

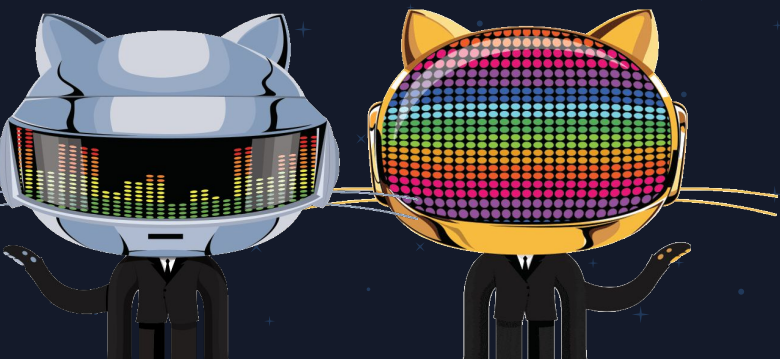
DOM Parte 2

Cordero Hernández, Marco R



En sesiones pasadas...

- **DOM**
 - Introducción
 - Navegación
 - Búsqueda de elementos
 - Colecciones



01

Atributos y propiedades

02

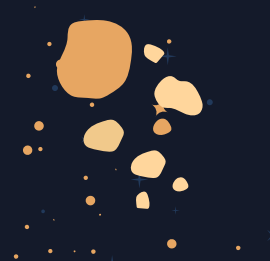
Modificación del documento

03

Operaciones adicionales

04

Eventos



01

ATRIBUTOS Y PROPIEDADES





Atributos y propiedades

- **Atributos:** Se refiere a la *información* asociada a los elementos del HTML
 - Atributos de HTML (name, text, placeholder, ...)
 - De tipo string
 - No son case-sensitive
- **Propiedades:** Se refiere a lo que está *dentro* de los objetos del DOM
 - *Todo* atributo de HTML tiene una propiedad equivalente asociada
 - Tiene un tipo particular según la especificación
 - *Sí es case-sensitive*



Métodos de atributos

- **elem.hasAttribute(nombre)**
 - Revisa existencia de atributo (true o false)
- **elem.getAttribute(nombre)**
 - Obtiene el valor de un atributo (si existe)
- **elem.setAttribute(nombre, valor)**
 - Asigna un valor a una propiedad
- **elem.removeAttribute(nombre)**
 - Elimina el atributo de un elemento
- **elem.attribute**
 - *Colección* de todos los atributos



Métodos de atributos

- Cuando se trabaje con propiedades, la recomendación es utilizar las propiedades predefinidas del **DOM**.

Las excepciones son:

- Cuando se quiere acceder a atributos creados por el usuario con “data-**algo**” → `document.body.dataset.algo`
- Cuando se quiere revisar tal cual la forma en que el valor de alguna propiedad está especificada (para revisar el string)

