Главное управление образования Гомельского облисполкома
Учреждение образования
«Гомельский государственный машиностроительный колледж»

Учебная дисциплина
«Веб-программирование для мобильных устройств»

Инструкция
по выполнению лабораторной работы №8
«Внедрение сценариев на языке JavaScript в веб-документ. Обработка событий с использованием DHTML. Использование управляющих конструкций при написании сценариев на языке JavaScript»

Гомель
2021

Составитель: _____ Гаврилова В.М.

Обсуждено и одобрено на заседании цикловой комиссии «Программируемые мобильные системы»

Протокол от «01» июня 2021 № 10

_____ В.М. Гаврилова

Лабораторная работа №8

Тема работы: «Внедрение сценариев на языке JavaScript в веб-документ. Обработка событий с использованием DHTML. Использование управляющих конструкций при написании сценариев на языке JavaScript»

1. Цель работы

Научить внедрять сценарии на языке JavaScript в веб-документ. Обучить обработке событий на веб-странице средствами DHTML. Углубить знания по использованию управляющих конструкций при написании сценариев на языке JavaScript.

2. Задание

1. Создать сценарий, в котором имеется функция, запускающая метод alert() с текстом «Был щелчок!», если пользователь щелкнул кнопкой мыши по странице.
2. Создать сценарий, в котором при изменении текста в текстовом поле, появится новое окно с текстом «Текст был изменен!!!».
3. Написать сценарий, в котором функция вычисляет среднюю зарплату за 3 месяца. Событие происходит при потере фокуса.
4. Написать сценарий, который определяет зарплату работнику, работающему на условиях часовой оплаты. Событие происходит при получении фокуса. Ввод данных осуществить с помощью окна prompt(). (Необходимо ввести количество часов и оплату за час.).
5. Написать программу для нахождения корней квадратного уравнения. Для вывода данных использовать метод alert(), а для обработки событий использовать onBlur .
6. Написать программу «Обмен валют».
7. Написать программу «Виртуальный светофор»: когда указатель мыши попадает на ячейку таблицы, ячейка окрашивается в соответствующий цвет
8. Изменить содержимое окна после двойного щелчка мыши.
9. Написать сценарий, в котором при наведении указателя мыши на текстовое поле ваша фамилия, имя, отчество изменяется на фамилию, имя, отчество вашей бабушки.
10. Создайте страницу с изображением и подписью под ним. При щелчке по подписи, она должна менять свой цвет.
11. Создайте кнопку, которая будет скрывать себя по нажатию.
12. Создайте страницу с изображениями автомобилей. При наведении на изображение в текстовом поле должна отображаться марка и год выпуска автомобиля.

3. Оснащение работы

Технические средства обучения:

- IBM – совместимый компьютер;

Электронные средства обучения:

- веб-браузер (Opera, Google Chrome, Mozilla Firefox и др.);
- html-редакторы (Notepad++, Sublime Text и др.).

4. Основные теоретические сведения

Dynamic HTML – это *набор технологий*, работающих на стороне клиента и призванных преодолеть статичность традиционных веб-страниц. Точнее говоря, это технологии, которые обеспечивают:

- динамическое формирование веб-страницы в процессе ее загрузки;
- динамическое изменение веб-страницы в ответ на действия пользователя.

Для достижения перечисленных целей используются следующие методы:

- динамическое изменение атрибутов и стилей элементов, составляющих HTML-документ;
- динамическое извлечение данных из внешних источников и включение их в веб-страницу;
- использование динамически загружаемых шрифтов;
- поддержка визуальных и мультимедийных эффектов при отображении страниц;
- механизмы сохранения информации на компьютере-клиенте между сессиями работы.

Объектная модель документа

DOM (от англ. *Document Object Model* – «объектная модель документа») – это независящий от платформы и языка программный интерфейс, позволяющий программам и скриптам получить доступ к содержимому документов, а также изменять содержимое, структуру и оформление документов.

