

# Экспресс-курс «JavaScript в веб-разработке»



JavaScript  
Courses

[js.courses.dp.ua/express](http://js.courses.dp.ua/express)

[vk.com/js.express](https://vk.com/js.express)

# Что и зачем будем учить?

**JavaScript как язык программирования.** Синтаксис языка, особенности реализации ООП в JavaScript. Подходы к решению типичных задач с использованием JavaScript.

**Клиентский JavaScript.** JavaScript в браузере. Инфраструктура браузера. Объектная модель документа (DOM), событийная модель, AJAX.

**Технологии на базе JavaScript и практики.** JavaScript библиотеки (jQuery).

# Акценты

- Что является базовым;
- Понять суть и принципы.

# Кто проводит курс?

**Анатолий Кигель**

***>10 лет в сфере веб-разработки.***

***7 лет опыта преподавательской деятельности в НГУ.***

***Технический консультант дизайн-студии «Свой стиль».***

[anatoliy.kigel@gmail.com](mailto:anatoliy.kigel@gmail.com)

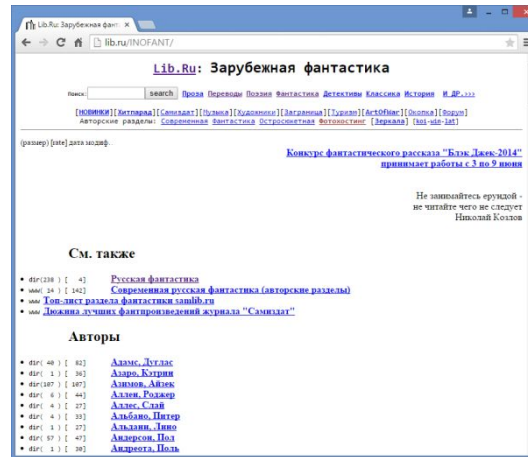
[vk.com/anatoliy\\_kigel](https://vk.com/anatoliy_kigel)

# Зачем нужен JavaScript?

Чтобы «оживить» web-страницы, повысить интерактивность страниц сайта.

# HTML статичен

Что неудивительно,  
ведь HTML не является языком  
программирования.



Типичный сценарий работы с сайтом без JavaScript: любое действие  
требует перехода на другой URL и/или полной перезагрузки  
страницы.

# Задачи JavaScript

Реагирование на действия  
пользователя

Манипуляция HTML-документом

*Всё что изменяется на странице без  
перезагрузки страницы это*

***JavaScript\****

*\*В CSS3 появилась возможность создания анимации без  
помощи JS.*

**HTML** (*HyperText Markup Language*) – язык разметки текста, по сути **структурирует** (определяет структуру текста) и выполняет роль **контейнера для текста** (данных, информации).

Язык **HTML** интерпретируется браузерами и отображается в виде документа в удобной для человека форме.

**Данные** + **Метаданные**

**Текст** + **Как текст отобразить**

**<title> Page title </title>**



# HTML- ДОКУМЕНТ

*состоит*

*из:*

**<tag** **attr="value"** **>Text data</tag>**

**Теги** как контейнер для блока текста  
+ **атрибуты** (*свойства, уточняют  
задачи тега, теги могут быть без  
атрибутов*);

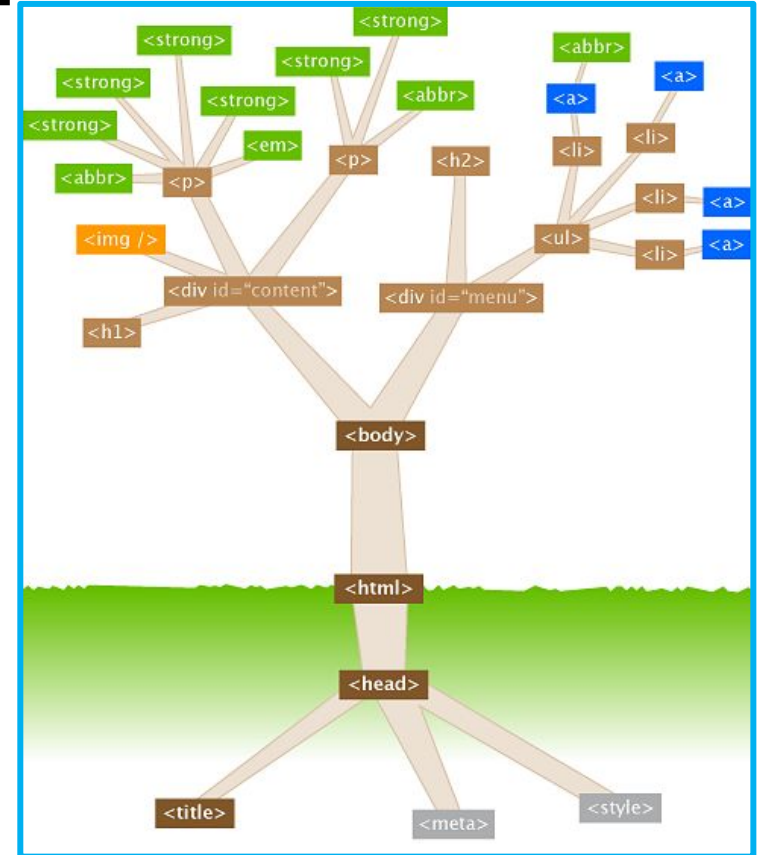
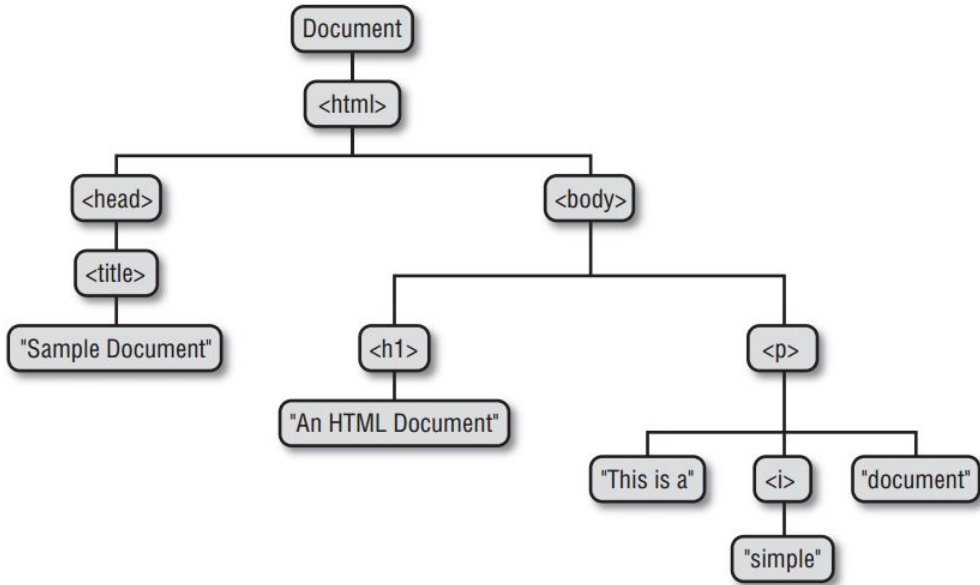
**Текстовые** данные (содержимое,  
контент).

# HTML- документ

```
1 <html>
2   <head>
3     <title>Sample Document</title>
4   </head>
5   <body>
6     <h1>An HTML Document</h1>
7     <p>This is a <i>simple</i> document.</p>
8   </body>
9 </html>
```

*Древоподобная структура HTML-  
документа*

# ДОКУМЕНТ



## Древовидная структура HTML-

# Объектная модель документа Document Object Model (DOM)

# CSS

CSS ([англ.](#) *Cascading Style Sheets* — *каскадные таблицы\** стилей) — язык **описания внешнего вида документа**, написанного с использованием HTML.

*\* таблицы здесь вообще не при чём.*

## Зачем?

*Разделение данных (тегов и текста) и их оформления;*

*Повторное использование кода.*

# Синтаксис CSS

**CSS селектор**, говорит к каким элементам будет применяться описываемый стиль (**css selector**).

Значение которое устанавливается для свойства (**value**).

```
div { color: red; font-size: 16pt; }
```

Имя свойства, которое устанавливается (**property**).

## CSS отвечает за такие аспекты

1. *Внешний вид элемента (цвет, шрифт, прозрачность и т.д. );*
2. *Размеры элемента (высота, ширина, границы, отступы и т.д.);*
3. *Положение элемента на странице;*
4. *Спецэффекты, анимация.*

*\* под элементом, подразумевается тег.*

## Задача JavaScript – манипуляция HTML-документом

1. *Обработка событий;*
2. *Изменение содержимого элемента и/или его свойств (в т. ч. стилей);*
3. *Удаление элемента;*
4. *Добавление элемента на страницу;*
5. *Изменение позиции элемента в документе;*

# HTML + CSS + JavaScript



*Безальтернативная тройка технологий  
front-end.*



# Версии JavaScript

*JavaScript – реализация языка  
ECMAScript*

*ECMAScript 3*

*ECMAScript 5*

*ECMAScript 6 => ECMAScript 2015*

*to be continue...*

<http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-262.pdf>

\* Стандарт определяет, что входит в сам языке, но не инфраструктуру где он работает. Инфраструктуру определяет стандарт DOM (Document Object Model).

