10. JavaScript - язык разработки клиентских веб-приложений

Цель: познакомиться с языком JavaScript и его синтаксисом.

10.1 Теоретические сведения

JavaScript - интерпретируемый язык программирования, стандартизированный международной организацией ECMA в спецификации ECMA-262. Языки JavaScript, JScript и ActionScript являются расширением стандарта ECMA-262.

JavaScript - интерпретируемый, объектно-ориентированный язык. Хотя он имеет существенно меньшее количество возможностей, чем такие объектно-ориентированные языки как C++ и Java.

Распространенным заблуждением является то, что JavaScript аналогичен или тесно связан с Java. Оба языка имеют C-подобный синтаксис, являются объектно-ориентированными и как правило широко используются в клиентских веб-приложениях, однако:

Java	**JavaScript**
использует парадигму ООП из C++	использует прототипный подход
имеет статическую типизацию	имеет динамическую типизацию (значение переменной может содержать объекты любого типа и даже функции)
загружается из скомпилированного байт-кода	интерпретируется напрямую из файла

10.1.2 Структура языка

Структурно JavaScript можно представить в виде объединения трех четко различимых друг от друга частей:

- ядро (ECMAScript),
- объектная модель браузера (Browser Object Model),
- объектная модель документа (Document Object Model или DOM).

10.1.2.1 Ядро

Спецификация ECMAScript описывает типы данных, инструкции, ключевые и зарезервированные слова, операторы, объекты, регулярные выражения, не ограничивая разработчиков производных языков от расширения их новыми составляющими.

10.1.2.2 Объектная модель браузера

Объектная модель браузера - специфичная для каждого браузера часть языка, опосредующая взаимодействие между ядром и объектной моделью документа. Основное предназначение объектной модели браузера - управление окнами веб-браузера и обеспечение их взаимодействия. Каждое из окон браузера представляется объектом window.

Помимо управления окнами, в рамках объектной модели браузера, браузерами обычно обеспечивается поддержка следующих возможностей:

- управление фреймами;
- исполнение кода и зацикливания с задержкой;
- системные диалоги;
- управление адресом открытой страницы;
- управление информацией о браузере;
- управление информацией о параметрах монитора;
- ограниченное управление историей просмотра страниц;
- поддержка работы с HTTP cookie.

10.1.2.3 Объектная модель документа

Объектная модель документа - интерфейс программирования приложений для HTML и XML - документов. Согласно DOM документу можно поставить в соответствие дерево объектов, обладающих рядом свойств, которые позволяют производить с ним различные манипуляции:

- получение узлов,
- изменение узлов,
- изменение связей между узлами,
- удаление узлов.

10.1.3 Основы JavaScript (ECMAScript)

Если сравнить JavaScript с традиционными языками разработки десктопных приложений, то возможности JavaScript существенно ограничены:

- язык не позволяет разрабатывать самостоятельные приложения;
- сценарии на JavaScript могут выполняться только при помощи интерпретатора, в частности веб-браузером.

- JavaScript - язык без строгого контроля типов. Поэтому не требуется объявлять тип переменных явно. Кроме того, во многих случаях JavaScript исполняет преобразования автоматически, когда они необходимы. Например, при сложении строки и числа, число будет преобразовано в строку.

Код на JavaScript пишется в текстовом формате, и организован в инструкции, блоки, состоящие из связанных наборов инструкций, и комментариев. В пределах инструкции можно использовать переменные и данные, такие как строки, числа и выражения. Для объявления конца инструкции используется точка с запятой (;). Группа JScript -инструкций, заключенная в фигурные скобки {}, называется блоком.

Комментарием в JavaScript является текст, расположенный после // до конца строки. Многострочный комментарий начинается с /*, и заканчивается */.

Знак равенства (=) используется в JavaScript как присваивание. Следующий код

подразумевает "Присвоить значение 3.14 переменной Pi".

