

# Front end разработка на JavaScript

---

**JS**  
**COURSE**  
**ORT DNIPRO**

---

[ORT.DP.UA/JS](https://ort.dp.ua/js)

Первым делом

**Наша группа: JS10**

**js10.ortdnipro.org**



Общение при помощи  
мессенджера

**Telegram**, а для  
обмена материалами  
и домашних заданий  
будем использовать

**GitHub**

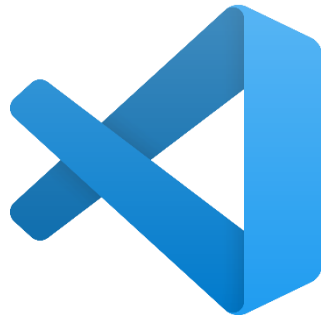


Что нам понадобится

# На понадобятся



## Браузер(ы)



## Visual Studio Code

*Или любой другой текстовый редактор*

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

# Live Server



**Live Server** ritwickdey.liveserver

Ritwick Dey | 6 850 405 | ★ ★ ★ ★ ★ | Repository | License | v5.6.1

Launch a development local Server with live reload feature for static & dynamic pages

**Disable ▼** **Uninstall** *This extension is enabled globally.*

**Live Server** – расширение к **Visual Studio Code** позволяющая запускать локальный **веб-сервер** с целью отладки и тестирования

<https://marketplace.visualstudio.com/items?itemName=ritwickdey.LiveServer>

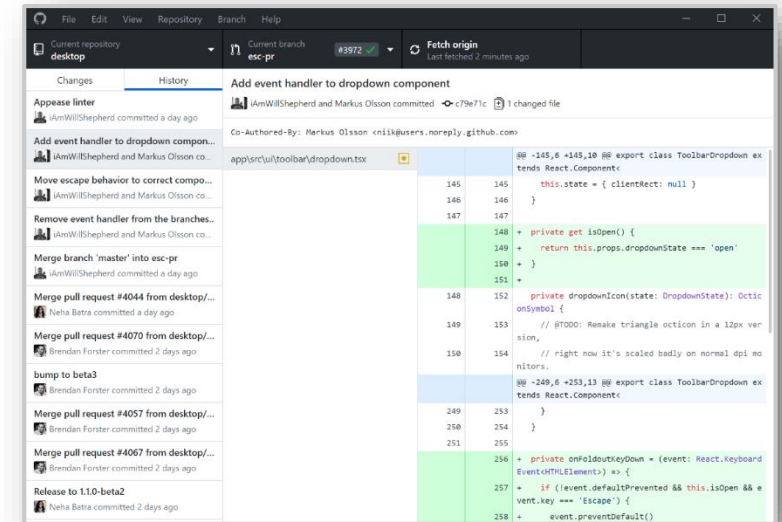
# Git клиент (для работы с GitHub)



или

<https://git-scm.com/>

Консольная версия  
(также используется VSCode)



