

Курс «Программирование на JavaScript»
Программное обеспечение. Работа в командной строке.
Система GIT

Содержание

1. Структура курса.....	3
2. Рекомендуемая экосистема	5
3. Командная строка.....	5
4. Установка JavaScript.....	10
5. Система управления версиями (Git).....	10
6. Jupyter notebook	13
7. IDE Brackets	16
8. Домашнее задание (разделы 1 и 2)	20

1. Структура курса

Материалы курса находятся в репозитории¹ **Gitlab**.

Ссылка: <https://gitlab.com/thisroot/js-course-may-2017/> (перейдите по ссылке)

Первая часть курса – основы JavaScript (перейдите в папку JavaScript-ОСНОВЫ) .

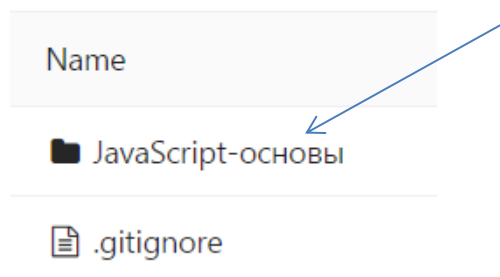


Рис. 1.1. Первая часть курса – основы JavaScript

Блок основы JavaScript состоит из разделов (пока доступ есть не ко всем разделам – доступ к разделам будет даваться последовательно мере прохождения курса)

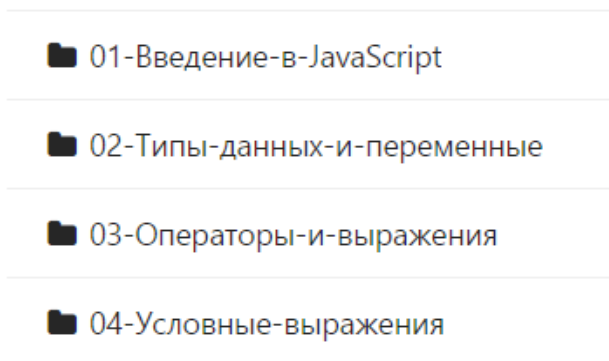


Рис. 1.2. Разделы курса

¹ Репозиторий, хранилище — место, где хранятся и поддерживаются какие-либо данные. Чаще всего данные в репозитории хранятся в виде файлов, доступных для дальнейшего распространения по сети [см. <https://ru.wikipedia.org/wiki/Репозиторий>].

Структура разделов (на примере [02-Типы-данных-и-переменные](#)):

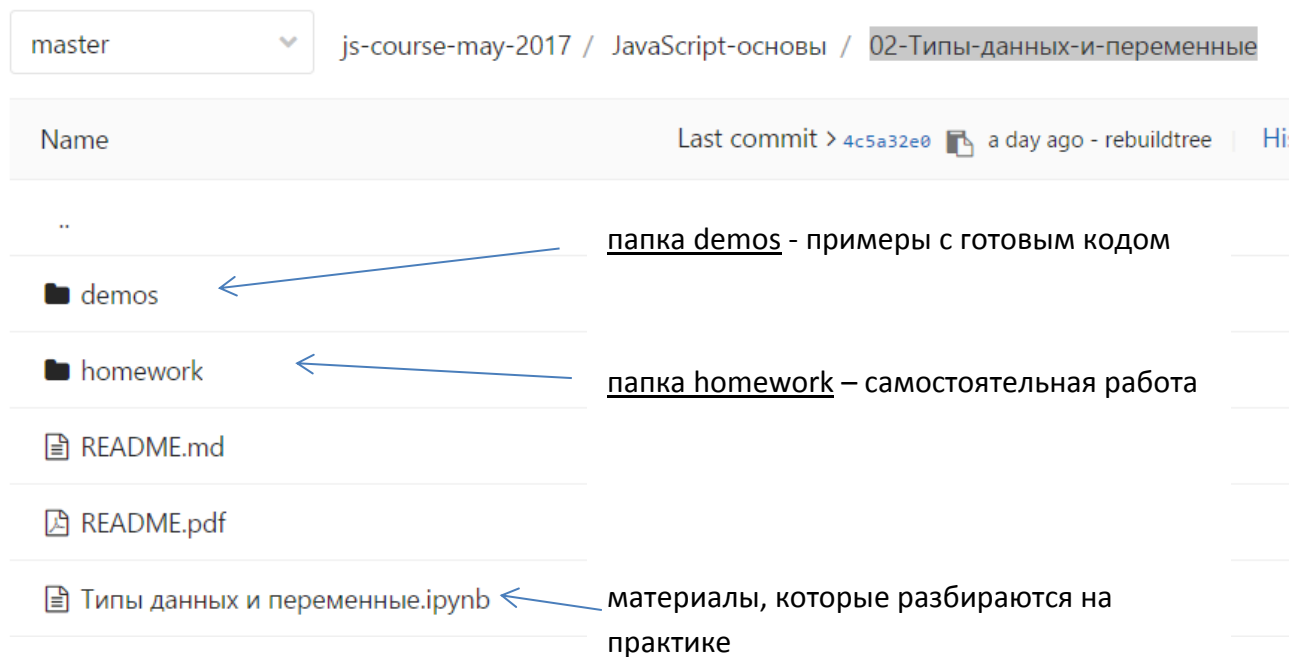


Рис. 1.3. Структура курса

Полное содержание курса:

Часть 1: Основы языка JavaScript

- Введение в JavaScript
- Типы данных и переменные
- Операторы и выражения
- Условные выражения
- Циклы
- Массивы
- Функции
- Объекты
- Методы в массивах и объектах
- Строки
- Регулярные выражения

Часть 2: JavaScript пользовательский интерфейс и объектная модель документа

- DOM Объектная модель документа
- DOM Операции с деревом объектов
- DOM Событийная модель
- jQuery Обзор
- jQuery Плагины
- JQuery Популярные библиотеки
- HTML Шаблоны
- DOM Эффективность

Рис. 1.4. Содержание курса

2. Рекомендуемая экосистема

Для изучения материалов курса **установите следующее программное обеспечение:**

- [Crome browser](#) - популярный браузер с продвинутыми средствами разработчика
- [Node.js](#) - транслятор JavaScript ²
- [IDE Brackets](#) – IDE (интегрированная среда разработки) для разработки HTML/CSS/JS приложений с удобной функцией **Live preview**
- [Jupyter notebook](#) - браузерный редактор кода (входит в дистрибутив Anaconda)
- [JavaScript](#) - ядро JavaScript для Jupyter notebook (**установку см. ниже на стр. 4**)
- [GIT](#) - Система управления версиями
- [MobaXterm](#) - работа в командной строке (вместо стандартной командной строки, устанавливайте по желанию)

3. Командная строка

Выполнять команды (создавать папки, просматривать их содержимое, запускать приложения и т.д.) на компьютере можно через привычный графический интерфейс операционной системы (большинство работают в Windows) или выполнять те же команды с помощью командной строки (консоли)

² <https://ru.wikipedia.org/wiki/Node.js>

Преимущество командной строки состоит в том, что она позволяет выполнять команды без участия графического интерфейса. Скорость выполнения команд гораздо выше, чем в графическом интерфейсе.

Часто специализированные программы, которые используют программисты, доступны исключительно через взаимодействие посредством интерфейса командной строки (CLI).

