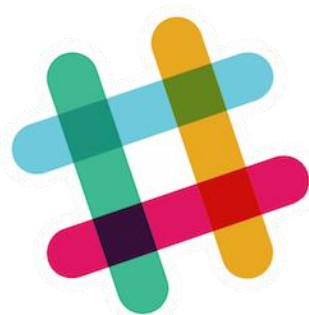
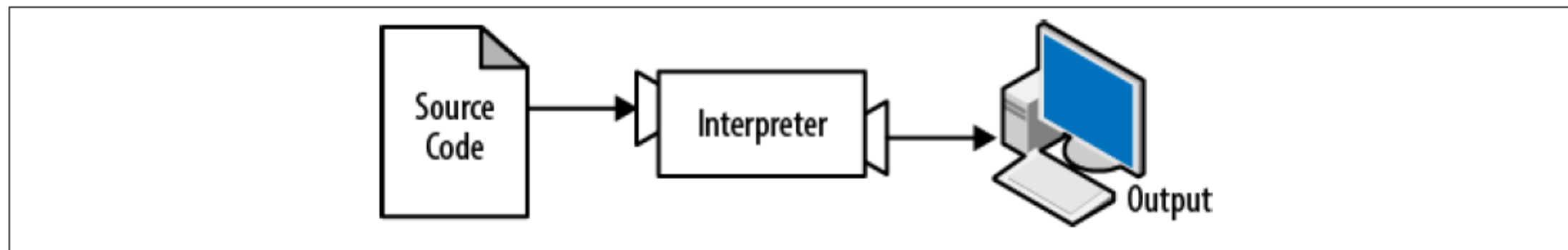


# Введение в JavaScript. Основные конструкции языка, переменные, типы данных.

# JavaScript

- «Сделать веб-страницы живыми»
- JavaScript не Java
- Для работы нужен интерпретатор
- Работает везде где есть интерпретатор

# Интерпретатор



# Движки браузеров

- Chrome – V8
- Mozilla Firefox – SpiderMonkey
- Safari – JavaScriptCore
- Internet Explorer – Chakra
- Microsoft EDGE – ChakraCore

# Тег script

- Нужен для встраивания JavaScript в html
- Указывается в любом месте документа
- С помощью атрибута src можно загрузить файл

# Стандартизация языка

- Разные движки (V8, SpiderMonkey, ..)
- Разные системы (Android, IOS, Windows, MacOS ..)
- Разные платформы (Браузеры, сервера, десктоп)

Разный JavaScript?

# Стандарт языка

Спецификация (стандарт, определение) языка программирования — это предмет документации, который определяет язык программирования, чтобы пользователи и разработчики языка могли согласовывать, что означают программы на данном языке.

© Wikipedia

# ECMAScript

- ECMAScript 5 (ES5) – 2009 год
- ECMAScript 6 (ES6, ES2015) – 2015 год
- ECMAScript 7 (ES7, ES2016) – 2016 год
- ...
- ECMAScript 2019



# Старый(Legasy) код

```
1  // es5
2  var a;
3
4  // es6+
5  let x;
6
```

# Комментарии

- Описывают код
- Полностью игнорируются движком JS
- Бывают двух видов

# Однострочные комментарии

```
1 // Создаём переменную по старому
2 var a;
3
4 let x; // Создаём переменную по новому
5
```

```
1 // Создаём переменную по старому
2 и назовём её a
3 var a;
4
5 // Создаём переменную по новому и назовём её x
6 let x;
7
```



# Многострочные комментарии

```
1  /*  
2  |   Создаём переменную по старому  
3  |   и назовём её a  
4  */  
5  var a;
```

```
/* Этот код не выполнится  
var a;  
let x;  
*/
```

# Переменные

# Переменные

- Хранят в себе данные
- Обязательно должны иметь имя
- Можно изменять значение при необходимости

# Создание переменной



# Создание переменной

```
let price; // price -> ?  
price = 100; // price -> 100
```

Создаём переменную без значения

В существующую переменную указываем значение

```
let price;  
price = 100;  
let discount = 20;  
price = price - discount;
```



# Старый стандарт

Вместо let используется var

```
var price;  
price = 100;  
var discount = 20;  
price = price - discount;
```

