Публикация на тему

API web-хранилища

JavaScript API web-хранилища - предоставляют набор методов и свойств для взаимодействия с данными на жестком диске пользователя.

Автор
Михалькевич Александр Викторович

Публикация

Наименование API web-хранилища

Автор А.В.Михалькевич

Специальность JavaScript API web-хранилища - предоставляют набор методов и свойств для взаимодействия с данными на жестком диске пользователя.,

Анотация

Anotation in English

Ключевые слова

Количество символов 17514

Содержание

Введение
1 Возможности web-хранилища
2 Создание и извлечение данных
3 Удаление данных
4 Сохранение чисел и дат
5 Слежение за областью HTML5-хранилища
Заключение
Список использованных источников
Приложения

Введение

Рассмотрены localStorage и sessionStorage - механизмы взаимодействия с данными на стороне клиента.

1 Возможности web-хранилища

API Web Storage (web хранилища) — это, по сути, следующая ступень развития файлов cookie. Этот API позволяет записывать данные на жесткий диск пользователя и обращаться к ним, как это делается в настольных приложениях. Процессы хранения и извлечения данных применимы в двух ситуациях: когда данные доступны в течении одного сеанса и когда данные хранятся долго, до тех пор, пока пользователь сам их не удалит. Таким образом API разделен на две части: sessionStorage и localStorage:

sessionStorage. Это маханизм хранения, удерживающий данные на протяжении сеанса одной страницы. В отличии от настоящих сеансов, доступ к информации есть только у одного окна или вкладки браузера. Как только окно или вкладка закрывается, эта информация удаляется.

localStorage. Этот механизм работает аналогично системам хранения настольных приложений. Данные записываются навсегда. Приложение, сохранившее их может обращаться к ним в любой момент.

Оба механизма работают через один и тот же интерфейс и предлагают одинаковые методы и свойства. Поэтому, для тестирования работы обоих механизмов можно использовать один html-шаблон

2 Создание и извлечение данных

И sessionStorage и localStorage сохраняют данные в форме отдельных элементов. Элементом считается пара из ключевого слова и значения. Каждое значение перед помещением в строку необходимо конвертировать в строку.

Для создания и извлечения элементов из пространства хранилища предназначены два новых метода:

setItem(key, value). Для создания, где key – это ключевое слово, value – это значение.

getItem(key). Для извлечения по ключевому слову.

```
function initiate(){
  var button = document.getElementById('save');
  button.addEventListener('click', newitem);
}
function newitem(){
  var keyword = document.getElementById('keyword').value;
  var value = document.getElementById('text').value;
  sessionStorage.setItem(keyword, value);

  show(keyword);
}
function show(keyword){
  var databox = document.getElementById('databox');
  var value = sessionStorage.getItem(keyword);
```

```
  databox.innerHTML = ' ' + keyword + ' - ' + value + ' \n ';
}
addEventListener('load', initiate);
```

Функция newitem() выполняется каждый раз, когда пользователь щелкает на кнопке формы. Эта функция создает элемент и добавляет в него информацию, полученную из формы, а затем вызывает функцию show(). Функция show() в свою очередь извлекает элемент из хранилища по ключевому слову, используя метод getItem(), а затем выводит его на экран.

Помимо этих методов, API хранения предоставляет упрощенный способ создания и извлечения элементов из пространства хранилища, в котором ключевое слово элемента используется, как свойство. Можно переменную ключевого слова заключать в квадратные скобки.

sessionStorage[ключевое_слово] = значение

, а можно передать строку в качестве имени свойства, например

sessionStorage.myitem = значение

```
function initiate(){
  var button = document.getElementById('save');
  button.addEventListener('click', newitem);
}
function newitem(){
  var keyword = document.getElementById('keyword').value;
  var value = document.getElementById('text').value;
  sessionStorage[keyword] = value;

  show(keyword);
}
function show(keyword){
  var databox = document.getElementById('databox');
  var value = sessionStorage[keyword];
  databox.innerHTML = '
' + keyword + ' - ' + value + '
';
}
addEventListener('load', initiate);
```

Рассмотрим методы и свойства API, позволяющие манипулировать данными:

length. Возвращает число элементов, помещенных в хранилище данным приложением.

key(index). Элементы записываются в хранилище последовательно, и им автоматически присваиваются порядковые номера, начиная с 0. С помощью данного метода можно извлечь определенный элемент или даже всю информацию, содержащуюся в хранилище, если пройтись по нему в цикле.

```
function initiate(){
  var button = document.getElementById('save');
```

```
  button.addEventListener('click', newitem);
  show();
}
function newitem(){
  var keyword = document.getElementById('keyword').value;
  var value = document.getElementById('text').value;

  sessionStorage.setItem(keyword, value);
  document.getElementById('keyword').value = '';
  document.getElementById('text').value = '';
  show();
}
function show(){
  var databox = document.getElementById('databox');
  databox.innerHTML = '';
  for(var f = 0; f < sessionStorage.length; f++){
    var keyword = sessionStorage.key(f);
    var value = sessionStorage.getItem(keyword);
    databox.innerHTML += '
' + keyword + ' - ' + value + '
';
  }
}
addEventListener('load', initiate);
```

Задача данного листинга вывести полный список элементов из хранилища. Мы немного усовершенствовали функцию show(), применив свойство length и метод key(). Для этого создали цикл for, начинающийся с 0, и заканчивающийся порядковым номером последнего элемента из хранилища. Функция show() вызывается из функции init(). Таким образом, она выводит список элементов из хранилища на экран сразу же, как только приложение запускается.