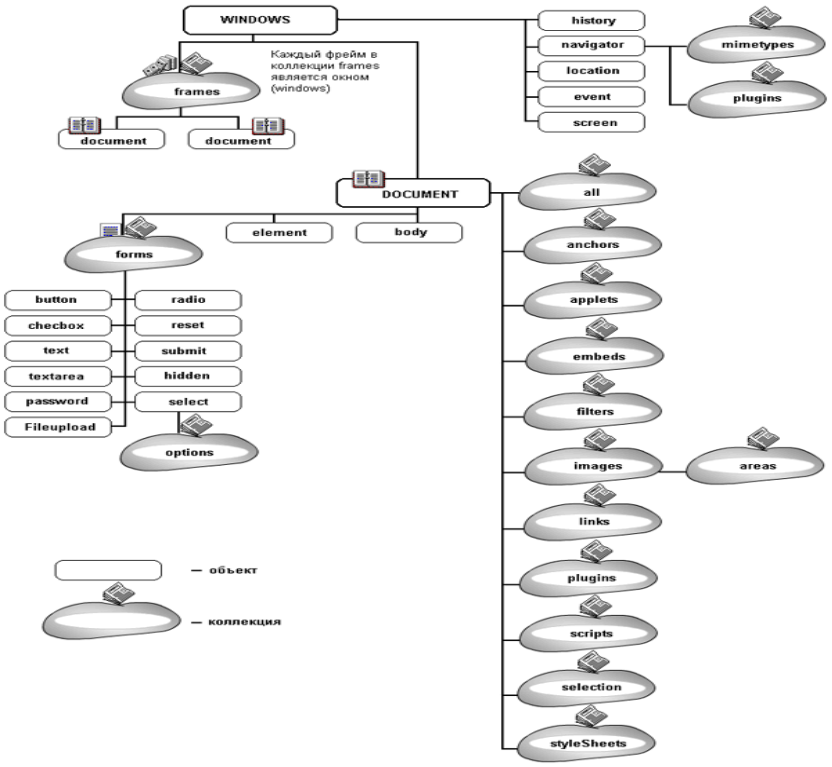
Модель DOM не накладывает ограничений на структуру документа. Любой документ известной структуры с помощью DOM может быть представлен в виде дерева узлов, каждый узел которого представляет собой элемент, атрибут, текстовый, графический или любой другой объект. Узлы связаны между собой отношениями родительский-дочерний.

Изначально различные браузеры имели собственные модели документов (DOM), не совместимые с остальными. Для того, чтобы обеспечить взаимную и обратную совместимость, специалисты международного консорциума W3C классифицировали эту модель по уровням, для каждого из которых была создана своя спецификация. Все эти спецификации объединены в общую группу, носящую название W3C DOM.

Базовая объектная модель документа

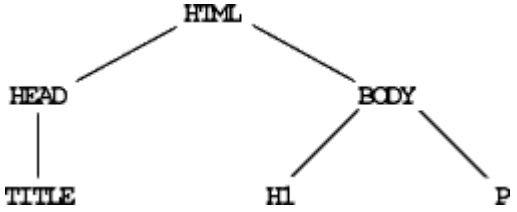
Каждый элемент дерева соответствует элементу HTML и, следовательно, имеет тег(и), содержимое и набор атрибутов. Для перехода к объектной модели документа остается сделать единственный шаг: назвать все элементы дерева объектами, а их атрибуты сделать доступными для чтения и для изменения из сценариев и аплетов. В результате дерево элементов HTML-документа становится динамически управляемым; более того, теперь мы можем легко добавлять к каждому элементу новые свойства, помимо стандартных атрибутов HTML.

Именно такой подход был положен в основу динамической модели HTML обозревателей Microsoft, а затем принят за основу стандартов W3C, получивших название *объектная модель документа* (Document Object Model или DOM). При этом W3C расширил понятие DOM на любые XML-документы, рассматривая HTML DOM как специализированный частный случай с дополнительными возможностями.