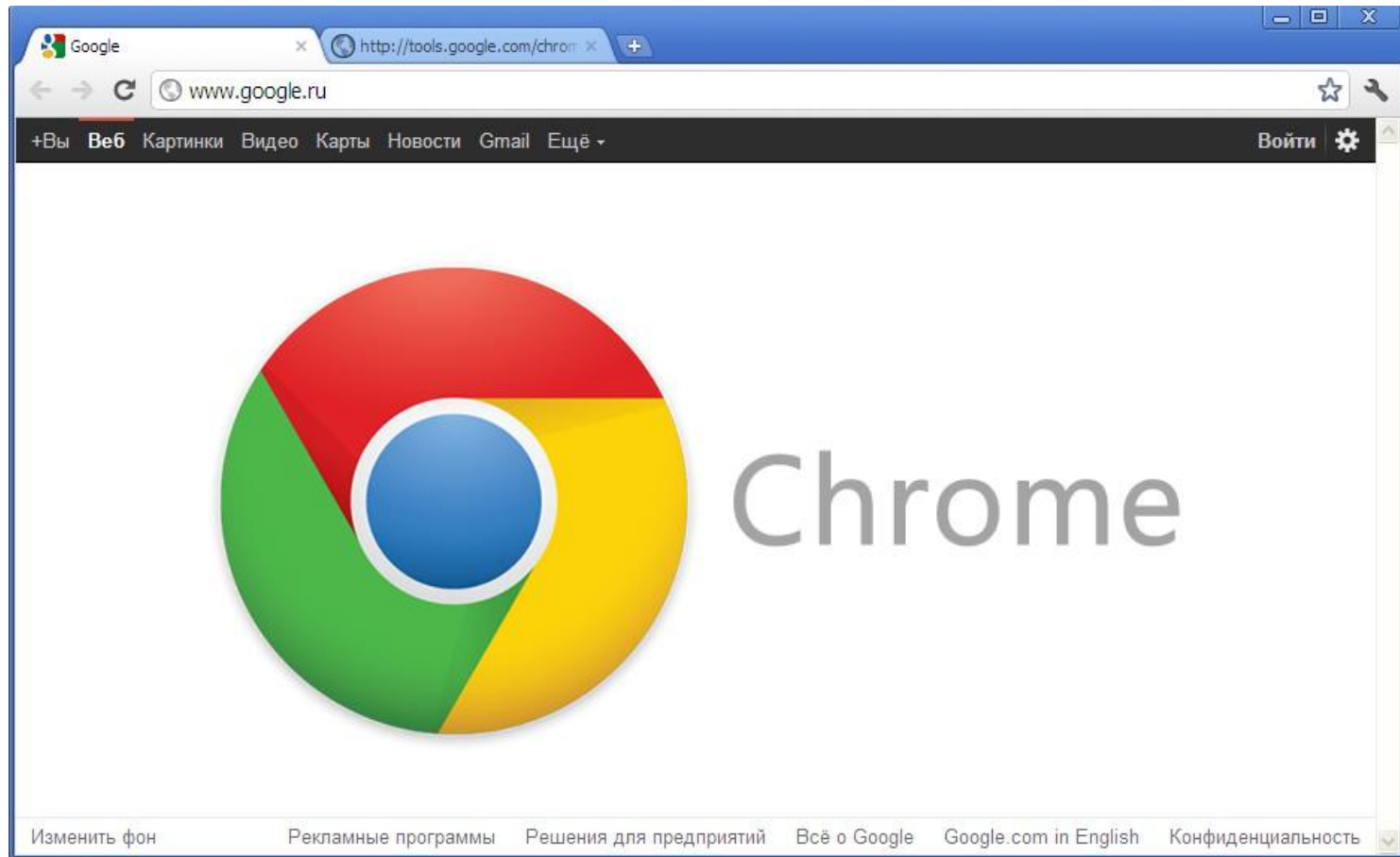
# Версии JavaScript

*Стандарт впереди планеты всей, однако, поддержка....*

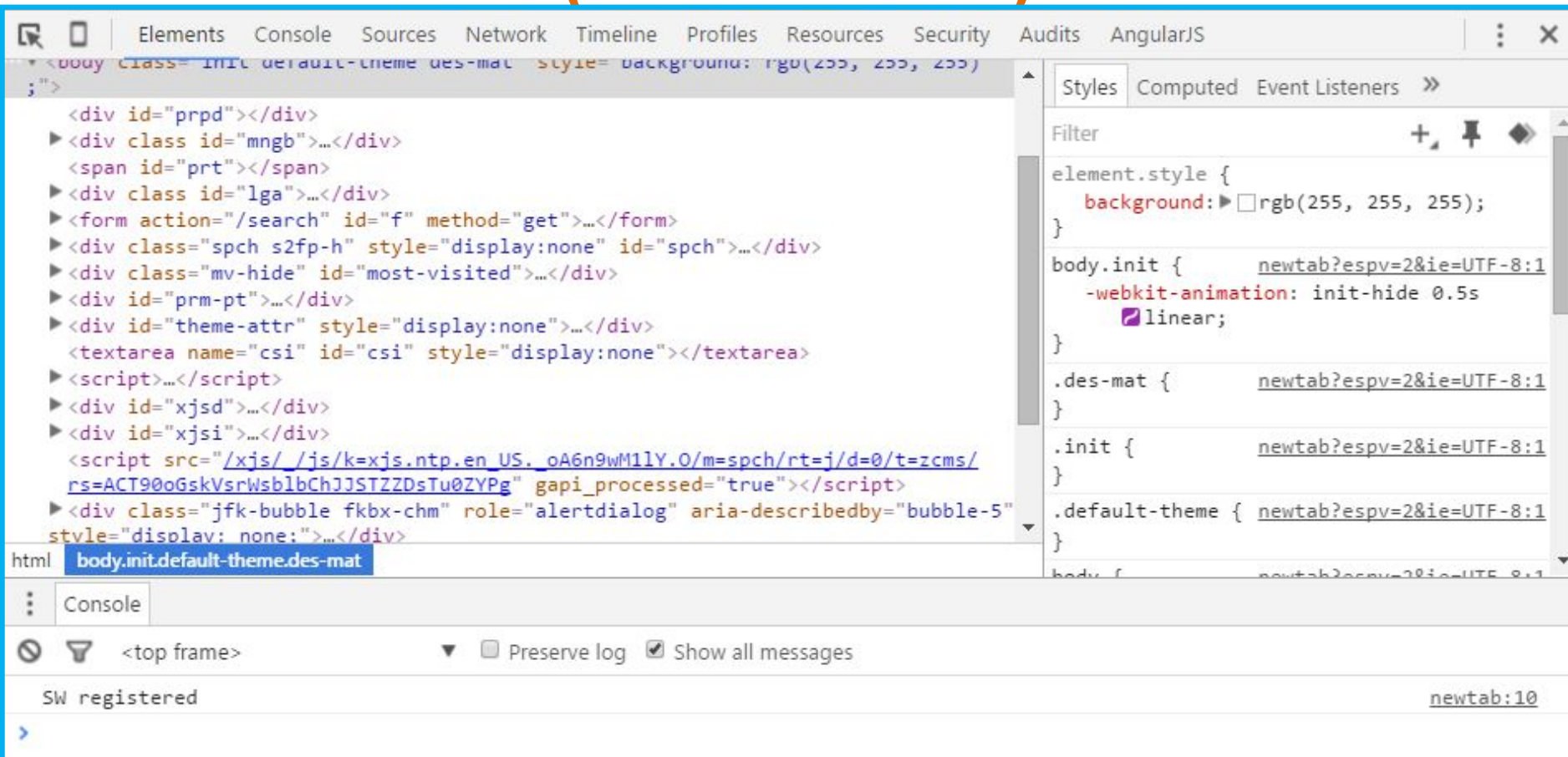
		Compilers/polyfills										Desktop browsers																		Servers/																							
		58%		74%		42%		60%		16%		15%		60%		79%		90%		63%		85%		90%		90%		93%		97%		98%		10%		21%		53%		99%		99%		11%		10%		18%		50%		56%	
Feature name		Traceur	Babel + core-js <sup>[1]</sup>	Closure	Type-Script + core-js	es6-shim	IE 11	Edge 12 <sup>[3]</sup>	Edge 13 <sup>[3]</sup>	Edge 14 <sup>[3]</sup>	FF 38 ESR	FF 45 ESR	FF 46	FF 47	FF 48	FF 49	CH 50, OP 37 <sup>[3]</sup>	CH 51, OP 38 <sup>[3]</sup>	CH 52, OP 39 <sup>[3]</sup>	SF 6.1, SF 7	SF 7.1, SF 8	SF 9	SF TP	WK	KQ 4.14 <sup>[4]</sup>	PJS	Node 0.12 <sup>[5]</sup>	Node 4 <sup>[6]</sup>	Node 5 <sup>[5]</sup>																								
Optimisation																																																					
proper tail calls (tail call optimisation)		0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2							
Syntax																																																					
default function parameters		7/7	4/7	4/7	4/7	5/7	0/7	0/7	0/7	0/7	7/7	3/7	4/7	4/7	4/7	4/7	7/7	7/7	7/7	0/7	0/7	0/7	7/7	7/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7	0/7					
rest parameters		5/5	4/5	3/5	2/5	4/5	0/5	0/5	5/5	5/5	5/5	4/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	0/5	0/5	0/5	5/5	5/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5						
spread (...) operator		15/15	15/15	13/15	12/15	4/15	0/15	0/15	12/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	15/15	0/15	5/15	9/15	15/15	15/15	0/15	0/15	0/15	0/15	0/15	0/15	0/15	0/15	0/15	0/15	0/15	0/15	0/15	0/15	0/15	0/15	0/15	0/15	0/15	0/15	0/15	0/15	0/15						
object literal extensions		6/6	6/6	6/6	4/6	6/6	0/6	0/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	6/6	0/6	1/6	5/6	6/6	6/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6	0/6						
for...of loops		7/9	9/9	9/9	6/9	3/9	0/9	0/9	6/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	9/9	9/9	0/9	2/9	8/9	9/9	9/9	0/9	0/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9	7/9						
octal and binary literals		4/4	2/4	4/4	4/4	4/4	2/4	0/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	4/4	0/4	0/4	4/4	4/4	4/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4	0/4						
template literals		5/5	4/5	4/5	3/5	3/5	0/5	0/5	4/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	5/5	0/5	0/5	5/5	5/5	5/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5						
RegExp "v" and "u" flags		5/5	3/5	3/5	0/5	0/5	0/5	0/5	2/5	4/5	4/5	2/5	2/5	5/5	5/5	5/5	5/5	5/5	5/5	0/5	0/5	0/5	5/5	5/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5	0/5						
destructuring declarations		21/22	20/22	21/22	18/22	15/22	0/22	0/22	0/22	0/22	21/22	19/22	19/22	19/22	21/22	21/22	21/22	21/22	21/22	22/22	0/22	9/22	19/22	22/22	22/22	0/22	0/22	0/22	0/22	0/22	0/22	0/22	0/22	0/22	0/22	0/22	0/22	0/22	0/22	0/22	0/22	0/22	0/22	0/22	0/22	0/22	0/22						
destructuring assignment		23/24	23/24	24/24	15/24	19/24	0/24	0/24	0/24	0/24	23/24	20/24	21/24	21/24	23/24	23/24	23/24	23/24	23/24	24/24	0/24	12/24	21/24	24/24	24/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24						
destructuring parameters		22/23	19/23	20/23	17/23	15/23	0/23	0/23	0/23	0/23	22/23	18/23	18/23	18/23	19/23	19/23	19/23	22/23	22/23	23/23	0/23	10/23	18/23	23/23	23/23	0/23	0/23	0/23	0/23	0/23	0/23	0/23	0/23	0/23	0/23	0/23	0/23	0/23	0/23	0/23	0/23	0/23	0/23	0/23	0/23	0/23	0/23	0/23					
Unicode code point escapes		2/2	1/2	1/2	1/2	1/2	0/2	0/2	2/2	2/2	2/2	0/2	1/2	1/2	1/2	1/2	2/2	2/2	2/2	0/2	0/2	2/2	2/2	2/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2						
new.target		2/2	0/2	0/2	0/2	0/2	0/2	0/2	1/2	2/2	0/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	2/2	0/2	0/2	0/2	2/2	2/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2	0/2						
Bindings																																																					
const		16/16	14/16	14/16	14/16	14/16	0/16	12/16	12/16	12/16	16/16	10/16	12/16	12/16	12/16	12/16	16/16	16/16	16/16	1/16	1/16	1/16	16/16	16/16	2/16	1/16	1/16	8/16	8/16																								
let		12/12	10/12	10/12	10/12	10/12	0/12	10/12	10/12	10/12	12/12	0/12	10/12	10/12	10/12	10/12	12/12	12/12	12/12	0/12	0/12	0/12	12/12	12/12	0/12	0/12	0/12	6/12	6/12																								
block-level function declaration <sup>[12]</sup>		Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	Yes	Yes	No	No	Flag	Yes	Yes																									
Functions																																																					
arrow functions		13/13	11/13	9/13	11/13	9/13	0/13	0/13	8/13	12/13	13/13	8/13	13/13	13/13	13/13	13/13	13/13	13/13	13/13	0/13	0/13	0/13	13/13	13/13	0/13	0/13	0/13	9/13	10/13																								
class		24/24	17/24	19/24	9/24	19/24	0/24	0/24	0/24	24/24	24/24	0/24	24/24	24/24	24/24	24/24	24/24	24/24	24/24	0/24	0/24	16/24	24/24	24/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24	0/24							
super		8/8	7/8	4/8	4/8	7/8	0/8	0/8	0/8	8/8	8/8	0/8	8/8	8/8	8/8	8/8	8/8	8/8	8/8	0/8	0/8	6/8	8/8	8/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8	0/8								
generators		27/27	24/27	24/27	16/27	0/27	0/27	0/27	0/27	27/27	27/27	20/27	25/27	25/27	25/27	25/27	25/27	27/27	27/27	27/27	0/27	0/27	0/27	27/27	27/27	0/27	0/27	0/27	20/27	20/27																							

<http://kangax.github.io/compat-table/es6/>

# Инструменты: Браузер Chrome



# Инструменты: Консоль разработчика в браузере (клавиша F12)



*Подробная информация о том, как браузер «понимает» созданный вами код.*

Інструменти: <http://google.com>



Поиск в Google

Мне повезёт!

Google.com.ua пропонується на: [українська](#)

**Презентация доступна по  
адресу:**

[js.courses.dp.ua/express](https://js.courses.dp.ua/express)

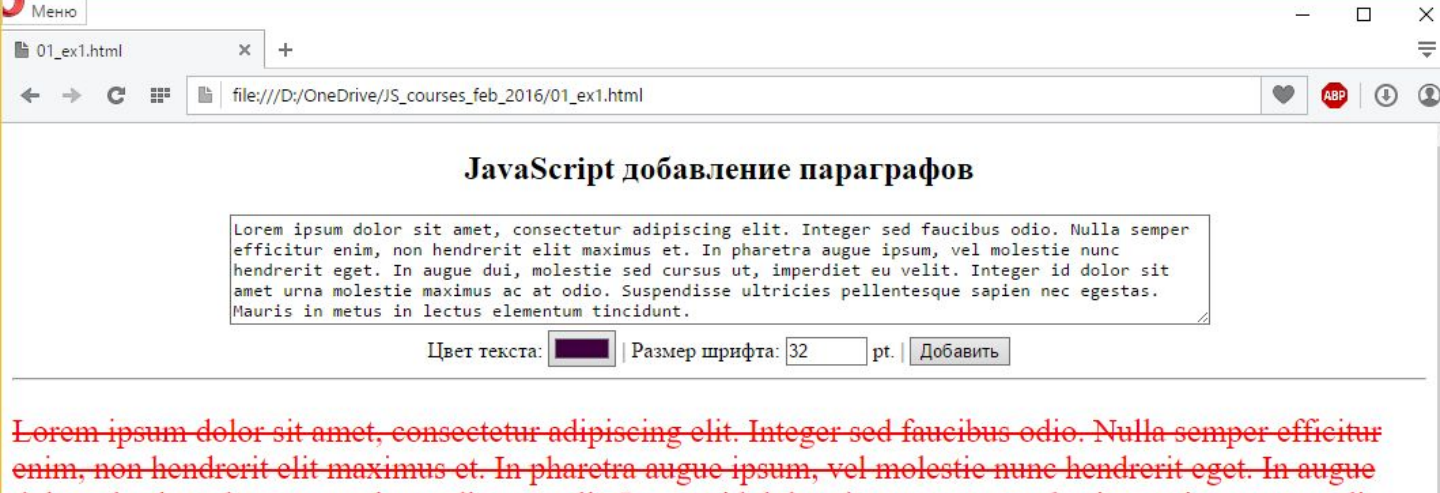
[vk.com/js.express](https://vk.com/js.express)

