**Лабораторная работа №4**

**Работа с куки (cookie) и Local Storage в JavaScript**

Куки - это небольшой объем данных, которые хранятся вэб браузером. Они позволяют Вам сохранять определенную информацию о пользователе и получать ее каждый раз, когда он посещает Вашу страницу. Каждый пользователь имеет свой собственный уникальный набор куков.

Обычно куки используются веб сервером для выполнения таких функций как отслеживание посещений сайта, регистрации на сайте и сохранения сведений о заказах или покупках.

В JavaScript куки доступны с помощью свойства cookie объекта document. Создать куки можно следующим образом:

**document.cookie = "name=значение; expires=дата; path=путь; domain=домен; secure";**

Получить весь сохраненый набор куков так:

**var x = document.cookie;**

Для сохранения куки нужно присвоить document.cookie текстовую строку, которая содержит свойства куки, которые мы хотим создать:

**document.cookie = "name=значение; expires=дата; path=путь; domain=домен; secure";**

Свойства описаны в таблице:

|  |  |  |
| --- | --- | --- |
| **Свойство** | **Описание** | **Пример** |
| *name*=*значение* | Устанавливает имя куки и его значение. | username=John |
| expires=*дата* | Устанавливает дату истечения срока хранения куки. Дата должна быть представлена в формате, который возвращает метод toGMTString() объекта Date. Если значение expires не задано, куки будет удалено при закрытии браузера. | expires= 13/06/2003 00:00:00 |
| path=*путь* | Данная опция устанавливает путь на сайте, в рамках которого действует куки. Получить значение куки могут только документы из указанного пути. Обычно данное свойство оставляют пустым, что означает что только документ установивший куки может получит доступ к нему. | path=/demo/ |
| domain=*домен* | Данная опция устанавливает домен, в рамках которого действует куки. Получить значение куки могут только сайты из указанного домена. Обычно данное свойство оставляют пустым, что означает, что только домен установивший куки может получит доступ к нему. | domain=ruseller.com |
| secure | Данная опция указывает браузеру, что для пересылки куки на сервер следует использовать SSL. Очень редко используется. | secure |

Пример установки куки:

**document.cookie = "username=John; expires=15/02/2019 00:00:00";**

Данный код устанавливает куки username, и присваивает ему значение "John", которое будет храниться до 15-го февраля 2019 года (используется Европейский формат времени!).

var cookie\_date = new Date ( 2003, 01, 15 );

document.cookie = "username=John; expires=" + cookie\_date.toGMTString();

Данный код выполняет точно такое же действие, как и предыдущий пример, но для установки даты используется метод Date.toGMTString(). Учтите, что нумерация месяца в объекте Date начинается с 0, то есть февраль - это 01.

**document.cookie = "logged\_in=yes";**

Данный код устанавливает куки logged\_in, и присваивает ему значение "yes". Так как атрибут expires не установлен, то куки удалится при закрытии браузера.

**var cookie\_date = new Date ( ); // Текущая дата и время**

**cookie\_date.setTime ( cookie\_date.getTime() - 1 );**

**document.cookie = "logged\_in=; expires=" + cookie\_date.toGMTString();**

Данный код устанавливает куки logged\_in и присваивает строку хранения значение времени за одну секунду перед текущим - такая операция приведет к немедленному удалению куки. Ручной способ удалить куки!

**Перекодирование значения куки!**

Следует перекодировать значение куки для корректного хранения и отображения таких символов как пробел и двоеточие. Такая операция гарантирует, что браузер корректно воспримет значение. Перекодирование легко выполняется функцией JavaScript escape(). Например:

**document.cookie = "username=" + escape("John Пупкин") + "; expires=15/02/2019 00:00:00";**

**Пример1**

Т.к. браузеры Хром, Яндекс, Опера и т.д. не работают с куки с локальных адресов, проверку примера можно произвести в IE или на web-сервере.

