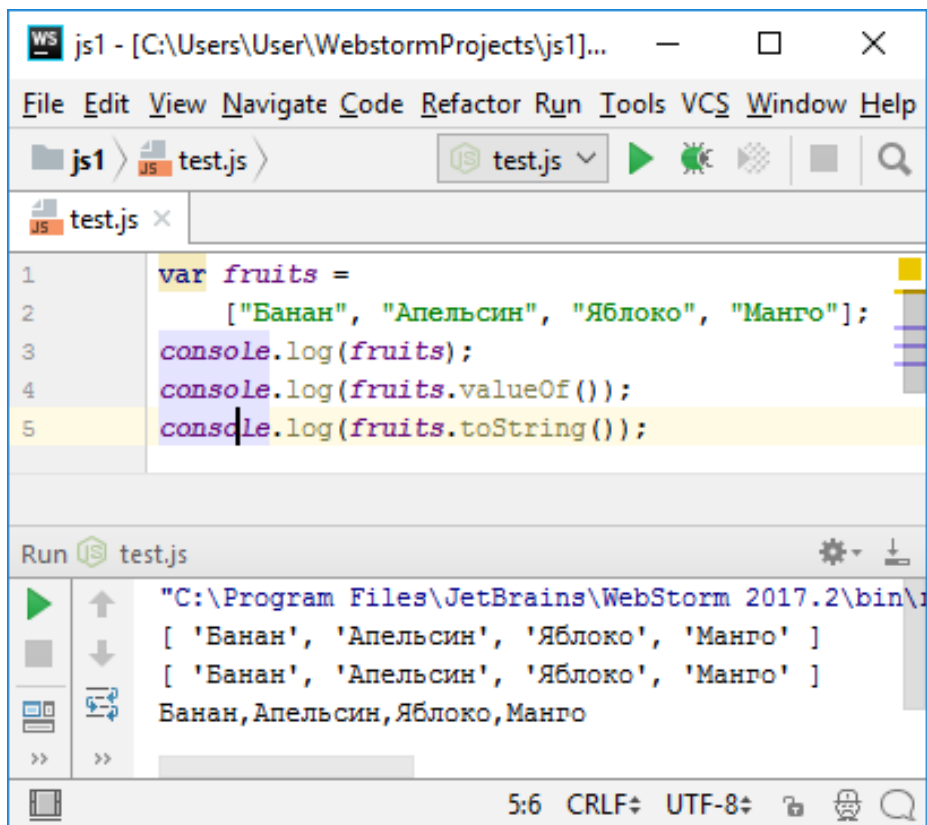
The code is being executed, and the output is shown in the 'Run' console at the bottom. The output consists of the following lines:

```
a
b
c
a
b
c
Д
а
```

Below the output, it says 'Process finished with exit code 0'. The status bar at the bottom indicates '6:1 CRLF UTF-8'.

Массивы (5), valueOf, toString

```
var fruits =  
    [ "Банан", "Апельсин", "Яблоко", "Манго" ] ;  
console.log( fruits ) ;  
console.log( fruits.valueOf() ) ;  
console.log( fruits.toString() ) ;
```



The screenshot shows an IDE window titled 'js1 - [C:\Users\User\WebstormProjects\js1]...'. The menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The toolbar shows icons for file operations and running code. The editor displays the following code in 'test.js':

```
1 var fruits =  
2   [ "Банан", "Апельсин", "Яблоко", "Манго" ] ;  
3   console.log(fruits);  
4   console.log(fruits.valueOf());  
5   console.log(fruits.toString());
```

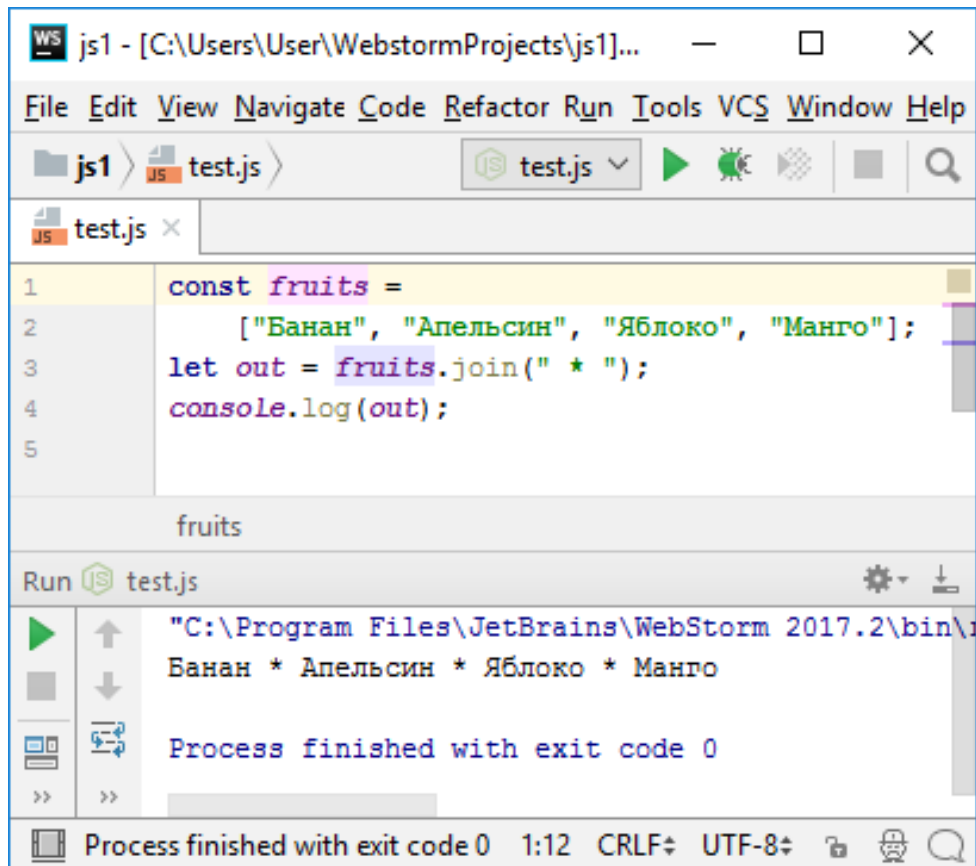
Below the editor is a 'Run' panel for 'test.js'. It shows the execution path and the output of the three log statements:

```
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\...  
[ 'Банан', 'Апельсин', 'Яблоко', 'Манго' ]  
[ 'Банан', 'Апельсин', 'Яблоко', 'Манго' ]  
Банан,Апельсин,Яблоко,Манго
```

The status bar at the bottom indicates '5:6 CRLF UTF-8'.

Массивы (6), join

```
const fruits =  
    ["Банан", "Апельсин", "Яблоко", "Манго"];  
let out = fruits.join(" * ");  
console.log(out);
```



The screenshot shows a code editor window titled 'js1 - [C:\Users\User\WebstormProjects\js1]...'. The code in 'test.js' is as follows:

```
1 const fruits =  
2     ["Банан", "Апельсин", "Яблоко", "Манго"];  
3 let out = fruits.join(" * ");  
4 console.log(out);  
5
```

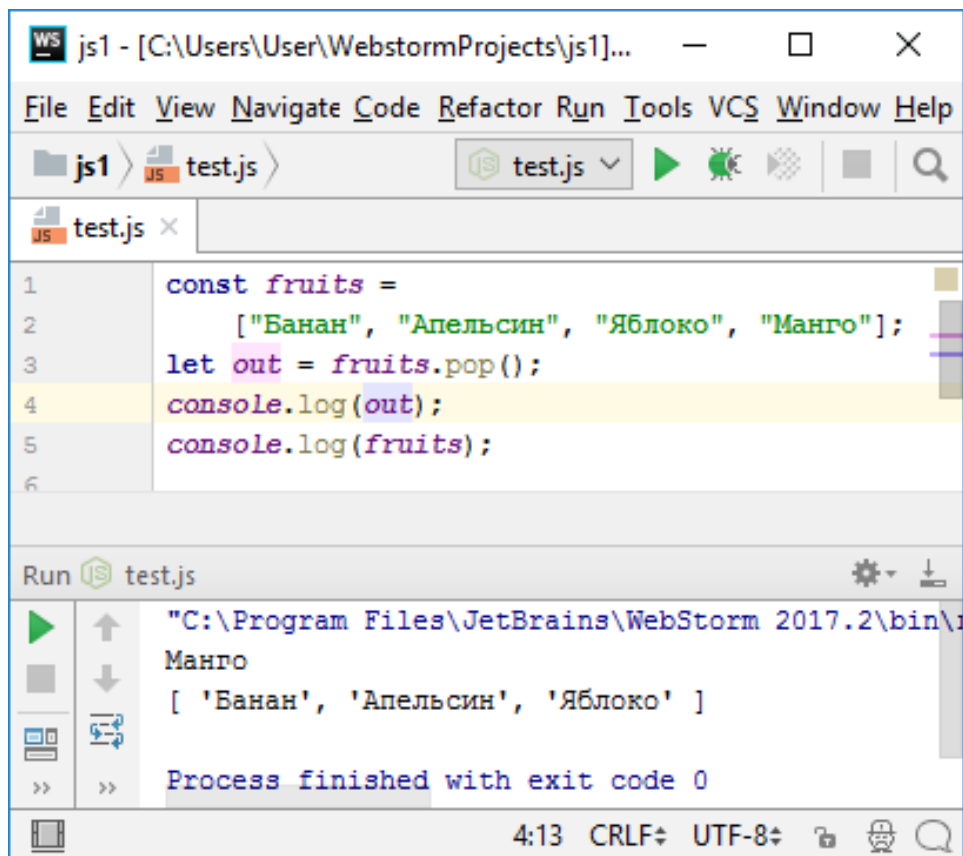
Below the code editor, the 'Run' tab is active, showing the output of the program:

```
Run test.js  
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\...  
Банан * Апельсин * Яблоко * Манго  
Process finished with exit code 0
```

The status bar at the bottom indicates 'Process finished with exit code 0', '1:12', 'CRLF', and 'UTF-8'.

Массивы (7), pop

```
const fruits =  
    [ "Банан", "Апельсин", "Яблоко", "Манго" ];  
let out = fruits.pop();  
console.log(out);  
console.log(fruits);
```



The screenshot shows an IDE window titled 'js1 - [C:\Users\User\WebstormProjects\js1]...'. The menu bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, and Help. The toolbar shows icons for running and debugging. The editor displays the following code in 'test.js':

```
1  const fruits =  
2      [ "Банан", "Апельсин", "Яблоко", "Манго" ];  
3  let out = fruits.pop();  
4  console.log(out);  
5  console.log(fruits);  
6
```

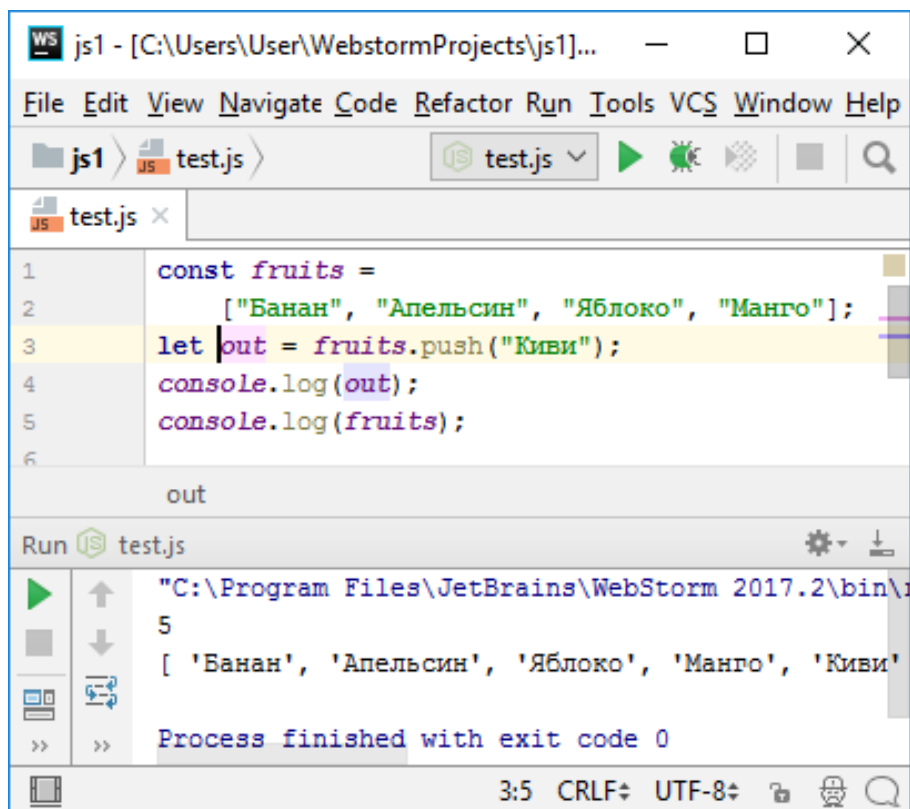
The 'Run' button is highlighted. Below the editor, the 'Run' panel shows the output of the execution:

```
Run test.js  
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\...  
Манго  
[ 'Банан', 'Апельсин', 'Яблоко' ]  
Process finished with exit code 0
```

The status bar at the bottom indicates the time is 4:13, the encoding is CRLF, and the character set is UTF-8.

Массивы (8), push

```
const fruits =  
    [ "Банан", "Апельсин", "Яблоко", "Манго" ] ;  
let out = fruits.push( "Киви" ) ;  
console.log( out ) ;  
console.log( fruits ) ;
```



The screenshot shows an IDE window titled 'js1 - [C:\Users\User\WebstormProjects\js1]...'. The code editor displays the following JavaScript code:

```
1 const fruits =  
2     [ "Банан", "Апельсин", "Яблоко", "Манго" ] ;  
3 let out = fruits.push( "Киви" ) ;  
4 console.log( out ) ;  
5 console.log( fruits ) ;  
6
```

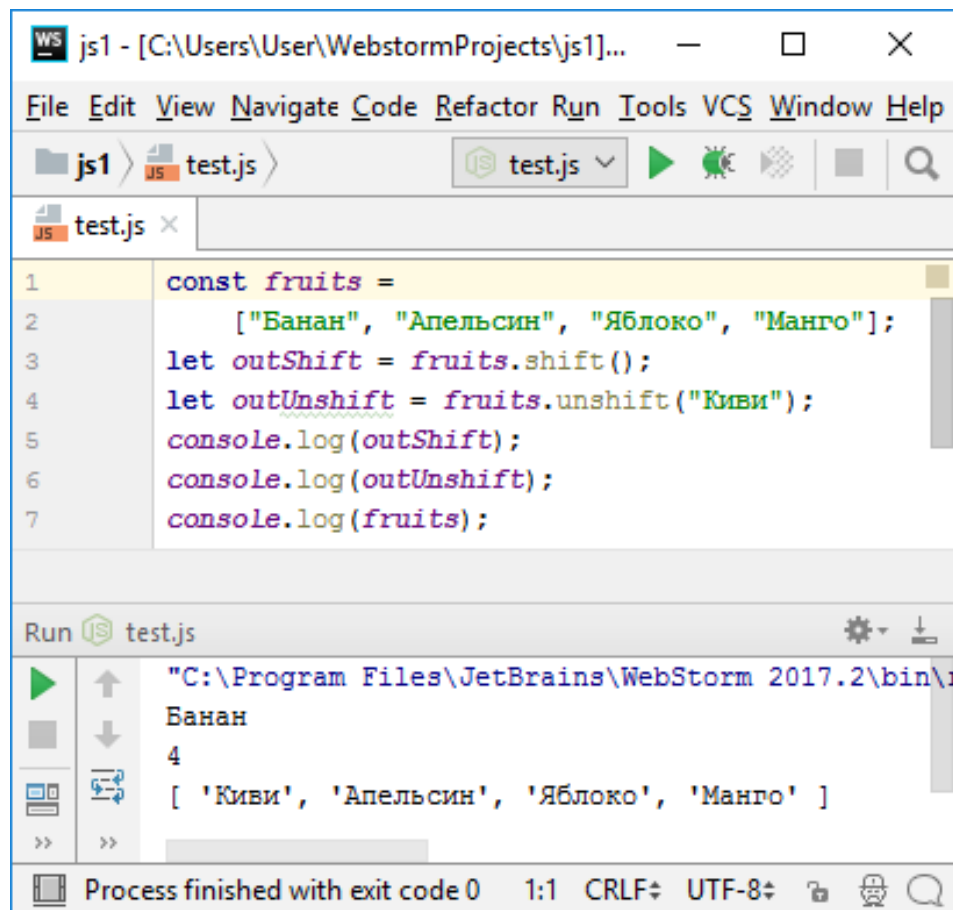
Below the code editor, the 'Run' tab shows the output of the program:

```
Run test.js  
out  
5  
[ 'Банан', 'Апельсин', 'Яблоко', 'Манго', 'Киви' ]  
Process finished with exit code 0
```

The status bar at the bottom indicates the file encoding is UTF-8 and the line ending is CRLF.