10.1.4 Типы данных

JavaScript - язык с нестрогим контролем типов, переменные в JavaScript не имеют строго фиксированного типа. Переменные имеют тип, эквивалентный типу значения, которое они содержат. Однако, в некоторых случаях, необходимо принудительное преобразование переменной в определенный тип. Числа могут быть объявлены как строки, а строки необходимо преобразовать в числовой тип. Для этого применяют функции parseInt() и parseFloat().

В JavaScript используется шесть типов данных. Основные из них - числа, строки, объекты, логический. Остальные два – null и undefined (т.е. неопределенный).

Строки объявляются при помощи двойных кавычек или апострофов. Строка может состоять из нуля или более символовunicode. Когда количество символов равно нулю, это называется пустой строкой ("").

JavaScript поддерживает числа как целые, так и с плавающей запятой. Также существуют специальные представления чисел, например NaN (не число).

Примеры чисел:

```
3.14 // Вещественное число
15 // Целое число
0177 // Восьмеричное число 177
0XA8 // Шестнадцатиричное число A8
```

10.1.5 Операторы

Язык поддерживает условные выражения if и if...else. При использовании нескольких условий одновременно можно использовать операторы || (ИЛИ) или && (И).

В JavaScript существует несколько типов циклов: for, for...in, while, do...while и switch.

10.1.6 Функции

В JavaScript имеется два вида функций: встроенные и определяемые. Программист имеет возможность создавать собственные функции. Определение функции состоит из объявления параметров и блока инструкций JavaScript.

При вызове функции в JavaScript действуют следующие правила передачи аргументов функции:

- Аргументы примитивных типов передаются функции по значению.
- Объекты (и встроенные, и определенные пользователем) передаются по ссылке. Это означает, что все изменения свойств объекта в теле функции производятся непосредственно в самом объекте, а не в его локальной копии и, следовательно, сохраняются после возврата из функции.

В общем случае любой объект JavaScript определяется через функцию. Для создания объекта используется конструктор, который в свою очередь вводится через Function. Таким образом, с функциями в JavaScript связаны следующие ключевые вопросы:

- функция как тип данных;
- функция как объект;
- функция как конструктор объектов.

10.1.7 Объекты

Объект - это главный тип данных JavaScript. Объекты в JavaScript, по-сути, являются совокупностями методов и свойств.

Сценарии JavaScript могут использовать объекты следующих видов:

- клиентские объекты, входящие в модель DOM, т.е. отвечающие тому, что содержится или происходит на Web -странице в окне браузера. Они создаются браузером при разборе (парсинге) HTML - страницы. Примеры: window, document, location, navigator и т.п.
- серверные объекты, отвечающие за взаимодействие клиент-сервер.

- встроенные объекты, представляющие различные типы данных, свойства, методы, присущие самому языку JavaScript, независимо от содержимого HTML -страницы. Примеры: Array, String, Date, Number, Function, Boolean, Math.
- пользовательские объекты, которые создаются программистом в процессе создания сценария с использованием конструкторов типа объектов (класса).

10.1.8 Операторы работы с объектами

Оператор for (переменная in объект) позволяет перебрать все свойства объекта. Например:

```
for (d in document)
  document.write("document."+d+" = <b>"+ document[d]+"</b><br>");
```

10.1.9 Клиентские объекты

Для создания механизма управления страницами на клиентской стороне используется объектная модель документа (DOM -Document Object Model). Суть модели в том, что каждому HTML -контейнеру соответствует объект, который характеризуется:

- свойствами;
- методами;
- событиями.

10.1.10 Массивы

Массивы в JavaScript могут быть встроенными, или коллекциями, (например, document.links, document.images) и определяемые пользователем. Для массивов определено несколько методов: join(), reverse(), sort() и другие. Свойство length позволяет узнать количество элементов в массиве.

Элементы массива нумеруются с нуля. В языке JavaScript массив может состоять из разнородных элементов. Элементами массива могут быть также массивы.

