Programación Avanzada
Grado en Ingeniería Informática en Sistemas de Información - Curso 2017/2018

EPD 8: JavaScript avanzado

La entrega del trabajo se hará a través de la tarea correspondiente en el Campus Virtual. Pasado el límite de entrega se aceptará el envío del trabajo, con una penalización de 2 puntos sobre 10 de la calificación por cada hora o fracción de retraso. La entrega consistirá en un único fichero comprimido en formato ZIP cuyo nombre deberá ser de la forma *equipoXX.zip*, donde *XX* serán dos cifras que indicará el número del equipo. Por ejemplo, equipo07.zip. Este fichero contendrá una serie de carpetas cuyo nombre deberá ser de la forma *ejY* o *pZ*, donde *Y* y *Z* representan, respectivamente, el número de cada ejercicio o problema del presente guión. Dentro de dichas carpetas se incluirán exclusivamente los archivos necesarios en la resolución del correspondiente ejercicio o problema. Las rutas de los ficheros empleados serán relativas, a fin de que las resoluciones a los ejercicios y problemas puedan ser examinadas en cualquier equipo. Cualquier entrega que no cumpla las reglas de nombrado, el formato de compresión del archivo o el contenido de los archivos del mismo, será penalizada con 2 puntos sobre 10 por cada incumplimiento.

Objetivos

- Tratar eventos en JavaScript.
- Manejar objetos propios de JavaScript.

Conceptos

1. Eventos

JavaScript aplica el modelo de programación basada en eventos. En este tipo de paradigma de programación, los *scripts* esperan la interacción del usuario para comenzar el procesamiento y generar un resultado.

La interacción que realiza el usuario se transmite a los programas mediante una serie de llamadas que denominamos genéricamente como *eventos*. JavaScript define numerosos eventos que permiten una interacción completa entre el usuario y las páginas/aplicaciones web. Por ejemplo, la pulsación de una tecla constituye un evento, así como pinchar o mover el ratón, seleccionar un elemento de un formulario, redimensionar la ventana del navegador, etc.

JavaScript permite asignar un código a cada uno de los eventos. De esta forma, cuando se produce cualquier evento, JavaScript ejecuta dicho código. Como buena práctica, el código asociado a un evento se limita, prácticamente, a invocar una función, que es el lugar donde realmente realizaremos el procesamiento necesario para tratar el evento, consiguiendo así un código más legible. Este tipo de funciones se denominan "*event handlers*" o en castellano "manejadores de eventos".

Puede encontrar más información en [1], capitulo 6, pág. 167-197 y en [2] capítulo 6 pág. 63-80.

2. *Objetos*

JavaScript es un lenguaje interpretado de secuencias de comandos (*scripts*) orientado a prototipos. La necesidad de un intérprete para ejecutarse (habitualmente en un navegador web, aunque también hay usos del mismo en servidor) limita su funcionalidad. Aunque tiene menos capacidades que los lenguajes de altas prestaciones, como por ejemplo C++ o Java, JavaScript ofrece unas características básicas asimilables a las de un lenguaje orientado a objetos. De esta forma, se puede decir que JavaScript es un lenguaje *basado* en objetos (que no *orientado* a objetos).

En JavaScript, los objetos pueden ser vistos como colecciones de propiedades y métodos. El lenguaje sólo admite tres tipos de objetos:

1. Objetos intrínsecos (Array, Boolean, Date, Function, Math, Number, Object y String).
2. Objetos del navegador (window, document, form, frame, history, location).
3. Objetos creados por el usuario.

Los primeros dos tipos forman parte de la implementación del lenguaje, es decir, ya están construidos y disponibles para que el programador los utilice. El tercer tipo lo conforman los objetos creados por el programador, para dar solución a sus necesidades.

Puede encontrar más información sobre el modelo de objetos en JavaScript en [1] capitulo 5 pág. 141-165, capitulo 7 pág. 199-210 y en [2] capitulo 5, pág. 53-62.

