



Лекция 1

Знакомимся

Цели на сегодня

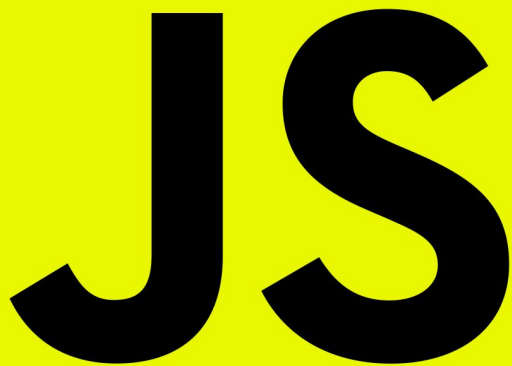
- Понять, для чего нужен JavaScript;
- Узнать, как записывать данные в память;
- Разобраться со всеми примитивными типами данных;
- Пробежаться по арифметическим операторам;
- Немного поговорить про типизацию.

Подключение скриптов в HTML

Для подключения скриптов используется тег `<script>`, который может находиться в `<head>` или перед `</body>`. Очередность подключения скриптов имеет значение, поскольку браузер парсит код построчно.

Тег **`<script>`** имеет следующие важные атрибуты:

- **src** – позволяет указать источник подключения скрипта;
- **async** – позволяет загружать скрипт в фоновом режиме без блокировки загрузки остального контента. Скрипт выполнится, как только будет загружен;
- **defer** – работает так же, но скрипт будет выполнен после полной загрузки страницы и, к тому же, сохраняется очередность выполнения.

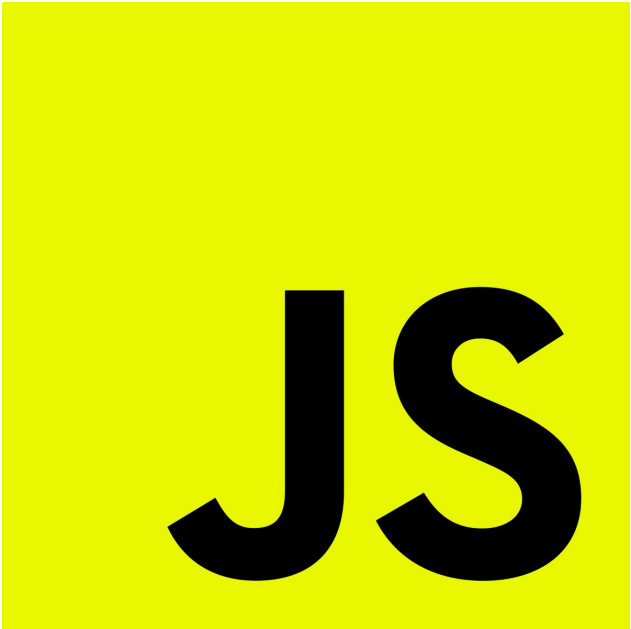
A large, bold, black 'JS' logo is centered on a bright yellow rectangular background. The letters are in a sans-serif font, with the 'J' and 'S' being slightly larger than the 'S'.

JavaScript –

скриптовый язык программирования с динамической типизацией и автоматическим управлением памятью.

ECMAScript –

стандарт языка JavaScript.



