Язык JavaScript (DOM)

DOM

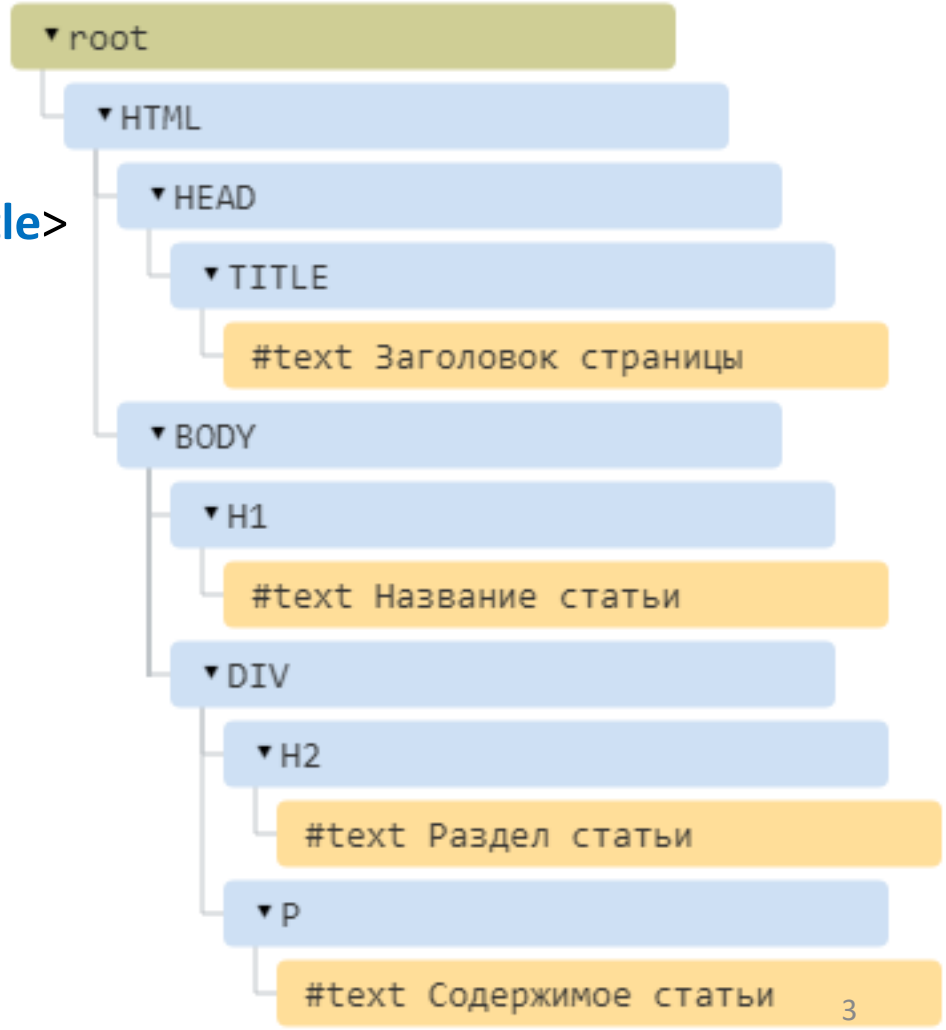
Объектная модель документа (DOM — Document Object Model) - это множество объектов, которые браузер создаёт в памяти компьютера (браузера) на основе кода HTML.

Веб-разработчик при создании динамических страниц работает не с HTML кодом данной страницы, а с объектами (DOM), которые браузер создаёт на основе этого кода.

DOM

В качестве примера рассмотрим следующий код HTML:

```
<html>
  <head>
    <title>Заголовок страницы</title>
  </head>
  <body>
    <h1>Название статьи</h1>
    <div>
      <h2>Раздел статьи</h2>
      <p>Содержимое статьи</p>
    </div>
  </body>
</html>
```



DOM

Дерево DOM - это множество узлов (объектов, элементов) связанных друг с другом. При этом каждый элемент HTML образует узел в этом дереве.