Массивы (9), shift, unshift

```
const fruits =  
    ["Банан", "Апельсин", "Яблоко", "Манго"];  
let outShift = fruits.shift();  
let outUnshift = fruits.unshift("Киви");  
console.log(outShift);  
console.log(outUnshift);  
console.log(fruits);
```



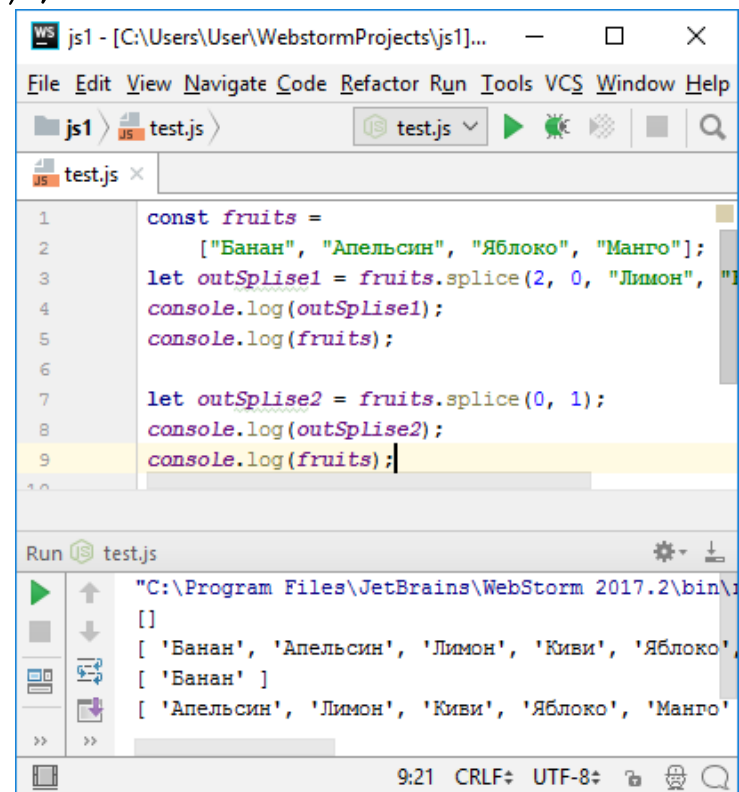
```
js1 - [C:\Users\User\WebstormProjects\js1]...  
File Edit View Navigate Code Refactor Run Tools VCS Window Help  
js1 test.js test.js  
test.js  
1 const fruits =  
2     ["Банан", "Апельсин", "Яблоко", "Манго"];  
3 let outShift = fruits.shift();  
4 let outUnshift = fruits.unshift("Киви");  
5 console.log(outShift);  
6 console.log(outUnshift);  
7 console.log(fruits);  
Run test.js  
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\...  
Банан  
4  
[ 'Киви', 'Апельсин', 'Яблоко', 'Манго' ]  
Process finished with exit code 0 1:1 CRLF UTF-8
```

Массивы (А), splice

37

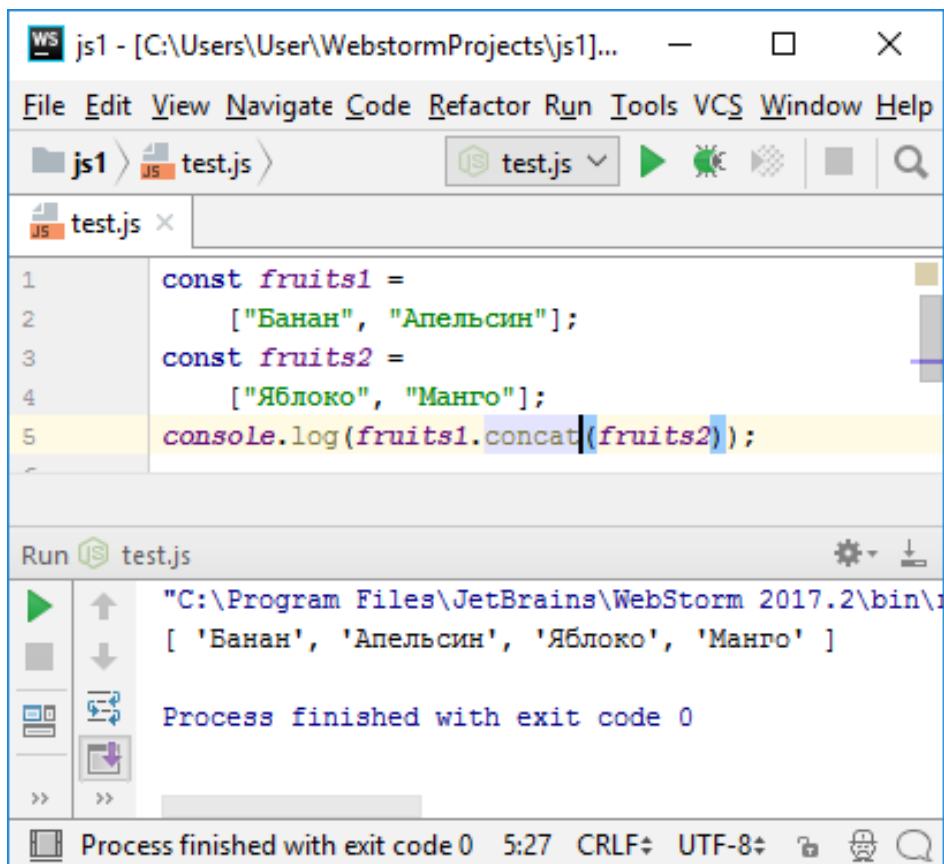
```
const fruits =  
    ["Банан", "Апельсин", "Яблоко", "Манго"];  
let outSplice1 = fruits.splice(2, 0, "Лимон", "Киви");  
console.log(outSplice1);  
console.log(fruits);
```

```
let outSplice2 = fruits.splice(0, 1);  
console.log(outSplice2);  
console.log(fruits);
```



Массивы (В), concat

```
const fruits1 =  
    [ "Банан", "Апельсин" ] ;  
const fruits2 =  
    [ "Яблоко", "Манго" ] ;  
console.log ( fruits1.concat ( fruits2 ) ) ;
```



The screenshot shows a web browser window titled 'js1 - [C:\Users\User\WebstormProjects\js1]...' with a menu bar (File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help) and a toolbar. The main editor displays a file named 'test.js' with the following code:

```
1 const fruits1 =  
2   [ "Банан", "Апельсин" ] ;  
3 const fruits2 =  
4   [ "Яблоко", "Манго" ] ;  
5 console.log ( fruits1.concat ( fruits2 ) ) ;
```

Below the editor is a 'Run' button and a 'test.js' label. The output console shows the execution result:

```
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\...  
[ 'Банан', 'Апельсин', 'Яблоко', 'Манго' ]  
Process finished with exit code 0
```

The status bar at the bottom indicates 'Process finished with exit code 0', '5:27', 'CRLF', 'UTF-8', and some icons.

Массивы (C), sort

```
const fruits =
    ["Банан", "Апельсин", "Яблоко", "Манго"];
console.log(fruits.sort());
console.log(fruits.reverse());
```

```
const numbers = [22, 12, 17, 3];
console.log(numbers.sort());
console.log(numbers.sort(
    function(a, b) { return a - b; }
));
```

Array.from(<псевдомассив>)

The screenshot shows a code editor window titled 'js1 - [C:\Users\User\WebstormProjects\js1]...' with a menu bar (File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help). The editor contains the following JavaScript code:

```
1 const fruits =
2   ["Банан", "Апельсин", "Яблоко", "Манго"];
3 console.log(fruits.sort());
4 console.log(fruits.reverse());
5
6 const numbers = [22, 12, 17, 3];
7 console.log(numbers.sort());
8 console.log(numbers.sort(
9   function(a, b) { return a - b; }
10  ));
```

Below the editor is a 'Run' panel showing the output of the code execution:

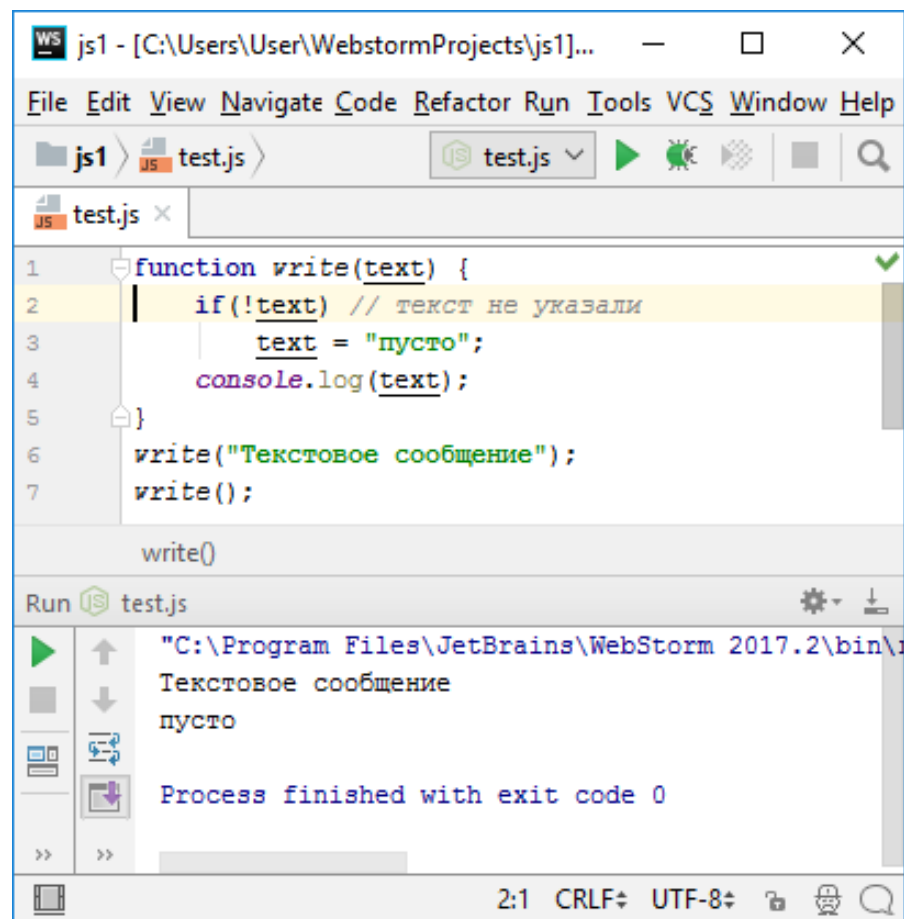
```
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\...
[ 'Апельсин', 'Банан', 'Манго', 'Яблоко' ]
[ 'Яблоко', 'Манго', 'Банан', 'Апельсин' ]
[ 12, 17, 22, 3 ]
[ 3, 12, 17, 22 ]
```

The process finished with exit code 0. The status bar at the bottom shows '8:20 CRLF UTF-8'.

ФУНКЦИИ (1)

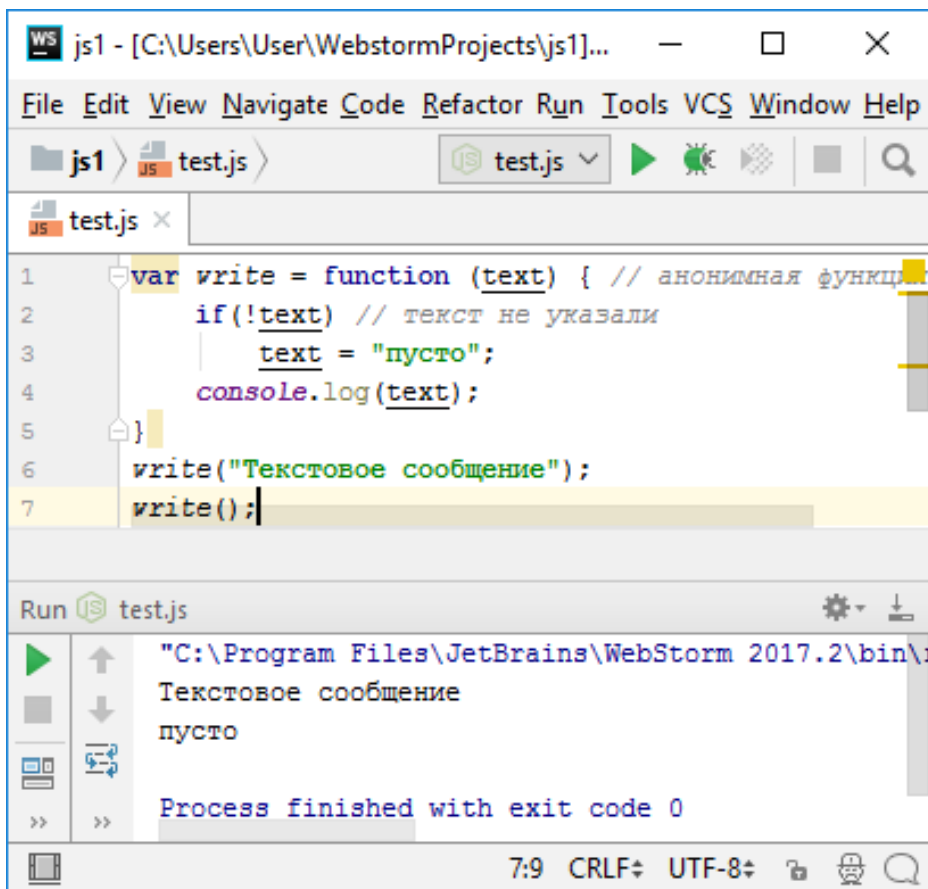
40

```
function write(text) {  
    if(!text) // текст не указали  
        text = "пусто";  
    console.log(text);  
}  
write("Текстовое сообщение");  
write();
```



Функции (2), анонимная

```
var write = function (text) { // анонимная функция
    if(!text) // текст не указали
        text = "пусто";
    console.log(text);
}
write("Текстовое сообщение");
write();
```



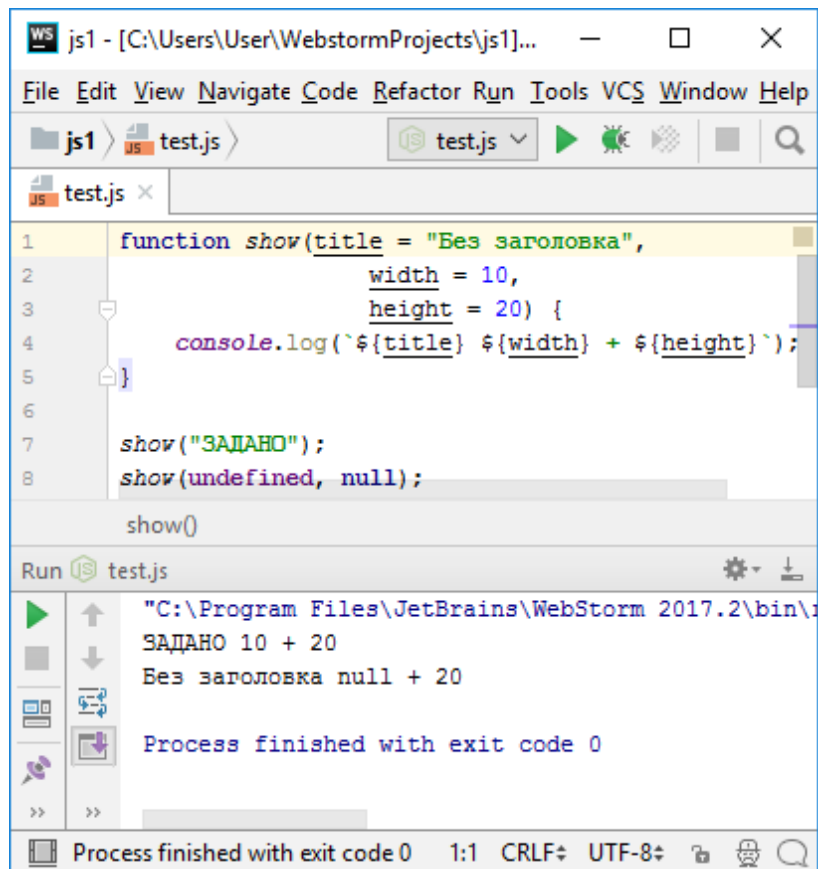
Функции (3), по умолчанию

42

```
function show(title = "Без заголовка", width = 10, height = 20) {  
    console.log(`${title} ${width} + ${height}`);  
}
```

```
show("ЗАДАНО");
```

```
show(undefined, null);
```



The screenshot shows an IDE window titled 'js1 - [C:\Users\User\WebstormProjects\js1]...'. The editor displays a JavaScript function `show` with default parameters `title = "Без заголовка"`, `width = 10`, and `height = 20`. The function logs a string template ``${title} ${width} + ${height}``. Below the function definition, two calls are made: `show("ЗАДАНО");` and `show(undefined, null);`. The Run window at the bottom shows the output of these calls: `ЗАДАНО 10 + 20` and `Без заголовка null + 20`. The process finished with exit code 0.

```
1 function show(title = "Без заголовка",  
2           width = 10,  
3           height = 20) {  
4     console.log(`${title} ${width} + ${height}`);  
5   }  
6  
7   show("ЗАДАНО");  
8   show(undefined, null);  
  
show()
```

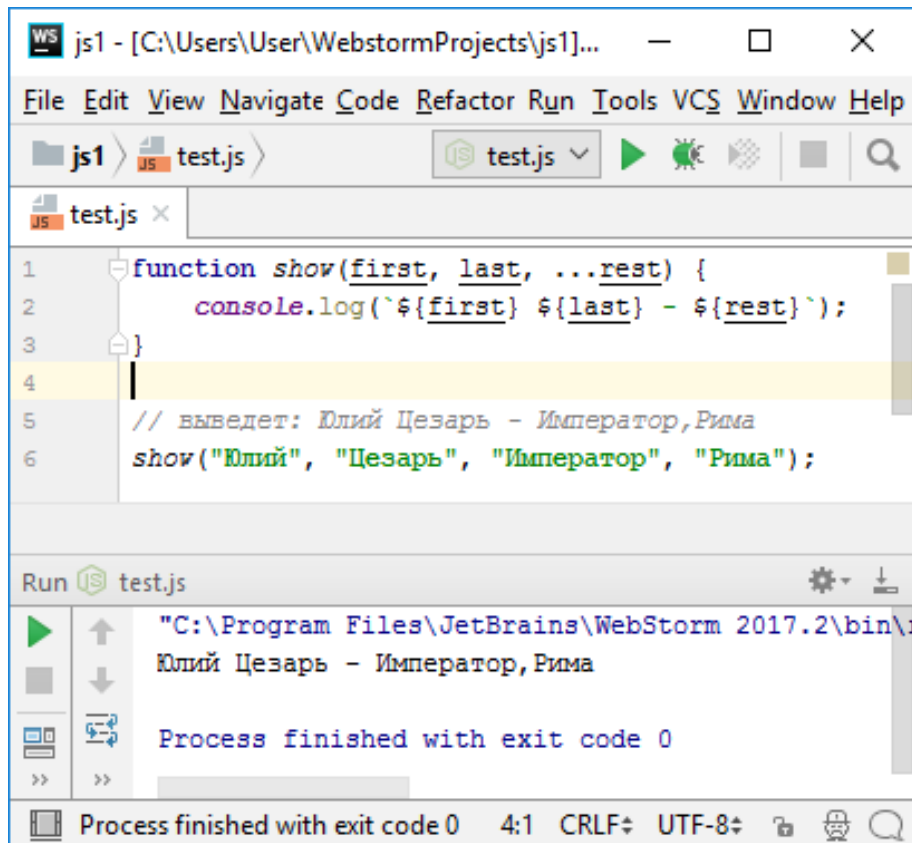
Run test.js

"C:\Program Files\JetBrains\WebStorm 2017.2\bin\...
ЗАДАНО 10 + 20
Без заголовка null + 20
Process finished with exit code 0

Функции (4), остаточные параметры⁴³

```
function show(first, last, ...rest) {  
    console.log(`${first} ${last} - ${rest}`);  
}
```

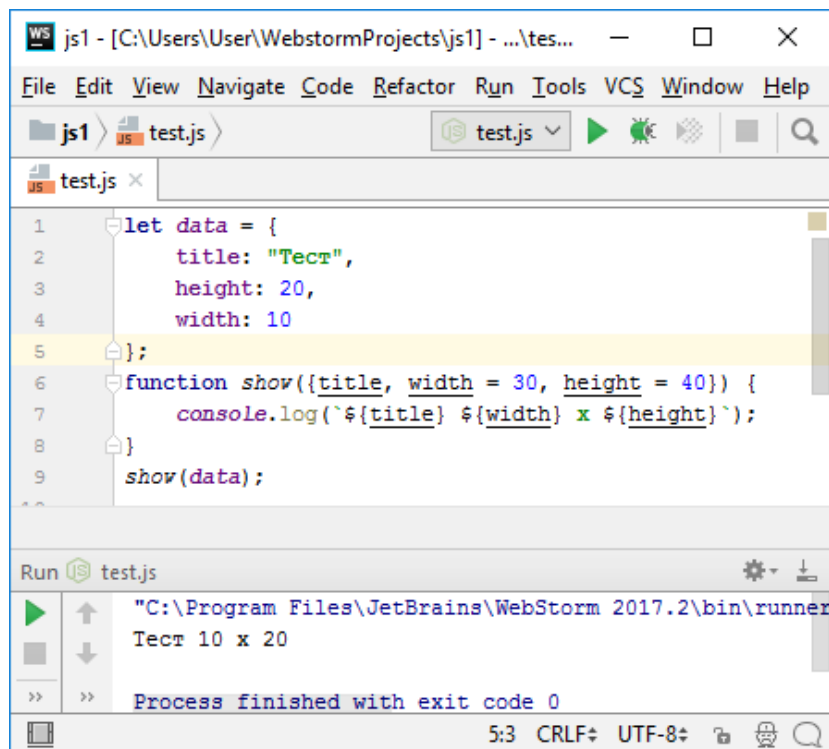
```
// выведет: Юлий Цезарь - Император, Рима  
show("Юлий", "Цезарь", "Император", "Рима");
```



