**rAlert(…)** – мигалка

**confirm("…")** – окно с запросом на да или нет возвращает булевый ответ

**prompt("… , текс по умолчанию")** – возвращает введенный текс

**Console.log/.warn** /**error**/**info** – вывод в консоль чего-то

**Eval(“…”)** - забирает строковое выражение как обыкновенное

**Void** – убирает возвращаемое значение

**Math**

**Math.pi(…)** – число пи

**Math.abc(…)** – по модулю

**Math.asin(…)/atan/acos/actan** – арксинус косину тангенс котангенс

**Math.atan2(… , …)** – соотношение первого числа ко второму

**Math.ceil(…)** – округление до большего числа

**Math.floor(…)** – округление к меньшему

**Math.cos(…)/sin/tan** – синус косинус тангенс

**Math.exp(…)** – число “е” в степени

**Math.log(…)** - натуральный логарифм

**Math.max(……)** – вывод максимального

**Math.min(……)** – вывод минимального

**Math.pow(число, степень)** – подводит число в степень

**Math.random()** – рандомное число от 0 до 1

**Math.round(…)** – округление до ближайшего целого

**Math.sqrt(…)** – квадратный корень

**Number.NEGATIVE\_INFINITY** – минус бесконечность

**Number.POSITIVE\_INFINITY** – плюс бесконечность

**... .toString()** – число в строковый тип

**... .toLocaleString()** – число в строковый тип относительно характеристик пк

**… .toFixed(1-20)** – кол-во цифр после точки

**… .toPrecision(1-20)** – точность возвращаемого числа

**isNaN(…)** – возвращает булевое значение является ли переменная NaN

**isFinite(…)** - возвращает булевое значение является ли переменная бесконечностью

**Boolean(…)** – преобразует переменную в булевый тип

**Number(…)** – преобразует переменную в числовой тип

**String(…)** – преобразует переменную в строковый тип

**parseInt (…)** – преобразует переменную в целочисленный тип

**parseFloat (…)** – преобразует переменную в дробный тип

**\’’** – показывает, что это не закрывающиеся скобки

**Functions**

**… .length** – возвращает длину строки

**… .toUpperCase()** – возвращает ту же строку заглавными буквами

**… .toLowerCase()** – возвращает ту же строку мелкими буквами

**… .split(“…“)** – разбивает строку на части, разделенные тем что вставили

**… .indexOf(“…”)** – выводит позицию, где начинается слово или -1, если слова нет

**… .onfocus** – нажата переменная

**… .onblur** – не нажатая переменная

**... .checked** – нажатый checkbox

**… .style. ... = “…”** – выставление CSS свойства

**.className = "…”** – менять имя класса для смены функций в CSS

**... = function (…) {…}** – в функцию можно передать любую букву и потом использовать ее в самой функции как объект элемента, функции ниже выполняются только с этой буквой

**… .altKey** – возвращает булевое значение была ли нажата кнопка Alt

**… .ctrlKey** – возвращает булевое значение была ли нажата кнопка Ctrl

**… .clientX** – расположение мышки по оси Х

**… .clientY** – расположение мышки по оси Y

**… .keyCode** – какая кнопка на клаве была нажата

**… .disabled="true or false"** – отключение и подключение частей форм

**Date**

**… = new Date()** – воз вращает текущую дату

Все следующие метолы зависят от объекта, созданного выше

**… .getMonth()** – возвращает 0-11

**… .getYear()** – год минус 1900

**… .getFullYear()** – год

**… .getDate()** – 1-31 день месяца

**… .getDay()** – 0-6 день 0-вс

**… .getHours()** – 0-23

**… .getTime** – возвращает миллисекунды с указанного времени

**+ тоже самое с set** – установить дату в переменную

**Date.now()** – кол-во миллисекунд от 1970

**… .toLocaleString()** – дата и время в регионе сервера

**… . toLocaleDateString()** – дата в регионе

**… . toLocaleTimeString()** – время в регионе

**Document**

**Document.getElementById(“…”)** – возвращает все элементы по ID

**Document.getElementByTagName(“…”)** – возвращает все элементы по тегу

**document.getElementsByName("…")** – возвращает элемент по тегу name

**document.getElementsByClassName("p")** – возвращает элементы но тегу class

**document.querySelector(". … или #... или тег")** – поиск как в CSS ( возвращает первый элемент )

**document.querySelectorAll(". … или #... или тег")** – поиск как в CSS ( возвращает все элементы )

**var …= document.createDocumentFragment()** – создает объединение узлов под одной шапкой

**… .innerHTML = "…"** – вставить/заменить элемент

**… .outerHTML** - возвращает HTML элемент с тегами, которые образуют его

**… .insertAdjacentHTML("способ","элементы с тегами вокруг")** – вставка текстовых элементов в узле ( способы: **Beforebegin** – перед выбранным элементом, **afterbegin** – после выбранного элемента, **beforeend** – перед закрывающим тегом элемента, **afterend** – за тегом выбранного элемента )

