# Лабораторная работа № 10 Сериализация объектов. Библиотека jQuery

**Цель работы:** ознакомиться с нотацией JSON, научиться использовать библиотеку jQuery.

**Краткие теоретические сведения:**

**Сериализация объектов**

Для хранения и обмена данными между клиентом и сервером чаще всего используется формат JSON. JSON (JavaScript Object Notation) – способ записи данных в виде строки. Поскольку формат JSON является только текстовым, его можно легко отправлять на сервер и с сервера и использовать в качестве формата данных на любом языке программирования. Например, клиент использует JavaScript, а сервер написан на Ruby/PHP/Java или любом другом языке.

JavaScript предоставляет следующие методы для сериализации и десериализации объектов:

* **JSON.stringify(obj)** используется для преобразования объектов в строку JSON;
* **JSON.parse(str)** используется для преобразования строки в формате JSON обратно в объект.

JSON поддерживает три типа данных: примитивные значения, объекты и массивы. Примитивные значения представляют стандартные строки, числа, значение null, логические значения true и false.

<script>

const student = {

name: "Alex",

age: 21,

marks: {

OAIP: 9,

"System programming": 8,

RPI: 9,

ACS: 8

}

};

let str = JSON.stringify(student);

console.log(str); // {"name":"Alex","age":21,"marks":{"OAIP":9,"System programming":8,"RPI":9,"ACS":8}}

let alex = JSON.parse(str);

console.log(alex); // {name: "Alex", age: 21, marks: {…}}

</script>

**Web storage**

Для хранения данных на стороне клиента в HTML5 была внедрена новая концепция – web-storage. Web storage состоит из двух компонентов: **session storage** и **local storage**.

Session storage представляет временное хранилище информации, которая удаляется после закрытия браузера.

Local storage представляет хранилище для данных на постоянной основе. Данные из local storage автоматически не удаляются и не имеют срока действия. Объем local storage составляет в Chrome и Firefox 5 Mб для домена.

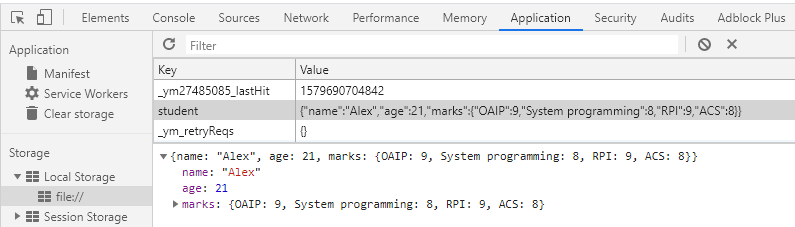
Все данные в web storage представляют набор пар ключ-значение. То есть каждый объект имеет уникальное имя-ключ и определенное значение.

Для работы с local storage в javascript используется объект **localStorage**, а для работы с session storage – объект **sessionStorage**. Эти объекты предомтавляют следующие методы для работы с данными:

* **setItem()** – для сохранения данных;
* **getItem()** – для получения сохраненных данных;
* **removeItem()** – для удаления объект из хранилища;
* **clear()** – для для полного удаления всех объектов из хранилища.

Например:

localStorage.setItem("student", JSON.stringify(student));



**Библиотека jQuery**

Современное веб-программирование и создание веб-приложений уже невозможно представить без использования языка JavaScript. Однако в настоящее время, все чаще используется не чистый код JavaScript, а JavaScript-фреймворки и библиотеки. Одной из таких библиотек является jQuery.

Подключив библиотеку jQuery вместо десятков команд на JavaScript можно написать несколько команд. Команды основаны на селекторах и классах CSS.

Библиотеку jQuery можно скачать с сайта <http://code.jquery.com/>. Если нет желания скачивать библиотеку её можно подключить с CDN (Content Delivery Network) компаний Google или Microsoft.

Вся работа с библиотекой сводится к выбору (query) HTML элемента и выполнения над ним различных действий

Основной синтаксис:

$(selector).action(),

где $() – функция доступа (определения) jQuery. (jQuery() – полное название функции $), selector – "query(запрос)" для определения HTML элементов, action() – действия над элементом (элементами).

Примеры:

$(this).hide() // скрывает текущий элемент.

$("p").hide() // скрывает все абзацы.

$(".test").hide() // скрывает все элементы класса "test".

$("#test").hide() // скрывает элемент с id="test".

**Создание элементов DOM**

$("<div>Hi</div>");

$("<div>")

Можно использовать кавычки, можно апострофы. Но наличие тегов обязательно. Задавать просто текст нельзя. Сама по себе эта команда ничего не выведет, надо этот узел привязать к родительскому элементу. Делается это так:

$("<div>Hi</div>").insertAfter("#a1")

Здесь a1 – это идентификатор объекта, после которого необходимо вставить элемент.

**Работа с полученным набором значений**

В терминах jQuery эти наборы называют обернутыми.

Для определения количество выбранных элементов используется свойство length или метода size, который также возвращает число выбранных элементов:

let num1 = $("tr:nth-child(odd)").length;

let num2 = $("tr:nth-child(odd)").size();

Для получения доступа к элементу выборки можно использовать индекс или использовать функцию eq(index). Второй способ более удобен так как функция возвращает объект:

let firstElement = $("tr:nth-child(odd)")[0];.

let elem = $("tr:even").eq(0);.

Для перебора элементов выборки можно использовать стандартные методы JavaScript, а также использовать специальный метод each:

$(function(){

$("tr:even").each(function(index, elem){

console.log(index + ". " + elem.innerHTML);

});

});

В метод each в качестве параметра передается безымянная функция, которая принимает два параметра: index - индекс элемента в наборе и elem - сам элемент.