... ⇔ spread ⇔ rest
Неопределенное число
аргументов

ФУНКЦИИ (5), ОБЪЕКТ

```
let data = {  
  title: "Тест",  
  height: 20,  
  width: 10  
};  
  
function show({title, width = 30, height = 40}) {  
  console.log(`${title} ${width} x ${height}`);  
}  
  
show(data);
```



Функции (6), имя функции

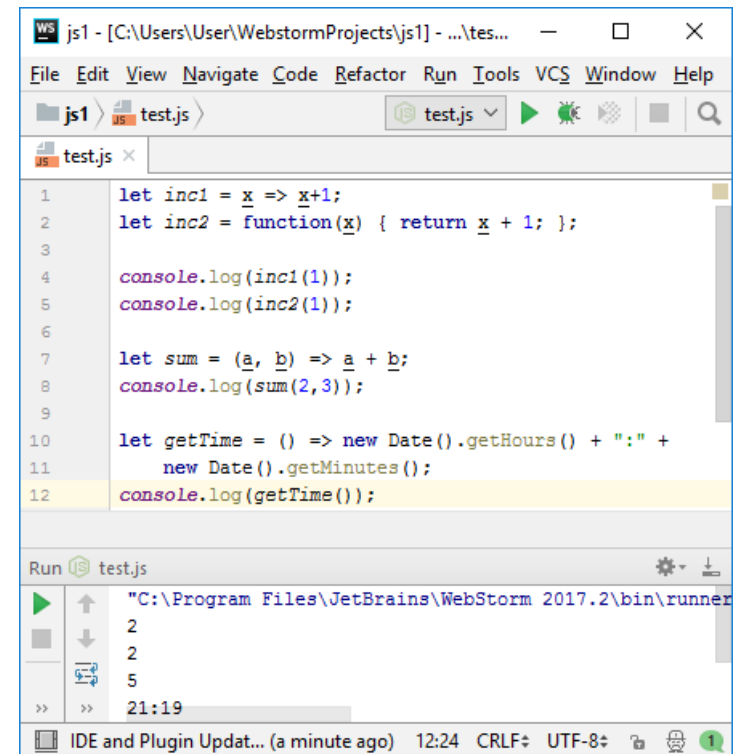
```
let data = { title: "Тест", height: 20, width: 10};
function show({title = "Без названия",
               width = 30, height = 40} = {}) {
    console.log(`${title} ${width} x ${height}`);
}
show(data);
show();
console.log(show.name);
```

```
js1 - [C:\Users\User\WebstormProjects\js1] - ...tes...
File Edit View Navigate Code Refactor Run Tools VCS Window Help
js1 test.js test.js
test.js x
1 let data = {
2     title: "Тест",
3     height: 20,
4     width: 10
5 };
6 function show({title = "Без названия",
7               width = 30, height = 40} = {}) {
8     console.log(`${title} ${width} x ${height}`);
9 }
10 show(data);
11 show();
12 console.log(show.name);
13
Run test.js
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\runner
Тест 10 x 20
Без названия 30 x 40
show
Process finished with exit code 0 5:3 CRLF UTF-8
```

Функции (7), стрелочные функции

46

```
let inc1 = x => x+1;
let inc2 = function(x) { return x + 1; };
console.log(inc1(1));
console.log(inc2(1));
let sum = (a, b) => a + b;
console.log(sum(2, 3));
let getTime = () => new Date().getHours() + ":" +
    new Date().getMinutes();
console.log(getTime());
```



The screenshot shows a webStorm IDE window titled 'js1 - [C:\Users\User\WebstormProjects\js1] - ...\tes...'. The editor displays the same JavaScript code as the previous block. Below the editor, the 'Run' panel shows the output of the code execution. The output consists of four lines: '2', '2', '5', and '21:19', which correspond to the values logged by the console.log statements in the code. The status bar at the bottom indicates 'IDE and Plugin Updat... (a minute ago)', '12:24', 'CRLF', 'UTF-8', and a green icon with the number '1'.

```
1 let inc1 = x => x+1;
2 let inc2 = function(x) { return x + 1; };
3
4 console.log(inc1(1));
5 console.log(inc2(1));
6
7 let sum = (a, b) => a + b;
8 console.log(sum(2, 3));
9
10 let getTime = () => new Date().getHours() + ":" +
11     new Date().getMinutes();
12 console.log(getTime());
```

Run test.js

"C:\Program Files\JetBrains\WebStorm 2017.2\bin\runner
2
2
5
21:19

IDE and Plugin Updat... (a minute ago) 12:24 CRLF UTF-8 1

Функции (8), видимость переменных, вложенные функции

```
let x = 2
let y = 3
function sum() {
  console.log(x + y)
}
sum() // 5
```

```
function sum2() {
  let x = y = 5
  function add() {
    console.log(x + y)
  }
  add()
}
sum2() // 10
```

Функции (9), рекурсия, arguments

48

```
let fact = function f(n) {  
  if(n < 1) return 1  
  let n1 = fact(n - 1) // Или f  
  let n2 = arguments.callee(n - 2)  
  return n1 + n2  
}  
fact(5) // 13
```

arguments – массив параметров функции
arguments.length – длина массива
arguments[0] – первый элемент массива

Функции (A), замыкания (Closures)

49

```
function obj(name) {  
  function get() {  
    return name  
  }  
  return get  
}  
let o = obj("Test")  
o() // "Test"
```

```
let getCode = function() {  
  // "Снаружи" code не изменить  
  let code = '12345678'  
  return function() {  
    return code  
  }  
}()  
getCode() // Вернёт code
```

```
function book(title) {  
  return {  
    setTitle(newTitle) {  
      title = newTitle  
    },  
    getTitle() {  
      return title  
    }  
  }  
}  
let b = book("Title 1")  
b.getTitle() // "Title 1"  
b.setTitle("Title 2")  
b.getTitle() // "Title 2"
```

Функции (В), this, setTimeout

```
function User1(years) {  
  this.age = years  
  setTimeout(function() {  
    console.log(`Age ${this.age}`)  
  }, 500)  
}  
new User1(5) // Age undefined
```

```
function User2(years) {  
  let self = this  
  self.age = years  
  setTimeout(function() {  
    console.log(`Age ${self.age}`)  
  }, 500)  
}  
new User2(6) // Age 6
```

```
function User3(years) {  
  this.age = years  
  setTimeout(() => {  
    console.log(`Age ${this.age}`)  
  }, 500)  
}  
new User3(7) // Age 7
```

Операторы

- `+` `-` `/` `*` `%`
 - `<<` `>>` `>>>`
 - `&` `^` `|` `~`
 - `and`, `xor`, `or`, `not`
 - `=` `+=` `-=` `/=` `*=` `%=` `<<=` `>>=` `>>>=` `&=` `^=` `|=`
 - `==` `===` `!=` `!==`
 - `>` `>=` `<` `<=`
 - `++` `--`
 - `&&` `||` `!`
 - `?:`
 - `,`
 - **delete** – удаление
 - **typeof** – определение типа (встроен.)
 - **void** – вычисление без return
 - **in** – проверка наличия свойства
 - **instanceof** – определение типа (user)
 - **new** – создание экземпляра
 - **super** – обращение к «родителю»
- ```
let y = 2 > 3 ? "a" : "b"
let x = (1+1, 2+2, 5)
console.log(x, y) // 5 b
```

```
let fun = (x)=>{console.log(x)} 51
let str = "round"
let int = 1
let date = new Date()
typeof fun // "function"
typeof str // "string"
typeof int // "number"
typeof date // "object"
typeof true // "boolean"
typeof null // "object"
typeof 42 // "number"
typeof {} // "object"
```

```
a = 42
var b = 3.14
delete a // true
delete b // false
obj = new Number()
obj.d = 42
delete Math.E // false
delete obj.d // true
delete obj // true
```

# Встроенные объекты

52

## Number

Number.MAX\_VALUE  
Number.MIN\_VALUE  
Number.NaN  
Number.NEGATIVE\_INFINITY  
Number.POSITIVE\_INFINITY  
Number.parseFloat()  
Number.parseInt()  
Number.isFinite()  
Number.isInteger()  
Number.isNaN()  
toExponential()  
toFixed()  
toPrecision()

## Math

Math.PI  
Math.E  
abs()  
sin(), cos(), tan()  
asin(), acos(), atan(), atan2()  
sinh(), cosh(), tanh()  
asinh(), acosh(), atanh()  
pow(), exp(), expm1(), log10(), log1p(), log2()  
floor(), ceil()  
min(), max()  
random()  
round()  
sqrt(), cbrt(), hypot()  
sign()

## Date

setTime  
parse

...

# Регулярные выражения

```
let reg = /p(.+)p/g
reg = new RegExp("p(.+)p", "g") // Альтернатива
reg.exec("Проверка")
// ['ровер', 'ове', index: 1, input: 'Проверка', groups: undefined]
"Проверка".match(reg) // ['ровер']
```

## RegExp

exec

test

## String

match

search

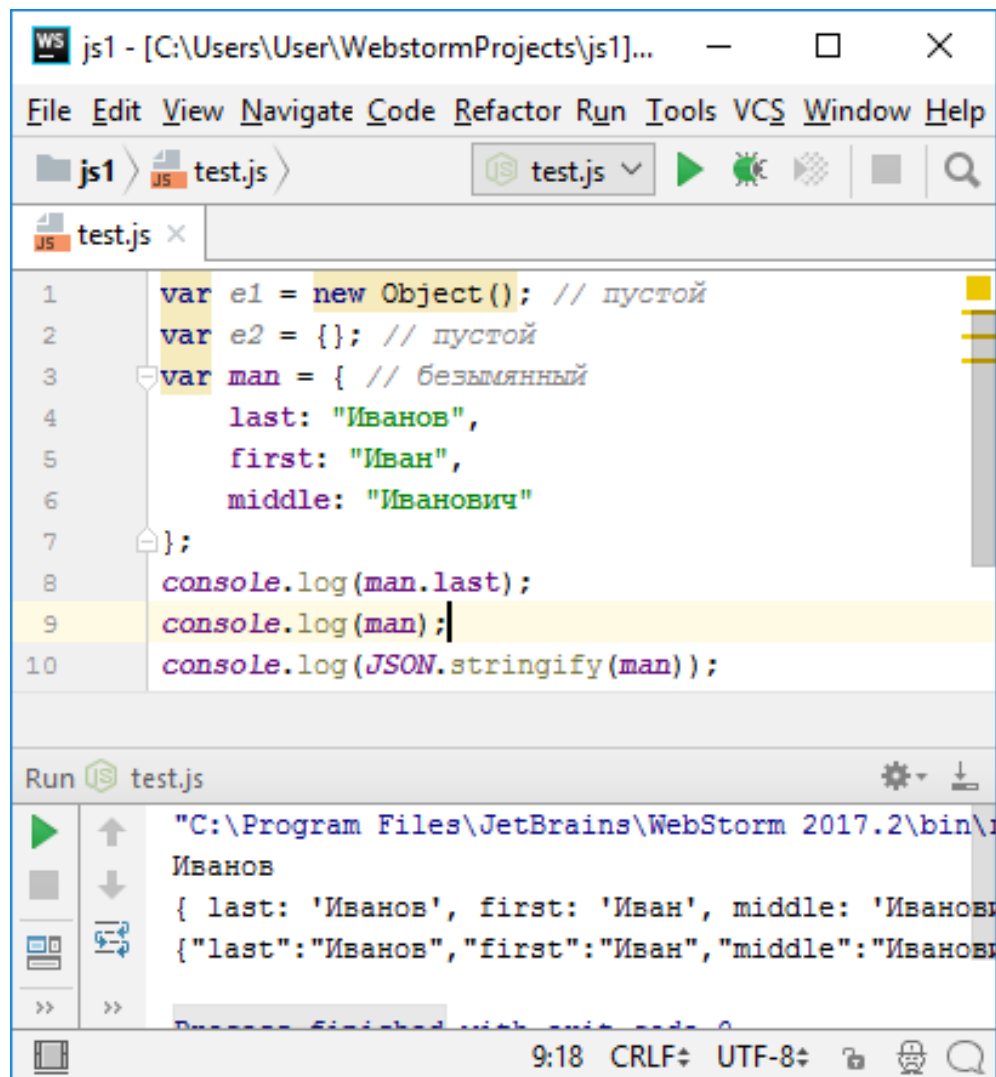
replace

split

# Создание объектов (1), Object

```
var e1 = new Object(); // пустой
var e2 = {}; // пустой
var man = { // безымянный
 last: "Иванов",
 first: "Иван",
 middle: "Иванович"
};
console.log(man.last);
console.log(man);
console.log(JSON.stringify(man));
```

JSON



The screenshot shows an IDE window titled 'js1 - [C:\Users\User\WebstormProjects\js1]...'. The editor displays the same JavaScript code as the previous block. The 'Run' button (a green play icon) is visible in the top right of the editor area. Below the editor, the 'Run' console shows the output of the code execution:

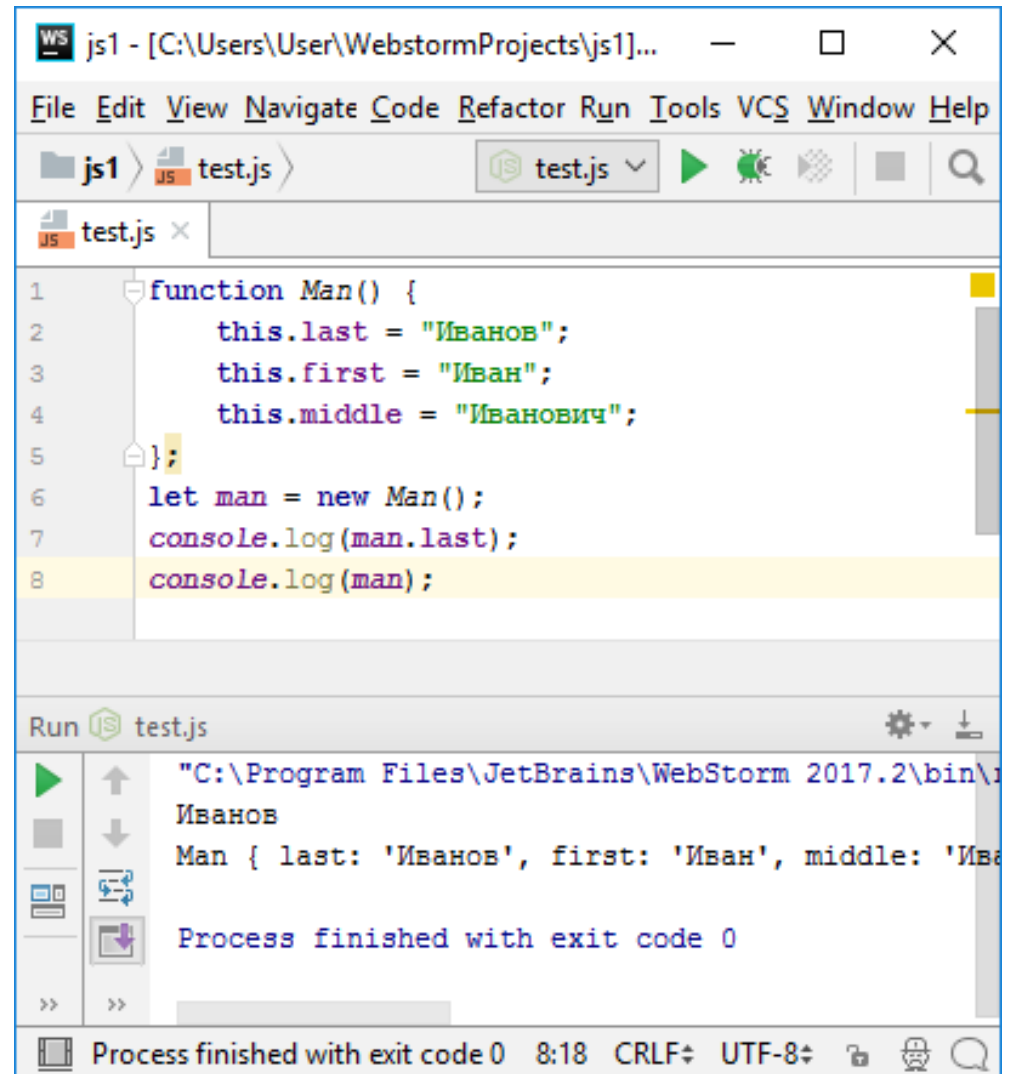
```
Run test.js
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\...
Иванов
{ last: 'Иванов', first: 'Иван', middle: 'Иванович' }
{"last":"Иванов","first":"Иван","middle":"Иванович"}
```

The status bar at the bottom indicates the time is 9:18, the encoding is CRLF, and the character set is UTF-8.

