Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Филиал

«Минский радиотехнический колледж»

Учебная дисциплина «Программные средства создания Internet-приложений»

**Инструкция**

по выполнению лабораторной работы

«Создание HTML-страниц с использованием сценариев JavaScript. Использование управляющих конструкций в JavaScript»

Минск

2017

**Лабораторная работа № 17**

**Тема работы: «Создание HTML-страниц с использованием сценариев JavaScript. Использование управляющих конструкций в JavaScript»**

**1. Цель работы**

Формирование практических умений создания простейших скриптов на языке программирования JavaScript и добавления их на HTML-страницу.

**2. Задание**

Создать HTML-документ используя средства DHTML.

**3. Оснащение работы**

ПК, текстовый редактор, браузер.

**4. Основные теоретические сведения**

Сценарии (скрипты) JavaScript размещаются непосредственно в коде HTML-страницы.

Встраивание сценариев для обработки событий. Поведение каждого элемента зависит от событий, которые постоянно происходят в веб-браузере. Допустим, пользователь выполнил следующие действия: щелкнул кнопкой мыши на тексте, переместил указатель мыши и загрузил HTML-документ. Веб-браузер сам решает, как реагировать на каждое событие, однако вы можете «перехватить» отдельные события и задать для них новое поведение. В языке HTML многие теги имеют атрибуты, соответствующие различным событиям. Эти атрибуты могут содержать код языка JavaScript, который выполняется в ответ на событие, например:

<p onclick="alert('Не нажимать!');">Это простой текст</p>

Встраивание сценариев в тег <script>. В HTML есть специальный контейнерный тег <script>. Если при разборе кода HTML-документа веб-браузер встретит этот тег, то он интерпретирует последующий до закрывающего тега </script> текст как сценарий на каком-либо языке.

Язык сценария указывается с помощью атрибута type. Например, для указания языка JavaScript используется следующий код:

<script type="text/javascript">

//Здесь располагается код сценария

</script>

Этот атрибут можно и не указывать, так как значение «text/javascript» является значением по умолчанию.

Включение кода JavaScript в состав HTML-документа может осуществляться при помощи тега <SCRIPT>. Рассмотрим на примере варианты использования данного тега.

<html>

<head><title>использование javascript</title></head>

<body>

<p>обычный текст html-документа</p>

<script language="javascript"> document.write("<b>текст, созданный сценарием"); document.write("javascript</b>");

</script>

</body>

</html>

Атрибут language в данном случае можно не указывать, он по умолчанию принимает значение JavaScript.

Код JavaScript также можно разместить в отдельном файле с расширением .js и подключить его к основному HTML-документу. Для этого в теге <SCRIPT> указывается атрибут src со значением, соответствующим адресу файла со сценарием JavaScript.

Например:

<SCRIPT SRC="Text.js"> </SCRIPT>

Основные правила синтаксиса:

1) **чувствительность к регистру**. Все ключевые слова пишутся в нижнем регистре. Все переменные и названия функций пишутся точно так же, как и были определены (например, переменные Str и str являются разными переменными или document.write(“Hello”)  Document.Write(“Hello”). Во втором случае будет выдана ошибка.);

2) **пробелы, табуляция и перевод строки**. Эти символы игнорируются в JavaScript, так что можно использовать их для форматирования кода с тем, чтобы его было удобно читать;

3) **символ точка с запятой (;)**. Все операторы должны быть разделены этим символом. Если оператор завершается переводом строки, то точку с запятой можно опустить. При этом нужно следить за тем, чтобы при разрыве строки одного оператора, новая строка не начиналась бы с самостоятельного оператора;

4) **комментарии**. JavaScript игнорирует любой текст расположенный между символами /\* и \*/. Также игнорируется текст начинающийся символами // и заканчивающийся концом строки;

5) **идентификаторы**. Идентификаторами являются имена переменных, функций, а также меток. Идентификаторы образуются из любого количества букв ASCII, подчеркивания (\_) и символа доллара ($). Первым символом не может быть цифра, а в версии JavaScript 1.0 не допускается использования и символа $;

6) **ключевые слова**. Ключевые слова не могут использоваться в качестве идентификаторов. Ключевыми словами являются: break, case, continue, default, delete, do, else, export, false, for, function, if, import, in, new, null, return, switch, this, true, typeof, with.

Для ввода и вывода данных можно воспользоваться методами браузера и загруженного в него документа.

Объект, представляющий свойства браузера, называется window (окно), а три его метода, предназначенных для ввода и вывода данных посредством диалоговых окон:

- **alert();**

**- prompt()**;

- **confirm()**.

Метод **alert()** выводит на экран модальное окно с сообщением. Модальное окно означает, что выполнение сценария и дальнейшее взаимодействие со страницей приостанавливается до тех пор, пока не закроется данное окно, в данном случае, пока не будет нажата кнопка **OK** для продолжения работы.

Синтаксис метода:

**alert(сообщение)**

*Пример:*

<script>

  alert("Hello world!");

</script>

Метод **prompt()** выводит на экран модальное окно приглашения на ввод данных пользователем.