*Группа для вопросов, обсуждений, объявлений  
(и презентации там тоже будут).*



# #Первый пример

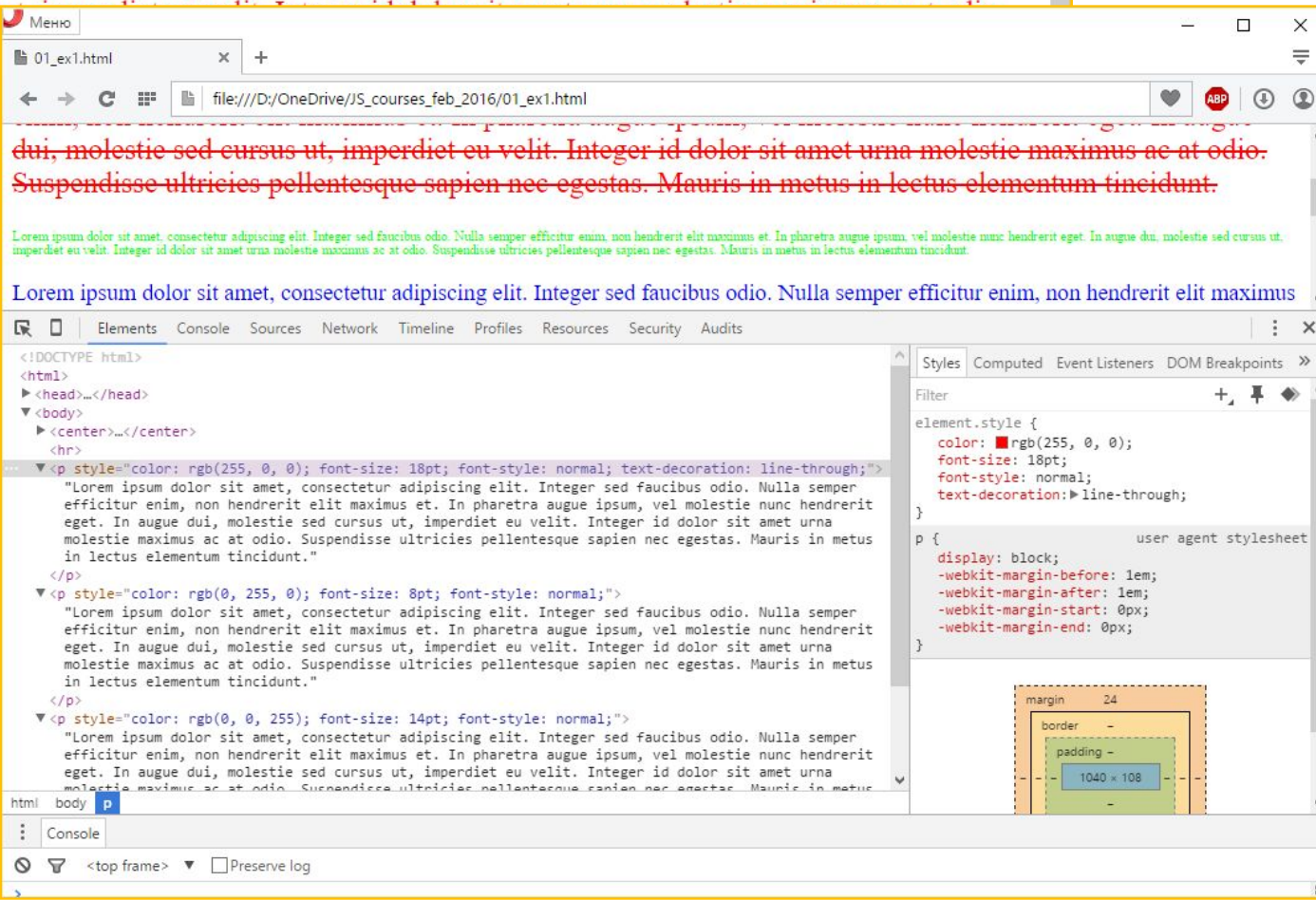
```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <script>
5     window.onload = function(){ //функция выполниться когда документ будет полностью загружен.
6
7         document.querySelector("button").onclick = function(){ //функция будет выполнена при клике по кнопке "Добавить".
8
9             var new_element                = document.createElement("p"); //Создаём новый элемент (тег) <p>
10             new_element.innerHTML          = document.querySelector("textarea").value; //Заполняем тег содержимым поля ввода.
11             new_element.style.color        = document.querySelector("#color").value; //Задаём цвет текста, на основе поля "Цвет текста".
12             new_element.style.fontSize     = document.querySelector("#fontsize").value + "pt"; //Задаём размер шрифта на основе поля "Размер шрифта".
13
14             new_element.onclick            = function(event_args){ //При клике по элементу...
15                 event_args.target.style.textDecoration = "line-through"; //...ставим стиль "зачеркнутый".
16             };
17
18             new_element.ondbclick          = function(event_args){ //При двойном клике по элементу
19                 event_args.target.remove(); //...удаляем элемент.
20             };
21
22             new_element.onmouseover        = function(event_args){ //При наведении курсора на элемент...
23                 event_args.target.style.backgroundColor = '#E0E0E0'; //...делаем элемента серым.
24             };
25
26             new_element.onmouseleave      = function(event_args){ //При уходе курсора с элемента...
27                 event_args.target.style.backgroundColor = 'FFFFFF'; //...ставим фон обратно белым.
28             };
29
30             document.body.appendChild(new_element); //Добавляет новосозданный элемент в тело документа (в тег <body>).
31         }
32     }
33 </script>
34 </head>
35 <body>
36     <center>
37         <h2>JavaScript Text Processor</h2>
38         <textarea cols='100' rows='5'></textarea>
39         <div>
40             Цвет текста: <input type='color' id='color'>
41             | Размер шрифта: <input type='number' id='fontsize' min='8' max='48' value='18'> pt.
42             | <button>Добавить</button>
43         </div>
44     </center>
45     <hr>
46 </body>
47 </html>
```



~~Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer sed faucibus odio. Nulla semper efficitur enim, non hendrerit elit maximus et. In pharetra augue ipsum, vel molestie nunc hendrerit eget. In augue dui, molestie sed cursus ut, imperdiet eu velit. Integer id dolor sit amet urna molestie maximus ac at odio. Suspendisse ultricies pellentesque sapien nec egestas. Mauris in metus in lectus elementum tincidunt.~~

*Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer sed faucibus odio. Nulla semper efficitur enim, non hendrerit elit maximus et. In pharetra augue ipsum, vel molestie nunc hendrerit eget. In augue dui, molestie sed cursus ut, imperdiet eu velit. Integer id dolor sit amet urna molestie maximus ac at odio. Suspendisse ultricies pellentesque sapien nec egestas. Mauris in metus in lectus elementum tincidunt.*

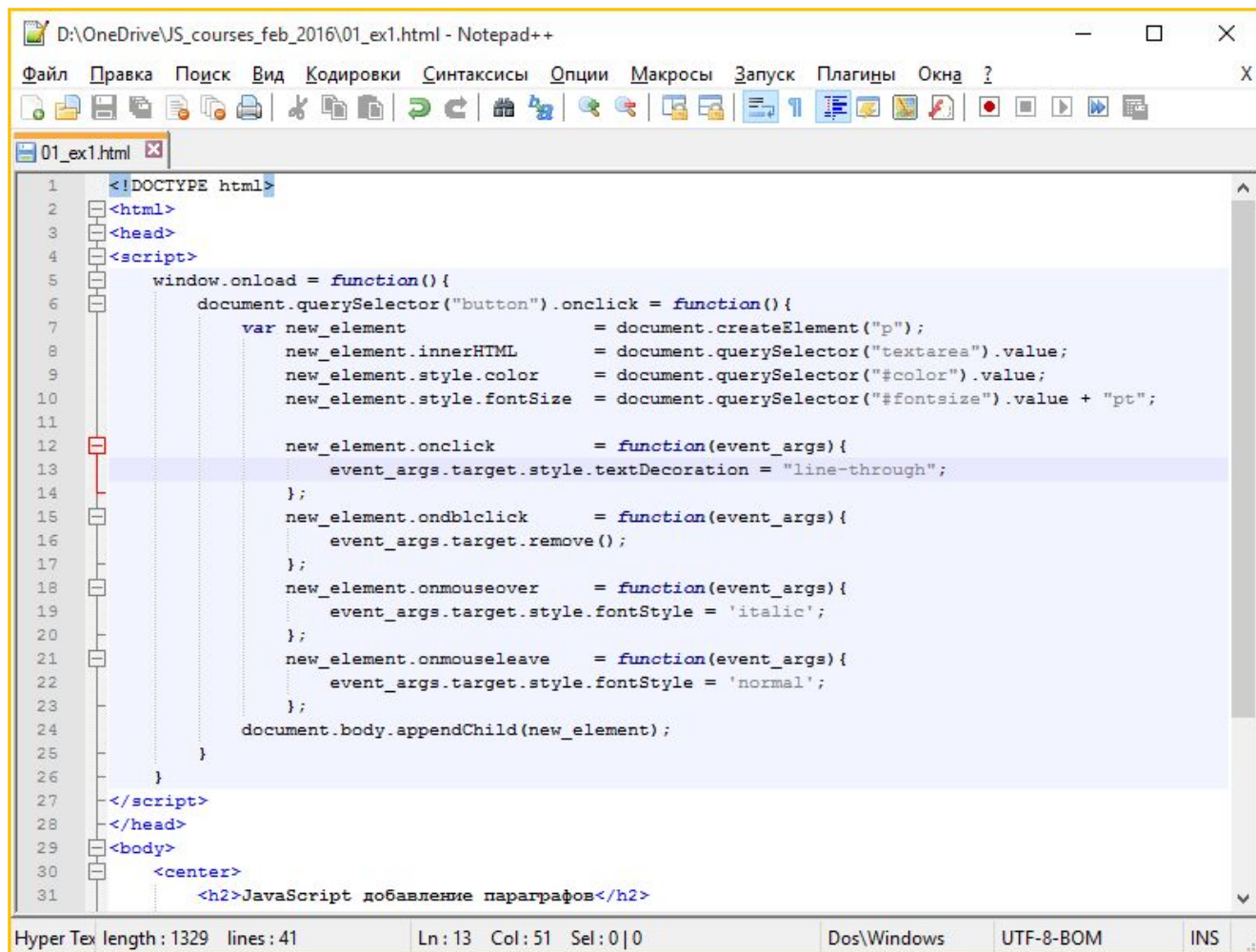
~~**Lorem ipsum**~~  
~~**Integer sed fa**~~



#Первый  
пример



# Инструменты: Notepad++



The screenshot shows the Notepad++ application window with the file 'D:\OneDrive\JS\_courses\_feb\_2016\01\_ex1.html' open. The menu bar includes 'Файл', 'Правка', 'Поиск', 'Вид', 'Кодировки', 'Синтаксисы', 'Опции', 'Макросы', 'Запуск', 'Плагин', 'Окна', and '?'. The toolbar contains various icons for file operations and editing. The editor displays the following code:

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <script>
5     window.onload = function(){
6         document.querySelector("button").onclick = function(){
7             var new_element
8                 = document.createElement("p");
9             new_element.innerHTML
10                = document.querySelector("textarea").value;
11             new_element.style.color
12                = document.querySelector("#color").value;
13             new_element.style.fontSize
14                = document.querySelector("#fontsize").value + "pt";
15
16             new_element.onclick
17                = function(event_args){
18                 event_args.target.style.textDecoration = "line-through";
19             };
20             new_element.ondblclick
21                = function(event_args){
22                 event_args.target.remove();
23             };
24             new_element.onmouseover
25                = function(event_args){
26                 event_args.target.style.fontStyle = 'italic';
27             };
28             new_element.onmouseleave
29                = function(event_args){
30                 event_args.target.style.fontStyle = 'normal';
31             };
32             document.body.appendChild(new_element);
33         }
34     }
35 </script>
36 </head>
37 <body>
38     <center>
39         <h2>JavaScript добавление параграфов</h2>
```

The status bar at the bottom shows 'Hyper Tex length: 1329 lines: 41', 'Ln: 13 Col: 51 Sel: 0|0', 'Dos\Windows', 'UTF-8-BOM', and 'INS'.

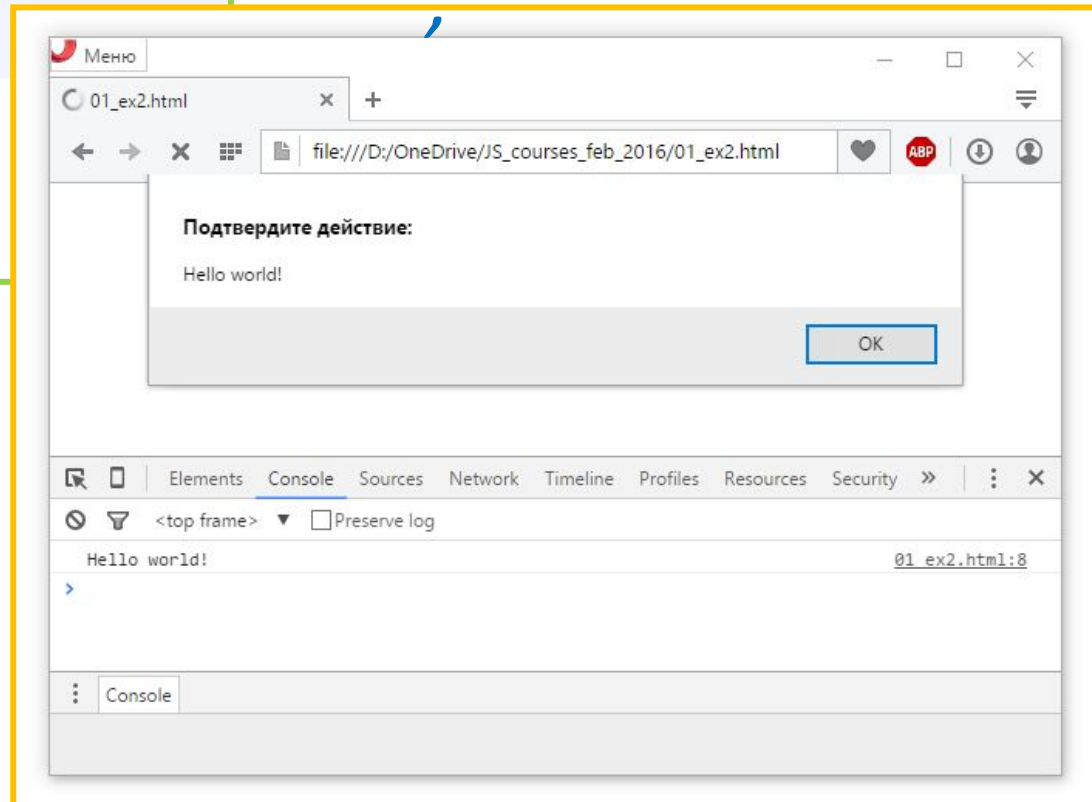
# Инструменты: служебные функции

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <script>
5
6     var message = "Hello world!";
7
8     console.log(message);
9
10    alert(message);
11
12 </script>
13 </head>
14 <body>
15 </body>
16 </html>
```