# Создание объектов (2), литерал

55

```
function Man() {
 this.last = "Иванов";
 this.first = "Иван";
 this.middle = "Иванович";
};
let man = new Man();
console.log(man.last);
console.log(man);
```



# Создание объектов (3), create

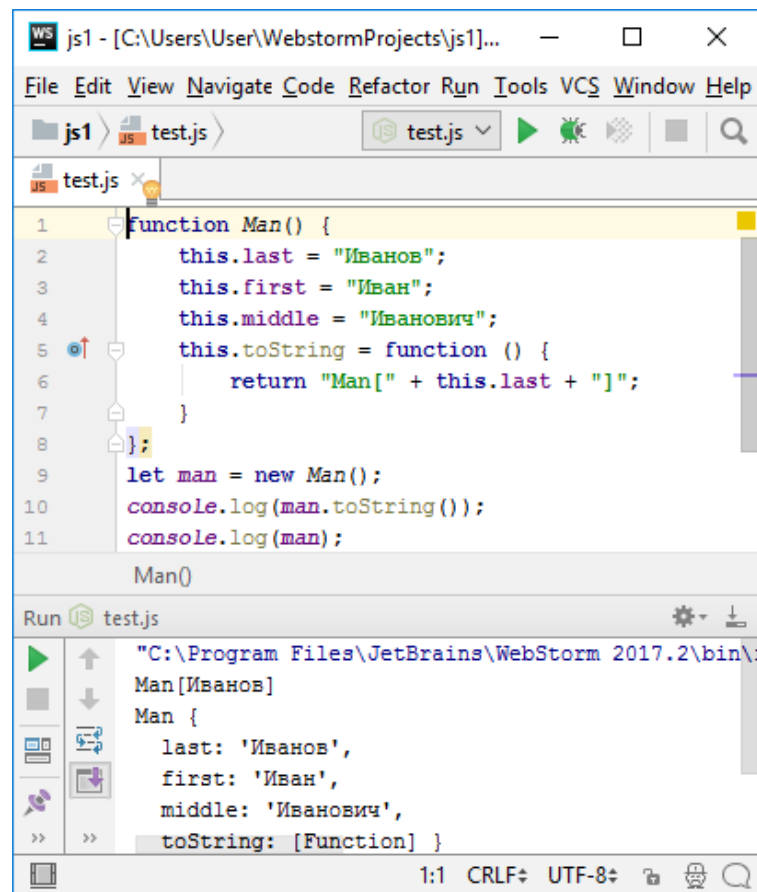
```
let Animal = {
 type: 'Cat'
}
let cat = Object.create(Animal);
let fish = Object.create(Animal);
fish.type = 'Fish';
console.log(cat.type); // Cat
console.log(fish.type); // Fish
```



# Метод в объекте (1)

```
function Man() {
 this.last = "Иванов";
 this.first = "Иван";
 this.middle = "Иванович";
 this.toString = function () {
 return "Man[" + this.last + "]";
 }
};

let man = new Man();
console.log(man.toString());
console.log(man);
```

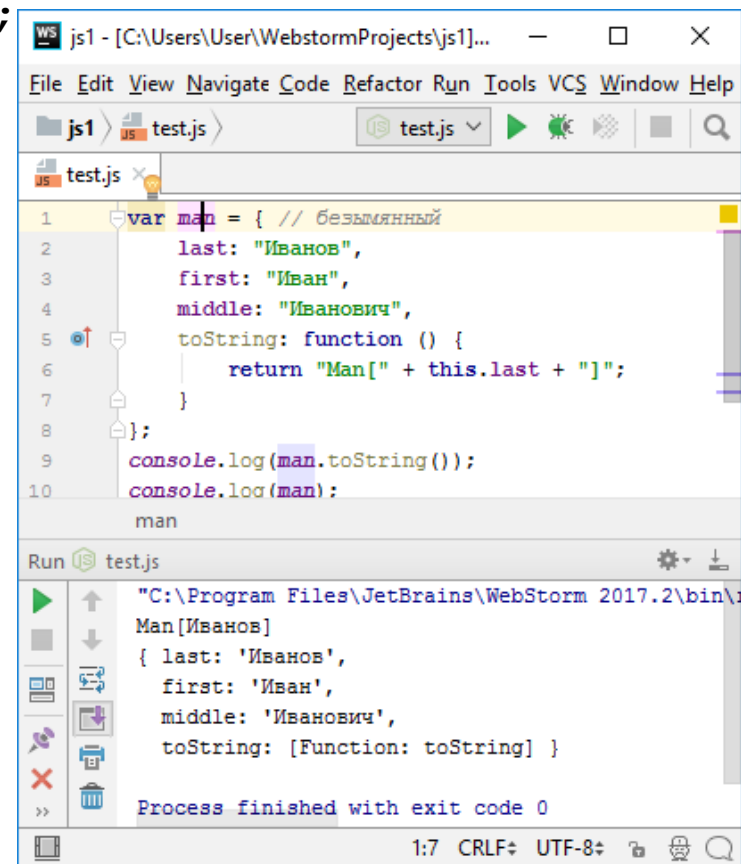


# Метод в объекте (2)

58

```
var man = { // безымянный
 last: "Иванов",
 first: "Иван",
 middle: "Иванович",
 toString: function () {
 return "Man[" + this.last + "]";
 }
};

console.log(man.toString());
console.log(man);
```

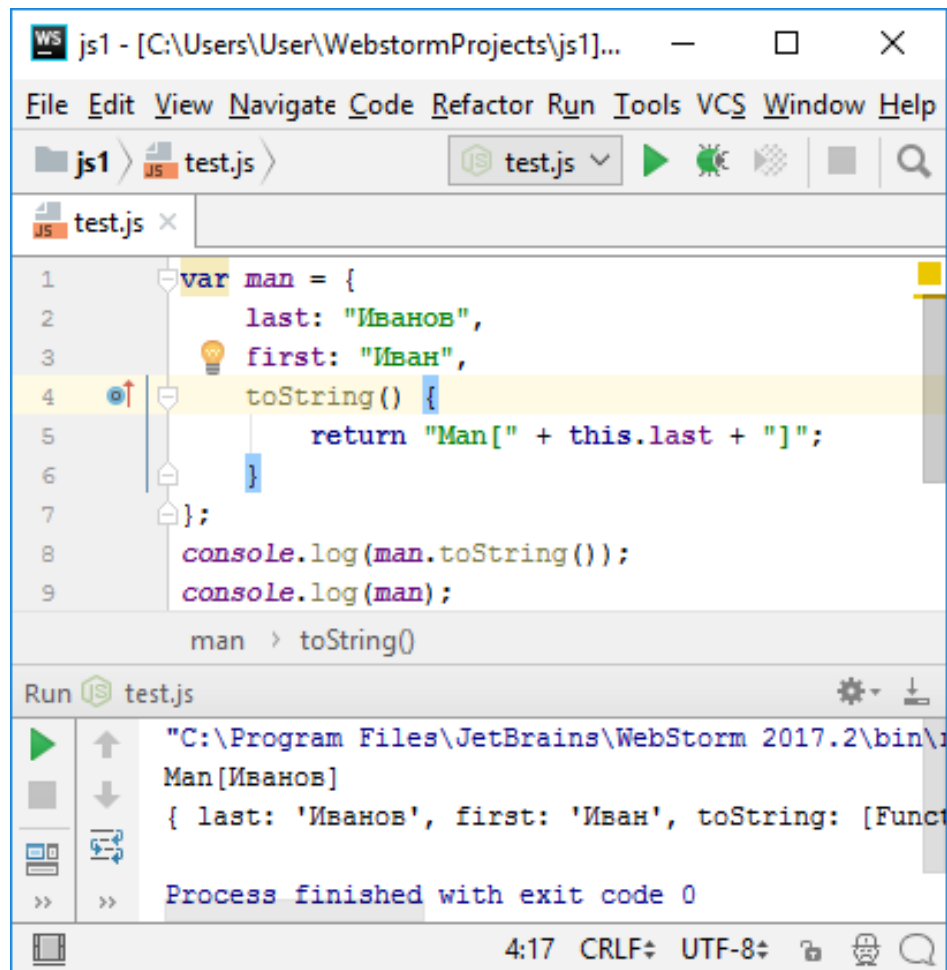


# Метод в объекте (3)

59

```
var man = {
 last: "ИВАНОВ",
 first: "ИВАН",
 toString() {
 return "Man[" + this.last + "]";
 }
};

console.log(man.toString());
console.log(man);
```

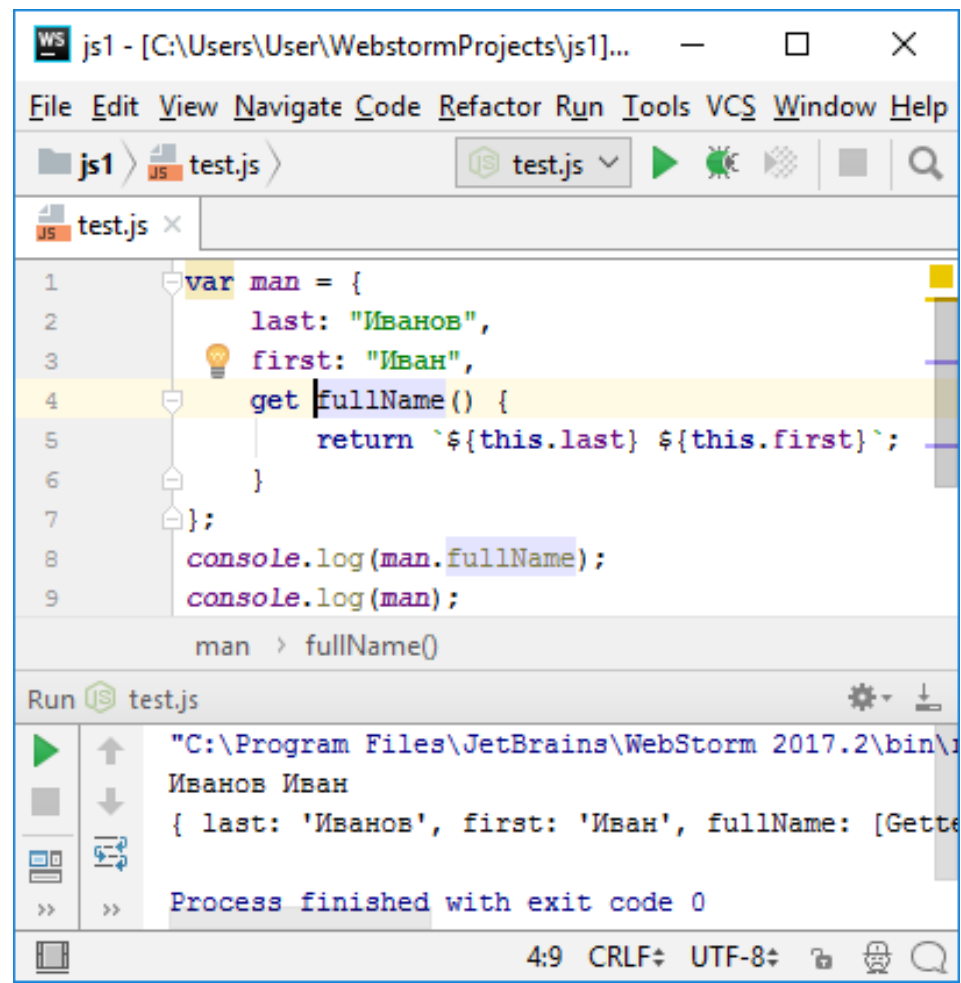


# Метод в объекте (4), get/set

60

```
var man = {
 last: "ИВАНОВ",
 first: "Иван",
 get fullName() {
 return `${this.last} ${this.first}`;
 }
};

console.log(man.fullName);
console.log(man);
```



The screenshot shows a code editor window with the following code in `test.js`:

```
1 var man = {
2 last: "ИВАНОВ",
3 first: "Иван",
4 get fullName() {
5 return `${this.last} ${this.first}`;
6 }
7 };
8 console.log(man.fullName);
9 console.log(man);
```

The console output shows the result of the execution:

```
man > fullName()
ИВАНОВ Иван
{ last: 'ИВАНОВ', first: 'Иван', fullName: [Getter]
Process finished with exit code 0
```

# Метод в объекте (5), []

61

```
let methodName = "name";
```

```
let man = {
```

```
 // в квадратных скобках может быть любое
```

```
 // выражение, которое должно вернуть
```

```
 // название метода
```

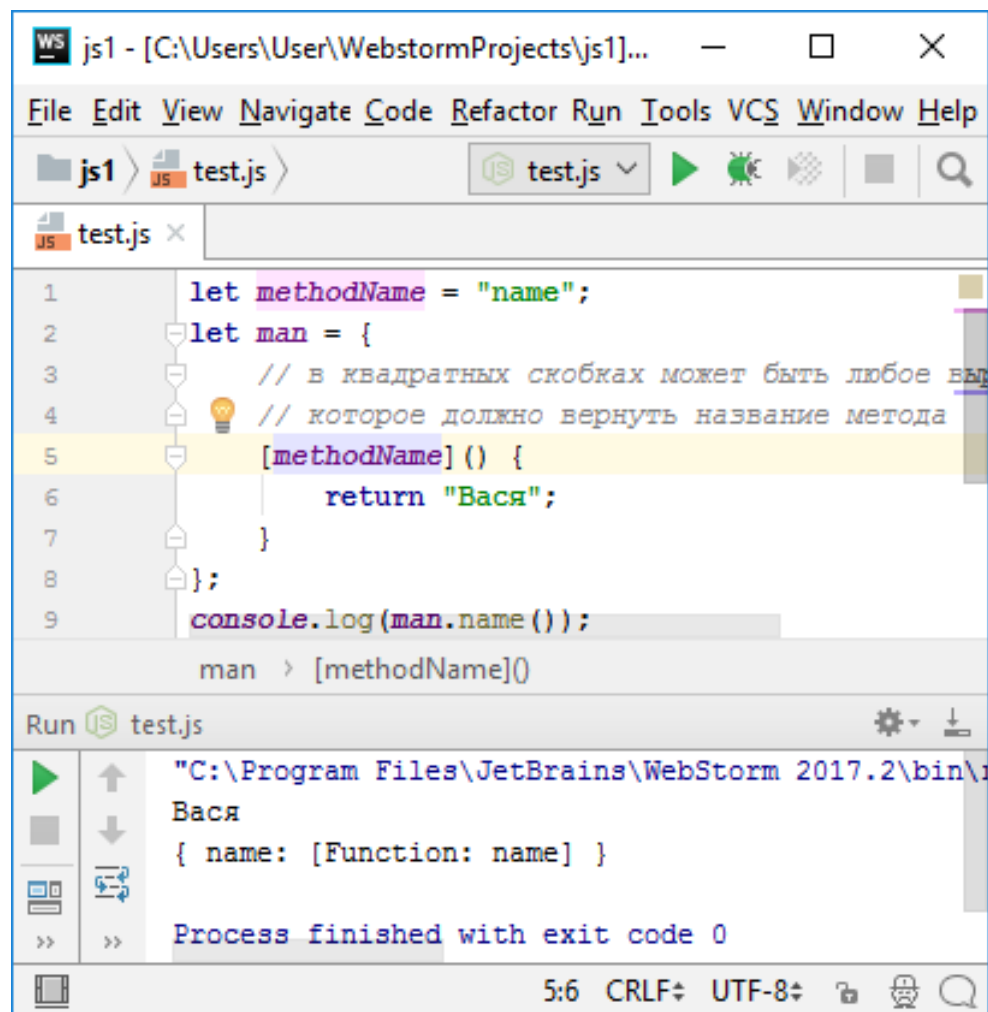
```
 [methodName] () {
 return "Вася";
```

```
 }
```

```
};
```

```
console.log(man.name());
```

```
console.log(man);
```

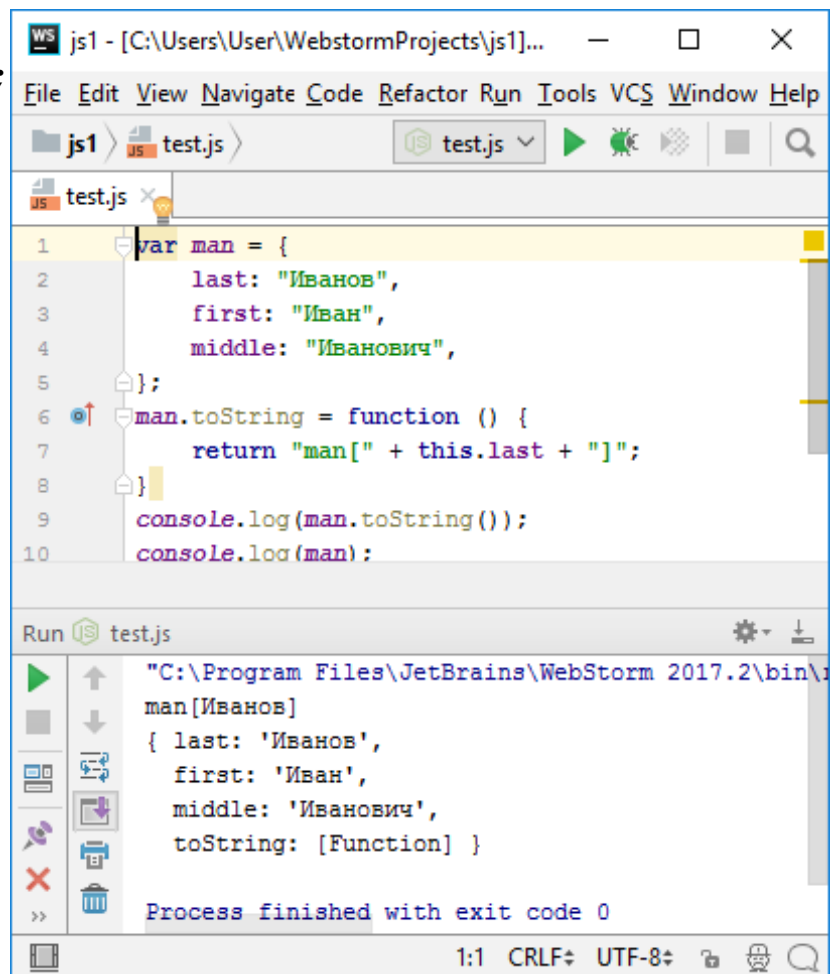


# Добавление метода

```
var man = {
 last: "Иванов",
 first: "Иван",
 middle: "Иванович",
};

man.toString = function () {
 return "man[" + this.last + "]";
}

console.log(man.toString());
console.log(man);
```



# Поля объекта (1)

```
var man = {
 last: "Иванов",
 first: "Иван",
 middle: "Иванович",
};
man.age = 24;
delete man.middle;
for (var key in man)
 console.log(man[key]);
console.log(man);
```

## Список свойств объекта:

- циклы **for...in**

Перечисляемые свойства объекта и прототипов

- **Object.keys(obj)**

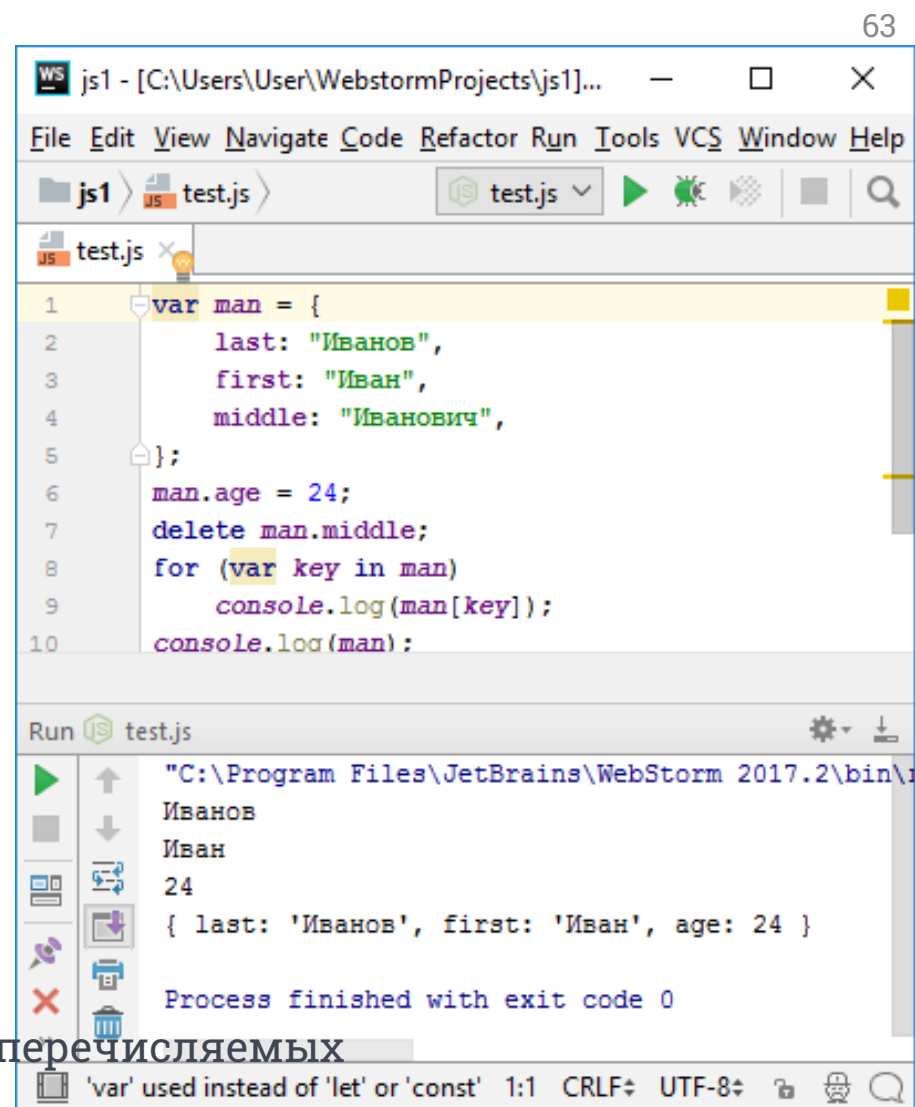
Массив со всеми собственными именами перечисляемых свойств объекта man.

