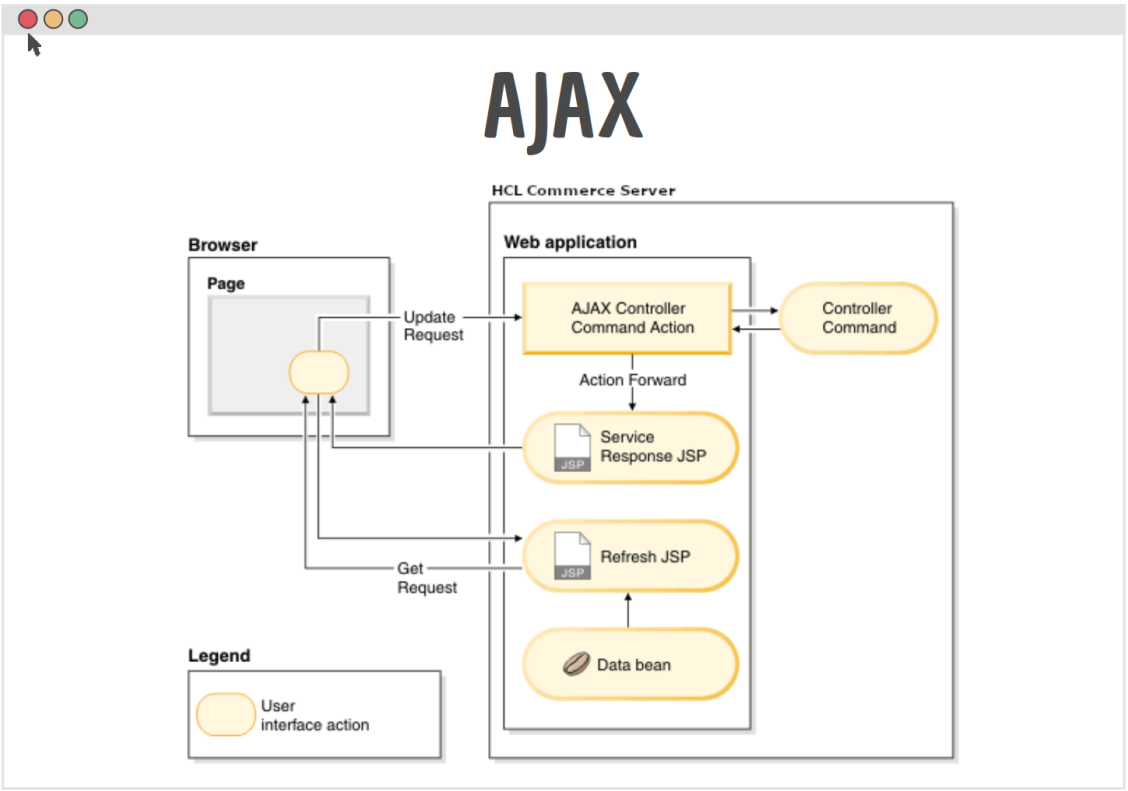
PROGRAMACIÓN WEB 2



Profesor(a):
Carlo Jose Luis Corrales Delgado

Estudiante:
Mamani Anahua, Victor Narciso
Mamani Huarsaya, Jorge Luis
Quispe Marca Edysson Darwin
Velarde Saldaña Jhossep Fabritzio
Zuñiga Villacorta Peter Sebastian

Repositorios GitHub:
https://github.com/jorghee/tarea_ajax
https://github.com/jorghee/ajax_markdown

