# Лабораторная работа № 1

## ОСНОВНЫЕ ПОНЯТИЯ JAVASCRIPT.

## РАБОТА С ОКНАМИ

Цель работы: ознакомиться с основными понятиями языка JavaScript, способами размещения JavaScript-кода, изучить основные свойства и методы работы с окнами.

**Задания:**

1. Изучите теоретические сведения.
2. Выполните все примеры, описанные в теоретической части.
3. Оформите отчет о лабораторной работе.

**ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

**JavaScript** предназначен для создания интерактивных html-документов. Основные области использования JavaScript:

* Создание динамических страниц, т.е. страниц, содержимое которых может меняться после загрузки.
* Проверка правильности заполнения пользовательских форм.
* Решение "локальных" задач с помощью сценариев.
* JavaScript-код - основа большинства Ajax-приложений.

JavaScript позволяет создавать приложения, выполняемые на стороне клиента, т.е. эти приложения выполняются браузером на компьютере пользователя.

Программы (сценарии) на этом языке обрабатываются встроенным в браузер интерпретатором. К сожалению, не все сценарии выполняются корректно во всех браузерах, поэтому тестируйте свои javascript-программы в различных браузерах.

Язык JavaScript регистрозависимый, т.е. заглавные и прописные буквы алфавита считаются разными символами.

Прежде, чем приступить к написанию сценариев, необходимо ознакомиться с основными понятиями, такими как литералы, переменные и выражения.

**Литералы**

**Литералы** - это простейшие данные, с которыми может работать программа.

1. Литералы целого типа - целые числа в представлении:

* десятичном, например: 15, +5, -174.
* шестнадцатеричном, например: 0х25, 0хff. Шестнадцатеричные числа включают цифры 0 - 9 и буквы a, b, c, d, e, f. Записываются они с символами 0х перед числом.
* восьмеричном, например: 011, 0543. Восьмеричные числа включают только цифры 0 - 7.

1. Вещественные литералы - дробные числа. Целая часть отделяется от дробной точкой, например: 99.15, -32.45. Возможна экспоненциальная форма записи, например: 2.73e-7. В привычном виде это 2.73Х10-7, но в JavaScript знак умножения и 10 заменяются символом e.
2. Логические значения - из два: истина (true) и ложь (false).
3. Строковые литералы - последовательность символов, заключенная в одинарные или двойные кавычки. Например: "ваше имя", 'ваше имя'.

**Переменные**

**Переменные** используются для хранения данных. Переменные определяются с помощью оператора **var**, после которого следует имя переменной. Имя переменной должно начинаться с буквы латинского алфавита или с символа подчеркивания. Само имя может включать буквы латинского алфавита, цифры и знак подчеркивания.

Например:

**var test**

**var \_test**

**var \_my\_test1**

Каждой переменной можно присвоить значение либо при ее инициализации (объявлении), либо в коде самой программы. Оператор присваивания обозначается знаком равно (=), но знак равенства здесь не имеет своего прямого значения. В данном случае он лишь указывает на то, что данной переменной присвоили значение.

Например:

**var a=15**

**var b=23.15**

**var c='выполнено'**

**var s=true**

Каждая переменная имеет тип, определяемый значением переменной. Так в данном примере: переменные a и b имеют тип number, переменная c имеет тип string, а переменная s - логический тип.

**Выражения**

**Выражения** строятся из литералов, переменных, знаков операций и скобок. В результате вычисления выражения получается единственное значение, которое может быть числом, строкой или логическим значением.

В выражении a\*b, a и b называются операндами, а \* - знак операции. В JavaScript определены следующие **операции**:

|  |  |
| --- | --- |
| **Операция** | **Название** |
| + | сложение |
| - | вычитание |
| \* | умножение |
| / | деление |
| % | остаток от деления целых чисел |
| ++ | увеличение значения операнда на единицу |
| -- | уменьшение значения операнда на единицу |

Операторы в выражении вычисляются слева направо в соответствии с приоритетами арифметических операций. Изменить порядок вычисления можно с помощью скобок.

Для каждого из арифметических операторов есть форма, в которой одновременно с заданной операцией выполняется присваивание:

|  |  |
| --- | --- |
| **оператор** | **эквивалентный оператор присваивания** |
| X+=Y | X=X+Y |
| X-=Y | X=X-Y |
| X\*=Y | X=X\*Y |
| X/=Y | X=X/Y |
| X%=Y | X=X%Y |

В данном случае сначала вычисляется правый операнд, а затем полученное значение присваивается левому операнду.

