JAVASCRIPT

CLASE 11

Material complementario

jQuery

CODER HOUSE

JavaScript: librerías

Liberia: definición

Una librería alude a uno o más archivos de JavaScript que contienen objetos y funciones programadas por terceros; está destinada a realizar tareas específicas en una aplicación o web.

Si bien es posible realizar desarrollos íntegros sin librerías, es común trabajar con ellas porque nos brindan múltiples funcionalidades pre-hechas, las cuales permiten agilizar el tiempo de desarrollo de una aplicación (no sería lo mismo si tuviéramos que hacer esto desde cero). Así, al dar a las librerías la posibilidad de solucionar las operaciones básicas y repetitivas de cada proyecto, nosotros podemos abocar nuestro tiempo a desarrollar las funcionalidades principales, nuevas, o específicas, de nuestra aplicación.

Para saber cómo usar una librería, debemos acceder a su documentación, la cual está compuesta por un documento o página web donde se detalla cómo funciona cada elemento de ella.

Es parte del trabajo del/la desarrollador/a realizar el análisis para determinar si es necesario utilizar una librería, cuál de ellas es la ideal para el proyecto, e investigar cómo integrar correctamente a nuestro trabajo. Cuando estamos desarrollando una app o web, podemos crear una librería externa con el objetivo de emplear las herramientas que nos ofrece en nuestro código. El proceso de referencia en el HTML es idéntico al de cualquier script JavaScript:

```
<script src="js/libreria.js"></script>
```

Uso de librerías

Origen de la librería

Para utilizar un librería en nuestro proyecto debemos referenciarla, y la forma de hacerlo puede cambiar en función de dónde se encuentra el origen de la misma. Tenemos dos formas de cargar una librería según su origen, ambas son completamente válidas, y en términos de performance no existe una gran diferencia asociada a la forma en que la misma se cargue:

- Cargar mediante URL: se carga el archivo de la librería de forma remota, leyéndolo desde una URL.
- Cargar localmente: se descarga el archivo a nuestro servidor, y se ejecuta localmente.

JavaScript minificado

Cuando utilizamos librerías externas, estas suelen brindar dos formas de trabajarlas: el formato comprimido y el descomprimido. En el descomprimido, el código JS se presenta de forma clara, legible y fácilmente editable, Este formato es el idóneo para cuando estamos en etapa de desarrollo, ya que nos permite analizar y editar la librería según nuestras necesidades.

```
/*!
 * jQuery JavaScript Library v3.6.0
 * https://jquery.com/
 *
 * Includes Sizzle.js
 * https://sizzlejs.com/
 *
 * Copyright OpenJS Foundation and other contributors
 * Released under the MIT license
 * https://jquery.org/license
 *
 * Date: 2021-03-02T17:08Z
 */
( function( global, factory ) {

	"use strict";

	if ( typeof module === "object" && typeof module.exports === "object" ) {

		// For CommonJS and CommonJS-like environments where a proper `window`
		// is present, execute the factory and get jQuery.
		// For environments that do not have a `window` with a `document`
		// (such as Node.js), expose a factory as module.exports.
		// This accentuates the need for the creation of a real `window`.
		// e.g. var jQuery = require("jquery")(window);
		// See ticket #14549 for more info.
		module.exports = global.document ?
			factory( global, true ) :
			function( w ) {
				if ( !w.document ) {
					throw new Error( "jQuery requires a window with a document" );
				}
				return factory( w );
			};
```

