



UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
PROGRAM MICROSOFT IT ACADEMY
MTA-98-375

HTML 5 Application Development Fundamentals



Microsoft
IT Academy

Lección 9: Trabajando con gráficas y datos
con JavaScript



UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERÍA Y ARQUITECTURA
PROGRAM MICROSOFT IT ACADEMY
MTA-98-375

CONTENIDO:

Animaciones con JavaScript	1
Trabajando con elementos gráficos.....	5
Enviando y recibiendo información	8
Cargando y guardando archivos.....	11
Validando entradas con JavaScript	13
Manejo de Cookies.....	15
Almacenamiento local (localStorage) con JavaScript.....	17

Animaciones con JavaScript

Ejercicio 1.

Se creara una animación de un párrafo utilizando JavaScript, con métodos se ira asignando diferentes posiciones para que se observe un movimiento, utilizando recursividad se verá que el movimiento de realiza poco a poco y no de una vez.

1. Crear un proyecto en WebMatrix y agregar un nuevo documento HTML, asignarle el nombre de index.html.
2. Se creara el documento HTML, sin estilo como se muestra a continuación:

```
<body onload = "init();">
<h1>Animacion en JavaScript</h1>
<p id = "original">Se observa como me muevo en pantalla.</p>
</body>
```

Animacion en JavaScript

Se observa como me muevo en pantalla.

3. Ahora que se tiene los elementos que se desean animar, se establecerá la función init(), la cual ya es llamada por el manejador de eventos onLoad, en esta se declararan algunas variables, como también, se llamara a otro método que es el que realizara la animación.

```
function init() {
    parrafo = document.getElementById("original");
    parrafo.style.position = "absolute";
    actual = 0;
    moverParrafo();
}
```

4. Se procede a agregar el método moverParrafo(), ya que es el que se encarga de mover el párrafo, y es convocado primeramente por la función init.

```
function moverParrafo() {
    siguiente = actual + "px";
    actual += 1;
    if (actual > 300) {
        bandera = 1;
    }
    parrafo.style.left = siguiente;
    var velocidad = 18;
    setTimeout(moverParrafo, velocidad);
}
```

Con estas funciones ya se podrá observar el movimiento constante del párrafo.

Animacion en JavaScript

Se observa como me muevo en pantalla.

Animacion en JavaScript

Se observa como me muevo en pantalla.

5. Debido a que el elemento al llegar al límite establecido (300) regresa a su punto de partida, realizaremos unos cambios en las funciones para que no solo aparezca al inicio de nuevo, si no se observe esa transición también, por lo cual se observara como que el elemento rebotara de lado a lado.

```
function moverParrafo() {  
  if (bandera == 0) {  
    siguiente = actual + "px";  
    actual += 1;  
    if (actual > 300) {  
      bandera = 1;  
    }  
    parrafo.style.left = siguiente;  
    var velocidad = 18;  
    setTimeout(moverParrafo, velocidad);  
  }  
  else {  
    siguiente = actual + "px";  
    actual -= 1;  
    if (actual < 1) {  
      bandera = 0;  
    }  
    parrafo.style.left = siguiente;  
    var velocidad = 18;  
    setTimeout(moverParrafo, velocidad);  
  }  
}
```

```
function init() {  
  parrafo = document.getElementById("original");  
  parrafo.style.position = "absolute";  
  actual = 0;  
  bandera = 0;  
  moverParrafo();  
}
```

Ahora se observa que el elemento rebota de lado a lado sin detenerse.


Ejercicio 2.

Se creara un proyecto de animación, similar al anterior, pero en este se observara la interacción con el usuario para que este altere la velocidad de movimiento del elemento.

1. Crear un proyecto en WebMatrix y agregar un nuevo documento HTML, asignarle el nombre de index.html.
2. Buscar en internet una imagen, descargarla y luego copiarla y pegarla en el proyecto, asignarle el nombre de imagen y la terminación correspondiente.
3. Se creara el documento HTML, sin estilo como se muestra a continuación:

```
<body onload = "init();">
<h1>Animacion de imagen y control de velocidad</h1>
<form>
<input id = "velocidad" type = "number" value = "18" min = "5" max = "100"></input>
</form>
</img>
</body>
```

Animacion de imagen y control de velocidad



4. Teniendo los elementos que se quieren animar y los parámetros a utilizar se procede a crear el método init(), el cual es convocado cuando la página carga completamente.

```
function init() {
    imagen = document.getElementById("original");
    imagen.style.position = "absolute";
    actual = 0;
    moverImagen();
}
```

5. Luego se agrega el método moverImagen() que es el que se encarga de mover la imagen por la pantalla sin detenerse. Se observa la funcionalidad correcta de la aplicación, modificando el campo de texto asignado se observa que la velocidad aumenta si el valor disminuye y viceversa.

```
function moverImagen() {  
    sig = actual + "px";  
    actual += 1;  
    if (actual > 300) {  
        actual = 0;  
    }  
    imagen.style.left = sig;  
    var velocidad = document.getElementById("velocidad").value;  
    setTimeout(moverImagen, velocidad);  
}
```

Animacion de imagen y control de velocidad



Trabajando con elementos gráficos.

