APIs (Parte I)

Fundamentos de jQuery

Competencias

- Distinguir las características de la librería jQuery para comprender sus ventajas en el desarrollo web.
- Desarrollar algoritmos transformando código de JavaScript a jQuery para utilizar correctamente la sintaxis de la librería.

Introducción

Actualmente, el mercado del desarrollo web se ha disparado de forma exponencial, lo que ha derivado en la creación de herramientas muy potentes basadas en otras ya existentes. JavaScript, por su parte, al estar completamente integrado en los navegadores web, también se ha visto afectado por estos cambios y ha impulsado a que las empresas detrás de estas herramientas busquen como estandarizar el lenguaje.

jQuery nace para facilitar el desarrollo web, reduciendo la cantidad de líneas de código al programador, permitiendo generar un orden y una estructura al momento de escribir JavaScript. jQuery como tal, en estos momentos se sigue utilizando de forma complementaria a otros frameworks de desarrollo web, pero marca las bases para la evolución web tal y como la conocemos ahora, con una infinidad de opciones, librerías y formas de aplicación de éstas.

Aprender a utilizar estos recursos es clave para el mundo profesional, puesto que ofrece herramientas adicionales que facilitan y agilizan los desarrollos, por lo que su manejo es altamente valorado en las organizaciones.

¿Qué es jQuery?

jQuery es una librería de JavaScript creada en el año 2006, con una gran capacidad y adaptación a los cambios, ya que permite crear páginas web dinámicas de forma muy rápida y con la menor cantidad de código posible en JavaScript. El lema principal de jQuery es "escribir menos, hacer más".

Ejemplo de código en JavaScript

```
document.getElementById('test');
```

Ejemplo de código en jQuery

```
$('#test');
```

Se puede apreciar en el ejemplo anterior que jQuery posee un carácter especial representado por $, con el cual hace referencia a que se está utilizando un método o función de dicha librería.

Veamos cómo podemos seleccionar el siguiente elemento HTML utilizando JavaScript y jQuery:

```
<div id="identificador" class="mi-clase">
   Soy un div
</div>
```

Con JavaScript el elemento div lo podríamos acceder de la siguiente manera como revisamos en clases pasadas:

```
document.getElementsByClassName("mi-clase")[0];
```

Pero con jQuery podríamos acceder a los elementos utilizando diferentes selectores, por ejemplo para acceder al elemento div sería:

```
$('.mi-clase')
```

En el siguiente ejemplo se observa la diferencia entre la creación de una animación con jQuery versus trabajar con JavaScript como tal.

Ejemplo de código en JavaScript

```
var s = document.getElementById('thing').style;
s.opacity = 1;
(function fade() {
   (s.opacity-=.1) < 0 ? s.display="none" : setTimeout(fade,40)
})();
```

Ejemplo de código en jQuery

```
$('#thing').fadeOut();
```

La diferencia está a la vista, el bloque de código con JavaScript es complejo de leer y de entender qué tipo de animación se está generando. Sin embargo, si observamos el trozo de código escrito con jQuery, se entiende que se genera una animación con un componente de id="thing"

Gracias a la infinidad de funciones que entrega jQuery, es que la web moderna ha podido desarrollar de forma mucho más eficiente controladores de eventos, manipulación de objetos del dominio (de ahora en adelante, conocido como DOM), llamadas a API con AJAX y una multitud de Plugins creados por la comunidad de forma abierta para la utilización en múltiples proyectos. De esta forma, transformamos un lenguaje puro como JavaScript, añadiendo una capa de funcionalidad que permite a los desarrolladores implementar soluciones complejas de forma mucho más sencilla.

Ejercicio guiado: Transformando JavaScript a jQuery

El siguiente código, escrito con la sintaxis de JavaScript puro, obtiene un elemento con id 'text', una vez que el usuario realiza click. Se solicita transformar el código JavaScript con la librería de jQuery, para evidenciar sus diferencias. El código es el siguiente:

```
let text = document.getElementById('texto');
text.addEventListener('click', function(){
   document.write('click sobre el texto');
});
```

Para implementar jQuery en este código y transformar así la nomenclatura de JavaScript, se debe:

- ***Paso 1*:** Llevar la instrucción `let text = document.getElementById('texto');` a jQuery, utilizando el método o símbolo especial "$" que permite abreviar el document.getElementById, indicando en los paréntesis del método que estaremos buscando, un selector denominado "texto" el cual es un "id". Ese resultado lo guardamos en la variable "text".

```
let text = $('#texto');
```

- ***Paso 2:*** Como ya se encuentra el elemento guardado en una variable, se puede utilizar directamente para asignarle un evento, en este caso, el evento "click" de JavaScript. El cual, en jQuery viene asignado con una función que se ejecutará cuando el evento se active y realizará todos los procesos que contenga dentro de ella. Es decir, el evento "click" pasa a ser un método en jQuery exclusivo para ese tipo de evento. Quedando el código de la siguiente manera:

```
let text = $('#texto');
text.click(function(){
    document.write('click sobre el texto');
});
```

Ejercicio propuesto (1)

Transformar el siguiente código de JavaScript a jQuery.

```
let button = document.getElementById('button');
button.addEventListener('click', function(){
    alert('click sobre el botón');
});
```

¿Por qué utilizamos jQuery?

jQuery posee una cantidad enorme de ventajas por las cuales los desarrolladores se sienten atraídos a utilizar sus beneficios. Entre ellos se destacan algunos:

- Licencia MIT y Open Source: Esto quiere decir que su uso no posee ninguna restricción.
- Soporte: Actualización constante y comunidad activa en todo el mundo.
- Compatibilidad con todos los navegadores web en el mercado.
- Manipulación del modelo de objeto de dominio de forma mucho más natural que puramente con JavaScript.
- Manipulación dinámica de estilos CSS: Es compatible con CSS3.
- Integración con AJAX abriendo la librería a proyectos complejos con comunicación con servidores.
- Creación de animaciones de forma intuitiva y sencilla.
- Compatibilidad con una enorme y extensa lista de Plugins creados a partir de jQuery.
- Librería ligera en comparación con su versatilidad.
- Capacidad de optimización de tiempo y orden de código en proyectos de mayor complejidad.

Todas estas características hacen que jQuery sea uno de los favoritos del mercado, pero es importante señalar que existen actualmente otros frameworks de desarrollo más potentes y completos que jQuery como: React JS, Vue JS, Angular JS, entre otros. Hay que tener presente que jQuery permite trabajar con proyectos de mediana envergadura sin la necesidad de integrar elementos complejos en proyectos que no necesitan tantos recursos. En este sentido, jQuery es un equilibrio entre una librería robusta, ligera y sencilla de aprender.

Integrando jQuery en un Proyecto

Es importante señalar que dependiendo del proyecto en el cual se esté trabajando, se puede integrar jQuery. En esta sección se explicarán varias formas de integrar la librería.

Primera opción: Utilizar un recurso en un servidor remoto (CDN)

La primera opción es importar la librería jQuery como un <script> en la cabeza del archivo principal o main de un proyecto, referenciando el recurso de tipo text/JavaScript.

CDN (Content Delivery Network), no es otra cosa que un servicio que nos permite incluir las librerías de código de jQuery desde los servidores de algunas importantes empresas. En este caso, utilizaremos el enlace que nos proporciona Google, tal como se muestra en el ejemplo a continuación:

```
<!DOCTYPE html>
  <html>
    <head>
      <script type="text/JavaScript"
src="http://ajax.googleapis.com/ajax/libs/jQuery/x.x.x/jQuery.min.js"></
script>
    </head>
    <body>
       <div>old content</div>
        <script></script>
    </body>
  </html>
```

El siguiente código debería ir dentro de las etiquetas <script></script>, pero a modo de ejemplo las sacamos e incluimos a continuación:

```
    //oculta los divs de la página
    $('div').hide();
    //actualiza el contenido de todos los divs
    $('div').text('new content');
    //añade una clase a todos los divs
    $('div').addClass("updatedContent");
    //muestra todos los divs
    $('div').show();
```

Analicemos en profundidad el script que contiene las interacciones escritas en jQuery. Como se puede observar, los métodos o funciones de jQuery comienzan siempre con un token de identificación: $, el cual señala que es una función de jQuery. Es importante, marcar la diferencia, puesto que JavaScript como tal, puede tener funcionalidades bajo el mismo nombre que en jQuery, por ende, el no colocar el símbolo $ puede hacer entender al navegador que estás utilizando una función pura de JavaScript obteniendo errores o resultados no deseados.

