

在如今的互联网时代,无论对于互联网企业还是互联网的使用者,地理位置都是一个十分重要的信息。团购网站可以根据用户的地理位置信息为用户提供最近的娱乐地点,天气预报网站也可以根据用户的地理位置信息提供其所在城市未来两周的天气状况等,许许多多的网站都在使用着用户地理位置信息,尽管地理定位信息可以说是用户十分重要的个人隐私,也有很多用户不愿意暴露自己的地理位置相关信息。但在另一方面,用户的地理位置信息确实让互联网相关企业能够更有针对性地为用户提供服务。不过还是要提醒读者,在使用不安全的网站,或是有较大公开性的社交平台时,在决定是否提供地理位置信息前,请三思而后行。

不同浏览器对于地理定位(Geolocation) API 的提供有所差异,尤其是使用 PC 浏览器的读者,在试着编写并运行本章代码的过程中,有很大可能无法得出相应的结果。在此建议,本章地理定位 API 的代码,请部署在服务器端,如有条件尽量使用具有 GPS 定位系统的移动设备(如智能手机、智能平板)来访问服务器相关页面,这样有更大的可能性成功运行。对于使用 PC 进行访问,可能会出现地理位置获取失败的情况,建议更换不同浏览器进行尝试。

## 5.1 浏览器如何获取地理信息

可以通过 HTML5 地理定位功能从浏览器获取地理位置信息,那么浏览器又是如何知道地理位置信息的呢?

浏览器大致会通过以下几种方式来确定地理位置信息,并根据实际情况调整获取方式,通常在使用浏览器定位服务时无法知道浏览器到底是使用哪种方式获取的地理位置信息。

### 1. GPS

全球定位系统(Global Positioning System,GPS)是由美国国防部研制建立的一种具有全方位、全天候、全时段、高精度的卫星导航系统,能为全球用户提供低成本、高精度的三维位置、速度和精确定时等导航信息。目前 GPS 被广泛用于导航、防盗等设备中,为大量企业和个人提供定位服务。

对于 HTML5 的学习者来说,需要知道的是,目前市面上大量的智能手机、智能平板、导航设备等都配备有 GPS 系统,能够获取到较高精度的地理定位信息。而 GPS 定位系统在笔记本电脑上的配置情况远不如移动设备。

## 2. IP 地址

IP 地址是互联网协议地址(Internet Protocol Address)的简写。它为互联网的每一个网络和每一台主机分配一个逻辑地址,大部分普通民用网络 IP 地址由互联网服务提供商(Internet Service Provider,ISP)统一提供和管理。

HTML5 地理定位相关的 API 有时会通过 IP 地址来判断用户的地理位置信息,很多从 PC 所获取的地理定位信息,很大可能是通过 IP 地址来确定当地互联网提供商服务器的位置来确定的。尽管这样的定位方式不够精确,但是这种方式还是确定了用户所在城镇等信息,可以为天气预报网站、网络购物平台所使用。

## 3. 手机信号基站

尽管市面上大量的手机都配备了 GPS 服务,但是 GPS 服务耗时较长,有时不够稳定精确。手机信号基站也是一个获取地理定位信息的选择。采用这种方法,可以通过计算手机与相邻各基站间的方位信息,来确定当前设备的地理坐标信息。

## 4. Wi-Fi

Wi-Fi 定位通过周围一个或多个的 Wi-Fi 热点信息,再通过一些距离计算可以获取到地理位置信息。这种方法在室内使用且周围有多个 Wi-Fi 热点时精度相对较高且速度较快,它也是浏览器获取地理定位信息的方式之一。

## 5.2 获取访客经纬度信息

经度与纬度共同组成了地理坐标系统,这样就可以唯一确定地球上的一点。通过地理定位相关 API,可以获取访客的地理坐标信息。请看如下实例,其在浏览器中的展示效果如图 5.1 所示。

