Экспресс-курс «JavaScript в веб-разработке»

JavaScript Courses

js.courses.dp.ua/express

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

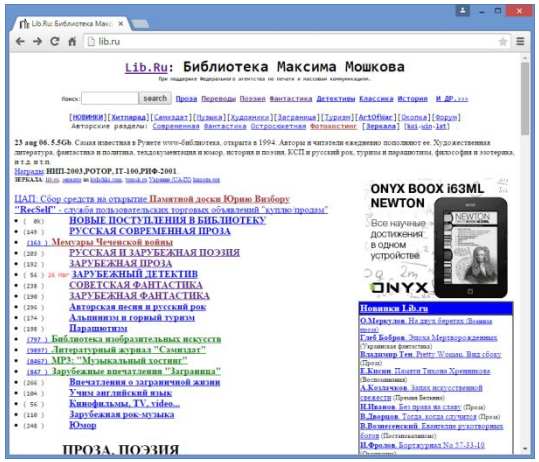
vk.com/anatoliy_kigel

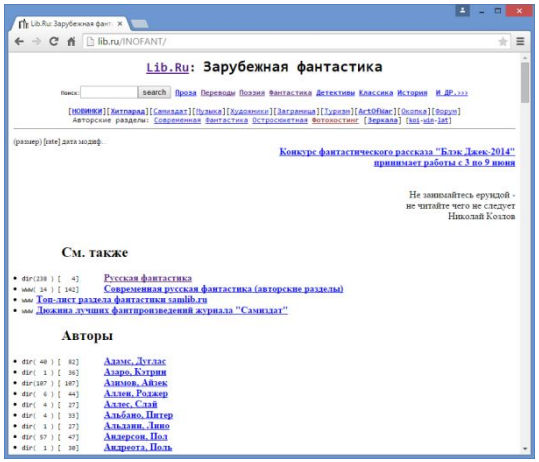
Зачем нужен JavaScript?

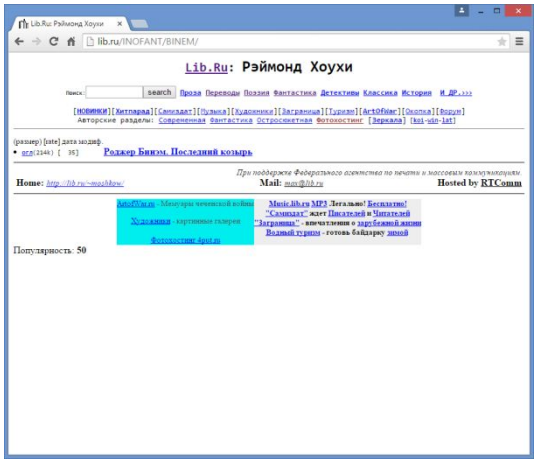
Чтобы «оживить» web-страницы, повысить интерактивность страниц сайта.

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****

**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);

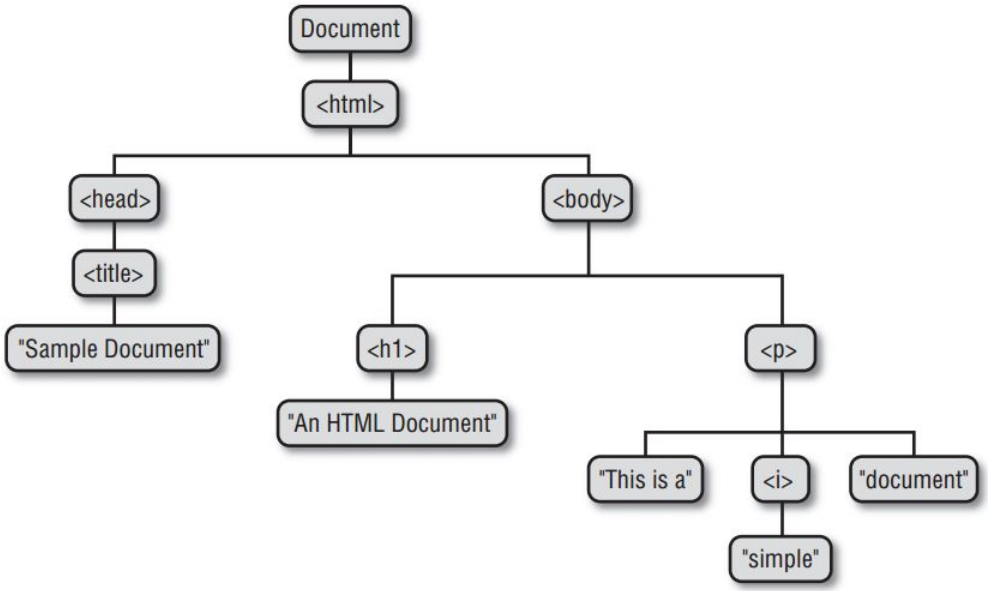
Текстовые данные (содержимое, контент).

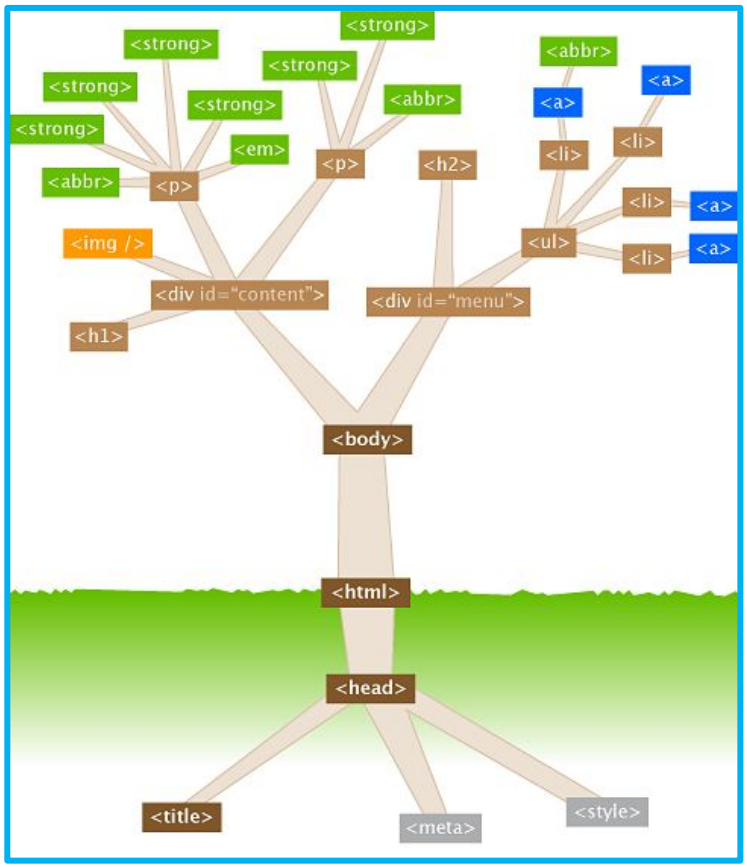
HTML-документ

```
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>
```

Древовидная структура HTML-документа

HTML-документ





Древовидная структура HTML-документа

Объектную 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 применятся 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

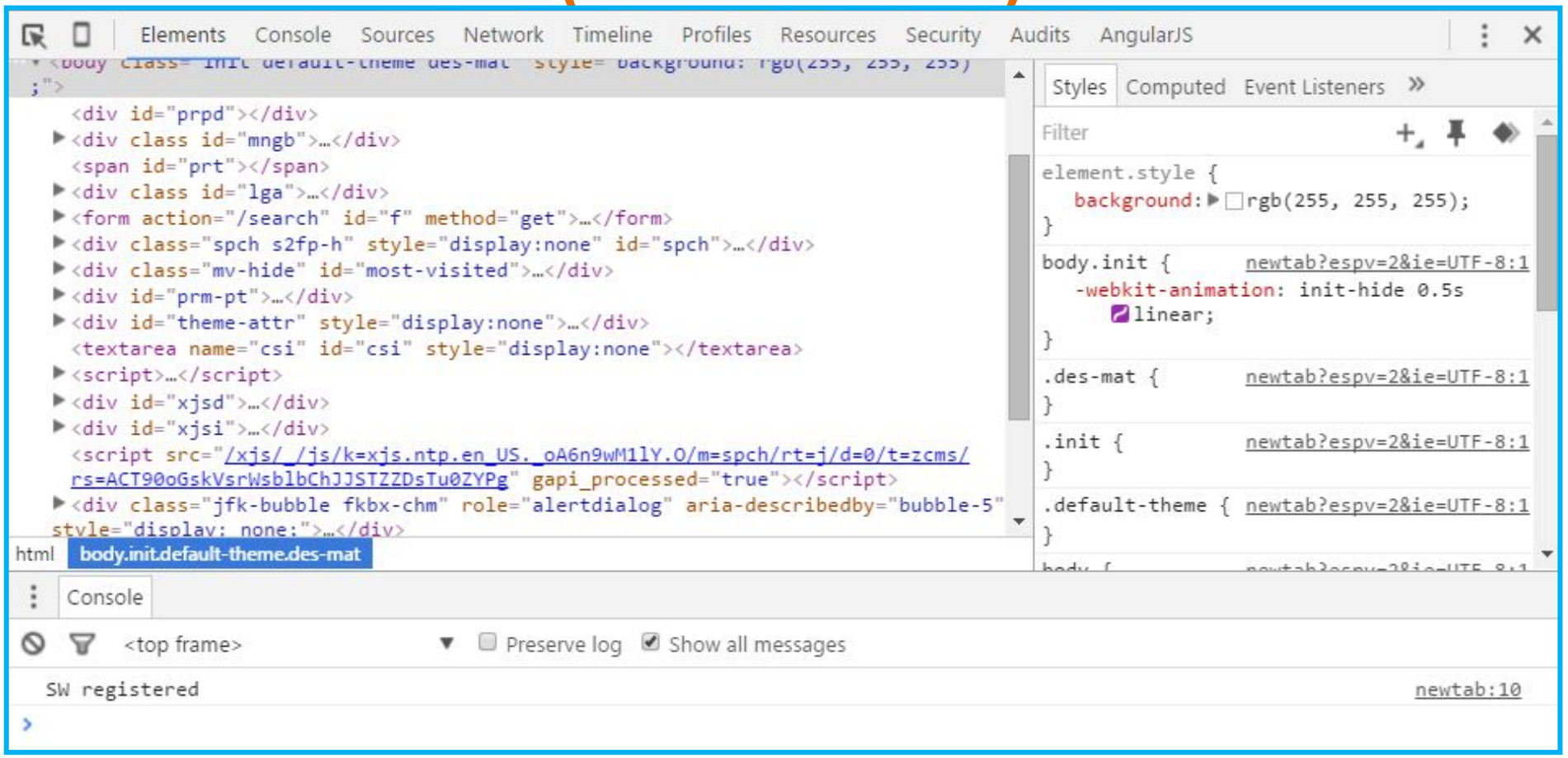
Стандарт впереди планеты всей, однако, поддержка….

