

Основы Интернет-технологий

Введение в JavaScript

Лекция 12

JavaScript

JavaScript — объектно-ориентированный скриптовый язык программирования. Он обычно используется как встраиваемый язык для программного доступа к объектам приложений и наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам, под которой подразумеваются следующие задачи:

- добавление "интеллекта" в страницу (определение типа и версии браузера, получение тек. даты и времени и т.п.);
- отслеживание и обработка событий, возникающих на странице в результате действий пользователя;
- проверка допустимости данных форм перед отправкой их на сервер;
- изменение содержимого или оформления элементов страницы;
- изменение структуры страницы;
- создание сложных анимационных эффектов на странице (сюда можно отнести "хвост" за курсором мыши, выпадающие или раскрывающиеся меню и т.п.);
- работа с дополнительными окнами (открытие, закрытие, изменение их размера, положения и содержимого);

JavaScript

JavaScript — это язык программирования, основанный на объектах: и языковые средства, и возможности среды представляются объектами, а сценарий (программа) на JavaScript — это набор взаимодействующих объектов.

Объект JavaScript — это неупорядоченный набор свойств, каждое из которых имеет нуль или более атрибутов, которые определяют, как это свойство может использоваться. Например, если атрибуту свойства `ReadOnly` (неизменяемый) присвоено значение `true` (истина), то все попытки программно изменить значение этого свойства будут безрезультатны.

Свойства — это контейнеры, которые содержат другие объекты, примитивные значения и методы:

Примитивное значение — это элемент любого из встроенных типов: `Undefined`, `Null`, `Boolean`, `Number` и `String`;

объект — это элемент еще одного встроенного типа `Object`;

метод — функция, ассоциированная с объектом через свойство.

JavaScript

JavaScript содержит несколько встроенных объектов, таких, как **Global, Object, Error, Function, Array, String, Boolean, Number, Math, Date, RegExp.**

Кроме того, JavaScript содержит набор *встроенных операций*, которые, строго говоря, не обязательно являются функциями или методами, а также *набор встроенных операторов*, управляющих логикой выполнения программ.

Синтаксис JavaScript в основном соответствует синтаксису языка Java, но упрощен в сравнении с ним, чтобы сделать язык сценариев легким для изучения.

Так, к примеру, декларация переменной не содержит ее типа, свойства также не имеют типов, а декларация функции может стоять в тексте программы после ее вызова.свойство.

JavaScript

Программа (сценарий) на языке JavaScript — это текст, состоящий из операторов, блоков, т. е. взаимосвязанных наборов операторов, и комментариев. Операторы могут содержать переменные, константы и выражения. С

ледующий пример начинается с определения функции, которое состоит из блока, содержащего два оператора. За определением следуют два оператора, не образующих блока.

```
function convert(inches)
{
    cm = inches * 2.54;    // Эти два оператора заключены в блок.
    meters = inches / 39.37;
}
convert(inches); // Эти два оператора не образуют блока.
km = meters / 1000;
```

Переменные JavaScript

Переменные используются в качестве символических имен, принимающих различные значения. Имена переменных задаются идентификаторами. *Переменная создается в момент ее декларации.*

JavaScript позволяет *декларировать переменную* двумя способами:

- С помощью ключевого слова `var`, например, `var x;` или `var x = 21;`
- Просто присваиванием переменной значения, например `x = 21;`

Если декларация переменной не содержит присваивания ей значения, то ее значением считается *undefined*. В самом общем виде декларация переменных имеет вид:

```
var      идентификатор[=инициализатор]?[,идентификатор  
[=инициализатор]?]
```

Инициализатор — это любое выражение, значение которого присваивается переменной при ее создании.

Каждая переменная имеет свою область действия, в которой она определена и в которой она имеет значение. Область действия переменной определяется положением ее декларации в тексте программы.

