Лабораторная работа №4
Основы javaScript

Цель:

Сценарий

Ниже вы увидите несколько сценариев, но все они составлены по одной схеме: для каждой команды объект.свойство (object.property) создается переменная, затем переменные помещаются в команду document.write() для вывода.

Свойства объекта navigator

```
<html>
<head></head>
<body>
<script>
var an = navigator.appName;
var av = navigator.appVersion;
var acn = navigator.appCodeName;
var ua = navigator.userAgent;

document.write("Вы пользуетесь <B>" +an+ "</B>, версия " +av+ ".<BR>Кодовое название "
+acn+ ", заголовок " +ua+ "." );
</script>
</body>
</html>
```

!!! Текст в скобках должен быть весь на одной строке.

Объект navigator имеет четыре свойства. Обратите внимание на заглавные буквы!

- appName сообщает название браузера, например, Explorer.
- appVersion сообщает версию браузера и платформу, на которой он работает.
- appCodeName сообщает кодовое имя, данное браузеру, например, Netscape называет свой браузер Mozilla.
- userAgent сообщает версию используемого браузера.

Зная браузер пользователя и его версию, можно дать команду: "Если браузер такой-то, сделать то-то."

Задание 1: Создать приведенный пример скрипта в составе HTML-документа. При этом оформить операторы скрипта в виде функции, описанную в заголовке HTML-документа, а вызов функции организовать через кнопку в теле HTML-документа.

Свойства объекта document

```
<html>
<head></head>
<body>
<script>
var bgc = document.bgColor;
var fgc = document.fgColor;
var lc = document.linkColor;
var al = document.alinkColor;
var vlc = document.vlinkColor;
var url = document.location;
var ref = document.referrer;
var t = document.title;
var lm = document.lastModified;
document.write("Цвет фона этой страницы <B>" +bgc+ "</B>.");
document.write("<BR>Цвет текста этой страницы <B>" +fgc+ "</B>.");
document.write("<BR>Цвет ссылок этой страницы <B>" +lc+ "</B>.");
```

```
document.write("<BR>Цвет активной ссылки этой страницы <B>" +al+ "</B>.");
document.write("<BR>Цвет посещенной ссылки этой страницы <B>" +vlc+ "</B>.");
document.write("<BR>URL этой страницы <B>" +url+ "</B>.");
document.write("<BR>До этого вы были на странице <B>" +ref+ "</B>.");
document.write("<BR>Заголовок этой страницы <B>" +t+ "</B>.");
document.write("<BR>Последние изменения в документ внесены: <B>" +lm+ "</B>.");
</script>
</body>
</html>
```

Код работает в IE, в других браузерах требует использования `document.bgColor.valueOf()`, а также перед выводом явного назначения свойств документа, например, `document.bgColor="#110011"`.

Обратите внимание! Цвета можно задавать через указание зарезервированных слов, например, для светлосерого `document.bgColor=LightGray`. Полная таблица цветов и их зарезервированных названия в файле: ***названия цветов в виде литералов.png***

Ниже перечислены некоторые свойства:

- bgColor возвращает шестнадцатеричный код цвет фона.
- fgColor возвращает шестнадцатеричный код цвета текста.
- linkColor возвращает шестнадцатеричный код цвета ссылки.
- alinkColor возвращает шестнадцатеричный код цвета активной ссылки.
- vlinkColor возвращает шестнадцатеричный код цвета посещенной ссылки.
- location возвращает URL страницы.
- referrer сообщает, с какой страницы пришел пользователь. Если информация недоступна, то возвращается пустое место.
- title возвращает заголовок документа HTML, т .е. текст между командами TITLE.
- lastModified сообщает дату, когда были внесены последние изменения в страницу (на самом деле дату, когда страница была загружена на сервер или сохранена последний раз на жестком диске).
- cookie (не показано) возвращает текстовый файл cookie.
- anchors (не показано) возвращает количество анкеров HREF на странице.
- forms (не показано) возвращает массив (список) объектов формы на странице.
- links (не показано) возвращает количество всех отдельных ссылок.

Задание 2: Создать приведенный пример скрипта в составе HTML-документа. При этом оформить операторы скрипта в виде функции, описанную в заголовке HTML-документа, а вызов функции организовать через кнопку в теле HTML-документа.

Индивидуальное задание 1:

Вариант 1: Используя предыдущий пример и оператор if сделайте автоматическую подстройку цветовых свойств страницы в зависимости от текущего времени. Для утренних часов сделать бежевую гамму, для дневных голубую или желтую, для вечерних серую, для ночных – синюю.

Вариант 2: Используя предыдущий пример и оператор if сделайте автоматическую подстройку цветовых свойств страницы в зависимости от текущего времени года, которое автоматически определяется по текущей дате. Для зимы сделать бело-голубую гамму, для весны – оранжевую/зеленую, для лета – зеленую/желтую, для осени – красную/серую.

Вариант 3: Используя предыдущий пример и оператор if сделайте автоматическую

подстройку цветовых свойств страницы в зависимости от декады месяца. Для первой декады сделать красную/фиолетовую гамму, для второй декады - бежевую, для третьей - коричневую.

Вариант 4: Используя предыдущий пример и оператор if сделайте автоматическую подстройку цветовых свойств страницы в зависимости от дня месяца и первой или второй половины дня. Для четных дней первой половины дня сделать голубую гамму, для четных дней второй половины дня - желтую, для нечетных дней первой половины дня – зеленую, для четных дней второй половины дня – фиолетовую.

Свойства объекта history

```
<html>
<head></head>
<body>
<SCRIPT LANGUAGE="javascript">
    var h = history.length;
    document.write("До этого вы посетили " + h + " страниц.");
</SCRIPT>
</body>
</html>
```

Объект history позволяет переместиться на одну или несколько страниц вперед или назад.
Объектом является журнал посещений history. Это список страниц, которые посетил браузер во время работы. Список истории позволяет реализовать кнопку "Назад" и просмотреть еще раз любую страницу.
Также объект имеет свойство length (протяженность). Оно позволяет определить в сценарии количество страниц в папке "history".
Существует также метод go() (пойти), который позволяет передвигаться по history.length с указанным шагом.

Задание 3: Создать приведенный пример скрипта в составе HTML-документа. При этом оформить операторы скрипта в виде функции, описанную в заголовке HTML-документа, а вызов функции организовать через кнопку в теле HTML-документа.