*console.log(...)*

*alert(...)*

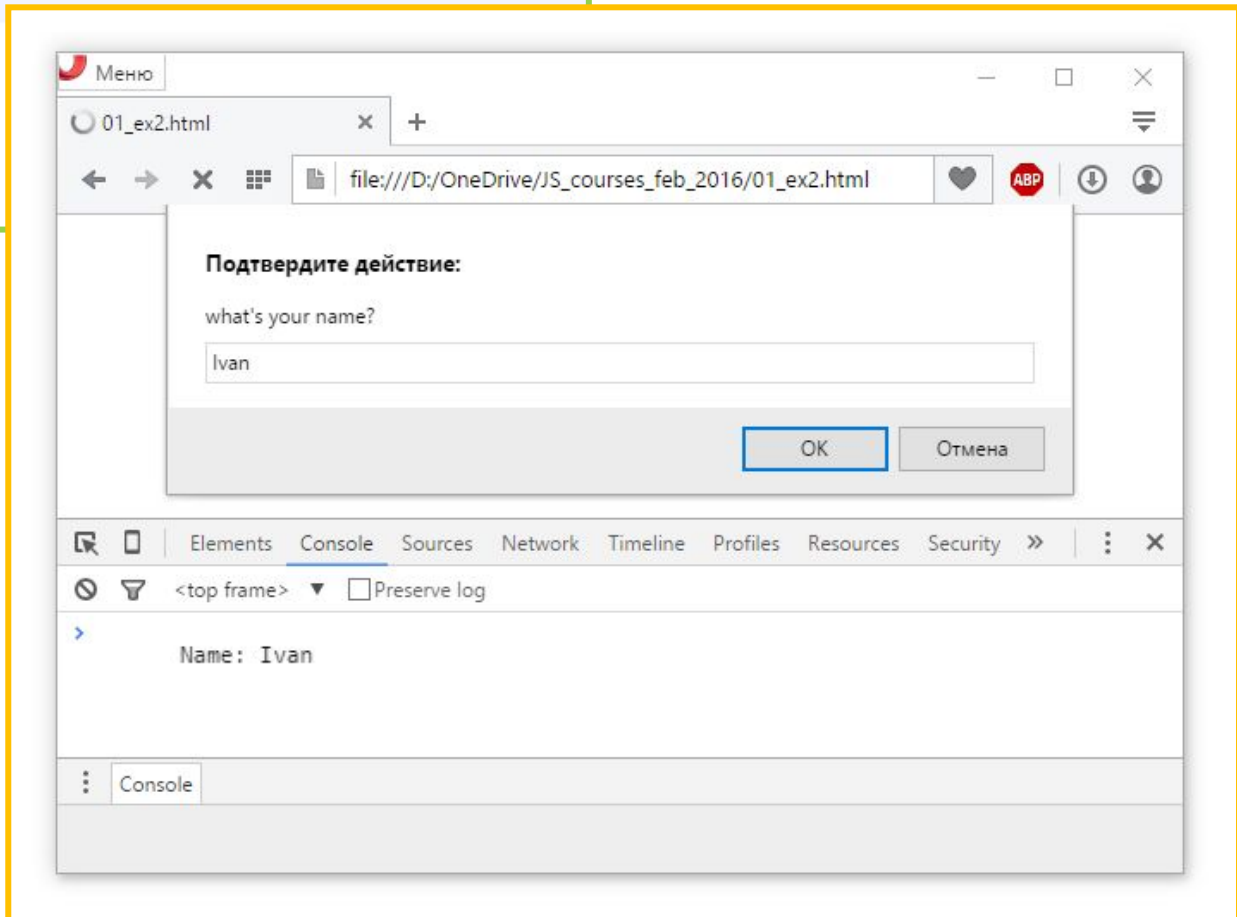
;



# Инструменты: служебные функции

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <script>
5
6     var result = prompt("what's your name?");
7
8     console.log("Name: " + result);
9
10 </script>
11 </head>
12 <body>
13 </body>
14 </html>
```

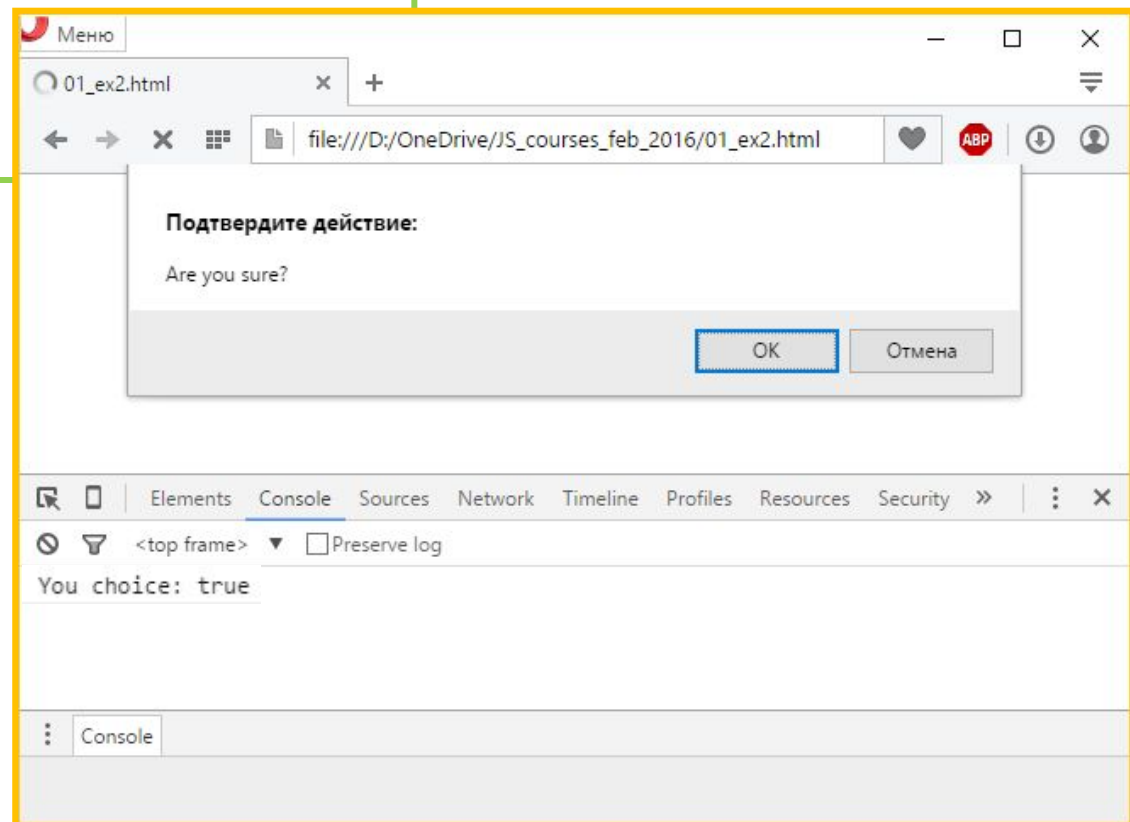
*prompt(...)*



# Инструменты: служебные функции

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4  <script>
5
6      var result = confirm("Are you sure?");
7
8      console.log("You choice: " + result);
9
10 </script>
11 </head>
12 <body>
13 </body>
14 </html>
```

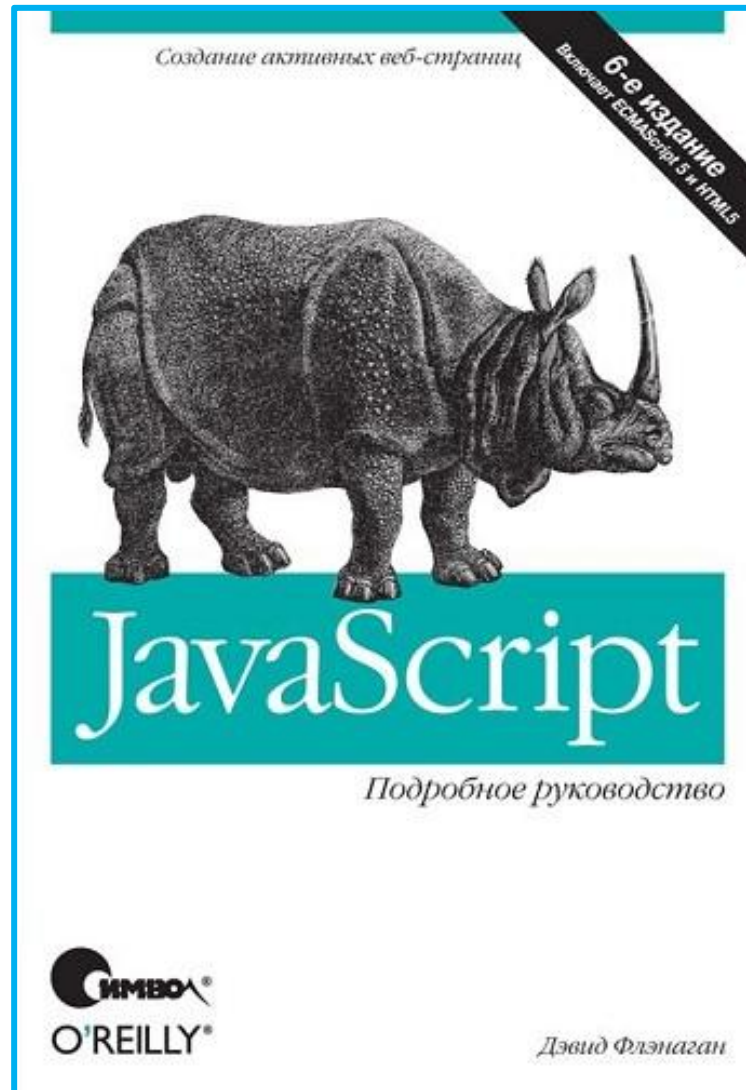
*confirm(...)*  
;



# Инструменты: «допустимый» синтаксис

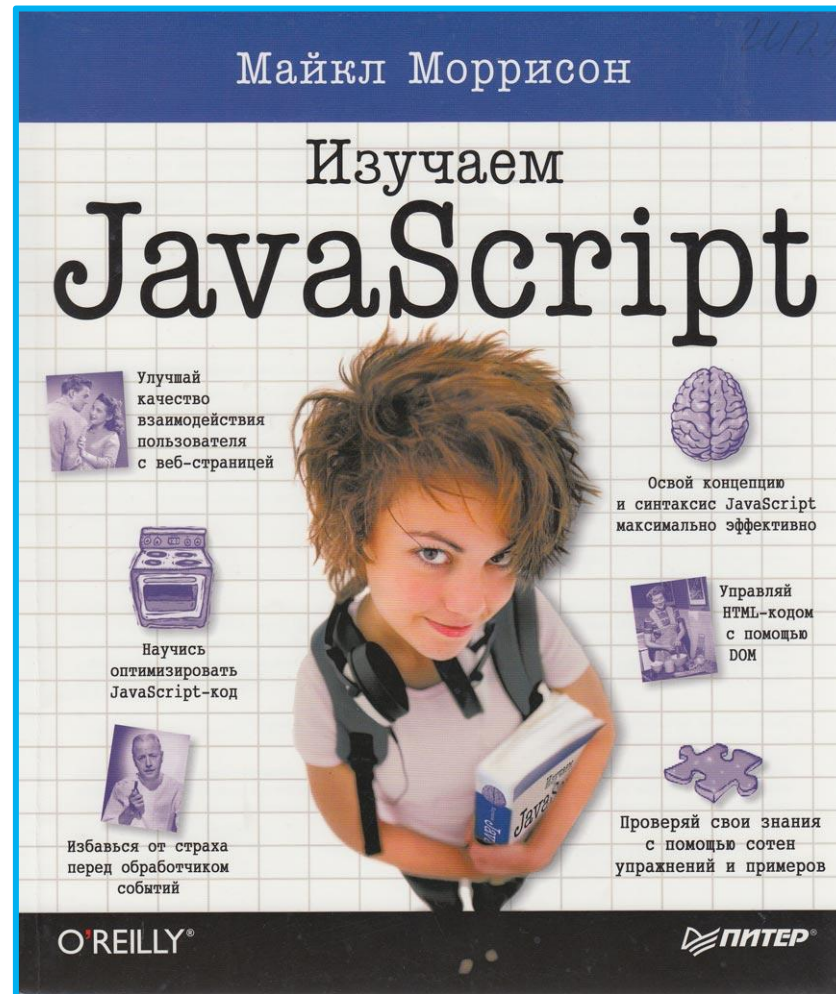
```
4 <script>  
5  
6     var message = "Text, text, text.";  
7  
8     console.log(message);  
9  
10 </script>
```

# JavaScript. Подробное руководство, 6-е издание (2012)





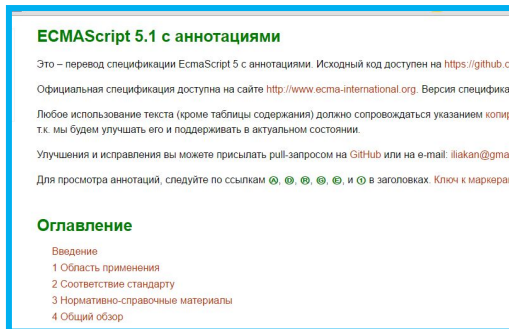
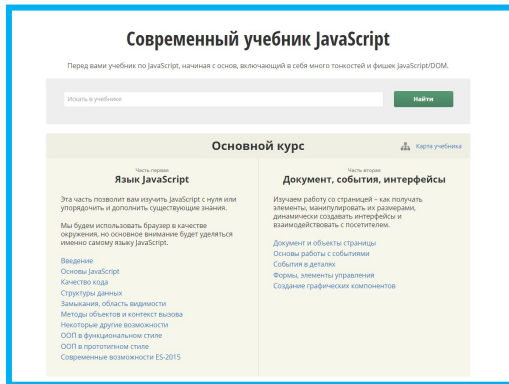
# Изучаем JavaScript, (2012)