Синтаксис метода:

|  |  |
| --- | --- |
| 1 | **var имя\_переменной = prompt(msg, defaultText)** |

где:

**имя\_переменной** – имя используемой переменной, которой будет присвоено значение возвращаемое методом prompt(),

**msg** – сообщение, которое будет показано пользователю (обычно это вопрос),

**defaultText** – строка, которая отображается по умолчанию в поле ввода, обычно второй аргумент оставляют пустым и записывают так - "":

*Пример:*

<script>

  var myName = prompt("Как тебя зовут?","");

  alert("Привет " + myName + "!");

</script>

*Пользователь должен, что-нибудь ввести и нажать****OK****, или отменить ввод нажав на****CANCEL****. Метод* ***prompt()*** *возвращает, то, что ввел пользователь - строку или специальное значение* [***null***](https://puzzleweb.ru/javascript/5_types.php#a9)*, если ввод был отменен.*

Метод **confirm()** действует очень похоже на **alert()**, выводит окно с вопросом **question** с двумя кнопками: **OK** и **CANCEL**.

<SCRIPT LANGUAGE="javascript">

confirm("Уверены, что хотите войти?")

</SCRIPT>

*Одна команда сама по себе многого не дает. Нет никакой разницы, какой ответ будет выбран - "ОК" или "ОТМЕНА". Но стоит добавить функции IF (если) и ELSE (иначе), и готовы отличные эффекты.*

*Пример:*

<SCRIPT LANGUAGE="javascript">

if (confirm("Уверены, что хотите на прошлый урок?")) {

parent.location='http://www.jsp.newmail.ru/les12.htm';

alert("Счастливого пути");

}

else {

alert("Tогда оставайтесь");

}

</SCRIPT>

*В нашем случае сonfirm предлагает альтернативу: <OK> и <Отмена>. Можно сказать, Да и Нет. Обратите внимание на скобки. После команды IF всегда идут скобки, но, как известно, команды сonfirm тоже требует скобок. Следовательно, берем две пары скобок, одна внутри другой.*

*Сразу же после этого идут команды, выполняемые при каждом варианте ответа. Обратите внимание на фигурные скобки{}. Ведь в действительности это функции. Первая из них показывает, что должно произойти, если пользователь выберет ОК (или Да).*

{

parent.location='http://www.jsp.newmail.ru/les12.htm';

alert("Счастливого пути");

}

***parent.location*** *означает ссылку. Дальше идет обыкновенная команда alert. Не забудьте про точки с запятой в конце строк.*

*если выбрать ОК, запускается функция IF (если). Отмена - другой выбор. Сразу после фигурной скобки идет команда ELSE (иначе), как бы <если нет>. И тогда следующий текст...*

else {

alert("Tогда оставайтесь");

}

*...означает: если нет, тогда вызвать окно и не менять страницу.*

*Все это вместе и дает пользователю возможность выбора: входить или не входить.*

**метод document.write()**

Метод **document.write()** выводит на страницу переданные ему аргументы.

Синтаксис метода:

**document.write(arg1,arg2,arg3,...);**

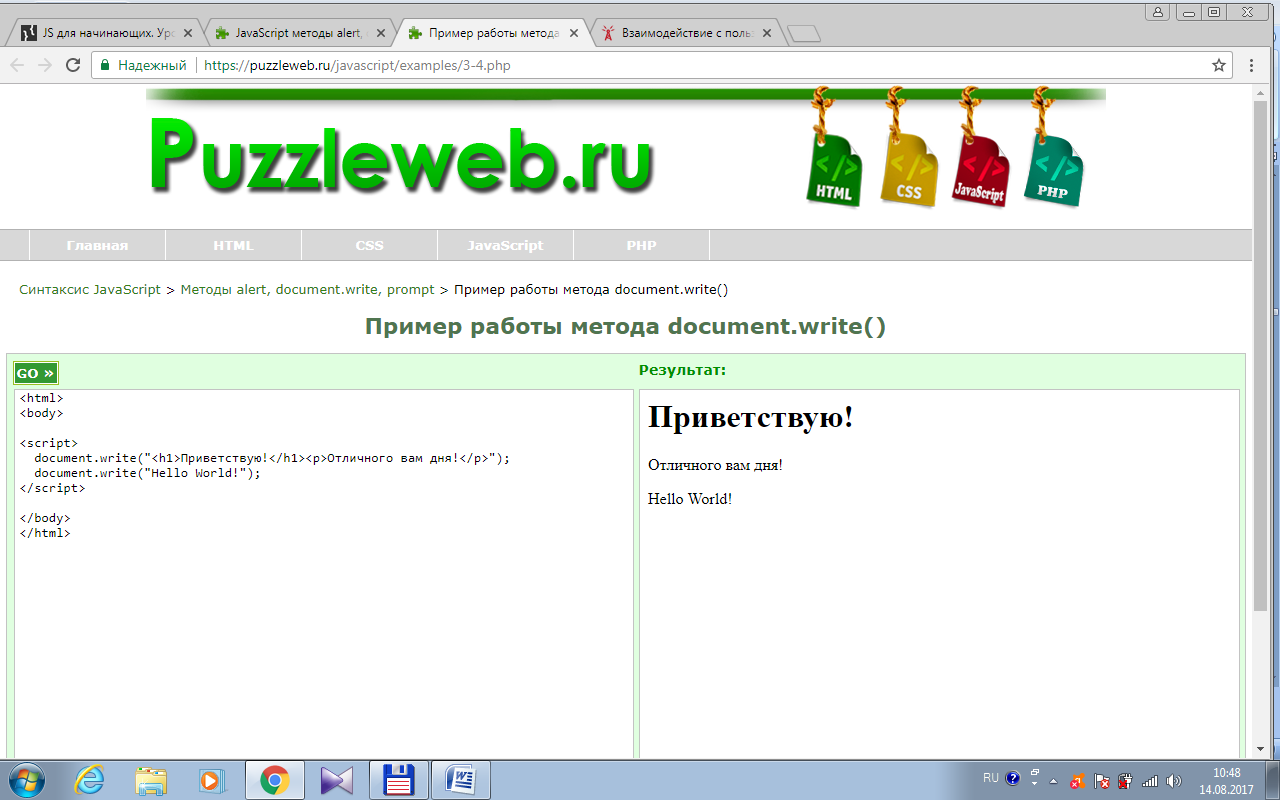
Аргументов может быть любое количество, и они могут быть любых типов, при выводе они преобразуются в строки:

