Сентябрь 8

PROGOROD WEB-GIS Javascript API

Версия 0.0.4

Описание программных интерфейсов для разработчиков.

121354, г. Москва, ул. Дорогобужская, д. 14, стр.1Е (с) Тел.: +7 (495) 661-80-21, web: http://pro-gorod.ru, ООО «ПроГород»

Содержание

ОБЩЕЕ ОПИСАНИЕ 4

Стандартные элементы управления 4
Масштабирование 4
Линейка 4
Маршрутизация 4
Информационные элементы 4
Масштаб 4
Координаты 4
Миникарта 4
Лицензионная информация 4
Базовый функционал JavaScript API 4
Поддержка множественных растровых слоев 4
Callback'и 5
Пользовательские пиктограммы 5
Включение/отключение стандартных, информационных и нестандартных элементов управления 5
Создание/удаление стандартных и нестандартных объектов 5
Преобразование координат 5
Расчет оптимального зума 5
Вычисление расстояния между двумя точками 5
Дополнительный функционал 5
Widgets 5
Прямое и обратное геокодирование 5
Кроссдоменные асинхронные запросы (AJAX) 5
Обработчик стандартных географических форматов 5
Безопасность 6
Ограничения ключа API 6

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ 6

Поддерживаемые раузеры 6
Плагины и др. 6
Внешние библиотеки 6

МЕТОДЫ JAVASCRIPT API 6

Начало работы с API 6
Процедура получения ключа API 6
Подключение JavaScript API 7
Инициализация карты 7
Конструктор 7
Центрирование карты при инициализации (и после) 8
Задание дополнительных растровых слоев 8
Удаление дополнительных растровых слоев 8
Включение стандартных элементов управления 9
Отключение стандартных элементов управления 9
Включение нестандартных элементов 9
Отключение нестандартных элементов управления 9
Задание смещения по оси X,Y блокам элементов управления 9

Панель стандартных элементов управления 9
Панель нестандартных элементов управления 9
Использование callback'ов в стандартных и нестандартных элементах управления 9
События к которым применимы callback'и 9
Определение и вызов callback'ов 9
Включение информационных элементов 10
Масштаб карты 10
Координаты 10
Миникарта 10
Лицензионная информация 10
Отключение информационных элементов 10
Масштаб карты 10
Координаты 10
Миникарта 10
Лицензионная информация 10
Управление стандартными объектами методами Javascript 10
Перечень стандартных геометрических DOM-объектов 11
Перечень стандартных геометрических Canvas-объектов 11
Создание стандартных геометрических DOM-объектов 11
Создание стандартных геометрических Canvas-объектов 15
Управление нестандартными методами Javascript API 19
Управление объектом «Маршрутизация» 19
Управление дополнительным функционалом Javascript API 20
Геокодирование 20
Widgets API 22
Вспомогательные методы 22
Показ/скрытие текущей геометрии 22
Получение координат окна/центра/зума 22
Центрирование по оптимальному зуму по bbox'у объекта 22
Конвертирование координат 22
Расчет расстояния между двумя точками 23
Кроссдоменные асинхронные запросы (AJAX) 23
Форматирование строки в JSON 23
Обработчик стандартных географических форматов 24

ПЕРЕЧЕНЬ ИЗМЕНЕНИЙ 24

Общее описание

PROGOROD WEB-GIS Javascript API является клиентским приложением на Javascript, взаимодействующим с серверной частью PROGOROD HTTP API. Средой исполнения приложения являются стандартные браузеры (см. «Технические требования»). Взаимодействие с серверной частью осуществляется по протоколу HTTP методами GET и POST.

Стандартные элементы управления

Стандартными инструментами являются элементы управления интегрированные во Javascript-компонент, позволяющие управлять картой и создавать определенные API географические объекты.

Масштабирование

Переключатель уровня приближения (далее «зум») карты. В текущей конфигурации доступно 14 уровней (от 1 до 14).

Линейка

Частный случай полилинии. Создает стандартный объект, недоступный через Javascript API.

Маршрутизация

Частный случай полилинии и метода AJAX. Графическое отображение маршрута, проложенного по двум вершинам.

Информационные элементы

Масштаб

Отключаемая шкала текущего масштаба.

Координаты

Отображение текущих координат. Инструмент работает в двух режимах: динамический и статический. В статическом отображаются координаты центра области видимости, обновляющиеся при окончании действия перемещения по карте. В динамическом отображаются координаты мыши.

Миникарта

В левом нижнем углу карты отображается уменьшенное изображение текущего положения с уменьшением шага зума на 2 от текущего.

Лицензионная информация

Неотключаемая информация об авторских правах и картографической лицензии в правом нижнем углу карты.

