**Назначение применение синтаксис и возможности языка JavaScript**

Многие современные технологии, которые сделали возможным само существование WWW давно превзошли возлагавшиеся на них первоначально надежды. В начале web разрабатывалась как сетевая среда управления статистическими текстами и изображениями. Она постоянно улучшается, изменяется и популяризуется разработчиками web технологий, где улучшение и достижения поставленных целей разработчики выполняют различную работу и стараются еще больше расширить использование ресурсов web. Один из основных технологий который дал разработчикам осуществить свои идеи выполнять различные нестандартные действия в пространстве web является язык JavaScript после внедрение в компьютер клиента этот язык может превратить страницу со статическим содержимым в привлекательный интерактивный и весьма разумный документ. Приложения так называемые сценарии JavaScript могут выполнять различные операции, как приветствие приведение к движению различных объектов изменение размеров цвета и других различных параметров объектов, которые сделают web-страницу более наглядной. На ряду с языком JavaScript существуют другие при помощи которых обеспечиваются полноценное использование ресурсов web страниц. Один из способов повышения эффективности работы по управлению пользователем содержимым страницы является установление с сервером которая обслуживает страницу. Предназначение серверных сценариев – это самая широкая начиная с проверки пароля до задержки выполнения определенного действия. Написание специальных серверных сценариев требует навыков программирования. Ряд таких создается с использованием фундаментальных языков программирования таких как Perl, Python, Java, C или C++. Разработчику не удается самостоятельно создать серверные приложения если он не знаком с технологиями PHP ASP Dot.net и call fusion. Многие серверы позволяют выполнять серверные сценарии написанные на JavaScript. Не зависимо от того какой язык используется для написания серверного сценария решения поставленной перед ним задачи требует обязательного вмешательства разработчика web-страницы или администратора web-сервера, которые настраиваю вспомогательные программы, применяемые для обработки поступающей информации и сохранение готовых данных. На разных этапах развития WWW перед браузером ставилась задача отображать только не большое количество типов данных. Вся система по форматированию данных и рисунков была встроена в браузер и предназначалась для управления исключительно операционной системой компьютера. Ограниченные числом используемых типов данных разработчики приложили не мало усилий, чтобы добиться расширения возможностей браузеров клиентов по обработке данных, представленных в других форматах. Тогда казалось несбыточной мечтой, что когда-нибудь удастся создать браузер, который сможет загружать и обрабатывать звуковые файлы хотя бы одного формата. Одним из способов данной проблемы было разрешение на использование в браузере системы определения типа данных входящего файла и запуска в компьютере клиента специальных приложений обеспечивающий обработку его содержимого. Начиная с Netscape Navigator 2 (начало 1996 года ) применение в браузерах программных надстройках позволило разработчикам расширить возможности браузера без всякой необходимости внесение существенных изменений в их код. В отличии от вспомогательных приложений с помощью надстройки можно реализовать плавное перемещение содержимого файла в html документ. Наиболее распространёнными примерами надстройки являются программные модули, которые используют их для воспроизведения звуковых и видео файлов, расположенных на сервере.

На подобии этого имеются различные надстройки, как:

* Adobe Acrobat Reader
* Macromedia Flash
* adBlock
* Kaspersky

Впервые язык написания сценариев для браузера Netscape появился в ноябре 1995 года, тоe есть составлялся проект его создания. Данный язык в начале назывался LiveScript и разрабатывался параллельно с программным обеспечение для web-сервера Netscape. Язык имел единый синтаксис и должен был служить двум целям: *первой целью* было создание такой среды разработки сценариев, которую администраторы web-серверов могли бы использовать для управления работы сервера и *второй целью* подключение к другим службам это к удаленным базам данным и средством поиска информации. Расширить марку действенного в названии Netscape стало использовать термин LiveWire для обозначение среды JavaScript применяемой сервером для подключения к базам данных. С точки зрения клиента разработчики для улучшения web-страниц могли самыми различными способами использовать сценарий написанные с использованием этого нового языка. В начале декабря 1995 года в преддверии формального выхода в свет NN2 компании Netscape и Sun любезно объявили, что язык создания сценариев в дальнейшем будет именоваться, как JavaScript.

Не смотря на то, что язык JavaScript ориентирован в основном на Netscape Microsoft признала его потенциальные возможности и обеспечило популярность JavaScript включив его под именем Jscript в браузере под название Internet Explorer 3. Даже если Microsoft будет продолжать использовать язык VBScript, чем это принято во всем мире несомненным остается факт, что JavaScript доступен большему числу браузеров и операционных систем.

