Динамический HTML

HTML является языком разметки и не имеет каких-либо средств, которые могли бы использоваться для изменения содержимого страницы. Эту проблему решает использование языка DHTML (Dynamic HTML), поддерживающего средства программирования на клиентской стороне. Для этого в DHTML встроена поддержка скриптового языка JavaScript (должен поддерживаться браузером). Возможности динамического управления содержимым становятся доступны при внедрении в веб-страницу кода JavaScript .

Это делается с помощью тега script, размещаемого в нужном месте веб-страницы и которым выделяют начало и конец исходного кода или указывают на подгружаемый из сети файл с исходным кодом: Для внедрения скриптов в веб-документ используется контейнерный тег <script>...</script>, внутри которого записываются команды JavaScript (в общем случае и ряда других яыков: VBScript, php, tcl/tk ...).

Если этот тег ипользуется в теле документа (внутри тега body), то исполнение скрипта осуществляется по мере отображения веб-страницы в браузере. Если же контейнер script описан внутри тега head, то обращение к скрипту возможно только явным образом, например, через вызов функции.

```
<!-- внедрение скрипта в разметку -->

<script type="text/javascript"> код скрипта </script>
```

Имеется возможность вынести код JavaScript в отдельный файл (как правило с расширением .js), который затем подключить к документу следующим образом:

```
<html> <head>

<!-- загрузка скрипта из внешнего файла -->

<script type="text/javascript">
src="http://example.com/scripts.js">

</script> </head>
```

... Такой способ внедрения скриптов позволяет создавать своего рода библиотеки скриптов и использовать их на всех страницах сайта.

Асинхронные скрипты: defer/async

Браузер загружает и отображает HTML постепенно. Особенно это заметно при медленном интернет-соединении: браузер не ждёт, пока страница загрузится целиком, а показывает ту часть, которую успел загрузить.

Если браузер видит тег <script>, то он по стандарту обязан сначала выполнить его, а потом показать оставшуюся часть страницы.

Кардинально решить эту проблему помогут атрибуты async или defer:

Атрибут async

Поддерживается всеми браузерами, кроме IE9-. Скрипт выполняется полностью асинхронно. То есть, при обнаружении <script async src="..."> браузер не останавливает обработку страницы, а спокойно работает дальше. Когда скрипт будет загружен – он выполнится.

Атрибут defer

Поддерживается всеми браузерами, включая самые старые IE. Скрипт также выполняется асинхронно, не заставляет ждать страницу, но есть два отличия от async.

Первое – браузер гарантирует, что относительный порядок скриптов с defer будет сохранён.

То есть, в таком коде (с async) первым сработает тот скрипт, который раньше загрузится:

```
<script src="1.js" async></script>

<script src="2.js" async></script>
```

А в таком коде (с defer) первым сработает всегда 1.js, а скрипт 2.js, даже если загрузился раньше, будет его ждать.

```
<script src="1.js" defer></script>

<script src="2.js" defer></script>
```

Поэтому атрибут defer используют в тех случаях, когда второй скрипт 2.js зависит от первого 1.js, к примеру – использует что-то, описанное первым скриптом.

Второе отличие – скрипт с defer сработает, когда весь HTML-документ будет обработан браузером.

Например, если документ достаточно большой, то скрипт async.js выполнится, как только загрузится – возможно, до того, как весь документ готов. А defer.js подождёт готовности всего документа.

Это бывает удобно, когда мы в скрипте хотим работать с документом, и должны быть уверены, что он полностью получен.

Атрибуты async/defer – только для внешних скриптов

Атрибуты async/defer работают только в том случае, если назначены на внешние скрипты, т.е. имеющие src. При попытке назначить их на обычные скрипты <script>…</script>, они будут проигнороированы.

Синтаксис JavaScript

Язык JavaScript синаксически близок к языкам C/C++, Java, PHP и другим C-подобным языкам. Поэтому для тех, кто знаком с такими языками не составит труда разобраться с основными языковыми конструкциями.