Таким образом, DOM — это модель HTML- и XML-документов, независимая от платформы и языка программирования, которая определяет:

- интерфейсы и объекты, которые используются для представления документа и манипулирования им;
- семантику этих интерфейсов и объектов, включая их атрибуты и реакцию на события;
- взаимосвязи между этими интерфейсами и объектами.



Каждый элемент представляет собой узел.

Таблица 1 - Свойства узлов

Свойство	Изменяемое	Описание
Attributes	Нет	Атрибуты узла.
childNodes	Нет	Список детей.
firstChild	Нет	Первый ребенок.
lastChild	Нет	Последний ребенок.
localName	Нет	Локальное имя в пространстве имен.
namespaceURI	Нет	URI пространства имен.
nextSibling	Нет	Следующий узел дерева.
nodeName	Нет	Имя узла.

nodeType	Нет	Тип узла.
nodeValue	Да	Значение узла.
ownerDocument	Нет	Документ — владелец узла.
parentNode	Нет	Отец данного узла.
Prefix	Да	Префикс пространства имен.
previousSibling	Нет	Предыдущий узел дерева.

Таблица 2 – Методы узлов для работы с узлами

Свойство	**Описание**
appendChild	Добавляет сына в конец списка детей.
cloneNode	Создает копию данного узла.
hasAttrbutes	Проверяет наличие у узла атрибутов.
hasChildNodes	Проверяет наличие у узла детей.
insertBefore	Вставляет новый узел перед заданным сыном.
isSupported	Проверяет свойство реализации DOM.

Окончание таблицы 2

Свойство	**Описание**
Normalize	Нормализует текстовое содержимое узла.
removeChild	Удаляет заданного сына.
replaceChild	Заменяет заданного сына новым узлом.

Таблица 3 – Методы узлов для работы с атрибутами

Метод	**Описание**
getAttribute	Возвращает значение заданного атрибута.
getAttributeNS	Возвращает значение заданного атрибута с учетом пространства имен.
getAttributeNode	Возвращает заданный узел Attr.
getAttributeNodeNS	Возвращает заданный узел Attr с учетом пространства имен.
getElementsByTagName	Возвращает список потомков, имеющих заданный тег.
getElementsByTagNameNS	Возвращает список потомков, имеющих заданный тег, с учетом пространства имен.
hasAttribute	Проверяет наличие заданного атрибута.
hasAttributeNS	Проверяет наличие заданного атрибута с учетом пространства имен.
removeAttribute	Удаляет заданный атрибут.
removeAttributeNS	Удаляет заданный атрибут с учетом пространства имен.

removeAttributeNode	Удаляет заданный узел Attr.
setAttribute	Добавляет новый атрибут.
setAttributeNS	Добавляет новый атрибут с учетом пространства имен.
setAttributeNode	Добавляет новый узел Attr.
setAttributeNodeNS	Добавляет новый узел Attr с учетом пространства имен.

События DHTML

onActivate

Наступает, когда элемент страницы получает фокус ввода.

onAbort

Наступает, когда пользователь прерывает загрузку графического изображения.

Действие по умолчанию: прерывание загрузки соответствующего файла.

onAfterPrint

Наступает сразу после вывода на принтер или предварительный просмотр текущей Web-страницы.

Событие может пригодиться, например, если требуется изменить текст или стиль печатаемого документа после распечатки.

Допустим, у нас текст написан шрифтом размером 9pt. Нам надо перед распечаткой страницы сделать текст 12pt, а после распечатки опять вернуть к 9pt.

Для этого можно написать следующий скрипт:

```
function bodyBeforePrint() {
    document.body.currentStyle.fontSize = "12pt";
}
function bodyAfterPrint () {
  document.body.currentStyle.fontSize = "9pt";
}
```

А в теге `<body>` надо записать следующее:

```
<body onBeforePrint="bodyBeforePrint()" onAfterPrint="bodyAfterPrint()">
```

onAfterUpdate

Событие наступает после того, как данные будут перенесены из элемента управления в соответствующее поле базы данных.

