Document Object Model (DOM)

JavaScript Courses

www.courses.dp.ua

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

состоит из:

<tag attr="value”>Text data</tag>

Теги как контейнер для информации
+**атрибуты**

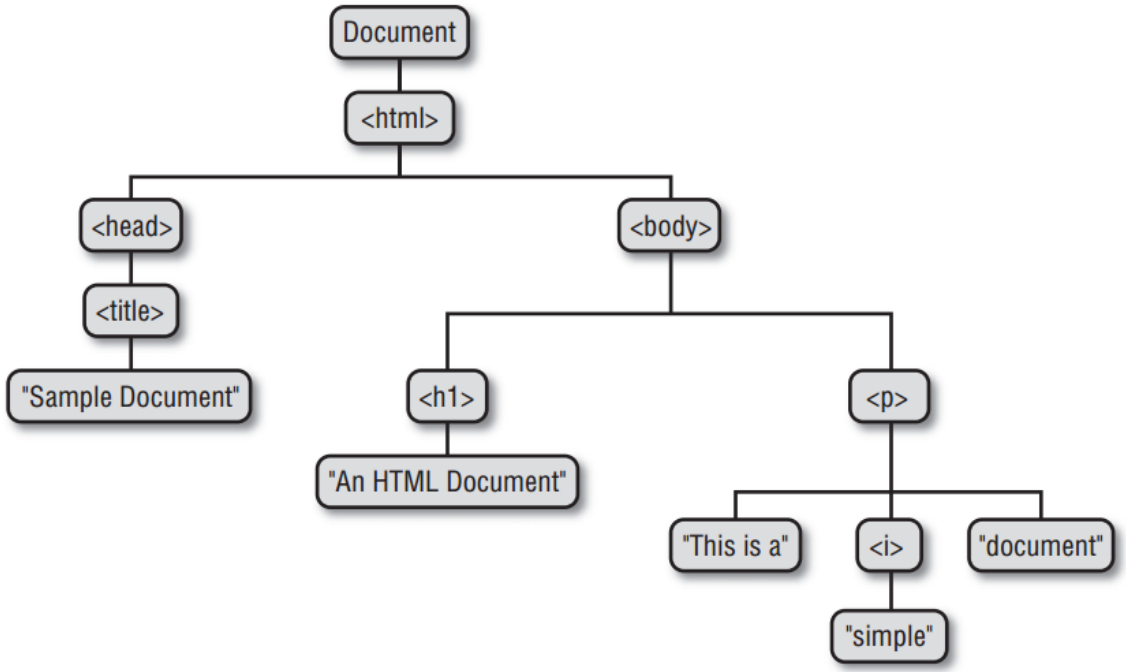
Текстовые данные (содержимое, контент)

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



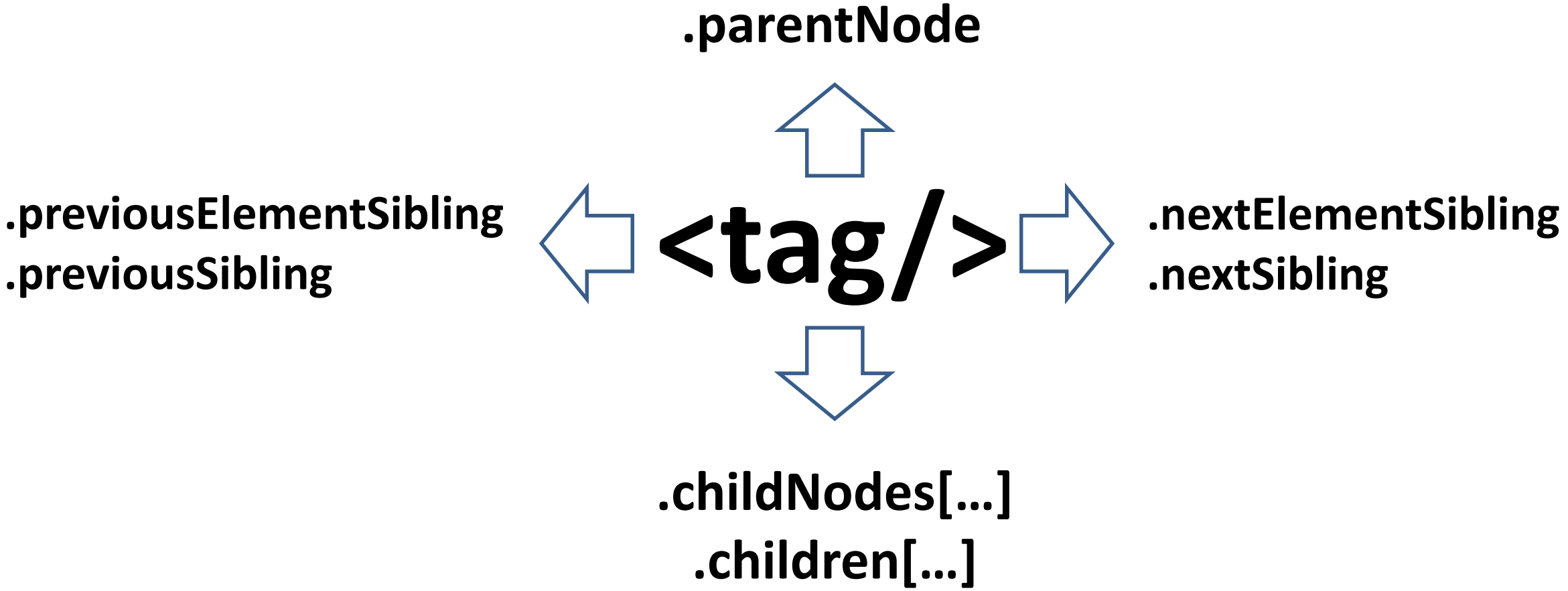
*В JavaScript **каждый тег** дерева представлен **объектом** (часто используется термин: узел, node). У каждого элемента есть один родительский элемент, и множество дочерних элементов (от 0 до ∞).*