Переменные

Для объявления переменных используется ключевое слова var. Переменные можно сразу инициализировать. Можно объявить несколько переменых сразу, разделив их запятыми.

```
var color = "#FFF", fsize = 1024 , total_count = 0, i;

var average = null; var c = 3; d = 0; //Ошибка!
```

Непроинициализированые переменные будут иметь неопределенное значение (undefined). Объявлять переменные можно в любом месте скрипта, но до первого обращения Типы данных переменным в javascript назначаются автоматически. Так же автоматически выполняется приведение типов.

Объявления массивов данных могут выполняться статически и динамически. Индексирование элементов начинается с нуля. Элементы массива могут быть проинициализированы при создании.

```
var weekdays = ["Пн", "Вт", "Ср", "Чт", "Пт"]; //
статический массив из пяти элементов

// динамическое объявление массива путем создание экземпляра
встроенного класса Array

var myarr; myarr = new Array(256);
```

```
myarr[0] = 255; myarr[1] = 254;
var x = myarr[7];
```

Операторы

Комментарии - предназначены для пояснения фрагментов кода или исключения фрагментов кода из обработки. Игнорируются при выполнении программы.

```
// Это однострочный комментарий.
/* Это многострочный комментарий. Он может объединять
несколько строк и его можно использовать в любом месте
программы */
```

Условный оператор if предназначен для ветвления программы в зависимости от значения (true | false) логического выражения:

```
if (условие) {блок операторов1} [else {блок операторов2}]
```

Оператор выбора switch также как и условный оператор предназначен для выполнения ветвтления алгоритма, но позволяет анализировать множество возможных результатов проверки условия.

Оператор break позволяет прервать выполнение оператора, если его не указать, то будут выполнены все последующие операторы.

```
switch (условие) {
case значение1 : {блок операторов1; break;}
case значение2 : {блок операторов2; break;}
case значение3 : {блок операторов3; break;} ...
[default : {блок операторов по умолчанию};]
}
```

Цикл со счетчиком for. Используется для циклов с заданным числом итераций (примечание: на самом деле конструкция for может использоваться и для построения любых циклов, все зависит от того, как и какие значения указаны в качестве параметров цикла).

```
for ([начальное значение]; [условие]; [приращение]) {блок операторов;}
```

Цикл с постусловием do...while. Выполняется, пока условие является истинным. Всегда выполняется хотя бы один раз.

```
do {блок операторов;} while (условие)
```

Цикл с предусловием while. Выполняется, если условие является истинным. Может не выполниться ни разу.

```
while (условие) {блок операторов;}
```

Операторы break и continue -используются для прерывания выполнения цикла или завершения текущей итерации.

Поэлементный цикл for (... in ...) применяется для выполнения команд над каждым элементом массива или коллекции.

```
for (переменная in массив|объект|коллекция) {блок операторов;}
```

Функции. Представляют возможность создания повторно используемого кода. Функция может принимать параметры и возвращать значение в вызывающую программу. Если в функции не предусмотрен возврат значения, то она работает как процедура.

```
function имяФункции ([список параметров]) { блок операторов; [return возвращаемоеЗначение;] }
```

Встроенные объекты JavaScript

JavaScript предлагает разработчику некоторый набор библиотечных функций, оформленных в виде свойств и методов различных объектов. Обращение к этим свойствами методам - через точечную нотацию. Кратко рассмотрим некоторые встроенные объекты JavaScript.

Объект Math

Объект Math представляет математические константы и функции. Константы представлены свойствами объекта, а функции - его методами. Их назначение понятно из названий:

Свойства: E, LN2, LN10, LOG2E, LOG10E, PI, SQRT1_2, SQRT2.

Методы: abs, acos, asin, atan, ceil, cos, exp, floor, log, max, min, pow, random, round, sin, sqrt, tan.

Примеры использования:

```
var r = 1.8, theta = 30, a, x, y, D;

var rnd = Math.round(Math.random()*99)+1;

D = Math.PI*r*r; x = Math.max(1,7,5,9);

y = Math.pow(2,10);

with (Math) { y = r*sin(theta); x = r*cos(theta); }
```

Объект string

Встроенный объект string представляет литерал (строку символов), заключенный в одинарные или двойные кавычки или вычислимое выражение, которое может быть интерпретировано как строка. Для объекта string определены следующие свойства и методы:

Свойства: length (длина строки).

