

Front end разработка на JavaScript

JS
COURSE
ORT DNIPRO

ORT**DNIPRO**.ORG/**JS**

Первым делом

Наша группа: JS12

<https://js12.site>



Общение при помощи
мессенджера

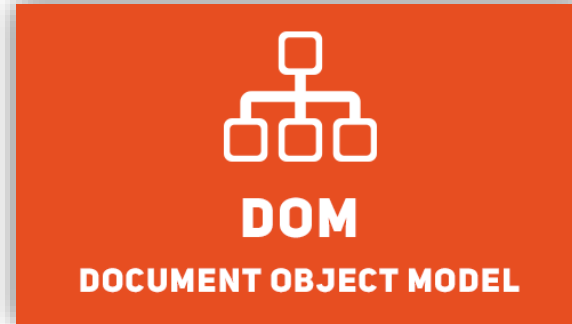
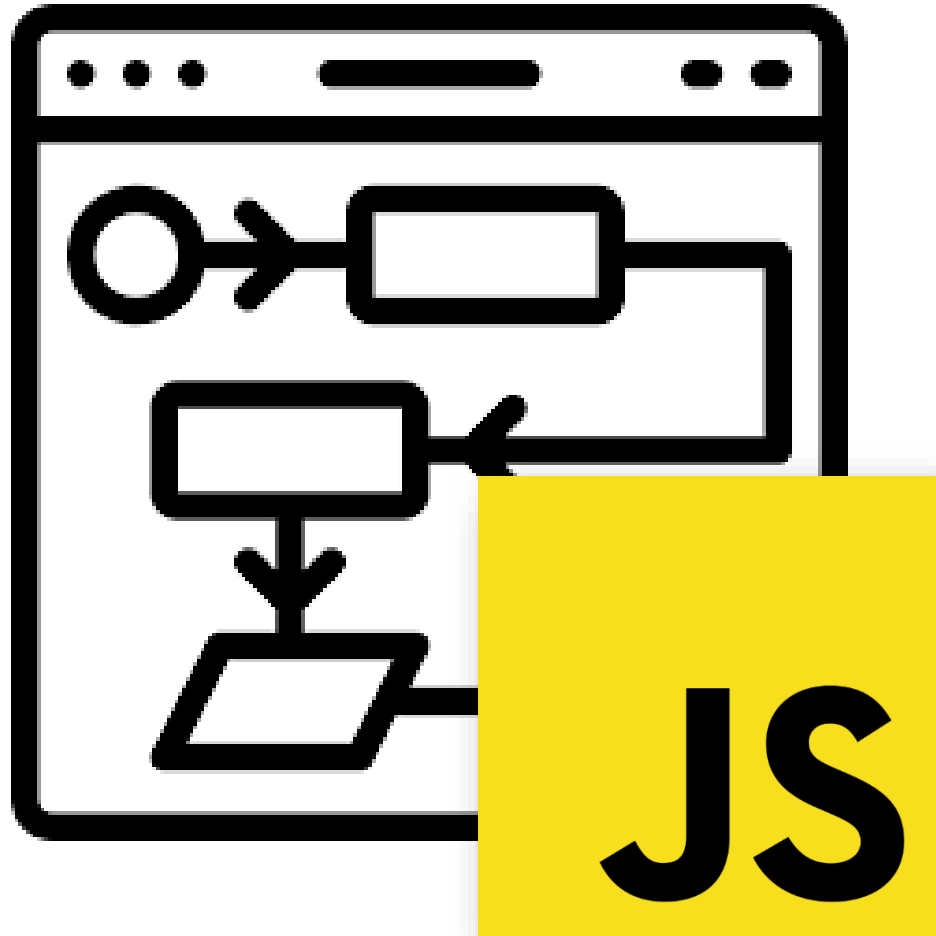
Telegram, а для
обмена материалами
и домашних заданий
будем использовать

GitHub



О чём курс?

О программировании и веб-разработке с применением языка JavaScript



Поехали!

