



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 10: Utilizando JavaScript para
interfaces táctiles, uso de recursos del
sistema y más.



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

CONTENIDO:

Responder a las interfaces táctiles.....	1
Trabajando con API's adicionales en HTML5.	3
Uso de Recursos del sistema.....	9

Responder a las interfaces táctiles.

Ejercicio 1.

Se creara un proyecto para comprar las capacidades de lectura de toques de la pantalla del dispositivo adonde se está mostrando la página web, esto servirá para saber cuándo trabajar los eventos de toque en nuestra página web y hacer más apreciable la funcionalidad para el usuario.

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>
  <canvas id="canvas" width="100"
    height="100"></canvas>
  <br />
  <p>Has clic en la caja para ver si es una pantalla táctil.</p>
</body>
```

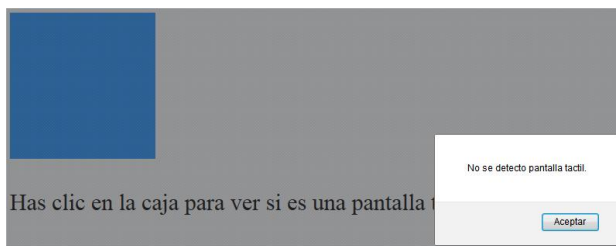


Has clic en la caja para ver si es una pantalla táctil.

3. Ahora se procederá a agregar un poco de estilo y las funciones necesarias para que la aplicación web pueda comprobar si existe o no una pantalla táctil en el dispositivo que se utilizara para ver la pagina web.

```
<style type="text/css">
#canvas{background-color: dodgerblue;}
</style>
<script type="text/javascript">
document.addEventListener("DOMContentLoaded", init, false);
function init() {
var canvas = document.getElementById("canvas");
if ("ontouchstart" in document.documentElement) {
canvas.addEventListener("touchstart", detect, false);
}
else {
canvas.addEventListener("mousedown", detect, false);
}
}
function detect() {
if ("ontouchstart" in document.documentElement) {
alert("Una pantalla tactil fue detectada!");
}
else {
alert("No se detecto pantalla tactil.");
}
}
}</script>
```

Al probarlo en una computadora normal sale lo siguiente.



Al probarlo en un dispositivo móvil se muestra lo siguiente.



Trabajando con API's adicionales en HTML5.

Ejercicio 1.

Se creara un proyecto para verificar si el navegador con el que se está abriendo la página está permitido para utilizar geolocalización, si este lo permite se mostrara la longitud y latitud.

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="IniciarLocalizacion()">
<div id="message">Buscando localizacion</div>
</body>
```

3. Teniendo los elementos a utilizar se codificara lo necesario en el JavaScript para obtener la localización del dispositivo, si este no se puede acceder se mostrara en pantalla que hubo algún inconveniente.

```
var messageDiv = document.getElementById('message');

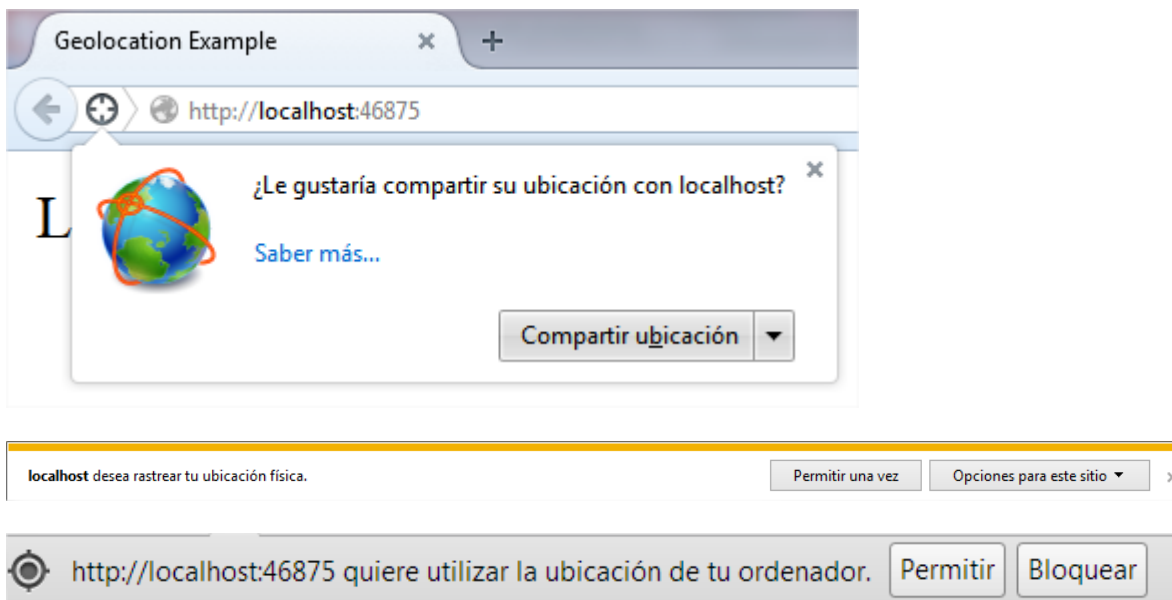
function IniciarLocalizacion() {
    var geolocation = navigator.geolocation;
    if (geolocation) {
        try {
            navigator.geolocation.getCurrentPosition(successCallback, errorCallback);
        } catch (err) {
            messageDiv.innerHTML = 'Error';
        }
    } else {
        messageDiv.innerHTML = 'Su navegador no soporta la geolocalización.';
    }
}
```

