PLAN 6: Manipulación del DOM

Módulo 2
Dedicación 7 días (35 horas)
Entrega 5 de junio de 2024

Descripción y objetivos

En esta sexta PLA jugamos con toda una serie de mecanismos diferentes mediante los cuales podemos modificar la estructura de una página web desde nuestro código JavaScript. Gracias a lo que se conoce como Document Object Model o DOM, JavaScript puede acceder y alterar los elementos de un documento HTML. Esto nos permite a los desarrolladores crear interacciones dinámicas y adaptativas con los usuarios, convirtiendo las páginas estáticas en entornos interactivos y vivos.

> El objetivo de las PLA 5, 6 y 7 es añadir interactividad a una página web para convertirla en una auténtica aplicación web, gracias a la versatilidad que nos ofrece el lenguaje de programación JavaScript.

Recuerde que, en paralelo a la resolución de la PLAN, conviene que siga trabajando en su proyecto final. Ahora ya sabe cómo estructurar una página web con HTML, cómo añadir estilos con CSS (opcionalmente con Bootstrap o con algún otro framework similar) y cómo añadir funcionalidad con código JavaScript. Con lo que veremos en esta PLAN también sabrá cómo modificar dinámicamente su contenido.

Consideraciones previas

Para resolver esta PLAN le resultará de especial interés la sección Document Object Model (DOM) de la MDL. El primer artículo introductorio que enlazan es un buen punto de partida.

Pero no se limite sólo a esta fuente, ya que en Internet encontrará una gran cantidad de información de gran interés al respecto. Y, como sabe, tiene también a su alcance el libro JavaScript Notes for Professionals donde, utilizando el índice o el buscador, encontrará parte de la información que necesita. Sin embargo, para esta PLAN este recurso no es tan útil como en otros PLAN, aunque sí que puede ir bien para encontrar cómo hacer determinadas cosas en JavaScript, no necesariamente relacionadas con la manipulación del DOM.

Por otra parte, aunque puede avanzar en el estudio de JavaScript sin tener que dominar aún los conceptos de callback function e higher order function de entender , conviene que dedique algunas horas a leer y mirar estos dos artículos:

- JavaScript Callback Function – Explained in Plain English (también disponible en forma de vídeo, recomendado hacer ambas cosas: leer el artículo y mirar el vídeo)
- What are Higher-Order Functions in JavaScript?

En cuanto al DOM, tal y como explican en JavaScript HTML DOM, es necesario saber que nos permite hacer un montón de cosas con un documento web, como por ejemplo:

- change all the HTML elements en la página change
- all the HTML atributes in the page change all the
- CSS styles in the page remove existing HTML
- elements and attributes add new HTML elements and
- attributes
- react to all existing HTML eventos en la página create
- new HTML eventos en la página

Y todo ello sin que el usuario tenga que refrescar la página ni tener que realizar ninguna nueva llamada al servidor. ¿Se da cuenta de la enorme cantidad de posibilidades que esto pone en nuestras manos?

Vídeos recomendados

En el canal de YouTube de Bluuweb encontrará la playlist Curso de JavaScript Moderno con una serie de vídeos que muestran un ejemplo bastante completo de manipulación del DOM. Recomendado mirar a estos seis, que suman un par de horas:

- #01 ¿Qué es el DOM?
- #02 Leer y Modificar HTML #03
- createElement #04
- Fragmento + createElement (no más reflow)
- #05 Template vs innerHTML vs createElement #06
- addEventListener (click), Event delegation y stopPropagation

> Si va justos de tiempo prescinde, de momento, de los vídeos #04 y #05, pero vuelva más adelante. En el mismo canal puede encontrar más vídeos con contenido interesante, aunque el estilo del autor es bastante peculiar, especialmente cuando hace sesiones en directo.

Importante

Ésta es la primera PLAN con un nivel de dificultad elevado en este curso. Vaya despacio, paso a paso, y no se desespere si desde el principio no sabe casi ni por dónde empezar. Se espera, más que nunca,

participación en el foro. No por compartir código con soluciones , claro, pero sí para compartir ideas («yo empezaría haciendo esto», «yo creo que tal elemento lo podemos recuperar teniendo en cuenta que cumple tal condición», etc).

No es necesario resolver todos los ejercicios perfectamente. Explique, en cada caso, qué ha hecho, hasta dónde ha llegado, cuál era la forma en que intentaba resolver el problema, etc.

Únicamente como referencia, el archivo scripts6-solution.js con todas las soluciones ocupa algo más de 300 líneas de código y se han definido una docena de funciones. Sus números pueden variar.

Tareas a resolver

Todas las tareas de esta PLAN deben resolverse en el archivo scripts6.js proporcionado, cada una en el espacio donde se indica en el mismo archivo. Aunque en algún momento le puede parecer que necesita también modificar el archivo index6.html o que le sería muy útil poder modificar también el archivo de estilos styles6.css , debe asumir que, debido a los requerimientos de un cliente en este caso imaginario, estos dos archivos no pueden alterarse en modo alguno. Sólo puede modificar el archivo scripts6.js .