10.1.11 Встраивание JavaScript в веб-страницы

Возможны 3 варианта встраивания JavaScript кода в веб-страницы:

- Расположение внутри страницы. Для добавления JavaScript -кода на страницу, можно использовать теги <script></script> .

Наример:

```
<script type="text/javascript">
alert('Hello, World!');
</script>
```

- Расположение внутри тега. Спецификация HTML описывает набор атрибутов, используемых для задания обработчиков событий.

Пример использования:

```
<a href="delete.php" onclick="return confirm('Вы уверены?');">Удалить</a>
```

- Вынесение в отдельный файл. Можно сохранить сценарий в отдельном файле, а потом подключить его с помощью конструкции

```
<script type="text/javascript" src="URL_файла_со_сценарием"></script>
```

10.1.12 Обработка событий в JavaScript

Популярность JavaScript во многом обусловлена именно тем, что написанный на нем сценарий может реагировать на действия пользователя и другие внешние события. Каждое из событий связано с тем или иным объектом: формой, гипертекстовой ссылкой или даже с окном, содержащим текущий документ.

Обработка события производится с помощью специально предназначенного для этого фрагмента кода, называемого обработчиком события.

10.1.13 Регулярные выражения

Регулярные выражения - система поиска текстовых фрагментов в электронных документах, основанная на специальной системе записи образцов для поиска.

Образец, задающий правило поиска, называется " шаблоном ".

При описании структуры шаблона используются:

- гибкая система квантификаторов (операторов повторения);
- операторы описания наборов символов и их типа (числовые, нечисловые, специальные).

Для того, чтобы задать положение искомого фрагмента внутри строки, можно использовать один из следующих операторов:

Представление	Позиция
^	Начало строки
$	Конец строки
\b	Граница слова
\B	Не граница слова
(?=шаблон)	Искомая строка следует после указанной строки (с просмотром вперед)
(?!=шаблон)	Искомая строка не следует после указанной строки (с просмотром вперед)
(?<=шаблон)	Искомая строка следует после указанной строкой (с просмотром назад)
(?<!шаблон)	Искомая строка не следует после указанной строки (с просмотром назад)

Кроме того, язык регулярных выражений предоставляет набор квантификаторов, позволяющих указать число повторений шаблона:

Представление	Число повторений
{n}	Ровно n
{m, n}	От m до n включительно
{m,}	Не менее m
{,n}	Не более n

Имеются и более простые квантификаторы:

Представление	Число повторений	Эквивалент
*	Ноль или более	{0,}
+	Одно или более	{1,}
?	Ноль или одно	{0,1}

Для задания внутри шаблона группы символом можно использовать следующие операторы:

Оператор	Описание
[xyz]	Любой символ из указанного множества
[^xyz]	Любой символ не входящий в указанное множество
[x-z]	Любой символ из указанного диапазона
[^x-z]	Любой символ не входящий в указанный диапазон
. (точка)	Любой символ кроме символов разрыва или переноса строки
\w	Любой буквенно-цифровой символ, включая символ подчеркивания
\W	Любой не буквенный символ
\d	Любая цифра
\D	Любой нецифровой символ
\s	Любой не отображаемый символ
\S	Любой символ (кроме неотображаемых символов)

Для группировки отдельных частей шаблона можно использовать следующие операторы:

Оператор	Описание
( )	Поиск группы символов внутри скобок и сохранение найденного соответствия
(?:)	Поиск группы символов внутри скобок без сохранения найденного соответствия
|	Комбинирование частей в одно выражения с последующим поиском любой из частей в отдельности. Аналогично оператору "ИЛИ".