**… .setAttribute(«название атрибута», «значение»)** – установка значений атрибутам

**... .nodeValue** – текстовый узел элемента

**… .getAttribute("…")** - получать доступ к атрибутам из html ( например ссылка href )

**… .setAttribute("атрибут", "значение")** – изменение значения атрибутов

**… . value = “…”** – работа с элементом

**… attributes[“…”]. …** - доступ к атрибутам и их значениям ( только для чтения к примеру **value** )

**… .hasAttribute("…")** – проверка на наличие атрибута

**… .removeAttribute("…")** – удаление атрибута

**… .createTextNode("…")** – создать текстовый документ

**… .createElement("div")** – создать узел в теге …

**… .appendChild(…)** – в зависимости от текущего узла создать дочерний узел и вставить в него элемент в конец

**Родительский узел.insertBefore(что вставить, перед чем)** – вставка узла ( метод указывается у родительского элемента и первым принимает новый узел вторым – узел перед которым он будет стоять )

**Родительский узел.removeChild(…)** – удаление дочернего узла ( в переменную передаётся узел который нужно удалить )

**… .replaceChild(новый узел, стары узел)** – замена узлов

**document.forms** – обращение к формам

**… .submit()** – подтвердить выполнение формы

**… .reset()** – сброс формы

**document.images** – обращение ко всем картинкам

**… .documentElement** – узел html самый главный

**... .body** – тег body

**… .title** – тег title

**… .head** – тег head

**… .anchors** – все якоря

**… .applets** – все джава аплеты

**… .forms** – все формы

**… .images** – все картинки

**… .links** – все ссылки

**... .parentNode** – родительский узел

**… .childNodes** – возвращает дочерние элементы

**… .lastChild** – последний дочерний элемент

**… .firstChild** – первый дочерний элемент

**… .nextSibling** – следующий братский элемент

**… .previousSibling**– предведущий братский элемент

**… .previousElementSiblingSibling -** предведущий братский узел

**… .nextElementSibling** – следующий братский узел

**… .lastElementChild** – последний дочерний узел

**… .firstElementChild** – первый дочерний узел

**... .nodeType** – тип узла ( для типов element = 1, attribute = 2, text = 3, comments = 8, document = 9 )

**… .nodeName** – возвращает название тега

**… .childElementCount** – количество дочерних элементов типа element ( **children.length** тоже самое )

**… .src** – адрес файла

**... .innerText** – текстовый элемент узла

**Window**

**window.onload = function () {}** – проверка все ли элементы загружены

**var … = window.open("…", “имя”, “heigtht = … , width = … , menubar = 1 или 0, toolbar = 1 ил 0, scrollbars = 1 или 0 ”)** – открывает новую страницу в браузере, имя нужно чтоб в одном окне открыть несколько сайтов по очереди ( после у объекта которому мы присвоили новое окно можно вызвать функцию **… .close();** - что закроет окно, как и все остальные функции window )

через переменную, которой присвоили новое окно можно вызывать функции другой страницы

если одна страница была открыта с помощью другой, то через объект **opener** можно обращаться к глобальным переменным родительской страницы

**... .focus()** – делает окно в фокусе через другой скрипт

**… .blur()** – снимает фокус

**… .print(…)** – принтер

**… .scrollTo(по оси х, по оси у)** – прокрутка стр.

**… .moveTo(по оси х, по оси у)** – перемещение нового окна браузера относительно родительского по координатам

**… .moveBy(по оси х, по оси у)** - многоразовое перемещение браузера относительно родительского

на такие координаты

**… .resizeTo(… , …)** – смена размеров окна

**… .resizeBy(… , …)** – изменение размера на

**… .scrollby(… , …)** – прокрутка на такое кол-во пикселей

**Parent. …** - родительская переменная или функция

**Frames. …** - дочерняя переменная или функция

**Top. …** - элемент или функция самой базовой стр.

**Location**

**Location** – возвращает адрес

**Location.hash** – возвращает хеш сайта ( в адресной строке после # )

**Location.hostname** – имя хоста ( начало адреса )

**Location.href** – полная адресная строка

**Location.pathname** – возвращает путь к странице без домена и параметров

**Location.protocol** – возвращает протокол ( file: или http: )

**Location.search** – возвращает параметры адреса

**Location.assign(“…”)** – перейти на адрес…

**location.reload()** – перезагрузка страницы

**location.replace(“…”)** – замена текущего адреса другим ( с этим методом в истории браузера память про посещении вашей страницы будет заменена памятью про новую страницу )