Node/Узел/Тег/Элемент

Каждый тег представлен объектом

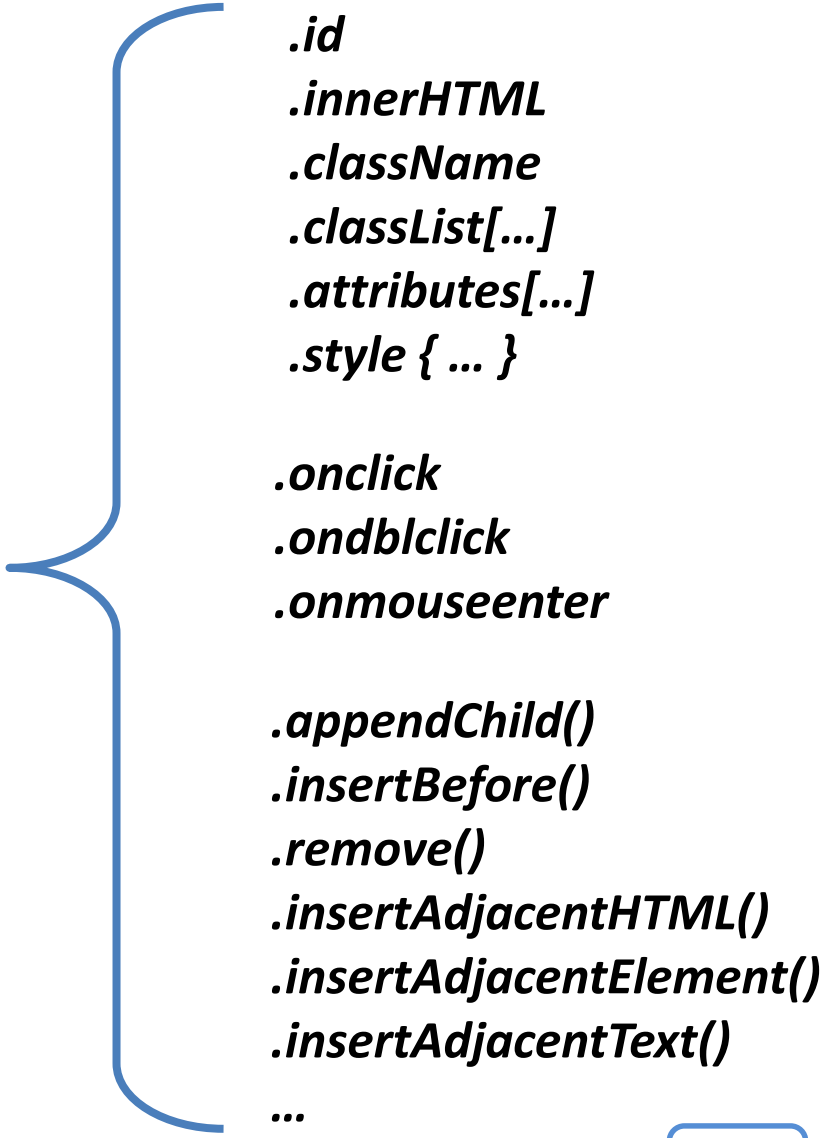
Воздействие на свойства и методы которого позволяют управлять внешним видом тега на странице.