# JavaScript.ru

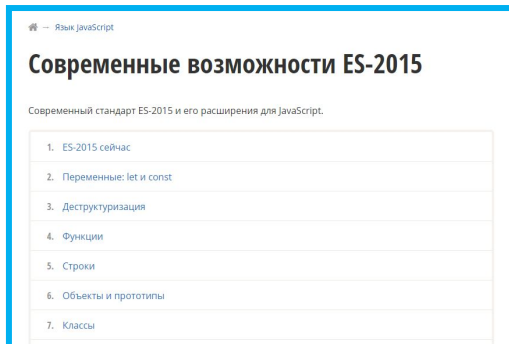
## Современный учебник JavaScript

<http://learn.javascript.ru/>



## ECMAScript 5.1 с аннотациями

<http://es5.javascript.ru/>

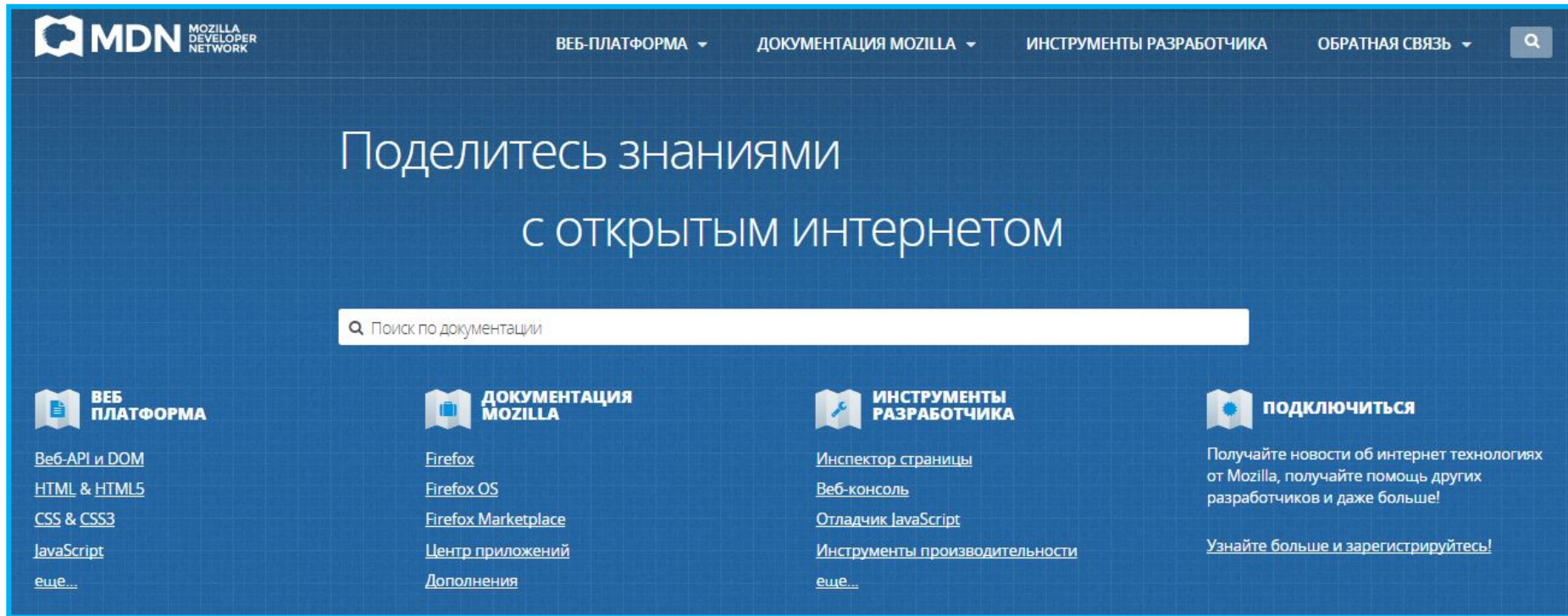


## Что нового в ES2015

<http://learn.javascript.ru/es-modern>

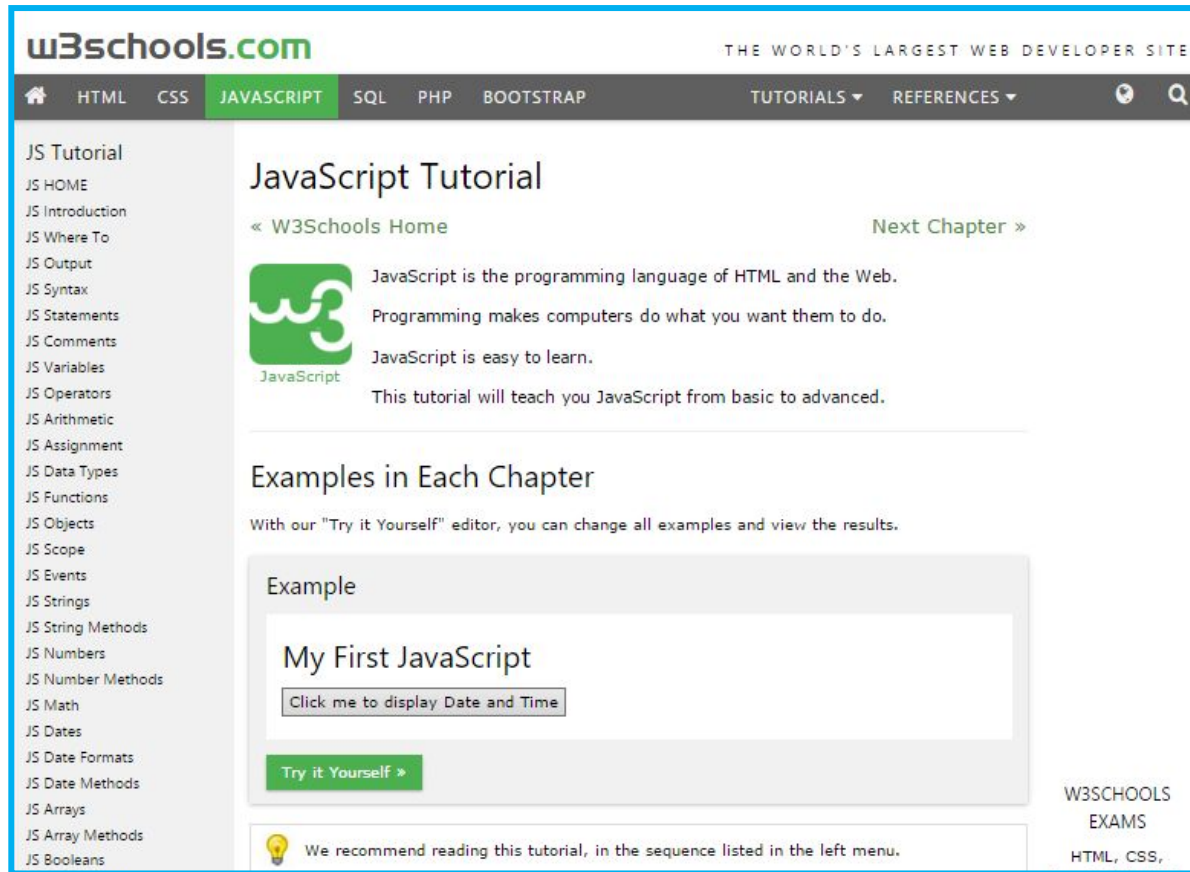


# Mozilla Developer Network. Энциклопедия JavaScript



<https://developer.mozilla.org>

# W3schools JavaScript Tutorial



The screenshot shows the W3schools website's JavaScript tutorial page. The header includes the W3schools logo, navigation links for HTML, CSS, JAVASCRIPT (highlighted), SQL, PHP, and BOOTSTRAP, and dropdown menus for TUTORIALS and REFERENCES. A left sidebar lists the JavaScript tutorial topics from JS HOME to JS Booleans. The main content area is titled 'JavaScript Tutorial' and includes a 'W3Schools Home' link, a 'Next Chapter' link, a green 'w3' logo with 'JavaScript' text, and three introductory sentences about JavaScript. Below this is a section 'Examples in Each Chapter' with a description of the 'Try it Yourself' editor. An example box titled 'My First JavaScript' contains a button that says 'Click me to display Date and Time' and a 'Try it Yourself' button. At the bottom, a lightbulb icon suggests reading the tutorial in sequence, and the text 'W3SCHOOLS EXAMS HTML, CSS,' is visible on the right.

w3schools.com THE WORLD'S LARGEST WEB DEVELOPER SITE


HTML CSS **JAVASCRIPT** SQL PHP BOOTSTRAP TUTORIALS REFERENCES

JS Tutorial

- JS HOME
- JS Introduction
- JS Where To
- JS Output
- JS Syntax
- JS Statements
- JS Comments
- JS Variables
- JS Operators
- JS Arithmetic
- JS Assignment
- JS Data Types
- JS Functions
- JS Objects
- JS Scope
- JS Events
- JS Strings
- JS String Methods
- JS Numbers
- JS Number Methods
- JS Math
- JS Dates
- JS Date Formats
- JS Date Methods
- JS Arrays
- JS Array Methods
- JS Booleans

## JavaScript Tutorial

« W3Schools Home Next Chapter »

 JavaScript

JavaScript is the programming language of HTML and the Web.  
Programming makes computers do what you want them to do.  
JavaScript is easy to learn.  
This tutorial will teach you JavaScript from basic to advanced.

### Examples in Each Chapter

With our "Try it Yourself" editor, you can change all examples and view the results.

#### Example

#### My First JavaScript

Click me to display Date and Time

Try it Yourself »

W3SCHOOLS EXAMS  
HTML, CSS,

We recommend reading this tutorial, in the sequence listed in the left menu.

<http://www.w3schools.com/js/>

# Если нужно подтянуть HTML и/или CSS



<http://www.w3schools.com/html/>  
<http://www.w3schools.com/css/>

# JavaScript как язык программирования. Синтаксис.

# JavaScript как язык программирования

*его*

*концепции*

Переменные / Типы /  
Операции  
Ветвления (условные  
операторы)  
Циклы / Массивы (структуры  
данных)  
Функции  
Объекты  
и  
и

# Переменные и

```
1 <script>
2
3   var a = "Elena";
4
5   var b = 35;
6
7   console.log(a, typeof(a));
8   console.log(b, typeof(b));
9
10  b = "77";
11
12  console.log(b, typeof(b));
13
14 </script>
```

Elena string	<u>ex2.html:7</u>
35 "number"	<u>ex2.html:8</u>
77 string	<u>ex2.html:12</u>

Ключевое слово **var** – используется при объявлении переменной. В **ECMAScript-2015** добавилось ключевое слово **let**, основное отличие в области видимости переменной объявленной с его помощью, и **const** - позволяющий объявлять константы. Оператор/функция **typeof** – позволяет получить название текущего типа для переменной.

В **JavaScript** отсутствует жёсткая типизация данных, при которой тип переменной определяется при её объявлении. В **JavaScript** тип переменной определяется при присвоении ей значения. И может меняться при каждом новом присвоении.

# Операторы, операнды и

## операции...

**Унарный оператор** – тот который взаимодействует только с одной переменной (операндом).

**Бинарный оператор** – тот который взаимодействует с двумя переменными (операндами).

```
1  <script>
2
3      var a = 5;
4      var b = 6;
5
6      var c = (a + b) * 4 - (++b);
7
8  </script>
```



# Операторы и операции (их приоритеты)

operator	Описание
<code>.[ ]()</code>	Доступ к полям, индексация массивов, вызовы функций и группировка выражений
<code>++ -- - ~ ! delete new typeof void</code>	Унарные операторы, тип возвращаемых данных, создание объектов, неопределенные значения
<code>* / %</code>	Умножение, деление, деление по модулю
<code>+ - +</code>	Сложение, вычитание, объединение строк
<code>&lt;&lt; &gt;&gt; &gt;&gt;&gt;</code>	Сдвиг битов
<code>&lt; &lt;= &gt; &gt;= instanceof</code>	Меньше, меньше или равно, больше, больше или равно, instanceof
<code>== != === !==</code>	Равенство, неравенство, строгое равенство, строгое неравенство
<code>&amp;</code>	Побитовое И
<code>^</code>	Побитовое исключающее ИЛИ
<code> </code>	Побитовое ИЛИ
<code>&amp;&amp;</code>	Логическое И
<code>  </code>	Логическое ИЛИ
<code>?:</code>	Условный оператор
<code>= OP=</code>	Присваивание, присваивание с операцией (например <code>+=</code> и <code>&amp;=</code> )
<code>,</code>	Вычисление нескольких выражений



# Операторы и операции и типы

```
1 <script>
2
3   var a = 5;
4   var b = 7;
5   var c = a + b;
6   var d = a * b;
7   console.log(c, typeof(c));
8   console.log(d, typeof(d));
9
10  b = "7";
11
12  c = a + b;
13  d = a * b;
14  console.log(c, typeof(c));
15  console.log(d, typeof(d));
16
17 </script>
```



12	"number"	ex2.html:7
35	"number"	ex2.html:8



?!?

Поскольку **JavaScript** относится к не типизированным (т.е. тип данных определяется автоматически), то в нём должна быть мощная система приведения типов. Т.е. возможность автоматического преобразования одних типов переменных в другие.

# Типы данных (переменных)

Тип данных – пометка для компьютера как относиться к тем или иным данным (переменным) и какие операции с ними возможно проводить.

*Тип определяет возможные значения и их смысл, а также **операции** которые возможны над этим типом данных.*



*Разные типы требуют разного подхода.*

# Типы данных (переменных)

Тип данных – пометка для компьютера как относиться к тем или иным данным и какие операции с ними возможно проводить.

