

Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет информатики и радиоэлектроники»
Филиал
«Минский радиотехнический колледж»

Учебная дисциплина «Программные средства создания Internet-приложений»

Инструкция
по выполнению лабораторной работы
«Использование объектов HTML и объектной модели документа»

Минск
2021

Лабораторная работа № 23

Тема работы: Использование объектов HTML и объектной модели документа

1. Цель работы

Формирование умений использования методов доступа к элементам HTML-документа средствами DOM и иерархической структуры DOM для организации доступа к HTML-элементам.

2. Задание

Выполнить задания в соответствии с порядком выполнения лабораторной работы.

3. Оснащение работы

ПК, редактор исходного кода, браузер.

4. Основные теоретические сведения

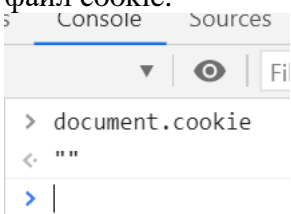
Объектная Модель Документа (DOM) – это программный интерфейс (API) для HTML и XML документов. DOM предоставляет структурированное представление документа и определяет то, как эта структура может быть доступна из программ, которые могут изменять содержимое, стиль и структуру документа. Представление DOM состоит из структурированной группы узлов и объектов, которые имеют свойства и методы. По существу, DOM соединяет веб-страницу с языками описания сценариев либо языками программирования.

Объект **document** представляет собой HTML документ, загруженный в окно (вкладку) браузера. С помощью свойств и методов данного объекта можно получить доступ к содержимому HTML-документа, а также изменить его содержимое, структуру и оформление.

Свойства и методы объекта document, доступные сценарию JavaScript, приведены в таблице 23.1.

Таблица 23.1 – Свойства и методы объекта document

Свойство/метод	Назначение
<u>document.doctype</u>	Возвращает Document Type Definition (DTD) текущего документа <!DOCTYPE html>
document.documentElement	Возвращает Element, который является первым дочерним элементом документа. Для HTML документов это HTML элемент <pre>> let a=document.documentElement; <> undefined > a; <> <html> <head></head> ><body>...</body> </html> ></pre>
document.documentURI	Возвращает URL документа "https://mrk-bsuir.by/ru"
<u>parentNode.childElementCount</u> <u>t</u>	Возвращает количество дочерних по отношению к parentNode элементов document.body.childElementCount; //2

<pre> 1 <!DOCTYPE html> 2 <html> 3 <head> 4 5 </head> 6 <body> 7 <h1> Заголовок</h1> 8 <p> Просто абзац </p> 9 обычный текст 10 11 </body> 12 13 </html> 14 </pre>	
<p><u>parentNode.children</u></p>	<p>Возвращает массив, содержащий все дочерние по отношению к parentNode элементы (за исключением узлов, не являющихся элементами)</p> <p>document.body.children // [h1, p]</p>
<p><u>parentNode.firstChild</u></p>	<p>Возвращает первый дочерний по отношению к parentNode элемент, или null, если таковых не имеется</p> <p>document.body.firstChild //<h1> Заголовок</h1> document.head.firstChild //null</p>
<p><u>parentNode.lastElementChild</u></p>	<p>Возвращает первый дочерний по отношению к parentNode элемент, или null, если таковых не имеется</p> <p>document.body.lastElementChild // <p> Просто абзац </p></p>
<p><u>document.activeElement</u></p>	<p>Возвращает элемент, который в текущий момент находится в фокусе</p> <p>document.activeElement</p>
<p><u>document.anchors</u></p>	<p>Возвращает массив локальных меток, размещенных в документе. Эти метки применяются для организации ссылок внутри документа. Если меток в пределах документа нет, то возвращается пустой массив</p> <p>document.anchors //HTMLCollection [], length: 0</p>
<p><u>document.body</u></p>	<p>Возвращает элемент <body> текущего документа</p> <pre> > document.body < ▾ <body> <h1> Заголовок</h1> ▸ <div>...</div>
 <input> </body> </pre>
<p><u>document.cookie</u></p>	<p>Возвращает список файлов cookie, разделенных точкой с запятой, для этого документа или устанавливает один файл cookie.</p> 
<p><u>document.defaultView</u></p>	<p>Возвращает объект window, который связан с document текущей страницы или null если document не доступен. Это свойство доступно только для чтения.</p>

