Explicación de los scripts utilizados

Archivo Map_url_parser.js

En este script validamos las coordenadas introducidas aplicando el uso de expresiones regulares. Para mantener el código más organizado hemos optado por crear un objeto que contenga cada expresión regular a usar, aportando además mayor capacidad de reutilización al mismo. Las expresiones regulares se encargan de validar tanto el formato de entrada como la validez de los mismos. Esto es, comprueban que la latitud sea menor o igual que 90°, por ejemplo. La explicación de estas RegExps no procede pues lo único que haríamos sería leerlas aplicando las reglas ya conocidas. Tomando ventaja del operador "?" somos capaces de resolver nuestro problema en pocas líneas. Además hemos limitado el número de decimales para evitar que los datos introducidos puedan exceder un tamaño razonable. Las RegExps no contienen modificadores. Como solo queremos buscar una ocurrencia no los necesitábamos.

Función parse_cardinal_coordinates():

Esta función será llamada cuando la coordenada introducida sea válida y su formato sea lo que hemos llamado cardinal (Y X°X'X", Y X°X'X"). Lo que haremos será pasarle la cadena de caracteres que contiene las coordenadas, buscaremos todos los valores numéricos con una expresión regular y los almacenaremos en un array. En función de los caracteres [NSEO] ajustaremos el signo de la petición a Google maps para más tarde sustituir los valores correctos en una cadena que teníamos preparada con un caracter específico que reemplazar. Haremos esto último también con expresiones regulares. Tras acabar el proceso colocaremos la URL de la localización en el cuadro de texto con ID "map".

Función parse_decimal_coordinates():

La idea detrás de esta función es totalmente análoga a la anterior. Corregimos el contenido de los valores encontrados para formar una petición correcta y luego la mostramos como el valor del elemento cuyo ID es "map". Como en este caso tenemos menos valores que reemplazar hemos generado la URL con una concatenación en vez de utilizar la potencia del método replace().

Función validate_input():

En esta función nos encargamos de comprobar si las coordenadas introducidas son válidas o no. Para ello utilizaremos las RegExps que definimos al principio como parámetro del método search(). Cabe destacar que nuestra comprobación es que este método devuelva un 0. Si bien nuestras expresiones regulares comprueban que las coordenadas se encuentren estrictamente entre el inicio y fin de la cadena a buscar queríamos hacer una segunda comprobación. En ausencia de "^" y "$" en las RegExps veríamos que comprobar que las coordenadas empiezan en el índice 0 de la cadena nos asegura que no hay datos "basura" antes de la misma. En cualquier caso, cuando se encuentre una coordenada válida veremos que si es cardinal cambiaremos el estado de unas "flags" definida en el archivo Globals.js. Con esto indicamos que hemos detectado una coordenada válida y que ésta es cardinal. Tras ello cambiaremos la clase del elemento "input" para que muestre un fondo verde que le indique al usuario que el formato de sus coordenadas es correcto. Hacemos esto manipulando el DOM a través del método setAttribute tomando ventaja de que no hay que crear el atributo "class", solo modificarlo.

Haremos algo análogo en el caso de que las coordenadas sean decimales. Solo en el caso en el que no se detecte ninguna veremos que cambiamos el atributo "class" para que el fondo del elemento "input" sea blanco, indicando que el formato de las coordenadas no es válido. Está función es llamada cada vez que introducimos un caracter en el elemento "input" asociado. Hacemos esto a través del evento "oninput". Además, vemos que a esta función se le pasa el elemento "input" como parámetro, algo que conseguimos pasando la palabra clave "this" en la llamada a este método desde el código HTML.

Archivo Globals.js

En este archivo hemos definido el objeto FORM_STATUS que contiene las "flags" indicando la validez de cada campo del formulario. Por ahora solo utilizamos el miembro "coordinates" pero planeamos integrar esta estructura en el resto del formulario a medida que las prácticas lo requieran. Hemos definido también el objeto FORMATS que indica a través de una cadena el formato de coordenadas con el que debemos trabajar. Cabe destacar que este archivo ha de ser el primero que importemos en nuestro código HTML ya que el resto deben poder modificar esta variable.

