**Лабораторная работа №15**

**Тема работы:** Разработка программ с применением инкапсуляции в JavaScript

**Цель работы:** закрепить навыки по работе с инкапсуляцией в JavaScript.

**Теоретические сведения**

**Работа с JSON в JavaScript**

**Формат JSON**

Сам **JSON** представляет собой строку. В этой строке данные хранятся в формате, похожим на обычные объекты JavaScript. Но есть отличия: все строки (как ключи, так и значения) обязательно должны быть в двойных кавычках. Одинарные кавычки и ключи без кавычек запрещены.

Кроме того: числа в кавычки можно не брать, разрешены значения null и true/false. Также в качестве значений могут быть массивы и объекты любого уровня вложенности.

Давайте посмотрим на строки с корректным **JSON**:

var json **=** '[0, 1, 2, 3]';

var json **=** '{

"name": "Иван",

"age": 20,

"online": true,

"wife": null,

"arr": [1, 2, 3, 4],

}';

А вот этот JSON будет некорректным:

var json **=** '{

name: "Иван", //ключ без кавычек недопустим

"age": '20', //одинарные кавычки недопустимы

}';

var result **=** JSON.parse(json);

console.log(result);

Давайте теперь рассмотрим методы для работы с JSON.

**Метод JSON.parse**

Метод **JSON.parse** преобразует строку с JSON в обычный объект JavaScipt, с которым можно будет работать дальше обычными средствами:

var json **=** '{

"name": "Иван",

"age": 20,

"online": true,

"wife": null,

"arr": [1, 2, 3, 4],

}';

var result **=** JSON.parse(json);

console.log(result); //увидим объект

**Метод JSON.stringify**

Метод **JSON.stringify** преобразует объекты JavaScript в строку JSON. Посмотрим на следующих примерах:

var obj **=** {

name: 'Иван',

age: 20,

bool: true,

arr: [1, 2, 3, 4],

};

var json **=** JSON.stringify(obj);

console.log(json);

В результате преобразования получится такая строка:

'{

"name": "Иван",

"age": 20,

"empty": null,

"arr": [1, 2, 3, 4],

}';

Учтите, что NaN и Infinity преобразуются в **null**:

var obj **=** {

value1: null,

value2: NaN,

value3: Infinity,

};

json **=** JSON.stringify(obj);

console.log(json);

В результате преобразования получится такая строка:

'{

"value1": true,

"value2": null,

"value3": null,

}';

Преобразование объекта в JSON называется *сериализацией* (говорят: *сериализуем* объект в строку).

**Работа с локальным хранилищем**

Для работы с локальным хранилищем предназначен встроенный в браузер объект localStorage. У него есть 4 простых для понимания методов. Вот они:

//Сохранение значения:

localStorage.setItem('Ключ', 'Значение');

//Получение значения:

var value **=** localStorage.getItem('Ключ');

//Удаление значения:

localStorage.removeItem('Ключ');

//Очистка всего хранилища:

localStorage.clear();

С **localStorage** можно также работать как с обычным массивом:

//Сохранение значения:

localStorage['Ключ'] **=** 'Значение';

//Получение значения:

var value **=** localStorage['Ключ'];

//Удаление значения:

delete localStorage['Ключ'];

Кроме объекта **localStorage** есть также объект **sessionStorage**. Работа с ним осуществляется так же, единственное отличие в том, что все данные из него автоматически уничтожаются после закрытия браузера или вкладки с сайтом. Ну, а localStorage хранит данные длительное время, пока эти данные не будут удалены скриптом, или пользователь браузера очистит локальное хранилище с помощью настроек.

**Примеры**

В следующем примере мы запишем имя пользователя в локальное хранилище:

localStorage.setItem('name', 'Иван');

Через некоторое время получим это имя обратно:

alert(localStorage.getItem('name'));

Как вы видите - ничего сложно здесь нет, все гораздо проще, чем та же работа с куками.

**Сохранение объектов**

Локальное хранилище не способно хранить объекты и массивы JavaScript, хотя это зачастую бывает удобно. Но способ есть - нужно сериализовать эти данные в формат JSON - получится строка, которую уже можно будет сохранить в localStorage. Затем, когда нам понадобится достать этот объект обратно - преобразуем строку из JSON обратно в объект - и спокойно им пользуемся.

Давайте рассмотрим этот процесс на примере. Сериализуем объект и сохраним в локальное хранилище:

//Дан объект:

var obj **=** {name: 'Иван', arr: [1, 2, 3]};

//Сериализуем его в '{"name": "Иван", "arr": [1, 2, 3]}':

var json **=** JSON.stringify(obj);

//Запишем в localStorage с ключом obj:

localStorage.setItem('obj', json);

Через некоторое время получим объект обратно:

//Получим данные обратно из localStorage в виде JSON:

var json **=** localStorage.getItem('obj');

//Преобразуем их обратно в объект JavaScript:

var obj **=** JSON.parse(json);

console.log(obj);

**Дополнительные возможности**

Определение количества записей в хранилище: alert(localStorage.length).

Определение названия ключа по его номеру: alert(localStorage.key(номер)).

При выполнении операций с хранилищем, срабатывает событие **onstorage**. Если к этому событию привязать функцию - то в ней будет доступен объект Event со следующими свойствами:

<body onstorage="func()">

function func(event) {

var key **=** event.key; //ключ изменяемых данных

var oldValue **=** event.oldValue; //старое значение

var newValue **=** event.newValue; //новое значение

var storageArea **=** event.storageArea; //storageArea

}

**Порядок выполнения работы**

**Работа с JSON.parse**

1. Дана JSON строка '["Коля", "Вася", "Петя"]'. Преобразуйте ее в массив JavaScript и выведите на экран элемент "Петя".

**Работа с JSON.stringify**

1. Дан объект {a: 'aaa', b: 'bbb'}. Преобразуйте его в JSON.

**Работа с локальным хранилищем**

**Основы работы**

1. Дан текстареа. В него вводится текст. Сделайте так, чтобы после захода на эту страницу через некоторое время, введенный текст остался в текстареа. Текст должен запоминаться в локальном хранилище.

**Закрепление**

1. Дан текстареа. В него можно ввести данные, затем поредактировать их, затем еще поредактировать. Пусть текстареа хранит историю своих изменений (даже после перезагрузки страницы). Сверху над текстареа должны появится стрелочки, с помощью которых можно перемещаться по истории.
2. Дана форма с инпутами, текстареа, чекбоксами, радио кнопочками, селектами и тп. Пользователь вводит какие-то данные и закрывает страницу (не факт, что он заполнил всю форму). Сделайте так, чтобы при следующем заходе на страницу введенные им ранее данные стояли на своих местах. Сделайте ваш скрипт как можно более универсальным.
3. Реализуйте записную книгу, хранящую данные в локальном хранилище.
4. Реализуйте органайзер, хранящий данные в локальном хранилище.