JavaScript

Основы программирования, часть 5

Web.Dev.
Courses

sp.courses.dp.ua

Объекты / Objects

установить время()

звонить()

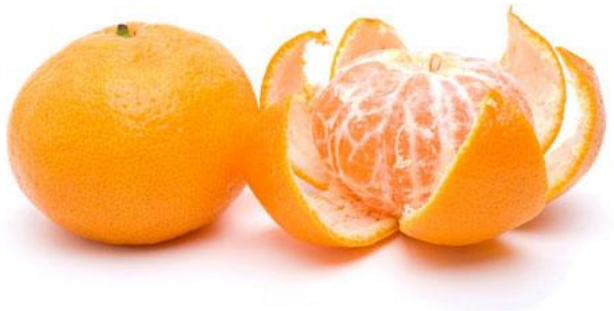
узнать время()

<< Действия()/Методы() >>

пахнуть()

почистить()

выжать сок()



циферблат: механика

вес: 300 грамм

материал: металл

<< Свойства / Аттрибуты >>

параметры

цвет: оранжевый

сахар: 15%

форма: шаровидная

сорт: «Дэнси»

Объекты

Могут содержать в себе другие объекты.

Объекты

Объект *в программировании объединение переменных и функций которые их обрабатывают.*

```
1  <script>
2      var str = "Lorem ipsum dolor sit amet.";
3      console.log(str, typeof(str));
4
5      console.log("Свойство length: ", str.length);
6      console.log("Метод slice()", str.slice(4,9));
7      console.log("Метод toUpperCase()", str.toUpperCase());
8  </script>
```

```
Lorem ipsum dolor sit amet. string                    ex06.html:4
Свойство length:  27                                  ex06.html:6
Метод slice() m ips                                   ex06.html:7
Метод toUpperCase() LOREM IPSUM DOLOR SIT AMET.       ex06.html:8
```

В JavaScript практически всё является объектом, хорошим примером объекта является строка.

Объекты

Объекты *- сложные переменные. Но не потому что их трудно понять или тяжело использовать, а потому что они состоят (сложены) из других переменных и функций.*

```
<script>

    var arr = ["Olga", "Elena", "Petr", "Sergey", "Maria"];

    console.log(arr);
    console.log("Length: " + arr.length);

    arr.push("Ivan");

    console.log(arr);
    console.log("Length: " + arr.length);

    console.log(arr.reverse());

</script>
```

```
▶ ["Olga", "Elena", "Petr", "Sergey", "Maria"]        index.html:5
Length: 5                                             index.html:6
                                                      index.html:10
▶ ["Olga", "Elena", "Petr", "Sergey", "Maria",
  "Ivan"]
Length: 6                                             index.html:11
                                                      index.html:13
▶ ["Ivan", "Maria", "Sergey", "Petr", "Elena",
  "Olga"]
```

Переменные (**данные**) входящие в состав объекта называют **атрибутами** или **свойствами** объекта.

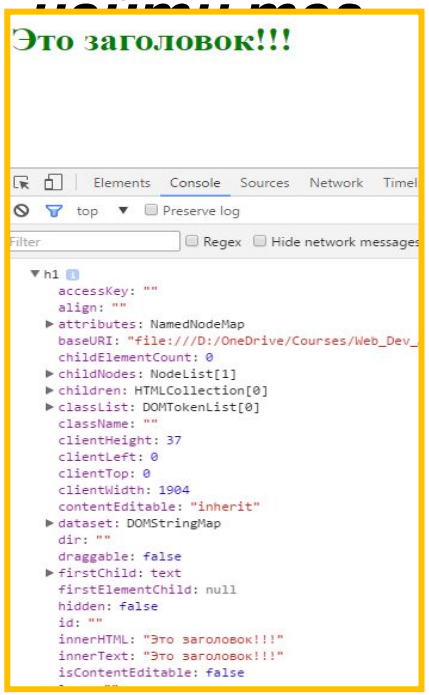
Функции (**действия**) входящие в состав объекта называют **методами** объекта.