**history**

**history.length** – возвращает количество переходов на этом окне

**history.back()** – открывает прошлую страницу

**history.forward()** следующая страница

**history.go(…)** – переход на страницу из истории, которая находиться через несколько шагов ( можно передавать числа + и - , это простая итерация по истории браузера )

**navigator (clientInformation )**

**navigator.userAgent** – возвращает полную информацию о браузере и операционной системе ( можно использовать регулярные выражения чтоб получить версию браузера )

**navigator.platform** – возвращает тип операционной системы

**navigator.javaEnabled()** – возвращает может ли браузер сам читать Джава скрипт ( true false )

**navigator.geolocation** – расположение пользователя ( создание объекта )

**navigator.cookieEnabled** – включены ли у пользователя куки

**navigator.onLine** – возвращает в сети браузер или нет

**screen**

**screen.width** – возвращает ширину экрана

**screen.height** – возвращает высоту экрана

**screen.availWidth** – доступная ширина

**screen.availHeight** – доступная высота

**Таймеры**

**setTimeout( назв. функции, милисеык);** - таймер одноразовый при рекурсии вызывать функцию без дужек

**setInterval( назв. функции, милисек);** - цикличный таймер при рекурсии вызывать функцию без дужек

**clearInterval( переменная, которая содержит интервал)** – удаление интервала

**Array**

**For(var … in …) {…}** – создает цикл для массива и автоматом записывает его элементы в переменную

**… .join()** – соединяет массивы

**… .reverse()** – меняет порядок массива

**… .sort( function (a, b) { if (a < b ) {return -1;} if (a > b ) {return 1;} return 0; } );** - функция сортировки

( **return -1** – сначала **а** потом **b**, **return 1** – сначала **b** потом **а**, **return 0** – не менять

**… .concat(…)** – добавляет в массив элементы

**… .slice(… , …)** – вставить кусок по индексам

**… .splice(… , … , ……. )** – вставка и удаление элементов, первые два значения отвечают за позицию вставки и удаления, остальное – новые элементы

**… .push(…)** – вставляет элементы в конец

**… .pop()** – удаляет последний элемент

**… .unshift(…)** – вставляет элементы в начало

**... .shift()** – удаляет первый элемент

**Delete …** - удаляет ячейку

**Array.isArray(…)** – проверяет является ли переменная массивом

**String**

**… .indexOf(“…”)** – возвращает первый индекс начала строки, которую искали

**… .lastIndexOf(“…”)** – возвращает последний индекс начала строки, которую искали

**… .split(“…”)** – разбивает строку на части и возвращает массив, разделение элементом в душках

**… . replace(“что заменить”,”на что заменить”)** – замена части строки

**… . substr(начальный индекс , длина)** – возвращает элемент по поиску по индексам

**… . substring(начальный индекс , конечный индекс)** – возвращает элемент выделенный индексами

**… .toUpperCase()** – возвращает переменную но с заглавными буквами

**… .toLowerCase()** – возвращает переменную но с мелкими буквами

**… .search(регулярное выражение)** – поиск по регулярным выражениям возвращает индекс ( если совпадений много вернётся первый элемент)

**… .match(регулярное выражение)** – поиск по регулярным выражениям возвращает элементыre0 ( если совпадений много вернётся массив)

**… .bold()** – жирный шрифт

**… .fontcolor(“…”)** – цвет

**… .fontsize(…)** – размер

**… .link(“куда”)** – ссылка

**… .localeCompare(...)** – сравнивает какая переменная больше если основная строка меньше вернется

-1, одинаковые 0, больше 1

**Functions**

**Arguments** – массив элементов который используется в функции в которой нет принимаемых параметров

**Objects**

**Var … = new Object();** - создание объекта

**Var … = { … : … , … : … итд };** - создание объекта

**Если метода или функции не существует ее можно создать во все объекта**

**…["…”]** – получение свойств через переменные объектов ( можно использовать в цикле for in без дужек )

**Var … = Object.create(…)** – наследование

**Delete …**  - удаление свойств объектов

**… .hasOwnProperty("…")** – проверка есть ли такое свойство у объекта ( при наследовании элемента все равно выдает false )

