

# **МДК.01.01. Эксплуатация информационной системы**

## **Практическое занятие №8**

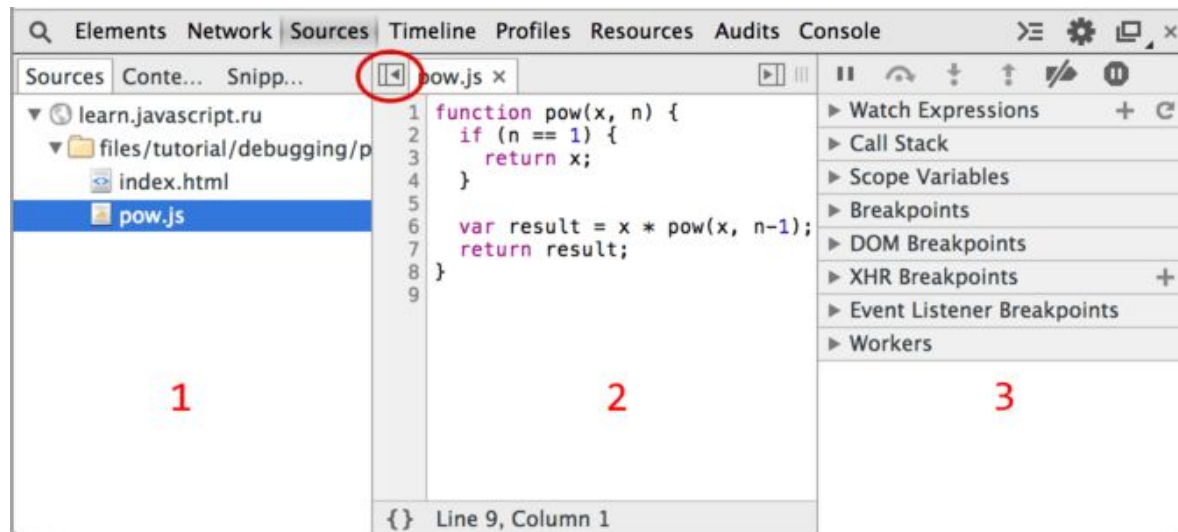
### **1) Отладка в браузере Chrome**

# Отладка в браузере Chrome (п.3.1)

<https://learn.javascript.ru>

/

Отладка – это процесс поиска и исправления ошибок в скрипте. Все современные браузеры и большинство других сред разработки поддерживают инструменты для отладки – специальный графический интерфейс, который сильно упрощает отладку. Он также позволяет по шагам отследить, что именно происходит в нашем коде



Включить инструменты разработчика в браузере Chrome можно, нажав Ctrl+Shift+I

Elements Network Sources Timeline Profiles Resources Audits Console

pow.js x

```
1 function pow(x, n) {
2   if (n == 1) {
3     return x;
4   }
5
6   var result = x * pow(x, n-1);
7   return result;
8 }
9
```

Watch Expressions + C

Call Stack

Scope Variables

Breakpoints

- pow.js:6  
var result = x \* pow(x, n-1);

DOM Breakpoints

XHR Breakpoints +

Event Listener Breakpoints

Workers

{ } Line 9, Column 1

Elements Network Sources Timeline Profiles Resources Audits Console

pow.js x

```
1 function pow(x, n) {
2   if (n == 1) {
3     return x;
4   }
5
6   var result = x * pow(x, n-1);
7   return result;
8 }
9
```

Watch Expressions + C

Call Stack

pow pow.js:6

(anonymous function) index.html:13

Paused on a JavaScript breakpoint.

Scope Variables

Local

- n: 3
- result: undefined
- this: Window
- x: 5

Global Window

Breakpoints

- pow.js:6  
var result = x \* pow(x, n-1);

{ } Line 6, Column 1

Пример кода для отладки в браузере Chrome:

**файл script.js**

```
var k = 4;
for (var i = 0; i < 3; i++) {
  alert( "Значение параметра 'c' функции calc(a, b, c) " + "= "
+ i);
  var test = calc(k, 2, i);
  alert("Результат выполнения функции " + test);
}
function calc(a, b, c) {
  return b*b - a*c;
}
```

