**Контрольная работа 12**

**Тема: «Приемы программирования на JavaScript»**

**Пример 1. Программа выдачи простого предупреждения при выборе гипертекстовой ссылки. Схема URL - JavaScript**

<HTML>

<HEAD>

<title>Самый первый пример JavaScript</title>

</HEAD>

<BODY>

<center>

<h1>Мой первый пример JavaScript</h1>

<hr>

</center>

В текст этого документа внедрена гипертекстовая ссылка

<a href="javascript:window.alert('Do you speak English?')">

"Don`t click here"

</a>.

Любопытно, что будет, если все-таки выбрать?

</BODY>

</HTML>

В данном примере среди текста документа расположена гипертекстовая ссылка "Don`t click here". Если ее выбрать при просмотре (кликнуть мышкой), то на экране появится окно-предупреждение с вопросом: "Do you speak English?".

Генерация этого окна осуществляется специальным методом window.alert, который выполняется при выборе гипертекстовой ссылки. Если быть более точным, то в качестве URL информационного ресурса, который следует загрузить при переходе по данной гипертекстовой ссылке, используется схема JavaScript - расширение спецификации URI для программирования сценариев просмотра гипертекстовых документов World Wide Web. В этом случае выполнение JavaScript-программы происходит при выборе гипертекстовой ссылки, а сам код программы записан как URL.

Добиться такого же эффекта можно и другим способом, не прибегая к новой схеме URL. Для этой цели можно использовать событие, которое генерируется программой-навигатором при выборе гипертекстовой ссылки - Click.

**Пример 2. Программа выдачи простого предупреждения по событию Click при выборе гипертекстовой ссылки**

<HTML>

<HEAD>

<title>Самый первый пример JavaScript</title>

</HEAD>

<BODY>

<center>

<h1>Мой первый пример JavaScript</h1>

<hr>

</center>

В текст этого документа внедрена гипертекстовая ссылка

<a href="javascript:void(0)" onClick="window.alert('Do you speak

English?')">"Don`t click here"</a>. Любопытно, что будет,

если все-таки выбрать?

</BODY>

</HTML>

Обойтись без новой схемы URL здесь также не удается, но она используется только для того, чтобы после выбора гипертекстовой ссылки в рабочем поле программы навигатора не появлялось пустой страницы, или не приходилось загружать вновь старую страницу. Для обработки события используется конструкция onClick, которая реализует обращение к обработчику (handler) данного события, который в свою очередь вызывает выполнение кода, записанного вслед за onClick.

Размещение кода программы на JavaScript непосредственно в тагах HTML является обычным делом, но не всегда бывает удобным. Наиболее часто JavaScript-код выносят в специальный HTML-контейнер SCRIPT(</script>.....</script>). Для того, чтобы продемонстрировать применение этого подхода, видоизменим наш пример следующим образом:

**Пример 3. Применение контейнера SCRIPT для размещение JavaScript-кода**

<HTML>

<HEAD>

<title>Самый первый пример JavaScript</title>

<script language="JavaScript">

<!-- Start the text of programme

function question()

{

window.alert("Do you speak English?");

}

// -->

</script>

</HEAD>

<BODY>

<center>

<h1>Мой первый пример JavaScript</h1>

<hr>

</center>

В текст этого документа внедрена гипертекстовая ссылка

<a href="javascript:void(0)" onClick="question()">"Don`t click

here"</a>. Любопытно, что будет, если все-таки выбрать?

</BODY>

</HTML>

Данный пример развивает применение JavaScript-кода для обработки события Click. Но только в этом случае мы не размещаем весь код обработки события в атрибуте onClick. В данный атрибут помещается только вызов функции, которая будет обрабатывать это событие. Само тело функции размещено в заголовке HTML-документа внутри тагов <script ...> и </script>

Для того, чтобы защитить текст функции от интерпретации браузерами, которые не поддерживают JavaScript, мы поместили код в HTML-комментарий. При этом последняя строка этого комментария должна начинаться как комментарий JavaScript, чтобы браузер, понимающий JavaScript, не интерпретировал ее как часть JavaScript-кода.

Различные браузеры поддерживают разные версии JavaScript, что накладывает определенные ограничения при написании сценариев JavaScript. Для того, чтобы максимально адаптировать свою программу к конкретному типу программного обеспечения, часто включают проверку версии и имени программы-браузера в JavaScript.

**Пример 4.Получение типа программы просмотра HTML-страниц**

<HTML>

<HEAD>

<title>Test of Browser name</title>

</HEAD>

<BODY>

<h1 align=center>Проверка имени типа браузера;</h1>

<hr>

Для того, чтобы получить имя вашей программы просмотра

выберите кнопку "Browser"<br>

<center>

<form name=kuku>

<input type=button name=browser value=Browser

onClick="window.alert(window.navigator.appName)">

</form>

</BODY>

</HTML>

Данная программа в точности повторяет пример 2.10, но в окне предупреждения выдает имя программы просмотра HTML-страниц (window.navigator.appName). Вообще говоря, в простом сообщении о типе программного обеспечения большого смысла нет, но если вставить проверку данного имени в текст HTML-страницы и реализовать условную компиляцию страницы, то тогда обращение к данной конструкции JavaScript будет оправданным:

**Пример 5.Условная генерация текста страницы по типу программы просмотра**

<HTML>

<HEAD>

<title>Test of Browser name</title>

</HEAD>

<BODY>

<h1 align=center>Проверка имени типа браузера</h1>

<hr>

<script language=JavaScript>

<!--

if(window.navigator.appName == "Netscape")

{

document.write("<br><center><font color=steelblue size=+5>");

document.write("У вас хороший навигатор.");

document.write("</font></center>");

}

else

{

document.write("<font color=red size=+3>Необходим Netscape Navigator

версии 3.0 и выше.</font>");

window.alert("Down load new version of your browser now.");

}

// -->

</script>

</BODY>

</HTML>

В данном примере текст JаvaScript-программы размещен непосредственно в теле документа. При его загрузке, когда HTML-интерпретатор доходит до контейнера SCRIPT, вызывается JavaScript-интерпретатор. В этот момент будет проверяться условие, которое содержится в операторе if. В зависимости от результата проверки этого условия остальной текст страницы примет тот или иной вид в зависимости от типа программы просмотра.

Вообще говоря, проверить тип программы просмотра можно на сервере протокола HTTP и передать программе просмотра уже готовую страницу без условной генерации ее содержания. Но это возможно только в том случае, когда автор страницы имеет возможность программировать на машине где установлен сервер и имеет возможность администрировать этот сервер. В ряде случаев, когда место под Website арендуется и в договоре аренды нет пункта, обеспечивающего управление ресурсами сервера, в этом случае программы с условной генерацией содержания страниц бывают чрезвычайно полезными. Другой случай - это работа в локальном режиме без сервера. Здесь JavaScript является единственным средством управления просмотром. Существует еще ряд случаев, когда применение контейнера SCRIPT в теле документа является вполне оправданным, но на них мы остановимся позже в контексте решения конкретных задач управления сценариями просмотра.

Однако, чаще всего, текст JavaScript-кода размещают в виде описания функций в заголовке документа, что делает использование такого кода более удобным. Связано это с двумя моментами, которые следует учитывать при написании JavaScript-программ: область действия описания программы (из каких частей документа или страницы она видна, а следовательно, на нее можно сослаться как на функцию, и принципы интерпретации кода при просмотре документа.

Если код размещается в теле документа, то он интерпретируется только в случае переразметки отображаемого документа (загрузка, изменение параметров окна, перезагрузка). Если текст размещен в заголовке, то на него можно сослаться из любого места документа, а точнее из любого обработчика событий, которые указываются как атрибуты HTML-тагов.

**Пример 6. Размещение скрипта в отдельном файле (netscape.jsc)**

<HTML>

<HEAD>

<title>Test of Browser name</title>

</HEAD>

<BODY>

<h1 align=center>Проверка имени типа браузера</h1>

<hr>

<script language=JavaScript src=netscape.jsc>

</script>

</BODY>

</HTML>

В данном случае текст условной генерации страницы размещен во внешнем файле. При загрузке страницы этот текст докачивается программой просмотра и исполняется так же, как если бы он размещался в документе. Любопытно, что при просмотре текста документа через опцию "View Source" текст скрипта не отображается, что дает возможность скрыть его содержание от пользователя. В самом файле, который содержит конструкции JavaScript, HTML-таги не используются:

**Пример 7. Содержание файла netscape.jsc, ссылка на который установлена в атрибуте SCR тага <SCRIPT >.**

if(window.navigator.appName == "Netscape")

{

document.write("<br><center><font color=steelblue size=+5>");

document.write("У вас хороший навигатор.");

document.write("</font></center>");

}

else

{

document.write("<font color=red size=+3>Необходим Netscape Navigator

версии 3.0 и выше.</font>");

window.alert("Down load new version of your browser now.");

}

На этом можно закончить вступительную часть, посвященную примерам JavaScript-кода, и перейти к более планомерному изложению приемов программирования на JavaScript, если бы не одно "но". Дело в том, что любой автор, который собирается излагать программирование на JavaScript, встречается с проблемой постепенного наращивания сложности примеров. Материал надо начинать излагать "от печки", но вот этой самой печки нет. Все программы на JavaScript (Client-site JavaScript) - это в той или иной мере программы обработки событий, которые связаны с теми или иными информационными объектами. Без изучения этих объектов нельзя начинать ничего программировать.

Однако, существует лазейка в этом заколдованном круге, которой мы и воспользуемся. Это схема JavaScript универсального локатора ресурсов (URL). В наших примерах мы уже использовали эту схему. Она помогала нам открывать окно-передупреждение при выборе гипертекстовой ссылки и избегать перезагрузки страниц. Мы будем пользоваться этой схемой вызова JavaScript-кода до тех пор, пока не появится в нашем рассмотрении объект (контейнер HTML) с атрибутом обработки события.

После этого небольшого отступления начнем рассматривать приемы программирования на JavaScript в соответствии с иерархией объектов этого языка, начиная с самого старшего объекта и двигаясь вглубь дерева объектов: от объекта "окно" к элементам формы.

**Рекомендуемая литература**

«Информатика и образование», №8, 2000 Е. В. Давыдова.

Пайк М. «Internet в подлиннике» СПб.: BHV-Санкт-Петербург, 1996.

Перри П. Дж. «Секреты World Wide Web» Киев: Диалектика, 1996.

Шарф Д. «HTML 4.0: Справочник» СПб: Питер, 1998.

Вуд Л. «Web-графика: Справочник» СПб: Питер, 1998.