Después de la obtención se crearan los métodos de respuesta tanto como, para cuando se obtienen los datos, como para cuando no se obtienen.

```
function successCallback(location) {
    messageDiv.innerHTML = "<p>Latitud: " + location.coords.latitude + "</p>";
    messageDiv.innerHTML += "<p>Longitud: " + location.coords.longitude + "</p>";
}

function errorCallback() {
    messageDiv.innerHTML = 'Hubo un error en el acceso a su posicion';
}
```

Al probarlo en los navegadores estos pedirán permisos para acceder a su ubicación.



Cuando estos permisos son aceptados se muestra como a continuación.

Latitud: 13.7272125
Longitud: -89.2161141

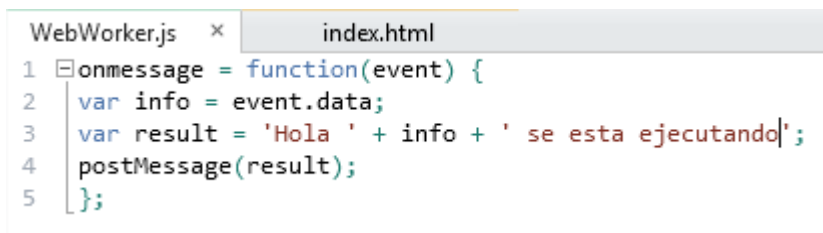
Ejercicio 2.

Se creara un ejemplo de uso de WebWorkers donde se ejecutara una alerta en diferentes hilos, se observara el uso de ellos desde otro archivo 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:

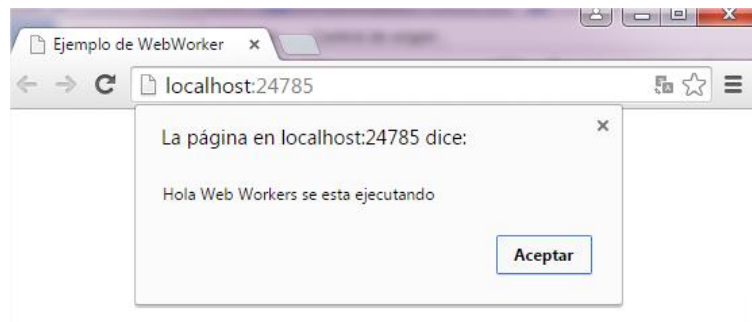
```
<!doctype html>
<html lang="en">
<head>
<script>
    var worker = new Worker('WebWorker.js');
    var info = 'Web Workers';
    worker.postMessage(info);
    worker.onmessage = function (event) {
        alert(event.data);
    };
</script>
<title>Ejemplo de WebWorker</title>
</head>
<body>
</body>
</html>
```

3. Para que la aplicación funcione correctamente es necesario agregar el archivo WebWorker.js, por lo cual se creara un nuevo archivo colocándole ese nombre, posteriormente se codificara el fragmento de código que se muestra a continuación en ese archivo.



```
WebWorker.js x index.html
1 onmessage = function(event) {
2   var info = event.data;
3   var result = 'Hola ' + info + ' se esta ejecutando';
4   postMessage(result);
5 }
```

Al tener estos archivos completos podremos ver su funcionamiento.



