

HTML



CSS



HTML / CSS / JS

Краткий экскурс в мир frontend'a

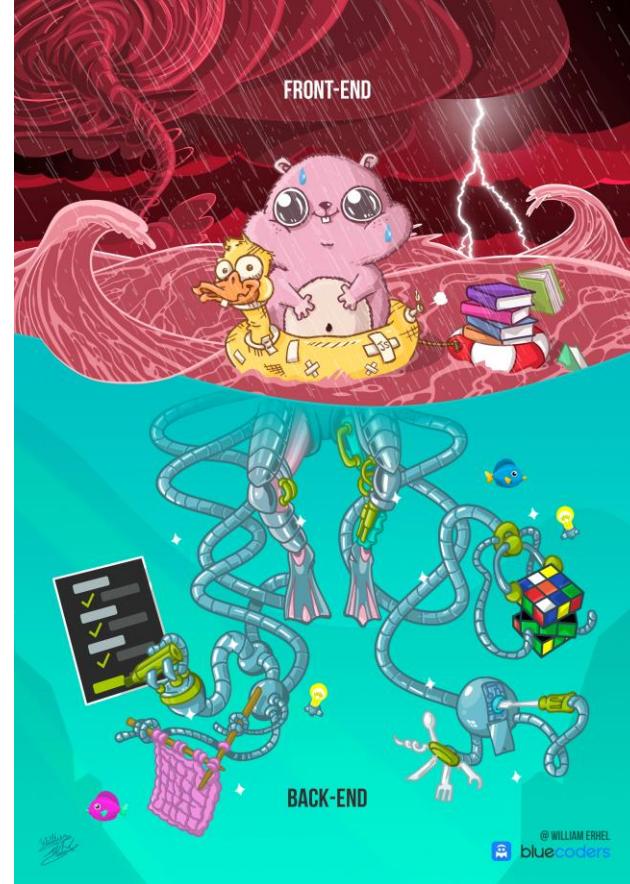
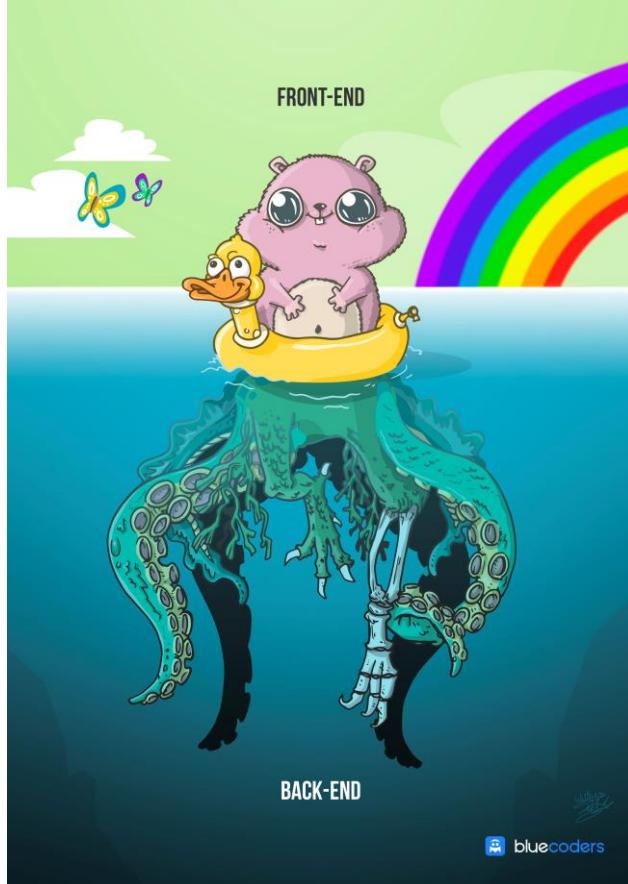
JavaScript



