<https://vk.com/jsspec>

JavaScript. Уровень 2. Расширенные возможности

# Объектная модель браузера

## Окружение: JavaScript, BOM, DOM

Window : JavaScript, BOM, DOM

## Обработка событий

<div **onmousemove**="console.log(Math.random())">Двигать мышкой</div>

<button **onclick**="console.log(window)">Нажать</button>

<button **onclick**="alert(window)">Нажать</button>

<div **onclick**="document.write('Привет, мир ♫')">Нажать</div>

<https://www.w3.org/TR/html/dom.html#sec-global-attributes>

## Объект Window

HTML:

<h1>Window:</h1>

<ul>

<li onclick="log(window.navigator)">Navigator

<li onclick="log(window.screen)">Screen

<li onclick="log(window.history)">History

<li onclick="log(window.location)">Location

<li onclick="log(window.document)">Document

</ul>

JS:

function log(el){

console.clear();

console.log(el);

}

## Другие объекты BOM

<button onclick="console.log(**window**)">Нажать</button>

## Диалоговые окна Window

<h1>Диалоговые окна Window</h1>

<ul>

<li onclick="**alert**('Уведомляем')">уведомление</li>

<li onclick="var t = **confirm**('Хорошо?'); console.log(t)">подтверждение</li>

<li onclick="var t = **prompt**('Введите','123'); alert(t)">ввод данных</li>

</ul>

## Таймеры Window

<button onclick="**setTimeout**(test1,2000)">🕐через 2сек</button>

<button onclick="setTimeout('test2()',3000)">🕐через 3сек</button>

<button onclick="setTimeout(test3,4000,'⏰')">🕐через 4сек</button>

<button onclick="setTimeout(test4,0)">🕐После загрузки страницы</button>

<button onclick="setTimeout(test4,0)">🕐После загрузки страницы</button>

<button onclick="**setInterval**(test5,5000)">🕐Каждые 5 сек</button>

function test1(){ alert("через 2000мс");}

function test2(){ alert("через 3000мс");}

function test3(a){ alert(a);}

function test4(){ alert("Есть!");}

function test5(){ console.log((new Date()).toLocaleTimeString());}

## Остановка таймера

<button onclick="t1 = setTimeout(test1,5000)">

Запустить однократный с интервалом 5сек🕐

</button>

<button onclick="**clearTimeout(t1)**">

Остановить

</button>

<hr>

<button onclick="t2 = setInterval(test1,5000)">

Запустить многократный с интервалом 5сек🕐

</button>

<button onclick="**clearInterval(t2)**">

Остановить

</button>

JS: function test1(){ console.log((new Date()).toLocaleTimeString()); }

## Window: окна

var title = "Test";

//var win = open("http://htmllab.ru","nazvanie\_okna","**width**=400,**height**=300,**top**=100,**left**=300");

var win2 = **open**("","win","width=400,height=300");//открытие окна

win2.**moveTo**(100,50);//сместить в точку по оси x на 100, по y на 50

win2.**resizeTo**(600,400);//изменить ширину и высоту

win2.**moveBy**(30,30);//сместить относительно текущего положения на 30 по x и y

win2.document.write("<mark>тест</mark>");

win2.document.write("<script>alert(opener.title)<\/script>");

win2.**scrollTo**(0,100);//прокрутить окно на 100px по вертикали

## Тест

<https://goo.gl/forms/s6eYboTolQWkCgDW2>

## Практическая работа

Написать скрипт, который открывает новое окно и выводит в него текущее время.

