

# Document Object Model (DOM)



JavaScript  
Courses

[www.courses.dp.ua](http://www.courses.dp.ua)

## Hello, Document Object Model!

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Integer a aliquet metus. Cras luctus metus velit, ac rhoncus velit scelerisque non. Etiam vitae condimentum sem. Maecenas at egestas dui. Ut sem tellus, condimentum a turpis laoreet, rhoncus aliquet ex. Ut sit amet felis lorem, Donec dapibus cursus risus nec aliquet. Etiam nibh dolor, porttitor et ipsum vitae, pharetra congue justo. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque nec nisi eu lorem interdum eleifend. Proin in bibendum magna.



Integer lobortis metus enim, tincidunt sagittis nibh pretium id. Integer sodales finibus iaculis. In quis ex magna, Aenean nec porta eros. Integer dapibus, nulla ac egestas viverra, orci massa pharetra felis, eu sodales sapien metus ac dolor. Curabitur venenatis iaculis metus. Pellentesque diam nisi, dictum et sodales vitae, auctor et mauris. Aliquam at tortor eu diam congue auctor sed fringilla massa. Aliquam erat volutpat.

Aliquam tempus lacus in neque dapibus, id sodales sem sodales. Duis eros magna, convallis nec placerat eu, sollicitudin in turpis. Proin luctus mollis varius. Donec ornare eu sapien nec dapibus. Nunc cursus, nulla eget iaculis mollis, enim enim ornare odio, in scelerisque ipsum lacus at dui. Nullam finibus pharetra erat, a dapibus est lacinia sed. Nunc tempus mi rutrum est interdum egestas. Suspendisse vel metus quis metus pharetra auctor. Suspendisse eu felis augue. Sed in vehicula tellus. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In consectetur eget dui ut dapibus. Sed non libero ullamcorper, sodales mi vitae, cursus odio. Proin vel velit auctor, pulvinar purus sed, aliquam arcu. Sed eu aliquet diam. Duis sed nulla a leo rutrum viverra eget ac metus.



Quisque ac lacinia libero. Integer ultricies sollicitudin vestibulum. Cras vulputate est eget accumsan ullamcorper. Proin euismod dolor et tortor lacinia, non consequat lacus consequat. Sed ornare lectus ipsum, egestas tempor turpis accumsan ac. Mauris at ante auctor, lobortis dolor in, fermentum nunc. In ac egestas sem, ac finibus nisi. Ut rhoncus lobortis pellentesque. Donec eu est erat. Nam eu tincidunt nisi. Sed porta ligula id viverra sagittis. Ut sollicitudin, tellus id iaculis accumsan, nibh enim posuere enim, ut lacinia mauris lorem ut ipsum. Aenean condimentum nisi metus, nec porta est viverra nec. Ut tempor egestas efficitur.

**Используйте заготовку**  
**[./source/ex01.html](/source/ex01.html)**

# window.document

*Хранилище HTML-документа*

# DOM – Document Object Model

*(объектная модель документа)*

*Стандарт который определяет из каких объектов браузер собирает дерево документа, какие свойства и методы есть у этих объектов у этих.*

<https://learn.javascript.ru/document>

# Задача JavaScript – изменение HTML-документа

## 1. Добавление нового элемента:

*Создать новый элемент и присоединить его, в качестве дочернего, к одному из существующих элементов;*

## 2. Изменение элемента:

*Изменение свойств элемента (в т.ч. содержимого);  
Изменение его позиции в дереве документа;*

