UT010.1: Mapas con Leaflet

Contenido

1. Introducción

El término geoposicionamieto o geolocalización hace referencia a ubicar un objeto sobre la superficie terrestre utilizando las coordenadas de latitud y longitud.

Este tipo de funcionalidad es muy utiliza en muchas aplicaciones. Para implementar el geoposicionamiento vamos a utilizar la librería opensource **Leaflet** https://leafletjs.com/ ya que nos ofrece de manera gratuita la gestión los mapas.

Como capa para visualizar los mapas utilizaremos **Open Street Map** https://www.openstreetmap.org/.

2. API de Geolocalización

Este API permite obtener las coordenadas de latitud y longitud del dispositivo que estamos utilizando. Lo ideal es disponer de un dispositivo con GPS para obtener una ubicación más precisa, pero los equipos sin GPS pueden obtenerla basándose en la red a la que estamos conectados.

El objeto `navigator` dispone de una propiedad `geolocation` que dispone de un objeto para conseguir la propiedad latitud y longitud de la ubicación del dispositivo utilizando el método `getCurrentPosition()`. Este método debe recibir una función de *callback* que será invocada desde el objeto para que podamos obtener ambos datos. La función de *callback* recibe un argumento con un objeto literal con las propiedades `coords.latitude` y `coords.longitude`.

En la siguiente función chequeamos si podemos obtener las coordenadas, y en caso de obtenerlas, invocamos la función de *callback* para mostrarlas.

```
function getLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(showPosition);
  } else {
    alert("Geolocation no está soportada en el navegador.");
  }
}

function showPosition(position) {
  console.log(position.coords.latitude);
  console.log(position.coords.longitude);
}
getLocation();
```

Podemos pasar un segundo argumento al método `getCurrentPosition()` con una función de *callback* que será invocada en caso de producirse algún error. La función recibe un argumento con la indicación del error.

```
function showError(error) {
  switch(error.code) {
    case error.PERMISSION_DENIED:
      console.log('El usuario ha denegado el acceso a la localización.');
      break;
    case error.POSITION_UNAVAILABLE:
      console.log('Localización inalcanzable');
      break;
    case error.TIMEOUT:
      console.log('Ha saltado el time out.');
      break;
    case error.UNKNOWN_ERROR:
      console.log('Error desconocido.');
      break;
  }
}
```

Invocamos el método `getCurrentPosition()` de la siguiente forma.

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

Obtener las coordenadas de la ubicación es un dato privado, por lo que el usuario debe dar permisos específicos para que podamos acceder. Algunos navegadores **impiden acceder** a esta función si no lo hacemos mediante un **protocolo HTTPS**.

Otro método interesante para utilizar con un dispositivo móvil con GPS es `watchPosition()`, que genera la posición en rangos de tiempo, lo que nos permitiría mostrar el movimiento del dispositivo. Para detener este cálculo tenemos el método `clearWatch()`.

3. Uso de Leaflet y Open Street Map

Vamos a integrar la librería LeafLet y los mapas Open Street Map en nuestra página.

3.1. Creación del mapa

Empezamos con la creación del mapa. Veamos los pasos para generar un mapa en la página.

1. Importamos la hoja de estilos en el documento HTML.

```
<link rel="stylesheet"
    href="https://unpkg.com/leaflet@1.7.1/dist/leaflet.css"
    integrity="sha512-
xodZBNTC5n17Xt2atTPuE1HxjVMSvLVW9ocqUKLsCC5CXdbqCmblAshOMAS6/keqq/sMZMZ19
scR4PsZChSR7A=="
    crossorigin=""/>
```

2. Importamos la librería JavaScript de Leaflet en el documento HTML.

```
<script src="https://unpkg.com/leaflet@1.7.1/dist/leaflet.js"
    integrity="sha512-
XQoYMqMTK8LvdxXYG3nZ448hOEQiglfqkJs1NOQV44cWnUrBc8PkAOcXy20w0vlaXaVUearIO
BhiXZ5V3ynxwA=="
    crossorigin=""></script>
```