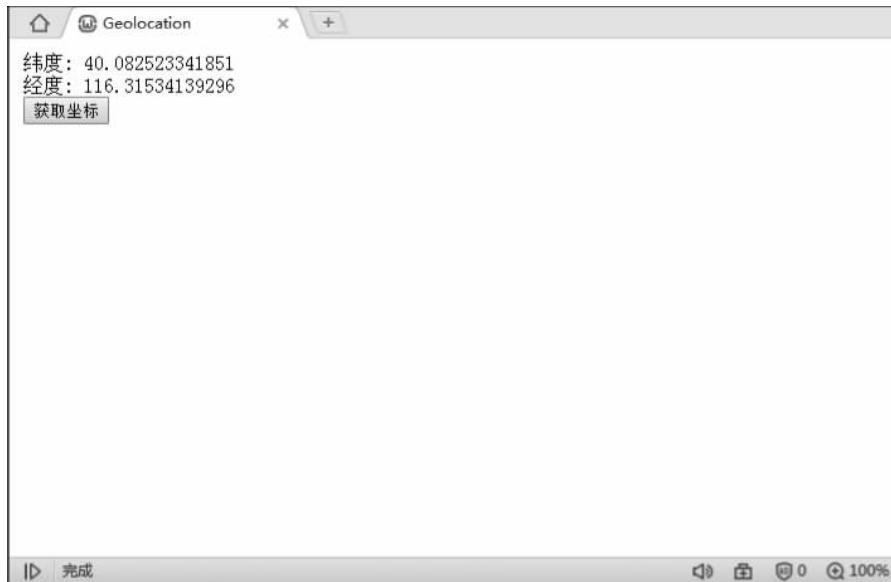


图 5.1 经纬度信息

文件名：获取访客经纬度信息.html

```
<!DOCTYPE HTML>
<html lang = "en-US">
<head>
    <meta charset = "UTF-8">
    <title>Geolocation</title>
</head>
<body>
    <div id = "myLocation">
    </div>
    <button id = "getLoc" onclick = "getMyLocation()">获取坐标</button>
    <script>
        function getMyLocation(){
            if (navigator.geolocation){
                navigator.geolocation.getCurrentPosition(displayLocation);
            }else{
                alert("您的浏览器不支持地理定位");
            }
        }
        function displayLocation(position){
            var latitude = position.coords.latitude;
            var longitude = position.coords.longitude;

            var div = document.getElementById("myLocation");
            div.innerHTML = "纬度：" + latitude + "<br>" + "经度：" + longitude;
        }
    </script>
</body>
</html>
```

上面这段代码首先在 HTML 页面中添加了一个按钮并为其指定单击事件以及事件处理函数 getMyLocation()。当运行这个页面并单击按钮后，就会执行 getMyLocation() 函数。这个函数首先进行了一次判断，判断当前浏览器环境下是否能够提供地理定位服务，如果没有则提醒用户。这个判断语句的条件就是 navigator 对象是否存在 geolocation 的相关属性或方法。其中 navigator 对象包含浏览器相关信息及其所运行环境信息，除了地理位置信息外，还包括浏览器名称、版本、浏览器所运行环境的操作系统以及操作系统语言等信息。相信很多读者在下载一些软件时，会发现单击下载直接获取的资源就是当前使用的操作系统版本；有时我们在国外上网，浏览一些本应是英文的网站时，发现网站的语言竟然是中文简体。这类服务很大程度上是依赖于 navigator 对象的相关信息来实现的。

在确定了当前浏览器支持地理定位服务后，就可以通过 navigator 对象下的 geolocation 的 getCurrentPosition() 方法来获取当前坐标。这个方法所需传入的是一个函数。通常获取地理定位的方法会耗费一段时间，尽管只是几百毫秒，但相比于页面其他代码执行的时间而言可以说是十分漫长的了，所以 getCurrentPosition() 方法应当是异步的，即执行获取地理位置信息的相关程序不应影响主页面的运行，所以需要提供一个函数，当获取地理位置信息成功后来运行这个函数，通过该函数将所获取的信息反馈给用户。在这里传入的就是

displayLocation() 函数。

在 displayLocation() 函数中,传入的参数即是 getCurrentPostion() 方法所返回的位置信息 position 对象。通过这个对象来读取相应的信息。在 displayLocation() 函数中,我们访问 position 对象的 coords(地理坐标) 属性下的 latitude(纬度) 和 longitude(经度),这样也就获取了访客的经纬度信息。如果需要,也可以通过 position 对象的 timestamp 属性来获取地理定位服务的相应时间戳,即返回地理定位信息后的时间,应注意这个时间戳信息的格式并不是常见可读的年、月、日、小时、分钟、秒,而是相对格林威治标准时间 1970 年 1 月 1 日 00 : 00 : 00.000(格里高利历,公历) 的以毫秒为单位的偏移量,即常见于计算机和电子设备的时间,它需要进行一定的换算才可阅读并应用。

当然对于 position 对象的 coords 属性,其包括的不只是经纬度信息,也包括海拔 altitude、方向 heading、速度 speed 等。不过其他信息的获取依赖于浏览器种类版本以及浏览器获取地理信息的方式。对于配备有 GPS 服务的智能手机或是智能平板来说,获取海拔、方向、速度等信息的可能性更大。

