Materia: Tecnologías para la Web.
Tema: JavaScript: Funciones.

1. Introducción

- La mejor forma de desarrollar y mantener un programa extenso es construirlo a partir de piezas pequeñas y simples, o módulos. Esta técnica se conoce como "divide y vencerás".

2. Módulos de programa en JavaScript

- Los programas en JavaScript se escriben mediante la combinación de nuevas funciones que el programador escribe, con funciones y objetos reempaquetados disponibles en JavaScript.
- El término método implica que la función pertenece a un objeto específico. Nos referimos a las funciones que pertenecen a un objeto de JavaScript específico como métodos; todas las demás se conocen como funciones.
- JavaScript cuenta con varios objetos que tienen una extensa colección de métodos para realizar cálculos matemáticos comunes, manipulaciones de cadenas, manipulaciones de fecha y hora, y manipulaciones de colecciones de datos conocidas como arreglos. Estos objetos facilitan nuestro trabajo, ya que proporcionan muchas de las herramientas que los programadores necesitan con frecuencia.
- Podemos definir funciones que realicen tareas específicas y usarlas en muchos puntos en una secuencia de comandos.
- Estas funciones se conocen como funciones definidas por el programador. Las instrucciones que definen a la función se escriben sólo una vez y se ocultan de otras funciones.
- Para invocar a las funciones se escribe el nombre de la función, seguido de un paréntesis izquierdo, después una lista separada por comas de cero o más argumentos, y luego un paréntesis derecho.
- Los métodos se llaman de la misma forma que las funciones, sólo que requieren el nombre del objeto al que pertenece el método y un punto antes del nombre del método.
- Los argumentos de las funciones pueden ser constantes, variables o expresiones.

3. Definiciones de funciones

- La instrucción `return` pasa la información del interior de una función de vuelta al punto en el programa en donde se llamó.
- Una función debe llamarse de manera explícita para que se ejecute el código en su cuerpo.
- El formato de la definición de una función es:

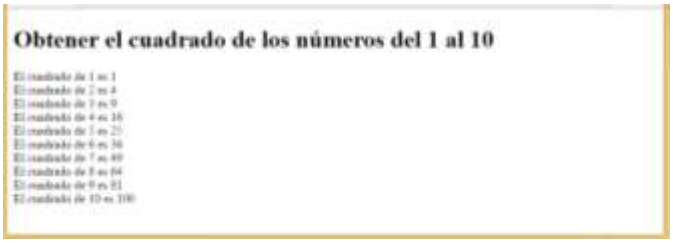
```
function nombreDeFunción( lista de parámetros){
      declaraciones e instrucciones
}
```

- Cada función debe realizar una sola tarca bien definida y d nombre de la función debe expresar esa tarca de manera efectiva. Esto promueve la reutilización de software.
- Existen tres formas de devolver el control al punto en donde se invocó a una función. Si la función no devuelve un resultado, el control regresa cuando el programa llega a la llave derecha de terminación de la función o cuando se ejecuta la instrucción `return;`. Si la función devuelve un resultado, la instrucción `return` expresión; devuelve el valor de expresión al que hizo la llamada.

EJEMPLO 1.
Archivo cuadradoent.html:

```
<!-- Función cuadrado definida por el programador. -->
<!DOCTYPE html>
<html>
      <head>
            <meta charset="utf-8" >
            <title>Una funci&oacute;n cuadrado definida por el programador</title>
            <style type="text/css">
                  p { margin: 0; }
            </style>
            <script>
```

```
                  document.writeln("<h1>Cuadrado de los n&uacute;meros del 1 al 10</h1>");
                  // obtener el cuadrado de los números del 1 al 10
                  for ( var x = 1; x <= 10; ++x )
                        document.writeln("<p>Cuadrado de " + x + ": " + cuadrado(x) +
"</p>");
                  // El cuerpo de la siguiente definición de la función cuadrado se ejecuta
                  // sólo cuando la función se llama de manera explícita.
                  function cuadrado( y ){
                        return y * y;
                  } // fin de la función cuadrado
            </script>
      </head><body></body> <!-- el elemento body est&aacute; vac&iacute;o -->
</html>
```