**“…” in …** - проверка есть ли такое свойство у объекта ( при наследовании все работает )

**… .propertyIsEnumerable("…")** – проверка есть ли такое свойство у объекта ( при наследовании элемента все равно выдает false )

**Get название() {…}** – метод который возвращает работу с элементами, не принимает параметров

**Set названиие(…) {…}** – устанавливает значение переменных ( создаться один метод и для него пишутся set и get )

**… .имя = function имя(…) {…};** - создание методов в объекте

**… : function имя(…) {…},** - создание метода при объявлении

**Function имя(…) {…} + var … = new имя** – создание базового метода при создании объекта ( в функции идет принятие и обработка элементов через оператор ***this чтоб задать имя нужно вписать его в […]***, а потом создаться переменная и ей присваивается функция с оператором **new** )

**События**

**Документ:**

**Window.onload** – действие после полной прогрузки сайта

**window.onbeforeunload** – перед уходом со страници ( **return confirm(“…”)** – задать вопрос перед уходом )

**window.onfocus** – в фокусе

**window.onblur** – при снятии фокуса

**window.onresize** – при изменении размера

**window.onscroll** – при прокрутке окна

**Форми:**

**… .onsubmit** – при подтверждении формы

**… .onreset** – при обновлении формы

**… .onchange** – при изменении элемента форм

**… .onfocus** – в фокусе

**… .onblur** – при снятии фокуса

**… .oninput** – при вводе текста в форму ( каждое нажатие )

**Мышка:**

**… .onclick** – при щелчке мыши

**… .onmousemove** – при наведении мышки на объект

**… .onmousedown** – при нажатии на мышку, наведенную на объект ( не отпуская )

**… .onmouseup** – отпуская нажатую мышку, наведенную на объект

**… .oncontextmenu** – нажав правой кнопкой мыши ( **return false** чтоб не выскочило окошко )

**... .ondbclick** – двойное нажатие мыши

**… .onmouseup** - при наведении мышки на объект

**… .onmouseout** – при убирании мышки с объекта

**Клава:**

**… .onkeypress** – при нажатии любой клавиши

**… .onkeydown** – нажав кнопку

**… .onkeyup** - отпустив ранее нажатую кнопку

**Присвоеие события:**

**… .addEventListener("событие", функция, false)** – добавление в объект функции ( все события с on пишутся без него, например **не onclick a click** )

**… .stopPropagation()** – остановить выполнение событий по дереву DOM ( стальные события, которые были активированы у родительских элементов отменяются )

