Урок 1

Основы языка Javascript

Общее знакомство с JavaScript, создание первого кода и его запуск.

Введение

Язык JavaScript является основным языком программирования в сети Интернет. Перво-наперво он позволяет расширять стандартное поведение пользователя в веб-страницах. Вместо сухих статичных страниц, которые просто показываются пользователю друг за другом, с JavaScript страница будет взаимодействовать с пользователями, откликаться на его действия, получать и применять данные из Интернет и делать многое другое.

JavaScript схож с Java только по названию – их не стоит путать. JavaScript был создан во время одного из пиков популярности Java, что позволило разработчикам языка успешно применить элементы синтаксиса языков семейства C. В целом Java и JavaScript совершенно непохожи.

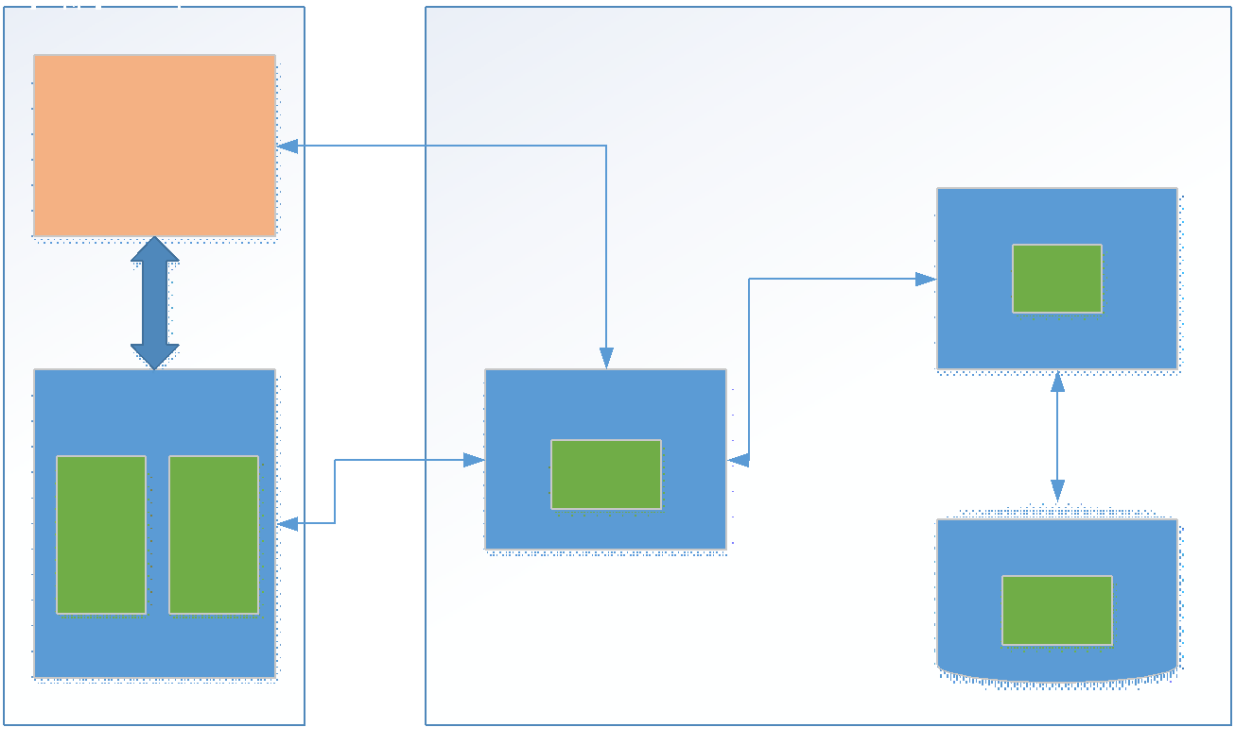
С опытом Вы вполне сможете программировать не шаблонное, а абсолютно новое отображение реакций на действия пользователя на своих страницах. Не так давно для решения таких задач предпочтительной была технология Flash. Однако с выходом HTML5 Flash ушёл в небытие из-за своей громоздкости и уязвимостей. В HTML5 JavaScript – стандартный язык сценариев.