1. Напишите код печати текущего времени через таймер:

var win = open("","win","width=400,height=300");

setInterval(

function(){

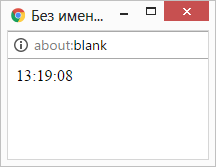
win.document.write((new Date().toLocaleTimeString()));

win.document.close();

}

,1000);

1. Создайте кнопку <button>, в обработчике onclick которой будет функция содержащая вышележащий код
2. Нажмите на кнопку и убедитесь, что всё работает нормально.
3. Открытое окно приблизительно будет выглядеть так



1. \*Доп. задание: Напишите код вывода чисел от 1 до 7 с интервалом в одну секунду. Для работы используйте setTimeout(). *Не пытайтесь применять циклы*. Например,

var i = 1, step = 7, t = 1000;

function anima(){

if(i <= step){

document.body.innerHTML += i++ + " ";

setTimeout(anima, t);

}

}

anima();

# 

## Интерфейсы

* HTMLFormElement
* HTMLInputElement
* HTMLTextAreaElement
* HTMLSelectElement, HTMLOptionElement
* HTMLImageElement
* HTMLTableElement
* <https://www.w3.org/TR/html/fullindex.html#element-interfaces>

## HTMLFormElement

* elements
* length
* name
* action
* method
* submit()
* reset()

## HTMLInputElement

* form
* defaultValue
* defaultChecked
* checked
* maxLength
* type
* value
* size
* name
* select(), click(), focus(), blur()

HTMLTextAreaElement

* form
* defaultValue
* value
* type
* cols
* rows
* disabled
* select(), focus(), blur()

## Правило обращения к атрибутам

for -> forHTML

class -> className

## Работа с CSS

div.style.color = "#369";

div.style.borderColor = "#333";

div.className = "some";

var elementClasses = elem.classList;

div.classList.add("some");

div.classList.remove("next");

console.log( div.classList.item(0));

div.classList.toggle("add");

## 

## 

## Практическая работа

Дана форма

<form action="" onsubmit="return false">

<div class="row"><input type="text" name="i1" id="i1"/></div>

<div class="row"><input type="text" name="i2" id="i3"/></div>

<div class="row"><input type="text" name="i3" id="i4"/></div>

<div class="row"><input type="text" name="i4" id="i4"/></div>

<div class="row"><input type="text" name="i5" id="i5"/></div>

<div class="row"><input type="submit" /></div>

</form>

Проверить поля формы на заполнение.

1. Убедитесь, что форма не отправляется - если в onsubmit формы или onclick гиперссылки написано return false, действие по умолчанию отменяется
2. Вместо false впишите вызов функции check(). Примечание: return должен остаться
3. В функции получить ссылку на коллекцию элементов input формы
4. Пройти в цикле (обычный for) по всем элементам
   1. Если элемент содержит пустое поле value, то окрасить его рамку в красный цвет
   2. Если все поля заполнены - разрешить отправку формы
   3. Если хотя бы одно поле не заполнено - выводить уведомление и форму не отправлять

## 

## 

## Элемент Select

HTMLSelectElement

* options
* length
* type
* selectedIndex
* value
* form
* add(), remove(), focus(), blur()

## HTMLOptionElement

* form
* defaultSelected
* text
* index
* value
* selected

## Добавление и удаление опций

<select id='s'>

<option>First</option>

</select>

var select = document.getElementById('s');

var defaultSelected = false, selected = false;

var option = new Option("Видимый текст", "значение",defaultSelected, selected);

select.add(option, select.options[0]);

**Стало**:

<select id='s'>

<option value="значение">Видимый текст</option>

<option>First</option>

</select>

select.remove(0);

## Практическая работа

Закрепляем работу со списками. Для разметки

**HTML**:

<input type="button" value="Добавить ⬆" onclick="addOption(0)"/>

<select id="select">

<option value="2018">2018</option>

</select>

<input type="button" value="Добавить ⬇" onclick="addOption(1)"/>

<input type="button" value="Очистить" onclick="clearSelect()"/>

**JS**:

function addOption(flag){

}

function clearSelect(){

}

Написать код добавления опций в верхнюю и нижнюю части списка, а также удаления всех пунктов списка.

