Librerías de basadas en JQuery y nuevas tendencias

Carlos Ortega Marchamalo & Pablo Collado Soto

Nosotros hemos estudiado y trabajado con JQuery. JQuery es una librería de JavaScript, es decir, código ya escrito por alguien que podemos emplear para no tener que "reinventar la rueda". En este caso JQuery nos permite utilizar selectores para obtener arrays de objetos que contienen representaciones de objetos del DOM. Hasta ahora nosotros manipulábamos el DOM directamente utilizando IDs para poder "llegar" a los elementos que queríamos modificar. Esto solía implicar una gran cantidad de código que no hacíamos más que reescribir con pequeñas modificaciones cada vez. Ahora, en vez de tener que reescribir y utilizar este código podemos utilizar los selectores que nos "dan" para condensar mucho nuestros archivos. Así, las funciones siblings(), children() o next() nos permiten recorrer todo el árbol DOM de manera rápida y sencilla.

Asimismo JQuery también facilita la manipulación de los elementos del DOM con funciones como show(), hide() o addClass(). El uso de eventos como los que se "lanzan" al pulsar en los botones es también mucho más sencillo. Lo que hacemos es añadirle un "listener" al botón en cuestión con una función que se ejecutará al recibir el evento. Ya no tenemos que incluir un atributo para cada evento asociado a cada botón lo que mejora en gran medida la legibilidad del código HTML. El "problema" de este modo de diseño es que las modificaciones se hacen sobre el DOM con lo que aislar problemas no es tan sencillo como podría parecer.

Otro punto en el que JQuery brilla es en el manejo de animaciones. Tenemos estructuras de colas FIFO por "debajo" que nos ayudan a encadenar varias mientras que ejecutar una animación es tan sencillo como llamar a animate() sobre un determinado objeto.

Además, JQuery facilita enormemente las interacciones con Ajax.

A pesar de todo esto, JQuery ha ido perdiendo algo de popularidad en detrimento de nuevas librerías y frameworks como React.js y Angular.js. Asimismo, existen librerías "por encima" de JQuery que le aportan funcionalidad adicional para manejar interfaces de usuario por ejemplo. Ahora pasamos a discutir sobre el resto de librerías del proyecto JQuery para luego visitar otros frameworks populares.

El resto de la familia JQuery

JQuery UI

JQuery resulta ser muy genérico en según y qué ocasiones. Motivado en parte por ésto nace JQuery User Interface (JQuery UI) que se encargará de generar interfaces de usuario utilizando las capacidades de JQuery. JQuery UI implementa todo lo que nos ofrece en hojas de estilo CSS así como en elementos HTML además de usar, como hemos dicho, JQuery por debajo. Si pensáramos en las librerías como una "pila" veríamos por tanto que JQuery UI está justo encima de JQuery. Al igual que en el modelo OSI JQuery ofrece servicios a su capa superior, en este caso, JQuery UI.

El diseño de interfaces con esta librería utiliza la idea de widgets que no son más que pequeños elementos con una funcionalidad determinada representados a través de objetos en JavaScript. Como podemos imaginar, estos objetos son instancias de clases definidas en la propia librería.

JQuery Mobile

Dada la demanda actual de páginas web desde smartphones JQuery decidió adaptarse a través de esta librería orientada a crear páginas con controles táctiles. JQuery Mobile pretende facilitar el desarrollo de páginas pensadas para ser vistas desde un teléfono tratando de ser tan independiente de la plataforma de visualización como pueda. Es decir, trata de funcionar de la misma manera en un dispositivo Android antiguo, uno moderno y otro basado en iOS. Más que aportar funcionalidad extra esta librería intenta conseguir una mayor portabilidad de todo lo que escribamos con JQuery.

Sizzle

Lo primero de todo es dejar claro que Sizzle no es una librería de JQuery sino un componente de éste. Al hablar sobre JQuery UI decíamos que si pensáramos en una "pila" de librerías tendríamos a JQuery debajo de JQuery UI. Pues bién, ahora Sizzle estaría por debajo de JQuery. Este componente es el "motor de selección" de JQuery. Vemos por tanto que cada vez que le pasamos una cadena a JQuery con $("selector") es en realidad Sizzle quien la interpreta para encontrar lo que queremos. Esta forma de diseño tiene sentido ya que hacemos que una parte crucial de JQuery sea lo más independiente posible del núcleo de la librería para poder

desarrollarlas en paralelo y facilitar el uso de Sizzle en cualquier otro proyecto.

A pesar de que Sizzle no sea una librería al uso nos ha parecido interesante comentar su funcionamiento ya que nos muestra que JQuery no es un "pegote" de código sino un proyecto relativamente extenso que requiere este componente para funcionar.

Treed

Treed es un plugin que se ejecuta "sobre" JQuery. Está pensado para facilitar la edición y visualización de estructuras de datos organizadas como árboles. Nos permite mostrar y ocultar nodos, renombrarlos, añadir y eliminar elementos… Podemos encontrar una demostración en: https://jaredforsyth.com/treed/.

JQuery Tools

En la línea de JQuery UI, JQuery Tools pretende incluir todos los elementos necesarios para crear interfaces de usuario web modernas. Sin embargo creemos que el desarrollo y soporte de la librería no están demasiado activos porque encontrar documentación o ejemplos de uso no ha sido fácil… Simplemente queremos reseñar que JQuery puede ser usado directamente o indirectamente a través de otras librerías que la utilizan como base.