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8">

<script>

function writeCookie(name, value, days) {

// По умолчанию куки являются временными, не имея срока хранения

var expires = "";

// Указав число дней, сделаем куки постоянными

if(days) {

var date = new Date();

date.setTime(date.getTime() + (days \* 24 \* 60 \* 60 \* 1000));

expires = "; expires=" + date.toGMTString();

}

// Присвоим куки имя, значение и срок хранения

document.cookie = name + "=" + value + expires + "; path=/";

}

function readCookie(name) {

// Найдем конкретный куки и вернем его значение

var searchName = name + "=";

var cookies = document.cookie.split(';');

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

var c = cookies[i];

while (c.charAt(0) == ' ')

c = c.substring(1, c.length);

if(c.indexOf(searchName) == 0)

return c.substring(searchName.length, c.length);

}

return null;

}

function readCookiePlace() {

var log=readCookie('Login');

var pw=readCookie('PWD');

document.getElementById('place').innerHTML="<HR/>"+

"Login: " + log + "<BR/>"+ "Password: " + pw +"<HR/>";

}

//Функция "удаляет" куки из браузера посредством установки срока хранения

// на одну секунду раньше текущего значения времени.

function delete\_cookie ( cookie\_name )

{

var cookie\_date = new Date ( ); // Текущая дата и время

cookie\_date.setTime ( cookie\_date.getTime() - 1 );

document.cookie = cookie\_name += "=; expires=" + cookie\_date.toGMTString();

}

//-->

</script>

</head>

<body>

<FORM NAME="Sel1">

<TABLE>

<TR><TD>Login:</TD>

<TD><INPUT NAME="Login" SIZE=20 ></TD></TR>

<TR><TD>Password:</TD>

<TD><INPUT NAME="Pas" SIZE=20 ></TD></TR>

</TABLE>

<!-- Кнопки готовности и сброса -->

<INPUT TYPE="button" VALUE="Запомнить" onClick="writeCookie('Login', Login.value,1); writeCookie('PWD', Pas.value,1); ">

<INPUT TYPE="button" VALUE="Вспомнить" onClick="alert('login='+ readCookie('Login') +' password=' +readCookie('PWD'));">

<INPUT TYPE="button" VALUE="Вывести в div c id=place" onClick="readCookiePlace();">

<INPUT TYPE="reset" VALUE="Сброс" >

</FORM>

<div id="place"></div>

</body>

</html>

**Задание 1**

Для сайта, созданного в лабораторной работе №1 организовать форму регистрации в левом верхнем углу. Хранение логина и пароля организовать в КУКИ. При перезагрузке страницы вывести приветствие по имени пользователя, например «Привет, Вася Пупкин!». Поля формы регистрации при этом заполняются логином и паролем из куки.

Используйте событие <body onload="функция()">

Для изменения значения поля input используйте

document.getElementById('ваш id').value = 'changed Value';



**JavaScript и Local Storage**

Local Storage (локальное хранилище). Эта технология поддерживается практически во всех современных браузерах.

Объем хранимой информации в LS по сравнению с cookie значительно выше: около 5Мб против 4Кб. К тому же, в LS данные хранятся в зашифрованном виде. Однако, как и в cookie, так и в LocalStorage, мы можем записывать только строковые данные.

Основные методы:

**localStorage.setItem('key', 'value');**

Обновляет или создает новую запись с ключом "key" и строковым значением "value"

**var lsData = localStorage.getItem('key');**

Возвращает данные связанные с ключом "key" или "null", если записи с таким ключом не обнаружено

