

WEB-ТЕХНОЛОГИИ - ЛЕКЦИИ

02.03.03 -Математическое обеспечение и администрирование информационных систем,
направленность (профиль) -разработка и администрирование информационных систем

<https://vikchas.ru>

Лекция 8. Тема «jQuery 2.0 для веб-технологий»

Часовских Виктор Петрович
доктор технических наук, профессор кафедры
ШИиКМ, ФГБОУ ВО «Уральский государственный
экономический университет

Екатеринбург 2025

Введение

В современном мире веб-технологии развиваются стремительными темпами, и разработчикам необходимо использовать инструменты, которые упрощают и ускоряют процесс создания веб-приложений. Одной из таких технологий является библиотека jQuery, которая предоставляет разработчикам удобный способ взаимодействия с DOM-структурой, обработки событий, создания анимаций и выполнения AJAX-запросов.

jQuery — это быстрая, небольшая и многофункциональная JavaScript-библиотека. Она делает такие вещи, как обход и манипулирование HTML-документом, обработка событий, анимация и Ajax, намного проще с помощью простого в использовании API, который работает в множестве браузеров.

Целью данной лекции является изучение возможностей библиотеки jQuery 2.0, её применение в современных веб-технологиях, а также демонстрация практических примеров использования данной библиотеки для решения типовых задач веб-разработки.

Актуальность данной темы обусловлена широким распространением jQuery в веб-разработке и необходимостью изучения её возможностей для создания современных, интерактивных и кросс-браузерных веб-приложений.

История развития jQuery

jQuery была создана Джоном Резигом в 2006 году и впервые представлена на BarCamp NYC. Основной целью создания библиотеки было упрощение взаимодействия с DOM-элементами, а также обеспечение кросс-браузерной совместимости.

jQuery быстро обрела популярность благодаря своей простоте и эффективности. По данным различных исследований, jQuery используется на большинстве веб-сайтов в интернете, и её популярность обусловлена следующими факторами:

- Простой и понятный синтаксис
- Кросс-браузерная совместимость
- Расширяемость с помощью плагинов
- Обширная документация и сообщество

Архитектура и основные компоненты jQuery 2.0

Архитектура jQuery 2.0 построена на модульном принципе, что позволяет разработчикам использовать только необходимые компоненты библиотеки. Основой jQuery является объект jQuery (или \$), который предоставляет доступ ко всем функциям библиотеки.

Основные компоненты jQuery 2.0:

1. **Ядро (Core)**— основные функции библиотеки, включая создание объекта jQuery, обработку селекторов и базовые утилиты
2. **Селекторы (Selectors)**— механизм выбора DOM-элементов с использованием CSS-селекторов
3. **Манипуляции с DOM (DOM Manipulation)**— функции для изменения структуры и содержимого DOM-элементов
4. **События (Events)**— механизм обработки событий пользовательского интерфейса
5. **Анимации (Effects)**— функции для создания анимаций и визуальных эффектов
6. **AJAX**— средства для выполнения асинхронных HTTP-запросов
7. **Утилиты (Utilities)**— набор вспомогательных функций для работы с массивами, объектами и т.д.
8. **Промисы (Promises/Deferred)**— реализация асинхронных операций с использованием паттерна Promise

Архитектура jQuery 2.0 обеспечивает простоту использования библиотеки и возможность её расширения с помощью плагинов. Модульность позволяет разработчикам включать в свои проекты только необходимые компоненты, что уменьшает размер итогового кода и повышает производительность.

Основы работы с jQuery 2.0

Для начала работы с jQuery 2.0 необходимо подключить библиотеку к HTML-документу. Это можно сделать двумя способами:

1. Скачать библиотеку и подключить локальную копию:

HTML

```
<script src="путь/к/jquery-2.0.3.min.js"></script>
```

2. Использовать CDN (Content Delivery Network):

HTML

```
<script src="https://code.jquery.com/jquery-2.0.3.min.js"></script>
```

После подключения библиотеки можно использовать функционал jQuery. Основной синтаксис jQuery выглядит следующим образом:

JAVASCRIPT

```
$(selector).action();
```

Где:

- \$ — это сокращение для объекта jQuery
- selector — это CSS-селектор для выбора HTML-элементов
- action — это функция, которую нужно выполнить для выбранных элементов

Пример использования jQuery:

JAVASCRIPT

```
$(document).ready(function() {  
    // Код выполнится после полной загрузки DOM  
    $("p").click(function() {  
        // При клике на параграф он скроется  
        $(this).hide();  
    });  
});
```

Функция `$(document).ready()` обеспечивает выполнение кода только после полной загрузки DOM-структуры документа, что предотвращает возможные ошибки при попытке взаимодействия с элементами, которые еще не были загружены.

Альтернативный, более краткий синтаксис:

JAVASCRIPT

```
$(function() {  
    // Код, который выполнится после загрузки DOM  
});
```

Селекторы и их применение

Селекторы в jQuery используются для выбора HTML-элементов на странице. jQuery поддерживает большинство CSS-селекторов, а также добавляет свои собственные.