Bibliografía Básica

1. JavaScript Professional Projects. P. Hatcher y J. Gosney. Course Technolgy, 2003. http://0-site.ebrary.com.athenea.upo.es/lib/bupo/Doc?id=10064361

2. Introducción a JavaScript. Javier Equíluz Pérez. http://librosweb.es/libro/javascript/

Experimentos

E1. (30 min.) En el presente experimento estudiaremos algunos ejemplos de código JavaScript para el manejo de eventos. Para ello, emplearemos un *script* cuya función es realizar una cuenta atrás. El usuario ha de introducir los valores de las horas, minutos y segundos correctamente para poder empezar dicha cuenta. Por tanto, podemos dividir este experimento en dos partes:

a) Validación de los campos a introducir. Dichos campos han de ser validados de modo que sólo puedan introducirse valores numéricos y, en caso de que el número de segundos o minutos sea mayor de 60, se actualicen los campos correspondientes. Las funciones del experimento correspondiente con esta parte son:

- esNumero(e): dado el evento *e*, comprueba si la tecla presionada es o no un número. En ese caso devuelve un valor *true*, y *false* en caso contrario.
- ActualizaHoras(): evita que el campo correspondiente a las horas esté vacío (cadena vacía).
- ActualizaMinutos(): comprueba que el rango introducido sea correcto y, en caso contrario (> 59), actualiza el campo de las horas.
- ActualizaSegundos(): comprueba que el rango introducido sea correcto y, en caso contrario (> 59), actualiza los campos correspondientes (minutos y horas).

b) Cronómetro. En este caso se hace uso de las funciones *setTimeout* para actualizar la cuenta y *clearTimeout* para cancelar la cuenta iniciada con la función anterior. Las funciones que corresponden al cronómetro son:

- detenerCrono(): para el cronómetro llamando a la función de JavaScript *clearTimeout*.
- IniciarCrono(): comienza la cuenta atrás y llama a *actualizaCrono*.
- ActualizaCrono(): función que actualiza el cronómetro. Para ello, usa la función de JavaScript *setTimeout* para que ejecute *ActualizaCrono* una décima más tarde. De esta forma, el cronómetro se irá actualizando diez veces por segundo.

Encontrará el código del experimento en el material adicional a este guión.

E2. (30 min.) Cree un formulario de *login* que impida enviar los datos al servidor hasta que ambos campos (usuario y contraseña) se encuentren rellenos ¿Puede reutilizar la misma función para comprobar ambos campos o necesita hacer una función para cada uno de ellos? ¿Qué es fundamental para poder reutilizar las funciones de comprobación? ¿Cómo se comporta la propagación de eventos en este caso? En base a lo anterior ¿Cómo podemos conseguir que el formulario no se envíe si no se cumplen las restricciones establecidas? ¿Por qué se debe procesar ese tipo de restricciones en el cliente, qué ventajas tiene? ¿Es suficiente con implementar estas restricciones en JavaScript o se debe hacer algo en el lado del servidor?

Ejercicios

EJ1. (30 min.) Cree una página, empleando JavaScript, que funcione como un lector de noticias. En ella, se incluirá una serie de titulares resumidos de noticias a modo de lista en la parte inferior de la página y una noticia ampliada en la parte superior (con el texto completo de la noticia). La noticia ampliada irá avanzando de manera automática cada 10 segundos recorriendo todas las noticias cuyos titulares se muestran en la parte inferior. Además, el usuario podrá pinchar directamente en uno de los titulares, mostrándose entonces su correspondiente noticia ampliada en la parte superior de la página. Al comenzar la ejecución la noticia que se mostrará ampliada será la primera. Utilice una matriz de dos columnas de tipo cadena: la primera con los titulares y la segunda con el texto completo de las noticias.