Методы (не все): anchor (якорь), bold (полужирное начертание), charAt (символ в позиции), fixed (преформат), fontcolor (цвет шрифта), fontsize (размер шрифта), indexOf (индекс первого вхождения символа), italics (курсив), link (гиперссылка), substring (подстрока), toLowerCase (строчные), toUpperCase (прописные).

Несколько примеров использования объекта string:

```
var hello = "Hello, ", w = "World!"; var str = hello + w; // конкатенация строк

document.write(str.bold());
document.write(str.toUpperCase());
document.write(hello.fontsize(6));

document.write(hello.substring(0,3));

document.write(hello.link("http://localhost"));
```

```
document.write(w.indexOf("l"));

alert("string lehgth = " + str.length);
```

Вывод данных в JavaScript

Результаты работы скрипта JavaScript могут быть отображены по меньшей мере двумя способами: в окно текущего веб-документа и в диалоговое окно. Для вывода данных в веб-документ можно использовать метод write объекта document. Примеры использования этого метода приведены при описании объекта string. Еще несколько примеров:

```
document.write("Hello, World!"); document.write("<h1>>Hello, World!</h1>"); // внедрение HTML в JavaScript

// обратите внимание на использование вложенных кавычек
document.write("<p><a href='http://localhost'>Link to localhost</a></p>");
```

Для вывода различных информационных сообщений, не относящихся напрямую к содержимому веб-страницы следует использовать метод alert, представляемый объектом window. Этот метод выводит модальное диалоговое окно, блокирующее выполнение скрипта до нажатия пользователем кнопки в этом диалоге.

```
alert(“Hello World!”);
```

Практическое задание

Выбрать 3 задания из списка и реализовать их в виде функции. Функции должны принимать на вход все необходимые данные (например, последовательности цифр или символов). Вывод реализовать в любом удобном формате.

1. Найти сумму элементов последовательности.
2. Найти минимальный элемент в последовательности.
3. Найти второй по величине элемент в последовательности.
4. Сколько раз в последовательности встречается заданное число?
5. Известно сопротивление каждого из элементов электрической цепи. Все элементы соединены параллельно. Определить общее сопротивление цепи.
6. Найти произведение элементов последовательности.
7. Найти сумму модулей элементов последовательности.

8. Сколько соответствующих элементов двух последовательностей с одинаковым количеством элементов совпадают?
9. Вычислить сумму квадратов элементов последовательности.
10. Определить среднее арифметическое элементов последовательности.
11. Определить среднее геометрическое элементов последовательности, содержащей положительные числа.
12. Найти произведение модулей элементов последовательности.
13. Определить, сколько раз встречается минимальный элемент в последовательности.
14. Определить, сколько раз встречается максимальный элемент в последовательности.
15. Выбрать максимальный из модулей элементов последовательности.
16. Сколько нулей в последовательности?
17. Напечатать true, если элементы последовательности упорядочены по возрастанию, и false в противном случае.
18. В последовательности натуральных чисел подсчитать их количество, оканчивающихся заданной цифрой.
19. В заданной последовательности определить максимальное количество подряд идущих положительных чисел.
20. Найти сумму тех членов последовательности, которые оканчиваются на заданную цифру.
21. Найти сумму чётных элементов последовательности целых чисел.
22. Определить количество нечётных отрицательных элементов в последовательности целых чисел.
23. Указать минимальный элемент среди нечётных чисел в последовательности, содержащей целые величины.
24. Найти сумму номеров тех элементов последовательности, которые отрицательны. Нумерацию элементов начать с единицы.
25. Определить количество ненулевых элементов последовательности.
26. Найти разность максимального и минимального элементов последовательности.
27. Между какими степенями двойки расположены все положительные элементы последовательности?
28. Если сумма элементов последовательности отрицательна, вывести –1, если положительна — 1, если равна нулю, то 0.
29. Найти наибольший общий делитель последовательности натуральных чисел.
30. Определить количество раз, которое меняется знак в последовательности чисел, отличных от нуля.
31. Сколько пробелов в данной последовательности символов?