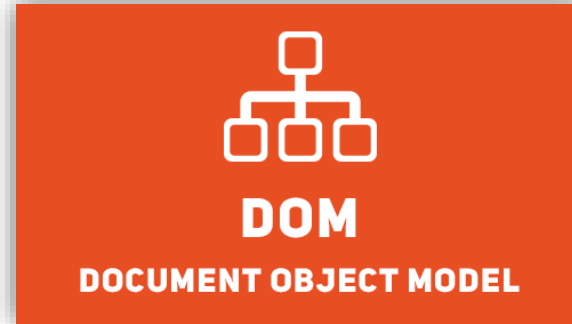
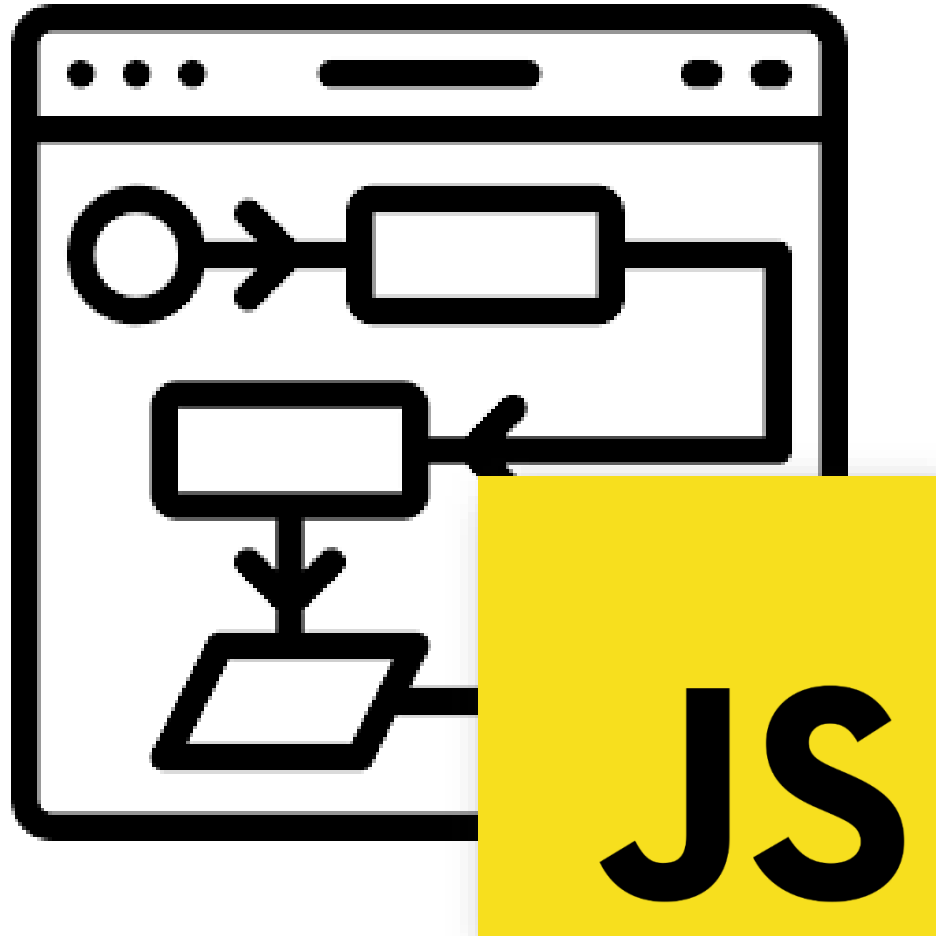
GitHub Desktop  
[WinX64, MacOS]

<https://desktop.github.com/>



О чём курс?

# О программировании и веб-разработке с применением языка JavaScript



Поехали!