Свойства элементов HTML-документа



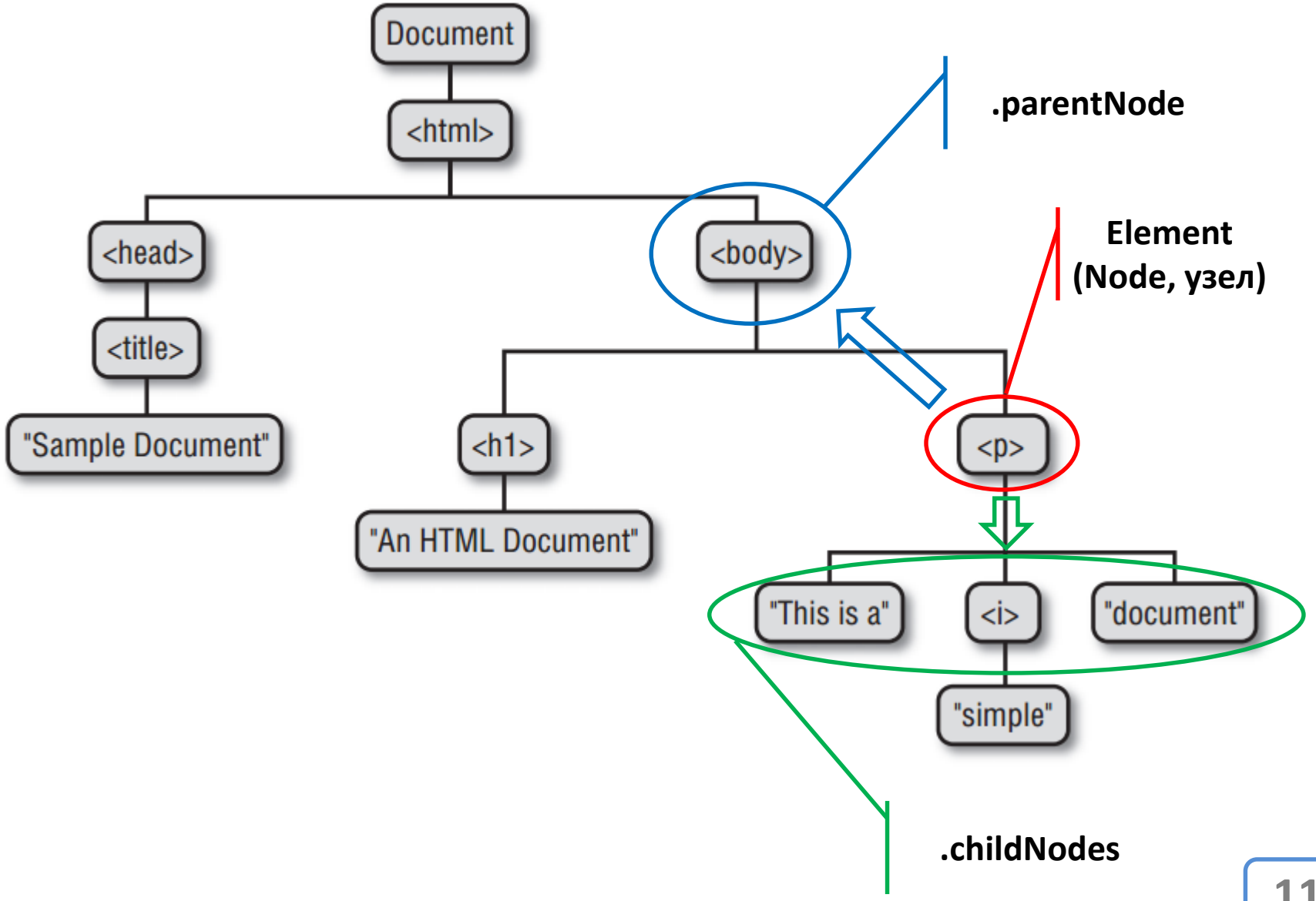
Каждый объект (элемент, тег) имеет среди своих свойств те которые хранят ссылку на родительский элемент (**parentNode**), на соседние элементы (**previousElementSibling** и **nextElementSibling**) и на перечень потомков (**childNodes** и **children**)

Свойства элементов HTML-документа



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.