Доступно только для элементов управления, привязанных к данным.

Действие по умолчанию: сохранение данных. Не прерывается присвоением значения false свойству returnValue.

onBeforeCopy

Наступает перед копированием данных из текущего элемента страницы в буфер обмена Windows.

Событие onBeforeCopy() можно использовать, чтобы разрешить или запретить пункт контекстного меню Скопировать.

Для этого достаточно присвоить свойству returnValue объекта event значение false. Значение false разрешает, т.к. в этом случае мы отменяем поведение меню по умолчанию.

Присвоив значение true мы, тем самым, запрещаем пункт контекстного меню.

По умолчанию пункт Скопировать разрешен всегда, тогда как пункты Вырезать и Вставить всегда запрещены, т.к. пользователь не имеет права изменять содержимое Web-страницы.

Например, мы хотим сделать так, чтобы пользователь не мог скопировать определенный текст на странице:

```
<p onBeforeCopy="window.event.returnValue = false">
Этот текст нельзя скопировать!
```

```
</p>
```

Данное событие браузера можно использовать для защиты фрагмента страницы от несанкционированного копирования.

Если вы хотите, чтобы пользователь не смог скопировать всю страницу, надо обработчик события применить для тела документа:

```
<body onBeforeCopy="window.event.returnValue = false">
```

onBeforeCut

Наступает перед переносом данных из текущего элемента страницы в буфер обмена Windows.

Событие onBeforeCut() можно использовать, чтобы разрешить или запретить пункт контекстного меню Вырезать.

Для этого достаточно присвоить свойству returnValue объекта event значение false. Значение false разрешает, т.к. в этом случае мы отменяем поведение меню по умолчанию.

Присвоив значение true мы, тем самым, разрешаем пункт контекстного меню.

По умолчанию пункт Вырезать запрещен всегда, т.к. пользователь не имеет права изменять содержимое Web-страницы.

Например надо разрешить пользователю вырезать часть страницы:

```
<p onBeforeCut="window.event.returnValue = false">
Этот текст можно вырезать!
</p>
```

В примере мы отменили поведение Web-обозревателя по умолчанию и разрешили пункт меню Вырезать.

Надо отметить, что если вы разрешаете пользователю вырезать текст на своей странице, то не придется писать для этого дополнительный код. Web-обозреватель выполняет заданные необходимые операции самостоятельно.

onBeforedeActivate

Наступает перед потерей фокуса текущим элементом страницы.

onBeforeEditFocus

Наступает перед переходом элемента страницы в режим редактирования.

Доступно только если свойство designMode установлено в on.

Действие по умолчанию: перевод элемента страницы в режим редактирования.

onBeforePaste

Наступает непосредственно перед вставкой данных из буфера обмена в текущий элемент страницы.

Событие onBeforePaste() можно использовать, чтобы разрешить или запретить пункт контекстного меню Вставить.

Для этого достаточно присвоить свойству returnValue объекта event значение false. Значение false разрешает, т.к. в этом случае мы отменяем поведение меню по умолчанию.

Присвоив значение true мы, тем самым, разрешаем пункт контекстного меню.

По умолчанию пункт Вставить запрещен всегда, т.к. пользователь не имеет права изменять содержимое Web-страницы.

Например надо разрешить пользователю вставить текст в элемент страницы:

```
<p onBeforePaste="window.event.returnValue = false">
Вставте сюда скопированный текст!
</p>
```

В примере мы отменили поведение Web-обозревателя по умолчанию и разрешили пункт меню Вставить.

Надо отметить, что если вы разрешаете пользователю вставлять текст на своей странице, то не придется писать для этого дополнительный код. Web-обозреватель выполняет заданные необходимые операции самостоятельно.

Только есть небольшая особенность: для того, чтобы вставить текст в элемент, не предназначенный для редактирования текста (все, кроме полей ввода), пользователю придется выделить фрагмент текста, куда будет вставлен текст.

onBeforePrint

Наступает перед выводом на принтер или предварительным просмотром текущей Web-страницы.

