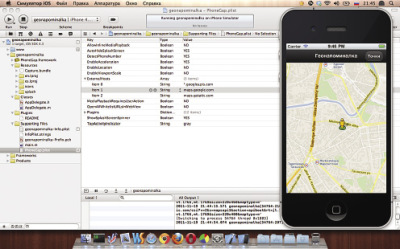
# Как создать программу для смартфона за полчаса

[**Журнал «Хакер»**](https://xakep.ru/)

**Сен 24, 2014**

**17 мин на чтение**

Изучить новый язык и среду разработки — это минимум, что от тебя потребуется, если ты захочешь написать свое первое мобильное приложение. Чтобы с пониманием набросать элементарный todo list для Android или iOS, не передирая пример из книжки, уйдет не меньше пары недель. Но можно не осваивать Objective-C или Java и при этом быстро разрабатывать приложения для смартфонов, если использовать такие технологии, как PhoneGap.

[](https://xakep.ru/wp-content/uploads/2014/09/02.-Zapusk-v-simulyator_opt.jpeg)Запуск приложения в симуляторе iOS

Если ты внимательно изучал нововведения, которые ожидают нас в Windows 8, то, возможно, заметил, что под ней можно будет разрабатывать приложения на HTML5. Идея, на самом деле, не новая — технологии, реализующие тот же подход для мобильных платформ, развиваются семимильными шагами. Одним из таких фреймворков, позволяющим разрабатывать приложения для смартфонов с помощью связки привычных для нас HTML, JavaScript и CSS!, как раз и является PhoneGap. Написанное с его помощью приложение подойдет для всех популярных платформ: iOS, Android, Windows Phone, Blackberry, WebOS, Symbian и Bada. Тебе не нужно будет изучать особенности программирования под каждую платформу (например, Objective-C в случае с iOS), разбираться с различными API и средами разработки. Все, что потребуется для создания кросс-платформенного мобильного приложения, — это знание HTML5 и специального PhoneGap API. При этом на выходе получится не тупая HTML-страница, «обрамленная» в интерфейс приложения, нет! API фреймворка позволяет задействовать практически все возможности телефона, которые используются при разработке с помощью нативных инструментов: доступ к акселерометру, компасу, камере (запись видео и фотосъемка), списку контактов, файловой системе, системе нотификаций (стандартных уведомлений на телефоне), хранилищам и т. д. Наконец, такое приложение может безболезненно обращаться к любому кросс-доменному адресу. Ты можешь воссоздать нативные элементы управления с помощью фреймворков вроде jQuery Mobile или Sencha, и конечная программа будет выглядеть на мобильном телефоне так, как будто она написана на нативном языке (ну или почти так). Лучше всего проиллюстрировать вышесказанное на деле, то есть написать приложение, поэтому предлагаю сразу приступить к практике. Засекай время — на все про все уйдет едва ли больше получаса.

**Другие полезности PhoneGap**

Кроме потрясающей платформы для мобильных приложений, PhoneGap также предоставляет сервис для сборки твоего приложения в «облаке». Под все платформы и в один клик! Сборщик условно бесплатный. Ты можешь зарегистрироваться на сайте PhoneGap Build ([build.phonegap.com](http://build.phonegap.com/)) и получить доступ к сборщику. С его помощью ты вправе собрать неограниченное число приложений с открытым исходным кодом и одно приложение с закрытыми исходниками. Понятно, что если нужно скомпилировать больше закрытых приложений, то придется немного заплатить.

Если тебе не хватает какого-нибудь функционала в «базовой комплектации» PhoneGap, то ты можешь расширить его возможности с помощью плагинов. Существует целый репозиторий ([github.com/phonegap/phonegap-plugins](http://github.com/phonegap/phonegap-plugins)), который включает в себя четыре раздела iPhone, Android, Palm, BlackBerry. Сейчас под iOS написано более 20 плагинов: BarcodeScanner (сканер штрих-кодов), AdPlugin (отображения рекламы iAd), NativeControls (нативные для iOS контролы) и другие.