Seguido del identificador, se observa una función que recibe parámetros. En el ejemplo, los parámetros que recibe para sus 4 acciones son div. Se debe considerar que al entregar como token de identificación la palabra o string div, jQuery interpretará que todas las etiquetas HTML de tipo div se verán afectadas por el cambio.

Las funciones reciben una instrucción que es aplicable a todos los div, pero necesitan adicionalmente recibir qué acción realizará en dicho componente. Es acá donde se introduce la función expresada por .func(params). Como se observa en los ejemplos, para todos los div se aplican funciones de **hide, text, addClass y show.**

La función hide esconde todos los elementos indicados en la instrucción anterior, es decir, esconde todos los divs a la vista del usuario. El esconder un componente no quiere decir que no exista, simplemente que no está a la vista del usuario, pero puede volver a reaparecer con la función show. Por otra parte, text añade un texto a un componente en el interior de sus etiquetas como un contenido. Por último, addClass añade una clase a un componente, en este caso, añade la misma clase a todos los div.

Ejercicio guiado: Implementando jQuery mediante CDN

Implementar la librería jQuery mediante el uso de CDN con el siguiente código, que obtiene el texto del elemento con id 'texto' cuando se recibe el evento click:

```
let text = $('#texto');
text.click(function(){
    document.write('click sobre el texto');
});
```

- ***Paso 1*:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

- ***Paso 2*:** En el index.html, escribir la estructura básica de un documento HTML como se muestra a continuación:

```
<!DOCTYPE html>
<html lang="en">
<head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width,
initial-scale=1.0">
      <title>Document</title>
</head>
<body>
      <h4>Este es un documento HTML con JavaScript</h4>
   <script src="script.js"></script>
</body>
</html>
```

- ***Paso 3*:** Al código anterior, es decir, en el archivo index.html, preparar el documento para poder implementar jQuery. Por ende, lo primero es incorporar el enlace o CDN de jQuery, que nos permita incluir esta librería en nuestro ejemplo. Puedes conseguir los enlaces en cualquiera de las opciones que facilita la página de jQuery en la

sección de descarga. Para este ejemplo, utilizaremos el enlace que se encuentra con la versión: 3.x. y será ubicado en la etiqueta <head> del documento HTML. Luego, se añadirá un id a la etiqueta <h4> con el nombre "texto", ya que es la variable text, carga el valor del id en ella.

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"><
/script>
   <title>Document</title>
</head>
<body>
   <h4 id="texto">Este es un documento HTML con JavaScript</h4>
   <script src="script.js"></script>
</body>
</html>
```

- ***Paso 4*:** En el archivo script.js, agregar el código dispuesto al inicio del ejemplo para poder ejecutarlo en el siguiente paso y ver el resultado. El código consiste en seleccionar un elemento mediante el tipo de selector "id", guardarlo en una variable y cuando se haga un clic sobre ese elemento se activa un mensaje que se mostrará en el documento:

```
let text = $('#texto');
text.click(function(){
    document.write('click sobre el texto');
});
```

- ***Paso 5*:** Ejecutar el código anterior y hacer un clic sobre el título con la etiqueta <h4>, el resultado en el navegador web será:

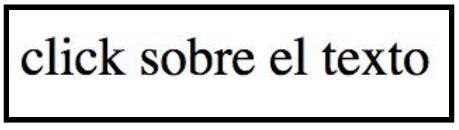

Imagen 1. Resultado al hacer clic sobre el texto
Fuente: Desafío Latam

Segunda opción: Descargar la librería y añadirla manualmente al proyecto

La diferencia con la primera opción es que al momento de subir tu página web, este archivo debe estar cargado, ya sea en el mismo servidor de la página web, o bien, en otro servidor propio donde se manejen los archivos. Esta forma, es recomendada para personas que sepan cómo configurar los entornos de trabajo. Para esta opción es necesario tener conocimientos de servidores, idealmente conocer lenguajes que permiten renderizar vistas, como Node.js.

Para esto, descargamos jQuery desde la página oficial y la incluimos en nuestro proyecto, dentro de assets/js. El código básico de HTML es el siguiente:

```
<!DOCTYPE html>
  <html>
    <head>
      <script type="text/JavaScript"
src="js/jquery-3.5.1.min.js"></script>
    </head>
    <body>
       <div>old content</div>
        <script></script>
    </body>
  </html>
```

Para efectos de esta unidad, cargaremos jQuery de forma remota (CDM), aunque le añade un poco más de tiempo de carga a la página que estamos desarrollando al solicitar un recurso externo, es mucho más fácil de implementar cuando se está comenzando un proyecto.

Para seguir profundizando contenidos, nos enfocaremos ahora en entender que es el modelo de objetos de dominio (DOM) y como manipularlo con jQuery.

Manipulación de elementos del DOM con jQuery

Competencias

- Integrar los conceptos de manipulación de DOM y selectores para resolver problemas utilizando la librería jQuery.
- Crear un script que manipule elementos del DOM con selectores utilizando la librería jQuery para resolver un problema planteado.

Introducción

Como hemos visto anteriormente, jQuery está pensado para facilitar el desarrollo, proporcionando una sintaxis sencilla para navegar por un documento, crear animaciones, seleccionar y manejar eventos del DOM.

A continuación, profundizaremos en los conceptos del DOM y las características que nos permiten interactuar con los elementos de una página HTML, seleccionando elementos y generando acciones sobre éstos con jQuery.

En la actualidad, se espera que una página web sea dinámica y responda a las necesidades del usuario, generando una experiencia interactiva e interesante para los visitantes. Esto lo podemos resolver de manera sencilla con los recursos que nos provee JavaScript a través de la librería jQuery.

¿Qué es Modelo de Objetos de Documento?

Recordemos que los objetos del DOM modelan tanto la ventana del navegador como el historial, el documento o página web y todos los elementos que pueda tener dentro la propia página, como párrafos, divisiones, tablas, formularios y sus campos, etc. A través del DOM, se puede acceder por medio de JavaScript, a cualquiera de estos elementos, es decir, a sus correspondientes objetos para alterar sus propiedades o invocar a sus métodos. Con todo, a través del DOM, queda disponible para los programadores de JavaScript cualquier elemento de la página para modificarlos, suprimirlos, crear nuevos elementos y colocarlos en la página, etc.

Veamos en acción qué es DOM

En el siguiente ejemplo, vemos un código escrito en HTML y JavaScript puro, en donde observamos que el body posee elementos p e input.

En el script, se describe una función que permite manipular el DOM, de modo que, buscando los elementos que tengan nombre x sean contabilizados y el valor numérico, sea añadido a la etiqueta p de id=demo.

```
<!DOCTYPE html>
<html>
   <head>
     <title>Ejemplo</title>
   </head>
   <body>
      <p>
          Cats: <input name="x" type="radio" value="Cats">
          Dogs: <input name="x" type="radio" value="Dogs">
      </p>
      <p><input type="button" onclick="getElements()" value="How many
elements named x?"></p>
      <p id="demo"></p>
      <script>
         function getElements() {
             var x = document.getElementsByName("x");
             document.getElementById("demo").innerHTML = x.length;
         }
      </script>
   </body>
</html>
```

Dicha acción. ¿Puede ser realizada con jQuery?

La respuesta es sí y lo veremos a continuación transformando el código de la función getElements de JavaScript a jQuery, si hay instrucciones que no entiendes aún, no te preocupes por que las veremos más adelante, lo importante es que aprecies la diferencia de sintaxis al realizar las mismas acciones de JavaScript con jQuery.

Lo primero que necesitamos es encontrar el selector de jQuery que nos permita contar los elementos que tengan nombre x, para eso utilizaremos el siguiente selector:

```
var totalElements = $('input[name="x"]').length
console.log(totalElements)
```

Si vemos el resultado deberíamos obtener:

```
2
```

Ahora que ya tenemos el total de elementos con el nombre x, debemos mostrar en el elemento con "id=demo", para eso utilizaremos un selector de ids y luego haremos uso del método text para agregar el total de elementos. Si te fijas obtenemos el mismo resultado, pero utilizando jQuery.