JS

!=



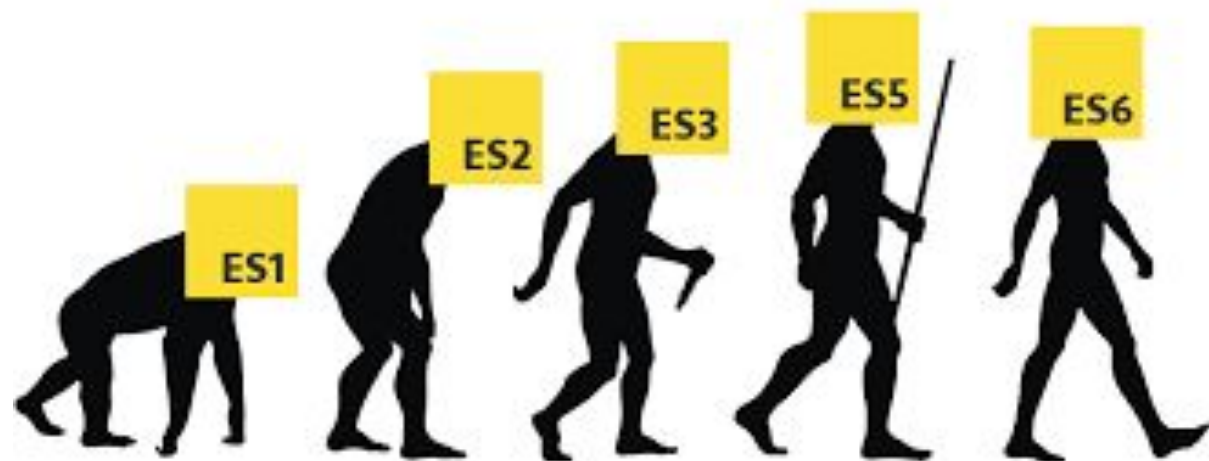
1997

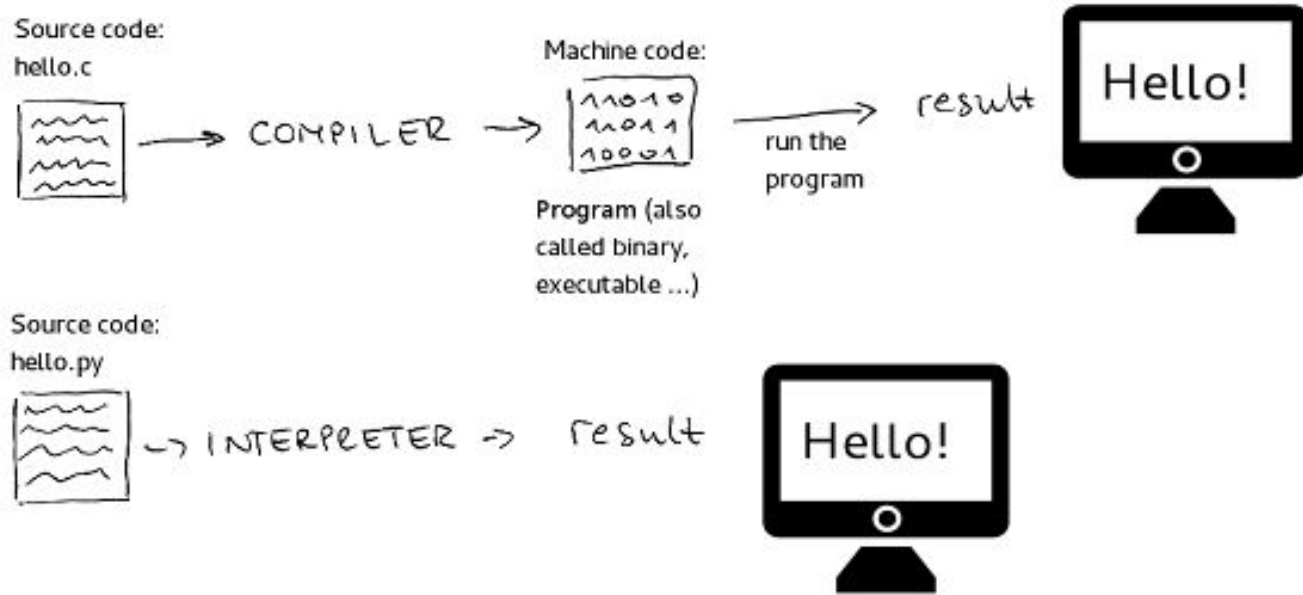
1998

1999

2009

2015





Компилируемые: C, C++, Go, Haskell, Rust;

Интерпретируемые: JavaScript, PHP, Perl, Ruby, Python;

ПРОГРАММЕРСКАЯ ОТМАЗКА #1 ДЛЯ
ЗАКОННОГО ОТЛЫНИВАНИЯ ОТ РАБОТЫ:

«МОЙ КОД КОМПИЛИРУЕТСЯ».

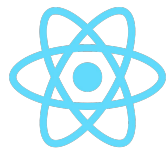


Где раньше использовался JavaScript?

- Создание всплывающих окон (попапов);
- Создание анимаций и переходов;
- Создание галерей и слайдеров;
- Создание мобильных меню;
- Работа с сервером;
- Управление HTML и CSS;

В браузере

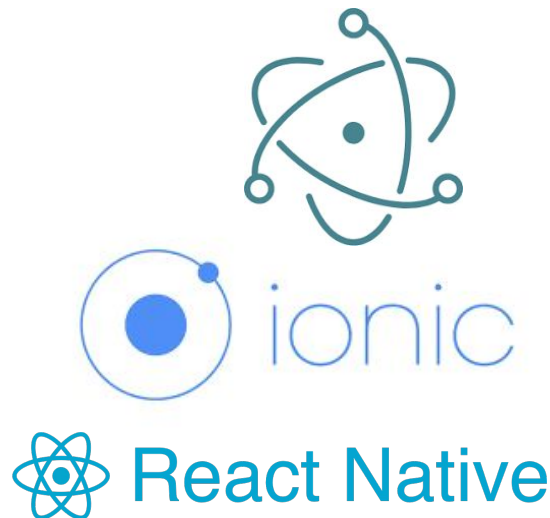
Где сейчас используется JavaScript?