Контекст исполнения

Существуют три типа исполняемого кода JavaScript, называемых **контекстом исполнения**:

- **Глобальный контекст**, т. е. исходный текст сценариев, не включая тела функций.
- **Локальный контекст**, т. е. исходный текст сценариев, являющийся телом функ-ции, а также аргумент конструктора встроенного объекта Function. Точнее говоря, если последним параметром конструктора Function является строка текста, то она интерпретируется как тело функции.
- **Eval-контекст**, т. е. аргумент метода eval. Если параметром метода eval является строка текста, то она интерпретируется как программа на языке JavaScript, имею-щая тот же контекст, в котором был вызван этот метод.

Константы

Константы используются для задания постоянных значений. В JavaScript имеется несколько типов констант, соответствующих его встроенным типам, а именно:

- нулевая константа null типа Null;
- логические константы true (истина) и false (ложь) типа Boolean;
- строковые константы типа String, например, "Привет всем!";
- числовые константы типа Number, например, 21 или 3.1415926.

Выражения и операции

Выражения в JavaScript, как и в других языках программирования, представляют со-бой комбинации переменных, констант и операций, дающие осмысленный результат. Этот результат может быть числом, текстовой строкой, логическим значением или объектом. Со-ответственно все выражения JavaScript подразделяются на арифметические, строковые, логические и объектные.

Существует два типа выражений: те, которые присваивают значение некоторой пере-менной (например, $x = 2 + 3$), и те, которые просто имеют некое значение (например, $2 + 3$). Первый тип выражений называется операциями присваивания.

Все **операции в JavaScript** подразделяются на следующие:

- операции сравнения;
- арифметические операции;
- битовые операции;
- логические операции;
- строковые операции;
- операции присваивания;
- прочие операции. Number, например, 21 или 3.1415926.

Операции сравнения

Операции сравнения сравнивают два операнда и возвращают логическое значение, означающее результат этого сравнения. Строки сравниваются в лексикографическом порядке в кодировке Unicode. Если типы операндов различны, то делается попытка преобразовать их к одному типу. При этом:

- Операции "больше", "меньше", "не больше" и "не меньше" сначала пытаются преобразовать операнды в числа, а, если это невозможно, то в строки, а затем производят их сравнение.
- Операции "равно" и "не равно" пытаются преобразовать операнды в строки, затем в числа и в логические значения, а затем производят их сравнение.
- Операции "тождественно" и "не тождественно" не преобразовывают типы данных: два операнда считаются тождественно равными, если они имеют одинаковые типы и одинаковые значения.

Арифметические операции

Арифметические операции применяются к числовым операндам и возвращают числовое значение, означающее результат операции. Если типы операндов различны, то делается попытка преобразовать их к числовому типу.

При этом:

- Операция "сложение" выполняется только тогда, когда оба операнда являются числами или логическими значениями. Если хотя бы один операнд является строкой, то производится конкатенация строк.
- Остальные операции преобразуют операнды в числа, а затем выполняют операцию.
- Операции "инкремент" и "декремент" применяются только к переменным.

Битовые и логические операции

Битовые операции применяются к числовым операндам, представленным как двоичные числа (т. е. как цепочки из 32 битов), и возвращают числовое значение, означающее результат операции. Перед выполнением операции операнды преобразуются в целые числа, а результат операции преобразуется в Number.

Логические операции применяются к логическим операндам и возвращают логическое значение, означающее результат операции. Если типы операндов различны, то делается попытка преобразовать их к логическому типу.

Арифметические операции

применяются к числовым операндам и возвращают числовое значение, означающее результат операции. Если типы операндов различны, то делается попытка преобразовать их к числовому типу.

При этом:

- Операция "сложение" выполняется только тогда, когда оба операнда являются числами или логическими значениями. Если хотя бы один операнд является строкой, то производится конкатенация строк.
- Остальные операции преобразуют операнды в числа, а затем выполняют операцию.
- Операции "инкремент" и "декремент" применяются только к переменным.

Строковые операции

На сегодняшний день JavaScript поддерживает единственную **строковую операцию**, а именно конкатенацию строк, которая обозначается символом "+". Если хотя бы один операнд является строкой, то результатом операции является слияние строк-операндов.

Операции присваивания присваивают левому операнду результат операции, который определяется правым операндом и самой операцией. Базовая операция присваивания имеет вид $a = b$, что означает: присвоить переменной a значение операнда b .