```
var totalElements = $('input[name="x"]').length
$('#demo').text(totalElements)
```

Manipulando DOM con jQuery

Al igual que con JavaScript, jQuery nos permite manipular cualquiera de los elementos del DOM, pero de manera muy sencilla utilizando el objeto $, el cual es acompañado de algún selector como los que se utilizan en CSS: podemos añadir, eliminar o modificar elementos del DOM. Para asegurarse que todos los elementos HTML se han cargado completamente en el DOM, jQuery dispone de un método llamado $(document).ready, que permite ejecutar nuestro jQuery una vez que todos los elementos hayan sido cargados y así evitar posibles errores.

```
$(document).ready(function() {
    // Escribe acà el código que desees que se ejecute una vez cargados
los elementos HTML del DOM
});
```

Selector ID

El selector de ID es uno de los selectores más utilizados, tanto en jQuery como en JavaScript puro.

La forma correcta de escribir un selector con jQuery es:

```
$(document).ready(function() {
    $('#some');
});
```

En donde #some es el id de algún componente HTML. Recuerda siempre que considerando buenas prácticas de programación, el id de algún componente debe ser único. A diferencia de las clases que pueden ser utilizadas en más de una ocasión.

Si tienes recuerdos de CSS, comprenderás que el identificador # es precisamente para indicar que se busca en un elemento de tipo id. Si por algún motivo existen varios elementos en la página con el mismo id la función retornará el primer elemento encontrado con ese id.

Veamos el siguiente ejemplo accediendo al elemento h1 utilizando selector ID.

```
<h1 id="soy-un-h1">Soy un titulo</h1>
```

Como hemos visto, para acceder a un elemento utilizando el selector id debemos usar # seguido del nombre del id del elemento como se verá en el siguiente código:

```
$(document).ready(function() {
   $('#soy-un-h1');
});
```

Ejercicio guiado: Seleccionar elementos por id con jQuery

Seleccionar mediante jQuery los elementos que se encuentran en un archivo HTML, guardarlos en una variable y mostrar las variables mediante un console.log, siendo estos elementos una etiqueta <p>, una etiqueta <input> y una etiqueta <button>, cada uno con un id distinto, como se muestra a continuación:

```
<p id="texto">Este es un documento HTML con JavaScript</p>
<input id="entrada" type="text" placeholder="Ingresa el texto aquí">
<button id="btn">Clic Aquí</button>
```

- ***Paso 1*:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

- ***Paso 2*:** En el index.html, escribir la estructura básica de un documento HTML, agregar el CDN para tener acceso a jQuery, enlaza el archivo externo de JavaScript y finalmente agrega las etiquetas indicadas al inicio del ejercicio, de esta forma:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"><
/script>
   <title>Document</title>
</head>
<body>
   <p id="texto">Este es un documento HTML con JavaScript</p>
   <input id="entrada" type="text" placeholder="Ingresa el texto aquí">
   <button id="btn">Clic Aquí</button>
   <script src="script.js"></script>
</body>
</html>
```

- ***Paso 3:*** En el archivo script.js, agregar el método llamado $(document).ready que permite ejecutar nuestro jQuery una vez que todos los elementos hayan sido cargados y así evitar posibles errores. Dentro del método, será donde agregaremos el código para seleccionar los elementos del DOM que poseen id:

```
$(document).ready(function() {
   let texto = $('#texto');
   let entrada = $('#entrada');
   let btn = $('#btn');

   console.log(texto);
   console.log(entrada);
   console.log(btn);
});
```

- ***Paso 4*:** Al ejecutar el código anterior, el resultado en la consola sería:

```
Object { 0: p#texto, length: 1 }
Object { 0: input#entrada, length: 1 }
Object { 0: button#btn, length: 1 }
```

En el resultado anterior, se puede observar como jQuery almacena un objeto por cada elemento que guardamos en las variables, cada objeto en su interior en la primera posición (0), agrega la etiqueta que contiene el id que almacenamos, mientras que en la última posición, agrega el largo que contendrá ese elemento, en este caso, como estamos trabajando mediante el selector de id, el largo siempre será igual a uno, pero cuando estemos trabajando con selectores como el de clases, la información puede ser variada y traer más de un elemento.

Ejercicio propuesto (2)

Selecciona, almacena en variables y muestra cada una de ellas para los elementos HTML mediante los selectores del tipo id de jQuery.

```
<h1 id="titulo_1">Este es un documento HTML con JavaScript</h1>
<h2 id="titulo_2">Este es un documento HTML con JavaScript</h2>
<p>El texto <span id="texto_1">indicado</span> fue seleccionado</p>
<button id="btn">Enviar</button>
```

Selector de Clase

El selector de clase es otro de los métodos más utilizados, tanto en jQuery como en JavaScript puro.

La forma correcta de escribir un selector de clase con jQuery es:

```
$(document).ready(function() {
   $('.some');
});
```

En donde .some es la clase de algún componente HTML. El identificador es precisamente para indicar que se busca un elemento de tipo class y, si por algún motivo, existen varios elementos en la página con la misma clase, la función retorna un arreglo con todos los componentes que poseen dicha clase.

La diferencia con un selector id, es que el selector de clase está pensado para ser utilizado en uno o más elementos. Veamos el siguiente ejemplo accediendo al elemento h1 pero ahora utilizando selector de clase.

```
<h1 class="soy-un-h1">Soy un titulo</h1>
```

Como hemos visto, para acceder a un elemento utilizando el selector id debemos usar "." seguido del nombre del id del elemento, como se verá en el siguiente código:

```
$(document).ready(function() {
   $('.soy-un-h1');
});
```

Ejercicio guiado: Selectores de clase con jQuery

Seleccionar mediante los selectores de clases de jQuery los elementos que se encuentran en un archivo HTML, guardalos en una variable y muestra las variables mediante un console.log, siendo estos elementos una etiqueta <p>, una etiqueta <input> y una <button>, como se muestra a continuación:

```
<p class="texto">Este es un documento HTML con JavaScript</p>
<input class="texto" type="text" placeholder="Ingresa el texto aquí">
<button class="texto">Clic Aquí</button>
```

- ***Paso 1*:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

- ***Paso 2*:** En el index.html, escribir la estructura básica de un documento HTML, agregar el CDN para tener acceso a jQuery, enlazar el archivo externo de JavaScript y finalmente agregar las etiquetas indicadas al inicio del ejercicio, como se muestra a continuación:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"><
/script>
   <title>Document</title>
</head>
<body>
   <p class="texto">Este es un documento HTML con JavaScript</p>
   <input class="texto" type="text" placeholder="Ingresa el texto aquí">
   <button class="texto">Clic Aquí</button>
   <script src="script.js"></script>
</body>
</html>
```

- ***Paso 3:*** En el archivo script.js, agregar el método llamado $(document).ready que permite ejecutar nuestro jQuery una vez que todos los elementos hayan sido cargados y así evitar posibles errores. Dentro del método, se agregará una única instrucción que permita captar los elementos que tienen la clase con el nombre "texto", en este caso, con una sola instrucción se puede hacer todo el proceso porque el selector de clases buscará todos los elementos que contengan la clase con ese nombre que estén disponibles en el DOM.

```
$(document).ready(function() {
   let texto = $('.texto');

   console.log(texto);
});
```

- ***Paso 4*:** Al ejecutar el código anterior, el resultado por consola sería:

```
Object { 0: p.texto, 1: input.texto, 2: button.texto, length: 3}
```

En el resultado anterior, se puede observar como jQuery almacena en un objeto todos los elementos del DOM que contienen la clase "texto" a modo de arreglo, ya que cada elemento está distribuido en una posición en específico de acuerdo al orden que fueron encontrados.

Ejercicio propuesto (3)

Selecciona, almacena en variables y muestra cada una de ellas para los elementos HTML mediante los selectores del tipo class de jQuery.

```
<h1 class="titulos">Este es un documento HTML con JavaScript</h1>
<h2 class="titulos">Este es un documento HTML con JavaScript</h2>
<p>El texto <span class="titulos">indicado</span> fue seleccionado</p>
<button class="titulos">Enviar</button>
```

Selección de Elementos en General

Como hemos visto, es posible seleccionar elementos del DOM a través de características como el ID y la clase, pero en códigos más complejos tendremos que combinar selectores para obtener los resultados esperados.

Observemos el siguiente ejemplo si quisiéramos seleccionar el div con clase some del siguiente HTML.

