Практическое занятие № 14
"Основы синтаксиса языка JavaScript: литералы, переменные, массивы, условные операторы, операторы циклов"

Цель работы:
1. Изучение особенностей синтаксиса языка JavaScript.
2. Получение практических навыков в создании HTML-документов с применением сценариев.

Оборудование и программное обеспечение:
1. ПК, локальная сеть.
2. ОС MS Windows 7 (10), офисный пакет MS Office, браузер MS Internet Explorer.

План занятия:
1. Освоение теоретической части занятия в соответствии с разделами.
2. Выполнение практической части занятия в соответствии с заданиями.
3. Повторение и закрепление основных изученных понятий и терминов.
4. Оформление отчета.

ЛЕКСИЧЕСКАЯ СТРУКТУРА

Лексическая структура языка программирования - это набор правил, которые определяют, какие символы могут содержать идентификаторы, какие символы используются для определения комментариев, как инструкции отделяются друг от друга и т.д. Лексическую структуру языка программирования иначе называют синтаксисом.

Каждое отдельное правило в лексической структуре называется лексемой.

Лексема - минимальная единица текста программы, которая имеет определённый смысл для интерпретатора и которая не может быть разбита на отдельные части. К лексемам относятся: ключевые слова, литералы, знаки операторов и т. д. Далее будет рассмотрен синтаксис JavaScript.

НАБОР СИМВОЛОВ

При написании программ на JavaScript используется набор символов Unicode. Он поддерживает практически все письменные языки, имеющиеся на планете в настоящее время. Это позволяет использовать в строковых литералах и идентификаторах не только те буквы, которые есть у вас на клавиатуре, но и любые другие из различных языков:

```
1    var something = "café";
2    var число = 123;
3
4    document.write("something: " + something + "<br>число: " + число)
```

ЧУВСТВИТЕЛЬНОСТЬ К РЕГИСТРУ

Язык JavaScript чувствителен к регистру символов. Это значит, что ключевые слова, имена переменных, функций и любые другие идентификаторы используемые в программе всегда должны содержать одинаковый набор прописных и строчных букв. Например, ключевое слово switch должно быть написано как switch, а не Switch или SWITCH, так же и имена переменных myVar, MYVAR или MyVar - будут считаться, как имена трех различных переменных.

```
1    <script>
2      var num = 2;
3      var Num = 4;
4
5      document.write("num: " + num + "<br> Num: " + Num);
6    </script>
```

ПРОБЕЛЬНЫЕ СИМВОЛЫ

Между лексемами могут вставляться разделители, в качестве которых используются пробельные символы: пробелы, табуляция и переносы строк. Число пробельных символов не ограничивается и зависит от предпочтений программиста, основная цель которого - создать наглядный и легко читаемый текст программы (исходный код). Интерпретатор игнорирует все пробельные символы между лексемами и воспринимает текст программы как сплошной поток кода, для него важно лишь определить границы между лексемами.

В примере ниже представлен JavaScript-код, в котором используются пробельные символы между лексемами:

```
1    function drawStarLine(numStars) {
2      for(var i = 0; i < numStars; i++) {
3        document.write("*");
4      }
```

```
5    }
6
7    drawStarLine(45);
```

Код примера можно было бы написать и в одну строку, не используя пробельных символов, но такой код будет менее удобен для чтения:

```
1    function drawStarLine(numStars){for(var i=0;i<numStars;i++){document.write("*");}}drawStarLine(45);
```

КОММЕНТАРИИ

Комментарий – это пояснение. С точки зрения интерпретатора, комментарий – это текст, который нужно игнорировать при анализе кода.

JavaScript поддерживает два вида комментариев: однострочный и многострочный. Всё, что идёт после символов // (два слэша) и до конца строки считается комментарием, такой вид комментария называется "однострочным". Любой текст между символами /* и */ считается комментарием, такой вид комментария называется "многострочным", так как он может состоять из нескольких строк:

```
1    // Это однострочный комментарий
2
3    /* Это многострочный комментарий */
4
5    /*
6    * Это ещё один многострочный комментарий. *
7    * Он расположен на нескольких строках.    *
8    * Звёздочки слева и справа вставлены      *
9    * для красоты.                            *
10   */
```

Комментарии используют для пояснения каких-нибудь участков кода.

Ещё одна важная роль комментариев, которую часто используют на практике – это временное отключение некоторой части кода. Таким образом комментарии используют, когда бывает трудно определить местонахождение ошибки или, когда некоторая часть кода временно не нужна.

Обратите внимание, что многострочные комментарии не могут быть вложенными, то есть нельзя вложить один многострочный комментарий в другой. Такая ошибка может появиться, например, когда необходимо временно отключить некоторую часть кода, а в нём уже содержится многострочный комментарий:

```
1    /*
2    var str = "Hello World!"; /* Этот комментарий создаст проблему */
3    */
```

ЛИТЕРАЛЫ

Литерал (константа) – это запись в исходном коде программы, представляющая собой обычное фиксированное значение.

Литералы представляют собой константы, непосредственно включаемые в текст программы, в отличие от прочих данных - констант и переменных, обращение к которым осуществляется посредством ссылок. Литералы не могут быть изменены в тексте программы. В следующем примере 14 и "Кот" это литералы, а num и fish - переменные:

```
1    var num = 14;
2    var animal = "Кот";
```

Представление каждого литерала зависит от конкретного типа данных: числовые, строковые, логические (булевы) и т.д. Ниже представлены литералы простых типов

```
1    9        // число девять
2    17.3     // число с плавающей точкой
3    "текст"  // строка текста
4    'h3'     // другая строка
5    true     // логическое значение (булево)
6    null     // пустой объект
7    undefined // неопределённое значение
```

Литералы являются важной частью любого языка программирования, так как написать программу без них невозможно.