<code>document.designMode</code>	<p>Переключает режим редактирования для всего документа. Допустимые значения: "on" и "off". <code>document.designMode = "on" "off";</code></p>
<code>document.dir</code>	<p>Свойство является строкой DOMString показывает направление текста на странице (слева направо или справа налево). 'ltr' - слева направо 'rtl' - справа налево</p>
<u><code>document.domain</code></u>	<p>Свойство domain у Document интерфейса получает/устанавливает доменную часть источника происхождения (origin) текущего документа. <code>document.domain // "mrk-bsuir.by"</code></p>
<code>document.embeds</code>	<p>Возвращает массив встроенных элементов <object> в текущем документе, только для чтения</p>
<u><code>document.forms</code></u>	<p>Возвращает массив встроенных элементов < form > в текущем документе</p> <pre> > document.forms < ▼ HTMLCollection [] ⓘ length: 0 ▶ __proto__: HTMLCollection > </pre>
<code>document.head</code>	<p>Возвращает элемент < head > в текущем документе</p>
<code>document.links</code> <pre> <body> <h1> Заголовок</h1> <div> <p> Просто абзац </p> обычный текст </div>
 <input /> Ссылка 1 Ссылка 2 Ссылка 3 </body> </pre>	<p>Возвращает массив всех гиперссылок в документе</p> <pre> > document.links; < ▼ HTMLCollection(3) [a, a, a] ⓘ 0: a 1: a 2: a length: 3 ▶ __proto__: HTMLCollection > </pre>
<code>document.location</code>	<p>Возвращает URI (Унифицированный Идентификатор Ресурса, последовательность символов, идентифицирующая физический или абстрактный ресурс, который не обязательно должен быть доступен через сеть Интернет, причем, тип ресурса, к которому будет получен доступ, определяется контекстом и/или механизмом) текущего документа</p> <pre> > document.location < Location {href: "https://mrk-bsuir.by/ru", ancestorOrigin: "https://mrk-bsuir.by", protocol: "https:", host: "mrk-bsuir.by", ...} > </pre>
<u><code>document.readyState</code></u>	<p>Возвращает статус загрузки документа loading - страница все еще загружается; interactive - страница уже загружена и DOM дерево построено, но дополнительные ресурсы, такие как изображения и iframe, все еще загружаются;</p>

