Курс: Javascript. Событийно-ориентированное программирование

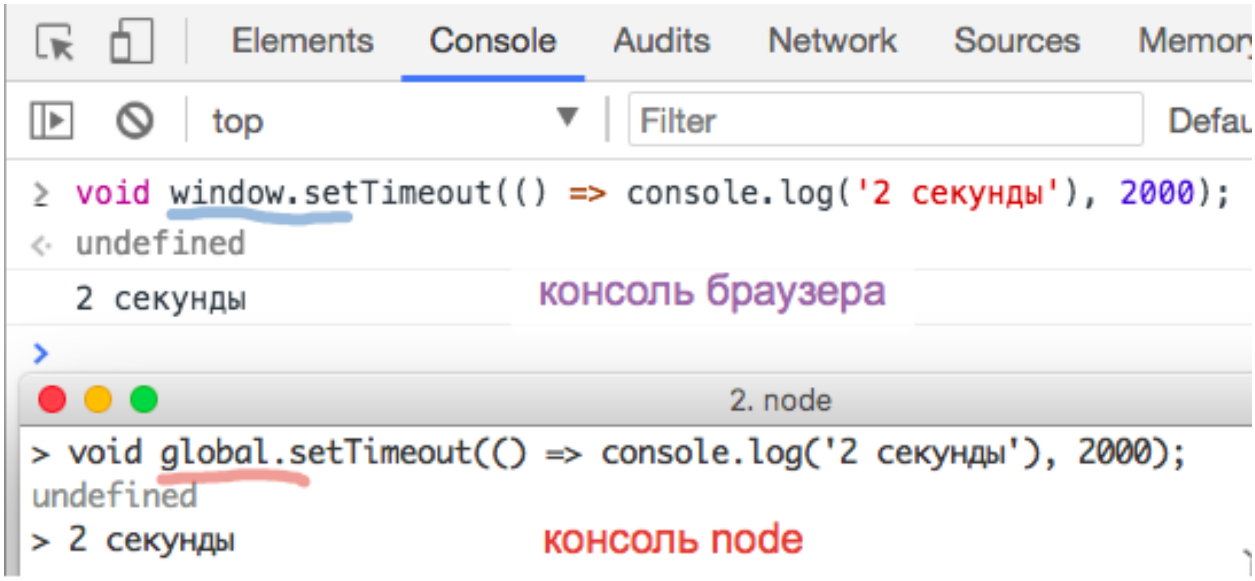
Тема 1. Объектная модель документа.
Манипулирование элементами по их селекторам (querySelector). Изменение свойств каскадных стилей. Создание элементов и фрагментов DOM (documentFragment).

Теоретические сведения
План:
Что такое DOM?
Что такое селектор?
Что такое фрагмент?
Как получать доступ к элементам по их селекторам?
Как создавать и изменять элементы и фрагменты DOM?

Наша задача — научиться управлять элементами веб-страницы с помощью JavaScript.

DOM является первым среди API, которыми «от природы» может оперировать JavaScript. Эта аббревиатура расшифровывается как Document Object Model — объектная модель документа. Под документом понимается веб-страница, загруженная в браузер. Браузер содержит программные компоненты, которые отвечают за загрузку и анализ (парсинг) документа на языке разметки, таком как HTML или SVG. В результате парсинга и возникает DOM — программная иерархическая структура, которая является моделью документа в том смысле, что моделирует взаимоотношения между его составными частями. Они с программной точки зрения представляют собой объекты, т.е. имеют свойства и методы. Также они являются источниками событий. Объект на вершине иерархии называется window. Он является также синоним глобального объекта для JavaScript в браузерной среде.

Функция setTimeout, которая осуществляет асинхронный запуск другой функции, является глобальным методом — методом глобального объекта.



В браузере мы можем использовать обращение window.setTimeout, а в консоли node мы можем использовать обращение global.setTimeout. И в обоих случаях мы можем написать просто setTimeout.

Аналогичным образом, сценарий на языке JavaScript работает с CSS Object Model (Объектная модель CSS) —древовидной структурой, подобной DOM, содержащей образ свойств и селекторов, к которым эти свойства должны быть применены, а также определяющей API для работы с этой структурой из JS.