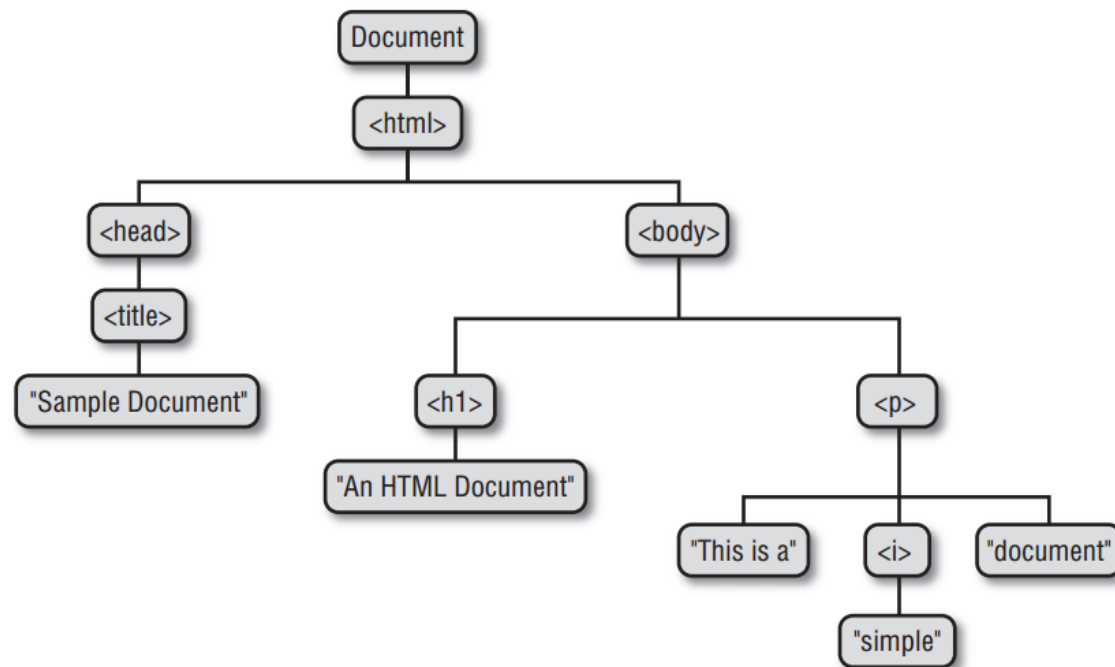
## 3. Удаление элемента (из дерева документа).