Feature name	Current browser	Compilers/polyfills: Traceur	Babel + core-js[1]	Closure	Type-Script + core-js	es6-shim	Desktop browsers: IE 11	Edge 12[3]	Edge 13[3]	Edge 14[3]	FF 38 ESR	FF 45 ESR	FF 46	FF 47	FF 48	FF 49	CH 50, OP 37[0]	CH 51, OP 38[0]	CH 52, OP 39[0]	SF 6.1, SF 7	SF 7.1, SF 8	SF 9	SF TP	WK	KQ 4.14[4]	PJS	Servers/r: Node 0.12[5]	Node 4[5]	Node 5[5]
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	2/2	2/2	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	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
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	5/5	0/5	0/5	0/5	5/5	5/5	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	15/15	0/15	5/15	9/15	15/15	15/15	0/15	0/15	0/15	0/15	15/15
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	6/6	0/6	1/6	5/6	6/6	6/6	0/6	0/6	0/6	6/6	6/6
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
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	4/4	0/4	0/4	4/4	4/4	4/4	0/4	0/4	0/4	4/4	4/4
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	5/5	0/5	0/5	5/5	5/5	5/5	0/5	0/5	0/5	5/5	5/5
RegExp "y" 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	5/5	0/5	0/5	0/5	5/5	5/5	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
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
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
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	1/2	2/2	2/2	2/2	0/2	0/2	2/2	2/2	2/2	0/2	0/2	0/2	2/2	2/2
new.target	2/2	0/2	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	2/2
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	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	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[12]	Yes	Yes	Yes	Yes	No	No	Yes	Yes	Yes	Yes	No	No	Yes	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	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	24/24	0/24	0/24	16/24	24/24	24/24	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	8/8	0/8	0/8	6/8	8/8	8/8	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/

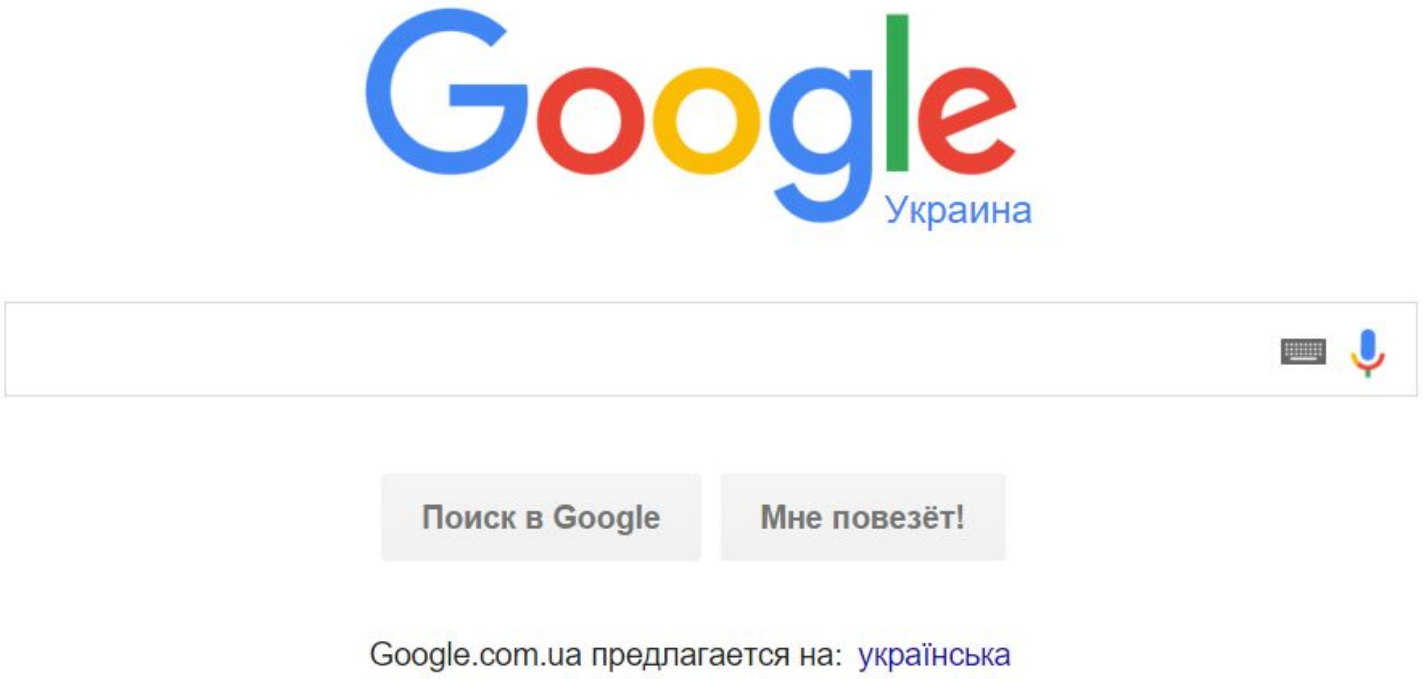
Инструменты: Браузер Chrome

Инструменты: Консоль разработчика в браузере (клавиша F12)



Подробная информация о том, как бразуер «понимаем» созданный вами код.

Инструменты: http://google.com



Презентация доступна по адресу:

js.courses.dp.ua/express

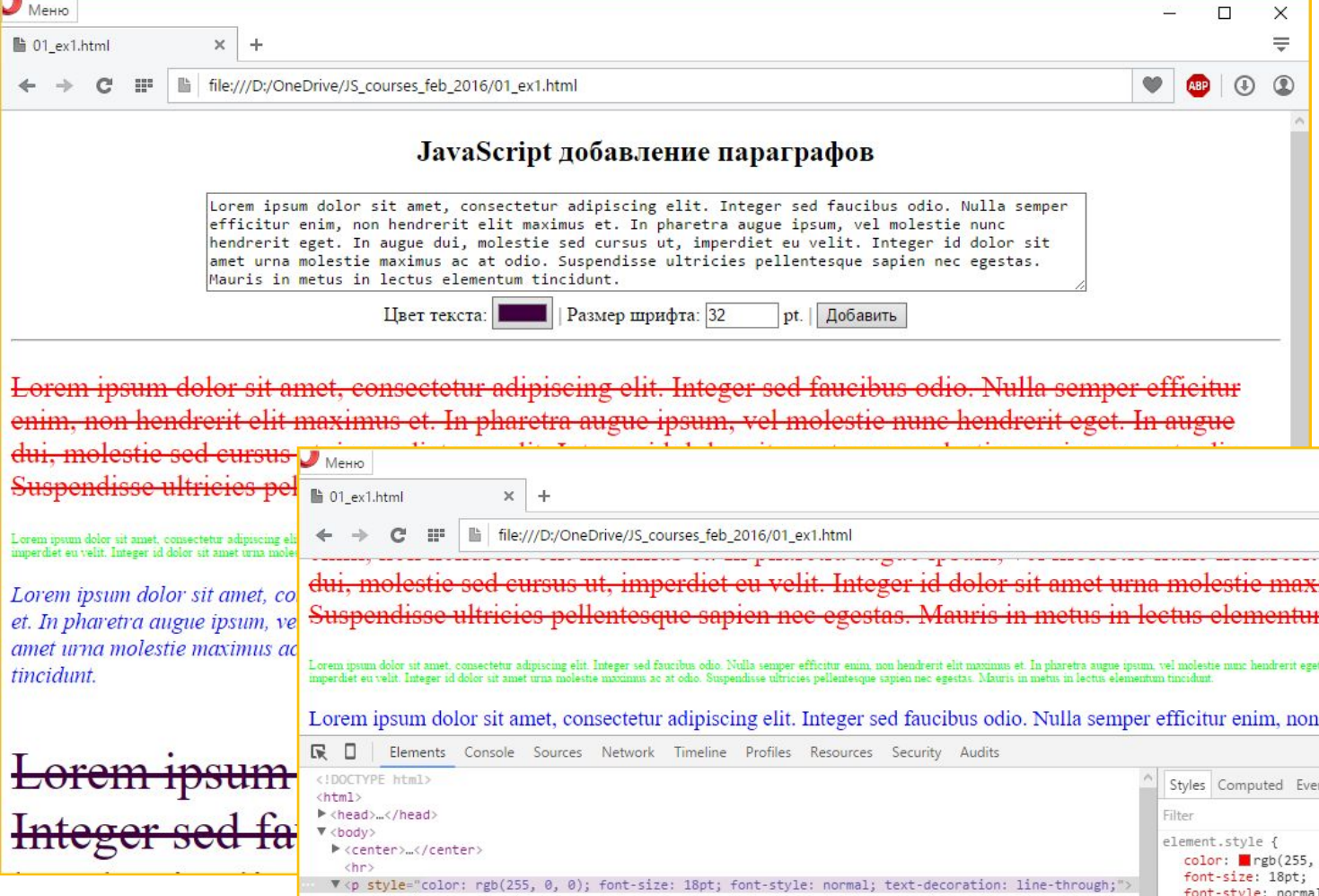
vk.com/js.express

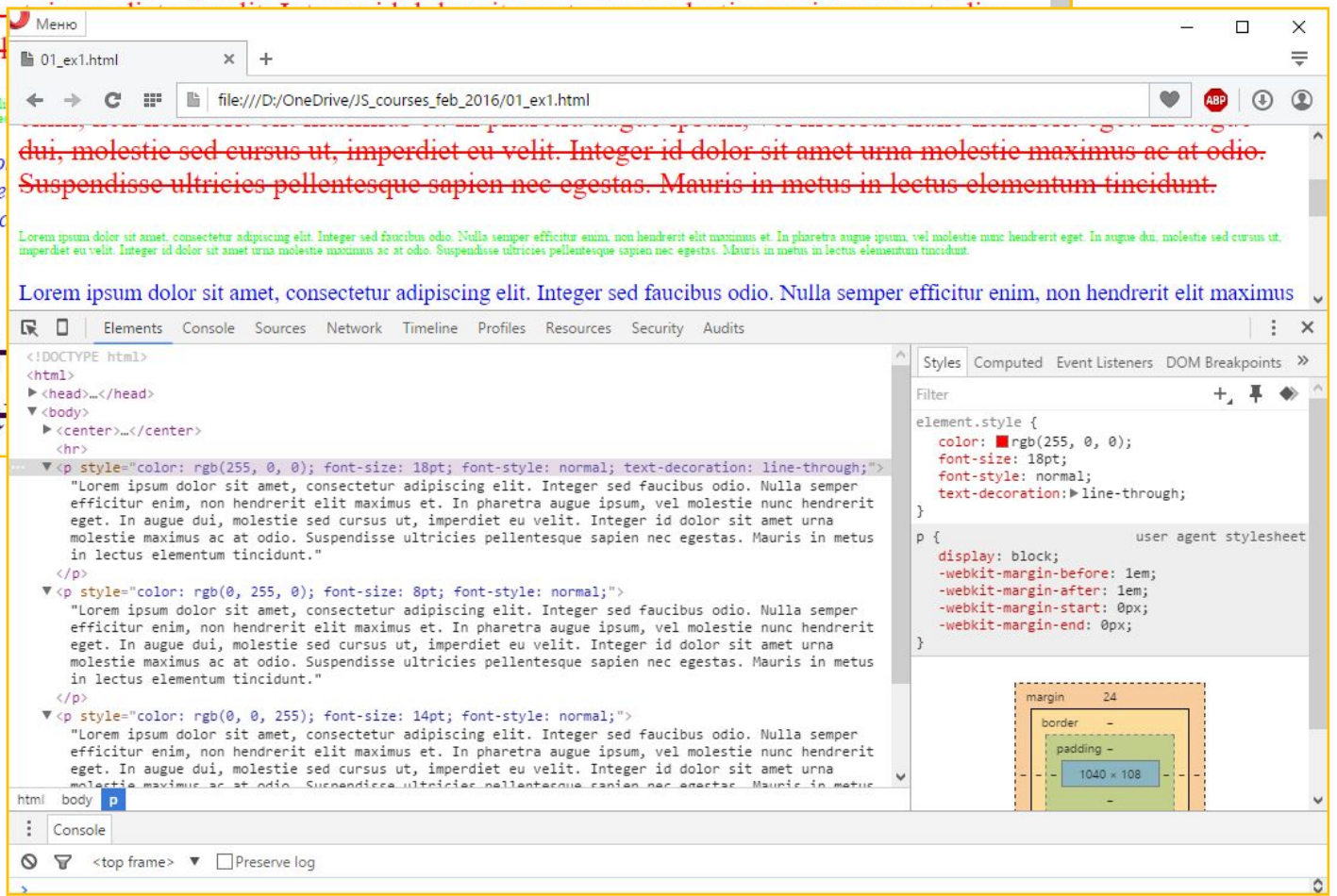
Группа для вопросов, обсуждений, объявлений (и презентации там тоже будут).

#Первый пример

```
<!DOCTYPE html>
<html>
<head>
<script>
    window.onload = function(){ //Функция выполниться когда документ будет полностью загружен.

        document.querySelector("button").onclick = function(){ //Функция будет выполнена при клике по кнопке "Добавить".

            var new_element                 = document.createElement("p"); //Создаём новый элемент (тег) <p>
                new_element.innerHTML       = document.querySelector("textarea").value; //Заполняем тег содержимым поля ввода.
                new_element.style.color     = document.querySelector("#color").value; //Задаём цвет текста, на основе поля "Цвет текста".
                new_element.style.fontSize  = document.querySelector("#fontsize").value + "pt"; //Задаём размер шрифта на основе поля "Размер шрифта".

                new_element.onclick         = function(event_args){ //При клике по элементу...
                    event_args.target.style.textDecoration = "line-through"; //...ставим стиль "зачеркнутый".
                };

                new_element.ondblclick      = function(event_args){ //При двойном клике по элементу
                    event_args.target.remove(); //...удаляем элемент.
                };

                new_element.onmouseover     = function(event_args){  //При наведении курсора на элемент...
                    event_args.target.style.backgroundColor = '#E0E0E0'; //...делаем элемента серым.
                };

                new_element.onmouseleave    = function(event_args){ //При уходе курсора с элемента...
                    event_args.target.style.backgroundColor = '#FFFFFF'; //...ставим фон обратно белым.
                };

            document.body.appendChild(new_element); //Добавляет новосозданный элемент в тего документа (в тег <body>).

        }
    }
</script>
</head>
<body>
    <center>
        <h2>JavaScript Text Processor</h2>
        <textarea cols='100' rows='5'></textarea>
        <div>
            Цвет текста: <input type='color' id='color'>
            | Размер шрифта: <input type='number' id='fontsize' min='8' max='48' value='18'> pt.
            | <button>Добавить</button>
        </div>
    </center>
    <hr>
</body>
</html>
```

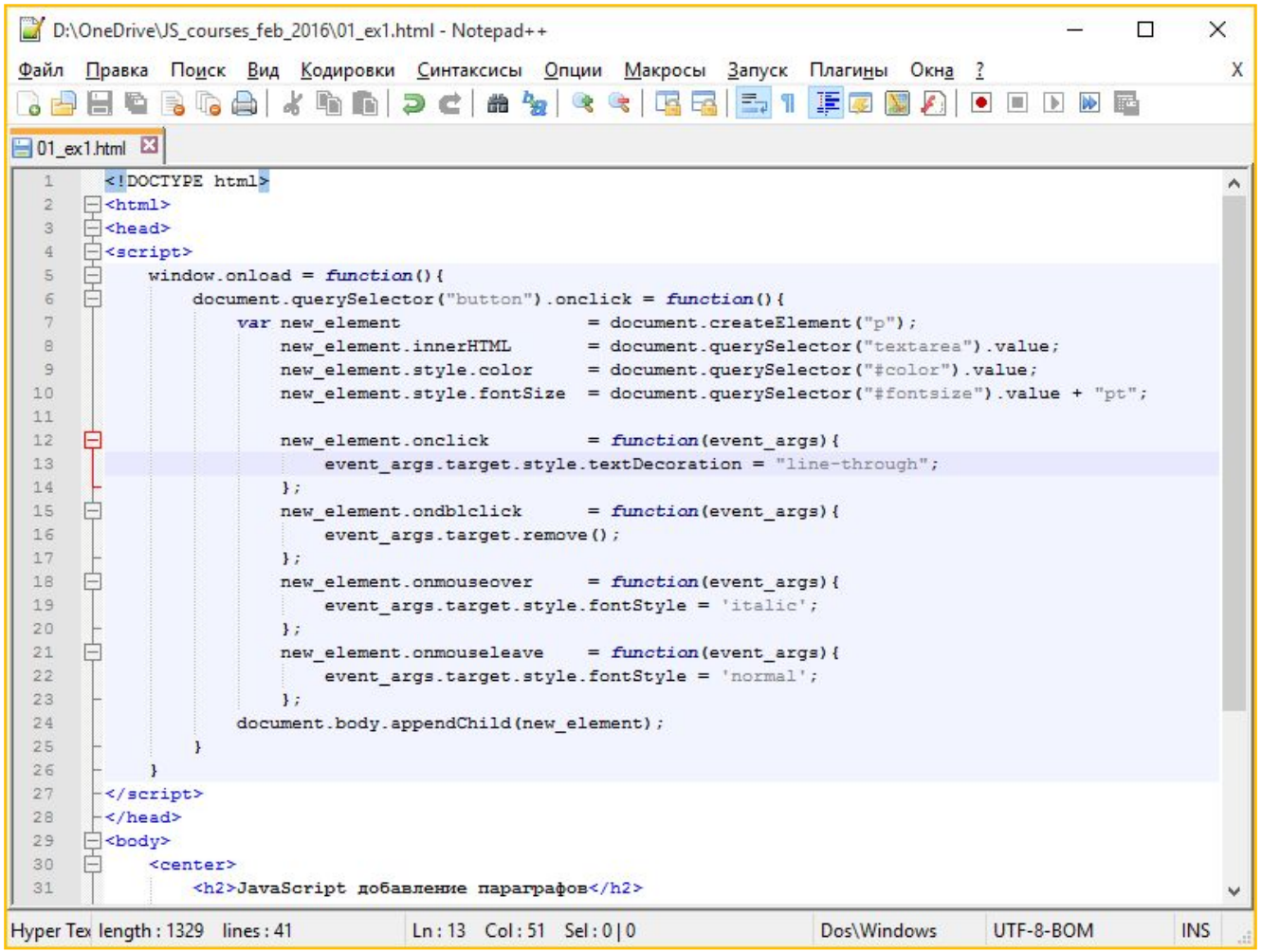
http://js.courses.dp.ua/express/01/jsexample.html





#Первый пример

Инструменты: Notepad++



Инструменты: служебные функции

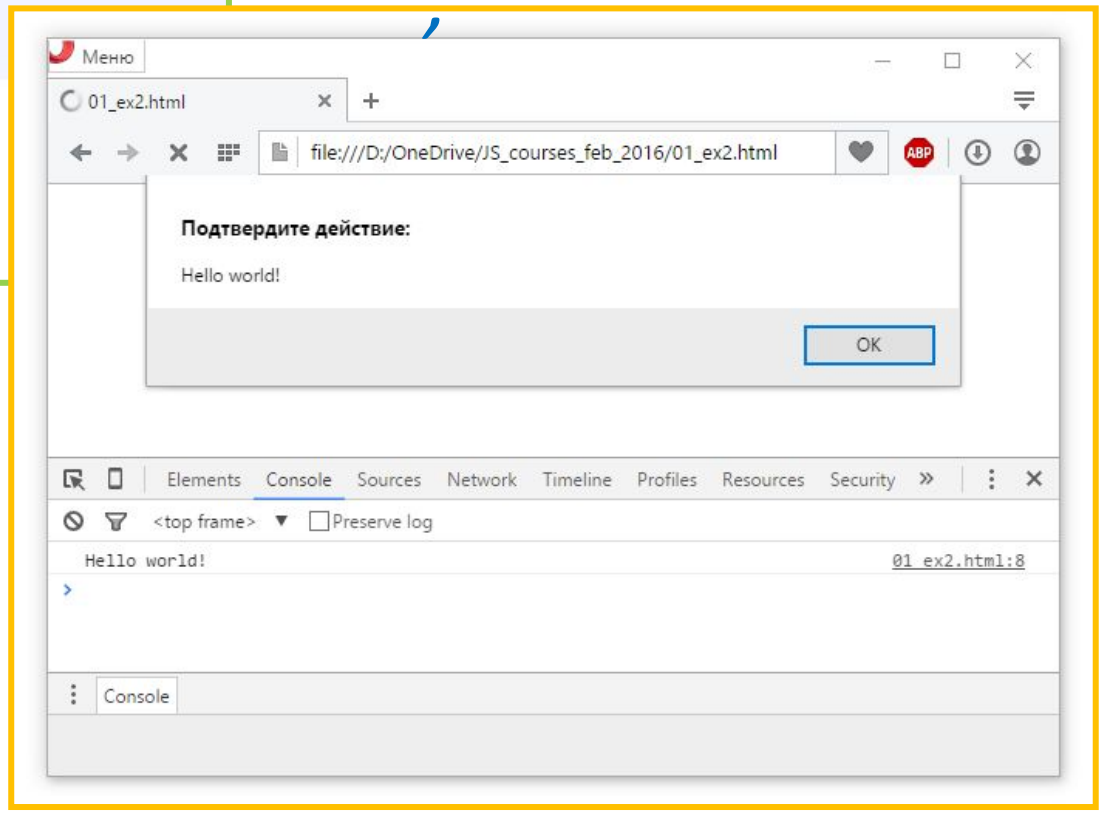
```
1   <!DOCTYPE html>
2   <html>
3   <head>
4   <script>
5
6       var message = "Hello world!";
7
8       console.log(message);
9
10      alert(message);
11
12  </script>
13  </head>
14  <body>
15  </body>
16  </html>
```

console.log(…)