ES

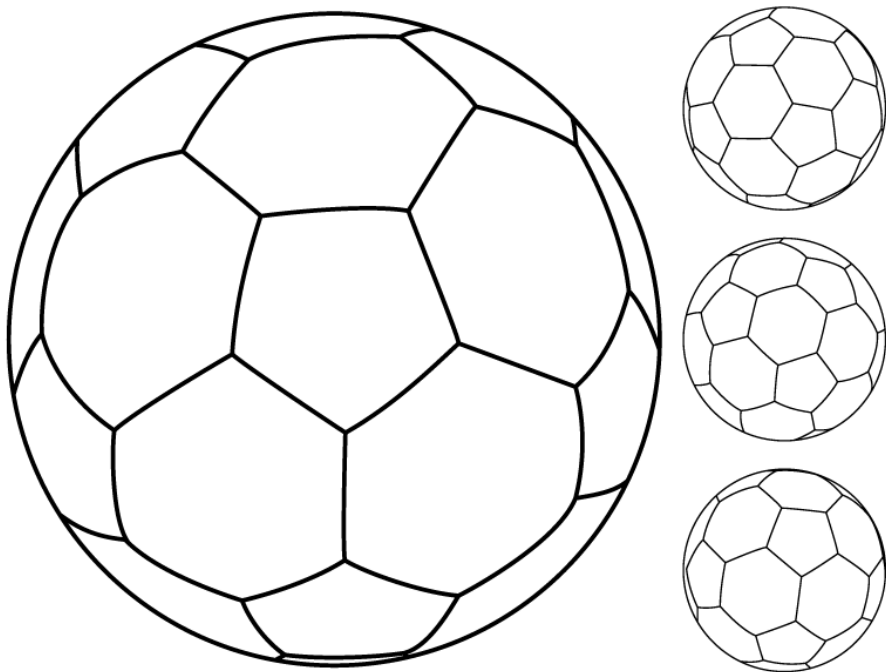
ECMAScript

vs

JS

JavaScript

ECMAScript



Спецификация...

JavaScript



...и её реализация

Развитие JavaScript...



JavaScript – язык программирования

1. Императивный

2. Интерпретируемый

3. Не типизированный

1. 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. Переменные объявленные через **var** добавляются в качестве свойств к глобальному объекту **window/globalThis** и могут «затереть» существующие служебные свойства.

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, 9999999999999999999999999999999n
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

Несмотря на наличие механизма автоматического приведения типов может возникать ситуации требующие принудительного преобразования типов (чаще всего **string** к **number**), для этого есть ряд возможностей. В первую очередь при помощи соответствующих функций-конструкторов **Number()**, **String()**, **Boolean()**, **BigInt()**, **Symbol()** и т.д.

2. 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>

3. 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
```

4. 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

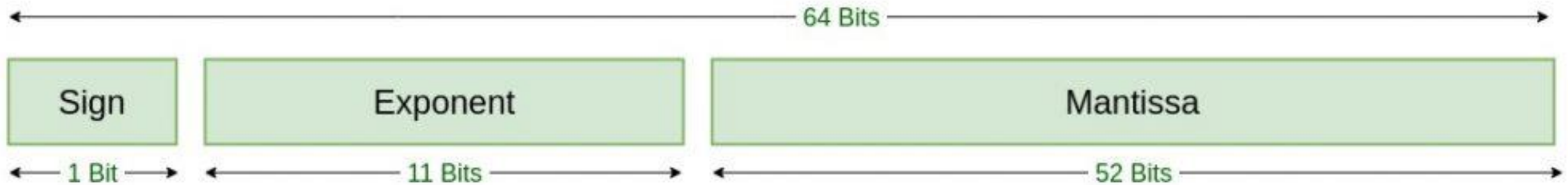
Сравнение чисел с учётом погрешности

```
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

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

5. Двоичная система счисления 0/1

BIN: 0; DEC: 0; HEX: 0
BIN: 1; DEC: 1; HEX: 1
BIN: 10; DEC: 2; HEX: 2
BIN: 11; DEC: 3; HEX: 3
BIN: 100; DEC: 4; HEX: 4
BIN: 101; DEC: 5; HEX: 5
BIN: 110; DEC: 6; HEX: 6
BIN: 111; DEC: 7; HEX: 7
BIN: 1000; DEC: 8; HEX: 8
BIN: 1001; DEC: 9; HEX: 9
BIN: 1010; DEC: 10; HEX: a
BIN: 1011; DEC: 11; HEX: b
BIN: 1100; DEC: 12; HEX: c
BIN: 1101; DEC: 13; HEX: d
BIN: 1110; DEC: 14; HEX: e
BIN: 1111; DEC: 15; HEX: f
BIN: 10000; DEC: 16; HEX: 10
BIN: 10001; DEC: 17; HEX: 11
BIN: 10010; DEC: 18; HEX: 12
BIN: 10011; DEC: 19; HEX: 13
BIN: 10100; DEC: 20; HEX: 14
BIN: 10101; DEC: 21; HEX: 15
BIN: 10110; DEC: 22; HEX: 16