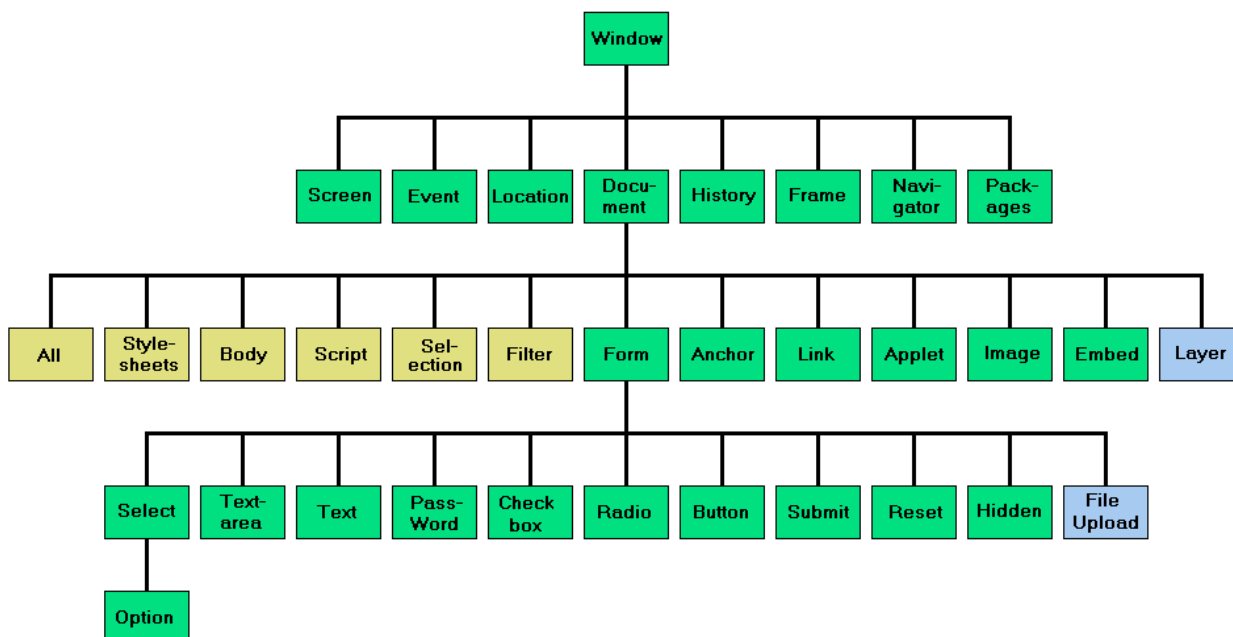
	<p>complete - страница и все дополнительные ресурсы уже загружены. Это состояние указывает, что событие load уже вызвано</p> <pre>> document.readyState</pre> <pre>< "complete"</pre>
<code>document.referrer</code>	<p>Возвращает URI страницы, с которой был совершен переход на текущую. Вернет пустую строку, если пользователь попал на страницу напрямую (не через ссылку, а, например, через закладку). Т.к. свойство возвращает строку, это не дает доступ к DOM ссылающейся страницы</p> <pre>> document.referrer</pre> <pre>< ""</pre>
<code>document.scripts</code>	<p>Возвращает массив всех элементов <code><script></code> в документе</p> <pre>document.scripts // HTMLCollection []</pre>
<code>document.URL</code>	<p>Возвращает строку, содержащую URL-адрес текущего документа</p> <pre>> document.URL</pre> <pre>< "https://mrk-bsuir.by/ru"</pre>
<code>document.createAttribute(String name)</code>	<p>Создает новый атрибут узла и возвращает его</p> <pre>let atr = document.getElementById("atr"); let a = document.createAttribute("my_atr"); a.value = "my_val"; atr.setAttributeNode(a); console.log(atr.getAttribute("my_atr")); // my_val</pre>
<code>document.createComment(String comment)</code> <pre><body> <h1> Заголовок</h1> <div id="atr"> <p> Просто абзац </p> обычный текст </div>
 <input /> Ссылка 1 Ссылка 2 Ссылка 3 </body></pre>	<p>Создаёт новый комментарий и возвращает его</p> <pre>let com=document.createComment("New comment"); > document.body.appendChild(com);</pre> <pre>< <!--New comment--> ▼ <body> <h1> Заголовок</h1> .. ▶ <div id="atr">...</div> == \$0
 <input> Ссылка 1 Ссылка 2 Ссылка 3 <!--New comment--> </body></pre>
<code>document.createTextNode(String text)</code>	<p>Создает и возвращает новый текстовый узел</p> <pre>let newtext = document.createTextNode("динамически встроенный обычный текст"); document.body.appendChild(newtext); document.body.appendChild(newtext);</pre> <pre>< "динамически встроенный обычный текст" <!--New comment--> "динамически встроенный обычный текст" </body></pre>
<code>document.getElementsByClassName(String className)</code>	<p>Возвращает массивоподобный (итерируемый) объект всех дочерних элементов, соответствующих всем из указанных имен классов. В случае вызова по отношению к объекту 'document', поиск происходит по всему</p>

