Игорь Борисов

# JavaScript. Уровень 2 Расширенные возможности (JavaScript в окружении браузера)

http://igor-borisov.ru

# Темы курса

- Объектная модель браузера
- Объектная модель документа и базовая модель событий
- Объектная модель документа и модель событий W3C DOM2
- Практикум
- Создание офф-лайн приложения

# Запуск файла в браузере

/****** Скрипт встроен в HTML-файл **********/

```
<!doctype html>
<html>
  <head>
    <title>Это HTML-файл</title>
    <script>
      // Здесь будет наш скрипт,
      // который будет исполняться
    </script>
  </head>
  <body>
  </body>
</html>
```

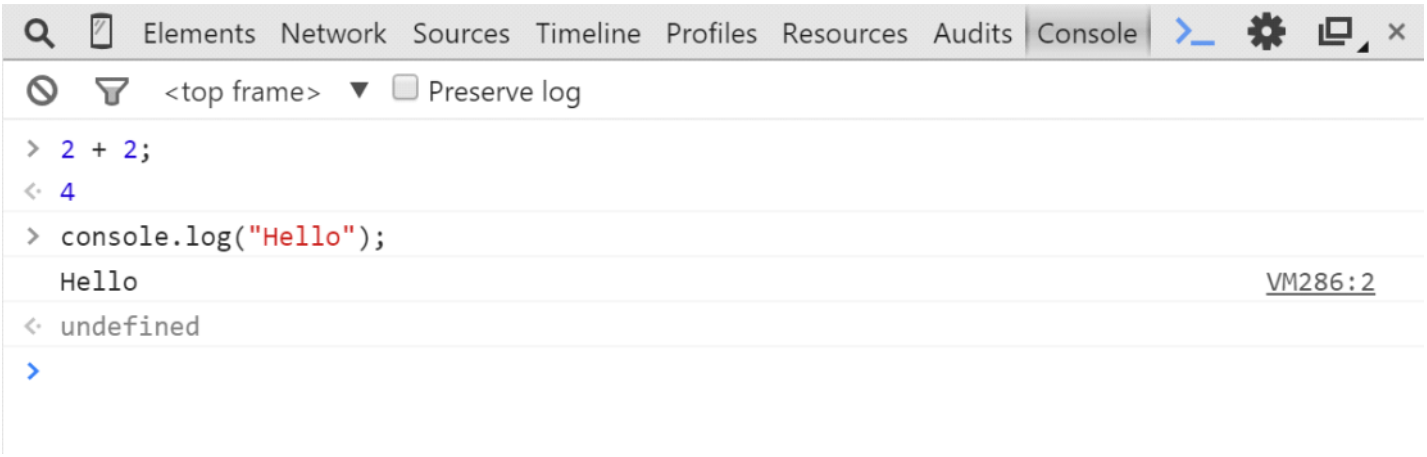
/******Скрипт подключен к HTML-файлу **********/

```
/*
  Это JavaScript-файл по имени, например, file.js
  Здесь будет наш скрипт, который будет подключаться к HTML-файлу
  и исполняться
*/
```

```html
<!doctype html>
<html>
  <head>
    <title>Это HTML-файл</title>
    <!--
      Здесь подключается внешний файл
      с нашим скриптом
    -->
    <script src="file.js"></script>
  </head>
  <body>
  </body>
</html>
```

# Вывод в консоль браузера

- F12




```
<script>
  console.log('Hello');
</script>
```

- Ctrl + L
  - Очистить консоль
- Стрелка вверх/Стрелка вниз
  - Предыдущая/Следующая команды
- Enter
  - Исполнить скрипт
- Ctrl + Enter
  - Многострочный ввод

# Обработка событий

*Attributes defined elsewhere*

- id, class (document-wide identifiers)
- lang (language information), dir (text direction)
- title (element title)
- style (inline style information)
- disabled (disabled input controls)
- accesskey (access keys)
- tabindex (tabbing navigation)
- onfocus, onblur, onclick, ondblclick, onmousedown, onmouseup, onmouseover, onmousemove, onmouseout, onkeypress, onkeydown, onkeyup

```
<button onclick="console.log('Ура')">Ура</button>


<script>
  function foo(){
    console.log('Ура');
  }
</script>
<button onclick="foo()">Ура</button>
```

Модуль 1

# JavaScript. Уровень 2
# Объектная модель браузера

# Темы модуля

- Объектная модель браузера
- Объект Window
- Объект History
- Объект Navigator
- Объект Screen
- Объект Location
- Методы объекта Document
- Методы объекта Window

# Объект Window

- window
  - ссылка на объект Window
- navigator
  - ссылка на объект Navigator
- screen
  - ссылка на объект Screen
- history
  - ссылка на объект History
- location
  - ссылка на объект Location
- document
  - ссылка на объект Document
- методы

# Объект History

```
// Сколько страниц посетили в данном окне?
history.length

// Загружаем предыдущий документ
history.back()
// Загружаем следующий документ
history.forward()
// Загружаем следующий документ отстоящий от текущего на 2
шага
history.go(2)
```

# Объект Navigator

```
// Примеры на Google Chrome

navigator.appCodeName // Mozilla
navigator.appName // Netscape
navigator.appVersion //5.0 (Windows NT 6.3; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.99
Safari/537.36
navigator.userAgent // Mozilla/5.0 (Windows NT 6.3; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/39.0.2171.99
Safari/537.36
navigator.platform // Win32
navigator.product // Gecko
navigator.onLine // true
navigator.language // ru
navigator.languages // ["ru-RU", "ru", "en-US", "en",
"da"]
navigator.vendor // Google Inc.
navigator.cookieEnabled // true
```

# Объект Screen

```
screen.width // 1920
screen.height // 1080

screen.availWidth // 1920
screen.availHeight // 1032

screen.availTop // 48
screen.availLeft // 0

screen.colorDepth // 24
```

# Объект Location

```
// Значения свойств для адреса
http://www.company.com:8080/section/web/file.html?x=10&y=20#metka

location.href
//http://www.company.com:8080/section/web/file.html?x=10&y=20#metka
location.protocol // http:
location.port // 8080
location.hostname //www.company.com
location.host //www.company.com:8080
location.pathname // /section/web/file.html
location.search // ?x=10&y=20
location.hash // #metka

// Приведение к строке
location.toString()
//http://www.company.com:8080/section/web/file.html?x=10&y=20#metka


// Повторный запрос
location.reload();

// Загрузить ресурс с добавлением в историю
location.assign("http://www.company.com");

// Загрузить ресурс без добавления в историю
location.replace("http://www.company.com");
```

# Объект Document: запись в документ

```
document.open();
document.write("<h1>Заголовок первого уровня</h1>");
document.write("<p>Первый абзац</p>", "<p>Второй
абзац</p>");
document.close();
```