EJ2. (25 min.) Cree una aplicación que nos ayudará a invertir en bolsa. Para ello, la web permitirá elegir el número de compañías en las que desea invertir. Ésta simulará el precio de las acciones de estas compañías en los últimos 20 días, dando a cada día un valor aleatorio entre 1 y 50. Deberá mostrar una tabla en la cual cada fila represente a una compañía. En el primer valor de cada fila se mostrará el nombre de cada compañía, seguido del primer precio de la acción. Para los siguientes valores deberá mostrarse una imagen con una flecha hacia arriba si ha subido con respecto al valor anterior, una flecha para abajo si ha bajado, o una imagen de "=" si se ha mantenido el precio. Por ejemplo:

Compañía Y	27	↑	↑	↓	↓	↓	↑	↓	↑	↑	↓	=	↑	=	↑	↑	↓	=	↑	=

Además la aplicación mostrará en otra tabla ordenada las compañías según el balance que hayan tenido las acciones en los últimos 5 días. Para realizar esto, deberá analizar los últimos 5 valores de cada compañía y ordenar según se haya mantenido o subido más el precio. Ejemplo:

Compañía Y (últimos 5 días): ↑↓=↑=
Compañía P (últimos 5 días): ↓=↓↑=
(Situará antes a la compañía 1 que a la compañía 2.)

Problemas

P1. (25 min.) Realice una página web que nos diga donde podremos ver una película en los cines de la ciudad. Para ello la página mostrará:

- Un listado para cada cine con las películas y las horas a las que se proyectan.
- Un desplegable con todas las películas que se pueden ver, para que el usuario elija la película.
- Un campo de texto donde el usuario introducirá la hora a partir de la cual está disponible para ver la película.

De esta forma, la página resaltará con algún estilo en cada cine las sesiones a partir de las cuales el usuario podría ir a ver la película. Dicho campo sólo permitirá la introducción de valores numéricos, por lo que la hora se deberá escribir como si fuera un único número (por ejemplo, para las 12:34 habrá que introducir las 1234). Si lo que se intenta escribir es una letra o símbolo no se aceptará la pulsación de dicha tecla y ésta no tendrá efecto (no se escribirá el carácter). Para ello, haga uso del método charCodeAt() y compruebe el código ASCII de cada carácter y elimine el mismo en caso de que no se sigan las reglas establecidas.

P2. (30 min.) Cree un código en JavaScript que, al ser incluido en una página, abra una ventana nueva donde indique la siguiente información utilizando las funciones DOM:

1. El número medio de palabras por párrafo de la página.
2. Listado con los textos alternativos de las imágenes que contiene la página.
3. Mostrar los dos primeros párrafos en negrita.
4. Mostrar la URL del primer hiperenlace de la página.

Aplique el código sobre una página extraída de alguno de los artículos incluidos en la tabla de contenidos que se expone en esta web: https://en.wikibooks.org/wiki/LaTeX.

P3. (90 min.) Implementar una página para crear tableros de "hundir la flota", en la que el usuario pueda configurar un tablero de casillas con agua y barcos. Para ello, habrá que crear una página con un tablero de 25x25 casillas (en una tabla HTML) en las cuales, por defecto, aparecerán imágenes de agua. El usuario dispondrá en la parte inferior de la página de dos imágenes: una de agua y otra de barco. Él podrá hacer click en una de estas imágenes y seleccionar una casilla del tablero para introducir esa imagen. Recuerde que un barco en el juego de hundir la flota puede ocupar varias casillas porque éste sea más largo. No se considerarán barcos en diagonal, sólo barcos que se extienden vertical u horizontalmente en el tablero. Cada vez que el usuario agregue imágenes a las casillas del tablero, se deberá comprobar que no existen barcos demasiado largos (de más de N casillas consecutivas). Para ello, en la parte inferior de la página habrá un deslizador tipo HTML5 para indicar la longitud máxima (N) de los barcos en el juego (por defecto, el deslizador valdrá 5). El objetivo de la página será introducir 6 barcos en el tablero. Por tanto, cada vez que el usuario añada una imagen al tablero, también se comprobará que ya se hayan introducido los 6 barcos. Cuando esto ocurra, se mostrará un aviso tipo alert indicando tal circunstancia. Deberá almacenar el tablero en una matriz de JavaScript y hacer los recorridos y búsquedas oportunas para identificar automáticamente dónde están los barcos y por tanto saber cuántos hay y que longitud tienen. Implemente todas las funciones en JavaScript que considere para descomponer adecuadamente las funcionalidades requeridas.