	<p>документу, включая корневой элемент. Вызывать <code>getElementsByClassName()</code> можно также применительно к любому элементу: возвращены будут лишь те элементы, которые являются потомками указанного корневого элемента и имеют при этом указанные классы</p> <pre>document.getElementsByClassName('test'); document.getElementById('main').getElementsByClassName('test');</pre>
<p><code>document.getElementsByTagName(String tagName)</code></p> <pre> 6 <body> 7 <h1> Заголовок</h1> 8 <div id="atr"> 9 <p> Просто абзац </p> 10 обычный текст 11 </div> 12
 13 <input /> 14 Ссылка 1 15 Ссылка 2 16 Ссылка 3 17 </body></pre>	<p>Возвращает массив элементов с указанным именем тега</p> <pre>document.getElementsByTagName('a');</pre> <p>► <code>HTMLCollection(3) [a, a, a]</code></p>
<p><code>document.getElementById(String id)</code></p>	<p>Возвращает ссылку на элемент по его идентификатору (ID)</p> <pre>> let elem = document.getElementById("atr"); < undefined > elem; < ► <div id="atr">...</div></pre>
<p><code>document.querySelector(String selector)</code></p>	<p>Возвращает первый элемент документа, который соответствует указанному селектору или группе селекторов. Если совпадений не найдено, возвращает значение <code>null</code></p> <pre>> document.querySelector('a');</pre> <p>◀ <code>Ссылка 1 </code></p>
<p><code>document.querySelectorAll(String selector)</code></p> <pre> 14 Ссылка 1 15 Ссылка 2 16 Ссылка 3 17 </body> Ссылка 1 Ссылка 2 Ссылка 3 </body></pre>	<p>Возвращает статический (не динамический) массив, содержащий все найденные элементы документа, которые соответствуют указанному селектору</p> <pre>> document.querySelectorAll('a');</pre> <p>◀ ► <code>NodeList(3) [a, a, a]</code></p> <pre>> document.querySelectorAll('.one');</pre> <p>◀ ► <code>NodeList [a.one]</code></p>
<p><code>document.close()</code></p>	<p>Метод завершает запись в документ, открытый с помощью <code>document.open()</code></p> <pre>document.open(); document.write("<p>The one and only content.</p>"); document.close();</pre>
<p><code>document.getElementsByName(String name)</code></p> <pre> Ссылка 1 Ссылка 2 Ссылка 3 </body></pre>	<p>Возвращает массив элементов с заданным именем</p> <pre>> document.getElementsByName('nam');</pre> <p>◀ ► <code>NodeList [a]</code></p> <pre>> console.log(document.getElementsByName('nam')[0].tagName);</pre> <p>A</p>

<code>document.getSelection()</code> Заголовок <div>Просто абзац</div> обычный текст	Возвращает объект Selection, в котором содержатся данные о тексте, выделенном в документе на данный момент <pre>> document.getSelection() < Selection {anchorNode: text, anchorOffset: 1, focusNode: text, focusOffset: 13, isCollapsed: false, ...}</pre>
<code>document.hasFocus()</code>	Метод возвращает значение Boolean, указывающее имеет ли документ или любой элемент внутри документа фокус <pre>> document.hasFocus(document.getElementsByTagName("input")) < false > document.hasFocus() < false</pre>
<code>document.write(String text)</code>	Пишет строку в поток документа, открытый с помощью document.open() <code>document.write("<h1>Новый заголовок!</h1>");</code>
<code>document.writeln(String text)</code>	Выводит в документ строку со знаком перевода каретки в конце. <code>document.writeln("<p>введите пароль:</p>");</code>

При загрузке HTML-документа в браузер почти для каждого его элемента иницируется объект, доступный из сценария. Эти объекты группируются в коллекции, рисунок 16.1:

- **images**: содержит коллекцию всех объектов изображений (элементов img);
- **links**: содержит коллекцию ссылок – элементов <a> и <area>, у которых определен атрибут href;
- **anchors**: предоставляет доступ к коллекции элементов <a>, у которых определен атрибут name (якорям);
- **forms**: содержит коллекцию всех форм на веб-странице;
- **embeds**: содержит список всех встроенных элементов на странице (тег <embed>);
- **scripts**: содержит список всех скриптов на странице (тег <script>).



Кроме того, все объекты принадлежат еще и коллекции **all** – коллекция всех тегов и элементов в основной части документов. Коллекция *all* не входит в стандарт DOM W3C, но поддерживается всеми наиболее популярными браузерами.

Доступ к коллекции можно осуществить двумя способами:

```
document.коллекция [индекс]
document.коллекция ["значение_id"]
```

где:

- **индекс** – номер элемента, соответствующий порядку его упоминания в HTML-коде (нумерация начинается с нуля);

- **значение_id** – значение атрибута id в теге, задающем элемент.

Обращение к объекту по числовому индексу достаточно удобно, если число элементов в коллекции невелико и постоянно. В большинстве же практических случаев надежнее доступ к объектам по их идентификаторам (значениям атрибута id).

```
<img id='picture' src='test/flowers.jpg'>
```

Тогда доступ к значению, указывающему на источник изображения, можно получить с помощью любого из следующих выражений:

```
document.getElementById('picture');
document.all['picture'];
document.images['picture'];
document.images[0]; //
```

Для работы с изображениями в JavaScript используется объект **Image**, который является очередным свойством объекта Document.