Запуск командной строки

Нажать комбинацию клавиш `win + R` и выполнить команду `cmd` (клавишу `win` находится слева от `Alt`) или Пуск - Выполнить – `cmd`.

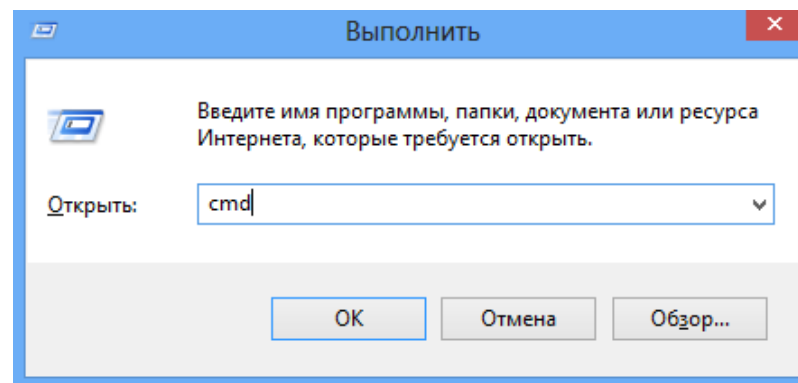


Рис. 3.1

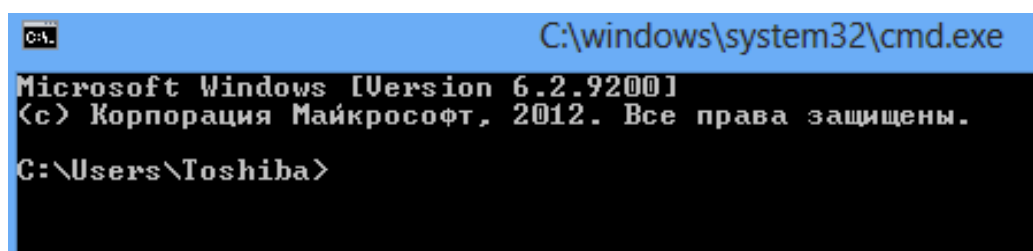


Рис. 3.2

Синтаксис командной строки

`dir` - просмотр содержимого текущего каталога (папки)

`cd` – переход в каталог.

`cd C://` - переход в корень диска C:

`cd ../Documents/Course` - переход по относительному пути - от текущего положения

`mkdir js` - создать каталог (где `js` – имя папки)

Практическая работа с командой строкой

1. Создадим папку `js-course-ivanov` на диске `C:` (где `ivanov` – Ваша фамилия)

- Запустим командную строку (комбинация клавиш `win + R`, выполнить команду `cmd`)

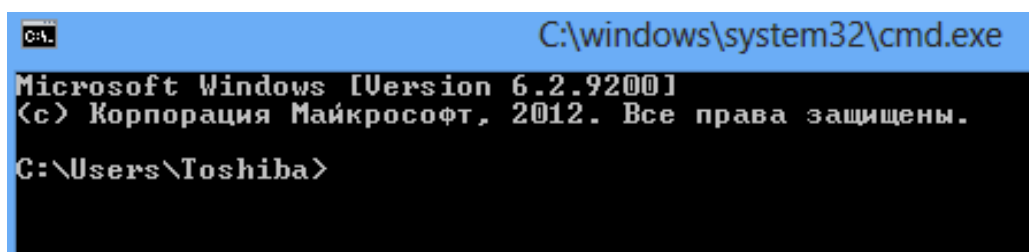


Рис. 3.3

Сейчас мы находимся в папке `Toshiba` (у Вас другая папка)

- Перейдем в корень диска `C:`: выполнив команду `cd C://`

```
C:\Users\Toshiba>cd c://
```

Рис. 3.4

Теперь мы находимся в корне диска `C:`:

```
c:\>
```

Рис. 3.5

Содержимое диска `C:` можно посмотреть, набрав команду `dir`

```
c:\>dir
Содержимое папки c:\
02.08.2014  11:47    <DIR>          DSSSL
11.06.2015  22:39    <DIR>          EzImplant-CDViewer
30.05.2014  18:52    <DIR>          iee.unn.ru
22.11.2013  22:03    <DIR>          Intel
15.02.2013  21:38    <DIR>          KAU
07.04.2013  20:39         254 966 135 kidsmile.mp4
06.01.2014  12:23    <DIR>          lj1010 series
26.07.2012  10:33    <DIR>          PerfLogs
13.05.2017  11:09    <DIR>          Program Files
12.05.2017  23:41    <DIR>          Program Files (x86)
```

Рис. 3.6

- Создадим папку js-course-ivanov (где ivanov – Ваша фамилия), выполнив команду mkdir js-course-ivanov

```
c:\>mkdir js-course-ivanov
c:\>
```

Рис. 3.7

Мы создали папку js-course-ivanov, но не перешли в неё. Поэтому конечной является строка c:\>

Чтобы войти в папку нужно выполнить команду cd js-course-ivanov

```
c:\>cd js-course-ivanov
```

Рис. 3.8

Очень удобно работать в командной строке, используя клавишу Tab, которая реализует автозаполнение командной строки.

Например, находясь в командной строке можно набрать **первые буквы** папки js-course-ivanov:


```
c:\>cd js-
```

Рис. 3.9

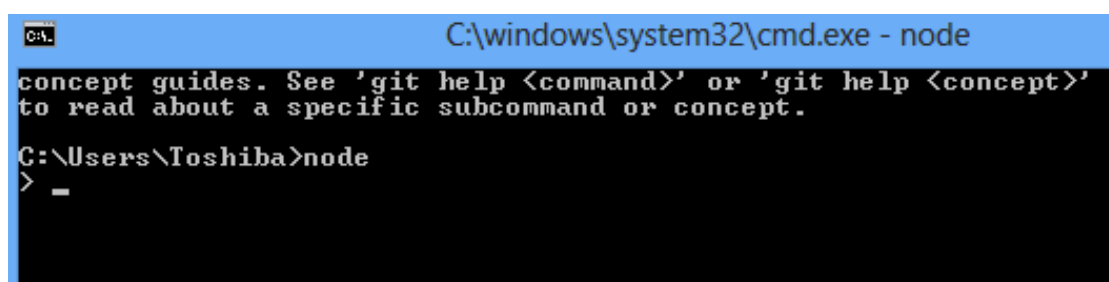
Далее нажимаем клавишу **Tab**, и название созданной ранее папки заполнится автоматически:

```
c:\>cd js-course-ivanov,
```