P4. (20 min.) Realice una aplicación para jugar al ahorcado. Para ello, el programa pedirá al usuario una palabra secreta, y mostrará una línea por cada letra de la palabra, con a su vez cada letra invisible para el usuario. Así otro usuario que juegue sabrá el número de letras de la palabra a adivinar. De igual forma, la aplicación contará con un campo donde el usuario podrá ir introduciendo letras. Si alguna de ellas estuviera contenida en la palabra, se pondrá visible en su posición. Si fallara, se mostrará un contador con el número de fallos registrados. El juego terminará si el usuario acaba adivinando la palabra o bien comete 5 fallos. Para facilitar la solución al juego, la página contará con un apartado donde estará oculta la palabra a adivinar. Al pasar el ratón por encima ésta se mostrará, volviéndose a ocultar cuando se quite el puntero de encima. Ejemplo: (palabra a adivinar: murciélago)

Pase el ratón por aquí para ver la solución:	M _ _ _ _ _ _ _ _ _ O	Número de fallos: 2

P5. (40 min.) Crearemos un código JavaScript que permita crear una página en la que se mostrará una tabla de imágenes aleatorias organizadas por combinaciones de categorías de imagen. En concreto, la tabla tendrá dos columnas e inicialmente sin ninguna fila. Las dos columnas corresponderán con estas dos categorías fijas: "man" y "woman". Cada fila de la tabla contendrá posteriormente imágenes cuyas categorías serán tanto una que desee el usuario como cada una de las dos categorías fijas. Para ello, haremos uso del servicio de https://source.unsplash.com/featured/. Este servicio, ya probado en el problema P2 de la EPD5, devolverá en cada llamada una imagen aleatoria con las categorías especificadas mediante parámetros GET. La página incluirá un cuadro de texto para que el usuario escriba la categoría que desee (por ejemplo, la categoría "leg") y un botón que permite añadir una nueva fila a la tabla de imágenes con dos imágenes cuyas categorías serán: "man,leg" y "woman,leg", respectivamente. Además, habrá un segundo botón que permitirá eliminar la última fila de la tabla de imágenes. Para evitar problemas de cacheo de las imágenes, añada a la URL un parámetro *get* generado a partir de un sello de tiempo (P.ej. https://source.unsplash.com/featured/?man,arm&123456789). Utilice las funciones DOM para llevar a cabo las funcionalidades necesarias.

P6. (55 min.) Realice una aplicación de recogida de los datos personales de las familias españolas. Para ello, primero se pedirá en un campo el número de miembros y se creará dinámicamente tantos sub-formularios como miembros se hayan especificado. Para cada miembro se pedirá una serie de datos, cada uno de ellos con unas restricciones.

- Nombre: Podrá contener sólo letras. El campo no puede estar vacío.
- Apellidos: Podrá contener sólo letras. El campo no puede estar vacío.
- Altura (en cm) : Campo numérico, comprendido entre 100 y 299. No puede estar vacío.
- Estado civil: lista desplegable que contendrá los valores "--Seleccione uno--", "Soltero", "Casado", "Divorciado", "Viudo". Deberá haber elegido al menos alguno de los 4 últimos.
- Página web privada: El formato aceptado será del tipo: www.paginaWeb.com/cadena. No puede estar vacío.
- Número de teléfono: Campo numérico para guardar el móvil del usuario. Debe empezar por 6, de longitud 9 y no puede estar vacío ni contener letras.
- Sexo: Conjunto de botones de radio que dará a elegir entre "Mujer" u "Hombre". No podrá dejar sin escoger alguna de las dos opciones.

