第 8 章

介绍地理定位和数据离线支持

您正在首次访问一个城市，并在寻找宾馆附近的观光点。但是您不知道要去哪里，要寻找哪些地点。因此，如果有应用程序可帮助您对此类位置进行定位并提供反馈，那将是非常有用的。这可通过实现 Geolocation API 来做到。

此外，您可能希望用户在未连接到因特网的时候也能够访问网页。

本质章介绍 Geolocation API。它进一步讨论如何在网站上实现离线支持。

目标

在本章中，您将学习：

- 实现地理定位
- 实现离线支持


NIIT
training
china
.com

实现地理定位

ShopHere 是总部在 Ohio 的大型零售商店。该商店出售服装、配饰和鞋。此外，它还销售家具和电器，例如冰箱、空调、笔记本电脑和移动设备。作为吸引新客户和建立稳定客户群的业务策略的一部分，商店提供折扣商品和特价商品。此外，商店的官网具有一项功能，允许用户分享他们当前的位置，将根据该位置对他们进行定向以到达最近的商店。

此类功能通过使用 Geolocation API 合并到网站中。启用了 Geolocation API 的网站可对用户的当前地理位置进行定位，显示该位置周围的景点，或者将用户从当前位置引导到目标位置。

实现 Geolocation API

Geolocation API 使网站能够检索用户的当前地理位置。此 API 允许您创建应用程序来告诉用户如何从当前位置到达目标位置。此类应用程序的示例是针对目标位置提供方向的地图应用程序。

用户位置不仅是通过代码或浏览器检索的。Geolocation API 使用某些功能，例如 Global Positioning System (GPS)、设备的 IP 地址、最近的移动电话发射塔以及用户设备中的用户输入来检索用户位置。

使用 Geolocation API 检索到的用户位置几乎是准确的，具体取决于用于检索位置的来源的类型。例如，IP 地址提供与用户位置接近的位置，而 GPS 能够给出更准确的结果。

识别用户位置有时可能损害用户隐私。因此，用户位置只有在用户同意的情况下才可用。将显示提示，询问用户是否希望分享当前位置并指定收集此数据的原因。此外，应该指定将在何处分享该数据。

地理定位对于用于移动设备的应用程序最为有益。这是因为当用户旅行时，Geolocation API 不断更新其位置。对于桌面应用程序，用户位置保持不变，只能设置一次。

Geolocation API 提供以下方法来确定用户位置：

- `getCurrentPosition()`
- `watchPosition()`

getCurrentPosition()

`getCurrentPosition()` 方法用于检索用户的当前地理位置。位置是作为一组坐标而检索的。

`getCurrentPosition()` 方法的语法是：

```
getCurrentPosition(CallbackFunction, ErrorHandler, Options);
```

在上述语法中：

- `CallbackFunction`：这是开发人员定义的用于检索当前位置的函数。
- `ErrorHandler`：这是检索用户位置时发生错误时调用的函数名称。这是可选参数。

- Options：此可选参数指定用于检索用户地理位置信息的一组选项，例如检索位置信息时的超时。

getCurrentPosition() 方法调用 callbackFunction，后者将 position 对象取作参数。此对象将用户的当前地理位置指定为一组地理坐标。position 对象在成功检索位置时返回两个属性 coords 和 timestamp。timestamp 属性返回检索位置的日期和时间。coords 属性返回 latitude 和 longitude 之类的各种特性。下表中描述了这些特性。

特性	*描述*
coords.latitude	*将纬度指定为十进制数。*
coords.longitude	*将经度指定为十进制数。*
coords.accuracy	*指定位置的精度。*
coords.altitude	*指定高于海平面的高度（米）。*
coords.altitudeAccuracy	*指定高度的精度。*

coords 属性的特性

使用 getCurrentPosition() 方法检索用户位置前，需要检查浏览器是否支持地理定位功能。为此，可使用 navigator 对象的 geolocation 属性。navigator 对象包含浏览器相关信息。思考以下代码段，它检查浏览器是否支持地理定位功能：

```
if(navigator.geolocation)
var geo=navigator.geolocation
else
{
alert("Your browser does not support geolocation")
}
```

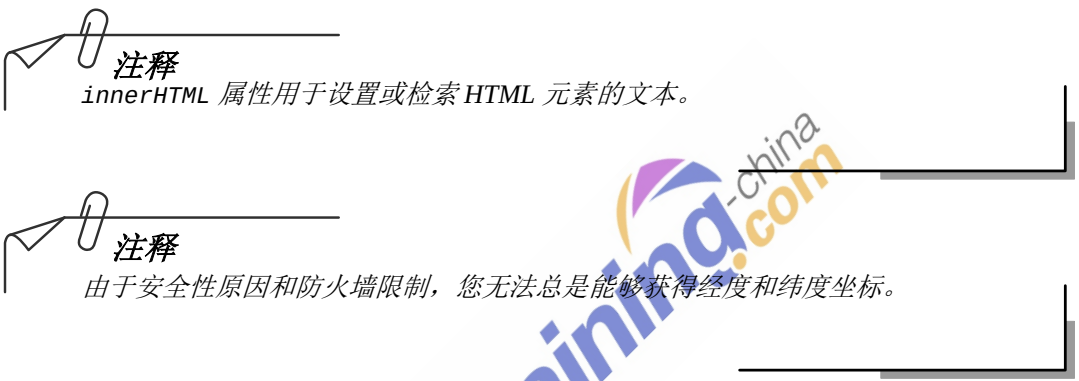
上述代码段检查浏览器是否支持地理定位功能。如果不支持，将显示相关消息。

思考以下代码段，它使用 getCurrentPosition() 方法检索用户位置：

```
<!DOCTYPE HTML>
<HTML>
<BODY>
<P ID="button">Click here to know your location coordinates:</P>
<BUTTON onclick="getLocation()">Get Location</Button>
<SCRIPT>
var geo=document.getElementById("button");
function getLocation()
{
if (navigator.geolocation)
{
navigator.geolocation.getCurrentPosition(getPosition);
}
else{geo.innerHTML="Geolocation is not supported by this browser.";}
```

```
}
function getPosition(position)
{
geo.innerHTML="Latitude:" + position.coords.latitude +
"<BR>Longitude:" + position.coords.longitude;
}
</SCRIPT>
</BODY>
</HTML>
```

上述代码段创建 **Get Location** 按钮，单击该按钮时将调用 `getLocation()` 方法。此方法首先检查浏览器是否支持地理定位功能。如果支持此功能，将调用 `getCurrentPosition()` 方法。此方法调用 `getPosition()` 方法，后者以经度和纬度值的形式检索用户位置。



watchPosition()

`watchPosition()` 方法返回用户的当前位置并在用户移动时不断更新位置。它最常用于 GPS 之类的设备，在用户旅行时通知他们的当前位置。`watchPosition()` 方法取的参数与 `getCurrentPosition()` 方法相同，并返回相同对象。

思考以下代码段，它使用 `watchPosition()` 方法检索用户位置：

```
function getLocation()
{
if (navigator.geolocation)
{
navigator.geolocation.watchPosition(getPosition);
}
else{geo.innerHTML="Geolocation is not supported by this browser.";}
}
function getPosition(position)
{
geo.innerHTML="Latitude:" + position.coords.latitude +
"<br>Longitude:" + position.coords.longitude;
}
```

上述代码段通过使用 `watchPosition()` 方法检索用户在旅行时的位置。

也可使用 `clearWatch()` 方法停止跟踪用户位置。此方法停止对 `watchPosition()` 方法的调用。

处理错误

有时，用户不允许访问其位置。这可能引起您网站上发生错误。当用户在移动设备上检查当前位置而该设备不在覆盖区域或者网络连接超时的情况下，也会发生该错误。作为网站开发人员，您需确保网站能够有效处理错误。为此，需要思考错误的所有可能原因，这些错误是应用程序中可能发生的并且可以通过有效的错误处理技术处理的。

要处理启用地理定位的网站上发生的错误，可使用 `getCurrentPosition()` 方法。此方法的第二个参数是错误处理程序函数。此函数可以处理以下错误代码：

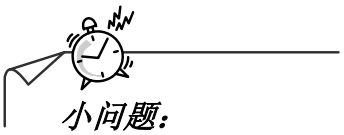
- `PERMISSION_DENIED`：指定用户拒绝了分享位置的请求。
- `POSITION_UNAVAILABLE`：指定无法检索用户的当前位置。
- `TIMEOUT`：指定给出的用于检索用户位置的时间超出了最大限制。
- `UNKNOWN_ERROR`：指定发生了未知或未定义的错误。

错误处理程序函数接受 `PositionError` 对象作为参数。此对象有两个属性 `code` 和 `message`。code 指定错误代码，message 指定发生错误时要显示的消息。

思考以下代码段，它处理检索用户位置时的错误：

```
function showError(PositionError)
{
switch(PositionError.code)
{
case PositionError.PERMISSION_DENIED:
x.innerHTML="User denied the request for tracking the location"
break;
case PositionError.POSITION_UNAVAILABLE:
x.innerHTML="User's location is not available"
break;
case PositionError.TIMEOUT:
x.innerHTML="The request to retrieve user's location is timed out"
break;
case PositionError.UNKNOWN_ERROR:
x.innerHTML="An unknown error occurred"
break;
}
}
```

上述代码段创建函数 `showError()`，它将在检索用户位置时发生错误的情况下调用。此函数接受 `switch` 构造中定义的错误代码。例如，如果用户拒绝访问其位置，网页上将显示错误 `User denied the request for tracking the location`。



小问题：

coords 属性的以下哪个特性用于指定高于海平面的高度（米）？

1. `coords.latitude`
2. `coords.accuracy`
3. `coords.altitude`
4. `coords.altitudeAccuracy`

答案：

3.`coords.altitude`

活动 8.1：实现地理定位

实现离线支持

有时，您应该已经注意到：当您在网站冲浪时，显示 Browser could not open the Web page 之类的消息。这是因为与因特网的连接已丢失。在此类情况下，您没有办法访问正在冲浪的网站。然而，如果您希望在未连接到因特网的情况下也能够对网站有某种程度的访问，可使用离线支持功能。例如，ShopHere 的主管们经常需要到不同城市出差以推广其产品。他们需要在产品推广旅行期间访问公司网站。然而，网站并不是每次都可访问，特别是在网络不可用的偏远地区。在此类情况下，可使用 HTML 的离线支持功能来确保主管们在没有因特网连接的情况下也能访问网站。此外，使用离线支持功能可避免加载网站所需的正常网络请求。

在 HTML 具有离线支持功能前，启用离线存储的网站是使用 cookie 和插件创建的。cookie 是用户浏览网站时从网站发出并存储在用户浏览器上的一小段数据。当用户重新访问同一网站时，将检索 cookie 中存储的数据。不只使用 Cookie 进行离线存储，因为它们存储的数据量有限。此外，Cookie 使网络活动变慢，因为在服务器上对它们进行传入和传出。

通过 HTML 的离线支持功能克服了这些限制。离线支持功能提供以下好处：

- 确保网站可用，甚至在用户未连接到网络时也是如此
- 降低服务器上的网络负载

可通过以下方式使网站离线工作：

- 实现客户端存储
- 实现应用程序高速缓存

实现客户端存储

客户端存储是指将数据本地存储在用户浏览器上的过程。又称为 Web 存储。使用客户端存储所存储的数据只有在请求（并非针对每个服务器请求）时才被检索。而且，可实现客户端存储来存储大量数据，
客户端存储以键/值对的形式存储数据。

客户端存储可使用以下对象实现：

- `localStorage`
- `sessionStorage`

localStorage

`localStorage` 对象允许存储不具有任何到期日的数据。这表示使用 `localStorage` 对象存储的数据在关闭浏览器后不会被删除，将在重新打开浏览器时可用。`localStorage` 对象只以字符串的形式存储数据。使用此对象，高速缓存的数据可在所有浏览器窗口中访问。

思考以下代码段，它使用 `localStorage` 存储用户信息：

```
<DIV ID="str"></DIV>
<SCRIPT>
if(typeof(Storage)!=="undefined")
```

```
{
localStorage.name="John";
document.getElementByIDgetElementById("str").innerHTML="Name:" +
localStorage.name;
}
else
{
document.getElementByIDgetElementById("str").innerHTML="Your browser does not
support local storage";
}
</SCRIPT>
```

在上述代码段中，typeof() 方法首先检查浏览器是否支持 Web 存储。如果支持，localStorage 对象将用户名本地存储在浏览器中，方法是使用键 name，该键被赋值 John。

以后，存储在 localStorage 对象中的信息将通过该键进行检索，然后赋给 ID 为 str 的 <DIV> 标记的 innerHTML 属性。如果浏览器不支持 Web 存储，文本 Your browser does not support local storage 将显示在 <DIV> 标记中。

sessionStorage

sessionStorage 对象用于仅针对一个会话存储数据。这表示数据将在用户关闭浏览器时被删除。通过 sessionStorage 存储的数据仅限于为其创建的浏览器使用。思考以下示例：您需要对用户在当前会话中单击按钮的次数进行计数。为此，您可使用 sessionStorage 对象，如以下代码段所示：

```
<HEAD>
<SCRIPT>
function clickCounter()
{
if(typeof(Storage)!=="undefined")
{
if (sessionStorage.clickcount)
{
sessionStorage.clickcount=Number(sessionStorage.clickcount)+1;
}
else
{
sessionStorage.clickcount=1;
}
document.getElementById("btn").innerHTML="You have clicked it" +
sessionStorage.clickcount + " time(s) in this session.";
}
else
{
document.getElementById("btn").innerHTML="Your browser does not support Web
storage";
}
}
</SCRIPT>
</HEAD>
<BODY>
<p><BUTTON onclick="clickCounter()" type="button">Click Here</BUTTON></P>
<DIV id="btn"></DIV>
<P>Click the button to see the increase in counter.</P>
```

```
<P>Close the browser and try again, the counter will be reset.</P>
</BODY>
```

上述代码段创建将在单击按钮时调用的 `clickCounter()` 函数。

在函数主体中，`typeof()` 方法首先检查浏览器是否支持 Web 存储。如果支持，`clickCount` 键将与 `sessionStorage` 对象一起使用以存储用户在当前会话中单击按钮的次数。每次用户单击按钮时，键 `clickCount` 的值将增 1。由于赋给 `sessionStorage` 对象的键值总是字符串，因此它将通过 `Number()` 方法转换为数字，然后再增 1。

存储在 `sessionStorage` 对象中的最近值是通过键 `clickCount` 检索的并赋给 `ID` 为 `btn` 的 `<DIV>` 标记的 `innerHTML` 属性，即显示给用户的属性。

实现应用程序高速缓存

浏览网站时，必须面对这样的情况：网络连接丢失，您单击浏览器的“后退”按钮以查看上一页面。然而，由于您未连接到因特网，并且浏览器未相应地缓存页面，因此您无法查看该页面。要在此类情况下查看上一页面，需要重新连接到因特网。为了解决此类问题，开发网站时，可指定浏览器应该缓存的文件，这样在离线的情况下刷新页面时，将能够查看页面。缓存网站的这一过程称为应用程序高速缓存。

应用程序高速缓存提供以下优点：

- **离线浏览**：指定在用户未连接到因特网的情况下也可查看网站。
- **高速**：指定当用户请求已位于缓存中的网页时，将从缓存（而不是服务器）进行检索。因此，由于不访问网络，加载网页将更快，因为不需要与服务器进行连接。
- **服务器负载降低**：指定：缓存的网页将总是可通过缓存使用，除非浏览器检测到服务器上已更新了缓存清单或者用户清除了浏览器缓存。那样的话，浏览器将下载新版本的清单以及清单中列出的资源。因此，发送给服务器的请求数将更少，这样就降低了服务器上的负载。

要实现应用程序高速缓存，需要创建文本文件 manifest。此文件包含需要缓存以供无网络连接时使用的资源列表。清单文件还包含不应缓存的文件或页面的列表。需要以 .appcache 扩展名保存该清单文件。

清单文件分为以下部分：

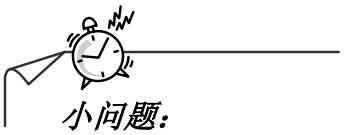
- **Cache**：列出需要在首次下载后缓存的文件。
- **Network**：列出不应该缓存的文件。
- **Fallback**：指定当用户尝试获取未缓存文件时将执行的任务。

要在 HTML 文档中引用清单文件，需要在开始 `<HTML>` 标记内添加 `manifest` 属性，如以下代码段所示：

```
<HTML manifest="HotelFacilities.appcache">
```

上述代码段为 `HotelFacilities.appcache` 清单文件启用应用程序高速缓存。

一旦在本地机器上缓存了应用程序，浏览器将显示缓存的文件，即使它已在服务器上更改或更新也是如此。因此，为了确保浏览器更新缓存，需要修改清单文件。



小问题：

清单文件的以下哪个部分列出不应该缓存的文件？

1. `Cache`
2. `Network`
3. `Fallback`
4. `FestCache`

答案：

2.`Network`

活动 8.21：实现离线支持

练习问题

1. 以下哪个函数用于检索用户的当前地理位置？
   a. `watchPosition()`
   b. `getPosition()`
   c. `getCurrentPosition()`
   d. `getPosition()`

2. 以下哪个对象包含浏览器相关信息？
   a. `navigator`
   b. `geolocation`
   c. `position`
   d. `error`

3. 以下哪个错误代码指定不能检索用户当前位置？
   a. `PERMISSION_DENIED`
   b. `POSITION_UNAVAILABLE`
   c. `TIMEOUT`
   d. `UNKNOWN_ERROR`

4. 以下哪个对象允许仅以字符串形式存储数据？
   a. `localStorage`
   b. `sessionStorage`
   c. `currentStorage`
   d. `Storage`

5. 以下哪个扩展名用于保存清单文件？
   a. `.cache`
   b. `.appcache`
   c. `.appmanifest`
   d. `.manifestcache`

小结

在本章中，您学习了：

- Geolocation API 使网站能够检索用户的当前地理位置。
- Geolocation API 提供以下方法来确定用户位置：
  - `getCurrentPosition()`
  - `watchPosition()`
- 要处理启用地理定位的网站上发生的错误，可使用 `getCurrentPosition()` 方法。此方法的第二个参数是错误处理程序函数。
- 离线支持功能提供以下好处：
  - 确保网站可用，甚至在用户未连接到网络时也是如此
  - 降低服务器上的网络负载
- 客户端存储是指将数据本地存储在用户浏览器上的过程。
- 客户端存储可使用以下对象实现：
  - `localStorage`
  - `sessionStorage`
- `localStorage` 对象允许存储不具有任何到期日的数据。
- `sessionStorage` 对象用于仅针对一个会话存储数据。这表示数据将在用户关闭浏览器时被删除。
- 要实现应用程序高速缓存，需要创建文本文件 manifest。
- 需要以 .appcache 扩展名保存该清单文件。