Основные типы селекторов:

1. Селекторы по тегу:

JAVASCRIPT

`$("p")` // Выбирает все элементы `<p>`

2. Селекторы по ID:

JAVASCRIPT

`$("#myId")` // Выбирает элемент с `id="myId"`

3. Селекторы по классу:

JAVASCRIPT

`$(".myClass")` // Выбирает все элементы с `class="myClass"`

4. Комбинированные селекторы:

JAVASCRIPT

`$("p.myClass")` // Выбирает все элементы `<p>` с классом `"myClass"`

5. Селекторы атрибутов:

JAVASCRIPT

`$("[href]")` // Выбирает все элементы с атрибутом `href`

`$("[href='#']")` // Выбирает все элементы с атрибутом `href` равным `"#"`

6. Селекторы позиции:

JAVASCRIPT

`$("li:first")` // Выбирает первый элемент `<li>`

`$("li:last")` // Выбирает последний элемент `<li>`

`$("li:even")` // Выбирает четные элементы `<li>`

`$("li:odd")` // Выбирает нечетные элементы `<li>`

7. Фильтры по содержимому:

JAVASCRIPT

`$("div:contains('текст')")` // Выбирает все `div`, содержащие указанный текст

Пример использования селекторов:

JAVASCRIPT

`$(function() {`

```
// Выбираем все параграфы внутри элемента с id="content"
$("#content p").css("color", "red");

// Выбираем все элементы с классом "highlight"
$(".highlight").css("background-color", "yellow");

// Выбираем все ссылки, которые открываются в новом окне
$("a[target='_blank']").addClass("external-link");
});
```

Манипуляции с DOM-элементами

Объектная Модель Документа (DOM) — это программный интерфейс для HTML, XML и SVG документов. Она предоставляет структурированное представление документа в виде дерева, где каждый узел представляет часть документа, такую как элемент, атрибут или текстовый контент. DOM позволяет программам изменять структуру, стиль и содержание документа, связывая веб-страницы со скриптами или языками программирования.

jQuery предоставляет обширный набор методов для манипуляции с DOM-элементами, включая изменение содержимого, атрибутов, стилей и структуры.

Получение и изменение содержимого:

JAVASCRIPT

```
// Получение HTML-содержимого
var content = $("#element").html();

// Изменение HTML-содержимого
$("#element").html("<p>Новое содержимое</p>");

// Получение текстового содержимого
var text = $("#element").text();

// Изменение текстового содержимого
```

```
$("#element").text("Новый текст");
```

```
// Получение значения input
```

```
var value = $("#myInput").val();
```

```
// Изменение значения input
```

```
$("#myInput").val("Новое значение");
```

Работа с атрибутами:

JAVASCRIPT

```
// Получение значения атрибута
```

```
var src = $("#myImage").attr("src");
```

```
// Установка значения атрибута
```

```
$("#myImage").attr("src", "new-image.jpg");
```

```
// Удаление атрибута
```

```
$("#myLink").removeAttr("target");
```

Работа с классами:

JAVASCRIPT

```
// Добавление класса
```

```
$("#element").addClass("highlight");
```

```
// Удаление класса
```

```
$("#element").removeClass("highlight");
```

```
// Переключение класса (если класс есть - удаляет, если нет - добавляет)
```

```
$("#element").toggleClass("highlight");
```

```
// Проверка наличия класса
```

```
if ($("#element").hasClass("highlight")) {
```

```
// Действия, если класс присутствует  
}
```

Работа со стилями CSS:

JAVASCRIPT

```
// Установка одного стиля
```

```
$("#element").css("color", "red");
```

```
// Установка нескольких стилей
```

```
$("#element").css({  
    "color": "red",  
    "font-size": "16px",  
    "backgroundColor": "black" // Можно использовать camelCase  
});
```

```
// Получение значения стиля
```

```
var color = $("#element").css("color");
```

Изменение структуры DOM:

JAVASCRIPT

```
// Создание нового элемента
```

```
var newElement = $("<div>Новый элемент</div>");
```

```
// Добавление элемента в конец
```

```
$("#container").append(newElement);
```

```
// Добавление элемента в начало
```

```
$("#container").prepend(newElement);
```

```
// Добавление перед элементом
```

```
$("#element").before(newElement);
```


// Добавление после элемента

```
$("#element").after(newElement);
```

// Обертывание элемента

```
$("#element").wrap("<div class='wrapper'></div>");
```

// Удаление элемента

```
$("#element").remove();
```

// Удаление содержимого элемента

```
$("#element").empty();
```

Размеры и позиция элементов:

JAVASCRIPT

// Получение ширины и высоты

```
var width = $("#element").width();
```

```
var height = $("#element").height();
```

// Установка ширины и высоты

```
$("#element").width(200);
```

```
$("#element").height(100);
```

// Получение позиции относительно документа

```
var position = $("#element").offset();
```

```
console.log(position.top, position.left);
```

// Получение позиции относительно родителя

```
var position = $("#element").position();
```

```
console.log(position.top, position.left);
```

Обработка событий

jQuery предоставляет удобный способ обработки событий, обеспечивая кросс-браузерную совместимость и дополнительную функциональность по сравнению с нативным JavaScript.