*Тип определяет возможные значения и их смысл, а также **операции** которые возможны над этим типом данных.*

**5 типов: number, string, boolean, function, object.**

1 «служебный» тип:

**undefined**  
**+1 тип добавлен в ECMAScript-2015:**

**symbol.**

*Javascript не типизированный язык. Тип переменной не указывается при объявлении и может меняться по ходу выполнения программы.*

## Pascal/Delphi

```
1 var
2 a: integer;
3 b: real;
4 c: string;
```

## C/C++/C#/Java

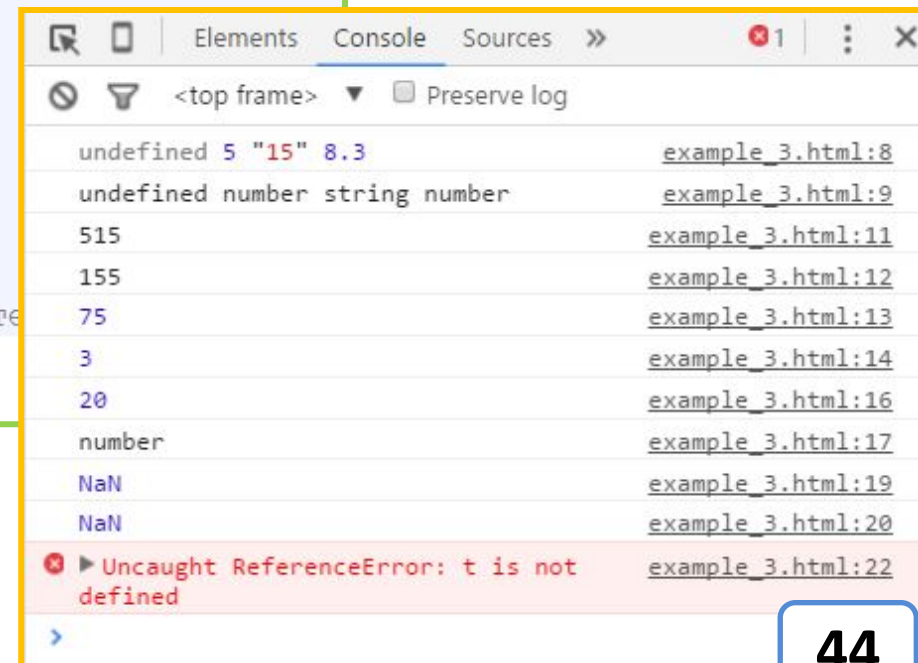
```
1 int a;
2 double b;
3 char c;
```

## JavaScript

```
1 var a = 123;
2 var b = "text";
3 var c = 234.55;
4 var d;
```

# Операции, типы, приведение типов

```
1 <!DOCTYPE html><html><head>
2 <script>
3   var a;
4   var b = 5;
5   var c = "15";
6   var d = 8.3;
7
8   console.log(a, b, c, d);
9   console.log(typeof(a), typeof(b), typeof(c), typeof(d));
10
11  console.log(b + c);
12  console.log(c + b);
13  console.log(b * c);
14  console.log(c / b);
15
16  console.log(b + Number(c));
17  console.log(typeof(Number(c)));
18
19  console.log(a + b);
20  console.log("text" * b);
21
22  console.log(b + t);
23  console.log("Это сообщение вы не увидите");
24 </script>
25 </head><body></body></html>
```



***Number()** – преобразует значение переменной к типу **number**, если это возможно.*

# Преобразование типов в JavaScript

Где зарыта  
собака?

```
1 <script>
2
3   var a = prompt("Введите любое число от 1 до 100");
4
5   console.log(a);
6
7   console.log(typeof(a));
8
9 </script>
```

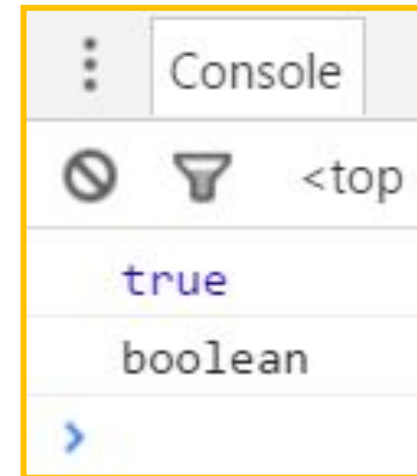
*Ввод данных из **элементов ввода** на странице или из функции **prompt** всегда возвращает строку, нужно это учитывать при дальнейшем использовании данных.*

**Подробнее:**

<https://learn.javascript.ru/types-conversion>

# Тип Boolean

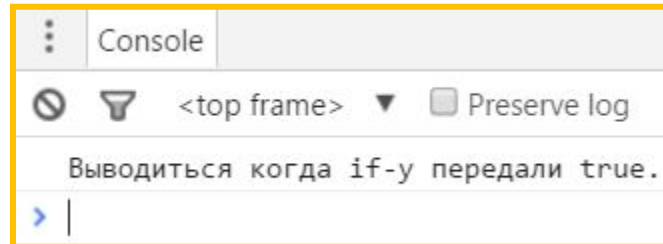
```
1 <script>
2
3     var a = true;
4
5     var b = false;
6
7     console.log(a);
8
9     console.log(typeof(a));
10
11 </script>
```



Переменная типа **boolean** содержит один из всего 2 вариантов значения: истина (**true**) или ложь (**false**).

# Оператор if-else

```
1 <script>
2
3   var bool_type_variable = true; //или false
4
5   if( bool_type_variable ) {
6
7       console.log("Выводиться когда if-у передали true.");
8
9   }else{
10
11       console.log("Выводиться когда if-у передали false.");
12   }
13 </script>
```



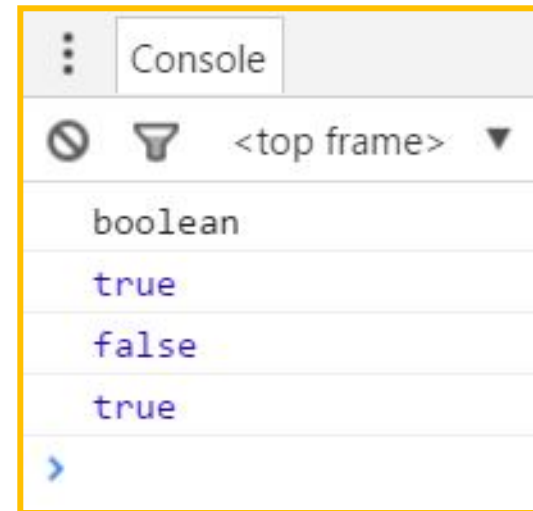
Оператор *if-else* в зависимости от переданного (*true* или *false*) значения выполняет один из двух блоков кода (**первый** или **второй**, соответственно), другой блок при этом не выполняется.



# Откуда берётся boolean?

>	<	>=	<=	==	!=	!	===	!==
---	---	----	----	----	----	---	-----	-----

```
1 <script>
2
3   var a = 3 < 4;
4   var b = 56 == 77;
5   var c = "UA" == "UA";
6
7   console.log(typeof(a));
8   console.log(a);
9   console.log(b);
10  console.log(c);
11
12 </script>
```



*Операторы сравнения*



# Откуда берётся boolean?

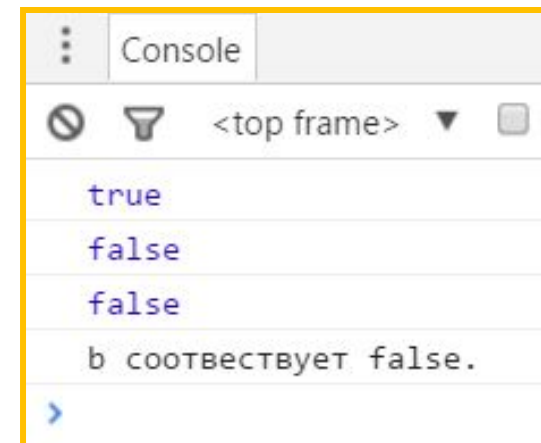
*undefined, NaN => false;*

*Number: 0 => false; все остальные =>*

*String: "" => false; все остальные =>*

*Object: null => false; существующий объект =>*

```
1 <script>
2
3   var a = 23;
4   var b = "";
5   var c = null;
6
7   console.log(Boolean(a));
8   console.log(Boolean(b));
9   console.log(Boolean(c));
10
11   if(b){
12       console.log("b соответствует true.");
13   }else{
14       console.log("b соответствует false.");
15   }
16
17 </script>
```



```
Console
<top frame>
true
false
false
b соответствует false.
```

*Из преобразование типов.*

# Откуда берётся boolean?

>	<	>=	<=	==	!=	!	===	!==
---	---	----	----	----	----	---	-----	-----

```
1 <script>
2
3   var a = 6;
4   var b = 51;
5   var c = "51";
6   var d = "6";
7
8   console.log(a > b);
9   console.log(a > c);
10  console.log(d > c);
11
12 </script>
```

?!?

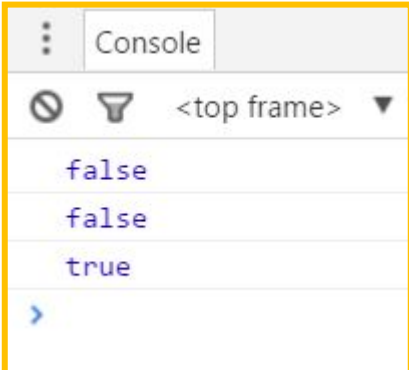
*Сложности с типами*

# Откуда берётся boolean?

>	<	>=	<=	==	!=	!	===	!==
---	---	----	----	----	----	---	-----	-----

*Сравнение строк осуществляется посимвольно.  
Сравниваться коды символов в таблице  
кодировки.*

```
1 <script>
2
3   var a = "Ivan";
4   var b = "Iven";
5
6   console.log(a == b);
7   console.log(a > b);
8   console.log(a < b);
9
10 </script>
```

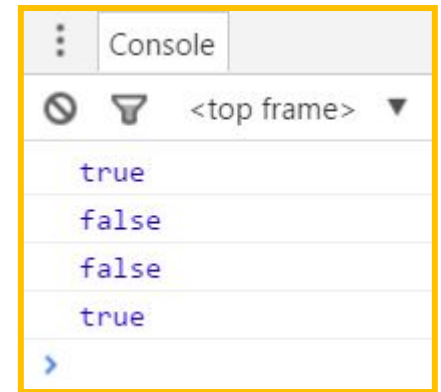


```
Console
<top frame>
false
false
true
>
```

# Откуда берётся boolean?

>	<	>=	<=	==	!=	!	===	!==
---	---	----	----	----	----	---	-----	-----

```
1 <script>
2
3   var a = "5";
4   var b = 5;
5
6   console.log(a == b);
7
8   console.log(a === b);
9
10  console.log((typeof(a) == typeof(b)) && (a == b));
11
12  console.log(typeof(a) == "string");
13
14 </script>
```



*Сравнение с учетом типа*

# Откуда берётся boolean?

```
1  <script>
2
3      var user_email = prompt("Enter email: ");
4
5      var is_email_correct = user_email.match(/\S+@\S+\.\S+/);
6
7      if(is_email_correct){
8
9          alert("email корректный.");
10
11      }else{
12
13          alert("email не корректный!");
14      }
15
16  </script>
```

*Различные функции проверки данных, но так или иначе внутри них проверка сводится к операторам сравнения.*

# Логические операторы



```
1 <script>
2   //...
3
4   if( (user_name == "ivan") && (user_password == "12345")){
5       //...
6   }
7
8   //...
9
10  if( (user_height > 145) || (user_age > 12) ){
11      //...
12  }
13  //..
14
15  var is_blocked = true;
16
17  if( !is_blocked ){
18      //...
19  }
20 </script>
```

*Когда нужны «сложные»  
условия*

# Логические операторы

<b>&amp;&amp;</b>	<b>  </b>	<b>!</b>
-------------------	-----------	----------

<b>&amp;&amp;</b>	False	True
False	False	False
True	False	True

<b>  </b>	False	True
False	False	True
True	True	True

<b>!</b>	False	True
	True	False

*Таблицы  
истинности*

# Логические операторы

&&		!
----	--	---

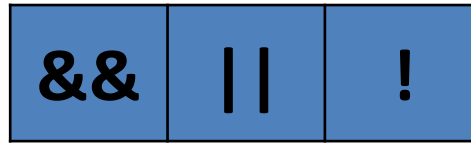
```
1 <script>
2
3   var a = 5;
4   var b = 5;
5
6   if( (a++ > 0) || (b++ > 0) ){
7       //...
8   }
9
10  console.log(a, b);
11
12 </script>
```

?!?

*Есть  
нюансы*



# Логические операторы



*Логические операторы && и || могут не проверять правый операнд, если значение левого операнда уже достаточно для итогового результата выражения.*

*Есть  
нюансы*

# Логические операторы

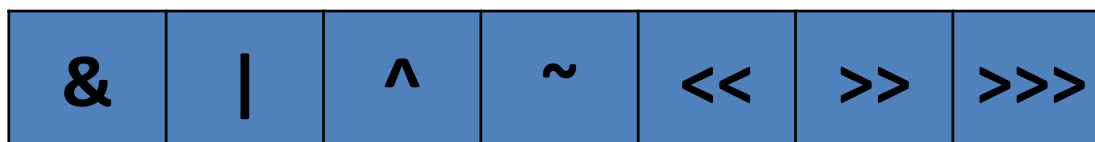
&&		!
----	--	---

```
1 <script>
2
3   var a = 5 && 7;
4   var b = 0 && 4;
5   var c = 8 || 1;
6
7   console.log(a, b, c);
8
9 </script>
```

?!?

*Есть  
нюансы...*

# Побитовые операторы



```
1  <script>
2
3      var a = 21;
4      var b = 10;
5
6      var logic    = a && b;
7      var bits     = a & b;
8
9      console.log( logic, bits );
10
11 </script>
```

?!?

*Не путайте логически и побитовые операторы, их результат далеко не всегда совпадает*

# Побитовые операторы

&		^	~	<<	>>	>>>
---	--	---	---	----	----	-----

&

21	=>	0	0	0	1	0	1	0	1
10	=>	0	0	0	0	1	0	1	0
0	<=	0	0	0	0	0	0	0	0

|

21	=>	0	0	0	1	0	1	0	1
10	=>	0	0	0	0	1	0	1	0
31	<=	0	0	0	1	1	1	1	1

*Побитовые операторы  
осуществляют действия с числами в  
двоичном виде*

**Если какое-либо действие (или блок действий) нужно повторить многократно (здесь и сейчас, без перерывов на другие действия) то циклы в помощь**

```
1 <script>
2
3     var secret_key = "123";
4     var user_key;
5
6     do{
7
8         user_key = prompt("Введите пароль: ");
9
10        if(secret_key != user_key) alert("Пароль не верный!");
11
12    }while(secret_key != user_key);
13
14    alert("Пароль правильный!");
15
16
17 </script>
```

*Классический цикл **do/while**, выполняется пока **условие** истинно (**true**)*

# while, do/while

```
1 <script>
2
3   var a = 3;
4
5   var b = 7;
6
7   while(a < b) {
8       //Какие-то действия...
9       a++; //!!!
10
11   }
12
13   console.log(a);
14
15 </script>
```

```
1 <script>
2
3   var a = 3;
4
5   var b = 7;
6
7   do{
8       //Какие-то действия...
9       a++; //!!!
10
11   }while(a < b);
12
13   console.log(a);
14
15 </script>
```

**While** – проверяет условия перед входом в цикл, **do/while** после выполнения каждой итерации (шага) цикла. Т.е. в цикле **do/while** тело выполниться минимум один раз.

