Respuestas a las prácticas JavaScript

D

Respuestas de la parte 3

Capítulo 1

1. Nada. JavaScript está integrado en todos los navegadores actuales (IE, Firefox, Opera, Chrome, Netscape, etc.). Lo que sí puede ser necesario es simplemente activar la opción del uso de JavaScript dentro del navegador.
2. No, no se puede ni leer ni escribir en una base de datos directamente desde código JavaScript; estas acciones se deben realizar mediante un lenguaje del servidor, por ejemplo, ASP o PHP.
3. No, está prohibida la lectura y escrituras de archivos de todo tipo desde código JavaScript. Si se quiere realizar este tipo de operaciones en el disco del servidor se debe apelar a uno de los lenguajes del servidor, por ejemplo, ASP o PHP. Si se quiere realizar este tipo de operaciones en el cliente se debería utilizar código Java o funciones ActiveX, pero son acciones que el cliente debe autorizar.
4. Falso.
5. Verdadero.
6. Falso. Existen casos en que la compatibilidad puede ser total, por lo que la respuesta Verdadero en esos casos sería correcta, pero la respuesta correcta es Falso porque las implementaciones de JavaScript son diferentes entre los distintos navegadores, por lo que es probable que tengamos que programar algunas instrucciones específicas para las distintas versiones.

7. Falso.

Capítulo 2

1. Falso.
2. Con la etiqueta <script> y </script>
3. Falso.
4. Indefinido, número, cadena y booleano.
5. Hay una conversión automática y el resultado es 400, como si la variable var1 fuese de tipo number.
6. Esta función es adecuada para comparar una variable en una sentencia if:

```
if (var1 = 2) {
...
}
```

7. 6 iteraciones.

8. Verdadero.
9. Mediante un archivo de texto con la extensión .js que posteriormente se incluye en la página HTML con esta sintaxis:

```
<SCRIPT language="javascript" src="miArchivo.js"></SCRIPT>
```

10. Mediante este código se sabe inmediatamente cómo es el estado del navegador respecto a la gestión de JavaScript:

```
<SCRIPT language=javascript>
alert("¡OK Javascript!");
</SCRIPT>
<NOSCRIPT>
No está activado JavaScript.
</NOSCRIPT>
```

11. Correcciones de errores y mantenimiento evolutivo.
12. if, if/else y switch
13. for, while y do

14. Las condiciones que se evalúan pueden ser simples o complejas y, en este caso, se unen mediante los operadores lógicos && y || (respectivamente, Y - O)
15. No.
16. Sí, pero deben ser los últimos de la lista de parámetros.
17. Sí, una variable puede contener un tipo de dato en un momento y después se le puede asignar cualquier otro tipo de datos.
18.

```
iLínea = iLínea < 60  ?  iLínea+ : 0;
```

19. No, pero es recomendable.
20. En una línea se puede utilizar //:

```
// esto es un comentario
```

El modo /* ...*/ permite comentarios de varias líneas:

```
/* Esto

es
un comentario */
```

Capítulo 3

1. Falso, los conceptos más complejos de la teoría no están implementados pero de todas maneras se puede considerar un lenguaje orientado a objetos dado que básicamente gestiona objetos de tipo tipo intrínseco y personalizados.
2. Es una entidad que reúne datos y funciones para gestionar esos datos. En terminología POO son propiedades y métodos.
3. Verdadero. Aunque el término método se suele reservar para las funciones relacionadas a un objeto determinado. Las funciones auxiliares de un programa se denominan simplemente funciones aunque si somos estrictos y consideramos al programa como un objeto, tampoco es incorrecto denominar métodos a esa funciones auxiliares (aunque no es lo habitual).
4. Se utiliza la sintaxis del punto:

```
nombreObjeto.nombrePropiedad
```

5. Cada objeto debe tener su constructor en el que se crea su instancia en memoria y opcionalmente se inicializan sus propiedades.
6. No es obligatorio que un objeto tenga propiedades o métodos (salvo el constructor), pero lo lógico es que al menos tenga alguna de las dos cosas. El método constructor es obligatorio.
7. Utilizando el operador new.
8. La variable miEquipo2 se crea correctamente. Potencialmente podría haber un problema si se piensa que las dos variables son diferentes. Las dos variables ocupan la misma memoria por lo que no son independientes entre sí.
9. Dentro del código relacionado a un objeto, la palabra clave this representa al propio objeto.
10. Un objeto A puede tener propiedades de distintos tipos (números, cadenas, fechas, objetos personalizados de otras clases, etc.). Estos objetos están contenidos dentro del objeto A.
11. Se define un nombre de método al que se le asigna una función (por ejemplo, fnImprimir()) que se desarrolla aparte.

```
...
// definición de métodos
this.imprimir = fnImprimir;
...
```

12. Son dos objetos que tienen dos nombres diferentes pero que comparten la memoria, por ejemplo, al modificar uno también se modifica el otro.