La página asegurará que los campos están completos de forma adecuada. La validación del formulario se realizará en el momento del envío, mostrándose los mensajes adecuados (incluyendo instrucciones de formato) en caso de error y cancelando el envío del formulario. Organice el código de forma tal que éste sea modular, dedicando una función por cada tipo de validación.

Ampliación de Bibliografía

1. The book of JavaScript a practical guide to interactive Web pages. Dave Thau. No Starch Press, 2006. http://0-site.ebrary.com.athenea.upo.es/lib/bupo/Doc?id=10158196
2. Learn JavaScript in a weekend.Jerry Lee Ford.Premier Press, 2004. http://0-site.ebrary.com.athenea.upo.es/lib/bupo/Doc?id=10054331

Datos de la Práctica

Autor del documento: Federico Divina (Noviembre 2007).

Revisiones:

1. Miguel García Torres (Noviembre 2008).
2. Federico Divina (Abril 2010).
3. Carlos D. Barranco (Mayo 2010). Revisión del texto y de formato.
4. Francisco A. Gómez Vela (Abril 2011). Revisión de texto, formato, enlaces y de enunciado en los ejercicios 3 y 4 con sus respectivos cambios en el material a entregar. Cambio de enunciado del problema 2, incluyendo un nuevo material. Cambio del problema 3 para que aporte mas funcionalidad.
5. Carlos D. Barranco (Mayo 2012). Revisión del texto de la sección conceptos, solventar el fallo en la referencia bibliográfica [1]. y ajustar la referencia que se hace a los números de página de la referencia [2]. Nuevo enunciado para los ejercicios 1 y 2, así como de los problemas 1 y 4. Finalmente, se ha mejorado el formato del ejercicio 4 y la redacción del problema 3.
6. Carlos D. Barranco (Diciembre 2013). Arreglos en el texto de la sección de conceptos y en el experimento 1. Renovación de los ejercicios 3 y 4 y de los problemas 2 y 3.
7. Carlos D. Barranco (Noviembre 2014). Mejora del texto de la sección de conceptos. Inclusión del experimento 2. Reajuste de tiempos de experimentos, ejercicios y problemas. Reorganización de los ejercicios.
8. Pablo Soler (Noviembre 2015). Revisión texto envío de trabajos. Cambio en los enunciados de los ejercicios 1 y 2, así como de los problemas 1 y 3. Reajuste de tiempos en los ejercicios.
9. Carlos D. Barranco (Noviembre 2015): Mejoras en texto y formato de la sección conceptos y experimento 1. Detallado de las cuestiones a abordar en el experimento 2. Mejoras en los texto de ampliación de bibliografía.
10. Ricardo – León Talavera Llames (Noviembre 2016): Modificación del enunciado del ejercicio 2, así como de los problemas 1, 4 y 6. Modificación de los tiempos de cada ejercicio.
11. Carlos D. Barranco (Noviembre 2016): Mejora en la redacción de la sección de conceptos, así como en experimentos, ejercicios y problemas.
12. Gualberto Asencio Cortés (Noviembre 2017): Modificación del ejercicio 1 y problemas 2, 3 y 5.
13. Carlos D. Barranco (Noviembre 2017): Corrección de formato en ejercicio 2.

Estimación temporal:

- Parte presencial: 120 minutos.
    - Explicación inicial: 5 minutos.
    - Experimentos: 60 minutos.
    - Ejercicios: 55 minutos.
- Parte no presencial: 270 minutos.
    - Lectura y estudio del guión y bibliografía básica: 10 minutos
    - Problemas: 260 minutos