Получим все изображения на странице:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
</head>
<body>
  
  
  
  <script>
let images = document.images;
// изменим первое изображение
images[0].src="pics/picture_4.jpg";
images[0].alt="Новая картинка";
// перебирем все изображения
for(let i=0; i<images.length;i++){

  document.write("<br/>" + images[i].src);
  document.write("<br/>" + images[i].alt);
}
document.write(document.images.length); // возвращает количество изображений
                                         на странице
  </script>
</body>
</html>
```

Конструктор **Image()** создает новый экземпляр HTMLImageElement:

```
Image([unsigned long width, unsigned long height])
```

```
let img = new Image(100, 200);
img.src = ' test/flowers.jpg ';
console.log(img);
```

Эквивалентно document.createElement('img'):

```
let a=document.createElement('img');
document.body.append(a);
```

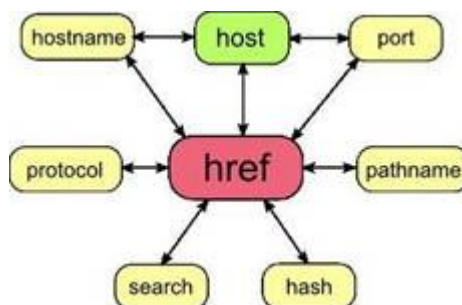


```
a.id="new";
document.getElementById('new');
document.getElementById('new').setAttribute('src',' test/flowers.jpg');
```

Обращение к объекту *Image* очень простое: сначала пишется объект *Document*, а затем его свойство с именем объекта *Image* (это имя задается с помощью атрибута "name"). В результате выполнения этого скрипта Вы увидите: "[object HTMLImageElement]".

```
<script>
document.write(document.img1)
document.img1.border = 5;
document.img1.width=200;
document.write("Ширина изображения - " + document.img1.width + "<br />");
document.write("Высота изображения - " + document.img1.height);
</script>
```

При работе с ссылками в JavaScript используется объект **Link**, который также является очередным свойством объекта *Document*. Объект **Link**, по сути, представляет собой HTML-ссылку.



Свойство объекта **Link** **hostname** содержит часть URL, которая отвечает за имя хоста:

```
<a href = 'http:// mrk-bsuir.by/mysql.html'>Ссылка</a>
<script >
document.write(document.links[0].hostname);    // "mrk-bsuir.by"
</script>
```

В данном примере мы создали сначала ссылку, а уже в скрипте получили объект *Link*, используя массив *links* в объекте *Document*. Данный массив содержит все ссылки на странице. Поэтому мы легко можем всегда к нему обратиться. А, получив объект *Link* из массива, прочитали его свойство, значение которого оказалось "mrk-bsuir.by". Также это свойство можно изменить:

```
document.links[0].hostname = "google.ru";
```

после выполнения скрипта адрес, на который ведёт данная ссылка, поменяется.

Свойство объекта **Link** **pathname** показывает путь в ссылке:

```
document.write(document.links[0].pathname); //"/mysql.html"
```

Свойство **href** объединяет свойства **hostname** и **pathname** и содержит путь, указанный в атрибуте **href**:

```
document.write(document.links[0].href); // 'http:// mrk-bsuir.by/mysql.html'
```

Как и все свойства, свойство **href** у объекта *Link* также доступно на запись.

Имеются и другие свойства, но их использование крайне редкое. Как правило, хватает и трёх рассмотренных. На картинке указаны ещё несколько других свойств:

- *port* – номер порта URL;

- *hash*- строка, следующая в URL за символом #, с помощью которого указывается, к какому якорю внутренней ссылки следует переместиться при загрузке документа;
- *host* – часть URL «хост:порт», значение порта содержится лишь тогда, когда оно явно было указано в URL-адресе;
- *protocol* – начальная часть, определяющая протокол, за которой следует двоеточие, например «http:»;
- *search* – строка запроса или данные URL, следующие после имени файла (включая разделительный символ ?).

Рассмотрим получение всех ссылок на странице:

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
</head>
<body>
  <a href="article1.html">Статья 1</a>
  <a href="article2.html">Статья 2</a>
  <a href="article3.html">Статья 3</a>
<script>
var links = document.links;
links[0].innerText="Замена";
links[0].href="article2.html";

for(var i=0; i<links.length;i++){

  document.write("<br/>" + links[i].innerText);
  document.write("<br/>" + links[i].href);
}
</script>
</body>
</html>
```

Опять же, так как у ссылки определен атрибут href, то при переборе ссылок можно получить его значение.