对于许多互联网企业来说,获取访客的地理位置信息之后,就可以大致判断出访客的所在国家、所在城市以及与访客相关的距离等信息,这为下一步提供差异化、具体化的服务带来了可能。当然作为用户,个人地理位置信息属于个人隐私,对于很多人来说处于“被监视”的状态会有一些不适。所以在通过浏览器试图获取地理位置信息坐标前,浏览器会主动询问用户是否共享个人的坐标位置。用户可以选择同意或拒绝,也可以通过浏览器调整与个人隐私相关的设置,对于某些或是所有的网站始终提供或不提供获取地理位置信息的权限,这些选择的权利始终在用户一边。

## 5.3 错误处理

相信不少读者在测试上述代码时或多或少遇到了一些困难。地理定位相关的 API 最好的测试方法是将代码部署在服务器上,然后通过配备有 GPS 的手机或平板访问,才更有可能获取更为准确的地理位置信息。通过 PC 的浏览器访问服务器的相关代码,仍有较大可能无法完成获取。这其中的原因一方面是各个浏览器种类和版本对于同一获取地理位置信息的具体实现有些区别,在实际应用上,效果可能有些差异;另一方面,地理定位的获取即便对于那些带有 GPS 服务的设备来说,也有很多不稳定性,有时费时较长、有时不够精确。当然,如果访客拒绝提供自己的相关地理位置信息,获取地理信息的过程也不能顺利完成。所以对于这些常见的问题,错误处理是必不可少的,在出现可能的问题导致获取过程失败时,需要将错误及时告知用户和相关服务程序。下面通过具体实例来学习地理定位的错误处理方法,其在浏览器中的展示效果如图 5.2 和图 5.3 所示。

文件名: 错误处理方法.html

```
<!DOCTYPE HTML>
<html lang = "en - US">
<head>
<meta charset = "UTF - 8">
<title>Geolocation</title>
```



图 5.2 错误处理—不可用错误



图 5.3 错误处理—用户拒绝

```

</head>
<body>
    <div id = "myLocation">
    </div>
    <button id = "getLoc" onclick = "getMyLocation()">获取坐标</button>
    <script>
        function getMyLocation(){
            if (navigator.geolocation){
                navigator.geolocation.getCurrentPosition(displayLocation,displayError);
            }else{
                alert("您的浏览器不支持地理定位");
            }
        }
        function displayLocation(position){
            var latitude = position.coords.latitude;
            var longitude = position.coords.longitude;

            var div = document.getElementById("myLocation");
            div.innerHTML = "纬度：" + latitude + "<br>" + "经度：" + longitude;
        }
        function displayError(error){
            var errorTypes = {
                0:"未知错误",
                1:"地理定位请求被用户拒绝",
                2:"当前地理定位不可用",
                3:"请求超时"
            };
            var errorMessage = errorTypes[error.code];
            //在以下两种情况下，错误信息还包括一些具体内容
            if (error.code == 0 || error.code == 2){
                errorMessage = errorMessage + "<br>" + error.message;
            }
            var div = document.getElementById("myLocation");
            div.innerHTML = errorMessage;
        }
    </script>
</body>
</html>

```

上述实例在运行过程中产生了两个错误，分别是地理位置不可用以及用户拒绝提供地理位置信息。还是之前的 navigator 对象下 geolocation 的 getCurrentPosition()方法，不同的是这次传入了两个函数，来处理这个方法的返回对象。getCurrentPosition()方法有两个可选的完成函数：第一个默认是成功之后的完成函数，另一个可选的就是出现错误时的完成函数。在实例中，指定完成函数为 displayError()并编写函数的代码。

```

function displayError(error){
    var errorTypes = {
        0:"未知错误",
        1:"地理定位请求被用户拒绝",
    }
}

```

```

        2:"当前地理定位不可用",
        3:"请求超时"
    };
    var errorMessage = errorTypes[error.code];
    //在以下两种情况下,错误信息还包括一些具体内容
    if (error.code == 0 || error.code == 2){
        errorMessage = errorMessage + "<br>" + error.message;
    }
    var div = document.getElementById("myLocation");
    div.innerHTML = errorMessage;
}

```

在这个 `displayError()` 函数中,传入的参数即是 `getCurrentPosition()` 方法在执行出错时所返回的 `error` 对象。`error` 对象包含属性 `code`,`error.code` 的标准即是 `errorTypes` 内所指定的几种错误,在这里将错误信息中文写入 `errorTypes` 数组。如果不写这一步也可以直接通过 `error.message` 来直接获取相应的错误信息。应注意,当错误代码为 0 或 2 时, `error.message` 会包含一些具体可能的错误,在这里需要在所构造的错误信息字符串的基础上连接 `error.message` 来构成一个完备的错误信息。

