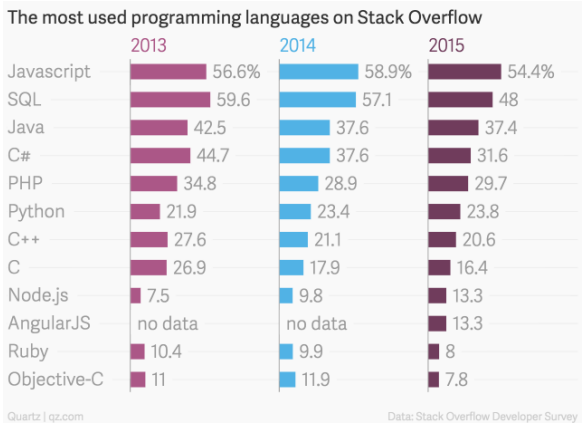
Javascript

Андрей Редькин, СФУ
andrew.redkin@gmail.com

Язык Javascript

- Современный JavaScript – это «безопасный» язык программирования общего назначения. Он не предоставляет низкоуровневых средств работы с памятью, процессором, так как изначально был ориентирован на браузеры, в которых это не требуется.

Популярность Javascript



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…
- …и многое, многое другое!

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.
- JavaScript, работающий в одной вкладке, не может общаться с другими вкладками и окнами
- Из JavaScript можно легко посылать запросы на сервер, с которого пришла страница. Запрос на другой домен тоже возможен, но менее удобен, т. к. и здесь есть ограничения безопасности.

Тенденции развития

- HTML 5 – эволюция стандарта HTML, добавляющая новые теги и, что более важно, ряд новых возможностей браузерам
- EcmaScript 6 будет шагом вперёд в улучшении синтаксиса языка

Альтернативные браузерные технологии

- Java – язык общего назначения, на нём можно писать самые разные программы. Для интернет-страниц есть особая возможность – написание апплетов
- Adobe Flash – кросс-браузерная платформа для мультимедиа-приложений, анимаций, аудио и видео. Flash-ролик – это скомпилированная программа, написанная на языке ActionScript.

Hello World!

- Программы на языке JavaScript можно вставить в любое место HTML при помощи тега SCRIPT

```
<!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>
```

- Асинхронная загрузка

```
<script src="1.js" async></script>
<script src="2.js" async></script>
```

- Отложенная загрузка

```
<script src="1.js" defer></script>
<script src="2.js" defer></script>
```

Переменная

- Для объявления переменной используется ключевое слово **var**

```
var message;
message = 'Hello'; // сохраним в переменной строку
```

Директива `"use strict"`и ставится в начале скрипта.

```
"use strict";

// этот код будет работать по современному стандарту ES5
...
```

Шесть типов данных, typeof

- Число «number»
- Строка «string»
- Булевый (логический) тип «boolean»
- Специальное значение «null»
- Специальное значение «undefined»
- Объекты «object»

```
typeof undefined // "undefined"

typeof 0 // "number"

typeof true // "boolean"

typeof "foo" // "string"

typeof {} // "object"

typeof null // "object"  (1)

typeof function(){} // "function"
```

Преобразование типов для примитивов

- В JavaScript есть три преобразования:
- 1 Строковое: String(value) – в строковом контексте или при сложении со строкой. Работает очевидным образом.
- 2 Численное: Number(value) – в численном контексте, включая унарный плюс +value. Происходит при сравнении разных типов, кроме строгого равенства.
- 3 Логическое: Boolean(value) – в логическом контексте, можно также сделать двойным НЕ: !!value.

Функции

```
function showMessage() {
  alert( 'Привет всем присутствующим!' );
}
```

```
function showMessage() {
  var message = 'Привет, я - Вася!'; // локальная переменная

  alert( message );
}

showMessage(); // 'Привет, я - Вася!'

alert( message ); // <-- будет ошибка, т.к. переменная видна только внутри
```

```
var userName = 'Вася';

function showMessage() {
  var message = 'Привет, я ' + userName;
  alert(message);
}

showMessage(); // Привет, я Вася
```

Параметры

- При вызове функции ей можно передать данные, которые та использует по своему усмотрению.

```
function showMessage(from, text) { // параметры from, text

  from = "** " + from + " **"; // здесь может быть сложный код оформления

  alert(from + ': ' + text);
}

showMessage('Маша', 'Привет!');
showMessage('Маша', 'Как дела?');
```