Базовый функционал Javascript API

Поддержка множественных растровых слоев

Рабочая область приложения позволяет одновременно отображать несколько растровых картографических слоев, выполненных в текущей проекции.

Callback'и
Ряд элементов управления снабжен callback'ами, что дает возможность использовать дополнительный функционал сразу после выполнения какого-либо события, созданного этим элементом.

Пользовательские пиктограммы
Все графические элементы (пиктограммы) стандартных и нестандартных элементов управления могут изменяться пользователем на свои собственные путем передачи в соответственные управляющие объекты ссылок на необходимые графические элементы.

Включение/отключение стандартных, информационных и нестандартных элементов управления
Все элементы управления могут быть включениы или отключены по желанию пользователя методами Javascript API.

Создание/удаление стандартных и нестандартных объектов
Используя Javascript API пользователь в любой момент может создать любой объект стандартного типа (см. «Перечень стандартных объектов») средствами DOM или Canvas.

Преобразование координат
Преобразование абсолютных координат в координаты проекции Меркатора и обратно.

Расчет оптимального зума
Расчет оптимального зума по bbox'у объекта.

Вычисление расстояния между двумя точками
Возвращает метрическое расстояния между двумя заданными в формате lon/lat точками.

Дополнительный функционал

Widgets
Втсроенный API, позволяющий интегрировать собственные приложения в среду карты.

Прямое и обратное геокодирование
Адресный поиск с нечеткой логикой — прямое геокодирование. Обратное геокодирование — поиск в заданом радиусе по переданным координатам. Методы возвращают объект с координатами, которые дальше могут быть отображены на карте и т.п. В объекте содержится также подробная описательная информация и точки интересов (далее «POI»).

Кроссдоменные асинхронные запросы (AJAX)
Встроенный метод, позволяющий делать кросдоменные AJAX-запросы средствами Flash.

Обработчик стандартных географических форматов
В текущей версии поддерживается обработка и отображение форматов KML, NMEA.

Безопасность

Доступ к API осуществляется по 128-битному ключу.

Ограничения ключа API

- Ограничения API уникальны для каждого ключа.
- Доменное ограничение. Один ключ на один домен.
- Географическое ограничение. Ограничение на географическую зону.
- Ограничение на кол-во загрузок API.

Технические требования

Поддерживаемые раузеры

- Firefox 3.0 и выше
- IE 7 и выше
- Google Chrome 4.0 и выше
- Safari 3.0 и выше

Плагины и др.

- Javascript
- Flash player не ниже 10
- Отсутствие с пользовательской стороны ограничений на загрузку и выполнение Javascript и Flash.

Внешние библиотеки

- Swfobject
- JSON с 0.1.9

Методы Javascript API

Начало работы с API

Получите уникальный ключ API. Подключение API возможно только по уникальному ключу, определяющему объем доступного пользовательского функционала в текущей загрузке клиентского приложения.

Процедура получения ключа API

- Получить уникальный ключ можно сделав запрос по электронной почте api@pro-gorod.ru.
- В запросе укажите URL сайта, на котором будет использоваться API, а также ваш электронный адрес.
- В течении 2-дней вам будет выдан 128-битный ключ в шестнадцатиричном формате (32 символа 0-9a-f).
- До подписание договора и оплаты услуг по договору ключ имеет тридцатидневную триал-версию.
- После подписания договора и оплаты услуг ключ действует сроком 1 год с момента подписания договора.
- По техническим вопросам обращаться по адресу api@pro-gorod.ru.

Подключение Javascript API

Общий случай включения в заголовок (x)HTML-документа с передачей ключа API:

```
<script type="text/Javascript" charset="utf-8"
src="http://js.tmcrussia.com/map.js?e16a5f67e5cc3887282038bf6bc27657&amp;version=0.0.4"></script>
```

Инициализация карты

Инициализация карты должна происходить после формирования DOM. По-умолчанию включен только один элемент управления — «Масштабирование». Наиболее общий случай:

```
<script type="text/Javascript">
window.onload  =  function() {
        var map, params, el = document.getElementById('myMap');
        try {
                params = {
                        roundRobin: {
                                tiles: [
                                        'http://h01.tiles.maps.rambler.ru/map/',
                                        'http://h02.tiles.maps.rambler.ru/map/',
                                        'http://h03.tiles.maps.rambler.ru/map/',
                                        'http://h04.tiles.maps.rambler.ru/map/'
                                       ]

                        },
                        zoom: 9
                };
                map = new Map(el,params);
                map.setCenter({lon: 4187521.87177, lat: 7473956.84308, zoom: 10});
        } catch (e) {
                alert(e);
        }
}
</script>
```