Основные методы для обработки событий:

JAVASCRIPT

// Обработка клика на элементе

```
$("#element").click(function(event) {
```

```
    // Действия при клике
```

```
    console.log("Элемент был нажат");
```

```
    // Предотвращение стандартного действия (например, перехода по ссылке)
```

```
    event.preventDefault();
```

```
    // Остановка распространения события
```

```
    event.stopPropagation();
```

```
});
```

// Обработка события наведения мыши (mouseenter и mouseleave)

```
$("#element").hover(
```

```
    function() {
```

```
        // Действия при наведении мыши
```

```
        $(this).addClass("hover");
```

```
    },
```

```
    function() {
```

```
        // Действия при уходе мыши
```

```
        $(this).removeClass("hover");
```

```
    }
```

```
);
```

// Универсальный метод для привязки обработчиков событий

```
$("#element").on("click", function() {
```

```
    // Действия при клике
```

```
});
```

// Привязка нескольких обработчиков одновременно

```
$("#element").on({
```

```
    click: function() {
```

```
        // Действия при клике
```

```
    },
```

```
    mouseenter: function() {
```

```
        // Действия при наведении мыши
```

```
    },
```

```
    mouseleave: function() {
```

```
        // Действия при уходе мыши
```

```
    }
```

```
});
```

// Делегирование событий (обработка событий на динамически созданных элементах)

```
$("#container").on("click", ".item", function() {
```

```
    // Этот обработчик будет работать для всех элементов с классом .item
```

```
    // внутри #container, даже если они будут созданы после привязки обработчика
```

```
});
```

// Удаление обработчика события

```
$("#element").off("click");
```

Основные события, поддерживаемые jQuery:

1. События мыши:

- click — клик на элементе
- dblclick — двойной клик
- mouseenter — указатель мыши входит в область элемента
- mouseleave — указатель мыши покидает область элемента
- mousemove — движение мыши внутри элемента
- mousedown — нажатие кнопки мыши
- mouseup — отпускание кнопки мыши

2. События клавиатуры:

- keydown — нажатие клавиши
- keyup — отпускание клавиши
- keypress — нажатие клавиши (символьной)

3. События формы:

- submit — отправка формы
- change — изменение значения элемента формы
- focus — элемент получает фокус
- blur — элемент теряет фокус

4. События документа/окна:

- load — загрузка ресурса завершена
- resize — изменение размера окна
- scroll — прокрутка элемента или окна
- unload — выгрузка документа

Пример обработки событий формы:

JAVASCRIPT

```
$(function() {  
    // Обработка отправки формы  
    $("#myForm").submit(function(event) {  
        // Предотвращаем стандартную отправку формы
```

```
event.preventDefault();

// Получаем данные из полей формы
var name = $("#name").val();
var email = $("#email").val();

// Проверка данных
if (name === "" || email === "") {
    alert("Пожалуйста, заполните все поля");
    return;
}

// Здесь можно выполнить AJAX-запрос для отправки данных на сервер
console.log("Форма отправлена с данными:", name, email);
});

// Обработка изменения полей ввода
$("#myForm input").change(function() {
    // Когда пользователь изменяет значение любого поля ввода
    console.log("Поле изменено:", $(this).attr("id"));
});
});
```

AJAX-запросы с использованием jQuery

AJAX (Asynchronous JavaScript and XML) — это технология, которая позволяет обновлять данные на веб-странице без её полной перезагрузки. Она используется для асинхронного обмена данными между браузером и сервером, что делает веб-приложения более быстрыми и интерактивными.

AJAX работает следующим образом: JavaScript отправляет запрос на сервер в фоновом режиме, сервер обрабатывает запрос и возвращает только необходимые данные. Эти данные могут быть в формате JSON, XML, HTML

или обычного текста. После получения ответа JavaScript обновляет только нужные части страницы, не затрагивая весь интерфейс.

jQuery значительно упрощает работу с AJAX (Asynchronous JavaScript and XML), предоставляя удобные методы для выполнения асинхронных HTTP-запросов.