Некоторые типы DOM-элементов предоставляют для удобства дополнительные свойства, специфичные для их типа.

Элемент <table> в дополнение к стандартным свойствам поддерживает следующие:

table.rows – возвращает коллекцию всех строк <tr> таблицы;

```
document.getElementsByTagName('table')[0].rows[0].style.color='green';
```

table.tBodies – возвращает коллекцию элементов таблицы <tbody> (по спецификации их может быть больше одного);

```
document.getElementsByTagName('table')[0].tBodies;
```

table.caption/tHead/tFoot – возвращает ссылки на элементы таблицы <caption>, <thead>, <tfoot>;

```
document.getElementsByTagName('table')[0].caption; //null, если заголовок отсутствует
```

<thead>, <tfoot>, <tbody> предоставляют свойство rows:

tbody.rows – коллекция строк <tr> секции.

<tr>:

tr.cells – коллекция <td> и <th> ячеек, находящихся внутри строки <tr>;

tr.sectionRowIndex – возвращает номер строки <tr> в текущей секции <thead>/<tbody>/<tfoot>;

tr.rowIndex – возвращает номер строки <tr> в таблице (включая все строки таблицы).

<td> and <th>:

td.cellIndex – номер ячейки в строке <tr>;

td.colSpan – задает или возвращает значение атрибута colspan;
td.headers – задает или возвращает значение атрибута headers;
td.rowSpan – задает или возвращает значение атрибута rowspan.

```
document.getElementsByTagName('table')[0].rows[1].cells[0].colSpan='6';
```

```
<table id="table">
  <tr>
    <td>один</td><td>два</td>
  </tr>
  <tr>
    <td>три</td><td>четыре</td>
  </tr>
</table>

<script>
  // выводит содержимое первой строки, второй ячейки
  alert( table.rows[0].cells[1].innerHTML ) // "два"
</script>
```

Методы объекта Table

Метод	Описание
<u>createCaption()</u>	Создает пустой элемент <Caption> и добавляет его в таблицу
<u>createTFoot()</u>	Создает пустой элемент <tfoot> и добавляет его в таблицу
<u>createTHead()</u>	Создает пустой элемент <thead> и добавляет его в таблицу
<u>deleteCaption()</u>	Удаляет первый элемент <Caption> из таблицы
<u>deleteRow()</u>	Удаление строки (<TR>) из таблицы
<u>deleteTFoot()</u>	Удаляет элемент <tfoot> из таблицы
<u>deleteTHead()</u>	Удаление элемента <thead> из таблицы
<u>insertRow()</u>	Создает пустой элемент <TR> и добавляет его в таблицу

Методы объекта TableRow

Метод	Описание
<u>deleteCell()</u>	Удаление ячейки из текущей строки таблицы
<u>insertCell()</u>	Вставка ячейки в текущую строку таблицы

```
document.getElementsByTagName('table')[0].createCaption();
document.getElementsByTagName('table')[0].caption.textContent='Заголовок';
```

5. Порядок выполнения работы

1. Создайте html-документ, содержащий не менее 5 элементов (заголовки, абзацы, элементы форм и т.д.). Предусмотрите наличие двух пустых блоков div1 и div2. В первый блок необходимо вывести все дочерние узлы элемента body. Во второй блок вывести все элементы документа.