Рис. 3.10

Работа с javascript в командной строке

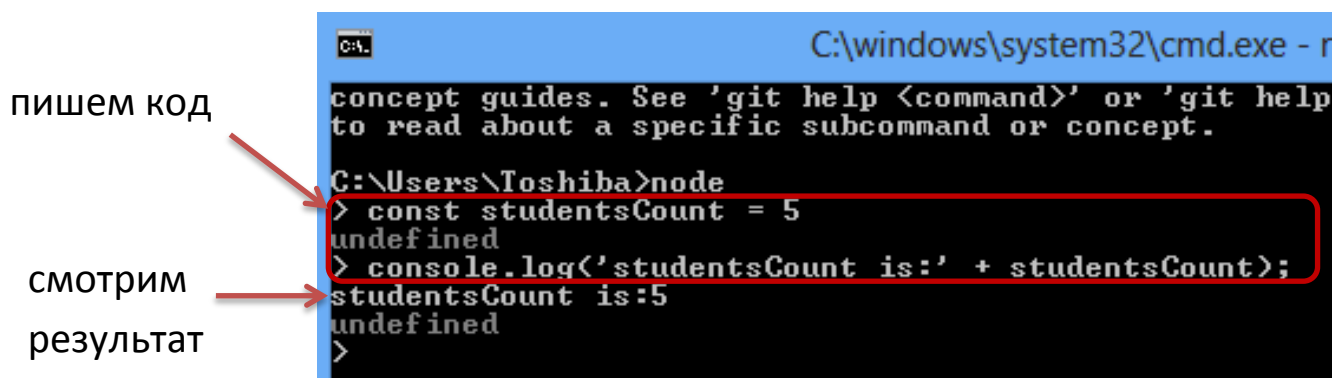
Находясь в командной строке, вводим команду **node**



```
C:\windows\system32\cmd.exe - node
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
C:\Users\Toshiba>node
> _
```

Рис. 3.11

и прописываем код



```
C:\windows\system32\cmd.exe - r
concept guides. See 'git help <command>' or 'git help
to read about a specific subcommand or concept.
C:\Users\Toshiba>node
> const studentsCount = 5
undefined
> console.log('studentsCount is:' + studentsCount);
studentsCount is:5
undefined
>
```

пишем код

смотрим результат

Рис. 3.12

4. Установка IJavaScript

IJavaScript - ядро JavaScript для Jupyter notebook³

Для установки IJavaScript в командной строке набрать `npm install -g ijavascript`, нажать Enter

```
c:\>npm install -g ijavascript
```

Рис. 4.1

Должна пройти установка ijavascript

```
c:\>npm install -g ijavascript
C:\Users\Toshiba\AppData\Roaming\npm\ijs -> C:\Users\Toshiba\AppData\Roaming\npm\node_modules\ijavascript\bin\ijavascript.js
C:\Users\Toshiba\AppData\Roaming\npm\ijsinstall -> C:\Users\Toshiba\AppData\Roaming\npm\node_modules\ijavascript\bin\ijsinstall.js
C:\Users\Toshiba\AppData\Roaming\npm\ijsconsole -> C:\Users\Toshiba\AppData\Roaming\npm\node_modules\ijavascript\bin\ijsconsole.js
C:\Users\Toshiba\AppData\Roaming\npm\ijskernel -> C:\Users\Toshiba\AppData\Roaming\npm\node_modules\ijavascript\lib\kernel.js
C:\Users\Toshiba\AppData\Roaming\npm\ijsnotebook -> C:\Users\Toshiba\AppData\Roaming\npm\node_modules\ijavascript\bin\ijsnotebook.js
C:\Users\Toshiba\AppData\Roaming\npm
-- ijavascript@5.0.19
```

Рис. 4.2

Далее нужно активировать ijavascript командой `ijsinstall`

```
c:\>ijsinstall
```

Рис. 4.3

5. Система управления версиями (Git)

Система управления версиями (от англ. Version Control System, VCS или Revision Control System) — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости

³ Jupyter - браузерный редактор кода, который мы будем использовать для интерактивной работы с кодом javascript непосредственно в браузере

возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое. (В нашем курсе, мы будем использовать систему Git).

Для хранения данных в системах управления версиями создаются репозитории.

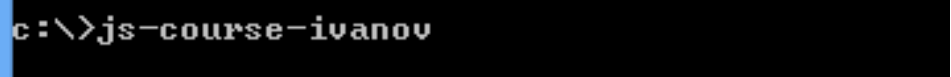
Напомним ссылку на репозиторий нашего курса:

<https://gitlab.com/thisroot/js-course-may-2017/> - там будут храниться все материалы курса.

Вы можете использовать данную ссылку для изучения материалов курса, а также создать локальный репозиторий (копию) на своем компьютере, выполнив команду `git clone`.

Создадим локальный репозиторий в созданной ранее папке `js-course-ivanov` (где `ivanov` – Ваша фамилия).

- войдем в каталог `js-course-ivanov` с помощью команды `cd`.
- находясь в папке `js-course-ivanov`



```
c:\>js-course-ivanov
```

Рис. 5.1

выполнить команду `git clone` - *адрес репозитория*.



```
c:\js-course-ivanov>git clone https://gitlab.com/thisroot/js-course-may-2017
```

Рис. 5.2

Произойдет клонирование репозитория

```
Cloning into 'js-course-may-2017'...  
warning: redirecting to https://gitlab.com/thisroot/js-course-may-2017.git/  
remote: Counting objects: 169, done.  
remote: Compressing objects: 100% (80/80), done.  
remote: Total 169 (delta 4), reused 169 (delta 4)  
Receiving objects: 100% (169/169), 627.68 KiB | 593.00 KiB/s, done.  
Resolving deltas: 100% (4/4), done.
```

Рис. 5.3

Посмотрим содержимое клонированного репозитория с помощью команды `dir`.

```
c:\js-course-ivanov\js-course-may-2017>dir
```

Рис. 5.4

```
Содержимое папки C:\jsgit\js-course-may-2017  
21.05.2017  01:10    <DIR>          .  
21.05.2017  01:10    <DIR>          ..  
21.05.2017  01:10                22 .gitignore  
21.05.2017  01:10    <DIR>          JavaScript-основы  
21.05.2017  01:10                1 158 LICENSE  
21.05.2017  01:10                5 882 README.md  
21.05.2017  01:10               46 135 README.pdf  
                4 файлов                53 197 байт  
                3 папок                1 615 286 272 байт свободно
```

Рис. 5.5

Напоминаем, для того, чтобы войти в папку нужно использовать команду `cd`

```
c:\js-course-ivanov>cd js-course-may-2017
```

Рис. 5.6

6. Jupyter notebook

Jupyter - браузерный редактор кода, который входит в дистрибутив Anaconda. Приложение нам понадобится для интерактивной работы с кодом javascript.

Запустим Jupyter, находясь в папке репозитория, выполнив команды `jupyter notebook`.

```
c:\js-course-ivanov\js-course-may-2017>jupyter notebook
```

Рис. 6.1

Jupyter должен открыться в браузере.

