TEL-RAN
by Starta Institute

Frontend Basic
LESSON 10

Anna Khachaturyan

TEL-RAN
by Starta Institute

Анна Хачатурян

Front End/Gen Tech Teacher

- Since 2018 in IT
- Full Stack Developer at Web Magnat
- QA Engineer/Web Developer at Central Bank of RA
- Lecturer at Plekhanov Russian University of Economics
- TA at Picsart Academy
- Teacher at Tel-Ran

https://t.me/anny_khachaturyan

TEL-RAN
by Starta Institute

ВАЖНО:

- Если у Вас возник вопрос в процессе занятия, пожалуйста, поднимите руку и дождитесь, пока преподаватель закончит мысль и спросит Вас, также можно задать вопрос в чате или когда преподаватель скажет, что начался блок вопросов.
- Организационные вопросы по обучению решаются с кураторами, а не на тематических занятиях.
- Вести себя уважительно и этично по отношению к остальным участникам занятия.
- Во время занятия будут интерактивные задания, будьте готовы включить камеру или демонстрацию экрана по просьбе преподавателя.

TEL-RAN
by Starta Institute

ПЛАН ЗАНЯТИЯ

1. Повторение изученного
2. Вопросы по повторению
3. Введение в DOM
4. Практика

TEL-RAN
by Starta Institute

1 ПОВТОРЕНИЕ

TEL-RAN
by Starta Institute

2 ВОПРОСЫ ПО ПОВТОРЕНИЮ

TEL-RAN
by Starta Institute

3 ВВЕДЕНИЕ В DOM

DOM

Основой HTML-документа являются теги.

В соответствии с объектной моделью документа («Document Object Model», коротко DOM), каждый HTML-тег является объектом. Вложенные теги являются «детьми» родительского элемента. Текст, который находится внутри тега, также является объектом.

Все эти объекты доступны при помощи JavaScript, мы можем использовать их для изменения страницы.

Например, document.body – объект для тега <body>.

Если запустить этот код, то <body> станет красным:

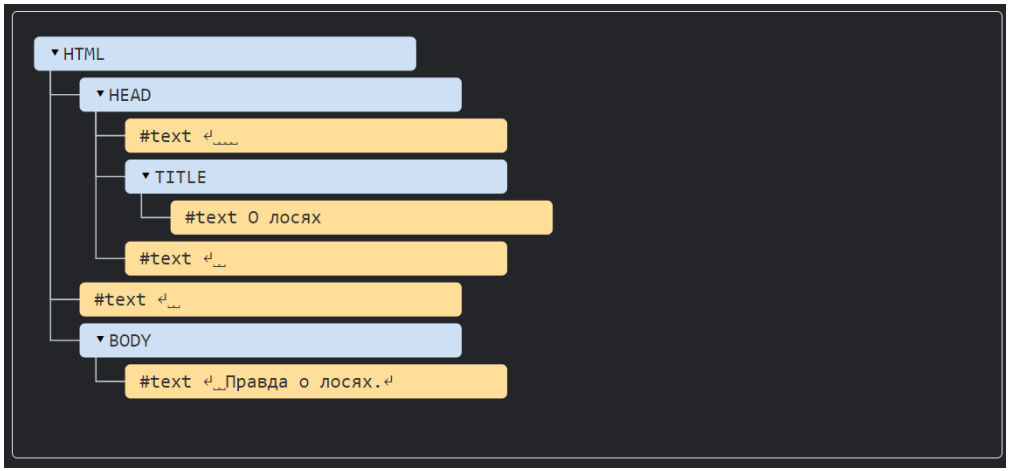
```
document.body.style.background = 'red'; // сделать фон красным
```

Это был лишь небольшой пример того, что может DOM. Скоро мы изучим много способов работать с DOM, но сначала нужно познакомиться с его структурой.

Пример DOM

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

DOM – это представление HTML-документа в виде дерева тегов. Вот как оно выглядит:



Каждый узел этого дерева – это **объект**.

Теги являются **узлами-элементами** (или просто элементами). Они образуют структуру дерева: <html> – это корневой узел, <head> и <body> его дочерние узлы и т.д.

Текст внутри элементов образует **текстовые узлы**, обозначенные как #text. Текстовый узел содержит в себе только строку текста. У него не может быть потомков, т.е. он находится всегда на самом нижнем уровне.

Например, в теге <title> есть текстовый узел "О лосях".

Обратите внимание на специальные символы в текстовых узлах:

- перевод строки: ↵ (в JavaScript он обозначается как \n)
- пробел: ␣

Пробелы и переводы строки – это полноправные символы, как буквы и цифры. Они образуют текстовые узлы и становятся частью дерева DOM. Так, в примере выше в теге <head> есть несколько пробелов перед <title>, которые образуют текстовый узел #text (он содержит в себе только перенос строки и несколько пробелов).

