Введение

ECMAScript

То есть ECMAScript — это язык, для которого описан и опубликован стандарт синтаксиса, типов данных, блоков, функций и прочих особенностей. JavaScript при этом — это другой язык, реализующий стандарт ECMAScript.

В 2015 вышел обновленный стандарт «ES6», расширивший синтаксические возможности языка. ES6 (2015) поддерживаеся всеми популярными браузерами.

Выражения

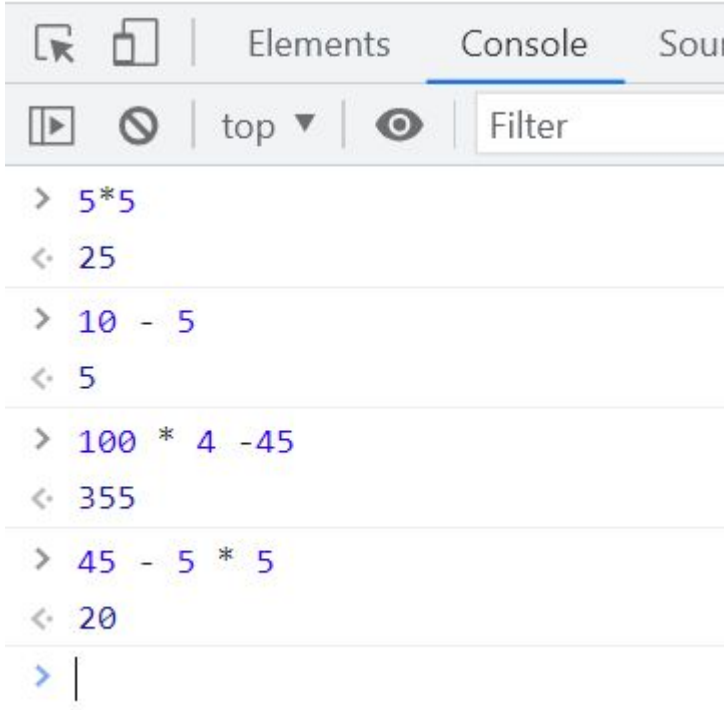
Выражение – это тоже одно из ключевых понятий в JavaScript. Оно представляет собой некоторый кусок кода, который всегда **возвращает значение**.

- `'Alexander'` – это выражение, которое представляет собой строку; результатом этого выражения будет эта же строка;`//'Alexander'`
- `7 + 3` – это выражение, в котором используется оператор `+`; результатом этого выражения будет число `10`; `//10`
- `'Alexander' + ' ' + 'Maltsev'` – результатом этого выражения будет новая строка `'Alexander Maltsev'`; // `'Alexander Maltsev'`

Простейшие вычисления в консоли

Поддерживаются следующие математические операторы:

- Сложение `+ (2+3)`,
- Вычитание `- (5-2)`,
- Умножение `* (5*4)`,
- Деление `/ (10/2 =5.0)`,
- Взятие остатка от деления `% (5 % 2 =1)`,
- Возведение в степень `** (2**3=8 , 4 **(1/2))`.



> - ввод пользователя
< - вывод результата

Переменные

Для хранения данных в программе используются **переменные**.

Переменные предназначены для хранения каких-нибудь временных данных или таких данных, которые в процессе работы могут менять свое значение.

Переменные

Каждая переменная имеет **имя.**

Имя представляет собой произвольный набор алфавитно-цифровых символов, знака подчеркивания (_) или знака доллара ($), причем названия не должны начинаться с цифровых символов. То есть мы можем использовать в названии буквы, цифры, подчеркивание. Однако все остальные символы запрещены. Например, правильные названия переменных:

```
$commision
someVariable
product_Store
income2
myIncome_from_deposit
```

Синтаксис

JavaScript относится к регистрозависимым языкам. Это разные имена:

world

World

WORLD

Стили написания Camel case :

```
let worldSpace, getSize, setSize;
```

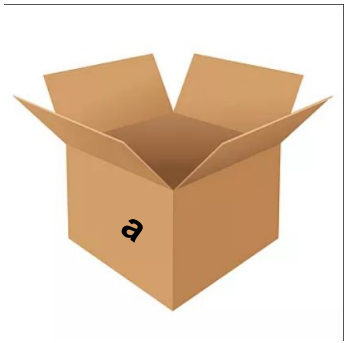
Переменные

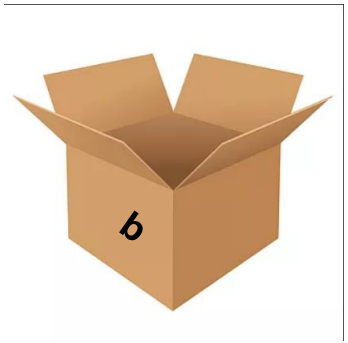
Для создания переменной в JavaScript используйте ключевое слово `let`.