DOM – Document Object Model

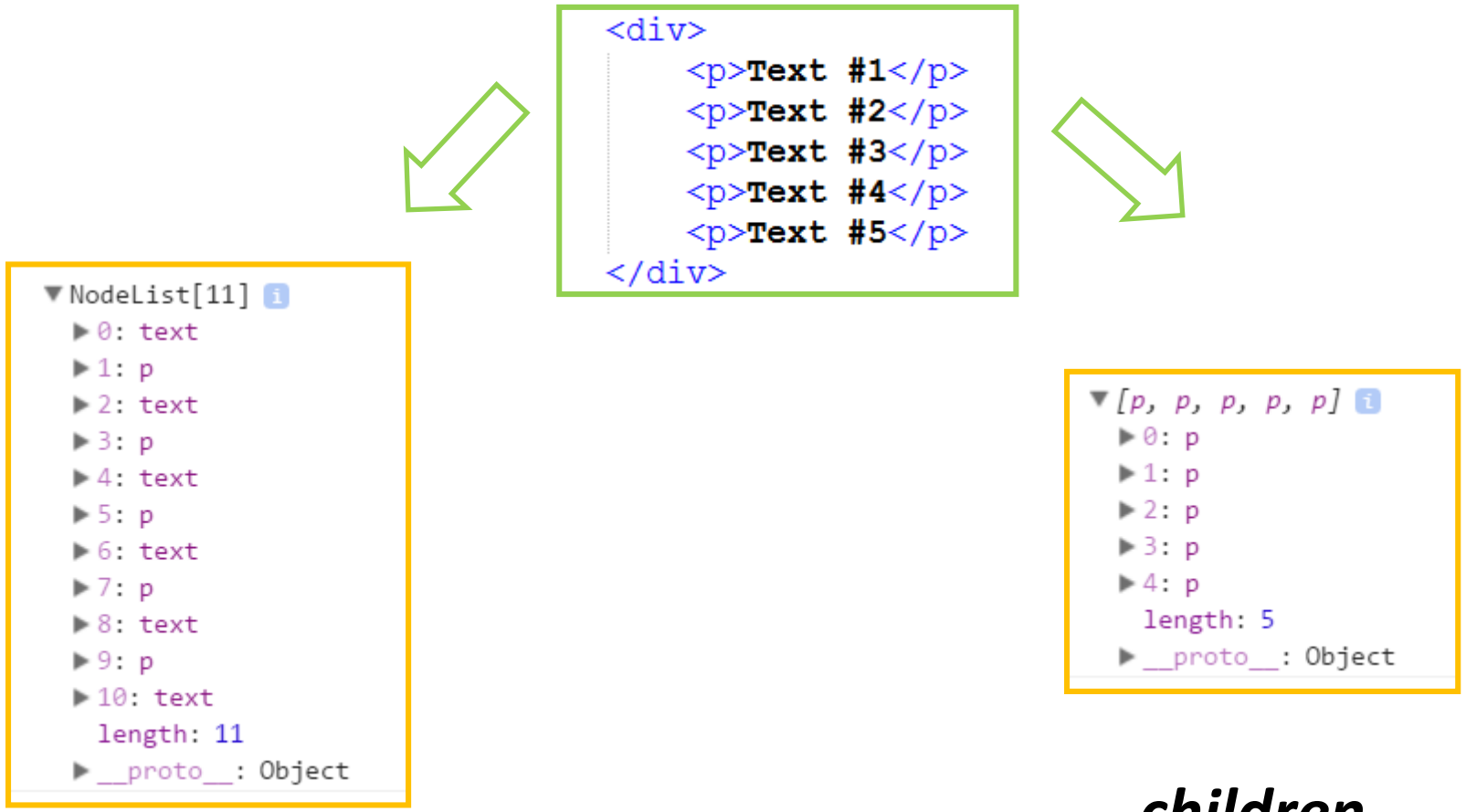


window.document – корень дерева документа

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

window.document.childNodes *– массив с тегами верхнего уровня (т.е.* ***html*** *и* ***doctype****).*

Свойство .children – тоже что и .childNodes но без «текстовых фрагментов»



.childNodes

.children

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

```
<script>
    abc.style.color     = "red";
    abc.style.border    = "3px dotted green";
    abc.style.textAlign = "center";
    abc.innerHTML       = "Hello world!!!";
</script>
<h1 id="abc"></h1>
```

```
<h1 id="abc"></h1>
<script>
    abc.style.color     = "red";
    abc.style.border    = "3px dotted green";
    abc.style.textAlign = "center";
    abc.innerHTML       = "Hello world!!!";
</script>
```

Код из тега script выполняется в тот момент когда браузер дойдёт до тега, если к этому моменту браузер еще не успел обработать разметку, то нашему коду не с чем будет работать.

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?

Используйте заготовку

./source/ex01.html

Теги у которых есть атрибут id доступны сразу из 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.*

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() и .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
45          special.innerHTML = "Hello world!!!";
46
47      </script>
```

*Свойство **.innerHTML** – можно не только считывать но и устанавливать. Изменение свойства **.innerHTML** – автоматически 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.*

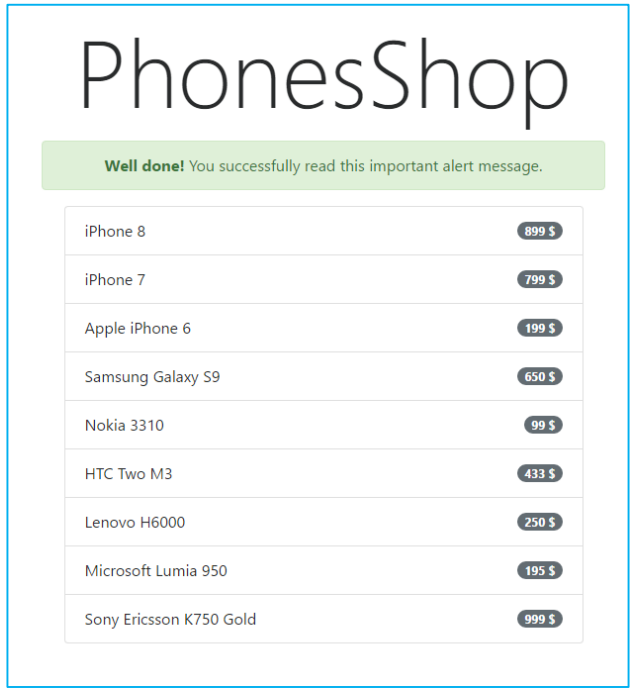
.classList.contains('cat') *– метод проверяет наличие у тега заданного класса (возвращает true/false).*

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

.attributes *– хранит коллекцию с атрибутами тега.*

Немного практики #1

«Переключающийся баннер»



Попробуем сделать баннер который будет показывать предложения из списка товаров. Наша цель:
./source/ex02_demo.html

Возьмите заготовку ./source/ex02.html

«Переключающийся баннер»