Два свойства объекта location

```
<SCRIPT LANGUAGE="javascript">
var hst = location.host;
document.write("Страница находится на <B>" + hst + "</B>.");

var hstn = location.hostname;
document.write("Страница находится на <B>" + hstn + "</B>.");
</SCRIPT>
```

Выше представлены два свойства объекта location: host, и hostname. Команды равноценны, так как выполняют одну и ту же задачу — сообщают URL в текстовом формате или адрес IP, в зависимости от сервера. Но... location.host сообщает URL плюс "порт", с которым соединен пользователь. location.hostname сообщает только URL.
Если вы получаете одинаковый результат от обеих команд, значит, ваш сервер не соединил вас со специальным портом. Говоря техническим языком, свойство "порта" — null.
Кстати, эти две команды не работают, если просматривать страницу с жесткого диска. Результат может быть только в том случае, если она размещается на сервере, так как сценарию требуется URL для анализа.

Существует множество других свойств, с которыми вы встретитесь во время уроков. Здесь даны общие представление о свойствах — как они используются и что делают наиболее часто используемые.

Задание 4: Создать приведенный пример скрипта в составе HTML-документа. При этом оформить операторы скрипта в виде функции, описанную в заголовке HTML-документа, а вызов функции организовать через кнопку в теле HTML-документа.

Индивидуальное задание 2:

Вариант 1: Используя одну из вышеприведенных команд типа объект.свойство, напишите скрипт, который создает ссылку на страницу документа HTML на каком-либо сервере. Например, если вы находитесь на www.izi.vlsu.ru, то скрипт должен создать ссылку на www.izi.vlsu.ru/index.html. Привязать этот код к кнопке.
Вариант 2: Используя одну из вышеприведенных команд типа объект.свойство, напишите скрипт, который создает ссылку на страницу документа HTML на каком-либо сервере. Например, если вы находитесь на www.izi.vlsu.ru, то скрипт должен создать ссылку на домен www.vlsu.ru/index.html. Привязать этот код к кнопке.

Вариант 3: Используя одну из вышеприведенных команд типа объект.свойство, напишите скрипт, который создает ссылку на страницу документа HTML на каком-либо сервере и покажет номер порта. Например, если вы находитесь на www.izi.vlsu.ru, то скрипт должен создать ссылку на www.izi.vlsu.ru/index.html и отразить порт. Привязать этот код к кнопке.

Вариант 4: Используя одну из вышеприведенных команд типа объект.свойство, напишите скрипт, который создает ссылку на страницу документа HTML на каком-либо сервере и покажет номер порта. Например, если вы находитесь на www.izi.vlsu.ru, то скрипт должен создать ссылку на домен www.vlsu.ru/index.html и отразить порт. Привязать этот код к кнопке.

Также, какое бы свойство ни использовалось, присвойте ему имя переменной. Имейте в виду, что страница должна находиться на сервере, на жестком диске сценарий не работает, так как там нет никакого location.host.

Общая иерархия объектов

- Window
  - Parent
  - Self
    - Location
      - Href
    - Document
      - Image
        - Src
      - Form
        - Text
        - Submit
        - Checkbox
  - Top
  - Frames

Результат действия иерархии

Все ссылки начинаются с наивысшего объекта, window (окно браузера), и идут по

нисходящей. Окна и рамки (frames) принадлежат объекту window. На них не нужно ссылаться, если только их не больше одного. Top, self, parent и frames — "встроенные" имена для окон.
Вот несколько примеров. Обратите внимание на иерархию.

```
document.mypic.src = "pic1.gif"
```

в самом начале window не требуется. Предполагается, что это все и так находится внутри окна. Команда document.mypic.src указывает на изображение с именем mypic, и изменяет его содержимое на "pic1.gif". В данном случае document (документ) — это страница на которой находится элемент, mypic — имя элемента, а SRC — источник элемента ("pic1.gif").

```
document.write(location.href)
```

write( ) — это метод объекта document. location.href содержит полный URL окна. Обратите внимание, что location и document находятся на одном уровне. Это значит, что вы получаете адрес документа того же уровня.

Разбор иерархии объектов

- Window
  - Parent
  - Self
    - Location
      - Href
    - Document
      - Image
        - Src
      - Form
        - Text
        - Submit
        - Checkbox
  - Top
  - Frames
- Самая большая путаница в том, что некоторые объекты также являются и свойствами.
- window — только объект.
- document является свойством окна, но в свою очередь и объектом.
- form — это свойство документа, но также и объект со своими свойствами!
- value (значение) и SRC (источник) — только свойства!
- Здесь представлены не все объекты и свойства. Однако этого достаточно, чтобы понять концепцию в целом... Все ссылки начинаются сверху от window и идут по нисходящей. То есть, нельзя написать document.mytext.myform или mypic.src.document. Это неправильный порядок, следует писать слева направо от более общего к более конкретному.
- Важное замечание: чтобы показать содержимое поля формы, необходимо использовать свойство value (значение), например, document.myform.mytext.value! Если написать просто document.myform.mytext, то будет получена информация о поле формы, но не о его содержании!

Считайте value ("значение") некоторым показателем того, что нечто имеется или отсутствует в определенное время. Поле с флажком может иметь значение "on" или "off", в зависимости от того, задан он или нет. Текстовое поле может иметь значение "hidden" (скрытое), если вы не хотите, чтобы пользователь его видел. Текстовое поле, как указано выше, может содержать какую-то запись. Она будет значением этого поля.

Обработчика событий

Команды последействия: onUnLoad и onMouseOut

Это два обработчика событий, которые необходимо иметь в своем арсенале: onMouseOut и onUnload (обратите внимание на заглавные буквы). Они рассматриваются в одном уроке, потому что начинают действовать после того, как что-то сделано.
onMouseOver вызывает некое событие, если навести мышь, к примеру, на ссылку. В противоположность ей onMouseOut начинает действовать, если курсор увести со ссылки. Мы также знаем, что команда onLoad запускает сценарий, когда страница загружается. Команда onUnload действует, когда пользователь уходит со страницы.