Ejercicio 3.

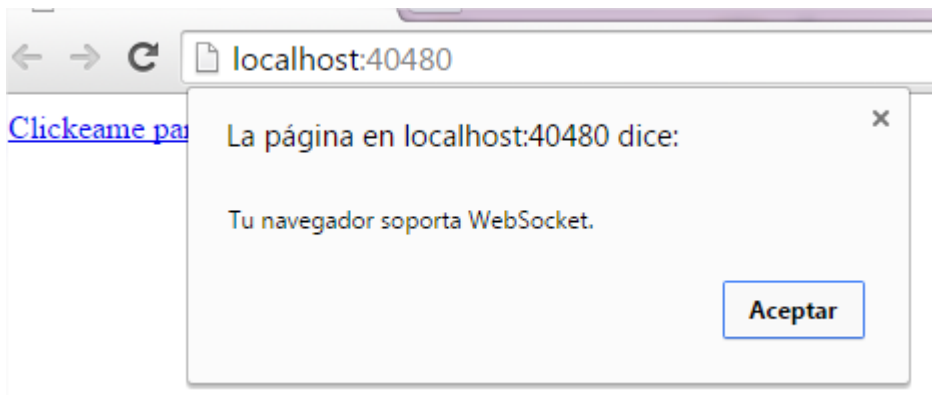
Se creara un proyecto de prueba de WebSocket, para verificar que el navegador soporte este tipo de conexiones, se hará una llamada al WebSocket local.

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:

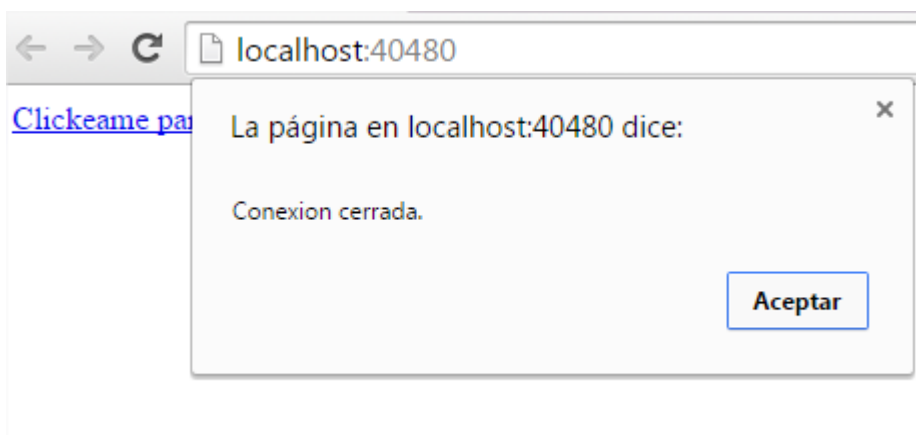
```
<!doctype html>
<html>
<head>
<script type="text/javascript">
    function WebSocketTest() {
        if ("WebSocket" in window) {
            alert("Tu navegador soporta WebSocket.");
            var socket = new
            WebSocket("ws://localhost:9998/echo");
            socket.onopen = function () {
                socket.send("COnectado");
                alert("Conectado.");
            };
            socket.onmessage = function (e) {
                var received_msg = e.data;
                alert("Mensaje recibido.");
            };
            socket.onclose = function () {
                alert("Conexion cerrada.");
            };
        }
        else {
            alert("Este navegador soporta WebSocket.");
        }
    }
</script>
</head>
<body>
<div>
<a href="javascript:WebSocketTest()">Clickeame para conectar al WebSocket</a>
</div>
</body>
</html>
```

[Clickeame para conectar al WebSocket](#)

Se podrá observar al ejecutar el programa que cada navegador mostrara si se puede o no utilizar WebSocket para realizar conexiones a servidor.



Luego de un momento se presenta otra alerta indicando que la conexión ha sido cerrada.



Ejercicio 4.

Se creara un proyecto para la verificación de soporte de la API File en el navegador que se esta utilizando.

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>
<head>
<meta charset="utf-8" />
<title>File API Browser Support Check</title>
<script>
    if (window.File && window.FileReader && window.FileList && window.Blob) {
        alert('Bien!! el navegador soporta la API File.');
```

Al ejecutar esta aplicación se presenta un alert si soporta o no la API File el navegador.

