

第五章 Geolocation API

内容安排

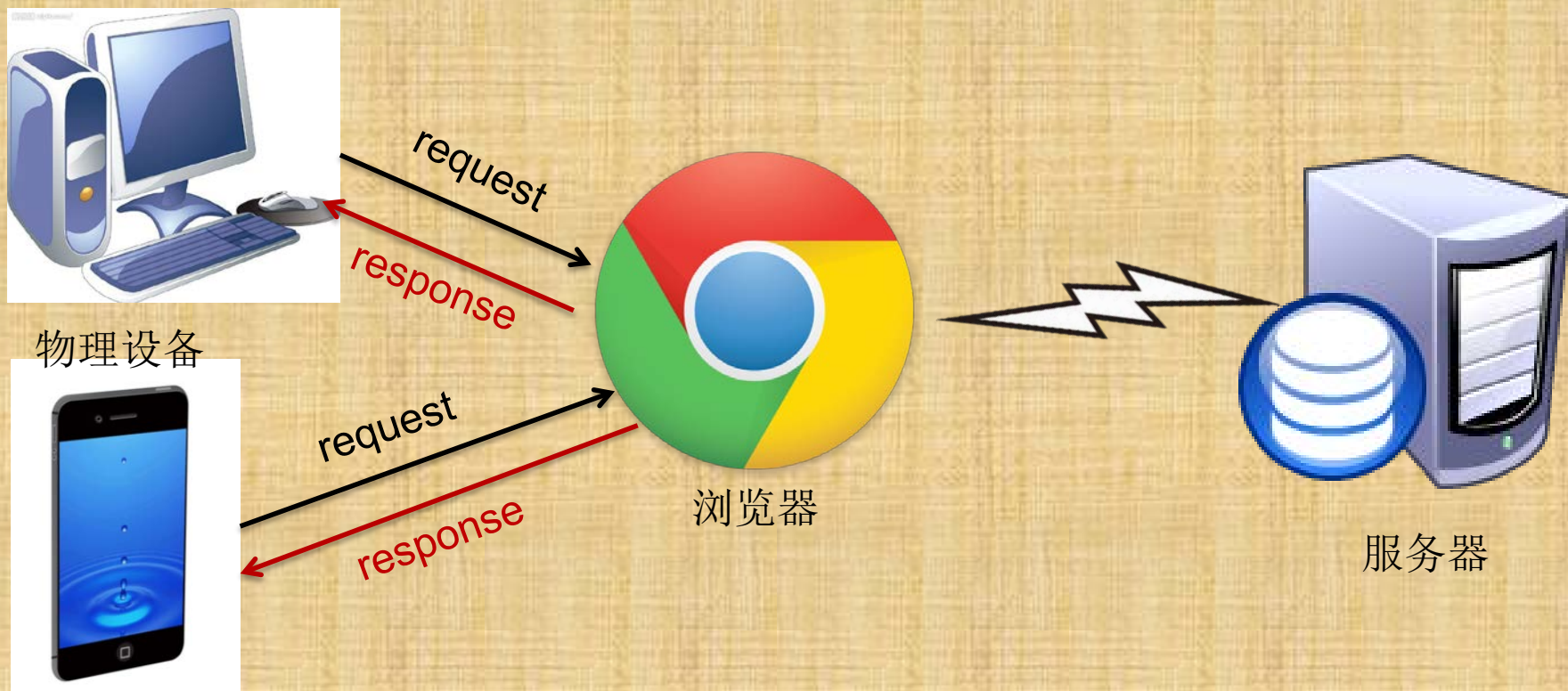
- 5.1 位置信息
- 5.2 HTML5 Geolocation的浏览器支持情况
- 5.3 隐私
- 5.4 使用HTML5 Geolocation API
- 5.5 进阶功能
- 5.6 课后思考
- 5.7 小结

5.1 位置信息

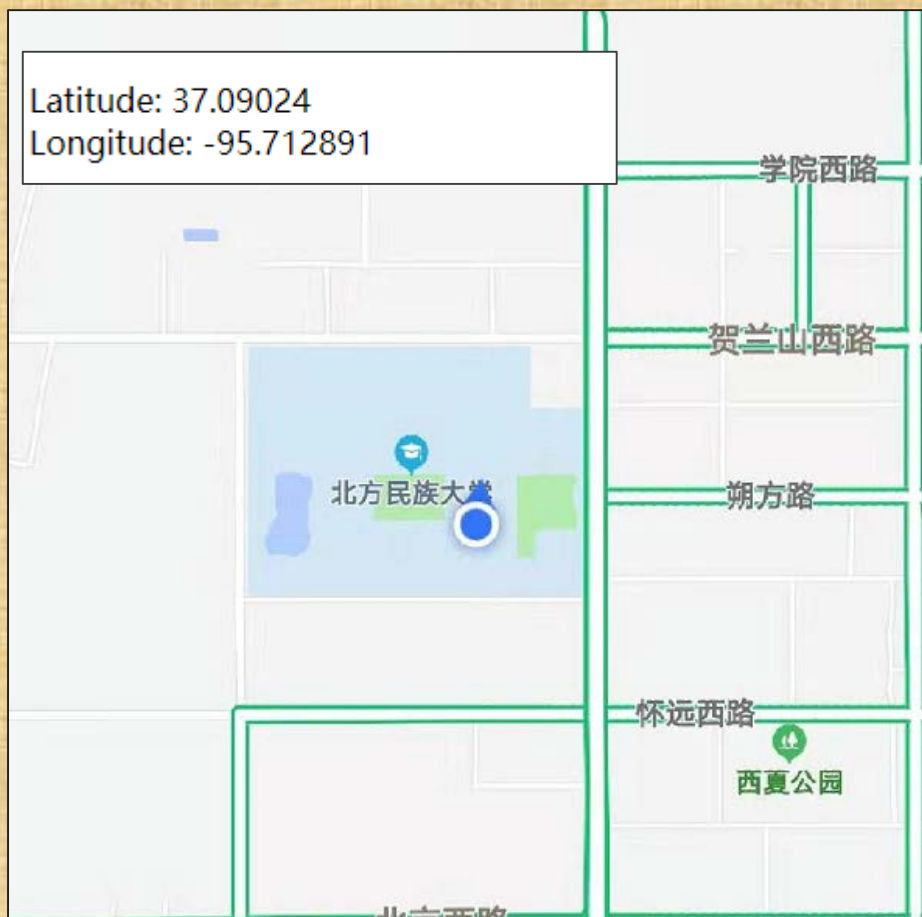
- 维度和经度坐标
- 位置信息从和而来
- **IP**地址地理定位数据
- **GPS**地理定位数据
- **Wi-Fi**地理定位数据
- 手机地理定位数据
- 用户自定义的地理定位数据

□ 维度和经度坐标

HTML5 Geolocation API的使用方法相当简单：请求位置信息→用户同意→位置信息返回给用户。



位置信息主要有一对纬度和经度坐标组成。如以北方民族大学为例：纬度：37.09024 经度：-95.712891



经纬度有两种格式：十进制格式；DMS（角度）格式（例如： $37^{\circ} 5'24.864''$ ）。

Geolocation为十进制格式。

除了坐标外，还提供坐标的准确度、行驶方向、速度、海拔等。

□ 位置信息从何而来

HTML5 Geolocation API不指定设备使用何种技术定位，仅用于检索位置信息的API，因此，得到的数据具备一定程度的准确性。

设备可使用的数据源：

- 1、IP地址；
- 2、GPS之类的定位设备；
- 3、RFID、Wi-Fi和蓝牙设备的MAC地址；
- 4、移动通信的2G、3G、4G等设备的ID；
- 5、用户自定义数据。

□ IP地址地理定位数据

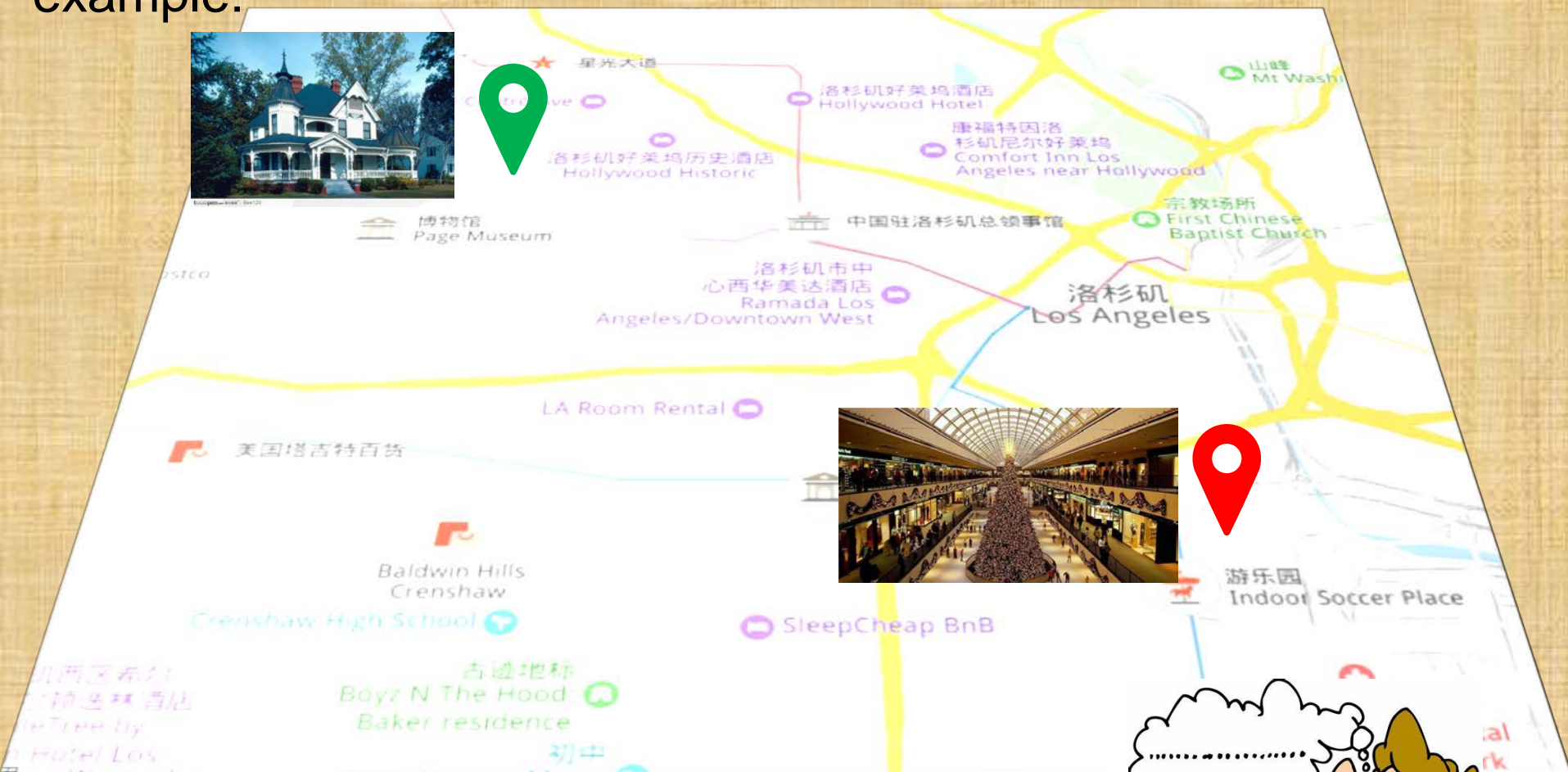
基于IP地址的地理定位的实现方式：自动查找用户的IP地址，然后检索其注册的物理地址。

但如果用户的IP地址时ISP（服务提供商）提供的，其位置往往就由服务提供商的物理地址决定，该地址可能距离用户数千米，因此容易造成定位不准确。

基于IP地址的地理位置数据的优缺点

优点	缺点
任何地方都可用	不精确
在服务器端处理	运算代价大

example:



定位正确（家）

定位错误（ISP：互联网服务供应商）



□ 卫星地理定位数据

卫星定位系统，主要有：

- 1、美国GPS；
- 2、俄罗斯GLONASS；
- 3、欧盟GALILEO系统；
- 4、中国北斗系统；
- 5、日本的QZSS准天顶卫星系统。

□ 卫星地理定位数据

卫星定位时通过收集运行在地球周围的多个卫星的信号实现的，如果没有被遮挡则定位精准，但是，其定位时间可能比较长，因此不适合应用与需要快速响应的应用程序里。

表 基于**GPS**的地理定位数据的优缺点

优点	缺点
很精确	定位时间长
	室内效果不好
	需要额外硬件设备
	用户耗电量大

□ Wi-Fi地理定位数据

基于手机的地理定位信息时通过用户到一些基站的三角距离确定的。通常基于Wi-Fi和基于GPS的地理定位信息结合使用。

表 基于手机的地理定位数据的优缺点

优点	缺点
相当精确	在基站较少的偏远读取效果不好
可在室内使用 可以简单、快捷定位	



□ 用户自定义的地理定位数据

用户输入地址、邮编等信息，然后根据这些信息进行定位。前提时这些地址和邮编已经有位置信息。

表 用户自定义的地理定位数据的优缺点

优点	缺点
用户可以获得比程序定位更准确的位置数据	可能会不准确，特别是当用户位置变更之后
允许地理定位服务的结果作为备用位置信息	
用户自行输入可能比自动检测更快	


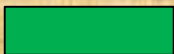
5.2 HTML5 Geolocation的浏览器支持情况

 不支持
 支持

IE	Edge [*]	Firefox	Chrome	Safari	Opera
			49		
			1 61		
			1 63		
			1 69		
			1 70		
			1 71		
			1 72	5.1	
		52	1 73	10.1	
8		1 60	1 74	11.1	
9	17	1 67	1 75	12	1 60
11	18	1 68	1 76	12.1	1 62

主流PC端浏览器支持情况

5.2 HTML5 Geolocation的浏览器支持情况

 不支持
 支持

iOS Safari *	Opera Mini *	Android Browser *	Opera Mobile *	Chrome for Android	Firefox for Android	IE Mobile	UC Browser for Android	QQ Browser	Baidu Browser
9.3									
¹ 10.3									
¹ 11.2									
¹ 11.4		4.3							
¹ 12.1		4.4.4							
¹ 12.3	all	¹ 67	46	¹ 75	¹ 67	11	12.12	¹ 1.2	¹ 7.12

主流移动端浏览器支持情况

5.2 HTML5 Geolocation的浏览器支持情况

Geolocation是HTML5规范中第一批被完整包含并实现的功能之一，所以主流浏览器基本都支持它。

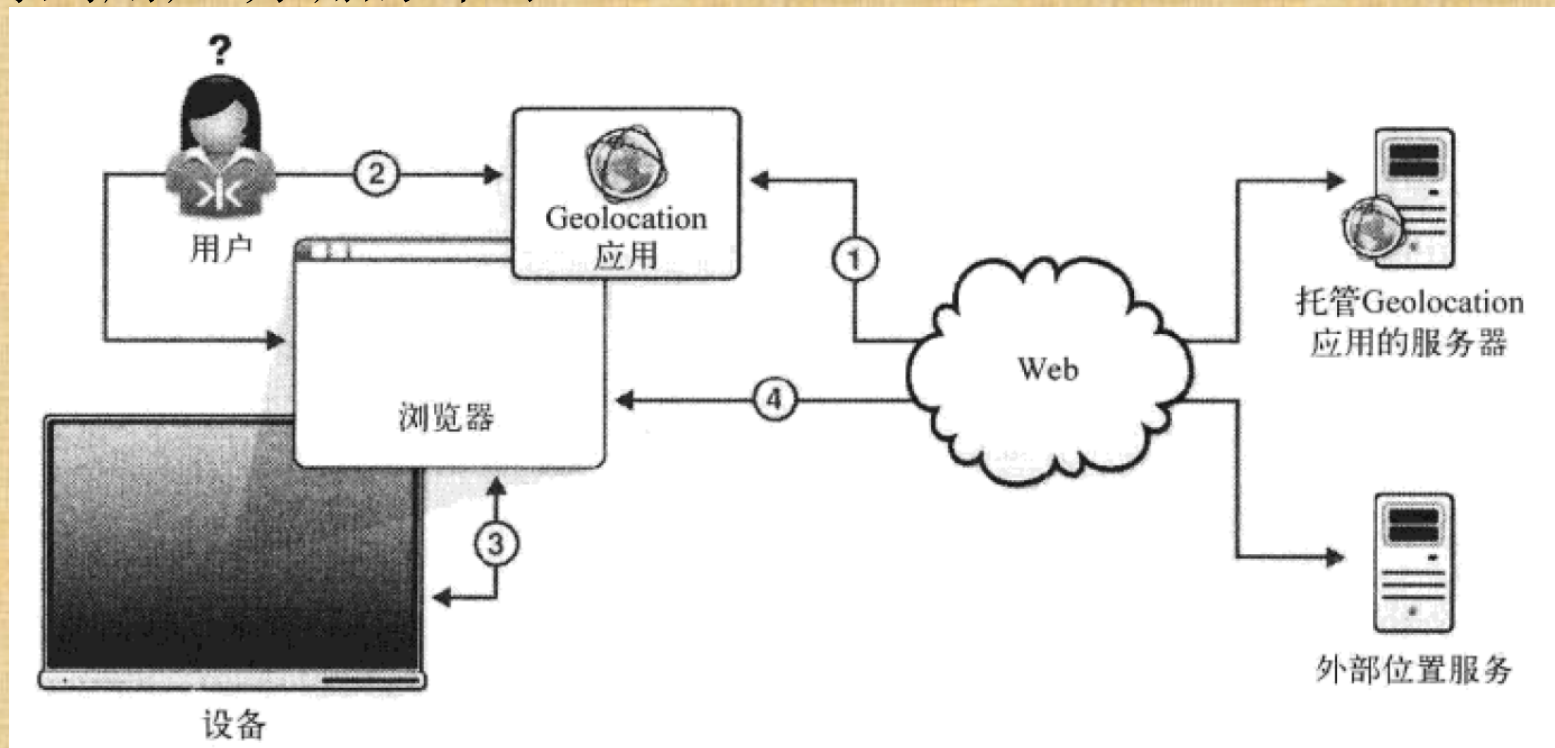
如果必须支持早期版本的浏览器，最好在调用API之前，先对浏览器进行针对Geolocation API支持的**检测**。

5.3 隐私

- 触发隐私保护机制
- 处理位置信息

□ 触发隐私保护机制

HTML5 Geolocation应用程序获取用户的位置信息需要得到用户明确的许可。



HTML地理定位浏览器和设备的隐私架构

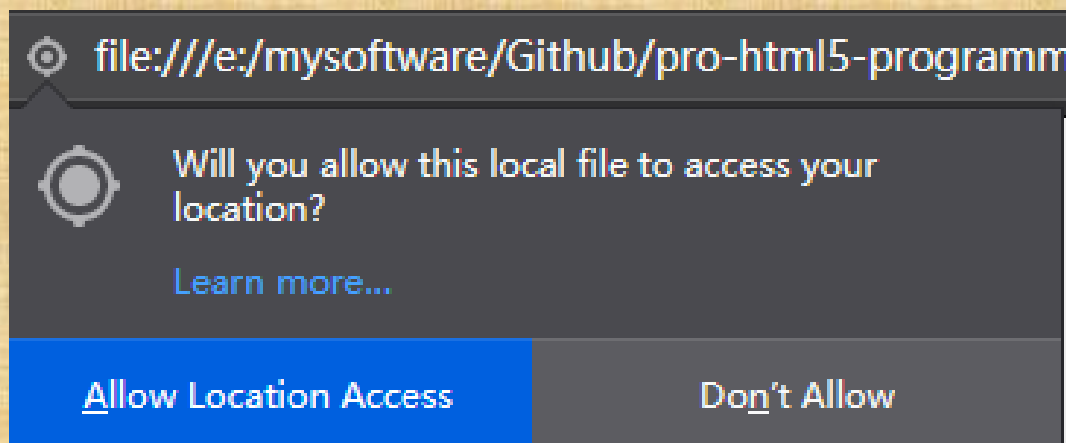
□ 触发隐私保护机制

HTML5地理定位浏览器和设备之间的交互：

- 1) 用户从浏览器中打开位置感知应用程序；
- 2) 应用程序Web页面加载，然后通过**Geolocation**函数调用请求位置坐标。浏览器拦截这一请求，然后请求用户授权。我们假设用户同意；
- 3) 浏览器从其宿主设备中检索坐标信息。例如，IP地址、**Wi-Fi**或**GPS**坐标。这是浏览器内部功能；
- 4) 浏览器将坐标发送给受信任的外部定位服务，它返回一个详细位置信息，并将该位置信息发回给**HTML5 Geolocation**应用程序。

□ 触发隐私保护机制

访问使用HTML5 Geolocation API的页面时，会触发保护机制。用户可以选择允许访问或者拒绝访问。



Firefox中调用HTML Geolocation API时触发的通知栏

□ 处理位置信息

位置数据属于**敏感信息**，接受到数据之后，我们必须小心处理、存储和重传。如果用户没有授权存储这些数据，任务完成后应立即删除。避免造成信息泄露等不安全因素。

在收集地理定位数据时，应用程序应该着重提示用户：

- 会收集位置数据；
- 为什么收集位置数据；
- 位置数据将保存多久；

- 怎样保证数据的安全；
- 位置数据怎样共享、和谁共享；
- 用户怎样检查和更新他们的位置数据。

5.4 使用HTML5 Geolocation API

- ❑ 浏览器支持性检查
- ❑ 位置请求

□ 浏览器支持性检查

作为开发人员，在调用HTML5 Geolocation API函数前，需要确保浏览器是否对其支持。可用代码进行检测：

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>HTML5 Geolocation浏览器支持性检测</title>
</head>
<body>

<script type="text/javascript">

  // 判断浏览器是否支持geolocation
  if(navigator.geolocation){
    //浏览器支持geolocation
    alert("浏览器支持！")

  }else{
    //浏览器不支持geolocation
    alert("浏览器不支持！");
  }

</script>
</body>
</html>
```

显示结果：

此网页显示
浏览器支持！

确定

ps:如果显示浏览器不支持，请升级浏览器或下载主流最新版本浏览器！

□ 位置请求

主要包括两种类型：

- 单次定位请求；

应用场景：附近商家定位等。

- 重复性的位置更新请求。

应用场景：计算行走路程、GPS导航等。

□ 位置请求

➤ 单次定位请求:

```
void getCurrentPosition(in PositionCallback successCallback,  
                        in optional PositionErrorCallback errorCallback,  
                        in optional positionOptions options);
```

— :定位成功参数 **—** :定位出错回调参数 **—** :其他操作

这个函数由 `navigator.geolocation` 对象调用，在脚本中需要先取得此对象。如前所述，应该确保一个合适的后备函数，用来应对浏览器不支持的情况。这个函数接受一个必选参数和两个可选参数。

□ 位置请求

successCallback:

表示调用getCurrentPosition函数成功以后的回调函数，该函数带有一个参数，对象字面量格式，表示获取到的用户位置数据。该对象包含两个属性 **coords** 和 **timestamp** (在实际应用中不常见)。其中 **coords** 属性包含以下6个值：

表 coords常用属性

属性值	含义	属性值	含义
accuracy	精确度	latitude	纬度
longitude	经度	altitude	海拔
heading	方向	speed	速度

□ 位置请求

```
function showPosition(position)
{
    x.innerHTML="纬度: " + position.coords.latitude +
    "<br>经度: " + position.coords.longitude +
    "<br>精确度: " + position.coords.accuracy +
    "<br>时间戳: " + position.timestamp;
}
```

示例中，showPosition()用来获取当前位置。

Timestamp是一份能够表示一份数据在一个特定时间点已经存在的完整的可验证的数据。主要是为用户提供一份电子证据，在电子商务、金融活动中使用较多。

□ 位置请求

errorCallback:

因为位置计算服务可能出差错，而应用程序对错处理非常重要。函数表示返回的错误代码。它包含以下两个属性：**message**：错误信息、**code**：错误代码。

表 错误编号详情

编号	代码	含义
1	PERMISSION_DENIED	表示用户拒绝浏览器获取位置信息的请求
2	POSITION_UNAVAILABLE	表示网络不可用或者连接不到卫星等
3	TIMEOUT	表示获取超时。
4	UNKNOWN_ERROR	表示未知错误

□ 位置请求

```
function showError(error)
{
    switch(error.code)
    {
        case error.PERMISSION_DENIED:
            x.innerHTML="用户拒绝对获取地理位置的请求。"
            break;
        case error.POSITION_UNAVAILABLE:
            x.innerHTML="位置信息是不可用的。"
            break;
        case error.TIMEOUT:
            x.innerHTML="请求用户地理位置超时。"
            break;
        case error.UNKNOWN_ERROR:
            x.innerHTML="未知错误。"
            break;
    }
}
```

showError()
函数用来显示
错误情况。利
用switch循环
语句输出可能
出现的差错。

□ 位置请求

options:

如果要同时处理正常情况和错误情况，有三个参数可选。

表 可选的地理定位请求特性

参数名	值	含义
<code>enableHighAccuracy</code>	布尔值	表示是否启用高精度模式，如果启用这种模式，则更耗时。
<code>timeout</code>	整数	表示在指定的时间内获取位置信息，否则触发 <code>errorCallback</code> 。
<code>maximumAge</code>	整数/常量	表示浏览器重新获取位置信息的时间间隔。

□ 位置请求

```
navigator.geolocation.getCurrentPosition(showPosition, showError, {  
    enableHighAccuracy: true,  
    maximumAge: 3000,  
    timeout: 10000});
```

这个调用告诉HTML5 Geolocation，任何处理时间超过10s(10000ms)的位置请求都应该触发一个错误；并每3秒(3000ms)获取一次位置数据。

`enableHighAccuracy`默认值为true，默认值为false（使用默认时可以忽略），表示使用高精度定位。

完整代码:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>单次定位请求</title>
  </head>
  <body>
    <p id="demo">点击按钮获取您当前坐标（可能需要比较长的时间获取）： </p>

    <button onclick="getLocation()">点我</button>

    <script>
      var x=document.getElementById("demo");

      //判断浏览器的支持性
      function getLocation()
      {
        if (navigator.geolocation)
```

```

if (navigator.geolocation)
{
    navigator.geolocation.getCurrentPosition(showPosition, showError, {
        enableHighAccuracy:true,
        maximumAge:3000,
        timeout:10000});
}
else
{
    x.innerHTML="该浏览器不支持定位。";
}

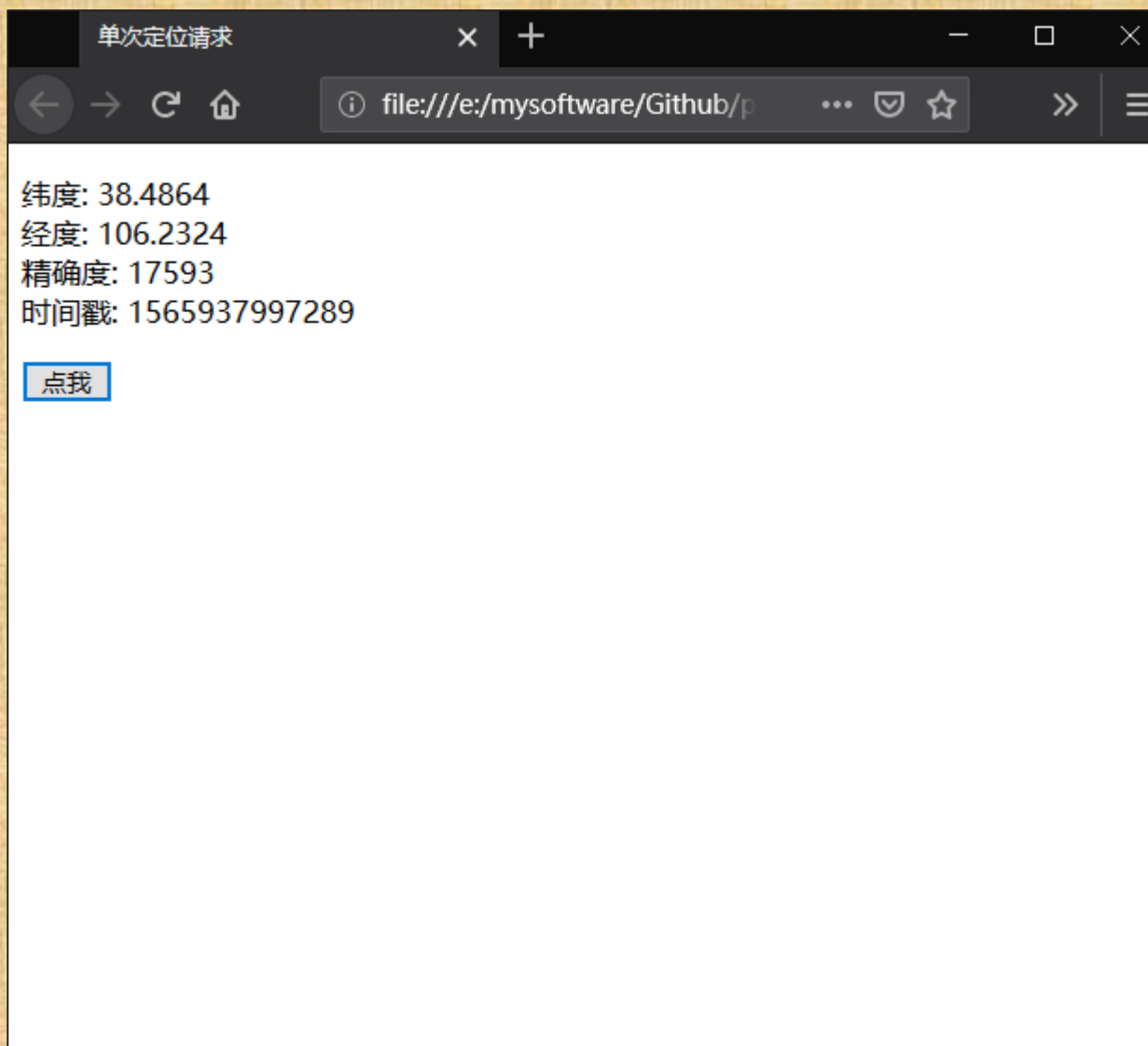
//显示位置信息
function showPosition(position)
{
    x.innerHTML="纬度: " + position.coords.latitude +
    "<br>经度: " + position.coords.longitude +
    "<br>精确度: " + position.coords.accuracy +
    "<br>时间戳: " + position.timestamp;
}

```



```
//错误处理
function showError(error)
{
    switch(error.code)
    {
        case error.PERMISSION_DENIED:
            x.innerHTML="用户拒绝对获取地理位置的请求。"
            break;
        case error.POSITION_UNAVAILABLE:
            x.innerHTML="位置信息是不可用的。"
            break;
        case error.TIMEOUT:
            x.innerHTML="请求用户地理位置超时。"
            break;
        case error.UNKNOWN_ERROR:
            x.innerHTML="未知错误。"
            break;
    }
}
</script>
</body>
</html>
```

运行结果：



□ 位置请求

➤ 重复定位请求:

有时候位置跟新一次是不够的，Geolocation可以在单次请求用户位置和以一定间隔多次请求用户位置间相互转换。

```
navigator.geolocation.getCurrentPosition(showPosition,  
                                          showError,  
                                          options);
```



```
navigator.geolocation.watchPosition(showPosition,  
                                     showError,  
                                     options);
```

□ 位置请求

在实际应用情况中，应用程序不再需要接收有关用户的持续位置更新，则只需调用`clearWatch()`函数。

```
var watchId = navigator.geolocation.watchPosition(updateLocation,  
                                                    handleLocationError,  
                                                    options);  
//持续更新坐标信息，对坐标信息进行应用.....  
  
//停止接收位置更新信息  
navigator.geolocation.clearWatch(watchId);
```

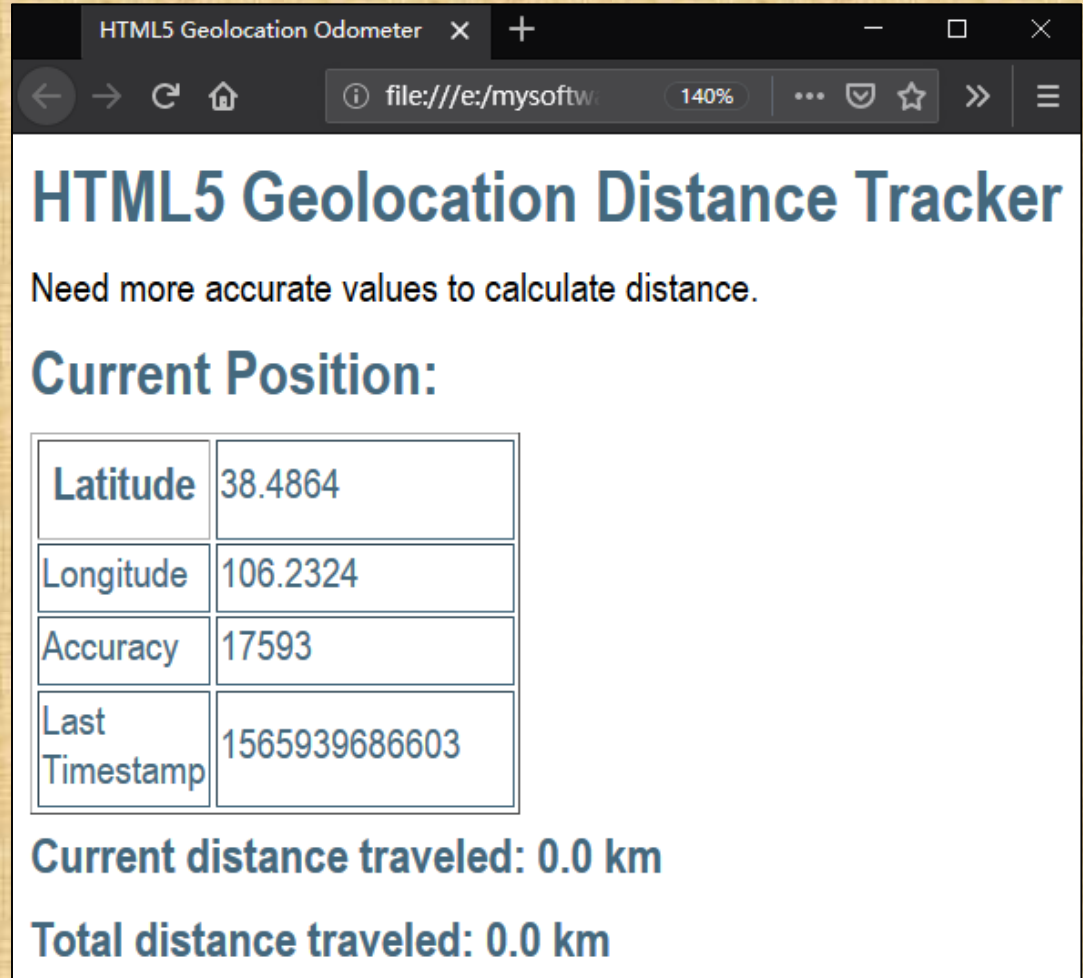
该函数会通知Geolocation服务，不在接收位置信息。`watchId`表示一个唯一的监视请求一边将来取消监视。

5.5 使用HTML5 Geolocation构建应用

- 编写HTML显示代码
- 处理Geolocation数据

5.5 使用HTML5 Geolocation构建应用

基于之前所学，我们使用多次请求特性构建一个简单有用的Web应用——距离跟踪器。其最终效果如图所示：



HTML5 Geolocation Odometer

file:///e:/mysoftw 140%

HTML5 Geolocation Distance Tracker

Need more accurate values to calculate distance.

Current Position:

Latitude	38.4864
Longitude	106.2324
Accuracy	17593
Last Timestamp	1565939686603

Current distance traveled: 0.0 km

Total distance traveled: 0.0 km

5.5 使用HTML5 Geolocation构建应用

距离计算使用公式Haversine公式实现，它能够根据经纬度来计算地球上两点间的距离，其数学公式如下：

$$d = 2\arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos\phi_1\cos\phi_2\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

下页演示了用JavaScript代码实现Haversine公式。针对目标，我们编写了将角度转换成弧度的方法，还提供了distance()函数来计算两个经纬度表示的位置间的距离。

5.5 使用HTML5 Geolocation构建应用

```
Number.prototype.toRadians = function() {  
    return this * Math.PI / 180;  
}  
  
function distance(latitude1, longitude1, latitude2, longitude2) {  
    // R是地球的半径，单位是千米  
    var R = 6371;  
    var deltaLatitude = (latitude2-latitude1).toRadians();  
    var deltaLongitude = (longitude2-longitude1).toRadians();  
    latitude1 = latitude1.toRadians(), latitude2 = latitude2.toRadians();  
    var a = Math.sin(deltaLatitude/2) *  
            Math.sin(deltaLatitude/2) +  
            Math.cos(latitude1) *  
            Math.cos(latitude2) *  
            Math.sin(deltaLongitude/2) *  
            Math.sin(deltaLongitude/2);  
    var c = 2 * Math.atan2(Math.sqrt(a),  
                           Math.sqrt(1-a));  
    var d = R * c;  
    return d;  
}
```


□ 编写HTML显示代码

```
<body onload="loadDemo()">
  <h1>HTML5 Geolocation Distance Tracker</h1>
  <p id="status">HTML5 Geolocation is <strong>not</strong> supported in your browser.</p>
  <h2>Current Position:</h2>
  <table border="1">
    <tr>
      <th width="40" scope="col"><h5>Latitude</h5></th>
      <td width="114" id="latitude">?</td>
    </tr>
    <tr>
      <td> Longitude</td>
      <td id="longitude">?</td>
    </tr>
    <tr>
      <td>Accuracy</td>
      <td id="accuracy">?</td>
    </tr>
    <tr>
      <td>Last Timestamp</td>
      <td id="timestamp">?</td>
    </tr>
  </table>
  <h4 id="currDist">Current distance traveled: 0.0 km</h4>
  <h4 id="totalDist">Total distance traveled: 0.0 km</h4>
  <script>...</script>
</body>
```

□ 处理Geolocation数据

添加loadDemo()方法

```
function loadDemo() {  
    if(navigator.geolocation) {  
        updateStatus("HTML5 Geolocation is supported in your browser.");  
        navigator.geolocation.watchPosition(updateLocation,  
                                              handleLocationError,  
                                              {enableHighAccuracy:true,  
                                              maximumAge:0,  
                                              timeout:10000});  
    }  
}
```

loadDemo()方法在页面加载完成时执行，并检测浏览器是否支持Geolocation，然后使用状态更新功能检测结果。

□ 处理Geolocation数据

添加handleLocationError()函数（即出错处理方法）：

```
function handleLocationError(error) {  
    switch(error.code)  
    {  
        case 0:  
            updateStatus("There was an error while retrieving your location: " + error.message);  
            break;  
        case 1:  
            updateStatus("The user prevented this page from retrieving a location."+ error.message);  
            break;  
        case 2:  
            updateStatus("The browser was unable to determine your location: " + error.message);  
            break;  
        case 3:  
            updateStatus("The browser timed out before retrieving the location."+ error.message);  
            break;  
    }  
}
```

□ 处理Geolocation数据

添加updateLocation()函数

```
function updateLocation(position) {  
    var latitude = position.coords.latitude;  
    var longitude = position.coords.longitude;  
    var accuracy = position.coords.accuracy;  
    var timestamp = position.timestamp;  
  
    document.getElementById("latitude").innerHTML = latitude;  
    document.getElementById("longitude").innerHTML = longitude;  
    document.getElementById("accuracy").innerHTML = accuracy;  
    document.getElementById("timestamp").innerHTML = timestamp;  
}
```

在此函数中我们将使用最新数据来更新页面并计算路程。

□ 处理Geolocation数据

添加updateLocation()函数

```
function updateLocation(position) {  
    var latitude = position.coords.latitude;  
    var longitude = position.coords.longitude;  
    var accuracy = position.coords.accuracy;  
    var timestamp = position.timestamp;  
  
    document.getElementById("latitude").innerHTML = latitude;  
    document.getElementById("longitude").innerHTML = longitude;  
    document.getElementById("accuracy").innerHTML = accuracy;  
    document.getElementById("timestamp").innerHTML = timestamp;  
}
```

在此函数中我们将使用最新数据来更新页面并计算路程。
并将这些数据更新到表格中。

□ 处理Geolocation数据

忽略不准确的更新：

```
// 完整性测试.....如果accuracy值过大，不要计算距离
if (accuracy >= 500) {
    updateStatus("Need more accurate values to calculate distance.");
    return;
}
```

准确度以m为单位，显示不准确的值会向用户提错误的位置信息，因此，这段代码将过滤掉所有低精度的位置更新数据，即忽略超过500米的精度。

□ 处理Geolocation数据

添加计算距离的代码:

```
// 计算距离
if ((lastLat != null) && (lastLong != null)) {
    var currentDistance = distance(latitude, longitude, lastLat, lastLong);
    document.getElementById("currDist").innerHTML =
        "Current distance traveled: " + currentDistance.toFixed(2) + " km";

    totalDistance += currentDistance;

    document.getElementById("totalDist").innerHTML =
        "Total distance traveled: " + currentDistance.toFixed(2) + " km";
}
lastLat = latitude;
lastLong = longitude;
updateStatus("Location successfully updated.");
```

```
}
```

□ 处理Geolocation数据

```
totalDistance += currentDistance;
```

```
document.getElementById("totalDist").innerHTML =  
    "Total distance traveled: " + currentDistance.toFixed(2) + " km";
```

上述代码将更新移动的总距离显示给用户，同时还储存当前值以备后面作比较。

`toFixed(2)`表示为使用四舍五入并保留两位小数点的方法显示数据。

完整代码示例：

```
<!DOCTYPE html>
<head>
  <meta charset="utf-8">
  <title>HTML5 Geolocation Odometer</title>
  <link rel="stylesheet" href="styles.css">
</head>

<body onload="loadDemo()">
  <h1>HTML5 Geolocation Distance Tracker</h1>
  <p id="status">HTML5 Geolocation is <strong>not</strong> supported in your browser.</p>
  <h2>Current Position:</h2>
  <table border="1">
    <tr>
      <th width="40" scope="col"><h5>Latitude</h5></th>
      <td width="114" id="latitude">?</td>
    </tr>
    <tr>
      <td> Longitude</td>
      <td id="longitude">?</td>
    </tr>
    <tr>
      <td>Accuracy</td>
      <td id="accuracy">?</td>
    </tr>
```

```

        <tr>
            <td>Last Timestamp</td>
            <td id="timestamp"?</td>
        </tr>
    </table>
    <h4 id="currDist">Current distance traveled: 0.0 km</h4>
    <h4 id="totalDist">Total distance traveled: 0.0 km</h4>
    <script>...</script>
</body>

```

```

<script type="text/javascript">

```

```

    var totalDistance = 0.0;
    var lastLat;
    var lastLong;

```

```

    Number.prototype.toRadians = function() {
        return this * Math.PI / 180;
    }

```

```

    function distance(latitude1, longitude1, latitude2, longitude2) {
        // R是地球的半径，单位是千米
        var R = 6371;
        var deltaLatitude = (latitude2-latitude1).toRadians();
        var deltaLongitude = (longitude2-longitude1).toRadians();
        latitude1 = latitude1.toRadians(), latitude2 = latitude2.toRadians();
    }

```

```

    var a = Math.sin(deltaLatitude/2) *
        Math.sin(deltaLatitude/2) +
        Math.cos(latitude1) *
        Math.cos(latitude2) *
        Math.sin(deltaLongitude/2) *
        Math.sin(deltaLongitude/2);
    var c = 2 * Math.atan2(Math.sqrt(a),
        Math.sqrt(1-a));

    var d = R * c;
    return d;
}

function updateStatus(message) {
    document.getElementById("status").innerHTML = message;
}

function loadDemo() {
    if(navigator.geolocation) {
        updateStatus("HTML5 Geolocation is supported in your browser.");
        navigator.geolocation.watchPosition(updateLocation,
                                            handleLocationError,
                                            {enableHighAccuracy:true,
                                             maximumAge:0,
                                             timeout:10000});
    }
}

```

```

function updateLocation(position) {
    var latitude = position.coords.latitude;
    var longitude = position.coords.longitude;
    var accuracy = position.coords.accuracy;
    var timestamp = position.timestamp;

    document.getElementById("latitude").innerHTML = latitude;
    document.getElementById("longitude").innerHTML = longitude;
    document.getElementById("accuracy").innerHTML = accuracy;
    document.getElementById("timestamp").innerHTML = timestamp;

    // 完整性测试.....如果accuracy值过大, 不要计算距离
    if (accuracy >= 500) {
        updateStatus("Need more accurate values to calculate distance.");
        return;
    }

    // 计算距离
    if ((lastLat != null) && (lastLong != null)) {
        var currentDistance = distance(latitude, longitude, lastLat, lastLong);
        document.getElementById("currDist").innerHTML =
            "Current distance traveled: " + currentDistance.toFixed(2) + " km";

        totalDistance += currentDistance;
    }
}

```



```

        document.getElementById("totalDist").innerHTML =
            "Total distance traveled: " + currentDistance.toFixed(2) + " km";
    }
    lastLat = latitude;
    lastLong = longitude;
    updateStatus("Location successfully updated.");
}

function handleLocationError(error) {
    switch(error.code)
    {
        case 0:
            updateStatus("There was an error while retrieving your location: " + error.message);
            break;
        case 1:
            updateStatus("The user prevented this page from retrieving a location."+ error.message);
            break;
        case 2:
            updateStatus("The browser was unable to determine your location: " + error.message);
            break;
        case 3:
            updateStatus("The browser timed out before retrieving the location."+ error.message);
            break;
    }
}

</script>
</body>
</html>

```

5.5 进阶功能

这里我们简单演示HTML5利用百度地图API进行地理定位。在调用百度地图的API前我们必须先**申请密钥**，即ak值。

申请地址为：<http://lbsyun.baidu.com/>，其代码效果如下所示：

```
<script type="text/javascript" src="http://api.map.baidu.com/api?v=2.0&ak=">  
</script>
```

如果需要更加深入的学习地图定位知识可以参考市场上主流的地图技术文档，如：

高德地图：<https://lbs.amap.com/>

腾讯地图：<https://lbs.qq.com/>

谷歌地图：<https://code.google.com/apis/console/>

1.5 进阶功能

HTML5 Geolocation is supported in your browser.

效果展示:



完整代码演示：

```
<!DOCTYPE html>
<html>
<title>HTML5调用百度地图API进行地理定位实例</title>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <script type="text/javascript" src="http://api.map.baidu.com/api?v=2.0&
      ak= " ></script>
  </head>
  <body style="margin:50px 10px;">
    <div id="status" style="text-align: center"></div>
    <div style="width:600px;height:480px;border:1px solid gray;
      margin:30px auto" id="container"></div>
    <script type="text/javascript">
      //默认地理位置设置为上海市浦东新区
      var x=121.48789949,y=31.24916171;
      window.onload = function() {
        if(navigator.geolocation) {
          navigator.geolocation.getCurrentPosition(showPosition);
          document.getElementById("status").innerHTML =
            "HTML5 Geolocation is supported in your browser.";
        }
      }
    </script>
  </body>
</html>
```



```

// 百度地图API功能
var map = new BMap.Map("container");           //创建地图实例
var point = new BMap.Point(x,y);
map.centerAndZoom(point,12);                     //获取当前视野的中心点
var geolocation = new BMap.Geolocation();
geolocation.getCurrentPosition(function(r){
    if(this.getStatus() == BMAP_STATUS_SUCCESS){
        var mk = new BMap.Marker(r.point);
        map.addOverlay(mk);
        map.panTo(r.point);
    }
    else {
        alert('failed'+this.getStatus());
    }
    },{enableHighAccuracy: true})
return;
}
alert("你的浏览器不支持获取地理位置！");
};

```

```
function showPosition(position){  
    x=position.coords.latitude;  
    y=position.coords.longitude;  
}  
</script>  
</body>  
</html>
```

5.5 课后思考

- ❑ 有哪些方法可以提高Geolocation API检索到的数据的准确性？
- ❑ timeout和maximumAge区别？
- ❑ 如何实时对位置信息进行监控？

5.6 小结

- ❑ 本章讲述了HTML5 Geolocation的位置信息——经纬度和其他特性，以及获取途径。
- ❑ 探讨了由HTML5 Geolocation引发的隐私问题。
- ❑ 最后创建了位置感知的Web应用程序。