Если шаблон поиска включает специальные (как правило неотображаемые) символы, для их описания можно использовать следующие обозначения:

Обозначение	Описание
\0	Символ с нулевым кодом
\n	Символ новой строки
\r	Символ начала строки
\t	Символ табуляции
\v	Символ вертикальной табуляции
\xxx	Символ, имеющий заданный восьмеричный ASCII код xxx
\xdd	Символ, имеющий заданный шестнадцатиричный ASCII код dd
\uxxxx	Символ, имеющий ASCII код выраженный ЮНИКОДОМ xxxx

Квантификаторам в регулярных выражениях соответствует максимально длинная строка из возможных (т.е. квантификаторы являются "жадными"). Это может приводить к некоторым проблемам. Например, шаблон (<.*>) описывающий на первый взгляд теги HTML на самом деле будет выделять более крупные фрагменты в документе.

Квантификатор	Описание
*?	"не жадный" эквивалент *
+?	"не жадный" эквивалент +
{n,}?	"не жадный" эквивалент {n,}

10.1.13.1 Примеры регулярных выражений

Дата в формате yyyy-MM-dd:

```
20\d\d\-(0[1-9]|1[0-2])\-(0[1-9]|[12][0-9]|3[01])
```

IP адрес:

```
\b(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\b
```

Доменное имя:

```
^([a-zA-Z0-9]([a-zA-Z0-9\-]{0,61}[a-zA-Z0-9])?\.)+[a-zA-Z]{2,6}$
```

Вещественное число:

```
[-+]?(?:\b[0-9]+(?:\.[0-9]*)?|\.[0-9]+\b)(?:[eE][-+]?[0-9]+\b)?
```

Запись числа в римской системе счисления:

```
^(?i:(?=[MDCLXVI])((M{0,3})((C[DM])|(D?C{0,3}))?((X[LC])|(L?XX{0,2})|L)?((I[VX])|(V?(II{0,2}))|V)?))$
```

10.1.13.1 Регулярные выражения в JavaScript

При поиске по тексту можно использовать шаблон, описывающий подстроку. В JavaScript такой шаблон может быть описан с помощью объекта RegExp. В простейшем случае такой шаблон описывает отдельный символ, однако имеет смысл его использовать для регулярных выражений.

Для объекта RegExp поддерживаются следующие методы:

- Конструктор RegExp. В аргументе задается регулярное выражение.
- Метод test выполняет поиск по шаблону.
- Метод exec выполняет поиск подстроки по шаблону и возвращает найденные соответствия; если соответствий нет, возвращается значение null.
- Метод compile применяется для изменения ранее созданного шаблона.

Аргументом в последних трех методах является строка, в которой выполняется поиск по шаблону.

10.2. Контрольные вопросы

1. Какие есть различия между Java и JavaScript?
2. Как можно структурно представить JavaScript?
3. Сколько типов данных исользуется в JavaScript? Перечислите их.
4. Какие правила передачи аргументов функции действуют при вызове функции в JavaScript?
5. Объекты каких видов могут использовать сценарии JavaScript?
6. Какие методы определены для массивов?
7. Какие варианты встраивания JavaScript кода в веб-страницы?
8. С помощью чего производится обработка события?
9. Что такое регулярные выражения?
10. Что такое объектная модель документа? Какие манипуляции можно осуществлять с её помощью?
11. Какие виды функций существуют в JavaScript?
12. Что описывает спецификация ECMAScript?
13. С чем связано то или иное событие в JavaScript?
14. Какие методы поддерживаются для объекта RegExp?

10.3. Задания

1. Напишите примеры по каждому из вариантов встраивания JavaScript кода в HTML-станицу: (R1001)
   - С помощью тега <script></script>
   - Расположение внутри тега
2. Решите следующие задачи: (R1002)

- Создайте массив, состоящий из 3 элементов, имеющий разные типы данных. Выведите на экран первый и второй элементы массива. Замените третий элемент массива, после чего выведите его значение на экран. Добавьте в массив начальный и конечный элементы при помощи специальных методов. Выведите все элементы массива на экран, после чего подсчитайте его окончательную длину.
- Дан следующий текст: «Every man in the world! Every woman on earth!» Замените слова «man» и «woman» на «person» при помощи методов регулярных выражений replace и compile.