Для того, чтобы можно было сравнивать два значения в JavaScript определены операции сравнения, результатом которых может быть только логическое значение: true или false:

|  |  |
| --- | --- |
| **операция** | **название** |
| < | меньше |
| <= | меньше или равно |
| == | равно |
| != | не равно |
| >= | больше или равно |
| > | больше |

В JavaScript определены логические операции:

&& - логическое И (AND),

|| - логическое ИЛИ (OR),

! - логическое НЕ (NOT).

Результаты воздействия логических операторов на различные комбинации значений операндов показаны в таблице:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **A** | **B** | **A&&B** | **A||B** | **!A** |
| true | true | true | true | false |
| true | false | false | true | false |
| false | true | false | true | true |
| false | false | false | false | true |

Проще говоря, значение выражения A&&B истинно, если истинны оба оператора, и ложно в противном случае. Значение выражения A||B истинно, если значение хотя бы одного операнда истинно, и ложно в противном случае. Если значение операнда А истинно, то !A - ложно и наоборот.

Для строковых литералов определена операция конкатенация строк, т.е. их объединение. Обозначается эта операция знаком плюс (+). Результатом выполнения операции является также строка.

Пример:

**var st1="Привет";**

**var st2="Вася";**

**var st3=st1+st2;**

В результате в переменной st3 будет значение "Привет Вася".

В конце приведем таблицу приоритетов операций в порядке их убывания. Приоритет операций определяет порядок, в котором выполняются операции в выражении.

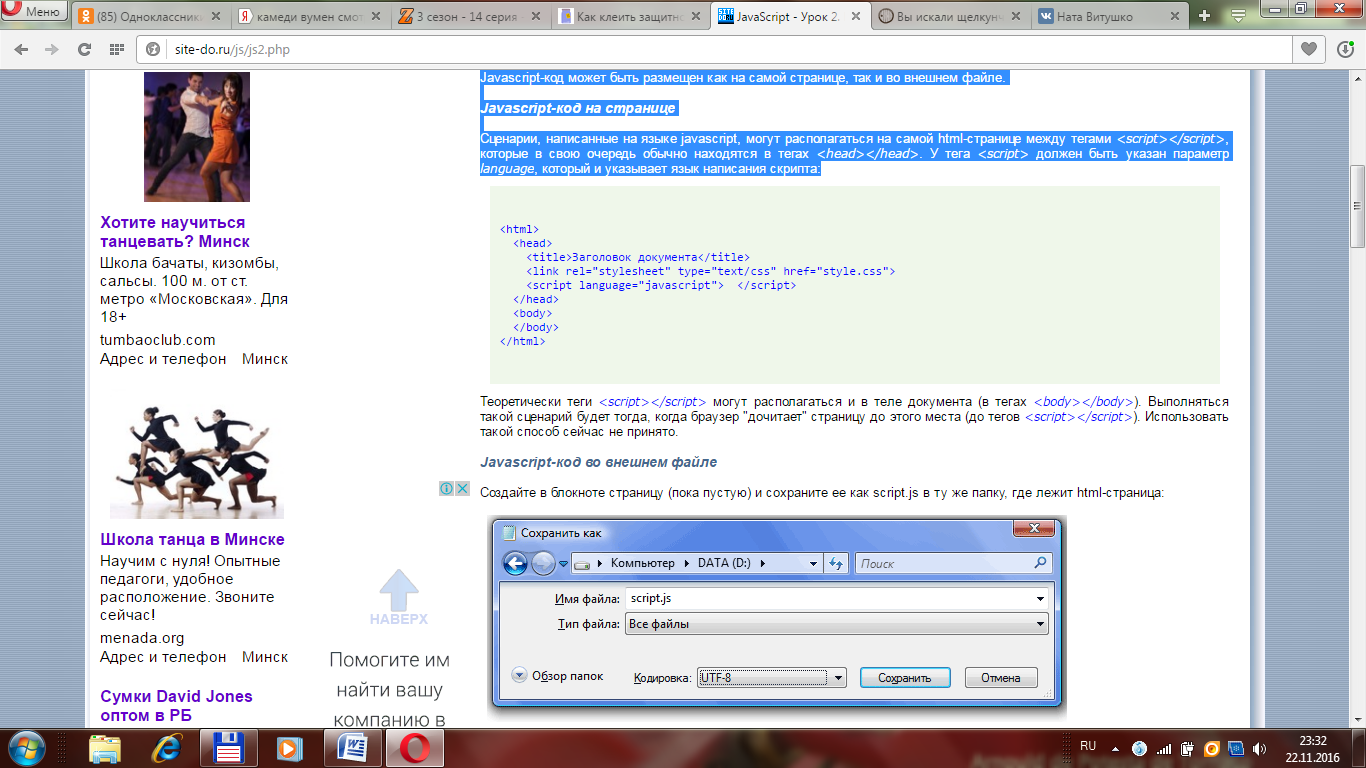
|  |  |
| --- | --- |
| **название** | **обозначение** |
| инкремент | ++ |
| декремент | -- |
| отрицание | ! |
| унарный минус | - |
| умножение | \* |
| деление, остаток от деления | /,% |
| сложение | + |
| вычитание | - |
| сравнение | <, >, <=, >= |
| равенство | == |
| не равенство | != |
| логическое И | && |
| логическое ИЛИ | || |
| присваивание | =, +=, -=, \*=, /=, %=, != |

