

Web-технологии и web-дизайн

Основы Javascript



Беркунский Е.Ю., кафедра ИУСТ, НУК
eugeny.berkunsky@gmail.com
<http://www.berkut.mk.ua>



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.

Характеристика *Javascript*

Некоторые важнейшие характеристики *Javascript* :

1. Язык объектно-ориентированного программирования. Объекты в языке имеют «тип», «атрибуты» и «методы»

```
"John,Paul,George,Ringo".split(",").length
```

2. Переменные не имеют заранее заданного типа, то есть в разные моменты времени могут содержать значения разных типов

```
let number = 25; number = (number < 0); number = "25";
```

3. Типы объектов могут быть: `number`, `string`, `function`, `object`, `undefined`. Оператор `typeof` позволяет «вычислить» тип объекта.

```
typeof 25 == "number"    typeof null == "object"
```

Основные встроенные типы

Есть набор встроенных «классов», порождающих «объекты», различающиеся набором атрибутов и методов. Программисты могут динамически изменять поведение этих «классов» и создавать свои собственные. Каждый «класс» является объектом, у которого есть «прототип», определяющий набор атрибутов и методов у всех вновь создаваемых объектов этого класса.

Типы, встроенные в язык, это:

- `Number` : 64-х-разрядные числа с плавающей точкой.
- `String` : строки с символами в формате Unicode.
- `Array` : массивы с переменными границами.
- `Function` : Функции. Каждая функция, кроме того, может служить конструктором объекта.
- `Boolean`, `Date`, `Math`, `RegExp` : логические значения, даты,...

Описание переменных:

```
let count = 25,  
    msg = 'Сообщение об ошибке';  
let 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 (let i = 0; i < 100; ++i) { ... }  
if (x * y < 100) { ... } else { ... }  
try { ... } catch (e) { ... } finally { ... }
```

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

- `window` : представляет «глобальный контекст» и позволяет работать с атрибутами и методами окна.
- `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>`, `` и т.п.
- события (events), возникающие от действий пользователя, например, нажатие кнопки мыши (`click`), загрузка новой страницы (`load`) и т.д.



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> ... </script>  
    <script src="scripts/myscript1.js"/>  
  </head>  
  <body>  
    <script> ... </script>  
    <script 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.

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

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

[helloworld.html](#)

Два простых примера (продолжение)

Во втором примере датчик случайных чисел используется для генерации случайной ссылки (из заданного набора):

```
<body>
  <p>
    <script>
      let rand = Math.random();          // в диапазоне: [0, 1)
      let numb = Math.floor(rand * 10);
      let image = "images/image" + numb + ".jpg";
      let insert = "<img class=\"floatRight\" src=\"\" +
                  image + \"\" alt=\"фотография цветочка\"/>";
      document.write(insert);
    </script>
  </p>
</body>
```

[randomPicture.html](#)

Тип String

Строки заключаются либо в апострофы, либо в двойные кавычки

```
let slogan = "Don't be evil!";  
let image = '';
```

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

Операции над строками: `+` `<` `>` `==` `!=`

| | | | |
|----------------------------|--------------------|-------------------------|--------------------|
| <code>"2" + "3"</code> | <code>"23"</code> | <code>"a" == "A"</code> | <code>false</code> |
| <code>"10" < "5"</code> | <code>true</code> | <code>5 == "5"</code> | <code>true</code> |
| <code>10 < "5"</code> | <code>false</code> | <code>5 === "5"</code> | <code>false</code> |
| <code>5 + "5"</code> | <code>"55"</code> | | |

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(3) | "g" |
| "Google".indexOf("o") | 1 |
| "Google".lastIndexOf("o") | 2 |
| "Google".replace("o", "oo") | "Gooogle" |
| "Google".replace(/o/g, "oo") | "Goooogle" |
| "Google".split("o") | ["G", "", "gle"] |
| "Google".substr(1, 3) | "oog" |
| "Google".substring(1, 3) | "oo" |
| "Google".toLowerCase() | "google" |
| "Google".toUpperCase() | "GOOGLE" |

