Javascript

Данильченко Анна Александровна

Преподаватель кафедры программного обеспечения систем ЖГТУ

Особенности JavaScript

- Поддерживает концепцию ООП
- Клиентский язык
- Исходный код скриптов открыт
- Результат выполнения зависит от браузера

Краткое введение в *Javascript*

Javascript это:

1. Интерпретируемый язык. Его интерпретатор обычно встроен в браузер.
2. Основное назначение – определять «динамическое» поведение страниц при загрузке (формирование страницы перед ее открытием) и при работе пользователя со страницей (UI элементы).
3. Текст на *Javascript* может быть вложен в HTML-страницу непосредственно или находиться в отдельном файле (как CSS).
4. Похож на языки *Java* и *C#* синтаксически, но сильно отличается от них по внутреннему содержанию.

```
<!DOCTYPE html>
<html>
<head>
    <title>Hello World</title>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
</head>
<body>
    <script type="text/javascript">
        document.write("Hello World");
        document.write("<br />Hello World");
    </script>
</body>
</html>
```

Некоторые сведения о синтаксисе

Описание переменных:

```
var count = 25,
  msg = 'Сообщение об ошибке';
var nullVar;  // получает начальное
значение null
```

Операции такие же, как в Java и C#, но более широко используется преобразование типов

```
+ - * / % ++ -- = += -= *=
/= %= == != > < >= <= && || !

2 + '3' == '23', но   2 + 3 == 5
```

Многие операторы очень похожи на соответствующие операторы Java и C#, но могут иметь некоторые отличия в семантике.

```
for (var i = 0; i < 100; ++i) { ... }
if (x * y < 100) { ... } else { ... }
try { ... } catch (e) { ... } finally { ... }
```

Объекты, встроенные в браузеры

При программировании можно использовать ряд встроенных объектов. Основные из них это:

- **window** : представляет «глобальный контекст» и позволяет работать
- **document**: загруженная страница со своей структурой элементов
- **navigator** : объект, представляющий браузер и его свойства.
- **location** : характеристики текущего URL (порт, хост и т.п.).
- объекты, представляющие элементы различных типов в HTML-странице, такие как **<body>**, **<link>**, **<img>** и т.п.
- события (events), возникающие от действий пользователя, например, нажатие кнопки мыши (**click**), загрузка новой страницы (**load**) и т.д.

Включение Javascript в HTML-страницу

Фрагменты кода можно включать в заголовок или тело HTML-документа. Кроме того, можно разместить код в отдельном файле, а в HTML-странице разместить ссылку на этот файл.

```
<html>
 <head>
  <script type="text/javascript"> … </script>
  <script type="text/javascript"
src="scripts/myscript1.js/>
 <head>
 <body>
  <script type="text/javascript"> … </script>
  <script type="text/javascript"
src="scripts/myscript2.js/>
 </body>
</html>
```

Код, ссылки на который размещены в заголовке, просто подсоединяется к странице и может быть использован, например, для определения реакций на пользовательские события.

Код, ссылки на который размещены в теле, исполняется при загрузке страницы и может непосредственно использоваться для формирования содержания страницы во время загрузки.

Размещение тега script

Интуиция подсказывает, что нужно размещать в теге `head`

```
<head>
    <link rel="stylesheet" href="styles.css">
    <script src="script.js"></script>
</head>
<body>
    <!-- Rendering content -->
</body>
```

Размещение тега script

Хорошая практика - подключать `script` в конце тега `body`

```
<head>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <!-- Rendering content -->
    <script src="script.js"></script>
</body>
```

Переменные

Переменная - именованная ячейка памяти.