Связи между узлами (объектами, элементами) определяются на основе того что каждый элемент в HTML документе вложен в какой-то другой элемент. Элемент, который содержит другие элементы, по отношению к ним является **родителем**. У любого элемента в HTML коде есть свой родитель и притом только один. Если элемент содержит другие элементы, то для него они являются **дочерними (детьми, прямыми потомками)**. Один элемент может содержать сколько угодно много дочерних элементов. **Если все элементы HTML кода "выстроить" в зависимости от их отношения друг к другу, то у нас в результате получится дерево.** Строится это дерево браузером сверху вниз.

ТИПЫ УЗЛОВ (NODE) ДОКУМЕНТА

У каждого узла есть свойства:

❖ **nodeType** – возвращает тип узла, который определяется числом от 1 до 12;

Код типа (nodeType)	Тип узла	Описание
1	ELEMENT_NODE	Узел элемента
3	TEXT_NODE	Текстовый узел (#text)
8	COMMENT_NODE	Узел комментария (#comment)
9	DOCUMENT_NODE	Узел документа (#document)

ТИПЫ УЗЛОВ (NODE) ДОКУМЕНТА

У каждого узла есть свойства:

❖ **nodeName** — возвращает имя узла. Если узел является элементом, то свойство nodeName возвращает имя тега. Для других типов узлов данное свойство будет возвращать различные имена: "#text" (для текстовых узлов), "#comment" для комментариев, "#document" для документа и .т.д.

❖ **nodeValue** — строка, представляющая значение узла. Возвращает следующие значения в зависимости от типа узла:

- возвращает null для узла элемента и узла документа;
- возвращает содержимое для текстового узла;
- возвращает контент для узла комментария.

ПЕРЕМЕЩЕНИЕ ПО DOM (ТОЛЬКО ДЛЯ ЧТЕНИЯ)

```
document; //объект document, корень в DOM
document.documentElement; //доступ к <html>
document.body; //доступ к <body>
```

Свойства узлов (elem) возвращающие только теги	Свойства узлов (elem) возвращающие элементы (теги, текст, комментарии и т.п.)
elem.children – возвращает псевдомассив дочерних тегов	elem.childNodes - возвращает псевдомассив всех дочерних элементов
elem.firstElementChild – возвращает первый дочерний тег	elem.firstChild – возвращает первый дочерний элемент
elem.lastElementChild – возвращает последний дочерний тег	elem.lastChild – возвращает последний дочерний элемент
elem.parentElement – родительский узел	elem.parantNode – родительский узел
elem.previousElementSibling – предыдущий сосед тег	elem.previousSibling – предыдущий элемент тег
elem.nextElementSibling – следующий сосед тег	elem.nextSibling – следующий сосед элемент

ПОИСК ЭЛЕМЕНТОВ В DOM

- **document**.getElementById('id элементов') – поиск по id элемента (существует только в контексте document).
- **document**.getElementsByName("name элемента") – позволяет получить все элементы с одинаковыми. значениями, заданные атрибутом name (существует только в контексте document).
- **elem**.getElementsByTagName('имя тега') – возвращает их в виде списка все элементы с заданным тегом. Может искать внутри любого элемента.
- **elem**.getElementsByClassName('имя класса') – возвращает коллекцию элементов с указанным именем класса. Может искать внутри любого элемента.
- **document**.querySelector('CSS селектор') – возвращает первый элемент, соответствующий селектору CSS.
- **document**.querySelectorAll('CSS селектор') – возвращает все элементы, соответствующие селектору CSS.

DOM: АТРИБУТЫ ЭЛЕМЕНТОВ

У любого элемента (узла) есть свойство **attributes**, с помощью которого Вы можете получить коллекцию его атрибутов (узлов), в виде объекта NamedNodeMap. Доступ к атрибуту (узлу) в этой коллекции осуществляется по его индексу. У любого атрибута есть свойства **name** и **value**, с помощью которых Вы можете получить имя атрибута и его значение. Для работы с узлами используются следующие методы:

- getNamedItem('имя_атрибута'),
- setNamedItem(**атрибут_узел**),
- removeNamedItem('имя_атрибута').