Capítulo 4

1. Es un conjunto de datos a los que se accede con un mismo nombre de variable y utilizando un índice que puede ser numérico o una clave.
2. El objeto Array.
3. No hay un límite físico pero desde el punto de vista lógico habitualmente se utilizan matrices entre 1 y 3 dimensiones.
4. El primer elemento tiene índice 0, el último lo da la propiedad length - 1. Siendo length la cantidad total de elementos.
5. La sentencia for.
6. La sentencia for…in

7.

```
<SCRIPT language=javascript>
function listarDiagonal(mat) {
    for (var fila = 0; fila <mat.length; fila++){
         if (mat[fila][fila] != undefined) {
              document.write("Elemento mat[", fila, "][", fila,"] ",
               mat[fila][fila], "<BR />");
         }
    }
}

var mat = new Array();
mat[0] = new Array(1, 4, 5);
mat[1] = new Array(8, 6, 0);
mat[2] = new Array(1, 5, 8);
mat[3] = new Array(2, 7, 5);
listarDiagonal(mat);
</SCRIPT>
```

8. 1, 1, 1, 2, 2, 2

9. El método joint() convierte una matriz en una cadena de caracteres formada por cada uno de los elementos de la matriz separados por un carácter que se define como parámetro del método.

10. Una matriz con este contenido: 1, 2, 3.

Capítulo 5

1. Realmente no; pero sí se podría optimizar el objeto Date proporcionado por JavaScript incluyéndolo dentro de un objeto personalizado para así añadirle los métodos que nos hagan falta. De esta manera aprovechamos la implementación vigente del objeto Date y simplemente le añadimos lo que nos parezca que es necesario, por ejemplo, funciones especiales de formateo, diferencia de fechas teniendo en cuenta sólo los días comerciales, etc.

2. El milisegundo.

3. Se debe informar el número de mes real - 1 debido a que el dato se basa en 0, es decir, enero es 0.

4. La función devuelve el desfase en minutos entre la hora local y el horario del meridiano Greenwich. La hora local se toma del equipo del cliente (no del servidor de la página web).

5. Mediante el uso de la función setTimeout() o también setInterval.

Capítulo 6

1. Falso. Es un objeto String aunque no se haya creado con el operador new.
2. Simplemente concatenando una cadena vacía, la variable resultante será una cadena.

```
var miNro=5000.66;
var miCadena= miNro + ""; // ahora miCadena será "5000.66"
```

3. Con el operador + o con el método concat().
4. Con el método charAt().
5. Mediante el método substring().
6. El método slice() es idéntico al método substring(). En el método substring() se indica la posición de inicio y de fin de la subcadena, mientras que el método substr() se indica la posición de inicio y la longitud.
7. indexOf().
8. Funcionan todos de modo similar. Básicamente rodean una cadena con unas etiquetas HTML, que varían según el método utilizado. Por ejemplo, el método anchor() añade etiquetas <A> </A>.
9. JavaScript permite evaluar una cadena de caracteres como si se tratase de código fuente JavaScript. La función eval() ejecuta el código y retorna el resultado.
10. Representan un mecanismo que simplifica ciertas manipulaciones de las cadenas de texto, como verificaciones de formatos complejos (emails, número telefónicos, etc.), extracciones de subcadenas, etc.
11. test().
12. Devuelve la primera aparición del patrón definido en la expresión regular.
13. Pertenece al objeto String y opera de modo similar al método exec() del objeto RegExp, salvo que devuelve una matriz de resultados de todas las apariciones del patrón buscado.
14. Mediante una expresión regular se define un grupo de caracteres y después el método replace() permite reemplazar cualquier elemento del grupo de la expresión por un carácter determinado que se pasa como segundo parámetro del método.