# Структура HTML-документа

```
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*

# Древовидная структура HTML-документа



В JavaScript **каждый тег** дерева представлен **объектом** (часто используется термин: узел, *node*). У каждого элемента есть один родительский элемент, и множество дочерних элементов (от 0 до  $\infty$ ).

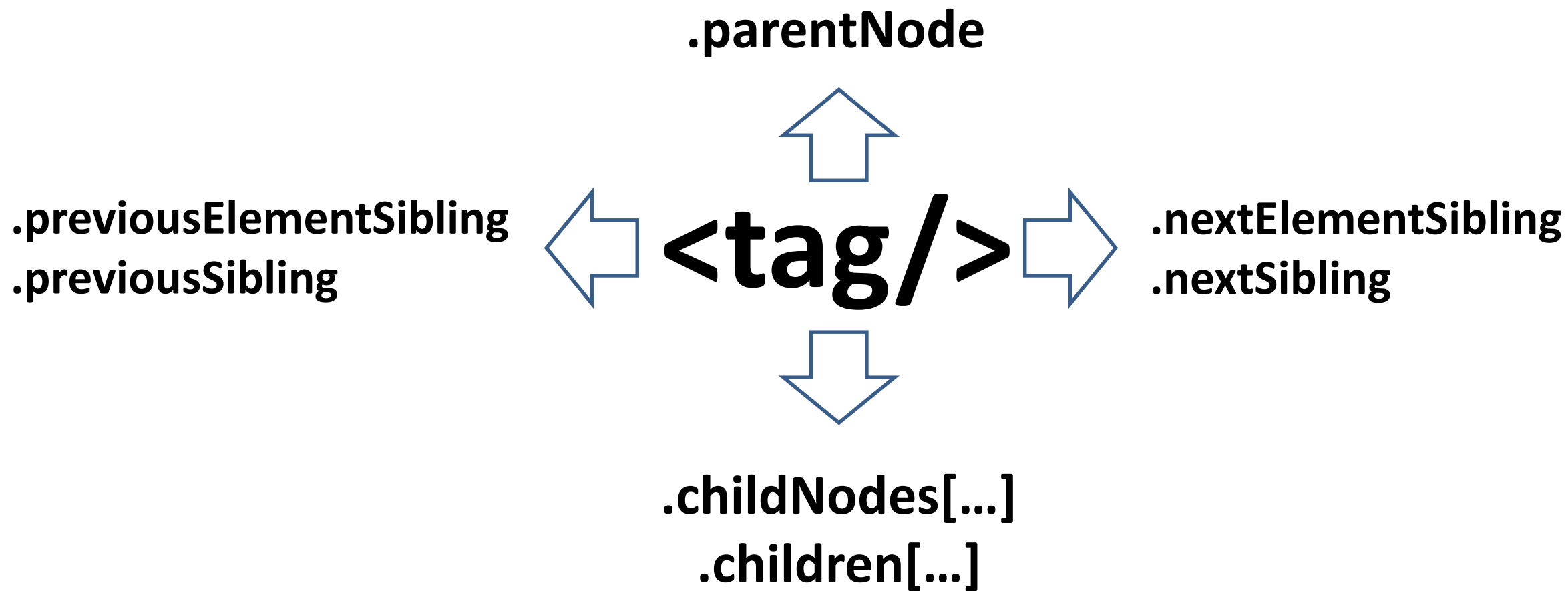
# Node/Узел/Тег/Элемент

*Каждый тег представлен объектом*

*Воздействие на свойства и методы  
которого позволяют управлять  
внешним видом тега на странице.*



# Свойства элементов HTML-документа



Каждый объект (элемент, тег) имеет среди своих свойств те которые хранят ссылку на родительский элемент (**parentNode**), на соседние элементы (**previousElementSibling** и **nextElementSibling**) и на перечень потомков (**childNodes** и **children**)