Событие может пригодиться, например, если требуется изменить текст или стиль печатаемого документа перед распечаткой.

Допустим, у нас текст написан шрифтом размером 9pt. Нам надо перед распечаткой страницы сделать текст 12pt, а после распечатки опять вернуть к 9pt.

Для этого можно написать следующий скрипт:

```
function bodyBeforePrint() {
  document.body.currentStyle.fontSize = "12pt";
}

function bodyAfterPrint () {
  document.body.currentStyle.fontSize = "9pt";
}
```

А в теге <body> надо записать следующее:

```
<body onBeforePrint="bodyBeforePrint()" onAfterPrint="bodyAfterPrint()">
```

Действие по умолчанию: вывод на принтер или предварительный просмотр текущей Web-страницы.

onBeforeUnload

Возникает перед выгрузкой страницы при переходе на другую страницу или закрытия окна.

Действие по умолчанию: сигнализация, что Web-страница сейчас будет закрыта.

onBeforeUpdate

Наступает перед переносом данных из элемента управления в соответствующее поле базы данных.

Доступно только для элементов управления, привязанных к данным.

Действие по умолчанию: сигнализация, что данные были изменены.

onBlur

Событие происходит при переходе фокуса с этого элемента с помощью курсора мышки или последовательности перехода.

onBounce

Срабатывает, когда бегущая строка достигла границы и меняет направление.

Допускается, когда значение behavior установлено на "alternate".

Действие по умолчанию: изменение направления движения текста.

onCellChange

Событие наступает, когда изменяются данные, находящиеся в базе данных.

Действие по умолчанию: сигнализация об изменении данных в базе данных.

onChange

Событи происходит при потере управляющим элементом фокуса ввода, если его значение было изменено с момента получения фокуса.

Элемент <input>, <select>, <textarea>.

Действие по умолчанию: изменение содержимого элемента управления.

onClick

Событие происходит при однократном щелчке левой кнопки мышки на элементе, при нажатии клавиши <Enter> на форме, использовании клавиши-ускорителя или выборе пункта в списке.

Это событие полезно, если Вам надо обработать нажатие пользователя на тот или иной элемент страницы.

Пример, приведенный далее, показывает как можно обработать событие onClick():

```
<html>
<head>
<title>Событие onClick()</title>
```

```
<script>
function clickAlert(str) {
  alert("Вы нажали на "+str);
}
</script>
<body>
<table border=1>
<tr>
  <td onclick="clickAlert('левую ячейку!')">Левая ячейка</td>
  <td onclick="clickAlert('правую ячейку!')">Правая ячейка</td>
</tr>
<tr>
  <td colspan=2>
  <input type=button onclick="alert('Вы нажали на кнопку!')"
   value="Нажми для проверки работы события onClick()">
  </td>
</tr>
</table>
</body>
</html>
```

onContextMenu

Наступает при щелчке правой кнопкой мыши по элементу страницы, перед выводом контекстного меню.

Действие по умолчанию: вывод контекстного меню.

onControlSelect

Это событие наступает при попытке пользователя выбрать элемент страницы, имеющий собственный пользовательский интерфейс.

Действие по умолчанию: активизация ползунков изменения размера.

onCopy

Наступает при копировании данных из текущего элемента страницы в буфер обмена Windows.

Действие по умолчанию: копирование выделенного фрагмента в буфер обмена.

Это событие можно использовать для операций копирования, если данные, помещаемые в Буфер обмена, нужно предварительно обработать.

```
<p onCopy="funCopy();" id="id1"
  onBeforeCopy="window.event.returnValue = false;">
Скопируйте этот текст!
</p>
<script>
function funCopy() {
  window.event.returnValue = false;
  window.clipboardData.setData("Text", id1.innerText);
}
</script>
```

В вышеприведенном примере реализована специальная операция копирования текста.

Содержимое элемента id1 копируется в Буфер обмена целиком.

onCut

Событие наступает при переносе данных из текущего элемента страницы в буфер обмена Windows.