Вызов через ready-callback:

```
map.ready = function() {
  console.log(map.getBboxCoords());
};
```

(x)HTML: `<div id="myMap"></div>`

Конструктор

В конструктор передается объект и параметры инциализации. В противном случае будет возвращен Exception.
Экземпляр объекта: new PGmap
Параметры:
(object) {

(object) roundRobin: { //клиентский roundrobin для обхода лимита на единовременные запросы из браузера
(array) tiles: [
(string) URL,
(string) URL…
]
},
(int) zoom: 9, //текущий зум
(int) minZoom: 4, //минимальный зум
(bool) noKinematicDrop: 1, //эффект кинетики при перемемщении
(bool) noSmoothZoom: 1 //отключение плавного зума

}
Пример:

```
params = {
    roundRobin: {
        tiles: [
            'http://h01.tiles.tmcrussia.com/map/',
            'http://h02.tiles.tmcrussia.com/map/',
            'http://h03.tiles.tmcrussia.com/map/',
            'http://h04.tiles.tmcrussia.com/map/'
        ]
    },
    zoom: 9
    minZoom: 4,
    noKinematicDrop: 1,
    noSmoothZoom: 1
};
map = new PGmap(mapNode, params);
```

Центрирование карты при инициализации (и после)

Метод: setCenter
Параметры:
(object) {
lon: (float) lon,
lat: (float) lat,
zoom: (int) zoom — диапозон от 0 до 16
}
Пример: map.setCenter({lon: 4187521.87177, lat: 7473956.84308, zoom: 10})

Задание дополнительных растровых слоев

Метод: layers.create
Параметры: (string) URL
Пример:

```
map.layers.create('http://traffic.tmcrussia.com/');  //(пробочный слой)
map.layers.create('http://yoursite.com/lv2/');
```

Удаление дополнительных растровых слоев

Метод: layers.remove
Параметры: (int) index

Пример:

```
map.layer.remove(0);
map.layer.remove(1);
```

Включение стандартных элементов управления

По умолчанию включен только элемент «Масштабирование»

…

Отключение стандартных элементов управления

…

Включение нестандартных элементов

…

Отключение нестандартных элементов управления

…

Задание смещения по оси x,y блокам элементов управления

Панель стандартных элементов управления

…

Панель нестандартных элементов управления

…

Использование callback'ов в стандартных и нестандартных элементах управления

События к которым применимы callback'и

- Изменение положения видимой области карты
- Изменение зума

Определение и вызов callback'ов

Вызов callback'а определен с соглашением camel case как постфикс к типу элемента (за исключением событий):

- Событие перемещения по карте MAP_MOVE, MAP_MOVED
- Событие изменения зума MAP_ZOOME, MAP_ZOOMED
- События маршрутизации:

  1) route.callBack.get // возникает по созданию маршрута

  Флаг error в объекте маршрута может иметь следующие значения.

  0 — нет ошибки
  1 — необходим перезапуск сервиса (серверная сторона)
  2 — не установлен финиш
  3 — не установлен старт
  4, 5, 6 — зарезервированные значения
  7 — неправильная точка старта
  8 — неправильная промежуточная точка (suberror - номер, не считая старта)
  9 — неправильная точка финиша
  10 — изолированная точка старта
  11 — изолированная промежуточная точка (suberror - номер, не считая

старта)
12 — изолированная точка финиша
13 — несвязный путь
14 — маршрутизация не удалась (suberror=0 — мало памяти, ошибка в данных и т.п. точно не известно; =1 — с первой попытки был код 1, но перезапуститься не удалось; =2 — с первой попытки был код 1, после перезапуска опять 1)

Пример:

```
map.event.user(PGmap.Events.MAP_MOVE, function(){
        console.log('mapmover' + new Date().getTime());
});
```

Включение информационных элементов

Масштаб карты

…

Координаты

…

Миникарта

Beta hook:

```
map.globals.minimap ={width: 200, height: 140, shift: 7, border:'1px dotted #98a', frameRelMax: 0.4};
minimap01 = new PGmap.MinimapCont(map.globals); //создание, подключение миникарты
mapNode.appendChild(minimap01.element);
```

Лицензионная информация

Не имеет кастомизации. Отключать разрешается только при наличии соответственного лицензионного соглашения и договора.

Отключение информационных элементов

Масштаб карты

…

Координаты

…

Миникарта

…

Лицензионная информация

Не имеет кастомизации. Отключать разрешается только при наличии соответственного лицензионного соглашения и договора.

Управление стандартными объектами методами Javascript

Javascript-версия API обладает встроенными методами для отображения геометрических элементов. Есть два возможных способа реализации: DOM () и Canvas.