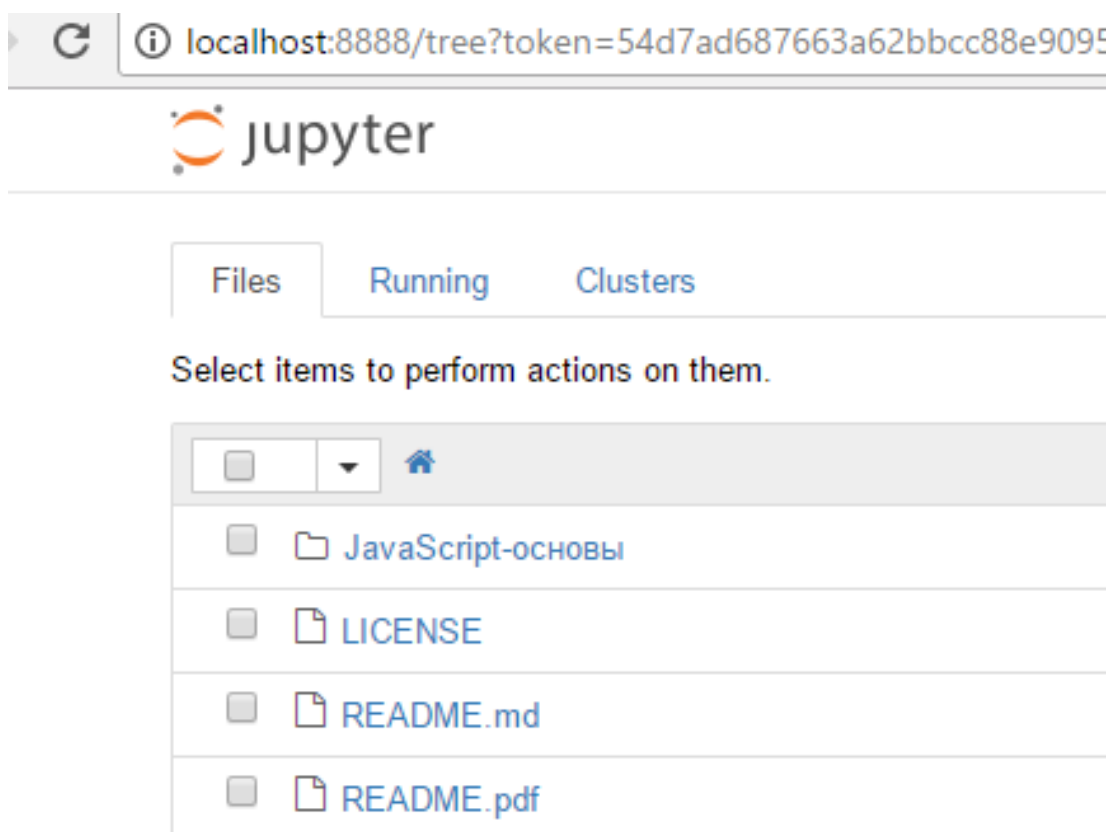


Рис. 6.2

Теперь Вы можете работать в локальном репозитории на своем компьютере в приложении Jupyter notebook..

Заметим, что физически репозиторий храниться по траектории:

 ▶ Компьютер ▶ T130981000A (C:) ▶ js-course-ivanov ▶ js-course-may-2017

Находясь в браузере, щелкнем по папке JavaScript-основы, далее 01-Введение-в-JavaScript и затем Введение в JavaScript.ipynb. Здесь нам доступны материалы раздела Введение в JavaScript.

Перейдем в раздел 02-Типы-данных-и-переменные – Типы данных и переменные.ipynb

Найдем поле с кодом и щелкнем в него мышкой.

```
In [ ]: var a = 1;
        var b = 2;

        var greaterAB = (a > b);
        console.log(`1 > 2 ? ${greaterAB}`); // false

        var equalA1 = (a === 1);
        console.log(`1 == 1 ? ${equalA1}`); // true»
        console.log(true, false);|
```

Рис. 6.2

Для исполнения кода в этом примере нажмем Shift + Enter.

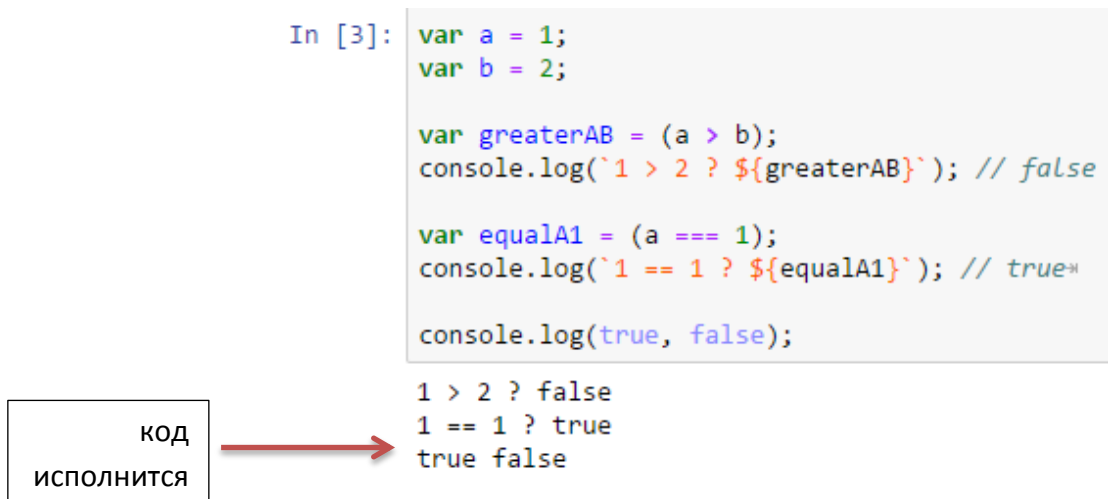
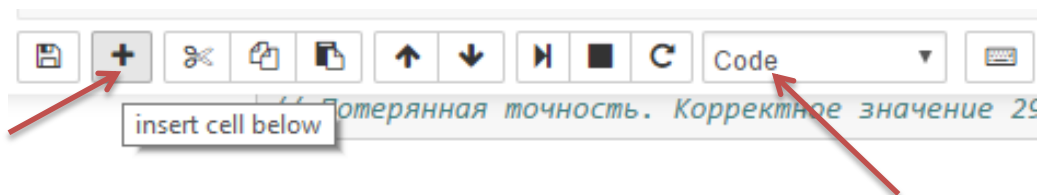


Рис. 6.3

В этом основное удобство работы в jupyter notebook – исполнять код (в данном случае готовых примеров) в браузере.

Вы можете добавить чистое поле и исполнять свои код.



Числа с плавающей точкой

- Является диапазонами всех действительных чисел
- Могут быть заданы с точностью до определенного значения
- Могут вести себя ненормально при вычислениях

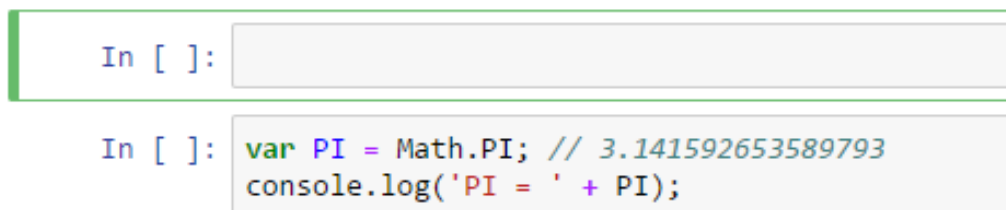


Рис. 6.4

Чтобы добавить код в соответствующем поле должно быть установлено значение Code.

Code ▼

Вы можете добавлять и другие типы полей:

Markdown ▼
Code
Markdown
Raw NBConvert
Heading

Markdown – текстовое поле

Heading – заголовок

7. IDE Brackets

Откройте js-файл из локального репозитория в Brackets (например из папки demos).