Ejercicio 1.

Se utilizarán los métodos de objetos en JavaScript para agregar elementos de tipo imagen, utilizando una función se toma ciertos parámetros de la imagen los cuales son asignados al elemento `` que va a ser agregado, si este no se puede agregar, se utiliza el atributo alternativo también.

1. Crear un proyecto en WebMatrix y agregar un nuevo documento HTML, asignarle el nombre de index.html.
2. Se creará el documento HTML, sin estilo como se muestra a continuación:

```
<body>
<form>
  <input id = "nombre" type = "text"></input>
</form>
<button onclick="mostrarImagen(276,110, 'Imagen');">Mostrar Imagen</button>
</body>
```



3. Descargue varias imágenes de su navegador, y agréguelas a su proyecto con los nombres que usted desee. Estas serán agregadas al documento.
4. Para que la aplicación funcione correctamente es necesario codificar la función `mostrarImagen()` con sus parámetros correspondientes, ancho, alto y un texto alternativo.

```
<script type="text/javascript">
  function mostrarImagen(width, height, alt) {
    var img = document.createElement("img");
    img.src = document.getElementById("nombre").value;
    img.width = width;
    img.height = height;
    img.alt = alt;
    document.body.appendChild(img);
  }
</script>
```

Ahora se puede observar el correcto funcionamiento de la aplicación web, si no se encuentra la imagen con el nombre especificado, siempre genera el recuadro con las dimensiones establecidas, pero coloca el texto alternativo en vez de la imagen. Asegúrese de escribir la terminación del tipo de dato de la imagen.



Ejercicio 2.

Se creara un proyecto para el uso de JavaScript en Canvas, con el cual se actualizara la pantalla del usuario dinámicamente cada momento.

1. Crear un proyecto en WebMatrix y agregar un nuevo documento HTML, asignarle el nombre de index.html.
2. Se creara el documento HTML, sin estilo como se muestra a continuación:

```
<body onload = "init();">
<h1>Reloj analogo utilizando JavaScript en canvas</h1>
<canvas id = "reloj" width = "200" height = "200"></canvas>
</body>
```

3. En el elemento Canvas creado, se pretende crear un reloj con actualización dinámica de las agujas, primeramente se creara la función init(), la cual establecerá ciertos elementos para iniciar a configurar nuestro elemento Canvas.

```
function init() {
    var canvas =
    document.getElementById("reloj");
    dctx = canvas.getContext('2d');
    dctx.fillStyle = "black";
    centroX = 100;
    centroY = 100;
    longitud = 100;
    mostrarAgujas();
}
```

Con esto se especifica el centro del reloj y la longitud de las agujas.

4. Al crear el init, se procederá a crear 3 funciones diferentes, una para mostrar las agujas, la cual calculara con la hora actual la dirección que debe de tener las agujas, dependiendo de los segundos, minutos y horas obtenidos, después llamara a otra función con la cual cada una de estas agujas es mostrada en pantalla, y esta otra función llamara a la función que con un cálculo matemático se encarga de dibujar ya en pantalla cada una de las agujas.

```
function mostrarAgujas() {  
    dctx.clearRect(0, 0, 200, 200);  
    var hora = new Date();  
    seg = hora.getSeconds();  
    min = hora.getMinutes() + seg / 60;  
    horas = hora.getHours() + min / 60;  
    mostrarAguja(seg / 60, 0.002);  
    mostrarAguja(min / 60, 0.005);  
    mostrarAguja(horas / 12, 0.01);  
    var vel = 1000;  
    setTimeout(mostrarAgujas, vel);  
}  
  
function mostrarAguja(fraccion, ancho) {  
    dctx.beginPath();  
    dctx.moveTo(centroX, centroY);  
    dibujarAguja(fraccion - ancho);  
    dibujarAguja(fraccion + ancho);  
    dctx.fill();  
}  
  
function dibujarAguja(fraccion) {  
    dctx.lineTo(centroX + longitud * Math.sin(2 * Math.PI * fraccion), centroY - longitud * Math.cos(2 * Math.PI * fraccion));  
}
```

Reloj analogo utilizando JavaScript en canvas



Enviando y recibiendo información

Ejercicio 1.