```
<div class="another"></div>
<div class="some"></div>
<div></div>
```

Se observa que tenemos 3 elementos div uno con clase another, otro con clase some y el tercero sin ninguna clase. Si necesitamos acceder al elemento con clase some podríamos hacerlo con el selector div.some, de la siguiente manera:

```
$(document).ready(function() {
    $('div.some');
});
```

Se debe tener cuidado con no dejar un espacio entre el elemento y la clase (div .some), ya que nos estaría seleccionando un elemento (cualquiera) con clase some que se encuentre dentro de un div.

Veamos otro ejemplo, donde necesitamos seleccionar varios elementos (diferentes) con jQuery, este requerimiento lo podemos realizar utilizando un selector múltiple con el que se pueden agrupar varios elementos separados por coma (,).

```
$(document).ready(function() {
   $('div, span');
});
```

En el ejemplo anterior se puede observar que el selector considera todos los div y los span presentes en el sitio. De esta forma, se pueden generar selectores específicos y complejos los cuales pueden localizar un elemento del DOM de forma precisa.

Ejercicio guiado: Accediendo a los elementos del DOM

Utilizando la selección de elementos generales de jQuery, se solicita identificar la primera etiqueta <a> sin agregar clases o id al elemento que contiene el HTML. Finalmente, para mostrar el resultado, se debe agregar un color de fondo al texto 'jQuery supports'. Utilizar el siguiente código:

```
<ul class="my-list">
   <li>
      <a href="http://jQuery.com">jQuery supports</a>
      <ul>
         <li><a href="css1">CSS1</a></li>
         <li><a href="css2">CSS2</a></li>
         <li><a href="css3">CSS3</a></li>
         <li>Basic XPath</li>
      </ul>
   </li>
   <li>jQuery also supports
      <ul>
         <li>Custom selectors</li>
         <li>Form selectors</li>
      </ul>
   </li>
</ul>
```

- ***Paso 1*:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

- ***Paso 2*:** En el index.html, escribir la estructura básica de un documento HTML, agregar el CDN para tener acceso a jQuery, enlaza el archivo externo de JavaScript y finalmente agregar las etiquetas indicadas al inicio del ejercicio, como se muestra a continuación:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"><
/script>
   <title>Document</title>
</head>
<body>
   <ul class="my-list">
      <li>
         <a href="http://jQuery.com">jQuery supports</a>
         <ul>
            <li><a href="css1">CSS1</a></li>
            <li><a href="css2">CSS2</a></li>
            <li><a href="css3">CSS3</a></li>
            <li>Basic XPath</li>
         </ul>
      </li>
      <li>jQuery also supports
         <ul>
            <li>Custom selectors</li>
            <li>Form selectors</li>
         </ul>
      </li>
   </ul>
   <script src="script.js"></script>
</body>
</html>
```

- ***Paso 3*:** En el archivo script.js, implementar los atributos y etiquetas que ya tiene nuestro HTML, ya que no podemos realizar modificación alguna, por lo tanto:
    - El primer elemento que aprovecharemos será la etiqueta <ul> que posee una clase denominada "`my-list`". Siendo específicos en el sector indicando, que sería la etiqueta ul que contenga la clase "`my-list`", quedando "`ul.my-list`".
    - Luego, para indicar que seleccionaremos el primer elemento hijo de la etiqueta <ul>, utilizamos el selector especial de CSS ">", este selector nos permite seleccionar sólo hijos directos de un elemento padre. En este caso "`ul.my-list > li`" seleccionará solo los dos li descendientes del padre ul con la clase my-list.

- Luego, la siguiente parte del selector "`li > a`" nos seleccionará los elementos que sean hijos directos de li y sólo cumpliría esta condición el siguiente elemento: `<a href="http://jQuery.com">jQuery supports</a>`:

```
$(document).ready(function() {
    $('ul.my-list > li > a');
    console.log($('ul.my-list > li > a'));
});
```

- ***Paso 4*:** Al ejecutar el código anterior, el resultado en la consola sería:

```
Object { 0: a, length: 1, prevObject: {…} }
```

Se puede observar en el resultado mostrado anteriormente, que el objeto trae un solo elemento, es decir, una sola etiqueta <a>, la cual corresponde a la primera etiqueta <a> que existe en el documento HTML.

- ***Paso 5:*** Para darle un poco más de colorido a nuestro elemento seleccionado, trabajaremos con el método CSS de jQuery, al cual le indicamos la propiedad y valor de CSS que deseamos agregar como un elemento en línea al elemento con el atributo style de HTML.

```
$(document).ready(function() {
     $('ul.my-list > li > a').css('background','red');
});
```

- ***Paso 6*:** Al ejecutar el código anterior, el resultado sería:

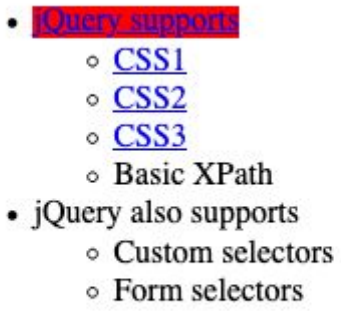

Imagen 2. Elemento seleccionado con un background:red
Fuente: Desafío Latam

En un principio estos conceptos pueden sonar un poco más complejos, pero con la práctica, se asimilarán. Cabe destacar, que la notación de los selectores como tal es idéntica a como

se expresan en CSS, puedes revisar el siguiente link donde se explica con más detalle los selectores de CSS.

Ejercicio propuesto (4)

Para el siguiente HTML, selecciona mediante los elementos generales todos los <li> que tienen el texto "Nested item". Igualmente añadele un color al texto, en este caso debe ser el color rojo.

```
<ul class="todos">
  <li>Item 1</li>
  <li>Item 2
      <ul>
         <li>Nested item 1</li>
         <li>Nested item 2</li>
         <li>Nested item 3</li>
      </ul>
  </li>
  <li>Item 3</li>
</ul>
```

Selección de elementos del DOM con jQuery

Competencias

- Reconocer los filtros como herramientas que permiten acotar la selección de elementos con jQuery, para utilizar correctamente los recursos al resolver un problema.

- Crear un script que permita la selección y manipulación de elementos del DOM utilizando la librería jQuery para resolver un problema planteado.

Introducción

Como vimos, jQuery nos ofrece una sintaxis sencilla para utilizar selectores y realizar acciones sobre los elementos del DOM. En comparación a JavaScript, la librería permite escribir un código mucho más limpio en menos líneas.

A continuación, veremos algunos recursos para realizar filtros, que son un tipo especial de selectores que nos facilitan el trabajo de obtener la información que vamos a manipular, de una forma más sencilla.

Utilizar filtros es una tarea recurrente en todo proyecto web, por lo que aprender estos conceptos nos será de mucha utilidad al desarrollar un sitio dinámico.

Filtros con jQuery

Los filtros son selectores especiales de jQuery que permiten acotar el espectro de una selección, limitado a la condición que se especifica en dicho filtro. Los filtros se verán marcados con el símbolo `:`.

Filtros de posición

```
$(document).ready(function() {
   $('ul.my-list > li > a:first');
});
```

El filtro :first retorna el primer elemento a de los existentes en los ítems de la lista de clase my-list.

```
$(document).ready(function() {
   $('ul.my-list > li > a:gt(1)');
});
```

El filtro :gt(1) (greater than) retorna los elementos que sean mayores al índice 1 de a presentes en la lista. Eso quiere decir que excluye a los elementos que están en la posición 0 y 1.

Bajo esa misma lógica encontramos filtros :last que devuelven un último elemento, :eq(n) (equal to) que devuelve el elemento que está en el índice n, :lt(n) (less than) que devuelve los elementos que están en el índice menor que el valor n.