Существует всего два исключения из этого правила:

1. По историческим причинам пробелы и перевод строки перед тегом <head> игнорируются
2. Если мы записываем что-либо после закрывающего тега </body>, браузер автоматически перемещает эту запись в конец body, поскольку спецификация HTML требует, чтобы всё содержимое было внутри <body>. Поэтому после закрывающего тега </body> не может быть никаких пробелов.

Все, что есть в HTML, даже комментарии, является частью DOM.

Даже директива <!DOCTYPE...>, которую мы ставим в начале HTML, тоже является DOM-узлом. Она находится в дереве DOM прямо перед <html>. Мы не будем рассматривать этот узел, мы даже не рисуем его на наших диаграммах, но он существует.

Даже объект document, представляющий весь документ, формально является DOM-узлом.

Существует 12 типов узлов. Но на практике мы в основном работаем с 4 из них:

1. document – «входная точка» в DOM.
2. узлы-элементы – HTML-теги, основные строительные блоки.
3. текстовые узлы – содержат текст.
4. комментарии – иногда в них можно включить информацию, которая не будет показана, но доступна в DOM для чтения JS.

Чтобы посмотреть структуру DOM в реальном времени, попробуйте Live DOM Viewer http://software.hixie.ch/utilities/js/live-dom-viewer/

Просто введите что-нибудь в поле, и ниже вы увидите, как меняется DOM.

Доп. https://learn.javascript.ru/dom-nodes

Поиск элемента в DOM

TEL-RAN
by Starta Institute

querySelectorAll

Самый универсальный метод поиска – это elem.querySelectorAll(css), он возвращает все элементы внутри elem, удовлетворяющие данному CSS-селектору.

Следующий запрос получает все элементы <li>, которые являются последними потомками в <ul>:

HTML

```
    <ul>
        <li>First</li>
        <li>Middle</li>
        <li>Last</li>
    </ul>

    <ul>
        <li>Text1</li>
        <li>Text2</li>
        <li>Text3</li>
        <li>Text4</li>
    </ul>
```

JS

```
let elements = document.querySelectorAll('li');

console.log(elements); //получаем список
```

Свойство **innerHTML** позволяет получить HTML-содержимое элемента в виде строки.

Мы также можем изменять его. Это один из самых мощных способов менять содержимое на странице.

Элементы можно получить с помощью индекса:

```
console.log(elements[0].innerHTML);
console.log(elements[1].innerHTML);
```

Либо с помощью цикла:

```
    let elements = document.querySelectorAll('li');

    for (let elem of elements) {
      console.log(elem.innerHTML);
    }
```

querySelector

Метод elem.querySelector(css) возвращает первый элемент, соответствующий данному CSS-селектору.

Иначе говоря, результат такой же, как при вызове elem.querySelectorAll(css)[0], но он сначала найдёт все элементы, а потом возьмёт первый, в то время как elem.querySelector найдёт только первый и остановится. Это быстрее, кроме того, его короче писать.

Как получить конкретно текст из элемента?

Свойство **innerText** - показывает всё текстовое содержимое, которое не относится к синтаксису HTML. То есть любой текст, заключённый между открывающими и закрывающими тегами элемента будет записан в innerText. Причём если внутри innerText будут ещё какие-либо элементы HTML со своим содержимым, то он проигнорирует сами элементы и вернёт их внутренний текст.

innerHTML - покажет текстовую информацию ровно по одному элементу. В вывод попадёт и текст, и разметка HTML-документа, которая может быть заключена между открывающими и закрывающими тегами основного элемента.

Работа с картинками и ссылками и со всеми атрибутами

TEL-RAN
by Starta Institute

Свойство **setAttribute**() предназначен для добавления в элемент нового атрибута с указанным значением. Если указанный атрибут уже существует, то значение этого атрибута изменяется на значение, переданное методу в качестве второго аргумента.

```
element.setAttribute(имяАтрибута, значение);
```

Пример:

HTML

```
    <a>Google</a>
```

JS

```
document.querySelector('a').setAttribute('href', 'https://www.google.com/');
```

Еще несколько свойств.

```
let elem = document.querySelector('p');

console.log(elem.hasAttribute('id')); //проверяет наличие атрибута

console.log(elem.getAttribute('id')); //получает значение атрибута

elem.setAttribute('class', 'text'); //устанавливает атрибут

elem.removeAttribute('id'); //удаляет атрибут
```

Пример:

TEL-RAN
by Starta Institute

HTML

```
    <a id="google">Google</a>
    <a id="facebook">Facebook</a>
```

JS

```
let link1 = document.querySelector('#google');
link1.setAttribute('href', 'https://www.google.com/');
link1.setAttribute('target', '_blank');

let link2 = document.querySelector('#facebook');
link2.setAttribute('href', 'https://www.facebook.com/');
link2.setAttribute('target', '_blank');
```

TEL-RAN
by Starta Institute

4 ПРАКТИКА

Задача: Написать программу, которая находит параграф по классу и заменяет в нем текст на “привет”.

```
document.querySelector('.paragraph').innerText = 'привет';
```