Сценарий

Следующий код использует события при перемещении указателя мыши:

```
<html>
<head>
    <title>Пример обработки событий мыши</title>
</head>
<body>
<A HREF="index.htm" onMouseOver="window.status='Эй! Убирайся с меня!';
return true"
onMouseOut="window.status='Так-то лучше, спасибо'; return true">
Наведите курсор на эту ссылку и сместите в сторону</A>
</body>
</html>
```

Задание 5: Создать приведенный пример, если статусы не отображаются в браузере, то исправить настройки браузера так, чтобы статус окна заработал.

Индивидуальное задание 3:

Вариант 1: Используя предыдущий пример, изменить действия при срабатывании событий мыши. А именно, при наведении на надпись – изменять цветовую гамму окна в темносинюю, при уводе курсора с надписи – возвращать прежнюю цветовую гамму окна.

Вариант 2: Используя предыдущий пример, изменить действия при срабатывании событий мыши. А именно, при наведении на надпись – вывести в цикле сообщение с просьбой ввести ключевое слово, когда ключевое слово (выбрано вами самостоятельно) будет введено, то выйти из цикла, при уводе курсора с надписи – выводить сообщение с помощью alert( ) «Курсор вне зоны надписи».

Вариант 3: Используя предыдущий пример, изменить действия при срабатывании событий мыши. А именно, при наведении на надпись – вывести в цикле сообщение с просьбой ввести результат умножения двух случайно определенных чисел в диапазоне от 1 до 10, если результат ввели правильно, то выйти из цикла, при уводе курсора с надписи – выводить сообщение с помощью alert( ) «Курсор вне зоны надписи».

Вариант 4: Используя предыдущий пример, изменить действия при срабатывании событий мыши. А именно, при наведении на надпись – вывести в цикле сообщение с просьбой ввести остаток от деления двух случайно определенных чисел в диапазоне от 1 до 10, если результат ввели правильно, то выйти из цикла, при уводе курсора с надписи – выводить сообщение с помощью alert( ) «Курсор вне зоны надписи».

Использование команды onUnload при уходе со страницы:

```
<html>
<head>
```

```
    <title>Untitled Page</title>
</head>
<BODY onUnload="alert('Уже уходите?')">

</body>
</html>
```

Задание 6: Создать приведенный пример, проверить работу.

Разбор сценария

Эффекты с мышью, как легко видеть, создаются с помощью команд onMouseOver и onMouseOut.

Обратите внимание, что эти две команды никак не связаны между собой. Вам не нужно, чтобы эти события происходили одновременно. Требуется, чтобы одно событие происходило, когда курсор мыши указывает на ссылку, а другое — когда курсор мыши смещается со ссылки. Поэтому нужно писать их как две совершенно независимые команды, каждая из которых содержит свою команду return true.

Сообщение при уходе со страницы создается с помощью команды onUnload="alert('Уже уходите?')", которая добавлена в строку BODY документа HTML. !!! Обратите внимание на двойные и одинарные кавычки. Внутри двойных — одинарные. Вторая двойная кавычка означает для браузера конец команды.

Пример

В этом задании используются onUnload, onMouseOver, и onMouseOut:

- Создайте страницу с гипертекстовой ссылкой.
- Когда курсор указывает на ссылку, в строке состояния должны появляться слова: "Привет, пользователь 'название браузера'!".
- Когда курсор уходит со ссылки, в строке состояния должен появляться текст: "Не скучаете у нас на 'URL страницы'?"
- Если щелкнуть на ссылке, должно появиться окно со словами: "Уже уходите? Сейчас всего только 'текущее время'";
- Время должно определяться с помощью функции.
- Не пользуйтесь командой onClick, чтобы вывести окно сообщения, возьмите команду onUnload.

Пример кода

```
<html>
<HEAD>
<SCRIPT>
  function goodbyedate()
  {
  var d = new Date();
  var h = d.getHours();
  var m = d.getMinutes();

  var t = h + ':' + m + ' ';

  alert
    (" Уже уходите?  Сейчас всего только " + t + ".");
  }
  </SCRIPT>
</HEAD>

<body bgcolor="FFFFCC" onUnload="goodbyedate();">

<A HREF="index.html"
  onMouseOver="window.status=
```

```
    'Привет, пользователь ' +navigator.appName;
     return true"
   onMouseOut="window.status=
    'Не скучаете у нас на ' +document.location+ '.';
     return true"> Проведите курсор мыши над этой ссылкой</A>

</BODY>

</html>
```

Задание 7: Создать приведенный пример, проверить работу. Функция, определенная в заголовке (между командами <head>) задает время. К этой функции обращается команда onUnload в строке <body> документа HTML. Переменная времени используется в команде alert. Команды onMouseOver и onMouseOut построены по той же схеме, что и в уроке, кроме переменных navigator.appName в onMouseOver и document.location в onMouseOut. Каждая из команд window.status помещена в двойные кавычки. Отрезки текста стоят в одинарных кавычках.

Открытие новых окон

window.open - в нем window (окно) — объект, а open (открыть) — метод, который на него воздействует.

```
<html>
<head>
    <title>Untitled Page</title>
</head>
<body>
<SCRIPT type="text/javascript" >

    window.open('example.html', 'Window_name1', config = 'height=300,width=300')

</SCRIPT>

</body>
</html>
```

Задание 8: Создайте скрипт, представленный выше. Выполните, обратите внимание на ошибку. Теперь создайте еще страницу с произвольным содержанием, но с названием example.html. Запустите предыдущий скрипт повторно. Проверьте, с какими параметрами откроется новое окно в браузере.

Обратите внимание! Представленный здесь сценарий будет только открывать окно. В этом окне выводится документ example.html.

Расположение на странице

Начнем с расположения сценария на странице. До сих пор всегда говорилось, что лучше помещать скрипты выше в документе, чтобы они быстрее загружались в память компьютера и начинали работать без задержки. Когда речь идет о функции, сценарий помещается между командами <HEAD>. Здесь будет сделано другое предложение.

Если вы собираетесь открывать новое окно, поместите команды, которые это делают, ближе к концу документа HTML. Проще говоря, пусть они идут в последнюю очередь. Причина простая: сначала загрузится страница, а потом откроется окно. Если команды стоят в начале, то окно появится прежде, чем пользователь увидит страницу. Скорее всего он закроет новое окно, прежде чем его можно будет использовать.