> De nuevo, no modifique nada en los archivos index6.html y styles6.css . Sólo tiene que modificar el archivo scripts6.js , que es el que debe entregar al final. A la hora de corregir su PLAN se hará utilizando los archivos index6.html y styles6.css originales proporcionados.

> En este vídeo encontrará una pequeña demo de cómo debe funcionar el documento web index6.html una vez solucionadas todas las tareas propuestas.

1.Píldoras de teoría

Responda a las siguientes preguntas en la parte superior del documento scripts6.js en los bloques de comentario multilínea indicados. Asegúrese de que las líneas tienen un ancho máximo de 100 caracteres para mejorar su legibilidad. Los copypastes, citados o no, no puntúan.

1. ¿Cómo definirías qué es exactamente y para qué sirve el DOM?
2. La línea donde enlazamos los scripts al documento HTML incorpora el atributo defer . ¿Para qué sirve? ¿Qué diferencia hace ponerlo o no? Si no lo tiene claro, resuelva primero los ejercicios y después intente quitar el atributo y experimente con el resultado.
3. En los métodos getElementBy... y getElementsBy... hay una s que los diferencia. ¿Qué ¿implicaciones tiene esto en cuanto al valor que devuelven estos métodos? ¿Cuáles son los otros dos métodos que sirven casi por lo mismo?

2. Menú de opciones según rol seleccionado

En la página web encontrará un menú dividido en dos filas. En la primera podemos elegir entre dos roles diferentes: Usuario y Administrador . En la segunda hay muchas opciones disponibles según el rol escogido. Es necesario que escriba el código JavaScript necesario para implementar este menú.

Debe conseguir que si se elige el rol Usuario sólo se muestren las opciones en verde de la segunda fila, mientras que si se elige el rol Administrador todas las opciones de la segunda fila deben estar disponibles. Además, el rol seleccionado debe quedar remarcado añadiendo los estilos de la clase role-selected (ya existente en la hoja de estilos externa).

Para ello:

1. Piense bien con qué selectores puede recuperar tanto el listado de roles como el listado de opciones.
2. Necesitará añadir un evento a los dos elementos que hacen de botones para elegir el rol. Piense cuál evento es el más adecuado y qué función será necesario ejecutar cuando se produzca.

3. Clasificación de fútbol

En la página web encontrará una mesa vacía. Si mira el documento HTML verá que tiene definido un blog <tbody> , pero sin ningún elemento dentro. Esta tabla tiene el identificador clasificación .

En el código JavaScript encontrará definida una variable teams con cuatro selecciones nacionales de fútbol, cada una con el número de partidos que ha ganado, empatado y perdido. Escriba el código necesario para poblar esta tabla con el contenido de la variable teams . Para ello:

1. Aproveche que esta tabla tiene el identificador classification .
2. Cree una nueva fila <tr> por cada equipo con los elementos <td> correspondientes.
3. Añada estas filas al elemento <tbody> de la tabla.
4. En la primera fila entera añada la clase CSS classification-first (ya existente en la hoja de estilos externa).

> Los equipos deben aparecer, idealmente y como es habitual en estos casos, ordenados por puntos, de mayor a menor. El sistema de puntuación es el habitual: las victorias suman tres puntos, los empates un punto y las derrotas ninguna. Sin embargo, deje esta parte por el final si ve que le da problemas, ya que implementarla no es trivial.

4. Informe de clientes

En la página web encontrará una tabla con un listado de clientes, cada uno con su saldo y los días que faltan hasta el día en que se espera que se produzca el pago. Es necesario hacer tres cosas:

1. Cuando se carga la página deben verse en color rojo los valores negativos de la tercera columna. Se corresponden con las deudas que ya deberían haberse pagado y nos interesa que queden remarcadas. Utilice la clase unpaid (ya existente en la hoja de estilos externa).
2. Cuando se carga la página, la fila de los totales debe mostrar la suma de todos los importes de la columna en vez del 0.00 € que aparece por defecto.
3. La opción Remove unpaid no hace nada, pero al documento HTML tiene asociada una función a través del atributo onclick . Implemente esta función para que cuando el usuario pulse esta opción desaparezcan de la tabla todas las filas correspondientes a clientes morosos (los que ya deberían haber pagado). No debe esconder estas filas, debe eliminarlas del todo. Además, el saldo total deberá actualizarse para reflejar el total de las filas que permanezcan.

> Feo, ¿verdad, este uso del atributo onclick ? Sí, pero es uno de los mecanismos más sencillos para añadir interactividad a una página web y en algún proyecto antiguo se lo podría encontrar. Cuando esté en sus manos, opte siempre por asignar los eventos desde JavaScript.

