Введение в Javascript. Структура кода. Введение в типы данных. Команды вывода на экран.

Что такое JavaScript?

JavaScript - язык для составления скриптов, разработанный фирмой Netscape. С помощью JavaScript Вы можете легко создавать интерактивные Web-страницы. Далее, вы увидите, что можно сделать с помощью JavaScript, и даже более того - увидите, *как* это сделано.

JavaScript - это не Java!

- Многие люди считают, что JavaScript - это то же самое, что и Java, лишь потому, что эти языки имеют схожие названия. На самом деле это не так. Полагаю, что сейчас будет излишне показывать вам все различия между этими языками - так что запомните лишь то, что JavaScript - это не Java.

Запуск JavaScript

Что необходимо сделать, чтобы запускать скрипты, написанные на языке JavaScript? JavaScript может выполняться не только в браузере, а где угодно, нужна лишь специальная программа — интерпретатор. Процесс выполнения скрипта называют «интерпретацией».
Конечно же, перед тем, как приступить к работе с Java, вы должны познакомиться с основами другого языка - HTML.
Для разработки также обязательно нужен хороший редактор.

Выбранный вами редактор должен иметь в своем арсенале:

- Подсветку синтаксиса.
- Автодополнение.
- «Фолдинг» (от англ. folding) — возможность скрыть-раскрыть блок кода

Что умеет JavaScript?

- Создавать новые HTML-теги, удалять существующие, менять стили элементов, прятать, показывать элементы и т.п.
- Реагировать на действия посетителя, обрабатывать клики мыши, перемещения курсора, нажатия на клавиатуру и т.п.
- Посылать запросы на сервер и загружать данные без перезагрузки страницы (эта технология называется "AJAX").
- Получать и устанавливать cookie, запрашивать данные, выводить сообщения

Что не умеет JavaScript?

- JavaScript не может читать/записывать произвольные файлы на жесткий диск, копировать их или вызывать программы. Он не имеет прямого доступа к операционной системе. Современные браузеры могут работать с файлами, но эта возможность ограничена специально выделенной директорией — *«песочницей»*. Возможности по доступу к устройствам также прорабатываются в современных стандартах и частично доступны в некоторых браузерах.
- JavaScript, работающий в одной вкладке, не может общаться с другими вкладками и окнами, за исключением случая, когда он сам открыл это окно или несколько вкладок из одного источника (одинаковый домен, порт, протокол).

Структура кода

```
<!DOCTYPE HTML>
<html>
<head>
	<!-- Тег meta для указания кодировки -->
	<meta charset="utf-8">
 </head>
	<body>
	<p>Начало документа...</p>
	 <script>
		alert( 'Привет, Мир!' );
	</script>
	<p>...Конец документа</p>
</body>
</html>
```

Внешние скрипты, порядок исполнения

Если JavaScript-кода много — его выносят в отдельный файл, который подключается в HTML:

```
<script src="/path/to/script.js"></script>
```

В одном теге SCRIPT нельзя одновременно подключить внешний скрипт и указать код.

Вот так не сработает:

```
<script src="file.js">
alert(1); // так как указан src, то внутренняя часть тега игнорируется
</script>
```

Внешние скрипты, порядок исполнения

Нужно выбрать: либо SCRIPT идёт с src, либо содержит код. Тег выше следует разбить на два: один — с src, другой — с кодом, вот так:

```
<script src="file.js">
</script>
<script>
    alert( 1 );
 </script>
```

Асинхронные скрипты: defer/async

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>

Точка с запятой

Точку с запятой *во многих случаях* можно не ставить, если есть переход на новую строку.

Так будет работать:

```
alert('Привет')
alert('Мир')
```

Однако, важно то, что «во многих случаях» не означает «всегда»! Поэтому в JavaScript рекомендуется точки с запятой ставить. Сейчас это, фактически, стандарт, которому следуют все большие проекты.

Комментарии

Однострочные комментарии начинаются с двойного слэша //. Текст считается комментарием до конца строки:

```
// Команда ниже говорит "Привет"
alert( 'Привет' );
alert( 'Мир' ); // Второе сообщение выводим отдельно
```

Многострочные комментарии начинаются слешем-звездочкой "/*" и заканчиваются звездочкой-слэшем "*/", вот так:

```
/* Пример с двумя сообщениями. Это - многострочный комментарий. */
alert( 'Привет' );
alert( 'Мир' );
```

Введение в типы данных

Переменная состоит из имени и выделенной области памяти, которая ему соответствует.