# ES

**ECMAScript**

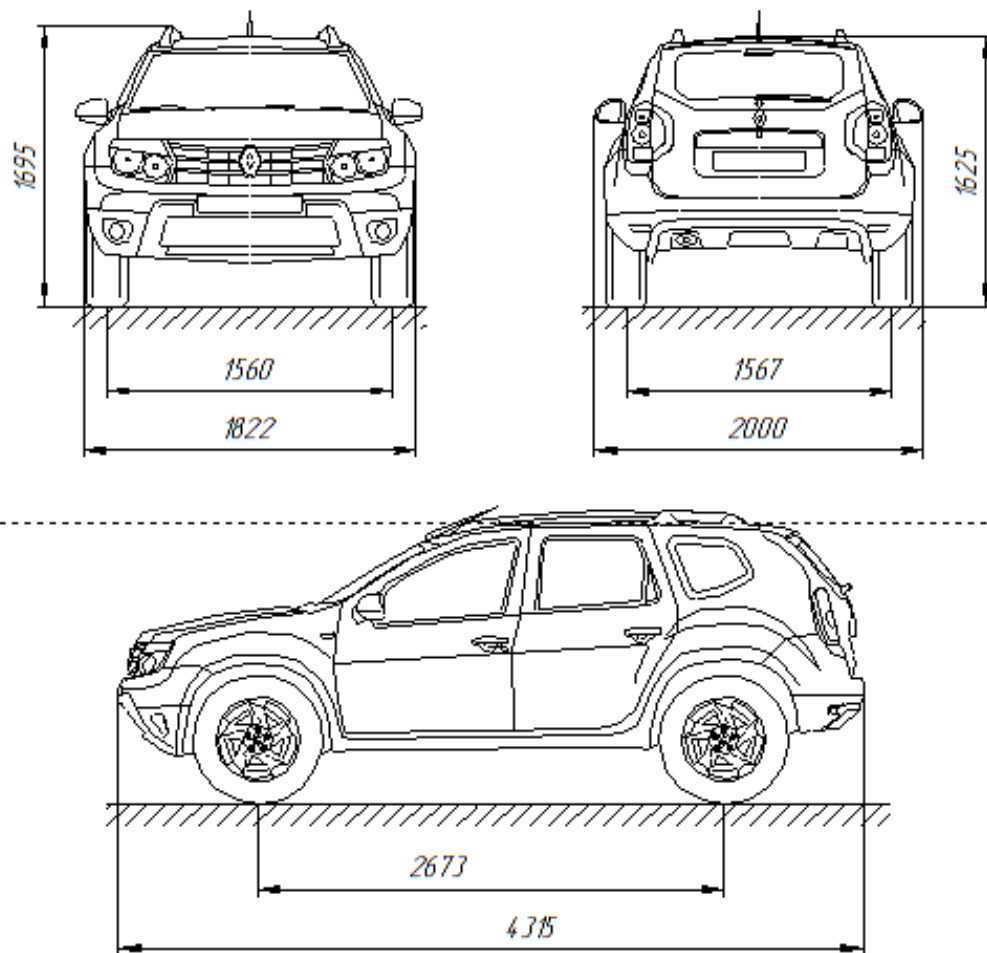
vs

# JS

**JavaScript**

# ECMAScript

# JavaScript



*Спецификация...*



*...и её реализация*

# JavaScript – язык программирования

**1. Императивный**

**2. Интерпретируемый**

**3. Не типизированный**

# JavaScript без полной разметки

F: > site > <> index.html > ...

```
1  <script>
2
3      let message = 'Hello world!';
4
5      console.log(message);
6
7  </script>
8
```

Мы имеем возможность ограничиться только тегам `<script></script>` для написания кода, и опускать полную разметку документа. **НО! чтобы корректно работал Live Server полная разметка таки нужна.**

# Служебные функции для ввода/вывода данных

*\* которые нам помогут прожить без разметки))*

**console.log(...);** - вывод в консоль браузера;

**alert(...);** - вывод во всплывающем окне;

**prompt(...);** - окно с запросом информации;

**confirm(...);** - окно для подтверждения;

**document.write()** – вывести данные в документ.

*\*\* кроме **console.log()** применение перечисленных методов считается плохой практикой в реальных задачах, но они могут нам помочь в процессе обучения.*



# Переменные и типы данных

# Переменные

```
1
2  var user_name    = "Elena";
3
4  let user_age     = 27;
5
6  const user_inn   = 3252873450;
7
8  console.log(user_name, typeof user_name);
9  console.log(user_age,  typeof user_age);
10 console.log(user_inn,  typeof user_inn);
11
```

Переменные объявляются при помощи ключевых слов **var**, **let** и **const**. Первые два способа отличаются областью видимости переменной которая создаётся. Третий создаёт переменную у которой нельзя заменить значения после инициализации.

Подробнее: <https://learn.javascript.ru/variables>

# Типы данных в JavaScript

```
3
4 undefined //undefined
5
6 number //42, -35.783, 4e18, NaN, Infinity ...
7
8 string //'Hello', "World", `!!!` ...
9
10 boolean //true, false
11
12 object //null, { prop:'value', ... } ...
13
14 symbol //Symbol('marker'), Symbol.for('label')
15
16 bigint //35n, 999999999999999999999999999999n
17
18 function //function(...){...}, (...)=> ...
19
```

Переменные могут хранить значение одного из поддерживаемых типов данных. В ходе выполнения кода может меняться как содержимое переменной так и его тип.

**Тип влияет на то какие операции могут быть выполнены с переменной.** Тип переменной можно получить при помощи оператора/функции **typeof**.

# Преобразование типов

```
3 let data = "12.35";  
4  
5 let a = Number(data);  
6  
7 let b = parseFloat(data);  
8 // let b = parseInt(...);  
9  
10 let c = +data;  
11  
12 console.log(a, b, c);  
13
```

Несмотря на наличие механизма автоматического приведения типов может возникать ситуации требующие принудительного преобразования типов (чаще всего **string** к **number**), для этого есть ряд возможностей.

# Операторы, операнды и операции

# Оператор присвоения

Чтобы сказать компьютеру, что именно нужно записать в переменную используется оператор присвоения =

**a = 2 + 3 \* 5;**



Оператор присвоения берёт то что справа от него и записывает в переменную имя которой расположено слева от него.