Действие по умолчанию: перенос выделенного фрагмента в буфер обмена.

Это событие можно использовать для операций вырезания, если данные, помещаемые в Буфер обмена, нужно предварительно обработать.

onDataAvailable

Наступает каждый раз, когда очередная партия данных переносится из базы данных на Web-страницу.

Действие по умолчанию: сигнализация, что данные готовы для отображения.

onDataSetChanged

Событие наступает, когда данные в базе данных изменяются, а также когда начальная порция данных готова для переноса на Web-страницу.

Действие по умолчанию: сигнализация, что данные изменились.

onDataSetComplete

Событие наступает, когда все данные перенесены из базы данных на Web-страницу.

Действие по умолчанию: сигнализация, что все данные готовы для отображения.

onDblClick

Событие происходит при двойном щелчке левой кнопкой мышки на элементе.

onDeactivate

Наступает при потере фокуса текущим элементом страницы.

onDrag

Наступает во время операции drag-n-drop в элементе-источнике.

Действие по умолчанию: вызов функции-обработчика, если такая есть.

onDragDrop

Событие наступает, когда пользователь "бросает" в окно Web-обозревателя ссылки на интернет-адреса.

Параметры:

type - тип события;

target - ссылка на элемент страницы, где оно наступило;

data - массив строк, каждая из которых содержит "брошенный" в окно адрес;

modifiers - модификаторы, указывающие, какая клавиша на клавиатуре была при этом нажата;

screenX и screenY - экранные координаты курсора мыши.

onDragEnd

Событие наступает, когда пользователь отпускает кнопку мыши, завершая операцию drag-n-drop, в элементе-источнике.

Действие по умолчанию: вызов функции-обработчика, если такая есть.

onDragEnter

Событие наступает, когда пользователь перетаскивает данные в допустимое место, в элементе-цели.

Действие по умолчанию: вызов функции-обработчика.

onDragLeave

Событие наступает, когда пользователь перетаскивает данные из допустимого места, в элементе-цели.

Действие по умолчанию: вызов функции-обработчика.

onDragOver

Событие наступает во время операции drag-n-drop в элементе-цели, когда пользователь перетаскивает данные над допустимым местом.

Действие по умолчанию: вызов функции-обработчика.

onDragStart

Событие наступает, когда пользователь начинает перетаскивать данные, в элементе-источнике.

Действие по умолчанию: вызов функции-обработчика.

onDrop

Событие наступает, когда пользователь отпускает кнопку мыши, завершая операцию drag-n-drop, в элементе-цели.

Действие по умолчанию: вызов функции-обработчика.

onError

Событие происходит при ошибке загрузки документа или графического изображения.

В обработчике этого события можно предусмотреть вывод каких-либо предупреждений о произошедшей ошибке.

Например, если адрес графического элемента задан в полном формате URL, а мы просматриваем локальную копию, не имея доступа к Internet.

onError=скрипт

Пример:

```
<html>
<head><title>Пример onError</title>
<script language="JavaScript">
function doError() {
alert ("Ошибка загрузки графики. Проверьте соединение.")
}
</script>
</head>
<body>
<img src="http://spravkaweb.ru/img/menu_all_out.gif" onError="doError()">
</body>
</html>
```

onErrorUpdate

Событие наступает, если при переносе измененных данных в базу данных происходит ошибка.

Действие по умолчанию: вызов функции-обработчика.

onFilterChange

Событие наступает, когда визуальный фильтр изменяет свое состояние или когда визуальное преобразование заканчивает свою работу.

Действие по умолчанию: сигнализация о завершении работы фильтра или преобразования.

onFinish

Событие наступает, когда очередной цикл движения текста в теге <MARQUEE> завершается.

Действие по умолчанию: прекращение цикла.

onFocus

Событие происходит при получении элементом фокуса с помощью мышки или последовательного перехода.

