Ноябрь | 17

PROGOROD WEB-GIS Javascript Flash API

Версия 0.2.8

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 5

Стандартные элементы управления 5
Масштабирование 5
Перемещение по карте 5
Точка 5
Полилиния 5
Полигон 5
Прямоугольник 5
Нестандартные элементы управления 5
Карандаш 5
Линейка 5
Скриншот 6
Маршрутизация 6
Информационные элементы 6
Масштаб 6
Координаты 6
Миникарта 6
Лицензионная информация 6
Базовый функционал Flash-компонента и Javascript Flash API 6
Поддержка множественных растровых слоев 6
Callback'и 6
Пользовательские пиктограммы 6
Включение/отключение стандартных, информационных и нестандартных элементов управления 6
Создание/удаление стандартных и нестандартных объектов 6
Преобразование координат 7
Расчет оптимального зума 7
Вычисление расстояния между двумя точками 7
Дополнительный функционал 7
Widgets 7
Пермалинки 7
Прямое и обратное геокодирование 7
Кроссдоменные асинхронные запросы (AJAX) 7
Обработчик стандартных географических форматов 7
Безопасность 7
Ограничения ключа API 7

ТЕХНИЧЕСКИЕ ТРЕБОВАНИЯ 8

Поддерживаемые раузеры 8
Плагины и др. 8
Внешние библиотеки 8

МЕТОДЫ JAVASCRIPT FLASH API 8

Начало работы с API 8
Процедура получения ключа API 8
Подключение Javascript Flash API 8
Инициализация карты 8

Конструктор 9
Центрирование карты при инициализации (и после) 9
Задание дополнительных растровых слоев 9
Удаление дополнительных растровых слоев 9
Центрирование карты после инициализации 10
Включение стандартных элементов управления 10
Перемещение по карте 10
Точка 10
Полилиния 10
Полигон 10
Прямоугольник 11
Масштабирование (зум) 11
Отключение стандартных элементов управления 12
Перемещение по карте 12
Точка 12
Полигон 12
Полилиния 12
Прямоугольник 12
Масштабирование (зум) 12
Включение нестандартных элементов 12
Карандаш 12
Маршрутизация 12
Скриншот 13
Линейка 13
Отключение нестандартных элементов управления 13
Карандаш 13
Маршрутизация 13
Скриншот 13
Линейка 13
Задание смещения по оси X,Y блокам элементов управления 14
Панель стандартных элементов управления 14
Панель нестандартных элементов управления 14
Использование callback'ов в стандартных и нестандартных элементах управления 14
Элементы, к которым применимы callback'и при создании через элементы управления и Javascript 14
События к которым применимы callback'и 14
Определение и вызов callback'ов 14
Включение информационных элементов 15
Масштаб карты 15
Координаты 16
Миникарта 16
Лицензионная информация 16
Отключение информационных элементов 16
Масштаб карты 16
Координаты 16
Миникарта 16
Лицензионная информация 17
Управление стандартными объектами методами Javascript 17
Перечень стандартных объектов 17
Создание стандартных объектов 17
Изменение текущих стандартных объектов 20
Получение текущих стандартных объектов 22

Удаление стандартных объектов 23
Управление нестандартными методами JavaScript Flash API 23
Управление объектом «Карандаш» 23
Управление объектом «Маршрутизация» 24
Управление дополнительным функционалом JavaScript Flash API 24
Геокодирование 25
Widgets API 27
Вспомогательные методы 30
Показ/скрытие текущей геометрии 30
Получение координат окна/центра/зума 30
Центрирование по оптимальному зуму по bbox'у объекта 30
Конвертирование координат 31
Расчет расстояния между двумя точками 31
Кроссдоменные асинхронные запросы (AJAX) 32
Форматирование строки в JSON 32
Обработчик стандартных географических форматов 32

ПЕРЕЧЕНЬ ИЗМЕНЕНИЙ 32

0.2.8 32
0.2.7 33
0.2.6 33
0.2.5 34
0.2.4 34
0.2.3 34
0.2.2 34

Общее описание

PROGOROD WEB-GIS Javascript Flash API является клиентским приложением на JavaScript и Flash, взаимодействующим с серверной частью. 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 Flash-компонент, позволяющие управлять картой и создавать определенные географические объекты.

Масштабирование

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

Перемещение по карте

Элемент управления, позволяющий перемещать область видимости карты методом drag'n'drop.

Точка

Создание и установка произвольной точки в произвольной видимости. Создает стандартный объект (см. «Перечень стандартных объектов»), доступный через Javascript Flash API.