**localStorage.removeItem('key');**

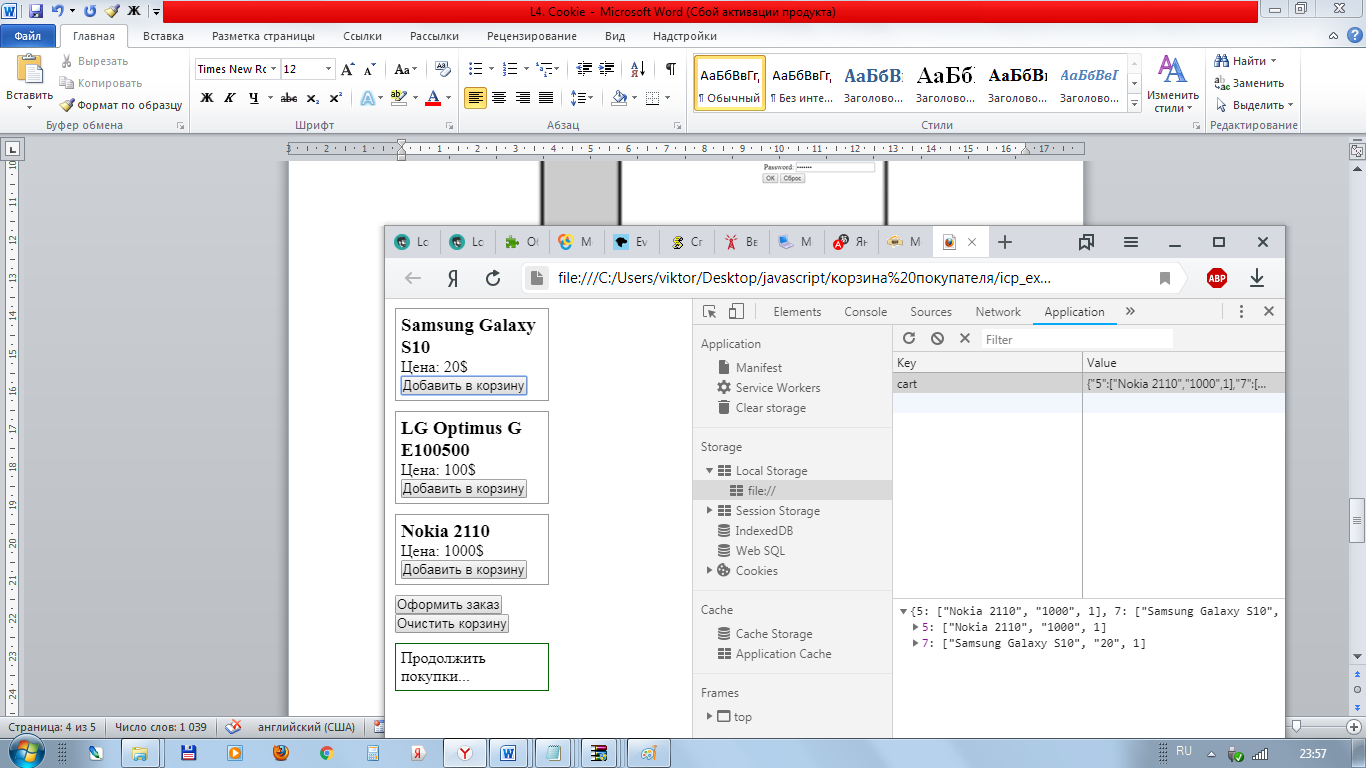
Удаляет данные со связанным ключом "key"

**localStorage.clear();**

Удаляет все записи из Local Storage

Посмотреть, что хранится в LocalStorage можно

Для браузеров Хром и Яндекс F12(Инструменты разработчика) 🡪 Application



Данные в LocalStorage хранятся в формате JSON - основанным на использовании текста, способе хранить и передавать структурированные данные.

Пример хранения заказа в формате JSON

{

"orderID": 12345,

"shopperName": "Ваня Иванов",

"shopperEmail": "ivanov@example.com",

"contents": [

{

"productID": 34,

"productName": "Супер товар",

"quantity": 1

},

{

"productID": 56,

"productName": "Чудо товар",

"quantity": 3

}

],

"orderCompleted": true

}

JavaScript имеет встроенный метод **JSON.stringify(),** который берет переменную и возвращает строку JSON, представляющую ее содержание.