18 de mayo, 2024

Los 8 ejercicios JavaScript con Ajax y Google Chart

En el priemr ejercicio tenemos una función de JavaScript llamada listRegions esta función recibe un arreglo de datos como parámetro y actualiza el contenido de un elemento HTML identificado por el id 'result'. Primero, borra cualquier contenido previo dentro de este elemento. Luego, itera sobre cada elemento del arreglo de datos. Para cada elemento, imprime el valor de la propiedad 'region' en la consola del navegador y crea un nuevo elemento de lista (li) con este valor como contenido de texto. Finalmente, añade este elemento de lista como hijo del elemento 'result' en el documento HTML.

```
// Autor: Sebastian Zuñiga
export function listRegions(data){
  document.getElementById('result').innerHTML = '';
  data.forEach(element => {
    console.log(element.region);
    document.getElementById('result').appendChild(document.createElement('li')).textContent = element.region;
  });
}
```

Figura 1: ejercicio1.js

En el segundo ejercicio tenemos la función totalConfirmed toma un arreglo de datos como entrada y calcula el total de casos confirmados por región a partir de esos datos. Luego, genera una lista en el documento HTML que muestra cada región junto con su total de casos confirmados.

```
export function totalConfirmed(data){
  // Obtener el elemento donde se mostrarán los resultados
  const resultDiv = document.getElementById('result');
  resultDiv.innerHTML = ''; // Limpiar cualquier contenido existente

  // Calcular el total de casos confirmados por región
  const regionTotals = data.reduce((acc, regionData) => {
    const totalConfirmed = regionData.confirmed.reduce((sum, dayData) => {
      return sum + parseInt(dayData.value, 10); // Sumar los valores confirmados por día
    }, 0);
    acc[regionData.region] = totalConfirmed; // Almacenar el total en el objeto acumulador
    return acc;
  }, {});

  // Crear una lista no ordenada para mostrar los resultados
  const ul = document.createElement('ul');
  for (const [region, confirmed] of Object.entries(regionTotals)) {
    const li = document.createElement('li'); // Crear un elemento de lista para cada región
    li.textContent = `${region}: ${confirmed}`; // Establecer el texto del elemento de lista
    ul.appendChild(li); // Agregar el elemento de lista a la lista no ordenada
  }
  resultDiv.appendChild(ul);
}
```

Figura 2: ejercicio2.js

En el tercer ejercicio la función top10Regions también recibe un arreglo de datos y muestra los 10 principales regiones junto con sus totales de casos confirmados en orden descendente en el documento

HTML. Comienza limpiando cualquier contenido previo dentro del elemento HTML identificado por el id 'result'. Luego, calcula el total de casos confirmados por región, similar a la función totalConfirmed. Después, ordena las regiones por el número total de casos confirmados en orden descendente y toma las primeras 10. Finalmente, crea una lista no ordenada en el documento HTML para mostrar los resultados y agrega cada región junto con su total de casos confirmados como elementos de lista.

```
export function top10Regions(data) {
  // Obtener el elemento donde se mostrarán los resultados
  const resultDiv = document.getElementById('result');
  resultDiv.innerHTML = ''; // Limpiar cualquier contenido existente

  // Calcular el total de casos confirmados por región
  const regionTotals = data.reduce((acc, regionData) => {
    const totalConfirmed = regionData.confirmed.reduce((sum, dayData) => {
      return sum + parseInt(dayData.value, 10); // Sumar los valores confirmados por día
    }, 0);
    acc[regionData.region] = totalConfirmed; // Almacenar el total en el objeto acumulador
    return acc;
  }, {});

  // Ordenar las regiones por número total de casos confirmados en orden descendente y obtener las 10 primeras
  const sortedRegions = Object.entries(regionTotals).sort((a, b) => b[1] - a[1]).slice(0, 10);

  // Crear una lista no ordenada para mostrar los resultados
  const ul = document.createElement('ul');
  for (const [region, confirmed] of sortedRegions) {
    const li = document.createElement('li'); // Crear un elemento de lista para cada región
    li.textContent = `${region}: ${confirmed}`; // Establecer el texto del elemento de lista
    ul.appendChild(li); // Agregar el elemento de lista a la lista no ordenada
  }

  // Agregar la lista no ordenada al elemento de resultados
  resultDiv.appendChild(ul);
}
```