**… .removeEventListener("событие", функция, false)** – удаляет функцию из объекта

**JQuery**

**jQuery(document).ready(function($) {...} );** - создание функции, которая выполниться по прогрузке страницы

**$(function($) {...});** - создание функции, которая выполниться по прогрузке страницы

**var… = $(“…”)** – захват элемента ( выборка по типу как в CSS ) + (

**“[…]”** – по атрибутам,

**“[… = value]”** – по атрибутам и их значениям,

**“[… != value]”** обратнее пред ведущему,

**“[… ^= …]”** – по первой части атрибута к примеру у name чтоб выделить только первую часть строки ,

**“[… $= …]”** – по последней части атрибута к примеру у name чтоб выделить только последнюю часть строки,

**“[… \*= …]”** – по любой части атрибута к примеру у name чтоб выделить только любую часть строки,

**“[… |= …]”** – по слову перед знаком ( - ) роще это префикс,

**“[…], […]”** – по нескольким значениям,

**“… :first”** – первый из выборки,

**“… :last”** – последний из выборки,

**“… :not(…)”** – отсеивание элементов,

**“… :even”** – отсеивание каждый второй начиная с первого,

**“… :odd”** – каждый второй начиная со второго,

**“… :gt(…)”** – все начиная с … индекса,

**“… :lt(…)”** – все до … индекса,

**“… :header”** – все заголовки,

**“… :animated”** – все анимации,

**“… :contains(…)”** – проверка на содержание части HTML кода,

**“… :empty”** – проверка на пустоту,

**“… :has(…)”** – проверка на наличие дочерних элементов (например img )

**“… :parent** – не пустые элементы,

**“… :first-child”** – первый элемент,

**“… :last-child”** – последний элемент,

**“… :nth-child(…)** – все элементы … позиции ( по индексу, **even** – только четные, **odd** – нечетные ),

**“… :only-child” –** только один ( без соседних узлов ),

**“… :button“** – поиск кнопок,

**“… :radio“** – поиск радиокнопок

**“… :checkbox“** – элементы checkbox

**“… :text“** – элементы text

**“… :password“** – элементы password

**“… :file“** – элементы полей загрузки файлов тип file

**“… :submit“** – кнопки отправки формы

**“… :reset“** – кнопки очистки форм

**“… :image“** – изображения для отправки форм

**“… :input“** – элементы форм

**“… :selected“** – отмеченные элементы типа option

**“… :checked“** – выбранные элементы типов radio or checkbox

**“… :enabled“** – формы со статусом enabled

**“… :disabled“** - формы со статусом disabled

**“parent > child”** – условие выбора чтоб первый элемент был родителем для второго

**“prev + next”** – выбрать следующий братский узел

**“prev ~ next”** – все следующие братские узлы

**“… :eq(…)”** – элемент за индексом ( выборка по индексу)

**“… :first-of-type”** – первый из списка

**“… :last-of-type”** – последний из списка

**“… :hidden”** – все скрытые элементы

**“… :nth-last-child(…)”** – все элементы по индексу с конца

**“… :nth-of-type(…)”** – выбирает братьев этого же типа по индексу

**“… :nth-last-of-type(…)”** – выбирает братьев этого же типа по индексу с конца

**“… :only-of-type”** – узел без братьев

**“… :root”** – корневой элемент (HTML)

**“… :visible”** – все видимые элементы )

**Методы**

**… .css( { “…” : “значение” } )** – задание CSS свойств

**… .val()** – получение значения value элемента ( если вписать что-то в душки то значение измениться )

**… .attr("…")** – возвращает значение атрибута ( не отображает параметры как **checked** у checkbox )

**… .prop("…")** - возвращает значение атрибута ( отображает параметры как **checked** у checkbox )

**… .attr("…", “…”)** – меняет значение атрибута

**… .prop("…", “…”)** – меняет значение атрибута

**… .removeAttr("…")** – удаляет атрибут

**… .each(function (v\_index, elem) { … } )** – работа с сразу всеми элементами выборки ( для обращения к элементам через оператор **this** или **$(elem)**, работает как цикл )

**… .map(function (v\_index, elem) { … } )** – возвращает массив элементов или объектов

**… .addClass("…")** – добавляет класс

**… .removeClass("…")** – удаляет класс

**… .toggleClass("…")** – проверяет есть ли у элемента этот класс, если да, о он его удаляет, если нет, то добавляет

**… .hasClass("…")** – проверка на наличие класса

**… .height()** – высота блока, если что-то внести, то высота измениться

**… .innerHeight()** – высота блока с учетом padding

**… .outerHeight(true)** – полная высота блока

**… .width()** – ширина блока, если что-то внести, то высота измениться

**… .innerWidth()** – ширина блока с учетом padding

**… .outerWidtht(true)** – полная ширина блока

**… .offset({top:"…",left:"…"})** – положение элемента на странице относительно всей страницы ( если ничего не вписывать в душки, то возвращает объект с двумя параметрами: **top** и **left** )

**… .position()** – положение элемента на странице относительно блока relative возвращает объект с двумя параметрами: **top** и **left**

**… .offsetParent()** – возвращает объект родительского блока

**… .scrollTop(...)** – возвращает значение или меняет его, на сколько был прокручен сайт вниз ( изначально всегда ноль и растет при прокрутке сайта вниз )

**… .scrollLeft(…)** – возвращает значение прокрутки влево или меняет его, насколько сайт был прокручен влево

**.css(“…”,”…”)**  - для установки значений или получения его не вводя второй параметр

**… .html(“...”)** – меняет html код элемента или возвращает его, если не вписывать параметры, также можно вписывать картинки и т.д. и даже функции

**… .text(“…”)** – как и с html но он исключает все теги и работа идет напрямую с текстом

**… .wrap("<…></…>")** – оборачивает выбранный элементы или элементы в теги

**… .wrapInner("<…></…>")** – оборачивает все внутренние элементы в тег

**… .unwrap()** – удаляет родительский узел

**… .append("…")** – вставить в конец дочерних элементов выбранного узла ( **appendTo** – наоборот )

**… .prepend("…")** – вставка в начало дочерних элементов выбранного узла ( **prependTo** – наоборот )

**… .after("…")** – вставляет элемент сразу после выбранного узла ( НЕ во внутрь ) ( **insertAfter** – наоборот )

**… .before("…")** – перед выбранным элементом ( НЕ во внутрь ) ( **insertBefore** – наоборот )

наоборот это когда ты сначала пишешь новый блок а после команды указываешь куда ее вставлять