# Операторы, операнды и операции...

Для выполнения действий (**операций**) над переменными или значениями (операндами) используются **операторы**, операторов существует много. С некоторыми из них все знакомы, например с арифметические операторами.

---

**Унарный оператор** – тот который взаимодействует только с одной переменной (операндом).

```
a++;  
b--;
```

**Бинарный оператор** – тот который взаимодействует с двумя переменными (операндами).

```
a = b**2 + 4*a*c;
```

---

У операторов есть приоритеты, какой приоритет выше, какой ниже запомнить непросто. Поэтому в случае сомнений какая операция будет первой а какая второй – смело используйте скобки. Принцип их применения такой же как и в математике – скобки повышают приоритет операции в них записанной.

```
a = (2+2) * 2;
```

«Скобками программу не испортишь» (с)

Подробнее: <https://learn.javascript.ru/operators>

## Операторы, операнды и операции...

### Что получится?

```
2  
3  let x = 5;  
4  
5      x = x++ + ++x;  
6  
7  console.log(x);  
8
```

?!?



# Выражения

По правую сторону от оператора присвоения может быть конкретное значение или же **выражение**, одна или несколько операций, результат выполнения которых будет записан в переменную имя которой стоит слева от знака присвоения. В выражении могут участвовать как и конкретные значения так и другие переменные.

$$a = b**2 + 4*a*c;$$

# NaN – Not a Number

Значение NaN (`typeof:number`) – в результате выполнения операции(й) означает, что среди операндов есть тот кто не являются `number`'ом или не может быть приведено к типу `number`.

# Математические функции

# Объект Math

```
2  
3   let data = 144;  
4  
5   let result = Math.sqrt(data);  
6  
7   console.log(data, result);  
8
```

**Math** - это встроенный объект с полями и методами для реализации математических постоянных и функций (в частности функции округления чисел).

Подробнее: <https://javascript.ru/math>

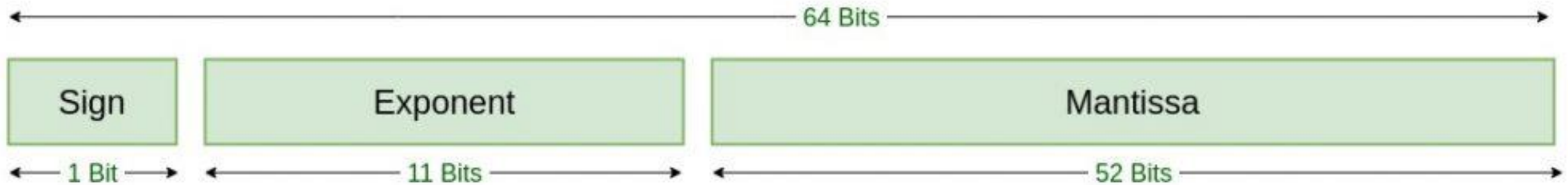
# Округление чисел

## Округление чисел в JS

```
2  let data = 723.53767347;
3
4
5  console.log( "Trunc",    Math.trunc(data) ); //Round to integer
6
7  console.log( "Floor",    Math.floor(data) ); //Round to integer
8
9  console.log( "Round",    Math.round(data) ); //Round to integer
10
11 console.log( "Ceil",     Math.ceil(data) ); //Round to integer
12
13 //If we need 2 digit after dot...
14 console.log(".toFixed()", +data.toFixed(2) );
15
16 //If we need 2 digit after dot... (second way)
17 data *= 100;
18 data = Math.round(data);
19 data /= 100;
20 console.log("Math 'focus'", data);
21
```

**IEEE 754 / float**

# Стандарт хранения вещественных чисел IEEE 754



$$D = (-1)^S \cdot \left(1 + \frac{M}{2^{52}}\right) \cdot 2^{E-1023}$$

Подробнее: [https://ru.wikipedia.org/wiki/IEEE\\_754-2008](https://ru.wikipedia.org/wiki/IEEE_754-2008)



# Полезные методы и свойства объекта Number

# Методы объекта **Number** и переменных типа **number**

`Number.isNaN()` 

Определяет, является ли переданное значение значением `NaN`.

`Number.isFinite()` 

Определяет, является ли переданное значение конечным числом.