Объявляется с помощью ключевого слова `var` и идентификатора.

```
var myVar;
```

Присваивание

```
var myVar; // объявление

myVar = 'my var'; // присваивание
```

Инициализация лучше

Плохо

```
var myVar; // объявление
myVar = 'my var'; // присваивание
```

Хорошо

```
var myVar = 'my var'; // инициализация
```

Единственная инструкция var - лучше

Плохо

```
var name = 'Артем';
var age = 25;
```

Хорошо

```
var name = 'Артем',
    age = 25;
```

Типы данных

5 примитивных типов

- Number
- String
- Boolean
- undefined
- null

undefined

- Значение переменных по умолчанию

```
var myVar; // undefined
```

null

Значение переменных которые ничего не содержат.

Плохо

```
var activeItem = {};
activeItem = undefined;
```

Хорошо

```
var activeItem = {};
activeItem = null;
```

Тип String

Строки заключаются либо в апострофы, либо в двойные кавычки

```
var slogan = "Don't be evil!";
var image ='<img class="to-right"
src="myphoto.jpg"/>';
```

escape-последовательности: \\ \/ \' \" \t \n

Операции над строками: + < > == !=

```
"2" +
"10" <
"3" <
10 < "5"
5 + "5"
```

```
"2
8"
10
false
55
"
```

```
"a" ==
"A" "5"
5 == "5"
```

```
fal
se
tru
e
fal
se
```

Атрибут строки: length – длина строки

```
"abc".length
== 3
```

Преобразования типов: String(n) Number(s)

```
String(10) < "5" ==
true
```

```
Number('3.' + '14') ==
3.14
```

Стандартные методы объектов типа String

charAt, indexOf, lastIndexOf,
replace, split,
substr, substring, toLowerCase,
toUpperCase

Примеры:

```
"Google".charAt(2)                  "g"
"Google".indexOf("o")               "i"
"Google".lastIndexOf("o")           2
"Google".replace("o","oo")          "Goooogle"
"Google".replace(/o/g,"oo")         "Google" (глобальный поиск)
"Google".split("o")                 ["G","","gle"]
"Google".substr(1,2)                "oo"
"Google".substring(1,2)             "o"
"Google".toLowerCase()              "google"
"Google".toUpperCase()              "GOOGLE"
```

Строки неизменяемы

```
var myString = "my string",
    subStr = myString.substring(3, 6);


myString; // "my string"
subStr; // "str"
```

Тип Number

Числа – это 64-х-разрядные двоичные числа с плавающей точкой.

```
Number.MIN_VALUE                5e-324
Number.MAX_VALUE                1.7976931348623157e+308
Number.NaN                      NaN
Number.POSITIVE_INFINITY        Infinity
Number.NEGATIVE_INFINITY        -Infinity
```

Операции над числами: `+ - * / % < > == !=`

```
1.1
```

Функции преобразования: `parseInt, parseFloat, Number, toString`

```
parseInt("3.14")                3
parseFloat("3.14")              3.14
Number("3.14")                  3.14
3.14.toString()                 "3.14"
isNaN(3.14/0)                   false
isNaN(0/0)                      true
```

Тип Boolean

Стандартные логические значения – **true** и **false**. Однако в качестве
условий можно использовать любое значение.

"Истинные" условия:

```
if (2 < 5)
if
('Google могуч и
ужасен')
```

"Ложные" условия:

```
if ("")
if (0)
if
(null)
```

Логические условия используются в условных операторах и операторах циклов.

```
if (x < y) { z = x; } else { z = y; }
while (x < 100) { x = x * 2; n++; }
do { x = Math.floor(x / 2); n++; } while (x >
for (var y = 0, x = 0; x < 100; ++x) { y += x; }
```

Оператор typeof

Определяет тип переменной

```
typeof 1; // "number"
typeof "some string"; // "string"
typeof true; // "boolean"
typeof undefined; // "undefined"
typeof function(){}; // "function"
```

Для всего остального `typeof` возвращает `"object"`, в том числе и для `null`.

Переменные

Объявить три переменные x=5; y=7;
abc=7.5
Вывести на экран в формате:
x=5;
y=7;
 abc=7.5

Типы переменных

Объявить 5 переменных разных типов данных.
i=-5,
d=7.3,
str=“Hello”,
b_1=false,
b_2=true.
Вывести результат на экран

Операции с переменными

Объявить две переменные x=5; y=7.5.

