**jQuery** — [библиотека JavaScript](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_JavaScript), фокусирующаяся на взаимодействии [JavaScript](https://ru.wikipedia.org/wiki/JavaScript) и [HTML](https://ru.wikipedia.org/wiki/HTML). Помогает легко получать доступ к любому элементу [DOM](https://ru.wikipedia.org/wiki/Document_Object_Model), обращаться к атрибутам и содержимому элементов [DOM](https://ru.wikipedia.org/wiki/Document_Object_Model), манипулировать ими, предоставляет удобный [API](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B9) для работы с [AJAX](https://ru.wikipedia.org/wiki/AJAX).

## **Возможности**

* Движок кросс-браузерных CSS-селекторов [Sizzle](https://ru.wikipedia.org/w/index.php?title=Sizzle&action=edit&redlink=1)[[4]](https://ru.wikipedia.org/wiki/JQuery#cite_note-4), выделившийся в отдельный проект;
* Переход по дереву DOM, включая поддержку [XPath](https://ru.wikipedia.org/wiki/XPath) как плагина;
* События;
* Визуальные эффекты;
* [AJAX](https://ru.wikipedia.org/wiki/AJAX)-дополнения;
* JavaScript-[плагины](https://ru.wikipedia.org/wiki/%D0%9F%D0%BB%D0%B0%D0%B3%D0%B8%D0%BD).

Точно так же, как [CSS](https://ru.wikipedia.org/wiki/CSS) отделяет визуализацию от структуры HTML, JQuery отделяет поведение от структуры HTML. Например, вместо прямого указания на обработчик события нажатия кнопки управление передаётся JQuery, которая идентифицирует кнопки и затем преобразует его в обработчик события клика. Такое разделение поведения и структуры также называется принципом [ненавязчивого JavaScript](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BD%D0%B0%D0%B2%D1%8F%D0%B7%D1%87%D0%B8%D0%B2%D1%8B%D0%B9_JavaScript). jQuery, как правило, включается в веб-страницу как один внешний JavaScript-файл. Вся работа с jQuery ведётся с помощью функции **$**. Если на сайте применяются другие JavaScript библиотеки, где $ может использоваться для своих нужд, то можно использовать её синоним — **jQuery**. Работу с jQuery можно разделить на 2 типа:

* Получение jQuery-объекта с помощью функции $(). Например, передав в неё CSS-селектор, можно получить jQuery-объект всех элементов HTML, попадающих под критерий и далее работать с ними с помощью различных методов jQuery-объекта. В случае, если метод не должен возвращать какого-либо значения, он возвращает ссылку на jQuery объект, что позволяет вести цепочку вызовов методов согласно концепции [текучего интерфейса](https://ru.wikipedia.org/wiki/Fluent_interface).
* Вызов глобальных методов у объекта $, например, удобных итераторов по массиву.

Типичный пример манипуляции сразу несколькими узлами DOM заключается в вызове $ функции со строкой селектора [CSS](https://ru.wikipedia.org/wiki/CSS), что возвращает объект jQuery, содержащий некоторое количество элементов HTML-страницы. Эти элементы затем обрабатываются методами jQuery.

**$("div.test").add("p.quote").addClass("blue").slideDown("slow");**

Находит все элементы div с классом test, а также все элементы p с классом quote, и затем добавляет им всем класс blue и визуально плавно спускает вниз. Здесь методы add, addClass и slideDown возвращают ссылку на исходный объект $("div.test"), поэтому возможно вести такую цепочку. **$.ajax** и соответствующие функции позволяют использовать методы [AJAX](https://ru.wikipedia.org/wiki/AJAX).

**$.ajax({**

**type: "POST",**

**url: "some.php",**

**data: {name: 'John', location: 'Boston'},**

**success: function(msg){**

**alert( "Data Saved: " + msg );**

**}**

**});**

Настройки beforeSend, error, dataFilter, success и complete (их описание есть в предыдущем разделе) позволяют установить обработчики событий, которые происходят в определенные моменты выполнения каждого ajax-запроса.

***beforeSend*** происходит непосредственно перед отправкой запроса на сервер.

***error*** происходит в случае неудачного выполнения запроса.

***dataFilter*** происходит в момент прибытия данных с сервера. Позволяет обработать "сырые" данные, присланные сервером.

***success*** происходит в случае удачного завершения запроса.

***complete*** происходит в случае любого завершения запроса.