Конфигурация нового окна

Общий формат открытия окна:

('URL документа в окне', 'Название нового окна',
config='параметры нового окна')
В рассматриваемом случае мы имеем:
('example.html', 'Window_name1', config='height=300,width=300')

- example.html— это URL страницы, которая появится в новом окне. Если страница располагается на другом сервере, то необходимо добавить http:// и так далее.
- Window_name1 — пример имени нового окна.
- config= указывает, что следующие команды относятся к конфигурации нового окна.

Команды config

Приведенные выше команды config открывают новое окно размером 300 на 300 пикселей. Обратите внимание, что команды height (высота) и width (ширина) разделены только запятой без пробелов, а значения поставлены в одинарные кавычки, так как эти два элемента являются подкомандами config и должны выполняться совместно. Пробел для браузера означает конец команды. Ошибка.
Для команды config существует множество подкоманд. Про высоту (height) и ширину (width) мы уже знаем, они определяются в пикселях. Остальные подкоманды употребляются со словами "yes" или "no" в зависимости от того, нужны ли в новом окне эти элементы. (Можно ставить "1" вместо "yes" и "0" вместо "no", но это не обязательно.)
Помните, никаких пробелов между подкомандами и используйте одинарные кавычки. Пробел равносилен ошибке.

- toolbar= отвечает за наличие панели инструментов во вновь открытом окне. Панель инструментов в верхней части окна браузера содержит кнопки НАЗАД, ВПЕРЕД, СТОП и т.д.
- menubar= отвечает за наличие строки меню с элементами ФАЙЛ, ПРАВКА, ВИД и т.д.
- scrollbars= отвечает за наличие полосы прокрутки.
- resizable= указывает, может ли пользователь изменить размер окна по своему желанию.
- location= отвечает за наличие адресной строки во вновь открытом окне, в которой выводится URL страницы.
- directories= отвечает за наличие строки каталогов в новом окне, которая содержит закладки и т.п.
- status= отвечает за наличие строки состояния.

От строки с заголовком избавиться невозможно, хотите вы этого или нет.
Может быть, вы считаете, что все вышеперечисленное — свойства. Нет. Если вам проще их запомнить, считая свойствами, — отлично, считайте их чем угодно. Но в действительности они называются характеристиками или атрибутами. Они действуют как параметры события JavaScript.

```
<html>
<head>
    <title>Untitled Page</title>
</head>
<body>
<SCRIPT type="text/javascript" >

window.open('example.html', 'Window_name1', config =
'height=300,width=300,toolbar=1,menubar=1,scrollbars=1,resizable=1,location=1,directories
=1,status=1')

</SCRIPT>
```

```
</body>
</html>
```

Задание 9: Создайте скрипт, представленный выше. Проверьте, с какими параметрами откроется новое окно в браузере, если поочередно заменять значения атрибутов с 1 на 0 или использовать yes/no.

Тэги в новом окне

Например, чтобы открыть главную страницу INTUIT в основном окне, надо поместить на ней следующий код:

```
<A HREF="http://www.intuit.ru" TARGET="main window"></A>
```

Основное окно всегда имеет по умолчанию имя "main". Поэтому в команду HREF документа HTML добавляется просто команда TARGET=" " с указанием main для окна, в которое должна загрузиться страница.

А если надо, чтобы страница загрузилась в новом окне? У этого окна, как было сказано выше, тоже есть имя, здесь оно названо Window_name1. Необходимо написать просто команду ссылки HREF с указанием окна Window_name1.

Можно открыть на самом деле несколько окон, добавляя несколько команд window.open. Надо только задать окнам различные имена. Можно создавать также ссылки между окнами, указывая необходимые имена окон.

Закрытие окна

Можно создать также в документе ссылку, которая будет закрывать окно. Вот как это делается:

```
<A HREF="" onClick="self.close()">Щелкните, чтобы закрыть</A>
```

Это обычная ссылка HREF, которая никуда не ведет. Задание ссылки таким образом позволяет избежать загрузки страницы. Закрывает окно команда onClick="self.close()".

self (само, себя) — это свойство может относиться к любому объекту. В нашем случае это свойство окна. Команда close (закрыть) закрывает окно.

Открытие окна по нажатию ссылки

Допустим, что требуется открыть окно по команде, а не когда пользователь заходит на страницу. Вот как это можно сделать:

```
<A HREF="les11.htm" onClick="window.open('opened.html', 'joe',
config='height=300,width=300')">Щелкните, чтобы открыть 'joe'</A>
```

Это ссылка HREF, которая направлена на саму себя. Команда onClick делает работу, а параметры содержатся в скобках ( ).

Индивидуальное задание 4:

Вариант 1: Напишите скрипт, который откроет новое окно с характеристиками:

1. размер 400 на 400 пикселей с желтым фоном страницы и содержит две ссылки:
   - одна откроет новую страницу в главном окне;
   - вторая откроет новую страницу в текущем окне.
2. страница, которая откроется в текущем окне, должна содержать ссылку, закрывающую окно.

Вариант 2: Напишите скрипт, который откроет новое окно с характеристиками:

1. размер 200 на 300 пикселей, содержит полосу прокрутки, сделайте фон страницы зеленым.
2. открытая страница содержит ссылку на izi.vlsu.ru, которая откроется в текущем окне.

Вариант 3: Напишите скрипт, который откроет новое окно с характеристиками:

1. размер 300 на 500 пикселей, содержит адресную строку и меню браузера, сделайте фон страницы голубым.
2. открытая страница содержит ссылку на intuit.ru, которая откроется в новом окне, и

на izi.vlsu.ru, которая также откроется в новом окне.

Вариант 4: Напишите скрипт, который откроет новое окно с характеристиками:

1. размер 300 на 300 пикселей, содержит меню браузера и строку состояния, сделайте фон страницы серым.
2. открытая страница содержит ссылку на intuit.ru, которая откроется в текущем окне, и ссылку, которая закроет текущее окно.