<script>

  document.write("<h1>Приветствую!</h1><p>Отличного вам дня!</p>");

  document.write("Hello World!");

</script>



Метод **document.write()** работает только на этапе загрузки страницы. Если document.write() вызвать после того, как страница загрузилась, результатом будет - перезаписанная страница, с текстом, который был добавлен с помощью **document.write()**.

Также существует метод **document.writeln(str)**, который добавляет после str символ перевода строки "\n".

*Пример:*

<html>

<body>

<script>

document.write('<style> td { color: #F40 } </style>');

</script>

<table>

<tr>

<script>

document.write('<td>')

</script>

Текст внутри TD.

<script>

document.write('</td>')

</script>

</tr>

</table></body>

</html>

*Написанные вами скрипты, особенно на первых парах, далеко не всегда будут работать так, как вам хочется. Этого не нужно бояться, ошибки при написании кода совершают и опытные программисты.*

*Какого рода ошибка бы ни была, лучшим способам понять откуда она взялась, является вывод как можно большого количества информации о ней. Для этой цели не всегда подойдёт alert или document.write, поэтому во многих браузерах есть javascript-консоль, на которую и выводится информация об ошибках.*

***console.log()*** *выводит значения в консоль окна Инструментов Разработчика браузера.*

https://true-coder.ru/js-dlya-nachinayushhix/js-dlya-nachinayushhix-urok-1-6-uchimsya-ispolzovat-js-konsol.html

**Инструкцией** называют строку кода, которая эквивалентна некоторому действию, *например:*

a = 1;

alert(a);

**Комментарии** - это строки кода, которые не обрабатываются транслятором. Комментарии в JS бывают двух типов - **однострочные** и **многострочные**. Однострочные начинаются с символов “**//**” и перестают быть комментарием при переходе на следующую строку. Многострочные начинаются с символов “**/\***” и заканчиваются символами “**\*/**”. Закрывающие комментарий символы могут находиться на строке, которая ниже текущей. *Рассмотрим примеры:*

**<script>**

**// This is a single-line comment**

**a = 54;**

**/\***

**This is multi-line comment.**

**We can insert some text here.**

**\*/**

**b = a + 1;**

**</script>**

**Переменная** - это символьное представление для некоторого значения, которое сохраняется в оперативной памяти.

Переменные в JS **не являются типизированными**. Это означает, что при создании переменной не указывается, какого типа информация будет в ней находиться. Для того, чтобы использовать переменную, ее необходимо **объявить** и **инициализировать**.

Объявление переменной реализуется при помощи ключевого слова **var**.

Инициализация переменной - это присвоение переменной некоторого начального значения. Инициализация также может иметь место при объявлении переменных.

var a, b;

var x = 12;

t = 10;

В сценариях JavaScript пременные могут хранить данные любых типов: числа, строки текста, логические значения, ссылки на объекты, а также специальные величины, например "нулевое" значение null или значение NaN, которое свидетельствует о недопустимости операции.

В javaScript не требуется, чтобы переменные были объявлены явным образом, прежде чем они будут использоваться в программе. Если переменная используется без объявления, интерпретатор javaScript создает переменную в месте ее первого использования в сценарии.

**Константы** - это символьные названия некоторых значений, которые не будут изменены в программе. Правила для составления названий констант такие же, как и для переменных. Константы объявляются при помощи ключевого слова **const**:

const pi = 3.1415;

**Выражение** - это некоторая конструкция, которая состоит из значений, переменных (констант) и операторов. Рассмотрим основные операторы JS. Выделяют:

1. арифметические;
2. логические;
3. сравнения.

К арифметическим операторам относятся следующие бинарные:

**+** (сложение), **-** (вычитание), **\*** (умножение), **/** (деление), % (взятие остатка от деления).

Приоритет этих операций совпадает с оригинальным приоритетом в математике. К унарным операторам относятся следующие:

++ инкремент (увеличить значение переменной на 1), -- декремент (уменьшить значение переменной на 1), - изменение знака на противоположный.

*Унарный оператор "+" также используется для преобразования строки к числу:*

alert(+"123"+1) // 124

*В отличие от некоторых других языков, оператор "+" делает конкатенацию, если хотя бы один операнд - строка, причем, не обязательно первый.*

*Примеры использования:*

a = 1;

x = 2.75;

a++; // 2

b = -a; // -2

y = 2 + 5; // 7

z = y % 2; // 1

x = x + 10; // 12.75

x--; // 11.75

а = 23 + 5; // 28

а = “строка1”+”строка2”; // “строка1строка2”

*В некоторых случаях, присваивание и арифметическая операция могут быть объединены в одно выражение - для этого существуют специальные операторы присваивания:*

<script>

a += 10; // same as a = a + 10;

a -= 10; // same as a = a - 10;

a \*= 5; // same as a = a \* 5;

a /= 5; // same as a = a / 5;

a %= 2; // same as a = a % 2;

</script>

[Логические](http://javascript.ru/basic/operators#logicheskie): **&&** (логическое И), **||** (логическое ИЛИ), ! (логическое НЕ).