EJEMPLO 2.
Archivo máximo.html:

```
<!-- Función máximo definida por el programador. -->
<!DOCTYPE html>
<html>
      <head>
            <meta charset="utf-8">
            <title>M&aacute;ximo de tres valores</title>
            <style type="text/css">
                  p { margin: 0; }
            </style>
            <script>
                  var entrada1 = window.prompt( "Escriba el primer número", "0" );
                  var entrada2 = window.prompt( "Escriba el segundo número", "0" );
                  var entrada3 = window.prompt( "Escriba el tercer número", "0" );
                  var valor1 = parseFloat( entrada1 );
                  var valor2 = parseFloat( entrada2 );
                  var valor3 = parseFloat( entrada3 );
                  var valorMax = maximo( valor1, valor2, valor3 );
                  document.writeln( "<p>Primer n&uacute;mero: " + valor1 + "</p>" +
                        "<p>Segundo n&uacute;mero: " + valor2 + "</p>" +
                        "<p>Tercer n&uacute;mero: " + valor3 + "</p>" +
                        "<p>El m&aacute;ximo es: " + valorMax + "</p>" );
                  // definición de la función máximo.
                  function maximo( x, y, z ){
                        return Math.max( x, Math.max( y, z ) );
                  }
            </script>
      </head> <body></body>
</html>
```

4. Notas sobre las funciones definidas por el programador

- Todas las variables declaradas con la palabra clave `var` en definiciones de funciones son variables locales; esto significa que sólo pueden usarse en la función en la que están definidas.
- Los parámetros de una función se consideran como variables locales. Cuando se hace la llamada a una función, los argumentos en la llamada se asignan a los parámetros correspondientes en la definición de la función.
- El código que se empaqueta como función puede ejecutarse desde varias ubicaciones en un programa mediante una llamada a la función.

5. Generación de números aleatorios

- El método `random` genera un valor de punto flotante desde `0.0` hasta `1.0`, sin incluir este último.
- JavaScript puede ejecutar acciones en respuesta a la interacción del usuario con un elemento en un formulario de HTML5. Esto se conoce como manejo de eventos de GUI.
- El manejador de eventos `click` de un demento de HTML5 indica la acción a realizar cuando el usuario del documento de HTML5 haga clic en el elemento.
- En la programación controlada por eventos, el usuario interactúa con un elemento, se notifica a la secuencia de comandos del evento y la secuencia de comandos procesa el evento, la interacción del usuario con la GUI "controla" el programa. La función que se llama cuando ocurre un evento se conoce como función manejadora de eventos o manejador de eventos.
- El método `getElementById`, que recibe un `id` como argumento, encuentra el elemento de HTML5 con un atributo `id` que coincida y devuelve un objeto de JavaScript que representa ese elemento.
- El factor de escala determina el tamaño del rango. El valor de desplazamiento se agrega al resultado para determinar en dónde comienza el rango.

EJEMPLO 3.
Archivos aleatorios.html:

```
<!-- Enteros aleatorios, desplazar y escalar. -->
<!DOCTYPE html>
<html>
      <head>
            <meta charset="utf-8">
            <title>Desplazar y escalar enteros aleatorios</title>
            <style type="text/css">
                  p, ol { margin: 0; }
                  li    { display: inline; margin-right: 10px; }
            </style>
            <script>
                  var valor;
                  document.writeln( "<p>N&uacute;meros aleatorios</p><ol>" );
                  for ( var i = 1; i <= 30; ++i ){
                        valor = Math.floor ( 1 + Math.random() * 6 );
                        document.writeln( "<li> " + valor + "</li>" );
                  } // fin del for
                  document.writeln( "</ol>" );
            </script>
      </head><body></body>
</html>
```

EJEMPLO 4.