Основные методы для работы с AJAX:

1. **\$.ajax()**— базовый метод для всех AJAX-запросов:

JAVASCRIPT

```
$.ajax({
    url: "api/data.json",
    type: "GET", // Метод HTTP (GET, POST, PUT, DELETE и т.д.)
    dataType: "json", // Тип ожидаемых данных (html, xml, json, text)
    data: { id: 123 }, // Данные, передаваемые на сервер
    success: function(response) {
        // Функция, выполняемая при успешном запросе
        console.log("Данные получены:", response);
    },
    error: function(xhr, status, error) {
        // Функция, выполняемая при ошибке
        console.error("Ошибка запроса:", error);
    },
    complete: function() {
        // Функция, выполняемая по завершении запроса (успешного или с ошибкой)
        console.log("Запрос завершен");
    },
    timeout: 5000 // Таймаут запроса в миллисекундах
});
```

Сокращенные методы для AJAX-запросов:

JAVASCRIPT

```
// GET-запрос
```

```
$.get("api/data.json", function(response) {  
    console.log("Данные получены:", response);  
});
```

```
// POST-запрос
```

```
$.post("api/save", { name: "John", age: 30 }, function(response) {  
    console.log("Данные сохранены:", response);  
});
```

```
// Загрузка JSON-данных
```

```
$.getJSON("api/data.json", function(data) {  
    console.log("JSON-данные получены:", data);  
});
```

Загрузка HTML-содержимого:

JAVASCRIPT

```
// Загрузка HTML-содержимого и вставка его в элемент
```

```
$("#container").load("content.html", function() {  
    console.log("Содержимое загружено");  
});
```

```
// Загрузка части содержимого из другой страницы
```

```
$("#container").load("page.html #section1", function() {  
    console.log("Секция загружена");  
});
```

Работа с промисами в AJAX:

JAVASCRIPT

```
// Использование методов Promise для AJAX-запросов
```

```
$.ajax({
```

```
    url: "api/data.json",
    type: "GET"
  })
  .done(function(response) {
    // Успешное выполнение
    console.log("Данные получены:", response);
  })
  .fail(function(xhr, status, error) {
    // Ошибка
    console.error("Ошибка запроса:", error);
  })
  .always(function() {
    // Выполняется всегда
    console.log("Запрос завершен");
  });
```

Пример комплексного AJAX-запроса с обработкой данных:

JAVASCRIPT

```
$(function() {
  $("#loadButton").click(function() {
    // Показываем индикатор загрузки
    $("#loading").show();

    // Выполняем AJAX-запрос
    $.ajax({
      url: "api/users",
      type: "GET",
      dataType: "json",
      timeout: 5000
    })
```



```
.done(function(users) {  
    // Очищаем контейнер  
    $("#userList").empty();  
  
    // Обрабатываем полученные данные  
    if (users.length > 0) {  
        // Создаем элементы списка для каждого пользователя  
        $.each(users, function(index, user) {  
            $("#userList").append(  
                $("<li>").append(  
                    $("<strong>").text(user.name),  
                    $("<span>").text(" - " + user.email)  
                )  
            );  
        });  
    } else {  
        $("#userList").append("<li>Пользователи не найдены</li>");  
    }  
})  
  
.fail(function(xhr, status, error) {  
    // Выводим сообщение об ошибке  
    $("#userList").html("<li class='error'>Ошибка при загрузке данных: " +  
error + "</li>");  
})  
  
.always(function() {  
    // Скрываем индикатор загрузки  
    $("#loading").hide();  
});  
});
```

});

Анимации и визуальные эффекты

jQuery предоставляет простые методы для создания анимаций и визуальных эффектов, которые работают во всех браузерах.

Основные методы анимации:

1. Простые эффекты:

JAVASCRIPT

// Скрытие элемента

```
$("#element").hide();
```

```
$("#element").hide("slow");    // с указанием скорости (slow, fast или  
миллисекунды)
```

```
$("#element").hide(1000);    // продолжительность в миллисекундах
```

// Показ элемента

```
$("#element").show();
```

```
$("#element").show("fast");
```

```
$("#element").show(500);
```

// Переключение видимости

```
$("#element").toggle();
```

```
$("#element").toggle(500);
```

// Плавное появление

```
$("#element").fadeIn(1000);
```

// Плавное исчезновение

```
$("#element").fadeOut(1000);
```

// Переключение прозрачности

```
$("#element").fadeToggle(1000);
```

```
// Установка прозрачности
```

```
$("#element").fadeTo(1000, 0.5); // полупрозрачный
```

```
// Скользящее раскрытие
```

```
$("#element").slideDown(1000);
```

```
// Скользящее скрытие
```

```
$("#element").slideUp(1000);
```

```
// Переключение скользящего эффекта
```

```
$("#element").slideToggle(1000);
```

2. Настраиваемые анимации:

JAVASCRIPT

```
// Анимация CSS-свойств
```

```
$("#element").animate({  
    opacity: 0.5,  
    width: "70%",  
    height: "300px",  
    marginLeft: "30px",  
    fontSize: "2em"  
}, 1000);
```

```
// Последовательная анимация
```

```
$("#element")  
    .animate({ opacity: 0.5 }, 500)  
    .animate({ width: "70%" }, 500)  
    .animate({ height: "300px" }, 500);
```

// Анимация с использованием очереди и функции обратного вызова

```
$("#element").animate({ width: "80%" }, 1000, function() {
```

```
    // Выполняется после завершения анимации
```

```
    $(this).animate({ height: "200px" }, 500);
```

```
});
```

