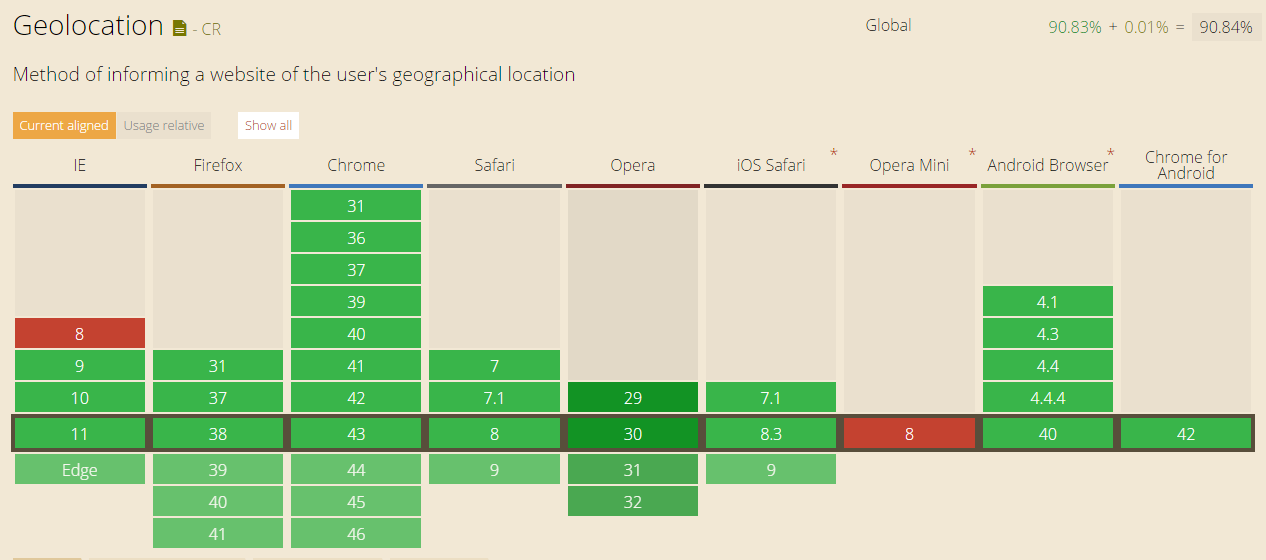
|  |
| --- |
| EPAM Systems |
| Geolocation API |
| HTML5 Geolocation API |

|  |
| --- |
| Oksana Spolnyk  [Date] |

# Определение

Геолокация это способ выяснить, где вы находитесь в мире и дополнительно обмен этой информацией с людьми, которым вы доверяете. Существует несколько способов выяснить ваше нахождение — по IP-адресу, подключению к беспроводной сети, через антенну к которой подключается ваш сотовый телефон во время разговора или GPS-оборудование вычисляющее широту и долготу на основе информации посылаемой спутниками в небе.

## Совместимость с браузерами



## Покажи мне код

API геолокации строится вокруг нового свойства глобального объекта navigator — navigator.geolocation. Простейшее использование API выглядит следующим образом.

**function** get\_location() {  
 navigator.geolocation.getCurrentPosition(show\_map);  
}

Здесь нет проверки, обработки ошибок и дополнительных опций. Ваше веб-приложение должно включать, по крайней мере, первые два пункта. Чтобы определить поддержку API геолокации, вы можете использовать Modernizr.

**function** get\_location() {  
 **if** (Modernizr.geolocation) {  
 navigator.geolocation.getCurrentPosition(show\_map);  
 } **else** {  
 // Нет встроенной поддержки  
 }  
}

Браузер никогда не заставит вас раскрыть свое текущее расположение удаленному серверу. Способ взаимодействия с пользователем отличается от браузера к браузеру. В Mozilla Firefox вызов функции getCurrentPosition() выведет информационную панель в верхней части окна браузера.

Функция, которая обращается к другой функции - show\_map и ждет от нее отзыв. Вызов getCurrentPosition() возвращает результат сразу, но это не означает, что у вас есть доступ к местоположению пользователя. В первый раз вы точно имеете информацию о местоположении в функции, которая выглядит следующим образом:

**function** show\_map(position) {  
 **var** latitude = position.coords.latitude;  
 **var** longitude = position.coords.longitude;  
 // Посмотрим карту или сделаем что-нибудь интересное!  
}

Функция вызывается с одним параметром, это объект с двумя свойствами: coords и timestamp.

*timestamp* содержит дату и время, когда было вычислено место. Так как это происходит асинхронно, вы не можете знать заранее, когда это произойдет. Пользователю потребуется некоторое время, чтобы прочитать информационную панель и согласиться указать свое местоположение. Устройства на базе GPS могут занять еще больше время для подключения к спутникам.