# Window: диалоговые окна

```
// Информирующее диалоговое окно
alert ("Привет");

// Подтверждающее диалоговое окно
confirm ("Вы уверены?");

// Диалоговое окно для ввода ответа на вопрос
prompt ("Введите ваш возраст", "25");
```

# Window: таймеры

```
var timer = setTimeout("alert('Hello')", 3000);

clearTimeout(timer);

var interval = setInterval("alert('Hello')", 3000);

clearInterval(interval);


// Вариант использования таймеров

function test(){
  alert('Hello');
}

// Имя функции без скобок, ссылка на функцию
setTimeout(test, 1000);
```

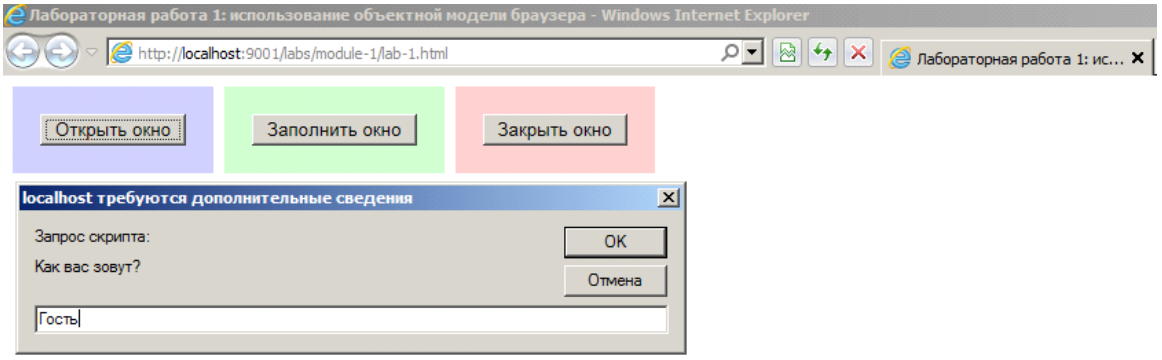
# Window: манипуляции с окном

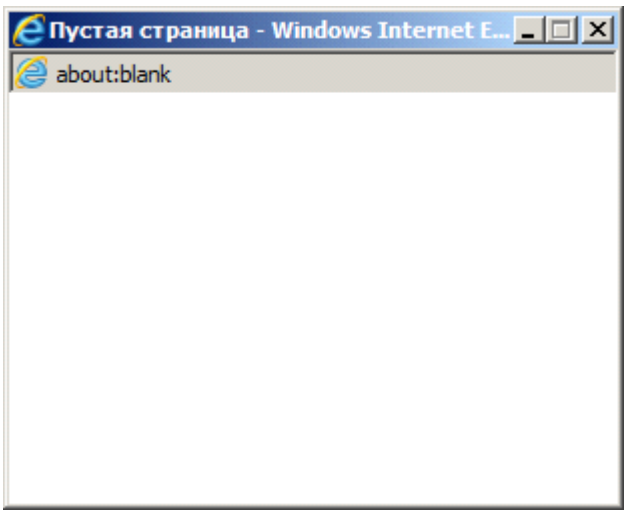
```
// Открываем окно
window.open("", "", "");

// Открываем окно, загружаем документ
window.open("http://ya.ru", "", "");

// Используем параметры
window.open("", "", "width=300, height=300");

// Ссылка на открытое окно
var w = window.open("", "", "top=300, left=300");

// Передаём фокус окну
w.focus();
// Снимаем фокус с окна
w.blur();

// Окно открыто?
console.log( w.closed ) // false

// Закрываем окно
w.close();

// Ссылка на открытое окно
var w = window.open("", "");
// w - объект window дочернего окна
// Пишем в дочернее окно
w.document.write("<h1>Hello</h1>");

// В дочернем окне
opener.location.pathname;
// opener - объект window родительского окна


// Открываем окно для манипуляций
var w = window.open("", "", "width=300, height=300);
w.focus();

// Позиционируем окно по координатам x и y
window.moveTo(200, 300);
// Позиционируем окно относительно текущей позиции
window.moveBy(20, 20);
```

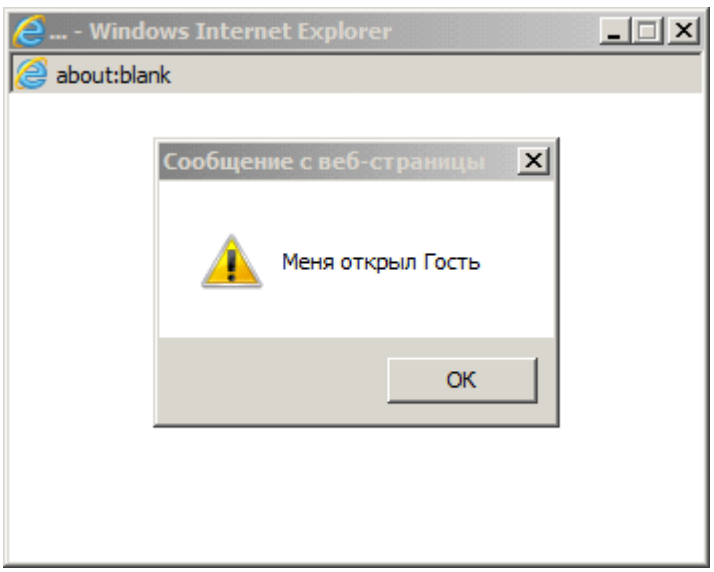
```
// Изменяем размер окна по ширине и высоте
window.resizeTo(100, 150);
// Изменяем размер окна относительно текущего размера
window.resizeBy(20, 30);
```

# Лабораторная работа 1

## Использование свойств и методов объектов браузера



# Содержание лабораторной работы 1

## Использование свойств и методов объектов браузера

### Упражнение 1: Создание нового окна

- В текстовом редакторе откройте файл **labs/module-1/lab-1.html**
- Найдите декларацию функции **winOpen()** и опишите её:
  - Проверьте, существует ли дочернее окно и не закрыто ли оно
  - Если окно не открыто, то
    - Откройте новое окно со следующими параметрами: **ширина**: **300px**, **высота**: **200px** и расположите его *строго по середине экрана*
    - Ссылку на новое окно присвойте глобальной переменной **win**
    - Передайте "фокус" созданному окну
- Сохраните файл **lab-1.html**, запустите его в браузере и убедитесь в работоспособности кода

### Упражнение 2: Закрытие дочернего окна

- Продолжайте работать с файлом **labs/module-1/lab-1.html**
- Найдите декларацию функции **winClose()** и опишите её:
  - Проверьте, существует ли дочернее окно и не закрыто ли оно
  - Если дочернее окно открыто, то
    - Передайте "фокус" созданному окну
    - Закройте его
- Сохраните файл **lab-1.html**, запустите его в браузере и убедитесь в работоспособности кода

### Упражнение 3: Запись данных в документ нового окна

- Продолжайте работать с файлом **labs/module-1/lab-1.html**

- Найдите декларацию функции **docWrite()** и опишите её:
    - Проверьте, существует ли дочернее окно и не закрыто ли оно
    - Если дочернее окно открыто, то
        - Передайте "фокус" созданному окну
        - Запишите в документ дочернего окна содержание, состоящее из полного набора элементов, включая **html**, **head**, **body**, **title**
        - Включите в содержание документа элемент **p** с содержимым:
          "Текст параграфа"
        - Включите в содержание документа элемент **script**. Скрипт должен вызывать метод **alert** объекта **Window** с текстом:
          "Меня открыло окно: [*значение свойства title объекта document родительского окна*]"
- Сохраните файл **lab-1.html**, запустите его в браузере и убедитесь в работоспособности кода
- В этом упражнении у вас может произойти ошибка. Подумайте, что стало её причиной? Как этого избежать?