`Number.isInteger()` 

Определяет, является ли тип переданного значения «числом», а само число — целым значением.

`Number.isSafeInteger()` 

Определяет, является ли переданное значение безопасным целым числом (числом в диапазоне от  $-(2^{53} - 1)$  до  $2^{53} - 1$ ).

...

Подробнее: [https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global\\_Objects/Number](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Number)

# Сравнение чисел с учётом погрешности

```
2
3   let a = 0.1;
4   let b = 0.2;
5   let c = a + b;
6   let control = 0.3;
7
8   console.log("c == control", c == control); //?!?!?!
9   console.log("c", c); //0.30000000000000004 WTF?!?!?!
10
11  //How compare two numbers?
12  let diff = Math.abs(c - control);
13  let isEqual = diff < Number.EPSILON;
14  console.log("c == control", isEqual);
15
```

Подробнее: [https://ru.wikipedia.org/wiki/IEEE\\_754-2008](https://ru.wikipedia.org/wiki/IEEE_754-2008)

# String

# Строки – текстовый тип данных

```
2
3   let name    = 'Jane';
4
5   let phrase  = `Hello ${name}! Welcome!`;
6
7   console.log(phrase);
8
```

Строки могут быть заданы при помощи одинарных и двойных кавычек. А с помощью обратных («косых») кавычек можно создать строку с подстановкой в неё значений переменных или выражений - т.н. шаблонные строки.

Подробнее: <https://learn.javascript.ru/string>

# Строки – текстовый тип данных

```
2  
3   let str = 'Hello world!';  
4  
5   console.log(str.length);  
6  
7   console.log(str[2], str[7]);  
8
```

У строк есть понятие длины (количества символов), узнать которую можно при помощи свойства **.length**, также есть возможность обращаться к конкретному символу по его номеру (индексу), при помощи оператора **[]**.

Подробнее: <https://learn.javascript.ru/string>

# Преобразование к строке

```
2
3   let a = 42;
4   let b = true;
5
6   console.log( a.toString() );
7   console.log( b.toString() );
8   console.log( String(a) );
9   console.log( String(b) );
10
```

Любые типы можно привести к строке, для этих целей можно вызвать метод **.toString()** или воспользоваться функцией **String()**

# Тип `boolean` и условные операторы



# Тип Boolean

```
2  
3   let a    = true;  
4   let b    = false;  
5  
6   console.log(a, typeof(a));  
7   console.log(b, typeof(b));  
8
```

Переменная типа **boolean** содержит одно из двух возможных значений: истина (**true**) или ложь (**false**).

```

2      Boolean(undefined); //false;
3
4
5      // Number (and BigInt) to Boolean
6      Boolean(42);         //true;
7      Boolean(-23.45);     //true;
8      Boolean(0);          //false;
9      Boolean(0.000001);   //true;
10     Boolean(NaN);         //false;
11
12     //String to Boolean
13     Boolean("Hello");     //true;
14     Boolean(" ");         //true;
15     Boolean("");          //false (if zero length);
16
17     //Object to Boolean
18     Boolean(null);        //false;
19     Boolean({ name: 'Jane' }); //true;
20     Boolean({});          //true;
21
22     //Symbol to Boolean
23     Boolean(Symbol('my-symbol')); //true;
24

```

## Откуда берётся **boolean**?

Из преобразования типов, явного (при помощи **Boolean()** или **!!**) или неявного (в условных операторах, циклах...).

# Оператор if-else – основной «клиент» boolean

```
2
3     if( some_boolean ){
4         //if some_boolean is true;
5     }else{
6         //if some_boolean is false;
7         //false branch is optional;
8     }
```

Оператор **if-else** в зависимости от переданного (**true** или **false**) значения выполняет один из двух блоков кода (**первый** или **второй**, соответственно), другой блок при этом не выполняется. Если значение переданное оператору **if** не является **boolean**'ом будет выполнено неявное преобразование. Ветка **else** не является обязательной.