Объект *coords* содержит свойства, такие как latitude и longitude, которые определяют физическое местоположение пользователя в мире.

|  |  |  |
| --- | --- | --- |
| coords.latitude | double | В градусах |
| coords.longitude | double | В градусах |
| coords.altitude | double или null | В метрах (высота) |
| coords.accuracy | double | В метрах (погрешность) |
| coords.altitudeAccuracy | double или null | В метрах |
| coords.heading | double или null | Градусы по часовой стрелке от севера |
| coords.speed | double или null | В метрах в секунду |
| timestamp | DOMTimeStamp | Как у объекта Date() |

Только три свойства будут гарантированно (coords.latitude, coords.longitude и coords.accuracy). Остальные, возможно, вернут null в зависимости от возможностей вашего устройства и сервера, с которым вы общаетесь. Свойства heading и speed при возможности вычисляются на основе предыдущей позиции пользователя.

## Обработка ошибок

Геолокация сложна и что-нибудь может пойти не так. Пользователь всегда прав. Второй аргумент функции getCurrentPosition(): это функция обработки ошибок.

navigator.geolocation.getCurrentPosition(show\_map, handle\_error);  
**function** handle\_error(err) {  
 **if** (err.code == 1) {  
// пользователь сказал нет!  
 }  
}

Если что-то пойдет не так, ваша функции для ошибок будет вызвана с объектом PositionError. Свойство code может принимать следующие значения:

* PERMISSION\_DENIED (1) если пользователь нажал на кнопку «Не сообщать» или иным образом запретил доступ к своему местоположению;
* POSITION\_UNAVAILABLE (2) если сеть не работает или нет связи со спутниками;
* TIMEOUT (3) если сеть работает, но вычисление положения занимает слишком много времени. Насколько это «слишком много»? Я покажу вам как это определить в следующем разделе;
* UNKNOWN\_ERROR (0) если что-нибудь еще пойдет не так.

Некоторые популярные мобильные устройства вроде телефонов iPhone и Android поддерживают два метода определения, где вы находитесь. Первый метод вычисляет ваше положение на основе относительной близости от различных сотовых вышек управляемых оператором телефона. Этот метод быстр и не требует никакого специального GPS-оборудования, но дает только общее представление о том, где вы находитесь. В зависимости от того, сколько сотовых вышек находится в вашем районе, «общее представление» может быть небольшим как городской квартал или намного больше (километры в любом направлении).

Функция getCurrentPosition() имеет необязательный третий аргумент, объект *PositionOptions*. Все свойства являются необязательными, вы можете задать любой, все или ни одного.

|  |  |  |  |
| --- | --- | --- | --- |
| enableHighAccuracy | Булевой | false | Если true, то медленнее |
| timeout | long | Нет | В миллисекундах |
| maximumAge | long | 0 | В миллисекундах |

Свойство **enableHighAccuracy** делает именно то, как и звучит. Если оно установлено как true, устройство поддерживает свойство и пользователь соглашается указать свое местоположение, то устройство попытается обеспечить высокую точность. Оба телефона iPhone и Android имеют отдельные разрешения для низкой и высокой точности позиционирования, поэтому не исключено, что вызов getCurrentPosition() с enableHighAccuracy:true не получится, а вызов enableHighAccuracy:false удастся.

Свойство **timeout** это количество миллисекунд, которое веб-приложение будет ожидать для получения положения. Таймер не начинает отсчет, пока пользователь не даст разрешение на попытку вычислить его положение. Он следит не за пользователем, а за сетью.

Свойство maximumAge позволяет устройству отвечать немедленно из закэшированной позиции. Например, вы обращаетесь к getCurrentPosition() в первый раз, пользователь соглашается, и ваша функция успешно возвращает положение вычисленное ровно в 10:00. Ровно через минуту, в 10:01 вы называете getCurrentPosition() снова установив свойство maximumAge в 75000.

navigator.geolocation.getCurrentPosition(  
 success\_callback, error\_callback, {maximumAge: 75000});

Вы тем самым говорите, что вам необязательно знать текущее положение пользователя. Вы были бы рады знать, где он был 75 секунд назад (75000 миллисекунд). Устройство знает, где пользователь был 60 секунд назад (60000 миллисекунд), так как оно рассчитало место после первого вызова getCurrentPosition(). Поэтому устройство не будет пересчитывать текущее местоположение пользователя. Оно вернет ту же информацию, что и в первый раз: ту же широту и долготу, ту же точность и то же время (10:00).

Прежде чем вы запросите местоположение пользователя, вы должны подумать о том, какая точность вам требуется, и установить соответствующее enableHighAccuracy. Если вам необходимо найти место еще раз, вы должны подумать о том, какая старая информация может пригодиться и установить соответствующее maximumAge. Если требуется искать локацию постоянно, то getCurrentPosition() не для вас, необходимо перейти на watchPosition().