;

alert(…)

;

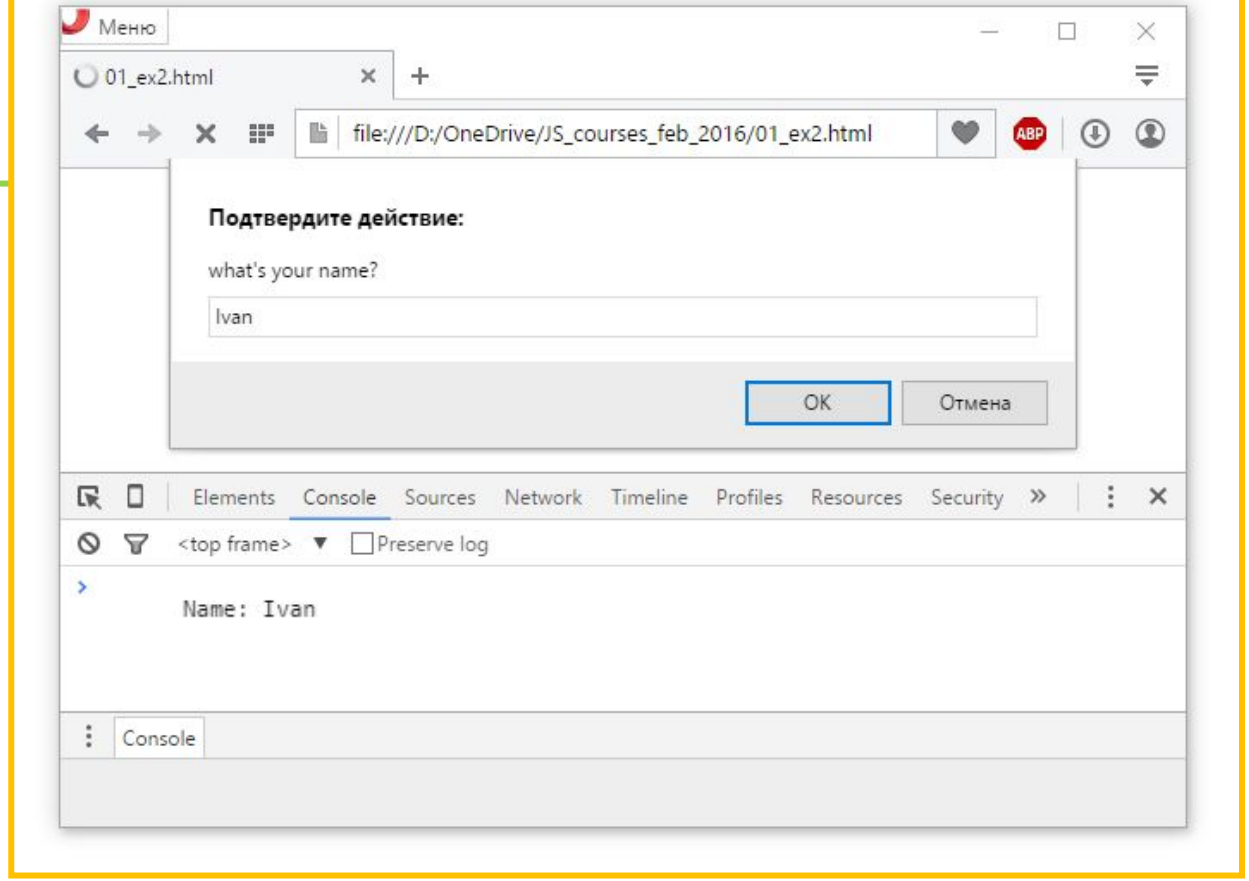


Инструменты: служебные функции

```
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(…)

.

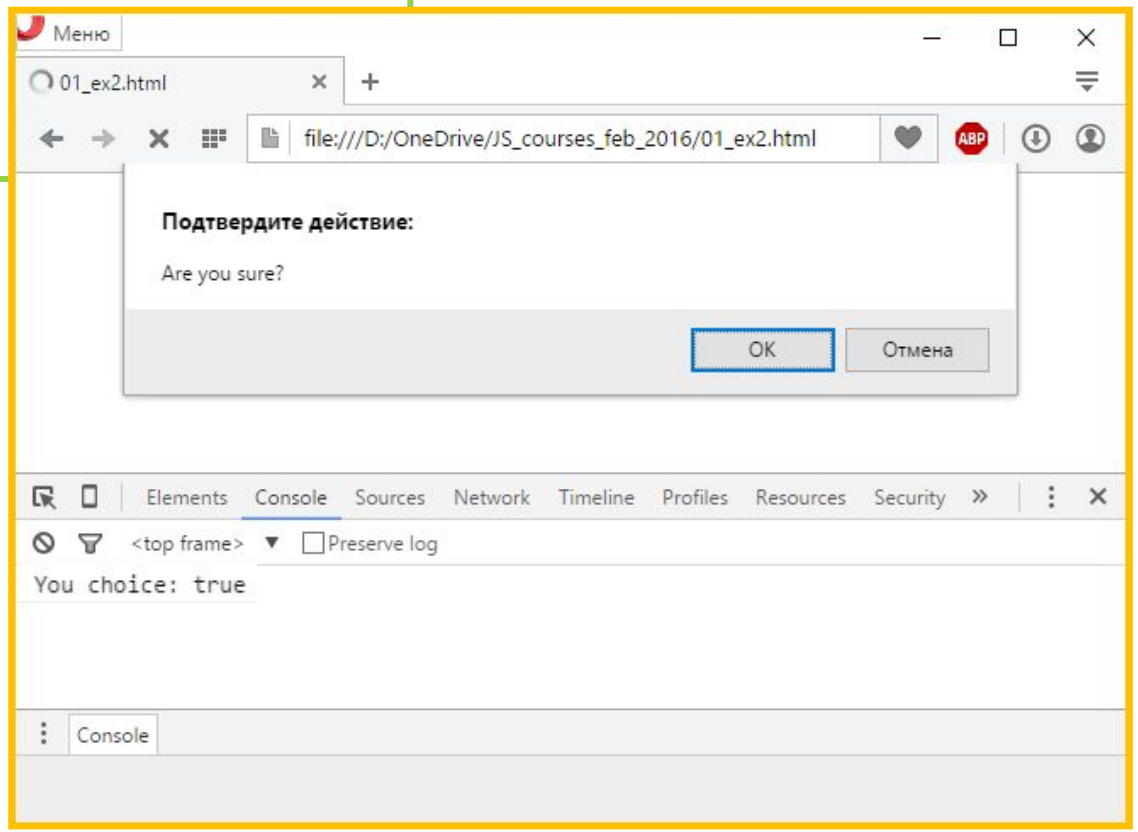


Инструменты: служебные функции

```
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(…)

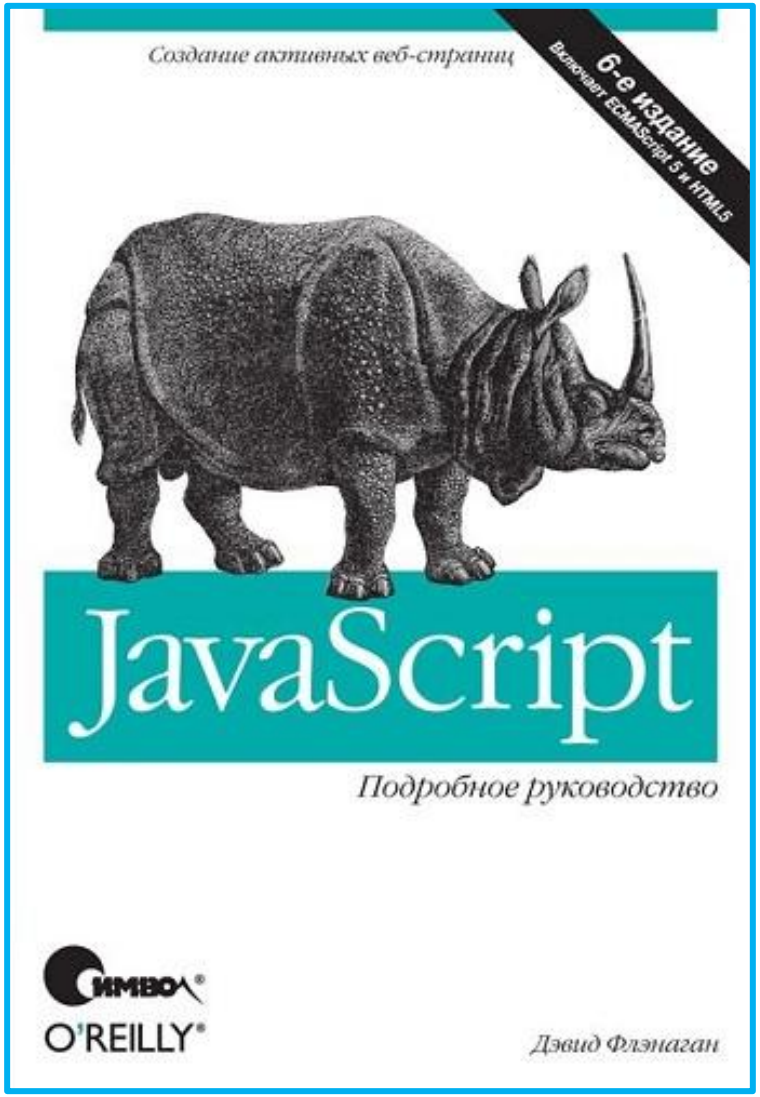
;