# Что мы изучили?

- Объектную модель браузера
- Часто используемые объекты Screen и Location
- Редко используемые объекты History и Navigator
- Запись содержимого в документ
- Исполнять отложенный код
- Возможность общения с пользователем посредством диалоговых окон
- Манипуляции с окнами

Модуль 2

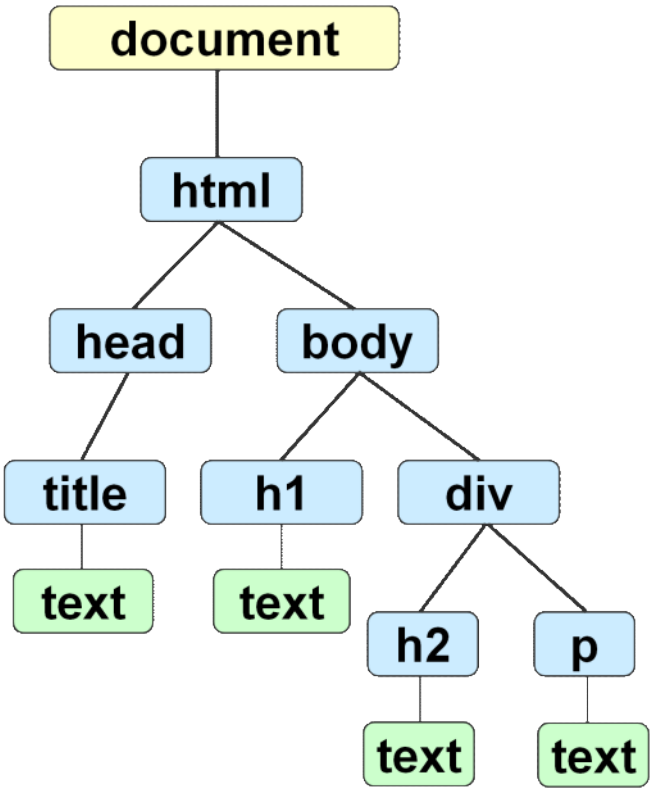
# JavaScript. Уровень 2
# Объектная модель документа

# Темы модуля

- Объектная модель документа и коллекции
- Свойства и методы основных HTML-элементов
- Манипуляции с узлами документа
- Базовая (исходная) модель обработки событий

# Объектная модель документа

```
<html>

  <head>
    <title>PAGE</title>
  <head>

  <body>
    <h1>PAGE</h1>
    <div>
      <h2>Раздел</h2>
      <p>Параграф</p>
    </div>
  </body>

</html>
```

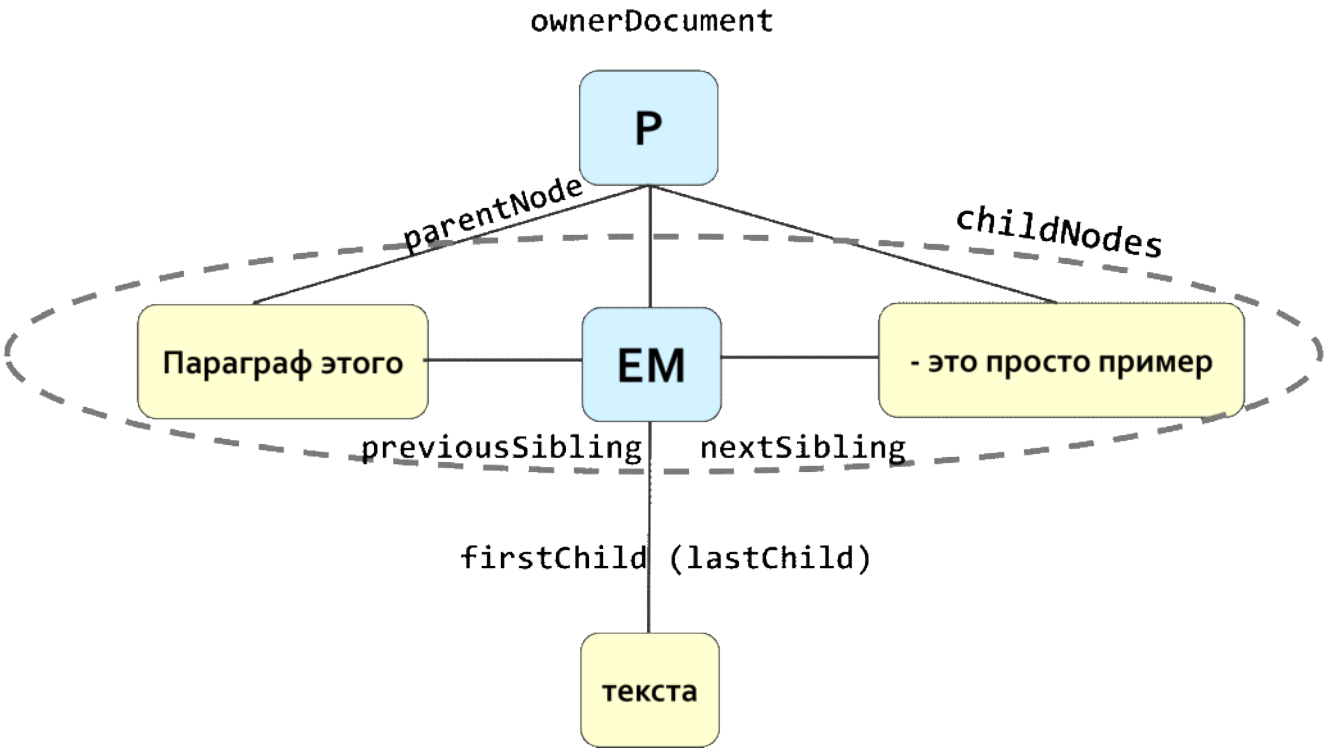


# Типы узлов

| Код типа | Тип узла | Описание | Пример |
|---|---|---|---|
| 1 | ELEMENT | Элемент |  . . .  |
| 2 | ATTRIBUTE | Атрибут элемента | align="center" |
| 3 | TEXT | Текстовый узел | Это текст |
| 8 | COMMENT | Комментарий | <!-- comment --> |
| 9 | DOCUMENT | Узел документа | document |
| 10 | DocumentType | Декларация типа документа | <!DOCTYPE HTML> |