Вывести на экран результаты математических операций:

+ - * / %

1.Добавить к переменной число 2.

2.Увеличить/уменьшить на 1.

Операции с переменными (строками)

Создать две строковые переменные
str_1=23,
str_2=12

1. Сложить строки
2. Перевести в числа и сложить

Операции с переменными (логические)

1. Объявить переменную b=x<y где x=5,y=10
2. Вывести значение переменной на экран
3. Вывести отрицание переменной
4. Вывести такую таблицу

B1	B2	Отрицание B1	B1 И B2	B1 ИЛИ B2	B1 ИСКЛЮЧАЮЩЕЕ ИЛИ B2
false	false	true	false	false	0
false	true	true	false	true	1
true	false	false	false	true	1
true	true	false	true	true	0

Условные операторы

```
if (утверждение) {
    //блок операторов
}
else {
    //блок операторов
}
```

1. Задание. Объявить переменную отвечающую за количество шин на складе.
Вывести: если шины
Есть на складе – «Шины есть»,
если переменная равна 0 то «Шин нет».
2. Если шин меньше 4 вывести «нет полного комплекта»

Составная инструкция (block statement)

Позволяет объеденять несколько инструкций в одну.

```
{
    statement;
    statement;
    statement;
}
```

Лучше всегда указывать фигурные скобки

```
if(выражение){
    инструкция1; // statement1
    инструкция2; // statement2
}else{
    инструкция3; // statement3
}
```

```
var kol=3;
if(kol>0)
{
    document.write("Шины есть<br>");
    if(kol<4){
        document.write("Нет полного комплекта");
    }
}
else
{
    document.write("Шины закончились");
}
```

```
Switch(переменная){
        case значение1:{
            //блок операторов
            break;}
        case значение2:
            //блок операторов
            break;
        default: //блок операторов
}
```

Вывести словами количество шин на складе. Если 1 то вывести «На складе 1 шина» и т.д. по умолчанию вывести шин много

Циклы!

Цикл for

Чаще всего применяется цикл for. Выглядит он так:

```
for (начало; условие; шаг) {
  // ... тело цикла ...
}
```

```
var kol=4;
for(i=0;i<kol;i++){
    document.write("Я число i="+i+"<br>");
}
```

Я число i=0
Я число i=1
Я число i=2
Я число i=3

Простой пример

Метод **document.write** используется для непосредственного включения
HTML-текста в содержимое страницы, например, можно сгенерировать длинный текст в параграфе:

```
<body>
 <p>
  <script
type="text/javascript">
   for (var i = 0; i < 100; ++i) {
    document.write("Hello,
world! ");
   }
  </script>
 </p>
</body>
```

Циклы

- С предусловием
  - `while`
  - `for`
- С постусловием
  - `do while`

Цикл while

Цикл `while` имеет вид:

```
while (условие) {
  // код, тело цикла
}
```

Пока `условие` верно — выполняется код из тела цикла.

```
var i=1;
var kol=4;
while(i<kol){
    document.write("Я число i="+i+"<br>");
    i++;
}
```

Я число i=1
Я число i=2
Я число i=3

Посчитать в цикле сумму чисел от 1 до 100. Вывести результат на экран.

Цикл do..while

Проверку условия можно поставить *под* телом цикла, используя специальный синтаксис `do..while`:

```
do {
  // тело цикла
} while (условие);
```

Цикл, описанный, таким образом, сначала выполняет тело, а затем проверяет условие.

Например:

```
var i = 0;
do {
  alert( i );
  i++;
} while (i < 3);
```

Синтаксис `do..while` редко используется, т.к. обычный `while` нагляднее — в нём не приходится искать глазами условие и ломать голову, почему оно проверяется именно в конце.