Создание и наполнение окна содержимым программно

В примере ниже создается функция, которая откроет новое окно, — причем и новое окно, и все его содержимое будет содержаться в том же документе HTML.

```
<html>
<head>
    <title>Untitled Page</title>
    <SCRIPT>
        function open1() {
            var OpenWindow = window.open('', 'newwin', config = 'height=300,width=300');
            OpenWindow.document.write("<HTML>");
            OpenWindow.document.write("<TITLE>Новое окно</TITLE>");
            OpenWindow.document.write("<BODY BGCOLOR='00ffff'>");
            OpenWindow.document.write("<CENTER>");
            OpenWindow.document.write("<font size=+1>Новое окно</font><P>");
            OpenWindow.document.write("<a href='http://www.intuit.ru' target='main'>Эта
ссылка<BR> откроется в основном окне</a><p>");
            OpenWindow.document.write("<P><HR WIDTH='60%'><P>");
            OpenWindow.document.write("<a href='' onClick='self.close()'>Эта ссылка
закроет окно</a><p>");
            OpenWindow.document.write("</CENTER>");
            OpenWindow.document.write("</HTML>");
        }
    </SCRIPT>
</head>

<body>

<button onclick="open1()">Запуск страницы в новом окне</button>

</body>
</html>
```

Задание 10: Создайте скрипт, представленный выше. Проверьте, с какими параметрами откроется новое окно в браузере.

Разбор сценария

В пример выше создаем переменную OpenWindow, под которой скрывается команда window.open(). Она выглядит следующим образом:

```
var OpenWindow=window.open("", "newwin", "height=300,width=300");
```

Формат знакомый. Единственная разница в том, что не указан URL. Пустые парные кавычки/апострофы говорят браузеру, что он должен искать в сценарии информацию о новом окне, — точно так же, как и в случае отсутствия URL в команде, которая закрывает окно. Оно бы не закрылось, если бы начала загружаться новая страница. То же самое и тут. Браузер стал бы загружать новую страницу, а не выполнять сценарий.

Теперь начинаем создавать страницу HTML, которая будет в новом окне. Вот первая строка текста:

```
OpenWindow.document.write("<HTML>");
```

Команда говорит, что строка текста должна быть записана в документ переменной OpenWindow (новое окно).

Каждая новая строка следует той же схеме. Помните: когда вы пишете HTML внутри команды document.write, вместо двойных кавычек с подкомандами ставьте одинарные.

Иначе будет ошибка.

Наконец обработчик событий в кнопке по onClick вызывает созданную функцию. Можно также запустить функцию в команде BODY в методе onLoad как `<body onLoad="open1()">`

Индивидуальное задание 5:

Вариант 1: Написать скрипт, создающий функцию, которая открывает окно. Документ, который появится в окне должен быть взять из индивидуального задания 3 по вашему варианту из лабораторной работы №3. Разумеется, добавьте еще ссылку, которая закроет окно.

Вариант 2: Написать скрипт, создающий функцию, которая открывает окно. Документ, который появится в окне должен быть взять из индивидуального задания 4 по вашему варианту из лабораторной работы №3. Разумеется, добавьте еще ссылку, которая закроет окно.

Вариант 3: Написать скрипт, создающий функцию, которая открывает окно. Документ, который появится в окне должен быть взять из индивидуального задания 2 по вашему варианту из лабораторной работы №3. Разумеется, добавьте еще ссылку, которая закроет окно.

Вариант 4: Написать скрипт, создающий функцию, которая открывает окно. Документ, который появится в окне должен быть взять из индивидуального задания 1 по вашему варианту из лабораторной работы №3. Разумеется, добавьте еще ссылку, которая закроет окно.

Работа с изображениями. Динамическое изменение изображений

В данном примере будут рассмотрены дополнительные возможности использования событий onMouseOver и onMouseOut. Как мы говорили ранее, любое событие может запускать на выполнение функцию или оператор JavaScript. Вспомните команду onLoad в теле документа HTML, которая вызывает код JavaScript из заголовка HEAD.

Представленные здесь два события происходят, когда указатель мыши перемещается на ссылку или смещается со ссылки.

Еще раз обратите внимание на то, что теги <SCRIPT> и </SCRIPT> не требуются. Обработчики событий onMouseOver и onMouseOut встраиваются в тег HTML <A HREF>. Также отметим, что для отключения вывода рамки вокруг изображения в теге <IMG SRC> включен атрибут BORDER="0".

```
<html>
<head>
    <title>Untitled Page</title>
</head>

<body>

<A HREF="http://www.intuit.ru"
onMouseOver="document.pic1.src='on.jpg'"
onMouseOut="document.pic1.src='off.jpg'">
<IMG SRC="off.jpg" BORDER=0 NAME="pic1">
</A>

</body>
</html>
```

Задание 11: Создайте пример, обратите внимание, чтобы в той же папке со скриптом были рисунки on.jpg и off.jpg. На странице выводится изображение off.jpg. Если навести на изображение указатель мыши, то изображение изменится на on.jpg. При смещении указателя мыши с изображения возвращается изображение off.jpg. Обратите внимание, что команда IMG связана с обработчиками onMouse в команде HREF через команду NAME="pic1". Это необходимо для связи команд.

Основные моменты:

1. onMouseOver и onMouseOut различают регистр. Нельзя менять заглавные и строчные буквы.

2. Так как необходимо ставить кавычки после onMouseOver= и onMouseOut=, то название файла *.jpg берется в одинарные кавычки, а не в двойные.

3. document.pic1.src следует иерархии объектов, document относится к текущему объекту документа HTML, а pic1 — это имя объекта-изображения (имя можно придумать какое угодно). src (источник) — это свойство объекта-изображения, которое указывает файл изображения.

4. В этом примере onMouseOver меняет источник изображения на on.jpg.

5. OnMouseOut заставляет объект изображение вывести off.jpg.

Динамическое изменение изображений с помощью функций

Когда нужна только одна команда JavaScript, можно включить ее в тег HTML <A HREF>. Для нескольких операторов JavaScript больше подходит функция. В реальном мире на странице часто требуется многократное изменение изображения с помощью JavaScript.

```
<HTML>
<HEAD>
  <title> Пример JavaScript </title>
<SCRIPT type="text/javascript">
    function up() {
        document.mypic.src = "on.jpg"
    }
    function down() {
        document.mypic.src = "off.jpg"
    }
</SCRIPT>
</HEAD>
  <BODY>
    <CENTER>
    <h2>Пример анимации</h2>

<A HREF="http://www.intuit.ru"
  onMouseOver="up()"  onMouseOut="down()">
<IMG SRC="off.jpg" NAME="mypic" BORDER=0></A>
  </BODY>
</HTML>
```

Задание 12: Создайте пример, обратите внимание, чтобы в той же папке со скриптом были рисунки on.jpg и off.jpg. Обратите внимание, что в сценарии созданы две функции.

```
function up() {
        document.mypic.src="up.gif"
}
function down() {
        document.mypic.src="down.gif"
}
```

