Aplicaciones Telemáticas, examen práctico, 15/05/2023

Grado en Ingeniería Telemática

Universidad Rey Juan Carlos

Instrucciones

1. Ejecuta en un terminal `~mortuno/prepara` y comprueba que esto ha creado los ficheros

   - `~/at.mayo.23/distancias.TULOGIN.js` donde resolverás el ejercicio 1.
   - `~/at.mayo.23/figuras.TULOGIN.html` y `~/at.mayo.23/figuras.TULOGIN.js` que contienen una copia de tus ficheros de la práctica 4.7. Deberás modificarlos para hacer el ejercicio 2.

2. Emplea lo visto en clase, y solo lo visto en clase (que puedes consultar en las transparencias). No será válido ni JavaScript más antiguo ni más avanzado.

Ejercicio 1. (3 puntos)

Para conocer la distancia entre dos puntos en el plano, basta aplicar el teorema de Pitágoras. Pero como la Tierra es esférica, para conocer la distancia entre dos puntos del planeta a partir de sus coordenadas es necesario aplicar la fórmula del semiverseno.

En el fichero `~/at.mayo.23/distancias.TULOGIN.js` encontrarás una implementación en JavaScript de la fórmula del semiverseno. Úsala para escribir un programa de acuerdo con la siguiente especificación.

1. El programa tendrá una función llamada *informe()* que recibirá un array de puntos.
2. Cada elemento de este array será a su vez otro array, formado por una cadena con el nombre del punto, su latitud y su longitud.
   Ejemplo:

```
p1 = ["madrid",  40.416, -3.703]
p2 = ["oviedo", 43.362, -5.848]
p3 = ["fuenlabrada", 40.284, -3.799]
p4 = ["sevilla",  37.385, -5.994]
puntos = [p1,p2,p3,p4]
```

3. La función *informe()* devolverá una lista con la distancia de cada punto con todos los demás (excluido él mismo).
4. Cada elemento de esta lista será a su vez otra lista

   - Su primer elemento será una cadena con el nombre del primer punto.
   - Su segundo elemento será una cadena con el nombre del segundo punto.
   - Su tercer elemento será un número entero con la distancia en kilómetros entre ambos puntos.

5. Ejemplo de valor devuelvo

```
[
  [ 'madrid', 'oviedo', 373 ],
  [ 'madrid', 'fuenlabrada', 17 ],
  [ 'madrid', 'sevilla', 391 ],
  [ 'oviedo', 'madrid', 373 ],
  [ 'oviedo', 'fuenlabrada', 382 ],
  [ 'oviedo', 'sevilla', 665 ],
  [ 'fuenlabrada', 'madrid', 17 ],
  [ 'fuenlabrada', 'oviedo', 382 ],
  [ 'fuenlabrada', 'sevilla', 374 ],
  [ 'sevilla', 'madrid', 391 ],
  [ 'sevilla', 'oviedo', 665 ],
  [ 'sevilla', 'fuenlabrada', 374 ]
]
```

6. La función *informe()* devolverá la lista requerida, pero no tendrá efectos laterales, incluyendo el no escribir nada en la salide estándar. En otras palabras: aunque es conveniente que uses trazas, bórralas o coméntalas cuando acabes.

7. El programa puede tener todas las funciones adicionales que te parezca conveniente.

Solución)

```
'use strict'
function semiverseno(coords1, coords2) {
    // Recibe dos parámetros, cada uno es una lista [latitud, longitud]
    // Devuelve la distancia en metros entre ambos puntos.
    // Extraido de https://github.com/dcousens/haversine-distance

    function toRad(x) {
      return x * Math.PI / 180;
    }

    var lat1 = coords1[0];
    var lon1 = coords1[1];

    var lat2 = coords2[0];
    var lon2 = coords2[1];

    var R = 6371000; // metros

    var x1 = lat2 - lat1;
    var dLat = toRad(x1);
    var x2 = lon2 - lon1;
    var dLon = toRad(x2)
    var a = Math.sin(dLat / 2) * Math.sin(dLat / 2) +
      Math.cos(toRad(lat1)) * Math.cos(toRad(lat2)) *
      Math.sin(dLon / 2) * Math.sin(dLon / 2);
    var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
    var d = R * c;

    return d;
}

function distancia(a,b){
    return semiverseno( [a[1],a[2]], [b[1],b[2]]) / 1000;
}

let p1 = ["madrid",  40.416, -3.703]
let p2 = ["oviedo", 43.362, -5.848]
let p3 = ["fuenlabrada", 40.284, -3.799]
let p4 = ["sevilla",  37.385, -5.994]

let puntos = [p1,p2,p3,p4]

function informe(puntos){
    let rval = []
    let elemento;
    for  (let a of puntos){
        for (let b of puntos){
            if (a !== b){
                elemento = [];
                elemento.push(a[0]);
                elemento.push(b[0]);
                elemento.push(Math.round(distancia(a,b)));
                rval.push(elemento);
            }
        }
    }
    return rval;
}

console.log(informe(puntos))
```

Ejercicio 2. (7 puntos)

Modifica los ficheros `~/at.mayo.23/figuras.TULOGIN.html` y `~/at.mayo.23/figuras.TULOGIN.js` según esta especificación:

1. Hasta ahora, tu programa o bien añadía una figura en modo texto o bien añadía una figura en modo gráfico. Una figura en modo texto siempre se quedaba en modo texto, una figura en modo gráfico siempre se quedaba en modo gráfico.

   Ahora, tu programa debe tener un único botón para añadir figuras. Y un segundo botón para elegir *modo gráfico* o *modo texto*.

   - En modo gráfico, todas las figuras se verán en modo gráfico. Incluyendo las que previamente fueron lanzadas en modo texto.
   - En modo texto, todas las figuras se verán en modo texto, incluyendo las que previamente fueron lanzadas en modo gráfico.

2. Tu programa siempre trabajará en modo *sin repetición*. Borra todo lo relativo al modo *con repetición*.

3. En tu programa se tiene que ir viendo el histórico de todas las figuras aparecidas.

   - Cuando sale una figura nueva no puede borrar la anterior.
   - Al cambiar el modo gráfico/texto, no pueden borrar las figuras.
   - Al pulsar *nuevo mazo*, las figuras sí se pueden borrar. O no. Como prefieras.

4. Cuando tu programa esté en *modo gráfico*, el botón de cambio de modo debe contener el texto *modo texto*, porque al pulsarlo este será el nuevo modo.

5. Cuando tu programa esté en *modo texto*, el botón de cambio de modo debe contener el texto *modo gráfico*, porque al pulsarlo este será el nuevo modo.

Sugerencia

- Cuando añadas una figura, añade simultáneamente el texto y la imagen. Pero con clases distintas. Luego, al cambiar de modo, haz que unas u otras sean visibles o invisibles, según corresponda.