# Массивы, когда переменных не хватает...

```
var a = [456, "lalala", 12.78, true];
```

**Массивы** – упорядоченный, сгруппированный набор элементов.

```
a = [ "a", "b", "c", "d", "e" ]
```

↑      ↑      ↑      ↑      ↑  
0      1      2      3      4





# В JavaScript массивы представляют собой гибрид классических массивов, стека, очереди и ассоциативных массивов.

```
1 <script>
2   var arr = [1, 3, "Elena", true];
3
4   console.log(arr);
5   console.log(arr.length);
6   //Свойство указывающее длину массива.
7   console.log(typeof(arr));
8
9   console.log(arr[0], typeof(arr[0]));
10  console.log(arr[2], typeof(arr[2]));
11  console.log(arr[3], typeof(arr[3]));
12
13  arr[0] = "Ivan";
14  console.log(arr[0], typeof(arr[0]));
15
16  arr.push(777);
17  //Добавление элемента в конец массива
18  console.log(arr, arr.length);
19
20  var x = arr.pop();
21  //Удаление последнего элемента из массива
22  console.log(arr, arr.length);
23  console.log(x);
24
25  arr.unshift("Julia");
26  //Добавление элемента в начало массива
27  console.log(arr, arr.length);
28
29  arr.shift();
30  //Удаление первого элемента массива
31  console.log(arr, arr.length);
32
33 </script>
```

[1, 3, "Elena", true]	exmaple.html:4
4	exmaple.html:5
object	exmaple.html:7
1 "number"	exmaple.html:9
Elena string	exmaple.html:10
true "boolean"	exmaple.html:11
Ivan string	exmaple.html:14
["Ivan", 3, "Elena", true, 777] 5	exmaple.html:18
["Ivan", 3, "Elena", true] 4	exmaple.html:22
777	exmaple.html:23
["Julia", "Ivan", 3, "Elena", true] 5	exmaple.html:27
["Ivan", 3, "Elena", true] 4	exmaple.html:31

*В JavaScript массивы не типизированы, т.е. могут одновременно хранить элементы*



# Цикл for и массивы

```
1 <script>
2
3   var mas = [5,34,"computer", true, "phone", 77.355];
4
5   for(var i = 0; i < mas.lenght; i++){
6       console.log(i, mas[i]);
7   }
8
9 </script>
```

0	5	<a href="#">exmaple.html:6</a>
1	34	<a href="#">exmaple.html:6</a>
2	"computer"	<a href="#">exmaple.html:6</a>
3	true	<a href="#">exmaple.html:6</a>
4	"phone"	<a href="#">exmaple.html:6</a>
5	77.355	<a href="#">exmaple.html:6</a>

*Цикл **for** удобен для тех случаев, когда заранее известно (или можно просчитать на основе уже имеющихся данных), сколько раз нужно будет повторить то или иное действие.*

# Метод `forEach` и массивы

```
1 <script>
2
3     var arr = [55, true, "Polina", 77.88];
4
5     arr.forEach(function(item, i, arr) {
6         console.log( i + ": " + item);
7     });
8
9 </script>
```

0: 55	<u>ex 02.html:7</u>
1: true	<u>ex 02.html:7</u>
2: Polina	<u>ex 02.html:7</u>
3: 77.88	<u>ex 02.html:7</u>

Метод ***forEach*** принимает функцию с тремя параметрами, которая будет вызвана для каждого элемента массива. Прервать такой «цикл» НЕВОЗМОЖНО.

# Контрольный вопрос #1

```
1 <script>
2
3   var mas = [5,34,"computer", true, "phone", 77.355];
4
5   for(var i = 0; i < mas.length; i++){
6       console.log(i, mas[i]);
7   }
8
9   console.log(i);
10
11 </script>
```

*Чему равно  $i$  после выполнения цикла?*

*Область видимости переменных – глобальная, переменная объявленная в блоке видна и за его пределами.*

## Контрольный вопрос #2

```
1 <script>
2
3     var i = 5;
4
5     while(i) {
6         console.log(i--);
7     }
8
9     console.log("После цикла: " + i);
10
11 </script>
```

*Что мы увидим в консоле?*

# «Дырки» в массивах




```
1 <script>
2
3   var arr = [5, 77, 23];
4
5   console.log("Length: ", arr.length);
6   for(var i = 0; i < arr.length; i++){
7       console.log(i + " : " + arr[i]);
8   }
9
10  arr[7] = 89;
11
12  console.log("Length: ", arr.length);
13  for(var i = 0; i < arr.length; i++){
14      console.log(i + " : " + arr[i]);
15  }
16
17 </script>
```

Length: 3	<a href="#">ex 02.html:5</a>
0 : 5	<a href="#">ex 02.html:7</a>
1 : 77	<a href="#">ex 02.html:7</a>
2 : 23	<a href="#">ex 02.html:7</a>
Length: 8	<a href="#">ex 02.html:12</a>
0 : 5	<a href="#">ex 02.html:14</a>
1 : 77	<a href="#">ex 02.html:14</a>
2 : 23	<a href="#">ex 02.html:14</a>
3 : undefined	<a href="#">ex 02.html:14</a>
4 : undefined	<a href="#">ex 02.html:14</a>
5 : undefined	<a href="#">ex 02.html:14</a>
6 : undefined	<a href="#">ex 02.html:14</a>
7 : 89	<a href="#">ex 02.html:14</a>

*Если в массиве задать элемент с индексом которого нет, то в нём появятся «дырки» - элементы без значений.*

# Массивы это коллекция пар «ключ => значение».

```
1 <script>
2   var arr = [1, 77, "Elena", 55.6, true];
3
4   console.log(arr.indexOf(55.6));
5
6   console.log(arr.indexOf("lalala"));
7
8   console.log(3 in arr);
9
10  console.log(7 in arr);
11 </script>
```

		<top frame>		<input type="checkbox"/> Preserve
3		example.html:4		
-1		example.html:6		
true		example.html:8		
false		example.html:10		
>				

```
0 => 1;
1 => 77;
2 => "Elena";
3 => 55.6;
4 => true;

arr.length == 5
```

*В классических массива все ключи – числа, как правило нумерация начинается с 0, и идёт без пропусков.*

# Ассоциативные массивы



*Массивы это хранилища пар  
«**ключ=>значение**», но ключом может  
выступать не только целые числа.*



# Ассоциативные массивы

```
1 <script>
2
3   var mas = [];
4
5   mas["name"]    = "Ivan";
6   mas["age"]     = 19;
7   mas["height"]  = 1.81;
8   mas["smoke"]   = false;
9
10  console.log(mas);
11
12  console.log(mas.name, mas["name"]);
13
14  console.log(mas.length);
15
16  console.log(Object.keys(mas));
17
18  console.log(Object.keys(mas).length);
19
20  mas.push("55");
21
22  console.log(mas, mas.length);
23
24 </script>
```

```
exmaple.html:10
[name: "Ivan", age: 19, height: 1.81, smoke:
false]
Ivan Ivan exmaple.html:12
0 exmaple.html:14
["name", "age", "height", "smoke"] exmaple.html:16
4 exmaple.html:18
exmaple.html:22
["55", name: "Ivan", age: 19, height: 1.81, smoke:
false]
1
```

*Массивы это хранилища пар «**ключ=>значение**», но ключом может выступать не только целые числа, но и строки.*



# Цикл for/in

```
1 <script>
2
3     var mas = [6, 8, 33, "text", true, 77.8];
4
5     for(var x in mas){
6         console.log(x, mas[x]);
7     }
8
9 </script>
```

0	6	<a href="#">exmaple.html:6</a>
1	8	<a href="#">exmaple.html:6</a>
2	33	<a href="#">exmaple.html:6</a>
3	text	<a href="#">exmaple.html:6</a>
4	true	<a href="#">exmaple.html:6</a>
5	77.8	<a href="#">exmaple.html:6</a>

*Цикл **for/in** позволяет перебрать ключи массива не заморачиваясь с их количеством.*

# Цикл for/in и ассоциативные массивы

```
1 <script>
2
3     var mas = [];
4
5     mas["name"] = "Elena";
6     mas.age     = 18;
7     mas.smoke   = false;
8
9     for(var x in mas){
10         console.log(x, mas[x]);
11     }
12
13 </script>
```

name Elena	<a href="#">example.html:10</a>
age 18	<a href="#">example.html:10</a>
smoke false	<a href="#">example.html:10</a>

*Цикл **for/in** позволяет перебрать ключи массива не заморачиваясь с их количеством.*

# Цикл for/in и свойства HTML элементов

```
1  <!DOCTYPE>
2  <html>
3  <body>
4      <h1>This is a text.</h1>
5  <script>
6      var element = document.querySelector("h1");
7
8      for(var property_name in element){
9          console.log(property_name, ":" ,element[property_name]);
10     }
11 </script>
12 </body>
13 </html>
```

```
ownerDocument : ▶#document
parentNode    : ▶<body>...</body>
parentElement  : ▶<body>...</body>
childNodes    : ▶[text]
firstChild     : "This is a text."
lastChild     : "This is a text."
previousSibling : ▶#text
nextSibling    : ▶#text
nodeValue     : null
textContent   : This is a text.
hasChildNodes  : function hasChildNodes() { [native code] }
normalize     : function normalize() { [native code] }
```

[http://js.courses.dp.ua/express/01/ex\\_01.html](http://js.courses.dp.ua/express/01/ex_01.html)

*Цикл **for/in** хорош для перебора  
свойств HTML элементов.*

# Объекты как ассоциативный массив

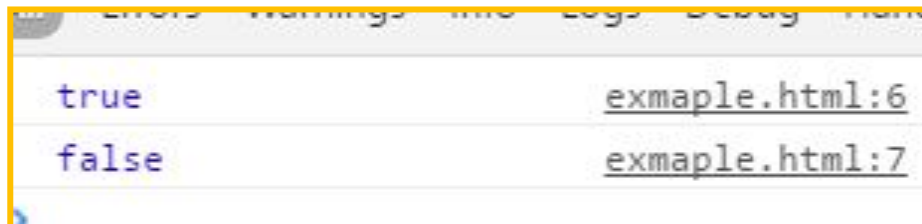
```
1 <script>
2
3   var mas = {name: "Ivan", age: 22};
4   console.log(mas);
5
6   for(var x in mas){
7       console.log(x, mas[x]);
8   }
9
10  mas["name"] = "John";
11  console.log(mas.name);
12
13  mas.push(55);
14
15 </script>
```

Object {name: "Ivan", age: 22}	exmaple.html:5
name Ivan	exmaple.html:8
age 22	exmaple.html:8
John	exmaple.html:13
Uncaught TypeError: mas.push is not a function	exmaple.html:15

Объекты в JavaScript также можно использовать как ассоциативный массив, но в таком случае не будут доступны методы-помощники.

# Как отличить массив от объекта

```
1 <script>
2
3   var arr1 = [1,2,3];
4   var arr2 = { title: "the news", year: 1999 };
5
6   console.log(Array.isArray(arr1));
7   console.log(Array.isArray(arr2));
8
9 </script>
```

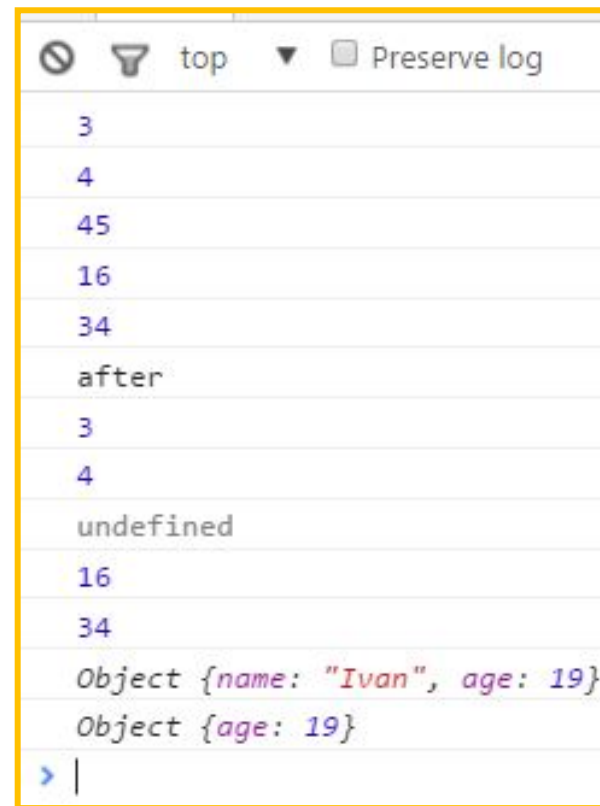


true	exmaple.html:6
false	exmaple.html:7

Метод ***Array.isArray()*** возвращает ***true*** если полученный объект является «классическим» массивом ***[ ]***, и ***false*** во всех остальных случаях.

# Удаление элементов массива

```
1 <script>
2
3   var arr = [3,4,45,16,34];
4
5   for(var i = 0; i < arr.length; i++){
6       console.log(arr[i]);
7   }
8
9   delete arr[2];
10  console.log("after");
11
12  for(var i = 0; i < arr.length; i++){
13      console.log(arr[i]);
14  }
15
16  //-----//
17
18  var mas = { name: "Ivan", age: 19 };
19
20  console.log(mas);
21
22  delete mas.name;
23
24  console.log(mas);
25
26 </script>
```

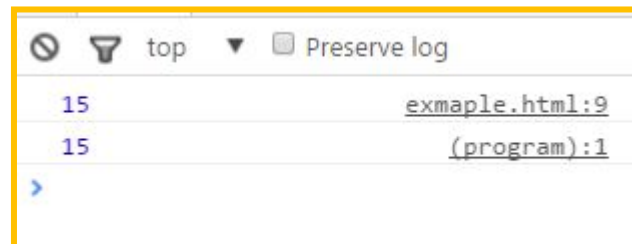


3  
4  
45  
16  
34  
after  
3  
4  
undefined  
16  
34  
Object {name: "Ivan", age: 19}  
Object {age: 19}

*Удаление элементов массива по сути образует «дырки».*

# eval

```
1 <script>
2
3   var a = 8;
4
5   var b = 7;
6
7   eval("var c = a + b;");
8
9   console.log(c);
10
11  eval("console.log(c);");
12
13 </script>
```



*eval()* выполняет **JavaScript** код,  
представленный строкой.