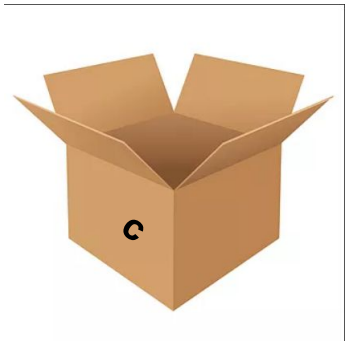
```
let Name;
```

```
let a, b, c;
```

```
let a; //ошибка!
```





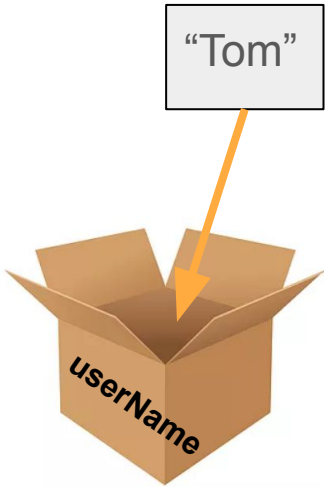


Создание переменных

После определения переменной ей можно присвоить какое-либо значение. Для этого применяется **оператор присваивания** (**=**):

```
let userName;

userName = "Tom";
```

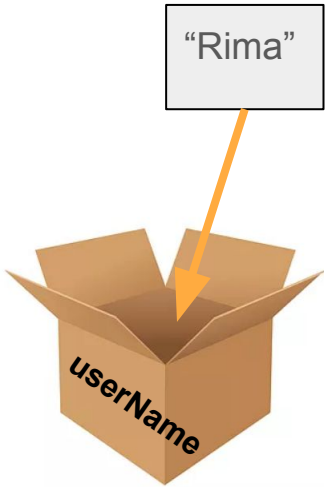
```
> let userName;

< undefined

> userName = "Tom";
< 'Tom'

> userName
< 'Tom'

>
```



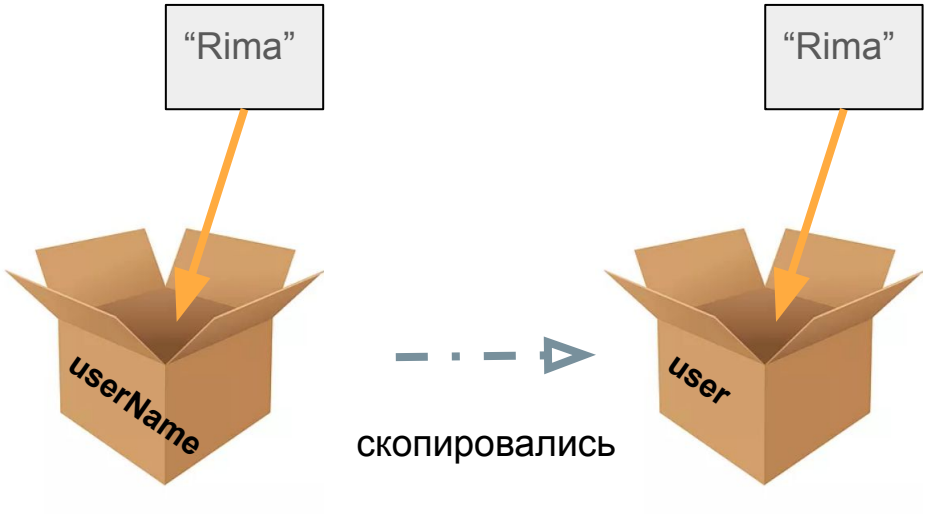
Строка сохраняется в области памяти, связанной с переменной. Мы можем получить к ней доступ, используя имя переменной.

То есть в данном случае переменная userName будет хранить строку "Tom".