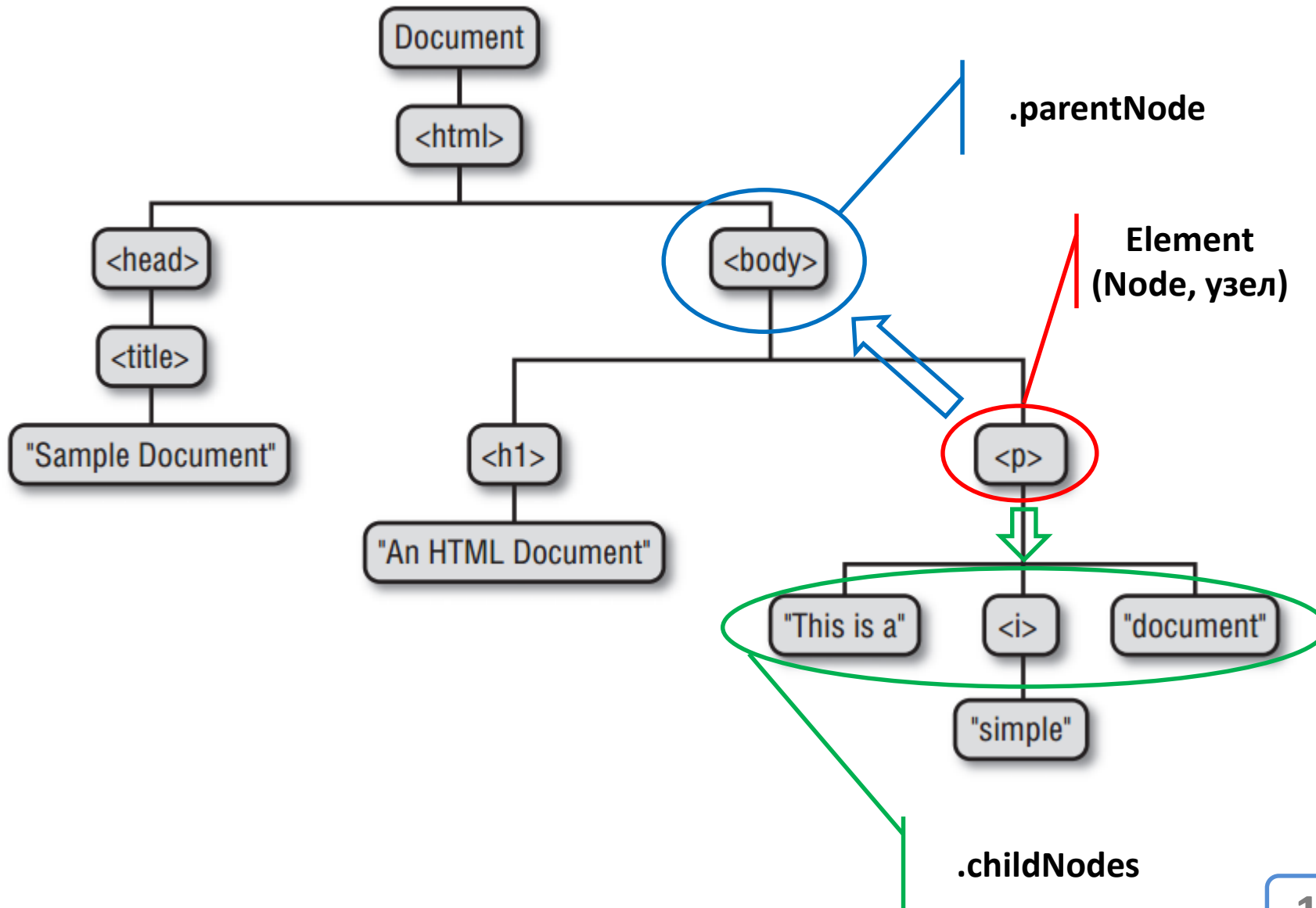
# Свойства элементов HTML-документа

**<tag/>**

Также среди свойств объекта (элемента) есть те которые позволяют управлять содержимым (атрибутами, стилями) или подпиской на событиями, а также ряд методов позволяющих добавлять/удалять элементы, и искать вложенные элементы.

*.id*  
*.innerHTML*  
*.className*  
*.classList[...]*  
*.attributes[...]*  
*.style { ... }*  
  
*.onclick*  
*.ondblclick*  
*.onmouseenter*  
  
*.appendChild()*  
*.insertBefore()*  
*.remove()*  
*.insertAdjacentHTML()*  
*.insertAdjacentElement()*  
*.insertAdjacentText()*  
...

# DOM – Document Object Model

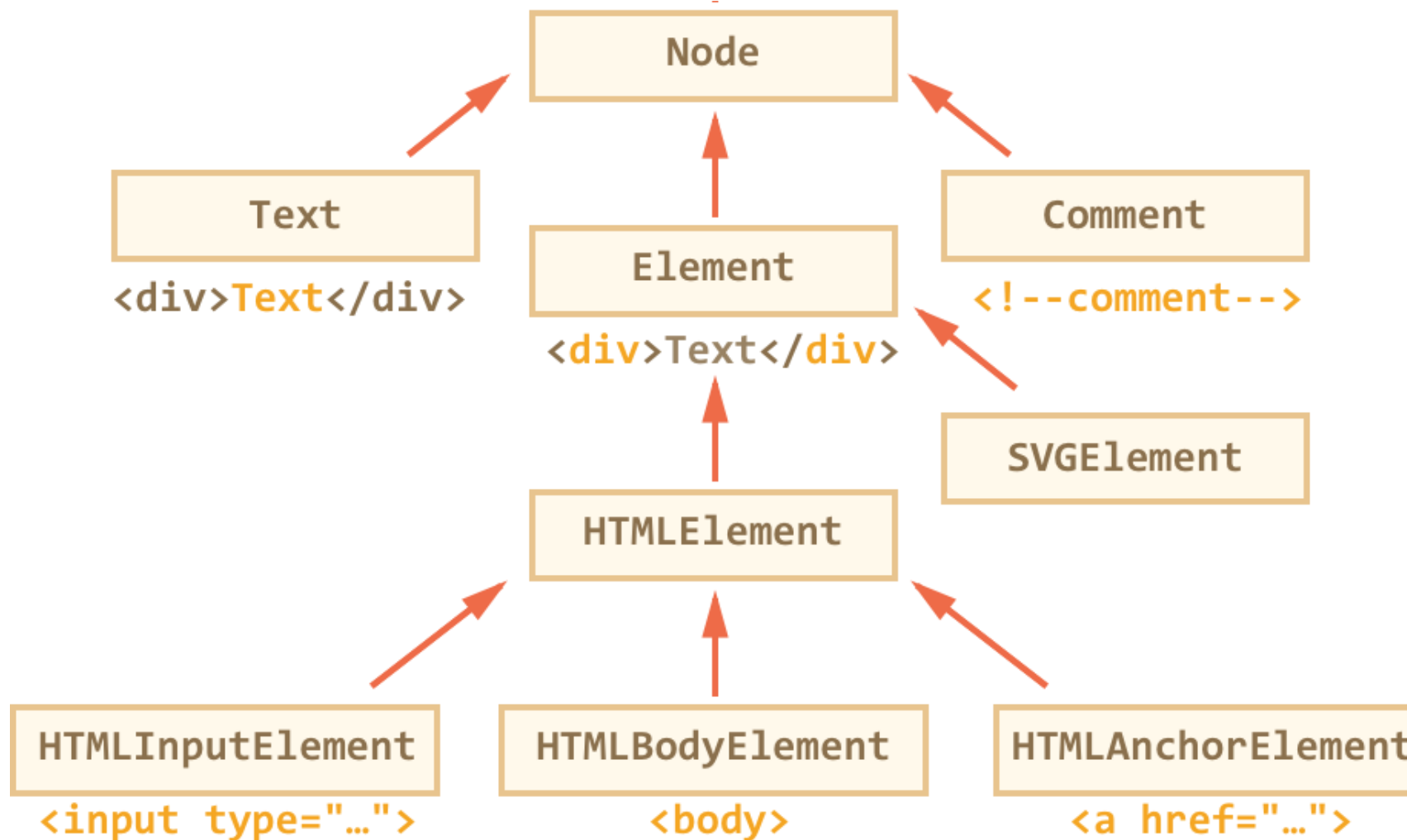


## **window.document – корень дерева документа**