**Что мы будем создавать**

В качестве целевой платформы возьмем iOS — да-да, деньги лежат в AppStore, и монетизировать свои разработки пока лучше всего там :). Но сразу внесу ясность: все то же самое, без изменений, можно провернуть, скажем, для Android. Долго думал, какой пример рассмотреть, так как писать очередную тулзу для учета списка дел совершенно не хотелось. Поэтому я решил создать приложение под названием «Геонапоминалка», навигационную прогу, назначение которой можно описать одной фразой: «Сообщи мне, когда я снова тут окажусь». В AppStore есть немало утилит, которые позволяют «запомнить» место, где пользователь припарковал машину. Это почти то же самое, только чуть попроще. Ты сможешь указать на карте города точку, задать для нее определенный радиус и запрограммировать сообщение. Когда ты в следующий попадешь в пределы окружности с указанным радиусом, приложение выдаст тебе уведомление, а точка будет удалена. Будем действовать по такому плану: сначала создадим простое веб-приложение, проверим его в браузере, а затем перенесем с помощью PhoneGap на платформу iOS. Очень важно написать в прототипе и протестировать в браузере на компьютере основную часть кода, поскольку отлаживать приложение в телефоне гораздо сложнее. В качестве каркаса мы возьмем JS-фреймворк jQuery c jQuery Mobile ([jquerymobile.com](http://jquerymobile.com/)), а в качестве движка карт — Google Maps v3. Приложение будет состоять из двух страниц: карты и списка точек.

* На карте устанавливается маркер твоего текущего положения. По клику на карте создается точка, к которой привязывается сообщение (вроде «машина рядом»). Точку можно удалить, кликнув на ней. Для перемещения маркера человека по карте используется геонавигационный API.
* На странице со списком точек должна иметься дополнительная кнопка «Удалить все точки», а рядом с каждой точкой — кнопка «Удалить эту точку». Если кликнуть по элементу в списке, соответствующая точка отобразится на карте. Настройки пользователя и список точек будем сохранять в localStorage.

**UI-фреймворки**

jQuery Mobile — это, конечно, не единственный фреймворк для создания мобильного интерфейса. На сайте PhoneGap приведен огромный список библиотек и фреймворков, которые ты можешь использовать ([phonegap.com/tools](http://phonegap.com/tools)): Sencha Touch, Impact, Dojo Mobile, Zepto.js и др.

**Каркас приложения**

Сразу объясняю, зачем мы будем использовать jQuery Mobile. Эта JS-библиотека предоставляет нам уже готовые элементы интерфейса мобильного приложения (максимально приближенные к нативным) для самых разных платформ. Нам ведь надо, чтобы на выходе было именно мобильное приложение, а не страничка из браузера! Так что качаем последнюю версию JQuery Mobile ([jquerymobile.com/download](http://jquerymobile.com/download)) и переносим в рабочую папку первые файлы приложения, которые нам понадобятся:

* images/ (перенеси сюда все изображения из одноименной папки архива jq-mobile);
* index.css;
* index.html;
* index.js;
* jquery.js;
* jquery.mobile.min.css;
* jquery.mobile.min.js.

Нужно сделать ресурсы в основном локальными, чтобы пользователь в будущем не тратил мобильный интернет. Теперь создаем каркас страниц в файле index.html. Приведенный ниже код описывает верхнюю часть страницы с картой, надписью «Геонапоминалка» и кнопкой «Точки».

**Страница с картой**

<div id="index" class="page-map" data-role="page" data-dom-cache="true">

<div data-role="header">

<h1>Геонапоминалка</h1>

<a id="menu-points" class="ui-btn-right" href="#points" data-transition="pop">Точки</a>

</div>

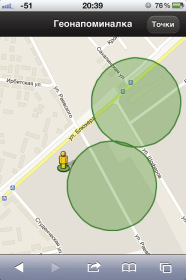
<div data-role="content">

<div id="map-canvas"><!-- Тут будет карта -->

</div>

</div>

</div>

[](https://xakep.ru/wp-content/uploads/2014/09/01.-Stranitsa-s-kartoj_opt.jpeg)Страница с картой, открытая в мобильном браузере. Это еще не iOS приложение.