# Связи между узлами

```
<p>Параграф этого <em>текста</em> - это просто пример</p>
```



# Выборка узлов-элементов

```
// В примерах не учитываются white-space
<body>
  <form action="" id="frm">
    <input type="text" name="search" value="Слово">
    <textarea name="desription"></textarea>
    <input type="submit" value="Послать">
    <input type="reset" value="Очистить">
  </form>
</body>

// Выбираем форму
var f = document.getElementById("frm");

// Объект FormHTMLElement
console.log( f.action ); // ""
f.action = "some-script";
console.log( f.action ); // "some-script"

// Выбираем все элементы input в контексте формы
var coll = document.getElementsByTagName("input");

/*
  Объект HTML-коллекция
 {
   0: element-1,
   1: element-2,
   2: element-3,
   length: 3
 }
*/

// Сколько элементов в коллекции?
console.log( col.length ); // 3

// Выбираем текстовое поле
var txt = coll[0];

// Объект InputHTMLElement
console.log( txt.value ); // Слово
txt.value = "Новое слово";
console.log( txt.value ); // Новое слово
console.log( txt.defaultValue ); // Слово
txt.defaultValue = "Слово по умолчанию;
```

```
// Выбираем всех детей формы
var children = f.childNodes;

// Сколько элементов в коллекции?
console.log( children.length ); // 4

// Тип второго ребёнка
console.log( children[1].nodeType ); // 1
// Что за HTML-элемент третий ребёнок?
console.log( children[2].nodeName ); // input
```

# Выборка текстовых узлов

```
// В примерах не учитываются white-space
<body>
  <p>Hello <b>my</b> world</p>
</body>

// Выбираем абзац
var p = document.body.firstChild;

// Выбираем содержимое текстовых узлов
var hello = p.firstChild.nodeValue; // Hello
var my = p.firstChild.nextSibling.nodeValue; // my
var world = p.lastChild.nodeValue; // world

console.log( hello + my + world );

// Выбираем содержимое всех текстовых узлов в контексте
абзаца
var str = p.textContent;
console.log( str ); // Hello my world
```

# События и методы элементов

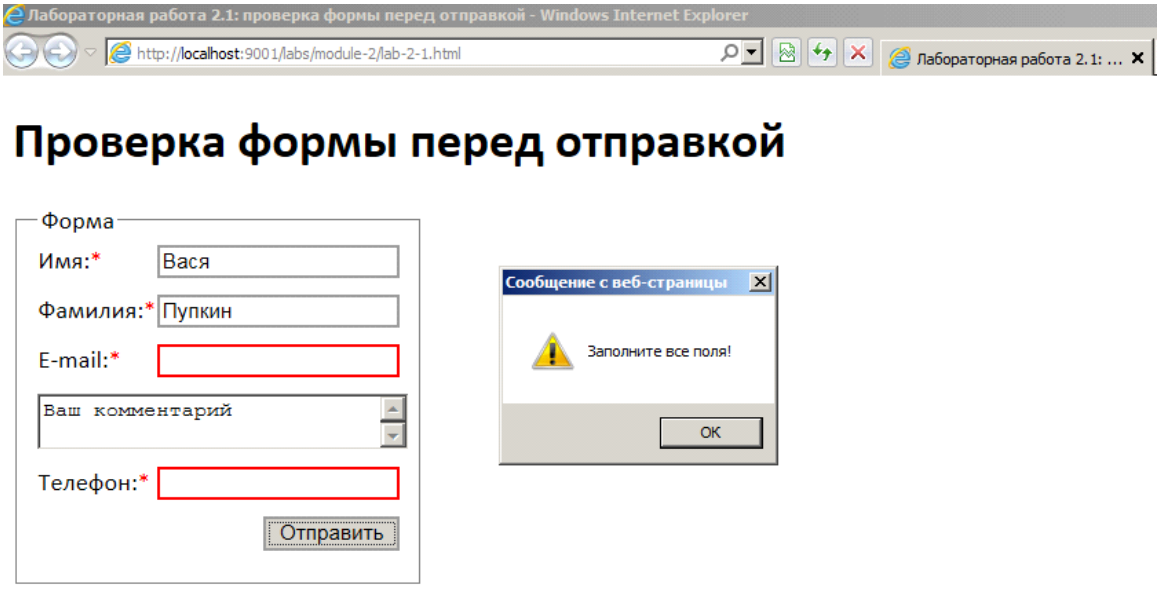
```
// В примерах не учитываются white-space
<body>
  <form action="" onsubmit="sendForm()">
    <input type="text" name="search" value="Слово">
  </form>
  <button id="send" onclick="foo(1)">Послать</button>
  <button id="clear" onclick="foo(0)">Очистить</button>
</body>

// Выбираем форму
var f;

window.onload = function(){
  f = document.getElementsByTagName("form")[0];
}

// Функции
function sendForm(){
  var txt = f.getElementsByTagName("input")[0];
  alert("Отправляем слово: " + txt.value);
}

function foo(btn){
  if (btn == 1){
    f.submit();
  }
  if (btn == 0){
    f.reset();
  }
}
```

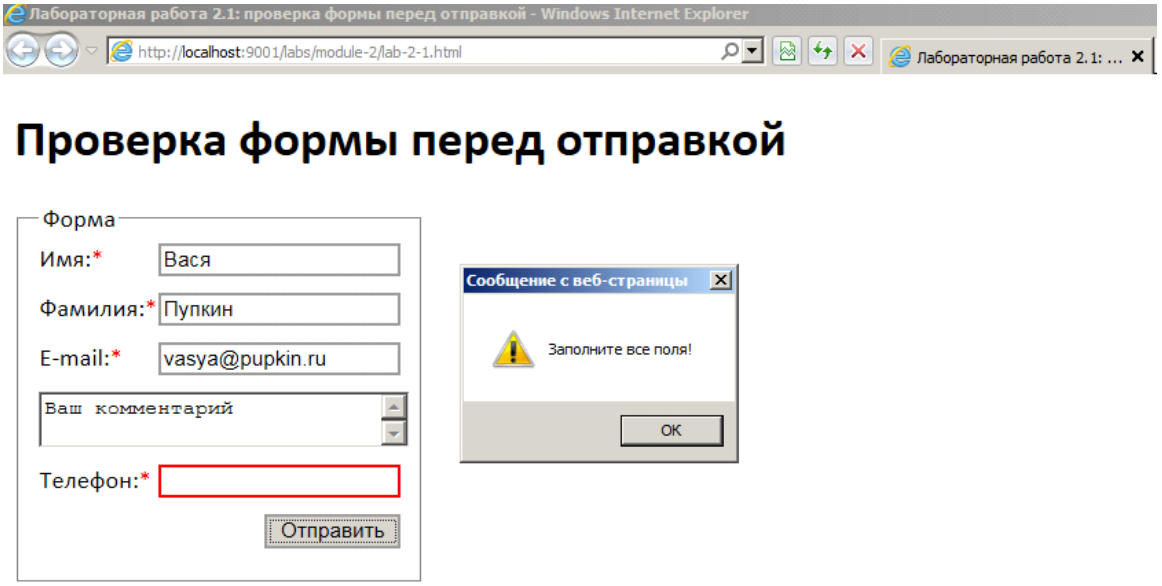
# Правила обращения к атрибутам элементов

```
// Правила именования
element.style.color = "red";
element.style.color = ""; // Значение по умолчанию
element.style.padding = '10px';

// border-top-width
element.style.borderTopWidth = '3px';

// Разрешение конфликта имен атрибутов
// для html-атрибута for
element.htmlFor = "x";
// для css-свойства float
element.style.cssFloat = "left";
// для html-атрибута class
element.className = "someClassName";
```