# Откуда берётся boolean?

## Операторы сравнения

>	<	>=	<=	==	!=	===	!==
---	---	----	----	----	----	-----	-----

```
2  
3     var a = 8;  
4     var b = 7;  
5  
6     var c = a >= b;  
7  
8     console.log(c, typeof(c));  
9
```

Подробнее: <https://learn.javascript.ru/comparison>

# Откуда берётся boolean?

>	<	>=	<=	==	!=	===	!==
---	---	----	----	----	----	-----	-----

```
2
3   let a = 6;
4   let b = 500;
5
6   let c = "6";
7   let d = "500";
8
9   console.log(a > b);
10  console.log(a > d);
11  console.log(c > d);
12
```

Есть нюансы с типами... При сравнении разных типов происходит их преобразование к **number**

# Откуда берётся boolean?

>	<	>=	<=	==	!=	===	!==
---	---	----	----	----	----	-----	-----

```
2
3   let a = "Ivan";
4   let b = "Iven";
5
6   console.log(a > b);
7   console.log(a < b);
8   console.log(a == b);
9
```

Сравнение строк  
осуществляется  
посимвольно.  
Выполняется  
сравнение кодов  
символов.

# Логические операторы

Когда нужны «сложные» условия

&&		
----	--	--

```
2
3      let a = 80;
4      let b = 500;
5
6      if( a > 1 && b < 1000 ){
7          //...
8      }else{
9          //...
10     }
11
```

Подробнее: <https://learn.javascript.ru/logical-ops>

# Логические операторы

## Таблицы истинности

<b>&amp;&amp;</b>	<b>False</b>	<b>True</b>
<b>False</b>	False	False
<b>True</b>	False	True

<b>  </b>	<b>False</b>	<b>True</b>
<b>False</b>	False	True
<b>True</b>	True	True

<b>!</b>	<b>False</b>	<b>True</b>
	True	False

Операторы логическое И (**&&**) и логическое ИЛИ (**||**) работают по такой схеме: Приводят левый операнд к **boolean**. Если по нему можно сделать выводы (будет выражение, в целом, верным или ложным), то возвращают левый операнд (в том типе в котором он и был). Если нет, то возвращают правый операнд (в том типе в котором он и был). Логические операторы **&&** и **||** могут не проверять правый операнд, если значение левого операнда уже достаточно для итогового результата выражения (*это важно если правый операнд - вызов функции*).



# «Многоэтажный» if-else-if

```
2
3   let t = 18; //temperature;
4
5   if( t < 0 ){ //from (-∞ to 0)
6       | //...very cold...
7   }else if(t < 14){ //from [0 to 14)
8       | //...cold...
9   }else if(t < 24){ //from [14 to 24)
10      | //...comfort...
11   }else if(t < 32){ //from [24 to 32)
12      | //...hot...
13   }else{ //from [32 to ∞)
14      | //...very hot...
15   }
```

Многоэтажных **if-else-if...**

Хорош для задач в которых больше чем два вариант развития событий, или когда значения нужно распределить по диапазонам.

Немного практики #1

***Задача:*** Мы знаем день, месяц и год рождения человека. Мы также знаем сегодняшний день, месяц и год, необходимо рассчитать сколько человеку полных лет на сегодняшний день.

Немного практики #2

**«Задача банкомата»** Программа спрашивает у пользователя сумму, а в ответ сообщает купюры каких номиналов, и в каком количестве необходимо выдать. При этом *суммарное количество купюр должно быть минимально возможным*. Для простоты будем считать, что в банкомате есть только купюры по **1, 5, 20, 50** гривен (при этом из количество не ограничено).

*Например: Пользователь вводит сумму: 552 грн.*

*В ответ программа выдаёт:*

50 грн. x 11;

20 грн. 0;

5 грн. x 0;

1 грн. x 2;

Немного практики #3

**Задача:** Тарифы банка за перевод средств с карты на карту: 1% за счёт личных средств и 4% в счёт кредитного лимита. Скрипт должен рассчитывать сумму комиссии за перевод (который хочет выполнить пользователь), и определять возможно ли выполнить перевод.