**РАЗМЕЩЕНИЕ JAVASCRIPT-КОДА**

JavaScript -код может быть размещен как на самой странице, так и во внешнем файле.

**JavaScript-код на странице**

Сценарии, написанные на языке JavaScript, могут располагаться на самой html-странице между тегами **<script></script>**, которые в свою очередь обычно находятся в тегах **<head></head>.** У тега **<script>** должен быть указан параметр **language**, который и указывает язык написания скрипта:



Теоретически теги **<script></script>** могут располагаться и в теле документа (в тегах <body></body>). Выполняться такой сценарий будет тогда, когда браузер "дочитает" страницу до этого места (до тегов <script></script>). Использовать такой способ сейчас не принято.

**JavaScript-код во внешнем файле**

Пример 1: Создайте в блокноте страницу (пока пустую) и сохраните ее как **script.js** в ту же папку, где лежит html-страница.

Это будет страница с JavaScript -кодом. Теперь необходимо подключить страницу script.js к html-странице. Для этого в html существует тег **<script>,** который и отвечает за подключение внешних файлов скриптов. Добавьте этот тег в html-страницу:



**JAVASCRIPT ОБРАБОТЧИКИ СОБЫТИЙ**

Пример 2: Создайте html-страницу и поместите в нее следующий код:



Откройте эту страницу в браузере, она выглядит так:

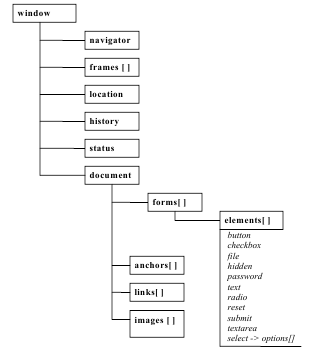


Браузер читает html-страницу, видит оператор для выполнения document.write("Моя первая страница."); и выполняет его.

Рассмотрим из чего состоит сам оператор (инструкция):



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



На самом верхнем уровне находится объект **window**, представляющий окно браузера и являющийся "родителем" всех остальных объектов. Расположенные ниже могут иметь свои подчиненные объекты. Так объект **document** (текущая страница) может иметь дочерний объект **form** (форма) и т.д.

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. Например:

* **document.write** позволяет писать текст в текущую страницу,
* **window.open** открывает новое окно браузера.

Также объекты имеют свойства, например:

* **document.bgcolor** содержит значение фонового цвета текущей страницы,
* **document.title** содержит заголовок страницы.

Все инструкции программного кода заканчиваются точкой с запятой.

Сценарий, который мы написали, выполнился немедленно после загрузки страницы. Однако в большинстве случаев нам нужно, чтобы сценарий выполнялся после каких-либо действий пользователя: нажатия на кнопку или ввода текста. Т.е. в ответ на пользовательское событие.

Для этого в тегах элементов страницы введены параметры обработки событий, задающие действия, выполняемые при возникновении события, связанного с элементом. Например:

**<div onClick="addText();"></div>**

Здесь **Click** - событие (щелчок по div-у), onClick - обработчик события, addText() - имя функции, которая сработает при возникновении этого события (щелчка по div-у).