# Имя переменной

- Имя переменной должно содержать только буквы, цифры или символы \$ и \_ (не может содержать -, +, \* и т.д.)
- Первый символ не должен быть цифрой
- Нельзя использовать зарезервированные слова (let, var, const, function, return и т.д.)

```
let price;  
let price2;  
let _price$;
```



```
let price-1;  
let 2price;  
let var;
```



# CamelCase

- Все слова пишутся слитно, первое слово начинается с маленькой буквы, все остальные с большой

```
let priceWithDiscount = 80;
```



↑ ↑ ↑  
price with discount

```
let price_with_discount = 80;
```



# Регистрозависим

Переменные со строчными или прописными буквами являются разными.

```
let price = 100;  
let Price = 200;  
let PRice = 300;  
let PRICE = 400;
```

```
alert(price); // 100  
alert(Price); // 200  
alert(PRice); // 300  
alert(PRICE); // 400
```

# Константы

Переменная, которую нельзя изменить после создания.

```
let price;  
price = 100;  
const discount = 20;  
price = price - discount;
```

```
discount = 50; // ERROR
```

# Типы данных

# Типы данных

- Number
- String
- Boolean
- Null
- Undefined
- BigInt
- Symbol
- Object

# Number

Представляет как целые числа, так и числа с запятой

```
let a = 5; // number  
let x = 5.123; // number
```

Доступны математические операции:

```
let a = 5 + 2 // сумма: 7  
let b = 5 - 2 // разница: 3  
let c = 5 * 2 // произведение: 10  
let d = 5 / 2 // частное: 2.5  
let e = 5 % 2 // остаток от деления: 1
```



# «Специальные числовые значения»

Бесконечность:

```
Infinity; // бесконечность  
+Infinity; // положительная бесконечность  
-Infinity; // отрицательная бесконечность
```

Не число:

```
NaN; // Not a number - не число  
let price = 5 / "Не число"; // NaN  
let doublePrice = 2 * price; // Всё ещё NaN
```

# Совмещённые операторы

```
let price = 100;  
price = price + 20;
```

```
let price = 100;  
price += 20;
```

```
price = price + 20;  
price = price - 20;  
price = price * 20;  
price = price / 20;  
price = price % 20;
```

```
price += 20;  
price -= 20;  
price *= 20;  
price /= 20;  
price %= 20;
```

# Increment/Decrement

```
let count = 1;  
count++  
count // 2  
count--  
count // 1
```

Увеличиваем значение  
в переменной на 1

Уменьшаем на 1

# String - строка

- Строки используются для работы с текстом
- Должны быть заключены в кавычки

Одинарные кавычки

```
let name = 'Строки используются для работы текстом';
```

Двойные кавычки

```
let name = "Строки используются для работы текстом";
```

Обратные кавычки

```
let name = `Строки используются для работы текстом`;
```

# Обратные кавычки

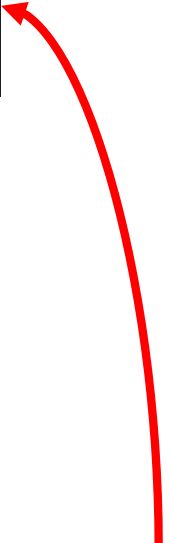
- Можно «встраивать» выражения

```
`3 плюс 2 равно ${3 + 2}` // 3 плюс 2 равно 5
```

Пишем `${}`, а между  
скобок выражение



Выражение выполнилось и  
в строку попал результат

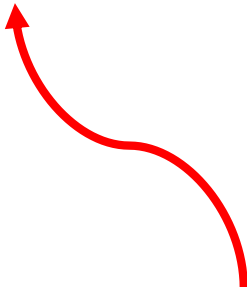


# Обратные кавычки

```
const price = 100;  
const summary = `Итого: ${price} руб. в месяц`; // Итого: 100 руб. в месяц
```