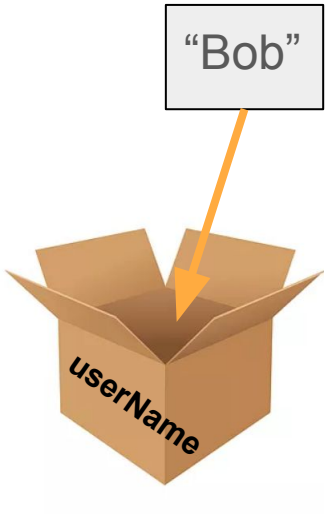
```
userName = "Rima";
```

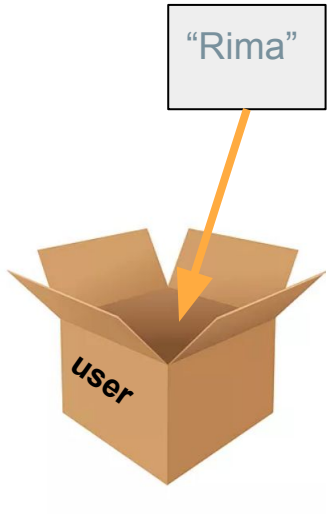


```
let user;
user = userName;
```



```
userName = "Bob";
```





Задание - создать переменную для хранения возраста

```
> let userAge; // выделяем память для хранения значения
  userAge = 25; // присваиваем значение
< 25
> userAge = 30;
< 30
>
```

Инициализация переменных

Можно сразу присвоить переменной значение при ее определении:

```
let adminName = "Jon";
```

Процесс присвоения переменной начального значения называется **инициализацией**.

Внедрение

1) <script></script> в теле документа и между ними поместить код.
2) <script src=”file.js”></script> Второй — написать код в виде отдельного файла (например, file.js) и подключить его в документ, указывая файл-источник .

Как правило, только простейшие скрипты помещаются в HTML. Более сложные выделяются в отдельные файлы.

Польза отдельных файлов в том, что браузер загрузит скрипт отдельно и сможет хранить его в кеше.

Другие страницы, которые подключают тот же скрипт, смогут брать его из кеша вместо повторной загрузки из сети. И таким образом файл будет загружаться с сервера только один раз.

Вывод в консоль console.log()

```
<body>
    <script>

        console.log('Hello, World!');

    </script>
</body>
```

console.log(“сообщение”)

Этот код выполняет очень простое действие: **выводит сообщение, указанное в круглых скобках в консоль**.

состоит:

- `console` – это **объект**. Объект в JavaScript – это набор свойств. Каждое свойство состоит из имени и значения («имя: значение»), имена также ещё называют ключами.
- `log` – это одно из свойств объекта `console`, а точнее его **метод**. Т.к. в качестве его значения выступает **функция**.
- `.` (точка) – это оператор, который мы используем **для доступа** к свойствам и методам объекта. В данном случае мы с помощью точки получаем доступ к методу `log` объекта `console`.
- `()` (скобки) – это часть синтаксиса JavaScript, с помощью которого в данном случае мы **вызываем функцию** `log`, которая является методом объекта `console`.
- `'Hello, World!'` – аргумент типа String (строка), т.е. **значение, которое мы передаём в метод** `log()`.

