

# Типы данных

Вукмирович Александр

# Содержание

- Логический (Boolean)
- Числа (Numbers)
- Строки (Strings)
- Массивы (Arrays)
- Объекты (Objects)
- Функции (Functions)

# Примитивы

- Логический (Boolean)
- Числа (Numbers)
- Строки (Strings)
- undefined и null

# Объекты (в широком смысле)

- Массивы (Arrays)
- Функции (Functions)
- Объекты (Objects)

# Логический (Boolean)

```
// Логическое "Да"
```

```
const trueVar = true;
```

```
// Логическое "Нет"
```

```
const falseVar = false;
```

# Логический (Boolean)

```
// Неявное сравнение
```

```
'1' == 1; // true
```

```
// Явное сравнение
```

```
1 === 1; // true
```

```
'1' !== 1; // true
```

# Логические операции

// Логическое И

`true && false;` // false

// Логическое ИЛИ

`true || false;` // true

// Логическое НЕ (отрицание)

`!false;` // true

# Числа (Numbers)

```
const intPositive = 1;  
const intNegative = -1;  
const floatPositive = 45.354435;  
const negativePositive = -0.374599;  
const exp = 5e2;  
const hex = 0xFF; // 255
```



# Базовая математика

$5 + 2 = 7;$     // сложение

$5 - 2 = 3;$     // вычитание

$5 * 2 = 10;$     // умножение

$5 / 2 = 2.5;$     // деление

$5 \% 2 = 1;$     // остаток от целочисленного деления

# Математические функции (Math)

```
Math.abs(-5); // 5 (абсолютное значение)
```

```
Math.floor(7.43); // 7 (округление к младшему)
```

```
Math.ceil(7.43); // 8 (округление к старшему)
```

```
Math.round(7.43); // 7 (округление к ближайшему)
```

```
Math.floor(7.53); // 7 (округление к младшему)
```

```
Math.ceil(7.53); // 8 (округление к старшему)
```

```
Math.round(7.53); // 8 (округление к ближайшему)
```

```
Math.sqrt(16); // 4 (извлечение квадратного корня)
```

```
Math.pow(2, 8); // 256 (возведение в степень), или 2**8
```

# Бесконечность (Infinity)

```
1 / 0 === Infinity; // true
```

```
1 / Infinity === 0; // true
```

# Not a Number (NaN)

```
Math.sqrt(-1);           // NaN
```

```
NaN !== NaN;             // true
```

```
isNaN(Math.sqrt(-1));    // true
```

```
Number.isNaN(Math.sqrt(-1)); // true, не поддерживается в IE
```

# Стандарт чисел

64-bit двойной точности

Соответствуют стандарту IEEE 754 (IEEE Standard for Floating-Point Arithmetic)

`Number.MAX_VALUE; // 1.79e+308`

`Number.MIN_VALUE; // 5e-324, самое близкое к нулю`

# Стандарт чисел

```
Number.MAX_SAFE_INTEGER; // 2^53 - 1 или 9007199254740991
Number.MAX_SAFE_INTEGER + 1 === Number.MAX_SAFE_INTEGER + 2; // true

Number.MIN_SAFE_INTEGER; // -(2^53 - 1) или -9007199254740991
Number.MIN_SAFE_INTEGER - 1 === Number.MIN_SAFE_INTEGER - 2; // true

Number.isSafeInteger(Math.pow(2, 54)); // false
Number.isSafeInteger(-Math.pow(2, 54)); // false
// все это не поддерживается в IE и Safari
```

# Особенности вычислений

```
0.1 + 0.2 = 0.30000000000000004;
```

```
0.1 + 0.2 !== 0.3; // true
```

```
Number.EPSILON; // 2-52, не поддерживается в IE
```

```
Math.abs((0.1 + 0.2) - 0.3) < Number.EPSILON; // true
```

Here is what you need to know about JavaScript's Number type

# Строки (Strings)

Строки полезны для хранения данных, которые можно представить в текстовой форме.