- **Object.getOwnPropertyNames(obj)**

Массив, содержащий все имена своих свойств (перечисляемых и неперечисляемых) объекта man.

## Проверка наличия свойства:

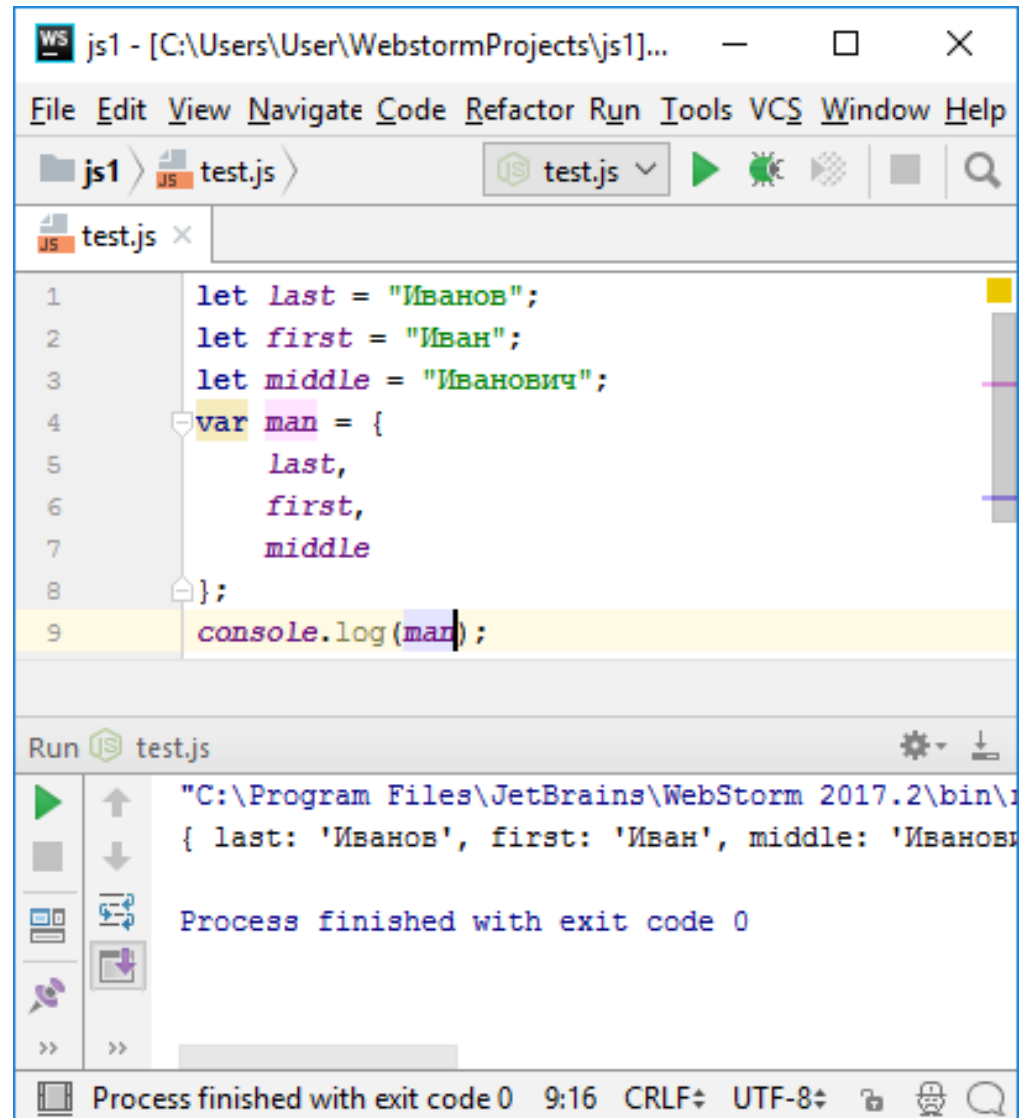
- **obj.hasOwnProperty(propertyName)**



# Поля объекта (2)

64

```
let last = "Иванов";
let first = "Иван";
let middle = "Иванович";
var man = {
 last,
 first,
 middle
};
console.log(man);
```



The screenshot shows an IDE window titled 'js1 - [C:\Users\User\WebstormProjects\js1]...'. The editor displays the same JavaScript code as the previous block. The file 'test.js' is open. The code is as follows:

```
1 let last = "Иванов";
2 let first = "Иван";
3 let middle = "Иванович";
4 var man = {
5 last,
6 first,
7 middle
8 };
9 console.log(man);
```

Below the editor, the 'Run' tab is active, showing the output of the execution. The output is:

```
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\...
{ last: 'Иванов', first: 'Иван', middle: 'Иванович' }
Process finished with exit code 0
```

The status bar at the bottom indicates 'Process finished with exit code 0', '9:16', 'CRLF', 'UTF-8', and other settings.

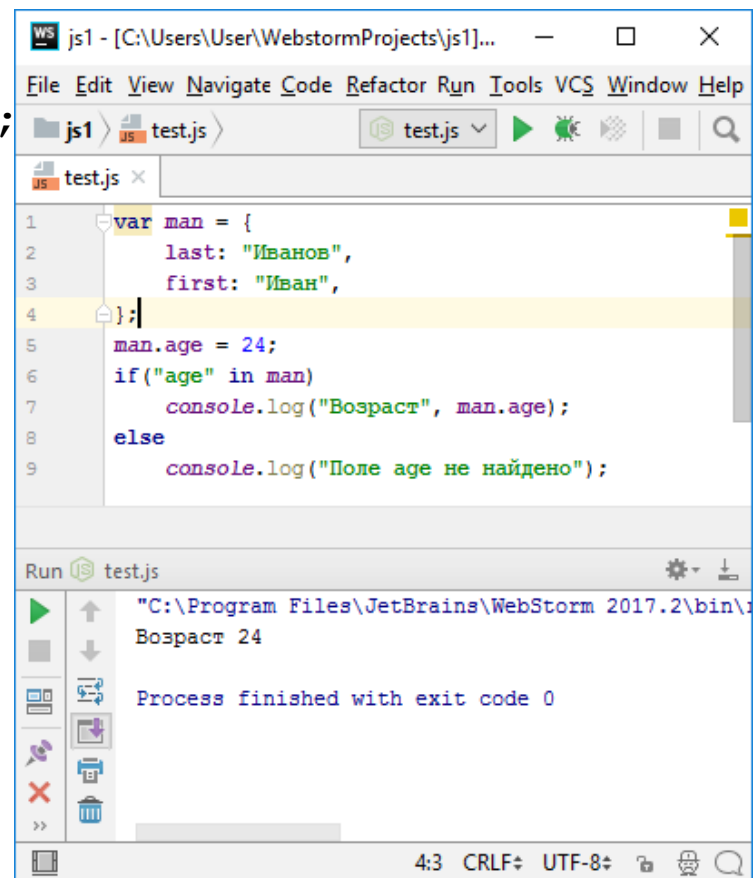


# Поиск поля по имени

65

```
var man = {
 last: "Иванов",
 first: "Иван",
};

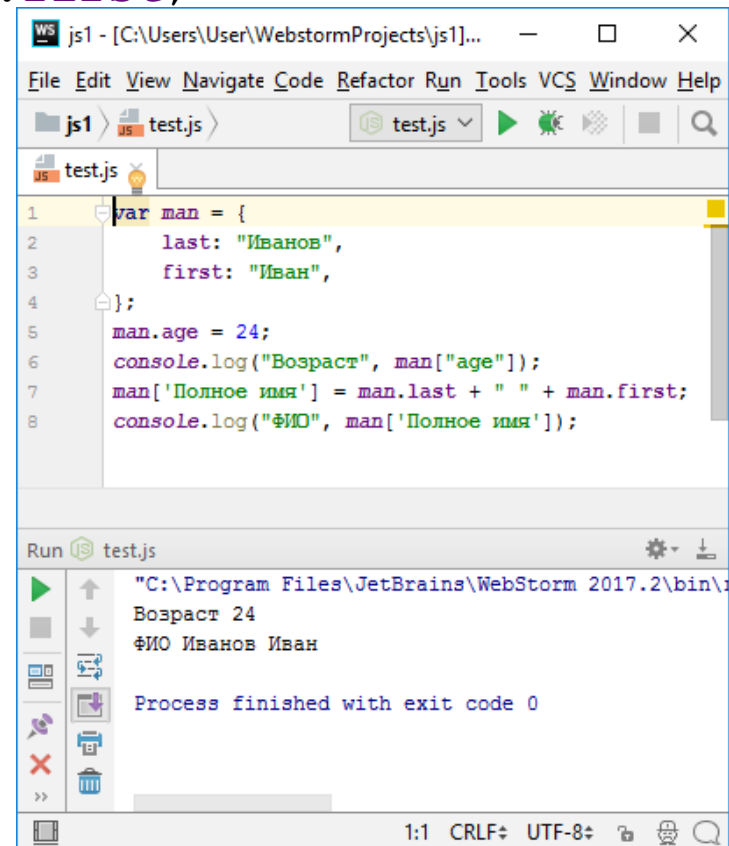
man.age = 24;
if("age" in man)
 console.log("Возраст", man.age);
else
 console.log("Поле age не найдено");
```



# Произвольные поля объекта

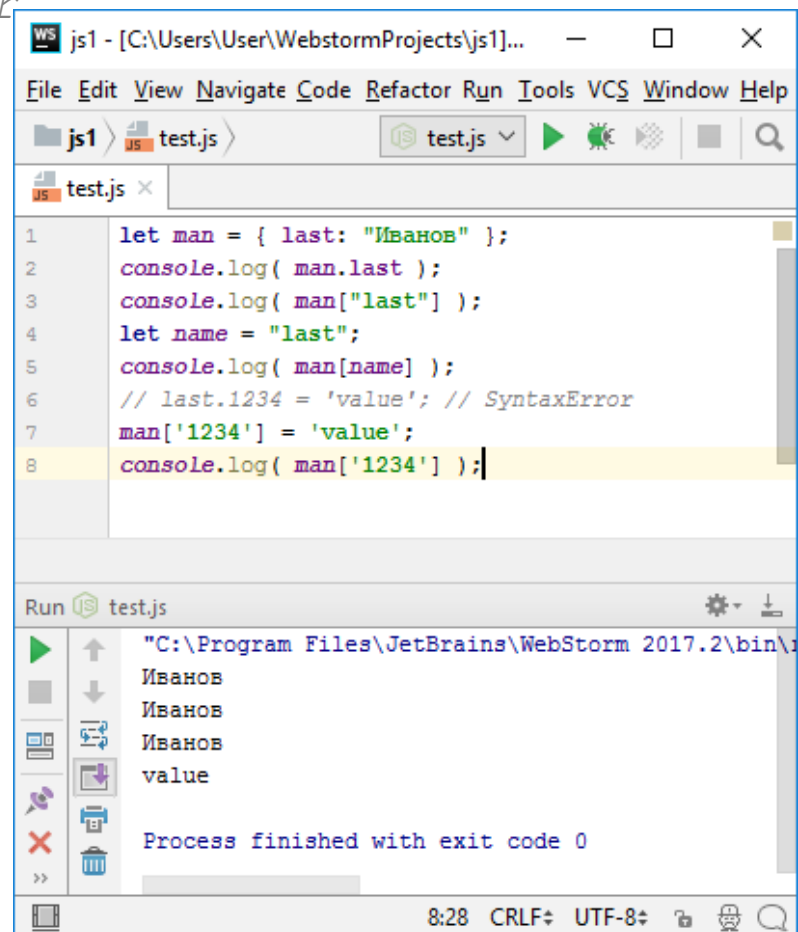
66

```
var man = {
 last: "Иванов",
 first: "Иван",
};
man.age = 24;
console.log("Возраст", man["age"]);
man['Полное имя'] = man.last + " " + man.first;
console.log("ФИО", man['Полное имя']);
```



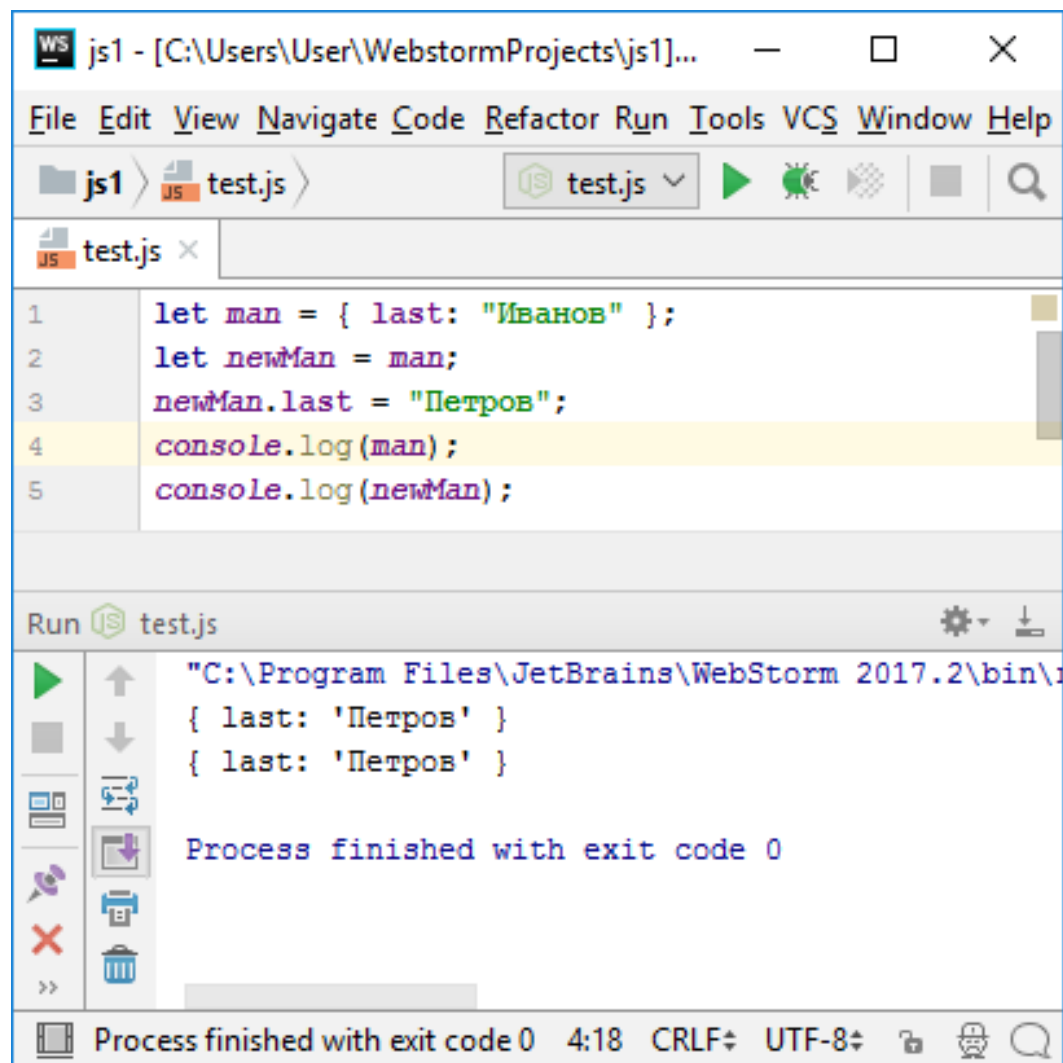
# Доступ к полям объекта

```
let man = { last: "Иванов" };
console.log(man.last);
console.log(man["last"]);
let name = "last";
console.log(man[name]);
// last.1234 = 'value'; // SyntaxError
man['1234'] = 'value';
console.log(man['1234']);
```



# Передача объекта по ссылке

```
let man = { last: "Иванов" };
let newMan = man;
newMan.last = "Петров";
console.log(man);
console.log(newMan);
```

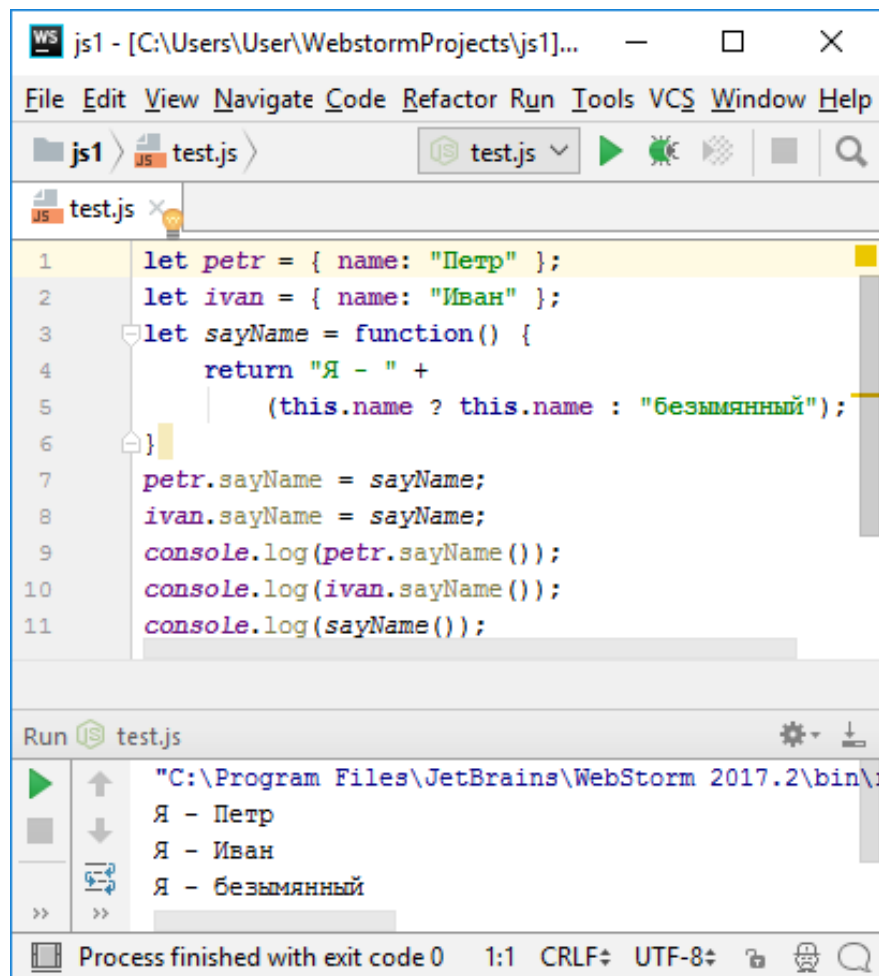


# Переприсваивание методов у объектов

69

```
let petr = { name: "Петр" };
let ivan = { name: "Иван" };
let sayName = function() {
 return "Я - " +
 (this.name ? this.name : "безымянный");
}

petr.sayName = sayName;
ivan.sayName = sayName;
console.log(petr.sayName());
console.log(ivan.sayName());
console.log(sayName());
```



The screenshot shows an IDE window titled 'js1 - [C:\Users\User\WebstormProjects\js1]...'. The editor displays the same JavaScript code as the previous block. Below the editor, the 'Run' button is clicked, and the output console shows the results of the execution:

```
Run test.js
"Я - Петр
Я - Иван
Я - безымянный
Process finished with exit code 0
```

# Object.assign

```
let user = { name: "Вася" };
let visitor = { isAdmin: false, visits: true };
let admin = { isAdmin: true };
```

```
Object.assign(user, visitor, admin);
```

```
console.log(user);
```