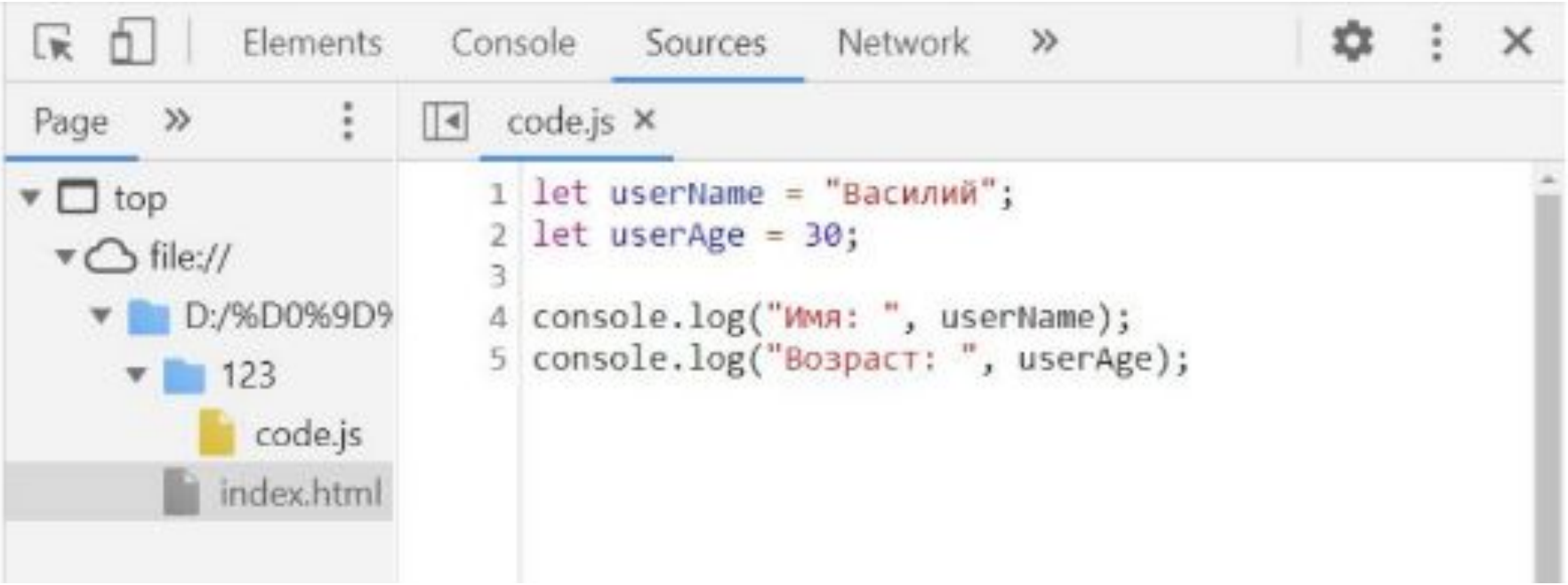
- результат возврата выражения console.log – undefined , так как ничего не возвращает

```
console.log(console.log('Hello, World!')); // undefined
```

Задание - в каком году родились

```
<body>
    <script>
        let age = 20;
        console.log("Ваш возраст", age);
        let year= 2022;
        let yearBirth = year - age ;
        console.log("Год рождения:", yearBirth);
    </script>
</body>
```

Вкладка Source



Задание

1. Объявите две переменные: admin и name.
2. Запишите строку "Petr" в переменную name.
3. Присвойте значение из переменной name в admin.
4. Выведите на экран значение admin
5. Объявите переменную для хранения почты и запишите значение,
6. Выведите значение переменной для хранения почты

Типы данных

JavaScript это динамический, слабо типизированный язык, поэтому переменная не ассоциируется с каким-либо типом данных, тип есть у значения переменной. То есть переменная может хранить значения разных типов.

В переменной можно хранить:

- **number**— любые числа
- **string**— строки
- **symbol** (символ, используется в особых случаях, появился в ES6
- **boolean**— (true/false) логическое значение

- **null** (специальное значение null, значение отсутсвует. Это значение применяется там, где ожидается получение объекта, но по каким-либо причинам данный объект не был получен)
- **undefined** (специальное значение undefined, значение неопределено.Этот тип имеет любая переменная, которой еще не было присвоено значение)
- **object**— объекты

Оператор **typeof**

Оператор `typeof` позволяет определить тип того или иного значения.

При этом у него имеется два синтаксиса, с круглыми скобками и без них:

```
typeof operand
```

```
typeof(operand)
```

Тип	Результат
Undefined	"undefined"
Null	"object"
Boolean	"boolean"
Number	"number"
String	"string"
Function	"function"
Любой другой объект	"object"

```
let name=”win”;

typeof name; //string

typeof(12.44) //number

typeof abc; //'undefined', если у переменной нет значения
```

Number (числа)

В современном JavaScript существует два типа чисел:

1. числа с плавающей точкой двойной точности» (double precision floating point numbers). Обычные числа в JavaScript хранятся в 64-битном формате
2. целые числами произвольной длины.

После объявления переменной, можно инициализировать ее числовым значением.

```
let age = 20;
let  number = 5.8;
```

Сокращение **e** 1e6= 1 000 000

let billion = 1e9; // 1 миллиард, буквально: 1 и 9 нулей(**1000000000**).Другими словами, `"e"` производит операцию умножения числа на 1 с указанным количеством нулей.

let ms = 0.000001; или **let ms = 1e-6;** // шесть нулей, слева от 1

`Самостоятельно запишите: сто, тысяча, 0,0001`

Округление

`Math.floor`	Округление в меньшую сторону	`Math.floor(3.1);`	3
`Math.ceil`	Округление в большую сторону	`Math.ceil(3.1);`	4
`Math.round`	Округление до ближайшего целого	`Math.round(3.1);`	3

Задание

Конвертируйте исходную сумму в доллар,евро. Округлить до целого

Точное округления метод toFixed(n)