1. Написать функцию addOption(flag), которая принимает аргумент flag и по его значению добавляет очередную опцию с годом.
2. Чтобы определить следующий год, нужно обратиться к первому элементу коллекции options HTML-элемента select и прочитать его значение value
3. Примечание: для добавления элемента коллекции в конец списка, нужно вторым аргументом передавать null
4. \* Доп. задание:

<div>

<textarea name="" id="" cols="20" rows="5"></textarea>

</div>

<input type="button" value="Добавить" onclick="add()"/>

1. Напишите функцию add(), которая будет брать все строки <textarea> и добавлять в виде набора опций в <select>

## Тест

<https://goo.gl/forms/dJFKmEneza7K2njV2>

## Элемент Table

**HTMLTableElement**: tHead, tFoot, tBodies[], rows[], caption, createCaption(), deleteCaption(), createTHead(), deleteTHead(), createTFoot(), deleteTFoot(), *insertRow(index)*, *deleteRow(index)*

**HTMLTableSectionElement**: rows[], *insertRow(index)*, *deleteRow(index)*

**HTMLTableRowElement**: cells, rowIndex, sectionRowIndex, *insertCell(index)*, *deleteCell(index)*

**HTMLTableCellElement**: cellIndex, colsSpan, rowSpan, textContent, ***innerHTML***

<input type="button" value="Добавить" onclick="addRow()"/>

<table id="table"></table>

function addRow(){

var t = document.getElementById("table"), tr, td;

tr = t.insertRow(0);

td = tr.insertCell(0);

td.innerHTML = "пункт";

td = tr.insertCell(1);

td.innerHTML = Math.round(Math.random()\*100);

}

## 

## Практическая работа

Создайте таблицу умножения (10 на 10 ячеек)

1. Создайте переменную table - ссылку на таблицу с идентификатором table: <table id="table"></table> и кнопку <input type="button" value="Построить" onclick="f()"/>

1. Описать функцию f()
2. Создать в цикле строки у таблицы
3. Во вложенном цикле создавать ячейки таблицы, заполняя их результатом перемножения счетчиков
4. Проверить работу скрипта, нажав на кнопку
5. Примерный вид результата

## 

1. Подсказка:

<input type="button" value="Построить" onclick="f()"/>

<table id="table"></table>

## 

## 

# Объектная модель документа: DOM

## Уровни DOM

DOM 0, DOM 1 Первые модели JavaScript. Поддерживает работу со стандартными коллекциями и другими элементами через стандартный набор функций

DOM 2 - более совершенные способы доступа к элементам страницы

DOM 3 - улучшенная обработка XML на основе XPath, сериализация данных и др.

DOM 4 - ждём

## DOM - дерево

<!DOCTYPE html>

<html>

<head>

<title>Document</title>

</head>

<body>

<h1>Header</h1>

<ul>

<li>item1</li>

<li>item2</li>

</ul>

</body>

</html>



## Выборка узлов DOM

var some = document.getElementById('some'); // выбор по идентификатору

var pElements = document.getElementsByTagName('p'); // выбор по тегу

var collection = document.getElementsByClassName('some'); // выбор по классу

// выбор коллекции по селектору

var pElements = document.querySelectorAll('div > p');

var pElement = document.querySelector('div > p'); //первый по селектору

## Типы узлов

|  |  |  |  |
| --- | --- | --- | --- |
| Номер типа | Тип узла | Описание | Пример |
| 1 | ELEMENT\_NODE | Элемент | <li> … </li> |
| 3 | TEXT\_NODE | Текстовый узел | слово |
| 8 | COMMENT\_NODE | Комментарий | <!-- комм --> |
| 9 | DOCUMENT\_NODE | Узел документа | document |
| 10 | DOCUMENT\_TYPE\_NODE | Декларация типа | <!doctype html> |
| 11 | DOCUMENT\_FRAGMENT\_NODE | Фрагмент док-та | создаётся программно |