Archivo imagenes1.html:

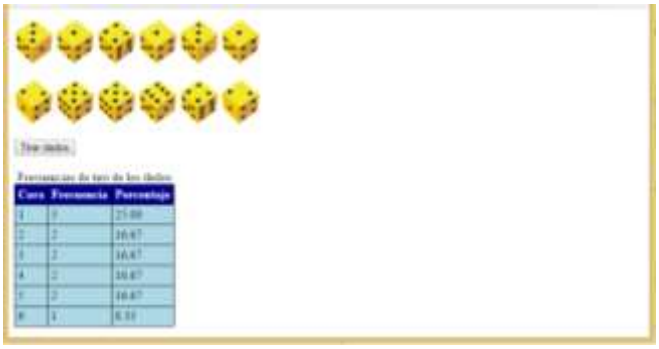
```
<!-- Generación de imágenes aleatorias de dados usando Math.random. -->
<!DOCTYPE html>
<html>
	<head>
		<meta charset="utf-8">
		<title>Im&aacute;genes aleatorias de dados</title>
		<style type="text/css">
			li { display: inline; margin-right: 10px; }
			ul { margin: 0; }
		</style>
		<script>
			// variables utilizadas para interactuar con los elementos img
			var imagenDado1;
			var imagenDado2;
			var imagenDado3;
			var imagenDado4;
			// registrar componente de escucha del botón y obtener los elementos img
			function iniciar(){
				var boton = document.getElementById( "botonTirar" );
				boton.addEventListener( "click", tirarDados, false );
				imagenDado1 = document.getElementById( "dado1" );
				imagenDado2 = document.getElementById( "dado2" );
				imagenDado3 = document.getElementById( "dado3" );
				imagenDado4 = document.getElementById( "dado4" );
			} // fin de la función iniciar
			// tirar los dados
			function tirarDados(){
				establecerImagen( imagenDado1 );
				establecerImagen( imagenDado2 );
				establecerImagen( imagenDado3 );
				establecerImagen( imagenDado4 );
			} // fin de la función tirarDados
			// seleccionar el origen de la imagen de un dado
			function establecerImagen( imgDado ){
				var valorDado = Math.floor( 1 + Math.random() * 6 );
				imgDado.setAttribute( "src", "dado" + valorDado + ".png" );
				imgDado.setAttribute( "alt", "dado de " + valorDado + " puntos" );
			} // fin de la función establecerImagen
			window.addEventListener( "load", iniciar, false );
		</script>
	</head>
	<body>
		<form action="#">
			<input id="botonTirar" type="button" value="Tirar dados">
		</form>
		<ol>
			<li><img id = "dado1" src = "blanco.png" alt = "imagen en blanco"></li>
			<li><img id = "dado2" src = "blanco.png" alt = "imagen en blanco"></li>
			<li><img id = "dado3" src = "blanco.png" alt = "imagen en blanco"></li>
			<li><img id = "dado4" src = "blanco.png" alt = "imagen en blanco"></li>
		</ol>
	</body>
</html>
```