Перечень стандартных геометрических DOM-объектов

- Точка (PGmap.Point)
- Полигон (PGmap.Polygon)
- Полилиния (PGmap.Polyline)

Перечень стандартных геометрических Canvas-объектов

- Круг (PGmap.Graphics.Circle)
- Эллипс (PGmap. Graphics.Ellipse)
- Полилиния (PGmap. Graphics.Path)
- Прямоугольник (PGmap. Graphics.Rect)
- Прямоугольник с закругленными углами (PGmap. Graphics.RoundRect)

Создание стандартных геометрических DOM-объектов

Создание объекта

Создает графический объект, с учетом параметра.

Метод: geometry.create
Параметры: (string) “point” | “rectangle” | “polyline” | “polygon” | “widget”, “route”, “ruler”)
Пример:
map.geometry. create ('point');

Добавление на карту

Созданный геометрический объект может быть добавлен на карту и в список элементов объекта map.geometry.
Метод: geometry.add
Параметры: (object) new PGmap.Point | PGmap.Polygon | PGmap.Polyline
Пример:
map.geometry.add(myLine);

Удаление с карты

Удаляет объект с карты и из списка элементов объекта map.geometry.
Метод: geometry.remove
Параметры: (object) ссылка на объект типа new PGmap.Point | PGmap.Polygon | PGmap.Polyline
Пример:
map.geometry. remove (myLine);

Получение объектов

Метод позволяет получить один, или несколько графических объектов, добавленных на карту и в список элементов объекта map.geometry. Если val – строка со значением “all”, то возвращается весь список элементов объекта map.geometry – map.geometry.elements. Если val – объект, то он может содержать либо только свойство type, имеющее одно из следующих значений: “points”, “rectangles”, “polylines”, “polygons”, “route”, или “ruler”.

В таком случае вернется массив всех элементов данного типа, добавленных в геометрию. Либо, кроме свойства type, объект val может

иметь свойство id[Number], и тогда вернется один объект указанного типа имеющего индекс id.

Метод: geometry.get
Параметры: (object) {type: “points” | “rectangles” | “polylines” | “polygons” | “route” | “ruler”.

} | (string = 'all') | (object) {id: (int) x}
Примеры:
map.geometry. get ({id: 1});
map.geometry. get ('all');
map.geometry. get ({type: 'points'});

Показ/скрытие объектов

Методы показывают и скрывают соответственно один, или

несколько графических объектов, добавленных на карту и в список элементов объекта map.geometry. Значения аргумента аналогичны описанным для метода geometry.get.

Метод: geometry.show, geometry.hide
Параметры: (object) {type: “points” | “rectangles” | “polylines” | “polygons” | “route” | “ruler”.

} | (string = 'all') | (object) {id: (int) x}
Примеры:
map.geometry. show ({id: 1});
map.geometry. show ('all');
map.geometry. hide ({type: 'points'});

Общие свойства геометрических DOM-объектов

.element [DOM Object]

Ссылка на DOM объект, представления точки (только для чтения).

.left .top [Number]

Положение точки в пикселях (только для чтения, актуальны только для типов point и widget, для widget, может быть не актуально, в зависимости от содержимого widget).

.globals [Object]

Ссылка на объект географической проекции карты (только для чтения).

Общие методы геометрических DOM-объектов

.update()

Обновляет объект с учетом текущего масштаба и параметров объекта (для widget может быть переназначено).

.show() .hide()

Показывает и соответственно скрывает объект (для widget может быть переназначено).

Точка

Конструктор: PGmap.Point
Параметры:
(object) {
    (object) coord: new PGmap.Coord(lon, lat, true) || {lon: x, lat: x}
}
Свойства:

.coords [PGmap.Coord]

Координаты точки, после их изменения необходимо вызвать метод .update().

.balloon [PGmap.Balloon]

Ссылка на балун, привязанный к точке.

Методы:

. addBalloon(balloon [PGmap.Balloon])

Привязывает к точке balloon. Если у объекта balloon выставлен флаг isClosing = true, то добавляется обработчик события клик на точке, по которому показывается balloon.

. removeBalloon()

Удаляет объект balloon, привязанный к точке.

. showBalloon() . hideBalloon()

Показывает и скрывает соответственно balloon, привязанные к точке.

Пример:

```
point = new PGmap.Point({
    coord: new PGmap.Coord(lon, lat, true),
    url: '1.png'
});
map.geometry.add(point);
```

Полигон

Конструктор: PGmap.Polygon
Параметры:
(object) {
    (array) points: [
        new PGmap.Coord(lon, lat, true) || {lon: x, lat: x},
        new PGmap.Coord(lon, lat, true) || {lon: x, lat: x}…
    ],
    (object) style: {
        color: (hexadecimal) ‘#ffff00’, //цвет линии
        lineHeight: (int) 2, //толщина линии
        backgroundColor: (hexadecimal) ‘#ffff00’ //цвет фона
    }
}
Свойства:

.points [Array [PGmap.Coord]]

Массив точек, по которым строится линия.