Ejercicio guiado: Filtros con jQuery

Implementar los filtros de posición en una tabla con tres filas y tres columnas, se debe seleccionar y modificar el color de fondo de la primera y última fila, además del color del texto de la primera fila y el elemento de la columna tres con fila dos. El HTML a continuación será el punto de partida para el ejercicio.

```
<table>
 <tr>
    <td>Row 1</td>
    <td>Row 2</td>
    <td>Row 3</td>
 </tr>
```

```
 <tr>
    <td>Row 4</td>
    <td>Row 5</td>
    <td>Row 6</td>
 </tr>
 <tr>
    <td>Row 7</td>
    <td>Row 8</td>
    <td>Row 9</td>
 </tr>
</table>
```

- ***Paso 1*:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

- ***Paso 2*:** En el index.html, escribir la estructura básica de un documento HTML, agregar el CDN para tener acceso a jQuery, enlazar el archivo externo de JavaScript y agregar el extracto de HTML indicado al inicio del ejercicio. Además, se agregará un estilo con CSS para darle bordes a la tabla y así visualizar mejor el ejemplo:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"><
/script>
   <style>
      table, tr, td {
         border: 1px solid black;
      }
   </style>
   <title>Document</title>
</head>
<body>
   <table>
    <tr>
       <td>Row 1</td>
       <td>Row 2</td>
       <td>Row 3</td>
    </tr>
```

```
    <tr>
       <td>Row 4</td>
       <td>Row 5</td>
       <td>Row 6</td>
    </tr>
    <tr>
       <td>Row 7</td>
       <td>Row 8</td>
       <td>Row 9</td>
     </tr>
   </table>
   <script src="script.js"></script>
</body>
</html>
```

- ***Paso 3***: En el archivo script.js se trabajará con los selectores del tipo filtro de jQuery para poder realizar las modificaciones pertinentes solicitadas en el enunciado. Una de ellas, es el cambio de color de fondo de la primera y última fila. Esto se puede lograr implementado los filtros lt y gt, los cuales, seleccionan todos los elementos en un índice mayor o menor (dependiendo del filtro) que el índice dentro del conjunto coincidente. Para cambiar el color de fondo se utilizará el método de jQuery denominado CSS, donde se pasa la propiedad más el valor.

```
$(document).ready(function() {
   $('td:gt(5)').css('backgroundColor','blue');
   $('td:lt(3)').css('backgroundColor','yellow');
});
```

- ***Paso 4***: Para poder terminar el ejercicio, aún hace falta modificar el color del texto de la primera fila y la fila dos, pero en la columna tres específicamente. El primer caso se puede lograr mediante el filtro ":first", que permite seleccionar el primer elemento del DOM que coincida con el selector indicado, mientras que para seleccionar y modificar el elemento de la segunda fila con la tercera columna, se puede implementar el filtro ":eq".

```
$(document).ready(function() {
   $('td:gt(5)').css('backgroundColor','blue');
   $('td:lt(3)').css('backgroundColor','yellow');
   $('tr:first').css('color','red');
   $('td:eq(5)').css('color','red');
});
```

- ***Paso 5*:** Al ejecutar el código anterior en el navegador web, el resultado se vería de la siguiente manera:

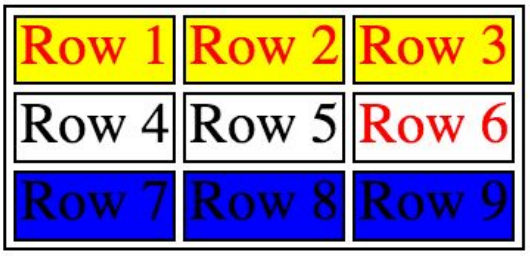

Imagen 3. Resultado en el navegador web.
Fuente: Desafío Latam

Ejercicio propuesto (5)

Para el siguiente HTML, selecciona mediante los filtros de posición el primer y último <li> perteneciente a la clase denominada "todos", añadiendo un color al texto a los elementos seleccionados, en este caso debe ser el color rojo.

```
<ul class="todos">
  <li>Item 1</li>
  <li>Item 2
      <ul>
         <li>Nested item 1</li>
         <li>Nested item 2</li>
         <li>Nested item 3</li>
      </ul>
  </li>
  <li>Item 3</li>
</ul>
```

Filtros hijo

Para implementar los filtros del tipo hijo con jQuery, se debe utilizar la pseudoclase "last-child", por ejemplo, si se tiene una estructura de listas no ordenadas y se desea seleccionar de esa lista solamente el último elemento y mostrar el texto que ese elemento posee, siendo la estructura del HTML la siguiente:

```
<ul>
   <li>item 1</li>
   <li>item 2</li>
   <li>item 3</li>
</ul>
```

Lo primero sería implementar la función principal de jQuery, luego, utilizando el método de jQuery "$" utilizar los selectores para indicar que apuntamos a la etiqueta padre "ul". Finalmente, acceder al último hijo de ella, en este caso al elemento "li", con ayuda de la pseudo clase "last-child":

```
$(document).ready(function() {
   $('ul li:last-child').html();
});
```

Por consiguiente, en el ejemplo presentado se observa que devuelve el último elemento hijo que está situado en una lista, mostrando específicamente el texto que posee el elemento. Este concepto es quizás un poco más complejo y específico pero es importante señalarlos. A modo de recomendación, para mantener un código ordenado, hay que evitar recurrir a filtros tan generalizados como los selectores o filtro hijo.

Ejercicio guiado: Filtros hijo

Se solicita seleccionar y modificar con el método CSS de jQuery los nombres de "Maria y Paola". La modificación debe ser del color de fondo y tamaño del texto. Siendo el extracto del HTML el siguiente:

```
<div>
  <span>Juan,</span>
  <span>Carlos,</span>
  <span>Manuel,</span>
  <span>María</span>
</div>
<div>
  <span>Rosa,</span>
  <span>Daniela,</span>
  <span>Pedro,</span>
  <span>Paola</span>
</div>
```

- ***Paso 1***: Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

- ***Paso 2***: En el index.html, escribir la estructura básica de un documento HTML, agregar el CDN para tener acceso a jQuery, enlazar el archivo externo de JavaScript y finalmente agregar el trozo de código mostrado en el enunciado del ejercicio:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"><
/script>
   <title>Document</title>
</head>
<body>
   <div>
     <span>Juan,</span>
     <span>Carlos,</span>
     <span>Manuel,</span>
     <span>María</span>
   </div>
   <div>
     <span>Rosa,</span>
     <span>Daniela,</span>
     <span>Pedro,</span>
     <span>Paola</span>
   </div>
   <script src="script.js"></script>
</body>
</html>
```

- ***Paso 3***: Ahora en el archivo script.js, se trabajará con los selectores de tipo filtro hijo de jQuery para poder realizar la selección de los elementos indicados. En este caso, consta en seleccionar y modificar con el método CSS los dos últimos elementos de cada etiqueta . Esto se puede lograr con el filtro del tipo "`:last-child`", aplicado a la etiqueta .

```
$(document).ready(function() {
   $('div span:last-child').css('backgroundColor','blue');
});
```

- ***Paso 4*:** Al ejecutar el código anterior, el resultado en el navegador web será:

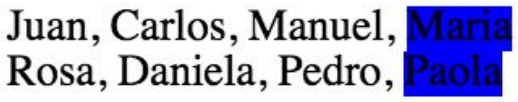

Imagen 4. Resultado en el navegador web.
Fuente: Desafío Latam

Ejercicio propuesto (6)

Utilizando el filtro hijo nth-child selecciona el segundo elemento de la lista agregando un color de fondo azul.

```
<ul>
   <li>uno</li>
   <li>dos</li>
   <li>tres</li>
   <li>cuatro</li>
   <li>cinco</li>
</ul>
```

Filtros form

También es posible añadir filtros a elementos del DOM de tipo form. Estos filtros son útiles para capturar información de un usuario mientras escribe algún valor en un formulario, cambia el estado de cierto elemento o simplemente desea enviar dichos datos a algún servidor para ser procesados.

```
$(document).ready(function() {
   $('input[type="checkbox"]:checked');
});
```

Como se observa en el ejemplo, seleccionamos los input de tipo checkbox que estén en un estado marcado checked. Esto nos permite gatillar alguna acción al momento que un usuario decide marcar algún componente de tipo checkbox.

Ejercicio guiado: Filtros form

Se solicita acceder y mostrar el elemento que ya viene seleccionado por defecto en la etiqueta <option>:

```
<select name="paises">
    <option value="Argentina">Argentina</option>
    <option value="Chile" selected>Chile</option>
    <option value="Brasil">Brasil</option>
</select>
```

- ***Paso 1*:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

- ***Paso 2*:** En el index.html, escribir la estructura básica de un documento HTML, agregar el CDN para tener acceso a jQuery, enlazar el archivo externo de JavaScript y finalmente agregar las etiquetas indicadas al inicio del ejercicio:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"><
/script>
   <title>Document</title>
</head>
<body>
      <select name="paises">
          <option value="Argentina">Argentina</option>
          <option value="Chile">Chile</option>
          <option value="Brasil">Brasil</option>
      </select>
      <script src="script.js"></script>
</body>
```

```
</html>
```

- ***Paso 3*:** Utilizando un filtro form, debemos acceder al elemento y filtrar por el selector principal ("select"), luego accedemos al hijo mediante el selector ">", siendo el elemento "option" el hijo directo, aquí si se aplica el filtro "selected", posteriormente con el método "val" capturamos el valor (value) que tiene el "option".

```
$(document).ready(function() {
   $('select > option:selected').val();
});
```

- ***Paso 4*:** El método "val" nos ayuda a obtener el valor del elemento, por ende, guardemos el resultado en una variable y usamos console.log para mostrar el valor.

```
$(document).ready(function() {
   let valorDeLaOpcion = $('select > option:selected').val();
   console.log(valorDeLaOpcion)
});
```

El resultado por consola es:

```
Chile
```

Ejercicio propuesto (7)

Escriba un filtro form para el siguiente elemento:

```
<input type="email" />
```

Otros filtros interesantes

La gama de filtros es enorme y la cantidad de acciones que se pueden realizar con ellos es compleja. Cabe destacar que existen filtros para obtener elementos animados (:animated), elementos ocultos (:hidden), un elemento raíz (:root), o elementos visibles (:visible). La gama de filtros siempre la puedes consultar al detalle en la documentación oficial de jQuery.

Eventos con jQuery

Competencias

- Reconocer los eventos, sus principales características y cómo interactuar con ellos, para utilizar correctamente los recursos de jQuery en sitios web dinámicos.

- Crear un script que permita el trabajo con eventos, utilizando la librería jQuery para resolver un problema planteado.

Introducción

Hasta ahora hemos visto las herramientas que nos provee jQuery para seleccionar y manipular elementos del DOM. Ahora sólo nos falta comprender qué son los eventos y cómo jQuery nos permite realizar acciones e interactuar con los elementos del DOM de manera dinámica.

A modo general, los eventos son acciones que ocurren dentro del HTML y pueden ser desencadenadas por situaciones del sistema o por la interacción del usuario. En cualquier caso, es posible capturar esta información y procesarla, por lo que es posible implementar lógica a través de estos elementos.

Con estas herramientas, estaremos en condiciones de crear un sitio web interactivo y mucho más robusto, que utilice los elementos dinámicos que se desencadenan en la ejecución del sitio y dar una mejor experiencia al usuario, de acuerdo a los requerimientos de la industria. Por estas razones, el manejo de estos elementos es altamente valorado en el mundo laboral.

¿Qué es un evento?

Conociendo los aspectos básicos para la selección de elementos del DOM con jQuery, ahora es necesario interactuar con dichos elementos de modo que generen dinamismo en la web. Para ello es necesario comprender el concepto de evento.

Un evento es una acción o función determinada que se ejecuta a partir de una interacción con nuestro sitio web, por ejemplo, el hacer click sobre un botón o pasar el mouse sobre algún elemento html. jQuery tiene una amplia gama de eventos propios que permiten manipular el DOM de una forma sencilla e intuitiva, sin perder eficiencia frente a lo que podría ser realizado con JavaScript Vanilla.

```
$(document).ready(function() {
   $('button').click(function () {
      alert("hola mundo");
   });
});
```

Como se observa en el ejemplo se está desencadenando un alert(), cuando estamos haciendo click sobre el elemento button de nuestro html. En general, los eventos en jQuery los podemos identificar o necesitar cuando queremos "hacer algo" a partir de una acción.

Ejercicio guiado: Eventos con jQuery

Imagina que en una página web solicitan que se visualice en un párrafo la selección que ha hecho el usuario en un formulario, mediante la etiqueta <select>, es decir, que si se elige Chile como opción dentro de la lista, este texto lo veamos reflejado en alguna parte del sitio.

Mediante jQuery y la captura de eventos, vamos a recoger esta información utilizando el método change, en base al siguiente código:

```
<p class="resultado"></p>
<select name="pais" id="pais">
   <option value="Argentina">Argentina</option>
   <option value="Perú">Perú</option>
   <option value="Chile">Chile</option>
   <option value="Venezuela">Venezuela</option>
</select>
```

- ***Paso 1***: Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

- ***Paso 2***: En el index.html, escribir la estructura básica de un documento HTML, agregar el CDN para tener acceso a jQuery, enlazar el archivo externo de JavaScript y finalmente agregar las etiquetas indicadas al inicio del ejercicio:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"><
/script>
   <title>Document</title>
</head>
<body>
      <p class="resultado"></p>
      <select name="pais" id="pais">
         <option value="Argentina">Argentina</option>
         <option value="Perú">Perú</option>
         <option value="Chile">Chile</option>
         <option value="Venezuela">Venezuela</option>
      </select>
      <script src="script.js"></script>
</body>
</html>
```

- ***Paso 3***: Para dar solución a lo solicitado, necesitamos usar algún evento de jQuery que capture el cambio en el selector para luego rescatar su valor. Tal como en JavaScript, en jQuery existe el evento "change" que se ejecutará cuando el elemento seleccionado tenga algún cambio. Por consiguiente, lo primero que haremos será acceder al elemento principal del formulario, es decir, a la etiqueta "select" que posee un `id="pais"`, mediante el selector de jQuery: `$('#pais')`. Aplicaremos el evento change, el cual, conlleva a una función que se ejecutará cada vez que ocurra ese evento en el selector indicado al método.

```
$(document).ready(function() {
    $('#pais').change(function() {
    })
});
```

- ***Paso 4***: Ya seleccionado el elemento, el cual reacciona cuando el usuario elija alguna de las opciones, ya está la mitad del trabajo hecho. Ahora necesitamos acceder a la opción seleccionada, obtener su valor y mostrarlo en el elemento p. Eso se logra indicando que al elemento con el id igual a "pais", es decir, el "select", se le aplique el método "children" para poder capturar el hijo que fue seleccionado, siendo el hijo directo la etiqueta <option> y en ella, activamos el filtro ":selected" más el método "val" para obtener el valor.

```
$(document).ready(function() {
   $('#pais').change(function() {
      let paisSeleccionado = $(this).children("option:selected").val();
      $('.resultado').text(paisSeleccionado);
   })
});
```

- ***Paso 5***: Si ejecutamos el código anterior, el resultado dependerá de la selección del usuario, por ejemplo, si selecciona Chile, el resultado será:

Imagen 5. Resultado en el navegador web.
Fuente: Desafío Latam

jQuery nos ofrece varios eventos que es importante conocer y tener presente a la hora de escribir código, te invito a que revises los eventos disponibles en el siguiente link

Ejercicio propuesto (8)

Cambia el color de fondo del div utilizando el evento de ratón: mouseenter

```
<div class="miCaja" style="width:100px;height:100px;border: 1px solid
black"></div>
```

Utilizando eventos con jQuery

Ahora vamos a profundizar las acciones que podemos realizar utilizando jQuery. Por ejemplo, uno de los métodos más implementado para manejar cualquier tipo de evento, es el método ".on()", el cual permite indicar el tipo de evento y agregar una función para el manejo de la acción. Incluso se puede especificar el elemento al cual se estará supervisando en busqueda del evento indicado.

En el siguiente código se utiliza el método ".on()" para capturar el evento "click" y mostrar por consola un mensaje:

```
$(document).ready(function() {
    $( "button" ).on( "click", function() {
        console.log( "Ocurrió un clic sobre la etiqueta p" );
    });
});
```

En el siguiente ejemplo, veremos que la función get recibe como parámetro un número, el cual es la posición del elemento que queremos retornar cuando se hace click sobre un botón implementando el método on().

```
$(document).ready(function() {
   $( "button" ).on( "click", function() {
       var imgElem = $('img[alt]').get(0);
   });
});
```

Dicha función realiza la misma acción que:

```
$(document).ready(function() {
   $( "button" ).on( "click", function() {
       var imgElem = $('img[alt]')[0];
   });
});
```

Aunque puede parecer innecesario, es importante destacar que un código legible es una mejor experiencia para el equipo de trabajo y para ti mismo al momento de revisar trabajos previos realizados por tí. En este sentido, es mucho mejor definir acciones que sean capaces de describir lo que hacen por sí mismas.

La siguiente función retorna todos los elementos marcados por el selector en forma de un arreglo de elementos de DOM, básicamente es un arreglo de tags de HTML, los cuales son guardados en una variable:

```
$(document).ready(function() {
   $( "button" ).on( "click", function() {
      var allLabeledButtons = $('label + button').toArray();
   });
});
```

La función toArray es muy útil para dar orden a un conjunto de elementos, como ya sabrán con JavaScript, los arreglos poseen una cantidad de funciones enormes como para manipular elementos.

Ahora, en el siguiente ejemplo definimos funciones encadenadas (chaining), es decir, una secuencia de condiciones en donde al terminar de realizar una acción ejecuta otra. En este caso, al selector de img le añadimos una clase llamada opaque, luego filtramos dentro de su contenido por la condición de que su título sea dog y a este componente le añadimos una clase llamada red-border. Como se puede observar, la capacidad de generar contenido dinámico con jQuery es muy amplio.

```
$(document).ready(function() {
   $('img')
        .addClass('opaque')
        .filter('[title*="dog"]')
        .addClass('red-border');
});
```

Ejercicio guiado: Seleccionar y cambiar el color de fondo

Se solicita seleccionar y cambiar el color de fondo a los elementos impares dentro de la lista desordenada cuando el usuario haga un click en el botón. Para ello se tiene el siguiente código en html:

```
<ul>
   <li>item 1</li>
   <li>item 2</li>
   <li>item 3</li>
   <li>item 4</li>
   <li>item 5</li>
   <li>item 6</li>
```

```
</ul>
<button id="btn">Seleccionar<button>
```

- ***Paso 1***: Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

- ***Paso 2***: En el index.html, escribir la estructura básica de un documento HTML, agregar el CDN para tener acceso a jQuery, enlazar el archivo externo de JavaScript y finalmente agregar las etiquetas indicadas al inicio del ejercicio:

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"><
/script>
   <title>Document</title>
</head>
<body>
      <ul>
         <li>item 1</li>
         <li>item 2</li>
         <li>item 3</li>
         <li>item 4</li>
         <li>item 5</li>
         <li>item 6</li>
      </ul>
      <button id="btn">Seleccionar</button>
      <script src="script.js"></script>
</body>
</html>
```

- ***Paso 3***: Para dar solución a lo solicitado, necesitamos usar algún evento de jQuery que permita capturar el evento “click” cuando active el botón. En este caso, uno de los posibles métodos que se pueden emplear, es el “.on()”, quién podrá ser aplicado directamente al botón, quedando:

```
$(document).ready(function() {
   $("#btn").on("click",function() {
        console.log( "Ocurrió un clic sobre el botón" );
    });
});
```

- **Paso 4:** Ahora como ya está activo y trabajando el evento, procedemos a seleccionar los elementos "li" en posiciones impares mediante el método filter, y con el pseudo elemento :even podemos seleccionar las posiciones impares (y con :odd las pares).

```
$( "li" ).filter( ":even" );
```

- ***Paso 5*:** Como ya seleccionamos los elementos, realicemos alguna acción sobre ellos, por ejemplo, cambiar el color de fondo:

```
$(document).ready(function() {
   $("#btn").on("click",function() {
        $( "li" ).filter( ":even" ).css( "background-color", "red" );
    });
});
```

- ***Paso 6*:** Si ejecutamos el código anterior, y hacemos un click en el botón, la selección ocurrirá sobre los ítem con número impar, siendo el resultado:

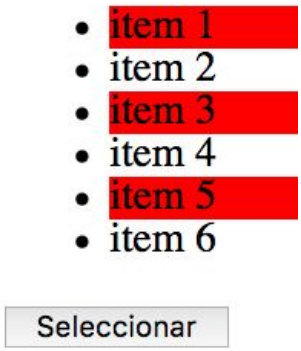

Imagen 6. Resultado en el navegador web.
Fuente: Desafío Latam

También podemos seleccionar varios elementos, iterarlos y aplicar alguna condición:

```
$(document).ready(function() {
    $('input').each(function(index, element) {
        var $element = $(element);
        $element.attr('value', $element.attr('placeholder'));
    });
});
```

Vemos que el ejemplo anterior es un poco más complejo. Donde seleccionamos todos los input y generamos una iteración entre ellos. Dicha función definida por each retorna una función callback con un índice y el elemento sobre el cual se está iterando, luego se crea una variable jQuery con dicho elemento y a cada uno de ellos se le añade con la función attr al atributo HTML "value" el elemento presente en el atributo placeholder.

¿Complejo verdad? Se darán cuenta que en la práctica, dichos elementos se vuelven intuitivos, la mejor forma de entender estos conceptos es escribiendo código, buscando ejemplos y leyendo documentación.

Ejercicio guiado: Contar los clicks sobre un elemento

Se solicita que al hacer click sobre una imagen, se muestre en consola la cantidad de clicks realizados y la hora. Por ende, para dar respuesta al ejemplo, se debe hacer lo siguiente:

- ***Paso 1*:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

- ***Paso 2*:** En el index.html, escribir la estructura básica de un documento HTML, agregar el CDN para tener acceso a jQuery, enlazar el archivo externo de JavaScript y finalmente agregar las etiquetas indicadas al inicio del ejercicio:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width,
initial-scale=1.0">
      <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"><
/script>
      <title>Ejemplo jQuery</title>
   </head>
```

```
   <body>
       <h3>Clic en la imagen</h3>
       <img id="ejemplo"
src="https://desafiolatam.com/assets/seo/desafiolatam_principal-031aadc5
74e79866fac011f0e4dae20b0d891d220e593b203b39568a495b894c.jpg" alt="DL"
/>
   <script src="script.js"></script>
</body>
</html>
```

- ***Paso 3:*** En el archivo script.js, crear una variable para llevar un contador, esto nos servirá para tener presente la cantidad de veces que está haciendo click el usuario sobre la imagen. Luego, utilizando el selector de id mediante jQuery, vamos a capturar el evento "click" sobre la imagen y dentro de la función se realizará el aumento del contador en uno y el mensaje que indicará la cantidad de clic que se llevan sobre la imagen.

```
$(document).ready(function() {
     let contador = 0;
     $('#ejemplo').on('click', function() {
          contador++;
          console.log('---> Clic '+contador+' vez!');
     });
});
```

- ***Paso 4:*** Al ejecutar el código anterior en el navegador web y si el usuario hace un click sobre la imagen, el resultado por consola sería:

```
---> Clic 1 vez!
```

- ***Paso 5:*** Funcionando hasta el momento el código realizado, procedemos a trabajar con una función externa que nos permitirá regresar la hora, minutos y segundos exactamente cuando el usuario haga un click sobre la imagen. Esto se puede lograr implementando el objeto Date() de JavaScript, que retorna la fecha y el tiempo actual en un solo formato. Pero, este objeto tiene distintos métodos que permiten extraer solo la hora getHours(), los minutos getMinutes(), los segundos getSeconds() e incluso los milisegundos getMilliseconds(). Al combinar todos estos métodos en una sola cadena de texto, podemos armar la hora exacta para cuando el usuario haga click en la imagen, pasando como argumento de la función el objeto new Date().

```
 $(document).ready(function() {
      function formatDate(date) {
           return (date.getHours() < 10 ? '0' : '') + date.getHours() +
':' + (date.getMinutes() < 10 ? '0' : '') + date.getMinutes() + ':' +
(date.getSeconds() < 10 ? '0' : '') + date.getSeconds() + '.' +
(date.getMilliseconds() < 10 ? '00' : (date.getMilliseconds() < 100 ?
'0' : '')) + date.getMilliseconds();
      };

      let contador = 0;
      $('#ejemplo').on('click', function() {
           contador++;
           console.log('--->' + formatDate(new Date()) + ' Clic
'+contador+' vez!');
      });
});
```

- ***Paso 6:*** Al ejecutar el código anterior y partiendo que el usuario hizo dos veces clic sobre la imagen, el resultado por consola sería:

```
--->18:30:19.483 Clic 1 vez!
--->18:31:01.003 Clic 2 vez!
```

Ejercicio propuesto (9)

Muestra la posición del elemento (item 1, item 2, entre otros) antes del primer elemento <ul>, además añade un color de fondo a los impares y otro color de fondo a los pares.

```
<ul>
   <li>item</li>
   <li>item</li>
   <li>item</li>
   <li>item</li>
   <li>item</li>
   <li>item</li>
</ul>
```

Animaciones

Dedicaremos un pequeño apartado a las animaciones en jQuery. Por excelencia, es una de las librerías en donde es más sencillo crear animaciones de forma rápida y efectiva.

