JQuery

> JQuery es una biblioteca de JavaScript que nos permitirá simplificar nuestro código **JS**, haciendo mucho más fáciles tareas comunes como:
>
> - Manipulación HTML/DOM
> - Manipulación CSS
> - Métodos de eventos HTML
> - Efectos y animaciones
> - AJAX
> - Utilidades
>
> De esta manera nuestro código para efectuar estas tareas ahora podría ser sólo una línea de código y de así nuestro código será mucho más limpio.

Instalación de JQuery

> Para agregar **JQuery** en nuestra página debemos agregar el script de **JQuery** en nuestro **HTML** de la siguiente manera:
>
> ```
> <script src="https://code.jquery.com/jquery-3.2.1.min.js"></script>
> ```

Sintaxis

> Para escribir ahora nuestro código **JS** usando **JQuery** tendremos que aprender su sintaxis:
>
> ```
>     $(selector).action();
> ```
>
> donde tenemos lo siguiente:
>
> - ***$*** para definir/acceso a jQuery
> - ***(selector)*** a "buscar o encontrar" elementos HTML
> - ***.action*** para ejercer algún cambio sobre el elemento seleccionado
>
> Al momento de escribir nuestros scripts con **JQuery** es bueno esperar a que nuestro documento **HTML** esté completamente cargado, para eso usaremos el **evento 'ready'** de nuestro documento.

```
$(document).ready(function () {
    // JS code here
});
```

Selectores

En **JQuery** usaremos los mismo selectores que en CSS de la siguiente manera:

- **Nombre de etiqueta**: Con el nombre de etiqueta podremos obtener todas las etiquetas presentes en nuestro documento.

```
<p>Pedro</p>
<p>Juan</p>
<p>Francisco</p>
<script>
    $(document).ready(function () {
      var p = $('p');
      /* nuestra variable p tendrá un arreglo de 3 elementos p
      con javascript sólo sería
      var p = document.getElementsByTagName('p');*/
    });
</script>
```

- **Identificador**: Podremos obtener el elemento que tenga el id seleccionado en el selector.

```
<p>Pedro</p>
<p id="juan">Juan</p>
<p>Francisco</p>
<script>
    $(document).ready(function () {
      var p = $('#juan');
      /* nuestra variable p tendrá el elemento p con id 'juan'
      con javascript sólo sería
      var p = document.getElementById('juan'); */
    });
</script>
```

- **Clase**: Podremos obtener el conjunto de elementos que tengan la clase del selector.

```
<p class="names">Pedro</p>
<p>Juan</p>
<p class="names">Francisco</p>
<script>
    $(document).ready(function () {
      var p = $('.names');
      /* nuestra variable p tendrá un arreglo de 2 elementos p con la clase 'names'
      con javascript sólo sería
      var p = document.getElementsByClassName('names'); */
    });
</script>
```

Eventos

Con **JQuery** también podremos escuchar eventos de nuestro DOM para ello tenemos que seleccionar el elemento y luego escuchar el evento a programar, su sintaxis será:

```
$(selector).on('eventsListenner', function () {...});
/* $ con JQuery seleccionamos el elemento html
.on es el método que nos permitirá escuchar un evento y en sus parametros
recibimos el nombre del evento a escuchar y luego la función donde irá la lógica */
```

- Ejemplos:
  - ***Click***:

    ```
    <button id="button">Click me!</button>
    <script>
    $(document).ready(function () {
      $('#button').on('click', function () {
          window.alert('Botón clickeado');
      });
    });
    </script>
    ```

  - ***Change***:

    ```
    <input id="textfield"/>
    <script>
    $(document).ready(function () {
      $('#textfield').on('change', function () {
          window.alert('Cambio en valor del input');
      });
    });
    </script>
    ```

  - ***Submit***:

    ```
    <form id="form_user">
    ...
    </form>
    <script>
    $(document).ready(function () {
      $('#form_user').on('submit', function () {
          window.alert('Formulario enviado');
      });
    });
    </script>
    ```

  - ***Hover***:

    ```
    <div class="container">
    ...
    </div>
    ```

```
<script>
$(document).ready(function () {
  $('.container').on('hover', function () {
      window.alert('Mouse pasó por encima del div con clase container');
  });
});
</script>
```

De esta manera podremos agregar un comportamiento lógico a nuestra aplicación web, interacción con el usuario, mejorar UX, validar campos de un formulario, etc.