EJEMPLO 5.
Archivo

```
<!-- Tirar 12 dados y mostrar las frecuencias. -->
<!DOCTYPE html>
<html>
      <head>
            <meta charset="utf-8">
            <title>Frecuencias al tirar los dados</title>
            <style type="text/css">
                  img   { margin-right: 10px; }
                  table { width: 200px;
                        border-collapse: collapse;
                        background-color: lightblue; }
                  table, td, th { border: 1px solid black;
                        padding: 4px;
                        margin-top: 20px; }
                  th    { text-align: left;
                        color: white;
                        background-color: darkblue; }
            </style>
            <script>
                  var frecuencia1 = 0;
                  var frecuencia2 = 0;
                  var frecuencia3 = 0;
                  var frecuencia4 = 0;
                  var frecuencia5 = 0;
                  var frecuencia6 = 0;
                  var totalDados = 0;
                  // registrar el manejador de eventos del botón
                  function iniciar(){
                        var boton = document.getElementById( "botonTirar" );
                        boton.addEventListener( "click" , tirarDados, false );
                  } // fin de la función iniciar
                  // tirar los dados
                  function tirarDados(){
                        var cara; // cara que se tiró
                        // iterar para tirar el dado 12 veces
                        for ( var i = 1; i <= 12; ++i ){
                              cara = Math.floor( 1 + Math.random() * 6 );
                              sumarTiros( cara ); // incrementa un contador de frecuencia
                              establecerImagen( i, cara ); // mostrar la imagen del dado
apropiado
                              ++totalDados; // incrementar el total
                        } // fin del ciclo de tirar dados
                        actualizarTablaFrecuencias();
                  } // fin de la función tirarDados
                  // incrementar el contador de frecuencias apropiado
                  function sumarTiros( cara ){
```

```
                  switch ( cara ){
                        case 1: ++frecuencia1; break;
                        case 2: ++frecuencia2; break;
                        case 3: ++frecuencia3; break;
                        case 4: ++frecuencia4; break;
                        case 5: ++frecuencia5; break;
                        case 6: ++frecuencia6; break;
                  } // fin de switch
            } // fin de la función sumarTiros
            // establecer origen de imagen para un dado
            function establecerImagen( numeroDado, cara ){
                  var imgDado = document.getElementById( "dado" + numeroDado );
                  imgDado.setAttribute( "src", "dado" + cara + ".png" );
                  imgDado.setAttribute( "alt" , "dado con " + cara + " punto(s)" );
            } // fin de la función establecerImagen
            // actualizar tabla de frecuencias en la página
            function actualizarTablaFrecuencias(){
                  var divTabla = document.getElementById( "divTablaFrecuencias" );
                  divTabla.innerHTML = "<table>" +
                        "<caption>Frecuencias de tiro de los dados</caption>" +
                        "<thead><th>Cara</th><th>Frecuencia</th>" +
                        "<th>Porcentaje</th></thead>" +
                        "<tbody><tr><td>1</td><td>" + frecuencia1 + "</td><td>" +
                        formatoPorcentaje(frecuencia1 / totalDados) + "</td></tr>" +
                              "<tr><td>2</td><td>" + frecuencia2 + "</td><td>" +
                        formatoPorcentaje(frecuencia2 / totalDados)+ "</td></tr>" +
                              "<tr><td>3</td><td>" + frecuencia3 + "</td><td>" +
                        formatoPorcentaje(frecuencia3 / totalDados) + "</td></tr>" +
                              "<tr><td>4</td><td>" + frecuencia4 + "</td><td>" +
                        formatoPorcentaje(frecuencia4 / totalDados) + "</td></tr>" +
                              "<tr><td>5</td><td>" + frecuencia5 + "</td><td>" +
                        formatoPorcentaje(frecuencia5 / totalDados) + "</td></tr>" +
                              "<tr><td>6</td><td>" + frecuencia6 + "</td><td>" +
                        formatoPorcentaje(frecuencia6 / totalDados) + "</td></tr>" +
                              "</tbody></table>";
            } // fin de la función actualizarTablaFrecuencias
            // aplicar formato al porcentaje
            function formatoPorcentaje( valor ){
                  valor *= 100;
                  return valor.toFixed(2);
            } // fin de la función formatoPorcentaje
            window.addEventListener( "load", iniciar, false);
      </script>
</head>
<body>
      <p>   <img id = "dado1" src = "blanco.png" alt = "imagen de dado 1">
            <img id = "dado2" src = "blanco.png" alt = "imagen de dado 2">
            <img id = "dado3" src = "blanco.png" alt = "imagen de dado 3">
            <img id = "dado4" src = "blanco.png" alt = "imagen de dado 4">
            <img id = "dado5" src = "blanco.png" alt = "imagen de dado 5">
            <img id = "dado6" src = "blanco.png" alt = "imagen de dado 6">
      <p>   <img id = "dado7" src = "blanco.png" alt = "imagen de dado 7">
            <img id = "dado8" src = "blanco.png" alt = "imagen de dado 8">
            <img id = "dado9" src = "blanco.png" alt = "imagen de dado 9">
            <img id = "dado10" src = "blanco.png" alt = "imagen de dado 10">
            <img id = "dado11" src = "blanco.png" alt = "imagen de dado 11">
            <img id = "dado12" src = "blanco.png" alt = "imagen de dado 12">
```

```
            <form action = "#">
                  <input id = "botonTirar" type = "button" value = "Tirar dados">
            </form>
            <div id = "divTablaFrecuencias"></div>
      </body>
</html>
```