Типы данных и операторы языка JavaScript

Для написания кодов и их подключения к web-странице необходимо использовать соответствующую команду, к которой относится к языку html которого называют тегом и используется следующим виде

<script>… </script>

В качестве типов данных в языке JavaScript могут использоваться следующие типы:

строковые - любая последовательность символов заключенных в кавычки “Hello world”

числовой – любое число, указанное без кавычек 54 5.29

логический true false

Null null  
объект – это некий программный объект определяемый своими свойствами и методами

Массивы также являются объектами

Функции

Если во время написания кода пользователю необходимо задавать число с помощью текстовых полей формы, то JavaScript такие значение принимает как строку и чтобы значение использовались как число его необходимо превести к числовому типу. Для того чтобы привести переменные к типу, который может использоваться в любом случае необходимо использовать ключевое слово var.

Для преобразования типов используются соответствующие команды.

s = 3 + 3 6

s = 3 + “3” 33

a = 5+ s 533

для преобразования типов из строки в число используется команды parseInt parseFloat

a = parseInt(“5”) a = 5

b = parseFloat (“1.6”) b = 1.6

Для того чтобы выполнять обратное преобразование можно к числовому значению прибавлять пустую строку тогда значение преобразуется в строку.

m = “” + 12 m = “12”

Операторы

Во время составления выражений можно использовать несколько различных операторов и операций

* = присваивание
* +
* -
* \*
* /
* %
* ++
* --
* ==
* !=
* ===
* +=
* -=
* /=
* \*=
* %=
* <<=
* >>=
* >>>=
* $=
* |=
* ^=
* &&
* ||
* ^
* ~
* !

Date

Boolean

Object

Array

Function

Number

String

Операторы и конструкторы в JavaScript

Операторы в языке JavaScript имеют такую же структуру и используются как в С подобных языках (C, C++, C#, Java)

If (условие) {операции}

If (условие) {операции} else {операции 2}

Тернарный оператор

?:

(условие) ? операция\_1: операция\_2

for (i=a; i<=b; i++) { операции }

while (условие) { операции }

do {

операции;

} while (условие)

break continue

for in

for (переменная in объект) {операции}

switch (выражение) {

case 1: операция 1; break;

…

default: операция;

}

try-catch-finally

try{

операции

} catch (e) {

операции

} finally {

операции

}

new with this

Функции

Для создания и использования функций в языке JavaScript используется ключевое слово function после него указывается имя функции далее в круглых скобка перечисляются параметры или аргументы и после внутри фигурных скобках описывается все необходимые операции. Результат функции всегда возвращается с использование ключевого слово return. В языке JavaScript ещё возможен реализация вложенных функций. Бывают случае что после реализации функции во время обращения функции указывается параметры количество, которых меньше тогда для тех параметров, которые были пропущены по умолчанию присваивается пустое значение null. Существует ряд методов, которые предназначены для вызова функции

apply ()

call ()

Данные методы даёт возможность вызвать функции имея только ссылку на неё чтобы возвращаемое значение получить в строковом виде можно использовать методы

toString ()

valueOf ()

Реализация рекурсивных функций в JavaScript допустима

Функции и Массивы

Для создания функций в языке JavaScript используется ключевое слово function после которого указывается название функции и далее в круглых скобках перечисляются все необходимые параметры и после этого внутри фигурных скобок описывается код функции. Функция может реализоваться и с параметром, и без параметра. Функция без параметра реализуется с указанием названия и пустых круглых скобок. Во время реализации функции в его коде можно задавать не только определённые команды JavaScript, но и ещё и других объектов, которые предназначаются управлению различных действий с различными частями веб-страниц. Самые основные объекты, предназначенные для реализации функций расмотрим чуть позже. Такие объекты относятся к языку HTML. В языке JavaScript допускается реализация вложенных функций. Для вызова функции необходимо в нужных частях кода языка HTML необходимо установить ссылку на определённую функцию. Например:

function loadPages () {

parent.frames[1].location.href = “page2.html”;

parent.frames[2].location.href = “instruct.html”;

}

Для определения функции с переходом с нажатием на определённую строку можно использовать следующий способ:

<a href = “javascript : loadPages ()”>Next</a>

<href= “javascript : void loadPages ()”>Next</a>

Существуют случае чтобы исключали по мере каких-либо операций в атрибуте href следует указать пустую функцию после оператора чтобы сработал обработчик событий onclick с это целью необходимо указать строчку:

<href= “javascript: void(0)”>Next</a> // то же самое, что и return 0

Массивы в языке JavaScript определяются с помощью переменной и поэтому при создании массива переменной присваивается объект нового массива. Для объявления массива используется ключевое слово new после которого следует ключевое слово объект Array() у которого имеется один необязательный параметр. Этот параметр играет роль изначального количества элементов в массиве. Объявление массива осуществляется с использованием ключевого слово var:

var stateNumbers = new Array ();

var stateNumbers = new Array (10);

Как и в других языках индексы элементов или порядковые номера элементов по умолчанию начинаются с нуля. Для определения количество элементов в массиве можно использовать ключевое слово length. При реализации веб-страниц в качестве элементов массива можно использовать не только значение определенного типа, но и ещё какие-либо объекты. К примеру, элементы формы, тогда в таком случае можно с легкостью реализовать динамические коды скриптов для однотипных объектов и использовать их с указанием индексов. При разработке веб-страниц существует несколько объектов которые можно использовать с целью реализации различных скриптов. Основными такими объектами являются window, location, navigation, document. Эти объекты в свою очередь составляют некоторую иерархическую структуру, имеющую следующий вид:

window

screen

navigator

location

history

document

Все перечисленные объекты в свою очередь имеют определенные методы и свойства. Использование свойств от методов отличается следующим образом

window.имя\_свойства

window.имя\_мотода(параметры)

для того чтобы создать новое окно с определённым названием которое можно использовать в дальнейшем для других действий можно использовать следующий код

var myWindow = window.open(“define.html” , “def”, “height=200, width=400”);

myWindow.close ();

self

Основные свойства и методы окна

status – свойства через которое представляется строка состояния в нижней части окна браузера.

alert() – метод генерирует диалоговое окно отображающий текст переданный в качестве параметра имеет предупреждающую иконку и одну кнопку ok.

confirm() – генерирует диалоговое окно выбора действий.

prompt() – генерирует диалоговое окно запроса, то есть окно ввода в котором выводиться текст передаваемый в первый параметр в поле ввода в котором может показываться значение по умолчанию переданное через второй параметр и кнопки отмена и ok.

onload – предназначена для реагирования на многочисленные события генерируемые системой или пользователем.

Объект location предназначен только для хранения адреса ссылки на документ. Наиболее часто используемые свойства данного объекта является свойства href при помощи которого задаётся адрес ссылки.

Объект navigator предназначен для определения и передаче на сервер сведение об используемом браузере и системе компьютера (ОС).

Все эти операции выполняются с помощью определённых свойств данного объекта, например через свойство userAgent возвращается основные сведение о программной среде браузера в виде текстовой строки.

appVersion для определении версии программного обеспечения или браузера.

appName для получения названия приложения или браузера.

Объект document, в данном объекте хранится все реальные содержимые страницы свойства и методы объекта воздействуют на внешний вид и содержимое страницы загруженный в окне.

Основные свойства и методы document:

1. write() позволяет динамически создавать содержимое страницы в процессе его загрузки;
2. forms() для использования объектов форм и его элементов на странице с указанием строкового значения внутри квадратных скобок forms[] можно установить новое значение в качестве имени формы. С помощью целых чисел, написанных внутри квадратных скобок, можно получить доступ к определённой форме по его порядковому номеру;
3. images() предназначен для использовании коллекции рисунков или картинок в document. Порядковые номера рисунков задаются после метода внутри квадратных скобок к которому необходимо получить доступ image[];
4. createElement() позволяет создать в памяти совершенно новый объект;
5. createTextNode() создаёт новую текстовый узел в текстовом содержимом document;
6. getElementById() предназначен для получения доступа к элементам с использованием строкового названия элемента;

Объектная модель документа

Объектная модель документа и его развитие являются самым захватывающим и запутанным процессом в браузерах поддерживающий сценарий. Самая первая объектная модель документа было предложена в браузере NN2 (Netscape Navigator 2). С момента появление первой объектной модели документа (DOM – Document Object Model) до самого последнего поколения в нем добавились множество различных характеристик разных типов и ряд средств, которые являются уникальными и используются в большинстве браузерах. Существуют различные стандартные модели объектной модели документа. Разработчиками этих стандартов является World Wide Web Consortium (W3C). Существует следующая иерархическая структура объектной модели документа которой является наиболее современным и широко используемым.