Manipulación de DOM

Obtención de contenidos

Con **JQuery** podremos obtener el contenido de los elementos seleccionados y para eso podemos ocupar 4 métodos.

- ***text()***: Con este método obtendremos todo el texto de los elementos hijos de nuestro nodo selecionado.

```
...
<div id="textExample"> Hola <b>Mundo</b> </div>
<script>
  /* Seleccionamos el elemento con id 'textExample'
  y extraemos el texto de su textNode */
  var text = $("#textExample").text();
  // nuestra variable text ahora tendrá 'Hola Mundo'
</script>
```

- ***html()***: Con este método obtendremos el **textNode** y elementos completos de nuestro nodo seleccionado, todo como una cadena de caracteres.

```
...
<div id="textExample"> Hola <b>Mundo</b> </div>
<script>
  /* Seleccionamos el elemento con id 'textExample' y
  extraemos el texto de su textNode incluido el html */
  var value = $("#textExample").html();
  // nuestra variable value ahora tendrá 'Hola <bMundo</b>'
</script>
```

- ***val()***: Con este método podemos obtener el valor de un elemento, este hace referencia al atributo **value** de una etiqueta.

```
...
<input type="text" id="test" value="Hello World!"/>
<script>
  /* Seleccionamos el elemento con id 'test' y
```

```
  extraemos el texto de su atributo value */
  var text = $("#test").val();
  // nuestra variable text ahora tendrá 'Hello World!'
</script>
```

- ***attr(param)***: Con este método podemos obtener el valor del atributo seleccionado de un elemento.

```
...
<input type="text" id="test" value="Hello World!"/>
<script>
  /* Seleccionamos el elemento con id 'test'
  y extraemos el texto de su atributo a elegir */
  var type = $("#test").attr('type');
  // nuestra variable type ahora tendrá 'text'
</script>
```

Modificador de contenidos

Con los 4 métodos anteriores obteniamos el contenido pero también utilizando los mismos podremos actualizar el contenido, el nuevo valor será el parámetro ocupado.

- ***text(params)***:

```
...
<div id="textExample"> Hola <b>Mundo</b> </div>
<script>
  /* Seleccionamos el elemento con id 'textExample'
  y extraemos el texto de su textNode */
  $("#textExample").text('Hello World!');
  // Ahora nuestro div mostrará 'Hello World!'
</script>
```

- ***html(params)***:

```
...
<div id="textExample"> Hola <b>Mundo</b> </div>
<script>
  /* Seleccionamos el elemento con id 'textExample'
  y extraemos el texto de su textNode */
  $("#textExample").html('Hello <b>World!</b>');
  /* Ahora nuestro div mostrará 'Hello World!'
  con la ultima palabra en negrita */
</script>
```

- ***val(params)***:

```
...
<input type="text" id="test" value="Hello World!"/>
<script>
  /* Seleccionamos el elemento con id 'test'
  y extraemos el texto de su atributo value */
  $("#test").val('Hola Mundo');
  /* Nuestro input dirá 'Hola Mundo'
```

```
    en vez de 'Hello World!' */
  </script>
```

- ***attr(param, new_value)***:

```
  ...
  <input type="text" id="test" value="Hello World!"/>
  <script>
    /* Seleccionamos el elemento con id 'test'
    y extraemos el texto de su atributo a elegir */
    $("#test").attr('type', 'password');
    /* Ahora nuestro input no mostrará su
    value por que será de tipo password */
  </script>
```

Añadir elementos hijos (childs-nodes)

Para añadir elementos como lo haciamos con el método **appendChild** en **JS**, con **JQuery** podremos agregar elementos hijos antes o después del **textNode** que ya tenga el elemento seleccionado.

- ***prepend(param)***: Agrega un nodo antes del **textNode**.

```
  ...
  <p id="paragraph"> Hola </p>
  <script>
    /* Seleccionamos el elemento con id 'paragraph'
    y agregamos el texto a su textNode */
    $("#paragraph").prepend('<b>Mundo</b>, ');
    // Ahora nuestro div mostrará 'Mundo, Hola'
  </script>
```

- ***append(param)***: Agrega un nodo después del **textNode**.

```
  ...
  <p id="paragraph"> Hola</p>
  <script>
    /* Seleccionamos el elemento con id
    'paragraph' y agregamos el texto a su textNode */
    $("#paragraph").append(', <b>Mundo</b>');
    // Ahora nuestro div mostrará 'Hola, Mundo'
  </script>
```

Añadir elementos hermanos (siblings-nodes)

Para añadir elementos como lo haciamos con el método **appendChild** en **JS**, con **JQuery** podremos agregar elementos hijos antes o después del **textNode** que ya tenga el elemento seleccionado.