Логические операторы, как правило, используются с булевыми значениями и возвращают булево значение.

Однако, в javascript операторы && и || возвращают не булево значение, а один из операндов, определивший значение выражения. Конечно, результат в таком случае может быть и не булевым.

«Логическое И» возвращает последний операнд, если все операнды верны. Если хоть один из операндов неверен, то возвратит первый неверный операнд.

Оператор «логического ИЛИ» возвращает первое верное значение. А если верных значений вообще нет, то последнее неверное.

Логическое НЕ, также удобно для преобразования в Boolean.

Сравнения (и отношения): == (равно), != (не равно), < (строго меньше), > (строго больше),<= (меньше или равно), >= (больше или равно), === (равно и тип данных операндов совпадает), !== (не равно или не совпадает тип данных операндов).

Операции отношения имеют равный приоритет. Операции сравнения имеют равный друг относительно друга приоритет, но ниже чем приоритет операций отношения.

*Пример:*

<html>

<head>

<title>Сравнение чисел</title>

<script language="JavaScript">

var first, //первая введенная пользователем строка

second; //вторая введенная пользователем строка

// считываем первое число как строку

first=window.prompt(”Введите первое целое число:”, “0”);

// считываем второе число как строку

second =window.prompt(”Введите второе целое число:”, “0”);

document.writeln(“<h1>Результаты сравнения</h1>”);

document.writeln(“<table border=’1’ width=’100%’>”);

if (first==second)

document.writeln(“<tr><td> “+ first +”==”+ second +” </td></tr>”);

if (first!=second)

document.writeln(“<tr><td> “+ first +”!=”+ second +” </td></tr>”);

if (first > second)

document.writeln(“<tr><td> “+ first +”>”+ second +” </td></tr>”);

if (first<second)

document.writeln(“<tr><td> “+ first +”<”+ second +” </td></tr>”);

if (first<=second)

document.writeln(“<tr><td> “+ first +”<=”+ second +” </td></tr>”);

if (first>=second)

document.writeln(“<tr><td> “+ first +”>=”+ second +” </td></tr>”);

// Окончание вывода результатов

document.writeln(“</table>”);

</script>

</head>

<body>

<p> обновите страницу для перезапуска сценария </p>

</body>

</html>

В примере использовали управляющую структуру **if**, которая используется в программах, в том числе, и для принятия решений на основе того, является ли условие этой структуры истинным или ложным. *Если условие выполняется (значение истинно), то выполняется оператор, включенный в эту структуру управления. Если условие не выполняется (значение ложно), то этот оператор не выполняется*

Сравнение == делается с приведением типов, а === без приведения типов, *например:*

0 == false  // верно

// но

0 !== false  //типы разные

Оператор присвоения. Этот оператор обозначается символом = и означает сопоставление некоторой переменной конкретного значения, например:

a = 1;

x = 2.75;

type = 'Sedan';

Основные операторы в порядке убывания приоритета.

|  |  |
| --- | --- |
| Оператор | выполняемая операция |
| . | доступ к свойству или методу объекта |
| [ ] | доступ к элементу массива |
| ( ) | вызов функции |
| ++ | приращение (инкремент) |
| -- | уменьшение (декремент) |
| - | унарный минус |
| ! | отрицание (логическое NOT) |
| Delete | удаление свойств объекта |
| New | создание экземпляра объекта |
| Typeof | возвращает тип операнда |
| Void | возвращает неопределенное значение |
| \* , / , % | арифметические умножение, деление, взятие по модулю |
| + , - | арифметические сложение и вычитание |
| + | сцепление (конкатенация) строк |
| < , <= , > >= | меньше, меньше или равно, больше, больше или равно |
| == , != | проверка равенства и неравенства |
| === , !== | проверка идентичности (т.е. равенства и неравенства без преобразования типа операндов) |
| && | логическое AND (второй операнд вычисляется, если первый равен true) |
| || | логическое OR (второй операнд вычисляется, если первый равен false) |
| = | Присваивание |
| += , -= , \*= и т.д. | присваивание с операцией |

Язык JavaScript поддерживает стандартный набор управляющих конструкций:

- последовательная структура;

- три типа структур выбора: if (структура с единичным выбором); if/else (с двойным выбором) , switch (с множественным выбором);

- четыре типа структур повторения: while, do/while, for, for/in.

Любая программа может быть составлена на основе выбора нужных для реализации алгоритма программы типов управляющих структур из восьми имеющихся, соединяемых друг с другом только двумя способами: сложение управляющих структур друг за другом и вложение управляющих структур друг в друга.

*Пример последовательной структуры:*

total=total+grade;

counter=counter+1;

*Пример использования if:*

if (first==second)

document.writeln(“<tr><td> “+ first +”==”+ second +” </td></tr>”);

*если выражение в условии принимает значение true – выполняется указанное действие, если false – не выполняется.*

Кроме того, в качестве логического выражения может выступать и последовательность условных операторов объединенных с помощью логического оператора ИЛИ (||) и логического оператора И (&&).