Инфраструктура браузера

Каждый элемент HTML-документа это объект

*Весь HTML-документ это один большой объект **window**, который состоит из множества других объектов, некоторые из которых тоже состоят из объектов и т. д.*

Каждому тегу соответствует один объект, который хранит всё содержимое, все стили и все атрибуты тега. И все их можно менять, но первым делом нужной




```
1  <html>
2  <head>
3  <script>
4      window.onload = function(){
5          var element = window.document.querySelector("h1");
6          element.style.color = "green";
7
8          console.dir(element);
9      }
10 </script>
11 </head>
12 <body>
13     <h1>Это заголовок!!!</h1>
14 </body>
15 </html>
```

window.onload –событие «когда документ загружен»

window *– т.н. глобальный объект в котором содержаться все инструменты (функции) для работы с HTML-документом.*

Однако браузер обрабатывает документ построчно, и если наш код находится в заголовке, то во время своей работы, часть документа с видимыми тегами еще не загрузилась. Т.е. у нас есть код который обрабатывает теги, но нет самих тегов.

Но есть событие ***onload*** *у объекта* ***window****. Функция которая подписана на это событие начнёт работать только тогда когда документ будет полностью загружен. Тем самым мы можем быть уверены, что нашему коду будет что обрабатывать.*

Событие window.onload

*Событие **window.onload** случается после окончания загрузки документа в браузер. И код который находится в функции ожидающей событие **window.onload** выполниться только после того как весь документ будет в браузере.*

```
<script>

    window.onload = function(){
        alert("A");
    }
    alert("B");

</script>
```

В противном случае JavaScript-коду просто нечего будет обрабатывать, ведь он будет запушен еще до того как будет загружено тело HTML-документа.

Поиск элементов на странице

document.querySelectorAll(*“css_selector”***)** – *возвращает массив объектов (элементов HTML-документа, тегов) которые соответствуют css-селектору который передан ей в качестве параметра;*

document.querySelector(*“css_selector”***)** – *возвращает первый объект (элемент HTML-документа, тег) из тех которые соответствуют css-селектору который передан ей в качестве параметра.*

id – *элементы у которых есть атрибут id можно использовать без поиска, такие элементы доступны как глобальные переменные (с именем равным id).*

Изменение содержимого элемента и/или его свойс

```
15  <script>
16      window.onload = function(){
17          var a = document.querySelector("h1");
18          a.innerHTML = "This is a SuperShop!";
19          a.style.color = "green";
20      }
21  </script>
```

```
<h1 style="color: green;">This is a SuperShop!</h1>
```

У всех тегов (элементов HTML-документа) есть ряд свойства определяющие его содержимое и внешний вид:

***.innerHTML** – свойство определяющее (или задающее) содержимое тега (его контент), т.е. всё то что находиться между открывающимся и закрывающимся тегом;*