The logo for ES6 (ECMAScript 2015) is a yellow square with the text "ES6" in black.

В браузере



На сервере



**На телефоне и
десктопе**

Переменные (Variables) –

это способ записать значение для дальнейшего использования. Тут подходит ассоциация с ящиком или коробкой, в которую мы положим то, что хотим использовать позже.



```
/*
```

Задавать переменные можно используя var, let и const.

var – устаревший способ, его на этом курсе использовать нельзя!

```
*/
```

```
var someNumber = 1;
```

```
var someString = '0 Captain! my Captain! our fearful trip is done';
```

// Переменную, заданную через var, легко переопределить:

```
var number = 0;
```

```
console.log(number); // 0
```

```
var number = 1;
```

```
console.log(number); // 1
```

ES6

```
// Предпочтительно задавать переменные через const и let.  
  
let someNumber = 1;  
const someString = '0 Captain! my Captain! our fearful trip is done';  
  
// От var они отличаются тем, что их нельзя прямо переопределить:  
  
let thing = 1; // 1  
let thing = 2; // SyntaxError: redeclaration of let thing
```



Ключевое слово

ES6



// Однако, переменную, заданную через let, можно переопределить иначе:

```
let thing = 1; // 1  
thing = 2; // 2
```

// С const так не выйдет:

```
const thing = 1; // 1  
thing = 2; // TypeError: invalid assignment to const 'thing'
```

Это не все особенности var, let и const, однако нам этого пока что достаточно

Основное правило выбора ключевого слова –

используем `const` во всех случаях, кроме тех, когда точно известно, что значение переменной будет изменяться (например в циклах).

Типы данных (Data Types)

1. Прimitives

- a. Числа (Numbers);
- b. Строки (Strings);
- c. Логические (Boolean)
- d. Символ (Symbol);
- e. null;
- f. undefined.

2. Сложные

- a. Объекты (Objects);
- b. Массивы (Arrays);
- c. Функции (Functions).

Числа (Numbers)

Числа (Numbers)

1. Целые (integer):

25

1000

25000

0x00

0x10

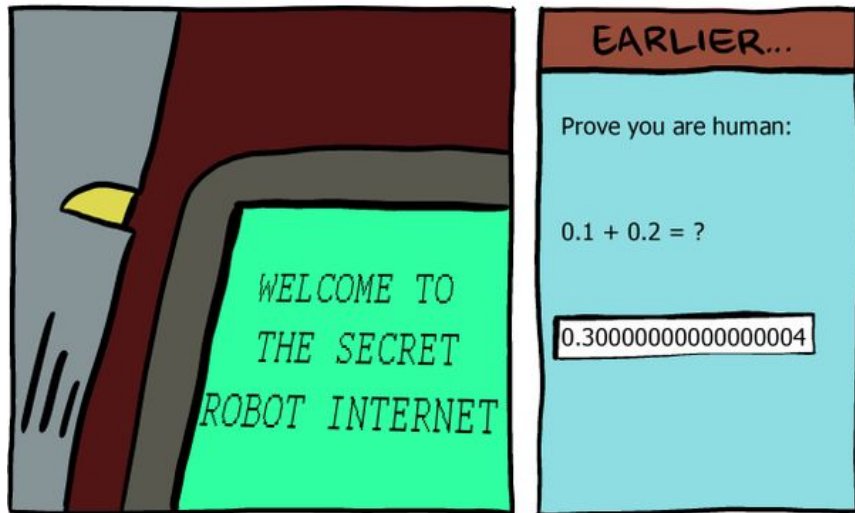
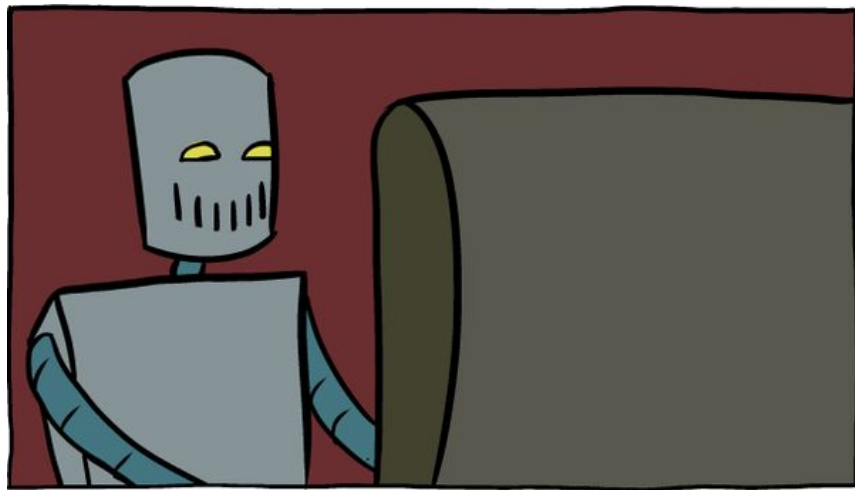
2. Дробные (floating points):

