JavaScript

Основы WEB-программирования

PROWEB

Программирование

Почему язык имеет значение

В начале, при зарождении **компьютерных** дисциплин, **не было** языков программирования. Программы выглядели так:

```
00110001 00000000 00000000

00110001 00000001 00000001
00110011 00000001 00000010
01010001 00001011 00000010
00100010 00000010 00001000
01000011 00000001 00000000
01000001 00000001 00000001
00010000 00000010 00000000
01100010 00000000 00000000
```

Почему язык имеет значение

Вот та же программа на JavaScript:

```
var  total = 0, count = 1;
for(count; count < 10;) {
    total += count;
    count++;
}
console.log(total);
// результат= 45
```

Что такое JavaScript?

JavaScript (часто просто **JS**) является **языком программирования**.

JavaScript был представлен в 1995 году как способ добавлять **программы** на веб-страницы в браузере **Netscape Navigator**

Ja vaScr ipt на Википедии

JavaScript не Java

Не следует путать **JavaScript** с языком программирования **Java**. И "**Java**", и "**JavaScript**" являются торговыми марками или зарегистрированными торговыми марками **Oracle** в США и других странах. Однако, у обоих языков различный синтаксис, семантика и применение.

Язык сценариев

JavaScript наиболее широкое применяется как язык **сценариев web-страниц**, но также используется и в других программных продуктах.

Интерпретация

```
var  total = 0, count = 1;
for(count; count < 10;) {
    total +=
    count;
    count++;
}
console.log(total);
// результат= 45
```



```
00110001 00000000 00000000

00110001 00000001 00000001
00110011 00000001 00000010
01010001 00001011 00000010
00100010 00000010 00001000
01000011 00000001 00000000
01000001 00000001 00000001
00010000 00000010 00000000
01100010 00000000 00000000
```

Интерпретация

Компиляция / Интерпретация

Компиляторы и **интерпретаторы** преобразуют исходный код в машинный код, только разными путями.

Компиляция

Компиляция — это когда исходный код программы, при помощи специального инструмента, другой программы, которая называется **«компилятор»**, преобразуется в другой язык, как правило — в машинный код. Этот машинный код затем распространяется и запускается.

Недостатки компиляции:

1. Программа имеет зависимость от **ОС**.
2. Сложность отладки кода программы.

Интерпретация

Интерпретация — это когда исходный код программы получает другой инструмент, который называют **«интерпретатор»**, и выполняет его **«как есть»**. При этом распространяется именно сам исходный код (скрипт).

Недостатки интерпретации:

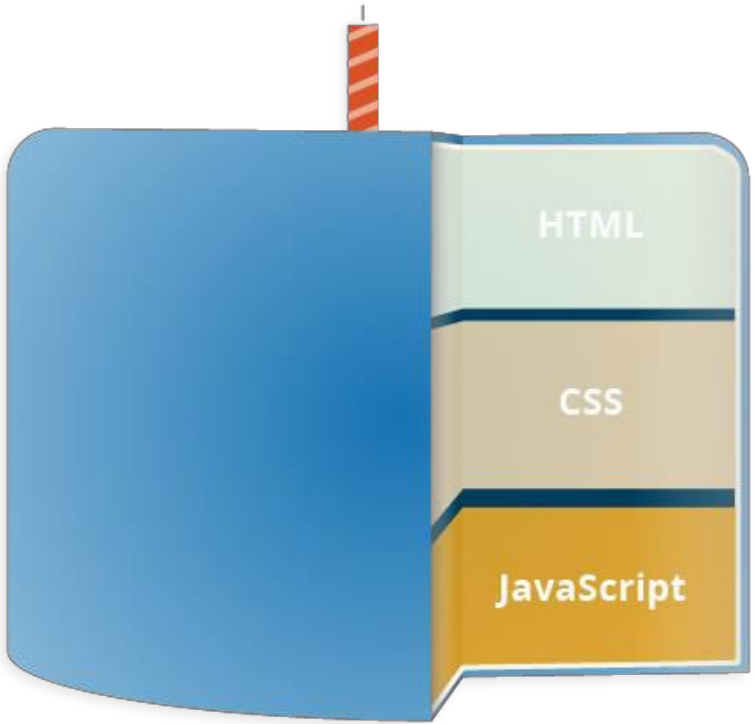
1. Для запуска программы требуется наличие **интерпретатора**.
2. Заметно ниже скорость работы.