Объекты DOM являются экземплярами соответствующих классов. Например, тело веб-страницы, представленное элементом <body></body>, это экземпляр класса HTMLBodyElement.

Класс Element является самым общим базовым классом, от которого наследуют все объекты в составе документа. В зависимости от языка разметки, такого как HTML или SVG, мы можем работать с классом HTMLElement или SVGElement.
Для DOM характерно говорить о реализации интерфейсов. Класс Elements наследует от своего родительского интерфейса Node и, по цепочке, от ещё более обобщённого интерфейса EventTarget.

От интерфейса Node наследуют свои методы и свойства такие интерфейсы как Document, Element, DocumentFragment.

DOM отнюдь не стопроцентно совпадает с JavaScript, например, в аспекте типов данных. В DOM существует такой тип как NodeList (document.querySelectorAll(...)), а в JavaScript его не существует. Кроме того, в DOM есть живые коллекции...

Упрощённо, схема работы браузера такова:

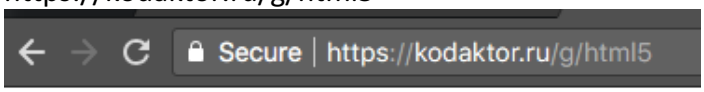
1. Загрузить и проанализировать HTML
2. Построить DOM
3. Загрузить и проанализировать стили

4. Построить CSSOM
5. На основе DOM и CSSOM построить дерево отображения (render tree)
6. Отрисовать результат

Перейдите по адресу https://kodaktor.ru/html5

и изучите структуру простой веб-страницы на языке разметки HTML5.

В режиме одиночной веб-страницы мы видим результат отображения этих тегов: https://kodaktor.ru/g/html5