3.14

0.5

0.333333

Для написания дробных
чисел используется точка,
а не запятая



Стоит учитывать, что из-за особенностей преобразования чисел с плавающей точкой, операции над дробными числами могут возвращать непредсказуемый результат.

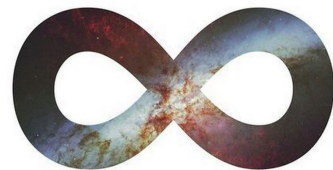
isFinite() может использоваться для проверки на бесконечность. Возвращает true или false

Бесконечность (Infinity) –

получается при попытке деления на ноль или работе с очень большими или очень маленькими числами.

```
console.log(1 / 0) // Infinity
```

```
console.log(-1 / 0) // -Infinity
```




/*

NaN (Not a Number) – значение, которое появляется, когда мы пытаемся произвести арифметическую операцию со значением, которое нельзя превести к числовому типу

*/

```
console.log('Maximumstart' * 2); // NaN  
console.log('Maximumstart' - 2); // NaN  
console.log('Maximumstart' / 2); // NaN  
console.log('Maximumstart' + 2); // Maximumstart2
```

```
console.log(0 / 0); // NaN  
console.log(Infinity / -Infinity); // NaN  
console.log(undefined + 2); // NaN
```

Строки (Strings)

В домашних заданиях и выпускных проектах нужно будет выбрать между одинарными и двойными

"Строка" –

в JavaScript это все, что написано в кавычках. Использование двойных или одинарных кавычек – дело вкуса.



```
/*
```

В JavaScript любые текстовые данные являются строками.
Можно использовать одинарные (' '), двойные (" ") и косые (` `) кавычки.

Одинарные и двойные кавычки абсолютно равноценны, при использовании косых появляются дополнительные плюшки.

```
*/
```

```
const firstPart = "Все равно любовь моя — тяжкая гиря ведь — ";
```

```
const secondPart = 'висит на тебе, куда ни бежала б.';
```

```
const phrase = firstPart + secondPart;
```

```
// "Все равно любовь моя — тяжкая гиря ведь — висит на тебе, куда ни бежала б."
```

```
const fakeBoolean = 'true';
```

```
const date = '20.03.2018';
```

```
const pi = '3.14';
```




// Сложение строк:

```
const date = "20.03.2018";  
console.log("Указанная дата:" + " " + date + "."); // Указанная дата: 20.03.2018.
```

// При использовании одинарных или двойных строк, переносы не учитываются:

```
const poetry = 'O Captain! my Captain! our fearful trip is done,  
The ship has weather'd every rack, the prize we sought is won,  
The port is near, the bells I hear, the people all exulting,';
```

// Раньше перенести строку можно было только с помощью управляющего символа:

```
const poetry = 'O Captain! my Captain! our fearful trip is done, \n' +  
'The ship has weather\'d every rack, the prize we sought is won, \n' +  
'The port is near, the bells I hear, the people all exulting,';
```

ES6

```
/*
  Интерполяция позволяет использовать переменные или выражения в
  строке и легко переносить строку
*/

const date = "20.03.2018";
const stringWithDate = `У вас запланирована встреча на ${date}`;

const value = `Ваше значение: ${27 / 3}`;

// Все поломалось :с
const foo = `0 Captain! my Captain! our fearful trip is done,
The ship has weather'd every rack, the prize we sought is won,
The port is near, the bells I hear, the people all exulting,`;

// А так все хорошо с:
const bar = `0 Captain! my Captain! our fearful trip is done,
The ship has weather'd every rack, the prize we sought is won,
The port is near, the bells I hear, the people all exulting,`
```

Логический (булев) тип данных (Boolean) –

тип данных, принимающий два возможных значения – **true** или **false**. Используется повсеместно, поскольку на его основе легко строить логику приложения.