Задача: Написать программу, которая находит параграфы и заменяет в каждом из них текст на “привет”.

```
let elements = document.querySelectorAll('p');

for (let elem of elements) {
  elem.innerText = 'привет';
}
```

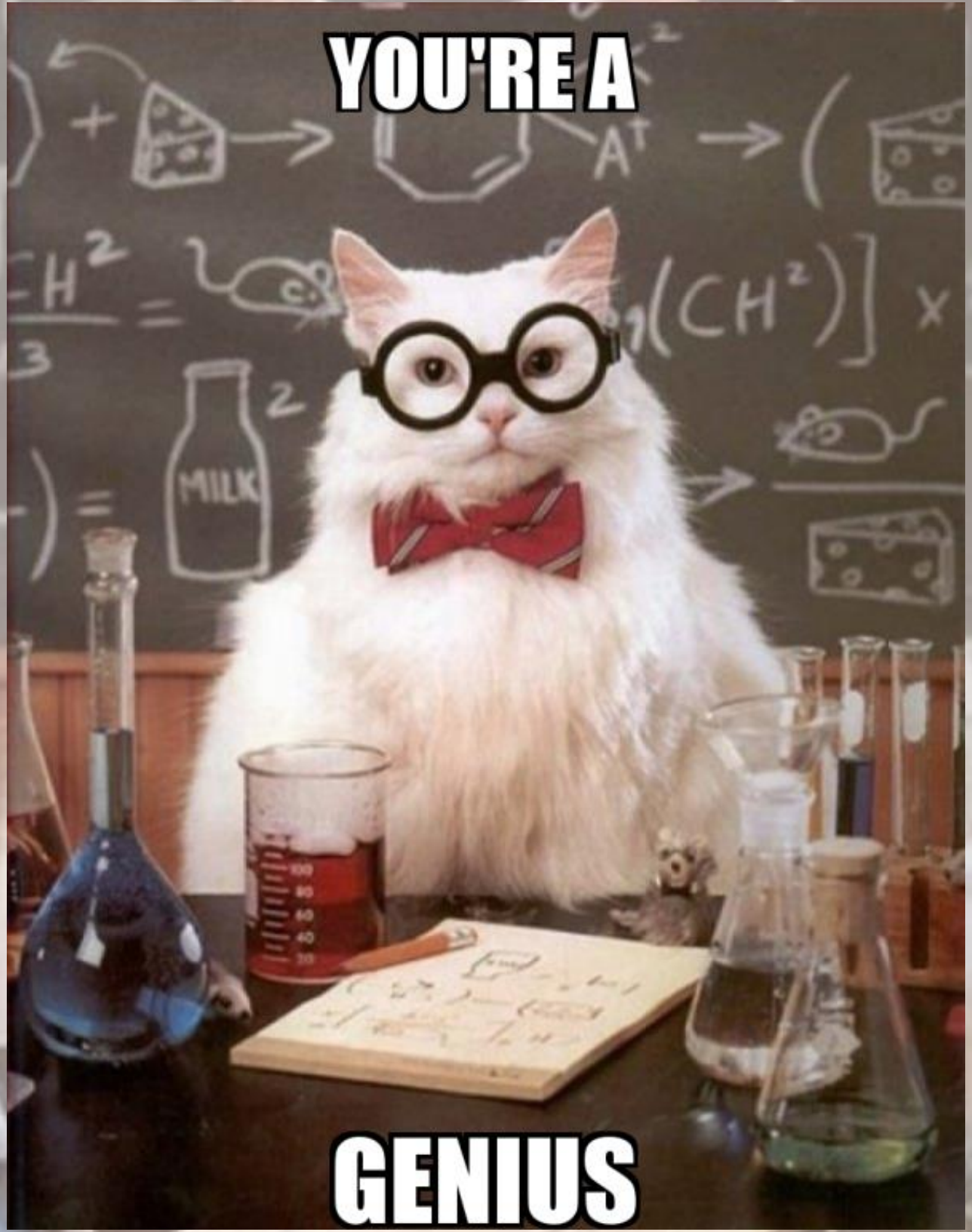
Задача: Написать скрипт, который находит параграфы и первым 3 параграфам меняет текст на указанные пользователем значения.

Задача: Написать скрипт, который находит картинки в блоке с классом main и первым 5 картинкам меняет url на указанное значение.

```
let images = document.querySelectorAll('.main > img');
let url = 'https://t3.ftcdn.net/jpg/03/31/21/08/360_F_331210846_9yjYz8hRqqvezWIIIcr1sL8UB4zyhyQg.jpg'

//Простой вариант

for(let i = 0; i < 5; i++){
    images[i].setAttribute('src', url);
}
```

```
//Сложный вариант

let i = 0;
for(let img of images){
    i++;
    if(i==6){
        break;
    }
    img.setAttribute('src', url);
}
```

Задача: Написать скрипт, который находит все ссылки на странице и формирует массив со всеми адресами. В итоге этот массив необходимо вывести в консоль.


YOU'RE A
MILK
GENIUS