.style [Object { lineHeight[Number] ,color[String]}]

Объект, описывающий внешний вид линии. После изменения одного из этих свойств необходимо вызвать метод .update().

Методы:

.setGeoArray(points [Array [x[Numaber], y[Number]]])

Метод определяет массив точек, по которым строится линия. Здесь points – массив вида [[x,y],[x,y]…[x,y]], где x,y – геокоординаты вида 37.418824, 55.712948.

Пример:

```
var polygon = new PGmap.Polygon({
        points: [
                new PGmap.Coord(4190877, 7457148),
                new PGmap.Coord(4207445, 7441492),
                new PGmap.Coord(4228877, 7448028)
        ],
        style: {
                color:"#ce5",
                lineHeight: 2,
                backgroundColor: "#8b9"
        }
});
map.geometry.add(polygon);
```

Полилиния

Конструктор: PGmap.Polyline

Параметры:

```
(object) {
        (array) points: [
                new PGmap.Coord(lon, lat, true) || {lon: x, lat: x},
                new PGmap.Coord(lon, lat, true) || {lon: x, lat: x}…
        ],
        (object) style: {
                color: (hexadecimal) ‘#ffff00’, //цвет линии
                lineHeight: (int) 2 //толщина линии в пикселях
        }
}
```

Свойства:

.points [Array [PGmap.Coord]]

массив точек, по которым строится линия.

.style [Object { lineHeight[Number] ,color[String]}]

объект, описывающий внешний вид линии. После изменения одного из этих свойств необходимо вызвать метод .update().

Методы:

.setGeoArray(points [Array [x[Numaber], y[Number]]])

метод определяет массив точек, по которым строится линия. Здесь points – массив вида [[x,y],[x,y]…[x,y]], где x,y – геокоординаты вида 37.418824, 55.712948.

Пример:

```
var polyline = new PGmap.Polyline({
        points: [
                new PGmap.Coord(4190877, 7477148),
                new PGmap.Coord(4207445, 7461492),
                new PGmap.Coord(4228877, 7468028)
        ],
        style: {
                color:"#895",
                lineHeight: 4
        }
});
map.geometry.add(polyline);
```

Создание стандартных геометрических Canvas-объектов

Методы и объекты модуля PGmap.Graphics

Создание Canvas-объекта

Создает и возвращает новый объект полотна (Canvas). Здесь необязательный аргумент el – DOM Object, в который будет добавлен объект Canvas и которому будет приписано свойство – el.graphic = Canvas. Необязательный аргумент p [Boolean] указывает, должен новый объект Canvas добавиться первым среди дочерних элементов объекта el. Также если аргумент p не задан вообще, то точкой от которой будет масштабироваться объект Canvas будет его центр, в ином случае это будет левый верхний угол объекта el.

Метод: setDrawleble
Параметры:

(DOM object) element, (Boolean) true|false

Примеры:
PGmap.Graphics. setDrawleble(document.getElementById('myId'),true);

Свойства Canvas-объекта

.parent

Ссылка на DOM объект, в который добавлен данный объект Canvas

.box

Объект рамки данного объекта Canvas. Здесь . box.x, . box.y — координаты левоверхнего угла рамки объекта, а . box.width, . box.height – ширина и высота рамки данного объекта

.elements

Массив дочерних графических объектов данного полотна (Canvas)

.element

Ссылка на DOM представление данного объекта Canvas

Методы Canvas-объекта

.shadow(off)

Показывать, или скрывать тень у всех объектов полотна (Canvas). Если необязательный аргумент off равен “off”, тень скрывается.

.scale(x,y)

Масштабировать все элементы полотна (Canvas). Здесь x и y – коэффициент изменения масштаба по оси X и по оси Y соответственно. (если x = 1 – настоящий размер).

.append(el)

Добавляет на полотно (Canvas) графический объект el.

.appendTo(el)

Добавляет данный объект Canvas в DOM объект el.

.remove(el)

Если задан необязательный аргумент el [графический объект], то данный метод удаляет графический объект el из данного полотна (Canvas), иначе удаляет данный объект Canvas из DOM модели.

.show(), .hide()

Показать скрыть объект Canvas.

.clear()

Очищает данное полотно (Canvas), удаляя из него все графические объекты.