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

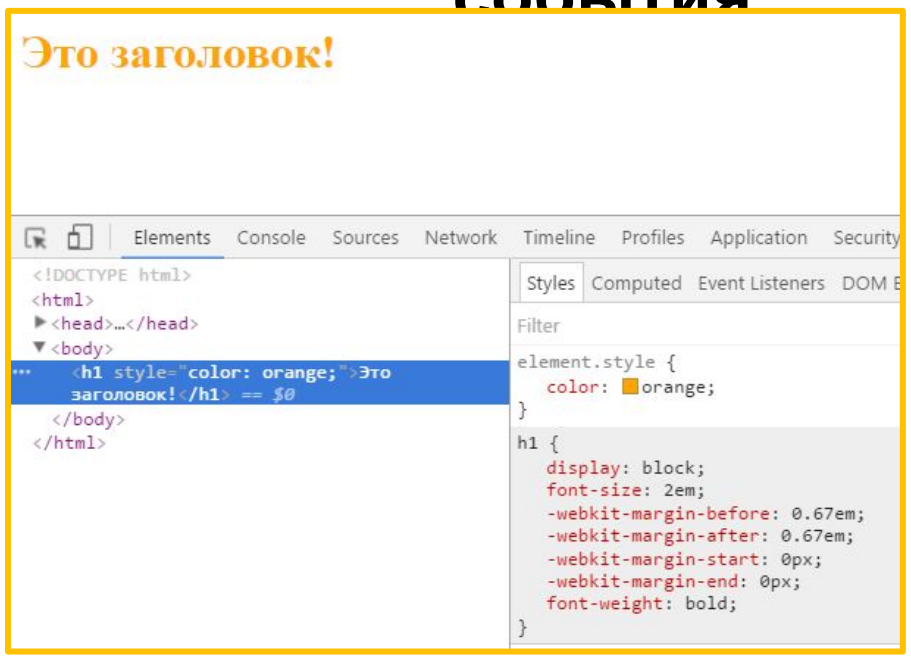
Объекты и

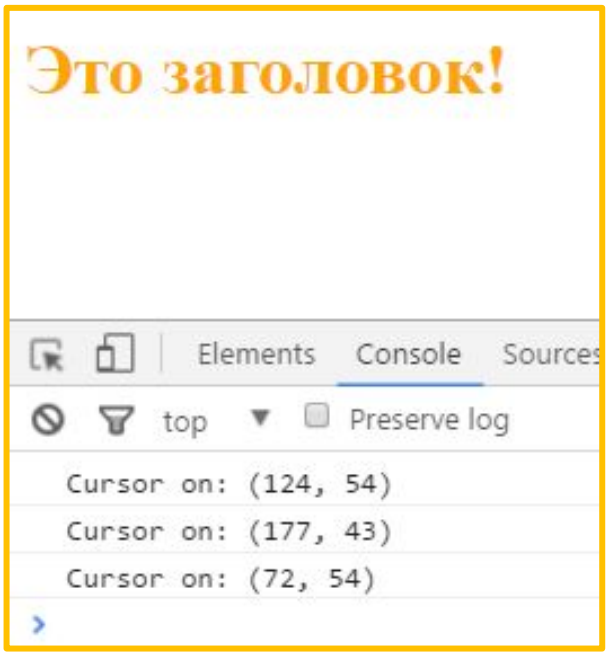
```
<!DOCTYPE html>
<html>
<head>
<script>

    window.onload = function(){

        var element = document.querySelector("h1");

        element.onclick = function(event_info){
            event_info.target.style.color = "orange";
            console.log("Cursor on: (" + event_info.pageX + ", " + event_info.pageY + ")");
        }
        element.ondblclick = function(e){
            e.target.style.color = "red";
        }

    }

</script>
</head>
<body>
    <h1>Это заголовок!</h1>
</body>
</html>
```

На событие можно подписываться присваивая функции-обработчики соответствующим свойствам элементов-объектов.

В качестве первого параметра функция-обработчик события принимает объект с подробной ***информацией о событии****.*

Объекты и события





На событие можно подписываться присваивая функции-обработчики соответствующим свойствам элементов-объектов.

В качестве первого параметра функция-обработчик события принимает объект с подробной ***информацией о событии****.*

Удаление элемента

```
<!DOCTYPE html>
<html>
<head>
<script>

    window.onload = function(){

        var element = document.querySelector("h1");

        element.onclick = function(event_info){
            event_info.target.remove();
        }
    }

</script>
</head>
<body>
    <h1>Это заголовок!</h1>
</body>
</html>
```