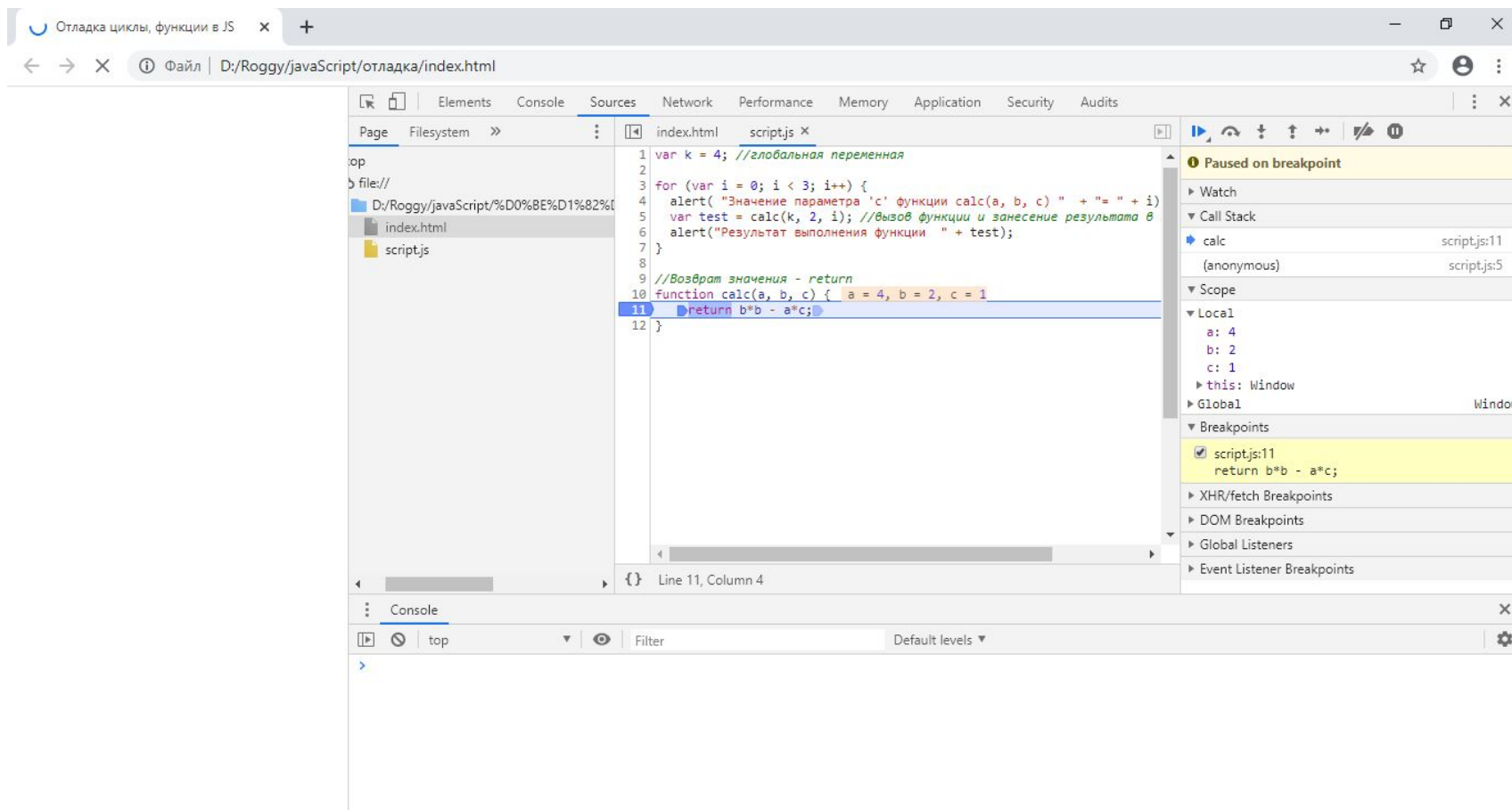
## **Задание:**

1. Изучить возможности отладчика кода в браузере Chrome.
2. Провести отладку кода (можно своего блока кода) с помощью браузера Chrome:
  - *Создать точку останова* (breakpoints) и в остановленном коде определить текущие значения переменных.
  - Выполнить код – по одной строке (по шагам) и посмотреть текущие значения переменных на каждом шаге.

**Scope** показывает текущие переменные.

В **Local** отображаются локальные переменные функций, а их значения подсвечены в исходном коде.

В **Global** перечисляются глобальные переменные (т.е. объявленные за пределами функций).





– сделать шаг. Быстрая клавиша – F11.

Здесь мы «заходим» во вложенные функции и шаг за шагом проходим по скрипту.

The screenshot shows a web browser window with the developer tools open. The browser's address bar shows the file path: `D:/Roggy/javascript/отладка/index.html`. The developer tools are in 'Sources' mode, showing the file `scriptjs` with the following code:

```
1 var k = 4; //глобальная переменная
2
3 for (var i = 0; i < 3; i++) {
4   alert("Значение параметра 'с' функции calc(a, b, c) " + " = " + i);
5   var test = calc(k, 2, i); //вызов функции и занесение результата в
6   alert("Результат выполнения функции " + test);
7 }
8
9 //Возврат значения - return
10 function calc(a, b, c) {
11   return b*b - a*c;
12 }
```

The debugger is paused at line 11, `return b*b - a*c;`. The right-hand pane shows the 'Debugger paused' status and the current scope, which is the function `calc`. The console at the bottom is empty.