**if ( (x>1)&&(x<5)  )**

**{document.write(«Переменная x лежит в промежутке от 1 до 5, не включая точки 1 и 5″);}**

*Пример использования if/else:*

if (first==second)

document.writeln(“<tr><td> “+ first +”==”+ second +” </td></tr>”);

else

document.writeln(“<tr><td> “+ first +”!=”+ second +” </td></tr>”);

**if ( логическое выражение) {**

**первая группа команд**

**} else {**

**вторая группа команд**

**};**

JavaScript поддерживает единственную тернарную (троичную) операцию «**?:**», которая используется в качестве сокращенной формы записи конструкции **if...else**:

document.writeln

(first==second ? “(”+ first +””==””+ second+”)”:”(+ first +”!=”+ second +)”);

Синтаксические конструкции, которые не возвращают значений, нельзя использовать в операторе '**?**'.

Для обработки сложных условий используются вложенные структуры **if...else**: когда одни структуры **if...else** помещаются внутрь других структур **if...else.**

if (age < 3) {

message = 'Здравствуй, малыш!';}

else if (age < 18) {

message = 'Привет!';}

else if (age < 100) {

message = 'Здравствуйте!';}

else {

message = 'Какой необычный возраст!';

}

var age = prompt('возраст?', 18);

var message = (age < 3) ? 'Здравствуй, малыш!' :

(age < 18) ? 'Привет!' :

(age < 100) ? 'Здравствуйте!' :

'Какой необычный возраст!';

alert( message );

*Пример использования switch:*

*представляет собой более наглядный способ сравнить выражение сразу с несколькими вариантами*

**switch(n)**

**{**

**case 1:**

**выполнить блок кода 1**

**break;**

**case 2:**

**выполнить блок кода 2**

**break;**

**default:**

**исполняемый код, если n не равно 1 или 2**

**}**

switch(x) {

case 'value1': // if (x === 'value1')

...

[break]

case 'value2': // if (x === 'value2')

...

[break]

default:

...

[break]

}

Переменная **x** проверяется на строгое равенство первому значению value1, затем второму value2 и так далее.

Если соответствие установлено – switch начинает выполняться от соответствующей директивы case и далее, до ближайшего break (или до конца switch).

Если ни один case не совпал – выполняется (если есть) вариант default.

var a = 2 + 2;

switch (a) {

case 3:

alert( 'Маловато' );

break;

case 4:

alert( 'В точку!' );

break;

case 5:

alert( 'Перебор' );

break;

default:

alert( 'Я таких значений не знаю' );

}

В **case** могут быть любые выражения, в том числе включающие в себя переменные и функции:

var a = 1;

var b = 0;

switch (a) {

case b + 1:

alert( 1 );

break;

default:

alert('нет-нет, выполнится вариант выше')

}

Несколько значений **case** можно группировать:

var a = 2+2;

switch (a) {

case 4:

alert('Верно!');

break;

case 3: // (\*)

case 5: // (\*\*)

alert('Неверно!');

alert('Немного ошиблись, бывает.');

break;

default:

alert('Странный результат, очень странный');

}

[Тип имеет значение](https://learn.javascript.ru/switch" \l "тип-имеет-значение):

var arg = prompt("Введите arg?")

switch (arg) {

case '0':

case '1':

alert( 'Один или ноль' );

case '2':

alert( 'Два' );

break;

case 3:

alert( 'Никогда не выполнится' );

default:

alert('Неизвестное значение: ' + arg)

}

*При вводе 0 выполнится первый alert, далее выполнение продолжится вниз до первого break и выведет второй alert('Два'). Итого, два вывода alert. При вводе 1 произойдёт то же самое.*

*При вводе 2, switch перейдет к case '2', и сработает единственный alert('Два').*

*При вводе 3, switch перейдет на default. Это потому, что prompt возвращает строку '3', а не число. Типы разные. Оператор switch предполагает строгое равенство ===, так что совпадения не будет.*

*Пример использования while:*

*Если инструкция* ***if*** *является базовой управляющей инструкцией, позволяющей интерпретатору JavaScript принимать решения, инструкция* ***while*** *– это базовая инструкция, позволяющая JavaScript выполнять повторяющиеся действия.*

Оператор **while** составляет тело структуры повторений: если условие истинно, то выполняются соответствующие действия; когда условие станет ложным – выполнение структуры повторения будет завершено.

*Она имеет следующий синтаксис: Цикл while (цикл с условием):*

**while (выражение){**

**инструкция};**

*Инструкция while начинает работу с вычисления логического выражения. Если оно равно false, интерпретатор JavaScript пропускает инструкции цикла и переходит к дальнейшему исполнению программы, а если true, то исполняется инструкция, образующая тело цикла, и выражение вычисляется снова.  Цикл продолжается, пока выражение не станет равно false, тогда инструкция while завершит работу и JavaScript интерпретатор пойдет дальше.*

i **= 0;**

**while (i < 10) {**

**document.write(i + «<br>»);**

**i ++;**

**} //Данный код выведет столбик из 10 цифр.**

Условие в скобках интерпретируется как логическое значение, поэтому вместо **while (i!=0)** обычно пишут while **(i)**:

**var i = 3;**

**while (i) { // при i, равном 0, значение в скобках будет false и цикл остановится**

**alert( i );**

**i--;**

**}**

**var i = 0;**

**var j = 10;**

**while (i < 100) {**

**if (i == j)**

**break;**

**i++;**

**}**

**document.write(i);**

**// Output: 10**

*Пример использования do/while:*

Цикл **do...while** выполняется по меньшей мере один раз до вычисления значения условного выражения:

**do {**

**тело цикла**

**}**

**while(условие);**

**var i = 0;**

**do {**

**document.write(i + " ");**

**i++;**

**} while (i < 10);**

**// Output: 0 1 2 3 4 5 6 7 8 9**

*В следующем примере, операторы внутри цикла выполняются один раз, даже если условие не выполняется:*

**var i = 10;**

**do {**

**document.write(i + " ");**

**i++;**

**} while (i < 10);**

**// Output: 10**

*В любой строке блока****do…while****можно использовать оператор****break****, чтобы программа вышла из цикла, или оператор* ***continue****, чтобы перейти непосредственно к выражению****while****.*

*Пример использования for:*

**for (i = 1; i < 10; i++) {**

**тело цикла**

**}**

Здесь: **for (инициализация; условие продолжения цикла; приращение)**

**for (var i = 0; i <= 9; i++) {**

**document.write (i);**

**document.write (" ");**

**}**

**// Output: 0 1 2 3 4 5 6 7 8 9**

Когда значение  условия продолжения цикла становится ложным, управление передается за пределы цикла.

Все три выражения в структуре **for** не являются обязательными.

**for (var i = 20; i >= 2; i-=2)**

*управляющая переменная изменяется от 20 до 2 с шагом -2.*

**for (var j = 2; j <= 20; j+=3)**

*управляющая переменная принимает следующие значения: 2, 5, 8, 11, 14, 17,20.*

В любой строке блока операторов цикла **for** можно использовать оператор **break**, чтобы выйти из цикла, или оператор **continue**, чтобы передать управление следующей итерации цикла.

**var j = 0;**

**for (;;) {**

**if (j >= 5) {**

**break;**

**}**

**j++;**

**document.write (j + " ");**

**}**

**// Output: 1 2 3 4 5**

*С помощью оператора****for****создается бесконечный цикл, для выхода из которого используется оператор****break****.*

**<script type="text/javascript">**

**var i=0**

**for (i=0;i<=10;i++)**

**{**

**if (i==3)**

**{**

**continue;**

**}**

**document.write("Число i равно " + i);**

**document.write("<br />");**

**}**

**</script>**

*Ключевое слово continue прерывает текущую итерацию цикла и переходит к следующей итерации.*

**<script type="text/javascript">**

**var sum=0;**

**for (var number = 2; number <= 100; number+=2)**

**sum+=number;**

**document.writeln(“<big>”Сумма четных чисел" + “от 2 до 100 равна”+ sum +”</big>”);**

**</script>**

*Оператор* ***sum*** *из тела структуры можно перенести в правый раздел заголовка структуры* ***for****, отделив его запятой:*

**for (var number = 2; number <= 100; sum+=number, number+=2)**

*Пример использования for/in:*

**var arr = ["Яблоко", "Апельсин", "Груша"];**

**for (var key in arr) {**

**alert( arr[key] ); // Яблоко, Апельсин, Груша**

**}**

**var a = [0, 1, 2, 3, 4, 5, 6], props = '';**

**for (var i in a) {**

**props += i;**

**}**

**alert(props); // В большинстве случаев 0123456**

**<!DOCTYPE html PUBLIC "?//W3C//DTD XHTML 1.0 Transitional//EN"**

**"http://www.w3.org/TR/xhtml1/DTD/xhtml1?transitional.dtd">**

**<html xmlns="http://www.w3.org/1999/xhtml">**

**<head>**

**<title>JavaScript</title>**

**<script language="JavaScript" type="text/javascript">**

**var days = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"];**

**for (var day in days) {**

**document.write(days[day] + " ");**

**}**

**</script>**

**</head>**

**(body)**

**(/body)**

**</html>**

Выйти из цикла можно не только при проверке условия но и, вообще, в любой момент. Эту возможность обеспечивает директива **break** в структурах while, for, do/while, switch.

*Например, следующий код подсчитывает сумму вводимых чисел до тех пор, пока посетитель их вводит, а затем – выдаёт:*

**var sum = 0;**

**while (true) {**

**var value = +prompt("Введите число", '');**

**if (!value) break; // (\*)**

**sum += value;**

**}**

**alert( 'Сумма: ' + sum );**

*директива break в строке (\*), если посетитель ничего не ввёл, полностью прекращает выполнение цикла и передаёт управление на строку за его телом, то есть на alert.*

*Вообще, сочетание «бесконечный цикл + break» – отличная штука для тех ситуаций, когда условие, по которому нужно прерваться, находится не в начале-конце цикла, а посередине.*

Директива **continue** прекращает выполнение текущей итерации цикла в структурах while, for, do/while. Она прерывает не весь цикл, а только текущее выполнение его тела, как будто оно закончилось.

*Например, цикл ниже использует continue, чтобы не выводить чётные значения:*

**for (var i = 0; i < 10; i++) {**

**if (i % 2 == 0) continue;**

**alert(i);**

**}**

*для чётных i срабатывает continue, выполнение тела прекращается и управление передаётся на следующий проход for.*

[Метки для break/continue](https://learn.javascript.ru/while-for#метки-для-break-continue)

Бывает нужно выйти одновременно из нескольких уровней цикла.