## Связь между узлами

<p>просто <em>пример</em> текста</p>





|  |  |  |
| --- | --- | --- |
| Пример связи | Название | На что ссылается на |
| p.childNodes | дочерние узлы | коллекцию из двух текстовых узлов и узла EM |
| p.firstChild | первый дочерний узел | текстовый узел “просто” |
| p.lastChild | первый дочерний узел | текстовый узел “текста” |
| p.childNodes[1] | второй дочерний элемент | узел EM |
| em.parentNode | родительский элемент | узел P |
| em.nextSibling | ссылка на следующий элемент | текстовый узел “текста” |
| em.previousSibling | ссылка на предыдущий элемент | текстовый узел “просто” |
| em.firstChild | ссылка на первый дочерний | текстовый узел “пример” |

## Методы узлов Node

Свойства: .nodeName, .nodeType, .nodeValue

.hasChildNodes() - true, если есть дочерние элементы

.cloneNode(true) - клонирует узел

.appendChild(ch) - добавляет в качестве дочернего элемент ch

.insertBefore(el,p) - вставляет элемент el перед p

.replaceChild(el, p) - заменяем p элементом el

.removeChild(el) - удаляет дочерний элемент el

## Методы узлов Element

.getElementsByTagName(tag) - получает коллекцию по названию тега

.hasAttribute(a) - true, если есть атрибут а

.getAttribute(a) - возвращает значение атрибута а

.setAttribute(a,v) - устанавливает значение атрибута а в v

.removeAttribute(a) - удаляет атрибут

## Практическая работа

Дана форма со списком:

<form action="" id="addRow">

<input type="text" />

<input type="button" value="Добавить" onclick="add()"/>

</form>

<ul id="ul">

<li>Lorem.</li>

<li>Amet.</li>

<li>Ea.</li>

</ul>

Написать функцию add(), которая будет создавать клон первого элемента списка, заполнять его текст содержимым из <input type="text" /> и добавлять к ul

1. Написание функции add()

function add(){

let form = document.querySelector("#addRow");

let ul = document.querySelector("#ul");

let li = ul.firstChild.nextSibling.cloneNode(true);

let text = form.firstChild.nextSibling.value;

li.childNodes[0].nodeValue = text;

ul.appendChild(li);

}

1. \*Доп. задание: Добавьте кнопку “Показать структуру” и блочный элемент с идентификатором “structure”. Написать функцию, которая будет запускаться по нажатию на кнопку. Функция должна принимать ссылку на DOM-узел и показывать структуру этого узла (все вложенные дочерние элемента, дочерние элементы дочерних элементов и так далее)

## Тест

<https://goo.gl/forms/gFanSYg9NSMNHRE73>

## 

## 

## Создание узлов

.createElement(elementName)

.createTextNode(text)

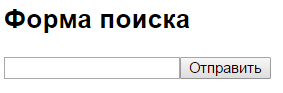
.createDocumentFragment()

.createComment(text)

## Практическая работа

Практика по созданию элементов.

1. Есть кнопка <input type="button" value="Добавить форму" onclick="addForm()"/>
2. Написать функцию addForm(), которая будет добавлять на странице форму поиска
3. Примечание: форма должна отправлять запрос в поисковую систему так, чтобы последняя показывала результат поиска по запросу
4. Приблизительный вид получившейся формы



# 

# 

# События и их обработка

## Типы событий

onclick, onmousedown, onmouseup, onmousemove, onmouseover, onmouseout

onkeydown, onkeypress, onkeyup

onfocus, onblur

onerror, onresize, onload

onsubmit, onreset, onselect, onchange, onscroll

## Модель событий W3C DOM



## 

## Регистрация обработчиков событий

function handler(ev){

//ev - ссылка на объект событие

//тело обработчика

}

var el = document.querySelector("#some");

el.onclick = handler; //базовая модель

el.addEventListener("click", handler); //модель событий W3

el.attachEvent("onclick", handler); //модель старого IE (на свалку)

## Отмена регистрации события

el.onclick = null; //базовая модель

el.removeEventListener("click", handler); //модель событий W3

el.detachEvent("onclick", handler); //модель старого IE (на свалку)