Методы Canvas-объекта для рисования

.beginFill(color,alpha)

Задает цвет (color) и прозрачность (alpha) для заливки фигур, которые будут далее нарисованы методами полотна.

.endFill()

Очищает текущие настройки заливки для рисования по полотну.

.lineStyle(width,color,alpha)

Задает ширину линии (width), цвет (color) и прозрачность линии (alpha), которые будут использоваться далее для рисовании по полотну.

.drawPath()

Метод рисует на полотне фигуру «путь» с параметрами, заданными ранее методами .lineStyle() и .beginFill(). Аргументы этого метода такие же как в методе .Path основного объекта PGmap.Graphics.

.drawCircle(), .drawEllipse(), .draw.Rect(), .drawRoundRect()

методы аналогичны методу .drawPath(), описанному выше.

.moveTo(x,y), .lineTo(x,y,dn), .curveTo(x,y,ax,ay,dn)

Методы для рисования линии («пути»), здесь x,y – координаты следующей точки, в которую метод . moveTo() – перенесет «карандаш», метод .lineTo() – нарисует прямую линию с параметрами, заданными методами .lineStyle() и .beginFill(), а метод .curveTo() – нарисует дугу с опорной точкой с координатами ax и ay. Необязательный параметр dn – означает, будет ли нарисована данная фигура сейчас же (если dn = true) или после вызова метода .drawAll().

```
.drawAll()
```

Метод рисует фигуру заданную путем последовательного вызова методов .moveTo(), .lineTo() и .curveTo().

Инициализация стандартных геометрических Canvas-объектов

```
.Circle(x [Number], y [Number], r [Number])
```

Создает фигуру круг с центром в точке с координатами x,y и радиусом r.

```
.Ellipse(x [Number], y [Number], rx [Number], ry [Number])
```

Создает фигуру эллипс с центром в точке с координатами x, y и радиусами по оси X – rx, по оси Y – ry.

```
.Path([commands [Array [Number]]], data [Array [ {x:[Number], y:[Number]} |
                [Number] ]])
```

Создает фигуру «путь» (это линия построенная по нескольким точками в соответствии с указанными командами, т.е. точки могут быть соединены между собой прямой, дугой, или просто разрывом). Здесь необязательный аргумент commands – массив содержащий коды команд соединения точек (1 – разрыв, до следующей точки линия прорисована не будет, 2 – прямая, до следующей точки будет прорисована прямая, 3 – дуга, следующая точка будет использована как вспомогательная для построения дуги до следующей после нее точки). Аргумент data – массив, содержащий координаты для построения фигуры, может быть задан как массив объектов типа { x: [Number], y: [Number]}, или просто как единый массив координат – [x1,y1,x2,y2,x3,y3,…,xn,yn].

```
.Rect(x [Number], y [Number], w [Number], h [Number]) –
```

Создает фигуру прямоугольник левый верхний угол которой будет в точке с координатами x, y, а ширина и высота будут соответственно w и h.

```
.RoundRect(x [Number], y [Number], w [Number], h [Number], rx [Number],
                ry[Number])
```

Создает фигуру прямоугольник с закругленными углами, левый верхний угол которой будет в точке с координатами x, y, ширина и высота будут соответственно w и h а радиусы закруглений – rx и ry.

Все объекты могут быть созданы без аргументов, их базовые параметры (координаты, радиусы и т.д.) могут быть заданы позднее через метод .attr().

Свойства стандартных геометрических Canvas-объектов

```
.paper
```

Ссылка на объект Canvas, к которому добавлен данный объект

.attrs

Ссылка на объект параметров данного объекта

.box

Массив координат, ограничивающих данный объект

.rect

Объект, описывающий рамку данного объекта, здесь .rect.x, .rect.y – координаты левого верхнего угла рамки объекта, а .rect.width, .rect.height – ширина и высота рамки данного объекта

.node

Ссылка на DOM представление данного объекта

Методы стандартных геометрических Canvas-объектов

.attr(n[String | Object], [v])

Метод позволяет задать, или получить один, или несколько параметров графического объекта. Здесь n – может быть строкой и тогда этот аргумент означает имя параметра, если при этом передан аргумент v, то v – это новое значение параметра с именем n.Если аргумент v не передан, то метод вернет текущее значение параметра с именем n. Если n – это объект, то он должен иметь формат – {“Имя параметра”: “значение параметра”} и по мену устанавливаются перечисленные в нем параметры.