JavaScript - **интерпретируемый** язык программирования

Интерпретация

Во все основные браузеры встроен **интерпретатор** JavaScript, именно поэтому они могут выполнять скрипты на странице.

Слоёный пирог



Слоёный пирог

- **HTML** - это **язык разметки**, который мы используем для визуального и смыслового структурирования нашего web контента.

Слоёный пирог

- **HTML** - это **язык разметки**, который мы используем для визуального и смыслового структурирования нашего web контента.
- **CSS** - это **язык стилей** с помощью которого мы придаем стиль отображения нашего HTML контента.

Слоёный пирог

- **HTML** - это **язык разметки**, который мы используем для визуального и смыслового структурирования нашего web контента.
- **CSS** - это **язык стилей** с помощью которого мы придаем стиль отображения нашего HTML контента.
- **JavaScript - язык программирования**, который позволяет Вам создать динамически обновляемый контент, управляет мультимедиа, анимирует изображения и т.д.

Слоёный пирог

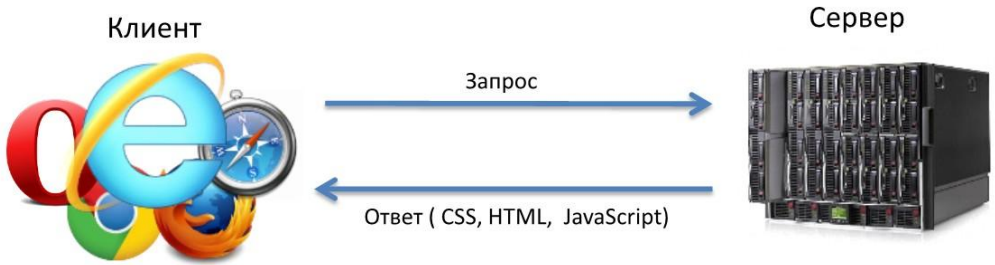
JavaScript это язык, который позволяет Вам применять сложные вещи на **web-странице** — каждый раз, когда на **web-страниц** происходит что-то большее, чем просто её статичное отображение.

— можете быть уверены, что скорее всего, не обошлось без **JavaScript**.

JavaScript — это третий слой слоёного пирога стандартных web технологий, два из которых (**HTML** и **CSS**)

Где применяется JavaScript?

- **WEB-сайты**
- Мобильные приложения
- Десктопные программы
- Сервер и вспомогательные инструменты



Возможности

- Добавлять различные **эффекты анимации**
- Реагировать на **события** - обрабатывать перемещения указателя мыши, нажатие клавиш с клавиатуры
- Осуществлять **проверку ввода данных** в поля формы до отправки на сервер, что в свою очередь снимает дополнительную нагрузку с сервера
- Определять браузер
- Изменять содержимое **HTML-элементов**, добавлять новые теги, изменять стили

Типы данных


ЭКОВОЗ
ПЛАСТИК
PLASTIC
ЭКОВОЗ
СТЕКЛО
GLASS
ЭКОВОЗ
БУМАГА
PAPER
ЭКОВОЗ
ТБО
SOLIDWASTE

Типы данных

В JavaScript типы данных можно разделить на две категории:

- **простые** (их также называют **примитивными**) типы
- **составные** (их также называют **ссылочными** или **объекты**)

Типы данных

К категории простых типов относятся:

- **string** - текстовые строки (обычно их называют просто - строки)
- **number** - числа
- **boolean** - логические (булевы) значения

Так-же к простым типам относятся два специальных значения:

- **null – Литерал (**запись в исходном коде компьютерной программы, представляющая собой фиксированное значение**)**
- **undefined (**не определено**)**

Строки (string)

- Записываются внутри одинарных или двойных кавычек

```
‘Hello, World’;
“Hello, World”;
```

Числа (number)

- Записываются без дополнительных символов
- Поддерживается работа с целыми и дробными числами, дробная часть отделяется **точкой**
- существует два дополнительных значения: `Infinity` и `NaN`
- `Infinity(Бесконечность) – Числовой тип данных`
- `NaN(Not a number) – Числовой тип данных`