Mozilla Developer Network



# Создание строки

// Пустая строка

```
const emptyString = '';
```

// Длина строки

```
emptyString.length; // 0
```

# Создание строки

// Можно использовать одинарные кавычки

```
const russianString = 'строка текста';
```

```
russianString.length; // 13
```

// Можно использовать двойные кавычки

```
const russianString = "строка текста";
```

```
russianString.length; // 13
```

# Создание строки

```
const escapeCodesString = 'a\'b'; // a'b
```

```
escapeCodesString.length; // 3
```

```
const escapeCodesString = "a\"b"; // a"b
```

```
escapeCodesString.length; // 3
```

# Создание строки

```
const escapeCodesString = 'a\\b'; // a\b
```

```
escapeCodesString.length; // 3
```

```
const escapeCodesString = 'a\n\tb'; // a  
                                     //      b
```

```
escapeCodesString.length; // 4
```

# Создание строки

```
// Поддерживаются все символы из Unicode  
const utf8String = '中文 español English বাংলা 日本語 ਪੰਜਾਬੀ';  
  
utf8String.length; // 53
```

# Создание строки

// Соединение строк

```
const concatString = 'Hello,' + ' World!'; // Hello, World!
```

```
const concatString = "Hello," + " World!"; // Hello, World!
```

# Создание строки

```
const world = 'World';  
const concatString = 'Hello, ' + world + '!'; // Hello, World!
```

```
const world = 'World';  
const concatString = `Hello, ${world}!`; // Hello, World!
```

# Строки являются незаписываемыми

```
const russianString = 'кот';
```

```
// Возможно обращение к символу по индексу  
russianString[1]; // 'о'
```

```
// Редактирование невозможно  
russianString[1] = 'и';  
russianString; // 'кот'
```



# Срезы из строк

```
// Обрезаем строку до 140 символов под длину твита
const longString = 'Очевидно проверяется, что математический анализ
существенно масштабирует интеграл по поверхности, что неудивительно.
```

Первая производная, очевидно, позиционирует  
ортогональный определитель.'

```
const shortString = longString;
```

```
if (longString.length > 140) {
  shortString = longString.slice(0, 139) + '...';
}
```

```
shortString; // 'Очевидно проверяется, что математический анализ
// существенно масштабирует интеграл по поверхности, что неудивительно.
// Первая производная, оч...'
shortString.length; // 140
```

# Поиск в строке

```
const tweet = 'PWA. Что это такое? Третий доклад на WSD в Питере Сергея Густуна #wstdays';
```

```
// Находим индекс первого вхождения подстроки в строке  
tweet.indexOf('#wstdays'); // 65
```

```
// Искомая подстрока отсутствует  
tweet.indexOf('#fronttalks'); // -1
```

# Поиск в строке

```
const tweet = 'PWA. Что это такое? Третий доклад на WSD в Питере Сергея Густуна #wstdays';
```

```
// Искомая подстрока присутствует  
tweet.includes('#wstdays'); // true
```

```
// Искомая подстрока отсутствует  
tweet.includes('#fronttalks'); // false
```

# Сравнение строк

```
'hi' === 'hi'; // true
```

```
'a' < 'b';      // true
```

```
'a' < 'ab';     // true
```

```
'bar' < 'foo';  // true
```

```
'1' > '12';     // false
```

```
'2' > '12';     // true
```

```
'12' < '5';     // true
```

# Сравнение строк

```
'foo'.localeCompare('bar');           // 1  
'abc'.localeCompare('b');             // -1  
'Привет!'.localeCompare('Привет!'); // 0
```