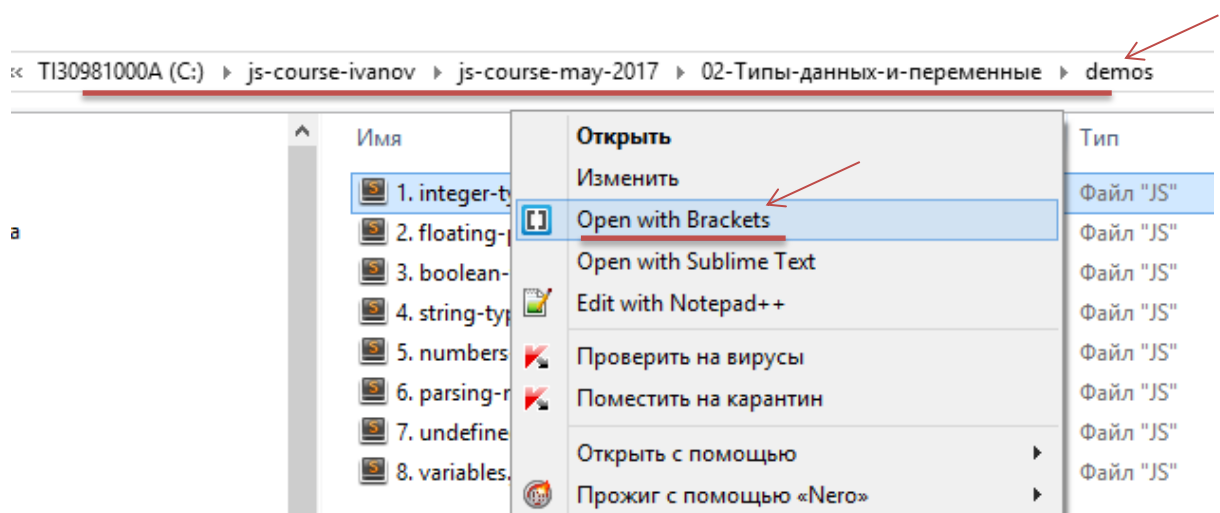


Рис. 7.1

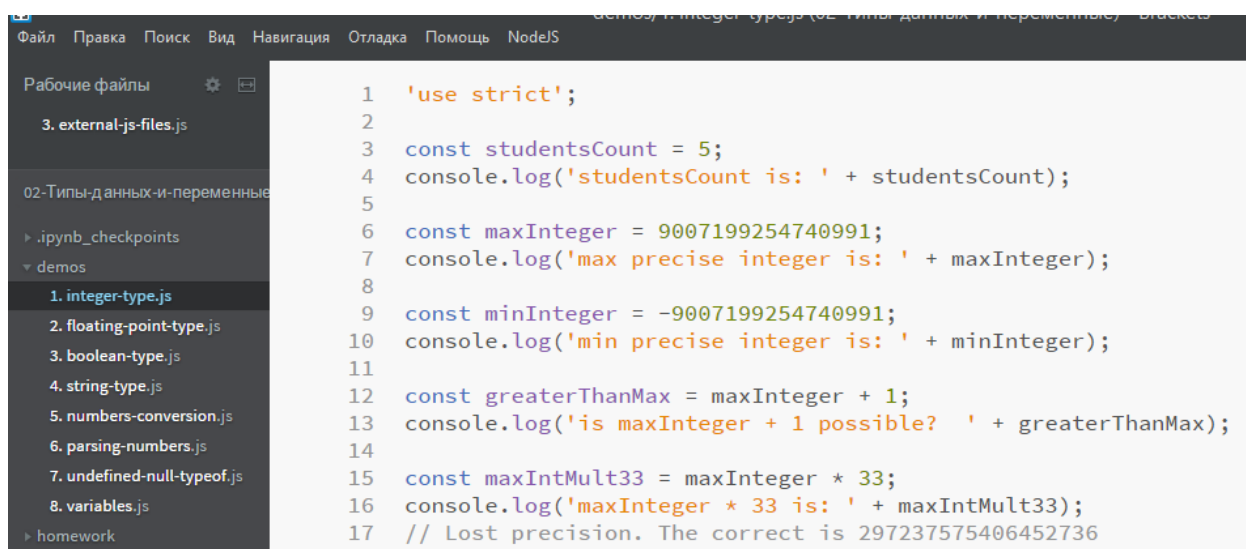


Рис. 7.2

Для работы с кодом установим расширение NodeJS integration.