Атрибут страницы data-dom-cache=»true» необходим для того, чтобы она не выгружалась из памяти. Для кнопки «Точки» используется data-transition=»pop», чтобы страница «Список точек» открывалась с эффектом «Всплытие». Подробнее о том, как устроены страницы jQuery Mobile, можно почитать в хорошем мануале ([bit.ly/vtXX3M](http://bit.ly/vtXX3M)). По аналогии создаем страницу со списком точек:

Страница со списком точек

<div id="points" class="page-pints" data-role="page" data-dom-cache="true">

<div data-role="header">

<!--Удаляет все точки из списка-->

<a id="delete-all" href="#" data-theme="b" data-icon="delete">Удалить все</a>

<h1>Точки</h1>

<!--Кнопка Карта-->

Карта

</div>

<div>

<!--Список точек-->

<ul id="list" data-role="listview" data-inset="true" data-split-icon="delete"></ul>

</div>

</div>

Для кнопки «Карта» тоже пропишем data-transition=»pop», но добавим атрибут data-direction=»reverse», чтобы страница «Карта» открывалась с эффектом «Затухание». Те же атрибуты пропишем в шаблоне точки. Все, наш каркас готов.

**Создание приложения**

Теперь надо отобразить карту, для чего мы возьмем стандартный API Google Maps, который используется миллионами разных сайтов:

var latLng = new gm.LatLng(

this.options.lat, this.options.lng);

this.map = new gm.Map(element, {

zoom: this.options.zoom, // Выбираем начальный зум

center: latLng, // Устанавливаем начальный центр

mapTypeId: gm.MapTypeId.ROADMAP, // Обычная карта

disableDoubleClickZoom: true,

// Отключаем автозум по тапу/двойному клику

disableDefaultUI: true

// Отключаем все элементы интерфейса

});

Здесь Gm — это переменная, ссылающаяся на объект Google Maps. Параметры инициализации я хорошо закомментировал в коде. Следующий шаг — отрисовка маркера человечка на карте:

this.person = new gm.Marker({

map: this.map,

icon: new gm.MarkerImage(PERSON\_SPRITE\_URL,

new gm.Size(48, 48))

});

В качестве PERSON\_SPRITE\_URL используется адрес спрайта человечка из Google-панорам. Его статический адрес —[maps.gstatic.com/mapfiles/cb/mod\_cb\_scout/cb\_scout\_sprite\_api\_003.png](http://maps.gstatic.com/mapfiles/cb/mod_cb_scout/cb_scout_sprite_api_003.png). Пользователь будет добавлять точки, кликая на карте, поэтому, чтобы их отрисовывать, мы будем слушать событие click:

gm.event.addListener(this.map, 'click', function (event) {

self.requestMessage(function (err, message) {

// Метод, возвращающий текст, введенный пользователем

if (err) return;

// Метод добавляет точку в список активных и

// отрисовывает ее на карте

self.addPoint(event.latLng,

self.options.radius, message);

self.updatePointsList(); // Перерисовываем список точек

});

}, false);

Я привожу бОльшую часть кода — остальное ищи на диске. Дальше нам нужно научить приложение перемещать иконку пользователя по карте. В прототипе мы задействуем Geolocation API (тот, который используется в том числе в десктопных браузерах):

if (navigator.geolocation) {

// Проверяем, поддерживает ли браузер геолокацию

function gpsSuccess(pos) {

var lat, lng;

if (pos.coords) {

lat = pos.coords.latitude;

lng = pos.coords.longitude;

} else {

lat = pos.latitude;

lng = pos.longitude;

}

self.movePerson(new gm.LatLng(lat, lng));

// Перемещаем иконку пользователя

}

// Каждые три секунды запрашиваем текущее

// положение пользователя

window.setInterval(function () {

// Запрашиваем текущее положение

navigator.geolocation.getCurrentPosition(gpsSuccess,

$.noop, {

enableHighAccuracy: true,

maximumAge: 300000

});

}, 3000);

}

Метод movePerson с помощью простой процедуры getPointsInBounds() проверяет, не находится ли пользователь в какой-нибудь активной точке. Последний вопрос — где хранить список точек? В HTML5 появилась возможность использовать localStorage, так что не будем ей пренебрегать (предоставляю тебе самостоятельно разобраться с этими участками кода, которые я хорошо закомментировал). Итак, приложение, работающее в браузере, готово!