3 Удаление данных

Для удаления данных предназначены два метода:

- **removeItem(key)**. Удаляет один элемент по ключевому словуа
- **clear()**. Очищает пространство хранилища. Удаляются все находящиеся в нем элементы.

```
function removeItem(keyword){
  if(confirm('Are you sure?')){
    sessionStorage.removeItem(keyword);
    show();
  }
}
function removeAll(){
  if(confirm('Are you sure?')){
    sessionStorage.clear();
    show();
```

```
  }
}
```

4 Сохранение чисел и дат

Т.к. сохраняемые данные автоматически преобразуются в текст, то перед выводом, если мы хотим получить число, данные нужно преобразовать с помощью функции Number():

```
var value = Nubmer(sessionStorage[keyword]);
```

При преобразовании типов, следует проявлять осторожность. Для некоторых типов данных существуют удобные процедуры преобразования, но например, если мы сохранили следующую дату:

var today = new Date();

Этот код сохранит не объект даты, а текстовую стоку. Например, Sat Jun 2013 13:30:46. К сожалению, не существует легкого способа преобразования этого текста обратно в дату.

Чтобы решить эту проблему, мы должны явно преобразовать дату в текст, а потом выполнить обратное преобразование.

```
var today = new Date();
sessionStorage['session_started'] = today.getFullYear() + “/”
   + today.getMonth() + “/” + today.getDate();
…
today = new Date(sessionStorage['session_started']);
alert(today.getFullYear);
```

В результате выполнения этого кода, появится окно сообщения, подтверждающее успешное восстановление объекта даты.

Сохранение объектов

Для того, чтобы сохранить объект, необходимо его преобразовать в текст. Существует стандартный способ, позволяющий это делать, называемый кодированием JSON.

Создадим функцию PersonalityScore:

```
function PersonalityScore(o, c, e, a, n){
 this.openness = o;
 this.cons = c;
 this.extraversion = e;
 this.agreeable = a;
 this.neuroticism = n;
}
```

Создаем объект PersonalityScore

```
 var score = new PersonalityScore(o, c, e, a, n);
```

Для преобразования объекта в формат JSON, вызовем метод JSON.stringify().

```
sessionStorage['personalityScore'] = JSON.stringify(score);
```

Для обратного преобразования, воспользуемся методом JSON.parse:

```
var scope = JSON.parse(sessionStorage);
```

Если мы собираемся сохранять большие объемы данных и на длительное время, то необходимо использовать объект localStorage. При этом, решение о том, требуется ли информация, хранящаяся в объекте или нет, принимает сам пользователь.

Система localStorage использует такой же интерфейс, что и sessionStorage. Поэтому, для localStorage можно использовать те же методы и свойства, которые мы использовали ранее. Придется внести единственное изменение: заменить префикс session префиксом local.

```
localStorage.setItem(keyword, value);
```

5 Слежение за областью HTML5-хранилища

Если вы хотите программно отслеживать изменения хранилища, то должны отлавливать событие storage. Это событие возникает в объекте window, когда setItem(), removeItem() или clear() вызываются и что-то изменяют. Например, если вы установили существующее значение или вызвали clear() когда нет ключей, то событие не сработает, потому что область хранения на самом деле не изменилась.

Событие storage поддерживается везде, где работает объект localStorage, включая Internet Explorer 8. IE 8 не поддерживает стандарт W3C addEventListener (хотя он, наконец-то, добавлен в IE 9), поэтому, чтобы отловить событие storage нужно проверить, какой механизм событий поддерживает браузер (если вы уже проделывали это раньше с другими событиями, то можете пропустить этот раздел до конца). Перехват события storage работает так же, как и перехват других событий. Если вы предпочитаете использовать jQuery или какую-либо другую библиотеку JavaScript для регистрации обработчиков событий, то можете проделать это и со storage тоже.

```
if (window.addEventListener) {
  window.addEventListener("storage", handle_storage, false);
} else {
  window.attachEvent("onstorage", handle_storage);
};
```

Функция обратного вызова handle_storage будет вызвана с объектом StorageEvent, за исключением Internet Explorer, где события хранятся в window.event.

```
function handle_storage(e) {
  if (!e) { e = window.event; }
}
```

В данном случае переменная e будет объектом StorageEvent, который обладает следующими полезными свойствами.

Свойство	Тип	Описание
key	string	Ключ может быть добавлен, удален или изменен.
oldValue	любой	Предыдущее значение (если переписано) или null, если добавлено новое значение.
newValue	любой	Новое значение или null, если удалено.
url*	string	Страница, которая вызывает метод, приведший к изменению.

* Примечание: свойство url изначально называлось uri и некоторые браузеры поддерживали это свойство перед изменением спецификации. Для обеспечения максимальной совместимости вы должны проверить существует ли свойство url, и если нет проверить вместо него свойство uri.

Событие storage нельзя отменить, внутри функции обратного вызова handle_storage нет возможности остановить изменение. Это просто способ браузеру сказать вам: «Эй, это только что случилось. Вы ничего не можете сделать, я просто хотел, чтобы вы знали».

Заключение

Список использованных источников

Приложения