**Introducción**

En este momento acabamos de recibir mucha información referente al manejo de los elementos en un documento web y necesitamos asentarla.

La mejor forma de asentar contenidos es poniéndolos en práctica, y en este Lab vamos a hacer un ejercicio guiado donde utilizaremos todo lo visto hasta ahora relativo al DOM.

**Ejercicio guiado**

En este Lab vamos a hacer un ejercicio guiado que nos servirá de repaso y aplicación práctica de todo lo visto hasta ahora.

El objetivo del ejercicio es crear una utilidad JavaScript que, cuando la apliquemos a cualquier página web bien estructurada nos incluya, de forma dinámica, una tabla de contenidos de dicha página web.

**Paso previo**

Abrir y entender los documentos base. Los documentos base son dos archivos, un documento HTML que contiene una estructura de ejemplo de una página web y un documento JavaScript prácticamente vacío donde escribiremos el ejercicio.

Lo primero que debéis hacer es abrir y leer los documentos y aseguraros de entender todo.

Como aspectos importantes de estos documentos hay que fijarse en el script literal definido en la cabecera del documento HTML. Éste script invocará la función buildTOC() cuando el navegador haya terminado la carga de la página y por tanto todos los elementos de la misma están correctamente cargados en el DOM en memoria del navegador. Es en este momento cuando los elementos de la página pueden ser consultados/modificados de forma segura.

La función buildTOC() está definida en el archivo js/toc.js que, como podéis comprobar, se incluye en la cabecera del documento HTML.

El otro aspecto importante es el contenido HTML. Está compuesto por headers de diferente importancia. HTML nos proporciona 6 tags para definir headers dependiendo de su importancia, siendo h1 el más importante y h6 el menos importante. El objetivo de los headers en HTML es servir como encabezados de las diferentes secciones y subsecciones en que se divide el contenido del documento.

Además del header, cada sección incluye un párrafo.

Si abrimos este documento HTML en nuestro navegador, podemos ver como el navegador, por defecto, aplica estilos de letra diferentes a cada header dependiendo de su importancia asociada.

Una vez que tenemos nuestro documento base ya podemos empezar a escribir el código de la función buildTOC.

**Paso primero**

La función buildTOC creará un nuevo elemento de tipo <div>, que colocará al comienzo de la página y será dentro de él donde construya la tabla de contenidos.

Consultando la sección de manipulación de elementos de la documentación de jQuery, no nos será difícil crear una nueva div e insertarla al comienzo de la página.

<http://api.jquery.com/category/manipulation/dom-insertion-inside/>

Añadir este código al comienzo de la función buildTOC.

var toc = $('<div></div>');

$('body').prepend(toc);

Este código crea un nuevo div y lo añade, como hijo del body, y además, por delante del primer hijo que tuviera. Con esto nos aseguramos que la tabla se situará al comienzo de la página.

A continuación debemos recopilar todos los elementos relevantes para nuestra tabla de contenidos, es decir, todos los elementos header. Esto podemos hacerlo de diferentes maneras, pero la más sencilla es como veremos a continuación.

var headings = $("h1, h2, h3, h4, h5, h6");

Esta sentencia retornará todos los elementos h1, h2, … h6 presentes en el documento y ordenados en orden de aparición. A continuación recorreremos todos los elementos contenidos en headings para determinar su nivel de importancia y así saber el grado de indentación que tendrán.

Para esto hacemos uso de la función each. Que, de acuerdo con la documentación, ejecutará la función que recibe como argumento una vez por cada elemento del array.

<http://api.jquery.com/each/>

Es importante darse cuenta de que el contexto de la función es el propio elemento cada vez, dicho de otra manera, dentro de la función argumento, this apunta al cada elemento del array sobre el que estamos iterando.

headings.each(function(index){

var heading = $(this);

var tocEntry = $('<p/>', { html: $(this).html() });

toc.append(tocEntry);

});

Con este código, para cada elemento contenido en el array de headings, determinamos su nivel de importancia obteniendo el número que acompaña a la ‘h’ en el tag y lo guardamos en level.

A continuación creamos un nuevo elemento de tipo párrafo con el contenido del heading y le asignamos una clase CSS de nombre ‘toc-’ seguido de un número que es correspondiente al nivel de importancia del header (toc-1 para h1, toc-2 para h2 y así sucesivamente).

Si guardamos el archivo y recargamos la página, podremos ver como se añade la tabla dinámicamente al comienzo del documento. Es cierto que no es muy atractiva visualmente, pero lo importante es conseguir la funcionalidad de añadir la tabla dinámicamente.

Ahora aplicaremos una pequeña indentación proporcional al nivel de importancia, para que la tabla tenga un aspecto más similar a un índice o tabla de contenidos.

Para ello vamos a aplicar a cada párrafo una clase CSS que indicará su indentación. Dichas clases CSS serán dependientes del nivel de importancia del header, así la clase CSS que apliquemos al párrafo correspondiente a un header h1 indentará 0px, la correspondiente a un header h2 15px y así sucesivamente. Este será el código de las clases CSS, que se deberá añadir al comienzo del documento, en la sección <head>

<style>

.toc-1 { margin-left: 0px }

.toc-2 { margin-left: 15px }

.toc-3 { margin-left: 30px }

.toc-4 { margin-left: 45px }

.toc-5 { margin-left: 60px }

.toc-6 { margin-left: 75px }

</style>

Para calcular qué clase CSS debemos aplicar a cada párrafo extraeremos la importancia correspondiente parseando su propiedad tagName.

var level = parseInt($(this).prop('tagName').charAt(1));

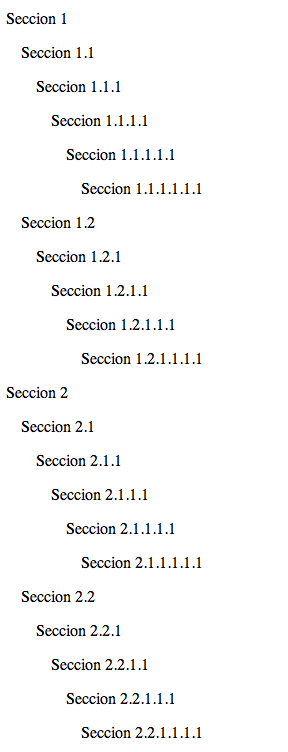
Y ahora que ya tenemos el valor de 1 a 6 en la variable level, bastará con modificar el código de creación de párrafos en la función buildTOC para incluir la clase CSS correspondiente a cada párrafo.

var tocEntry = $('<p/>', {

html: $(this).html(),

'class': 'toc-' + level

});

Fijaros, que en este caso, el nombre de la propiedad class del objeto atributos que pasamos para crear el párrafo va entre comillas, debido a que class es una palabra reservada de JS, y si no lo pusiéramos entre comillas, daría lugar a un error de sintaxis.

Con esto debería apareceros una tabla como la de la imagen. Si no os saliera, repasad los pasos y los posibles mensajes de error que aparezcan en la consola. Si aún así no os sale, entonces preguntad!, pero que nadie se quede sin conseguirlo, ya que éste ejercicio es la base del próximo.

Este Lab no finaliza cuando hayáis conseguido escribir el ejercicio y que funcione, sino cuando hayáis entendido minuciosamente todo el código y todo lo que hace para conseguir el resultado. Preguntad todo lo que sea necesario.