«Источники знаний»



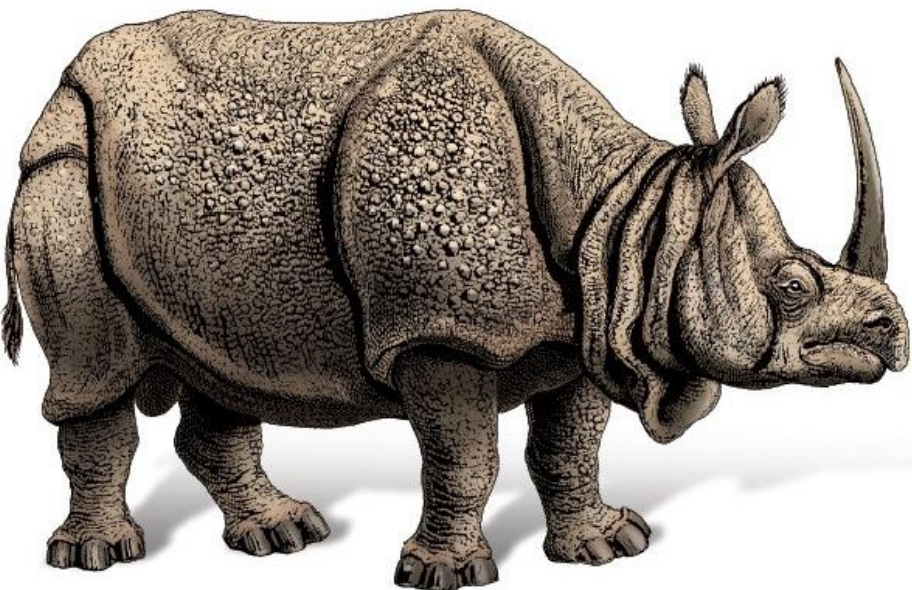
O'REILLY®

Seventh  
Edition

# JavaScript

## The Definitive Guide

Master the World's Most-Used  
Programming Language



David Flanagan

**Flanagan David**

**JavaScript: The Definitive  
Guide: Master the World's  
Most-Used Programming  
Language. 7th edition, 2020.**

# Современный учебник Javascript

Перед вами учебник по JavaScript, начиная с основ, включающий в себя много тонкостей и фишек JavaScript/DOM.

[смотреть на Github](#)

Поделиться:

[НАЙТИ](#)

## Содержание

Первые две части посвящены JavaScript и его использованию в браузере. Затем идут дополнительные циклы статей на разные темы.

<http://learn.javascript.ru/>

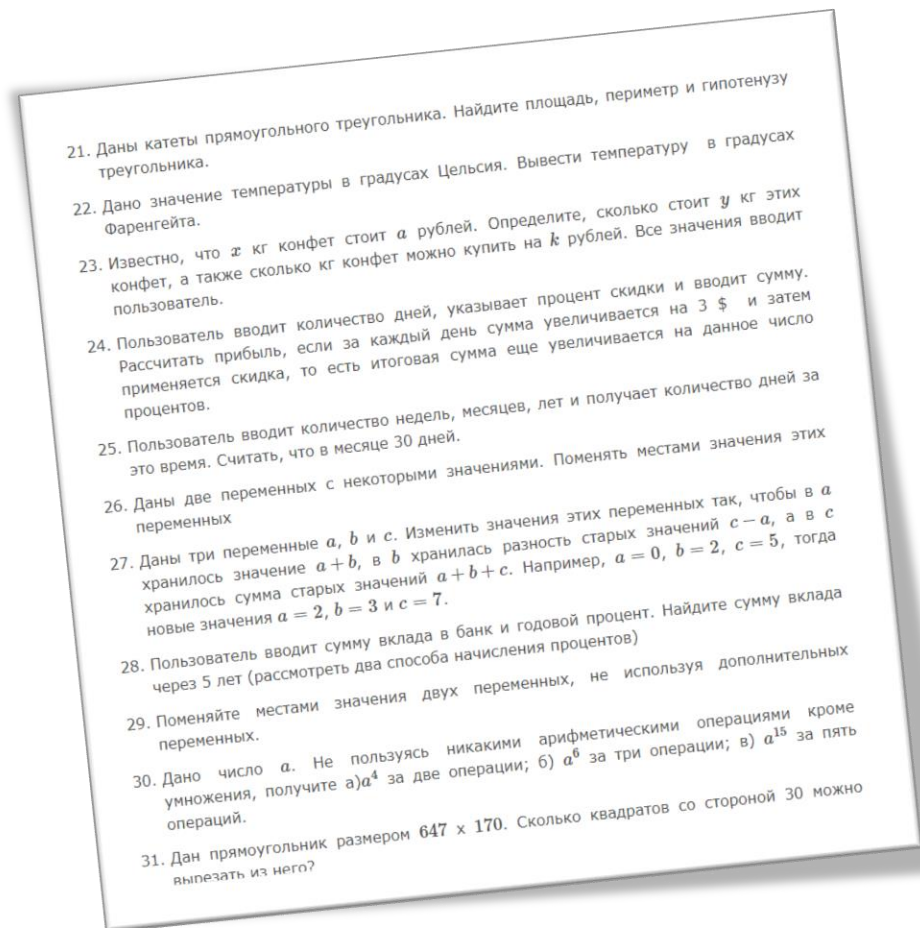
К следующему занятию будет  
полезно почитать о...