Метод toFixed(n) округляет число до `n` знаков после запятой и возвращает строковое представление результата.

```
let num = 12.34;

num.toFixed(1); // "12.3" строка

+num.toFixed(1); // 12.3 строка
```

Специальные значения Number

- **«NaN»** (Not-a-Number) применяется при невозможности преобразовать результат к числу, то есть результатом не мб число, по сути, обозначает ошибку; (0/0)
- **«Infinity»** — для представления положительной бесконечности (+∞) ;(10/0)
- «-**Infinity»** — для представления отрицательной бесконечности (–∞) (-10/0);

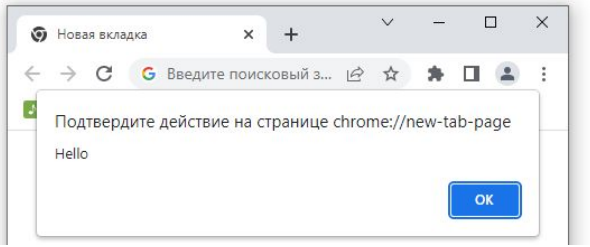
Всплывающие окна

Небольшое всплывающее окно с сообщением называется ***модальным окном***.
Понятие *модальное* означает, что пользователь не может взаимодействовать с интерфейсом остальной части страницы, нажимать на другие кнопки и т.д. до тех пор, пока взаимодействует с окном.

alert()

alert(“Сообщение”) предназначена для вывода в браузере **предупреждающего модального диалогового окна с некоторым сообщением и кнопкой «ОК»**.

```
> alert("Hello")
>
```



При его появлении **дальнейшее выполнение кода страницы прекращается** до тех пор, пока пользователь не закроет это окно или не нажмет кнопку “ОК”

Кроме этого, оно также блокирует возможность взаимодействия пользователя с остальной частью страницы.

Применение этого окна в основном используется для вывода некоторых данных при изучении языка JavaScript, в реальных проектах команда `alert()` не используется.

Если alert сообщение нужно вывести на нескольких строках, то в этом случае следует воспользоваться «символом перевода строки», который в JavaScript записывается как `\n`:

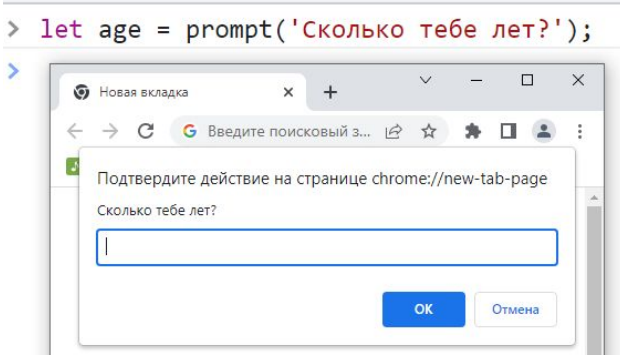
```
alert('Hello\n By');
```

prompt(title, [default])

Метод `prompt()` предназначен для вывода диалогового окна с сообщением, текстовым полем для ввода данных и кнопками «ОК» и «Отмена». Это окно предназначено для запроса данных, которые пользователю нужно ввести в текстовое поле.

```
let result = prompt(message, default);
```

title -Текст для отображения в окне.
default - Необязательный второй параметр, который устанавливает начальное значение в поле для текста в окне.
result - результат ввода . В result возвращается значение, введенное пользователем (строка) или null. Если пользователь не ввёл данные (поле ввода пустое) и нажал на «ОК», то в result будет находиться пустая строка. Пользователь также может отменить ввод нажатием на кнопку «Отмена» или нажав на клавишу Esc. В этом случае значением result станет null.



Задание

1. Запросить имя пользователя,
2. запросить возраст пользователя
3. вывести в модальном окне имя и возраст
4. вывести , сколько осталось до пенсии

```
let name = prompt('как вас зовут?','');

let age = +prompt('Сколько тебе лет?', 100);

alert("Ваше имя"+name+"\nваш возраст"+age);

alert(age+5);
```

Метод confirm()

Метод `confirm()` объекта `window` применяется для вывода модального диалогового окна с сообщением и кнопками «ОК» и «Отмена». Оно обычно используется для запроса у пользователя разрешения на выполнение того или иного действия.

```
let result = confirm(question);
```

В переменную `result` возвращается:

- `true` - если пользователь нажал на кнопку «ОК»;
- `false` - в остальных случаях.