**Пример2. Преобразование объекта JavaScript в строку JSON**

<html>

<head>

<script type="text/javascript">

var cart = {

"orderID": 12345,

"shopperName": "Ваня Иванов",

"shopperEmail": "ivanov@example.com",

"contents": [

{

"productID": 34,

"productName": "Супер товар",

"quantity": 1

},

{

"productID": 56,

"productName": "Чудо товар",

"quantity": 3

}

],

"orderCompleted": true

};

alert ( JSON.stringify( cart ) );

</script>

</head>

</html>

Существует несколько способов разобрать строку JSON в JavaScript, но самый безопасный и надежный - использовать встроенный метод **JSON.parse().** Он получает строку JSON и возвращает объект или массив JavaScript, который содержит данные.

**Пример3. Преобразование строки JSON в объект JavaScript**

<html>

<head>

<script type="text/javascript">

var jsonString = ' \

{ \

"orderID": 12345, \

"shopperName": "Ваня Иванов", \

"shopperEmail": "ivanov@example.com", \

"contents": [ \

{ \

"productID": 34, \

"productName": "Супер товар", \

"quantity": 1 \

}, \

{ \

"productID": 56, \

"productName": "Чудо товар", \

"quantity": 3 \

} \

], \

"orderCompleted": true \

} \

';

var cart = JSON.parse ( jsonString );

alert ( cart.shopperEmail );

alert ( cart.contents[1].productName );

</script>

</head>

</html>

**Пример 4. Хранение сведений о выбранных товарах в корзине покупателя**

Используемые методы.

Метод **document.querySelectorAll()** возвращает список элементов HTML-документа, которые соответствуют указанной группе селекторов .С полученными элементами можно будет производить различные манипуляции: менять их текст, атрибуты, css стили и так далее.

Метод возвращает псевдомассив элементов, а не один элемент, поэтому по этому набору сначала нужно пройтись циклом и для каждого элемента сделать то, что вам нужно.

Метод **addEventListener** является современным способом назначить или удалить обработчик, и при этом позволяет использовать сколько угодно любых обработчиков.

Назначение обработчика осуществляется вызовом addEventListener с тремя аргументами:

element.addEventListener(event, handler[, phase]);

event

Имя события, например click

handler

Ссылка на функцию, которую надо поставить обработчиком.

phase

Необязательный аргумент, «фаза», на которой обработчик должен сработать.

Метод **attachEvent** альтернатива методу **addEventListener** для IE.

<!doctype html>

<html>

<head>

<meta charset="utf-8">

<title>Корзина покупателя</title>

<style type="text/css">

\* {margin: 0; padding: 0;}

#wrapper {

width: 50%;

margin: 10px;

}

#cart\_content {

margin-top: 10px;

padding: 5px;

border: 1px solid #060;

}

.item\_box {

border: 1px solid #999;

margin-bottom: 10px;

padding: 5px;

}

.shopping\_list {

width: 100%;

margin-top: 10px;

border-collapse: collapse;

}

.shopping\_list td,

.shopping\_list th {

padding: 10px;

border: 1px solid #AAAAAA;

}

</style>

</head>

<body>

<div id="wrapper">

<div class="item\_box">

<h3 class="item\_title">Samsung Galaxy</h3>

<p>Цена: <span class="item\_price">50</span>$</p>

<button class="add\_item" data-id="7">Добавить в корзину</button>

</div>

<div class="item\_box">

<h3 class="item\_title">LG </h3>

<p>Цена: <span class="item\_price">200</span>$</p>

<button class="add\_item" data-id="2">Добавить в корзину</button>

</div>

<div class="item\_box">

<h3 class="item\_title">Lenovo</h3>

<p>Цена: <span class="item\_price">1000</span>$</p>

<button class="add\_item" data-id="5">Добавить в корзину</button>

</div>

<p><button id="checkout">Оформить заказ</button> &nbsp; <button id="clear\_cart">Очистить корзину</button></p>

