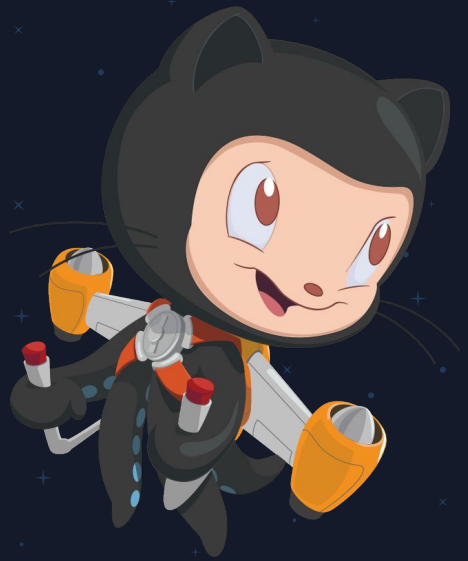
Introducción a DOM

Cordero Hernández, Marco R

En sesiones pasadas...



- Sistema de archivos (fs)
- Postman
- Express Router
- Páginas estáticas

01

Introducción al DOM

02

Navegación del DOM

03

Búsqueda de elementos

04

Colecciones

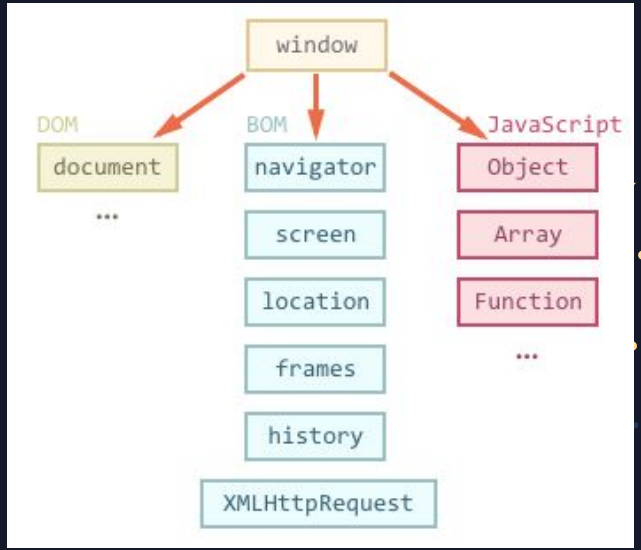
01

INTRODUCCIÓN AL DOM

Introducción

JS permite el control del sitio web de manera *dinámica*. Para lograr esto, implementa el **D**ocument **O**bject **M**odel (**DOM**):

- Da acceso al contenido de la página
- Permite crear y cambiar lo que sea
- Describe la estructura del documento



Introducción

A través del **DOM**, es posible acceder al objeto ***window***, el cual hará referencia al objeto raíz o global de un navegador.

También se cuenta con el **B**rowser **O**bject **M**odel (**BOM**):

- Provee objetos que dan información adicional al **DOM**
- Otorga información sobre el navegador y el sistema operativo
- Permite obtener y cambiar la URL
- Permite manipular el tamaño de la ventana
- … entre muchos otros

```html
<!DOCTYPE html>
<html>
<head>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <!-- Comentario -->
    <h1 class="head1">Título</h1>
    <hr>
    <p id="p1"> Hola</p>
</body>
</html>
```

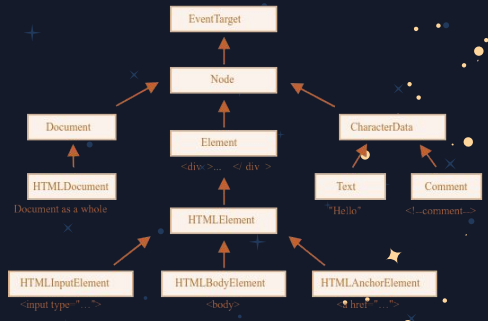
```
├─DOCTYPE: html
└─HTML
  ├─HEAD
  │ └─LINK rel="stylesheet" href="style.css"
  ├─#text:
  └─BODY
    ├─#text:
    ├─#comment: comentario
    ├─#text:
    ├─H1 class="head1"
    │ └─#text: Título
    ├─#text:
    ├─HR
    ├─#text:
    ├─P id="p1"
    │ └─#text: Hola
    └─#text:
```

nodeType

Para visualizar el tipo de nodo se puede usar la propiedad **nodeType**.