02

MODIFICACIÓN DEL DOCUMENTO

Crear e insertar

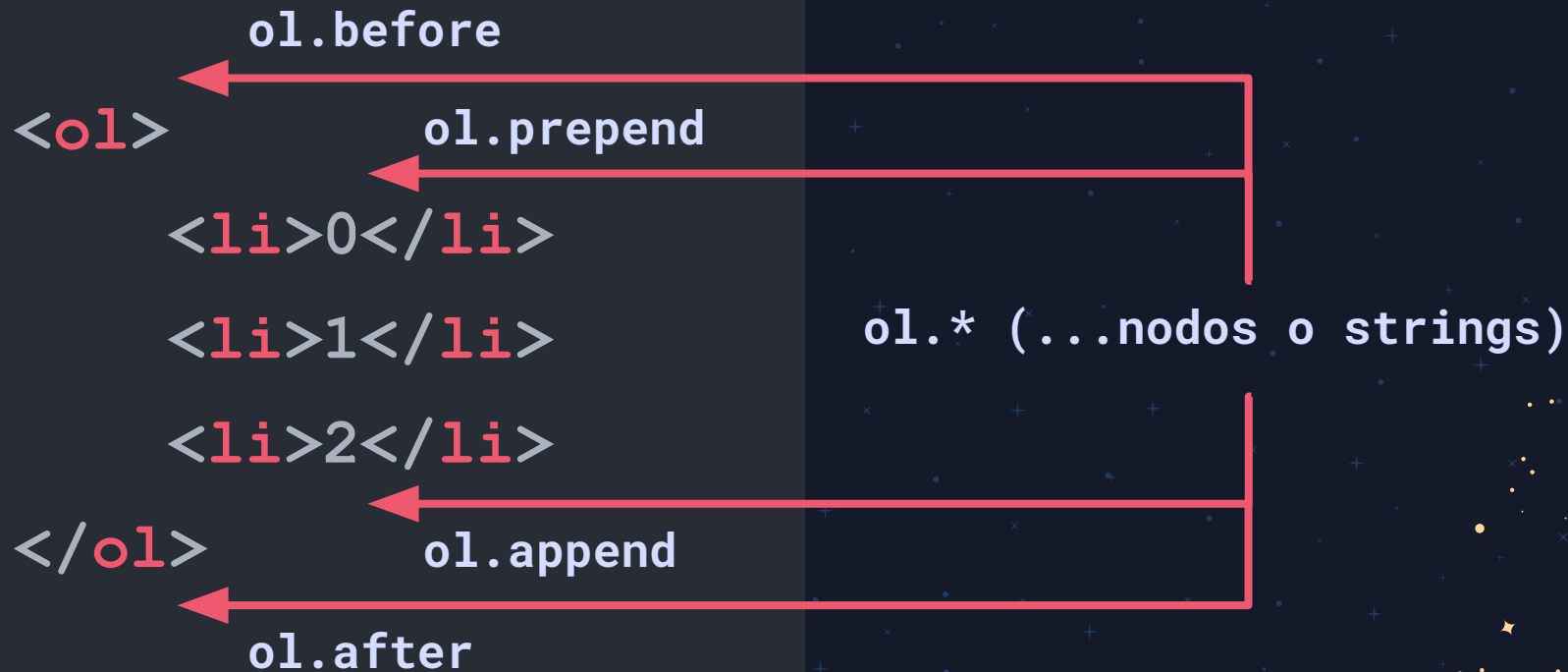
Crear elementos y nodos de texto

- `let varElem = document.createElement(tag)`
- `let varText = document.createTextNode(texto)`

Insertar los elementos/nodos

- `elementoPadre.appendChild(nodo)`
 - Añade un hijo al final
- `elementoPadre.insertBefore(nodo, siguienteHermano)`
 - Añade el nodo justo antes del *siguienteHermano*
- `elementoPadre.replaceChild(nodo, hijoPorBorrar)`
 - Reemplaza el *hijoPorBorrar*
- `nodo.[prepend | append | before | replaceWith](...nodos o strings)`
 - Añade el nodo especificado al inicio, al final, antes, después o lo reemplaza por los nuevos nodos o cadenas dadas

Crear e insertar



Ejercicio

Crea un nuevo HTML y pon una lista *no ordenada* que tenga 2 elementos arbitrarios (pueden ser • HTML y • CSS)

En la consola (o desde un archivo externo) realiza lo siguiente:

- Crea un párrafo con otro texto arbitrario (ej. Git) y guárdalo en una variable
- Crea un párrafo con *otro* texto arbitrario (ej. MongoDB) y guárdalo en una variable
- Inserta el primer párrafo antes de la lista no ordenada
- Inserta el segundo párrafo después de la lista no ordenada
- Crea 3 elementos ***li*** con más texto aleatorio
 - Inserta uno como el primer elemento de la lista
 - Inserta otro entre HTML y CSS (los dos originales)
 - Inserta otro después de CSS (al final de la lista)