Capítulo 7

1. Nos permite saber si una variable contiene un valor numérico o no.
2. Según se quiera contemplar decimales o no hay dos funciones:

```
var miCadena= "5000.66";
var miEntero=parseInt(miCadena); // miEntero será 5000
var miFloat=parseFloat(miCadena); // miFloat será 5000.6
```

3. La función isNaN() analiza todo el contenido de la variable, de principio a fin, para determinar false o true si la variable es o no es un número. No devuelve nunca un valor que no sea true o false. En cambio, las funciones parseInt() o parseFloat() devuelven la parte numérica encontrada en la parte inicial de una variable, analizándola de izquierda a derecha.
4. Hay dos modos:

   Con la función toString()

   Simplemente concatenando al número una cadena de caracteres.
5. Básicamente todas las funciones matemáticas.
6. No, el objeto se crea automáticamente.
7. Falso. El resultado es -Infinity o -Infinity.
8. Falso. Tiene también propiedades con las principales constantes matemáticas, por ejemplo, Math.PI.

Capítulo 8

1. Se utiliza la opción fullscreen para abrir la ventana popup.
2. El método reload() recarga la página en los navegadores más modernos; el siguiente es un código alternativo que funciona en todos los navegadores:

```
window.location=document.location;
```

3. Los navegadores no permiten esa opción.
4. Por razones de seguridad, no se puede evitar.
5. No, el método window.print() no permite cambiar datos de la impresora desde el código JavaScript; se deberá utilizar el cuadro de diálogo de la impresora.

6. Con el siguiente código se detecta:

```
<SCRIPT language=javascript>
  var miPopup=window.open("ventanaPopup.html",
              "Popup","width=100,height=100")
  if (!miPopup) {
    alert("¡No se puede mostrar una ventana emergente debido al
         bloqueo de popups!");
  }
</SCRIPT>
```

7. El evento onload se produce después de la carga de la página. Si se utiliza este evento en la etiqueta BODY podemos definir funciones JavaScript que se ejecutarán justo después de la carga de la página.
8. El método open() tiene un parámetro para indicar las funcionalidades de la ventana popup. Si no se especifica nada, se asumen los valores predeterminados.
9. Mediante el uso de la propiedad status de esa ventana.
10. Se utiliza la propiedad status de la ventana. Esta zona de la ventana se suele utilizar para configurar mensajes de ayuda para el usuario.
11. Se utiliza e método alert() del objeto window.
12. Se utiliza el método confirm() del objeto window.
13. El método más directo es utilizar prompt().

l4. El objeto opener es un subobjeto del objeto window que nos da acceso a la ventana padre.

15. Falso.
16. Verdadero.

Capítulo 9

1. Mediante lo siguiente:

- Nombre del archivo cookie para si identificación
- Un dominio para el que el archivo cookie tiene validez
- Una ruta que define el acceso de la página para la que el archivo cookie es legible

2. La principal limitación es que nunca podemos estar seguros si el navegador tiene habilitado el uso de las cookies.

3. El objeto document tiene una propiedad para almacenar las cookies.
4. Es un archivo de texto. No es ejecutable.
5. Ninguna.
6. Por razones de seguridad no es posible ejecutar un archivo exe desde código JavaScript. La solución es crear un enlace a un archivo exe.
7. No se puede utilizar JavaScript para estos fines, dado que JavaScript sólo tiene a lo que sucede en el cliente pero no en el servidor. Esta tarea se debe desarrollar con código del servidor (ASP, PHP, etc.)
8. El objeto navigator contiene la propiedad cookieEnabled que nos permite averiguar si el navegador tiene habilitado el uso de cookies (true) o no (false).

Capítulo 10

1. La etiqueta <img> nos permite incluir imágenes en la página. Dentro de JavaScript se utiliza la propiedad document.images.
2. El objeto Image no tiene métodos. Las propiedades nos permiten ajustar:

- alt: Texto alternativo.
- complete: Indicador de finalización de la carga.
- fileSize: Tamaño en bytes de la imagen.
- height: Altura de la imagen.
- id: Identificador.
- name: Nombre de la imagen.
- src: Dirección de la descarga de la imagen.
- width: Ancho de la imagen.

3. No, no es posible. El cliente siempre tiene medios para quedarse con una copia de la imagen.
4. Es una array de objetos de tipo Image.

Capítulo 11

1. En el atributo onFocus de la etiqueta del elemento se define un gestor del evento, que es simplemente una función JavaScript.

