

Введение в DOM

Урок 6





Кадочников Алексей

Frontend-разработчик

- ☀ Веб-разработчик со стажем более 9 лет
- ☀ Преподаватель GeekBrains с 2015 года
- ☀ Автор курсов по Frontend на портале Geekbrains
- ☀ Работал в таких компаниях, как VK и Wizard-C



План курса

1

Знакомство с javascript

2

Основы javascript

3

Функции в JavaScript

4

Циклы и массивы

5

Работа с объектами в JavaScript

6

Введение в DOM

7

Работа с DOM

8

Основы событий в JavaScript

9

Работа с событиями в JavaScript

10

Основы шаблонизации, работа с JSON

11








Работа с медиа файлами

12

Основы API, итоги курса



Что будет на уроке сегодня

-  Введение в DOM
-  Основы управления структурой DOM
-  Создание и добавление новых узлов
-  Клонирование узлов
-  Удаление узлов
-  Замена узла
-  Работа с свойствами и методами



Вводная информация

Основная область применения — использование JavaScript в браузерной среде.

Современные браузеры — это комплексные программные продукты, по сложности и функциональности сопоставимые с операционными системами. И действительно, если посмотрим на список существующих API браузера, найдём практически всё, что «умеют» приложения на наших с вами компьютерах и даже больше.

API браузера встроены в веб-браузер и используют данные браузера и компьютерной среды для осуществления более сложных действий с этими данными. Они не часть языка, API браузера строятся на основе встроенных функций JavaScript для увеличения возможностей разработчиков при написании кода.

Начнём с самого базового API, отвечающее за программное (объектное) представление веб-страницы (HTML-документа), а именно: DOM, или Document Object Model (объектная модель документа).



Введение в DOM

Документ, загруженный в каждую вкладку браузера, представлен объектной моделью документа (DOM). Это представление «древовидной структуры», созданное браузером.

```
1 <!DOCTYPE html>
2 <html>
3
4 <head>
5   <meta charset="utf-8">
6   <title>Простая страница</title>
7 </head>
8
9 <body>
10   <section>
11     <p>Пример текстовой ноды со <a id="myLink"
12 href="https://example.com">ссылкой</a></p>
13     
14   </section>
15 </body>
16 </html>
```



Введение в DOM

Он позволяет легко получить доступ к структуре HTML путём использования языков программирования.

```
DOCTYPE: html
HTML
  HEAD
    #text:
    META charset="utf-8"
    #text:
    TITLE
      #text: Простая страница
    #text:
  #text:
  BODY
    #text:
    SECTION
      #text:
      P
        #text: Пример текстовой ноды со
        A id="myLink" href="https://example.com"
          #text: ссылкой
      #text:
      IMG src="image.png" alt="Image example"
      #text:
    #text:
```



Введение в DOM

Мы видим, что каждый HTML-элемент и текст в документе имеют собственную запись в дереве — каждый из них называется узлом (node). Для описания узла и его положения в дереве используются термины:

- **Element node** — элемент, как он существует в DOM.
- **Root node** — верхний узел в дереве, который в случае HTML всегда представляет собой HTML-узел. Другие типы разметки, такие как SVG и пользовательский XML, имеют разные корневые элементы.
- **Child node (узел-ребёнок)** — узел, находящийся прямо внутри другого узла. Так, IMG в приведённом выше примере считается дочерним элементом SECTION.
- **Descendant node (узел-потомок)** — узел внутри дочернего элемента. IMG в приведённом выше примере считается дочерним элементом SECTION и потомком для родителя SECTION. IMG не ребёнок BODY, так как находится на двух уровнях ниже дерева в дереве, но он считается потомком BODY.
- **Ancestor node (узел-прародитель)** — один из родительских узлов родителя текущего узла. Любой узел, для которого текущий узел представляется потомком, будет его прародителем.



- **Parent node (узел-родитель)** — узел, в который входит текущий узел. Например, BODY — родительский узел SECTION в приведённом выше примере.
- **Sibling nodes (родственный узел)** — узлы, лежащие на одном уровне в дереве DOM. Например, IMG и P — братья и сёстры в приведённом выше примере.
- **Text node** — узел, содержащий текстовую строку.





Основы управления структурой DOM

В JavaScript есть множество способов выбора элемента и хранения указателя на него в переменной. `Document.querySelector()` — рекомендуемый современный подход, который считается удобным, потому что позволяет выбирать элементы, применяя селекторы CSS. Вышеупомянутый запрос `querySelector()` соответствует первому элементу `<a>`, который появляется в документе.