Полилиния

Создание произвольной полилинии в области видимости карты. Создает стандартный объект, доступный через Javascript Flash API.

Полигон

Создание многоугольника в области видимости карты. Создает стандартный объект, доступный через Javascript Flash API.

Прямоугольник

Частный случай полигона. Создает стандартный объект, доступный через Javascript Flash API.

Нестандартные элементы управления

Карандаш

Позволяет рисовать произвольные объекты на карте с возможностью экспорта (в т.ч. на печать) видимой области карты и объектов на ней, включая вышеперечисленные.

Линейка

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

Скриншот
Сохранение выделенной области карты на физический жесткий диск в виде графического jpeg-файла.

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 правом верхнем 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 Flash-компонента и Javascript Flash 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 Flash API.

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

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.

Пермалинки
Ссылка на состояние карты, включающая позиционирование, зум и стандартные объекты.

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
Безопасность ключа API и пользовательских прав, обеспеченных этим ключом, защищена двусторонним шифрованием уровня обмена клиент-сервер при помощи алгоритма RSA и защитой от декомпилирования.

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
- ~~CrossXHR~~ (отсутствует с 0.1.8)
- JSON с 0.1.9

Методы Javascript Flash 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 Flash API

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

```
<script type="text/javascript" charset="utf-8"
src="http://flash.tmcrussia.com/map.js?e16a5f67e5cc3887282038bf6bc27657
&amp;version=0.2.8"></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;
        var el = document.getElementById('myMap');
        try {
                map = new Map(el);
                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:

Конструктор

В конструктор передается объект. В противном случае будет возвращен Exception.

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: layer.create
Параметры: (string) URL
Пример:

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

Удаление дополнительных растровых слоев

Метод: layer.remove
Параметры: (int) index
Пример:

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

Центрирование карты после инициализации

Метод: updateMap удален с версии 0.2.1 См. метод setCenter

Включение стандартных элементов управления

По умолчанию включен только элемент «Масштабирование»
Начиная с версии 0.2.5 параметры пользовательской кастомизации стандартных контролов временно изолированы. Функционал будет возвращен в версии 0.2.7.

Перемещение по карте

Метод: setControl
Параметры:
(string) 'navigate',
(object) {
upState: (string) '/images/button.png',
downState: (string) '/images/button.png',
overState: (string) '/images/button.png'
}
Пример: map.setControl('navigate');

Точка

Метод: setControl
Параметры:
(string) 'point',
(object) {
upState: (string) '/images/button.png',
downState: (string) '/images/button.png',
overState: (string) '/images/button.png'
}
Пример: map.setControl('point');

Полилиния

Метод: setControl
Параметры:
(string) 'polyline' ,
(object) {
upState: (string) '/images/button.png',
downState: (string) '/images/button.png',
overState: (string) '/images/button.png'
}
Пример: map.setControl('polyline');

Полигон

Метод: setControl
Параметры:
(string) 'polygon' ,
(object) {
upState: (string) '/images/button.png',
downState: (string) '/images/button.png',
overState: (string) '/images/button.png'
}

Пример: map.setControl('polygon');

Прямоугольник

Метод: setControl
Параметры:
(string) 'rectangle' ,
(object) {
upState: (string) '/images/button.png',
downState: (string) '/images/button.png',
overState: (string) '/images/button.png'
}
Пример: map.setControl('rectangle');

Масштабирование (зум)

Включено по умолчанию.
Метод: setControl
Параметры:
(string) 'slider',
(object){
plus: (object){
upState: (string) '/images/button.png',
downState: (string) '/images/button.png',
overState: (string) '/images/button.png'
},
minus: (object){
upState: (string) '/images/button.png',
downState: (string) '/images/button.png',
overState: (string) '/images/button.png'
},
offsetX: (int) x,
offsetY: (int) y
}
Пример:

```
var p = 'sys_pix/controls/m-';
var e = '.png';
map.setControl('slider', {
        plus: {
                upState:   p+'plus'+e,
                downState: p+'plus_f3'+e,
                overState: p+'plus_f2'+e
        },
        minus: {
                upState:   p+'minus'+e,
                downState: p+'minus_f3'+e,
                overState: p+'minus_f2'+e
        },
        offsetX: -30,
        offsetY: -10
});
```

Отключение стандартных элементов управления

Перемещение по карте
Метод: unsetControl
Параметры: (string) 'navigate'
Пример: map.unsetControl('navigate');

Точка
Метод: unsetControl
Параметры: (string) 'navigate'
Пример: map.unsetControl('navigate');

Полигон
Метод: unsetControl
Параметры: (string) 'polygon'
Пример: map.unsetControl('polygon');

Полилиния
Метод: unsetControl
Параметры: (string) 'polyline'
Пример: map.unsetControl('polyline');

Прямоугольник
Метод: unsetControl
Параметры: (string) 'rectangle'
Пример: map.unsetControl('rectangle');

Масштабирование (зум)
Метод: unsetControl
Параметры: (string) 'slider'
Пример: map.unsetControl('slider');

Включение нестандартных элементов
С версии 0.2.5 по 0.2.7 параметры пользовательской кастомизации нестандартных контролов временно изолированы.

Карандаш
Метод: setControl
Параметры:
(string) 'freeHand' ,
(object) {
	upState: (string) '/images/button.png',
	downState: (string) '/images/button.png',
	overState: (string) '/images/button.png'
}
Пример: map.setControl('freeHand');

Маршрутизация
Метод: setControl
Параметры:
(string) 'route' ,
(object) {

upState: (string) '/images/button.png',
downState: (string) '/images/button.png',
overState: (string) '/images/button.png'
}
Пример: map.setControl('route');

Скриншот

Метод: setControl
Параметры:
(string) 'screenShot' ,
(object) {
upState: (string) '/images/button.png',
downState: (string) '/images/button.png',
overState: (string) '/images/button.png'
}
Пример: map.setControl('screenShot');

Линейка

Метод: setControl
Параметры:
(string) 'ruler' ,
(object) {
upState: (string) '/images/button.png',
downState: (string) '/images/button.png',
overState: (string) '/images/button.png'
}
Пример: map.setControl('ruler');

Отключение нестандартных элементов управления

Карандаш

Метод: unsetControl
Параметры: (string) 'freeHand'
Пример: map.unsetControl('freeHand');

Маршрутизация

Метод: unsetControl
Параметры: (string) 'route'
Пример: map.unsetControl('route');

Скриншот

Метод: unsetControl
Параметры: (string) 'screenShot'
Пример: map.unsetControl('screenShot');

Линейка

Метод: unsetControl
Параметры: (string) 'ruler'
Пример: map.unsetControl('ruler');

Задание смещения по оси x,y блокам элементов управления

Панель стандартных элементов управления

Метод: setControl
Параметры:
(string) 'baseControls',
(object){
offsetX: (int) x,
offsetY: (int) y
}
Пример: map.setControl('baseControls', {offsetX: -25, offsetY: 45});

Панель нестандартных элементов управления

Метод: setControl
Параметры:
(string) 'geometryControls',
(object){
offsetX: (int) x,
offsetY: (int) y
}
Пример: map.setControl('geometryControls', {offsetX: -25, offsetY: 45});

Использование callback'ов в стандартных и нестандартных элементах управления

Элементы, к которым применимы callback'и при создании через элементы управления и Javascript

- Точка (point)
- Полигон (polygon)
- Полилиния (polyline)
- Прямоугольник (rectangle)
- Инструмент рисования (freeHand)
- Инструмент маршрутизации (route)

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):

- Точка pointCallBack
- Полигон polygonCallBack
- Полилиния polylineCallBack
- Прямоугольник rectangleCallBack
- Инструмент рисования freeHandCallBack
- Событие перемещения по карте mapMoveCallBack
- Событие изменения зума mapZoomCallBack
- 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)

2) route.callBack.onMouseOver // возникает по наведению на вершину
3) route.callBack.onMouseOut // возникает по снятию наведения с вершины
4) route.callBack.onClick // происходит непосредственно после клика
5) route.callBack.onDrag // drag

Каждый callback получает коллекции объектов своего типа на входе при передаче анонимной функции.
Пример:

```
map.pointCallBack = function(e){
        console.log(e); //все объекты на карте
        /*
        ...например, удалить все точки кроме поставленной...
        */
}
```

Включение информационных элементов

Масштаб карты

Метод: setControl
Параметры:

```
(string) 'scale',
(object) {
        topBorder: 'FF0000',
        background: 'FF0000',
        bottomBorder: '00FF00',
        color: '00FF00',
        size: 16,
        offsetX: (int) x1,
        offsetY: (int) y1
}
```

Пример: map.setControl('scale',{topBorder: 'FF0000', background: 'FF0000', bottomBorder: '00FF00', color: '00FF00', size: 16});

Координаты

Метод: setControl
Параметры:
(string) 'coords',
(object){
    color: (hexadecimal) 'FF0000',
    size: (int) 10,
    mouse: (boolean) true,
    offsetX: (int) -10,
    offsetY: (int) -10
}
Пример: map.setControl('coords');

Миникарта

Метод: setControl
Параметры:
(string) 'miniMap',
(object) {
    width: (int) x,
    height: (int) y,
    isClosing: (boolean) true,
    topBorder: (hexadecimal) 'f407bf' ,
    leftBorder: (hexadecimal) 'f407bf' ,
    rightBorder: (hexadecimal) 'f407bf' ,
    bottomBorder: (hexadecimal) 'f407bf' ,
    offsetX: (int) x1,
    offsetY: (int) y1
}
Пример: map.setControl('miniMap',{width: 300, height: 300, isClosing: true});

Лицензионная информация

Не имеет кастомизации.

Отключение информационных элементов

Масштаб карты

Метод: unsetControl
Параметры: (string) 'scale'
Пример: map.unsetControl('scale');

Координаты

Метод: unsetControl
Параметры: (string) 'coords'
Пример: map.unsetControl('coords');

Миникарта

Метод: unsetControl
Параметры: (string) 'miniMap'

Пример: map.unsetControl('miniMap');

Лицензионная информация

Не имеет кастомизации.

Управление стандартными объектами методами Javascript

Перечень стандартных объектов

- Точка (point)
- Полигон (polygon)
- Полилиния (polyline)
- Прямоугольник (rectangle)

Создание стандартных объектов

Точка

Метод: createGeometry
Параметры:

```
(object) {
        type: (string) 'point',
        lon: (float) x,
        lat: (float) y,
        content: (object) {
                title: (string) '',
                body: (string) '',
                width: (int) x,
                height: (int) y,
                isHidden: true, //балун скрыт при старте
                isClosing: (boolean) true, //балун закрываемый (контролы)
                background: (hexadecimal) 'f407bf' | (string) url,
        }
        onClick: (function) console.log, //дополнительные эвенты
        onMouseOver: (function) console.log,
        onMouseOut: (function) console.log
        url: (string) 'http://yoursite.ru/piture.png',
        rotate: (int) 120 //угол поворота изображения из параметра url
        direction:  'back' //необязательный
}
```

Пример:

```
map.createGeometry({
                type: 'point',
                lon: 4165423,
                lat: 7466185,
                content: {
                   title: 'title',
                   body: 'body',
                   width: 200,
                   height: 150,
```

```
            // background: 'f407bf'
            background: 'cloud.png'
        },
        url: '/resources/dynamic_controls/cars/car1.png',
        rotate: true,
        direction: true

    });
```

Полигон

Для задаия полигона достаточно двух точек: первой и третьей. Остальные могут быть переданы как false.

Метод: createGeometry

Параметры:

```
(object) {
    type: (string) 'polygon',
    points: (array) [
        (object) {
        lon: (float) x,
        lat: (float) y,
        url: (string) 'http://yoursite.ru/piture.png',
        direction: (string) 'back',
        offset:
            (object){
                x:(int) x,
                y:(int) y
            }
        },
        (object){...},
        (object){...},
        (object){...}
    ],
    style: (object) {
        color: (hexadecimal) 'ff0000',
        lineHeight: (int) 2,
        backgroundColor: (string) 'none' //отключить фон и сделать карту внутри полигона доступной для инструмента «Перемещение по карте»
    }
}
```

Пример:

```
map.createGeometry({
    type: 'polygon',
    points: [
        {lon: 4170877, lat: 7477148, url: '/resources/dynamic_controls/cars/car1.png', direction: 'back', offset: {x: 1, y: 1}},
        {lon: 4187445, lat: 7461492, url: '/resources/dynamic_controls/cars/car2.png'},
```

```
		{lon: 4208877, lat: 7468028, url:
'/resources/dynamic_controls/cars/car3.png'}
	],
	style: {
		color: 'FF0000',
		lineHeight: 2
	}
});
```

Полилиния

Метод: createGeometry
Параметры:

```
(object) {
	type: (string) 'polyline',
	points: (array) [
		(object) {
		lon: (float) x,
		lat: (float) y,
		url: (string) 'http://yoursite.ru/piture.png',,
		direction: (string) 'back',
		offset:
			(object){
				x:(int) x,
				y:(int) y
			}
		},
		(object){...},
		(object){...}
	],
	style: (object) {
		color: (hexadecimal) 'ff0000',
		lineHeight: (int) 2
	}
}
```

Пример:

```
map.createGeometry({
	type: 'polyline',
	points: [
		{lon: 4170877, lat: 7477148, url:
'/resources/dynamic_controls/cars/car1.png', direction: 'back', offset: {x: 1, y:
1}},
		{lon: 4187445, lat: 7461492, url:
'/resources/dynamic_controls/cars/car2.png'},
		{lon: 4208877, lat: 7468028, url:
'/resources/dynamic_controls/cars/car3.png'}
		],
	style: {
		color: 'FF0000',
		lineHeight: 2
```

```
        }
});
```

Прямоугольник

Метод: createGeometry

Параметры:

```
(object) {
        type: (string) 'polygon',
        points: (array) [
                (object) {
                lon: (float) x,
                lat: (float) y,
                url: (string) 'http://yoursite.ru/piture.png',,
                direction: (string) 'back',
                offset:
                        (object){
                                x:(int) x,
                                y:(int) y
                        }
                },
                (object){...},
                (object){...},
                (object){...}
        ],
        style: (object) {
                color: (hexadecimal) 'ff0000',
                lineHeight: (int) 2
        }
}
```

Пример:

```
map.createGeometry({
        type: 'rectangle',
        points: [
                {lon: 4187445, lat: 7461492,url:
'/resources/dynamic_controls/cars/car2.png'}, //ltc
                {url: '/resources/dynamic_controls/cars/car2.png'}, //rtc
                {lon: 4208877, lat: 7468028, url:
'/resources/dynamic_controls/cars/car2.png'}, //rbc
                {url: ''} //lbc
        ],
        style: {
                color: 'FF0000',
                lineHeight: 2
        }
});
```

Изменение текущих стандартных объектов

Задание стилей по-умолчанию

Метод: setStyle

Параметры:
(object){
type: (string) 'polyline' | 'polygon'|'point' | 'rectangle'|'freehand',
color: (hexadecimal) ‘FFFFF’,
lineHeight: (int) x, //px
backgroundColor: (hexadecimal) ‘FFFFF’,
opacity: (float) 0-1,
url: (string) 'http://site.ru/image.png'
}
Примечание: заданные параметры действительны для следующих объектов
color — 'polyline' | 'polygon'|'rectangle'|'freehand'
lineHeight — 'polyline' | 'polygon'|'rectangle'|'freehand'
backgroundColor — 'polygon'|'rectangle'
opacity — 'polygon'|'rectangle'
url — 'polyline' | 'polygon'|'point' | 'rectangle'|'freehand'
Пример:
```
map.setStyle({type: 'polygon', backgroundColor: 'FF0000', lineHeight: 5});
```

Добавление новых вершин в полигон и полилинию

Метод: addPointToGeometry
Параметры:
(object) {
type: (string) 'polyline' | 'polygon',
id: (int) 0,
lon: (float) 4187445,
lat: (float) 7477148,
url: (string) '/resources/dynamic_controls/cars/car1.png',
direction: (string) 'back', //@deprecated
offset: (object) {
x: (int)1,
y: (int) 1
}
}
Пример:
```
map.addPointToGeometry({
type:'polyline',
id: 0,
lon: 4187445,
lat: 7477148,
url: '/resources/dynamic_controls/cars/car1.png',
direction: 'back',
offset: {
x: 1,
y: 1
}
});
```

Обновление вершины в полигонах и полилиниях

Метод: updatePoint

Параметры:
(object){