Поскольку язык JavaScript входит в Топ-10 языков программирования, Ваши знания и опыт будут востребованы. JavaScript не только один из самых популярных языков программирования, но он также поддерживается всеми современными браузерами, что делает запуск программ на нём очень лёгким. Но сейчас JavaScript – это не только язык программирования веб-интерфейсов. Всё больше серверных реализаций языка набирает популярность и завоёвывает умы программистов.

Принципы работы JavaScript

Вы уже умеете создавать HTML-страницы с их структурой, умеете наполнять их содержимым, верстать их с макетов и применять стили. Но пока они не обладают уникальным поведением. Всё, что они умеют – это предоставлять навигацию по ссылкам, реагировать на наведение мышки и т.п.

Современные веб-страницы явно имеют больше требований к своему поведению. Они должны быть интерактивными, т.е. должны взаимодействовать с пользователем. Тут-то и вступает в игру JavaScript. Давайте посмотрим, где же находится JavaScript при применении его для оформления поведения веб-страниц.



- HTML определяет контент страниц, их структуры;
- CSS определяет стиль оформления страницы;
- Сервер обрабатывает HTTP-запросы, при необходимости передавая их интерпретатору;
- Интерпретатор формирует HTML-документ, при необходимости обращаясь к БД.

JavaScript на этой знакомой Вам схеме новичок. Как мы уже условились, он умеет определять поведение страниц. Например, странице нужно реагировать в момент, когда пользователь кликает по кнопке «Отправить заказ». Появляется возможность наращивания программного кода, выполняющего различные динамические операции.

Принципы написания кода на JavaScript

Если Вы уже знакомы с некоторыми языками программирования, то точно знаете, что программа появляется в процессе вполне определенных шагов. Сначала пишется код, при необходимости он компилируется, собирается в пакет, устанавливается на компьютер. Язык JavaScript несколько отличается по своему циклу разработки от данной схемы. Разработчик подключает код JavaScript к странице, которая загружается в браузер. После этого уже сам браузер совершает всё необходимое, чтобы выполнить код.

1. Страница создаётся по стандартной схеме. Верстается структура HTML, к ней применяются стили, в неё добавляется содержимое.
2. К странице подключается код JavaScript. На примере Вы увидите, что это происходит ровно также, как и подключение файла CSS. Т.е. код можно вынести в отдельный файл. Впрочем, также, как и с CSS, никто не запрещает писать JavaScript код прямо внутри разметки страницы.
3. При загрузке страницы браузером происходит построение DOM-модели. Браузер находит код JavaScript и сразу же начинает читать его, подготавливая к запуску. Если в коде будет найдена ошибка, браузер будет старается продолжить чтение, избегая отказа от показа пользователю запрошенной страницы.
4. Браузер начинает выполнять JavaScript-код в момент обнаружения. При этом нужно помнить, что код выполняется вплоть до закрытия страницы (а не до окончания её формирования, как

это происходит, например, в PHP). Актуальные версии JavaScript имеют высокую производительность. Однако, надо понимать, что код выполняется на стороне клиента и расходует именно клиентские ресурсы, а не ресурсы сервера.

Теперь научимся подключать JavaScript код к странице. Это делается при помощи HTML-тега <script>. Напишем простую страницу. Пока не будем вникать в код JavaScript, просто подключим его и посмотрим, что произойдет

```
<head>
<script>
setTimeout(reminder, 5000);
function reminder() {
 alert("Так и будешь смотреть на эту скучную страницу?");
}
</script>
</head>
```

При запуске кода в браузере Вы увидите, что раз в 5 секунд браузер будет возвращать Вам оповещение с заданным текстом. Разумеется, это всего лишь малая толика тех возможностей языка, с которыми нам ещё предстоит познакомиться. Однако, этот «Hello, World» пример даёт общее представление о том, как происходит подключение и вызов скрипта.

Бывают случаи, когда код выносят в отдельный файл, подключаемый в HTML-коде:

- Если кода много;
- Если код работает на многих страницах сайта.