```
20 <script>
21
22     console.dir(window.document.childNodes) ;
23
24 </script>
```

***window.document.childNodes*** – коллекция с тегами верхнего уровня.

# Типы объектов в иерархии документа



## Типы объектов в иерархии документа

Element type	NodeType
Element	1
Attribute	2
Text	3
Comment	8
Document	9

*Свойство **.nodeType** хранит тип узла (в виде числа), по нему можно определить к какой категории относится тот или иной узел.*

Когда выполняется  
код в теге `<script>` ?

# JavaScript в HTML

*<script></script>*

*Тег скрипт может быть размещен в любом месте HTML-документа, с помощью него можно либо непосредственно писать JS код, либо подключать внешний файл с кодом. Однако....*



# JavaScript в HTML

*Разрешить это неудобство (с выполнением кода сразу, а не когда страница полностью загрузится) можно разными способами, например:*

1. Разместить весь код в конце документа;
2. Разместить весь код во внешнем файле и подключить его с атрибутом **defer**;
3. Использовать события **onLoad** или **onDOMContentLoaded** (эти варианты мы рассмотрим детальнее когда будет говорить о событиях).

```
<script defer src="scripts/async.js"></script>
```

*Атрибут **defer** откладывает выполнение скрипта до тех пор, пока вся страница не будет загружена полностью. Работает только для внешних (подключаемых) файлов.*

**Как добраться до тега?**

## Теги у которых есть атрибут **id** доступны в виде свойств объекта **window**

```
43 <script>
44
45     window.special.style.backgroundColor = "yellow";
46
47     //special.style.backgroundColor = "yellow";
48
49 </script>
```

*Но только если **id** состоит из допустимых, в **JavaScript** символов, для именования переменных.*

# Поиск элементов в документе

*Выбор элемента с которым проводить манипуляции самая часто выполняемая операция в JS.*

**Выбор элемента по атрибуту id:**

```
document.getElementById("some_id") ;
```

*Возвращает один элемент атрибут (свойство) **id** равно «**some\_id**». Если такого элемента нет в документе, то возвращается **null**.*

# Поиск элементов в документе

**Выбор элементов по названию тега:**

```
document.getElementsByTagName ("tag_name") ;
```

**Выбор элементов по атрибуту name:**

```
document.getElementsByName ("attr_name") ;
```

**Выбор элементов по атрибуту class:**

```
document.getElementsByClassName ("class_name") ;
```

*Все эти функции возвращают псевдомассив с теми элементами которые подошли под условие.*

## Поиск элементов в документе

**Выбор всех элементов которые соответствуют CSS селектору:**

```
document.querySelectorAll("css_selector") ;
```

*Возвращает псевдомассив с теми элементами которые подошли под условие css-селектора.*

```
document.querySelector("css_selector") ;
```

*Возвращает первый найденный элемент который подошел под условие css-селектора (или **null** если ничего не найдено).*

# На практике

```
23  var a = document.getElementsByClassName("display-3");  
24  console.dir(a);  
25  
26  var b = document.getElementsByTagName("p");  
27  console.dir(b);  
28  
29  var c = document.querySelector("h1 + p");  
30  console.dir(c);  
31  c.style.backgroundColor = "yellow";  
32  
33  var d = document.querySelectorAll("#special img");  
34  console.dir(d);  
35  
36  d.forEach(function(element) {  
37      element.style.border = "3px solid orange";  
38  });
```

*Попробуем задействовать рассмотренные функции поиска элементов в дереве HTML-документа.*

## Вложенный поиск, т.е. поиск в результатах поиска