Insertar HTML

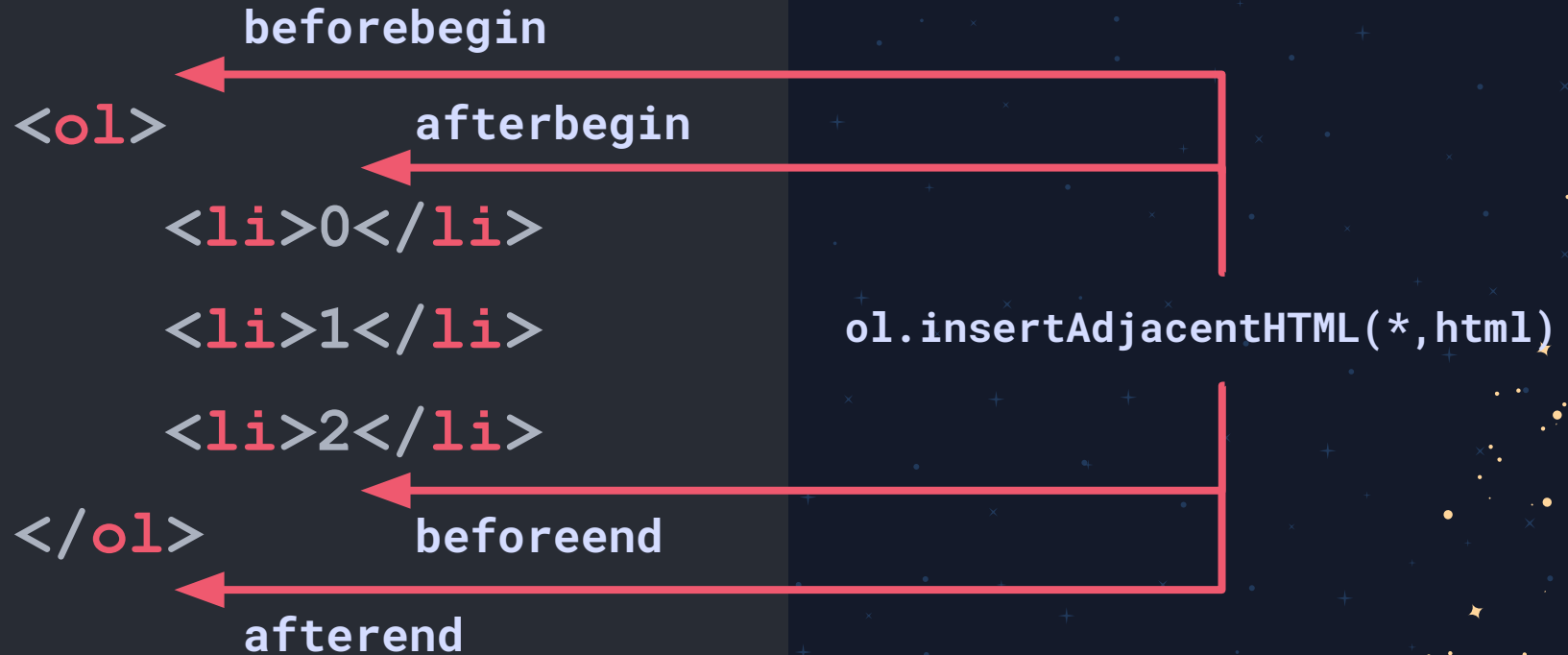
Para insertar contenido HTML es posible usar

- `elem.insertAdjacentHTML(target, html)`
- `elem.insertAdjacentText(target,html)`
- `elem.insertAdjacentElement(target,html)`

Donde **target** puede hacer referencia a

- `'beforebegin'` - Insertar antes del elemento
- `'afterbegin'` - Insertar dentro del elemento al inicio
- `'beforeend'` - Insertar dentro del elemento al final
- `'afterend'` - Insertar después del elemento