ИДЕНТИФИКАТОРЫ

Идентификатором называется последовательность букв, цифр, символов подчёркивания "_" и знаков доллара "$". Идентификаторы выступают в качестве имён переменных, функций, свойств объекта, и т. д. При выборе идентификатора необходимо учитывать следующие правила:

Идентификаторы не могут совпадать ни с одним из ключевых и зарезервированных слов JavaScript, т. к. они являются частью синтаксиса самого языка и имеют специальное значение для интерпретатора.

Идентификаторы могут состоять из одного и более символов.

Идентификаторы должны начинаться с буквы, символа подчёркивания или знака доллара. Далее могут следовать любые буквы, цифры, знаки доллара или символы подчёркивания в любом количестве.

Примеры допустимых идентификаторов в качестве имён переменных:

```
1  x
2  my_var
3  _myCar2
4  $cash
```

Чтобы лучше понимать код, при его чтении, идентификаторы желательно придумывать такие, которые будут соответствовать хранящимся в них данным:

```
1  var age = 30;           // Возраст
2  var hairColor = "black"; // Цвет волос
3  var year = 1900;        // Год
```

Есть два устоявшихся негласных стиля используемых для записи идентификаторов, состоящих из нескольких слов: camelCase и snake_case.

В JavaScript наиболее популярным стилем именования идентификаторов, состоящих из нескольких слов, является camelCase – "верблюжья" нотация (нотация – это устоявшиеся правила записи). Согласно этому стилю идентификаторы, которые состоят из одного слова, пишутся строчными буквами:

```
1  var color = "black";
2  var number = 7;
```

Если идентификатор состоит более, чем из одного слова, то первое слово пишется строчными буквами, а каждое последующее слово начинается с прописной (заглавной) буквы:

```
1  var myAge = 10;
2  var firstName = "Bilbo";
```

Верблюжья нотация получила своё название в результате того, что заглавные буквы внутри идентификатора напоминают горбы верблюда.

Менее популярным стилем именования идентификаторов, состоящих из нескольких слов, является snake_case – "змеиная" нотация. Согласно этому стилю идентификаторы, которые состоят из одного слова, пишутся строчными буквами:

```
1  var color = "синий";
2  var number = 176;
```

Если идентификатор состоит более, чем из одного слова, то слова разделяются символом нижнего подчёркивания, при этом каждое последующее слово пишется с маленькой буквы:

```
1  var my_age = 13;
2  var first_name = "Bilbo";
```

КЛЮЧЕВЫЕ И ЗАРЕЗЕРВИРОВАННЫЕ СЛОВА

Стандарт ECMA-262 определяет набор ключевых слов (keywords), имеющих особое значение для интерпретатора. В таблице ниже представлен полный список ключевых слов:

Break	delete	function	return	typeof
Case	do	if	switch	var
Catch	else	in	this	void
continue	false	instanceof	throw	while
debugger	finally	new	true	with
default	for	null	try	

Кроме того ECMA-262 содержит набор зарезервированных слов (reserved words). Эти слова в настоящее время не являются частью языка, но могут войти в его состав в будущих версиях языка:

class	const	enum	export	extends	import	super

В строгом режиме (strict mode) также запрещается использовать в качестве идентификаторов следующие зарезервированные слова:

implements	let	private	public	yield
interface	package	protected	static	

В строгом режиме также вводится ограничение на использование в качестве идентификаторов следующих слов:

arguments	eval

ИНСТРУКЦИИ

Инструкция – это указание на совершение какого-либо действия, например, создать переменную, запустить цикл, выйти из функции и т. п. Любая программа представляет собой последовательность выполняемых инструкций. Окончание инструкции обозначается символом ; (точка с запятой):

```
1  // Инструкция объявления и инициализации переменной
2  var num = 12;
```

Использование точки с запятой для указания конца инструкции не является обязательным условием. В JavaScript не обязательно в конце инструкций ставить точку с запятой, если они находятся на разных строках:

```
1  первая инструкция
2  вторая инструкция
```

Если инструкции находятся на одной строке, их надо обязательно разделять с помощью точки с запятой, тем самым сообщая интерпретатору, где заканчивается одна инструкция и начинается другая:

```
1  первая инструкция; вторая инструкция;
```

Однако хорошей практикой в программировании является использование точки с запятой всегда, даже если инструкции расположены на разных строках:

Выполнение практической части занятия:

Интегрируйте сценарий в HTML-страницу, используя специальный тег <SCRIPT> по данному шаблону: <SCRIPT> код сценария ...</SCRIPT>.
Этот тег является контейнером и его можно помещать в любую часть кода HTML страницы. Обычно его помещают в заглавную часть HTML страниц, т.е. между тегами <HEAD> …. </HEAD>.

Вариант 1: Задайте с помощью JavaScript-кода пробельные символы между лексемами для HTML-страницы с описанием текста.

Вариант 2: Задайте с помощью JavaScript-кода фиксированные значения литералов различных типов (от трех до пяти типов) для HTML-страницы.

Оформление отчета

Контрольные вопросы к практической работе № 14:

1. Дайте понятие лексической структуры языка программирования.
2. Что означают понятия: "набор символов", "чувствительность к регистру"?
3. Что означают понятия: "пробельные символы", "комментарии"?
4. Дайте понятие литерала.
5. Что означает понятие "идентификатор"?
6. Что относится к ключевым и зарезервированным словам и каково их назначение?
7. Назовите назначение инструкций в языке Java Script.