# Лабораторная работа 2.1

## Проверка заполнения полей формы перед отправкой



# Содержание лабораторной работы 2.1

## Проверка заполнения полей формы перед отправкой

### Упражнение 1: Описание функции проверки формы

- В текстовом редакторе откройте файл **labs/module-2/lab-2-1.html**
- Присвойте событию **onclick** кнопки со значением "**Отправить**" вызов функции **checkForm()**
- Найдите декларацию функции **checkForm()** и опишите её:
  - Выберите элемент **form** в переменную
  - Выберите все **нужные** элементы элемента **form**
  - Проверьте значения обязательных полей (они помечены звёздочкой)
  - Если хотя бы у одного из полей значения отсутствуют, то
    - Изменить цвет рамки пустых полей на красный
    - Выдать **alert** с предупреждением "Заполните обязательные поля"
    - Форму не отправлять
  - При последующей проверке, если поле с красной рамкой заполнено, вернуть цвет рамки в первоначальный
  - При заполнении всех обязательных полей отправить форму
- Сохраните файл **lab-2-1.html**, запустите его в браузере и убедитесь в работоспособности кода

# Элемент Select

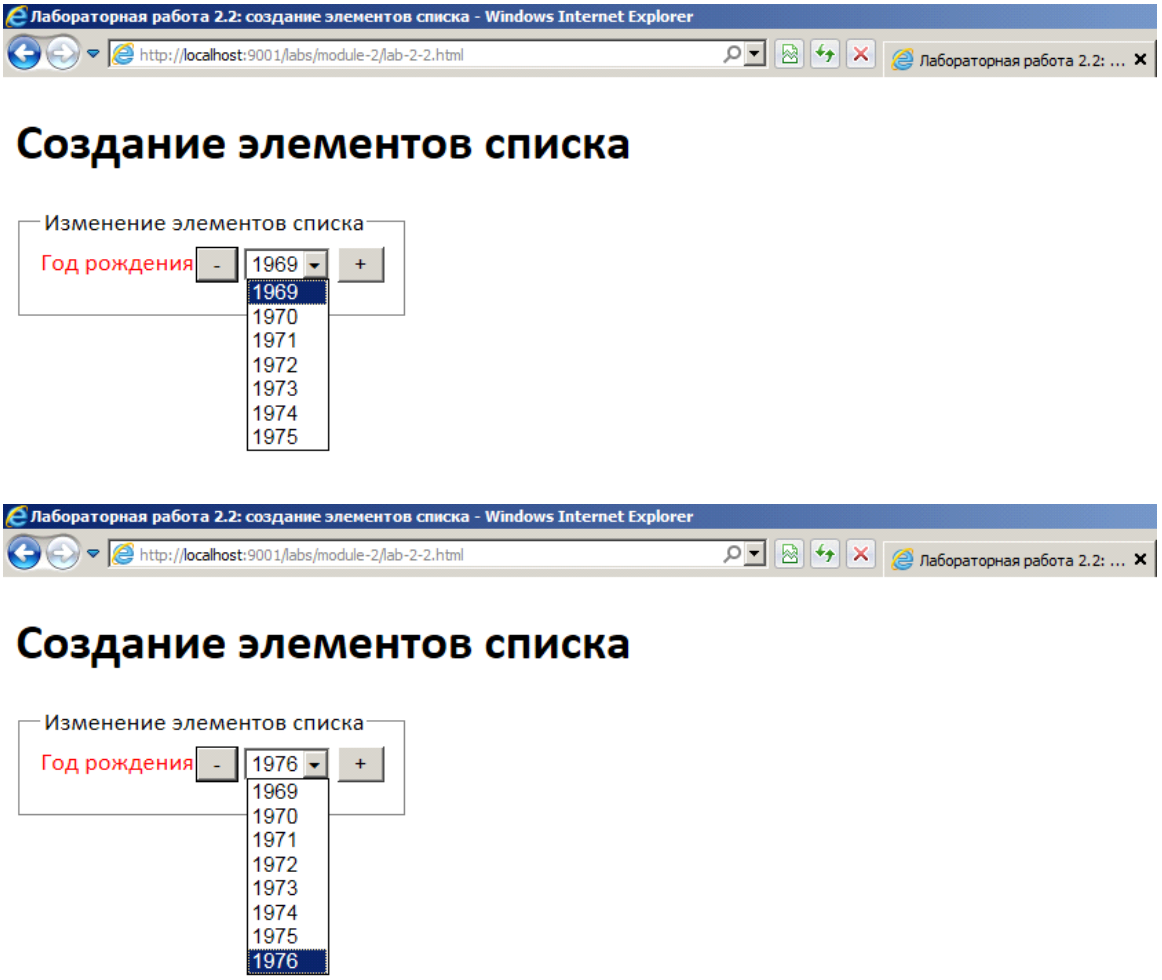
```
<form name="f" action="">
  <select name="book">
    <option value="1">JavaScript</option>
    <option value="2">PHP</option>
    <option value="3">XML</option>
    <option value="4">AJAX</option>
  </select>
</form>

// Выбираем список
var select = document.getElementsByTagName("select")[0];
// Количество элементов option
console.log( select.length ); // 4

// Выбираем options
var options = select.options;
console.log( select.length == options.length ); // true

// Делаем option выбранным по умолчанию
options[3].defaultSelected = true;
// Делаем option выбранным
options[2].selected = true;
// Меняем текст первого option
options[0].text = "ECMA-262";
// Меняем значение первого option
options[0].value = "10";

// Индекс выбранного элемента option
console.log( select.selectedIndex ); // 2
// select.selectedIndex === select.options[2]

// Текст выбранного элемента option
console.log( select.selectedIndex.text ); // XML
// Значение выбранного элемента option
console.log( select.selectedIndex.value ); // 3
```