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

| | |
|---------------------------------------|--------------------------------------|
| <code>Number.MIN_VALUE</code> | <code>5e-324</code> |
| <code>Number.MAX_VALUE</code> | <code>1.7976931348623157e+308</code> |
| <code>Number.NaN</code> | <code>NaN</code> |
| <code>Number.POSITIVE_INFINITY</code> | <code>Infinity</code> |
| <code>Number.NEGATIVE_INFINITY</code> | <code>-Infinity</code> |

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

| | |
|-----------------------|-------------------|
| <code>3.14 % 2</code> | <code>1.14</code> |
|-----------------------|-------------------|

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

| | |
|----------------------------------|---------------------|
| <code>parseInt("3.14")</code> | <code>3</code> |
| <code>parseFloat("*3.14")</code> | <code>NaN</code> |
| <code>Number("3.xaxa")</code> | <code>NaN</code> |
| <code>3.14.toString()</code> | <code>"3.14"</code> |
| <code>isNaN(3.14 / 0)</code> | <code>false</code> |
| <code>isNaN(0 / 0)</code> | <code>true</code> |

Тип Boolean

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

"Истинные" условия:

```
if (2 < 5)
if (25)
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 > 0);
for (let y = 0, x = 0; x < 100; ++x) { y += x; }
```

Объекты типа Date содержат дату в виде числа миллисекунд, прошедших с 1 января 1970 г. Диапазон от -10^8 до 10^8 дней от 1 января 1970 г.

Конструкторы:

```
let now = new Date();           // сейчас
let january1st1970 = new Date(0); // дата в миллисекундах
let gagarin = new Date(1961, 3, 12);
let newYear = new Date("January 1, 2009");
```

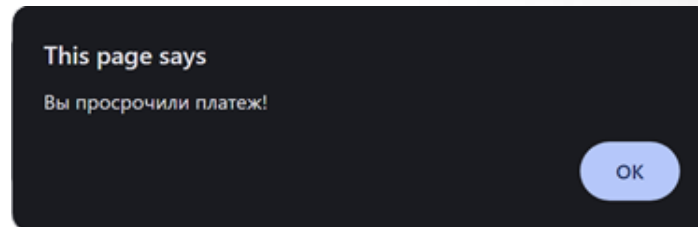
Методы, применимые для работы с датами: getDate, getMonth, getFullYear, getTime, getTimezoneOffset, setDate, setFullYear,...

```
function DaysToDate(day, month) {
    let now = new Date(), year = now.getFullYear();
    let bd = new Date(year, month-1, day);
    let fullDay = 24 * 60 * 60 * 1000;
    let diff = Math.ceil((bd - now) / fullDay);
    return diff < 0 ? diff + 365 : diff;
}
```

Сообщения, выдаваемые в рорир-окнах

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

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

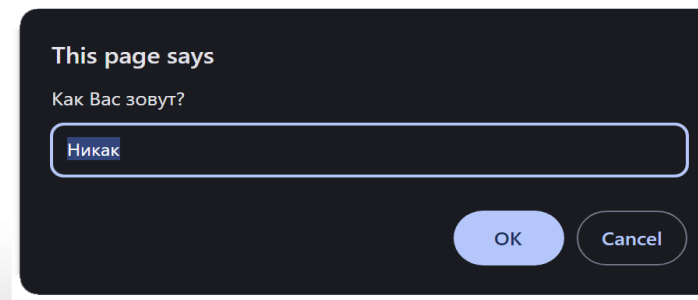


```
confirm('Вы этого хотите?');
```



Выдает **true** или **false**

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



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

События и реакции на них

Имеется большое количество событий, которые можно разделить на следующие классы:

- события от мыши (click, dblclick, mousedown,...);
- события от клавиатуры (keypress, keydown,...);
- события от элементов ввода (focus, submit, select,...);
- события страницы (load, unload, error,...);

Один из способов программирования состоит в определении реакции на события непосредственно в описании элемента, например:

```
<p>День независимости Украины  
  <span style="color: blue; text-decoration: underline;"  
    onclick=  
      "alert('Осталось ' + DaysToDate(24, 8) + 'дней');">  
    24 августа</span>.  
</p>
```

Недостаток этого способа: javascript-текст опять смешивается с содержанием страницы.