Двоичная система счисления

```
1  
2   for(let i = 0; i <= 100; i++){  
3  
4       console.log(`  
5           BIN: ${i.toString(2)};  
6           DEC: ${i.toString(10)};  
7           HEX: ${i.toString(16)};  
8       `);  
9  
10  }
```

Разрядность системы счисления зависит от количества цифр используемых для формирования чисел, в остальном отличий от привычной нам десятичной системы нет. В **JavaScript** метод **.toString(N)** позволяет вывести число в нужной системе счисления (разрядной которой задаётся параметром **N**).

6. Битовые операции

Битовые операции

```
1
2 let a = 67;
3
4 let b = 23;
5
6 let x = a & b; //Bitwise AND
7 let y = a | b; //Bitwise OR
8 let z = a ^ b; //Bitwise XOR
9
```

		AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Битовые операторы выполняют операции над битами числа

7. Симметричное шифрование

Симметричное шифрование

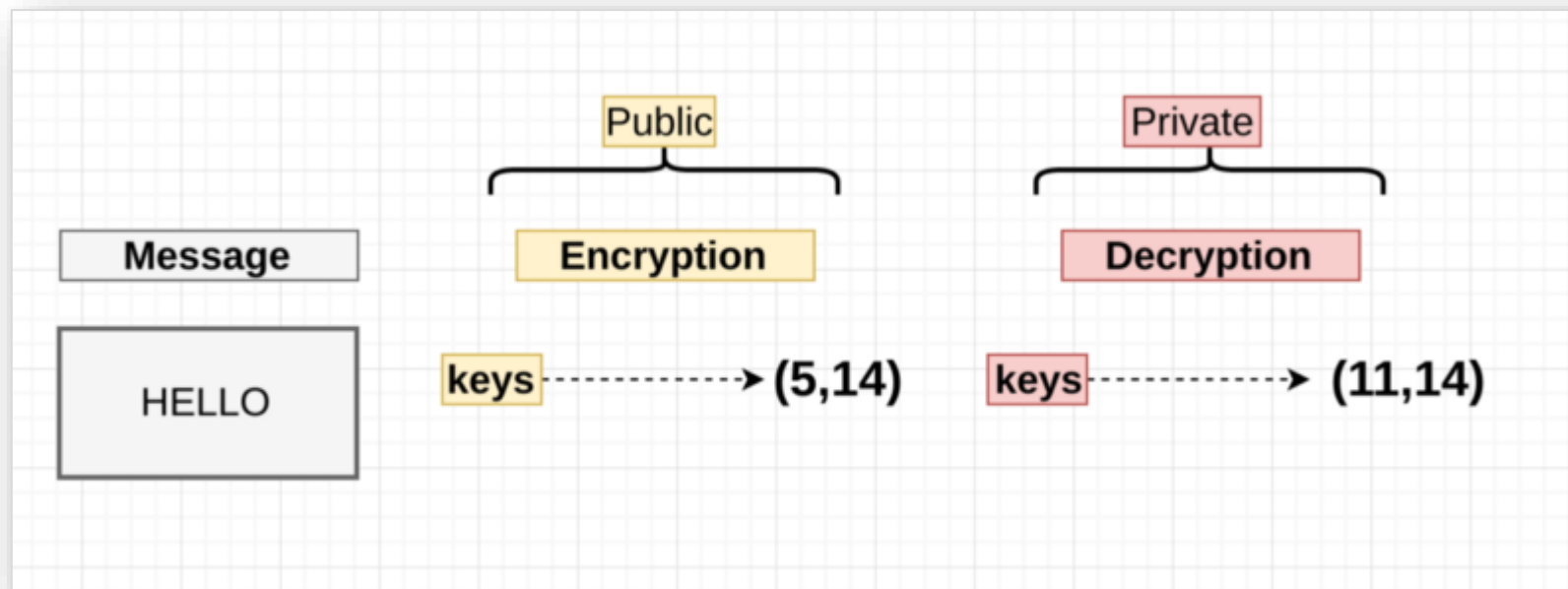
```
1
2 let myData = 'Forza Ferrari!';
3
4 const KEY = 42;
5
6 let chars = [...myData]
7   .map(i => i.charCodeAt(0));
8
9 console.log('Original Chars: ', chars);
10
11 /* Encrypt */
12
13 let encryptedChars = chars.map(i => i ^ KEY);
14
15 console.log('Encrypted Chars: ', encryptedChars);
16
17 console.log('Encrypted Text: ', String
18   .fromCharCode(...encryptedChars));
19
20 /* Decrypt */
21
22 let decryptedChars = encryptedChars.map(i => i ^ KEY);
23
24 console.log('Decrypted Chars: ', decryptedChars);
25
26 console.log('Decrypted Text: ', String
27   .fromCharCode(...decryptedChars));
28
29
```