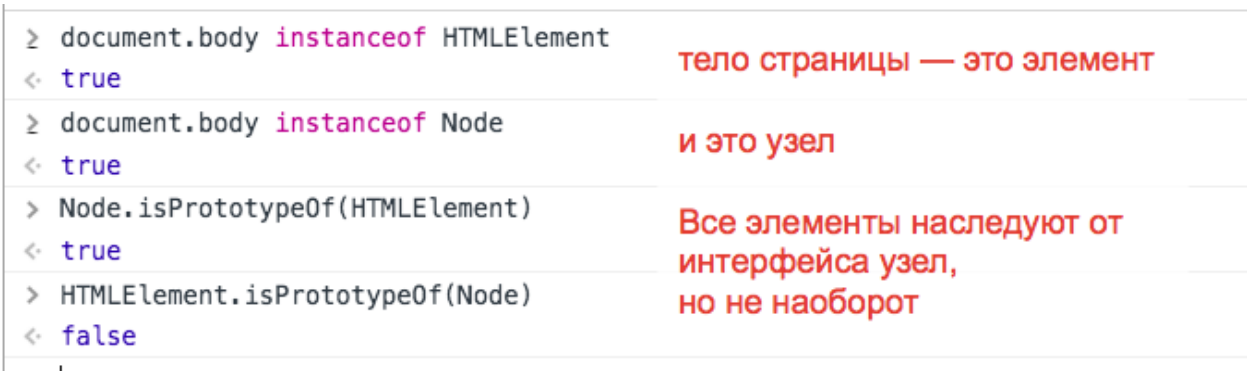
HTML5

W3C

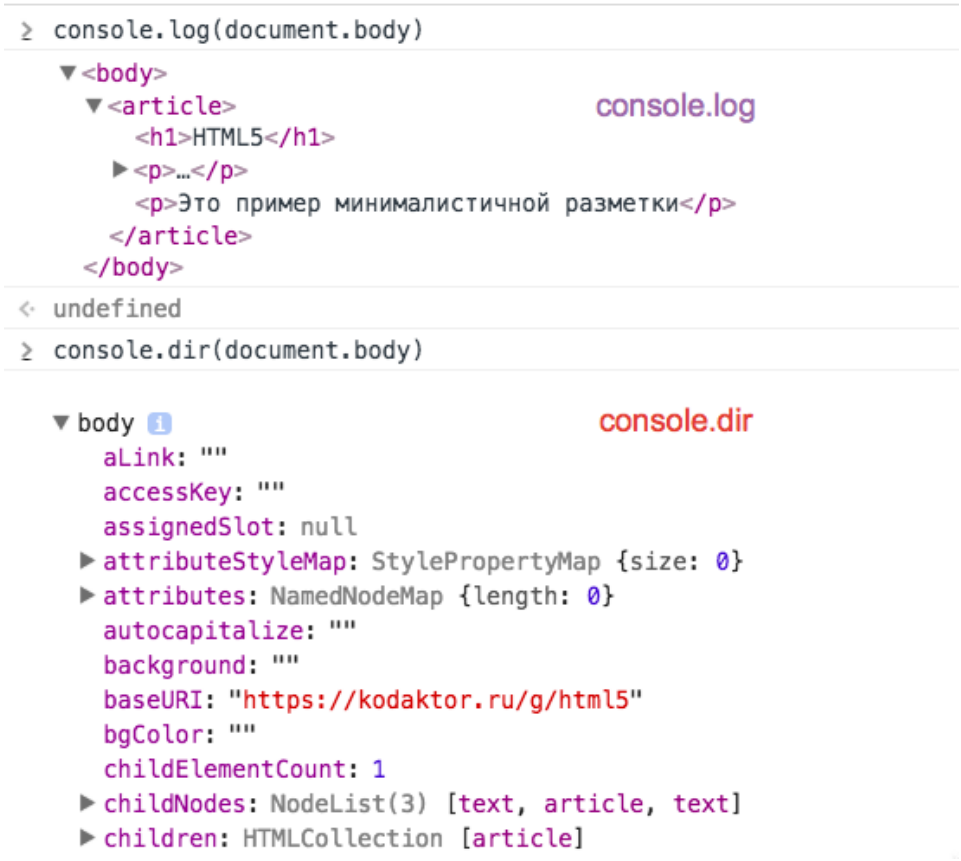
Это пример минималистичной разметки

Перейдя в консоль (веб-инструменты), мы теперь можем исследовать структуру DOM для этого документа.

Например, исследуем тело веб-страницы.



Когда мы просто пишем в консоли document.body, это то же самое, что написать console.log(document.body). Есть ещё удобный метод dir, позволяющий показать информацию в более подробной древовидной форме:



С точки зрения узлов (Node) у тела три дочерних узла: элемент article и два переноса строки. И у него только один дочерний HTMLElement — article.

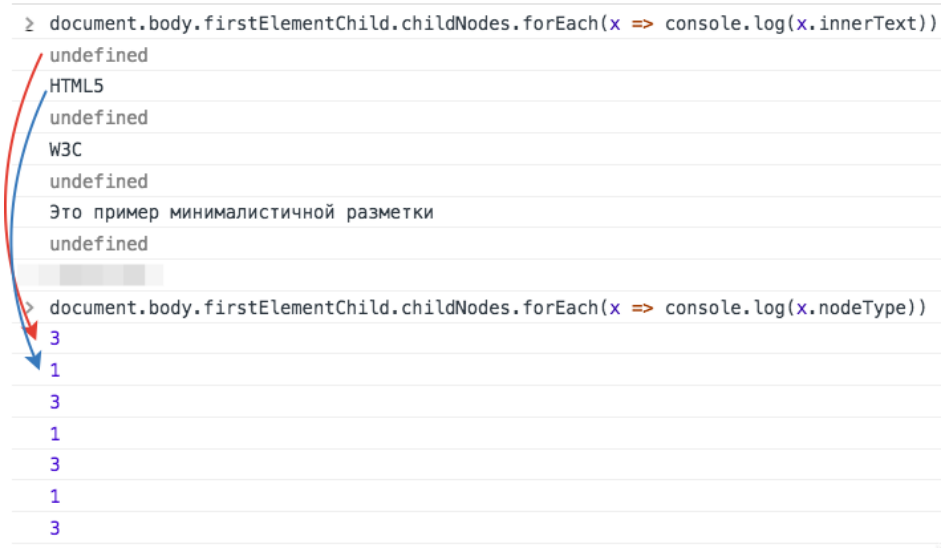
Первым потомком-узлом тела страницы является текст (хотя это просто перевод строки).