- Возврат значения

```
function calcD(a, b, c) {
   return b*b - 4*a*c;
}

var test = calcD(-4, 2, 1);
alert(test); // 20
```

Функциональные выражения

```
function sayHi() {   // (1)
  alert( "Привет" );
}

var func = sayHi;    // (2)
func(); // Привет    // (3)

sayHi = null;
sayHi();             // ошибка (4)
```

```
var sum = new Function('a,b', ' return a+b; ');

var result = sum(1, 2);
alert( result ); // 3
```

```
function ask(question, yes, no) {
  if (confirm(question)) yes()
  else no();
}

ask(
  "Вы согласны?",
  function() { alert("Вы согласились."); },
  function() { alert("Вы отменили выполнение."); }
);
```

Строки

```
var text = "моя строка";

var anotherText = 'еще строка';

var str = "012345";
```

Специальные символы

Символ	Описание
\b	Backspace
\f	Form feed
\n	New line
\r	Carriage return
\t	Tab

```
var str = "My\n"; // 3 символа. Третий - перевод строки

alert( str.length ); // 3
```

```
var str = "Я - современный браузер!";
alert( str[0] ); // "Я"
```

```
var str = "Widget with id";

alert( str.indexOf("Widget") ); // 0, т.к. "Widget" найден прямо в начале str
alert( str.indexOf("id") ); // 1, т.к. "id" найден, начиная с позиции 1
alert( str.indexOf("widget") ); // -1, не найдено, так как поиск учитывает регистр
```

Объекты

- Объекты в JavaScript сочетают в себе два важных функционала.
- Первый – это ассоциативный массив: структура, пригодная для хранения любых данных.
- Второй – языковые возможности для объектно-ориентированного программирования.

Создание объектов

```
1. o = new Object();
2. o = {}; // пустые фигурные скобки
```

```
// при присвоении свойства в объекте автоматически создаётся "ящик"
// с именем "name" и в него записывается содержимое 'Вася'
person.name = 'Вася';

person.age = 25; // запишем ещё одно свойство: с именем 'age' и значением 25
```

```
var menuSetup = {
  width: 300,
  height: 200,
  title: "Menu"
};

// то же самое, что:

var menuSetup = {};
menuSetup.width = 300;
menuSetup.height = 200;
menuSetup.title = 'Menu';
```

Объекты как ассоциативные массивы

- Ассоциативный массив – структура данных, в которой можно хранить любые данные в формате ключ-значение.

```
var person = {};

person['name'] = 'Вася'; // то же что и person.name = 'Вася'
```

- перебор свойств

```
for (key in obj) {
  /* ... делать что-то с obj[key] ... */
}
```

Массивы с числовыми индексами

- Массив – разновидность объекта, которая предназначена для хранения пронумерованных значений и предлагает дополнительные методы для удобного манипулирования такой коллекцией.

```
var fruits = ["Яблоко", "Апельсин", "Слива"];
```

```
// микс значений
var arr = [ 1, 'Имя', { name: 'Петя' }, true ];

// получить объект из массива и тут же -- его свойство
alert( arr[2].name ); // Петя
```

Массивы:split и join

```
var names = 'Маша, Петя, Марина, Василий';

var arr = names.split(', ');

for (var i = 0; i < arr.length; i++) {
  alert( 'Вам сообщение ' + arr[i] );
}
```

```
var arr = ['Маша', 'Петя', 'Марина', 'Василий'];

var str = arr.join(';');

alert( str ); // Маша;Петя;Марина;Василий
```

Массивы: sort

```
function compareNumeric(a, b) {
  if (a > b) return 1;
  if (a < b) return -1;
}

var arr = [ 1, 2, 15 ];

arr.sort(compareNumeric);

alert(arr);  // 1, 2, 15
```

Массив: перебирающие методы

- **forEach(callback)**
  для каждого элемента массива вызывает функцию callback. Этой функции он передаёт три параметра callback(item, i, arr):
  - item – очередной элемент массива.
  - i – его номер.
  - arr – массив, который перебирается.

```
var arr = ["Яблоко", "Апельсин", "Груша"];

arr.forEach(function(item, i, arr) {
  alert( i + ": " + item + " (массив:" + arr + ")" );
});
```

- **filter(callback)**
  создаёт новый массив, в который войдут только те элементы arr, для которых вызов callback(item, i, arr) возвратит true

```
var arr = [1, -1, 2, -2, 3];

var positiveArr = arr.filter(function(number) {
  return number > 0;
});

alert( positiveArr ); // 1,2,3
```