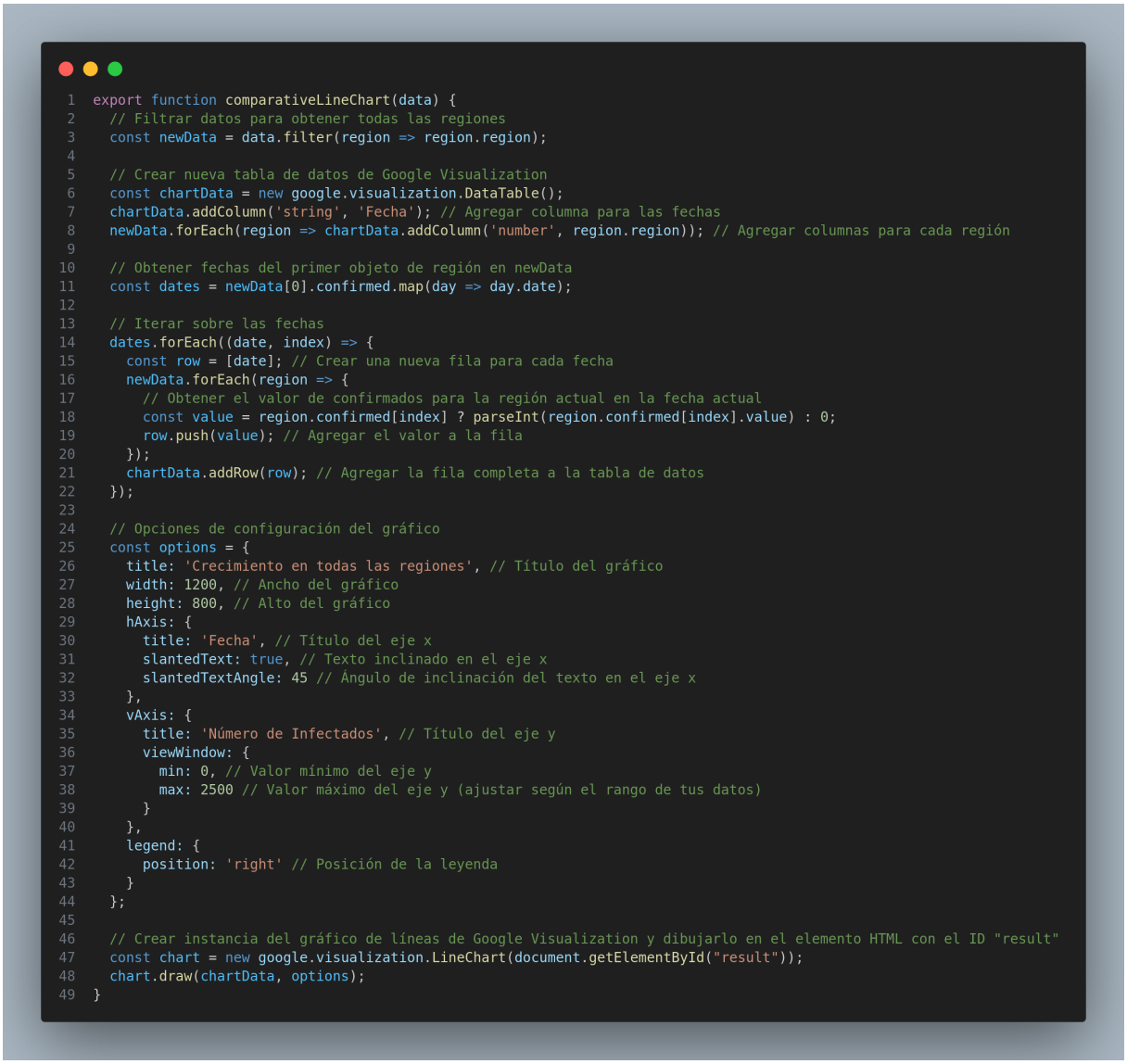
Figura 3: ejercicio3.js

En el cuarto ejercicio la función arequipaInfected se encarga de visualizar los datos de infectados en la región de Arequipa a lo largo del tiempo. Comienza filtrando los datos para obtener solo la información relacionada con la región de Arequipa. Luego, crea una nueva tabla de datos de Google Visualization y agrega una columna para las fechas y columnas para cada región en Arequipa. Posteriormente, itera sobre las fechas y para cada fecha, crea una nueva fila y agrega los valores de infectados correspondientes a cada región en esa fecha. Establece opciones de configuración para el gráfico, como título, dimensiones y estilos de ejes, y finalmente, crea una instancia del gráfico de líneas de Google Visualization y lo dibuja en el elemento HTML con el ID result".

```js
// Esta función visualiza los datos de infectados en la región de Arequipa en el tiempo de los valores
export function arequipaInfected(data){
  // Filtrar los datos para obtener solo la información relacionada con la región de Arequipa
  const newData = data.filter(region => region.region == 'Arequipa');
  // Crear una nueva tabla de datos de Google Visualization
  const chartData = new google.visualization.DataTable();
  // Agregar una columna para las fechas
  chartData.addColumn('string', 'Fecha');
  // Agregar columnas para cada región en Arequipa