```
74  <script>
75      var i = 0;
76
77      setInterval(function(){
78
79          phones_ul.children[i ? i-1 : phones_ul.children.length - 1].
            classList.remove("list-group-item-info");
80
81          alert_div.innerHTML = phones_ul.children[i].innerHTML;
82          phones_ul.children[i].classList.add("list-group-item-info");
83
84          var badge = alert_div.querySelector(".badge-default");
85
86          badge.classList.remove("badge-default");
87          badge.classList.add("badge-danger");
88
89          if(++i >= phones_ul.children.length){
90              i = 0;
91          }
92
93      }, 4000);
94  </script>
```

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>
```

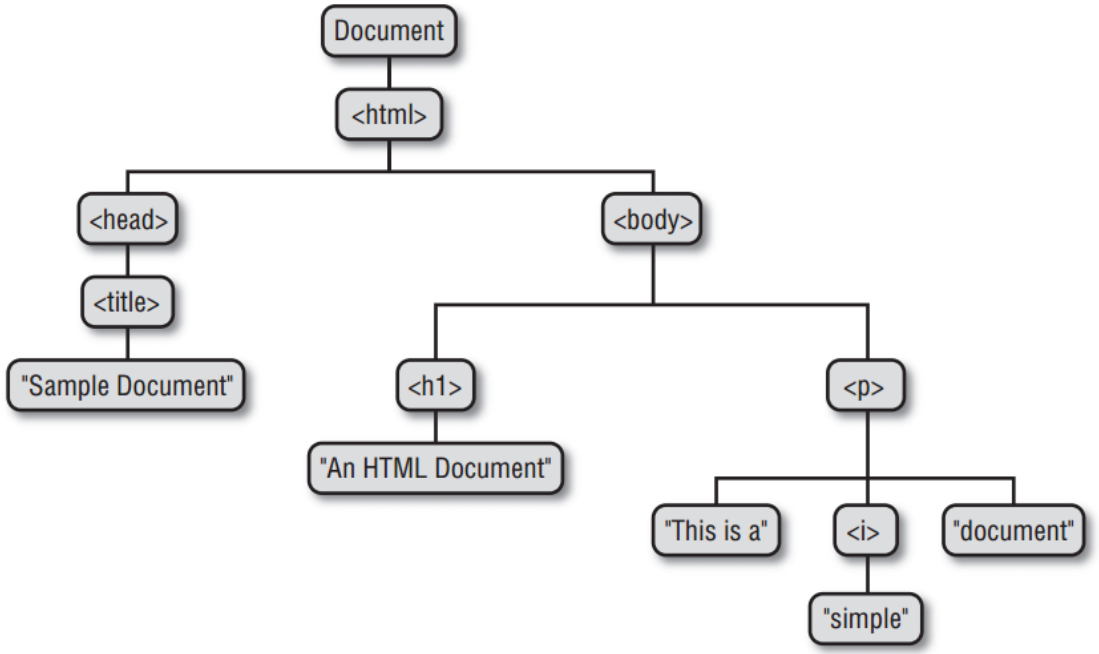
```
▶ ownerDocument: document
  parentElement: null
  parentNode: null
  prefix: null
```