Для *объявления* или, другими словами, *создания переменной* используется ключевое слово var. После объявления, можно записать в переменную данные. Эти данные будут сохранены в соответствующей области памяти и в дальнейшем доступны при обращении по имени:

```
var message;
message = 'Hello!';
alert( message ); // выведет содержимое
                  переменной
```

Число «number»

```
var n = 123;
n = 12.345;
```

Единый тип *число* используется как для целых, так и для дробных чисел.

Существуют специальные числовые значения Infinity (бесконечность) и NaN (ошибка вычислений).

Например, бесконечность Infinity получается при делении на ноль:

```
alert( 1 / 0 ); // Infinity
```

Ошибка вычислений NaN будет результатом некорректной математической операции, например:

```
alert( "нечисло" * 2 ); // NaN, ошибка
```

Эти значения формально принадлежат типу «число», хотя, конечно, числами в их обычном понимании не являются.

Строка «string»

```
var str = "Мама мыла раму";
str = 'Одинарные кавычки тоже подойдут';
```

В JavaScript одинарные и двойные кавычки равноправны. Можно использовать или те или другие.

Тип *символ* не существует, есть только *строка*.

Булевый (логический) тип «boolean»

У него всего два значения: true (истина) и false (ложь).

Как правило, такой тип используется для хранения значения типа да/нет, например:

```
var checked = true; // поле формы помечено галочкой
checked = false; // поле формы не содержит галочки
```

Специальное значение «null»

Значение null не относится ни к одному из типов выше, а образует свой отдельный тип, состоящий из единственного значения null:

```
var age = null;
```

В JavaScript null не является «ссылкой на несуществующий объект» или «нулевым указателем», как в некоторых других языках. Это просто специальное значение, которое имеет смысл «ничего» или «значение неизвестно».

В частности, код выше говорит о том, что возраст age неизвестен.

Специальное значение «undefined»

Значение undefined, как и null, образует свой собственный тип, состоящий из одного этого значения. Оно имеет смысл «значение не присвоено».

Если переменная объявлена, но в неё ничего не записано, то её значение как раз и есть undefined:

```
Var x; alert( x ); // выведет "undefined"
```

Можно присвоить undefined и в явном виде, хотя это делается редко:

```
var x = 123; x = undefined;
alert( x ); // "undefined"
```

В явном виде undefined обычно не присваивают, так как это противоречит его смыслу. Для записи в переменную «пустого» или «неизвестного» значения используется null.

Объекты «object»

Первые 5 типов называют *«примитивными»*. Особняком стоит шестой тип: *«объекты»*. Он используется для коллекций данных и для объявления более сложных сущностей. Объявляются объекты при помощи фигурных скобок {...}, например:

```
var user = { name: "Вася" };
```

Оператор typeof

Оператор typeof возвращает тип аргумента. У него есть два синтаксиса: со скобками и без:

1. Синтаксис оператора: typeof x.
2. Синтаксис функции: typeof(x).

Работают они одинаково, но первый синтаксис короче.

Результатом typeof является строка, содержащая тип:

```
typeof undefined // "undefined"
typeof 0 // "number"
typeof true // "boolean"
typeof "foo" // "string"
typeof {} // "object"
```

Команды вывода на экран: alert

Синтаксис:

alert(сообщение)

alert выводит на экран окно с сообщением и приостанавливает выполнение скрипта, пока пользователь не нажмёт «ОК». Окно сообщения, которое выводится, является *модальным окном*. Слово «модальное» означает, что посетитель не может взаимодействовать со страницей, нажимать другие кнопки и т.п., пока не разберётся с окном. В данном случае – пока не нажмёт на «ОК».

Команды вывода на экран: prompt

Функция prompt принимает два аргумента:

```
result = prompt(title, default);
```

Она выводит модальное окно с заголовком title, полем для ввода текста, заполненным строкой по умолчанию default и кнопками OK/CANCEL. Пользователь должен либо что-то ввести и нажать OK, либо отменить ввод кликом на CANCEL или нажатием Esc на клавиатуре.

Вызов prompt возвращает то, что ввёл посетитель — строку или специальное значение null, если ввод отменён.

Команды вывода на экран: confirm

Синтаксис:

```
result = confirm(question);
```

confirm выводит окно с вопросом question с двумя кнопками: OK и CANCEL.

Результатом будет true при нажатии OK и false – при CANCEL(Esc).

Например:

```
var isAdmin = confirm("Вы - admin?");
alert( isAdmin );
```