DOM: АТРИБУТЫ ЭЛЕМЕНТОВ

```
<p id="description" class="text">I LOVE JAVASCRIPT</p>

// Пример №1 (выводим атрибуты и их значения):
let elementP = document.getElementById("description");
let attrP = elementP.attributes;
for (let i=0; i<attrP.length; i++) {
    console.log(attrP[i].name + " = " + attrP[i].value);
}
//Пример №2 (создаем новый атрибут):
let newAttr = document.createAttribute("style");
newAttr.value = "text-align: center;";
//добавляем атрибут к элементу
elementP.attributes.setNamedItem(newAttr);
```

DOM: АТРИБУТЫ ЭЛЕМЕНТОВ

У каждого элемента (узла) есть методы (**getAttribute**(), **setAttribute**(), r**emoveAttribute**(), **hasAttribute**()), которые позволяют работать с его атрибутами более просто.

Пример:

```
//получить элемент с id="myAnchor"
let myAnchor = document.getElementById("myAnchor");
//получить значение атрибута rel у элемента
let url = myAnchor.getAttribute("href");
//выведем значение атрибута в консоль браузера
console.log(url);
//удалить атрибут href у элемента
myAnchor.removeAttribute("href");
```

DOM: АТРИБУТЫ ЭЛЕМЕНТОВ

Ещё пример:

```
//получить все элементы a в HTML документе
let elements = document.getElementsByTagName("a");
//перебрать полученную коллекцию элементов в цикле
for (let i = 0; i < elements.length; i++) {
    //если указанный элемент имеет атрибут target, то
    if (elements[i].hasAttribute("target")) {
        //установить атрибуту targer значение _self
        elements[i].setAttribute("target","_self");
    }
}
```

DOM: АТРИБУТЫ ЭЛЕМЕНТОВ

Для работы с атрибутом **style** предназначено свойство **style**:

```
//получить значение свойства CSS (имяСвойстваCSS записанное в CamelCase)
element.style.имяСвойстваCSS
//установить свойству CSS (имяСвойстваCSS) значение value
element.style.имяСвойстваCSS = value
//удалить значение свойства CSS (имяСвойстваCSS)
element.style.имяСвойстваCSS = ""
//получить количество свойств CSS, содержащихся в объекте style
element.style.length
//получить CSS свойство, хранящееся в 1 элементе списка style
//отсчёт элементов в CSSStyleDeclaration начинается с 0
element.style[0]
//получить значение свойства color
element.style["color"]
```

DOM: АТРИБУТЫ ЭЛЕМЕНТОВ

Для получения или установления атрибута style целиком, т.е. в виде строки, используется свойство cssText:

```
//получить элемент, именующий id="intro"
let elementIntro = document.getElementById("intro");
/*установить значение "font-size:40px;color:blue;"
атрибуту style*/
elementIntro.style.cssText = "font-size:40px; color:blue;";
```

DOM: АТРИБУТЫ ЭЛЕМЕНТОВ

Для того чтобы управлять отдельными классами элемента необходимо использовать свойство classList.

Свойства объекта classList:

- length - количество классов у элемента.

Методы объекта classList:

- add(class1,class2,...) - добавляет один или несколько к элементу классов. Если указанный класс уже есть у элемента, то он не будет добавлен.
- contains(class) - возвращает true или false в зависимости от того имеет ли элемент указанный класс.

DOM: АТРИБУТЫ ЭЛЕМЕНТОВ

Методы объекта classList:

- item(index) - возвращает имя класса по его индексу из коллекции.
- remove(class1,class2,...) - удаляет один или несколько указанных у элемента классов.
- toggle(class,[**true**|**false**]) - переключает указанное имя класса у элемента. Метод toggle имеет 2 параметра: class (обязательный) - указывается имя класса, который надо переключить; true|false (необязательный) - логическое значение, которое принудительно заставляет включить или выключить указанный класс у элемента.