  newData.forEach(region => chartData.addColumn('number', region.region));

  // Extraer las fechas del primer objeto de región en newData
  const dates = newData[0].confirmed.map(day => day.date);
  // Iterar sobre las fechas
  dates.forEach((date, index) => {
    // Crear una nueva fila para cada fecha
    const row = [date];
    // Iterar sobre cada región en newData
    newData.forEach(region => {
      // Obtener el valor de infectados para la región actual en la fecha actual
      const value = region.confirmed[index] ? parseInt(region.confirmed[index].value) : 0;
      // Agregar el valor a la fila
      row.push(value);
    });
    // Agregar la fila completa a la tabla de datos
    chartData.addRow(row);
  });

  const options = {
    title: 'Infectados en Arequipa',
    width: 1200, // Ancho del gráfico
    height: 800, // Alto del gráfico
    hAxis: {
      title: 'Fecha', // Título del eje x
      slantedText: true,
      slantedTextAngle: 45
    },
    vAxis: {
      title: 'Número de Infectados', // Título del eje y
      viewWindow: { //Min y Max para los numeros de infectados
        min: 0,
        max: 1200
      }
    },
    legend: {
      position: 'right'
    }
  };

  // Crear una instancia del gráfico de líneas de Google Visualization y dibujarlo en el elemento HTML con el ID "result"
  const chart = new google.visualization.LineChart(document.getElementById("result"));
  chart.draw(chartData, options);
}
```