[holidays.html](#)

Существует несколько способов создания массива:

```
let holidays = ["1 января", "7 января", "24 августа"];  
let holidays = new Array("1 января", "7 января", "24 августа");  
let holidays = new Array(3);  
holidays[0] = "1 января";  
holidays[1] = "7 января";  
holidays[2] = "24 августа";
```

Атрибут массива: `length` – длина массива.

```
let myArray = new Array();  
myArray[2] = new Date(2024, 1, 23);  
myArray[5] = new Date(2024, 5, 9);  
myArray.length == 6
```

Тип Array (продолжение)

Методы, определенные для работы с массивом:

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

```
let names = ["Петя", "Вася"];
```

```
names = names.concat(["Сергея", "Наташа"], ["Оля", "Люба"]);
```

```
names == ["Петя", "Вася", "Сергея", "Наташа", "Оля", "Люба"]
```

```
let s = names.join(';');
```

```
s == "Петя;Вася;Сергея;Наташа;Оля;Люба"
```

```
let e = names.pop();
```

```
e == "Люба"
```

```
names == ["Петя", "Вася", "Сергея", "Наташа", "Оля"]
```

```
let l = names.push("Саша");
```

```
l == 6
```

```
names == ["Петя", "Вася", "Сергея", "Наташа", "Оля", "Саша"]
```

`shift` и `unshift` – точно так же, как `pop` и `push`, но с началом массива.

```
names = names.slice(1, 4);
```

```
names == ["Вася", "Сергея", "Наташа", "Оля"]
```

Тип Array (продолжение)

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

```
let names = ["Вася", "Сереза", "Наташа", "Оля"];
```

```
names.reverse();
```

```
names == ["Оля", "Наташа", "Сереза", "Вася"]
```

```
names.sort();
```

```
names == ["Вася", "Наташа", "Оля", "Сереза"]
```

```
let a = [5, 3, 40, 1, 10, 100].sort();
```

```
a == [1, 10, 100, 3, 40, 5]
```

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

```
a == [1, 3, 5, 10, 40, 100]
```

```
names.splice(1, 2, "Саша", "Таня", "Нина");
```

```
names == ["Вася", "Саша", "Таня", "Нина", "Сереза"]
```

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

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

Работа с таймером

Можно создать таймер и определить реакцию на событие от таймера.

```
let timer = setTimeout(func, timeinterval);
```

`func` – это функция или строка с кодом; `timeinterval` – время в миллисекундах. Таймер срабатывает один раз и запускает функцию.

```
function launchTimer() {  
    setTimeout("alert('Шахтер – чемпион!');", 2024);  
}
```

Теперь можно запустить этот таймер, например, по событию `click`:

```
<body>  
    <p>Нажми <span onclick="launchTimer();">сюда!</span></p>  
</body>
```

Пока событие еще не случилось, таймер можно остановить:

```
let timer = setTimeout(func, timeinterval);  
clearTimeout(timer);
```

[settimer.html](#)

Работа с интервальным таймером

Таймер может срабатывать многократно через равные промежутки времени. Такой таймер создается с помощью функции `setInterval` и останавливается с помощью функции `clearInterval`.

```
let timer = setInterval(func, timeinterval);
```

```
function launchInterval() {  
    timer = setInterval("alert('Шахтер – чемпион!');", 2024);  
}
```

```
function stopTimer() {  
    if (timer) clearInterval(timer);  
    timer = null;  
}
```

```
<body>  
    <p>Нажми <span onclick="launchInterval();">сюда,</span>  
        чтобы запустить.</p>  
    <p>Нажми <span onclick="stopTimer();">сюда,</span>  
        чтобы остановить.</p>  
</body>
```

[setinterval.html](#)

Вопросы?



Web-технологии и web-дизайн

Основы Javascript



Беркунский Е.Ю., кафедра ИУСТ, НУК
eugeny.berkunsky@gmail.com
<http://www.berkut.mk.ua>