Nuevas tendencias

Cuando JQuery vio la luz en el 2006 fue un gran avance. Le facilitó la vida a muchos programadores web permitiendo un manejo más sencillo del DOM de cada página. Con el tiempo han ido apareciendo otras soluciones como React y Angular que si bien no son librerías (en realidad son frameworks) incorporan y extienden la funcionalidad de JQuery. El ecosistema de JS cambia constantemente y ahora la tendencia es utilizar estos dos sistemas. Veremos lo que nos ofrece cada uno y haremos una pequeña comparación con JQuery.

Antes de comenzar nos gustaría distinguir dos términos que a veces se toman como sinónimos cuando no lo son: librería (library) y framework.

Una librería no es más que un conjunto de clases, funciones y objetos instanciados ya escritos por otra persona o entidad. Nosotros podemos instanciar cualquier clases y utilizar todos los

métodos y funciones que vengan definidas. Podemos emplear todo esto como queramos e integrarlo de la mejor manera posible en la estructura de nuestra página o proyecto. En definitiva, se nos proporciona código ya escrito que nos aporta nuevas funcionalidades y que podemos usar como queramos. Para poder emplear una librería de manera eficiente no nos queda otra que leernos el código y comprenderlo o recurrir a la documentación de la misma para ver qué métodos y funciones tenemos a nuestra disposición.

Un framework va un paso más allá que las librerías. Si bien son parecidas en cuanto a que ambas son código ya escrito el framework nos ofrece "espacios en blanco" en los que escribir nuestro código con el que particularizar el framework con funcionalidad genérica para nuestra aplicación. Al contrario que las librerías, un framework nos impone una forma de trabajar; no podemos hacer lo que queramos. Además no podemos (o no deberíamos) modificar el código del framework aunque sí podríamos hacerlo con el de las librerías (aunque no es lo normal). Si queremos ampliar la funcionalidad del framework nos toca escribir el código para hacerlo. Un framework también permite reutilizar el código de manera más eficiente y desplegar aplicaciones en mucho menos tiempo. Podemos pensar en un framework como una "aplicación" que no hace nada. Nos dan la estructura y solo tenemos que decir lo que queremos que ocurra. Las librerías solo nos dan herramientas para hacer lo que deseemos…

¿Qué es mejor? Depende de lo que necesitemos. Un framework es más "potente" pero nos restringe más nuestras opciones. Deberemos juzgar cada caso y aplicar la solución correcta en función de lo que necesitemos.

React

React es un framework desarrollado por Facebook que se utiliza para construir elementos de interfaces como lo que vemos en una página web. Con React creamos "componentes" que incorporan la funcionalidad del elemento así como su apariencia lo que nos permite hacer una programación mucho más modular. React es más eficiente que JQuery a la hora de manipular el DOM con lo que el rendimiento no se ve afectado en páginas con un DOM "grande". Para conseguir esta mejora React solo re-renderiza los elementos que cambien en la página y trabaja con un DOM "virtual" en vez del DOM original lo que reduce los tiempos de las distintas operaciones. React define la interfaz de los componentes con JSX que es una extensión de HTML. Además, al ser un framework nos "impone" un

estilo de trabajo; React no es simplemente un conjunto de clases y funciones que podemos emplear como nos dé la gana… Vemos por tanto que React nos ofrece mucha más funcionalidad que JQuery pero nos "pide" que sigamos unas pautas al usarlo.

Angular

Angular es otro framework, en este caso desarrollado por Google. Con Angular se facilita enormemente la creación de "Aplicaciones de Una Página". Esto es, generamos el código en HTML estático y con una serie de atributos adicionales Angular nos genera una página con "todo" cargado, no tenemos que cargar más elementos en respuesta a las peticiones del usuario. Angular logra ésto a través del uso del patrón MVC (MOdel-View-Controller) común a muchos sitios web. No obstante, al estar orientada a las Aplicaciones Web y no tanto a las Páginas Web puede resultar un poco rígida para según qué situación…

Vemos que ambos frameworks aportan una gran cantidad de funcionalidad. Entonces… ¿por qué seguir usando JQuery? JQuery es una librería así que por definición nos da más libertad para hacer lo que queremos. En vez de decirnos cómo trabajar nos aporta herramientas para que hagamos lo que nos parezca. Esto conlleva que la instalación y el "overhead" que le metemos al sistema es bastante menor de lo que sería con un framework. Con JQuery podemos probar ideas de manera más rápida y sencilla y para proyectos como el nuestro es suficiente. Nosotros manipulamos un DOM bastante pequeño y no hacemos demasiados cambios con lo que no le sacamos tanto partido a herramientas tan ambiciosas como React y Angular.

Glider

Para hacer una galería más dinámica hemos optado por utilizar Glider. Es un Plugin escrito en JavaScript que NO emplea JQuery. Instanciando las clases que define podemos crear galerías de imágenes dinámicas de manera mucho más sencilla que "ensuciándonos" las manos con código… Para "instalarlo" tenemos que incluirlo en nuestro servidor y referenciar el archivo .js que contiene todo lo necesario para funcionar desde la página en la que insertamos la galería. Asimismo vemos que para disponer la interfaz de la galería Glider hace uso de su propia hoja de estilos CSS por lo que también debemos incluirla. Vemos que si empleamos muchas de estas librerías se facilita nuestra labor pero también cargamos nuestra página con funcionalidades que pueden no

ser siempre necesarias… Son herramientos que debemos usar con cuidado.