Лабораторная работа 2

Парсинг XML с использованием JavaScript

Задание

Создать HTML страницу с встроенным скриптом, который будет обрабатывать выданный вам XML файл. Вам необходимо вывести на экран все элементы третьего, последнего уровня вложенности одного из элементов второго уровня.

Дополнительное задание: в HTML файле создать поле ввода, в которое вводиться номер элемента второго уровня, для которого выведутся элементы третьего уровня.

Теоретические сведения

Основы XMLHttpRequest

Объект `XMLHttpRequest` (или, как его кратко называют, «XHR») дает возможность из JavaScript делать HTTP-запросы к серверу без перезагрузки страницы.

Несмотря на слово «XML» в названии, `XMLHttpRequest` может работать с любыми данными, а не только с XML.

Использовать его очень просто.

Пример использования

Как правило, `XMLHttpRequest` используют для загрузки данных.

Для начала посмотрим на пример использования, который загружает файл `phones.xml` из текущей директории и выдаёт его содержимое:

```
// 1. Создаём новый объект XMLHttpRequest
var xhr = new XMLHttpRequest();

// 2. Конфигурируем его: GET-запрос на URL 'phones.xml'
xhr.open('GET', 'phones.xml', false);

// 3. Отсылаем запрос
xhr.send();
```

```
// 4. Если код ответа сервера не 200, то это ошибка
if (xhr.status != 200) {
  // обработать ошибку
  alert( xhr.status + ': ' + xhr.statusText ); // пример вывода: 404: Not Found
} else {
  // вывести результат
  alert( xhr.responseXML ); // responseText -- текст ответа.
}
```

Настроить: open

Синтаксис:

```
xhr.open(method, URL, async, user, password)
```

Этот метод – как правило, вызывается первым после создания объекта `XMLHttpRequest`.

Задаёт основные параметры запроса:

- `method` – HTTP-метод. Как правило, используется GET либо POST, хотя доступны и более экзотические, вроде TRACE/DELETE/PUT и т.п.

- `URL` – адрес запроса. Можно использовать не только http/https, но и другие протоколы, например `ftp://` и `file://`.

- При этом есть ограничения безопасности, называемые «Same Origin Policy»: запрос со страницы можно отправлять только на тот же `протокол://домен:порт`, с которого она пришла. В следующих главах мы рассмотрим, как их можно обойти.

- `async` – если установлено в `false`, то запрос производится синхронно, если `true` – асинхронно.

«Синхронный запрос» означает, что после вызова `xhr.send()` и до ответа сервера главный поток будет «заморожен»: посетитель не сможет

взаимодействовать со страницей – прокручивать, нажимать на кнопки и т.п. После получения ответа выполнение продолжится со следующей строки.

«Асинхронный запрос» означает, что браузер отправит запрос, а далее результат нужно будет получить через обработчики событий, которые мы рассмотрим далее.

- `user`, `password` – логин и пароль для HTTP-авторизации, если нужны.

Вызов `open` не открывает соединение

> Заметим, что вызов `open`, в противоположность своему названию (`open` – англ. «открыть») не открывает соединение. Он лишь настраивает запрос, а коммуникация инициируется методом `send`.

Отослать данные: send

Синтаксис:

```
xhr.send([body])
```

Именно этод метод открывает соединение и отправляет запрос на сервер.

В `body` находится *тело* запроса. Не у всякого запроса есть тело, например у GET-запросов тела нет, а у POST – основные данные как раз передаются через `body`.

Отмена: abort

Вызов `xhr.abort()` прерывает выполнение запроса.

Ответ: status, statusText, responseText

Основные свойства, содержащие ответ сервера:

`status`

HTTP-код ответа: `200`, `404`, `403` и так далее. Может быть также равен `0`, если сервер не ответил или при запросе на другой домен.

`statusText`

Текстовое описание статуса от сервера: `OK`, `Not Found`, `Forbidden` и так далее.

`responseText`

Текст ответа сервера.

`responseXML`

Если сервер вернул XML, снабдив его правильным заголовком `Content-type: text/xml`, то браузер создаст из него XML-документ. По нему можно будет делать запросы `xhr.responseXml.querySelector("...")` и другие.

getElementsByTagName

Метод `elem.getElementsByTagName(tag)` ищет все элементы с заданным тегом `tag` внутри элемента `elem` и возвращает их в виде списка.

Регистр тега не имеет значения.

Например:

```
// получить все recipe-элементы
var elements = document.getElementsByTagName('recipe');
```

Например, найдём все элементы `input` внутри таблицы:

```
<table id="age-table">
  <tr>
    <td>Ваш возраст:</td>
    <td>
      <label>
        <input type="radio" name="age" value="young" checked>
младше 18
      </label>
      <label>
        <input type="radio" name="age" value="mature"> от 18 до
50
      </label>
      <label>
        <input type="radio" name="age" value="senior"> старше
60
      </label>
    </td>
  </tr>
```

```
</table>


<script>
  var tableElem = document.getElementById('age-table');
  var elements = tableElem.getElementsByTagName('input');

  for (var i = 0; i < elements.length; i++) {
    var input = elements[i];
    document.write(input.value + ', ');
  }
</script>
```

Можно получить всех потомков, передав звездочку `'*'` вместо тега:

```
// получить все элементы документа
document.getElementsByTagName('*');

// получить всех потомков элемента elem:
elem.getElementsByTagName('*');
```

CORS политика

CORS — это механизм безопасности, который позволяет веб-странице из одного домена обращаться к ресурсу с другим доменом (кросс-доменным запросом). Без таких функций, как CORS, веб-сайты ограничиваются доступом к ресурсам одного и того же происхождения через так называемую политику единого происхождения.

При выполнении запросов на локальном компьютере, несмотря на то что запрос к XML файлу не является кросс-доменным, браузер, тем не менее, закрывает доступ. Это связано с тем, что браузер при открытии локальных файлов в заголовке Access-Control-Allow-Origin посылает null. Для корректной работы необходимо запустить на локальном компьютере собственный веб-сервер, однако это ограничение можно обойти, открыв браузер Chrome в небезопасном режиме, позволяющем подобные операции.

Для выполнения этой лабораторной работы запустите браузер Chrome с параметром --allow-file-access-from-files

Для Windows:

```
chrome.exe --allow-file-access-from-files
```

Для Mac:

```
open /Applications/Google\ Chrome.app/ --args
--allow-file-access-from-files
```