**Категорически не рекомендуется  
использовать!!!**

# Структуры данных ECMAScript-2015

**Map** – коллекция для хранения записей вида ключ:значение (по сути словарь - *dictionary*). В отличие от ассоциативных массивов, ключом может выступать не только строка но и другие типы данных.

```
1 <script>
2
3   var map = new Map();
4
5   map.set('1', 'Ivan');
6   map.set(1, 'Elena');
7   map.set(true, 123);
8
9
10  console.log(map.get(1));
11  console.log(map.get('1'));
12  console.log(map.get(false));
13
14  console.log("Map size: " + map.size);
15
16 </script>
```

Elena	<a href="#">ex 02.html:10</a>
Ivan	<a href="#">ex 02.html:11</a>
undefined	<a href="#">ex 02.html:12</a>
Map size: 3	<a href="#">ex 02.html:14</a>



# Структуры данных ECMAScript-2015

**Set** – коллекция для хранения множества значений, каждое из которых может встречаться только один раз. В отличие от ассоциативных массивов, ключом может выступать не только строка но и другие типы данных.

```
1 <script>
2
3   var set = new Set();
4
5   set.add(55);
6   set.add(true);
7   set.add("Elena");
8   set.add(55);
9   set.add(true);
10
11  console.log("Size :" + set.size);
12
13  console.log(set.has(44));
14  console.log(set.has(true));
15
16  set.delete(true);
17
18  console.log(set.has(true));
19
20 </script>
```

Size :3	<u>ex 02.html:11</u>
false	<u>ex 02.html:13</u>
true	<u>ex 02.html:14</u>
false	<u>ex 02.html:18</u>

# JSON <http://www.json.org/json-ru.html>

```
1 <script>
2
3   var message = '{"name": "Vasya", "age": "19", "languages": ["English","Russian","French"] }';
4
5   var person = JSON.parse(message);
6
7   for(var x in person){
8       console.log(x + ": " + person[x]);
9   }
10
11   console.log(person);
12
13 </script>
```

```
name: Vasya                                     example.html:8
age: 19                                          example.html:8
languages: English,Russian,French              example.html:8
▼ Object {name: "Vasya", age: "19", Languages: Array[3]} example.html:11
  age: "19"
  ▶ languages: Array[3]
    name: "Vasya"
  ▶ __proto__: Object
```

Декодирование из JSON

*JSON - текстовый формат текстовый формат обмена данными, основанный на JavaScript и обычно используемый именно с этим языком. А по простому, это текстовый формат передачи массивов и объектов в JS.*

# JSON <http://www.json.org/json-ru.html>

```
1 <script>
2
3   var arr = [3, 67, "Computer", 'IFC', true, null, {title:"Harry Potter", autor: "J.K.Rowling"}, 768.22];
4
5   console.log(arr);
6
7   var text_format = JSON.stringify(arr);
8
9   console.log(text_format);
10
11 </script>
```

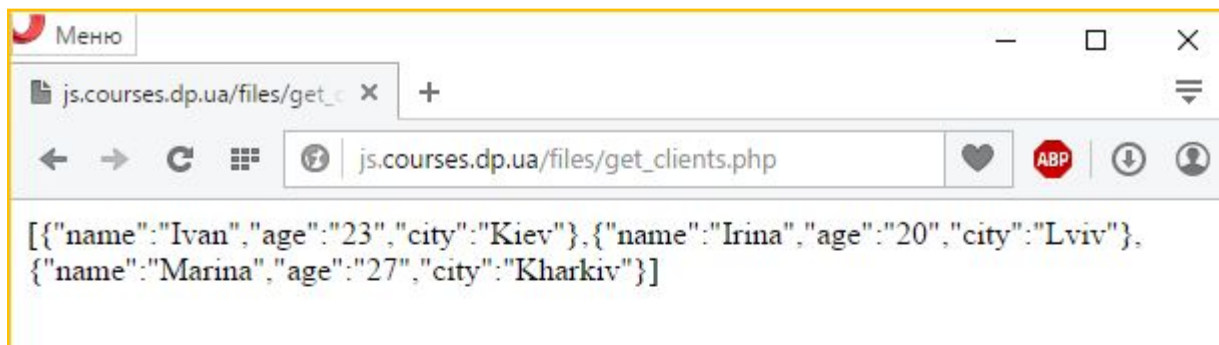
Кодирование в  
JSON

```
► [3, 67, "Computer", "IFC", true, null, Object, 768.22]      exmaple.html:5
[3,67,"Computer","IFC",true,null,{"title":"Harry Potter","autor":"J.K.Rowling"},768.22]  exmaple.html:9
```

*JSON - текстовый формат текстовый формат обмена данными, основанный на JavaScript и обычно используемый именно с этим языком. А по простому, это текстовый формат передачи массивов и объектов в JS.*

# JSON в реальности

[http://js.courses.dp.ua/express/01/get\\_clients.php](http://js.courses.dp.ua/express/01/get_clients.php)



```
1 <?php
2     header('Access-Control-Allow-Origin: *');
3
4     $arr = array(
5         array("name" => "Ivan", "age" => "23", "city" => "Kiev"),
6         array("name" => "Irina", "age" => "20", "city" => "Lviv"),
7         array("name" => "Marina", "age" => "27", "city" => "Kharkiv")
8     );
9
10    echo json_encode($arr);
11
12 >
```

Сценарий на PHP кодирует данные в формат JSON и возвращает строку с закодированными данными вместо страницы при запросе на адрес:

[http://js.courses.dp.ua/files/get\\_clients.php](http://js.courses.dp.ua/files/get_clients.php). А наш JS код декодирует данные и наполняет ими страницу.

# JSON в реальности

```
1 <html>
2 <head></head>
3 <body>
4   <h1>Clients</h1>
5   <div id="clients"></div>
6   <script>
7     var XHR = new XMLHttpRequest();
8     XHR.onload = function() {
9
10      var data = XHR.responseText;
11      console.log("Recieve data: " + data);
12
13      var clients_arr = JSON.parse(XHR.responseText);
14      console.log(clients_arr);
15
16      var clients_div = document.getElementById("clients");
17
18      for(var id in clients_arr){
19        var new_div = document.createElement("div");
20
21        for(var field in clients_arr[id]){
22          new_div.innerHTML += "<b>" + field + "</b>: " +
23            clients_arr[id][field] + "; ";
24        }
25        clients_div.appendChild(new_div);
26      }
27    };
28    XHR.open("get", "http://js.courses.dp.ua/files/get_clients.php",
29      true);
30    XHR.send();
31  </script>
32 </body></html>
```



## Clients

**name:** Ivan; **age:** 23; **city:** Kiev;  
**name:** Irina; **age:** 20; **city:** Lviv;  
**name:** Marina; **age:** 27; **city:** Kharkiv;

# JSON в реальности

The screenshot displays a web browser's developer tools interface. The top section shows the HTML structure with a `<div id="clients">` element containing three `<div>` elements, each representing a client's data. The middle section shows the CSS styles for the selected `div` element, including `display: block;` from the user agent stylesheet. The bottom section shows the console logs, including a message "Recieve data: [{"name": "Ivan", "age": "23", "city": "Kiev"}, {"name": "Irina", "age": "20", "city": "Lviv"}, {"name": "Marina", "age": "27", "city": "Kharkiv"}]" and a subsequent log showing the data as an array of objects.

```
<html>
  <head></head>
  <body>
    <h1>Clients</h1>
    <div id="clients">
      <div>
        <b>name</b>
        ": Ivan; "
        <b>age</b>
        ": 23; "
        <b>city</b>
        ": Kiev; "
      </div>
      <div>...</div>
      <div>...</div>
    </div>
    <script>...</script>
  </body>
</html>
```

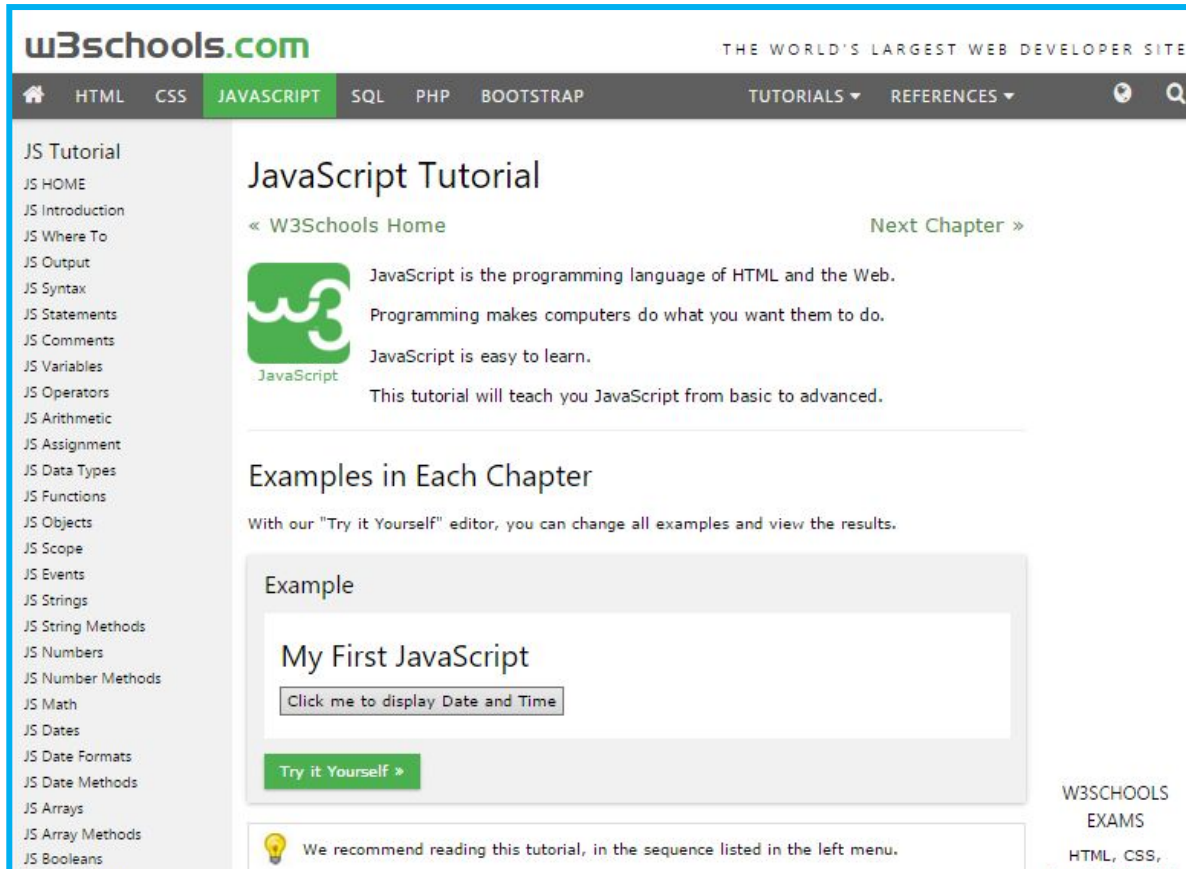
html body div#clients div

Filter element.style { }  
div { display: block; } user agent stylesheet

margin -  
border -  
padding -  
1013 x 18

Filter Recieve data: [{"name": "Ivan", "age": "23", "city": "Kiev"}, {"name": "Irina", "age": "20", "city": "Lviv"}, {"name": "Marina", "age": "27", "city": "Kharkiv"}] json\_example.html:11  
[Object, Object, Object] json\_example.html:14  
0: Object  
age: "23"  
city: "Kiev"  
name: "Ivan"  
\_\_proto\_\_: Object  
1: Object  
2: Object

# W3schools JavaScript Tutorial



The screenshot shows the W3schools website's JavaScript tutorial page. The header includes the W3schools logo, navigation links for HTML, CSS, JAVASCRIPT (highlighted), SQL, PHP, and BOOTSTRAP, and dropdown menus for TUTORIALS and REFERENCES. A left sidebar lists the JavaScript tutorial topics from JS HOME to JS Booleans. The main content area is titled 'JavaScript Tutorial' and includes a 'W3Schools Home' link, a 'Next Chapter' link, a green 'w3' logo with 'JavaScript' text, and three introductory sentences about JavaScript. Below this is a section 'Examples in Each Chapter' with a description of the 'Try it Yourself' editor. An example box titled 'My First JavaScript' contains a button that says 'Click me to display Date and Time' and a 'Try it Yourself' button. At the bottom, a lightbulb icon suggests reading the tutorial in sequence, and a 'W3SCHOOLS EXAMS' link is visible.

w3schools.com THE WORLD'S LARGEST WEB DEVELOPER SITE


HTML CSS **JAVASCRIPT** SQL PHP BOOTSTRAP TUTORIALS REFERENCES

JS Tutorial

- JS HOME
- JS Introduction
- JS Where To
- JS Output
- JS Syntax
- JS Statements
- JS Comments
- JS Variables
- JS Operators
- JS Arithmetic
- JS Assignment
- JS Data Types
- JS Functions
- JS Objects
- JS Scope
- JS Events
- JS Strings
- JS String Methods
- JS Numbers
- JS Number Methods
- JS Math
- JS Dates
- JS Date Formats
- JS Date Methods
- JS Arrays
- JS Array Methods
- JS Booleans

## JavaScript Tutorial

« W3Schools Home Next Chapter »

 JavaScript

JavaScript is the programming language of HTML and the Web.  
Programming makes computers do what you want them to do.  
JavaScript is easy to learn.  
This tutorial will teach you JavaScript from basic to advanced.

### Examples in Each Chapter


With our "Try it Yourself" editor, you can change all examples and view the results.

#### Example

#### My First JavaScript

Click me to display Date and Time

Try it Yourself »

 We recommend reading this tutorial, in the sequence listed in the left menu.

W3SCHOOLS EXAMS  
HTML, CSS,

<http://www.w3schools.com/js/>

**Презентация доступна по  
адресу:**  
[js.courses.dp.ua/express](https://js.courses.dp.ua/express)

[vk.com/js.express](https://vk.com/js.express)

*Группа для вопросов, обсуждений, объявлений  
(и презентации там тоже будут).*