## Объект события и свойства

type, target/srcElement

button, altKey, ctrlKey, shiftKey, clientX, clientY

keyCode

## Отмена события по умолчанию/всплытия

function handler(ev){ return false } //базовая модель

function handler(ev){ ev.preventDefault(); } //модель событий W3

function handler(ev){ ev.returnValue = false; } //модель старого IE (на свалку)

if/try,catch

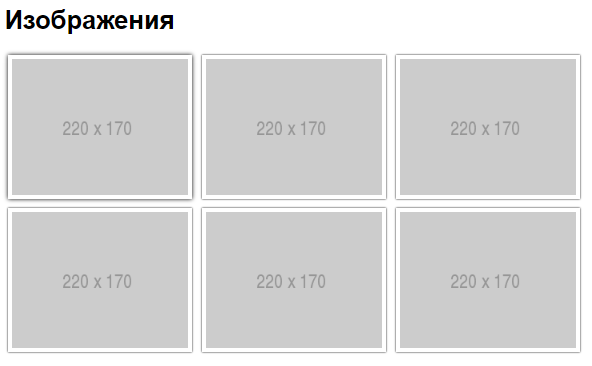
ev.stopPropagation(), ev.cancel

## 

## 

## Практическая работа

Работа на закрепление материала по работе с событиями - создание эффекта лайтбокса. Дан ряд изображений. Создать обработчик события на нажатии - должен затеняться фон (появляться <div> с полупрозрачностью) и выводиться увеличенное изображение:



1. На каждое изображение в блоке с идентификатором #gallery зарегистрировать обработчик нажатия showImage()
2. Реализовать обработчик нажатия: создавать на лету и прикреплять на странице <div> с полупрозрачностью (идентификатор #darkbox) и изображение в блоке #img
3. Реализовать обработчик нажатия на затемнённый фон так, чтобы #darkbox и #imgBox удалялись
4. \* Доп. задание (опционально): Создать массив с названиями изображений и выводить их в #gallery в момент формирования страницы. Примечание: не забудьте "навесить" обработчики.

HTML:

<div class="container">

<h1>Изображения</h1>

<div id="gallery">

<img src="http://placehold.it/220x170" alt="" />

<img src="http://placehold.it/220x170" alt="" />

<img src="http://placehold.it/220x170" alt="" />

<img src="http://placehold.it/220x170" alt="" />

<img src="http://placehold.it/220x170" alt="" />

<img src="http://placehold.it/220x170" alt="" />

</div>

</div>

<div id="darkbox"></div>

<div id="imgBox">

<img src="http://placehold.it/650x470" alt="" />

</div>

CSS:

|  |
| --- |
| \* {  font-family: Arial; } #gallery{  width: 750px; } #gallery img{  box-shadow: 0 0 2px #000;  margin: 4px;  border: 5px solid #fff; }  #gallery img:hover{  box-shadow: 0 0 7px #000; } #darkbox{  background: rgba(0,0,0,.5); } #darkbox, #imgBox, #imgBox img {  position: fixed;  margin: auto;  top: 0;  left: 0;  bottom: 0;  right: 0; } #imgBox img {  box-shadow: 0 0 10px #000;  border: 5px solid #fff;  border-radius: 5px; } |

## Тест

<https://goo.gl/forms/UVSGoGW2wz0Kgsjn1>

## 

## 

## Практическая работа

## 

## 

## Курсы для дальнейшего обучения

JavaScript. Уровень 3. React и JSX <http://www.specialist.ru/course/react>

JavaScript. Уровень 3а. jQuery. <http://www.specialist.ru/course/kveri>

JavaScript. Уровень 3. jQuery Расширенные возм. <http://www.specialist.ru/course/kveri2>

JavaScript. Уровень 3в. Node.js <http://www.specialist.ru/course/node>

JavaScript. Уровень 3г. HTML5 API <http://www.specialist.ru/course/api>