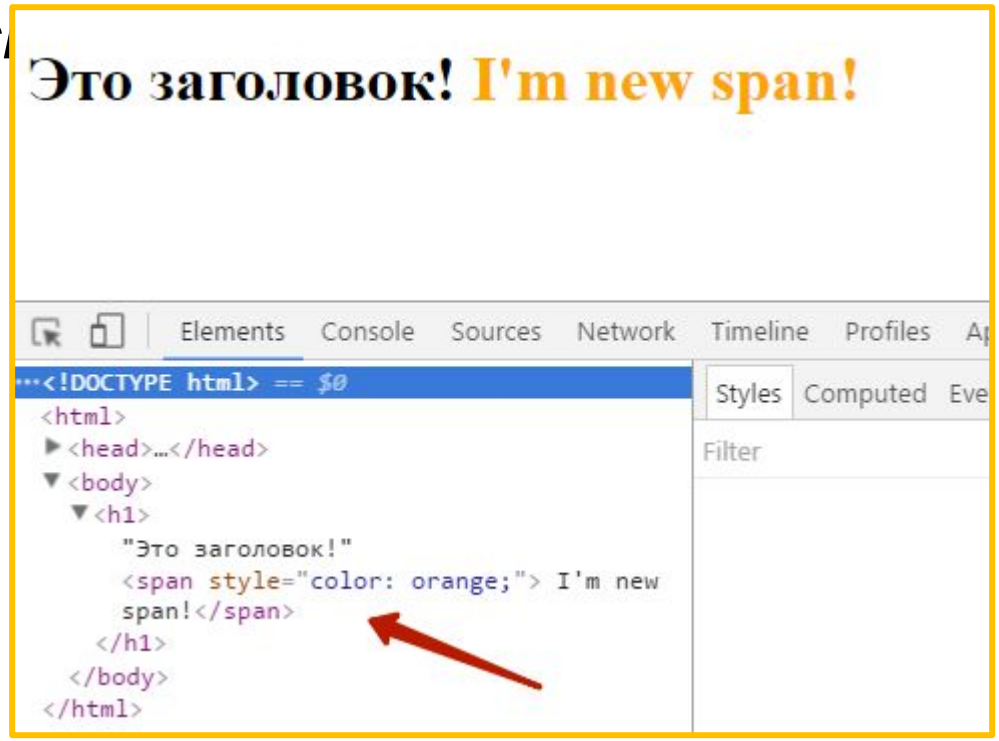
*Метод **.remove()** – удаляет элемент (объект, тег) из HTML-документа. Если у него есть дочерние элементы (теги-потомки) он также удалятся.*

Добавление элемента на страницу

Чтобы добавить элемент на страницу, необходимо определить к какому из существующих элементов его необходимо прикрепить.

```
<!DOCTYPE html>
<html>
<head>
<script>

    window.onload = function(){

        var element = document.createElement("span");
        element.innerHTML = " I'm new span!";
        element.style.color = "orange";

        var h1 = document.querySelector("h1");

        h1.appendChild(element);

    }

</script>
</head>
<body>
    <h1>Это заголовок!</h1>
</body>
</html>
```

Добавление элемента на с



document.createElement(*"tag_name"*) – *метод создающий пустой элемент (объект, тег), которые еще не входит в документ, но его свойства уже можно наполнять необходимыми данными;*

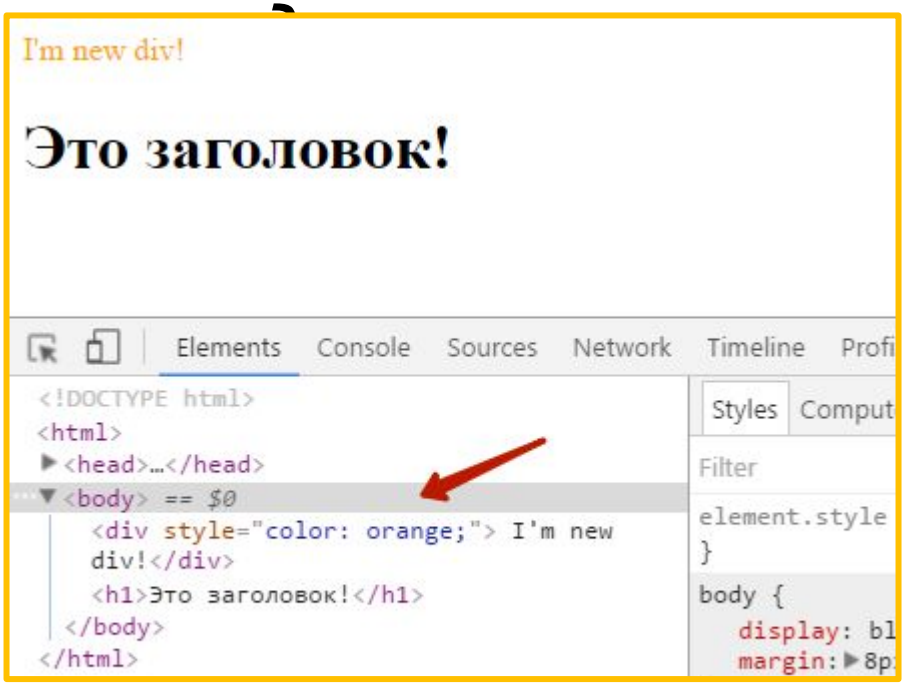
.appendChild(*new_element****)*** *– метод добавляющий новый элемент к потомкам того элемента у которого* ***.appendChild*** *вызывается.*