Разница между while и do...while

```
i = 1000;
do {
    document.write("<br />Мы в цикле");
} while (i < 100);

while (i < 100) {
    document.write("<br />Мы в цикле");
}
```

Continue и break

Вывести числа от 0 до 10 которые не делятся на 3.

Если делим число 8 на 3 то выходим из цикла.

Взаимодействие с пользователем: alert, prompt, confirm

alert

Синтаксис:

```
1 alert(сообщение)
```

`alert` выводит на экран окно с сообщением и приостанавливает выполнение скрипта, пока пользователь не нажмет «OK».

```
1 alert( "Привет" );
```

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 разберется с окном. В данном случае – пока не нажмет на «OK».

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 ввел посетитель — строку или специальное значение `null`, если ввод отменен.

```
var years = prompt('Сколько вам лет?', 100);

alert('Вам ' + years + ' лет!')
```

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:

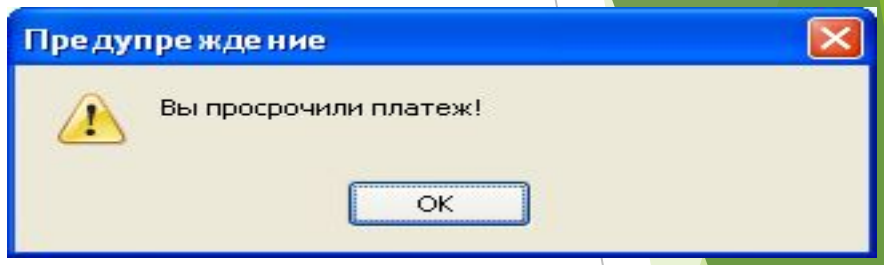
```
var isAdmin = confirm("Вы - администратор?");

alert( isAdmin );
```

Сообщения, выдаваемые в popup-окнах

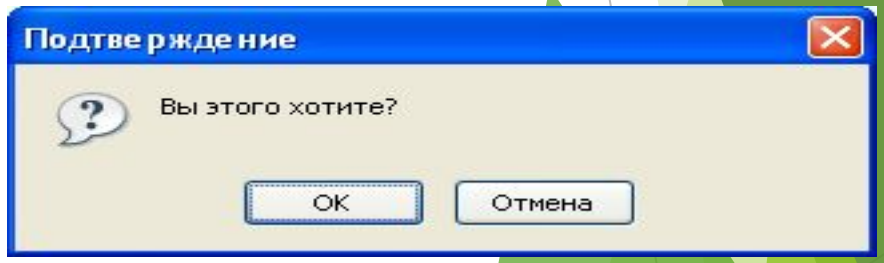
Три стандартные функции используются для генерации сообщений в popup-окнах: **alert**, **confirm**, **prompt**.

```
alert('Вы просрочили платеж?');
```



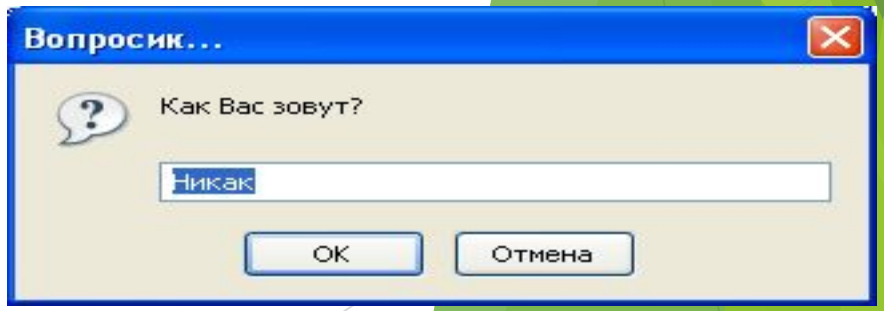

```
confirm('Вы этого хотите?');
```

Выдает **true** или **false**




```
var name = prompt('Как Вас зовут?',
  'Никак', 'Вопросик...');
```