Функции выполняют то же, что и команды в прошлом примере. Помните иерархию объектов? Сначала документ, потом имя, присвоенное объекту, и наконец SRC. Функции названы up() и down().

```
<HTML>
<HEAD>
  <title> Пример JavaScript </title>
<SCRIPT type="text/javascript">
function up() {
    document.mypic.src = "on.jpg"
}
function down() {
    document.mypic.src = "off.jpg"
}

function up2() {
    document.mypic2.src = "off.jpg"
}

function down2() {
    document.mypic2.src = "on.jpg"
}
</SCRIPT>
</HEAD>
  <BODY>
    <CENTER>
```

```
    <h2>Пример анимации</h2>

<A HREF="http://www.intuit.ru"
  onMouseOver="up()"  onMouseOut="down()">
<IMG SRC="off.jpg" NAME="mypic" BORDER=0></A>

<A HREF="http://www.intuit.ru"
  onMouseOver="up2()"  onMouseOut="down2()">
<IMG SRC="off.jpg" NAME="mypic2" BORDER=0></A>
  </BODY>
</HTML>
```

Задание 13: Создайте пример скрипта, выполните его. Измените его таким образом, чтобы было три ссылки в виде динамических изображений. Для первого оставьте зелено/красный круглый переключатель, для второго сделайте сине/желтый переключатель, для третьего – оранжево/ голубой переключатель.

Создание функций для обработки данных пользователя в формах

Формы всегда начинаются тегом <FORM> и заканчиваются тегом </FORM>. В этом ничего нового, простой HTML.

```
<html>
<head>
<SCRIPT type="text/javascript">
    function newcolor(color) {
        alert("Вы выбрали " + color)
        document.bgColor = color
    }
</SCRIPT>
</HEAD>
<BODY>
<p>Выберите цвет фона</p>
<FORM>
 <INPUT TYPE="button" VALUE="серый"
      onClick="newcolor('gray')">
 <INPUT TYPE="button" VALUE="светло-серый"
      onClick="newcolor('lightgray')">
</FORM>
</BODY>
</HTML>
```

Задание 14: Создайте пример скрипта, выполните его.

В скрипте при указании цвета используется литерал. Литерал является значением (VALUE), которое не изменяется. Литерал может быть числом, именем или любой случайной последовательностью чисел и имен. Помните только о том, что литерал невозможно изменить. Таким образом, следующий фрагмент сценария:

```
onClick="newcolor('lightgray')"
```

... определяет строку литерал "lightgray".

Обратите внимание: атрибут VALUE (значение) в команде INPUT не является свойством JavaScript, он выводит текст на кнопку. Он не влияет на свойства JavaScript.

Динамическое изменение содержимого форм

Будем передавать в функцию данные, которые пользователь введет в поле формы. Затем эти данные будут использованы для поиска в Google. Здесь понемногу начнем погружаться в API.

```
<html>
<head>
<SCRIPT type="text/javascript">
    function Gofindit() {
        var searchfor = document.formsearch.findthis.value;
        {
            var FullSearchUrl =
"https://www.google.ru/#hl=ru&q=" + searchfor;
            location.href = FullSearchUrl;
        }
    }

</SCRIPT>
</HEAD>
<BODY>

<FORM NAME="formsearch" action="">
Найдите в Google:
<INPUT TYPE="text" NAME="findthis" SIZE="40">
<INPUT TYPE="button" VALUE="Искать"
onClick="Gofindit()">


</FORM>

</BODY>
</HTML>
```

Задание 15: Создайте пример скрипта, выполните его. Если у вас есть доступ к Интернет, то автоматически подгрузится страница Google с результатами выполнения запроса по тем словам, которые вы введете на своей форме в поле с именем `findthis.`

В скрипте команда `https://www.google.ru/#hl=ru&q=` является фрагментом Google API для формирования простого поискового запроса. В переменную `searchfor` функции `Gofindit()` передается значение, введенное в элемент формы.

`<INPUT TYPE="text" NAME="findthis" SIZE="40">` - создается элемент формы типа «поле ввода» с именем `findthis` и размером для ввода `40.`

`<INPUT TYPE="button" VALUE="Искать" onClick="Gofindit()">` - создается элемент формы типа «кнопка» с надписью `Искать` и при нажатии на которую будет вызвана функция `Gofindit()`.

Разбор сценария

Этот сценарий снова требует четкого понимания иерархии объектов.

1. Во-первых, создаем функцию с переменной searchfor (искать) под названием formsearch, внутри элемента findthis (найти), который обладает свойством value (значение). Она будет результатом чего-то происходящего в объекте document.

2. Вторую функцию помещаем внутри первой. Видите вторую пару {фигурных скобок}?

3. Для второй функции создаем еще одну переменную FullSearchUrl, которая представляет собой адрес поисковой машины Google плюс значение переменной searchfor, полученное через команду document.formsearch.findthis.value.

4. Наконец, location.href приравнивается переменной FullSearchUrl. После выполнения функции пользователь попадет на итоговую страницу поиска.