**… .remove(“…”)** – удаляет элемент, в душках уточнение элемента по селекторам, оно необязательно

**… .detach(“…”)** – удаляет элемент с возможностью восстановления ( для восстановления нужно удаленный элемент поместить в переменную, а потом присвоить ей метод **appendTo(“…”)**  **prependTo(“…”)**  **insertAfter(“…”) insertBefore(“…”)** после который кому они относяться **)**

**… .empty()** – удаляет все дочерние элементы выбранного элемента

**… .replaceWith(…)** – переместить то что в душках на место элемента от которого вызван метод ( НЕ копирование, а перемещение с заменой )

**… .replaceAll(…)** – заменить элемент в душках элементом от которого вызывали ( НЕ копирование, а перемещение с заменой )

**… .clone()** – создает копию элемента (принимает 2 параметра true или false, первый отвечает за копирование атрибутов элемента, а второй за атрибуты дочерних элементов, по умолчанию false )

**function (v\_index, v\_value ) {…}** – в методы выше можно вписывать функции, которые автоматически записывают индекс в первый принимаемый параметр, а во второй значение

**function (v\_index, v\_class) {…}** – в методы выше можно вписывать функции, которые автоматически записывают индекс в первый принимаемый параметр, а класс во второй )

**$.cssHooks.my\_hook = {**

**get : function (elem) { return $(elem).css("…"); },**

**set : function (elem, value) { … }**

**}** – создание своего CSS свойства с кросбраузерностью, посте можно вызывать метод CSS путем **…**

**… .serialize()** – возвращает строку, приведенную из массива в тип как строкового объекта, это для передачи данных, например из форм, на сервер. (тип URL )

**$.param(…)** – делает из объекта строку типа URL ( для php )

**… .serializeArray()** – создает массив с массивами из формы ( можно проходить через двойные циклы for in )

**… .get(идентификатор)** – возвращает объект с свойствами, значение которых аналогично параметрам из html, например id

**… .toArray()** – работает как и **get**, но не принимает параметры и возвращает массив

**… .size()** – количество элементов в выборке

**$.isArray(…)** – проверка массив ли

**$.isEmptyObject(…)** –проверка на отсутствие элементов

**$.isFunction(…)** – является ли функцией

**$.isNumeric(…)** – является ли числом

**$.isPlainObject(…)** – проверка объекта, но его должен был создать пользователь

**$.merge(… , …)** – объединение массивов

**$.noop()** – пустая функция, короткая запись

**$.trim(…)** – убирает в строке все лишние пробелы

**$.type(…)** – возвращает класс элемента

**итерация по объекту DOM**

**… .children("…")** – поиск в дочерних элементах ( в душках уточнение, без него вернет все элементы )

**… .closest("…")** – поиск элемента … , который находиться ближе всего

**… .find("…")** – поиск в дочерних элементах ( можно передать объект )

**… .next()** – следующий братский узел ( в душках можно уточнять элементы )

**… .nextAll()** – все следующие братские элементы, включая этот ( в душках можно уточнять элементы )

**… .nextUntil(“…”)** – следующие братья до …

**… .parent()** – родительский элемент

**… .parents()** – все родительские элементы

**… .parentsUntil("…")** – все родительские до …

**… .prev()** – предвидущий братский узел ( в душках можно уточнять элементы )

**… .prevAll()** – все предвидущие братские элементы, включая этот ( в душках можно уточнять элементы )

**… .prevUntil(“…”)** – предвидущие братья до …

**… . .siblings()** – все братья, не включая элемент от которого вызывали

**Выборка**

**… .eq(…)** – поиск по индексу ( минусовые числа это отсчет с конца )

**… .filter(“…”)** – фильтр ( можно вставлять функцию )

**… .first()** – первый элемент

**… .has(“…”)** – условие на содержание

**… .is(“…”)** – проверка на содержание элементов ( булевое значение возвращает )

**… .last()** – последний элемент

**.not(function (v\_index) { if(…) {return false;}else {return true;}})** – отбрасывание элементов функцией, с помощью возврата да или нет

**… .slice(… , …)** – обрежет элемент по индексам

**… .add("…", расположение)** – добавление элементов к выборке ( второй параметр только уточняет расположение элементов, он не обязательный )

**… .contents()** – возвращает все как и метод children, но захватывает и все текстовые узлы

**… .end()** – возвращает элемент выборки перед тем, от которого взывали

**События**

**Методы:**

**event.preventDefault()** – отменяет все базовые действия по умолчанию ( у ссылки переход на страницу, у правой кнопки мыши вызов контекстного меню )