// Относительная анимация

```
$("#element").animate({
```

```
    width: "+=50px", // увеличить ширину на 50px
```

```
    height: "-=20px" // уменьшить высоту на 20px
```

```
}, 500);
```

// Анимация с использованием эффекта ускорения/замедления

```
$("#element").animate({ width: "80%" }, {
```

```
    duration: 1000,
```

```
    easing: "swing" // или "linear"
```

```
});
```

3. Управление анимациями:

JAVASCRIPT

// Остановка текущей анимации

```
$("#element").stop();
```

// Остановка всех анимаций в очереди

```
$("#element").stop(true);
```

// Остановка текущей анимации и переход к конечному состоянию

```
$("#element").stop(false, true);
```

```
// Остановка всех анимаций и переход к конечному состоянию
$("#element").stop(true, true);
```

```
// Задержка перед следующей анимацией
$("#element")
    .slideUp(500)
    .delay(1000) // задержка 1 секунда
    .slideDown(500);
```

Пример создания сложной анимации:

JAVASCRIPT

```
$(function() {
    $("#animateButton").click(function() {
        // Последовательная анимация с различными эффектами
        $("#box")
            .animate({ opacity: 0.8 }, 500)
            .animate({ width: "300px", height: "300px" }, 800)
            .animate({ borderRadius: "50%" }, 500)
            .animate({ backgroundColor: "#ff9900" }, 800)
            .animate(
                { marginLeft: "200px" },
                {
                    duration: 1000,
                    complete: function() {
                        // Функция, выполняемая после завершения анимации
                        $(this).text("Анимация завершена");
                    }
                }
            );
    });
});
```

```
$("#resetButton").click(function() {  
    // Остановка всех анимаций и возврат к исходному состоянию  
    $("#box").stop(true, false).removeAttr("style").text("");  
});  
});
```

Адаптация jQuery для современных веб-приложений

Несмотря на появление новых фреймворков, jQuery остается актуальной библиотекой для многих проектов. Рассмотрим способы адаптации jQuery 2.0 для использования в современных веб-приложениях.

Интеграция jQuery с современными фреймворками:

1. jQuery и React:

- Использование jQuery-плагинов в React-компонентах:

JAVASCRIPT

```
import React, { useEffect, useRef } from 'react';  
import $ from 'jquery';  
import 'some-jquery-plugin';
```

```
function MyComponent() {  
    const elementRef = useRef(null);  
  
    useEffect(() => {  
        // Инициализация jQuery-плагина  
        $(elementRef.current).somePlugin({  
            // настройки  
        });  
  
        // Очистка при размонтировании  
        return () => {
```

```

        $(elementRef.current).somePlugin('destroy');
    };
}, []);

return <div ref={elementRef}></div>;
}

```

2. jQuery и Vue.js:

- Использование jQuery в директивах Vue:

JAVASCRIPT

```

Vue.directive('jquery-plugin', {
  inserted: function (el, binding) {
    $(el).plugin(binding.value);
  },
  unbind: function(el) {
    $(el).plugin('destroy');
  }
});

```

3. jQuery и Angular:

- Интеграция через сервисы Angular:

JAVASCRIPT

```

@Injectable({
  providedIn: 'root'
})
export class JQueryService {
  initPlugin(element: ElementRef, options: any) {
    $(element.nativeElement).plugin(options);
  }
}

```

Оптимизация jQuery для мобильных устройств:

1. Использование облегченных альтернатив:

- Zepto.js — легковесная альтернатива с аналогичным API
- jQuery slim — облегченная версия jQuery без AJAX и анимаций

2. Оптимизация обработки событий для мобильных устройств:

- Использование делегирования событий
- Обработка специфичных мобильных событий (touchstart, touchmove, touchend)
- Использование jQuery Mobile для мобильных интерфейсов

3. Адаптация для PWA (Progressive Web Applications):

- Использование Service Workers для кэширования jQuery и зависимостей
- Оптимизация работы offline
- Минимизация влияния jQuery на время загрузки приложения

Итог по адаптации jQuery для современных веб-приложений:

1. jQuery 2.0 может успешно интегрироваться с современными фреймворками для решения специфических задач
2. Модульный подход и современные методы JavaScript делают jQuery более гибкой и соответствующей современным практикам разработки
3. Оптимизация для мобильных устройств позволяет использовать jQuery в мобильно-ориентированных приложениях
4. Правильное использование jQuery в современных проектах требует понимания её сильных и слабых сторон для выбора оптимальных сценариев применения

Практические примеры использования jQuery 2.0

Разработка интерактивной формы

В данном разделе будет разработана интерактивная форма регистрации с использованием jQuery 2.0, включающая валидацию полей, динамическое обновление и интерактивные элементы.