Графические фигуры поддерживают следующие параметры:

Только для фигуры “путь” (Path):

points , data – массив координат по которым будет строиться «путь», формат такой же как в методе Graphics.Path. commands – массив команд, по которым будет строиться «путь», формат такой же как в методе Graphics.Path (данный параметр устанавливается только вместе с points, или data).

path – объект полученный методом .attr(“path”), задает построение «пути».

stroke – задает цвет линии
stroke-width – задает толщину линии
stroke-opacity – задает прозрачность линии
fill – задает цвет заливки
fill-opacity – задает прозрачность заливки
x, y – координаты центра (для круга и эллипса), или координаты левого верхнего угла (для прямоугольников)
width, height – ширина и высота для прямоугольников.
r, rx, ry – радиусы для окружности и эллипса, или радиусы закруглений для прямоугольника с закругленными углами.

.appendTo(el [Object Canvas | DOM Object], [inhtml [Boolean]])

Добавляет графический объект в объект Canvas, или в DOM объект. Здесь el – элемент, в который будет добавлен данный объект, el может быть объектом Canvas, или DOM объектом. Если el – DOM объект, у которого есть

свойство graphic и это объект Canvas, то данный объект будет добавлен в el.graphic. Необязательный аргумент inhtml указывает должен ли данный объект добавиться в объект el (если это DOM объект) независимо от наличия у el свойства graphic.

.remove()

Удаляет данный графический объект объекта Canvas, если он был туда добавлен.

.show(), .hide()

Показать, скрыть графический объект.

.makeCanvas(el)

Данный метод создает и возвращает новый объект Canvas, добавляя в него данный графический объект. Здесь el – необязательный аргумент DOM Object, в который будет добавлен созданный объект Canvas.

Управление нестандартными методами Javascript API

Управление объектом «Маршрутизация»

Получение маршрута заданным координатам

Метод: route.get
Параметры:

```
(object){
        points: (array) [
                { lon: (float), lat: (float)},
                { lon: (float), lat: (float)},
                { lon: (float)', lat: (float)}
                ...
        ],
        method: (string) 'optimal' | 'shortest' | 'less_turns',  //необязательный
        (func) movemap,
        (string) url, //необязательный
        (func) callback
}
```

Пример:

```
var s = [{lon: 37.66, lat: 55.75}, {lon: 37.55, lat: 55.80}];
map.route.get(s, null, null, function(data){
        console.log(data);
});
```

Удаление маршрута

Метод: route.remove
Параметры: —
Пример: map.route.remove();

Удаление установленной пользователем точки маршрута

Число определенных пользователем или приложением точек вместе с начальной и конечной должно быть не менее 2.
Метод: route.removePoint

Параметры: (int) x
Пример: map.route.removePoint(1);

Управление дополнительным функционалом Javascript API

Полная документация по геокодеру выходит за пределы описания функционала непосредственно данного прилоения и высылается по требованию.

Геокодирование

Метод возвращает коллекцию объектов найденных адресов и POI с координатами.
Метод: search
Параметры:

```
(object) {
        type:  (string) ‘search’ (прямое) | ‘geocode’ (обратное) | ‘city’ | ‘pois’
        a: (string) ‘’ //аналогично type
        str: (string) ‘’ //только в случае search и geocode ,
        q|q[]: (string|array) ‘’,[‘’,’’] //аналог str
        lon: (float) x, //только в случае type: ‘geocode’, по умолчанию центр
        lat: (float) y, //только в случае type: ‘geocode’, по умолчанию центр
        r: (int) n, //радиус метрический, только в случае type: ‘geocode’
        lon1: (float) x,  //необязательный, по умолчанию текущий lbc
        lat1: (float) y,  //необязательный, по умолчанию текущий lbc
        lon2: (float) x,  //необязательный, по умолчанию текущий rtc
        lat2: (float) y,  //необязательный, по умолчанию текущий rtc
        z: (int [0-17]) z ,  //зум необязательный, по умолчанию текущий
},
(function) callback((object) result)
Результат callback’a:
(object)  {
        //найденные адреса
        addr: (object) {
                (int) id: {
                        addr: (string) '',  //строка адреса
                        id: (string) '',  //параметр для уточнеящего запроса
                        type: (string) '',  //тип
                        uuid: (string) '',
                        x: (string) '',  //lon
                        y: (string) '', //lat
                        zoom: (string) ''
                },
                ...
        }
        // возможно имевшиеся ввиду
        addr_alt: (object) {
                (int) id: {
                        addr: (string) '',
                        id: (string) '',
                        type: (string) '',
                        uuid: (string) '',
                        x: (string) '',
```

```
                y: (string) '',
                zoom: (string) ''
        },
        ...
},
poi: (object) {
        (bool) id: {
                24hour_flag:  (bool) false,
                emergency_flag: (string) '',
                info: (string) '', //описание
                poi_id: (string) '',
                time_domain: (string) '', //время работы POI
                addr: (string) ‘’,
                cats: (object) {
                        a_name: (string) ‘’,  //категория A
                        b_name: (string) ‘’,  //категория B (вложенная в A)
                        c_name: (string) ‘’   //категория C (вложенная в B)
                },
                name: (string)  ””, //название POI
                x: (string) '',
                y: (string) '',
                www: (string) '' //poi url
                tel: (string) ‘’ //телефон
        },
        ...
}},
//возможно имевшиеся ввиду
poi_alt: (object) {
        (bool) id: {
                24hour_flag:  (bool) false,
                emergency_flag: (string) '',
                info: (string) '', //описание
                poi_id: (string) '',
                time_domain: (string) '', //время работы POI
                addr: (string) ‘’,
                cats: (object) {
                        a_name: (string) ‘’,  //категория A
                        b_name: (string) ‘’,  //категория B (вложенная в A)
                        c_name: (string) ‘’   //категория C (вложенная в B)
                },
                name: (string)  ””, //название POI
                x: (string) '',
                y: (string) '',
                www: (string) '' //poi url
                tel: (string) ‘’ //телефон
        },
        ...
}
```

```
}
Пример:
map.search({type: 'search', str: 'ресторан'}, function(e) {
        if (e != false) {
                console.log(e);
        }
});
map.search({type: 'geocode',  lon: 4187521.87177, lat: 7473956.84308},
function(e) {
        if (e != false) {
                console.log(e);
        }
});
map.search({type: 'city'}, function(e){console.log(e);});
```