<div id="cart\_content"></div>

</div>

<script type="text/javascript">

var d = document,

itemBox = d.querySelectorAll('.item\_box'), // блок каждого товара

cartCont = d.getElementById('cart\_content'); // блок вывода данных корзины

// Функция кроссбраузерная установка обработчика событий

function addEvent(elem, type, handler){

if(elem.addEventListener){

elem.addEventListener(type, handler, false);

} else {

elem.attachEvent('on'+type, function(){ handler.call(elem);

});

}

return false;

}

// Получаем данные из LocalStorage

function getCartData(){

return JSON.parse(localStorage.getItem('cart'));

}

// Записываем данные в LocalStorage

function setCartData(o){

localStorage.setItem('cart', JSON.stringify(o));

return false;

}

// Добавляем товар в корзину

function addToCart(e){

this.disabled = true; // блокируем кнопку на время операции с корзиной

var cartData = getCartData() || {}, // получаем данные корзины или создаём новый объект, если данных еще нет

parentBox = this.parentNode, // родительский элемент кнопки &quot;Добавить в корзину&quot;

itemId = this.getAttribute('data-id'), // ID товара

itemTitle = parentBox.querySelector('.item\_title').innerHTML, // название товара

itemPrice = parentBox.querySelector('.item\_price').innerHTML; //стоимость товара

if(cartData.hasOwnProperty(itemId)){ // если такой товар уже в корзине, то добавляем +1 к его количеству

cartData[itemId][2] += 1;

} else { // если товара в корзине еще нет, то добавляем в объект

cartData[itemId] = [itemTitle, itemPrice, 1];

}

// Обновляем данные в LocalStorage

if(!setCartData(cartData)){

this.disabled = false; // разблокируем кнопку после обновления LS

cartCont.innerHTML = 'Товар добавлен в корзину.';

setTimeout(function(){

cartCont.innerHTML = 'Продолжить покупки...';}, 1000);

}

return false;

}

// Устанавливаем обработчик события на каждую кнопку &quot;Добавить в корзину&quot;

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

addEvent(itemBox[i].querySelector('.add\_item'), 'click', addToCart);

}

// Открываем корзину со списком добавленных товаров

function openCart(e){

var cartData = getCartData(), // вытаскиваем все данные корзины

totalItems = '';

console.log(JSON.stringify(cartData));

// если что-то в корзине уже есть, начинаем формировать данные для вывода

if(cartData !== null){

totalItems = '<table class="shopping\_list"><tr><th>Наименование</th><th>Цена</th><th>Кол-во</th></tr>';

for(var items in cartData){

totalItems += '<tr>';

for(var i = 0; i < cartData[items].length; i++){

totalItems += '<td>' + cartData[items][i] + '</td>';

}

totalItems += '</tr>';

}

totalItems += '<table>';

cartCont.innerHTML = totalItems;

} else {

// если в корзине пусто, то сигнализируем об этом

cartCont.innerHTML = 'В корзине пусто!';

}

return false;

}

/\* Открыть корзину \*/

addEvent(d.getElementById('checkout'), 'click', openCart);

/\* Очистить корзину \*/

addEvent(d.getElementById('clear\_cart'), 'click', function(e){

localStorage.removeItem('cart');

cartCont.innerHTML = 'Корзина очишена.';

});

</script>

</body>

</html>

**Задание 2**

Для формы регистрации из задания 1 организовать хранение логина и пароля в LocalStorage.

**Задание 3**

Поместить скрипт корзины покупателя на отдельную страницу своего сайта. В случае, если корзина не пустая, выводить на главную страницу сообщение «В корзине есть заказы»

**Задание 4 (на отлично)**

Доработать код примера 4 так, чтобы выводились общее количество и общая стоимость заказанных товаров.