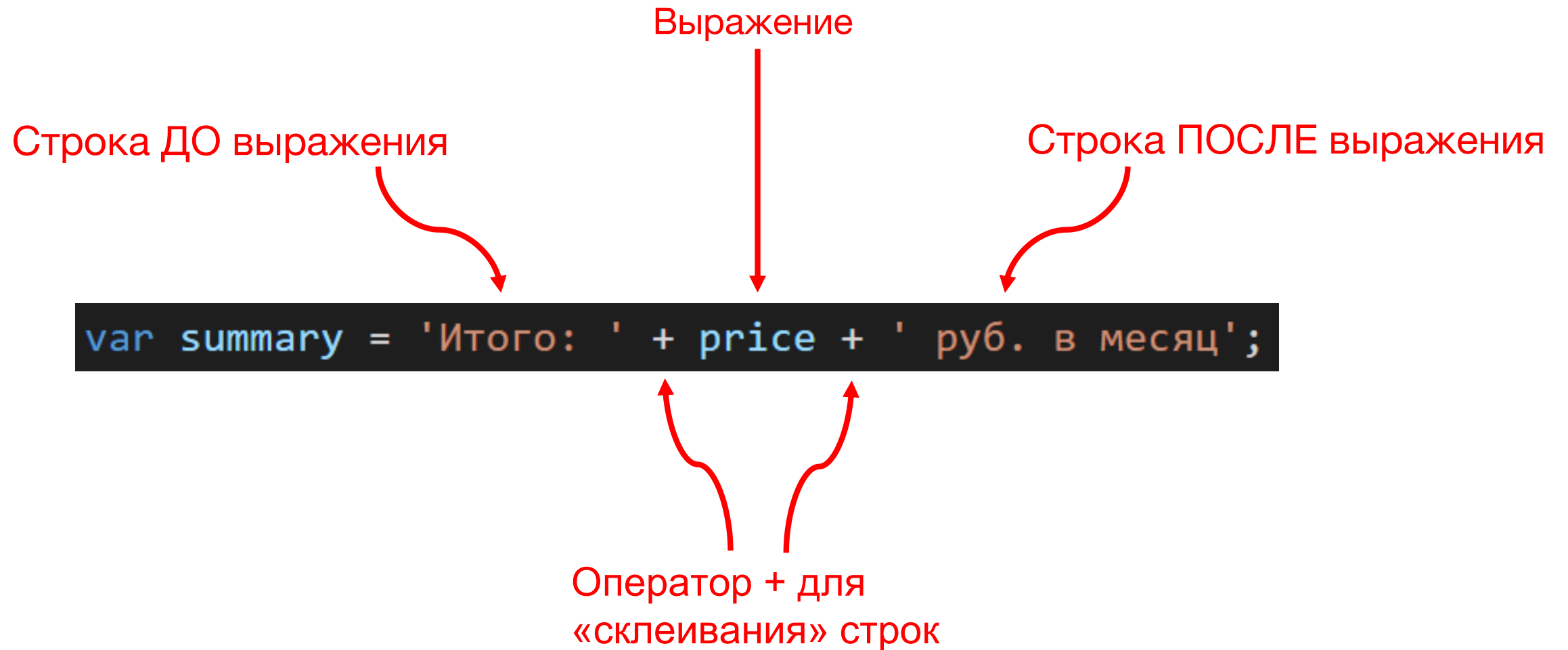


Вставляем переменную



В строку попало  
значение переменной

# А вот раньше... (ES5)



# Boolean – булевый/логический тип

- Может принимать только два значения
  - true – истина
  - false – ложь
- Используется для хранения значений да или нет

```
const isJavaScript = true; // да, истина  
const isJava = false; // нет, ложь
```

- Может быть результатом сравнения

```
const isBiggerThenZero = 100 > 0; // true
```



# Null

- Только одно значение null

```
let discount = null;
```

# Undefined

- Только одно значение undefined
- Означает «значение не было присвоено»

```
let price; // price -> undefined  
price = 100; // price -> 100  
price = undefined; // price -> undefined
```

# Object - объект

- Не является примитивным
- Позволяют хранить коллекции данных или более сложные объекты.

# Object - об'єкт

```
const emptyObj = {};  
const obj = {  
  key: 'value',  
  'other-key': 42,  
  innerObj: {  
    innerKey: 53,  
  },  
};
```

```
obj.key; // value  
obj.innerObj.innerKey // 53  
obj['other-key']; // 42
```

```
obj.key = 'Bee';  
obj.innerObj.innerKey = null;  
obj['other-key'] = 40;
```

# BigInt

- Может работать с гигантскими числами в отличие от number
- Работает только с целыми числами

```
const bigInt = 1234567890123456789012345678901234567890n;
```

# Symbol

- Используется для создания уникальных идентификаторов объектов
- Используется в библиотеках и фреймворках

# Типизация в JavaScript

- JavaScript — это динамически типизированный язык.

JavaScript

```
let number = 42;  
number = 'строка';
```



C#

```
int number = 45;  
number = "строка";
```



# Преобразование типов данных



# Преобразование типов данных

- Процесс конвертации значения из одного типа данных в другой (строку в число, число в логический и т.д.)