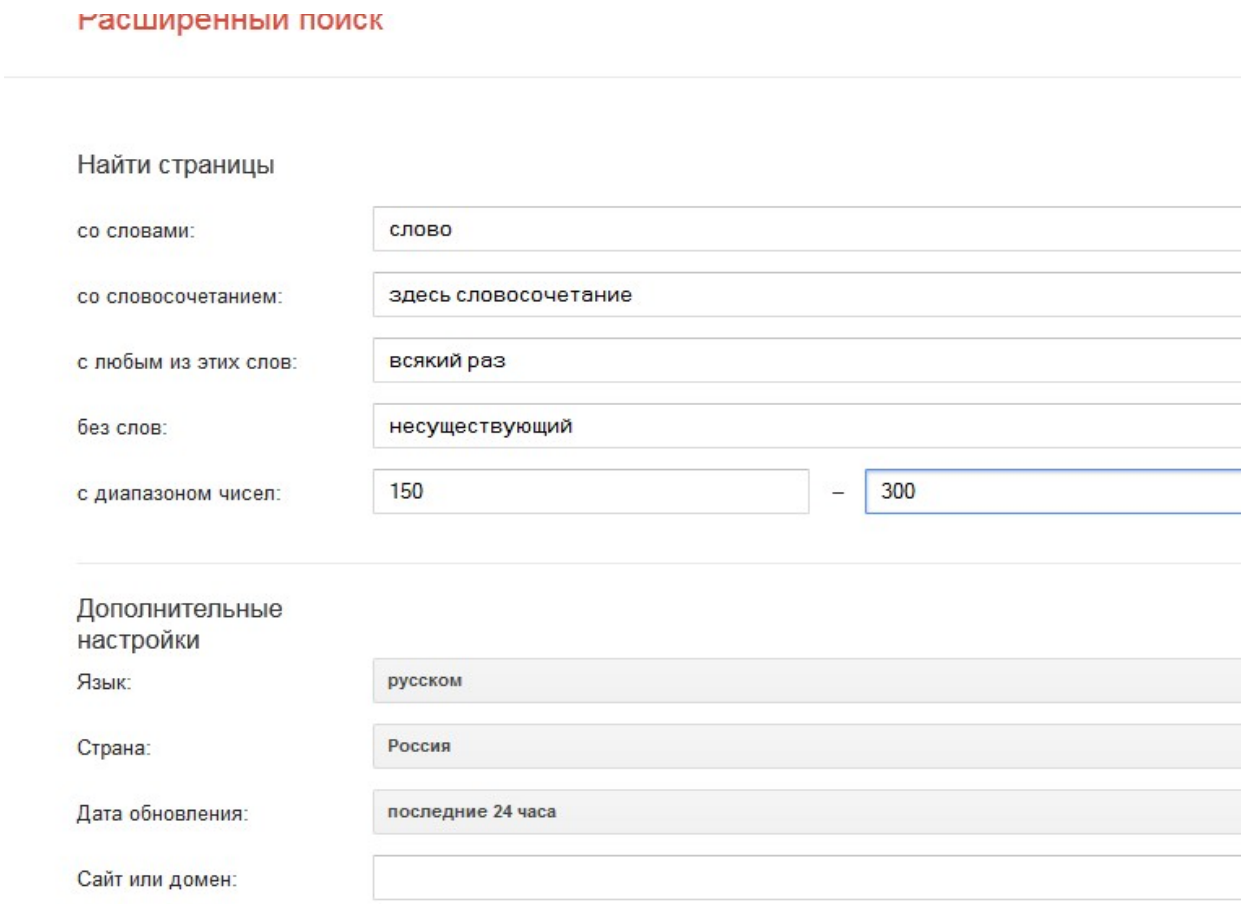
5. Теперь переходим к командам формы. Их две: текстовое поле (TEXT), куда пользователь вводит свой запрос, и кнопка, запускающая функцию.

6. Обратите внимание, что форма в целом называется formsearch. Затем для текстового поля задаем имя findthis.

7. Дальше соединяем кнопку с командой onClick, которая запускает функцию.

8. Наконец заканчиваем форму командой </FORM>. Готово.

Если в Google выбрать форму расширенного поиска, то она позволит настроить следующие поисковые элементы, например, введем следующие признаки для поиска:

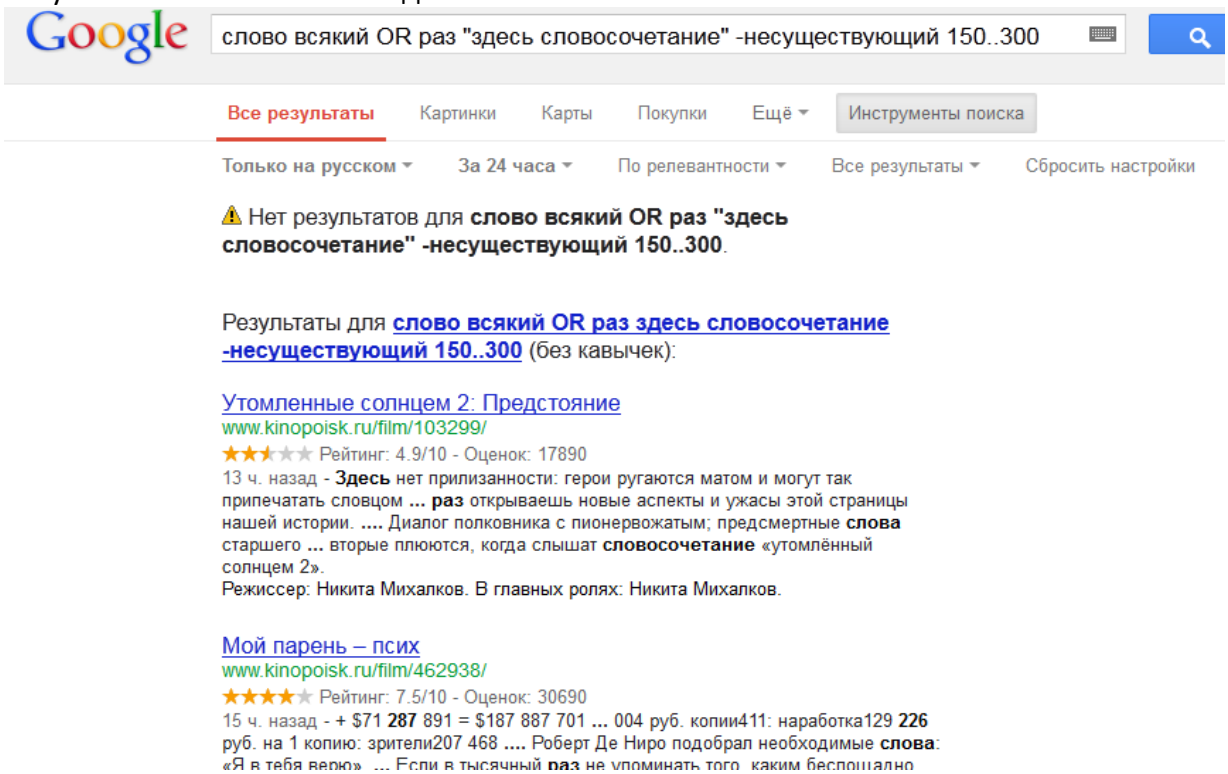


По приведенному выше примеру формы в результате будет сформирована API функция, которую можно будет внедрить в javascript:

https://www.google.com/search?hl=ru&as_q=слово&as_epq=здесь+словосочетание&as_oq=всякий+раз&as_eq=несуществующий&as_nlo=150&as_nhi=300&lr=lang_ru&cr=countryRU&as_qdr=d&as_sitesearch=&as_occt=any&safe=images&as_filetype=&as_rights=

Если проводить анализ ключевых параметров команды, то видно, что:
& - знак присоединения следующего параметра функции;
hl=ru – язык ввода русский;
as_q= …со словами;
as_epq= … со словосочетанием;
as_oq=… с любым из этих слов;
as_eq=… без слов;
as_nlo=… с диапазоном чисел – начало;
as_nhi=… с диапазоном чисел – конец;
lr=… язык ресурсов, в которых ищем;
cr=… страна – источник ресурса, в котором ищем;
и т.д.