**Запуск веб-приложения**

Как я уже говорил, отладку в основном необходимо выполнять на компьютере. Самый подходящий браузер для тестирования веб-приложений на компьютере — это Safari или Chrome. После отладки в этих браузерах ты можешь быть уверен в том, что твое приложение не «поедет» в браузере мобильного телефона. Оба этих браузера совместимы с большинством мобильных веб-браузеров, поскольку точно так же, как и они, построены на основе движка WebKit. После устранения всех багов можно переходить к запуску мобильного веб-приложения непосредственно на телефоне. Для этого настрой свой веб-сервер (пусть даже Denwer или XAMPP), чтобы он отдавал созданную страницу, и открой ее уже в браузере мобильного телефона. Приложение должно выглядеть примерно так, как показано на рисунке. Тут важно понимать, что будущее мобильное приложение, собранное для мобильной платформы с помощью PhoneGap, будет выглядеть почти один в один, за исключением того, что на экране не будет отображаться навигационная панель браузера. Если все хорошо, можно приступать к созданию из странички полноценного iOS-приложения. Заметь, что PhoneGap и IDE для мобильной разработки мы до этого момента даже не трогали.

**Подготовка**

Для того чтобы собрать приложение под iOS, тебе нужен компьютер с операционной системой Mac OS 10.6+ (или виртуальная машина на Mac OS 10.6), а также среда разработки Xcode с установленным iOS SDK. Если у тебя не установлен SDK, придется скачать с сайта Apple образ диска, включающий в себя Xcode и iOS SDK ([developer.apple.com/devcenter/ios/index.action](http://developer.apple.com/devcenter/ios/index.action)). Имей в виду, что образ весит около 4 Гб. Кроме этого, тебе понадобится зарегистрироваться на сайте Apple в качестве разработчика (если ты не собираешься публиковать свое приложение в AppStore, то это требование можно обойти). С помощью этого набора можно разрабатывать приложения на нативном для iOS языке Objective-C. Но мы решили пойти обходным путем и воспользоваться PhoneGap, поэтому нам еще нужно установить пакет PhoneGap iOS. Просто скачай архив с офсайта (<https://github.com/callback/phonegap/zipball/1.2.0>), распакуй его и в папке iOS запусти программу установки. Когда установка завершится, в меню проектов Xcode должна появиться иконка PhoneGap. После запуска придется заполнить несколько форм, но уже очень скоро ты увидишь рабочую область IDE с твоим первым приложением. Чтобы проверить, все ли работает, нажми кнопку Run — должен запуститься эмулятор iPhone/iPad с шаблонным приложением PhoneGap. Собранная программа выдаст ошибку с сообщением о том, что index.html не найден, — это нормально. Открой папку, в которой ты сохранил первичные файлы проекта, и найди в ней подпапку www. Перетащи ее в редактор, кликни на иконке приложения в списке слева и в появившемся окне выбери «Create folder references for any added folders». Если запустить программу еще раз, то все должно заработать. Теперь можно скопировать все файлы нашего прототипа в папку www. Пора подпилить наш прототип для работы на смартфоне в обработке PhoneGap.