Выдает **введенную строку** или **null**



Задание

1. Написать калькулятор простых операций с числами.
2. Ввод и вывод данных и операторов с помощью модальных окон

Массив в JavaScript

Это структура данных представленная в виде ячеек любого типа данных , объединенных под одним именем

Тип Array

Существует несколько способов создания массива:

```
var holidays = ["1 января", "7 января", "23 февраля"];
var holidays = new Array("1 января", "7 января", "23 февраля");
var holidays = new Array(3);
holidays[0] = "1 января";
holidays[1] = "7 января";
holidays[2] = "23 февраля";
var myArray = new Array();
myArray[2] = new Date(2008,2,23);
myArray[5] = new Date(2008,5,9);
myArray.length == 6
```

Атрибут length – длина массива

Максимальное количество ячеек

4 294 967 295

Многомерные массивы

В JavaScript нет понятия многомерного массива, однако в самих массивах может содержаться что угодно, в том числе ссылки на другие массивы.

```
var a0 = [10, 11, 12];
var a1 = [20, 21, 22];
var a2 = [30, 31, 32];
 var a = [a0, a1, a2];
 alert(a[0]); // 10,11,12
 alert(a[1]); // 20,21,22
alert(a[2]); // 30,31,32
```

Задание

1. Создать массив элементов (-7.5, 5, “str”, false). Вывести на экран элемент массива со значением 5.
2. Вывести на страницу в цикле все элементы массива.
3. Создать многомерный массив из трех массивов (в первом числа от 0 до 5, во втором от 0 до 10 в третьем от 0 до 15).
4. Вывести все элементы многомерного массива на экран

Тип Array (продолжение)

Методы, определенные для работы с массивом:

concat, join, pop, push, shift , unshift, slice

```
var names = ["Петя",
"Вася"];names.concat(["Сережа", "Наташа"],
["Оля", "Люба"]);
                names == ["Петя", "Вася", "Сережа", "Наташа",
                "Оля", "Люба"]
var s =
names.join(";");
                s == "Петя;Вася;Сережа;
var e =         Наташа;Оля;Люба"
names.pop();
                names == ["Петя", "Вася", "Сережа",
                "Наташа", "Оля"]
var l =
names.push("Саша")
;               names == ["Петя", "Вася", "Сережа", "Наташа",
                "Оля", "Саша"]
```

shift и **unshift** – точно также, как **pop** и **push**, но с началом массива

```
names =
names.slice(1, 4);["Вася", "Сережа",
                "Наташа", "Оля"]
```

Тип Array (продолжение)

Еще методы, определенные для работы с массивом:
reverse, sort, splice, toString

```
var names = ["Вася", "Сережка",
"Наташа", "Оля"];
names.reverse();   names == ["Оля", "Наташа",
                   "Сережка", "Вася"]
names.sort();      names == ["Вася", "Наташа", "Оля",
                   "Сережка"]
var a = [5, 3, 40, 1, 10,
100].sort();       a == [1, 10, 100, 3, 40, 5]

var a = [5, 3, 40, 1, 10,
100].sort(function(a,b){return a-b;});

names.splice(1, 2, "Саша",
"Таня", "Нина");   names == ["Вася", "Саша", "Таня",
                   "Нина", "Сережка"]
```

toString – точно так же, как **join(',')**.

```
                   names.toString() == "Вася,Саша,
                   Таня,Нина,Сережка"
```

Задание

1. Создать массив Array(15 7 8 10 -5 0 1)
2. Вывести длину массива;
3. Вывести массив в виде 15;7;8;10;-5;0;1
4. Отсортировать элементы массива по возрастанию, и все таки по возрастанию.
5. Отсортировать по убыванию.
6. Создайте два массива.1-й - это массив из пяти стран на ваш выбор.2-й - это массив количества населения в этих странах (этот массив состоит исключительно из цифр).А затем с помощью метода document.write() выведите на экран пять строк, где каждая строка - это название страны и численность ее населения.