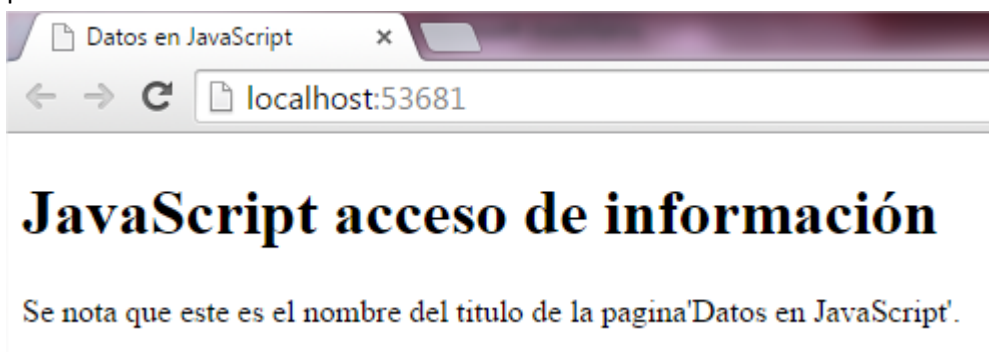
Se creara un archivo para observar el comportamiento de la página web para obtener información del navegador, utilizando diferentes métodos correspondientes a elementos en JavaScript.

1. Crear un proyecto en WebMatrix y agregar un nuevo documento HTML, asignarle el nombre de index.html.
2. Se creara el documento HTML, sin estilo como se muestra a continuación:

```
<!DOCTYPE html>

<html lang="en">
  <head>
    <meta charset="utf-8" />
    <title>Datos en JavaScript</title>
    <script type = "text/javascript">
      function init() {
        var parrafo = document.getElementById('parrafo');
        message = "Se nota que este es el nombre del titulo de la pagina'" + document.title + "'.";
        parrafo.innerHTML = message;
      }
    </script>
  </head>
  <body onload = "init();">
    <h1>JavaScript acceso de información</h1>
    <p id = "parrafo"></p>
  </body>
</html>
```

Se puede observar que la aplicación web, muestra el título de página, esto se realiza para reducir la cantidad de veces que el titulo se puede encontrar en nuestra página, par ano volver a escribirlo, accedemos a cierta información que ya se encuentra en nuestra página web y la presentamos adonde nosotros deseemos.



Ejercicio 2.

Se realizara un proyecto para el parseo de datos complicados, y se mostraran como corresponde, ya que a veces la transferencia de datos se recibe en cadenas de texto, las cuales poseen toda la información en una sola, siempre hay elementos que separan estas palabras, son los que se utilizan en este ejemplo.

1. Crear un proyecto en WebMatrix y agregar un nuevo documento HTML, asignarle el nombre de index.html.
2. Se creara el documento HTML, sin estilo como se muestra a continuación:

```
<body onload = "init();">
<h1>Parseando datos complejos</h1>
<p id = "parrafo">Bienvenido.</p>
</body>
```

3. Teniendo el elemento al cual se agregara texto, se procede a crear la función de init(), en la cual se realizara el parseo de una cadena de caracteres compleja, para esto crearemos dicha cadena e ingresaremos un valor determinado que se desea mostrar en específico.

```
datosEjemplo = "Lorena: 8.49; Maria: 9.36;Luis: 8.56; Salvador: 8.49; Andrea: 9.23; Rodrigo: 7.52";
datoAMostrar = "Lorena";
```

4. Luego creamos la función correspondiente, para que el elemento <script> se observe como a continuación.

```
<script type = "text/javascript">
    datosEjemplo = "Lorena: 8.49; Maria: 9.36;Luis: 8.56; Salvador: 8.49; Andrea: 9.23; Rodrigo: 7.52";
    datoAMostrar = "Lorena";
    function init() {
        var parrafo =
            document.getElementById("parrafo");
        var listaDatos = datosEjemplo.split(';');
        for (j = 0; j < listaDatos.length; j++) {
            elementos = listaDatos[j].split(':');
            var nombre = elementos[0].trim()
            if (nombre == datoAMostrar) {
                var message = "Dada la lista de nombres y notas '" +
                    datosEjemplo + "', el programa obtuvo la nota especifica de: " +
                    elementos[1].trim() + " que pertenece a " + nombre + ".";
                parrafo.innerHTML = message;
            }
        }
    }
</script>
```

Se observa el correcto funcionamiento de la aplicación, ya que se muestra todo el conjunto de elementos y luego se muestra el elemento específico, intenta cambiar el elemento a mostrar para ver si cambia el dato específico.



Parseando datos complejos

Dada la lista de nombres y notas 'Lorena: 8.49; Maria: 9.36; Luis: 8.56; Salvador: 8.49; Andrea: 9.23; Rodrigo: 7.52', el programa obtuvo la nota específica de: 8.49 que pertenece a Lorena.