Los 4 tipos de nos más comunes son:

- **Documento** (9): El punto de entrada al DOM
- **Nodos de elementos** (1): Etiquetas de HTML
- **Nodos de texto** (3): Contienen texto
- **Comentarios** (8): Pueden leerse desde el DOM



nodeType

```
ELEMENT_NODE = 1;
ATTRIBUTE_NODE = 2;
TEXT_NODE = 3;
CDATA_SECTION_NODE = 4;
ENTITY_REFERENCE_NODE = 5; // Histórico
ENTITY_NODE = 6; // Histórico
PROCESSING_INSTRUCTION_NODE = 7;
COMMENT_NODE = 8;
DOCUMENT_NODE = 9;
DOCUMENT_TYPE_NODE = 10;
DOCUMENT_FRAGMENT_NODE = 11;
NOTATION_NODE = 12; // Histórico
```

Manipulación de contenido

Para obtener y modificar el contenido de un elemento es posible usar los siguientes atributos:

- `elemento.`***`innerText`***
  - Manipula el texto inmediato de un elemento (solo para etiquetas HTML)
- `elemento.`***`innerHTML`***
  - Manipula texto, etiquetas y propiedades dentro de un elemento (*peligroso*)
- `nodo.`***`textContent`***
  - Manipula todo el texto de un nodo o elemento

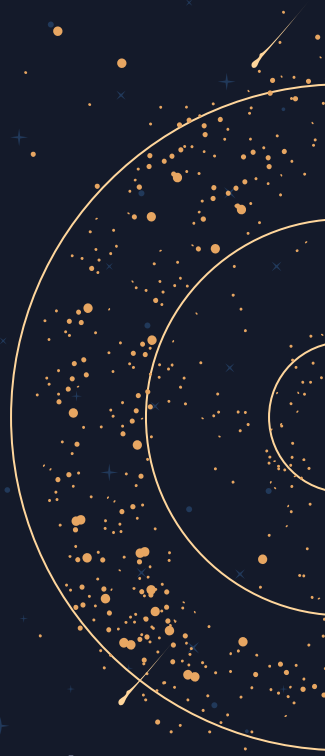
02

NAVEGACIÓN DEL DOM

Navegación

Usando ***document***

- **`document.documentElement`**
  - Elemento HTML raíz
- **`document.body`**
  - Elemento **<body>** *si es que existe*
- **`document.head`**
  - Elemento **<head>**

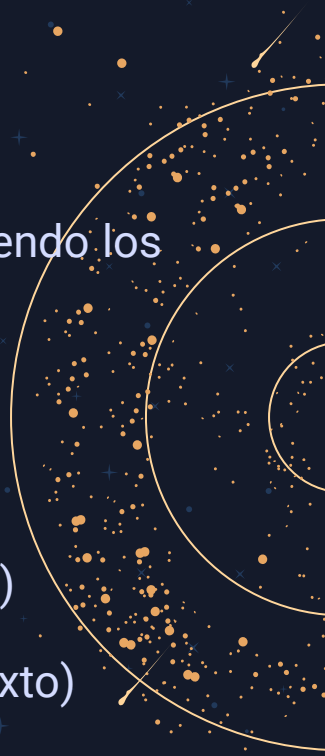


Caso de uso: Renderizar un cuerpo (body) distinto dependiendo de la versión del navegador o la región desde donde se accede al contenido