# Из числа в строку

Создали переменную с  
числом

Передали переменную с  
числом

```
const price = 100;  
const summary = `Итого: ${price} руб. в месяц`;  
'Итого: 100 руб. в месяц'
```

Превратилось в  
строку

# Из числа в строку

Неявное/автоматическое преобразование

```
let numberString = `${123}`; // '123'
```

Явное преобразование

```
let numberString = String(123); // '123'
```

# Boolean/Null/Undefined в строку

```
`${true}`; // 'true'  
`${false}`; // 'false'  
`${null}`; // 'null'  
`${undefined}`; // 'undefined'  
  
String(true); // 'true'  
String(false); // 'false'  
String(null); // 'null'  
String(undefined); // 'undefined'
```

# Object в строку

По умолчанию:

```
let obj = {  
  key: 'value',  
};  
  
`${obj}`; // '[object Object]'  
String(obj); // '[object Object]'
```

# Численное преобразование / Преобразование в число

В математических операциях

```
const num = '100' / '2'; // 50
```



```
const num = 100 / 2; // 50
```

# Численное преобразование / Преобразование в число

```
'100' / '2' // 50  
'100' / 2 // 50  
'100' / 'ДВА' // NaN  
'100' * '2' // 200  
'100' - '2' // 98  
'101' % '2' // 1
```

```
'100' + '2' // '1002'  
100 + '2' // '1002'
```

Думает, что надо  
«склеивать» строки

# Явное преобразование в число

```
Number('100') // 100  
Number('Не число') // NaN
```



# Остальные типы в число

```
Number(false) // 0
100 * false // 100 * 0 = 0

Number(true) // 1
100 * true // 100 * 1 = 100

Number(null) // 0
100 * null // 100 * 0 = 0

Number(undefined) // NaN
100 * undefined // 100 * NaN = NaN

Number({}) // NaN
100 * {} // 100 * NaN = NaN
```

# К логическому типу

```
Boolean(0) // false  
Boolean('') // false  
Boolean(NaN) // false  
Boolean(null) // false  
Boolean(undefined) // false
```

```
Boolean(1) // true  
Boolean('123') // true  
Boolean('0') // true  
Boolean({ key: 'value' }) // true  
Boolean({}) // true
```

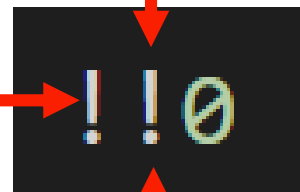
# Неявное преобразование к логическому типу

```
!!0 // false  
!!1 // true
```

# Неявное преобразование к логическому типу

Преобразует 0 в  
логический тип (false)

Второй оператор снова  
меняет значение (из true в  
false)



Полученное значение (false)  
меняется на  
противоположное (true)

# Ссылки

- var vs let
  - <https://medium.com/nuances-of-programming/%D0%B2-%D1%87%D1%91%D0%BC-%D1%80%D0%B0%D0%B7%D0%BD%D0%B8%D1%86%D0%B0-%D0%BC%D0%B5%D0%B6%D0%B4%D1%83-var-let-%D0%B8-const-%D0%B2-javascript-3084bfe9f7a3>
- Зарезервированные слова
  - [https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Lexical\\_grammar#%D0%9A%D0%BB%D1%8E%D1%87%D0%B5%D0%B2%D1%8B%D0%B5%D1%81%D0%BB%D0%BE%D0%B2%D0%B0](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Lexical_grammar#%D0%9A%D0%BB%D1%8E%D1%87%D0%B5%D0%B2%D1%8B%D0%B5%D1%81%D0%BB%D0%BE%D0%B2%D0%B0)
- BigInt
  - <https://learn.javascript.ru/bigint>
  - [https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global\\_Objects/BigInt](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/BigInt)

# Ссылки

- Symbol
  - <https://learn.javascript.ru/symbol>
  - [https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global\\_Objects/Symbol](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Symbol)
- <https://developer.mozilla.org/ru/docs/Web/JavaScript> - Справочник по JavaScript
- <https://learn.javascript.ru/> - Учебник по JS
- Про примитивные значения
  - [https://developer.mozilla.org/ru/docs/Web/JavaScript/Data\\_structures](https://developer.mozilla.org/ru/docs/Web/JavaScript/Data_structures) - пункт «Примитивные значения»