3. Tenemos que generar una capa para mostrar el mapa. Construimos la capa donde mostrar el mapa asociándole un identificador. Para permitir visualizar el mapa debemos indicarle una altura mediante CSS.

```
const main = document.querySelector('main');
main.replaceChildren();
main.insertAdjacentHTML('afterbegin', '<div class="container"><div
class="m-4" id="mapid"></div></div>');
const mapContainer = document.getElementById('mapid');
mapContainer.style.height = '350px';
mapContainer.style.border = '2px solid #faa541';
```

4. El método `map()` de la librería recibe como argumento el identificador de la capa donde vamos a mostrar el mapa. Para ubicar el mapa en una posición en concreto debemos invocar el método `setView()` sobre el objeto con el mapa, pasándole un array con las coordenadas de latitud y longitud, y el zoom para mostrar el mapa.

```
let map = L.map('mapid')
    .setView([38.990831799999995, -3.9206173000000004], 15);
```

5. De momento tan solo disponemos del contenedor del mapa, ahora debemos añadirle una capa con el mosaico de imágenes que componen el mapa. Como hemos comentado, para crear el mosaico vamos a utilizar Open Street Map, ya que se trata de un servicio gratuito. El método `tileLayer()` permite configurar la plantilla de URL que vamos a utilizar para obtener las imágenes del mosaico. Al método le pasamos la URL de Open Street Map y un objeto literal para la configuración, en este caso fijamos la propiedad `attribution`, para indicar de dónde estamos obteniendo los mapas, y la propiedad `maxZoom`, con el zoom máximo que vamos a permitir en el mapa. Por último, invocamos el método `addTop()` para asignar la plantilla de URL a nuestro mapa.

```
L.tileLayer('http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
        attribution: 'Map data &copy; <a href="http://openstreetmap.org">
OpenStreetMap</a> contributors, <a href="http://creativecommons.org/licen
ses/by-sa/2.0/">CC-BY-
SA</a>, Imagery © <a href="http://cloudmade.com">CloudMade</a>',
        maxZoom: 18
}).addTo(map);
```

La librería está desacoplada de la fuente de imágenes con los mapas, por lo que, además de Open Street Map, podríamos utilizar otros servicios con una visualización de mapas diferentes.

3.2. Marcadores, círculos y polígonos

Podemos añadir marcadores a nuestro mapa. El método `marker()` crea un marcador pasándole como argumentos el array con la latitud y la longitud. Para asignarlo al mapa utilizamos el método `addTo()`.

```
let marker = L.marker([38.990831799999995, -
3.9206173000000004]).addTo(map);
```

Si queremos generar un círculo utilizamos el método `circle()`. Además de las coordenadas, podemos pasar como argumento un objeto literal con las propiedades del círculo como el radio, el color, el relleno o la opacidad que le vamos a aplicar.

```
let circle = L.circle([38.98845529016365, -3.928015709152533], {
    color: 'red',
    fillColor: '#f03',
    fillOpacity: 0.5,
    radius: 100
}).addTo(map);
```

O también podemos generar un polígono con el método `polygon()`, pasando como argumentos las coordenadas para completar el polígono.

```
let polygon = L.polygon([
    [38.991470693002185, -3.919398307898519],
    [38.990696840612586, -3.917407036187797],
    [38.98752129002853, -3.915415763167403],
    [38.986720708675215, -3.9181623451986534]
]).addTo(map);
```

3.3. Bocadillos

Podemos vincular un bocadillo a un marcador utilizando el método `bindPopup()`. El método recibe como argumento el código HTML que se mostrará en el bocadillo. El método `openPopup()` permite mostrar el bocadillo.

```
marker.bindPopup('<strong>IES Maestre de Calatrava</strong><br>Nuestro ce
ntro de estudios.').openPopup();
circle.bindPopup('Diputación de Ciudad Real');
polygon.bindPopup('Ciudad deportiva de Ciudad Real');
```

También podemos crear un bocadillo sin asociarlo a un marcador en concreto.

```
let popup = L.popup()
    .setLatLng([38.994325866209785, -3.924805640781415])
    .setContent('Ronda de Toledo.')
    .openOn(map);
```

3.4. Eventos en el mapa

Podemos detectar el punto donde cliqueamos dentro de un mapa. El método `on()` permite asociar un evento al mapa, pasando el manejador de eventos que se ejecutará cuando se produzca. En el ejemplo estamos creando un evento *click*. El objeto con el evento dispone de una propiedad `latlng` donde podemos obtener las coordenadas del punto donde hemos cliqueado. El manejador de este ejemplo genera un marcador cada vez que pulsamos en un punto.

```
map.on('click', function(event) {
    L.marker([event.latlng.lat, event.latlng.lng]).addTo(map);
});
```