Симметричное шифрование – использует один и тот же ключ для шифровки и расшифровки данных.

https://ru.wikipedia.org/wiki/Симметричные_криптосистемы

8. Ассиметричное шифрование

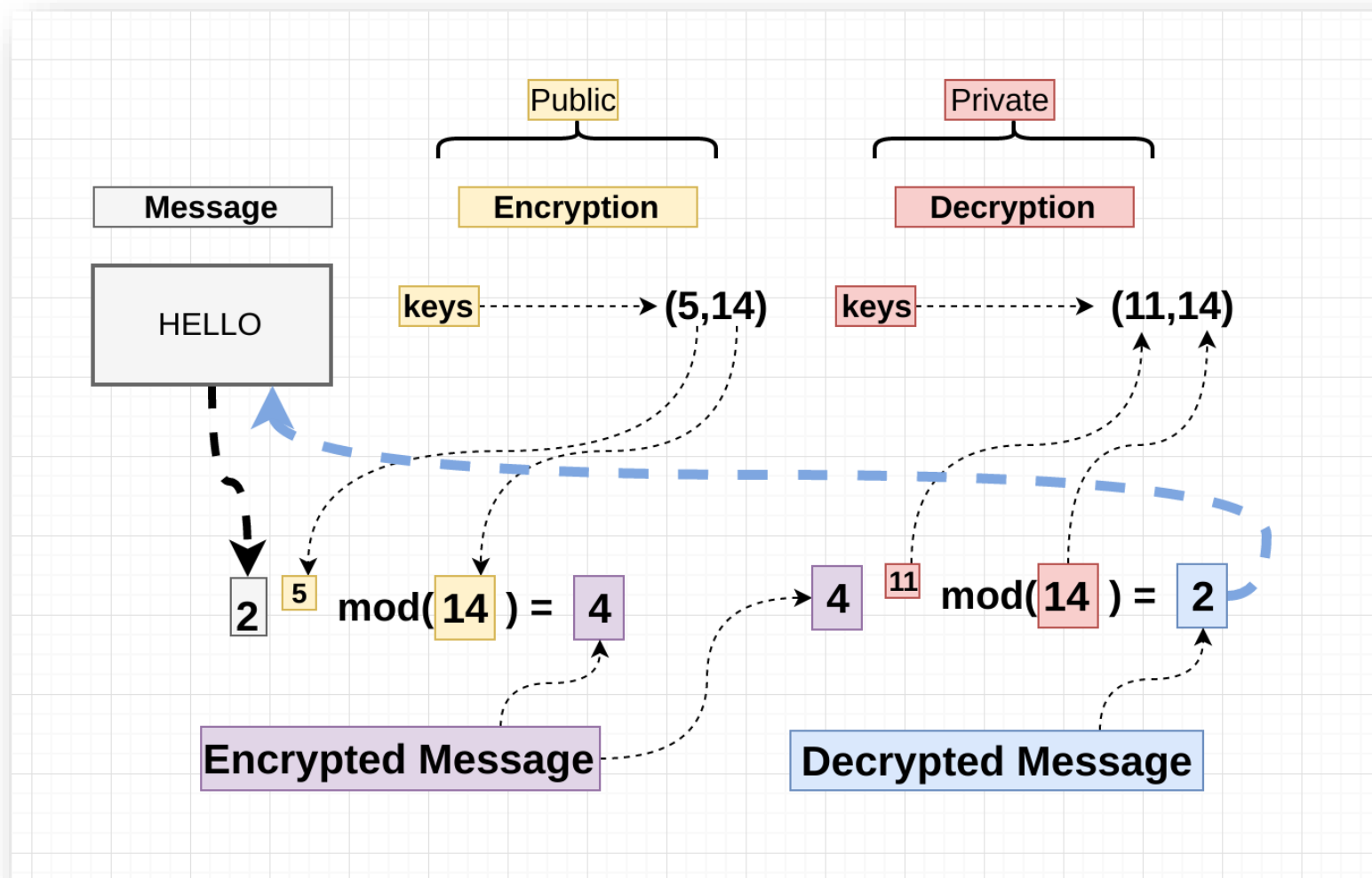
Ассиметричное шифрование (алгоритм RSA)