План

1. HTML – что это такое?
2. CSS. Наводим красоту.
3. JavaScript или ЯП «на коленке» за 10 дней.
4. Домашнее задание

Frontend



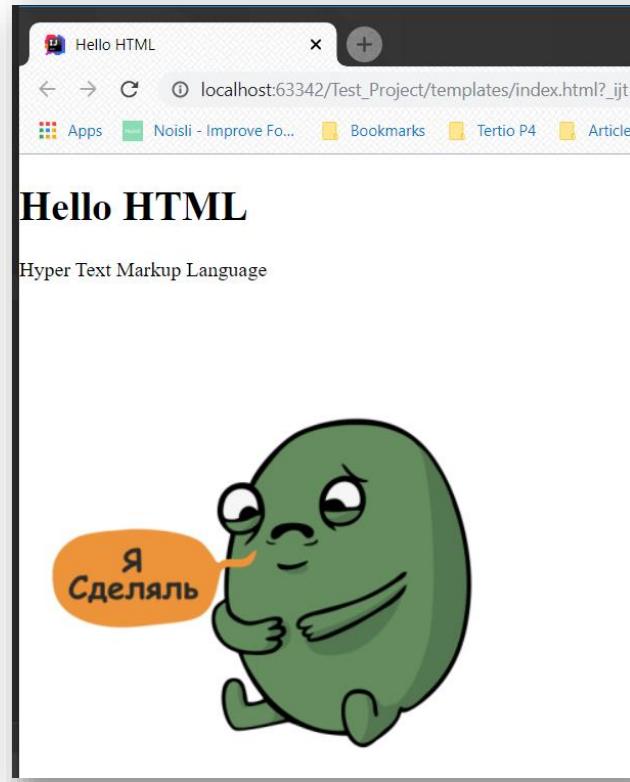
HTML



Структура HTML

```
<!doctype html>
<html lang="ru">
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8">
    <style type="text/css">
      body {
        padding: 0;
        margin: 0;
      }
    </style>
    <title>Hello HTML</title>
  </head>

  <body>
    <h1>Hello HTML</h1>
    <p>Hyper Text Markup Language</p>
    
  </body>
</html>
```



Структура HTML



Структура HTML

Void Element

```

```

Tag
Name

Attribute
Name

Attribute
Value

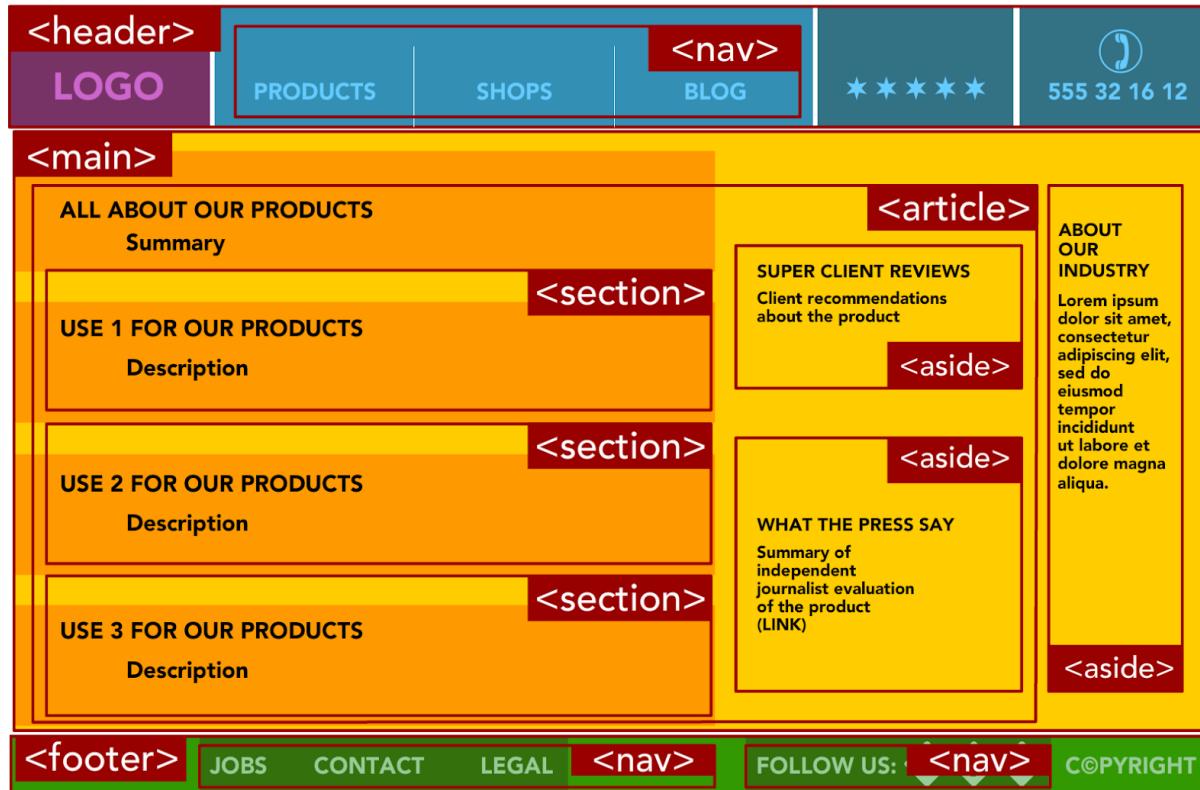
Attribute
Name

Attribute
Value

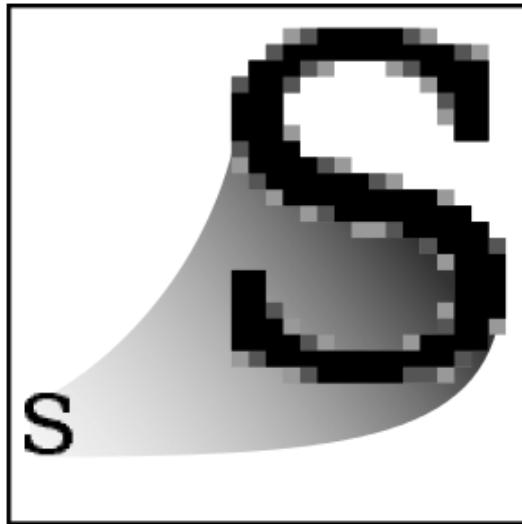
Структура HTML

Теги верхнего уровня	Теги заголовка документа	Структурные элементы: организация макета страницы	Текстовые элементы: определение контента	Строчные элементы: различный текст	Элементы общего назначения
<pre><!doctype html> <html> <head> <body></pre>	<pre><title> <meta> <link></pre>	<pre><header> <footer> <article> <section> <nav></pre>	<pre><h1> <h2> <h3> <p> <table></pre>	<pre><a> <q> <abbr> <small> <sub> <sup> <i>
</pre>	<pre><div> </pre>