# К следующему занятию...

1. Коллекции в JavaScript: *Массивы* (**Array**), *Ассоциативные массивы* (**Object**, **Map**), *Множества* (**Set**);
2. Spread оператор (**...**) – он же оператор «три точки», он же оператор **деструктуризации**;
3. Методы массива (**Array**) *.splice()*, *slice()*, *.join()*, *.includes()*, *indexOf()*, *.lastIndexOf()*, *.reverse()*.

Скучное но необходимое  
домашнее задание  
для тренировки

# Домашнее задание



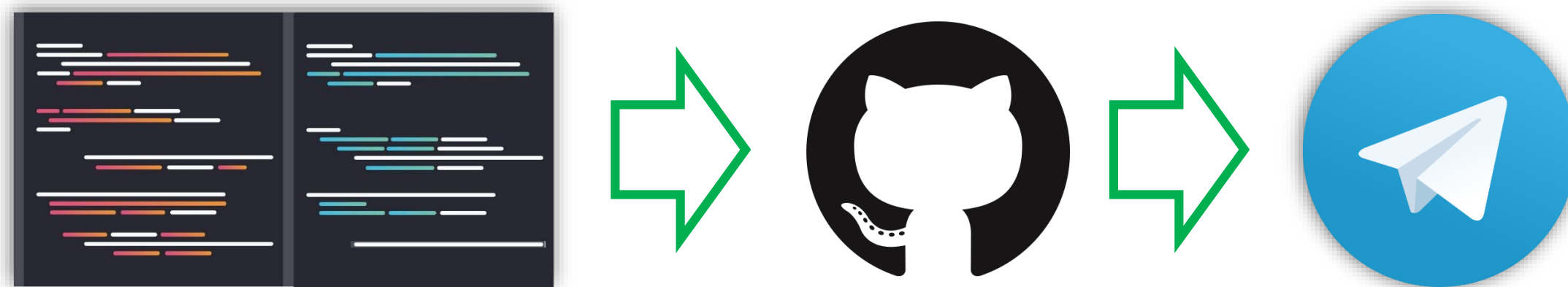
Первым делом необходимо **натренироваться применять базовые конструкции**. Поэтому напишите код который решит задачи (№7-№74) из раздела «Простейшая арифметика» и «Условный оператор и арифметика».

Решения этих задач **загружать на GitHub не нужно** (правильность их работы можно легко проверить с ответами, или калькулятором). Но если возникнут проблемы, не стесняйтесь задавать вопросы. Для этого **загрузите ваш код** (в котором возникли проблемы) **на GitHub**, и ссылку на него **с описанием проблемы** сбросьте в группу.

<http://www.itmathrepetitor.ru/prog/zadachi-na-vychisleniya/>

Домашнее задание  
/сделать

Каждое домашнее задание оформляйте в виде отдельного репозитория на GitHub, в названии которого **укажите код задания** (например: **A1 Federal Tax**)



Если есть проблемы, вопросы, трудности, делаем тоже самое – код с проблемой заливаем на **GitHub** и ссылку на него, с описанием вопроса в **группу**.



# Домашнее задание #А.1



Есть в США такой вид налога как **Federal Income Tax**, ваша задача написать налоговый калькулятор, который будет рассчитывать сумму налогов в зависимости от годового дохода человека. За основу взять ставки налога для доходов полученных за **2020** г. (и оплата в 2021 году), и для простоты - расчёт выполнять **только** для лиц не состоящих в браке: **single**.

<https://www.debt.org/tax/brackets/>

*О прогрессивном налогообложении в целом, с примерами:*

<http://allfi.biz/glossary/eng/P/progressive-taxation.php>

Не забудьте

**Наша группа: JS10**

**js10.ortdnipro.org**