# Создание узлов

```
<html>
  <head>
    <title>Создание узлов</title>
  </head>
  <body>
    <p id="p1">Paragraph number one</p>
  </body>
</html>

// Создаём элемент P
var p2 = document.createElement('p');
// Создаём текстовый узел
var t = document.createTextNode('Я — новый параграф');
// Добавляем текстовый узел в новый элемент P
p2.appendChild(t);
// Добавляем новый элемент P последним ребёнком body
document.body.appendChild(p2);
// Добавляем атрибут id новому элементу P
p2.setAttribute('id', 'p2');

// Выбираем первый абзац
var p1 = document.getElementById("p1");
// Вставляем (перемещаем) новый элемент P перед первым абзацем
document.body.insertBefore(p2, p1);

// Клонируем (копируем) новый узел со всем содержимым
var cloned = p2.cloneNode(true);
// Добавляем атрибут id новому элементу P
cloned.setAttribute('id', 'p3');
// Добавляем скопированный узел последним ребёнком body
document.body.appendChild(cloned);
// Удаляем p с id=2 через его родителя
p2.parentNode.removeChild(p2);
```

# Лабораторная работа 2.2

## Добавление элементов option в выпадающий список



# Содержание лабораторной работы 2.2

## Манипуляции с выпадающим списком

### Упражнение 1: Добавление элементов option в список

- В текстовом редакторе откройте файл **labs/module-2/lab-2-2.html**
- Создайте функцию **addYear()**, которая добавляет новые элементы **option** в список **year**
- Присвойте событию **onclick** кнопок **sub** и **add** вызов функции **addYear()** любым удобным способом
- Когда нажата кнопка **sub**, элемент добавляется в начало списка
- Когда нажата кнопка **add**, элемент добавляется в конец списка
- Опишите функцию **addYear()**:
  - Определитесь, куда надо добавлять новый элемент: в начало или в конец списка
  - Создайте новый элемент **option** и присвойте ему необходимые параметры
  - Добавьте новый элемент в список **year**
- Сохраните файл **lab-2-2.html**, запустите его в браузере и убедитесь в работоспособности кода

# Что мы изучили?

- Познакомились с понятием HTML-коллекций
- Познакомились с объектной моделью документа
- Научились манипулировать узлами HTML-дерева
- Познакомились с базовой (исходной) моделью обработки событий

Модуль 3

# JavaScript. Уровень 2
# Объектная модель документа и модель событий W3C DOM2

# Темы модуля

- Модель обработки событий W3C DOM2
- Демо-практикум: создание игры "Memory game"

# Основные типы событий

- Большинство объектов
  - onclick
  - onmousedown
  - onmouseup
  - onmousemove
  - onmouseover
  - onmouseout
- document, body, input, textarea
  - onkeydown
  - onkeypress
  - onkeyup
- window, body, a, button, input, label, select, textarea
  - onfocus
  - onblur
- window, body, iframe, img, object
  - onload
- window, body
  - onunload
- window, body, iframe
  - onresize
- form
  - onsubmit
  - onreset

- input, textarea
  - onselect

- input, textarea, select
  - onchange

- window, body, элементы с прокруткой
  - onscroll

# Модель событий W3C DOM

```
var p = document.getElementsByTagName('p')[0];

function foo(){
  console.log("По мне кликнули");
}

// Регистрация события
p.addEventListener('click', foo);

// Отмена регистрации события
p.removeEventListener('click', foo);
```

# Объект события и его свойства

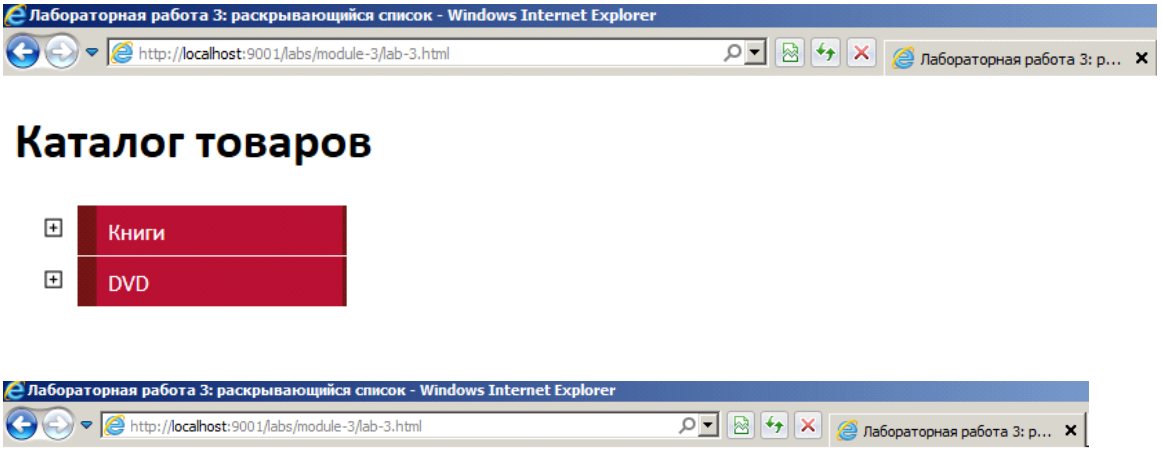
```
var p = document.getElementsByTagName('p')[0];
p.addEventListener('click', test);

function test(e){
  // Тип события
  console.log( e.type );

  // Ссылка на элемент, в котором зарегистрировано событие
  console.log( e.target ); this === target

  // Мышиные события

  // Какая кнопка мыши нажата?
  console.log( e.button );

  // Были ли нажаты соответствующие клавиши при нажатии
кнопки мыши?
  console.log( e.altKey );
  console.log( e.ctrlKey );
  console.log( e.shiftKey );


  // Координаты точки нажатия относительно клиентской
части окна браузера
  console.log( e.clientX );
  console.log( e.clientY );

  // Координаты точки нажатия относительно монитора
  console.log( e.screenX );
  console.log( e.screenY );

  // Событие клавиатуры
  // Какая клавиша нажата?
  console.log( e.keyCode );

 }

// Отслеживание текущего элемента при эффекте "всплытия
пузырька"
// Ссылка на элемент, на котором произошло событие
console.log( e.currentTarget );
```