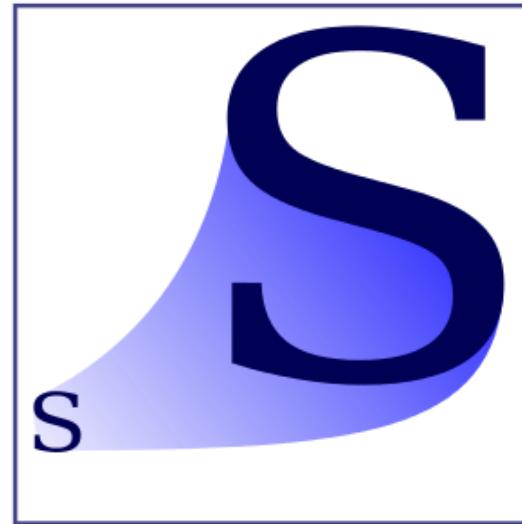
Структура HTML



Графика веб-страницы

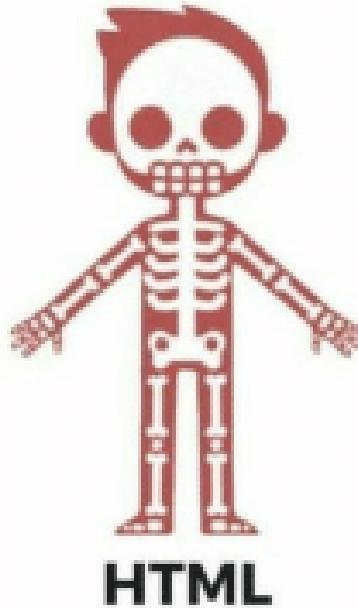


Raster
.jpeg .gif .png



Vector
.svg

HTML



CSS

Наследование в CSS – это способность элементов-потомков перенимать правила форматирования (свойства CSS), которые присвоены их предкам.
<http://htmlbook.ru/samcss/nasledovanie>

Каскадирование – одновременное применение разных стилевых правил к элементам документа — с помощью подключения нескольких стилевых файлов, наследования свойств и других методов. Чтобы в подобной ситуации браузер понимал, какое в итоге правило применять к элементу, и не возникало конфликтов в поведении разных браузеров, введены некоторые **приоритеты и специфичность**.

<http://htmlbook.ru/samcss/kaskadirovanie>

Подключение к HTML-документу

Внешние стили:

```
<!DOCTYPE html>
<html>
  <head>
    .....
    <link rel="stylesheet"
          type="text/css"
          href="style.css">
  </head>
  <body>
    .....
  </body>
</html>
```

```
<!DOCTYPE html>
<html>
  <head>
    .....
    <style media="all">
      @import url(style.css);
    </style>
  </head>
</html>
```

Подключение к HTML-документу

Внутренние стили:

```
<!DOCTYPE html>
<html>
  <head>
    .....
    <style>
      body {
        color: red;
      }
    </style>
  </head>
  <body>
    .....
  </body>
</html>
```

```
<!DOCTYPE>
<html>
  <head>
    .....
  </head>
  <body>
    <p style="font-size: 20px;
               color: green;
               font-family: arial, sans-
               serif">
      .....
    </p>
  </body>
</html>
```

Синтаксис CSS

```
селектор,  
селектор {  
    /* однострочный комментарий */  
    /* многострочный  
       комментарий */
```

свойство: значение;

свойство: значение;

свойство: значение;

}

Синтаксис CSS: селекторы

```

<body>

  <p>Lorem ipsum dolor sit amet</p>

  <p class="blueText uppercase bold">...</p>

  <p class="blueText">...</p>

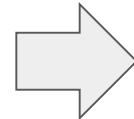
  <p>...</p>

  <p id="redText">...</p>

  <p id="greenText">...</p>

</body>