**event.isDefaultPrevented()** – возвращает булевое значение был ли использован метод **preventDefault**

**event.stopPropagation()** – останавливает перехват событий ( при выполнении события, оно подымается к своему родителю и если у родителя тоже описана функция работы с этим событием, он ее выполнит, а эта функция это предотвращает )

**event.isPropagationStopped()** – возвращает булевое значение был ли использован метод **stopPropagation**

**event.stopImmediatePropagation()** – вроде тоже что и **stopPropagation**

**event.isImmediatePropagationStopped()** – возвращает булевое значение был ли использован метод **stopImmediatePropagation**

**Свойства:**

**event.currentTarget** – то же что и this (при работе всеравно возвращает **this**элемента)

**event.data** – объект, который принимает данные, которые можно ввести при вызове события

**event.delegateTarget** – возвращает **this** ( при работе с детьми элемента возвращает их **this**)

**event.pageX** – координаты курсора по иксу

**event.pageY** – координаты курсора по игреку

**event.namespace** – строковом типе содержится пространство имен

**event.relatedTarget** – куда переместился курсор ( возвращает объект ) хз что это

**event.result** – если к одному объекту приклеить 2 обработчика событий и в первой из них есть **return**, то в этой переменной храниться возвращаемое значение

**event.target** – возвращает объект элемента, от которого было вызвано событие

**event.timeStamp** – возвращает время в секундах прошедшее с **1 января 1970г**

**event.type** – строково возвращает тип события

**event.which** – возвращает код нажатой клавиши

**$(this)** – обращение к элементу

**Мышка:**

**… .click(function (event) { … } )** – при щелчке мыши ( если не передавать параметры то при загрузке сайта или функции будет с эмулировано нажатие на этот объект )

**... .dblclick(function (event) { … } )** – двойное нажатие мыши ( остальное как и в простом нажатии )

**… .hover(function (event) {…}, function (event) {…})** – при наведении мыши – первая функция, при выводе мыши – вторая

**… .mousedown(function (event) {…})** – при нажатии на мышку, наведенную на объект ( не отпуская )

**… .mouseup(function (event) {…})** – отпуская нажатую мышку, наведенную на объект

**… .mousemove(function (event) {…})** – при движении мыши внутри объекта

**... .mouseenter(function (event) {…})** – наводя курсор

**... .mouseleave(function (event) {…})** – снимая курсор

**Клава**

**… .keydown(function (event) {…})**– при нажатии любой кнопки на клавиатуре

**… .keyup(function (event) {…})**– при отпускании любой кнопки на клавиатуре

**… .keypress(function (event) {…})** – при нажатии кнопки

**Формы**

**… .focus(function (event) {…})** – в фокусе

**… .blur(function (event) {…})** – потеря фокуса

**… .select(function (event) {…})** – при выделении текста внутри элемента

**… .submit(function (event) {…})** – при отправке формы ( вызывается для всей формы )

**… .change(function (event) {…})** – при изменении частей формы

**Документ**

**… .ready(function($) { … } );** - при полной загрузке данной части DOM без мультимедии ( в основном для document )

**… .load(function(event) { … } );** - после загрузки данного элемента с его дочерними

**… .unload(function(event) { … } );** - при покидании страницы ( не работает в Firefox методы типа alert )

**… .$(window).resize(function (event) {…});** - при изменении размера окна

**… .scroll(function(elem) {…});** - на прокрутке экрана

**Дополнительные методы, объединение событий**

**… .bind("события" ,объект, function(event) {…});** или **.bind({“…”:”…”, …});** - функция сразу для нескольких событий ( если вместо функции передать **false**, будут отменены базовыйе свойства элемента ) ( при создании новых объектов в элементе не применяет функцию к ним ) события через пробел

**… .delegate("поиск в дочерних", "события", объекты, function(event) {…});** - как и метод **bind**, но с уточнением для какого дочернего, а само свойство задаться для родителя ( при создании новых объектов в элементе применяет функцию и к ним )

**… .one("события" ,объект, function(event) {…});** или **.one({“…”:”…”, …});** - как и метод **bind**, но действие выполняется только один раз

**… .on("события", "поиск в дочерних ", объект, function(event) {…});** - как **bind** + **delegate**

**… .off("события", "поиск в дочерних ", имя функции)** – удаление обработчиков метода **on**, можно передать только событие и к нему будут удалены все функции