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

В качестве значения атрибута src указывается путь к файлу, содержащему скрипт. Браузер скачивает скрипт и выполняет его. Также есть возможность указать скрипт, который расположен на другом сервере. К примеру

```
<script src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.7/angular.min.js"></script>
```

Структура кода

В языке JavaScript код состоит из команд. Сама по себе команда описывает некую операцию. Таким образом, набор команд, из которых состоит код, и задаёт поведение страницы.

```
var car = "Audi";
var power = 500;
if (power < 400) {
 alert("This page is for fast cars only!");
} else {
 alert("Welcome " + car + " owner!");
}
```

Команды в данном коде, к примеру, объявляют переменные, в которых будут храниться некие, используемые в дальнейшем, значения. Эти значения в дальнейшем влияют на принятие решений.

В переменных могут храниться числа, строки, логические значений и другие данные. Мы познакомимся с ними позже. Вне зависимости от типа данных все переменные создаются по одному алгоритму.

Переменная объявляется, начиная с ключевого слова var. Затем указывается имя переменной. При необходимости можно указать начальное значение переменной через знак равенства, но это необязательно, т.к. значение можно присвоить позднее. Переменную можно объявить и без ключевого слова var, но в таком случае она сразу же становится глобальной (о глобальности и минусах такого решения мы поговорим чуть позже).

```
var power = 500;
var speed;
```

Имя переменной должно начинаться с буквы, подчеркивания или знака доллара. Потом может следовать любое количество букв, цифр, подчеркиваний и знаков доллара. В языке JavaScript есть набор ключевых слов, использовать которые в качестве имён переменных не следует. Ниже их полный список

- break;
- case;
- catch;
- class;
- const;
- continue;
- debugger;
- default;
- delete;
- do;
- else;
- enum;
- export;
- extends;
- false;
- finally;
- for;
- function;
- if;
- implements;
- import;
- in;
- instanceof;
- interface;
- let;
- new;
- package;
- private;
- protected;
- public;
- return;
- static;
- super;
- switch;
- this;
- throw;
- true;

- try;
- typeof;
- var;
- void;
- while;
- with;
- yield.

В JavaScript регистр символов играет роль практически везде. Поэтому будьте внимательны, давая имена своим переменным.

В момент написания кода имена переменных _qwe, $ или bar что-то означают лично для Вас. Но помните, что Ваш код может читать кто-то другой. И он этого не поймёт. Поэтому имена currentSpeed, isValid сразу же показывают, что за значение хранится в них.

Предположим, что в процессе написания кода Вам требуется подобрать имя для переменной, в которой хранится массив с чётными строками текста красного цвета. В данном случае удобно использовать т.н. стиль lowerCamelCase - начинайте с прописной буквы каждое слово, кроме первого: evenRedStringsText.

Переменные, имена которых начинаются с $, резервируются для библиотек JavaScript и хранения DOM-узлов (о них мы поговорим позже). Некоторые авторы всё же применяют такие имена, но это не рекомендуется.

Для выполнения неких действий в JavaScript применяются выражения. Они вычисляются, после чего становится доступен их результат в виде некоего значения. Например

```
var distance = time * speed;
var fullname = "Mr " + "Rick " + "Sanches";
power < 400; // логическое выражение
```

Как видите, в предыдущем примере использован так называемый комментарий. Комментарии могут находиться в любом месте программы и никак не влияют на её выполнение. Интерпретатор игнорирует их. Они используются для того, чтобы оставить какую-то важную информацию об участке кода или временно деактивировать участок кода, не удаляя его.

- Однострочные комментарии начинаются с двойного слэша //.
- Многострочные комментарии начинаются слешем-звездочкой «/*» и заканчиваются звездочкой-слэшем «*/»

Помимо переменных в JavaScript можно определять символьные имена для конкретных значений. К примеру, если у Вас есть цвет #FF7F00, каждый раз писать его значение неудобно. Но можно положить его в константу с удобным именем, что ускорит общение с ним.

Как правило, их называют большими буквами, через подчёркивание.

```
var COLOR_RED = "#F00";
var COLOR_GREEN = "#0F0";
var COLOR_BLUE = "#00F";
var COLOR_ORANGE = "#FF7F00";
var color = COLOR_ORANGE;
alert( color ); // #FF7F00
```

Внимательные заметят, что чисто технически – это та же обычная переменная. Но особое написание ссылается на договорённость не менять значение в процессе написания кода.