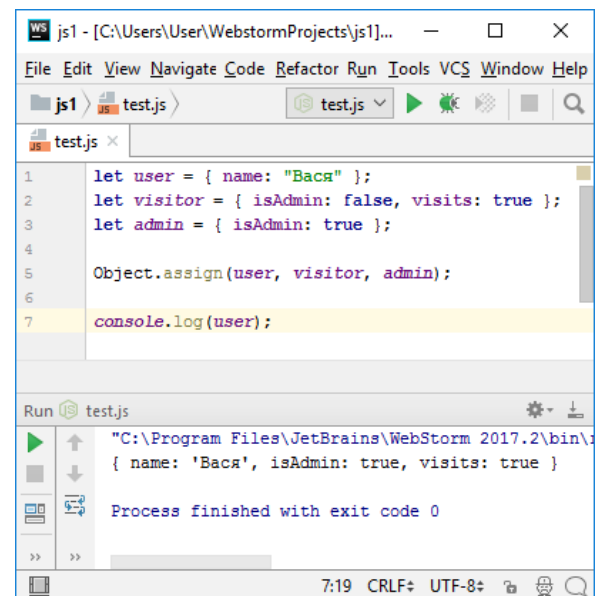
## Клонирование

```
let user = { name: "Вася" };
let visitor = { isAdmin: false, visits: true };
let admin = { isAdmin: true };
```

```
Object.assign(user, visitor, admin);
```

```
let clone = Object.assign({}, user);
```

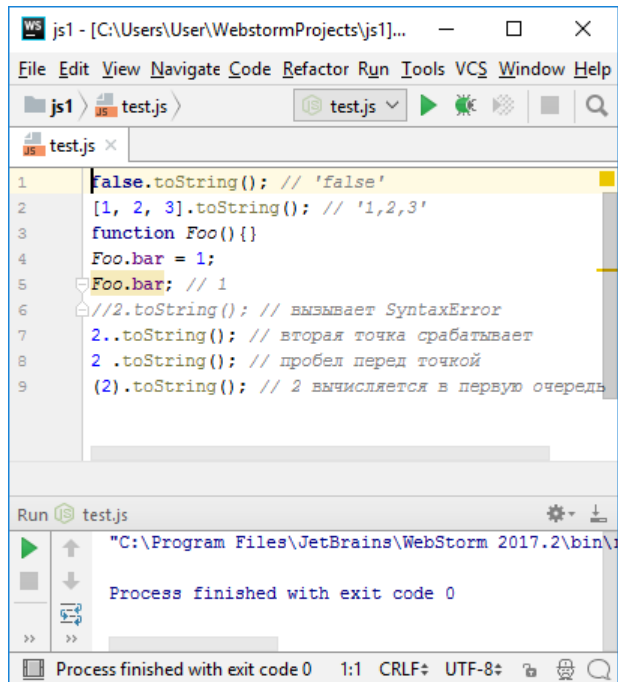
```
console.log(clone);
```



# Все сущности ведут себя как объекты

71

```
false.toString(); // 'false'
[1, 2, 3].toString(); // '1,2,3'
function Foo() {}
Foo.bar = 1;
Foo.bar; // 1
//2.toString(); // вызывает SyntaxError
2..toString(); // вторая точка срабатывает
2 .toString(); // пробел перед точкой
(2).toString(); // 2 вычисляется в первую очередь
```



# Прототипы

```
function Employee() {}
```

```
function Manager() {
 Employee.call(this)
}
```

```
//создаём пустой объект с прототипом от Employee
//и используем этот объект как прототип для Manager
```

```
Manager.prototype = Object.create(Employee.prototype)
Manager.prototype.constructor = Manager
```

```
function Worker() {
 Employee.call(this)
}
```

```
Worker.prototype = Object.create(Employee.prototype)
Worker.prototype.constructor = Worker
```

Прототип объекта — это объект, используемый в качестве шаблона, с целью получить изначальные свойства для нового объекта

## Поиск свойства у объекта:

1. Проверяется, существует ли локальное свойство с запрашиваемым именем
2. Если локального свойства нет, проверяется цепочка прототипов (`__proto__`)
3. Если один из объектов в цепочке прототипов имеет свойство с запрашиваемым именем, то возвращается значение
4. Иначе — объект его не имеет

В ES5 для прототипа был метод геттер:

`Object.getPrototypeOf(obj)`

В ES-2015 также добавился сеттер:

`Object.setPrototypeOf(obj, newProto)`

...А также «узаконено» свойство `__proto__`, которое даёт прямой доступ к прототипу.



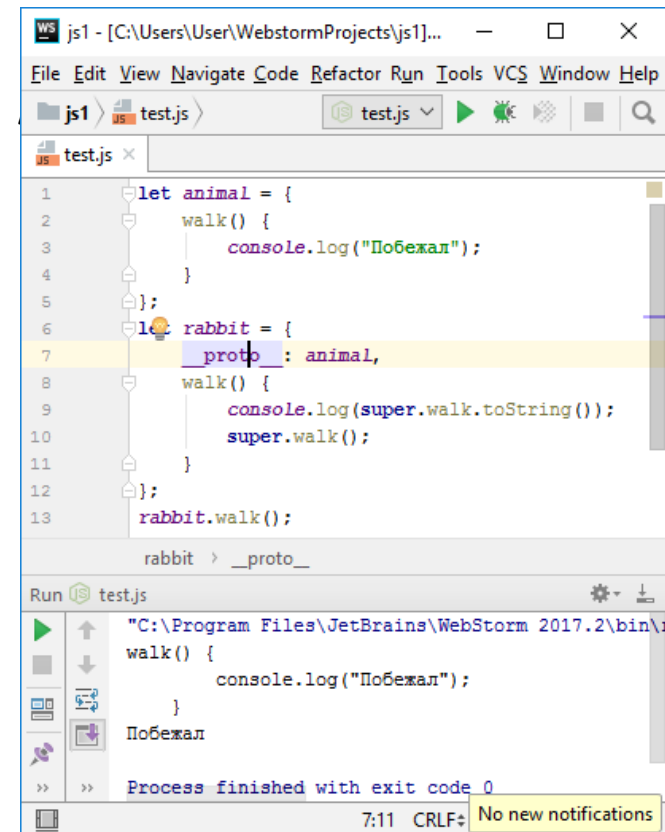
# super (1)

73

```
let animal = {
 walk() {
 console.log("Побежал");
 }
};

let rabbit = {
 __proto__: animal,
 walk() {
 console.log(super.walk.toString());
 super.walk();
 }
};

rabbit.walk();
```



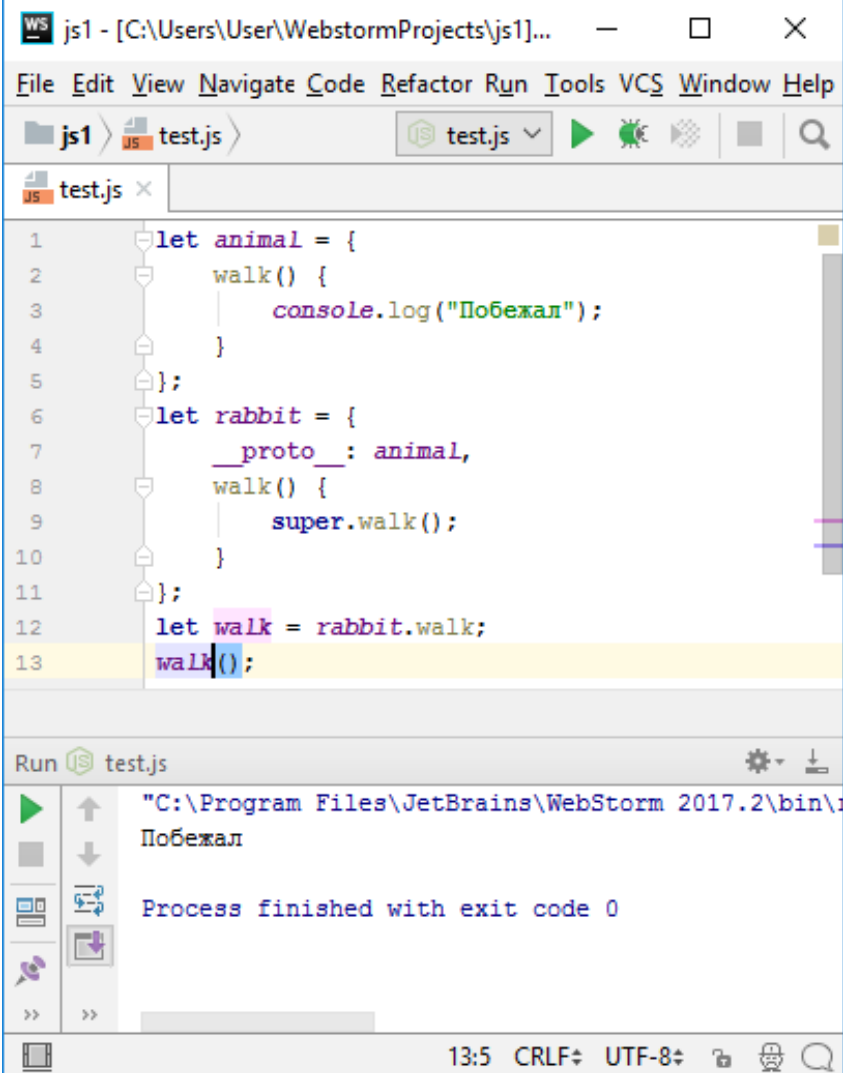
# super (2)

74

```
let animal = {
 walk() {
 console.log("Побежал");
 }
};

let rabbit = {
 __proto__: animal,
 walk() {
 super.walk();
 }
};

let walk = rabbit.walk;
walk();
```



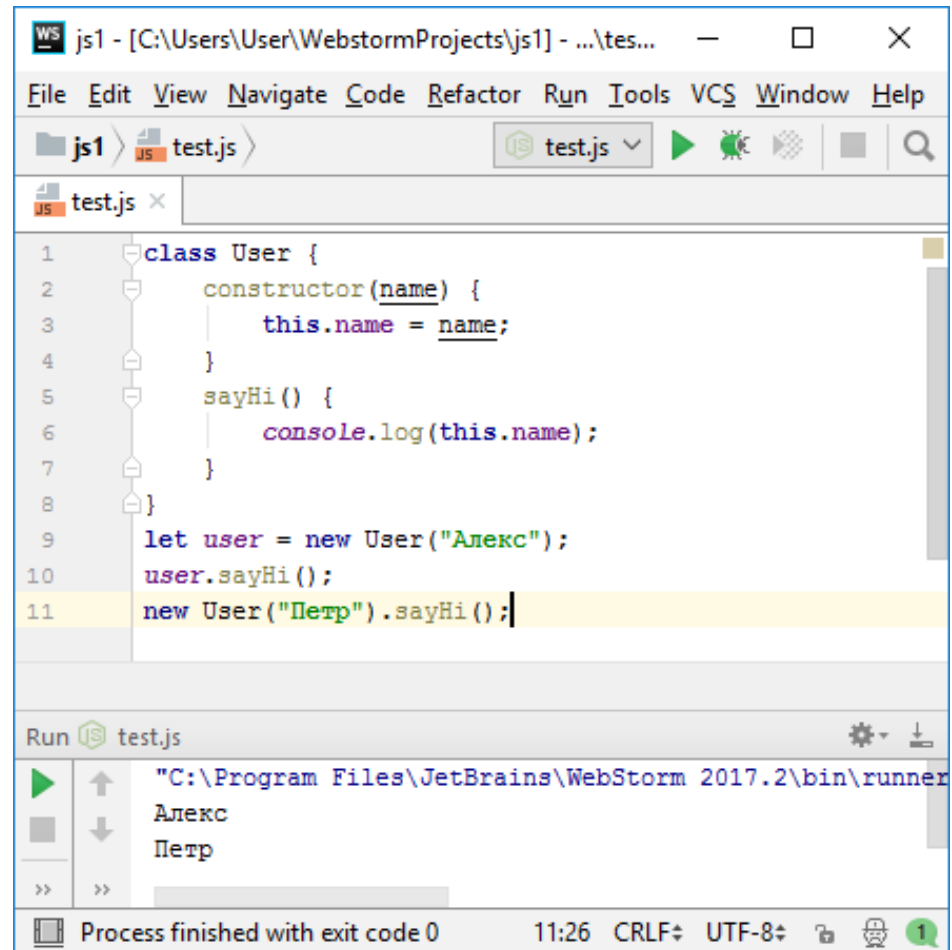
```
js1 - [C:\Users\User\WebstormProjects\js1]...
File Edit View Navigate Code Refactor Run Tools VCS Window Help
js1 > test.js > test.js
test.js x
1 let animal = {
2 walk() {
3 console.log("Побежал");
4 }
5 };
6 let rabbit = {
7 __proto__: animal,
8 walk() {
9 super.walk();
10 }
11 };
12 let walk = rabbit.walk;
13 walk();
Run test.js
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\...
Побежал
Process finished with exit code 0
13:5 CRLF UTF-8
```

# Классы (1)

75

```
class User {
 constructor(name) {
 this.name = name;
 }
 sayHi() {
 console.log(this.name);
 }
}

let user = new User("Алекс");
user.sayHi();
new User("Петр").sayHi();
```

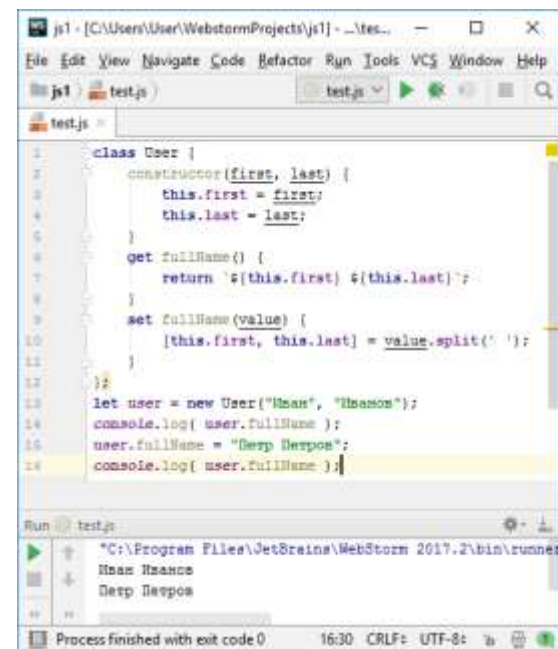


# Классы (2)

76

```
class User {
 constructor(first, last) {
 this.first = first;
 this.last = last;
 }
 get fullName() {
 return `${this.first} ${this.last}`;
 }
 set fullName(value) {
 [this.first, this.last] = value.split(' ');
 }
};

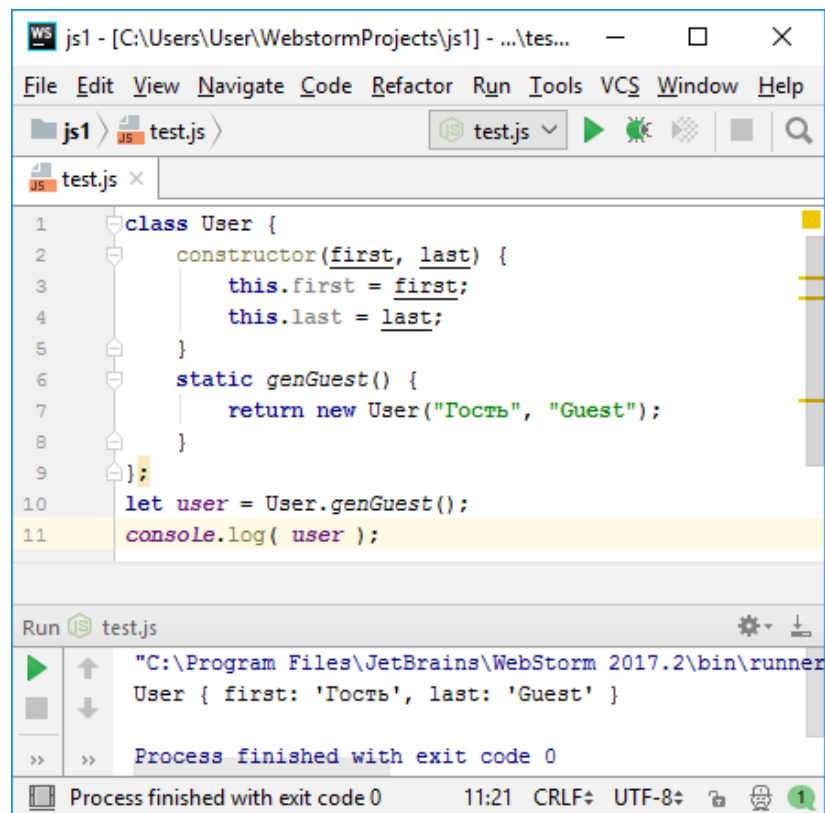
let user = new User("Иван", "Иванов");
console.log(user.fullName);
user.fullName = "Петр Петров";
console.log(user.fullName);
```



# Классы (3)

77

```
class User {
 constructor(first, last) {
 this.first = first;
 this.last = last;
 }
 static genGuest() {
 return new User("Гость", "Guest");
 }
};
let user = User.genGuest();
console.log(user);
```



```
js1 - [C:\Users\User\WebstormProjects\js1] - ...tes...
File Edit View Navigate Code Refactor Run Tools VCS Window Help
js1 test.js
test.js
1 class User {
2 constructor(first, last) {
3 this.first = first;
4 this.last = last;
5 }
6 static genGuest() {
7 return new User("Гость", "Guest");
8 }
9 };
10 let user = User.genGuest();
11 console.log(user);
Run test.js
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\runner
User { first: 'Гость', last: 'Guest' }
Process finished with exit code 0
Process finished with exit code 0 11:21 CRLF UTF-8
```

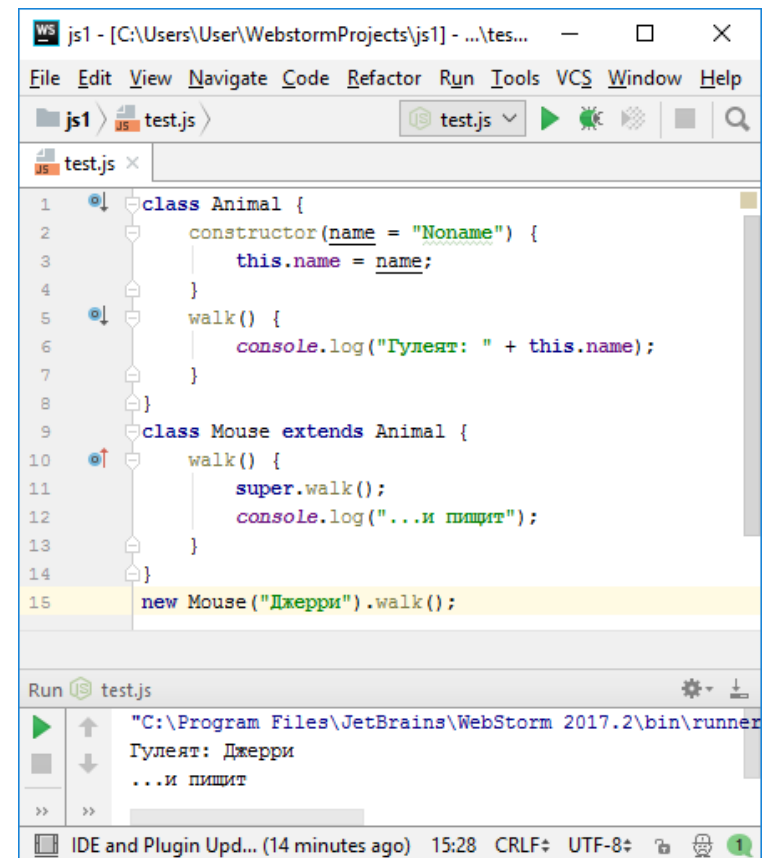
# Классы (4)