Прочие операции включают большой набор разнообразных операций, к которым относятся такие операции, как условная операция, операция запятая, операция delete, операция new и другие. Более подробно узнать о них можно по одной из предложенных в конце раздела ссылок.

Операторы

Управление последовательностью действий в ходе выполнения сценария осуществляется с помощью операторов. JavaScript содержит стандартный набор операторов, унаследованный от языков C++ и Java, а именно:

- условный оператор `if...else`;
- оператор выбора `switch`;
- операторы цикла `for`, `while`, `do...while`, `break` и `continue`;
- оператор итерации `for...in`;
- оператор указания объекта `with`;
- операторы обработки исключений `try...catch` и `throw`;
- операторы декларации функций `function` и возврата из функции `return`.

Любое выражение JavaScript также является оператором.

Условный оператор

Условный оператор `if...else` позволяет проверить определенное условие и, в зависимости от его истинности, выполнить ту или иную последовательность операторов.

Он имеет две формы:

`if (условие) оператор1`

`if (условие) оператор1 else оператор2`

Здесь условие — это любое выражение, значение которого может быть преобразовано к логическому типу, оператор1 и оператор2 — любые группы операторов JavaScript; если эти группы содержат более одного оператора, то они должны быть заключены в фигурные скобки `{}`.

Оператор выбора

Оператор выбора **switch** выполняет ту или иную последовательность операторов в зависимости от значения определенного выражения. Он имеет вид:

```
switch (выражение) {  
  case значение:  
    операторы  
    break;  
  case значение:  
    операторы  
    break;  
  ...  
  default:  
    операторы  
}
```

Здесь выражение — это любое выражение, значение — это возможное значение выражения, а операторы — любые группы операторов JavaScript.

Оператор цикла for

Цикл — это последовательность операторов, выполнение которой повторяется до тех пор, пока определенное условие не станет ложным. JavaScript содержит три оператора цикла: `for`, `while` и `do...while`, а также операторы `break` и `continue`, которые используются внутри циклов. Близок к операторам цикла и оператор итерации `for...in`, используемый при работе с объектами.

Оператор цикла `for` имеет вид:

`for` (инициализация; условие; изменение) оператор

Здесь инициализация и изменение — это любое выражения, условие — любое выражение, значение которого может быть преобразовано к логическому типу, оператор — любая группа операторов JavaScript; если эта группа содержит более одного оператора, то она должна быть заключена в фигурные скобки `{}`. Инициализация может содержать декларацию переменной.

Операторы цикла с условием

Оператор цикла while имеет вид:

while (условие) оператор

Здесь условие — любое выражение, значение которого может быть преобразовано к логическому типу, оператор - любая группа операторов JavaScript; если эта группа содержит более одного оператора, то она должна быть заключена в фигурные скобки {}.

Оператор цикла do...while имеет вид:

do оператор while (условие)

Здесь условие — любое выражение, значение которого может быть преобразовано к логическому типу, оператор - любая группа операторов JavaScript; если эта группа содержит более одного оператора, то она должна быть заключена в фигурные скобки {}. Последовательность операторов, выполнение которой повторяется до тех пор, пока определенное условие не станет ложным.

Оператор for...in

Оператор **for...in** выполняет заданные действия для каждого свойства объекта или для каждого элемента массива.

Он имеет вид:

for (переменная in выражение) оператор

Здесь переменная — это декларация переменной, выражение — любое выражение, значением которого является объект или массив, оператор — любая группа операторов JavaScript; если эти группа содержит более одного оператора, то она должны быть заключена в фигурные скобки {}.

Операторы `break` и `continue`

Оператор `break` прерывает выполнение текущего цикла, оператора `switch` или помеченный оператор и передает управление оператору, следующему за прерванным. Этот оператор может употребляться только внутри циклов `while`, `do...while`, `for` или `for...in`, а также внутри оператора `switch`. Он имеет две формы:

`break`

`break метка`

Оператор `continue` завершает текущую итерацию текущего цикла или цикла, помеченного соответствующей меткой, и начинает новую итерацию. Этот оператор может употребляться только внутри циклов `while`, `do...while`, `for` или `for...in`. Он имеет две формы:

`continue`