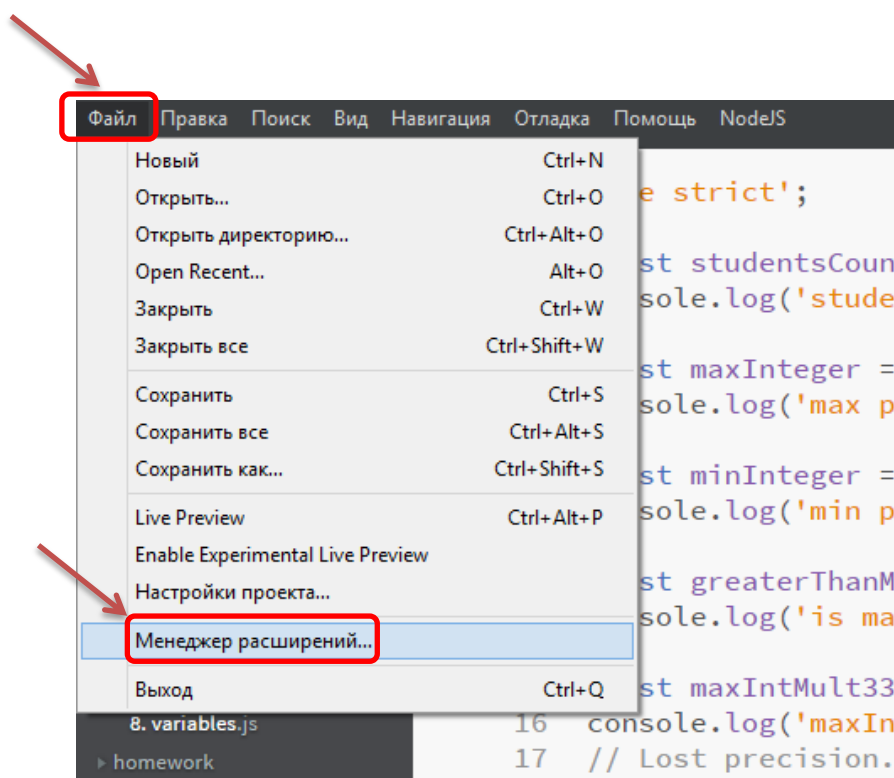


Рис. 7.3

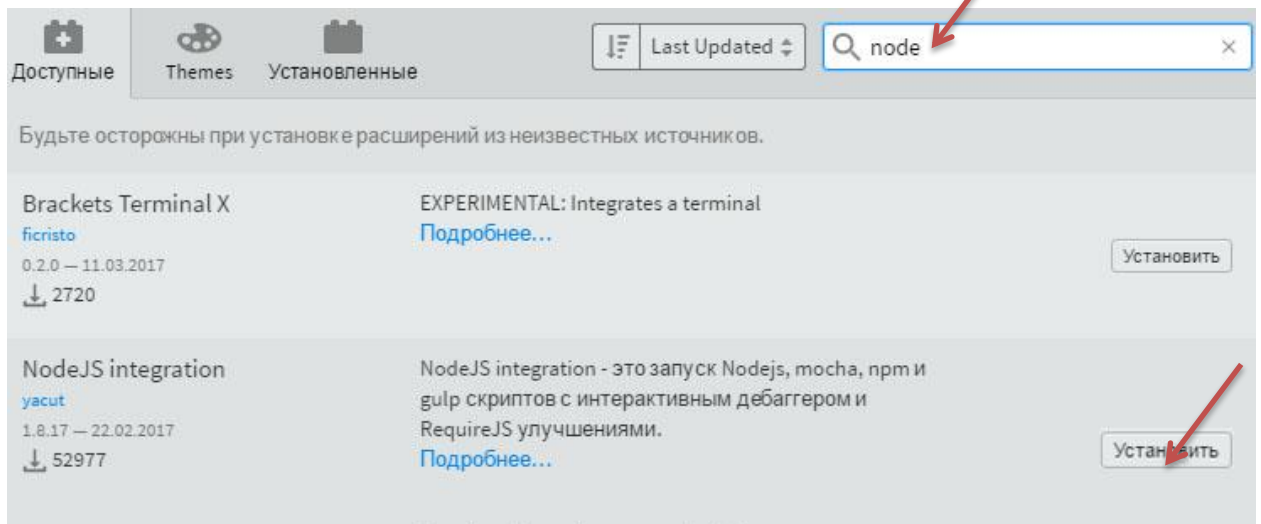


Рис. 7.4

В появившейся вкладке

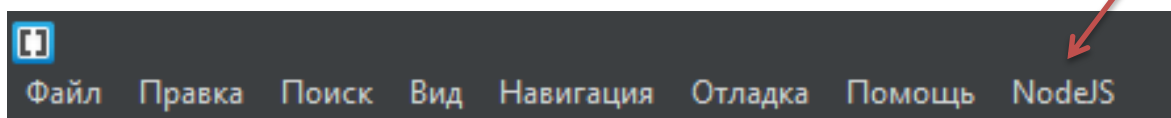


Рис. 7.5

Запускаем исполнение кода (или клавиши Ctrl + Shift + N)

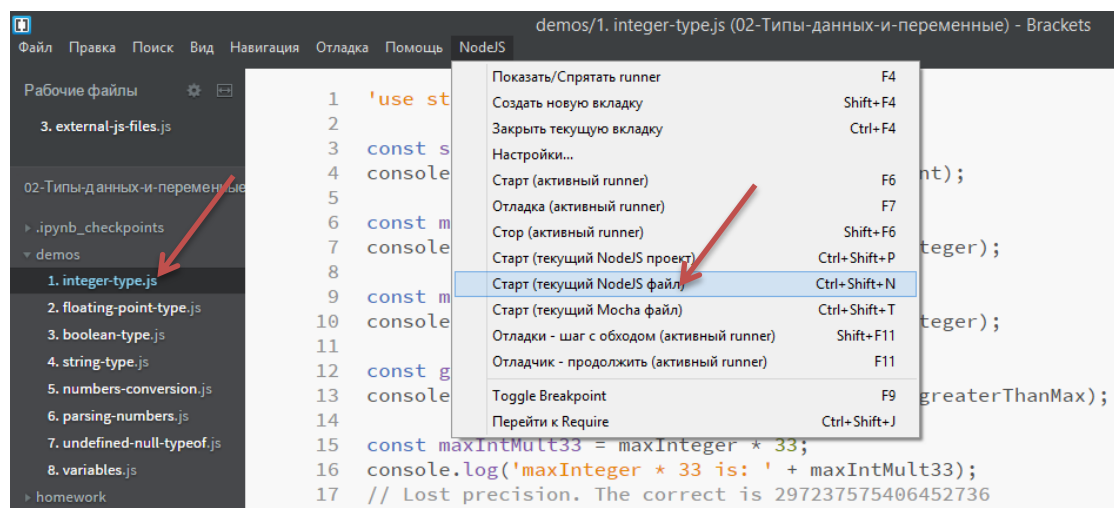
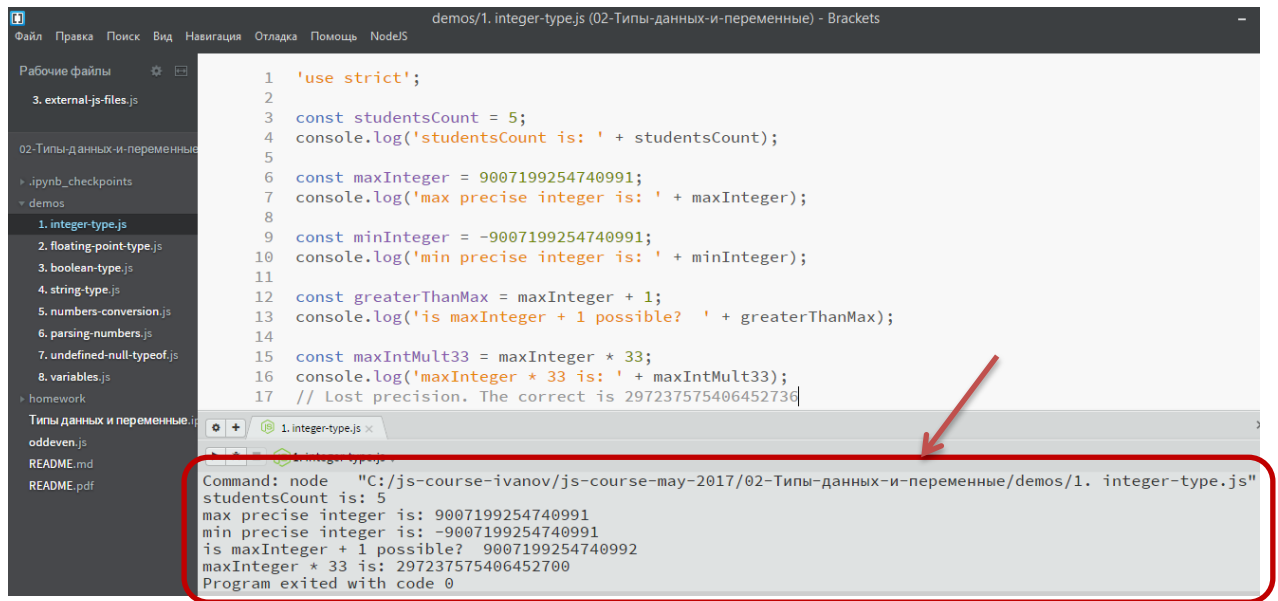


Рис. 7.6

Смотрим результат



The screenshot shows the Brackets IDE interface. The main editor displays a JavaScript file named `1. integer-type.js` with the following code:

```
1 'use strict';
2
3 const studentsCount = 5;
4 console.log('studentsCount is: ' + studentsCount);
5
6 const maxInteger = 9007199254740991;
7 console.log('max precise integer is: ' + maxInteger);
8
9 const minInteger = -9007199254740991;
10 console.log('min precise integer is: ' + minInteger);
11
12 const greaterThanMax = maxInteger + 1;
13 console.log('is maxInteger + 1 possible? ' + greaterThanMax);
14
15 const maxIntMult33 = maxInteger * 33;
16 console.log('maxInteger * 33 is: ' + maxIntMult33);
17 // Lost precision. The correct is 297237575406452736
```

The console output at the bottom shows the results of the code execution:

```
Command: node "C:/js-course-ivanov/js-course-may-2017/02-Типы-данных-и-переменные/demos/1. integer-type.js"
studentsCount is: 5
max precise integer is: 9007199254740991
min precise integer is: -9007199254740991
is maxInteger + 1 possible? 9007199254740992
maxInteger * 33 is: 297237575406452700
Program exited with code 0
```

A red arrow points from the console output to the `console.log` statement on line 16 of the code.