Para ello:

1. Aproveche que esta tabla tiene el identificador customers .
2. Investiga la función parseFloat() .
3. Tenga presente que para eliminar un elemento del documento HTML podemos utilizar varios métodos, pero es necesario que realmente se elimine, no queremos dejarlo en modo no visible. Debe desaparecer del DOM.

5.Valoración con estrellitas

En el documento HTML encontrará un contenedor que incluye otros cinco contenedores con diferentes clases CSS asignadas.

Es necesario que escriba el código JavaScript necesario para implementar un sistema de valoración clásico en el que el usuario puede hacer clic sobre cualquiera de las estrellitas para marcar una puntuación. Esta puntuación debe quedar recogida en la variable rating ya definida en el archivo JavaScript, que tomará valores entre 1 y 5. Para ello:

1. Itera los posibles conjuntos de elementos que obtenga con un bucle de tipo for...of .
2. Como en la tarea 2, necesita asignar algún evento a algún elemento del DOM.
3. Desde el evento que acaba de añadir al punto anterior, ponga o saque clases CSS según corresponda a cada uno de los contenedores .

6. Barra de breakout

En el documento HTML encontrará un contenedor con la clase bar-container y, dentro, otro contenedor con la clase bar .

Es necesario que escriba el código JavaScript necesario para implementar una pequeña parte del típico juego Breakout donde debemos ir moviendo la barra para evitar que una pelotita caiga al vacío. Esta barra debe poder moverse de forma horizontal pero no vertical.

Usted debe conseguir que, pulsando las teclas flecha izquierda y flecha derecha, la barra se mueva.
Para ello:

1. Asigne un evento al elemento <body> .
2. Tenga presente cómo funciona el mecanismo de cascada de CSS. Seguramente necesitará sobreescribir una de las propiedades definidas en la clase bar . Puede hacerlo añadiendo desde JavaScript un atributo style (CSS inline) que, por tanto, pasará por encima de cualquier estilo CSS externo.

7.Piloteta de breakout

En el documento HTML encontrará un contenedor con la clase ball-container y, dentro, otro contenedor con otra clase ball que no puede utilizar en ningún caso en su código . Por tanto, tendrá que encontrar otra manera de ganar acceso al elemento que le interesa.

Es necesario que escriba el código JavaScript necesario para implementar otra pequeña parte del juego Breakout, aquella en la que la pelotita va rebotando por las paredes.

Usted debe conseguir que, de forma automática cuando se carga la página, la pelotita se mueva y vaya rebotando.
Para ello:

1. No puede utilizar la clase baile de la pelotita para acceder a ella. Debe utilizar el método querySelector para acceder a la que hay dentro del contenedor con etiqueta ball-container .
2. Investigue el método setInterval() para ejecutar una función de forma repetida. En este caso, opte por un número de milisegundos entre 20 y 50.
3. Tenga presente cómo funciona el mecanismo de cascada de CSS. Seguramente necesitará sobreescribir algunas de las propiedades definidas en la clase baile . Puede hacerlo añadiendo desde JavaScript un atributo style (CSS inline) que, por tanto, pasará por encima de cualquier estilo CSS externo.

Super challenge opcional

¿Tiene tanto la barra como la pelotita del Breakout funcionando? Pues no está muy lejos de poder juntar ambas partes y crear un pequeño juego. Quizás puede añadir un marcador y algunas mejoras visuales, pero la parte principal ya la tiene. ¿Se anima a terminarlo?

Entrega

Un archivo `c2409_PLA6_Apellido_Nombre.js` que incluya todas sus respuestas. Añada los blogs de comentarios que considere oportunos cuando desee explicar algún concepto o hacer cualquier tipo de aclaración.

Notas finales

1. Como en todas las PLAN, será sensacional si utiliza un repositorio privado en GitHub con el nombre `cifo-frontend-2024` y suba el contenido de esta PLAN (directorio `plan6`). Idealmente, el historial de commits debería mostrar el proceso evolutivo de resolución de la PLAN, tarea a tarea, apartado a apartado.

2. Nunca copie / plagie ninguna respuesta ni fragmento de código sin añadir las referencias correspondientes. En programación es habitual —y correcto— copiar pequeños bloques de código de distintos sitios. Pero si una parte relevante de lo que entrega no es de elaboración propia es necesario que lo mencione. No hacerlo supondrá una calificación de cero puntos.
3. Recuerde que las guías de estilo son muy importantes y conviene que las tenga presentes desde el principio, a fin de conseguir escribir un código fuente limpio, elegante y menos propenso a errores. En el caso de JavaScript, al tratarse del principal lenguaje de programación que trabajamos en este curso, esto cobra especial importancia.
4. Si bien herramientas como GitHub Copilot o ChatGPT son de innegable ayuda a la hora de programar, es muy importante que entienda perfectamente todo el código que entregue y que lo sepa explicar cuándo se lo pidan. En caso contrario le será difícil superar entrevistas de selección laboral para acceder a posiciones de programación.

Albert Mata — hello@albertmata.net — última actualización: 6 de mayo de 2024