Перечислим события, которые поддерживаются JavaScript.

|  |  |  |
| --- | --- | --- |
| **событие** | **когда происходит** | **обработчик события** |
| Blur | потеря объектом фокуса | onBlur |
| Change | пользователь изменяет значение элемента | onChange |
| Click | пользователь щелкает мышью по объекту | onClick |
| DblClick | пользователь делает двойной щелчок мышью по объекту | onDblClick |
| DragDrop | пользователь перетаскивает мышью объект | onDragDrop |
| Error | возникновение javascript-ошибки | onError |
| Focus | окно или элемент формы получает фокус | onFocus |
| KeyDown | пользователь нажимает клавишу клавиатуры | onKeyDown |
| KeyPress | пользователь удерживает нажатой клавишу клавиатуры | onKeyPress |
| KeyUp | пользователь отпускает клавишу клавиатуры | onKeyUp |
| Load | документ загружается в браузер | onLoad |
| MouseDown | пользователь нажимает кнопку мыши | onMouseDown |
| MouseOut | указатель мыши выходит за пределы элемента | onMouseOut |
| MouseOver | указатель мыши помещается над элементом | onMouseOver |
| MouseUp | пользователь отпускает кнопку мыши | onMouseUp |
| Move | пользователь перемещает окно | onMove |
| Reset | пользователь нажимает кнопку "reset" формы | onReset |
| Resize | пользователь изменяет размеры окна или элемента | onResize |
| Select | пользователь выбирает элемент формы | onSelect |
| Submit | пользователь нажимает кнопку "submit" формы | onSubmit |
| Unload | пользователь закрывает документ | onUnload |

Хорошим стилем программирования является оформление действий, выполняемых при обработке событий, в виде функций.

Рассмотрим пример. 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>**

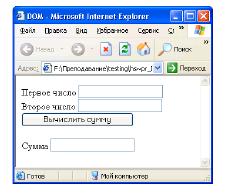
**Сумма <INPUT type="text" name="result">**

**</FORM>**

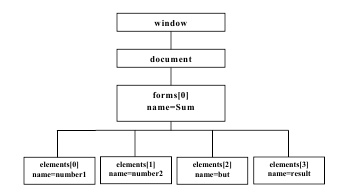
**</BODY>**

**</HTML>**

Результат создания документа pr\_DOM.html приведен на рисунке.



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



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");

В результате выполнения данного кода будет открыто окно, представленное на рисунке.



**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.

|  |  |
| --- | --- |
| **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

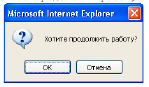
В JavaScript определены методы для вызова стандартных окон: **alert**, **confirm** и **prompt**.

Метод **alert("message")** вызывает диалоговое окно с сообщением, определенным параметром **message**, и кнопкой OK.

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

Пример приведен ниже, результат – на рисунке.

confirm("Хотите продолжить работу?");

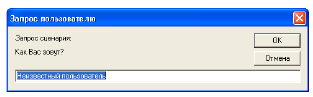


Метод **prompt ("message"[,"inputDefault"])** отображает диалоговое окно ввода текста. Параметр **message** определяет текст запроса пользователю, **inputDefault** указывает текст по умолчанию в поле ввода.

Пример приведен ниже, результат – на рисунке.

prompt ("Как Вас зовут?",

"Неизвестный пользователь");



**КОНТРОЛЬНЫЕ ВОПРОСЫ:**

1. Перечислите основные области использования JavaScript.
2. Дайте определение литерала, переменной, выражения.
3. Какие операции определены в JavaScript?
4. Заполните таблицу:

|  |  |
| --- | --- |
| **оператор** | **эквивалентный оператор присваивания** |
| X+=Y |  |
|  | X=X-Y |
|  | X=X\*Y |
| X/=Y |  |
|  | X=X%Y |

1. Какие логические операции определены в JavaScript? Как они обозначаются.
2. Где может быть размещен JavaScript –код? Приведите примеры.
3. Для чего используется DOM? Расшифруйте аббревиатуру.
4. Перечислите наиболее важные свойства окна. Каково их назначение.
5. Перечислите наиболее важные методы окна. Каково их назначение.

**ВОПРОСЫ ДЛЯ САМОСТОЯТЕЛЬНОГО ИЗУЧЕНИЯ:**

1. Типы данных JavaScript;
2. Правила определения имени переменной;
3. Операторы JavaScript;
4. Особенности использования функций **eval**, **parseInt** и **parseFloat**;
5. Способы доступа к объектам HTML-страницы;
6. Особенности обращения к отдельным элементам формы (например, спискам SELECT, переключателям RADIO и т. д.).