```
23     var first_row = document.querySelector(".jumbotron + .row");
24
25     var p_elements = first_row.querySelectorAll("p");
26
27     p_elements.forEach(function(element) {
28         element.style.border = "2px solid red";
29     });
```

**Функции поиска элементов можно применять к любому существующему элементу, а не только к документу. Когда функция поиска применяется к конкретному элементу, то поиск осуществляется среди его потомков.**



# «Живые» и статические коллекции

```
20 <script>
21
22 window.onload = function() {
23     var live_imgs = document.getElementsByTagName("img");
24     var static_imgs = document.querySelectorAll("img");
25
26     console.log("Before", "static " + static_imgs.length, "live " + live_imgs.length);
27
28     document.querySelector("img").remove();
29
30     console.log("After", "static " + static_imgs.length, "live " + live_imgs.length);
31 }
32
33 </script>
```

```
Before static 3 live 3
After static 3 live 2
> |
```

*Живые (**Live**) коллекции изменяют свой состав в зависимости от изменений в документа. Статические (**Static**) коллекции не изменяют свой состав после формирования.*

# Живые и статические коллекции

```
.querySelector() , .querySelectorAll()
```

*Возвращают статические коллекции, т.е.  
«слепок» на момент вызова функции.*

```
.getElementsBy...
```

*Возвращают живые коллекции, которые всегда актуальны. Т.е. массив с результатом работы этих функций всегда будет содержать актуальное количество результатов, что бы не происходило с документом. Псевдомассив **.children** также относится к «живым» коллекциям.*

*С живыми коллекциями нужно быть осторожным в том случае если вы перебираете её в цикле и изменяете её состав.*

```
<script>  
    var result = document.getElementsByTagName("p");  
    result = [...result];  
</script>
```

*Однако, и живую и статическую коллекцию можно конвертировать в классический массив.*

Как изменить тег?

# Свойство **.innerHTML** хранит содержимое тега

```
43 <script>  
44     special.innerHTML = "Hello world!!!";  
46 </script>
```

*Свойство **.innerHTML** – можно не только считывать но и устанавливать. Изменение свойства **.innerHTML** – влечёт перерисовку документа.*

# Полезные свойства элементов

**.className** – свойство содержит полный список всех классов которые присвоены тегу (одной строкой).

**.classList** – свойство содержит список всех классов которые присвоены тегу (в виде массива).

**.classList.add('cat')** – метод добавляет класс к тегу (если есть другие классы то они остаются).

**.classList.remove('cat')** – метод удаляет класс у тегу (если есть другие классы то они не затрагиваются).

**.classList.toggle('cat')** – метод удаляет класс у тегу, если он есть, или добавляет класс, если его нет.

**.classList.contains('cat')** – метод проверяет наличие у тега заданного класса (возвращает true/false).

**.style** – свойство определяющее объект со всеми поддерживаемыми браузером стилевые свойства (CSS).

**.style.setProperty('css-property', 'value')** – метод для установки стилового свойства (CSS).

**.attributes** – хранит коллекцию с атрибутами тега.

Как удалить тег?

## Удаление элементов из дерева документа

```
43  <script>
44
45      var tag = special;
46      special.remove();
47      console.dir(tag);
48
49  </script>
```