Insertar HTML





03

**OPERACIONES
ADICIONALES**

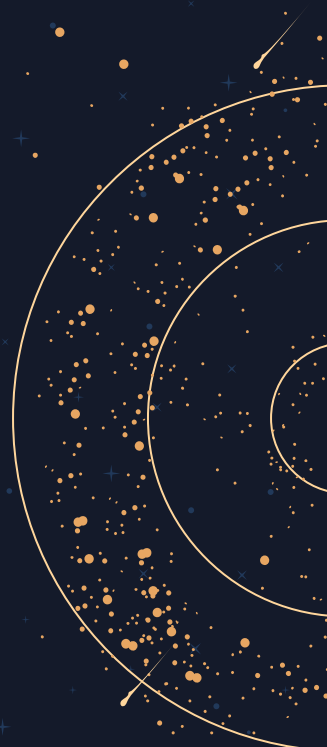
Operaciones adicionales

Clonación de nodos

- Clonación con *todos los atributos y sub-elementos*
 - `elem.cloneNode(true)`
- Clonación sin los elementos hijos
 - `elem.cloneNode(false)`

Eliminar nodos

- `parentElem.removeChild(node)`
- `node.remove()`



Ejercicio

- Crea una lista desordenada con 5 elementos
- Añade un id a la lista y a algún elemento de la lista
- Clona toda la lista y guárdala en una variable **nLista**
- De la variable **nLista** selecciona los elementos que tengan el atributo **id**
- Quita ese atributo de cada elemento
- Que aparezca tu **nLista** en el html después de la otra lista
- Elimina de la primera lista el segundo elemento



The background is a deep navy blue space filled with numerous small, light blue stars and cross-shaped constellations. Two large, stylized spiral galaxies are positioned in the corners: one in the top-left and one in the bottom-right. These galaxies are composed of concentric, swirling bands of dark blue and gold dots, with a dense, bright gold core. Several golden streaks, representing meteors or comets, are scattered across the scene, with one prominent streak in the upper right and others in the lower left and middle left.

04

EVENTOS

Eventos

Un **evento** es una señal de que algo ha ocurrido. Todos los nodos del DOM generan eventos.

Algunos ejemplos de eventos comunes son:

- **click**: Cuando el mouse da click a un elemento
- **contextmenu**: Cuando el mouse da click derecho en algún elemento
- **mouseover / mouseout**: Cuando el cursor “entra” o “sale” de un elemento

Eventos



- ***mousedown / mouseup***: Cuando un botón del mouse se presiona o se libera
- ***keydown / keyup***: Cuando se presiona o se libera una tecla
- ***DOMContentLoaded***: Cuando el HTML ya se haya cargado y procesado por completo
- ***change***: Genérico; cuando se modifica el valor o el contenido de algún elemento, en particular de formularios, input, select, textarea...

<https://www.toptal.com/developers/keycode>



04.1

EVENT HANDLERS

Handlers por atributos

Un **handler** es una *función* que se ejecuta cuando ocurre un evento particular sobre un *nodo*.

Existen múltiples formas de asignar un **handler**.

La forma directa de asignar un handler desde **HTML** es mediante atributos con la sintaxis **onevento**

Handlers por atributos



```
<button onclick="alert('Hola mundo')">Botón 1</button>
<button onclick="funcionExterna()">Botón 2</button>
<script>
  function funcionExterna() {
    alert('Hola mundo');
  }
</script>
```



Handlers por propiedad

Otra manera de asignar un handler es usar la propiedad del *DOM* onevento (directamente desde **JS**).

Solo se puede tener un handler por elemento, por lo cual, si se declaran dos, el último sustituirá al anterior

```
<button id="btn3">Botón 3</button>
<script>
    btn3.onclick = otraFuncion; // Sin
    paréntesis
    function otraFuncion() {
        alert('Hola mundo');
    }
</script>
```



Handlers por propiedad



```
<button id="btn4">Botón 4</button>
<script>
  // Callback directo en asignación
  btn4.onclick = function () {
    alert('Botón 4 presionado');
  }
</script>
```



Ejercicio



Crea un **HTML** que tenga 2 botones:

- Uno que diga *incrementar*
- Otro que diga *decrementar*

Después de estos, muestra un párrafo con texto **0**.

Al hacer click en los botones, el valor del párrafo debe incrementar o decrementar el valor mostrado en el párrafo.