# Преобразование строки в число

<code>Number('123');</code>	<code>// 123</code>	<code>parseFloat('123');</code>	<code>// 123</code>
<code>Number('12.8');</code>	<code>// 12.8</code>	<code>parseFloat('12.8');</code>	<code>// 12.8</code>
<code>Number('12.8 ');</code>	<code>// 12.8</code>	<code>parseFloat('12.8 ');</code>	<code>// 12.8</code>
<code>Number(' 12.8');</code>	<code>// 12.8</code>	<code>parseFloat(' 12.8');</code>	<code>// 12.8</code>
<code>Number(' ');</code>	<code>// 0</code>	<code>parseFloat(' ');</code>	<code>// NaN</code>
<code>Number('');</code>	<code>// 0</code>	<code>parseFloat('');</code>	<code>// NaN</code>
<code>Number('12.8s');</code>	<code>// NaN</code>	<code>parseFloat('12.8s');</code>	<code>// 12.8</code>
<code>Number('s12.8');</code>	<code>// NaN</code>	<code>parseFloat('s12.8');</code>	<code>// NaN</code>

# Полезные функции для работы со строками

`toLowerCase` — приводим строку к нижнему регистру

`toUpperCase` — приводим строку к верхнему регистру

`trim` — удаляет пробельные символы с обеих сторон

`startsWith` — начинается ли строка с подстроки

`endsWith` — заканчивается ли строка подстрокой

Все функции для работы со строками

Перерыв?



# Массивы (Arrays)

Массивы являются спископодобными объектами, чьи прототипы содержат методы для операций обхода и изменения массива.

Ни размер JavaScript-массива, ни типы его элементов не являются фиксированными.

Mozilla Developer Network

# Создание массива

// Пустой массив

```
const emptyArray = [];
```

```
emptyArray.length; // 0
```

// Массив чисел

```
const arrayOfNumbers = [1, 2, 3, 4];
```

```
arrayOfNumbers.length; // 4
```

// Массив строк

```
const arrayOfStrings = ['a', 'b', 'c'];
```

```
arrayOfStrings.length; // 3
```

# Добавление в массив

```
const emptyArray = [];
```

```
// Добавляем элементы
```

```
emptyArray.push('a');
```

```
emptyArray.push('b');
```

```
emptyArray; // ['a', 'b']
```

```
emptyArray.length; // 2
```

# Удаление из массива

```
const someArray = ['a', 'b'];
```

```
// Удаляем последний элемент
```

```
someArray.pop(); // 'b'
```

```
someArray;       // ['a']
```

```
someArray.length; // 1
```

# Объединение массивов

```
const arrayOfNumbers = [1, 2, 3, 4];  
const arrayOfStrings = ['a', 'b', 'c'];  
  
const concatedArray = arrayOfNumbers.concat(arrayOfStrings);  
  
concatedArray; // [1, 2, 3, 4, 'a', 'b', 'c']  
  
arrayOfNumbers; // [1, 2, 3, 4]  
arrayOfStrings; // ['a', 'b', 'c']
```

# Итерирование по массиву

```
const tweets = [  
  'Я и IoT, пятый доклад на WSD в Питере Вадима Макеев #wstdays',  
  'Вёрстка писем. Развенчиваем мифы. Четвёртый доклад на WSD в Питере Артура Коха #wst  
  'PWA. Что это такое? Третий доклад на WSD в Питере Сергея Густуна #wstdays',  
  'Pokémon GO на веб-технологиях, второй доклад на WSD в Питере Егора Коновалова #wstd  
  'Ого сколько фронтендеров. #wstdays',  
  '<head> — всему голова, первый доклад на WSD в Питере Романа Ганина #wstdays',  
  'Доброе утро! WSD в Питере начинается через 30 минут: программа, трансляция и хештег  
  'Наглядная таблица доступности возможностей веб-платформы Пола Айриша: Can I use + S  
  'Node.js, TC-39 и модули, Джеймс Снел о проблемах Node.js с асинхронными модулями ES  
  'Всегда используйте <label>, перевод статьи Адама Сильвера в блоге Академии HTML',  
  'JSX: антипаттерн или нет? Заметка Бориса Сердюка на Хабре',  
  'Как прятать инлайновые SVG-иконки от читалок, Роджер Йохансен объясняет, зачем это  
];  
  
tweets.length; // 12
```