Типы данных

Типы данных в JavaScript делятся на две группы: простые типы и объекты. Простые типы - это числа, текстовые строки и логические (или булевы) значения. Специальные значения null и undefined - это простые значения, но они не являются ни числами, ни строками, ни логическими значениями. Каждое из них определяет только одно значение своего собственного специального типа. Значение, которое не является числом, строкой, логическим значением или специальным значением null или undefined, является объектом. Объект - это коллекция свойств, каждое из которых имеет имя и значение.

Число «number»

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

Нужно учитывать, что тип число используется как для целых, так и для дробных чисел. Тип имеет специальные числовые значения Infinity (бесконечность) и NaN (ошибка вычислений, not a number).

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
```

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, но выберите для себя один тип, будьте последовательны.

Булевый (логический) тип «boolean»

У данного типа есть только два значения: true (истина) и false (ложь).

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

Специальное значение «null»

Как мы проговорили ранее, null не относится к вышеперечисленным типам. Он указывает на несуществующий объект. Т.е. это просто специальное значение для понятия «ничто».

```
var age = null;
```

Специальное значение «undefined»

Данный особый тип означает, что значение переменной не присвоено. Оно фигурирует тогда, когда переменная объявлена, но значение ей ещё не успели присвоить.

```
var x;
alert( x ); // выведет undefined
```

Объекты «object»

Вышеозначенные типы называют «примитивными». Особый тип: «объекты». Применяется он для хранения коллекций данных и объявления сложных сущностей. Подробно объекты изучаются в курсе JS Level 2.

В любом месте кода Вы можете узнать, какой тип имеет та или иная переменная. Для этого применяется оператор typeof. Возвращает он тип переданного аргумента в виде строкового значения.

```
typeof undefined   // undefined
typeof 0           // number
typeof true        // boolean
typeof "foo"       // string
typeof {}          // object
typeof null        // object
typeof function(){} // function
```

Обратите внимание на последние две строки, вернее, на их поведение.

1. Результат typeof null == "object"– это официально признанная ошибка в языке, которая сохраняется для совместимости. На самом деле null – это не объект, а отдельный тип данных.

2. Функции мы будем обсуждать позже. Но сейчас отметим, что они не являются отдельным базовым типом в JavaScript, а представляют собой подвид объектов. При этом typeof выделяет функции отдельно, возвращая для них "function". Это весьма удобно, так как позволяет легко определить функцию.

Стандарт языка

Довольно долго язык JavaScript развивался, сохраняя обратную совместимость. Новые возможности добавлялись, но старые – не менялись, чтобы не «сломать» уже существующие HTML/JS-страницы с их использованием. Разумеется, это привело к тому, что любая ошибка в дизайне языка становилась

зафиксированной в нем навсегда.

Так продолжалось до появления стандарта ECMAScript 5 (ES5), который добавил новые возможности, а также привнёс в язык множество исправлений. Эти исправления привели к тому, что старый код перестал работать.

Дабы нивелировать эту проблему разработчики языка решили, что по умолчанию опасные изменения будут выключены, а интерпретатор языка будет работать по-прежнему. Для того, чтобы перевести код в режим полного соответствия современному стандарту, требуется указать специальную директиву use strict.

Заметьте, что эта директива не поддерживается браузерами младше IE9.

Сама директива - это строка "use strict", которая ставится в начале скрипта:

```
"use strict";
```

При использовании этой директивы, к примеру, будет невозможно создание переменных без ключевого слова var, что разрешено в более старом стандарте.

Домашнее задание

1. Задать температуру в градусах по Цельсию. Вывести в alert соответствующую температуру в градусах по Фаренгейту. Подсказка: расчёт идёт по формуле: Tf = (9 / 5) * Tc + 32, где Tf – температура по Фаренгейту, Tc – температура по Цельсию
2. Объявить две переменные: admin и name. Записать в name строку "Василий"; Скопировать значение из name в admin. Вывести admin (должно вывести «Василий»).
3. * Чему будет равно JS-выражение 1000 + "108";
4. * Самостоятельно разобраться с атрибутами тега script (async и defer)

Дополнительные материалы

1. http://articles-hosting.ru/329/javascript-%E2%80%93-chto-eto-takoe.html - что такое JavaScript
2. https://habrahabr.ru/post/37619/ - обсуждение об изучении JavaScript

Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. «JavaScript. Подробное руководство» - Дэвид Флэнаган
2. «Изучаем программирование на JavaScript» - Эрик Фримен, Элизабет Робсон