DOM: АТРИБУТЫ ЭЛЕМЕНТОВ

```
/*Пример №1: добавить несколько классов к элементу с id="myID“ */
let myID = document.getElementById("myID");
myID.classList.add("col-xs-6","col-sm-4");

/*Пример №2: определить имеет ли элемент класс hidden и если имеет, то удалить его */
if (myID.classList.contains("hidden") {
    myID.classList.remove("hidden");
}
```

DOM: АТРИБУТЫ ЭЛЕМЕНТОВ

Свойства элемента, предназначенные для доступа к его атрибутам: className, id, title, lang, tabIndex, dir.

Существуют свойства для управления атрибутами, которые существуют только у определённых видов элементов. Например, у элемента (узла) **img**, есть ещё и другие специфичные для этого объекта свойства: src (источник картинки), width (ширина картинки), height (высота картинки), alt (текст, если картинка не отображается) и др.

DOM: ДОБАВЛЕНИЕ И УДАЛЕНИЕ УЗЛОВ

Добавление нового узла к дереву обычно осуществляется в 2 этапа:

1. Создать необходимый узел, используя один из следующих методов:

- createElement('имя_тега') - создаёт элемент (узел) с указанным именем (тегом).
- createTextNode('текст') - создаёт текстовый узел с указанным текстом.

2. Указать место в дереве, куда необходимо вставить узел. Для этого необходимо воспользоваться одним из следующих методов:

- appendChild(**node**) - добавляет узел как последний дочерний узел элемента, для которого вызывается данный метод.
- insertBefore(**node**, [**existingNode**]) - вставляет узел как дочерний узел элемента, для которого вызывается данный метод. existingNode (не обязательный параметр) - это дочерний узел элемента перед которым, необходимо вставить узел.

DOM: ДОБАВЛЕНИЕ И УДАЛЕНИЕ УЗЛОВ

Пример:

```
//создаём элемент (узел) li
let elementLI = document.createElement("li");
//создаём текстовый узел
let textSmart = document.createTextNode("Смартфон");
/*добавляем созданный текстовый узел как последний
дочерний элемент к только что созданному элементу li */
elementLI.appendChild(textSmart);
/*получаем элемент, к которому будет добавлен
созданный узел li как дочерний*/
let elementUL = document.getElementById("list");
/*добавляем созданный элемент li как последний
дочерний элемент к ul с id="list" */
elementUL.appendChild(elementLI);
```

DOM: ДОБАВЛЕНИЕ И УДАЛЕНИЕ УЗЛОВ

Работа с существующими узлами методами appendChild() и insertBefore() также осуществляется в 2 этапа:

1. Получить существующий узел в дереве.
2. Указать место, куда необходимо вставить узел, с помощью метода appendChild() или insertBefore(). !!!При этом узел будет удалён из предыдущего места.

DOM: ДОБАВЛЕНИЕ И УДАЛЕНИЕ УЗЛОВ

Удаление узла из дерева осуществляется в 2 этапа:

1. Получить (найти) этот узел в дереве
2. Вызвать у родительского узла метод removeChild(**node**), которому в качестве параметра необходимо передать узел, который мы хотим у него удалить.

Метод removeChild() возвращает в качестве значения удалённый узел или null, если узел, который мы хотели удалить, не существовал.

DOM: ДОБАВЛЕНИЕ И УДАЛЕНИЕ УЗЛОВ

Пример, удалить все дочерние узлы у элемента, имеющего id="myQuestion":

```
/*получить элемент, у которого мы хотим удалить все его дочерние узлы*/
let element = document.getElementById("myQuestion");
//пока есть первый элемент
while (element.firstElement) {
    //удалить его
    element.removeChild(element.firstChild);
}
```

DOM: МЕТОДЫ ДЛЯ КЛОНИРОВАНИЯ И ЗАМЕНЫ УЗЛОВ

Метод **replaceChild**() предназначен для замены одного дочернего узла другим. В качестве результата данный метод возвращает узел, который мы заменили новым узлом.

```
//создаём элемент (узел) li
let elementLI = document.createElement("li");
//создаём текстовый узел
let textSmart = document.createTextNode("Смартфон");
/*добавляем созданный текстовый узел как последний
дочерний элемент к только что созданному элементу li */
elementLI.appendChild(textSmart);
//получить элемент ul с id="myTech"
let myTech = document.getElementById("myTech");
/*заменим первый дочерний узел (элемент li с индексом 0)
элемента ul на только что созданный узел */
myTech.replaceChild(elementLI, myTech.childNodes[0]);
```