Действие по умолчанию: установка фокуса ввода на текущий элемент страницы.

onHelp

Возникает при нажатии клавиши F1.

Действие по умолчанию: открытие окна справки.

onKeyDown

Событие происходит при нажатии клавиши на клавиатуре.

Действие по умолчанию: возвращение кода нажатой клавиши.

onKeyPress

Событие происходит при нажатии и отпускании клавиши на клавиатуре.

Действие по умолчанию: возвращение кода нажатой клавиши в формате Unicode.

onKeyUp

Событие происходит при отпускании клавиши на клавиатуре.

Действие по умолчанию: возвращение кода нажатой клавиши.

onLayoutComplete

Событие наступает, когда Web-страница уже готова для печати на принтере или предварительного просмотра.

Действие по умолчанию: вызов функции-обработчика, если такая есть.

onLoad

Событие происходит, когда "браузер" заканчивает загружать окно или все фреймы элемента <frameset>.

Действие по умолчанию: отображение элемента или ее элемента.

onLoseCapture

Событие наступает, когда когда элемент страницы перестает перехватывать все события мыши.

Действие по умолчанию: вызов функции-обработчика, если такой есть.

onMouseDown

Событие происходит при нажатии кнопки мышки на элементе.

onMouseEnter

Событие наступает, когда пользователь помещает курсор мыши на элемент страницы.

onMouseLeave

Событие наступает, когда пользователь убирает курсор мыши с элемента страницы.

onMouseMove

Событие наступает, когда пользователь перемещает курсор мыши над элементом страницы.

onMouseOut

Событие происходит при перемещении курсора мышки за пределы элемента.

onMouseOver

Событие происходит при наведении курсора мыши на элемент.

onMouseUp

Событие происходит при отпускании кнопки мышки на элементе.

onMove

Событие наступает, когда пользователь перемещает окно Web-обозревателя или изменяет размера фрейма.

Параметры:

type - тип события;

target - ссылка на элемент страницы, где оно наступило;

X и Y - координаты левого верхнего угла окна или фрейма.

onPaste

Событие наступает при вставке данных из буфера обмена Windows в текущий элемент страницы.

Действие по умолчанию: вставка содержимого буфера обмена.

Это событие можно использовать для операций вставки, если данные, извлекаемые из Буфер обмена, нужно предварительно обработать.

Но перед этим необходимо отменить выполнение этих операций по умолчанию, присвоив false свойству returnValue объекта event.

onPropertyChange

Событие наступает при изменении одного из свойств текущего элемента страницы.

Действие по умолчанию: сигнализация об измененииее значения свойства.

onReadyStateChange

Событие наступает при изменении состояния элемента страницы.

Действие по умолчанию: сигнализация об изменении значения свойства readyState.

onReset

Событие происходит при сбросе формы.

onResize

Возникает при изменении размеров окна, фрейма или другого элемента страницы.

onResizeEnd

Событие наступает по окончания изменения пользователем размеров элемента страницы, имеющего собственный пользовательский интерфейс.

Действие по умолчанию: изменение размеров элемента страницы.

onResizeStart

Событие наступает, когда пользователь начинает изменять размеры элемента страницы, имеющего собственный пользовательский интерфейс.

onRowEnter

Событие происходит при переходе на другую запись базы данных.

Действие по умолчанию: сигнализация о готовности данных, содержащихся в новой записи.

onRowExit

Событие наступает перед переходом с текущей записи базы данных на другую.

Действие по умолчанию: сигнализация об изменении записи.

onRowsDelete

Событие наступает непосредственно перед удалением текущей записи из базы данных.

Действие по умолчанию: сигнализация об удалении записи.

onRowsInserted

Событие наступает сразу после новой записи в базу данных.

Действие по умолчанию: сигнализация о вставке новой записи.

onScroll

Возникает при скроллировании содержимого окна.

Действие по умолчанию: прокрутка содержимого элемента страницы.

onSelect

Событие происходит при выделении некоторого текста в текстовом поле.