*Удалить элемент из дерева документа можно вызывав у него метод **.remove()**, при этом все его дочерние элементы также исчезнут со странице. 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>
```

Изменение свойств элементов документа – влечет за собой перерисовку документа!

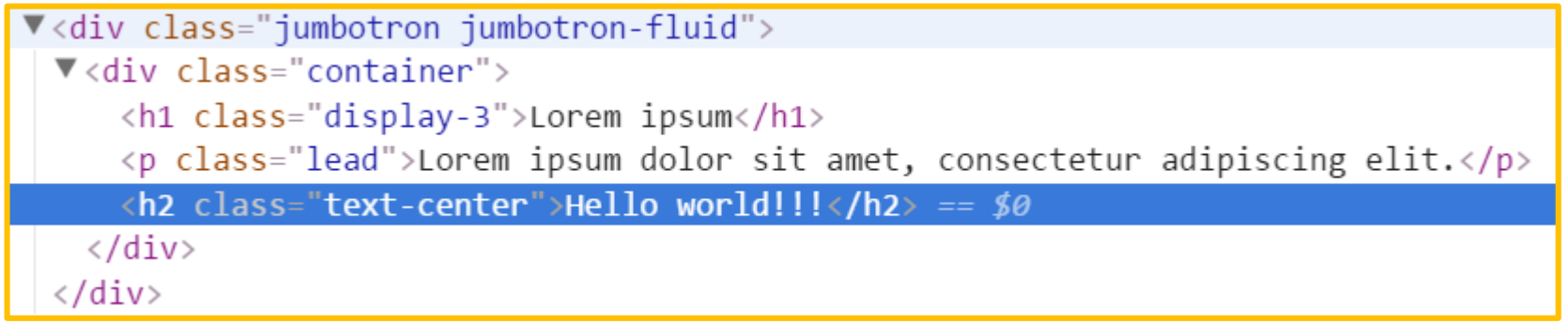
Добавление новых элементов к дереву документа

```
20      <script>
21
22          window.onload = function(){
23              var jumbotron = document.querySelector(".jumbotron > .container");
24
25              var tag = document.createElement("h2");
26
27              tag.innerHTML = "Hello world!!!";
28              tag.classList.add("text-center");
29
30              jumbotron.append(tag);
31          }
32
33      </script>
```

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

.appendChild() *– добавляет элемент к существующему, в качестве последнего потомка.*

Добавление новых элементов к дереву документа



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

.appendChild() *– добавляет элемент к существующему, в качестве последнего потомка.*

.appendChild() *– добавляет элемент к существующему, в качестве последнего потомка. Может быть вызвана для любого существующего тега (даже если он не входит в дерево – другими словами можно формировать ветку еще до того как «присоединять» её к дереву).*

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.*

Перемещение элементов

appendChild*, *insertBefore *– могут перемещать элемент по дереву документа. Т.е. при помощи этих методов можно «перемещать» элемент который уже в дереве, предварительно удалять его со старой позиции нет необходимости.*

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;
    ...
}
```

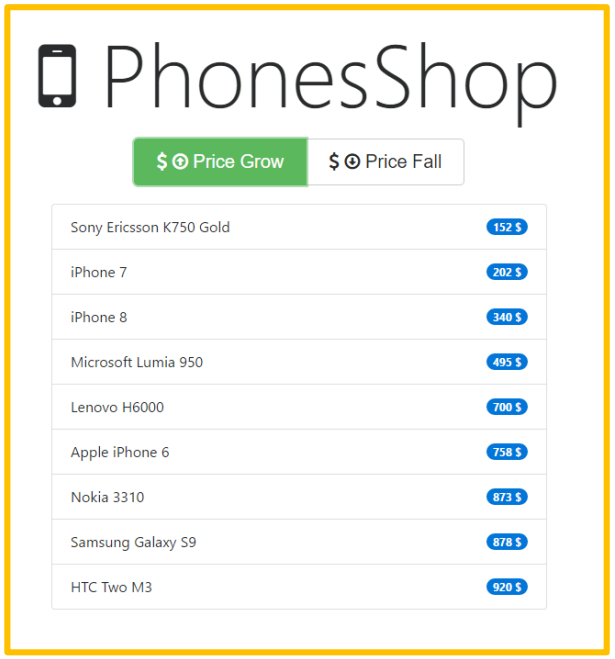
Установка атрибута «класс» для элемента, снимает необходимость задавать в коде 100500 свойств.

!Изменение свойств элемента который еще не вставлен в дерево, не влечёт перерисовку страницы!

Немного практики #2

Добавление элементов и сортировка

```
<!DOCTYPE html>
<html lang="en">
<head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-alpha.6/css/bootstrap.min.css" integrity=
    "sha384-rwoIResjU2yc3z8GV/NPeZWAv56rSmLldC3R/AZzGRnGxQQKnKkoFVhFQhNUwEyJ" crossorigin="anonymous">

    <!-- Bootstrap CSS -->
    <link href="https://maxcdn.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.css" rel="stylesheet" integrity=
    "sha384-wvfXpqpZZVQGK6TAh5PVlGOfQNHSoD2xbE+QkPxCAFlNEevoEH3Sl0sibVcOQVnN" crossorigin="anonymous">

    <style>
        .container, h1{
            margin-top: 10px;
        }

        h1, .btn-group{
            margin-bottom: 20px;
        }

    </style>
    <script>

        function generate_price(){
            return (Math.floor(Math.random()*900) + 100) + " $";
        }

        var phones = [
            {
                title: "iPhone 8",
                price: generate_price()
            },
            {
                title: "iPhone 7",
                price: generate_price()
            },
```



Воспользуйтесь заготовкой: ***./source/ex03.html***

Задание:

1) На основе массива ***phones*** *сгенерировать и вывести разметку со списком телефонов на страницу;*

2) Реализовать сортировку по возрастанию цены (после того как пользователь нажмёт соответствующую кнопоку).