Figura 4: ejercicio4.js

En el quinto ejercicio la función comparativeLineChart analiza datos sobre casos confirmados en todas las regiones y crea un gráfico de líneas que compara el crecimiento de estos casos a lo largo del tiempo. Para ello, primero filtra los datos para obtener información de todas las regiones disponibles. Luego, estructura estos datos en una tabla de Google Visualization, donde cada fila representa una fecha y cada columna representa una región, con los valores de casos confirmados en cada fecha. Se definen opciones para personalizar el aspecto del gráfico, como el título y los ejes x e y. Finalmente, utiliza la biblioteca de Google Visualization para dibujar el gráfico en un elemento HTML específico identificado por el ID result".



```
export function comparativeLineChart(data) {
  // Filtrar datos para obtener todas las regiones
  const newData = data.filter(region => region.region);

  // Crear nueva tabla de datos de Google Visualization
  const chartData = new google.visualization.DataTable();
  chartData.addColumn('string', 'Fecha'); // Agregar columna para las fechas
  newData.forEach(region => chartData.addColumn('number', region.region)); // Agregar columnas para cada región

  // Obtener fechas del primer objeto de región en newData
  const dates = newData[0].confirmed.map(day => day.date);

  // Iterar sobre las fechas
  dates.forEach((date, index) => {
    const row = [date]; // Crear una nueva fila para cada fecha
    newData.forEach(region => {
      // Obtener el valor de confirmados para la región actual en la fecha actual
      const value = region.confirmed[index] ? parseInt(region.confirmed[index].value) : 0;
      row.push(value); // Agregar el valor a la fila
    });
    chartData.addRow(row); // Agregar la fila completa a la tabla de datos
  });

  // Opciones de configuración del gráfico
  const options = {
    title: 'Crecimiento en todas las regiones', // Título del gráfico
    width: 1200, // Ancho del gráfico
    height: 800, // Alto del gráfico
    hAxis: {
      title: 'Fecha', // Título del eje x
      slantedText: true, // Texto inclinado en el eje x
      slantedTextAngle: 45 // Ángulo de inclinación del texto en el eje x
    },
    vAxis: {
      title: 'Número de Infectados', // Título del eje y
      viewWindow: {
        min: 0, // Valor mínimo del eje y
        max: 2500 // Valor máximo del eje y (ajustar según el rango de tus datos)
      }
    },
    legend: {
      position: 'right' // Posición de la leyenda
    }
  };

  // Crear instancia del gráfico de líneas de Google Visualization y dibujarlo en el elemento HTML con el ID "result"
  const chart = new google.visualization.LineChart(document.getElementById("result"));
  chart.draw(chartData, options);
}
```

Figura 5: ejercicio5.js

En el sexto ejercicio tenemos la función growthWithoutLimaCallao realiza un análisis similar a la función comparativeLineChart, pero excluye las regiones de Lima y Callao. Comienza filtrando los datos para excluir estas dos regiones y luego estructura los datos restantes en una tabla de Google Visualization. Posteriormente, crea un gráfico de líneas que muestra el crecimiento de casos confirmados para las regiones restantes a lo largo del tiempo. Se definen opciones de configuración para el gráfico, como el título y los ejes x e y. Finalmente, utiliza la biblioteca de Google Visualization para dibujar el gráfico en un elemento HTML específico identificado por el ID result".

```js
export function growthWithoutLimaCallao(data) {
  const newData = data.filter(region => region.region !== 'Lima' && region.region !== 'Callao');

  const chartData = new google.visualization.DataTable();
  chartData.addColumn('string', 'Fecha');
  newData.forEach(region => chartData.addColumn('number', region.region));

  const dates = newData[0].confirmed.map(day => day.date);
  dates.forEach((date, index) => {
    const row = [date];
    newData.forEach(region => {
      const value = region.confirmed[index] ? parseInt(region.confirmed[index].value) : 0;
      row.push(value);
    });
    chartData.addRow(row);
  });

  const options = {
    title: 'Crecimiento sin Lima y Callao',
    width: 1200,
    height: 800,
    hAxis: {
      title: 'Fecha',
      slantedText: true,
      slantedTextAngle: 45
    },
    vAxis: {
      title: 'Número de Infectados',
      viewWindow: {
        min: 0,
        max: 1000  // Ajustar según el rango de tus datos
      }
    },
    legend: {
      position: 'right'
    }
  };

  const chart = new google.visualization.LineChart(document.getElementById("result"));
  chart.draw(chartData, options);
}
```

