

## HINTS

### LISTA DE NODOS VS ARRAY

La lista que recibimos cuando capturamos nodos, por ejemplo, desde su clase, se parece mucho a un **Array**, pero son estructuras distintas, ya que la lista de Nodos (**NodeList**) es una colección de nodos del **DOM** y el **Array** es un tipo de dato propio de **JavaScript**, en el cual podemos almacenar cualquier dato. En el caso de ambos elementos, se accede a su largo por el método `length()` y a sus índices utilizando corchetes (`[]`), sin embargo, **NodeList** no cuenta con métodos como `map()`, `filter()` o `reduce()`.

### ATRIBUTOS

Además de la creación de nodos, podemos incorporar o modificar los atributos de cada uno. Para obtener el atributo, solamente debemos indicarlo:

```
1 dato.id;
```

También podemos utilizar el método `set()` para establecer un valor:

```
1 dato.setAttribute("id", "nombre");
```

Podemos hacer esto para indicar cualquier atributo, por ejemplo, un `id`, `class`, `for`, `placeholder`, etc.

Finalmente, también podemos eliminar un atributo usando la función `removeAttribute()`:

```
1 dato.removeAttribute("id");
```

## MANEJADORES DE EVENTOS

No solamente podemos manejar eventos a través de funciones externas (que es la forma más recomendable) como ya aprendimos en la lectura. Podemos manejar eventos como atributos, siendo esta la manera más sencilla pero menos eficaz, ya que convierte el código en menos manejable y legible, sin embargo, quizás podemos hacer uso de ella cuando la acción sea muy acotada. Para hacer esto, vamos a colocar la acción inmediatamente en la etiqueta **HTML**, de la siguiente manera:

```
1 <input type="button" value="pinchame" onclick="alert('hola') ">
```

Obteniendo como resultado, que cada vez que se presione el botón, se muestre una alerta con la frase “Hola”.

## VARIABLE THIS

Existe en **JavaScript** una variable especial llamada **this** que se crea automáticamente y hace referencia al elemento que ha provocado el evento. Para estos casos, indicas, haciendo uso de esta variable en el evento, que acción se ejecutará:

```
1 <div class="caja1" style="padding: 10px; background-color:
2 chartreuse;" onclick="this.style.backgroundColor='red';" >
3
4 <p>Caja que cambia de color</p>
5
6 </div>
```

Al hacer clic sobre el **<div>** cambiará su color de chartreuse a red.

En el caso de trabajar con funciones y utilizar esta variable, se pasará como variable a la función:

```
1 <div class="caja1" style="padding: 10px; background-color:
2 chartreuse;" onclick="cambiarColor(this);" >
3
4     <p>Caja que cambia de color</p>
5
6 </div>
```

## EVENTOS SIN DEFINICIÓN EN LA ETIQUETA HTML

Es posible trabajar con **JavaScript** puro y no indicar el evento directamente en la etiqueta **HTML**, para mantener el código fuente de nuestro archivo libre de código **JavaScript**. Esto lo haremos de la siguiente manera:

```
1 <div id="caja1" style="padding: 10px; background-color: rgb(231, 253,
2 208);" >
3
4     <p>Pinchame</p>
5
6 </div>
```

Y el JavaScript será:

```
1 function muestraMensaje() {
2     alert("hola");
3 }
4
5
6 document.getElementById("caja1").onclick = muestraMensaje;
```

## EVENTOS DE TECLADO

Además de los eventos que vimos en la lectura, que podemos llamar como eventos de ratón o mouse, existen eventos que se producen sin que el usuario haga algo específico, como podría ser eventos que se activen con solo la carga de la página o eventos de teclado, en los cuales se activará la acción a través del uso del teclado, por ejemplo, el evento `keydown()` que se produce cuando se pulsa cualquier tecla del teclado.

## USO DEL METODO HTML()

Para acceder al contenido de una etiqueta **HTML**, utilizamos el método `html()`. Si no lo colocamos en la declaración, estaremos accediendo a la etiqueta con todas sus propiedades. Veamos un ejemplo de ambos casos: utilizando el método `html()`.

```
1 $(document).ready(function() {  
2  
3 console.log($("#ol > li > a:eq(6)").html());  
4 });  
5
```

---

Elemento 7

---



```
1 $(document).ready(function() {  
2     console.log($("#ol > li > a:eq(6)"));  
3  
4 });  
5
```

```
▼ S.fn.init(1) ⓘ  
  ► 0: a  
    length: 1  
  ► prevObject: S.fn.init [document]  
  ► __proto__: Object(0)
```

## EVENTO ON CHANGE

Este evento se dispara en elementos `<input>`, `<select>` y `<textarea>` cuando el usuario confirma el valor de un elemento.

Si tenemos un elemento `<select>`:


```
1 <label>Elija un color:</label>  
2  
3     <select id="nieve" onchange="miFuncion()" name="nieve">  
4         <option value="">Seleccione Uno ...</option>  
5         <option value="verde">Verde</option>  
6         <option value="amarillo">Amarillo</option>  
7         <option value="rojo">Rojo</option>  
8     </select>  
9  
10  
11     <div id="resultado"></div>
```



Podemos agregar en él el evento **onchange** y con JavaScript indicar la acción de la función.

```
1 function miFuncion() {  
2     var captura = document.querySelector("#nieve");  
3     var valor = captura.value;  
4     document.getElementById("resultado").innerText = `Usted ha  
5 seleccionado ${valor}`  
6  
7 }
```

Y obtendremos como resultado una respuesta a cada acción del usuario:

Elija un color:    
Usted ha seleccionado amarillo