**Манипулирование объектами на странице**

Библиотека jQuery предлагает нам инструментарий для манипуляции свойствами и атрибутами элементов:

* **prop(property, [value])** получает или изменяет значение свойства элемента;
* **removeProp(property)** удаляет свойство;
* **attr(attributeб [value])** получить или изменяет значение атрибута элемента;
* **css**(**property, [value]**) применяется для работы со стилями элемента;
* **html()** используется для получения или установки разметки;
* **text()** используется для получения или установки текста;
* **val()** используется для получения значений элементов форм.

**Обработка событий**

Различные браузеры по-своему могут обрабатывать события, jQuery пытается сгладить эти неприятности.

Модель событий jQuery обладает следующими свойствами:

* поддерживает единый метод установки событий;
* позволяет устанавливать несколько обработчиков для события;
* использует стандартные названия типов событий;
* предоставляет единые методы отмены события и блокирования действий по умолчанию.

jQuery предоставляет специальный метод **on**, который позволяет создать и зарегистрировать обработчики как для существующих, так и для будущих элементов, которых еще не в структуре DOM. Этот метод имеет следующие варианты использования:

* on('событие', 'селектор', обработчик\_события);
* on('событие', 'селектор', данные\_события, обработчик\_события) то же самое, что и в предыдущем случае, плюс параметр для передачи данных события в объект Event (e.data).

Для удаления обработчика используется команда unbind()

Метод **trigger()** используется для вызова обработчиков событий вручную. Например, вызвав метод trigger для нужного элемента мы можем также и вызвать обработчик события. Он имеет следующие варианты использования:

* trigger('событие'): вызов обработчика для данного события;
* trigger(объект\_Event): вызов обработчика с использованием объекта Event, который содержит данные о том, какой именно обработчик надо вызвать.

Кроме специализированных методов jQuery предоставляет прямые методы для обработки событий. Эти методы, как правило, носят наименование обрабатываемого события, а в качестве параметра принимают функцию обработчика данного события:

$('button').first().click(function(){

$(this).css('background-color', 'silver');

});

**Эффекты анимации**

К **базовым эффектам** в jQuery относятся эффекты скрытия и отображения элементов, которые достигаются с помощью методов **show()**, **hide()** и **toggle()**.

Все эти методы могут использоваться в трех вариантах (рассмотрим на примере метод hide):

* hide(): метод без параметров
* hide([duration][, complete]): принимает два необязательных параметра. Параметр duration указывает как долго анимация элемента будет длиться. По умолчанию его значение равно 400 миллисекунд. Параметр complete представляет функцию, вызываемую методом по завершению анимации
* hide([duration] [, easing][, complete]): то же самое, только добавляется параметр easing, который принимает название функции плавности анимации в виде строки. По умолчанию его значение равно "swing". Также можно использовать значения 'slow' и 'fast', которые соответствуют длительности эффекта в 600 и 200 миллисекунд.

**Эффекты скольжения** позволяют нам плавно скрыть или раскрыть элемент. Эффекты скольжения реализованы в виде методов **slideUp()**, **slideDown()** и **slideToggle()**.

**Эффекты прозрачности** позволяют нам, плавно изменяя прозрачность элемента, скрыть его или отобразить. Эффекты прозрачности реализованы с помощью методов **fadeOut()**, **fadeIn()**, **fadeTo()** и **fadeToggle()**.

Для создания более сложных по характеру эффектов используется метод **animate()**:

animate(properties [,duration] [,easing] [,complete])

Обязательный параметр properties содержит набор css-свойств, у которых указываются финальные значения. Параметр duration указывает, как долго будет длиться изменение прозрачности элемента. Параметр easing принимает название функции плавности анимации в виде строки. Параметр complete представляет функцию обратного вызова, вызываемую методом по завершении анимации

**Задание к лабораторной работе:**

**Задание 1**. В соответсвии с вариантом задания 1 лабораторной работы №6 сформировать массив объектов. Сериализовать массив в формат JSON с последующим сохранением его в объекте localStorage. Используя библиотеку jQuery, написать сценарий, считывающий информацию из локального хранилища и формирующий таблицу из полученных данных. Предусмотреть сортировку отображаемых данных при щелчке по заголовкам таблицы.

**Задание 2**. Создать HTML-документ с реализацией стандартногокалькулятора, выполняющего арифметические операции над введенными данными и имеющего возможность сохранения результатов в памяти. При написании сценария использовать..

**Задание 3**. Создать HTML-документ с полем ввода даты. Используя библиотеку jQuery, сформировать календарь для текущего месяца введенной даты. Календарь оформить в виде таблицы, неделя начинается с понедельника, в ячейки таблицы выводится день месяца.

**Задание 4**. Воспользоваться созданным в предыдущем задании календарем. Сформировать строку JSON с информацией о мероприятиях на текущий месяц. Строка хранится в Local Storage. Предусмотреть выделения цветом дней месяца, для которых запланировано мероприятие. При щелчке на такой день, должна выдаваться полная информация о мероприятии. В скрипте предусмотреть возможность добавления мероприятия на выбранный день с сохранением информации в Local Storage. Для добавления мероприятия и просмотра информации использовать модальное окно.

**Контрольные вопросы для защиты:**

1. Какие механизмы предусмотены для хранения информации на стороне клиента?

2. Что такое JSON?

3. Как сериализуется объекты?

4. Какие средства используются для восстановления данных из JSON строки?

5. Что такое jQuery?

6. Как получить доступ к элементу, используя jQuery?

7. Как изменить свойства элемента, используя jQuery?

8. Как получить доступ к атрибутам, используя jQuery?

9. Как изменить содержимое разметки элемента, используя jQuery?

10. Как получить контекст элемента, используя jQuery?

11. Как задать стили элемента, используя jQuery?

12. Как назначить обработчик события, используя jQuery?