	type: (string) 'polygon'|’polyine’,
	id: (int) 0,
	pointId: (int) 2,
	url: (string) ‘’|false,
	direction: (string) 'back', //@deprecated
	offset: (object) {
		x: (int) 1,
		y: (int) 1
	}
}
Пример:
```
map.updatePoint({
	type: 'polygon',
	id: 0,
	pointId: 2,
	url: 'default',
	direction: 'back',
	offset: {
		x: 1,
		y: 1
	}
});
```

Получение текущих стандартных объектов

Получение всех объектов

Метод: getGeometry
Параметры: (string) 'all’
Формат ответа: коллекция типов объектов (см. далее)
Пример: var z = map.getGeometry('all’);

Получение всех объектов по типу

Метод: getGeometry
Параметры:
(object){
	type: (string) 'point' | 'polygon' | 'polyline' | 'rectangle'
}
Формат ответа: коллекция объектов, относящихся к заданному типу (см. далее)
Пример: var z = map.getGeometry({type: 'point'});

Получение объекта по типу и индексу

Метод: getGeometry
Параметры:
(object){
	type: (string) 'point' | 'polygon' | 'polyline' | 'rectangle',
	id: (int) 0
}
Формат ответа:

- Точка: { type: 'point', lon: '', lat: ''}
- Полигон: { type: 'polygon', points:[{ lon: '', lat: ''},{ lon: '', lat: ''}]}
- Прямоугольник: { type: 'rectangle', points:[{ lon: '', lat: ''},{ lon: '', lat: ''}]}
- Полилиния: { type: 'polyline', points:[{ lon: '', lat: ''},{ lon: '', lat: ''}]}

Пример: var z = map.getGeometry({type: 'point', id: 0});

Удаление стандартных объектов

Удаление всех объектов

Метод: deleteGeometry
Параметры: (string) 'all'
Пример: map.deleteGeometry('all');

Удаление всех объектов по типу

Метод: deleteGeometry
Параметры:
(object){
    type: (string) 'point' | 'polygon' | 'polyline' | 'rectangle'
}
Пример: map.deleteGeometry({type: 'point'});

Удаление объекта по типу и индексу

Метод: deleteGeometry
Параметры:
(object){
    type: (string) 'point' | 'polygon' | 'polyline' | 'rectangle',
    id: (int) 0
}
Пример: map.deleteGeometry({type: 'point', id: 0});

Управление нестандартными методами Javascript Flash API

Управление объектом «Карандаш»

Задание стилей объекту отрисовки

Созданному объекту возможно задать цвет и толщину линии в т.ч. в процессе использования инструмента.
Метод: setStyle()
Параметры:
(object) {
    type: (string) 'freeHand',
    color: (hexadecimal) 'FF0000',
    lineHeight: (int) n
}
Пример: map.setStyle({type: 'freeHand', color: 'FF0000', lineHeight: 5});

Возврат объекта отрисовки

Метод возвращает результат использования инструмента карандаш в формате SVG.

Метод: screenShot.getSVGFreeHand()
Параметры: —
Пример: var z = map.screenShot.getSVGFreeHand();

Возврат изображения с отрисованными объектами

Возвращает графический объект в формате base64. В качестве аргумента передается параметр «качество изображения» (числовое значение от 0 до 100).
Метод: screenShot.getScreen
Параметры: (int) [0-100]
Пример: var z = map.screenShot.getScreen(99);

Вывод на печать карты с отрисованными объектами

Отправляет текущую сцену на печать, выводит диалог печати.
Метод: screenShot.printMap()
Параметры: —
Пример: map.screenShot.printMap();

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
}
```