Figura 6: ejercicio6.js

En el septimo ejercicio la función compareRegions compara el crecimiento de casos confirmados entre múltiples regiones utilizando una estructura similar a las funciones anteriores, pero sin excluir regiones específicas. Comienza creando una tabla de datos de Google Visualization con columnas para las fechas y cada región. Luego, añade filas de datos asumiendo que todas las regiones tienen las mismas fechas. Finalmente, genera un gráfico de líneas con opciones de configuración como título y ejes, mostrando la comparación en un elemento HTML con ID result". Esta función simplifica el proceso de comparación al evitar la necesidad de excluir regiones específicas en el código.

```js
// Por el momento es la misma lógica del mostrar todas las regiones sin Lima y Callao, es
// decir, estamos repitiendo código con pequeños cambios.

export function compareRegions(data) {
  const chartData = new google.visualization.DataTable();
  chartData.addColumn("string", "Fecha");

  // Añadir columnas para cada región seleccionada
  data.forEach(region => chartData.addColumn('number', region.region));

  // Suponemos que todas las regiones tienen las mismas fechas
  const dates = data[0].confirmed.map(day => day.date);

  // Añadir filas de datos
  dates.forEach((date, index) => {
    const row = [date];
    data.forEach(region => {
      const value = region.confirmed[index] ? parseInt(region.confirmed[index].value) : 0;
      row.push(value);
    });
    chartData.addRow(row);
  });

  const options = {
    title: "Comparación entre Regiones",
    width: 1200,
    height: 800,
    hAxis: {
      title: "Fecha",
      slantedText: true,
      slantedTextAngle: 45
    },
    vAxis: {
      title: "Número de Infectados",
      viewWindow: {
        min: 0,
        max: 1000
      }
    },
    legend: {
      position: "right"
    }
  };

  const chart = new google.visualization.LineChart(document.getElementById("result"));
  chart.draw(chartData, options);
}

```

Figura 7: ejercicio7.js

En el octavo ejercicio la función drawComparativeChart, genera un gráfico comparativo del crecimiento de casos de infectados en Perú, excluyendo las regiones de Lima y Callao. Filtra la información para eliminar estas dos regiones, crea una tabla de datos y prepara filas con datos de casos confirmados por fecha y región. Luego, establece opciones para el gráfico, como título y ejes, y finalmente dibuja el gráfico de líneas en un elemento HTML específico. Esta función ofrece una forma clara de comparar el avance de la enfermedad en las regiones peruanas excluyendo Lima y Callao.