Фукнция **watchPosition()** имеет ту же структуру что и getCurrentPosition(). Она содержит две функции, одна требуется для успеха, вторая для ошибок, также может иметь необязательный объект PositionOptions, он имеет те же свойства, о которых вы узнали ранее. Разница в том, что ваша функция будет вызываться каждый раз при смене местоположения пользователя. Нет необходимости постоянного опроса позиции, устройство само определит оптимальный интервал опроса и вызовет функцию при изменении положения пользователя. Вы можете использовать эту возможность для обновления видимым маркеров на карте, предоставления инструкции, куда идти дальше или чего-нибудь еще. Это полностью зависит от вас.

Сама функция **watchPosition()** возвращает число, которое следует где-то хранить. Если вы хотите остановить слежение за положением пользователя, то можете вызвать метод clearWatch(), передать ему это число и устройство остановит вызов функции. Если вы когда-либо использовали функции setInterval() и clearInterval() в JavaScript, то это работает точно так же.

## Что насчет IE?

Internet Explorer не поддерживает API геолокации, который я только что описал. Открытый плагин Gears от Google работает на Windows, Mac, Linux, Windows Mobile, и Android и предоставляет возможности для старых браузеров. Одна из таких возможностей это Gears API геолокации. Это не то же самое, что API геолокации от W3C, но служит для той же цели.

Многие старые мобильные платформы имеют API для геолокации ориентированные на свои устройства. BlackBerry, Nokia, Palm и OMTP BONDI предоставляют свои собственные API для геолокации. Конечно, вся их работа отличается от Gears, который, в свою очередь, работает иначе, чем API геолокации от W3C.

## geo.js для спасения

geo.js это JavaScript-библиотека с открытым исходным кодом распространяемая по лицензии MIT. Она сглаживает различия между API от W3C, Gears и поставщиками мобильных платформ. Чтобы ее использовать, вам надо добавить два элемента <script> в нижнюю часть страницы. Технически, вы можете вставить их в любое место, но скрипты в <head> могут замедлить загрузку страницы, так что не делайте этого.

Первый скрипт geo\_init.js инициализирует Gears, если тот установлен. Второй скрипт geo.js.

<!DOCTYPE html>  
<html>  
<head>  
 <meta charset="utf-8">  
 <title>Погружение в HTML5</title>  
</head>  
<body>  
...  
<script src="gears\_init.js"></script>  
<script src="geo.js"></script>  
</body>  
</html>

Теперь вы готовы использовать любой установленный API геолокации.

**if** (geo\_position\_js.init()) {  
 geo\_position\_js.getCurrentPosition(geo\_success, geo\_error);  
}

Функция getCurrentPosition() переключает ваш браузер на запрос разрешения указать ваше местоположение. Если геолокация предоставляется через Gears, появится всплывающее диалоговое окно с запросом, что вы доверяете сайту использовать Gears. Если ваш браузер изначально поддерживает API геолокации, диалог будет выглядеть несколько иначе. Например, Firefox 3.5 изначально поддерживает API геолокации. Если попытаться определить ваше местоположение в Firefox 3.5, то появится информационная панель в верхней части страницы с запросом указать ваше местоположение с этого сайта.

Функция getCurrentPosition() принимает две вызываемых функции в качестве аргументов. Если getCurrentPosition() была успешна в поиске вашего местоположение, то есть вы дали разрешение, API сработает подобно магии — произойдет вызов функции, переданный в качестве первого аргумента. В этом примере функция успеха называется geo\_success.

## Живой пример

Вот живой пример с использованием geo.js. Он пытается получить ваше местоположение и отобразить карту на месте рисунка.

При загрузке страницы вызывается geo\_position\_js.init() для определения геолокации доступной через любой интерфейс, который поддерживает geo.js. Если это так, то появится ссылка, на которую можно нажать, чтобы посмотреть ваше местоположение. Щелчок по ссылке вызывает функцию lookup\_location() показанную здесь.

**function** lookup\_location() {  
 geo\_position\_js.getCurrentPosition(show\_map, show\_map\_error);  
}

Если вы дали согласие отследить ваше местоположение и службы в состоянии это сделать, geo.js вызывает первую функцию show\_map() с единственным аргументом loc. Объект loc содержит свойство coords в котором хранится широта, долгота и точность (в этом примере точность не используется). Остальное в функции show\_map() используется для создания встроенной карты через Google Maps API.

**function** show\_map(loc) {  
 $("#geo-wrapper").css({'width':'320px','height':'350px'});  
 **var** map = **new** GMap2(document.getElementById("geo-wrapper"));  
 **var** center = **new** GLatLng(loc.coords.latitude, loc.coords.longitude);  
 map.setCenter(center, 14);  
 map.addControl(**new** GSmallMapControl());  
 map.addControl(**new** GMapTypeControl());  
 map.addOverlay(**new** GMarker(center, {draggable: **false**, title: "Вы здесь"})); }

Если geo.js не может определить ваше местоположение, он вызывает вторую функцию show\_map\_error().

# Источники

http://htmlbook.ru/html5/geolocation