6. Ejemplo: Juego de probabilidad; introducción a los elementos audio y video de HTML5

- Un elemento audio de HTML5 incrusta audio en una página Web. Al establecer el atributo `preload` en "auto" indicamos al navegador que debe considerar descargar el clip de audio, de modo que esté listo para reproducirse.
- No todos los navegadores soportan los mismos formatos de archivos de audio, pero la mayoría soportan los formatos MP3, OGG y/o WAV. Por esta razón, podemos usar elementos `source` anidados en el elemento audio para especificar las ubicaciones de un clip de audio en distintos formatos. Cada elemento `source` especifica los atributos `src` y `type`. El atributo `src` especifica la ubicación del clip de audio. El atributo `type` especifica el tipo MIME del clip.
- Cuando un navegador Web que soporta el elemento audio encuentra los elementos `source`, selecciona el primer origen de audio que represente uno de los formatos soportados por el navegador.
- Al interactuar con un elemento audio de JavaScript, podemos usar el método `play` para reproducir el clip una vez.
- La función global de JavaScript `isFinite` devuelve `true` sólo si su argumento es un número válido en el rango soportado por JavaScript.
- El elemento video de HTML5 incrusta un video en una página Web.
- El atributo `controls` del elemento video indica que el reproductor de video en el navegador debe mostrar controles que permitan al usuario controlar la reproducción de video.
- Como con el audio, no todos los navegadores soportan los mismos formatos de archivos de video, pero la mayoría soportan los formatos MP4, OGG y/o WebM. Por esta razón es posible usar elementos `source` anidados en el elemento video para especificar las ubicaciones de los formatos múltiples de un clip de video.