```



```

p {
  color: gray;
}

.blueText {
  color: blue;
}

.uppercase {
  text-transform: uppercase;
}

.bold {
  font-weight: bold;
}

#redText {
  color: red;
}

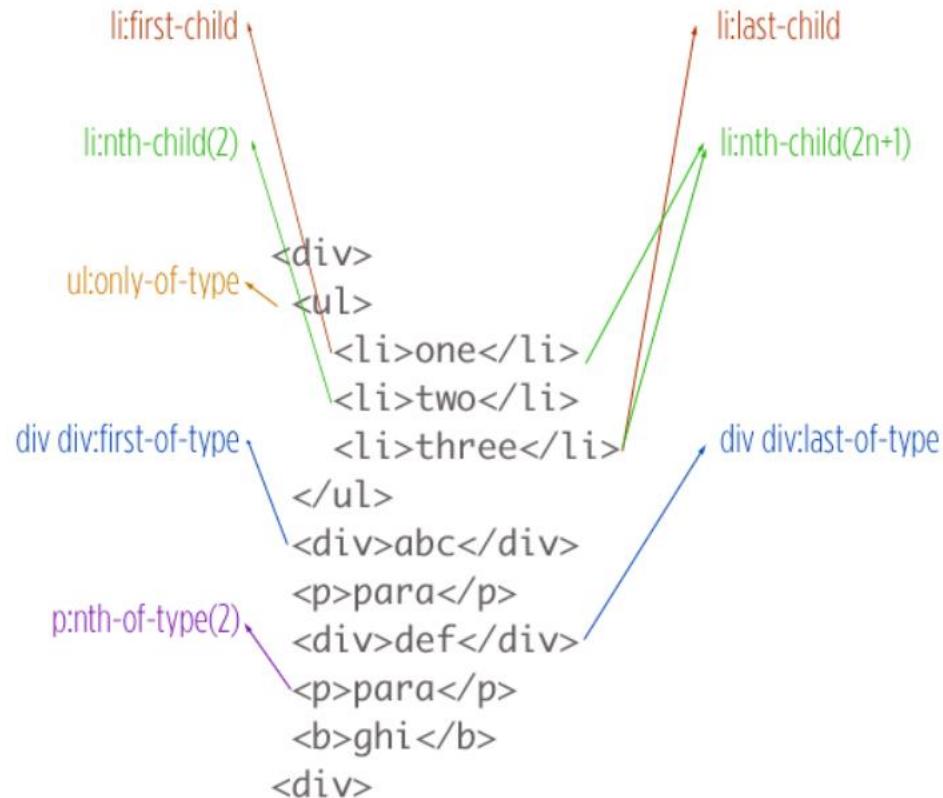
#greenText {
  color: green;
}

```

Синтаксис CSS: селекторы

Универсальный селектор	Селектор тегов	Селектор классов
<pre>* { margin: 0; padding: 0; }</pre>	<pre>p { font-family: arial, helvetica, sans-serif; }</pre>	<pre>.note { color: red; background-color: yellow; font-weight: bold; }</pre>
Селектор идентификаторов	Селектор атрибутов	Селектор потомков
<pre>#paragraph1 { margin: 0; }</pre>	<pre>a[href="http://www.somesite.com"] { font-weight: bold; }</pre>	<pre>div#paragraph1 p.note { color: red; }</pre>
Селектор дочерних элементов	Селектор элементов одного уровня	Селектор псевдоклассов и псевдоэлементов
<pre>p.note > b { color: red; }</pre>	<pre>h1 + p, h1 ~ p.bold { font-size: 24px; }</pre>	<pre>a:active, div::before { color: blue; }</pre>

Синтаксис CSS: псевдоклассы



Синтаксис CSS: псевдоэлементы



The screenshot shows a code editor with two panes. The left pane is labeled "HTML ▾" and contains the following code:

```
1 <div>Lorem ipsum dolor sit amet, consectetur adipisci-  
sicing elit. Quibusdam harum, illum laborum numquam  
quas aperiam placeat repellat at, iure eos corporis  
magnam reprehenderit id atque molestias suscipit  
porro odio autem?</div>
```

The right pane is labeled "CSS ▾" and contains the following CSS code:

```
1 div {  
2   border: 2px solid red;  
3   padding: 20px;  
4 }  
5  
6 div::before,  
7 div::after {  
8   display: block;  
9   content: '';  
10  width: 100%;  
11  height: 20px;  
12 }  
13  
14 div::before {  
15   background: blue;  
16 }  
17  
18 div::after {  
19   background: red;  
20 }
```

Three red arrows point from the CSS pseudo-element declarations (div::before, div::after) to the corresponding visual effects in the preview area below. The preview area is a red-bordered box containing the text "Lorem ipsum dolor sit amet, consectetur adipisci-...". The top half of this text is blue, and the bottom half is red, matching the background colors defined in the CSS.

<https://jsfiddle.net/zfpnxe7s/>

Приоритет стилей

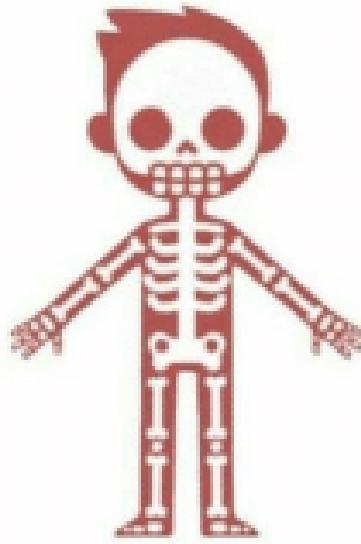
Порядок каскадирования и подсчет специфичности