```
► ownerDocument: document
  parentElement: null
  parentNode: null
  prefix: null
```

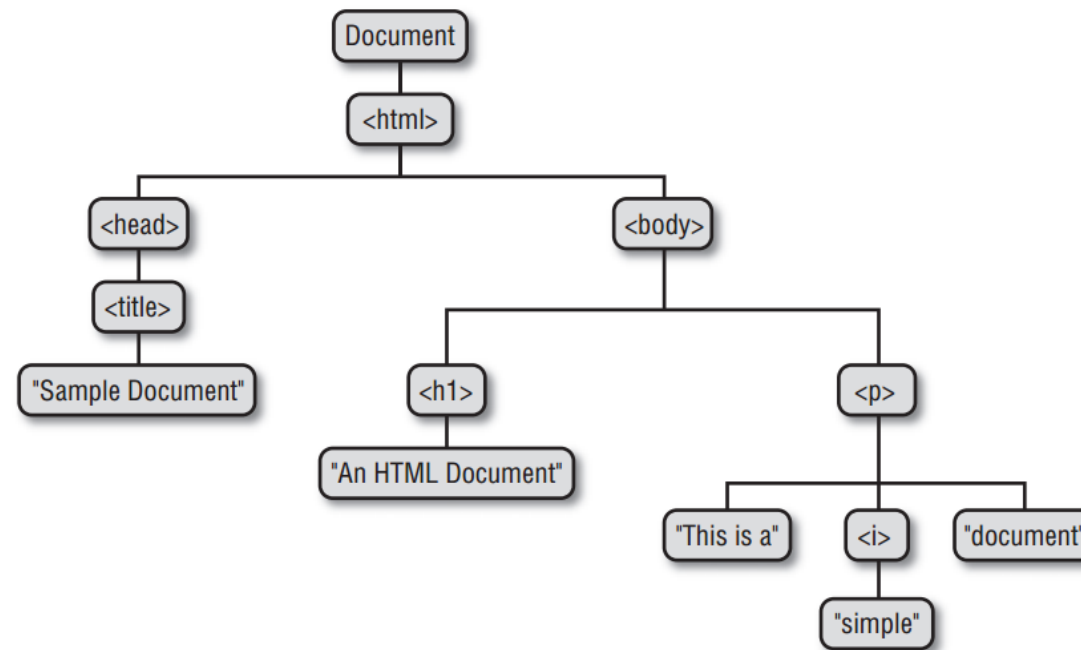
Удалить элемент из дерева документа можно вызывая у него метод **.remove()**, при этом все его дочерние элементы также исчезнут со страницы. Однако сам объект-тег не уничтожается. Его можно использовать в дальнейшем.



Как создать и добавить тег?

# Добавление новых элементов к дереву документа

*Вставить новый элемент в документ, можно прикрепив его к какому-либо существующему элементу. Т.е. прикрепить его к родительскому элементу (другими словами: сделать его дочерним для существующего элемента).*



# Добавление новых элементов к дереву документа

*Простейший вариант: просто добавить текстовую строку с нужными данным к свойству **.innerHTML**. Однако это не самый удобный вариант.*

```
var h1 = document.querySelector("h1");  
h1.innerHTML = "<b>Hello</b> to <i>all</i> <u>people</u>!";
```

```
▼ <h1 class="display-3">  
  <b>Hello</b>  
  " to "  
  <i>all</i>  
  <u>people</u>  
  "!"  
</h1>
```

## Добавление новых элементов к дереву документа

```
3  var alert_tag = document.createElement("div");
4
5  alert_tag.className = "alert alert-info text-center";
6
7  alert_tag.innerHTML = "Hello world!";
8
9  var h1 = document.querySelector("h1");
10
11  h1.insertAdjacentElement('afterend', alert_tag);
12
```

***document.createElement()*** – создаёт новый элемент (по имени тега). Этот элемент, после создания, еще не включен в дерево. Но его свойства уже можно изменять.

## Добавление новых элементов к дереву документа

```
<!-- beforebegin -->
<p>
  <!-- afterbegin -->
  foo
  <!-- beforeend -->
</p>
<!-- afterend -->
```

Варианты позиции для методов  
группы *.insertAdjacent...()*

**tag.insertAdjacentElement**(**position**, **element**)  
добавляет **элемент** к **существующему**, в указанную **позицию**.

<https://developer.mozilla.org/ru/docs/Web/API/Element/insertAdjacentElement>

Также существуют методы **tag.insertAdjacentHTML()** и **tag.insertAdjacentText()**

## Добавление новых элементов к дереву документа

```
23  var row = document.querySelector(".row");  
24  
25  var tag = document.createElement("p");  
26  tag.innerHTML = "Hello world!!!";  
27  tag.classList.add("text-center");  
28  tag.classList.add("col-12");  
29  
30  row.insertBefore(tag, row.firstChild);
```

***.insertBefore()*** – добавляет элемент в качестве дочернего, при этом позволяет указать перед каким из, уже существующих, потомков новый элемент должен быть размещён.

# Новые элементы, свойства и классы

```
var element = document.createElement("div");  
  
element.style.color      = "red";  
element.style.fontSize   = "32pt";  
element.style.margin     = "10px";  
//...
```

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