Por último, modificamos la posición del marcador inicial cada vez que pulsamos con el botón secundario.

```
map.on('contextmenu', function(event) {
    marker.setLatLng([event.latlng.lat, event.latlng.lng]);
});
```

4. Ejercicio. Crear un Geocoder

Un **Geocoder** es un servicio que dada una dirección nos devuelve las coordenadas con su latitud y su longitud. En Open Street Map disponemos de este servicio en la URL https://nominatim.openstreetmap.org/search. El servicio necesita varios parámetros.

- *format*: Indicamos el formato JSON.
- *limit*: Número de resultados que devuelve el servicio para una dirección determinada.
- *q*: La dirección que estamos buscando.

Creamos un formulario que recoja una dirección, y permita mostrar un mapa con la ubicación seleccionada.

Solución

```
const main = document.querySelector('main');
main.replaceChildren();
main.insertAdjacentHTML('afterbegin', `<div class="container p-4">
  <form id="fGeocoder" method="get">
    <h2>Ejemplo de uso de GeoCoder</h2>
    <div class="form-group row">
      <div class="col-sm-10">
        <label for="address" class="col-form-label">Dirección</label>
        <input type="text" name="q" class="form-control" id="address"
placeholder="Introduce la dirección a buscar">
      </div>
      <div class="col-sm-2 align-self-end">
        <button id="bAddress" class="btn btn-primary"
type="submit">Buscar</button>
      </div>
    </div>
    <div id="geocoderAddresses"></div>
    <div id="geocoderMap" class="my-2"></div>
  </form>
</div>`);

const form = document.forms.fGeocoder;
const addresses = document.getElementById('geocoderAddresses');
const mapContainer = document.getElementById('geocoderMap');
let map = null;
```

```
form.addEventListener('submit', function (event) {
  const url = new URL('https://nominatim.openstreetmap.org/search');
  url.searchParams.append('format', 'json');
  url.searchParams.append('limit', 3);
  url.searchParams.append('q', this.q.value);
  fetch(url, {
    method: 'get',
  })
    .then((response) => response.json())
    .then((data) => {
      const list = document.createElement('div');
      list.classList.add('list-group');
      data.forEach((address) => {
        list.insertAdjacentHTML('beforeend', `<a href="#" data-
lat="${address.lat}" data-lon="${address.lon}" class="list-group-item
list-group-item-action">
          ${address.display_name}</a>`);
      });
      addresses.replaceChildren();
      addresses.append(list);

      const links = document.getElementsByTagName('a');
      for (const link of links) {
        link.addEventListener('click', (event) => {
          for (const link of links) {
            link.classList.remove('active');
          }
          this.classList.add('active');
          if (map) {
            map.setView(new L.LatLng(event.currentTarget.dataset.lat,
event.currentTarget.dataset.lon), 15);
          } else {
            mapContainer.style.height = '350px';
            mapContainer.style.border = '2px solid #faa541';
            map =
L.map('geocoderMap').setView([event.currentTarget.dataset.lat,
event.currentTarget.dataset.lon], 15);
            L.tileLayer('http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.pn
g', {
              attribution: 'Map data &copy; <a
href="http://openstreetmap.org">OpenStreetMap</a> contributors, <a
href="http://creativecommons.org/licenses/by-sa/2.0/">CC-BY-SA</a>,
Imagery © <a href="http://cloudmade.com">CloudMade</a>',
              maxZoom: 18,
            }).addTo(map);
          }
```

```
          L.marker([event.currentTarget.dataset.lat,
event.currentTarget.dataset.lon]).addTo(map);
          event.preventDefault();
        });
      }
      console.log(data);
    })
    .catch((error) => {
      addresses.replaceChildren();
      addresses.insertAdjacentHTML('afterbefin', `<div class="text-
danger">
        <i class="bi bi-exclamation-circle-fill"></i>
        No se ha podido establecer la conexión con el servidor de mapas.
      </div>`);
    });

  event.preventDefault();
});
```