2. Se define un gestor para el evento onBlur de la etiqueta del elemento que se quiera controlar.
3. Sí, es posible. Se utiliza este tipo de codificación:

```
<input type="radio" onclick="función1();función2();función3()"  ...>
```

4. Sí, se genera el evento onmouseover.
5. Se define un gestor para el evento onChange de la etiqueta del elemento que se quiera controlar.
6. Se define un gestor para el evento onSelect de la etiqueta del elemento que se quiera controlar una selección de texto.

Capítulo 12

1. Los botones de función propios del navegador no se pueden gestionar desde JavaScript para que el navegador no pierda el control sobre sus teclas.
2. Por ejemplo:

```
document.onmousemove = obtenerPosMouse;
```

   También se puede definir la función a nivel de una etiqueta HTML:

```
<A href="javascript:void()" onmouseover="miFunción()"></A>
```

3. Verdadero.
4. Falso. Devuelve true si se acepta y false si se cancela o si se cierra la ventana con el botón Cerrar.
5. Sí, utilizando el evento onmousedown y analizando las propiedades del evento para diferenciar el botón principal del botón secundario.
6. onkeypress.
7. Varía según el tipo de navegador:

   IE: String.fromCharCode(event.keyCode)

   Netscape: String.fromCharCode(e.which)

   Firefox:String.fromCharCode(e.charCode)
8. HTML nos permite la reproducción de audio y sonido mediante el uso de la etiqueta <EMBED></EMBED>.

Capítulo 13

1. No es posible debido a que todos los navegadores poseen la opción Ver código fuente (o similar).
2. El valor del elemento elegido (podría ser más de uno en el caso de selección múltiple) se obtiene analizando el array options y verificando si la propiedad selected es true, por ejemplo:

```
if ( documento.formname.selectname.options[i].selected == true) {
...
}
```

3. Se verifica el atributo checked del elemento.
4. Es un área de texto normal pero no es visible.
5. El elemento de tipo hidden es invisible, en cambio, el elemento con el atributo disabled es visible de modo ensombrecido y no modificable por el usuario.
6. Recorriendo la matriz elements que pertenece al objeto document.nombreformulario.
7. Se verifica la propiedad checked de la matriz de elementos que forma el grupo de opciones.
8. La etiqueta textarea nos permite introducir cadenas de texto largas de varias líneas.
9. Hay dos modos, por índice o por nombre:

   Mediante un acceso por índice al array forms:

```
document.forms[0]
```

   También se puede emplear la siguiente sintaxis (suponiendo que el formulario HTML se denomina formCompra):

```
document.formCompra
```

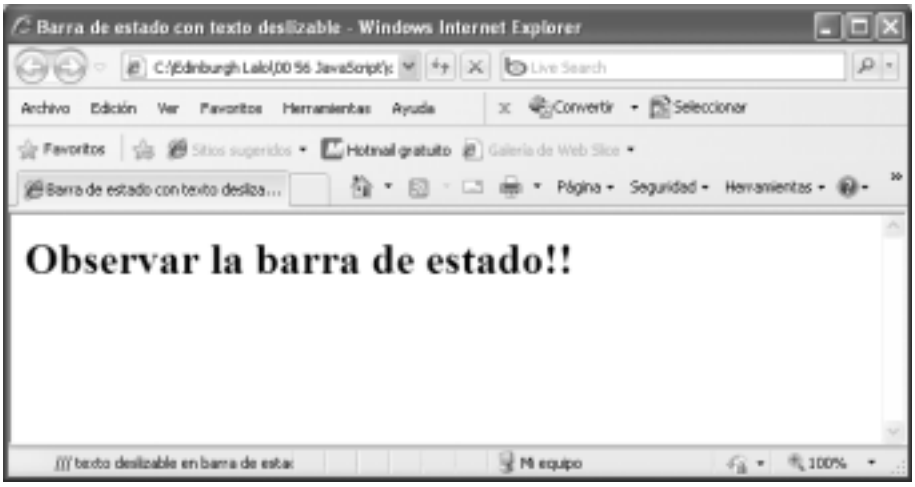
Capítulo 14

1. DHTML es la base de la manipulación dinámica de los componentes de una página, para conseguir cambios del aspecto estético, de las posiciones y del contenido.