Рис. 7.7

Здесь мы просто посмотрели как выполняется код в программе Brackets.

7. Выводы

Работать с js-КОДОМ в данном курсе мы будем в:

- Jupyter notebook (пример на стр. 13). В
- Brackets (пример на стр. 16)
- Командной строке (пример на стр. 5)

В данном пособии мы исполняли js-код этими способами. Проверьте, все ли у Вас получилось

8. Домашнее задание (разделы 1 и 2)

8.1. Повторите материалы курса

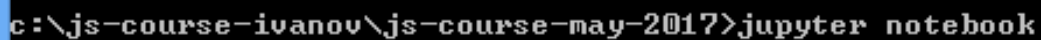
[01-Введение-в-JavaScript](#)

[02-Типы-данных-и-переменные](#)

8.2. Локальный репозиторий и Jupyter notebook

Скачать репозиторий <https://gitlab.com/thisroot/js-course-may-2017/> к себе на компьютер (если этого не сделали ранее). Как это сделать описано на стр. 11-12 (рис. 5.1-5.6).

Запустить локальный репозиторий в **Jupyter notebook**



```
c:\js-course-ivanov\js-course-may-2017>jupyter notebook
```

Рис. 8.1

Открываем папку JavaScript-основы



 JavaScript-основы	rebuildtree
 .gitignore	first

Рис. 8.2

далее 02-Типы-данных-и-переменные - Типы данных и переменные.ipynb

01-Введение-в-JavaScript

02-Типы-данных-и-переменные

03-Операторы-и-выражения

04-Условные-выражения

Рис. 8.3

demos

homework

Типы данных и переменные.ipynb

README.md

README.pdf

Рис. 8.4

Исполните код во всех поля Code.

```
In [1]: 'use strict';  
var studentsCount = 5;  
console.log('studentsCount is: ' + studentsCount);  
  
studentsCount is: 5
```

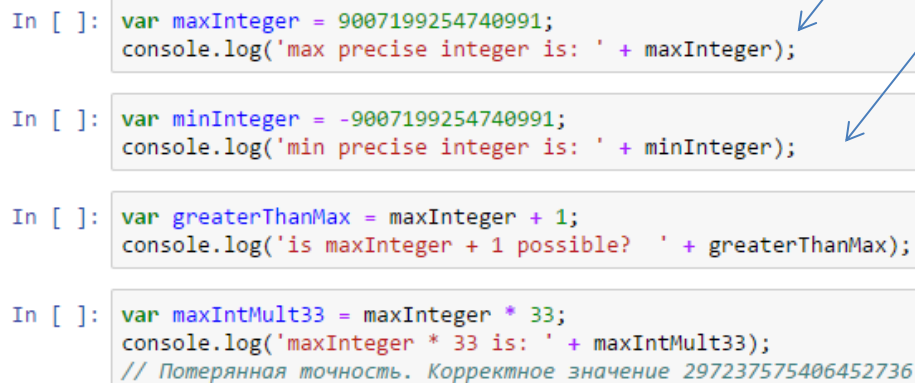
- определяем переменную `studentsCount`,
присваиваем значение 5

- с помощью `console.log` выводим результат

Нажимаем Shift + Enter и смотрим
результат

Рис. 8.5

Разберите все примеры (исполните код) в данном разделе



```
In [ ]: var maxInteger = 9007199254740991;
        console.log('max precise integer is: ' + maxInteger);

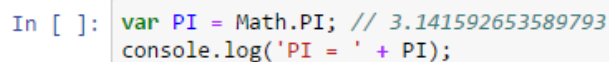
In [ ]: var minInteger = -9007199254740991;
        console.log('min precise integer is: ' + minInteger);

In [ ]: var greaterThanMax = maxInteger + 1;
        console.log('is maxInteger + 1 possible? ' + greaterThanMax);

In [ ]: var maxIntMult33 = maxInteger * 33;
        console.log('maxInteger * 33 is: ' + maxIntMult33);
        // Потерянная точность. Корректное значение 297237575406452736
```

Числа с плавающей точкой

- Являются диапазонами всех действительных чисел
- Могут быть заданы с точностью до определенного количества знаков после точки
- Могут вести себя ненормально при вычислениях



```
In [ ]: var PI = Math.PI; // 3.141592653589793
        console.log('PI = ' + PI);
```

И т.д.

Рис. 8.6

В этом разделе рассматривается в основном определение переменных и их вывод.

8.3. Примеры (demos)

Посмотрите примеры в папке demos (в локальном репозитории, который Вы скачали ранее, см. стр. 11-12)