**Перенос прототипа**

В первую очередь нужно подключить phonegap-1.2.0.js в твой индексный файл. PhoneGap позволяет ограничивать список доступных для посещения хостов. Предлагаю сразу настроить такой «белый список». В меню проекта открой Supporting Files/PhoneGap.plist, найди пункт ExternalHosts и добавь в него следующие хосты, к которым будет обращаться наше приложение (это сервера Google Maps): \*.gstatic.com, \*.googleapis.com, maps.google.com. Если их не указать, программа выдаст предупреждение в консоли и карта не отобразится. Для инициализации веб-версии нашего приложения мы использовали событие DOMReady или хелпер jQuery: $(document).ready(). PhoneGap генерирует событие deviceready, которое говорит о том, что мобильное устройство готово. Предлагаю этим воспользоваться:

document.addEventListener("deviceready", function () {

new Notificator($("#map-canvas")[0]);

// Если у пользователя нет интернета,

// сообщаем ему об этом

if (navigator.network.connection.type ===

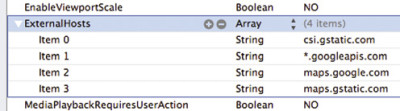
Connection.NONE) {

navigator.notification.alert("Нет интернет-соединения",

$.noop, TITLE);

}

}, false);

[](https://xakep.ru/wp-content/uploads/2014/09/03.-Propisyvaem-extern_opt.jpeg)Прописываем ExternalHosts

Это событие проверяет, есть ли у пользователя хоть какое-нибудь интернет-соединение. Если его нет, выводим соответствующее сообщение. Вместо функции navigator.notification.alert можно использовать более привычную alert, но ее минус в том, что она выглядит менее естественно для мобильного приложения. Сейчас нам хватит и этих знаний, но ты можешь подробнее прочитать о network.connection ([bit.ly/uEyRwz](http://bit.ly/uEyRwz)) и способах нотификации ([bit.ly/tkvzE2](http://bit.ly/tkvzE2)).

Запретим скроллинг:

document.addEventListener("touchmove", function (event) {

event.preventDefault();

}, false);

Затем заменим все вызовы alert и confirm на нативные, которые предоставляет нам PhoneGap:

navigator.notification.confirm('Удалить точку?',

function (button\_id) {

if (button\_id === 1) { // Нажата кнопка OK

self.removePoint(point);

}

}, TITLE);

Последнее, что нам нужно поменять, — это блок кода, перемещающий иконку пользователя по карте. Наш текущий код тоже работает, но работает менее оптимально (перемещает иконку, даже если координаты не изменились) и дает не такие богатые данные, как аналог в PhoneGap:

navigator.geolocation.watchPosition(function (position) {

self.movePerson(new gm.LatLng(

position.coords.latitude,

position.coords.longitude));

}, function (error) {

navigator.notification.alert(

'code: ' + error.code + '\nmessage: ' + error.message,

$.noop,

TITLE

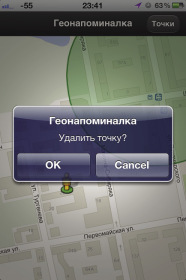
);

}, {

frequency: 3000

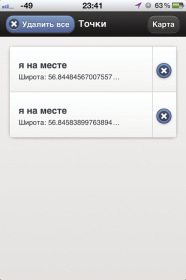
});

Этот код более изящный — он генерирует событие только тогда, когда координаты изменились. Жмем кнопку Run и убеждаемся, что только что созданное нами приложение отлично работает в симуляторе iOS-устройства! Пора приступать к запуску на реальном устройстве.

[](https://xakep.ru/wp-content/uploads/2014/09/04.-Nativnye-uvedomlen_opt.jpeg)Нативные уведомления в iOS