Widgets API

Методы встраивания пользовательских виджетов в PROGOROD Javascript API.
...

Вспомогательные методы

Показ/скрытие текущей геометрии

Методы скрывают текущую геометрию для стандартных и нестандартных объектов. По умолчанию скрывает/показывает все.
Методы: geometry.hide, geometry.show
Параметры:
(object) PGmap.Point | PGmap.Polygon | PGmap.Polyline
Пример: map.hideGeometry(), map.showGeometry({id: 1, type: 'polygon'})

Получение координат окна/центра/зума

Метод возвращает коллекцию, содержащую координаты области видимости (левый нижний и правый верхний углы), центра и зум.
Метод: getBboxCoords
Пример: console.log(map.getBboxCoords());

Центрирование по оптимальному зуму по bbox'у объекта

...

Конвертирование координат

Конвертирование координат из проекции Меркатора в абсолютные

Методы: PGmap.Utils.fromMercX, PGmap.Utils.fromMercY
Параметры: (float) lon, (float) lat
Пример: PGmap.Utils.fromMercX(4187445), PGmap.Utils.fromMercY(7477148)

Конвертирование абсолютных координат в координаты проекции Меркатора

Методы: PGmap.Utils.lonLatToXY
Параметры:

(object) {
(float) lon,
(float) lat,
(int) zoom
}
Пример: PGmap.Utils.lonLatToXY({lon: 4188208.9332545744, lat: 7473382.722287859, zoom: 12})

Получение пиксельного расстояния от левого верхнего угла рабочего приложения по координатам Меркатора

Метод доступен только после инициализации карты.
Методы: PGmap.Utils.mercX, PGmap.Utils.mercY
Параметры: (float) lon, (float) lat
Пример: mercX(lon), mercY(lat)

Перевод времени в абсолютные координаты

Методы: PGmap.Utils.timeToDegreesLon, PGmap.Utils.timeToDegreesLat
Параметры: (float) lon, (float) lat
Пример: PGmap.Utils.mercX(PGmap.Utils.timeToDegreesLon(03736.0067))

Расчет расстояния между двумя точками

Метод: PGmap.Utils.distVincenty
Параметры:
(float) lon1,
(float) lat1,
(float) lon2,
(float) lat2
Пример: PGmap.Utils.distVincenty(lon1, lat1, lon2, lat2)

Кроссдоменные асинхронные запросы (AJAX)

Использование данного функуонала требует разраешения с принимающей стороны в формате crossdomain.xml.
Метод: ajax
Параметры:
(object) {
url: (string) '',
(function) callback((string | object) result)
}
Пример:
map.ajax({
url: 'http://traffic.tmcrussia.com/traffic_status.js',
callback: console.log
});

Форматирование строки в JSON

Метод: JSON.parse()
Параметры: string
Пример: var jsonObject = JSON.parse('some: "object"');

Обработчик стандартных географических форматов

Обработчик стандартных географических форматов KML, nmea.
Результирующий объект попадает в отображение методом freeHand.
Метод: PGmap.GeoParser.parseKML, PGmap.GeoParser.parseNMEA
Параметры:
(string) KML | NMEA
Пример:

```
PGmap.GeoParser.parseKML(myDocumentString);
```

Перечень изменений