# Итерирование по массиву

```
const tweets = [...];  
  
for (let i = 0; i < tweets.length; i++) {  
    const tweet = tweets[i];  
  
    // Что-то делаем с конкретным твитом  
}
```

# Поиск в массиве твитов

```
// Найдём все строки, содержащие хештег #wstdays в массиве
const tweets = [...];

const result = [];

for (let i = 0; i < tweets.length; i++) {
  const tweet = tweets[i];

  if (tweet.includes('#wstdays')) {
    result.push(tweet);
  }
}
```



# Операции с массивом твитов

```
const tweets = [...];
```

```
// Постраничная навигация
```

```
const tweetsByPage = tweets.slice(5, 10);
```

# Операции с массивом твитов

```
const tweets = [...];
```

```
// Копируем оригинальную ленту в новый массив
```

```
const tweetsWithAdv = tweets.slice();
```

```
// Добавим немного рекламы в ленту, изменяя копию
```

```
tweetsWithAdv.splice(4, 0, 'Покупайте наших слонов!');
```

```
tweetsWithAdv.splice(9, 0, 'И натяжные потолки тоже отличные!')
```

```
tweets.length;           // 12
```

```
tweetsWithAdv.length;    // 14
```

# Сортировка массива

```
const arrayOfNumbers = [4, 1, 1000, 3, -1, 5];  
// сортирует исходный массив  
arrayOfNumbers.sort();  
arrayOfNumbers; // [ -1, 1, 1000, 3, 4, 5 ] (!)
```

```
const arrayOfStrings = ['ab', 'a', 'c', '10', '1', '2'];  
arrayOfStrings.sort();  
arrayOfStrings; // [ '1', '10', '2', 'a', 'ab', 'c' ]
```

# Полезные функции для работы с массивами

`shift` — выталкивает из массива первый элемент и возвращает его

`unshift` — добавляет элемент в начало массива

`reverse` — инвертирует порядок элементов в исходном массиве

`filter` — выбирает элементы исходного массива, удовлетворяющие условию, в новый массив

`map` — создает новый массив, содержащий преобразованные элементы исходного массива

`reduce` — преобразует исходный массив к одному новому значению

Все функции для работы с массивами

# Объекты (Objects)

Список, состоящий из пар с именем свойства и связанного с ним значения, которое может быть произвольного типа.

Mozilla Developer Network

# Создание объекта

```
// Пустой объект
```

```
const emptyObject = {};
```

```
// Присвоение значения свойства через точечную нотацию
```

```
emptyObject.propertyName = 'foo'; // { propertyName: 'foo' }
```

```
// Получение значения свойства через точечную нотацию
```

```
emptyObject.propertyName; // 'foo'
```

```
// Удаление свойства (пользоваться аккуратно)
```

```
delete emptyObject.propertyName; // true
```

```
emptyObject; // {}
```

# Обращение к свойствам объекта

// Объект с предопределенным набором свойств

```
const tweet = {  
  id: '782188596690350100',  
  text: 'Я и IoT, пятый доклад на WSD в Питере Вадима Макеева #wstdays',  
  user: {  
    id: 42081171,  
    name: 'Веб-стандарты',  
    screenName: 'webstandards_ru',  
    followersCount: 6443  
  },  
  hashtags: ['wstdays']  
};
```

```
tweet.id; // '782188596690350100'
```

```
tweet.user.screenName; // 'webstandards_ru'
```

// Получение значения свойства через квадратные скобки

```
tweet['i' + 'd']; // '782188596690350100'
```

# Итерирование по ключам объекта

```
const keys = Object.keys(tweet);  
  
keys; // ['id', 'text', 'user', 'hashtags']  
  
for (let i = 0; i < keys.length; i++) {  
  const key = keys[i];  
  const value = tweet[key];  
  
  // Что-то делаем с ключом и со значением  
}
```



# Проверка наличия свойства у объекта

```
tweet.hasOwnProperty('text'); // true  
tweet.hasOwnProperty('nonExistentProperty'); // false
```

```
'text' in tweet; // true  
'nonExistentProperty' in tweet; // false
```

```
tweet.nonExistentProperty; // undefined
```

# Функции (Functions)

Именованный блок кода, который позволяет переиспользовать существующий код.