2. DOM es el modelo de objeto documento que permite gestionar los componentes de una página web como una estructura jerárquica.
3. Es una herramienta muy interesante que nos permite analizar y modificar los nodos del modelo DOM lo que nos ayuda en el diseño de las aplicaciones web.
4. El objeto document es el elemento fundamental de la programación DHMTL, dado que es el que nos permite acceder y modificar todos los elementos de una página.
5. getElementById().
6. La clase style nos permite modificar dinámicamente la presentación de los elementos de un documento.
7. Mediante la propiedad className del elemento.
8. Visibilidad, posicionamiento, dimensionamiento, orden z, color, etc.
9. Texto deslizante: ¡Obviamente, no hay una única solución!

```
<html>
<head><title> Barra de estado con texto deslizable </title>

<script type="text/JavaScript">

// Variable globales
// accesibles desde todas
// las funciones
var iDesplaz = 0;
var txtMensaje = "";

////////////////////////////////////////
// Esta función se llama 1 vez,
// al cargarse la página
// define el mensaje que se mostrará
// en la barra de estado
// y pone en marcha el proceso iterativo
////////////////////////////////////////
function crearMensaje() {
    // se crea un relleno inicial
    // con 80 espacios
    txtMensaje = ""
    for (i = 1; i <= 80; i++) {
         txtMensaje += " ";
    }
    txtMensaje += "/// texto deslizable en barra de estado ///";
    // comienza el proceso iterativo
    hacerDeslizamiento();
}
////////////////////////////////////////
```

```
// Esta función se llama automáticamente
// gracias a la función setTimeout()
////////////////////////////////////////
function hacerDeslizamiento() {
    // cambia el mensaje
    // en cada ejecución de la función
    // cambia la posición de inicio de la cadena
    var txtDisplay = txtMensaje.substring(iDesplaz,
         txtMensaje.length);

    // coloca el texto en la barra de estado
    window.status=txtDisplay;

    iDesplaz++;
    // si se llega al final se empieza desde 1
    if (iDesplaz > txtMensaje.length){ iDesplaz = 1;}

    // pone en marcha el timer
    // llamada automatizada cada 200 ms
    setTimeout("hacerDeslizamiento()", 200);
}
</script>
</head>
<body onLoad="crearMensaje()">
<H1> Observar la barra de estado!!</H1>
</body>
</html>
```



Capítulo 15

1. El modelo de objeto documento, DOM, es un estándar para representar el documento independientemente del navegador que se utilice.

2. Falso. DOM se centraliza en el documento y el modelo BOM incluye acceso a todas las áreas del navegador. Otra gran diferencia es que el modelo BOM es más específico para cada navegador mientras que DOM pretende ser totalmente independiente del navegador.
3. Es una estructura jerárquica para visualizar el documento. Para DOM todo el documento es un nodo documento y cada uno de los elementos dependientes son otros tipos de nodos.
4. Es el nodo raíz y es de tipo Document.
5. Cada etiqueta HTML es un nodo de tipo elemento y cada uno de sus atributos es un nodo de tipo atributo. El texto contenido en los elementos HTML son nodos de tipo texto.
6. Mediante la propiedad childNodes() del objeto Node; el tipo de la propiedad es NodeList().
7. Mediante la propiedad parentNode.
8. insertBefore().
9. getElementByID(*id1*), getElementsByTagName(*etiq*), getElementsByTagNameNS().
10. setAttribute().

Capítulo 16

1. HTML (o XHTML) y hojas de estilos en cascada (CSS), DOM (*Document Object Model*), Objeto XMLHttpRequest, y XML, XSLT o JSON. Ajax requiere el uso de un lenguaje que se ejecuta en el cliente, por ejemplo, JavaScript.
2. Significa que el proceso no se detiene esperando la respuesta.
3. Falso. Puede funcionar en los dos modos.
4. Ajax intenta resolver dos problemas del esquema clásico:

- El tiempo de espera en cada interacción del usuario.
- No se hacen actualizaciones parciales de la página.

5. Las principales limitaciones son las siguientes:

- Es posible que el navegador no acepte el proceso de JavaScript, y por lo tanto de Ajax.

- El uso de Ajax sobrecarga al servidor.
- El código Ajax tampoco es tomado en cuenta por los robots de los motores de búsqueda para la indexación de los contenidos.
- Las peticiones Ajax pueden cambiar el aspecto de una página original de modo total pero esos cambios no quedan registrados en el historial (se modifica la funcionalidad de la tecla Atrás).

6. El objeto XMLHttpRequest permite enviar y recibir documentos XML en procesos de segundo plano.
7. El uso asíncrono requiere que el código en el navegador se pueda "enterar" de la llegada de la respuesta del servidor para realizar las acciones previstas con los datos recibidos.
8. Mediante la definición de una función gestora de la respuesta en onreadystatechange.