HTML-структура формы:

HTML

<!DOCTYPE html>


```
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Регистрационная форма</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <h1>Регистрация пользователя</h1>

    <form id="registrationForm">
      <div class="steps-indicator">
        <div class="step active" data-step="1">Персональные данные</div>
        <div class="step" data-step="2">Контакты</div>
        <div class="step" data-step="3">Дополнительно</div>
      </div>

      <div class="form-step" id="step1">
        <div class="form-group">
          <label for="firstName">Имя *</label>
          <input type="text" id="firstName" name="firstName" required>
          <div class="error-message"></div>
        </div>

        <div class="form-group">
          <label for="lastName">Фамилия *</label>
          <input type="text" id="lastName" name="lastName" required>
          <div class="error-message"></div>
        </div>
      </div>
    </form>
  </div>
</body>
</html>
```

```
</div>
```

```
<div class="form-group">
```

```
  <label for="birthDate">Дата рождения *</label>
```

```
  <input type="date" id="birthDate" name="birthDate" required>
```

```
  <div class="error-message"></div>
```

```
</div>
```

```
<div class="buttons">
```

```
  <button type="button" class="next-step">Далее</button>
```

```
</div>
```

```
</div>
```

```
<div class="form-step" id="step2" style="display: none;">
```

```
  <div class="form-group">
```

```
    <label for="email">Email *</label>
```

```
    <input type="email" id="email" name="email" required>
```

```
    <div class="error-message"></div>
```

```
  </div>
```

```
<div class="form-group">
```

```
  <label for="phone">Телефон</label>
```

```
  <input type="tel" id="phone" name="phone">
```

```
  <div class="error-message"></div>
```

```
</div>
```

```
<div class="form-group">
```

```
  <label for="address">Адрес</label>
```

```
  <textarea id="address" name="address" rows="3"></textarea>
```

</div>

<div class="buttons">

<button type="button" class="prev-step">Назад</button>

<button type="button" class="next-step">Далее</button>

</div>

</div>

<div class="form-step" id="step3" style="display: none;">

<div class="form-group">

<label>Интересы</label>

<div class="checkbox-group">

<label>

<input type="checkbox" name="interests" value="sports"> Спорт

</label>

<label>

<input type="checkbox" name="interests" value="music">

Музыка

</label>

<label>

<input type="checkbox" name="interests" value="art">

Искусство

</label>

<label>

<input type="checkbox" name="interests" value="tech">

Технологии

</label>

</div>

</div>

```

    <div class="form-group">
      <label for="subscribe">
        <input type="checkbox" id="subscribe" name="subscribe">
        Подписаться на рассылку
      </label>
    </div>

    <div class="buttons">
      <button type="button" class="prev-step">Назад</button>
      <button type="submit" class="submit-
btn">Зарегистрироваться</button>
    </div>
  </div>
</form>

<div id="formSummary" style="display: none;">
  <h2>Ваши данные</h2>
  <div id="summaryContent"></div>
  <button id="editForm">Редактировать</button>
</div>
</div>

<script src="https://code.jquery.com/jquery-2.0.3.min.js"></script>
<script src="form.js"></script>
</body>
</html>

```

CSS-стили для формы:

CSS

```
* {
```

```
    box-sizing: border-box;
}
```

```
body {
    font-family: Arial, sans-serif;
    line-height: 1.6;
    margin: 0;
    padding: 20px;
    background-color: #f5f5f5;
}
```

```
.container {
    max-width: 800px;
    margin: 0 auto;
    background-color: white;
    padding: 30px;
    border-radius: 5px;
    box-shadow: 0 2px 10px rgba(0, 0, 0, 0.1);
}
```

```
h1 {
    text-align: center;
    margin-bottom: 30px;
    color: #333;
}
```

```
.steps-indicator {
    display: flex;
    justify-content: space-between;
```

```
margin-bottom: 30px;  
position: relative;  
}
```

```
.steps-indicator::before {  
  content: " ";  
  position: absolute;  
  top: 50%;  
  left: 0;  
  right: 0;  
  height: 2px;  
  background-color: #ddd;  
  z-index: 1;  
}
```

```
.step {  
  position: relative;  
  z-index: 2;  
  background-color: white;  
  padding: 10px 15px;  
  border: 2px solid #ddd;  
  border-radius: 30px;  
  font-size: 14px;  
  color: #777;  
  transition: all 0.3s ease;  
}
```

```
.step.active {  
  border-color: #4CAF50;
```

```
    color: #4CAF50;
}
```

```
.step.completed {
    border-color: #4CAF50;
    background-color: #4CAF50;
    color: white;
}
```

```
.form-group {
    margin-bottom: 20px;
}
```

```
label {
    display: block;
    margin-bottom: 5px;
    font-weight: bold;
    color: #555;
}
```

```
input[type="text"],
input[type="email"],
input[type="tel"],
input[type="date"],
textarea {
    width: 100%;
    padding: 10px;
    border: 1px solid #ddd;
    border-radius: 4px;
```

```
    font-size: 16px;
}
```

```
input[type="text"]:focus,
input[type="email"]:focus,
input[type="tel"]:focus,
input[type="date"]:focus,
textarea:focus {
    outline: none;
    border-color: #4CAF50;
    box-shadow: 0 0 5px rgba(76, 175, 80, 0.3);
}
```

```
.checkbox-group {
    display: grid;
    grid-template-columns: repeat(2, 1fr);
    gap: 10px;
}
```

```
.error-message {
    color: #f44336;
    font-size: 14px;
    margin-top: 5px;
    min-height: 20px;
}
```

```
.buttons {
    display: flex;
    justify-content: space-between;
```



```
    margin-top: 30px;
}
```

```
button {
    padding: 10px 20px;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: 16px;
    transition: background-color 0.3s ease;
}
```

```
.next-step, .submit-btn {
    background-color: #4CAF50;
    color: white;
}
```

```
.next-step:hover, .submit-btn:hover {
    background-color: #45a049;
}
```

```
.prev-step {
    background-color: #f1f1f1;
    color: #333;
}
```

```
.prev-step:hover {
    background-color: #e0e0e0;
}
```

```
input.error {  
    border-color: #f44336;  
}
```

```
#formSummary {  
    margin-top: 30px;  
    padding: 20px;  
    border: 1px solid #ddd;  
    border-radius: 4px;  
    background-color: #f9f9f9;  
}
```

```
#summaryContent {  
    margin-bottom: 20px;  
}
```

```
#summaryContent dl {  
    display: grid;  
    grid-template-columns: 1fr 2fr;  
    gap: 10px;  
}
```

```
#summaryContent dt {  
    font-weight: bold;  
    color: #555;  
}
```

```
.highlight {
```

```
    animation: highlight 1s ease;
}
```

```
@keyframes highlight {
    0% { background-color: #fff9c4; }
    100% { background-color: transparent; }
}
```