78

```
class Animal {
 constructor(name = "Noname") {
 this.name = name;
 }
 walk() {
 console.log("Гулеят: " + this.name);
 }
}
```

```
class Mouse extends Animal {
 walk() {
 super.walk();
 console.log("...и пищит");
 }
}
```

```
new Mouse("Джерри").walk();
```



# Статические методы и свойства

79

## Метод 1

```
class User {
 static staticMethod() {
 console.log(this === User);
 }
}
User.staticMethod(); // true
```

## Метод 2

```
class User { }
User.staticMethod = function() {
 console.log(this === User);
};
User.staticMethod(); // true
```

## СВОЙСТВО

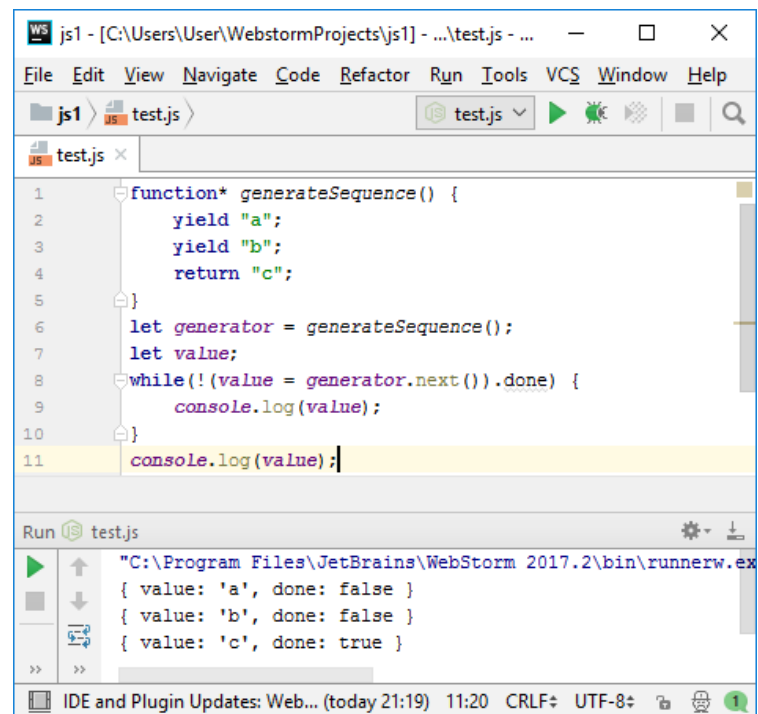
```
class Article {
 static publisher = "Франк";
}
console.log(Article.publisher); // Франк
```

# Генераторы (1)

```
function* generateSequence() {
 yield "a";
 yield "b";
 return "c";
}

let generator = generateSequence();
let value;
while(! (value = generator.next()).done) {
 console.log(value);
}

console.log(value);
```



The screenshot shows a webStorm IDE window with a file named 'test.js'. The code in the editor is identical to the code block on the left. The 'Run' button (a green play icon) is visible in the bottom right of the editor area. Below the editor, the 'Run' tool window is open, displaying the output of the program. The output consists of three lines, each representing a call to `console.log`:

```
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\runnerw.exe
{ value: 'a', done: false }
{ value: 'b', done: false }
{ value: 'c', done: true }
```

The status bar at the bottom of the IDE shows 'IDE and Plugin Updates: Web...' (today 21:19), '11:20', 'CRLF', 'UTF-8', and several icons.

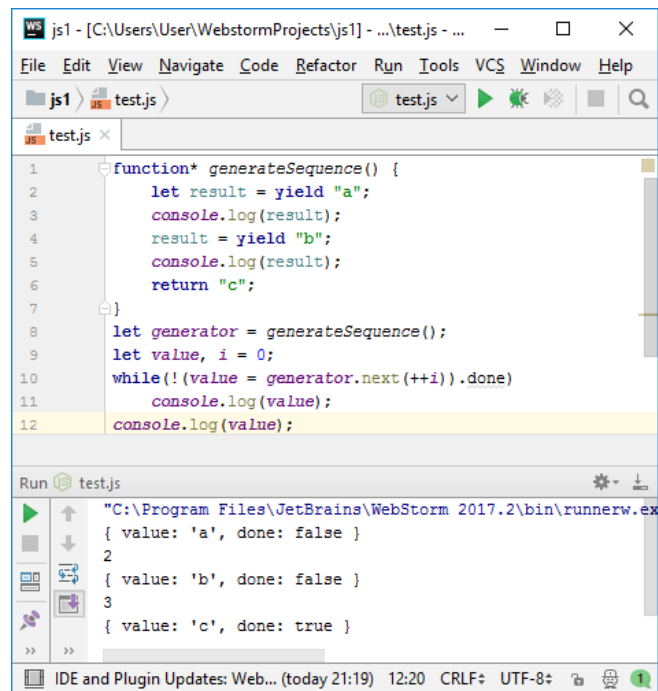


# Генераторы (2)

81

```
function* generateSequence() {
 let result = yield "a";
 console.log(result);
 result = yield "b";
 console.log(result);
 return "c";
}

let generator = generateSequence();
let value, i = 0;
while (!(value = generator.next(++i)).done) {
 console.log(value);
}
console.log(value);
```



The screenshot shows an IDE window with a file named `test.js`. The code in the editor is as follows:

```
1 function* generateSequence() {
2 let result = yield "a";
3 console.log(result);
4 result = yield "b";
5 console.log(result);
6 return "c";
7 }
8 let generator = generateSequence();
9 let value, i = 0;
10 while (!(value = generator.next(++i)).done) {
11 console.log(value);
12 }
13 console.log(value);
```

Below the editor, the `Run` tab is active, showing the output of the program:

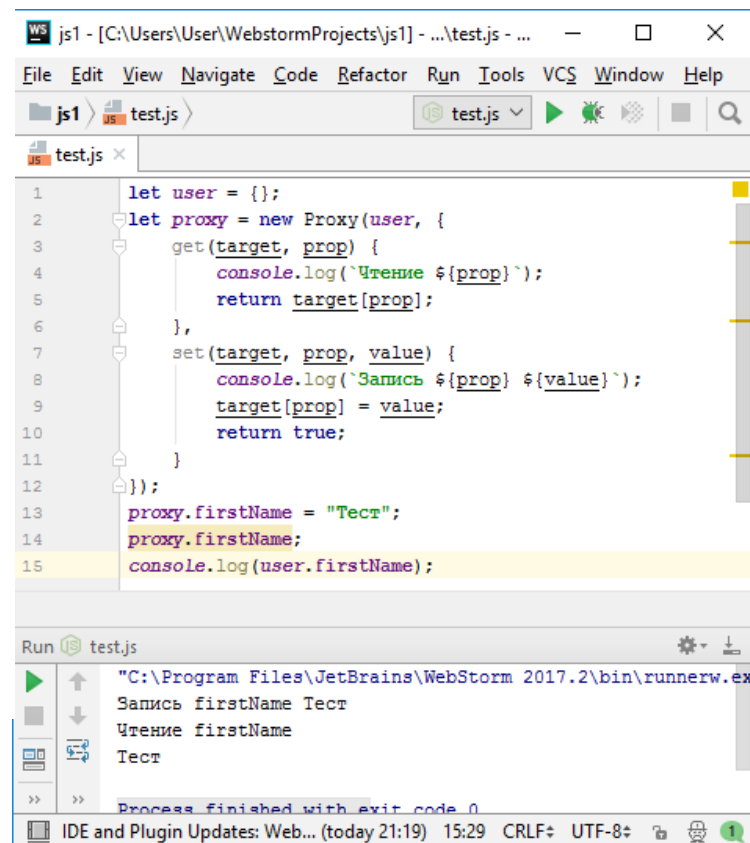
```
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\runnerw.exe"
{ value: 'a', done: false }
2
{ value: 'b', done: false }
3
{ value: 'c', done: true }
```

The status bar at the bottom indicates the IDE and Plugin Updates: Web... (today 21:19) 12:20 CRLF+ UTF-8+.

# Прокси (1)

82

```
let user = {};
let handler = {
 get(target, prop) {
 console.log(`Чтение ${prop}`);
 return target[prop];
 },
 set(target, prop, value) {
 console.log(`Запись ${prop} ${value}`);
 target[prop] = value;
 return true;
 }
}
let proxy = new Proxy(user, handler);
proxy.firstName = "Тест";
proxy.firstName;
console.log(user.firstName);
```



The screenshot shows an IDE window titled 'js1 - [C:\Users\User\WebstormProjects\js1] - ...\test.js - ...'. The code editor displays the following JavaScript code:

```
1 let user = {};
2 let proxy = new Proxy(user, {
3 get(target, prop) {
4 console.log(`Чтение ${prop}`);
5 return target[prop];
6 },
7 set(target, prop, value) {
8 console.log(`Запись ${prop} ${value}`);
9 target[prop] = value;
10 return true;
11 }
12 });
13 proxy.firstName = "Тест";
14 proxy.firstName;
15 console.log(user.firstName);
```

Below the code editor, the 'Run' tab shows the output of the program:

```
Run test.js
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\runnerw.exe"
Запись firstName Тест
Чтение firstName
Тест
Process finished with exit code 0
```

The status bar at the bottom indicates 'IDE and Plugin Updates: Web... (today 21:19) 15:29 CRLF+ UTF-8+ 1'.

# Прокси (2)

83

```
let handler = {
 get: function(target, name) {
 return name in target ? target[name] : Math.PI;
 }
};
let p = new Proxy({}, handler);
p.a = 1;
console.log(p.a, p.b); // 1 3.141592653589793
```

- Метод **Proxy.revocable()** создаёт отзываемый **Прокси**
- **Прокси** может быть отозван функцией **revoke**
  - отключает все ловушки-обработчики
- После этого любые операции над **Прокси** вызовут ошибку **TypeError**

# Map

84

```
let map = new Map();
```

```
map.set('1', 'строка');
```

```
map.set(1, 'число');
```

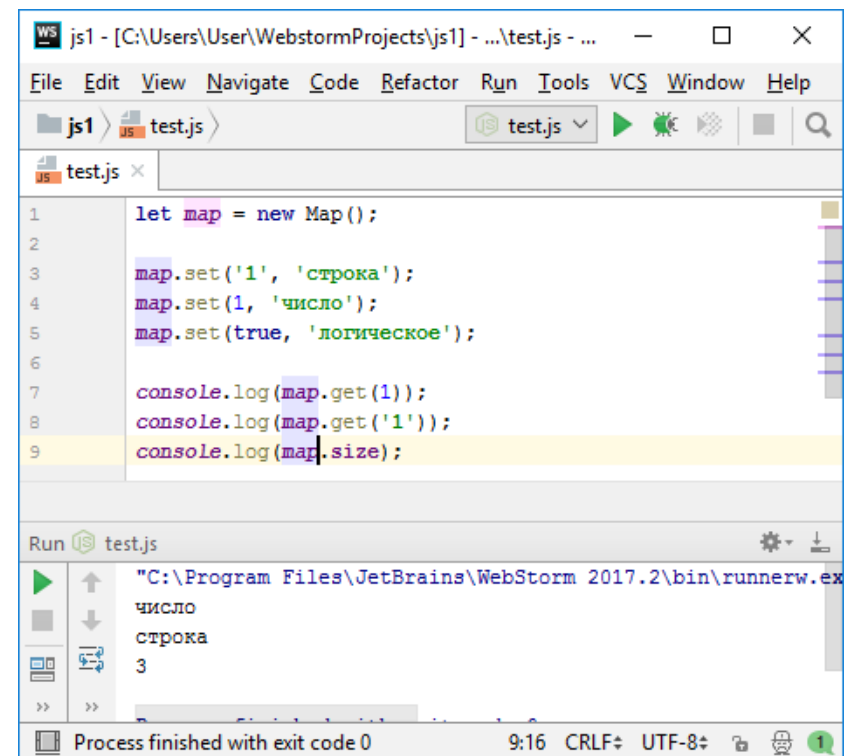
```
map.set(true, 'логическое');
```

```
console.log(map.get(1));
```

```
console.log(map.get('1'));
```

```
console.log(map.size);
```

## WeakMap



The screenshot shows an IDE window titled 'js1 - [C:\Users\User\WebstormProjects\js1] - ...test.js - ...'. The editor displays the following JavaScript code in 'test.js':

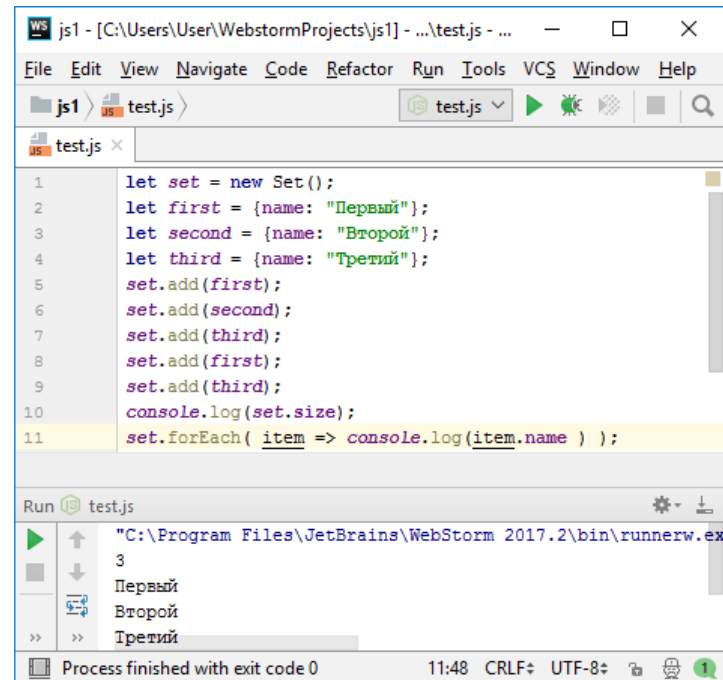
```
1 let map = new Map();
2
3 map.set('1', 'строка');
4 map.set(1, 'число');
5 map.set(true, 'логическое');
6
7 console.log(map.get(1));
8 console.log(map.get('1'));
9 console.log(map.size);
```

Below the editor, the 'Run' tab shows the execution output for 'test.js':

```
"C:\Program Files\JetBrains\WebStorm 2017.2\bin\runnerw.exe
число
строка
3
```

The status bar at the bottom indicates 'Process finished with exit code 0', the time '9:16', and the encoding 'CRLF UTF-8'.

```
let set = new Set();
let first = {name: "Первый"};
let second = {name: "Второй"};
let third = {name: "Третий"};
set.add(first);
set.add(second);
set.add(third);
set.add(first);
set.add(third);
console.log(set.size);
set.forEach(item => console.log(item.name));
```



The screenshot shows a code editor window titled 'js1 - [C:\Users\User\WebstormProjects\js1] - ...\test.js - ...'. The code in the editor is identical to the one in the first block. Below the editor, the 'Run' tab is active, showing the output of the program. The output consists of the number '3' followed by the names 'Первый', 'Второй', and 'Третий' on separate lines. The status bar at the bottom indicates 'Process finished with exit code 0' and the time '11:48'.

- `set.add(item)` – добавляет в коллекцию `item`
- `set.delete(item)` – удаляет `item` из коллекции
- `set.has(item)` – возвращает `true`, если `item` есть в коллекции
- `set.clear()` – очищает `set`

# Обещание / Promise (1)

- Синтаксис создания «Обещания»:

```
let promise = new Promise(function(resolve, reject) {
 // При вызове Promise выполняется данная функция
 // По завершении нужно вызвать одно из:
 // resolve(результат) - при успешном выполнении
 // reject(ошибка) - при ошибке
})
```

- Вызов «Обещания»:

```
promise.then(onFulfilled, onRejected)
```

- onFulfilled – функция, которая будет вызвана с результатом при resolve.
- onRejected – функция, которая будет вызвана с ошибкой при reject.

# Обещание / Promise (2)

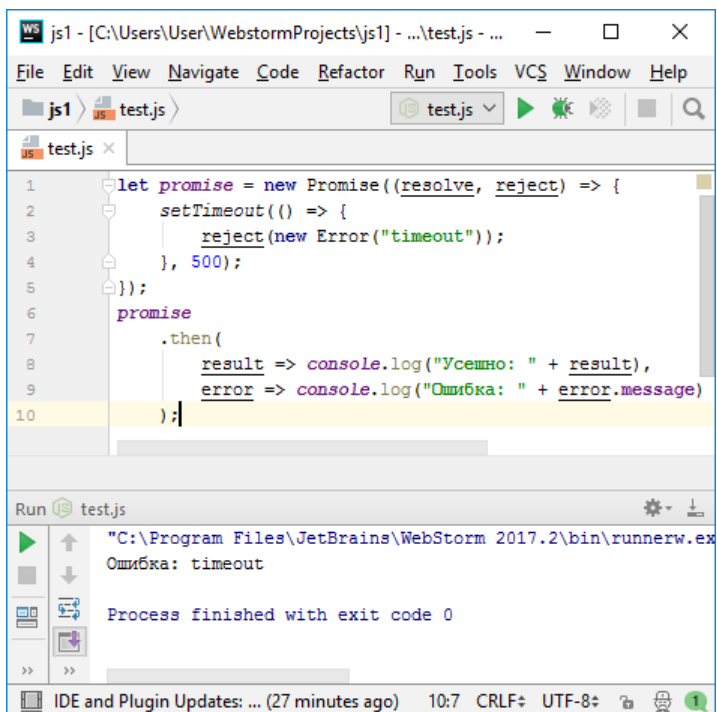
```
function myPromise() {
 return new Promise((resolve, reject) => {
 // Успех в половине случаев
 if (Math.random() > .5)
 resolve("Успех")
 else
 reject("Ошибка")
 })
}
myPromise().then(
 (success)=>console.log(success),
 (failure)=>console.log(failure)
)
```

# Обещание / Promise (2)

88

```
let promise = new Promise((resolve, reject) => {
 setTimeout(() => {
 reject(new Error("timeout"));
 }, 500);
});

promise
 .then(
 result => console.log("Успешно: " + result),
 error => console.log("Ошибка: " + error.message)
);
```



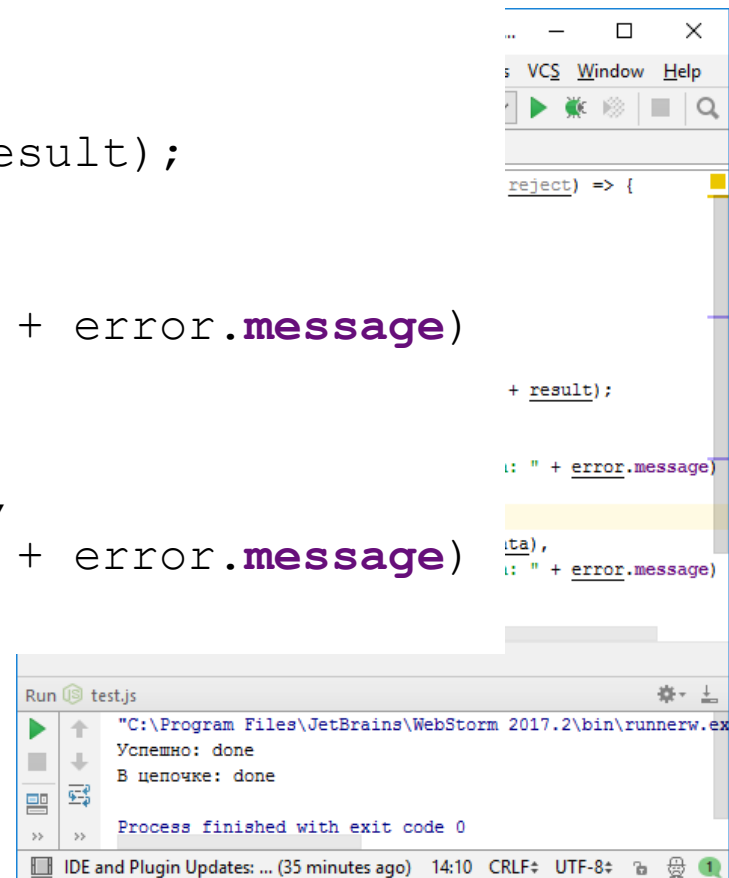


# Обещание / Promise (3)

89

```
let promise = new Promise((resolve, reject) => {
 setTimeout(() => {
 resolve("done");
 }, 500);
});

promise
 .then(
 result => {
 console.log("Успешно: " + result);
 return result;
 },
 error => console.log("Ошибка: " + error.message)
)
 .then(data =>
 console.log("В цепочке:", data),
 error => console.log("Ошибка: " + error.message)
);
```



# Обещание / Promise (4)

- А что после `catch`?
- Обработчик `.catch(onRejected)` получает ошибку и должен обработать её.
- Есть два варианта развития событий:
  - Если ошибка не критичная, то `onRejected` возвращает значение через `return`, и управление переходит в ближайший `.then(onFulfilled)`.
  - Если продолжить выполнение с такой ошибкой нельзя, то он делает `throw`, и тогда ошибка переходит в следующий ближайший `.catch(onRejected)`.