- 1) **style="..." = 1000**
- 2) **#id = 100**
- 3) **.class, :pseudo-class, [attr] = 10**
- 4) **tag, ::pseudo-element = 1**

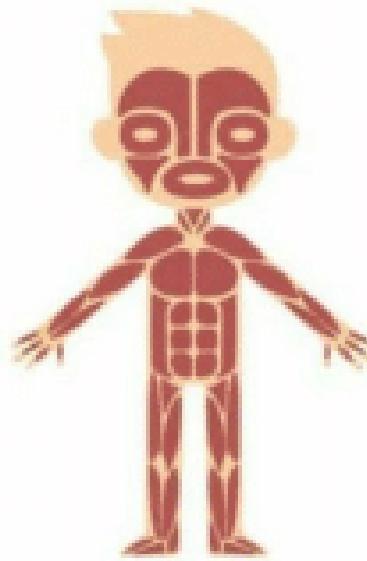
CSS – наводим красоту



HTML + CSS



HTML



JavaScript



CSS

JavaScript

- Язык Ja

1995 год

- Он был

Брендом

переди



«JScript

JavaScript

Объявление переменных



```
var number = 15;  
  
let string = 'Hello';  
  
const someone = { name: 'Petro' };
```

JavaScript

Объявление переменных. Область видимости

```
function outer() {  
  
    var a = 1;  
  
    if (true) {  
        var a = 2;  
    }  
  
    console.log(a); // 2  
}
```

```
function outer2() {  
  
    var a = 1;  
  
    if (true) {  
        let a = 2;  
    }  
  
    console.log(a); // 1  
}
```

JavaScript

- Зарезервированные слова

break	delete	for	let	super	void
case	do	function	new	switch	while
catch	else	if	package	this	with
class	enum	implements	private	throw	yield
const	export	import	protected	true	
continue	extends	in	public	try	
debugger	false	instanceof	return	typeof	
default	finally	interface	static	var	



1. Число “number”

```
var a = 123;  
var b = 12.345;
```

- Может быть как целым числом, так и дробным.
- Существуют специальные числовые значения **Infinity** и **NaN**

```
alert( 1 / 0 ); // Infinity
```

```
alert( "нечисло" * 2 ); // NaN
```

NaN != NaN

Проверка на NaN

```
if (myNum == NaN) { // Неправильно
  myNum = 0;
}
```

```
if (isNaN(myNum)) { // Правильно
  myNum = 0;
}
```

```
var testNaN = 0 / 0;
console.log(typeof testNaN); // number o_0
```

```
var testNaN = 0 / 0;
console.log(typeof testNaN); // number o_0
number
```

NaN === NaN	// false
NaN !== NaN	// true

isNaN(NaN)	// true
isNaN(0)	// false
isNaN('text')	// true
isNaN('0')	// false

2. Стока “string”

```
var str = "Я строка!";
```

```
var str = 'Одинарные ковычки тоже норм :)' ;
```

- Одинарные и двойные ковычки в JavaScript равноправны

3. Булевый (логический) тип “boolean”

- У него всего два значения: **true** и **false**

```
var isValid = true;
```

```
var isAnonymous = false;
```

4. Специальное значение “null”

- Значение **null** не относится ни к одному из предыдущих типов, а образует свой отдельный тип, состоящий из единственного значения **null**

```
var something = null;
```

Это просто специальное значение, которое имеет смысл “ничего” или “значение неизвестно”

5. Специальное значение “undefined”

- Значение **undefined** присваивается переменной, если в неё ничего не записано

```
var x;  
alert( x ); // выведет "undefined"
```

- **Undefined** можно присвоить в явном виде, но лучше для этого использовать **null**

6. Объекты “object”

- Все предыдущие типы были примитивными.
- В отличие от них, **object** является составным типом и может включать в себя другие объекты или примитивы

```
var cat = {  
    name : "Барсик",  
    color : "Чёрный"  
};
```

JavaScript

Типы данных

```
var cat = {  
    name: 'barsik',  
    color: 'black',  
    sayHello: function () {  
        console.log("Meow");  
    }  
}
```

cat.name

-> barsik

cat.sayHello()

-> Meow

cat['name']

-> barsik

cat['sayHello']()

-> Meow

JavaScript

7. Массивы

```
var mas = ['apple', 'orange', 'melon']

mas = ['ball', {name: 'cat'}, [1, 21], 100]

mas[1]
-> {name: 'cat'}
```



```
mas.length
-> 4
```

JavaScript

- ФУНКЦИИ

```
// Function Expression

var fact = function(x) {
  return x == 1 ? 1 : fact(x - 1) * x;
}

fact(5); // 120
```

```
// Function Declaration

function multiply(a, b) {
  return a * b;
}

multiply(2, 4); // 8
```

```
// Named Function Expression

var fact = function me(x) {
  return x == 1 ? 1 : me(x - 1) * x;
}

fact(4); // 24
```

JavaScript

- Стрелочные функции (ECMA-2015)

```
var sayHello = function (name) {
    console.log('Hello ' + name);
}

// ECMA-2015
var sayHello = (name) => console.log(`Hello ${name}`)

sayHello('test')
-> Hello test
```