```
.myclass{  
    color: red;  
    font-size: 32pt;  
    margin: 10px;  
    ...  
}
```

*Установка атрибута «класс» для элемента, снимает необходимость задавать в коде 100500 свойств.*

Немного практики



# Hello, Document Object Model!

Sony Ericsson K750 Gold	999 \$
iPhone 8	899 \$
iPhone 7	799 \$
Apple iPhone 6	199 \$
Samsung Galaxy S9	650 \$
Nokia 3310	99 \$
HTC Two M3	433 \$
Lenovo H6000	250 \$
Microsoft Lumia 950	195 \$

Сделаем «перелистывание», наша цель:

[./source/ex02\\_demo.html](#)

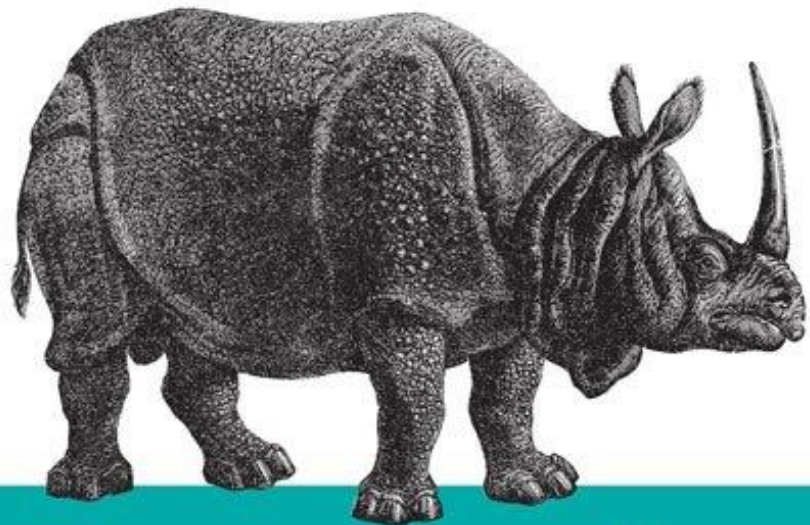
**Используйте заготовку [./source/ex02.html](#)**

Домашнее задание  
/узнать

***tag.appendChild(), tag.insertBefore()*** – узнайте о «классических» методах добавления элемента в дерево документа.

Создание активных веб-страниц

6-е издание  
Включает ECMAScript 5 и HTML5



# JavaScript

Подробное руководство

  
O'REILLY®

Дэвид Флэнаган

# Дэвид Флэнаган *JavaScript.* *Подробное руководство*

Предварительные знания –  
лучший помощник в обучении,  
поэтому к следующему занятию  
жду, что **вы прочтёте 17-ю главу**  
**«Обработка событий».**

# Современный учебник Javascript

Перед вами учебник по JavaScript, начиная с основ, включающий в себя много тонкостей и фишек JavaScript/DOM.



смотреть на Github

Поделиться:



НАЙТИ

## Содержание

Первые две части посвящены JavaScript и его использованию в браузере. Затем идут дополнительные циклы статей на разные темы.

**Предварительные знания – лучший помощник** в обучении, поэтому к следующему занятию жду, что **пройдёте разделы 2-й части 2.1-2.8, 3.1, 3.3, 3.6, 3.8, 3.9, 3.10, 3.11**

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

Домашнее задание  
/сделать

# Домашнее задание #Е.1

## DOM Homework

Microsoft Lumia 950	195 \$
Sony Ericsson K750 Gold	999 \$
iPhone 8	899 \$
Lenovo H6000	250 \$
Apple iPhone 6	199 \$
Samsung Galaxy S9	650 \$
Nokia 3310	99 \$
HTC Two M3	433 \$

Сделаем плавное перемещение элементов их «хвоста» списка в начало, наша цель : [./homework/hw01\\_demo.html](#) (Вам могут помочь свойства `.offsetTop` и `.offsetHeight`)

**Используйте заготовку [./homework/hw01.html](#)**

# Домашнее задание #Е.2

## PhonesShop

iPhone 8	397 \$
iPhone 7	337 \$
Apple iPhone 6	340 \$
Samsung Galaxy S9	908 \$
Nokia 3310	414 \$
HTC Two M3	656 \$
Lenovo H6000	584 \$
Microsoft Lumia 950	145 \$
Sony Ericsson K750 Gold	182 \$

В коде заготовлен массив ***phones*** выведите его в разметку (в виде *bootstrap-списка*) в **отсортированном по цене виде**.

**Используйте заготовку**  
[\*\*\*./homework/hw02.html\*\*\*](#)