Параметры:

type - тип события;

target - ссылка на элемент страницы, где оно наступило.

onSelection

Событие наступает, когда пользователь выделяет фрагмент содержимого страницы или поля ввода.

Действие по умолчанию: выделение фрагмента содержимого страницы или поля ввода.

onSelectionChange

Событие наступает, когда меняется тип выделения.

onSelectStart

Событие наступает, когда пользователь начинает выделять фрагмент содержимого элемента страницы.

Действие по умолчанию: выделение фрагмента содержимого элемнта страницы.

onStart

Событие наступает, когда текст в элементе <marquee> начинает двигаться.

Доступно, только если свойство loop установлено в 1.

Действие по умолчанию: начало движения. Не "всплывает" и не прерывается присвоением значения false свойству returnValue.

onStop

Событие наступает, когда пользователь прерывает загрузку текущей Web-страницы нажав кнопку Стоп или отжав ее.

onSubmit

Событие происходит при отправке формы.

Действие по умолчанию: отправка введенных в форму данных.

onUnLoad

Событие происходит, когда "браузер" удаляет документ из окна или фрейма.

Действие по умолчанию: выгрузка страницы.

Пример 1. Использование метода alert() и обработчика событий onBlur.

```
<html>
<head>
<script language="JavaScript">
<!--
function Age(form)
{
```

```
if (1*document.form1.a.value <= 18)
{
alert ("Возраст должен быть старше 18");
}
}
<!---->
</script>
</head>
<body>
<form name="form1">
Возраст: <input type="text" name="a" size=5 onBlur="Age()">
</form>
</body>
</html>
```

Пример 2. Использование обработчика событий onFocus.

```
<html>
<head>
<script language="JavaScript">
<!--//
function loan(obj)
{
var a=1*obj.a1.value;
var result=a*a;
obj.sr.value=result;
}
//-->
</script>
</head>
<body>
<form name="Form1">
<font size=3>
Сторона квадрата:  <input type="text" size=6 name="a1"><br>
Площадь квадрата:  <input type="text" size=6 name="sr" onFocus="loan(Form1)">
</form>
</body>
</html>
```

Пример 3. Использование метода confirm и метода close.

```
<html>
<head>
<script language="JavaScript">
<!--//
function exit()
{
var a=window.confirm("Хотите закрыть окно?")
if (a==true)
window.close()
}
//-->
</script>
</head>
<body>
<form name="Form1">
<font size=3>
<input type="button" value="Close" onClick="exit()">
```

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

Пример 4. Использование метода prompt.

```
<html>
<head>
<script language="JavaScript">
<!--//
function registration(obj)
{
var a=window.prompt("Введите Ваше имя"," ")
obj.regname.value=a;
}
//-->
</script>
</head>
<body>
<form name="form1">
<input type="button" value="Регистрация" onClick="registration(form1)">
<input type=text name=regname size=50>
</form>
</body>
</html>
```

5. Порядок выполнения работы

1. Создайте сценарий в соответствии с заданием.
2. Подключите сценарий к html-файле.
3. Просмотрите с помощью web-браузера созданный документ.

6. Форма отчета о работе

Номер учебной группы _____
Фамилия, инициалы обучающегося _____
Дата выполнения работы _____
Тема работы: _____
Цель работы: _____
Задание: _____
Оснащение работы: _____
Результат выполнения работы: _____

7. Контрольные вопросы и задания

1. Дать понятие DHTML?
2. Дать понятие DOM?
3. Назовите и охарактеризуйте события DHTML?

Рекомендуемая литература

1 Васильев, А.Н. JavaScript в примерах и задачах / А.Н. Васильев. – М.: Эксмо, 2018.

2 Вахтуров, В.В. JavaScript. Освой на примерах / В.В. Вахтуров. – СПб.:БХВ-Петербург, 2007.

3 Макфарланд, Д. JavaScript и jQuery. Исчерпывающее руководство / Д. Макфарланд. - М.: Эксмо, 2012.

4 Интернет-ресурс https://learn.javascript.ru/