JavaScript

Синтаксис

- Циклы

```
// 1
while (условие) {
|   ...
}
```

```
// 2
do {
|   ...
} while (условие)
```

```
// 3
for (var i = 0; i < 10; i++) {
|   ...
}
```

Метки

```
label:
for (;;) {
|   for (;;) {
|     break label;
}
}
```

JavaScript

Синтаксис

- Конструкция switch

```
var age = 18;

switch (age) {

    case '18':
        console.log( 'Не сработает' );
        break;

    case 18:
        console.log( 'А так сработает' );
        break;

    default:
        console.log( 'Любое другое значение' );
}
```

JavaScript

Синтаксис

- Условия

```
if (expression) {  
    //do something  
}
```

```
if (expression) {  
    //do something  
} else {  
    //do something  
}
```

```
if (expression) {  
    //do something  
} else if (expression) {  
    //do something  
} else {  
    //do something  
}
```

JavaScript

Подключение к странице

JavaScript добавляется на страницу как текст внутри тега **<script>**, либо как внешний файл через **<script src="путь"></script>**

```
<!DOCTYPE html>
<html>
<head>
    <title>Example java script</title>
    <meta charset="utf-8">
</head>

<body>
    Hello from html!
    <br>
</body>

<script>
    alert('Я javascript!')
</script>

</html>
```

```
<!DOCTYPE html>
<html>
<head>
    <title>Example java script</title>
    <meta charset="utf-8">
</head>

<body>
    Hello from html!
    <br>
</body>

<script src="script.js"></script>
</html>
```

JavaScript

Подключение к странице

Обратите внимание на то, что браузер выполняет код последовательно, т.е. если вначале HTML страницы будет подгрузка внешнего скрипта, остальная страница не отобразится, пока не будет загружен весь скрипт.

```
<html>
<head>
    <script src="big.js"></script>
</head>

<body>
    Этот текст не будет показан, пока браузер не выполнит big.js
</body>
</html>
```

Подключение к странице

- Существуют специальные атрибуты **async** и **defer** использующиеся для того, чтобы загружать скрипты асинхронно.
- **async** – загружает скрипты без сохранения порядка и не ждёт пока загрузится весь HTML
- **defer** – сохраняет последовательность загрузки и ждет, пока весь HTML-документ будет готов.

- Оператор проверки равенства “`==`”
- Если два значения относятся к одному типу, просто сравниваем их
- Если же значения относятся к разным типам, то пытаемся преобразовать их к одному типу и потом сравнить

JavaScript

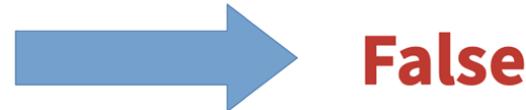
Динамическая типизация

$"22" == 22$ \downarrow $22 == 22$ \downarrow True	$" " == 22$ \downarrow $0 == 22$ \downarrow False	$"Text" == 22$ \downarrow $NaN == 22$ \downarrow False
True == $"22"$ \downarrow $1 == "22"$ \downarrow $1 == 22$ \downarrow False	False == 0 \downarrow $0 == 0$ \downarrow True	Undefined == Null \downarrow True

Динамическая типизация

Псевдоложь и псевдоправда:

- **undefined**
- **null**
- **0**
- **пустая строка**
- **Nan**



False

Все остальные значения JavaScript
Рассматривает как псевдоистину

Псевдоложь и псевдоправда:

```

if ([]){
  // true
}

var cat = { name: "Барсик" };
if (cat) {
  // true
}

if (1) {
  // true
}

var str = "Hello!";
if (str) {
  // true
}
  
```

```

var test;
if (test) {
  // false
}

var element = document.getElementById("elementThatDoesntExist");
if (element) {
  // false
}

if (0) {
  // false
}

if ("") {
  // false
}

if (NaN) {
  // false
}
  
```

Ещё больше преобразований:

```
var addi = 3 + "4";
```



При суммировании строки с числом выполняется конкатенация, а не суммирование.



Переменной *result* присваивается строка "34" (не 7).

```
var plusi = "4" + 3;
```

← То же самое... Получается "43".

JavaScript

Динамическая типизация

Ещё больше преобразований:

```
var multi = 3 * "4";
```

← Здесь JavaScript преобразует строку "4" в число 4, и умножает его на 3, получается 12.

```
var divi = 80 / "10";
```

← Стока "10" преобразуется в число 10. Затем 80 делится на число 10, получается 8.

```
var mini = "10" - 5;
```

← С вычитанием "10" преобразуется в число 10; получается 10 минус 5, то есть 5.