Por otro lado, en formato comprimido, comúnmente llamado minificado (o *minified,* en inglés), el contenido del archivo JS se encuentra escrito todo en una única línea, sin espacios en blanco, ni comentarios.

```
/*! jQuery v3.6.0 | (c) OpenJS Foundation and other contributors | jquery.org/license */
!function(e,t){"use strict";"object"==typeof module&&"object"==typeof module.exports?
module.exports=e.document?t(e,!0):function(e){if(!e.document)throw new Error("jQuery requires a window
with a document");return t(e)}:t(e)}("undefined"!=typeof window?window:this,function(C,e){"use strict";var
t=[],r=Object.getPrototypeOf,s=t.slice,g=t.flat?function(e){return t.flat.call(e)}:function(e){return
t.concat.apply([],e)},u=t.push,i=t.indexOf,n=
{},o=n.toString,v=n.hasOwnProperty,a=v.toString,l=a.call(Object),y={},m=function(e)
{return"function"==typeof e&&"number"!=typeof e.nodeType&&"function"!=typeof e.item},x=function(e){return
null!=e&&e===e.window},E=C.document,c={type:!0,src:!0,nonce:!0,noModule:!0};function b(e,t,n){var r,i,o=
(n=n||E).createElement("script");if(o.text=e,t)for(r in c)
(i=t[r]||t.getAttribute&&t.getAttribute(r))&&o.setAttribute(r,i);n.head.appendChild(o).parentNode.removeCh
ild(o)}function w(e){return null==e?e+"":"object"==typeof e||"function"==typeof e?
n[o.call(e)]||"object":typeof e}var f="3.6.0",S=function(e,t){return new S.fn.init(e,t)};function p(e){var
t=!!e&&"length"in e&&e.length,n=w(e);return!m(e)&&!x(e)&&("array"===n||0===t||"number"==typeof t&&0<t&&t-1
in e)}S.fn=S.prototype={jquery:f,constructor:S,length:0,toArray:function(){return
s.call(this)},get:function(e){return null==e?s.call(this):e<0?
this[e+this.length]:this[e]},pushStack:function(e){var t=S.merge(this.constructor(),e);return
t.prevObject=this,t},each:function(e){return S.each(this,e)},map:function(n){return
this.pushStack(S.map(this,function(e,t){return n.call(e,t,e)}))},slice:function(){return
this.pushStack(s.apply(this,arguments))},first:function(){return this.eq(0)},last:function(){return
this.eq(-1)},even:function(){return this.pushStack(S.grep(this,function(e,t)
{return(t+1)%2}))},odd:function(){return this.pushStack(S.grep(this,function(e,t){return
t%2}))},eq:function(e){var t=this.length,n=+e+(e<0?t:0);return this.pushStack(0<=n&&n<t?[this[n]]:
[])},end:function(){return
```

El uso de este formato se recomienda cuando se ha finalizado la etapa de desarrollo del proyecto, y está listo para pasar a producción, ya que la librería reduce el peso del archivo, como consecuencia de eliminar todos los saltos de línea y espacios innecesarios, evitándose así la edición del archivo JS,

Para identificar a los archivos, se suele respetar la siguiente convención de nombres:

- Archivo común: libreria.js
- Archivo minificado: libreria.min.js

Si por alguna razón requerimos modificar el Javascript una vez minificado, en la actualidad existen diversas herramientas online, que permiten descomprimir cualquier código Javascript minificado para poder revisarlo y modificarlo de ser necesario. Por ejemplo: https://herramientas-online.com/unminify-javascript-beautifier.html

jQuery

jQuery es una librería que nace de la necesidad de simplificar el acceso al DOM. Entre sus principales funcionalidades se encuentran la manipulación del DOM, controlar eventos, agregar animaciones, y ejecutar llamadas AJAX, entre otras características adicionales.

Si bien se puede interactuar con el DOM mediante las herramientas nativas de JS, lo que diferencia a jQuery es que es más práctico y eficaz en términos de legibilidad para el desarrollador, ya que en términos de cómputo resulta lo mismo.

En la actualidad, existen diversas opciones de librerías que realizan el mismo trabajo que jQuery, sin embargo, su presencia en el mercado es aún significativa, y existe un conjunto de tecnologías populares dependientes de la misma, convirtiéndola en una excelente candidata a ofertas laborales en migración de sistemas legados o escalado de software ya construidos con jQuery.[1]

El portal Built With nos provee estadísticas sobre las tecnologías de desarrollo empleadas para desarrollar los sitios y aplicaciones web más utilizados. Si realizamos la búsqueda discriminando por jQuery, podremos observar que la librería está presente en un porcentaje significativo de sitios web en la actualidad. Esto deja en evidencia la existencia de una oferta laboral amplia, principalmente en Argentina, donde la migración de sitios y aplicaciones a frameworks que empleen E56 podría llevar unos

años.