Предположим, мы хотим циклически вывести текст у элементов. Свойство childNodes возвращает структуру, упомянутую выше — NodeList.

Эта структура отличается от массивов (Array), которые встроены в JavaScript. Обе структуры имеют свойство length, хранящее количество элементов, и при работе с обеими используются квадратные скобки для доступа к отдельным элементам, но внутренне они действуют совершенно по-разному.

Определение NodeList в DOM (дается в спецификации HTML, а не ECMAScript 6) включает итератор по умолчанию, который действует, подобно итератору по умолчанию массива. Это означает, что NodeList можно использовать в цикле for..of.

В современной реализации у неё также есть метод forEach:



Мы видим, что у узлов, у которых свойство nodeType имеет значение 3, значение свойства innerText равно undefined. Это просто текстовые узлы, и у них перенос строки хранится как значение свойство nodeValue.

Окончательно, мы можем использовать это наблюдение чтобы вывести только текстовое содержание элементов:

```
document.body.firstElementChild.childNodes.forEach(x => x.nodeType === 1 ? console.log(x.innerText): '');
```

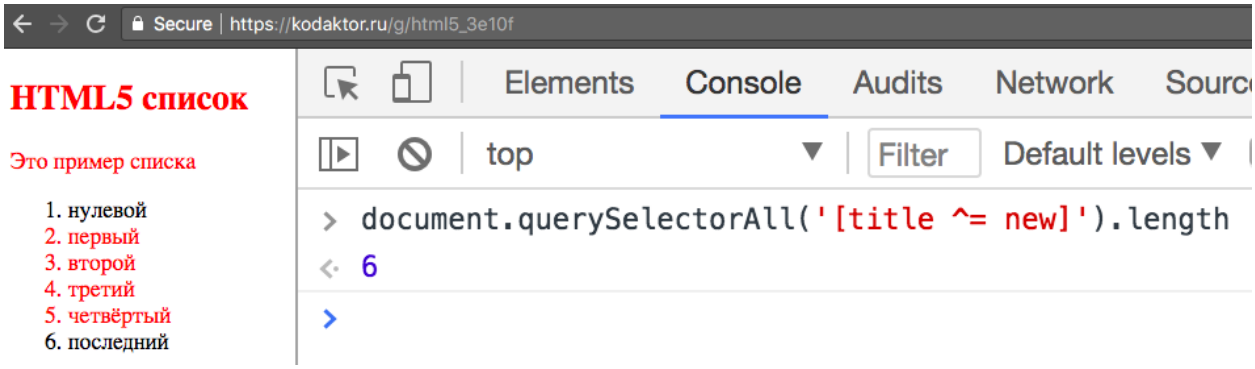
или

```
document.body.firstElementChild.childNodes.forEach(x => x.nodeType === 1 && console.log(x.innerText));
HTML5
W3C
Это пример минималистичной разметки
```

Понятие селектора объясняется в теме каскадных стилей, но напомним, что селекторы позволяют отбирать элементы веб-страницы и строить правила отображения.

Например на странице https://kodaktor.ru/html5_list есть элементы, у которых установлен атрибут title. Селектор [title=new] отбирает только те элементы, у которых значение атрибута title в точности равно слову new. Таких элементов пять. Это заголовок и четыре элемента списка.

В то же время если написать в селекторе [title ^= new] то к этим пяти добавится и шестой (абзац), потому что у него значение атрибута title начинается с этого слова.



Метод querySelectorAll существует у узлов типа Element и возвращает коллекцию типа NodeList. У неё есть свойство length.

Кроме того, мы можем использовать метод forEach, предоставив ему в качестве аргумента функцию. В эту функцию передаётся последовательно каждый элемент из коллекции. Поэтому мы можем, в частности, изменить стиль каждого элемента из коллекции: