MANEJO DEL DOM CON JQUERY

Modelo de Objetos del Documento

Primero… ¿Qué es jQuery?

- jQuery es un **framework basado en javaScript** para programación de páginas y aplicaciones web.
- Un framework es un **marco de trabajo** que permite **programar de modo más accesible y directo**. Llamados también librerías.
- Son como una **caja de herramientas** que nos permiten tener algunas **acciones resueltas para utilizarlas de modo directo y sencillo**, no hace falta programar todo (como con JS).
- Las herramientas se **construyen con funciones** que vamos llamando para acceder a esas acciones que queremos lograr.

Primero… ¿Qué es jQuery?

Para poder programar con jQuery, primero lo debemos linkear al html por encima de nuestro código:

1) Localmente

```
<script src="js/jquery.js"><script>
<script>
      // Acá nuestro código que usa jQuery
</script>
```

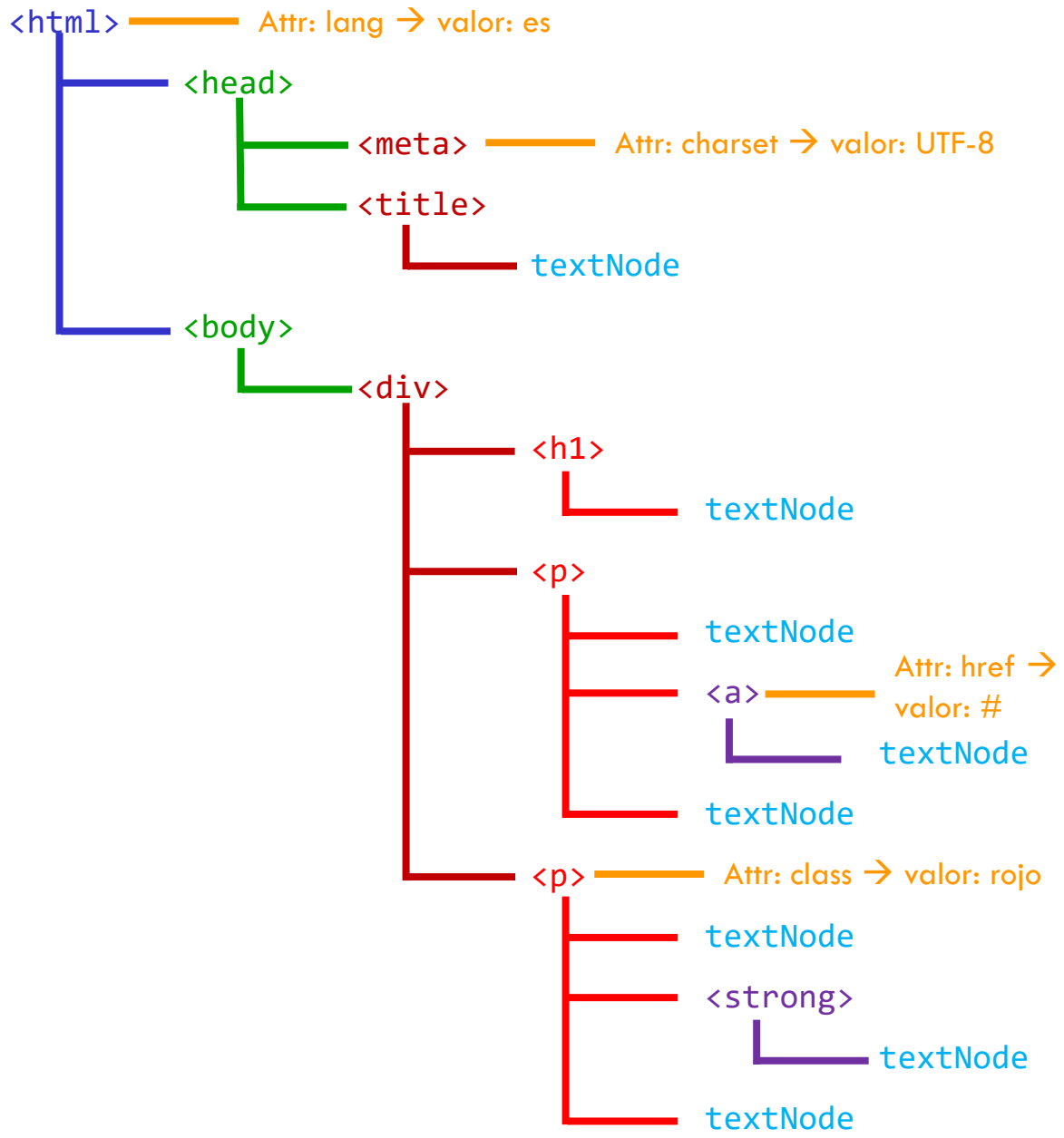
2) Desde un CDN (Content Delivery Network), por ejemplo, el de Google:

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.6.0/jquery.min.js">
</script>
<script>
      // Acá nuestro código que usa jQuery
</script>
```

El DOM

- Es una sigla: **Document Object Model** o **Modelo de Objetos del Documento** que trata a todas las **etiquetas** html como **nodos** y a los **textos** que contengan como **nodos de texto**.
- Cuando se **carga el html** se establece un **árbol de nodos** con las etiquetas para **poder accederlas** o **seleccionarlas**, "siguiendo una **ruta** o **secuencia de maquetación**" muy parecido a como funciona la **regla en cascada** de CSS.
- También se puede **obtener** cualquier **característica** de los nodos (etiquetas), es decir, a sus **atributos**.

```
<!DOCTYPE html>
<html lang="es“>
<head>
    <meta charset="UTF-8">
    <title>Ejemplo DOM</title>
</head>
<body>
    <div>
        <h1>El DOM</h1>
        <p>Soy un párrafo con <a href="#">un
        vínculo</a> adentro</p>
        <p class="rojo">Soy un párrafo con
        <strong>un strong</strong> adentro</p>
    </div>
</body>
</html>
```



DOM desde jQuery

- Existe **una restricción**: para poder **seleccionarlas** las etiquetas **deben existir** (debe haber sido leída por el browser). El **script debe incluirse** como respuesta al **evento de carga del html** o **antes del cierre del body** (ya se crearon las etiquetas).
- La función **`jQuery( )`** o **`$( )`**, es la **encargada de seleccionar etiquetas** del árbol de nodos del **DOM**.
- Recibe entre los paréntesis, **sintaxis CSS** (cualquier tipo de regla) para acceder o seleccionar a una (o varias) etiqueta html.

```
<div id="caja">Soy un DIV</div>
<script>
   var caja = $('#caja');
</script>
```

DOM desde jQuery

- Existe **una restricción**: para poder **seleccionarlas** las etiquetas **deben existir** (debe haber sido leída por el browser). El **script debe incluirse** como respuesta al **evento de carga del html** o **antes del cierre del body** (ya se crearon las etiquetas).
- La función **`jQuery( )`** o **`$( )`**, es la **encargada de seleccionar etiquetas** del árbol de nodos del **DOM**.
- Recibe entre los paréntesis, **sintaxis CSS** (cualquier tipo de regla) para acceder o seleccionar a una (o varias) etiqueta ht

```
<div id="caja">Soy un DIV</di
<script>
    var caja = $('#caja');
</script>
```

La función $() devuelve un OBJETO de jQuery que contiene a la/s etiqueta/s seleccionada/s (si vemos su estructura por consola notaremos que es muy parecido a un array donde en cada posición se guardó una etiqueta html).

Métodos de JQ

- Ese **object de JQ** tiene **métodos** (funciones) inventados por jQuery **para manipular** la/s etiqueta/s seleccionada/s.
- Algunas de esos métodos son **setters** (asignan valor) y otros son **getters** (traen u obtienen un valor).
- También tiene **funciones utilitarias** que no son ni setters ni getters sino que **cumplen una determinada función**.
- **TODOS** los métodos (setters, getters y utilitarios**) se acceden** a partir del **object** (el que obtuvimos al seleccionar la etiqueta) **por medio de puntos**.

Métodos de JQ

Dos opciones de trabajo:

1) Seleccionamos la etiqueta y la guardamos en una varible. Luego accedemos a los métodos por medio de ella

```
<div id="caja">Soy un DIV</div>
<script>
      var caja = $('#caja');
      caja.método();
</script>
```

2) Seleccionamos la etiqueta y accedemos directamente a los métodos sin guardarla en ningún lado:

```
<div id="caja">Soy un DIV</div>
<script>
      $('#caja').método();
</script>
```

Métodos para propiedades básicas

- Para **obtener propiedades** de las etiquetas tenemos estos métodos:
  - `$('#caja').html( ); // Pide el contenido de texto y etiquetas hijas de #caja.`
  - `$('#caja ').text( ); // Pide el contenido de #caja como texto plano.`
  - `$('#caja').val( ); // Pide el value de un control de formulario`
- **Sin argumentos**, se está **pidiendo** el valor (funcionan como getters).
  Con argumento, se **setea** o **reemplaza** el valor (funcionan como setters).
  - `$('#caja').html('Texto con <strong>etiquetas</strong>');`
  - `$('#caja').text('Texto plano');`
  - `$('#caja').val('Nuevo valor');`
- Para **cualquier atributo** de una etiqueta se usa la función **.attr()**
  - `$('#foto').attr('src'); // Pide el atributo source de la imagen.`
  - `$('#foto').attr('src', 'foto.png'); // Cambia el valor del atributo source.`

Manejo de propiedades CSS

- Para manejar **una propiedad CSS** se usa la función **`.css( )`** que recibe **uno** o **dos** argumentos (ambos como string).
- **Un solo argumento, obtiene el valor de la propiedad** que se pasa como parámetro.

```
/* OBTENER el valor de la propiedad COLOR */
var color = $('#caja').css( 'color' );
```

- Con **un segundo argumento** se setea **el nuevo valor** de esa propiedad.

```
/* CAMBIAR el valor de la propiedad COLOR */
$('#caja').css( 'color', 'green' );
```

- También **acepta un OBJECT de `js`** para modificar varias propiedades a la vez:

```
$('#objeto').css( {color:'purple', backgroundColor:'pink'} );
```

Crear etiquetas desde JQ

- La función **$( )** también sirve para **crear nuevas etiquetas** si **recibe**, como **string**, la **etiqueta** a crear, la función retorna un **OBJECT de jQuery** con la etiqueta creada.
- **IMPORTANTE**: la etiqueta la recibe con **sintáxis HTML** (con <>), sino va a interpretar que debe BUSCAR y seleccionar esa etiqueta:

```
var divs = $('div'); // Busca todas las etiquetas DIV.
var nuevoDiv = $('<div></div>'); // Crea un DIV vacío.
```

- Una vez creado, ese object tiene todos los métodos de jQuery: **.text( ), .val( ), .css( )**, etc.

Agregar la etiqueta creada al html

- Para **agregar** elementos al DOM hay dos funciones:

```
$('#contenedor').append( objeto_nuevo ); // Agrega el objeto_nuevo como último hijo de #contenedor seleccionado con la función $()

$('#contenedor').prepend( objeto_nuevo ); // Lo agrega como primer hijo.
```

- El argumento puede ser un OBJECT de jQuery (etiqueta creada desde código) o un **bloque de código HTML** bien formado como string:

```
$('#padre').append('<p>Nuevo <em>párrafo</em></p>' );
```

Otros métodos

- También se pueden hacer las siguientes acciones:

```
$('#objeto').after('<p>texto</p>'); // Agrega el elemento después de #objeto.
$('#objeto').before('<p>texto</p>'); // Lo agrega antes de #objeto
$('#objeto').wrap('<p></p>'); // Encierra a #objeto en la etiqueta indicada: <p>#objeto</p>.
$(‘#padre').children( ); // Obtiene todos los hijos de #padre
$(‘#padre').children(‘strong'); // Obtiene solo los hijos de un tipo.
```

- Y hay métodos para **moverse en el DOM**:

```
$('#objeto').next( ); // Selecciona el nodo siguiente de #objeto
$('#objeto').prev( ); // Selecciona el nodo anterior de #objeto
$('#objeto').parent( ); // Selecciona el nodo padre de #objeto
$('#objeto').parents( ); // Selecciona todos los padres de #objeto
```

Eliminar eltiquetas con JQ

- Para **eliminar elementos al DOM**:

```
$('#objeto').remove( ); // Busca al elemento #objeto y lo elimina de su contenedor.

$('#objeto').remove('.clase' ); // Busca al elemento #objeto y se fija si tiene aplicada la clase .clase, si es true lo quita (un filtro).
```

- Si se selecciona más de un elemento, remove( ) **elimina todos**:

```
$(‘ul li’).remove( ); // Elimina todos los li de la ul.
```