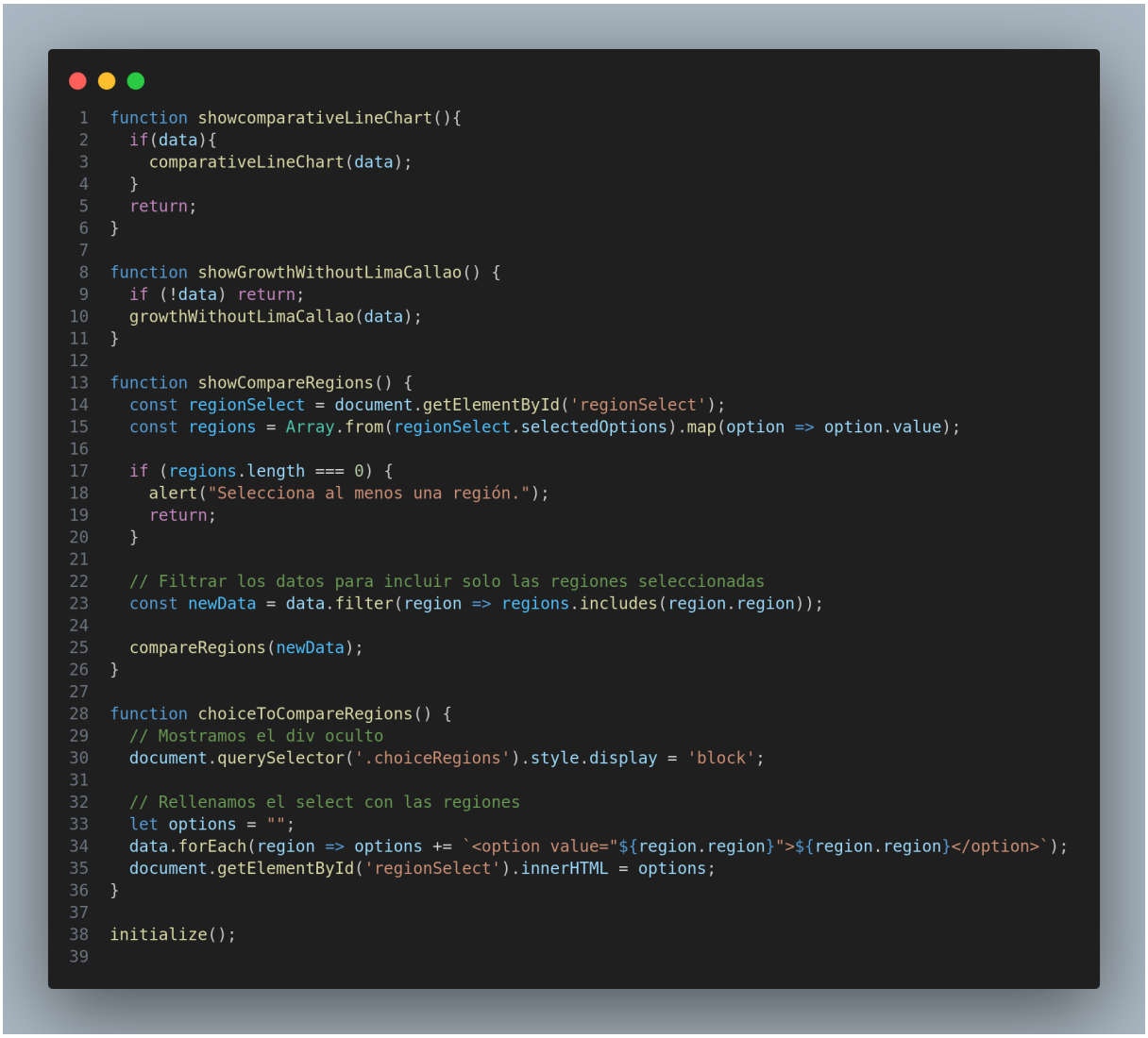
```javascript
// Autor: Jhossep Velarde
export function drawComparativeChart(info) {
  // Filter out Lima and Callao
  let filteredInfo = info.filter(region => region.region !== 'Lima' && region.region !== 'Callao');

  // Create the data table.
  let data = new google.visualization.DataTable();
  data.addColumn('string', 'Fecha');

  // Extract the unique dates
  let dates = filteredInfo[0].confirmed.map(entry => entry.date);

  // Add a column for each region
  filteredInfo.forEach(region => {
    data.addColumn('number', region.region);
  });

  // Prepare rows for the data table
  let rows = dates.map((date, index) => {
    let row = [date];
    filteredInfo.forEach(region => {
      row.push(parseInt(region.confirmed[index].value));
    });
    return row;
  });

  data.addRows(rows);

  // Set chart options
  let options = {
    title: "Crecimiento de Infectados en Perú (excepto Lima y Callao)",
    width: 1200,
    height: 800,
    hAxis: {
      title: "Fecha",
      slantedText: true,
      slantedTextAngle: 45,
    },
    vAxis: {
      title: "Número de Infectados"
    },
    chartArea: {
      width: '70%',
      height: '70%'
    }
  };

  // Instantiate and draw our chart, passing in some options.
  let chart = new google.visualization.LineChart(document.getElementById("result"));
  chart.draw(data, options);
}
```