*Например, внутри цикла по i находится цикл по j, и при выполнении некоторого условия мы бы хотели выйти из обоих циклов сразу:*

**outer: for (var i = 0; i < 3; i++) {**

**for (var j = 0; j < 3; j++) {**

**var input = prompt('Значение в координатах '+i+','+j, '');**

**// если отмена ввода или пустая строка -**

**// завершить оба цикла**

**if (!input) break outer; // (\*)**

**}**

**}**

**alert('Готово!');**

*для этого использована метка.*

Метка имеет вид:

**outer: for (var i = 0; i < 3; i++) { ... }**

Можно также выносить её на отдельную строку:

**outer:**

**for (var i = 0; i < 3; i++) { ... }**

Вызов break outer ищет ближайший внешний цикл с такой меткой и переходит в его конец.

*В примере будет разорван самый внешний цикл и управление перейдёт на alert.*

Директива continue также может быть использована с меткой, в этом случае управление перепрыгнет на следующую итерацию цикла с меткой.

*Пример:*

**var i = 0,**

**j = 8;**

**checkiandj: while (i < 4) {**

**console.log("i: " + i);**

**i += 1;**

**checkj: while (j > 4) {**

**console.log("j: "+ j);**

**j -= 1;**

**if ((j % 2) == 0)**

**continue checkj;**

**console.log(j + " является нечётным.");**

**}**

**console.log("i = " + i);**

**console.log("j = " + j);**

**}**

*Результат программы будет следующим:*

**"i: 0"**

**// начало checkj**

**"j: 8"**

**"7 является нечётным."**

**"j: 7"**

**"j: 6"**

**"5 является нечётным."**

**"j: 5"**

**// конец checkj**

**"i = 1"**

**"j = 4"**

**"i: 1"**

**"i = 2"**

**"j = 4"**

**"i: 2"**

**"i = 3"**

**"j = 4"**

**"i: 3"**

**"i = 4"**

**"j = 4"**

Для управления HTML-страницами и их элементами JavaScript использует объектную модель документов DOM (Document Object Model). Упрощенная схема объектной модели представлена на рисунке 1.

**window**

**navigator**

**frames [ ]**

**location**

**history**

**status**

**document**

**forms[ ]**

**elements[ ]**

**anchors[ ]**

**links[ ]**

**images [ ]**

*button checkbox file hidden password text radio reset submit textarea*

*select -> options[]*

Рисунок 1 – Объектная модель документа

Объекты появляются после загрузки документа браузером или как результат работы программы. Каждый объект характеризуется набором методов, свойств и возможных событий.

Свойства объектов соответствуют атрибутам определенных тегов; их использование позволяет получить или изменить характеристики окна браузера, загруженных документов и элементов HTML-страницы. Например, к свойствам изображения относятся src, width, height и т. д.

Методы объекта представляют собой действия, которые могут быть выполнены по отношению к данному объекту. Напри-мер, одним из методов объекта document является write, который помещает в документ определенный текст.

Создадим документ pr\_DOM.html:

<HTML>

<HEAD> <TITLE> DOM </TITLE> </HEAD>

<BODY>

<FORM name=Sum>

Первое число <INPUT type="text" name="number1"><BR> Второе число <INPUT type="text" name="number2"><BR>

<INPUT type="button" name="but"

value="Вычислить сумму"><BR><BR> Сумма <INPUT type="text" name="result">

</FORM>

</BODY>

</HTML>

При открытии данного документа создается объектная модель, представленная на рисунке 2.

**window**

**document**

**forms[0] name=Sum**

**elements[0] name=number1**

**elements[1] name=number2**

**elements[2] name=but**

**elements[3] name=result**

Рисунок 2 – Иерархия объектов документа pr\_DOM.html

Изменим описание кнопки на странице следующим образом:

<input type="button" name="but" value=" Вычислить сумму "

onClick="document.forms[0].elements[3].value= parseInt(document.forms[0].elements[0].value) + parseInt(document.forms[0].elements[1].value);">

В результате изменений при нажатии кнопки значения, введенные пользователем в два первых текстовых поля, будут переводиться в числовой тип данных (функцией parseInt) и суммироваться; результат будет помещен в третье текстовое поле.

Объект window можно не указывать (как в данном примере), если имеется в виду текущее окно браузера.

Для обращения к объектам страницы можно также использовать имя, указанное в атрибуте name.

Например, обработку нажатия кнопки можно определить следующим образом:

<input type="button" name="but" value=" Вычислить сумму "

onClick="document.Sum.result.value=parseInt(document.Sum.number1.value) + parseInt(document.Sum.number2.value);">

Для расчета суммы можно определить отдельную функцию summa( ) и вызвать ее при помощи атрибута onClick:

<html>

<head>

<title> DOM </title>

<script>

// определяем функцию summa()

function summa(){

// объявляем переменные n1 и n2

var n1=parseInt(document.Sum.number1.value); var n2=parseInt(document.Sum.number2.value);

/\*рассчитываем сумму и записываем ее в поле result\*/

document.Sum.result.value=n1+n2;

}

</script>

</head>

<body>

<form name=Sum>

...

<input type="button" name="but" value="Вычислить сумму" onClick="summa()">

...

</form>

</body>

</html>

Работа с окнами. Для открытия нового окна используется метод open объекта window.

Например:

My\_window=window.open("doc.html","wind","width=250,height=100,status=no, toolbar=no,menubar=no,scrollbars=no");

My\_window является переменной, с помощью которой можно получить доступ к открытому окну (например, для того чтобы закрыть его). В новое окно с именем wind будет загружен документ doc.html. Третий параметр метода open определяет характеристики окна. Возможные свойства окна приведены в таблице 1.