EJEMPLO 6.
Archivo craps.html:

```
<!-- Simulación del juego de Craps. -->
<!DOCTYPE html>
<html>
      <head>
            <meta charset="utf-8">
            <title>Simulaci&oacute;n del juego de Craps</title>
            <style type ="text/css">
                  p.rojo      { color: red }
                  img   { width: 54px; height: 54px; }
```

```
            div   { border: 5px ridge royalblue;
                        padding: 10px; width: 120px;
                        margin-bottom: 10px; }
            .punto      { margin: 0px; }
      </style>
      <script>
            // variables utilizadas para referirse a los elementos de página
            var puntoDado1Img;      // se refiere a la img de puntos del primer dado
            var puntoDado2Img;      // se refiere a la img de puntos del segundo dado
            var tiroDado1Img;       // se refiere a la img de tiro del primer dado
            var tiroDado2Img;       // se refiere a la img de tiro del segundo dado
            var mensajes;           // se refiere al párrafo "mensajes"
            var botonJugar;         // se refiere al botón lugar
            var botonTirar;         // se refiere al botón Tirar
            var tirodados;          // se refiere al audio para arrojar los dados
            // otras variables utilizadas en el programa
            var miPunto;            // punto si no gana/pierde en el primer tiro
            var valorDado1;         // valor del primer dado en el tiro actual
            var valorDado2;         // valor del segundo dado en el tiro actual
            // inicia un nuevo juego
            function iniciarJuego(){
                  // obtiene los elementos de página con los que vamos a interactuar
                  tirodados = document.getElementById( "tirodados" );
                  puntoDado1Img = document.getElementById( "puntoDado1" );
                  puntoDado2Img = document.getElementById( "puntoDado2" );
                  tiroDado1Img = document.getElementById( "tiroDado1" );
                  tiroDado2Img = document.getElementById( "tiroDado2" );
                  mensajes = document.getElementById( "mensajes" );
                  botonJugar = document.getElementById( "jugar" );
                  botonTirar = document.getElementById( "tirar" ) ;
                  // prepara la GUI
                  botonTirar.disabled = true; // deshabilitar botonTirar
                  establecerImagen( puntoDado1Img );  // restablecer para nuevo juego
                  establecerImagen( puntoDado2Img );  // restablecer para nuevo juego
                  establecerImagen( tiroDado1Img );   // restablecer para nuevo juego
                  establecerImagen( tiroDado2Img );   // restablecer para nuevo juego
                  miPunto = 0;                        // en este momento no hay punto
                  primerTiro();                       // tirar el dado para iniciar el
juego
            } // fin de la función iniciarJuego
            // realizar el primer tiro del juego
            function primerTiro(){
                  var sumaDeDados = tirarDados();     // primer tiro de los dados
                  // determinar si el usuario ganó, perdió o debe seguir tirando
                  switch (sumaDeDados){
                        case 7: case 11: // gana en el primer tiro
                              mensajes.innerHTML =
                                    "Ha ganado. Clic en Reproducir para nuevo juego.";
                              break;
                        case 2: case 3: case 12:      // pierde en el primer tiro
                              mensajes.innerHTML =
                                    "Ud pierde. Clic en Reproducir para nuevo juego.";
                              break;
                        default:          // recordar punto
                              miPunto = sumaDeDados;
                              establecerImagen ( puntoDado1Img, valorDado1 );
                              establecerImagen ( puntoDado2Img, valorDado2 );
```

```
                        mensajes.innerHTML = "Tire de nuevo";
                        botonTirar.disabled = false;  // habilitar botonTirar
                        botonJugar.disabled = true; //deshabilitar botonJugar
                        break;
                  } // fin de switch
            } // fin de la función primerTiro
            // se llama para los tiros subsiguientes de los dados
            function tirarDeNuevo(){
                  var sumaDeDados = tirarDados(); // tiro subsiguiente de los dados
                  if (sumaDeDados == miPunto){
                        mensajes.innerHTML =
                              "¡Ha ganado! Clic en Reproducir para jugar de nuevo.";
                        botonTirar.disabled = true; // deshabilitar botonTirar
                        botonJugar.disabled = false; // habilitar botonJugar
                  } // fin de if
                  else if (sumaDeDados == 7 ) { // craps
                        mensajes.innerHTML =
                              "Usted pierde. Haga clic en Jugar para jugar de nuevo.";
                        botonTirar.disabled = true; // deshabilitar botonTirar
                        botonJugar.disabled = false; // habilitar botonJugar
                  } // fin de else if
            } // fin de la función tirarDeNuevo
            // tirar los dados
            function tirarDados(){
                  tirodados.play(); // reproduce sonido de tiro de dados
                  // borra imágenes anteriores mientras se reproduce el sonido de tiro
                  valorDado1 = NaN;
                  valorDado2 = NaN;
                  mostrarDados();
                  valorDado1 = Math.floor( 1 + Math.random() * 6);
                  valorDado2 = Math.floor( 1 + Math.random() * 6);
                  return valorDado1 + valorDado2;
            } // fin de la función tirarDados
            // mostrar dados que se tiraron
            function mostrarDados(){
                  establecerImagen( tiroDado1Img, valorDado1 );
                  establecerImagen( tiroDado2Img, valorDado2 );
            } // f i n de la función mostrarDados
            // establecer origen de imagen para un dado
            function establecerImagen( imgDado, valorDado ){
                  if ( isFinite( valorDado ) )
                        imgDado.src = "dado" + valorDado + ".png";
                  else
                        imgDado.src = "blanco.png";
            } // fin de la función establecerImagen
            // registrar componentes de escucha de eventos
            function iniciar(){
                  var botonJugar = document.getElementById( "jugar" );
                  botonJugar.addEventListener( "click", iniciarJuego, false );
                  var botonTirar = document.getElementById( "tirar" );
                  botonTirar.addEventListener( "click", tirarDeNuevo, false );
                  var sonidoDados = document.getElementById( "tirodados" );
                  sonidoDados.addEventListener( "ended", mostrarDados, false );
            } // fin de la función iniciar
            window.addEventListener( "load", iniciar, false );
      </script>
</head>
```

```
	<body>
		<audio id = "tirodados" preload = "auto">
			<source src = "http://test.deitel.com/dicerolling.mp3"
				type = "audio/mpeg">
			<source src = "http://test.deitel.com/dicerolling.ogg"
				type = "audio/ogg">
			El navegador no soporta la etiqueta audio</audio>
		<p><a href = "ReglasCraps.html">Haga clic aqu&iacute; para un video corto
			que explica las reglas b&aacute;sicas de Craps</a></p>
		<div id = "puntoDiv">
			<p class = "punto">El punto es:</p>
			<img id = "puntoDado1" src = "blanco.png"
				alt = "Dado 1 de valor de punto">
			<img id = "puntoDado2" src = "blanco.png"
				alt = "Dado 2 de valor de punto">
		</div>
		<div class = "tiroDiv">
			<img id = "tiroDado1" src = "blanco.png"
				alt = "Dado 1 de valor de tiro">
			<img id = "tiroDado2" src = "blanco.png"
				alt = "Dado 2 de valor de tiro">
		</div>
		<form action = "#">
			<input id = "jugar" type = "button" value = "Jugar">
			<input id = "tirar" type = "button" value = "Tirar" >
		</form>
		<p id = "mensajes" class = "rojo">Haga clic en Jugar para iniciar el juego</p>
	</body>
</html>
```