Забавный случай

```
null > 0    // false
null == 0    // false
null >= 0   // true
```

JavaScript

jQuery

jQuery

 ANGULARJS
by Google

2006

2009

 React

2013

 
Vue.js

2016

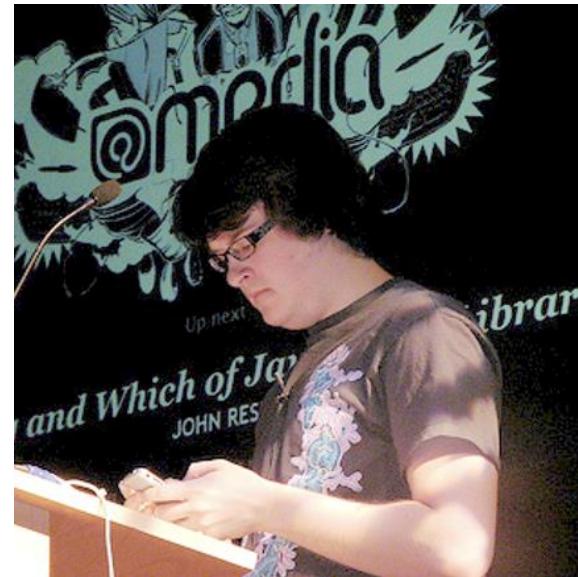
 **jQuery** ≠  JS
write less, do more.

JavaScript

jQuery - библиотека для работы с DOM деревом. Первый выпуск был в 2006 году. Основная особенность, упрощение API для работы с DOM-деревом.



<https://jquery.com/>



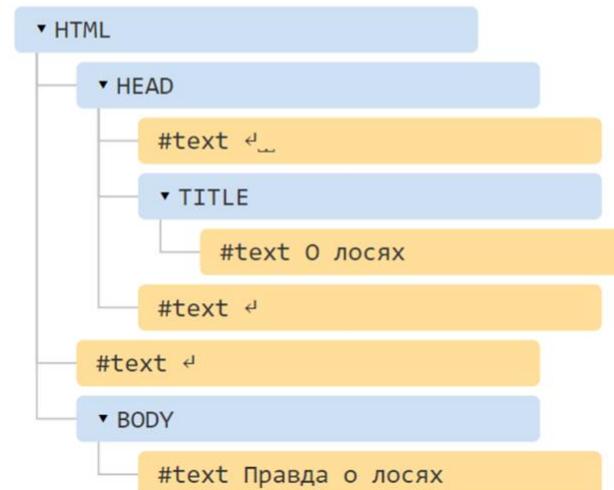
Джон Резиг

JavaScript

jQuery. DOM-дерево

Согласно DOM-модели, документ является иерархией, деревом.
Каждый HTML-тег образует узел дерева с типом «элемент».
Вложенные в него теги становятся дочерними узлами. Для представления текста создаются узлы с типом «текст»

```
<!DOCTYPE HTML>
<html>
<head>
    <title>О лосях</title>
</head>
<body>
    Правда о лосях
</body>
</html>
```



JavaScript

jQuery

Особенная функция jQuery

\$(/* *selector* */)

Значок доллара в этом случае - просто название функции.

```
const $ = function(selector) { ... }  
const jQuery = function(selector) { ... }
```

<https://plnkr.co/edit/iPuhXg1Wv6O1gnmAXleQ> - песочница с jQuery

<https://embed.plnkr.co/WIgOxgjlcbIQ8aF6GtKI/> - пример игры “змейка” с jQuery

JavaScript

jQuery. Селекторы

Селектор	Описание, пример
Элемента	Выбирает все элементы данного типа на странице, например, <code>\$("div")</code> .
Элемент1 элемент2	Выбирает все элементы2, вложенные непосредственно в элемент1, например, <code> \$("p img")</code> .
Класса	Выбирает все элементы заданного класса, например, <code> \$(".sidebar")</code> .
Идентификатора	Выбирает элемент с указанным идентификатором, например, <code> \$("#main")</code> .
Элемент класс	Выбирает из элементов данного типа только те элементы, которым назначен указанный класс, например, <code> \$("p.first")</code> .
Потомка	Выбирает все указанные элементы выбранного селектора, например, <code> \$(".sidebar a")</code> .
Дочерние	Выбирает элементы, соответствующие второму селектору, которые содержатся непосредственно внутри первого селектора, являющиеся дочерними по отношению к нему, например, <code> \$("body > p")</code> .
Сестринские	Выбирает элементы, соответствующие второму селектору, идущие непосредственно за первым элементом, являющимся для него сестринским, например, <code> \$("h2 + p")</code> .
	Выбирает элементы, соответствующие второму селектору, являющиеся сестринскими по отношению к первому элементу и расположенные после него, например, <code> \$("h2 ~ p")</code> .