Por otro lado, debemos identificar que muchos de los últimos cambios de ES5 en adelante, se introducen en base a la popularidad de ciertas formas de trabajo y notación popularizada por jQuery. Esto implica que entender jQuery es también comprender el funcionamiento de algunas de las características actuales de JavaScript.

Ventajas de jQuery

- Se utilizan los mismos selectores de CSS3 para operar sobre los elementos, permitiendo a diseñadores/as front end la trasposición y comparación entre tecnologías.
- Permite diseñar y estructurar la aplicación en base a una codificación ordenada y unificada.
- Es open source, resultando conveniente para estudiar y modificar el código fuente de la librería según nuestras necesidades. Existe una gran cantidad de adecuaciones de la librería para distintas situaciones de desarrollo, que podemos usar y tomar de ejemplo en internet: jQuery UI, jQuery Mobile, Lista de mejores plugins.
- Posee una comunidad de desarrollo activa, que brinda soporte y mantenimiento a la librería. Última actualización: jQuery 3.6.0 - 2 de marzo de 2021
- Permite acceder al DOM de una forma ágil y sencilla, empleando selectores.
- Integra una importante cantidad de plugins pre-definidos (animaciones, sliders, componentes, etcétera), y brinda también la posibilidad de crear los propios para expandir la librería.
- Permite manipular diversos elementos y modificarlos con una línea de código. (encadenamiento de métodos). Asimismo, es posible incluir saltos de línea en el encadenado, garantizando un código más legible.

Cargar jQuery

Para utilizar esta librería, tenemos dos opciones diferenciadas según su origen:

Cargar mediante URL: si queremos hacer uso de esta opción, debemos localizar la URL

de un servidor CDN que provea la librería jQuery, y luego cargarlo en el archivo HTML mediante una etiqueta <script> antes de cerrar el </body>:

```
<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
```

Ocupar la librería desde un servidor CDN nos brinda la posibilidad de mantenerla siempre actualizada. Si la comunidad de desarrolladores/as realiza algún arreglo o mejora, al agregarlo al servidor todos los proyectos en los que esté referenciado también se actualizarán.

Cargar localmente: para usar la librería localmente, debemos descargar el archivo .js de la página oficial de jQuery, y luego incluirlo en nuestra carpeta del proyecto. Finalmente, cargamos el script en el archivo HTML mediante una etiqueta <script> antes de cerrar el </body>

```
<script src="js/jquery-3.5.1.min.js"></script>
```

Utilizar esta opción nos brinda la ventaja de poder trabajar offline con la librería.

Selectores

Los selectores en jQuery hacen referencia a la forma de identificar los diferentes elementos del DOM, y poder operar sobre ellos. En vez de emplear los métodos getElementById(), getElementsByClassName() y getElementsByTagName() de document, con JQuery podemos usar los selectores para acceder a cualquier elemento del DOM, así como agregar un nuevo elemento, modificar su innerHTML, o asociar un evento.

Una vez que importamos la librería en nuestro proyecto, utilizamos los selectores escribiéndolos bajo el siguiente esquema: $('selector'). Si el selector aplicado coincide con más de un elemento, quedan todos referenciados en la variable.

Los selectores proponen dos prefijos para acceder a elementos del DOM: si es por ID, empleamos el prefijo "#", y si es por clase utilizamos ".". Al utilizar los mismos prefijos que CSS, los selectores de jQuery se trabajan de forma similar a los selectores de CSS, con lo

cual un/a desarrollador/a front end puede realizar el traspaso de tecnología fácilmente:

```
// Acceso equivalente: document.getElementById("listaPaises");
$("#listaPaises");
// Acceso equivalente: document.getElementsByClassName("paises")
$(".paises");
// Acceso equivalente: document.getElementsByTagName("li")
$("li");
```

En la actualidad, JavaScript cuenta con los métodos querySelector y querySelectorAll para brindar acceso al DOM usando selectores, que proveen un comportamiento similar al propuesto por jQuery en primer lugar.

Combinación de selectores

Los selectores se pueden utilizar para seleccionar cualquier elemento HTML, usando su etiqueta, su clase, su ID, o la combinación de las tres:

```
$( "div" );      //selecciona todos los <div> de la página
$( "a" );        //selecciona todos los <a>
$( "p, a" );     //seleccionar todas los <p>, y los <a>
$( "li.nombre-clase #caja" );
//seleccionar todo <li> con clase "nombre-clase", y que tengan un hijo con
ID "caja"
```

Este tipo de combinación se utiliza con la intención de acceder a elementos del DOM, bajo un criterio de búsqueda bien específico.