Результат выполнения команды ниже:



Таким образом, можно менять состав команды, удалять ненужные параметры или после знака равно ничего им не писать (что будет пониматься как игнорирование параметра), например:

https://www.google.com/search?hl=ru&as_q=слово&as_epq=здесь+словосочетание&as_oq=всякий+раз&as_eq=&as_nlo=150&as_nhi=300&lr=lang_ru&cr=countryRU&as_qdr=d&as_sitesearch=&as_occt=any&safe=images&as_filetype=&as_rights=

В результате можно создать форму, аналогичную расширенному поиску в Google:

```
<html>
<head>
<SCRIPT type="text/javascript">
    function Gofindit() {
        var search_word = document.formsearch.search_word.value;
        var search_combination = document.formsearch.search_combination.value;
        var any_word = document.formsearch.any_word.value;
        var not_word = document.formsearch.not_word.value;
        var number_min = document.formsearch.number_min.value;
        var number_max = document.formsearch.number_max.value;
        var FullSearchUrl = "https://www.google.com/search?hl=ru&as_q=" + search_word +
                            "&as_epq=" + search_combination +
                            "&as_oq=" + any_word +
                            "&as_eq=" + not_word +
                            "&as_nlo=" + number_min +
                            "&as_nhi=" + number_max +
                            "&lr=lang_ru" +
                            "&cr=countryRU" +
                            "&as_qdr=d" +
                            "&as_sitesearch=" +
                            "&as_occt=any" +
                            "&safe=images" +
                            "&as_filetype=" +
                            "&as_rights=";

        location.href = FullSearchUrl;

    }

</SCRIPT>
</HEAD>
<BODY>

<FORM NAME="formsearch" action="">
   Найдите в Google:
   <br>
   со словом:          <INPUT TYPE="text" NAME="search_word" SIZE="40">
   <br>
   со словосочетанием: <INPUT TYPE="text" NAME="search_combination" SIZE="40">
   <br>
   с любым из слов:    <INPUT TYPE="text" NAME="any_word" SIZE="40">
   <br>
   без слов:           <INPUT TYPE="text" NAME="not_word" SIZE="40">
   <br>
   в диапазоне числа от: <INPUT TYPE="text" NAME="number_min" SIZE="40">
                      до: <INPUT TYPE="text" NAME="number_max" SIZE="40">
   <br>
   <INPUT TYPE="button" VALUE="Искать" onClick="Gofindit()">

</FORM>

</BODY>
</HTML>
```

Задание 16: Создайте пример скрипта, выполните его. Если у вас есть доступ к Интернет, то автоматически подгрузится страница Google с результатами выполнения запроса по тем словам, которые вы введете на своей форме. Обратите внимание! Выделенные желтым элементы поисковых параметров либо имеют уже введенное значение (например, `lr=lang_ru`), либо не имеют вообще (например, `as_sitesearch=`).

Индивидуальное задание 6:

Вариант 1: Написать скрипт, который изменит предыдущий скрипт. А именно, разобраться, как указываются в расширенном поиске английский, немецкий и французский языки, добавить поле на форму, в которое пользователь будет вводить язык, а программно по условию if будет заполняться в параметре `lr=` соответствующий литерал (например, пользователь ввел «русский», в параметр по условию подставился литерал `lang_ru`).

Вариант 2: Написать скрипт, который изменит предыдущий скрипт. А именно, разобраться, как указываются в расширенном поиске страны Украина, Китай, Англия, Германия, добавить поле на форму, в которое пользователь будет вводить название страны, а программно по условию if будет заполняться в параметре `cr=` соответствующий литерал (например, пользователь ввел «Россия», в параметр по условию подставился литерал `countryRU`).

Вариант 3: Написать скрипт, который изменит предыдущий скрипт. А именно, разобраться с параметром `as_occt=`, добавить поле на форму, в которое пользователь будет вводить один из вариантов значений для параметра, а программно по условию if будет заполняться в параметре `as_occt=` соответствующий литерал.

Вариант 4: Написать скрипт, который изменит предыдущий скрипт. А именно, разобраться с параметром `as_filetype=`, добавить поле на форму, в которое пользователь будет вводить один из вариантов значений для параметра, а программно по условию if будет заполняться в параметре `as_filetype=` соответствующий литерал.

Индивидуальное задание 7:

Вариант 1: Измените сценарий задания 15 так, чтобы он вызывал другую поисковую систему - Rambler.

Вариант 2: Измените сценарий задания 15 так, чтобы он вызывал другую поисковую систему - Yahoo.

Вариант 3: Измените сценарий задания 15 так, чтобы он вызывал другую поисковую систему - Yandex.

Вариант 4: Измените сценарий задания 15 так, чтобы он вызывал другую поисковую систему - Mail.

Подсказка: чтобы увидеть, как формируется поисковая строка соответствующего поисковика, просто зайдите на их страницу напишите простой поисковый запрос. При нажатии «найти» у вас в адресной строке получится требуемая адресная строка, которую нужно будет забрать в код, но подменить конкретное значение поискового запроса на переменную, связанную с полем TEXT вашей формы.

Отчет по лабораторной работе

В соответствии со структурой заготовки отчета и примером оформления оформить в отчете все задания, выполняемые в ходе лабораторной работы, а также индивидуальные задания по вариантам. Файл с отчетом называть по шаблону:

Фамилия_лаб_раб_номер.

Отчет предоставляется в электронном виде либо лично преподавателю, либо на электронную почту для проверки. Также по результатам лабораторной работы на следующем за ней занятии проводится выборочный опрос по командам языка.

Источник теории: http://www.intuit.ru/department/internet/jspractics/class/free/