Добавление элементов и сортировка

```
72      //Insert data in document//
73      var ul = document.querySelector("ul.list-group");
74
75      list.forEach(function(elemenent){
76          var li = document.createElement("li");
77          li.className = "list-group-item justify-content-between";
78          li.innerHTML = elemenent.title;
79
80          var span = document.createElement("span");
81          span.className = "badge badge-primary badge-pill");
82          span.innerHTML = elemenent.price;
83          li.appendChild(span);
84
85          ul.appendChild(li);
86      });
87      // End insert data in document//
```

На основе массива создаём разметку...

Добавление элементов и сортировка

```
114     function draw_list(){
115         var ul = document.querySelector("ul.list-group");
116         ul.innerHTML = "";
117         phones.forEach(function(elemenent){
118             var li = document.createElement("li");
119             li.className = "list-group-item justify-content-between";
120             li.innerHTML = elemenent.title;
121 
122             var span = document.createElement("span");
123             span.className = "badge badge-primary badge-pill";
124             span.innerHTML = elemenent.price;
125             li.appendChild(span);
126 
127             ul.appendChild(li);
128         });
129     }
130 
131     draw_list();
132 
133     document.querySelector("button").onclick = function(){
134         phones.sort(function(a, b){
135             return parseInt(a.price) - parseInt(b.price);
136         });
137         draw_list();
138     }
```

Сортируем сначала сам массив и выводим его на место старого списка.

Домашнее задание
/узнать

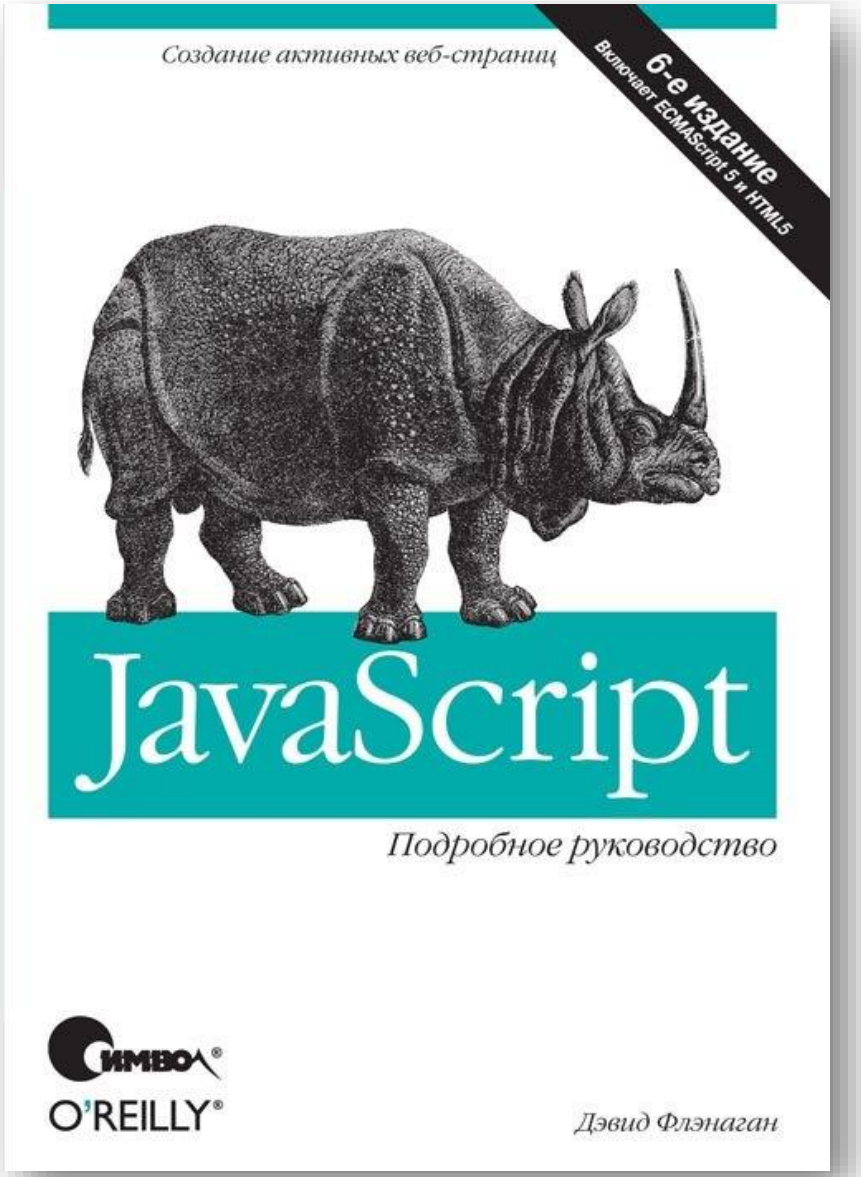
Узнать о следующих свойствах:

.firstChild;
.lastChild;
.nextSibling;
.previousSibling;
.nextElementSibling;
.previousElementSibling;

Вставка элемента в произвольное место документа

*element.**insertAdjacentElement()***
*element.**insertAdjacentHTML()***

Узнайте подробности использования этих методов!



