# События jQuery

# .ON()

- Устанавливает обработчики событий на выбранные элементы страницы. Имеет два варианта использования:

- .on(events, [selector], [data], handler)
- events — тип(ы) обрабатываемых событий. Например "click", "resize" и.т.д. Если необходимо привязать обработчик сразу на несколько типов событий, нужно перечислить их через пробел: "click resize ..."
- selector — селектор по которому будут фильтроваться элементы, лежащие внутри уже найденных. В итоге, обработчик будет срабатывать только в том случае, если событие «поднялось» от одного из отфильтрованных элементов.
- data — данные, передаваемые обработчику событий. В обработчике будут доступны в переменной event.data.
- handler — функция, которая будет установлена в качестве обработчика. Вместо функции, можно указать значение false, это будет эквивалентно установке такой функции: function(){return false;}.

- .on(events-map, [selector], [data])
- с помощью этого метода можно установить на выбранные элементы сразу несколько разных обработчиков событий, каждый из которых будет реагировать на свой тип события.

- events-map — объект, в котором нужно перечислить типы обрабатываемых событий и соответствующие им обработчики. Задается в формате {events-1:handler-1, events-2:handler-2, ...}, где events-i и handler-i соответствуют параметрам events и handler в первом варианте метода (описанном выше).

# ONE()

- Устанавливает обработчик события выбранным элементам страницы. Особенностью метода является то, что обработчик будет вызван не более одного раза, на каждом из элементов.
- .one(eventType, [eventData], handler(eventObject)):jQueryv:1.1
- eventType — тип обрабатываемого события. Например "click", "resize" и.т.д. (список всех событий см. ниже).
- eventData — данные, передаваемые обработчику событий. Они должны быть представлены в форме объекта, в формате: {fName1:value1, fName2:value2, ...}.
- handler(eventObject) — функция, которая будет установлена в качестве обработчика. При вызове она будет получать объект события eventObject.

# OFF()

- Удаляет с выбранных элементов страницы обработчики событий, установленные с помощью метода .on(). Имеет два варианта использования:

- .off([events], [selector], [handler]):jQueryv:1.7
- events — тип(ы) обрабатываемых событий. Например "click", "resize" и.т.д. Может быть указано сразу несколько типов, разделенных пробелами, а так же, могут быть указаны пространства имен.
- selector — селектор, указанный (если он был указан) при установке события в методе .on().
- handler — функция, которую необходимо удалить из обработчиков.

- .of(events-map, [selector]):jQueryv:1.7
- с помощью этого метода можно удалить с выбранных элементов сразу несколько разных обработчиков событий, установленных на разные типы событий.

- events-map — объект, в котором нужно перечислить типы событий и соответствующие им удаляемые обработчики. Задается в формате {events-1:handler-1, events-2:handler-2, ...}, где events-i и handler-i соответствуют параметрам events и handler в первом варианте метода (описанном выше).

# EVENT

- объект, содержащий данные о произошедшем событии. Передается во все обработчики событий, которые были установлены средствами библиотеки jQuery. Объект event немного отличается от стандартных объектов событий, которые получают обработчики, установленные обычными средствами javascript. Некоторые поля подвергаются изменению для обеспечения кроссбраузерности. Помимо этого, в event добавлены дополнительные поля и методы.

- Следующие поля гарантировано присутствуют в каждом объекте события (хотя некоторые из них могут иметь значение undefined):
- altKey, attrChange, attrName, bubbles, button, cancelable, charCode, clientX, clientY, ctrlKey, currentTarget, data, detail, eventPhase, fromElement, handler, keyCode, layerX, layerY, metaKey, namespace (только с jQuery-1.4.3 и старше), newValue, offsetX, offsetY, originalTarget, pageX, pageY, prevValue, relatedNode, relatedTarget, screenX, screenY, shiftKey, srcElement, target, toElement, view, wheelDelta, which

- event.currentTarget
- Содержит DOM-элемент, событие которого обрабатывается. Внутри обработчика, currentTarget всегда совпадает с переменной this. Однако, этот элемент может не являться источником события, поскольку оно могло быть передано от дочернего элемента, в результате "всплытия" события, вверх по иерархии DOM. Первоначальный источник события содержится в event.target.
- event.data
- Дополнительные данные, которые передаются обработчику при его установке.
- event.isDefaultPrevented()
- Определяет, вызывался ли метод event.preventDefault() на данном объекте событий.
- event.isPropagationStopped()
- Определяет, вызывался ли метод event.stopPropagation() на данном объекте событий.
- event.pageX, event.pageY
- Координаты курсора мыши относительно левого верхнего угла документа.
- event.relatedTarget
- Не текущий DOM-элемент, участвующий в событии. Для события mouseout будет содержать элемент, куда переместился курсор, а для mousein элемент, откуда курсор пришел.
- event.result
- Содержит значение, которое возвратил предыдущий обработчик этого события. Если предыдущего обработчика нет, возвратит undefined.
- event.timeStamp
- Содержит время, когда было произведено событие. Время представлено количеством секунд, прошедших с 1.01.1970.
- event.which
- При возникновении события, связанного с нажатием клавиш клавиатуры или кнопок на странице, это поле будет содержать информацию о нажатых клавишах или кнопках. В отличии от стандартных event.keyCode и event.charCode, содержимое в event.which кроссбраузерное.

# .TRIGGER()

- Вызывает событие у выбранных элементов, что приводит к запуску обработчиков этого события. Метод имеет два варианта использования:

- .trigger(eventType, [extraParameters]) eventType — тип вызываемого события. Например "click", "resize" и. т.д. extraParameters — массив дополнительных данных, передаваемых в обработчик.

- .trigger(eventObject)
- Позволяет повторно запустить событие из обработчика текущего события. Может быть полезно при необходимости вызывать событие через определенные периоды времени.
- eventObject — объект обрабатываемого события, который был передан в обработчик.

# .TRIGGERHANDLER()

- Вызывает выполнение обработчиков заданного события у выбранных элементов. Само событие, при этом, не происходит. Метод имеет один вариант использования:

- .triggerHandler(eventType, [extraParameters]):jQueryv:1.2
- eventType — тип вызываемого события. Например "click", "resize" и.т.д. подробнее о типах
- extraParameters — массив дополнительных данных, передаваемых в обработчик.

- Метод .triggerHandler() работает почти так же, как метод .trigger(), с несколькими отличиями:
- Метод .triggerHandler() не вызывает стандартных событий (к примеру отправку формы).
- В то время как .trigger() действует на все подходящие элементы в объекте jQuery, .triggerHandler() срабатывает только для первого найденного элемента.
- Метод .triggerHandler() возвращает то, что и метод обработчик, который сработает в результате “выстрела” события. Если ни один из обработчиков не будет вызван, то будет возвращён undefined

# РАЗЛИЧИЯ

- $( "#old" ).click(function() {
- $( "input" ).trigger( "focus" );
- });
- $( "#new" ).click(function() {
- $( "input" ).triggerHandler( "focus" );
- });
- $( "input" ).focus(function() {
- $( "Focused!" ).appendTo( "body" )
- });