Guía de ejercicios - APIs (III)

 ¡Hola! Te damos la bienvenida a esta nueva guía de estudio.

¿En qué consiste esta guía?

La siguiente guía de estudio tiene como objetivo practicar y ejercitar los contenidos que hemos visto en clase.

¡Vamos con todo!

Tabla de contenidos

¡Comencemos!

Actividad guiada 1: Muuri

Te gustaría realizar componentes dinámicos que funcionan en base al concepto drag & drop (arrastrar y soltar) Muuri es la solución. Es un plugin muy completo creado a base de jQuery que permite generar contenedores y arrastrarlos hacia otro contenedor, manipulando desde la interfaz del usuario la disposición de los elementos.

Crear dos bloques definidos, uno al lado del otro, con un texto en su interior, considerar que estos bloques se puedan arrastrar e intercambiar de posición entre ellos, partiendo del siguiente código:

```
<div class="grid">
   <div class="item">
      <div class="item-content">
        Esto puede ser cualquier texto.
      </div>
   </div>
   <div class="item">
      <div class="item-content">
        <div class="my-custom-content">
            Y aquí también!
        </div>
      </div>
   </div>
</div>
```

- ***Paso 1*:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

- ***Paso 2*:** En el index.html, escribir la estructura básica de un documento HTML, incorporar el enlace o CDN de jQuery y Muuri. Al igual que en el primer plugin que revisamos, es necesario incluirlo después que jQuery, de esta forma evitamos que la aplicación no funcione. Luego, agregar el extracto del código HTML entregado en el enunciado y enlaza el archivo externo denominado “script.js”:

```
<!DOCTYPE html>
<html lang="en">
<head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-
```

```
scale=1.0">
      <link rel="stylesheet" href="style.css">
      <title>Usando Muuri</title>
</head>
<body>
      <div class="grid">
         <div class="item">
            <div class="item-content">
               Esto puede ser cualquier texto.
            </div>
         </div>
         <div class="item">
            <div class="item-content">
               <div class="my-custom-content">
                  Y aquí también!
               </div>
            </div>
         </div>
      </div>
   <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"><
/script>
   <script
src="https://cdn.jsdelivr.net/npm/muuri@0.9.0/dist/muuri.min.js"></scrip
t>
   <script src="script.js"></script>
</body>
</html>
```

- ***Paso 3***: Crear el estilo necesario para poder contener los elementos en cuadros, agregando un color de fondo, ancho, alto, posición y como se mostrará cada uno de esos elementos.

```
.grid {
  position: relative;
}
.item {
  position: absolute;
  width: 100px;
  height: 100px;
  z-index: 1;
  background: #000;
  color: #fff;
```

```
}
.item.muuri-item-dragging {
  z-index: 3;
}
.item.muuri-item-releasing {
  z-index: 2;
}
```

- ***Paso 4:*** Queda entonces por trabajar con el archivo script.js, en donde lo primero a realizar es crear una variable que contenga un nuevo objeto de Muuri (no es un nombre al azar, la librería indica que debe ser creado así). En este objeto, se indica a cuál elemento del DOM le vamos a activar las características de Muuri, en este caso los que tengan clase `grid`, además se entrega la configuración o comportamiento que deben tener los elementos a los cuales le aplicamos la librería. Como el caso de `dragEnabled`, quien activa la opción de arrastrar y soltar, o `dragReleaseEasing`, quien indica cuál sería el tipo de efecto en el movimiento de los bloques. Las configuración del comportamiento de los elementos se encuentran mejor especificadas en la documentación de la librería, disponible en Muuri Doc.

```
var grid = new Muuri('.grid',
   {
      dragEnabled: true,
      dragReleaseEasing: 'ease',
   },
);
```

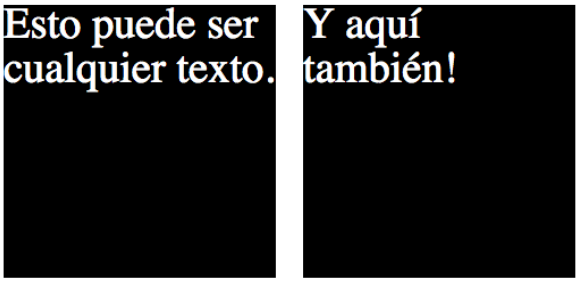

Imagen 1. Resultado de la aplicación.
Fuente: Desafío Latam

¡Manos a la obra! - Plugins

- Crea un plugin con jQuery para conectarse, traer y mostrar la información en el documento web para al siguiente api: https://jsonplaceholder.typicode.com/users.

Realiza la respectiva conexión mediante AJAX y muestra la información solo cuando un usuario haga click sobre un botón.

¡Manos a la obra! - jQuery, AJAX y CanvasJS 2

- Partiendo del ejercicio guiado "jQuery, AJAX y CanvasJS", muestra ahora la información del euro con respecto al peso chileno mediante un gráfico. La API tiene la opción de traer las tendencias del euro solamente.

¡Manos a la obra! - Muuri 2

- Construye un sitio que muestre los números del 1 al 10 y permita utilizar la función de arrastrar y soltar, que ofrece Muuri.