## 5.4 地理定位选项

除了上述的两个成功获取定位与错误处理的完成函数外,`getCurrentPosition()` 方法还可以传入其他的地理定位选项,分别是 `enableHighAccuracy`、`timeout` 以及 `maximumAge`。可以在上述实例中添加类似如下的代码片段来具体使用。

```

options = {
    enableHighAccuracy: true,
    maximumAge: 60000,
    timeout: 30000
};
navigator.geolocation.getCurrentPosition(displayLocation,displayError,options);

```

这段代码向 `getCurrentPosition()` 方法中传入了自行定义的 `options` 对象。它由 `enableHighAccuracy`、`timeout` 以及 `maximumAge` 三个属性值组成。

`enableHighAccuracy` 属性可选 `true` 或 `false`,表示是否要求高精度的地理定位检测。这个设定的具体执行效果依赖于发起请求的设备是否具有 GPS 定位系统服务,否则意义不大。这个属性的默认值是 `false`,即不执行高精度定位。在没有具体应用场景的情况下,也尽量避免将其设定为 `true` 来支持高精度定位。高精度的定位首先费时;其次对于移动设备来说,还会消耗一些额外的电量。

`maximumAge` 属性设定了缓存地理定位信息的时间长度,以毫秒为单位。在上面的代码中,`maximumAge: 60000` 代表了一次地理定位数据从获取到失效的持续时间为 60 000ms,即 60s。这样就避免了在实际应用中反复调用费时费电的地理定位服务,但在另一方面,当进行类似导航这样的追踪服务时,`maximumAge` 的属性值设定越高,追踪的效果

就越差,可能就不能满足实际应用场景的需要。默认情况下,maximumAge 的属性值为 0,即不进行坐标的缓存,每一次的调用意味着重新向浏览器请求地理信息服务,而不是直接使用不久前刚刚获取的数据。

timeout 属性设定了等待地理定位信息服务的时间上限,以毫秒为单位。在上面的代码中,timeout: 30000 即代表了从用户同意共享地理位置信息开始计时,经过 300 00ms,即 30s 后,若还未获取地理位置信息就会提前结束这一过程,返回一个 error 对象,且对应的 error.code 为 3,即请求超时。默认情况下一般是没有请求时间上限的,具体情况可能因浏览器种类和版本而异。

## 5.5 地理定位追踪

之前我们使用的是 navigator 下 geolocation 对象的 getCurrentLocation() 方法。这个方法在每一次调用之后都会返回发出请求的用户的地理位置信息。这种方法大部分时候被应用于获取用户某一时刻的固定地理位置信息。而在很多的应用场景中,需要持续地对用户的位置进行跟踪,比如导航的过程。这时若想完成对目标的追踪,就不仅仅需要一次地理位置信息获取,而是需要不断地获取地理定位信息。geolocation 对象的 watchPosition() 就可以注册监听器,并在设备的地理位置发生变化后被调用,这个方法具体传入的参数和 getCurrentLocation() 一样,需要传入一个获取定位成功的回调函数,可选参数分别是之前所提到的错误处理函数以及其他三个地理定位选项。大致使用方法如下。

```
options = {
  enableHighAccuracy: true,
  maximumAge: 60000,
  timeout: 30000
};
if (navigator.geolocation){
  id = navigator.geolocation.watchPosition(displayLocation, displayError, options);
} else{
  alert("您的浏览器不支持地理定位");
}
```

可以看出,watchPosition() 方法与 getCurrentLocation 方法的使用几乎相同。所不同的首先是二者的调用次数,watchPosition() 在设备地理位置发生改变时会被调用,而 getCurrentLocation() 方法则是一次性的。然后的一大区别是,watchPosition() 方法注册了一个监听器,用来监听地理定位是否变化。当然这个监听器会占用一些额外的运算资源,所以在一次监测行为结束后,要关闭监听器,并释放相应资源。在上述代码中我们发现,在调用 watchPosition() 方法时返回了一个 id 指代当前的监听行为。所以当我们要取消这次地理定位追踪时,只需调用 clearWatch() 方法并传入这个 id 值,即可完成取消。

```
navigator.geolocation.clearWatch(id);
```

## 5.6 习题

1. 分别站在服务提供者和用户的角度思考地理位置信息所带来的好处与隐患。
2. 解释地理坐标系的定义与应用。
3. 浏览器可能通过怎样的方式来获取用户的地理位置信息？
4. 任意选择地球上的一个城市，通过网络查询到这座城市的坐标，构建 Web 小应用来计算这个城市与当前位置的距离。
5. 自行搜索地图厂商所提供的地图组件，在浏览器中展示基于地图的当前位置，并构建简单的应用。