JavaScript с использованием jQuery 2.0:

JAVASCRIPT

```
$(function() {
    // Объект для хранения данных формы
    var formData = {};

    // Валидация полей в зависимости от шага
    function validateStep(step) {
        var isValid = true;

        // Проверка полей первого шага
        if (step === 1) {
            // Проверка имени
            var firstName = $("#firstName").val().trim();
            if (firstName === "") {
                showError($("#firstName"), "Пожалуйста, укажите имя");
                isValid = false;
            } else {
                clearError($("#firstName"));
                formData.firstName = firstName;
            }
        }
    }
}
```

```
// Проверка фамилии
var lastName = $("#lastName").val().trim();
if (lastName === "") {
    showError($("#lastName"), "Пожалуйста, укажите фамилию");
    isValid = false;
} else {
    clearError($("#lastName"));
    formData.lastName = lastName;
}

// Проверка даты рождения
var birthDate = $("#birthDate").val();
if (birthDate === "") {
    showError($("#birthDate"), "Пожалуйста, укажите дату рождения");
    isValid = false;
} else {
    var today = new Date();
    var selectedDate = new Date(birthDate);

    if (selectedDate > today) {
        showError($("#birthDate"), "Дата рождения не может быть в будущем");
        isValid = false;
    } else {
        var age = today.getFullYear() - selectedDate.getFullYear();
        if (age < 18) {
            showError($("#birthDate"), "Вам должно быть не менее 18 лет");
            isValid = false;
        } else {
```

```
        clearError($("#birthDate"));
        formData.birthDate = birthDate;
    }
}
}
}
```

// Проверка полей второго шага

```
if (step === 2) {
```

```
    // Проверка email
```

```
    var email = ($("#email").val()).trim();
```

```
    var emailRegex = /^[\\w-]+(\\.\\w-+)*@([\\w-]+\\.)+[a-zA-Z]{2,7}$/;
```

```
    if (email === "") {
```

```
        showError($("#email"), "Пожалуйста, укажите email");
```

```
        isValid = false;
```

```
    } else if (!emailRegex.test(email)) {
```

```
        showError($("#email"), "Пожалуйста, укажите корректный email");
```

```
        isValid = false;
```

```
    } else {
```

```
        clearError($("#email"));
```

```
        formData.email = email;
```

```
    }
```