Типы данных

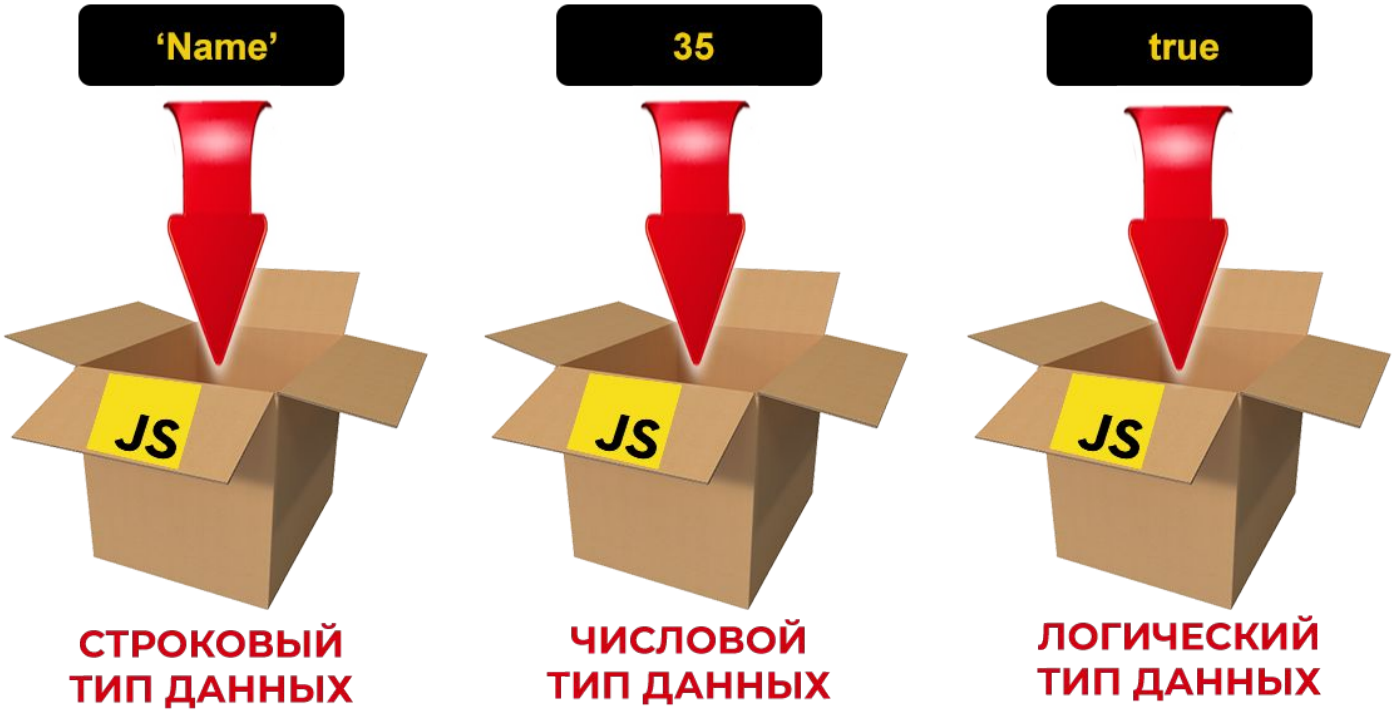
```
var a = 10;          // тип Number
var b = 0.8;         // тип Number

var c = "string";    // тип String
var d = 'string';    // тип String

var e = true;        // тип Boolean
var f = false;       // тип Boolean

var g = null;        // тип null
var h = undefined;   // тип undefined
```

Переменная



Переменная

Когда в программе необходимо **сохранить** значение, чтобы использовать его позже, это значение присваивается **переменной**. **Переменная** – это просто символьное имя для значения, которое обеспечивает возможность получить значение по имени, то есть, когда в программе указывается имя переменной вместо неё подставляется значение.