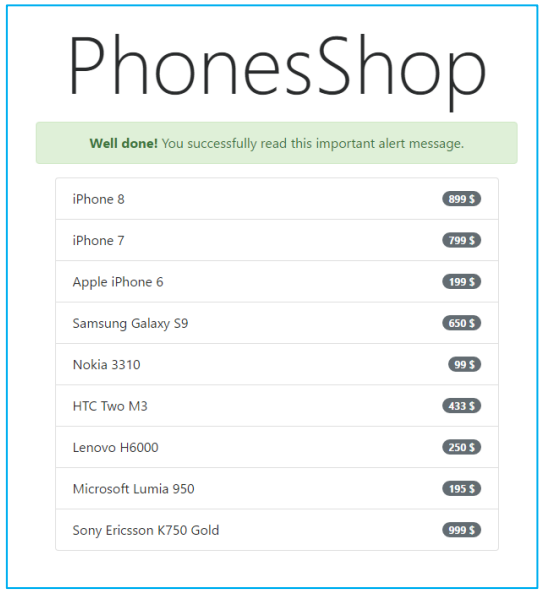
Дэвид Флэнаган *«JavaScript. Подробное руководство»*

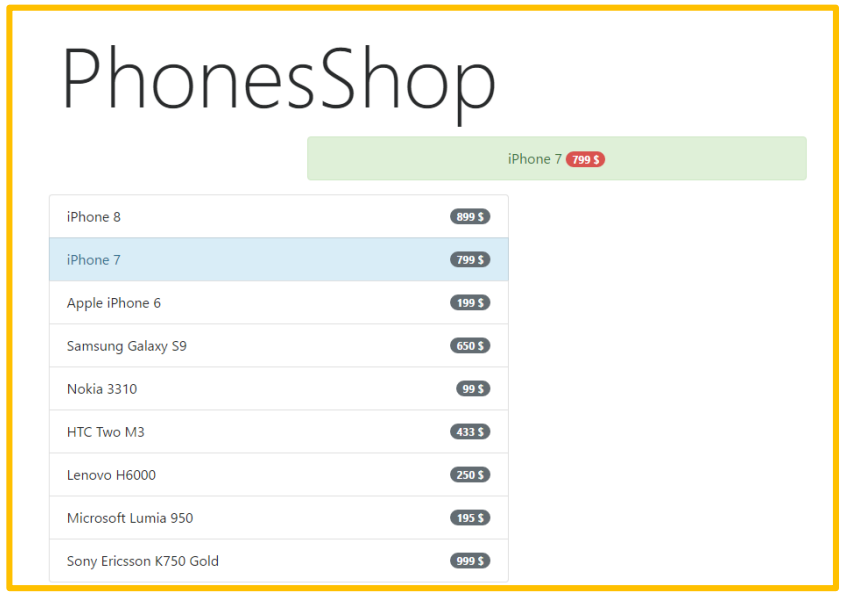
Предварительные знания – лучший помощник в обучении, поэтому к следующему занятию жду, что **вы прочтёте 17-ю главу - «Обработка событий»**.

Домашнее задание
/сделать

Домашнее задание #F.1

«Переключающийся баннер v1.1»



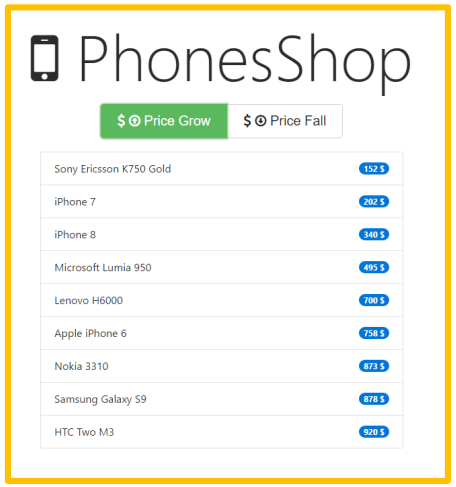


Сделайте баннер который будет «выезжать/уезжать» с предложениями из списка товаров. Наша цель:
./homework/hw_f1_demo.html

Используйте заготовку ./homework/hw_f1.html

Домашнее задание #F.2

Добавление элементов и сортировка



*Используйте заготовку: **./homework/hw_f2.html***

***Задание:** доделайте пример, а именно: реализуйте сортировку как по возрастанию цены и по убыванию, так чтобы кнопками-переключателями пользователь мог выбрать направление сортировки (и по состоянию кнопок было понятно какое направление выбрано).*
***Пример**: ./homework/hw_f2_demo.html*