Figura 8: ejercicio8.js

Tenemos un main donde se ejecutara el script importa ocho funciones desde archivos JavaScript externos, cada una diseñada para realizar una tarea específica en la visualización de datos sobre casos de infectados en regiones de Perú. Estas funciones abarcan desde la simple visualización de listas de regiones hasta la comparación detallada del crecimiento de casos confirmados en diferentes áreas geográficas. Al asociar cada función a un botón en la interfaz de usuario, el script permite una fácil interacción del usuario, lo que facilita la exploración y comprensión de los datos sobre la situación de la pandemia en Perú.

```js
import { listRegions } from "./ejercicio1.js";
import { totalConfirmed } from "./ejercicio2.js";
import { top10Regions } from "./ejercicio3.js";
import { arequipaInfected } from "./ejercicio4.js";
import { comparativeLineChart } from "./ejercicio5.js";
import { growthWithoutLimaCallao } from "./ejercicio6.js";
import { compareRegions } from "./ejercicio7.js";
import { drawComparativeChart } from "./ejercicio8.js";

google.charts.load('current', {'packages':['corechart']});
google.charts.setOnLoadCallback(initialize);

let data;

function loadFile() {
  let url = "http://localhost:8000/data.json";

  return fetch(url)
    .then(response => response.json())
    .then(jsonData => data = jsonData)
    .catch(error => console.log(error));
}

function initialize() {
  loadFile().then(() => {
    document.getElementById('listRegions').addEventListener('click', showlistRegions);
    document.getElementById('totalConfirmed').addEventListener('click', showtotalConfirmed);
    document.getElementById('top10Regions').addEventListener('click', showtop10Regions);
    document.getElementById('arequipaInfected').addEventListener('click', showarequipaInfected);
    document.getElementById('comparativeLineChart').addEventListener('click', showcomparativeLineChart);
    document.getElementById('growthWithoutLimaCallao').addEventListener('click', showGrowthWithoutLimaCallao);
    document.getElementById('choiceToCompareRegions').addEventListener('click', choiceToCompareRegions);
    document.getElementById('compareRegions').addEventListener('click', showCompareRegions);
    document.getElementById('dailyGrowthWithoutLimaCallao').addEventListener('click', () => drawComparativeChart(data));
  });
}
function showlistRegions(){
  if(data){
    listRegions(data);
  }
  return;

}
function showtotalConfirmed(){
  if(data){
    totalConfirmed(data);
  }
  return;
}

function showtop10Regions(){
  if(data){
    top10Regions(data);
  }
  return;
}

function showarequipaInfected(){
  if(data){
    arequipaInfected(data);
  }
  return;
}
```

Figura 9: main.js



```
function showcomparativeLineChart(){
  if(data){
    comparativeLineChart(data);
  }
  return;
}

function showGrowthWithoutLimaCallao() {
  if (!data) return;
  growthWithoutLimaCallao(data);
}

function showCompareRegions() {
  const regionSelect = document.getElementById('regionSelect');
  const regions = Array.from(regionSelect.selectedOptions).map(option => option.value);

  if (regions.length === 0) {
    alert("Selecciona al menos una región.");
    return;
  }

  // Filtrar los datos para incluir solo las regiones seleccionadas
  const newData = data.filter(region => regions.includes(region.region));

  compareRegions(newData);
}

function choiceToCompareRegions() {
  // Mostramos el div oculto
  document.querySelector('.choiceRegions').style.display = 'block';

  // Rellenamos el select con las regiones
  let options = "";
  data.forEach(region => options += `<option value="${region.region}">${region.region}</option>`);
  document.getElementById('regionSelect').innerHTML = options;
}

initialize();

```

Figura 10: main.js

La ejecucion de los 8 ejercicios en el index.HTML

El primero :

Figura 11: Poner en funcionamiento el contenedor Docker

El segundo :

Figura 12: Poner en funcionamiento el contenedor Docker

El tercero :

Figura 13: Poner en funcionamiento el contenedor Docker

El cuarto :

Figura 14: Poner en funcionamiento el contenedor Docker

El quinto :

Figura 15: Poner en funcionamiento el contenedor Docker

El sexto :

Figura 16: Poner en funcionamiento el contenedor Docker

El septimo :

Figura 17: Poner en funcionamiento el contenedor Docker

El octavo :

Figura 18: Poner en funcionamiento el contenedor Docker