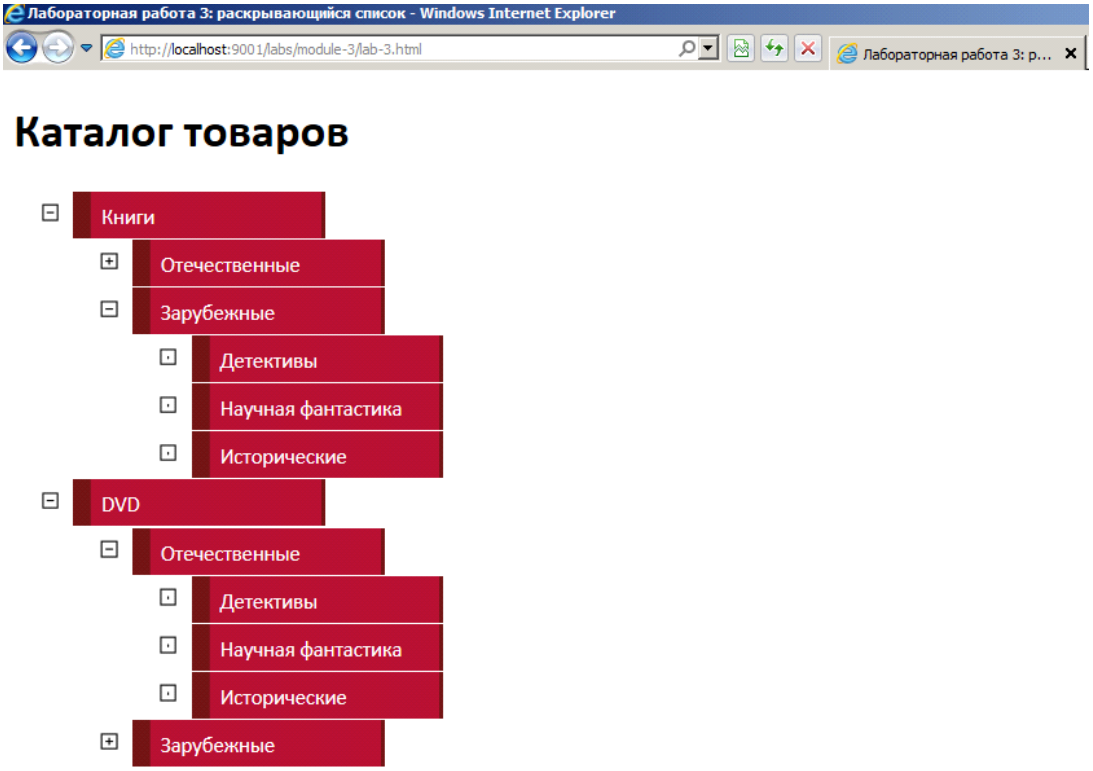
# Отмена специфических событий

```
var a = document.getElementsByTagName('a')[0];
a.addEventListener('click', foo);

function foo(e){
  // Отмена всплытия события
  e.stopPropagation();

  // Отмена действий по умолчанию
  e.preventDefault();
}
```

# Лабораторная работа 3

## Создание динамического раскрывающегося списка



# Содержание лабораторной работы 3

## Создание динамического раскрывающегося списка

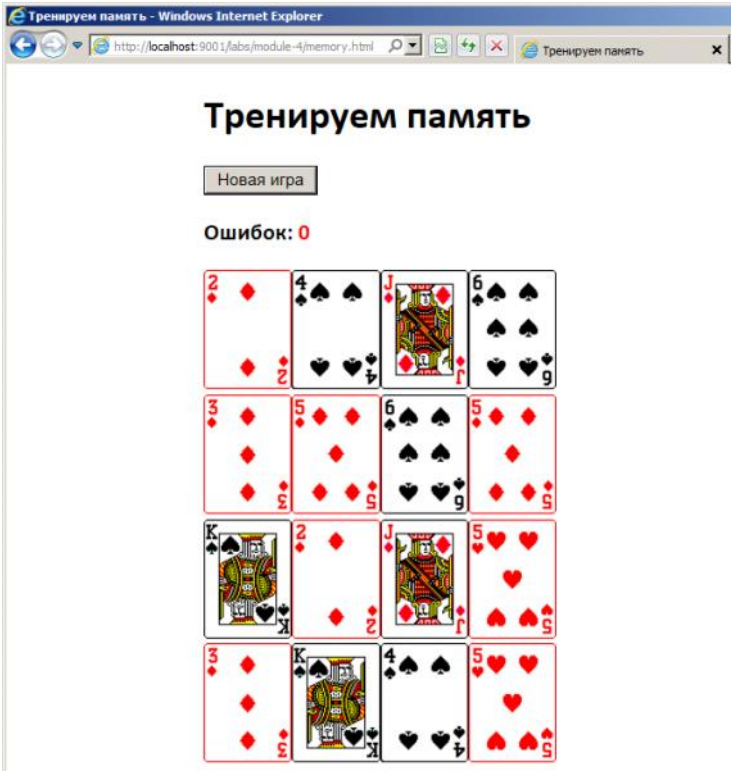
### Упражнение 1: Инициализация кода

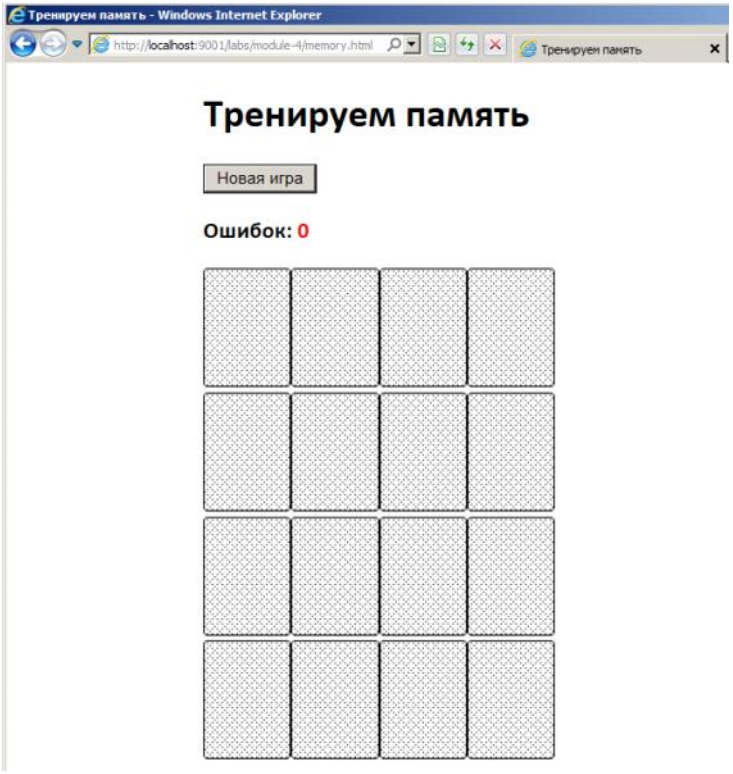
- В текстовом редакторе откройте файл **labs/module-3/lab-3.html**
- Найдите декларацию функции **init()** и опишите её:
    - Если элемент **LI** имеет вложенный список, то
        - присвойте данному элементу в качестве маркера картинку **imgs/plus.gif**
        - скройте вложенный список с помощью стилевого свойства **display**
    - Зарегистрируйте функцию **aClick()** в качестве обработчика события **onclick** для всех ссылок (элементов **A**)
    - Функция **init()** должна запуститься автоматически после загрузки страницы
- Сохраните файл **lab-3.html**, запустите его в браузере и убедитесь в работоспособности кода