EJEMPLO 7.
Archivo ReglasCraps.html:

```
<!-- Página Web que muestra video de las reglas básicas para el juego de dados Craps. -->
<!DOCTYPE html>
<html>
	<head>
		<meta charset = "utf-8">
		<title>Reglas de Craps</title>
	</head>
	<body>
```

```
            <p><a href = "Craps.html">Regresar al juego de Craps</a></p>
            <video controls>
                  <source src = "ReglasCraps.mp4" type = "video/mp4">
                  <source src = "ReglasCraps.webm" type = "video/webm">
                  Un jugador tira dos dados. Cada dado tiene seis caras que contienen
                  uno, dos, tres , cuatro, cinco y seis puntos, respectivamente. Se
                  calcula la suma de los puntos en las dos caras superiores. Si la
                  suma es 7 u 11 en el primer lanzamiento, el jugador gana. Si la suma
                  es 2, 3 o 12 en el 1er lanzamiento ( lo que se llama "craps" ), el jugador
                  pierde ( es decir, la "casa" gana). Si la suma es 4, 5, 6, 8, 9 o
                  10 en el 1er lanzamiento, esa suma se convierte en el "punto" del jugador.
                  Para ganar, debe seguir tirando los dados hasta que "salga su
                  punto" ( es decir, tire su valor de punto). Si tira un 7 antes de sacar
                  el punto, pierde.
            </video>
      </body>
<html>
```

7. Reglas de alcance

- Cada identificador en un programa tiene un alcance. El alcance de un identificador para una variable o función es la parte del programa en donde se puede hacer referencia al identificador.
- Las variables globales o variables a nivel de secuencia de comandos (es decir, las variables que se declaran en el elemento `head` del documento de HTML5, pueden usarse en cualquier parte de una secuencia de comandos y se dice que tienen alcance global. Así, cada función en la secuencia de comandos tiene la posibilidad de usar las variables.
- Los identificadores que se declaran dentro de una función tienen alcance de función (o local) y pueden usarse sólo en esa función. El alcance de una función comienza con la llave izquierda de apertura ({) de la función en la que se declara el identificador y termina en la llave derecha de terminación (}) de la función. Las variables locales de una función y los parámetros de una función tienen alcance de función.
- Si una variable local en una función tiene el mismo nombre que una variable global, la variable global se “oculta” del cuerpo de la función.

EJEMPLO 8.
Archivo alcance.html:

```
<!-- Ejemplo de alcance. -->
<!DOCTYPE html>
<htmtl>
      <head>
            <meta charset = "utf-8">
            <style type = "text/css">
                  p { margin: 0px; }
                  p.espacio { margin-top: lOpx; }
            </style>
            <script>
                  var salida; // almacena la cadena a mostrar
                  var x = 1; // variable global
                  function iniciar(){
                        var x = 5; // variable local para la función iniciar
                        salida = "<p>la x local en iniciar es " + x + "</p>";
                        funcionA(); // funcionA tiene la x local
                        funcionB(); // funcionA usa la variable global x
                        funcionA(); // funcionA reinicializa la x local
                        funcionB(); // la variable global x retiene su valor
                        salida += "<p class = 'espacio'>la x local en iniciar es " + x +
                              "</p>";
                        document.getElementById("resultados" ).innerHTML = salida;
```

```
                  } // fin de la función iniciar
                  function funcionA(){
                        var x = 25; // se inicializa cada vez que se llama a la funcionA
                        salida += "<p class = 'espacio'>la x local en la funcionA es " + x
+ " despu&eacute;s de entrar a funcionA</p>";
                        ++x;
                        salida += "<p>la x local en funcionA es " + x +
                        " antes de salir de funcionA</p>";
                  } // fin de funcionA
                  function funcionB(){
                        salida += "<p class='espacio'>la variable global x es " + x +
                        " al entrar a la funcionB";
                        x *= 10;
                        salida += "<p>la variable global x es " + x +
                        " al salir de la funcionB</p>";
                  } // fin de la funcionB
                  window.addEventListener( "load", iniciar, false );
            </script>
      </head>
      <body onload = "iniciar()">
            <div id = "resultados"></div>
      </body>
</html>
```