Откройте их в Brackets

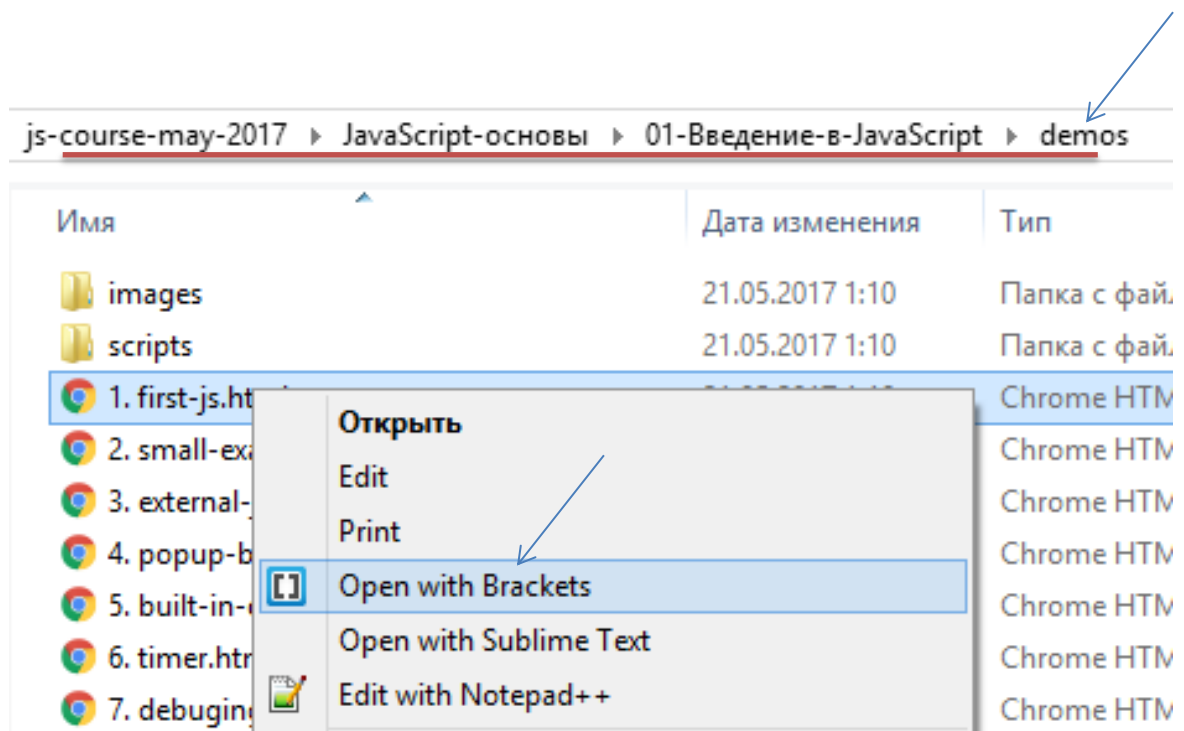


Рис. 8.7

Посмотрите код

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <title>Introduction to JavaScript - First JS</title>
5  </head>
6  <body>
7      <script>
8          alert("Hello JavaScript!");
9      </script>
10 </body>
11 </html>
```

Рис. 8.8

и посмотрите, что происходит в браузере с помощью Live Preview



Рис. 8.9

- Brackets

Откройте js-файл в Brackets.

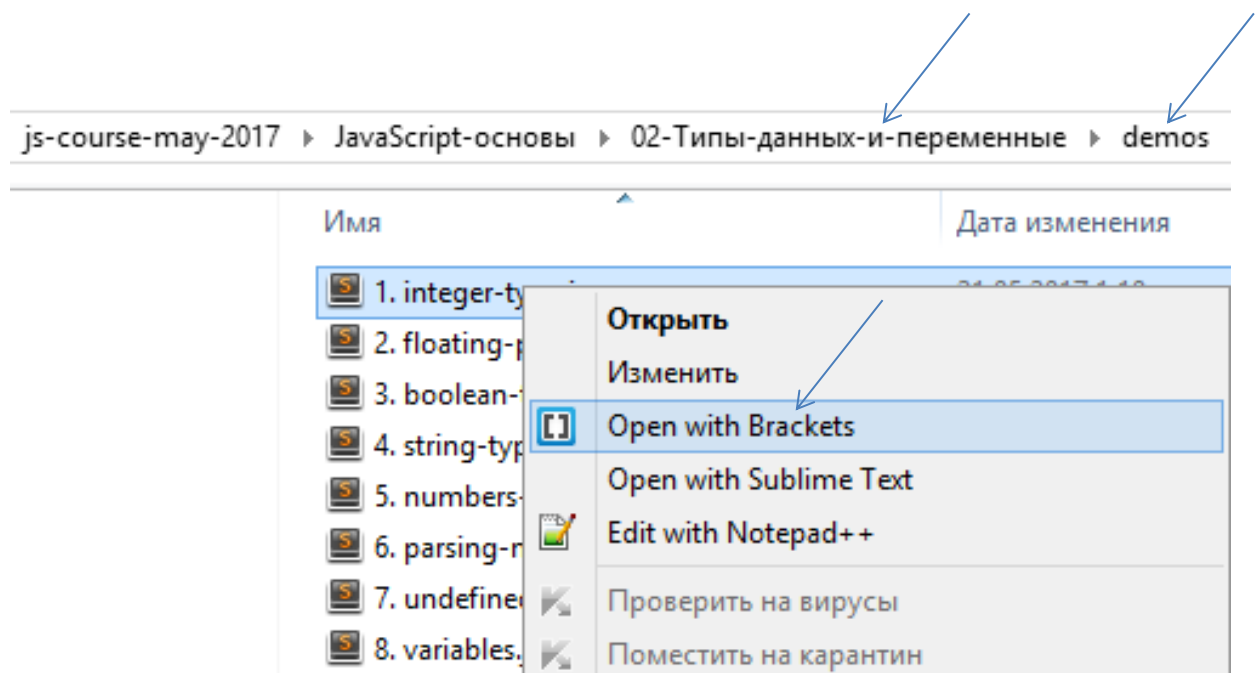


Рис. 8.10

Установите расширение NodeJS integration, если этого не сделали ранее (установка описана на стр. 17-18, рис. 7.3.-7.5.)

Исполните код

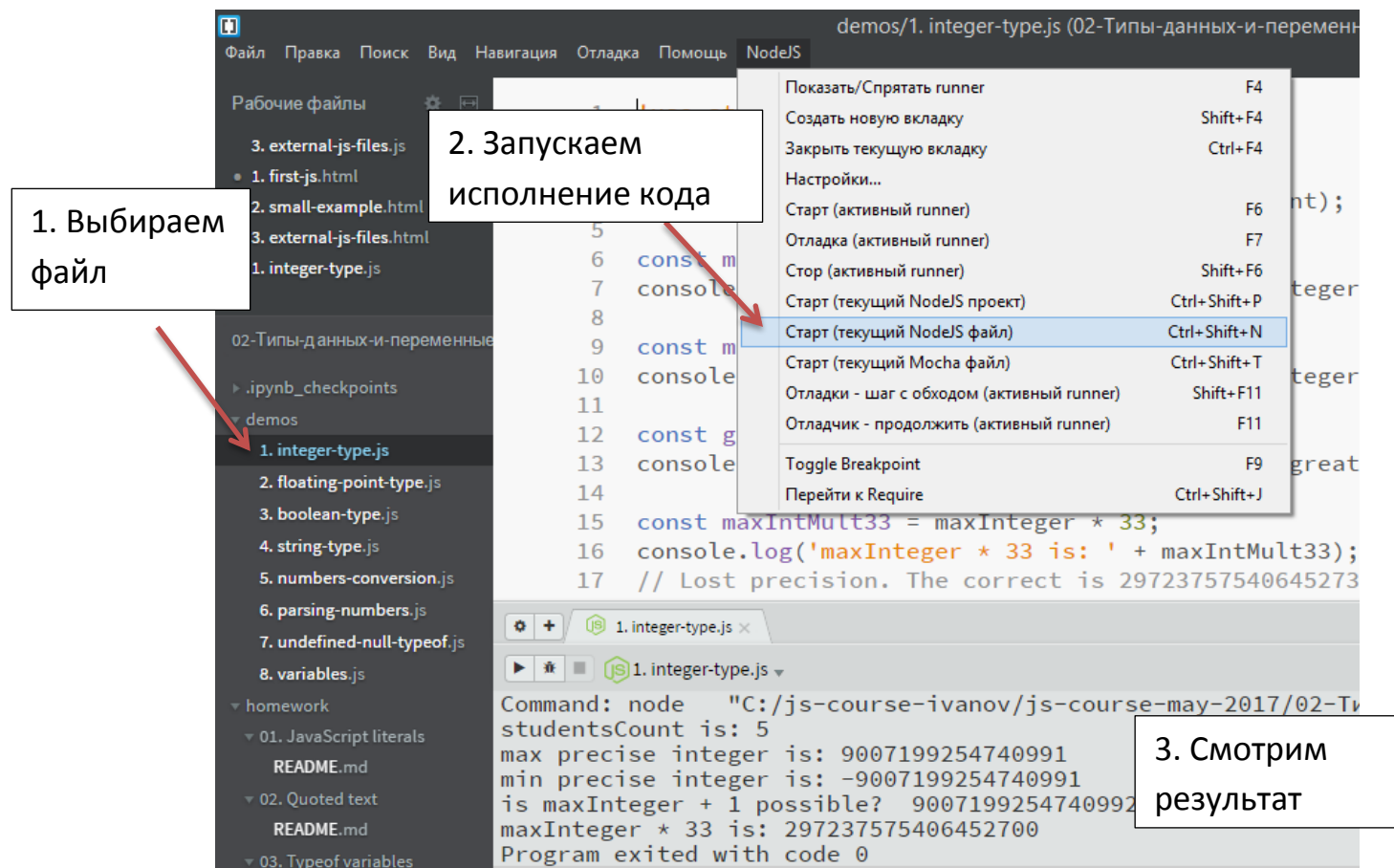


Рис. 8.11

Здесь мы посмотрели как выполняется js-код в Brackets.