Изменение позиции элемента в документе

```
<!DOCTYPE html>
<html>
<head>
<script>

    window.onload = function(){

        var element = document.createElement("div");
        element.innerHTML = " I'm new div!";
        element.style.color = "orange";

        var h1 = document.querySelector("h1");

        document.body.insertBefore(element, h1);
    }

</script>
</head>
<body>
    <h1>Это заголовок!</h1>
</body>
</html>
```

Метод ***.appendChild ()*** всегда добавляет элемент в конец списка. Но есть возможность задать позицию вставки элемента среди потомком тега. Для этого существует метод ***.insertBefore()***.

Изменение позиции элемента в



.insertBefore(*new_el, ref_el****)*** *– добавляет элемент в список дочерних элементов родителя перед указанным элементом.* ***new_el*** *– элемент который вставляется,* ***ref_el*** *– элемент перед которым вставляется элемент.*

Не менее полезные свойства объектов-тегов!

.parentNode *– свойство, которое содержит ссылку на родительский элемент (тег).*

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

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

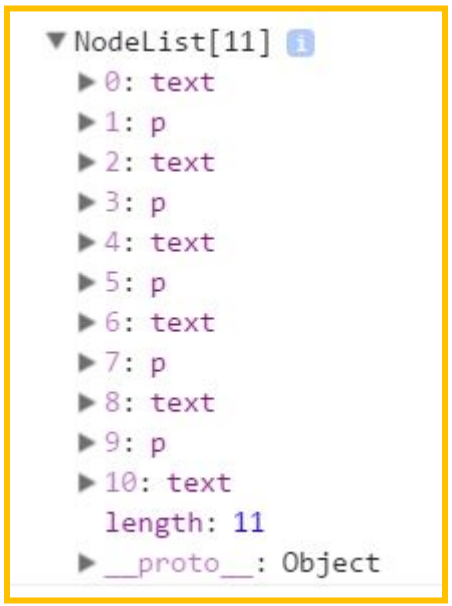
.classList.add('cat') *– метод добавляет класс к тегу (если есть другие классы то они остаются).*

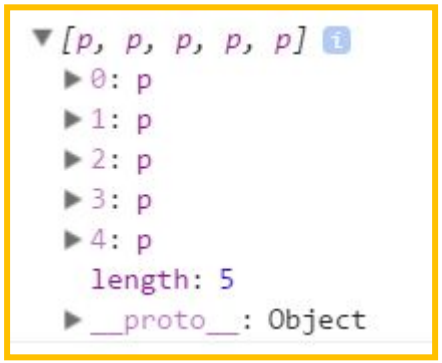
.classList.remove('cat') *– метод удаляет класс у тегу (если есть другие классы то они не затрагиваются).*

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

Свойства .children и .nodeChild
массивы с потомками тега

```
<div>
    <p>Text #1</p>
    <p>Text #2</p>
    <p>Text #3</p>
    <p>Text #4</p>
    <p>Text #5</p>
</div>
```





.childNodes

.children

Таймеры

В JavaScript есть возможность отложить какое-то действие на время, или выполнять действие многократно через заданные интервалы времени.

setTimeout(some_func, time, param) *– функция которая после истечения времени заданного в параметре* ***time*** *(задаётся в миллисекундах) выполнить один раз функцию имя которой передана в параметре* ***some_func****, если этой функции нужно передать какие-либо параметры их можно заранее указать в* ***param.***

setInterval(some_func, time, param) *– функция которая будет через каждый промежуток времени* ***time*** *(задаётся в миллисекундах) выполнять функцию имя которой передана в параметре* ***some_func****, если этой функции нужно передать какие-либо параметры их можно заранее указать в* ***param.***