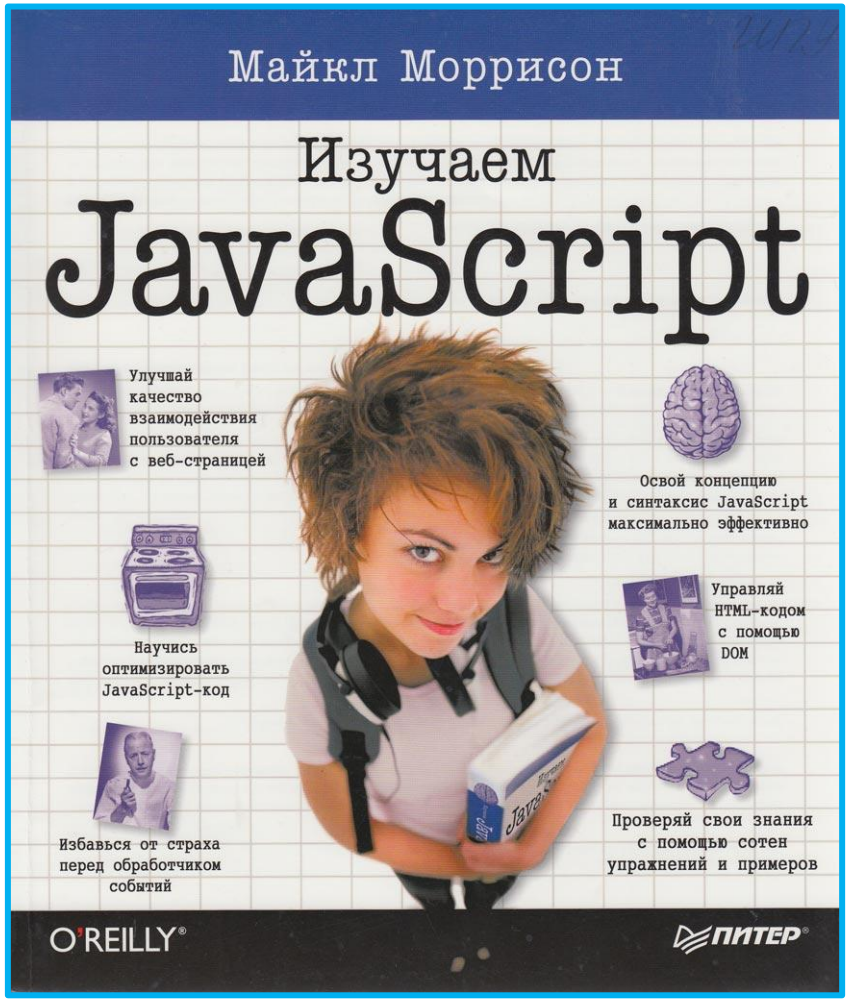
Инструменты: «допустимый» синтаксис

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

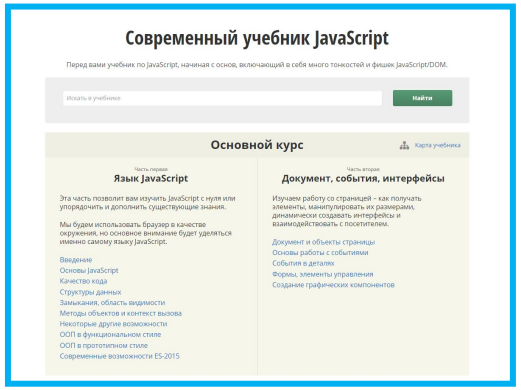
JavaScript. Подробное руководство, 6-е издание (2012)



Изучаем JavaScript, (2012)

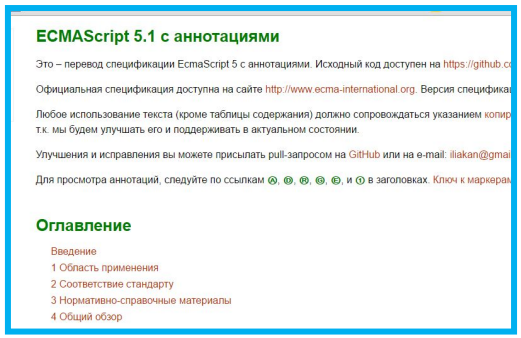


JavaScript.ru



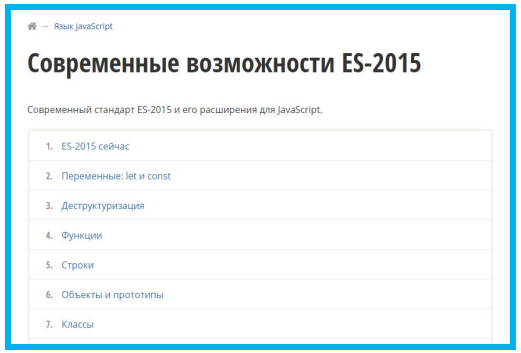
Современный учебник JavaScript

http://learn.javascript.ru/



ECMAScript 5.1 с аннотациями

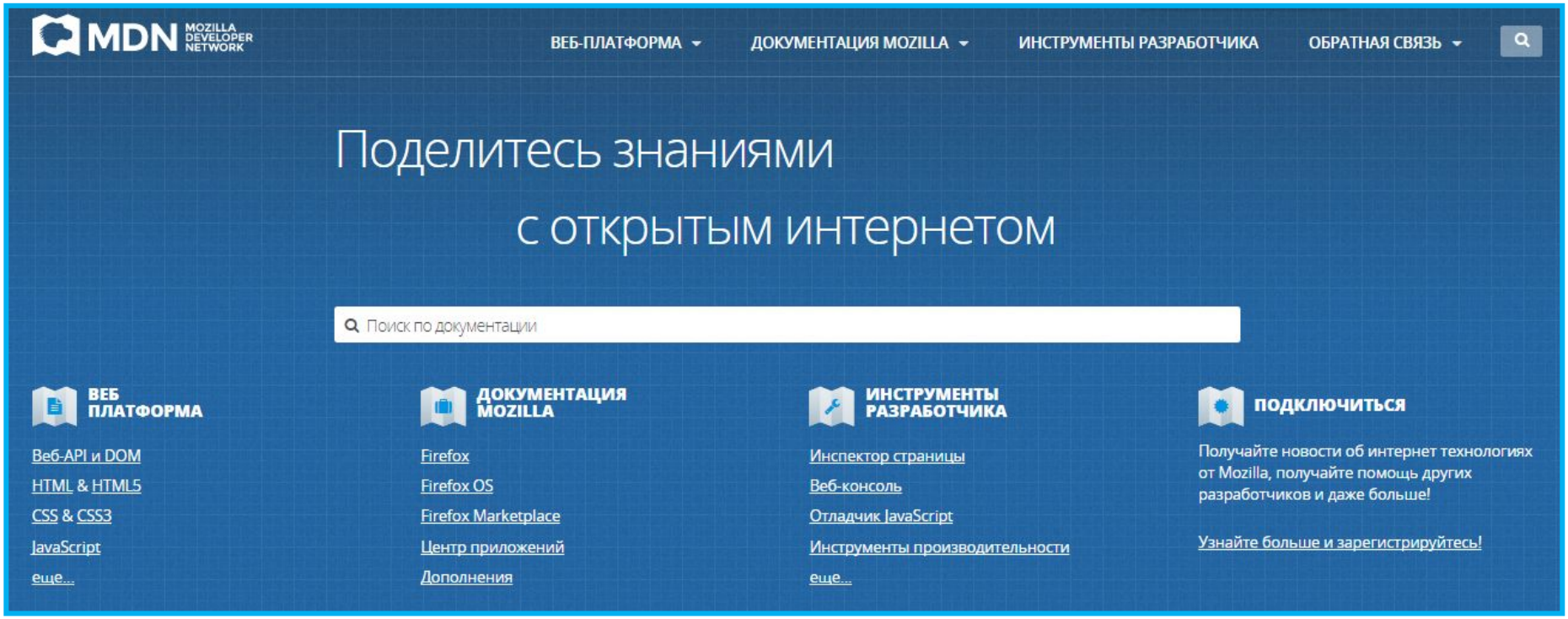
http://es5.javascript.ru/



Что нового в ES2015

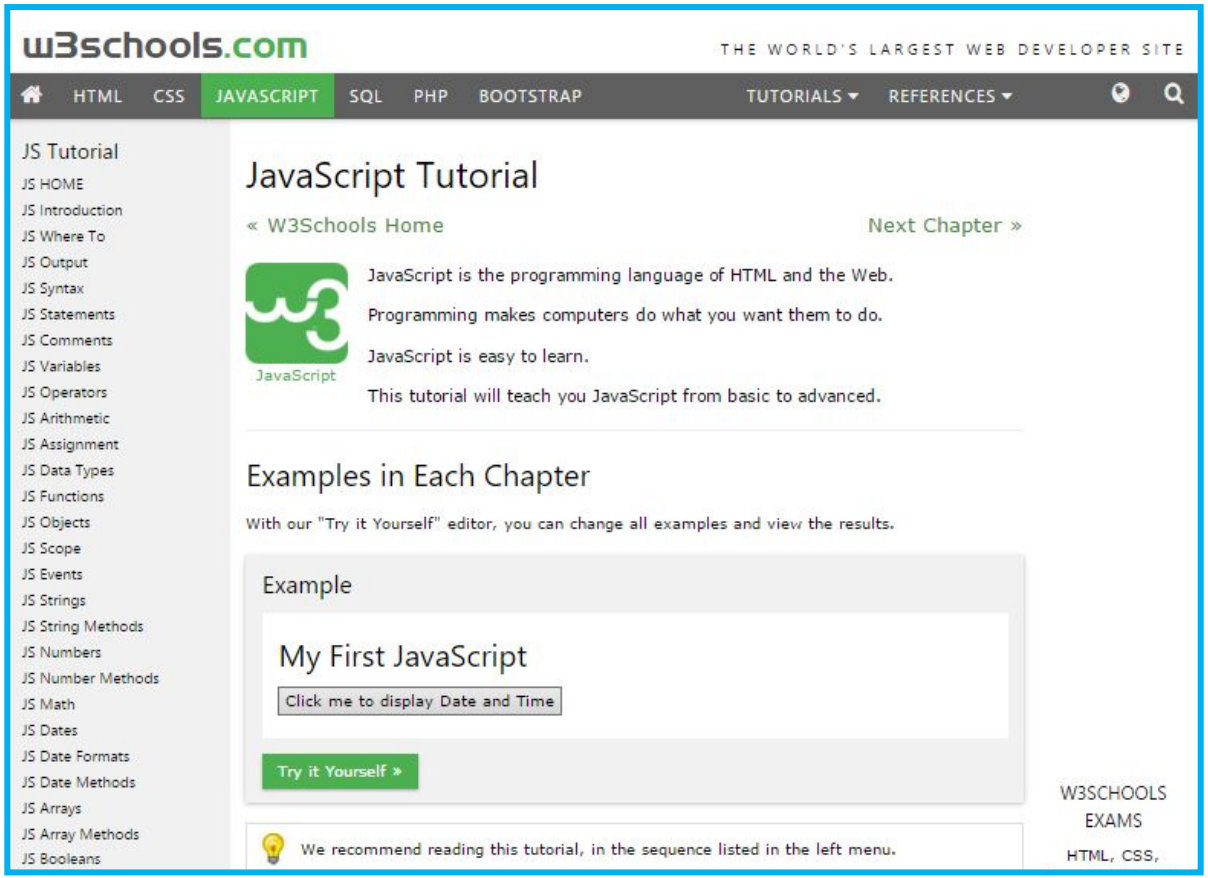
http://learn.javascript.ru/es-modern

Mozilla Developer Network. Энциклопедия JavaScript



https://developer.mozilla.org

W3schools JavaScript Tutorial



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

И

Объекты

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                    ex2.html:7
35 "number"                     ex2.html:8
77 string                       ex2.html:12
```

*Ключевое слово **var** – используется при объявлении переменной. В **ECMAScript-2015** добавилось ключевое слово **let**, основное отличие в области видимости переменной обвяленной 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
. [ ] ( )	Доступ к полям, индексация массивов, вызовы функций и группировка выражений
++ -- - ~ ! delete new typeof void	Унарные операторы, тип возвращаемых данных, создание объектов, неопределенные значения
* / %	Умножение, деление, деление по модулю
+ - +	Сложение, вычитание, объединение строк
<< >> >>>	Сдвиг битов
< <= > >= instanceof	Меньше, меньше или равно, больше, больше или равно, instanceof
== != === !==	Равенство, неравенство, строгое равенство, строгое неравенство
&	Побитовое И
^	Побитовое исключающее ИЛИ
|	Побитовое ИЛИ
&&	Логическое И
||	Логическое ИЛИ
?:	Условный оператор
= *OP*=	Присваивание, присваивание с операцией (например += и &=)
,	Вычисление нескольких выражений

https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/Operator_Precedence

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** относиться 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** которое 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** которое 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

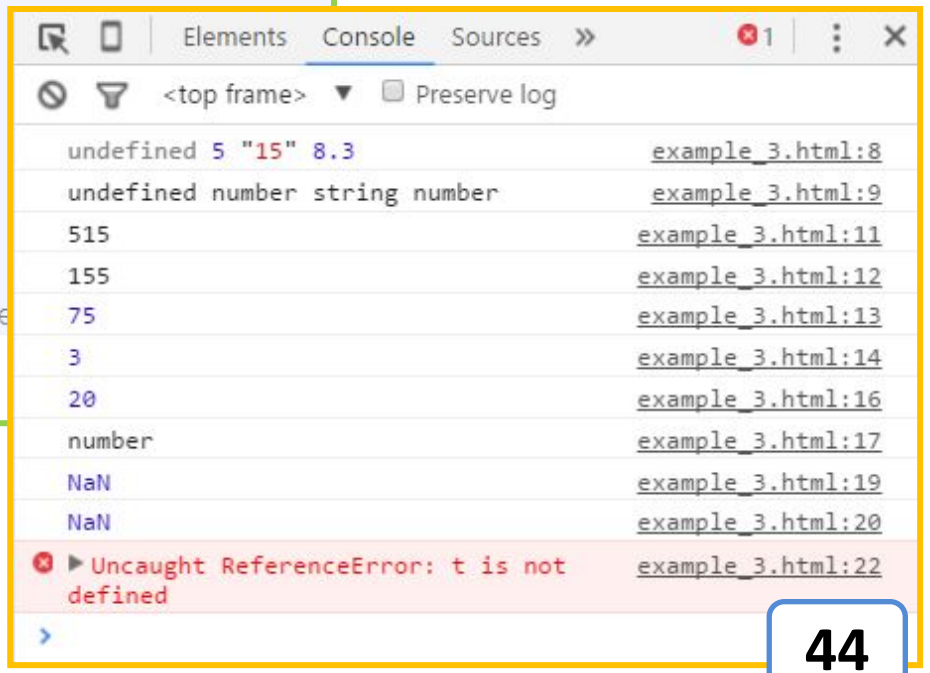
```
1  int a;
2  double b;
3  char c;
```

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?

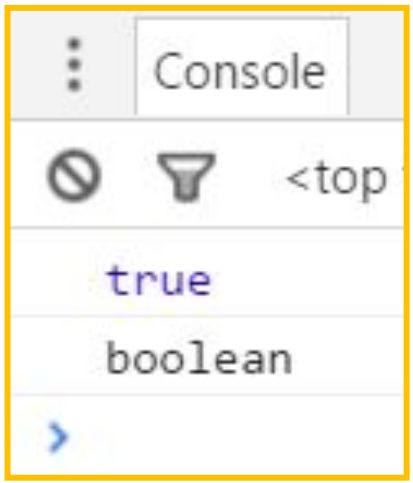
```
1  <script>
2
3      var a = prompt("Введие любое число от 1 до 100");
4
5      console.log(a);
6
7      console.log(typeof(a));
8
9  </script>
```

*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

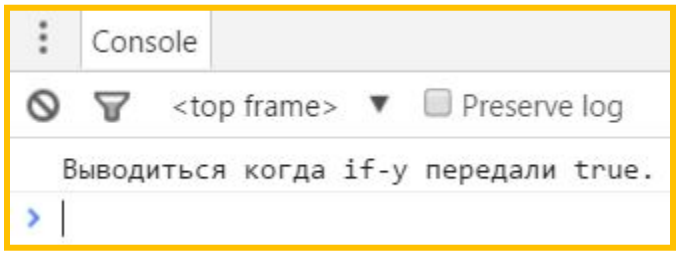
```
1  <script>
2
3      var a = true;
4
5      var b = false;
6
7      console.log(a);
8
9      console.log(typeof(a));
10
11 </script>
```



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

Оператор if-else

```
<script>

    var bool_type_variable = true; //или false

    if( bool_type_variable ) {

        console.log("Выводиться когда if-у передали true.");

    }else{

        console.log("Выводиться когда if-у передали false.");
    }
</script>
```

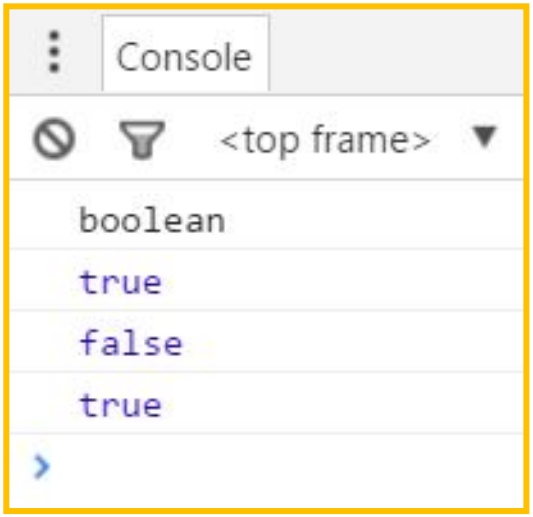


*Оператор **if-else** в зависимости от переданного (**true** или **false**) значения выполняет один из двух блоков кода (**первый** или **второй**, соответственно), другой блок при этом не выполняется.*

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?

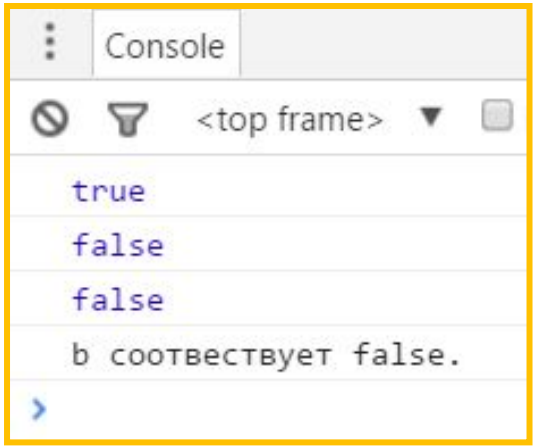
*undefined, NaN => **false**;*

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

***String:** “” => **false**; все остальные => true;*

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

```
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 соотвествует true.");
13     }else{
14         console.log("b соотвествует false.");
15     }
16
17 </script>
```



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>
```

?!?

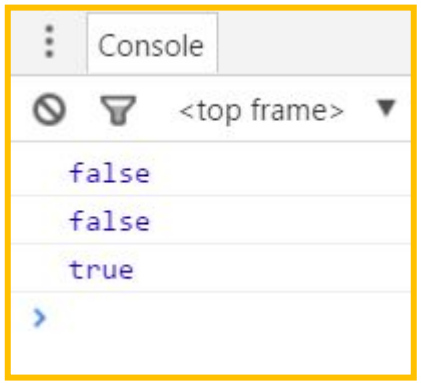
Сложности с типами

Откуда берётся boolean?

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

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

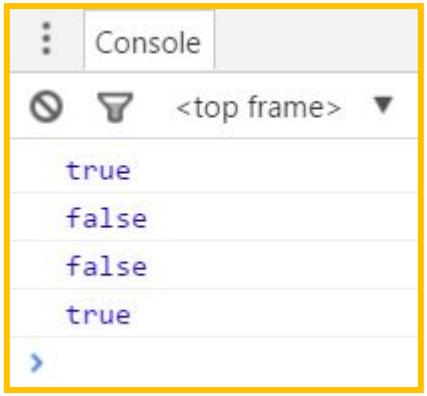
```
1  <script>
2
3      var a = "Ivan";
4      var b = "Iven";
5
6      console.log(a == b);
7      console.log(a > b);
8      console.log(a < b);
9
10 </script>
```



Откуда берётся boolean?

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

```
<script>

    var a = "5";
    var b = 5;

    console.log(a == b);

    console.log(a === b);

    console.log((typeof(a) == typeof(b)) && (a == b));

    console.log(typeof(a) == "string");

</script>
```



Сравнение с учетом типа

Откуда берётся boolean?

```
<script>

    var user_email = prompt("Enter email: ");

    var is_email_correct = user_email.match(/\S+@\S+\.\S+/);

    if(is_email_correct){

        alert("email корректный.");

    }else{

        alert("email не корректный!");
    }

</script>
```

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

Логические операторы

&&	||	!

```
<script>
    //...

    if( (user_name == "ivan") && (user_password == "12345")){
        //...
    }

    //...

    if( (user_height > 145) || (user_age > 12) ){
        //...
    }
    //..

    var is_blocked = true;

    if( !is_blocked ){
        //...
    }
</script>
```

Когда нужны «сложные» условия

Логические операторы

&&	||	!

&&	False	True
False	False	False
True	False	True

||	False	True
False	False	True
True	True	True

!	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

```
<script>

    var secret_key = "123";
    var user_key;

    do{

        user_key = prompt("Введите пароль: ");

        if(secret_key != user_key) alert("Пароль не верный!");


    }while(secret_key != user_key);

    alert("Пароль правильный!");

</script>
```

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

while, do/while

```
<script>

    var a = 3;

    var b = 7;

    while(a < b){

        //Какие-то действия...
        a++; //!!!
    }

    console.log(a);

</script>
```

```
<script>

    var a = 3;

    var b = 7;

    do{
        //Какие-то действия...
        a++; //!!!

    }while(a < b);

    console.log(a);

</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" ]
```

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

```
<script>
    var arr = [1, 3, "Elena", true];

    console.log(arr);
    console.log(arr.length);
    //Свойство указывающее длинну массива.
    console.log(typeof(arr));

    console.log(arr[0], typeof(arr[0]));
    console.log(arr[2], typeof(arr[2]));
    console.log(arr[3], typeof(arr[3]));

    arr[0] = "Ivan";
    console.log(arr[0], typeof(arr[0]));

    arr.push(777);
    //Добавление элемента в конец массива
    console.log(arr, arr.length);

    var x = arr.pop();
    //Удаление последнего элемента из массива
    console.log(arr, arr.length);
    console.log(x);

    arr.unshift("Julia");
    //Добавление элемента в начало массива
    console.log(arr, arr.length);

    arr.shift();
    //Удаление первого элемента массива
    console.log(arr, arr.length);

</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

```
<script>

    var mas = [5,34,"computer", true, "phone", 77.355];

    for(var i = 0; i < mas.lenght; i++){
        console.log(i, mas[i]);
    }

</script>
```

```
0 5                 exmaple.html:6
1 34                exmaple.html:6
2 "computer"        exmaple.html:6
3 true              exmaple.html:6
4 "phone"           exmaple.html:6
5 77.355            exmaple.html:6
```

*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          ex 02.html:7
1: true        ex 02.html:7
2: Polina      ex 02.html:7
3: 77.88       ex 02.html:7
```

*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 рано **i** после выполнения цикла?*

Область видимости переменных – глобальная, переменная обвяленная 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

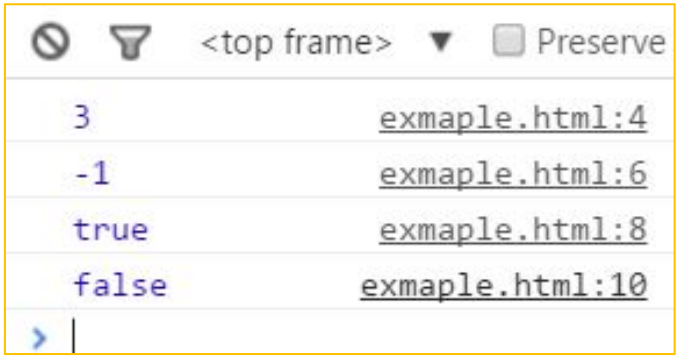
```
1  <script>
2
3      var arr = [5, 77, 23];
4
5      console.log("Length: ", arr.length);
6      for(var i = 0; i < arr.length; i++){
7          console.log(i + " : " + arr[i]);
8      }
9
10     arr[7] = 89;
11
12     console.log("Length: ", arr.length);
13     for(var i = 0; i < arr.length; i++){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

```
Length:  3                    ex 02.html:5
0 : 5                         ex 02.html:7
1 : 77                        ex 02.html:7
2 : 23                        ex 02.html:7
Length:  8                    ex 02.html:12
0 : 5                         ex 02.html:14
1 : 77                        ex 02.html:14
2 : 23                        ex 02.html:14
3 : undefined                 ex 02.html:14
4 : undefined                 ex 02.html:14
5 : undefined                 ex 02.html:14
6 : undefined                 ex 02.html:14
7 : 89                        ex 02.html:14
>
```

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

Массивы это коллекция пар «ключ => значение».

```
<script>
    var arr = [1, 77, "Elena", 55.6, true];

    console.log(arr.indexOf(55.6));

    console.log(arr.indexOf("lalala"));

    console.log(3 in arr);

    console.log(7 in arr);
</script>
```



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

```
<script>

    var mas = [];

    mas["name"]     = "Ivan";
    mas["age"]      = 19;
    mas["height"]   = 1.81;
    mas["smoke"]    = false;

    console.log(mas);

    console.log(mas.name, mas["name"]);

    console.log(mas.length);

    console.log(Object.keys(mas));

    console.log(Object.keys(mas).length);

    mas.push("55");

    console.log(mas, mas.length);

</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                    exmaple.html:6
1 8                    exmaple.html:6
2 33                   exmaple.html:6
3 text                 exmaple.html:6
4 true                 exmaple.html:6
5 77.8                 exmaple.html:6
```

*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                    exmaple.html:10
age 18                        exmaple.html:10
smoke false                   exmaple.html:10
```

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

Цикл for/in и свойства HTML элементов

```
<!DOCTYPE>
<html>
<body>
    <h1>This is a text.</h1>
    <script>
        var element = document.querySelector("h1");

        for(var property_name in element){
            console.log(property_name, ":" ,element[property_name]);
        }
    </script>
</body>
</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

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

Объекты как ассоциативный массив

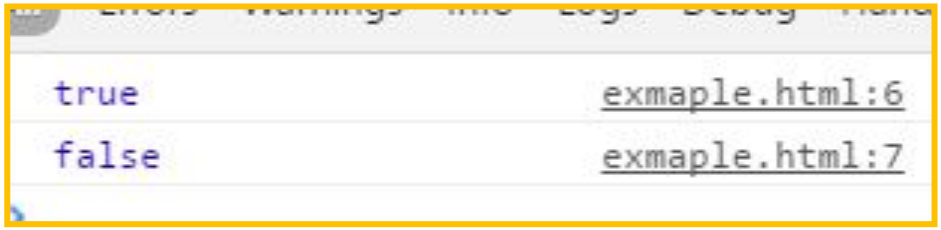
```
1  <script>
2
3      var mas = {name: "Ivan", age: 22};
4      console.log(mas);
5
6      for(var x in mas){
7          console.log(x, mas[x]);
8      }
9
10     mas["name"] = "John";
11     console.log(mas.name);
12
13     mas.push(55);
14
15 </script>
```

```
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       exmaple.html:15
not a function
```

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

Как отличить массив от объекта

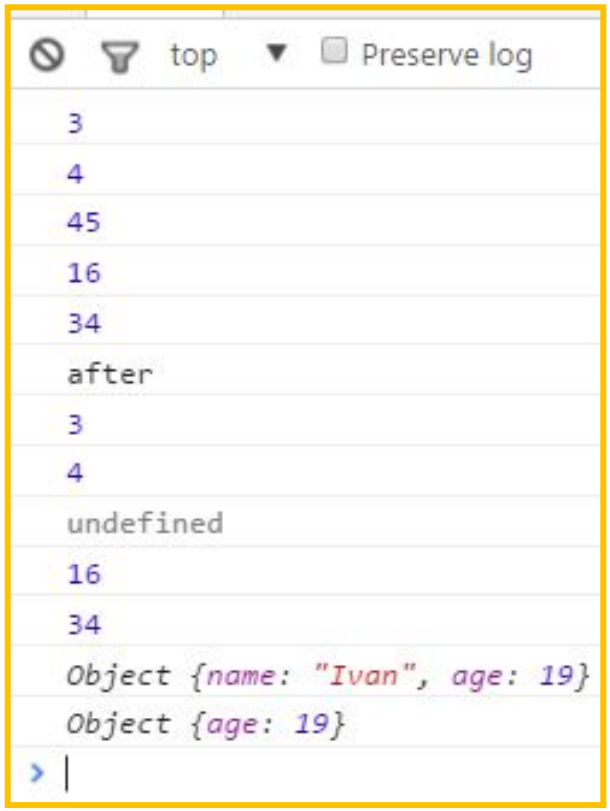
```
1  <script>
2
3      var arr1 = [1,2,3];
4      var arr2 = { title: "the news", year: 1999 };
5
6      console.log(Array.isArray(arr1));
7      console.log(Array.isArray(arr2));
8
9  </script>
```



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

Удаление элементов массива

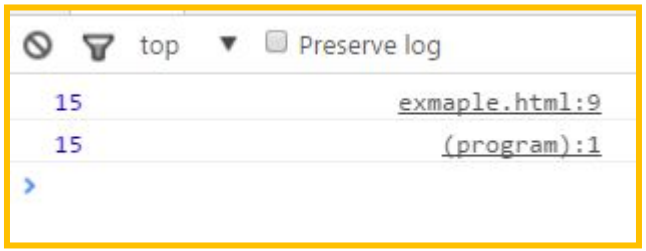
```
<script>

    var arr = [3,4,45,16,34];

    for(var i = 0; i < arr.length; i++){
        console.log(arr[i]);
    }

    delete arr[2];
    console.log("after");

    for(var i = 0; i < arr.length; i++){
        console.log(arr[i]);
    }

    //-----------------------//

    var mas = { name: "Ivan", age: 19 };

    console.log(mas);

    delete mas.name;

    console.log(mas);

</script>
```



Удаление элементов массива по сути образовывает «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 отличии от ассоциативных массивов, ключом может выступать не только строка но и другие типы данных.*