**Запуск на устройстве**

Подсоедини iPhone, iPod или iPad к компьютеру, на котором запущен Xcode. Программа определит новое устройство и попросит разрешения использовать его для разработки. Нет смысла ей отказывать :). Повторю еще раз: чтобы запустить написанное приложение на iOS, необходимо быть авторизированным разработчиком iOS (другими словами, быть подписанным на iOS Developer Program). Этим придется заморочиться только в случае разработки приложений для продукции Apple, с другими платформами (Android, Windows Phone) все намного проще. У тех, кто обучается в вузе, есть шанс получить доступ к программе бесплатно благодаря каким-нибудь льготам. Все остальные должны платить $99 в год для участия в программе. Apple выдает сертификат, которым ты сможешь подписывать свой код. Подписанное приложение разрешается запускать на iOS и распространять в App Store. Если ты не студент, а $99 для невинных экспериментов тебе пока жалко, то есть и другой способ — обмануть систему. Ты можешь создать самоподписанный сертификат для верификации кода и запустить мобильную программу на джейлбрейкнутом iOS-устройстве (не буду на этом останавливаться, потому что все максимально подробно расписано в этой статье: [bit.ly/tD6xAf](http://bit.ly/tD6xAf)). Так или иначе, ты вскоре увидишь работающее приложение на экране своего мобильного телефона. Останавливай секундомер. Сколько времени у тебя на это ушло?

[](https://xakep.ru/wp-content/uploads/2014/09/05.-Rabotayushhee-prilozhe_opt.jpeg)Работающее PhoneGap-приложение

**Другие платформы**

Кроме PhoneGap, существуют и другие платформы, позволяющие создавать мобильные приложения без использования нативных языков. Перечислим наиболее крутых игроков.

**Appcelerator Titanium (**[**www.appcelerator.com**](http://www.appcelerator.com/)**).**

Titanium умеет собирать приложения в первую очередь под Android и iPhone, но в нем также заявлена поддержка BlackBerry. Кроме самого фреймворка, проект предоставляет набор нативных виджетов и IDE. Ты можешь разрабатывать приложения на Titanium бесплатно, однако за поддержку и дополнительные модули придется заплатить (от $49 в месяц). Цена некоторых сторонних модулей доходит до $120 за год. Разработчики Appcelerator Titanium утверждают, что на основе их фреймфорка написано более 25 тысяч приложений. Исходный код проекта распространяется под лицензией Apache 2.

**Corona SDK (**[**www.anscamobile.com/corona**](http://www.anscamobile.com/corona)**).**

Эта технология поддерживает основные платформы — iOS и Android. Фреймворк нацелен в основном на разработку игр. Еще бы, ведь разработчики заявляют о высококачественной оптимизации на OpenGL. Бесплатной версии у платформы нет, а цена довольно-таки кусачая: $199 в год за лицензию для одной платформы и $349 в год для iOS и Android. Corona предлагает свою IDE и эмуляторы устройств. Приложения под Corona пишут на языке, похожем на JavaScript.

**Заключение**

Мы создали простое мобильное веб-приложение и в несколько простых шагов портировали его на платформу iOS с помощью PhoneGap. Мы не написали ни строчки кода на Objective-C, но получили программу приличного качества, потратив минимум времени на перенос и изучение API PhoneGap. Если ты предпочитаешь другую платформу, например Android или Windows Mobile 7, то ты так же легко, без каких-либо изменений под эти платформы, сможешь собрать наше приложение (для каждой из них есть хороший вводный мануал и видеоурок:[phonegap.com/start](http://phonegap.com/start)). Чтобы убедиться в состоятельности платформы, можно посмотреть на уже готовые приложения на PhoneGap, которые разработчики технологии собрали в специальной галерее ([phonegap.com/apps](http://phonegap.com/apps)). По факту PhoneGap — это идеальная платформа для создания как минимум прототипа будущего приложения. Ее главными преимуществами являются быстрота и минимум затратат, чем активно пользуются стартапы, которые во всех отношениях ограничены в ресурсах. Если приложение попрет, а внутренности на HTML+JS тебя по какой-то причине перестанут устраивать, всегда можно будет портировать приложение на нативный язык. Не могу не сказать, что PhoneGap изначально разрабатывался компанией Nitobi как открытый проект (репозиторий располагается на GitHub:[github.com/phonegap](http://github.com/phonegap)). Исходники и дальше будут оставаться открытым, хотя в октябре прошлого года компанию Nitobi купил Adobe. Нужно ли говорить, какие перспективы появляются у проекта при поддержке в лице такого гиганта?