Есть более старые методы для захвата ссылок на элементы. Например:

- **Document.getElementById()** выбирает элемент с заданным значением атрибута id — передаётся функции как параметр.
- **Document.getElementsByTagName()** возвращает коллекцию HTMLCollection, содержащую все элементы на странице этого типа — передаётся функции как параметр. Например, <p>, <a> и т. д.
- **Document.getElementsByClassName()** возвращает коллекцию HTMLCollection дочерних элементов, соответствующих всем указанным именам классов — передаётся функции как параметр. Например, 'class-one class-two' и т. д.



Создание и добавление новых узлов

Возвращаясь к текущему примеру, начнём с получения ссылки на наш элемент `<section>`. Добавим следующий код внизу существующего скрипта:

```
1 const sectionElement = document.querySelector('section')
```

Теперь создадим новый абзац, используя `Document.createElement()`, и передадим ему текстовое содержимое, как и раньше:

```
1 const paragraphElement = document.createElement('p')  
2 paragraphElement.textContent = 'Новый текст параграфа'
```



Создание и добавление новых узлов

Создадим новый абзац в конце раздела, воспользовавшись `Node.appendChild()`:

```
1 sectionElement.appendChild(paragraphElement)
```

Добавим дополнительный текстовый узел в наш новый абзац.

Сначала создадим текстовый узел, используя `Document.createTextNode()`. Затем возьмём ссылку на абзац и добавим к нему дополнительный текстовый узел:

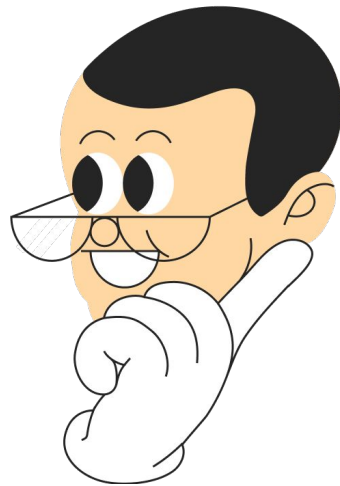
```
1 const paragraphElementText = document.createTextNode('Содержимое  
текстовой ноды')  
2 paragraphElement.appendChild(paragraphElementText)
```



Клонирование узлов

Когда узел уже добавлен в дерево, повторное добавление его другому родителю приводит к удалению из текущего. Для копирования узла применяется метод `Node.cloneNode(deep)`. Он возвращает дубликат узла, из которого этот метод вызван. Метод принимает единственный логический параметр `deep`, определяющий, надо ли клонировать дочерние элементы узла.

Важно! `cloneNode()` иногда приводит к дублированию идентификаторов элементов в документе.





Удаление узлов

Удалить узел из DOM можно двумя способами:

- через ссылку на родительский элемент, используя `parentNode.removeChild(child);`
- применив метод `Element.remove()`.



Удалим ранее созданный клон секции:

```
1 const sectionElementClone = document.querySelectorAll('section')[1]
2 sectionElementClone.parentNode.removeChild(sectionElementClone)
3 // sectionElementClone.remove()
```




Замена узла

DOM API содержит методы для замены одного узла другим или несколькими узлами. Как и в случае с удалением **доступны два разных способа**:

- через ссылку на родительский элемент, используя `parentNode.replaceChild(newChild, oldChild);`
- применив метод `Element.replaceWith(...nodes)`.

```
1 const divElement = document.createElement('div')
2 const paragraphElement = document.createElement('p')
3 divElement.appendChild(paragraphElement)
4
5 const spanElement = document.createElement('span')
6 const strongElement = document.createElement('strong')
7 paragraphElement.replaceWith(spanElement, strongElement)
8
9 console.log(divElement.outerHTML)
```











Работа с свойствами и методами

```
1 <div class="product">
2   
3   <h3>Название футболки</h3>
4   <p>Lorem ipsum dolor sit, amet consectetur adipisicing elit.
   Sequi, reiciendis!</p>
5   <button class="button">Купить</button>
6 </div>
7 <script>
8   const buttonEl = document.querySelector('.button');
9   const productImg = document.querySelector('.product__img');
10
11   productImg.onclick = function() {
12     productImg.src = 'img/photo2.jpg';
13   }
14
15   buttonEl.onclick = function() {
16     buttonEl.textContent = 'Товар добавлен в корзину'
17   }
18 </script>
```



Итоги урока

-  Узнали что такое DOM модель
-  Рассмотрели основы управления структурой DOM
-  Узнали как происходит создание и добавление новых узлов
-  Рассмотрели клонирование узлов
-  Разобрали удаление узлов
-  Разобрали замену узлов
-  Работа с свойствами и методами

Спасибо 
за внимание