```
<script>

    var map = new Map();

    map.set('1', 'Ivan');
    map.set(1, 'Elena');
    map.set(true, 123);


    console.log(map.get(1));
    console.log(map.get('1'));
    console.log(map.get(false));

    console.log("Map size: " + map.size);

</script>
```

```
Elena                                   ex 02.html:10
Ivan                                    ex 02.html:11
undefined                               ex 02.html:12
Map size: 3                             ex 02.html:14
```

Структуры данных ECMAScript-2015

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

```
<script>

    var set = new Set();

    set.add(55);
    set.add(true);
    set.add("Elena");
    set.add(55);
    set.add(true);

    console.log("Size :" + set.size);

    console.log(set.has(44));
    console.log(set.has(true));

    set.delete(true);

    console.log(set.has(true));

</script>
```

```
Size :3                    ex 02.html:11
false                      ex 02.html:13
true                       ex 02.html:14
false                      ex 02.html:18
```

JSON http://www.json.org/json-ru.html

```
<script>

    var message = '{"name": "Vasya", "age": "19", "languages": ["English","Russian","French"] }';

    var person = JSON.parse(message);

    for(var x in person){
        console.log(x + ": " + person[x]);
    }

    console.log(person);

</script>
```

```
name: Vasya                                                    exmaple.html:8
age: 19                                                        exmaple.html:8
languages: English,Russian,French                              exmaple.html:8
▼ Object {name: "Vasya", age: "19", languages: Array[3]}       exmaple.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

```
<script>

    var arr = [3, 67, "Computer", 'IFC', true, null, {title:"Harry Potter", autor: "J.K.Rowling"}, 768.22];

    console.log(arr);

    var text_format = JSON.stringify(arr);

    console.log(text_format);

</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


```php
<?php
    header('Access-Control-Allow-Origin: *');

    $arr = array(
        array("name" => "Ivan", "age" => "23", "city" => "Kiev"),
        array("name" => "Irina", "age" => "20", "city" => "Lviv"),
        array("name" => "Marina", "age" => "27", "city" => "Kharkiv")
    );

    echo json_encode($arr);

?>
```

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 . А наш JS код декодирует данные, и наполняет ими страницу.

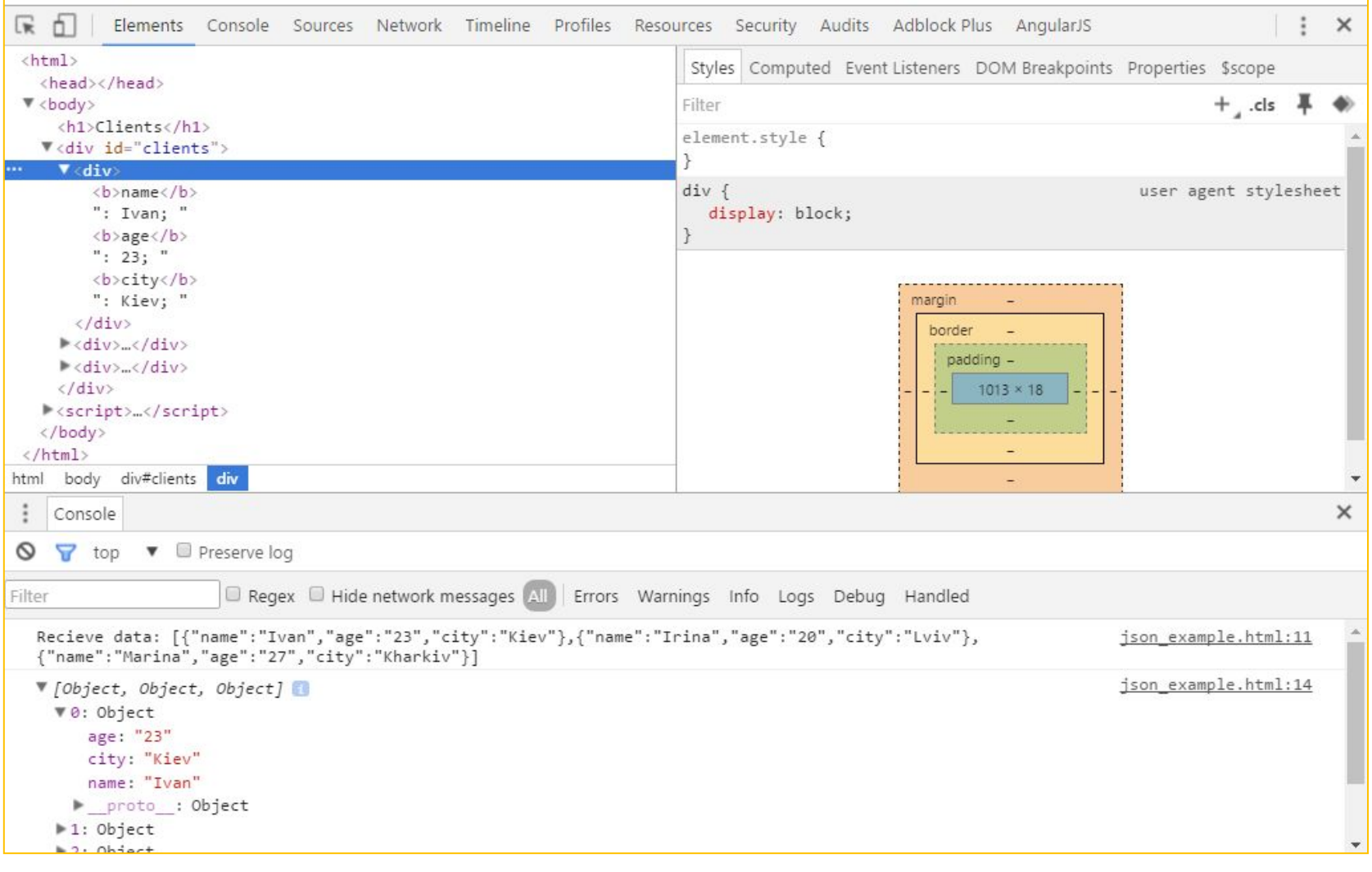
JSON в реальности

```
<html>
<head></head>
<body>
    <h1>Clients</h1>
    <div id="clients"></div>
    <script>
        var XHR = new XMLHttpRequest();
        XHR.onload = function(){

            var data = XHR.responseText;
            console.log("Recieve data: " + data);

            var clients_arr = JSON.parse(XHR.responseText);
            console.log(clients_arr);

            var clients_div = document.getElementById("clients");

            for(var id in clients_arr){
                var new_div = document.createElement("div");

                for(var field in clients_arr[id]){
                    new_div.innerHTML += "<b>" + field + "</b>: " +
                    clients_arr[id][field] + "; ";
                }
                clients_div.appendChild(new_div);
            }

        };
        XHR.open("get", "http://js.courses.dp.ua/files/get_clients.php",
        true);
        XHR.send();
    </script>
</body></html>
```

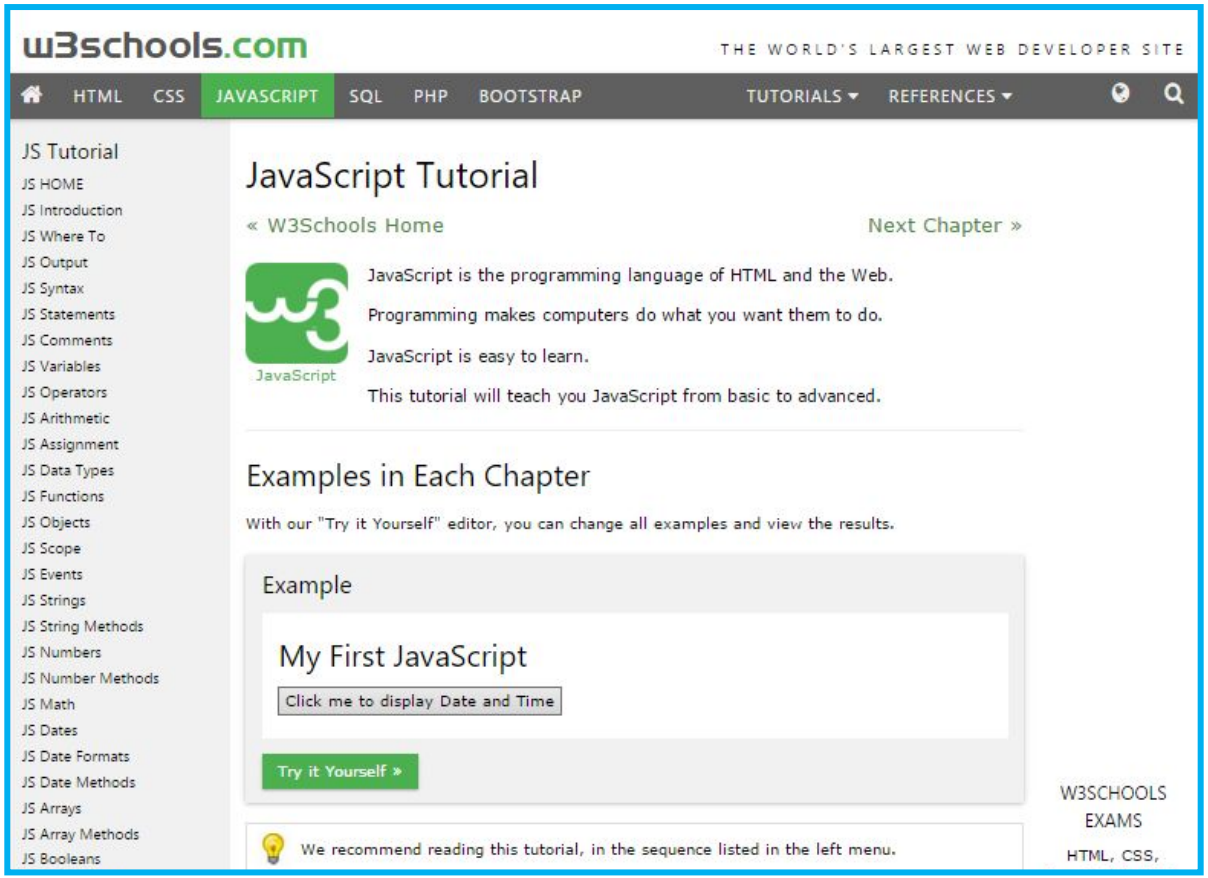
JSON в реальности

Clients

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



W3schools JavaScript Tutorial



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

Презентация доступна по адресу:

js.courses.dp.ua/express

vk.com/js.express

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