- ***before(param)***: Agrega el nodo hermano **antes** del nodo seleccionado.

```
  ...
  <ul>
```

```
    <li id="listItem">Gato<li>
</ul>
<script>
  /* Seleccionamos el elemento con id 'listItem'
  y agregamos el nodo hermano anterior */
  $("#listItem").before('<li>Perro</li>');
  // Ahora nuestra lista mostrará ' Perro, Gato'
</script>
```

- ***after(param)***: Agrega el nodo hermano **después** del nodo seleccionado.

```
...
<ul>
    <li id="listItem">Gato<li>
</ul>
<script>
  /* Seleccionamos el elemento con id 'listItem'
  y agregamos el nodo hermano siguiente */
  $("#listItem").after('<li>Perro</li>');
  // Ahora nuestra lista mostrará ' Gato, Perro'
</script>
```

Eliminar elementos

Con el uso de **JQuery** también podremos eliminar nodos de nuestro **DOM** o vaciar su contenido (Eliminar sus hijos), para ello podremos utilizar los siguientes métodos:

- ***remove()***: Con este método eliminaremos el nodo seleccionado.

```
...
<div id="primerDiv">Primer Div</div>
<div id="segundoDiv">Segundo Div</div>
<div id="tercerDiv">Tercer Div</div>
<script>
  /* Seleccionamos el elemento con id
  'primerDiv' y lo eliminamos del DOM */
  $("#primerDiv").remove();
  // Ahora nuestra página tendrá sólo 2 div
</script>
```

- ***empty()***: El método empty permitirá eliminar todos los **nodos hijos** del elemento seleccionado de está manera quedará **vacío**.

```
...
<div id="primerDiv">
    <div id="segundoDiv">Segundo Div</div>
    <div id="tercerDiv">Tercer Div</div>
</div>
<script>
  /* Seleccionamos el elemento con id
  'primerDiv' y lo eliminamos del DOM */
  $("#primerDiv").empty();
  // Ahora nuestro div quedará así '<div id="primerDiv"></div>'
</script>
```

Efectos

JQuery provee de varios efectos comunes para agregar a nuestra página web, dentro de los más comunes tenemos:

- ***Hide/Show***: Para ocultar y mostrar elementos con un suave estilo.

```
// Se ocultará los elementos p al hacer click en el elemento con id 'hide'
$("#hide").on('click', function () {
  $("p").hide();
});
// Se mostrará los elementos p al hacer click en el elemento con id 'show'
$("#show").on('click', function () {
  $("p").show();
});
```

- ***Fade***: Se utiliza para desvanecer el elemento seleccionado, para esto usaremos el método **fadeToggle**:

```
$("#div1").fadeToggle(); // el nodo aparecerá/desaparecerá con la velocidad normal del
$("#div2").fadeToggle("slow");  // el nodo aparecerá(desaparecera lento
$("#div3").fadeToggle(3000); // el nodo aparecerá/desaparecerá en 3 segundos
```

- ***Slide***: El efecto slide se ocupa para deslizar elementos, para utilizarlo ocuparemos el método **slideToggle**.

```
$("#panel1").slideToggle();
$("#panel2").slideToggle("slow");
$("#panel3").slideToggle(3000);
```

- ***Animate***: Es el método que nos permite crear animaciones personalizadas, su sintaxis será

```
$(selector).animate({
  cssPropertyName1: 'value1',
  cssPropertyName2: 'value2',
  ...
  cssPropertyNameN: 'valueN',
});
```

Ejemplo*:

```
<style>
  /* Estilo inicial del div*/
  div {
    background:#98bf21;
    height:100px;
    width:100px;
    position:absolute;
  }
```

```
    </style>
    ...
    <div>
        <!--Este div sufrirá un cambio en css mediante la animación de JQuery-->
    </div>
    <script>
      // Seleccionamos el div y aplicamos la animación con sus estilos
      $("div").animate({
        left: '250px',
        opacity: '0.5',
        height: '150px',
        width: '150px'
      });
    </script>
    ...
```