Таблица 1 – Свойства окна

|  |  |
| --- | --- |
| Свойство | Изменяемый параметр окна |
| width | Ширина окна в пикселях |
| height | Высота окна в пикселях |
| toolbar | Наличие панели инструментов |
| location | Наличие строки адреса |
| status | Наличие строки состояния |
| menubar | Наличие строки меню |
| scrollbars | Наличие полос прокрутки |
| resizable | Разрешает/запрещает изменение размеров окна |

Для того чтобы передать фокус некоторому окну, используется метод focus. Например:

My\_window.focus( );

передает фокус окну, с которым связана переменная My\_window. Для закрытия окна используется метод close( ). Например: window.close( ) // закрывает текущее окно My\_window.close( )

/\* закрывает окно, с которым связана переменная My\_window\*/

window.opener.close( )

//закрывает окно, из которого было открыто текущее

В JavaScript определены методы для вызова стандартных окон: alert, confirm и prompt.

Метод alert("message") вызывает диалоговое окно с со-общением, определенным параметром message, и кнопкой OK.

Метод confirm("message") вызывает диалоговое окно с указанным сообщением message и кнопками OK и Cancel.

Метод prompt ("message"[,"inputDefault"]) отображает диалоговое окно ввода текста. Параметр message определяет текст запроса пользователю, inputDefault указывает текст по умолчанию в поле ввода.

**5. Порядок выполнения работы**

1. Создайте HTML-страницу Lab17\_1.html. Установите название документа, цвет фона и текста.
2. Создайте сценарий JavaScript, который при загрузке данного документа будет вызывать окно, запрашивающее имя пользователя (используйте метод prompt).
3. Введенное имя сохраните в переменной name.
4. Выведите на страницу приветствие вида «Здравствуйте, name!». Если имя не введено, приветствие должно выглядеть следующим образом: «Здравствуйте! Было бы приятно с Вами познакомиться».
5. При щелчке левой кнопкой мыши на приветствии должно появляться окно confirm, в котором содержится вопрос «Вы хотите изменить имя пользователя?». Если ответ утвердительный, вновь вызовите окно запроса имени и соответствующим образом измените имя пользователя на странице.
6. Создайте документ Lab17\_2.html, в соответствии с вашим вариантом (таблица 2). Определите соответствующие поля формы для получения входных данных. Пользователю должна быть предоставлена возможность выбора формы предъявления результата: в новом окне, в стандартном окне alert или на той же странице.

Таблица 2 – Варианты заданий

|  |  |
| --- | --- |
| Вариант | Задание |
| 1 | Вычислите сумму n натуральных чисел |
| 2 | Определите n-й член ряда Фибоначчи |
| 3 | Вычислите факториал числа |
| 4 | Рассчитайте сумму двух чисел |
| 5 | Рассчитайте разность двух чисел |
| 6 | Рассчитайте произведение двух чисел |
| 7 | Определите частное двух чисел |
| 8 | Найдите корни квадратного уравнения |
| 9 | Определите, является ли високосным год, введенный пользователем |
| 10 | Определите сумму n членов ряда Фибоначчи |
| 11 | Определите длину окружности по ее радиусу |
| 12 | Определите результат вычисления арифметического выражения, введенного в текстовое поле |
| 13 | Найдите наименьший общий делитель двух чисел |
| 14 | Найдите наибольший общий делитель двух чисел |
| 15 | Найдите площадь прямоугольника по его ширине и высоте |
| 16 | Найти площадь треугольника |
| 17 | Рассчитать площадь круга вписанного в квадрат со стороной а |

**6. Форма отчета о работе**

*Лабораторная работа № \_\_\_*

*Номер учебной группы \_\_\_\_\_\_\_\_\_\_\_\_*

*Фамилия, инициалы учащегося \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Дата выполнения работы \_\_\_\_\_\_\_\_\_\_\_\_\_*

*Тема работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Цель работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Оснащение работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Результат выполнения работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**7. Контрольные вопросы и задания**

1. Укажите способы использования кода JavaScript в составе HTML-страницы.

2. На HTML-странице определена следующая форма:

<FORM name="f1">

<INPUT type="text" name="txt"> <br>

<INPUT type="button" name="but"> <br>

</FORM>

Каким образом можно поместить в поле ввода txt некоторый текст, изменить размер данного поля, изменить надпись на кнопке?

3. Как в JavaScript определяются комментарии?

4. Перечислите методы создания стандартных окон JavaScript.

5. Укажите параметры окна, которое открывается при выполнении следующего кода:

msgWindow=window.open("1.html", "displayWindow", "width=300,height=300,location=no, status=yes,toolbar=yes,menubar=no, scrollbars=no,resizable=no")

Как можно загрузить в данное окно документ new.html, используя атрибут TARGET тега <A>? Как можно закрыть данное окно?

**8. Рекомендуемая литература**

**Макфарланд, Д.** Новая большая книга CSS / Дэвид Макфарланд. — СПб.: Питер, 2016. — 720с.

**Никсон, Р.** Создаем динамические веб-сайты с помощью PHP, MySQL, JavaScript, CSS и HTML5 / Р. Никсон. — СПб.: Питер, 2016.

**Прохоренок, Н.А**. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Н.А.Прохоренок. – СПб.: БХВ-Петербург, 2010.

**Фрейн, Б.** HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств / Бен Фрейн. — СПб.: Питер Пресс, 2017. — 272с.