ФОТО

Как и в других языках программирования, которые поддерживают визуализацию так же язык JavaScript тоже вовремя использовании элементов формы существуют различные свойства и методы для объектов элемента формы с использованием которых можно реализовать выполнение различных операций. Такие свойства и методы имею почти такие же аналогичные названия, как и других языках программирования.

События основные понятия и способы использования

Наиболее важной стороной создания не навязчивых DOM сценариев является использование динамически связанных событий конечная цель написания полезного кода JavaScript состоит в получении веб-страницы, работающей для пользователей независимо от того какой браузер, они используют или на какой платформе работают. Для достижение этой цели устанавливается заданный набор свойств, которые будут использоваться и исключаются все браузеры, которые их не поддерживают. Для браузеров, не поддерживающих эти свойства, предлагаются работоспособные, но менее интерактивные версии веб страниц. Преимущество такой организации взаимодействия JavaScript HTML включают более совершенный код более понятные веб страницы и лучшее взаимодействие с пользователями. Всего этого можно достичь за счет использования событий DOM для улучшения взаимодействия происходящего веб приложениях. При разработке веб-страниц ранее использовались такие события как load, click, mouseover и другие, которые являются довольно старыми типами событий и хорошо поддерживаются всеми браузерами. По мере развития веб платформ появились более мощные прикладные интерфейсы на основе которой количество событий существенно увеличилось. Существуют события для работы с мышью, с клавиатурой, с html-формами и с объектом window которые относятся к категории старых типов событий.

Например

Submit, reset, click, change, focus, blur относятся к событиям формы.

События mousemove, mouseup, mousedown относятся к событиям мышки.

Событие keypress, keyup, keydown к событиям клавиатуры.

В основном все используемые события могут быть сгруппированы в некоторые обобщенные категории, которые являются:

1. Аппаратно-зависимые события ввода.

События из этой категории непосредственно связанны с конкретными устройствами ввода (мышь, клавиатура). Примеры: mousemove, mouseup, mousedown, keypress, keyup, keydown, touchmove, gesturechange.

1. Аппаратно-независимые события ввода.

Эти события не связанны с каким-то определенным устройствам. Например: click, textinput.

1. События пользовательского интерфейса.

Это высокоуровневые события, которые часто возникают в элементах html-форм составляющих пользовательский интерфейс веб-приложения. Пример: focus, change, submit.

1. Событие изменения состояния.

Некоторые события, относящиеся к данной категории не связанны непосредственно с деятельностью пользователя, но имеют отношения к сетевым операциям браузера указывают на переход к другому этапу операции или на изменение состояния. К событиям данной категории относятся: load, online, offline, loadstart, progress, loadend.

1. Прикладные события.

Некоторые прикладные интерфейсы, определяемые стандартом html-5 и связанными с ними спецификациями, включают собственные типы событий. Пример: интерфейс drag-and-drop определяет такие события как dragstart, dragenter, dragover, drop. Для тегов video и sound: waiting, playing, seeking, volumechange.

1. Обработчик ошибок и событий от таймеров.

Обработчик ошибок и событий от таймеров являются частью асинхронной модели программирования в клиентском JavaScript и похоже на обработчики обычных событий.

Стандарт html-5 и связанные с ним стандарты определяют основу новых API для новых веб-приложений. Многие из них могут быть доступны к использованию, а некоторые из них пока не реализованы во всех браузерах. Например: одно из широко рекламируемых особенностей html является возможность использования элементов audio и video. Основные события, которые используются для выполнения операций с этими элементами это canplay, canplaythrught, durationchange, emptied, ended, loadeddata, loadedmetadata, loadstart, pause, play, playing, progress, ratechange, seeked, seeking, stalled, suspend, timeupdate, volumechange, waiting.

События кода в JavaScript чаще всего используется в виде вызовом метода, если необходимо использовать событие условных выражений определить момент наступления и при этом выполнять соответствующие операции. Например:

while (! window.loaded() )

{

document.getElementById(“body”).style.border= “1px solid #000”;

}

function loaded()

{

document.getElementById(“body”).style.border= “1px solid #000”;

}

window.onload=loaded;

input text