Специальные значения `null` и `undefined`

`null` –

означает отсутствие какого-то значения, «ничего»;

`undefined` –

указывает на то, что значение не задано. Часто встречается при оглашении переменной без значения или обращении к несуществующей переменной.

Как узнать тип данных?

Оператор

```
// typeof – правильный способ узнать тип данных

typeof 2 // "number"
typeof '2'; // "string"

typeof 'true'; // "string"
typeof true; // "boolean"

typeof Symbol("name"); // "symbol"

typeof undefined; // "undefined"
typeof NaN; // "number"
typeof Infinity // ?

typeof null; // "object"
```

Математические операторы

Математические операторы

Для работы с переменными, со значениями, JavaScript поддерживает все стандартные операторы, большинство которых есть и в других языках программирования.

Унарным называется оператор, который применяется к одному операнду. Например, оператор унарный минус "-" меняет знак числа на противоположный.

Бинарным называется оператор, который применяется к двум операндам ("+", "-", "*", "/").

Тернарным называется оператор, который применяется к трем операндам (только оператор "?").



```
// Математические операторы
```

```
console.log(1 + 2); // 3  
console.log(2 - 3); // -1  
console.log(4 / 2); // 2  
console.log(10 * 10); // 100  
console.log(8 % 2); // 0
```

```
// Дополнительные операторы и константы есть в объекте Math
```

```
Math.sqrt(25); // 5  
Math.PI; // 3.141592653589793  
Math.E; // 2.718281828459045  
Math.pow(2, 4); // 16
```



```
/*
```

Часто нужно применить оператор к переменной и сохранить результат в ней же, для этого идеально подходят операторы сокращенной арифметики

```
*/
```

```
let x = 10;
```

```
x = x + 5;
```

```
x = x * 2;
```

// Эту запись можно сделать компактней:

```
let x = 10;
```

```
x += 5;
```

```
x *= 2;
```

Операторы сравнения



// Операторы сравнения возвращают булевы значения

```
console.log(10 > 5); // true
```

```
console.log(2 >= 2); // true
```

```
console.log(5 < 10); // true
```

```
console.log(5 <= 5); // true
```

// Оператор сравнения, который сравнивает значения, но

// не сравнивает типы данных

```
console.log(3 == '3'); // true
```

// Оператор сравнения, который сравнивает значения и типы данных

```
console.log(3 === '3'); // false
```

// Нестрогий оператор "не равно"

```
console.log(3 != '3'); // false
```

// Строгое "не равно"

```
console.log(3 !== '3'); // true
```

```
/*
    Проверка через оператор сравнения не работает
    NaN === NaN // false
    "someNotANumberText" === NaN // false
    Поэтому используем специальную функцию isNaN
*/

isNaN(2); // false
isNaN('2'); // false, потому что приводится к числовому типу

isNaN(true); // false, поскольку true = 1
isNaN(false); // false, поскольку false = 0
isNaN(null); // false
isNaN(undefined); // true

isNaN('Maximumstart'); // true

isNaN(NaN); // true
isNaN(Infinity); // ?
```

Функция

Преобразование типов для примитивов

<code>true</code> (Boolean)	->	1 (Number)
<code>false</code> (Boolean)	->	0 (Number)
<code>+"2"</code> / <code>Number("2")</code> (String)	->	2 (Number)

Контрольные вопросы

1. Как определить, четное число или нечетное?
2. Как получить бесконечность?
3. Что такое «динамическая типизация»?
4. Как объединить две строки в предложение?
5. Почему сравнение:

`100 === "100"`

Вернет `false`?

Самостоятельная работа

Написать две версии программы-калькулятора, использующей интерполяцию и конкатенацию:



```
const number = 2; // Тут подставляются разные числа
const result = Тут_происходит_магия;
console.log(result); // 'Результат сложения числа 2 и 100 = 102';
```



```
const number = 10; // Тут подставляются разные числа
const result = Тут_происходит_магия;
console.log(result); // 'Результат сложения числа 10 и 100 = 110';
```