Analizaremos un pequeño ejemplo a modo de introducción, de esta forma desglosamos cada una de las acciones presentes.

```
$('.icon-roll').click(function() {
   var $icon = $(this);
   $icon
        .closest('.module')
        .find('.body')
        .toggle('slow', function() {
            $icon.text($(this).is(':hidden') ? '+' : '-');
        });
});
```

Suponemos que existe un elemento con una clase llamada icon-roll, al hacer click en ese elemento se gatilla una función. Como ustedes ya deben saber, al referirnos al parámetro this estamos hablando directamente del elemento padre directo que tenemos. Es decir, en este caso, el elemento seleccionado.

Dicho elemento lo almacenamos en una variable y buscamos el primer ancestro del elemento con clase module, a dicho elemento buscamos otro con clase body y generamos una animación llamada toggle, la cual, recibe como parámetro una instrucción, en este caso slow y además una función, la cuál hará que el botón seleccionado cambie de estado de + a - respectivamente.

Esto traducido a una interfaz, básicamente dicha función está generando una caja que al hacer clic en el ícono + o -, esconde o muestra dicho contenido de una forma fluida y lenta.

Así como existe la función toggle, existen por ejemplo: fadeIn, fadeOut, slideIn, slideDown, entre otras. Invitamos a que investigues sobre las funcionalidades existentes para animar páginas web de forma dinámica.

Ejercicio guiado: Toggle

Desarrollar una aplicación que permita modificar el texto existente por “Latam” cuando el usuario haga click sobre un botón y, cuando vuelva a realizar la acción, el texto debe regresar a su condición original, en este caso “Desafío”. El extracto del HTML es el siguiente:

```
<p>Desafío</p>
<p style="display: none">Latam</p>
<button>Cambio</button>
```

- ***Paso 1*:** Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crea dos archivos, un index.html y un script.js.

- ***Paso 2*:** En el index.html, escribir la estructura básica de un documento HTML, agregar el CDN para tener acceso a jQuery, enlazar el archivo externo de JavaScript y finalmente agregar las etiquetas indicadas al inicio del ejercicio, como se muestra a continuación:

```
<!DOCTYPE html>
<html>
   <head>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width,
initial-scale=1.0">
      <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"><
/script>
      <title>Ejemplo jQuery</title>
   </head>
   <body>
       <p>Desafío</p>
       <p style="display: none">Latam</p>
       <button>Cambio</button>
       <script src="script.js"></script>
   </body>
</html>
```

- ***Paso 3:*** Al ejecutar el código anterior, el resultado en el navegador web será:

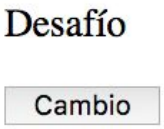

Imagen 7. Resultado en el navegador web.
Fuente: Desafío Latam

- ***Paso 4:*** En el archivo script.js, activar la función principal de jQuery y dentro de ella capturamos el elemento <button> y le asignamos un evento "click" con el método ".on()". Ahora dentro del método ".on()", se debe seleccionar el elemento <p> y aplicar el método ".toggle()". Este método se encarga de modificar el estilo en línea que posean las etiquetas, como la segunda etiqueta <p> contiene un estilo con "display: none", el toggle se encarga de llevarlo a block, mientras que al otro elemento <p>, le modifica la propiedad display a none.

```
$(document).ready(function() {
    $('button').on('click',function(){
        $('p').toggle();
    });
});
```

- ***Paso 5:*** Al ejecutar el código anterior en el navegador web y si el usuario hace un click sobre la el botón, el resultado será:

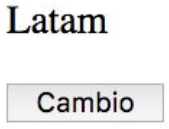

Imagen 8. Resultado en el navegador web.
Fuente: Desafío Latam

Ejercicio propuesto (10)

Muestra el texto que se encuentra en el párrafo con la etiqueta <p>, cuando el usuario haga un click sobre el botón denominado "Mostrar".

```
<button>Mostrar</button>
<p>Desafío Latam</p>
```

Resumen

jQuery es una librería de JavaScript de código abierto, que simplifica y optimiza la escritura de código, permitiendo interactuar con elementos HTML con una sintaxis sencilla, que extiende las posibilidades del lenguaje, produciendo un código más limpio, eficiente y accesible.

Esta es una de las principales razones por las que resulta importante aprender jQuery, puesto que permite manipular elementos del DOM, trabajar con CSS, entre otras acciones que veremos en la siguiente lectura a través de una sintaxis sencilla.

Aprender estos recursos nos permiten construir sitios web mucho más profesionales y entregar una experiencia de usuario mucho más interesante para los visitantes.

Solución de los ejercicios propuestos

1. Transforma el siguiente código de JavaScript a jQuery.

```
let button = $('#button');
button.click(function(){
    alert('click sobre el botón');
});
```

2. Selecciona, almacena en variables y muestra cada una de ellas para los elementos HTML mediante los selectores del tipo id de jQuery.

```
$(document).ready(function() {
   let titulo_1 = $('#titulo_1');
   let titulo_2 = $('#titulo_2');
   let texto_1 = $('#texto_1');
   let btn = $('#btn');
   console.log(titulo_1);
   console.log(titulo_2);
   console.log(texto_2);
   console.log(btn);
});
```

3. Selecciona, almacena en variables y muestra cada una de ellas para los elementos HTML mediante los selectores del tipo class de jQuery.

```
$(document).ready(function() {
   let titulos = $('.titulos');

   console.log(titulos);
});
```

4. Para el siguiente HTML, selecciona mediante los elementos generales todos los <li> que tienen el texto “Nested item”. Igualmente añadele un color al texto, en este caso debe ser el color rojo.

```
$(document).ready(function() {
     $('ul.todos > li > ul > li').css('color', 'red');
});
```

5. Para el siguiente HTML, selecciona mediante los filtros de posición el primer y último <li> perteneciente a la clase denominada “todos”, añadiendo un color al texto a los elementos seleccionados, en este caso debe ser el color rojo.

```
$(document).ready(function() {
     $('ul.todos > li:eq(0) ').css('color', 'red');
     $('ul.todos > li:eq(2) ').css('color', 'red');
});
```

6. Utilizando el filtro hijo nth-child selecciona el segundo elemento de la lista agregando un color de fondo azul.

```
$(document).ready(function() {
     $('ul li:nth-child(2)').css('background-color', 'red');
});
```

7. Escriba un filtro form para el siguiente elemento:

```
$(document).ready(function() {
     let entrada = $('input');
     console.log(entrada)
});
```

8. Cambia el color de fondo del div utilizando el evento de ratón: mouseenter

```
$(document).ready(function() {
     let miCaja = $('.miCaja');
     miCaja.on('mouseenter',function(){
          $(this).css('background-color','green');
     });
});
```

9. Muestra la posición del elemento (item 1, item 2, entre otros) antes del primer elemento <ul>, además, añade un color de fondo a los impares y otro color de fondo a los pares.

```
$(document).ready(function() {
      $('ul li').each(function(index){
            $('ul').before("<p>"+index+": "+$(this).text()+"</p>");
      });
      $('ul li:odd').css('background-color','green');
      $('ul li:even').css('background-color','blue');
});
```

10. Muestra el texto que se encuentra en el párrafo con la etiqueta <p>, cuando el usuario haga un click sobre el botón denominado “Mostrar”.

```
$(document).ready(function() {
      $('button').on('click',function(){
          $('p').show();
      });
});
```