// Проверка телефона (необязательное поле)

```
var phone = ($("#phone").val()).trim();
```

```
var phoneRegex = /^[+]?\\d{10,15}$/;
```

```
if (phone !== "" && !phoneRegex.test(phone)) {
```

```
        showError($("#phone"), "Пожалуйста, укажите корректный номер  
телефона");
```

```
        isValid = false;
```

```
    } else {
```

```
        clearError($("#phone"));
```

```
        formData.phone = phone;
```

```
    }
```

```
    // Сохранение адреса
```

```
    formData.address = ($("#address").val()).trim();
```

```
    }
```

```
    return isValid;
```

```
    }
```

```
// Отображение ошибки для поля
```

```
function showError($element, message) {
```

```
    $element.addClass("error");
```

```
    $element.next(".error-message").text(message);
```

```
}
```

```
// Очистка ошибки для поля
```

```
function clearError($element) {
```

```
    $element.removeClass("error");
```

```
    $element.next(".error-message").text("");
```

```
}
```

```
// Переключение на следующий шаг
```

```
$("#next-step").click(function() {
```

```
var currentStep = $(this).closest(".form-step").attr("id").replace("step", "");
currentStep = parseInt(currentStep);

if (validateStep(currentStep)) {
    // Скрываем текущий шаг
    $("#step" + currentStep).hide();

    // Показываем следующий шаг
    $("#step" + (currentStep + 1)).fadeIn(300);

    // Обновляем индикатор шагов
    $(".step[data-step=" + currentStep + "]").addClass("completed");
    $(".step[data-step=" + (currentStep + 1) + "]").addClass("active");
}
});

// Переключение на предыдущий шаг
$(".prev-step").click(function() {
    var currentStep = $(this).closest(".form-step").attr("id").replace("step", "");
    currentStep = parseInt(currentStep);

    // Скрываем текущий шаг
    $("#step" + currentStep).hide();

    // Показываем предыдущий шаг
    $("#step" + (currentStep - 1)).fadeIn(300);

    // Обновляем индикатор шагов
    $(".step[data-step=" + currentStep + "]").removeClass("active");
```

```
    $(".step[data-step=" + (currentStep - 1) +  
    ""].removeClass("completed").addClass("active");  
});
```

// Обработка отправки формы

```
$("#registrationForm").submit(function(e) {  
    e.preventDefault();
```

// Сохраняем данные с последнего шага

```
formData.interests = [];
```

```
$("#input[name='interests']:checked").each(function() {  
    formData.interests.push($(this).val());  
});
```

```
formData.subscribe = $("#subscribe").is(":checked");
```

// Рендерим сводку данных

```
renderSummary();
```

// Скрываем форму и показываем сводку

```
$("#registrationForm").hide();
```

```
$("#formSummary").fadeIn(300);
```

```
});
```

// Отображение сводки данных формы

```
function renderSummary() {
```

```
    var html = '<dl>';
```

```
    html += '<dt>Имя:</dt><dd>' + (formData.firstName || "") + '</dd>';
```

```
    html += '<dt>Фамилия:</dt><dd>' + (formData.lastName || "") + '</dd>';
```



```
html += '<dt>Дата рождения:</dt><dd>' + (formData.birthDate ?  
formatDate(formData.birthDate) : '') + '</dd>';
```

```
html += '<dt>Email:</dt><dd>' + (formData.email || '') + '</dd>';
```

```
if (formData.phone) {  
    html += '<dt>Телефон:</dt><dd>' + formData.phone + '</dd>';  
}
```

```
if (formData.address) {  
    html += '<dt>Адрес:</dt><dd>' + formData.address + '</dd>';  
}
```

```
if (formData.interests && formData.interests.length > 0) {  
    var interestsMap = {  
        'sports': 'Спорт',  
        'music': 'Музыка',  
        'art': 'Искусство',  
        'tech': 'Технологии'  
    };  
};
```

```
var interestsList = formData.interests.map(function(interest) {  
    return interestsMap[interest] || interest;  
}).join(', ');
```

```
html += '<dt>Интересы:</dt><dd>' + interestsList + '</dd>';  
}
```

```
html += '<dt>Подписка на рассылку:</dt><dd>' + (formData.subscribe ? 'Да'  
: 'Нет') + '</dd>';
```

```
html += '</dl>';
```

```
$("#summaryContent").html(html);  
}
```

```
// Форматирование даты
```

```
function formatDate(dateString) {  
    var date = new Date(dateString);  
    return date.toLocaleDateString('ru-RU');  
}
```

```
// Возврат к редактированию формы
```

```
$("#editForm").click(function() {  
    $("#formSummary").hide();  
    $("#registrationForm").fadeIn(300);
```

```
// Возвращаемся на первый шаг
```

```
$(".form-step").hide();  
$("#step1").show();
```

```
// Сбрасываем индикатор шагов
```

```
$(".step").removeClass("active completed");  
$(".step[data-step='1']").addClass("active");  
});
```

```
// Валидация полей формы при изменении значений
```

```
$("input, textarea").on("input change", function() {  
    var $this = $(this);
```

```
if ($this.hasClass("error") && $this.val().trim() !== "") {  
    // При вводе текста в поле с ошибкой сбрасываем ошибку  
    clearError($this);  
}  
});
```

```
// Форматирование телефонного номера  
$("#phone").on("input", function() {  
    var input = $(this).val().replace(/\D/g, ""); // Удаляем все нецифровые  
символы
```

```
if (input.length > 0) {  
    // Если первая цифра не '+', добавляем '+'  
    if (input.charAt(0) !== '+') {  
        input = '+' + input;  
    }  
}
```

```
// Ограничиваем длину номера  
if (input.length > 15) {  
    input = input.substr(0, 15);  
}
```

```
$(this).val(input);  
}  
});
```

```
// Анимация для чекбоксов  
$(".checkbox-group input").change(function() {
```

```
var $label = $(this).closest("label");
$label.addClass("highlight");
setTimeout(function() {
    $label.removeClass("highlight");
}, 1000);
});
});
```