Может иметь входные параметры и возвращать значение

Является объектом высшего порядка

# Декларация функции

```
function getFollowersCount() {  
    return 6443;  
}
```

# Декларация функции

```
function noop() {  
}
```

```
function noop() {  
    return undefined;  
}
```

```
function noop() {  
    return;  
}
```

# Декларация функции

```
function getAuthor(tweet) {  
    return tweet.user.screenName;  
}
```

# Передача по значению и по ссылке

```
const tweet = {  
  user: {  
    followersCount: 6443  
  }  
};  
  
function incrementFollowersCount(count) {  
  count++;  
}  
  
incrementFollowersCount(tweet.user.followersCount);  
  
tweet.user.followersCount; // 6444 6443
```

# Передача по значению и по ссылке

```
const tweet = {  
  user: {  
    followersCount: 6443  
  }  
};  
  
function incrementFollowersCount(user) {  
  user.followersCount++;  
}  
  
incrementFollowersCount(tweet.user);  
  
tweet.user.followersCount; // 6444
```

Функции — объекты высшего порядка

Они могут быть переданы в другие функции в качестве аргумента, а так же могут иметь личные свойства, как и другие объекты.



# Передача функций в качестве аргументов

```
const tweets = [  
  { hashtags: ['wstdays'], likes: 16, text: 'Я и IoT, пятый...' },  
  { hashtags: ['wstdays', 'mails'], likes: 33, text: 'Вёрстка писем...' },  
  { hashtags: ['wstdays'], likes: 7, text: 'PWA. Что это...' },  
  { hashtags: ['wstdays', 'pokemongo'], likes: 12, text: 'Pokémon GO на...' },  
  { hashtags: ['wstdays'], likes: 15, text: 'Ого сколько фронт...' },  
  { hashtags: ['wstdays', 'html'], likes: 22, text: '<head> – всему...' },  
  { hashtags: ['wstdays'], likes: 8, text: 'Доброе утро! WSD...' },  
  { hashtags: ['nodejs'], likes: 7, text: 'Node.js, TC-39 и модули,...' },  
  { hashtags: ['html'], likes: 28, text: 'Всегда используйте <label>...' },  
  { hashtags: ['svg'], likes: 19, text: 'Как прятать инлайновые...' },  
  { likes: 9, text: 'Наглядная таблица доступности...' },  
  { likes: 18, text: 'JSX: антипаттерн или нет?...' }  
];
```

# Передача функций в качестве аргументов

```
let result = [];  
  
// Выбираем только твиты с хештегом #wstdays  
  
function filterWithWstdaysHashtag(tweet) {  
  const hashtags = tweet.hashtags;  
  if (Array.isArray(hashtags) && hashtags.includes('wstdays')) {  
    result.push(tweet);  
  }  
}  
  
// Теперь в result лежат отфильтрованные твиты  
tweets.forEach(filterWithWstdaysHashtag);
```

# Передача функций в качестве аргументов

```
let result = tweets
    .filter(filterWithWstdaysHashtag)
    .map(render)
    .join('\n');

// Выбираем только твиты с хештегом #wstdays
function filterWithWstdaysHashtag(tweet) {
    const hashtags = tweet.hashtags;
    return Array.isArray(hashtags) && hashtags.includes('wstdays');
}

// Превращаем массив объектов твитов в HTML-строки
function render(tweet) {
    return `<dt>${tweet.text}</dt>
        <dd>${tweet.user}</dd>
        <dd>${tweet.hashtags.join(', ')}</dd>`;
}

// Теперь в result лежит HTML с деревом твитов
result = `<dl>${result}</dl>`;
```