Archivo Get_values.js

Función get_values():

Tan solo encontramos la función homónima al nombre del archivo. Al pinchar en un botón habilitado a tal efecto llamaremos a esta función que se encargará de comprobar la validez de los datos a través del objeto FORM_STATUS y de llamar a las funciones que deben procesarlos. Hemos comentado la comprobación que debería hacer a través de un bucle "for" y solo nos fijamos en el campo "coordinates", el único que sigue esta estructura. Así, si el campo es válido llamamos a la función a cargo de generar la URL en función del formato de las coordenadas que conocemos a través del atributo "coordinates" del objeto FORMATS, definido también en Globals.js. Tras finalizar veremos que aparece la URL en el sitio apropiado en nuestra página o, en caso de que las coordenadas no sean válidas hemos decidido no mostrar una alerta por ser ésta demasiado intrusiva. No veremos ningún cambio en el formulario.

Archivo Mail_validation.js

Función validate_mail():

Vemos que la única función de este archivo es validate_mail(). Nuestra estrategia ha sido utilizar una vez más expresiones regulares para la validación del formato del correo electrónico. Como el código es más manejable no hemos considerado necesario crear un objeto que contuviera la RegExp que empleamos. Cabe destacar que en vez de utilizar una lista de dominios concatenada con el operador tubería "|" tal y como veíamos que se sugería en la teoría hemos optado por permitir tantos sufijos para el dominio del correo como desee el usuario siempre que contenga al menos dos caracteres alfanuméricos. Así tomamos como válidas direcciones como las nuestras institucionales: ABCD@edu.uah.es. Cabe destacar que en esta RegExp hemos utilizado el modificador "i" para permitir caracteres alfabéticos en mayúscula y minúscula a la vez que manteníamos una RegExp concisa. Al igual que antes no creemos que explicar el funcionamiento de la RegExp aporte demasiado… Solo hay que aplicar las reglas de las mismas. Cabe destacar que hemos escrito la RegExp teniendo en cuenta las reglas de:
https://help.xmatters.com/ondemand/trial/valid_email_format.htm

Al igual que hacíamos en la función validate_input() de Map_url_parser.js aquí añadimos el atributo "class" para cambiar

el fondo del campo en el que introducimos el correo electrónico a verde y lo eliminamos del DOM cuando el correo no sea válido. Cabe destacar que el método setAttribute() se encarga de crear y añadir el nodo de atributo al padre de manera automática, no tenemos que hacerlo de manera automática.

Como aquí solo trabajamos con un campo hemos decidido que la estructura que hemos empleado antes para validar el formulario era innecesaria, con lo que en esta función encontramos toda la funcionalidad que atañe a la validación del correo electrónico.

Archivo Config.js

Función pais():

Mediante el evento de JavaScript "onchange" permite verificar la validez del texto introducido en el campo "País" del formulario.

Para ello, admite escribir la palabra por completo y en el momento en que el puntero para escribir cambia de campo se llama a esta función. Ella se encarga de comprobar el valor introducido en el apartado mencionado y compararlo con cada una de las opciones incluidas en la lista de datos mediante un bucle "for".

De este modo, se comprueba uno a uno si ambos valores coinciden y si, por el contrario, no son iguales se incrementa el valor de un contador iniciado a 0. Después de finalizar la comparación con todas las posibilidades existentes se verifica si el contador empleado es igual al número total de opciones incluidas en la lista, en cuyo caso se obtiene la certeza de que el texto introducido era incorrecto y se muestra una alerta que informa de ellos.

Por el contrario, si esta última revisión no resulta correcta quiere decir que sí ha existido alguna coincidencia con las opciones, por lo que se permite continuar sin ningún problema.

Función check_pswd():

Esta función también emplea el evento "onchange" y cuando se termina de escribir la contraseña y se de-selecciona el campo para ello verifica si la longitud del texto introducido es mayor o igual que 6, en cuyo caso, de no serlo, muestra una alerta indicando esta incidencia.

fFunción passwd():

Esta función hace uso del evento "oninput" y a medida que se van introduciendo caracteres en el campo para introducir la contraseña va siendo invocada, actuando únicamente cuando se cumple la condición imprescindible de que se hayan escrito al menos 6 caracteres.

Así, cuando la longitud de la clave está entre 6 y 8 el campo de seguridad aparece en rojo. Si se introducen menos de 12 letras, este campo pasa a estar en amarillo y si son más de 12 la seguridad se considera correcta y aparece en verde.

Función temp():

Esta función se vale del evento "onchange" de JavaScript. Después de haber introducido el valor para la temperatura máxima o mínima y se desea pasar a otro campo comprueba el valor introducido. Si este no cumple con lo exigido, es decir, no está entre 22° y 26° C, se muestra una alerta para informar de ello y el dato introducido aparece en rojo.

Carlos Ortega Marchamalo & Pablo Collado Soto