  Ahora es tu turno de investigar algunos plugins interesantes, la gama de posibilidades es infinita, con jQuery puedes hacer proyectos pequeños, complejos y de cualquier tema. La base siempre está en cómo aplicar las herramientas. Te invitamos a ser proactivo y aprender con estas herramientas, por tus medios. Puedes partir por la documentación oficial de jQuery y consultar el sitio de jQuery Script, que tiene mucha información relevante.

Soluciones

Plugins

1. Crea un plugin con jQuery para conectarse, traer y mostrar la información en el documento web para al siguiente api: https://jsonplaceholder.typicode.com/users. Realiza la respectiva conexión mediante AJAX y luego muestra la información solo cuando un usuario haga clic sobre un botón.

Archivo index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-
scale=1.0">
   <script
```

```
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"><
/script>
   <title>Document</title>
</head>
<body>
   <div id="miCaja"></div>
   <script src="script.js"></script>
   <script src="jQuery.datosGenerales.js"></script>
</body>
</html>
```

Archivo jQuery.datosGenerales.js

```
jQuery.fn.datosGenerales = function() {
      var element = this;
      $.ajax({
      type:"GET",
      url:"https://jsonplaceholder.typicode.com/users",
      dataType:"json",
      success: function(data) {
            console.log(data)
            data.forEach(datos => {
                  element.append(`<p>ID: ${datos.id} - Name:
${datos.name} - UserName: ${datos.username}</p>`);
            });
      }
      });
   return this;
};
```

Archivo script.js

```
$(document).ready(function(){
      $('#miCaja').datosGenerales();
});
```

jQuery, AJAX y CanvasJS 2

1. Partiendo del ejemplo realizado anteriormente, muestra ahora la información del euro con respecto al peso chileno mediante un gráfico. La API tiene la opción de traer las tendencias del euro solamente.

```
$(document).ready(function(){
      var dataPoints = [];

      var options =  {
      animationEnabled: true,
      theme: "light2",
      title: {
            text: "Tendencias del Euro"
      },
      axisX: {
            valueFormatString: "DD MMM YYYY",
      },
      axisY: {
            title: "EUR",
            titleFontSize: 24,
            includeZero: false
      },
      data: [{
            type: "spline",
            yValueFormatString: "€#,###.##",
            dataPoints: dataPoints
      }]
      };

      $.ajax({
      type:"GET",
      url:"https://mindicador.cl/api/euro",
      dataType:"json",
      success: function(datos) {
            let datosApi = datos.serie;
            console.log(datosApi);
            for (var i = 0; i < datosApi.length; i++) {
                  dataPoints.push({
                  x: new Date(datosApi[i].fecha),
                  y: datosApi[i].valor
                  });
            }
            $("#chartContainer").CanvasJSChart(options);
      },
      error: function(error) {
            console.log(error)
      }
      });
});
```

Muuri 2

1. Construye un sitio que muestre los números del 1 al 10 y permita utilizar la función de arrastrar y soltar, que ofrece Muuri.

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-
scale=1.0">
      <link rel="stylesheet" href="style.css">
      <title>Usando Muuri</title>
</head>
<body>
<div class="grid grid-1">
  <div class="item">
    <div class="item-content">1</div>
  </div>
  <div class="item">
    <div class="item-content">2</div>
  </div>
  <div class="item">
    <div class="item-content">3</div>
  </div>
  <div class="item">
    <div class="item-content">4</div>
  </div>
  <div class="item">
    <div class="item-content">5</div>
  </div>
  <div class="item">
    <div class="item-content">6</div>
  </div>
  <div class="item">
    <div class="item-content">7</div>
  </div>
  <div class="item">
    <div class="item-content">8</div>
  </div>
  <div class="item">
```

```
    <div class="item-content">9</div>
  </div>
  <div class="item">
    <div class="item-content">10</div>
  </div>
</div>
   <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"><
/script>
   <script
src="https://cdn.jsdelivr.net/npm/muuri@0.9.0/dist/muuri.min.js"></scrip
t>
   <script src="script.js"></script>
</body>
</html>
```

CSS

```
body {
  padding: 0;
}
.grid {
  position: relative;
}
.item {
  position: absolute;
  width: 200px;
  height: 200px;
  line-height: 200px;
  margin: 5px;
  z-index: 1;
}
.item.muuri-item-hidden {
  z-index: 0;
}
.item.muuri-item-releasing {
  z-index: 2;
}
.item.muuri-item-dragging {
  z-index: 3;
}
.item-content {
  position: relative;
  font-family: sans-serif;
  width: 100%;
  height: 100%;
```

```
  text-align: center;
  border: 1px solid #F44336;
  border-radius: 3px;
  font-size: 25px;
  color: #F44336;
  cursor: pointer;
}
.item.muuri-item-dragging .item-content,
.item.muuri-item-releasing .item-content {
  background: #FFCDD2;
}
```

JS

```
var grid1 = new Muuri('.grid-1', {
  dragEnabled: true,
  dragContainer: document.body,
  dragSort: function () {
    return [grid1]
  }
});
```