Пример: map.route.get([{lon:x, lat:y},{lon:xx, lat:yy}], method: 'optimal');

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 Flash 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 flash-виджетов в PROGOROD Javascript Flash API.

Метод добавления widget'а

Метод: widget.create
Параметры:

```
(object){
        url: (string),
        lon: (float), //левый нижний угол
        lat: (float)
}
```

Пример:

```
Map.widget.create({
      url: 'http://.../widget.swf',
      lon: 4187521,
      lat: 7473956
 });
```

Метод удаления widget'а

Метод: widget.remove
Параметры: —
Пример: map.widget.remove()

Получение всех widget'ов

Метод: widget()
Параметры: —
Пример: console.log(map.widget());

Метод отправки сообщения (вызов метода виджета)

Метод: widget.set();
Параметры:

```
(object){
      id: (int), //widget id
      func: (string), // название метода widget'a
      params: (object){}  // необязательные параметр; данный объект
передается в метод виджета в качестве параметра
   }
```

Пример:

```
map.widget.set({
```

```
    id: 1,
    func: 'someFunc',
    params: {enruleTheWorld: 'IDDQD'}
});
```

Интерфейсы Flash-приложения доступные в рамках Widgets API

```
/**
 * @return возвращает долготу в проекции Меркатора
 */
public final function get lon():Number

/**
 * @return возвращает широту в проекции Меркатора
 */
public final function get lat():Number

/**
 * @param val устанавливает долготу в проекции Меркатора
 */
public final function set lon(val:Number):void

/**
 * @param val устанавливает широту в проекции Меркатора
 */
public final function set lat(val:Number):void

/**
 * метод возвращает текущие координаты, bbox и зум
 *
 * @return coords, bbox
 */
protected final function getCoords():Object

/**
 * Метод преобразует координаты проекции Меркатора в географические координаты
 * @param lon долготу
 * @param lat широта
 * @return Point
 */
protected final function convertLonLattoXY(lon:Number, lat:Number):Point

/**
 * Метод преобразует географические координаты в координаты проекции Меркатора
 * @param x, локальный X данного объекта
 * @param y, локальный Y данного объекта
 * @return Object формат {lon: ..., lat: ... }
```

```
 */
protected final function convertXYtoLonLat(x:Number, y:Number):Object

/**
 * Метод загрузки контента
 * @param url строка адреса
 * @param call метод возврата результата
 * @param isBinary булевый, необходим в случае если нужно получить
бинарный контент
 *
 */
protected final function loadContent(url:String, call:Function, isBinary:Boolean =
false):*

/**
 * Метод парсера JSON
 * @rapam str строка
 * @return Object
 *
 */
protected function jsonParse(str:String):Object

/**
 * Метод преобразования долготы из проекции Меркатора в
географические координаты
 * @param x долгота в координатах проекции Меркатора
 * @return долгота в геокоординатах
 *
 */
protected final function fromMercX(x:Number):Number

/**
 * Метод преобразования широты из проекции Меркатора в
географические координаты
 * @param x широта в координатах проекции Меркатора
 * @return широта в геокоординатах
 *
 */
protected final function fromMercY(y:Number):Number

/**
 * Метод преобразования долготы из географических координат в
координаты проекции Меркатора
 * @param lon долгота в геокоординатах
 * @return долгота в координатах проекции Меркатора
 *
 */
protected final function mercX(lon:Number):Number
```

```
/**
 * Метод преобразования широты из географических координат в
координаты проекции Меркатора
 * @param lon широта в геокоординатах
 * @return широта в координатах проекции Меркатора
 *
 */
protected final function mercY(lat:Number):Number

/**
 * отправка сообщений в javascript
 *
 * @param func, строковое значение, название функции
 * @param params, параметр для передачи чего либо в js функцию
 * @return может вернуть данные любого типа, но не является
обязательным
 */
protected final function sendJs(func:String, params:* = null):*

/**
 * Метод для зачистки колбеков и мусора;
 * выполняется перед удалением со сцены
 */
public function clean():void
```

Вспомогательные методы

Показ/скрытие текущей геометрии

Методы скрывают текущую геометрию для стандартных и нестандартных объектов. По умолчанию скрывает/показывает все.
Методы: hideGeometry, showGeometry
Параметры:

```
(object) {
        id: (int) x,
        type: (strting) 'point'|'polygon'|'polyline'|'rectangle'|'route'|'freeHand'
}
```

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

Метод: setCenterByBbox
Параметры:

(object) {
lon1: (float) lon1,
lat1: (float) lat1,
lon2: (float) lon2,
lat2: (float) lat2

}
Пример: map.setCenterByBbox({lon1: 4037461.4063877454, lat1: 7649814.135811477, lon2: 4501405.325538139, lat2: 7292741.732982437})

Конвертирование координат

Конвертирование координат из проекции Меркатора в абсолютные
Методы: from_merc_x, from_merc_y
Параметры: (float) lon, (float) lat
Пример: from_merc_x(4187445), from_merc_y(7477148)

Конвертирование абсолютных координат в координаты проекции Меркатора
Методы: lonLatToXY
Параметры:
(object) {
(float) lon,
(float) lat,
(int) zoom
}
Пример: map.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: merc_x, merc_y
Параметры: (float) lon, (float) lat
Пример: merc_x(lon), merc_y(lat)

Перевод времени в абсолютные координаты
Методы: TimeToDegreesLon, TimeToDegreesLat
Параметры: (float) lon, (float) lat
Пример: merc_x(TimeToDegreesLon(03736.0067))

Расчет расстояния между двумя точками
Метод: distVincenty
Параметры:
(float) lon1,
(float) lat1,
(float) lon2,
(float) lat2
Пример: 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://site.ru/reply.php',
        callBack: function(e) {
                console.log(e);
        }
});
```

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: parseLog
Параметры:

```
(object) {
        type: (string) 'nmea ' | 'kml',
        str: (string) '' | url: (string) ''
}
```

Пример:

```
map.parseLog({
        type: 'nmea',
        url: 'http://site.ru/nmea.txt'
});
```

Перечень изменений

0.2.8

- В случае ошибки маршрутизации не отрисовывать стартовую и конечную точку
- Удалять ранее проложенный маршрут в любом случае при перестроении маршрута
- Внесены правки в геокодер в соответствии с новой спецификацией
- Возможность отключать элементы управления на лету
- добавление хинтов для ховера контролов из правой панели (точка, полигон и тп)

- Исправлено перекрытие балунов виджетами
- Не отборажаются до отрисовки перекрывающих контролов основные дефолтные контролы
- Внесена правка в загрузчик, вызывавшая конфликт внутри $(document).ready()
- Возможна передача неописанных параметров в методы геокодера
- URL серивисов за исключением авторизации возможно кастомизировать
- Все методы delete переименованы в remove!
- Метод route.deletePoint переименован в removePoint!
- Исправлен баг с «прыгающими» координатами
- Добавлен метод map.layer.remove()

0.2.7

- Внесены изменения в инструмент «Линейка» в т.ч. исправлен ряд ошибок.
- Добавлен список ошибок маршрутизации в описание callback'а.
- Исправлен баг Safari с нулевой дельтой (mousewheel).
- Карта зациклена по периметру.
- Добавлена возможность задания пользовательских настроек мини-карты.
- Добавлена возможность задания пользовательских настроек шкале масштаба.
- Добавлена возможность задания пользовательских настроек координатам.
- Добавлено изменение офсетов контролов после загрузки.
- Добавлена возможность поддержки дополнительных растровых слоев.
- Добавлено плавное смещение при использовании элемента управления перемещения по карте в рамках bbox.
- Переработанна и улучшена логика основных событий: удалена часть избыточных диспетчеров, класс контроллеров перенесен в статический слой.
- Переработан и улучшен зум: исправлен баг с позиционировнием при зуме, удалена часть таймеров.
- Значительно
- Исправлена работа обфускатора и защиты от декомпилляции.
- Добавлена поддержка цветовых схем в KML.
- Добавлено указание версионных ревизий API.
- Исправлен формат обратного геокодера, удалена переменная GET «radius» (заменена на «r»).
- Дополнена спецификация getSVGFreeHand, setStyle, show/hideGeometry, printMap.
- Изменены параметры (!) установки координат в информационном элементе.

0.2.6

- Включена и дополнена кастомизация контролов.
- Правки маршрутизации. Маршрут перестраивается при начале drag.
- Исправлен route.callBack.onClick
- Добавлен route.callBack.onDrag
- Добавлен метод получения пиксельного расстояния от левого верхнего угла flash-компонента по координатам Меркатора
- Добавлена возможность задания парамеров смещения панелям управления
- Расширен метод показа/скрытия геометрии (по типу и идентификатору)

- Исправлена система виджетов. Добавлена возможность множественного запуска.
- Добавлено описание доступных виджетам интерфейсов WEB-GIS JS API
- Исправлено название метода json_parse
- Исправлена ошибка рендеринга полигонов из KML
- Исправлен инструмент «Линейка». Изменена логика работы.
- Исправлены баги показа стандартных элементов управления.
- Объектный пул запросов. Ускорение загрузки внешних данных на 30-60%.
- Уменьшена скорость масштабирования при использовании стандартных манипуляторов.
- Изменен курсор состояния наведения на карту.
- Введено ограничение «верхнего» уровня (4)
- Исправлено перестроение маршрута при изменении масштаба
- Добавлен метод расчитвающий зум по bbox — setCenterByBbox.

0.2.5

- Правки в маршрутизацию: прокладка маршрута через несколько точек, через JS, колбэки на ховер вершин, промежуточные точки draggable с перестройкой маршрута.
- Система widget'ов — использование сторонних флэш-приложений с ограничееным доступом к внутренней среде карты.
- Удалены экспериментальные балуны. Возвращен прежний тип (<0.2.3).

0.2.4

- Инструмент маршрутизации и JS API

0.2.3

- Добавлен колбэк ready для инициализации карты.
- Изменен дизайн контролов. Временно отключена возможность кастомизации контролов. Удален типовой слайдер.
- Реализован инструмент «линейка».
- Инструент «скриншот» с сохранением на жестки диск
- Реализован новый тип балунов
- Зуммирование даблкликом вместо однокликового центрирования.
- Изменен шаблон лицензионной информации.
- Добавлен эффект кинтеического движения.
- Экспериментальные балуны на замену стандартны точек.

0.2.2

- Кэширование тайлов за пределами рабочего поля на половину его диагонали.
- Зум вынесен под тайлы. Перестроение сетки происходит поверх.
- Callback'и на зум и перемещение карты.
- Фокус зума изменен с половины расстояния между курсором и центром карты на курсор.
- Возможность запуска балунов в свернутом состоянии.
- Расширен метод setStyle.
- Дополнительные callback'и на объекты типа point.
- Дополнительные методы перевода времени в абсолютные координаты.
- Изменен формат входящих данных геокодера.

- Изменен формат получаемых данных группой методов getGeometry.
- Добавлены методы скрытия и показа стандартных и нестандартных объектов.