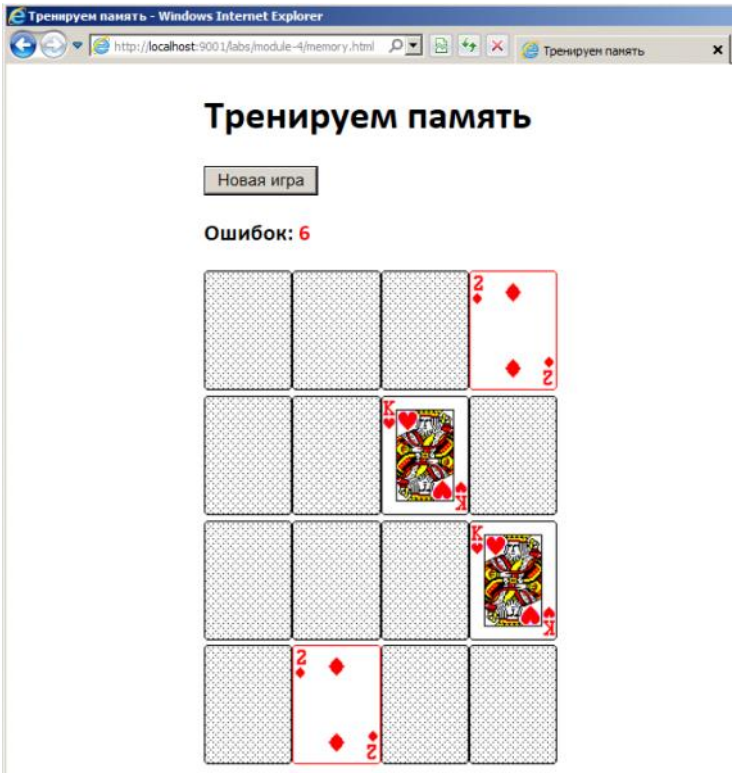
### Упражнение 2: Описание основной логики кода

- Продолжайте работать с файлом **labs/module-3/lab-3.html**
- Найдите декларацию функции **aClick()** и опишите её:
    - Получите ссылку на текущий элемент **A**
    - Найдите родительский элемент **LI**
    - Если у выбранного элемента **LI нет** вложенного списка, то разрешите переход по ссылке
    - Если у выбранного элемента **LI есть** вложенный список, то
        - Если список скрыт, его надо показать и присвоить элементу **LI** в качестве маркера картинку **imgs/minus.gif**
        - Если список показан, его надо скрыть и присвоить элементу **LI** в качестве маркера картинку **imgs/plus.gif**
        - Запретите переход по ссылке
    - Сохраните файл **lab-3.html**, запустите его в браузере и убедитесь в работоспособности кода

# Игра "Memory game"




Тренируем память - Windows Internet Explorer
http://localhost:9001/labs/module-4/memory.html
Тренируем память
Тренируем память
Новая игра
Ошибок: 6

# План действий

- Необходимые переменные
  - Время показа всех картинок в начале новой игры
  - Время показа картинок при выборе пары
  - Общее количество карт для показа
  - Счётчик неправильных ходов
  - Массив текущих карт, участвующих в игре
- Необходимые функции
  - Инициализации. Запускается после загрузки страницы
    - Создание необходимого количества элементов **IMG**
    - Добавление созданных элементов на страницу
    - Назначение обработчика события **onclick** для элементов **IMG**
  - Новой игры. Запускается при нажатии на кнопку "**Новая игра**"
    - Обнуляются и инициализируются все необходимые переменные
    - Перемешивается колода
    - Создаётся колода для текущей игры
    - Присваивание значений атрибута **src** картинкам
    - Временный показ картинок
    - Переворачивание картинок "рубашкой" вверх
  - Обработчика события onclick элементов **IMG**
    - Показ первой выбранной картинки
    - Показ второй выбранной картинки
    - При неправильном выборе, переворот обеих картинок "рубашкой" вверх и изменение счётчика неправильных ходов

# Что мы изучили?

- Познакомились с моделью обработки событий W3C DOM2
- Попрактиковались в изученных темах
- Написали небольшое клиентское приложение

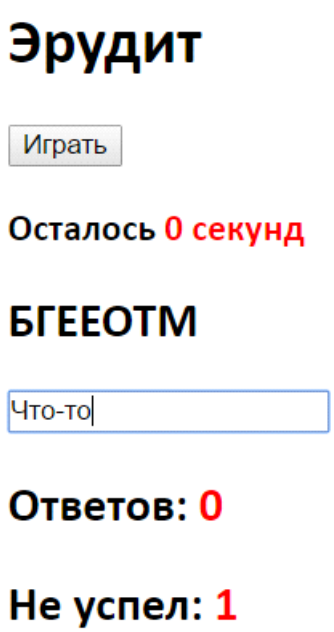
Модуль 4

# JavaScript. Уровень 2
# Практикум

# Темы модуля

- Практика: создание игры "Word scramble"
- HTML5 API: Application cache

# Игра "Word scramble"



# План действий

- Необходимые переменные
  - Текущее слово
  - Время для ответа
  - Оставшееся время для ответа
  - Ссылка на устанавливаемый таймер
  - Счётчик угаданных слов
  - Счётчик ошибок
  - Строковые переменные
- Необходимые функции
  - Инициализации. Запускается после загрузки страницы
    - Выборка необходимых элементов
    - Назначение необходимых обработчиков событий
  - Нового слова. Запускается при нажатии на кнопку "**Играть**"
    - Проверка, а есть ли ещё слова для игры?
    - Обнуляются и инициализируются все необходимые переменные
    - Выборка случайного слова из списка слов
    - Перемешивание выбранного слова
    - Отмена нажатия на кнопку
    - Запуск таймера
  - Исполнения таймера
    - Изменение счётчика оставшегося времени
    - Если время вышло, остановка таймера и восстановление возможности нажатия на кнопку
  - Обработчика события **onkeyup** текстового поля
    - Выбор текущего значения текстового поля
    - Сравнение текущего значения текстового поля с оригинальным словом
    - В случае успеха, изменение счётчика угаданных слов и остановка таймера

# HTML5 API: Application cache

```
<html manifest="site.manifest">

CACHE MANIFEST
# Версия 1
# Файлы доступны только по сети
NETWORK:
/folder/
/some-file.html
# Файлы доступны только из кэша
CACHE:
/imgs/
/app/somefile.html
/somefile.html
# По сети доступен первый файл
# Из кэша доступен второй файл
FALLBACK:
/online.html /offline.html
```

# Что мы изучили?

- Написали небольшое клиентское приложение
- Сделали наше приложение доступным для автономного использования

# Что почитать?

- Mozilla Developer Network
- W3C Document Object Technical Reports

# Что дальше?

- JavaScript. Уровень 3а. Использование библиотеки jQuery
- JavaScript. Уровень 3б. AJAX. Разработка веб-приложений для Web
- JavaScript. Уровень 3в. Серверное программирование на Node
- JavaScript. Уровень 3г. HTML5 API