8. Funciones globales de JavaScript

- JavaScript cuenta con varias funciones globales como parte de un objeto `Global`. Este objeto contiene todas las variables globales en la secuencia de comandos.
- No necesita usar el objeto `Global` de manera directa; JavaScript lo usa por usted.

9. Recursividad

- Una función recursiva se llama a sí misma, ya sea de manera directa o indirecta a través de otra función.
- Una función recursiva sabe cómo resolver sólo el caso más simple, o caso base. Si se hace una llamada a la función con un caso base, devuelve un resultado. Si se hace una llamada a la función con un problema más complejo, sabe cómo dividir el problema en dos piezas conceptuales: una pieza que la función sepa cómo procesar y una versión más simple o pequeña del problema original.
- La función invoca (llama) a una copia nueva de sí misma para trabajar sobre cl problema más pequeño; esta invocación se conoce como llamada recursiva o paso de recursividad.
- El paso de recursividad se ejecuta mientras la llamada original a la función sigue abierta (es decir, no ha terminado de ejecutarse).
- Para que la recursividad termine en un momento dado, cada vez que la función se llama a sí misma con una versión más simple del problema original, la secuencia de problemas cada vez más pequeños debe converger en el caso base. En ese punto, la función reconoce el caso base, devuelve un resultado a la copia anterior de la función y se lleva a cabo una secuencia de instrucciones `return` hasta que la llamada a la función original regresa el resultado final al que la llamó.

EJEMPLO 9.
Archivo PruebaFactorial.html:

```
<!-- Cálculo del factorial con una función recursiva. -->
<!DOCTYPE html>
```

```
<html>
	<head>
		<meta charset = "utf-8">
		<title>Funci&oacute;n factorial recursiva</title>
		<style type = "text/css">
			p	{ margin: 0px; }
		</style>
		<script>
			var salida = ""; // almacena la salida
			// calcula los factoriales de 0 a 10
			function calcularFactoriales(){
				for ( var i = 0; i <= 10; ++i )
					salida += "<p>" + i + "! = " + factorial( i ) + "</p>";
				document.getElementById( "resultados" ).innerHTML = salida;
			} // fin de la función calcularFactoriales
			// Definición recursiva de la función factorial
			function factorial( numero ){
				if ( numero <= 1 ) // caso base
					return 1;
				else
					return numero * factorial( numero - 1 );
			} // fin de la función factorial
			window.addEventListener( "load", calcularFactoriales, false );
		</script>
	</head>
	<body>
		<h1>Factoriales de 0 a 10</h1>
		<div id = "resultados" ></div>
	</body>
</html>
```

10. Comparación entre recursividad e iteración

- Tanto la iteración como la recursividad implican repetición: la iteración usa de manera explícita una instrucción de repetición; la recursividad logra la repetición por medio de llamadas repetidas a la función.
- Tanto la iteración como la recursividad implican una prueba de terminación: la iteración termina cuando falla la condición de continuación de ciclo; la recursividad termina cuando se reconoce un caso base.
- Ambos enfoques se aproximan en forma gradual a la terminación, ya sea mediante la repetición controlada por contador o mediante la recursividad: la iteración sigue modificando un contador hasta que éste asume un valor que hace que falle la condición de continuación de ciclo; la recursividad sigue produciendo versiones más simples del problema original hasta llegar al caso base.
- Por lo general, se prefiere una solución recursiva a una iterativa cuando la recursividad refleja con más naturalidad el problema y resulta en un programa fácil de comprender y de depurar. Otra razón de seleccionar una solución recursiva es que una solución iterativa no sea clara.