**… .unbind(“событие”)** – удаление событий **bind**

**… .undelegate(“событие”);** - удаление событий **delegate**

**… .trigger(“событие”);** - включение события у элемента

**Анимации**

**… .hide(…,"linear or swing", function() {…})** – анимация скрытия сжатием в угол лево верх

**… .show(…,"linear or swing", function() {…})** – анимация показа

**… .slideUp(…,"linear or swing", function() {…})** – анимация скрытия сжатием вверх

**… .slideDown(…,"linear or swing", function() {…})** – анимация показа после сжатия вверх

**… .slideToggle(…,"linear or swing", function() {…})** – анимация показа и сжатия и восстановления ( можно кликать много раз, оно переключается между **slideUp** и **slideDown** )

**… .fadeOut(…,"linear or swing", function() {…})** – исчезновение элемента

**… .fadeIn(…,"linear or swing", function() {…})** – появление элемента

**… .fadeTo(…,0 - 1, function() {…})** – исчезновение или появление элемента до определенного значения

**… .fadeToggle(…,"linear or swing", function() {…})** – исчезновение или появление

**… .animate({"…":"…", …}, время, "linear or swing", function(){…})** – анимация свойств CSS

**… .queue(function(){ … ; $(this).dequeue();})** – выполнить функцию между очередями (относительно того где она вызвана) после ее вызова анимации останавливаются и для продолжения нужно **$(this).dequeue()** - вызов следующей анимации

**… .clearQueue()** – убирает очередь из анимации

**… .stop(…, …)** – принимает 2 параметра true или false, останавливает выполнение очереди ( без параметров останавливает выполнение идущей очереди и запускает следующий элемент, первый как true – пропускает все анимации, второй как true – не смотря на пропуск анимации ставит элемент в позицию, которая была б по завершению отмененных анимации )

**… .delay(…)** – задержка между выполнениями очередей

**… .toggle(время, "linear or swing" ,function(){…})** – скрытие/показ анимации

**… .finish()** – мгновенное окончание всех анимацией, которые были вызваны до этой команды

**$.fx.interval = …;** - интервал между выполнением смены кадров

**$.fx.off = true;** - отключение длительности анимацией, все происходит мгновенно

**PHP обмен**

**$.ajax( { url:”...”,**

**type:”post/get”,**

**data:{…},** - параметры

**success:function() {…},**

**error:function() {…},**

**beforeSend:{…},**

**complete:{…},**

**datatype:”xml/html/script/json/text”, timeOut: … ,**

**username:”…”,**

**password:”…”,**

**crossDomain:true/false** - обращение к чужому серверу да или нет

**});**

**$.ajaxSetup({…});** - установка методов функции ajax по умолчанию

Нужно создавать отдельную страницу php, ибо если обрабатывать данные на текущей, то в echo вернётся полная хуйня. Echo возвращает данные на страницу, з которой вызывали и записывает их в d\_data (первый параметр функции success). В случае переброса массивов подключаем JSON

**JSON.parse(…)** – превращает массив типа JSON в JS

**json\_encode(…);** - массив PHP в JSON ( в php )

**JSON.stringify(…)** – массив JS в JSON

**canvas**

**var canv = document.getElementById("canvas");**

**var ctx = canv.getContext('2d');**

**var grad = ctx.createLinearGradient(…, …, …, … );** - добавление градиента (начальные и конечные координаты)

**grad.addColorStop('…', '…');** - добавление цвета в градиент ( позиция внутри градиента 0-1, цвет )

**ctx.fillStyle = grad или …;** -цвет заполнения (вписывается цвет или градиент )

**ctx.strokeStyle = 'blue';** -цвет рамки

**canv.width = …** - ширина блока

**canv.height = …** - высота блока

**ctx.lineWidth = …;** -ширина рамки

**ctx.strokeRect(…, …, …, …);** -рамка прямоугольника координата Х, У, ширина, высота

**ctx.fillRect(…, …, …, …);** - прямоугольник ( Х, У, ширина, высота)

**ctx.arc(Х, У, радиус, начало, конец, true/false); -** конур круга (Х, У, радиус, начало в рад, конец в рад, за часовой или против true/false) после круг нужно заполнить это только контур

**ctx.fill();** -заполнение

**ctx.scale(…, …);** -увеличние в ширину и высту (объявляеться перед рисованием)

**ctx.rotate(-0.4);** поворот

**ctx.beginPath();** -создать линию

**ctx.moveTo(…, …);** -кординаты начала

**ctx.lineTo(…, …);** -следующая точка

**ctx.closePath();** -соеденить конец с началом

**ctx.stroke();** -закрыть путь и тобразить

**ctx.font = "50px Arial"** - параметры текста размер и шрифт

**ctx.textAlign = "…";** - позицыя текста

**ctx.fillText("…", …, ..)** - заполнение (текст, Х, У)