Selectores por posición

Otra manera de emplear los selectores es definir el acceso al DOM teniendo presente la posición del nodo respecto a la página, o al nodo padre. Last (ultimo no de la página), first-child(primer hijo), last-child(último hijo), y otros identificadores señalados a continuación permiten definir distintos criterios de posición en el selector:

```
$( "p:last" );                 //Selecciona el último <p> de la página
$( "li:first-child" );         //Selecciona todos los <li> que son primeros hijos
$( "li:last-child" );          //Selecciona todos los <li> que son últimos hijos
$( "li:only-child" );          //Selecciona todos los <li> que sean hijos únicos
$( "li:nth-child(3)" );        //Selecciona todos los <li> que sean el 3er elemento
$( "tr:nth-child(odd)" );      //Selecciona todos los <tr> que sean impares
$( "tr:nth-child(even)" );    //Selecciona todos los <tr> que sean pares
$( "div:nth-child(3n)" );      //Selecciona cada tercer elemento <div>
```

Selectores de formulario

También podemos identificar a un grupo de selectores específicos cuya funcionalidad es acceder a elementos de un formulario, así como a diferentes etiquetas <input> según su tipo o propiedad:

```
$( ":text" );
$( ":checkbox" );
$( ":radio" );
$( ":image" );
$( ":submit" );
$( ":reset" );
$( ":password" );
$( ":file" );
$( ":input" );      //Selecciona los elementos input, textarea,select y button
$( ":button" );     //Selecciona los elementos button e input con atributo "type"="button"
$( ":enabled" );    //Selecciona los elementos del formulario activados
$( ":disabled" );   //Selecciona los elementos del formulario desactivados
$( ":checked" );    //Selecciona los radio buttons y checkboxes que están pulsados
$( ":selected" );   //Elementos de una lista de opciones que este seleccionados
```

Agregar elementos con jQuery

Hasta el momento, la forma que identificamos para incluir nodo al DOM es mediante el método ***createElement***, el cual nos permite especificar por parámetro el tipo de etiqueta

a crear. El método es útil para crear un nodo específico, pero ante la necesidad de incluir en el DOM una estructura HTML compleja, puede resultar poco práctico. jQuery propone el empleo de métodos ***append()*** y ***prepend()*** para añadir elementos al HTML en distinto orden. Veremos esto en los próximos párrafos.

jQuery append()

El método ***.append()*** inserta el contenido especificado como último hijo del elemento seleccionado, es decir, al final. Puede recibirse por parámetro una plantilla literal, como podemos ver en el ejemplo:

```
let producto    = { id: 1,  nombre: "Arroz", precio: 125 };
//Es posible usar plantillas de texto en el parámetro.
$("#app").append(`<div><h3> ID: ${producto.id}</h3>
                <p>  Producto: ${producto.nombre}</p>
                <b> $ ${producto.precio}</b></div>`);
```

jQuery prepend()

El método ***.prepend()*** inserta el contenido especificado como primer hijo del elemento seleccionado. Es decir que al usar prepend nos aseguramos que el elemento insertado sea el primero dentro de la etiqueta seleccionada:

```
let producto2    = { id: 2,  nombre: "Flan", precio: 150 };
//Es posible usar plantillas de texto en el parámetro.
$("#app").prepend(`<div><h3> ID: ${producto2.id}</h3>
                <p>  Producto: ${producto2.nombre}</p>
                <b> $ ${producto2.precio}</b></div>`);
```

Resaltamos que si bien los métodos ***append*** y ***prepend*** existen en JavaScript nativo, al día de hoy su funcionamiento no es idéntico a sus equivalentes en jQuery, principalmente porque el parámetro definido implica la utilización de un node si requerimos incluir un estructura HTML, como la visualizada en el ejemplo anterior.

Podemos verificar diferencias de notación y empleo entre jQuery y JavaScript ES6 consultando el siguiente recurso: primeros pasas para migrar de jQuery a JavaScript