DOM – Document Object Model

(объектная модель документа)

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

https://learn.javascript.ru/document

Всё, что мы рассмотрели и есть стандарт DOM

Немного практики

Немного практики

```
<!DOCTYPE html>
<html>
<head>
<title>DOM.task1</title>
<script>
    window.onload = function(){
        //Начало вашего кода.....

        //.....Конец вашего кода
    }
</script>
</head>
<body>
<h1>Каталог телефонов <button>Отсортировать</button></h1>
    <div id="phone_list">
        <div>iPhone 3Gs <span>582$</span></div>
        <div>Lenovo <span>997$</span></div>
        <div>HTC <span>698$</span></div>
        <div>Samsung <span>242$</span></div>
        <div>Fly <span>793$</span></div>
        <div>Asus <span>771$</span></div>
        <div>Nokia <span>108$</span></div>
        <div>LG <span>214$</span></div>
        <div>IPhone 5s <span>216$</span></div>
        <div>IPhone 6+ <span>326$</span></div>
        <div>IPhone 6s+ <span>799$</span></div>
    </div>
</body>
</html>
```

Каталог телефонов Отсортировать

iPhone 3Gs 582$
Lenovo 997$
HTC 698$
Samsung 242$
Fly 793$
Asus 771$
Nokia 108$
LG 214$
IPhone 5s 216$
IPhone 6+ 326$
IPhone 6s+ 799$

По нажатию на кнопку необходимо отсортировать список по возрастанию цены.

Скопируйте код:
http://courses.dp.ua/files/is/tasks/task1.html

Немного практики

```
<script>
    window.onload = function(){
        //Начало вашего кода.....
        var btn = document.querySelector("button");
        btn.onclick = function(){
            var list = document.querySelector("h1 + div");
            var elements = list.children;
            for(var i = 0; i < elements.length - 1; i++){
                for(var j = 0; j < elements.length - 1 - i; j++){
                    var price1 = parseInt(elements[j].querySelector("span").innerHTML);
                    var price2 = parseInt(elements[j+1].querySelector("span").innerHTML);
                    if(price1 > price2){
                        list.insertBefore(elements[j+1], elements[j]);
                    }
                }
            }
        }
        //.....Конец вашего кода
    }
</script>
```

По нажатию на кнопку необходимо отсортировать список по возрастанию цены.

Домашнее задание

Немного практики

```
<!DOCTYPE html>
<html>
<head>
<title>DOM.task1</title>
<script>
    window.onload = function(){
        //Начало вашего кода.....

        //.....Конец вашего кода
    }
</script>
</head>
<body>
<h1>Каталог телефонов <button>Отсортировать</button></h1>
    <div id="phone_list">
        <div>iPhone 3Gs <span>582$</span></div>
        <div>Lenovo <span>997$</span></div>
        <div>HTC <span>698$</span></div>
        <div>Samsung <span>242$</span></div>
        <div>Fly <span>793$</span></div>
        <div>Asus <span>771$</span></div>
        <div>Nokia <span>108$</span></div>
        <div>LG <span>214$</span></div>
        <div>IPhone 5s <span>216$</span></div>
        <div>IPhone 6+ <span>326$</span></div>
        <div>IPhone 6s+ <span>799$</span></div>
    </div>
</body>
</html>
```

Каталог телефонов Отсортировать

iPhone 3Gs 582$
Lenovo 997$
HTC 698$
Samsung 242$
Fly 793$
Asus 771$
Nokia 108$
LG 214$
IPhone 5s 216$
IPhone 6+ 326$
IPhone 6s+ 799$

По нажатию на кнопку необходимо отсортировать список по возрастанию цены. При повторном нажатии сортировка должна выполниться по убыванию цены. И так по очереди переключать направление сортировки.

Скопируйте код:
http://courses.dp.ua/files/is/tasks/task1.html