# Передача функций в качестве аргументов

```
const likesCount = tweets.reduce(getTotalLikes, 0)
```

```
function getTotalLikes(acc, item) {  
  return acc + item.likes;  
}
```

```
likesCount; // 194
```

// можно переписать так

```
const likesCount = tweets.reduce(function (acc, item) {  
  return acc + item.likes;  
}, 0);
```

```
likesCount; // 194
```

# Стрелочные функции

```
// обычная функция, сохраненная в переменную  
const addFunc = function (a, b) { return a + b; }
```

```
addFunc(2, 4); // 6
```

```
// стрелочная функция, сохраненная в переменную  
const addArrowFunc = (a, b) => a + b;
```

```
addArrowFunc(2, 4); // 6
```

# Передача функций в качестве аргументов

```
const hashtagsStat = tweets
    .reduce(flattenHashtags, [])
    .reduce(getHashtagsStats, {});
```

```
function flattenHashtags(acc, item) {
    return acc.concat(item.hashtags || []);
}
```

```
function getHashtagsStats(acc, item) {
    if (!(item in acc))
        acc[item] = 0;
}
```

```
acc[item]++;

return acc;
}
```

```
hashtagsStats; // { html: 2, mails: 1, nodejs: 1, pokemongo: 1, svg: 1, wstdays: 7 }
```

# Передача функций в качестве аргументов

```
const hashtagsStat = tweets

  .reduce(function (acc, item) {
    return acc.concat(item.hashtags || []);
  }, [])

  .reduce(function (acc, item) {
    if (!(item in acc)) {
      acc[item] = 0;
    }

    acc[item]++;

    return acc;
  }, {});

hashtagsStats; // { html: 2, mails: 1, nodejs: 1, pokemongo: 1, svg: 1, wstdays: 7 }
```

# Передача функций в качестве аргументов

```
const hashtagsStat = tweets
```

```
.reduce((acc, item) => acc.concat(item.hashtags || []), [])
```

```
.reduce((acc, item) => {  
  if (!(item in acc)) {  
    acc[item] = 0;  
  }  
  acc[item]++;  
  return acc;  
}, {});
```

```
hashtagsStats; // { html: 2, mails: 1, nodejs: 1, pokemongo: 1, svg: 1, wstdays: 7 }
```



# Передача функций в качестве аргументов

```
const hashtagsStats = {html: 2, mails: 1, nodejs: 1, pokemongo: 1, svg: 1, wstdays: 7};

const sorted = Object
  .keys(hashtagsStats) // ['html', 'mails', 'nodejs', 'pokemongo', 'svg', 'wstdays']
  .map(key => {
    return {
      tag: key,
      count: hashtagsStats[key]
    };
  }) // [ { tag: 'html', count: 2 }, { tag: 'mails', count: 1 } ,...]
  .sort((a, b) => {
    if (a.count !== b.count) {
      return a.count > b.count ? -1 : 1; // сортируем от большего к меньшему
    }
    return a.tag.localeCompare(b.tag);
  }) // [ { tag: 'wstdays', count: 7 }, { tag: 'html', count: 2 } ,...]
  .reduce((acc, item) => {
    acc[item.tag] = item.count;
    return acc;
  }, {});

sorted; // { wstdays: 7, html: 2, mails: 1, nodejs: 1, pokemongo: 1, svg: 1 }
```

# typeof (Определение типа в runtime)

```
typeof true;           // 'boolean'
typeof undefined;      // 'undefined'
typeof 'строка';       // 'string'
typeof { baz: 1 };     // 'object'

typeof 12.4;           // 'number'
typeof NaN;            // 'number'
typeof Infinity;       // 'number'

function foo() {}
typeof foo;            // 'function'

typeof [1, 2];          // 'object', используйте Array.isArray()
typeof null;           // 'object' (!)
```

Спасибо!  
Вопросы?