Ассиметричное шифрование – использует разные ключи (открытый и закрытый) для шифрования и расшифровки данных.

https://ru.wikipedia.org/wiki/Криптосистема_с_открытым_ключом

Ассиметричное шифрование (алгоритм RSA)



Ассиметричное шифрование – использует разные ключи (открытый и закрытый) для шифрования и расшифровки данных.

Алгоритм RSA / Генерация ключей

1. Выбираем два простых числа P и Q ;
2. Находим $N = P * Q$;
3. Находим $F = (Q - 1) * (P - 1)$;
4. Подбираем число E , которое должно быть простым, быть меньше F и их максимальный общий делитель был 1;
5. Выбираем число D удовлетворяющее $D * E \% F == 1$;
6. Теперь у нас есть пара ключей (E, N) и (D, N) ;

9. 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...).

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>

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

&&	False	True
False	False	False
True	False	True

 	False	True
False	False	True
True	True	True

!	False	True
	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*).

10. Немного практики

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.

«Источник знаний»

O'REILLY®

7-е
издание

JavaScript Полное руководство

Справочник по самому популярному
языку программирования



Дэвид Флэнаган

Дэвид Флэнаган

JavaScript: Полное
руководство, 7-е издание

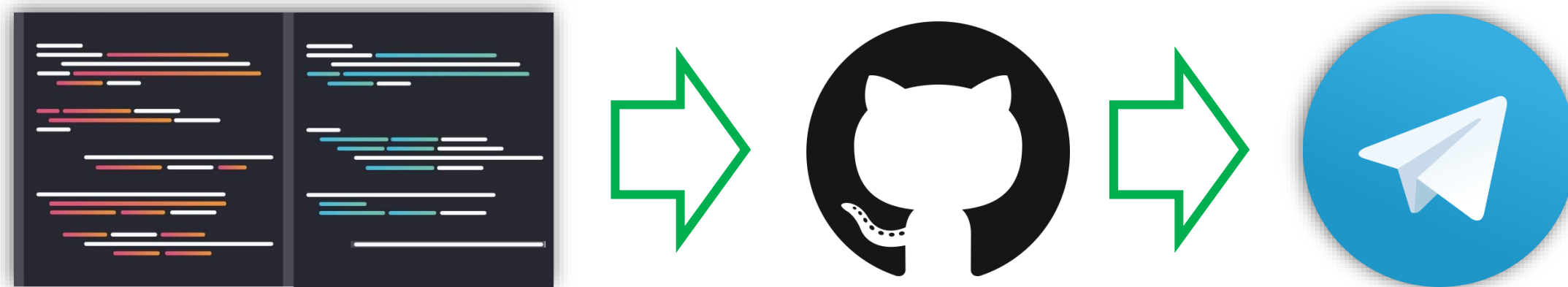
**К следующему занятию будет
полезно почитать о...**

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()*.

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 **2021** г. (с оплатой в **2022** году), и для простоты - расчёт выполнять **только** для лиц не состоящих в браке: **single**.

<https://www.bankrate.com/finance/taxes/tax-brackets.aspx>

В репозитории занятия есть тестер: `./src/homework-tester` для сверки. Расхождением в **~1-2 доллара** можно пренебречь.

О прогрессивном налогообложении в целом, с примерами:
<http://allfi.biz/glossary/eng/P/progressive-taxation.php>

Не забудьте

Наша группа: JS12

<https://js12.site>