# Обещание / Promise (5)

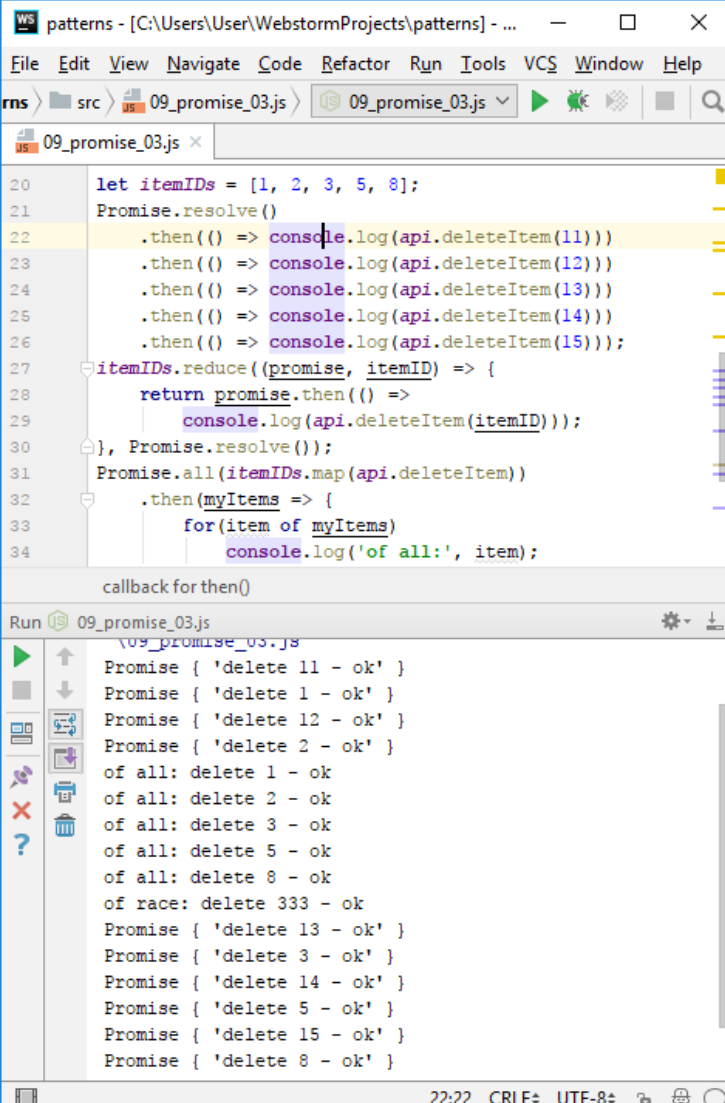
91

```
Promise.all([
 httpGet('/promise/data1.json'),
 httpGet('/promise/data2.json'),
 httpGet('/promise/data3.json')
]).then(results => {
 console.log(results);
});
```

# Обещание / Promise (6) – последовательное выполнение / all, race

92

```
let itemIDs = [1, 2, 3, 5, 8];
Promise.resolve()
 .then(() => console.log(api.deleteItem(11)))
 .then(() => console.log(api.deleteItem(12)))
 .then(() => console.log(api.deleteItem(13)))
 .then(() => console.log(api.deleteItem(14)))
 .then(() => console.log(api.deleteItem(15)));
itemIDs.reduce((promise, itemID) => {
 return promise.then(() =>
 console.log(api.deleteItem(itemID)));
}, Promise.resolve());
Promise.all(itemIDs.map(api.deleteItem))
 .then(myItems => {
 for(item of myItems)
 console.log('of all:', item);
 });
Promise.race([api.deleteItem(333),
 api.deleteItem(666)])
 .then(result => {
 console.log('of race:', result);
 });
```



The screenshot shows a code editor window titled 'patterns - [C:\Users\User\WebstormProjects\patterns] - ...'. The editor displays the same JavaScript code as the previous block. The console output window at the bottom shows the following logs:

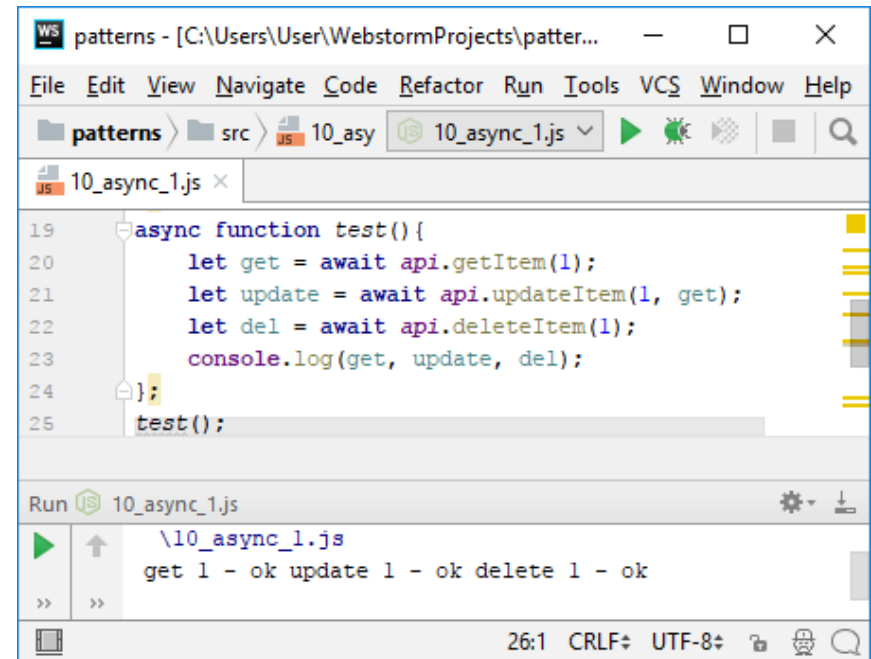
```
Promise { 'delete 11 - ok' }
Promise { 'delete 1 - ok' }
Promise { 'delete 12 - ok' }
Promise { 'delete 2 - ok' }
of all: delete 1 - ok
of all: delete 2 - ok
of all: delete 3 - ok
of all: delete 5 - ok
of all: delete 8 - ok
of race: delete 333 - ok
Promise { 'delete 13 - ok' }
Promise { 'delete 3 - ok' }
Promise { 'delete 14 - ok' }
Promise { 'delete 5 - ok' }
Promise { 'delete 15 - ok' }
Promise { 'delete 8 - ok' }
```

# async / await (1) – замена Обещания

93

```
async function test(){
 let get = await api.getItem(1);
 let update = await api.updateItem(1, get);
 let del = await api.deleteItem(1);
 console.log(get, update, del);
};
test();
```

```
Promise.resolve()
 .then(() => {
 return api.getItem(1)
 })
 .then(item => {
 return api.updateItem(1, item);
 })
 .then(() => {
 return api.deleteItem(1);
 })
 .catch(e => {
 console.log('error');
 })
```



# async / await (2) – sequence

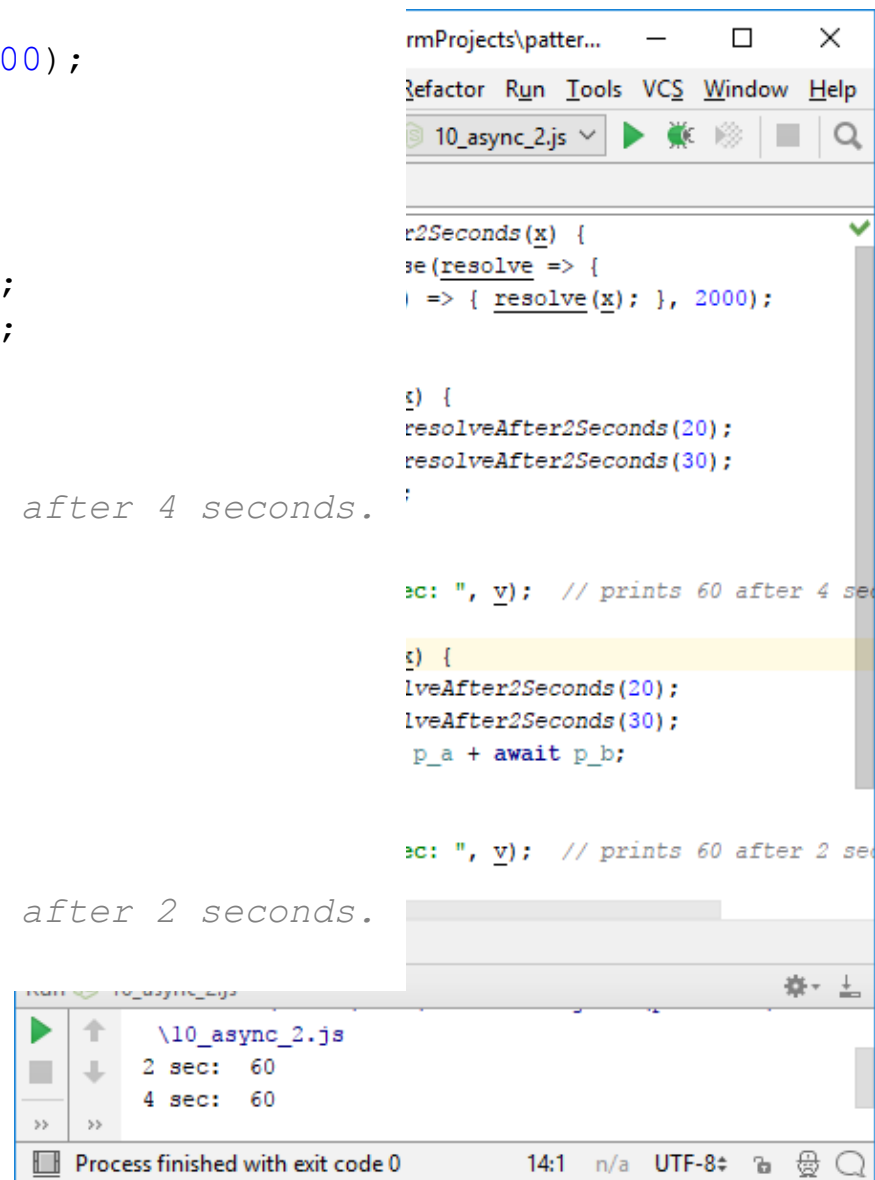
```
function resolveAfter2Seconds(x) {
 return new Promise(resolve => {
 setTimeout(() => { resolve(x); }, 2000);
 });
}
```

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

add1(10).then(v => {
 console.log("4 sec: ", v); // prints 60 after 4 seconds.
});
```

```
async function add2(x) {
 const p_a = resolveAfter2Seconds(20);
 const p_b = resolveAfter2Seconds(30);
 return x + await p_a + await p_b;
}

add2(10).then(v => {
 console.log("2 sec: ", v); // prints 60 after 2 seconds.
});
```



# Настройка контекста / call, apply, bind

```
function log(y) {
 if(this.x)
 console.log(this.x, y)
 else
 console.log("Unknown", y)
}
```

*// Указание контекста при вызове - **call** - параметры передаются через запятую*

```
log.call({x:"me"}, "1234") // me 1234
```

```
log.call(null, "1234") // Unknown 1234
```

*// Указание контекста при вызове - **apply** - параметры передаются как массив*

```
log.apply({x:"you"}, ["42"]) // you 42
```

```
log.apply(undefined, ["42"]) // Unknown 42
```

*// Настройка контекста - **bind***

```
let myLog1 = log.bind({x:"us"})
```

```
let myLog2 = log.bind(null)
```

```
myLog1("abc") // us abc
```

```
myLog2("abc") // Unknown abc
```

# Компактные конструкции

96

- nullish coalescing operator - ??
  - `let var2 = variable ?? "default value"`
  - Возвращает значение правого операнда в том случае, если значение левого равно **null** или **undefined**
- ИЛИ - ||
  - `let var2 = variable || "default value"`
  - Присвоение значения по умолчанию, если переменная «пустая»
- Двойное побитовое отрицание - ~~
  - Побитовый оператор НЕ (~) он берёт число, преобразует его в 32-битное целое число (отбрасывая «лишние» биты) и инвертирует биты этого числа
  - `~x → -(x+1)`
  - Быстрое преобразование  
`~~x → Math.floor(x)`

```
let x = Math.PI
```

```
console.log(Math.floor(x), ~x, ~~x) // 3 -4 3
```



# Приватные поля и методы / #

97

```
class Checker {
 #myLimit = 200;
 #checkValue(value) {
 if (value < 0) throw new Error("Слишком мало");
 if (value > this.#myLimit) throw new Error("Слишком много");
 }
}

let checker = new Checker();
// "снаружи" нет доступа к приватным полям и методам класса
checker.#checkValue(-1);
// SyntaxError: Private field must be declared in an enclosing class
checker.#myLimit = 1000;
// SyntaxError: Private field must be declared in an enclosing class
```

# Strict mode – «строгий» режим

98

- Устанавливается

- для файла в целом

- в начале файла нужно добавить строку

- 'use strict'
      - "use strict"

*// Синтаксис всего файла в строгом режиме*

```
'use strict';
let v = "Hi! I'm a strict mode script!";
```

- для отдельной функции

- в начале функции нужно добавить строку

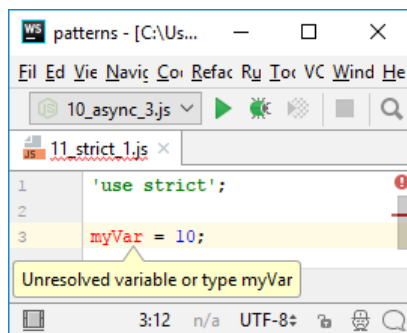
- 'use strict'
    - "use strict"

```
function strict() {
 // Строгий режим на уровне функции
 'use strict';
 function nested() {
 return 'And so am I!';
 }
 return "Hi! I'm a strict mode function! " + nested();
}
function notStrict() {
 return "I'm not strict.";
}
```

# «Строгий» режим (1)

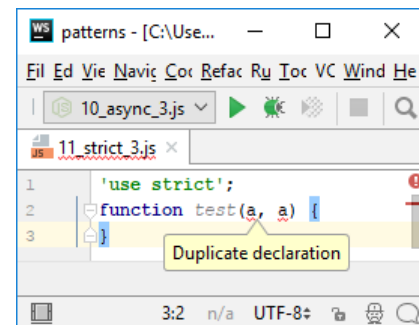
- Ошибка, если переменная не была объявлена

```
'use strict';
myVar = 10;
```



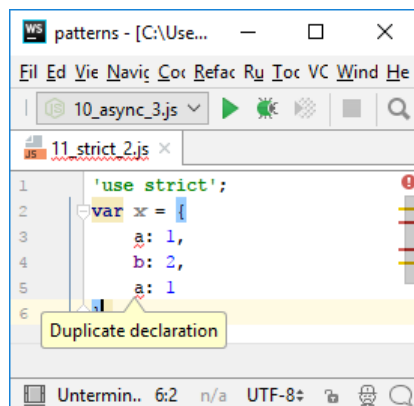
- Запрещает дублирование аргументов

```
'use strict';
function test(a, a) {
}
```



- Не допускает дублирование ключей в объекте

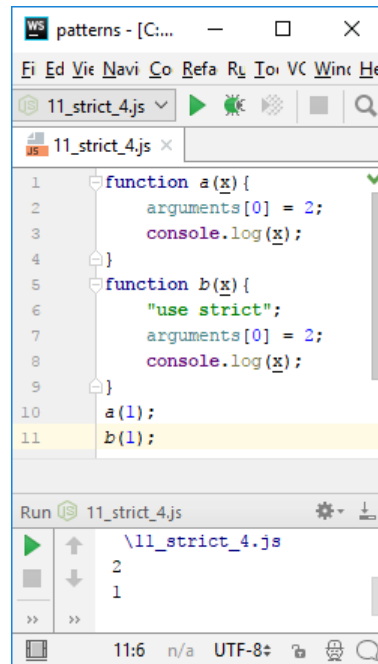
```
'use strict';
var x = {
 a: 1,
 b: 2,
 a: 1
}
```



# «Строгий» режим (2)

- Запрещает изменение **arguments**

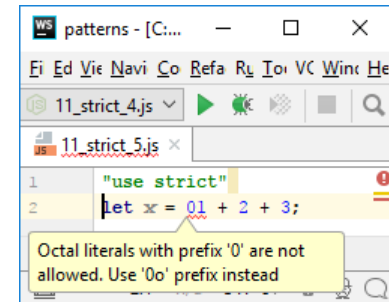
```
function a(x) {
 arguments[0] = 2;
 console.log(x);
}
function b(x) {
 "use strict";
 arguments[0] = 2;
 console.log(x);
}
a(1);
b(1);
```



- Контроль сложения в восьмеричном формате

"use strict"

```
let x = 01 + 2 + 3;
let x = 0o1 + 2 + 3;
```



- Не добавляются переменные после **eval**

```
var x = 17;
var evalX = eval("'use strict'; var x = 42; x;");
console.assert(x === 17);
console.assert(evalX === 42);
```

# Вопросы для самопроверки

101

- В чем отличие ECMAScript от JavaScript?
- Какие реализации JavaScript вы знаете?
- В чем отличия `var`, `let` и `const`?
- В чем отличия `==` и `===`?
- В чем отличия строк `""` и ````?
- Как можно сгенерировать и обработать исключительную ситуацию?
- В чем отличия `for..in` от `for..of`?
- В чем отличия `break` от `continue` и при чем тут метки?
- Как можно добавить объект в массив?
- Что такое замыкание?
- Какие варианты создания объекта Вы знаете?
- Что такое генератор? Прокси?
- В чем отличия `Map` от `WeakMap`?
- Зачем нужны `Promise`?
- Как использовать `async/await`?
- Зачем нужен строгий режим?