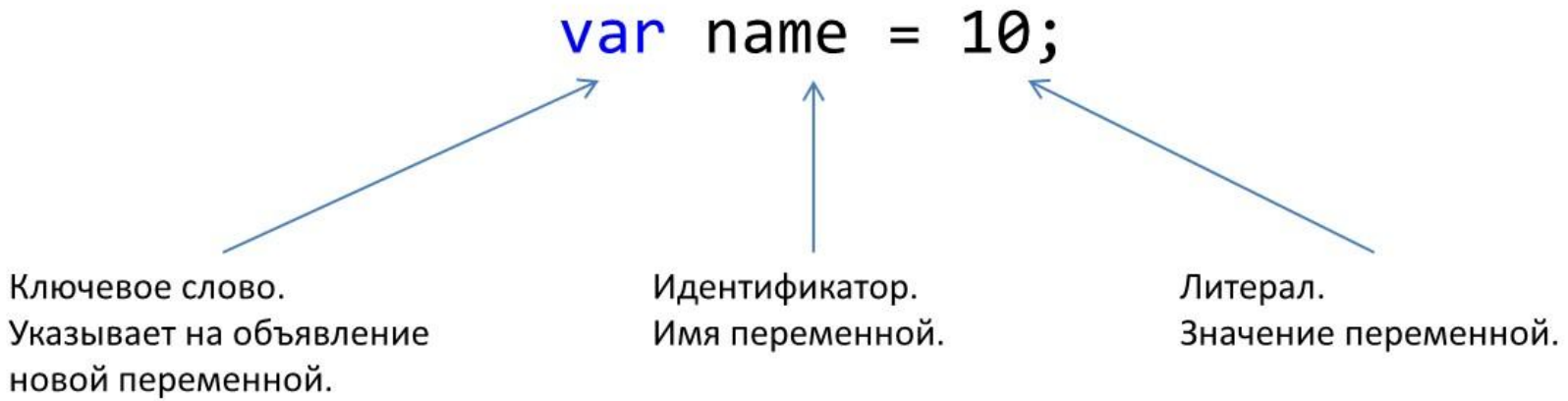
Объявление переменной

Прежде чем использовать **переменную**, её необходимо объявить. Переменные объявляются с помощью ключевого слова **`var`**, за которым следует имя переменной.

Один раз использовав ключевое слово **`var`**, можно объявить несколько переменных, перечислив их через запятую:

```
var num, num2;
```

Объявление переменной



Присваивания значения

Присвоить какое-либо значение переменной можно с помощью **оператора присваивания**, который обозначается символом равно **=**

```
var color = "чёрный";
```

Зачем нужны переменные?

Переменные помогают сделать программный код **понятнее**.
Рассмотрим небольшой пример:

Зачем нужны переменные?

```
totalPrice = 2.42 + 4.33; // Общая цена
```

Имеющиеся здесь числа могут означать что угодно. Чтобы стало понятно, что именно здесь суммируется, значение **2.42** можно присвоить переменной **`candyPrice`** (цена конфет), а **4.33** – переменной **`oilPrice`** (цена масла):

```
totalPrice = candyPrice + oilPrice;
```

Вместо того, чтобы вспоминать, что эти значения означают, можно увидеть, что в сценарии складывается цена конфет с ценой масла.

Как называть переменные (Индификаторы)?

Идентификатор должен быть одним словом:

 myName

 My Name

Имя не может начинаться с цифры:

 element21

 21element

Первым символом имени может быть символ ‘$’ или ‘_’ :

 $str

 _name

Как называть переменные (Индификаторы)?

В JavaScript наиболее популярным стилем именования идентификаторов, состоящих из нескольких слов, является **camelCase** – "верблюжья" нотация (нотация – это устоявшиеся правила записи). Согласно этому стилю идентификаторы, которые состоят из одного слова, пишутся строчными буквами:

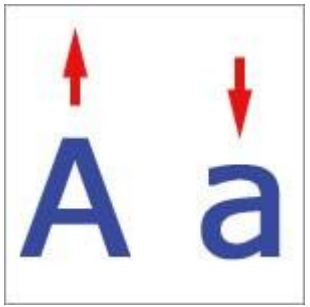
```
var color = "black";
var number = 7;
```

Если идентификатор состоит более, чем из одного слова, то первое слово пишется строчными буквами, а каждое последующее слово начинается с прописной (заглавной) буквы:

```
var myAge = 10;
var firstName = "Bilbo";
```

Регистр

JavaScript **чувствителен к регистру** символов. Это значит, что ключевые слова и любые другие идентификаторы, используемые в программе, всегда должны содержать одинаковый набор прописных и строчных букв. Например, ключевое слово **switch** должно быть написано как **`switch`**, а не **`Switch`** или **`SWITCH`**



Операторы

Оператор – это символ(ы) или ключевое слово, благодаря которому производятся некоторые виды вычислений с участием одного или нескольких значений. Значения, располагающиеся слева и справа от оператора, называются **операндами**.

Операторы

- Оператор с одним операндом называется **унарным**,
- с двумя – **бинарным**,
- с тремя – **тернарным**.

Виды операторов

Операторы

- Арифметические операторы (`+, -, *, /, %, ++, --`)
- Строковые оператор склеивания (`+`)
- Логические операторы (`&&, ||, !`)
- Операторы сравнения (`==, !=, ===, !==, >, >=, <, <=`)
- Операторы булевой логики (`true, false`)