2. Добавьте на html -страницу нумерованный список, содержащий названия месяцев и кнопку, по нажатию на которую будет создан 13-й пункт списка, содержащий маркированный список с перечнем дней недели.
3. Напишите сценарий, который по нажатию на кнопку выводит содержимое всех заголовков в пределах элемента main.
4. Получить доступ к одному и тому же элементу тремя разными способами, в том числе с использованием методов, использующих иерархическую структуру DOM. Значения, которые возвращает каждый из использованных методов сохранить в переменные. Выполнить проверку: равны ли переменные (ссылаются ли они на один и тот же узел).
5. Средствами JavaScript добавьте на веб-страницу таблицу в соответствии с вариантом, таблица 23.2

Вариант	Задание																							
1	Заголовок 1		Заголовок 2																					
	Ячейка 3	Ячейка 4																						
Добавьте для таблицы заголовков																								
2	<table><tr><td>Name</td><td>Surname</td><td>Telephone</td></tr><tr><td>Jack</td><td>Sales</td><td>555-5555</td></tr><tr><td>John</td><td>Admin</td><td>555-5555</td></tr><tr><td>James</td><td>Sales</td><td>555-5555</td></tr><tr><td>Total</td><td>Total</td><td>Total</td></tr></table>			Name	Surname	Telephone	Jack	Sales	555-5555	John	Admin	555-5555	James	Sales	555-5555	Total	Total	Total						
	Name	Surname	Telephone																					
	Jack	Sales	555-5555																					
	John	Admin	555-5555																					
	James	Sales	555-5555																					
Total	Total	Total																						
3	<table><tr><td></td><td>2013</td><td>2014</td><td>2015</td></tr><tr><td>Нефть</td><td>43</td><td>51</td><td>79</td></tr><tr><td>Золото</td><td>29</td><td>34</td><td>48</td></tr><tr><td>Дерево</td><td>38</td><td>57</td><td>36</td></tr></table>				2013	2014	2015	Нефть	43	51	79	Золото	29	34	48	Дерево	38	57	36					
		2013	2014	2015																				
	Нефть	43	51	79																				
	Золото	29	34	48																				
Дерево	38	57	36																					
4	<table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>4</td><td>5</td><td>6</td></tr><tr><td>7</td><td>8</td><td>9</td></tr></table>			1	2	3	4	5	6	7	8	9												
	1	2	3																					
	4	5	6																					
7	8	9																						
5	<table><tr><td>Признаки</td><td>Сычи</td><td>Ушастые совы</td><td>Филин</td></tr><tr><td>Количество видов</td><td>3</td><td>6</td><td>16</td></tr><tr><td>Большие уши</td><td>✗</td><td>✓</td><td>✓</td></tr><tr><td>Латинское наименование</td><td>Athene</td><td>Asio</td><td>Bubo</td></tr><tr><td>Итого по размерам</td><td>Мелкие</td><td>Средние</td><td>Крупные</td></tr></table>				Признаки	Сычи	Ушастые совы	Филин	Количество видов	3	6	16	Большие уши	✗	✓	✓	Латинское наименование	Athene	Asio	Bubo	Итого по размерам	Мелкие	Средние	Крупные
	Признаки	Сычи	Ушастые совы	Филин																				
	Количество видов	3	6	16																				
	Большие уши	✗	✓	✓																				
	Латинское наименование	Athene	Asio	Bubo																				
Итого по размерам	Мелкие	Средние	Крупные																					

6*. Добавить для объекта Element метод в прототип, который добавляет указанный дочерний узел.

Фамилия, инициалы учащегося _____

Дата выполнения работы _____

Тема работы: _____

Цель работы: _____

Оснащение работы: _____

Результат выполнения работы: _____

7. Контрольные вопросы и задания

1. Что представляет собой объектная модель документа (DOM)?
2. Что такое DOM-дерево?.
3. Перечислите известные Вам узлы DOM-дерева.
4. Перечислите известные способы обращений к элементам HTML?
5. Перечислите коллекции объектов, доступные из сценария?
6. Для каких целей используется объект **Image**? Приведите пример.
7. Опишите основные свойства объекта **Link**.

8. Рекомендуемая литература

1. **JAVASCRIPT.RU** [Электронный ресурс] / Современный учебник JavaScript – 2007—2020 Илья Кантор. – Режим доступа: <https://learn.javascript.ru>. – Дата доступа: 04.03.2020.
2. **Никсон, Р.** Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / Р. Никсон. 4-е изд. – СПб.: Питер, 2018.
3. **Симпсон, К.** ES6 и не только / К. Симпсон. – СПб.: Питер, 2017.
4. **Хавербеке, М.** Выразительный JavaScript. Современное веб-программирование / М. Хавербеке – СПб.: Питер, 2019.