DOM: МЕТОДЫ ДЛЯ КЛОНИРОВАНИЯ И ЗАМЕНЫ УЗЛОВ

Метод **cloneNode**(deep) создаёт копию узла, который он возвращает в качестве результата. Данный метод имеет один необязательный параметр (deep), который может принимать одно из следующих значений:

true - клонирует узел, его атрибуты и всех его потомков.

false (по умолчанию) - клонирует только узел и его атрибуты.

```
//получить элемент, имеющий id="myTech"
let myTech = document.getElementById("myTech");
//получить узел, который мы хотим клонировать
let myPhone = myTech.firstChild;
//клонировать узел
let myClone = myPhone.cloneNode(true);
//вставить узел в конец списка
myTech.appendChild(myClone);
```

DOM: СВОЙСТВА TEXTCONTENT, INNERHTML И ДР.

Свойство textContent позволяет получить текстовый контент указанного узла и всех его потомков. Данное значение можно представить как конкатенацию (сложение) все текстовых узлов, которые являются потомками узла, для которого вызывается данное свойство.
Кроме этого, данное свойство позволяет также установить элементу текстовый контент. При этом все дочерние узлы будут удалены и заменены единственным текстовым узлом, содержащий этот контент.

```
//Получить текстовый контент узла (node):
node.textContent;
//Установить узлу текстовый контент "Text":
node.textContent = "Text";
```

DOM: СВОЙСТВА TEXTCONTENT, INNERHTML И ДР.

node.innerText - это свойство, позволяющее задавать или получать текстовое содержимое элемента и его потомков.

В то время как **textContent** получает содержимое всех элементов, включая такие элементы как <**script**> и <**style**>, то свойство **innerText** этого не делает, а получает только значения тех элементов, которые отображает браузер пользователю.

Свойство **innerText** учитывает стили элемента и не будет возвращать текст скрытых элементов, тогда как свойство **textContent** это делает. Так как **innerText** следит за актуальным стилем CSS элемента, то это может сказаться на производительности, избегайте его использования, когда это возможно.

DOM: СВОЙСТВА TEXTCONTENT, INNERHTML И ДР.

Кроме свойства innerText также существует свойство outerText, которое возвращает текст аналогично свойству innerText. А вот при установлении значения свойству outerText для элемента, данное свойство заменяет не только содержимое расположенное между открывающим и закрывающим тегом элемента, но и сам элемент.

DOM: СВОЙСТВА TEXTCONTENT, INNERHTML И ДР.

Свойство innerHTML устанавливает или возвращает HTML контент в виде строки, расположенный между открывающим и закрывающим тегом элемента.
Пример:

```
let myP = document.getElementById("myP");
//получить HTML содержимое элемента, имеющего id="myP"
myP.innerHTML;
/*изменить HTML содержимое элемента, имеющего id="myP“ */
myP.innerHTML = "<em>Что-то новое</em>";
```

Свойство outerHTML устанавливает или возвращает HTML контент, представляющий сам элемент и его дочерние элементы.

ЧТО ПОЧИТАТЬ ДОПОЛНИТЕЛЬНО

https://learn.javascript.ru/dom-nodes - DOM-дерево
https://learn.javascript.ru/dom-navigation - Навигация по DOM-элементам
https://learn.javascript.ru/searching-elements-dom#matches - Поиск: getElement*, querySelector*
https://learn.javascript.ru/basic-dom-node-properties#textcontent-prosto-tekst - Свойства узлов: тип, тег и содержимое
https://learn.javascript.ru/dom-attributes-and-properties - Атрибуты и свойства
https://learn.javascript.ru/modifying-document - Изменение документа
https://learn.javascript.ru/styles-and-classes - Стили и классы