<https://html5book.ru/jquery-selectors/>

JavaScript

jQuery сравнение с современным js

Получение элементов DOM дерева

jQuery

```
// Выбираем все элементы с классом box
var $box = $('.box');
```

Vanilla JS

```
// тоже самое с vanilla JS
document.querySelector('.box');
document.querySelectorAll('.box');
```

JavaScript

jQuery сравнение с современным js

Работа со стилями элемента

jQuery

```
// Скрываем все .box
$('.box').hide();
```

Vanilla JS

```
// Проходимся по всему массиву элементов, чтобы скрыть все
// элементы с .box
document.querySelectorAll('.box').forEach(box => {
  box.style.display = 'none'
})
```

JavaScript

jQuery сравнение с современным js

Выбор дочернего элемента

jQuery

```
// Выбираем первый .box в .container
var container = $('.container');
container.find('.box');
```

Vanilla JS

```
// Выбираем первый .box в .container
var container = document.querySelectorAll('.container');
container.querySelector('.box');
```

JavaScript

jQuery сравнение с современным js

Выбор соседей

jQuery

```
// Отдаст следующий, предыдущий и родительский  
// элемент для .box
```

```
$('.box').next();  
$('.box').prev();  
$('.box').parent();
```

Так плохо

```
$('.box').next().next().next();
```

Vanilla JS

```
// Отдаст следующий, предыдущий и родительский элемент  
// для .box
```

```
var box = document.querySelector('.box');  
box.nextElementSibling;  
box.previousElementSibling;  
box.parentElement
```

Так лучше

```
$('.box').next('.selected').next('.main')
```

Создание элементов

Если вы хотите динамически создать элемент в JavaScript, чтобы добавить его в DOM, то вы можете вызывать `createElement()` на `document` и передать ему имя тега, как аргумент, чтобы указать какой именно элемент вы хотите создать:

```
// Создаем div и span
$("<div/>");
$("<span/>");

// Создаем div и span
document.createElement("div");
document.createElement("span");
```

Стилизация элементов

Если вы вызываете `css()` на элементе, чтобы поменять его CSS инлайново с помощью jQuery, то этого же эффекта вы можете добиться с помощью `.style` в чистом JavaScript.

```
// С jQuery
// Выбирает .box и меняет его цвет текста на #000
$(".box").css("color", "#000");

// Без jQuery
// Выбирает первый .box и меняет его цвет текста на #000
document.querySelector(".box").style.color = "#000";
```

JavaScript

jQuery сравнение с современным js

С jQuery вы можете передать объект с парами ключ-значения, чтобы стилизовать уже большое количество свойств за раз. В JavaScript вы можете выставить только одно значение за раз или указать все стили одной строкой:

```
// С jQuery
// Передаем несколько стилей
$(".box").css({
    "color": "#000",
    "background-color": "red"
});

// Без jQuery
// Выставляем цвет на #000 и делаем фон красным
var box = document.querySelector(".box");
box.style.color = "#000";
box.style.backgroundColor = "red";

// Выставляем все стили разом, но перезаписываем уже существующие
box.style.cssText = "color: #000; background-color: red";
```

jQuery сравнение с современным js

hide() и show()

Все удобства методов `hide()` и `show()` можно получить через свойство `.style`, выставив `display` на `none` или `block`:

```
// С jQuery
// Спрятать и показать элемент
$(".box").hide();
$(".box").show();

// Без jQuery
// Прячем и показываем элемент, изменения display на block или none
document.querySelector(".box").style.display = "none";
document.querySelector(".box").style.display = "block";
```

Работа с событиями

В jQuery есть огромное множество способов для того, чтобы слушать события, вместо `on()`, `bind()`, `live()` или `click()`, вы могли бы сделать всё тоже самое с помощью их эквивалента `addEventListener`:

```
// С jQuery
$(".button").click(function(e) { /* handle click event */ });
$(".button").mouseenter(function(e) { /* handle click event */ });
$(document).keyup(function(e) { /* handle key up event */ });

// Без jQuery
document.querySelector(".button").addEventListener("click", (e) => {
/* ... */ });
document.querySelector(".button").addEventListener("mouseenter", (e)
=> { /* ... */ });
document.addEventListener("keyup", (e) => { /* ... */ });
```

Вызываем и создаем события

Вы можете вручную вызывать события с помощью `trigger()` в jQuery, а также в чистом JS при помощи `dispatchEvent()`. Метод `dispatchEvent()` может быть вызван совершенно на любом элементе и берёт `Event`, как первый аргумент:

```
// С jQuery
// Вызываем myEvent на документе и .box
$(document).trigger("myEvent");
$(".box").trigger("myEvent");

// Без jQuery
// Создаем и запускаем myEvent
document.dispatchEvent(new Event("myEvent"));
document.querySelector(".box").dispatchEvent(new Event("myEvent"));
```

JavaScript

jQuery сравнение с современным js

А теперь всё вместе, как мы можем обновить текст и класс и добавить это всё в DOM:

```
// Создаём div
var element = document.createElement("div");

// Добавляем ему класс
element.classList.add("box");

// Указываем текст
element.textContent = "Text inside box";

// Вставляем его в .container
document.querySelector(".container").appendChild(element);
```

JavaScript

Подключение jQuery

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6   <!--Подключаем библиотеку-->
7   <script src="js/jquery-2.2.3.min.js"></script>
8 </head>
9 <body>
10 </body>
11 </html>
```

```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Document</title>
6   <!--Подключаем библиотеку-->
7   <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>
8 </head>
9 <body>
10 </body>
11 </html>
12
```

Полезные ссылки, ресурсы

- <http://htmlbook.ru/>
- <https://learn.javascript.ru/>
- <https://html5book.ru/>

Домашнее задание