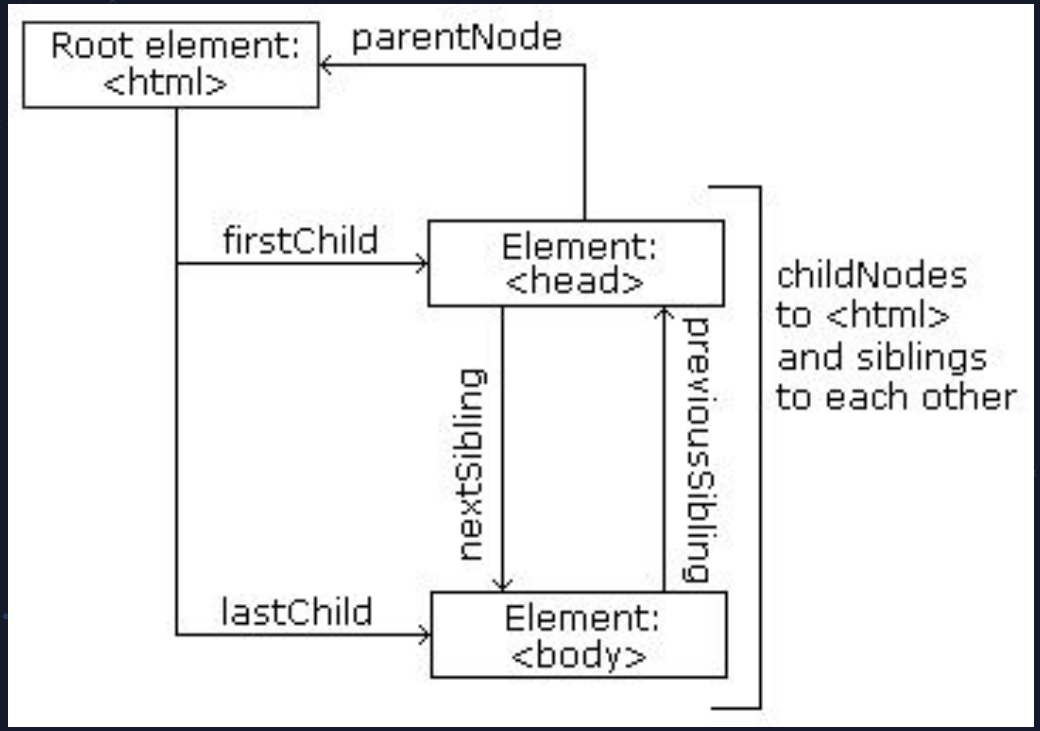
Navegación

Usando *nodos*

- **`childNodes`**
  - Regresa una *colección* de nodos hijos de un elemento, incluyendo los de tipo texto
- **`firstChild`**
  - Regresa el primer hijo de un nodo
- **`lastChild`**
  - Regresa el último hijo de un nodo
- **`previousSibling`**
  - Regresa el nodo adyacente previo (considera también el texto)
- **`nextSibling`**
  - Regresa el nodo adyacente siguiente (considera también el texto)
- **`parentNode`**
  - Regresa el padre del nodo



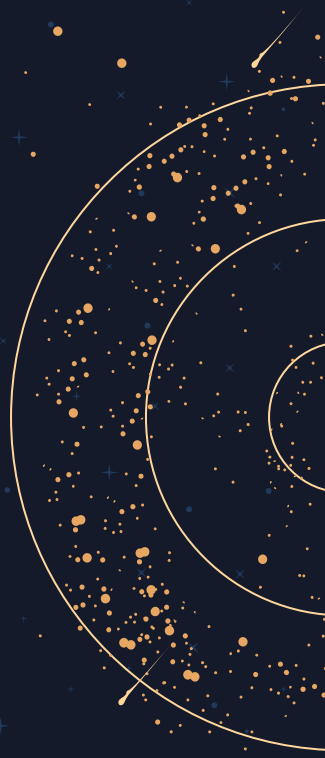
Navegación


Root element:
<html>
parentNode
firstChild
Element:
<head>
childNodes
to <html>
and siblings
to each other
nextSibling
previousSibling
lastChild
Element:
<body>

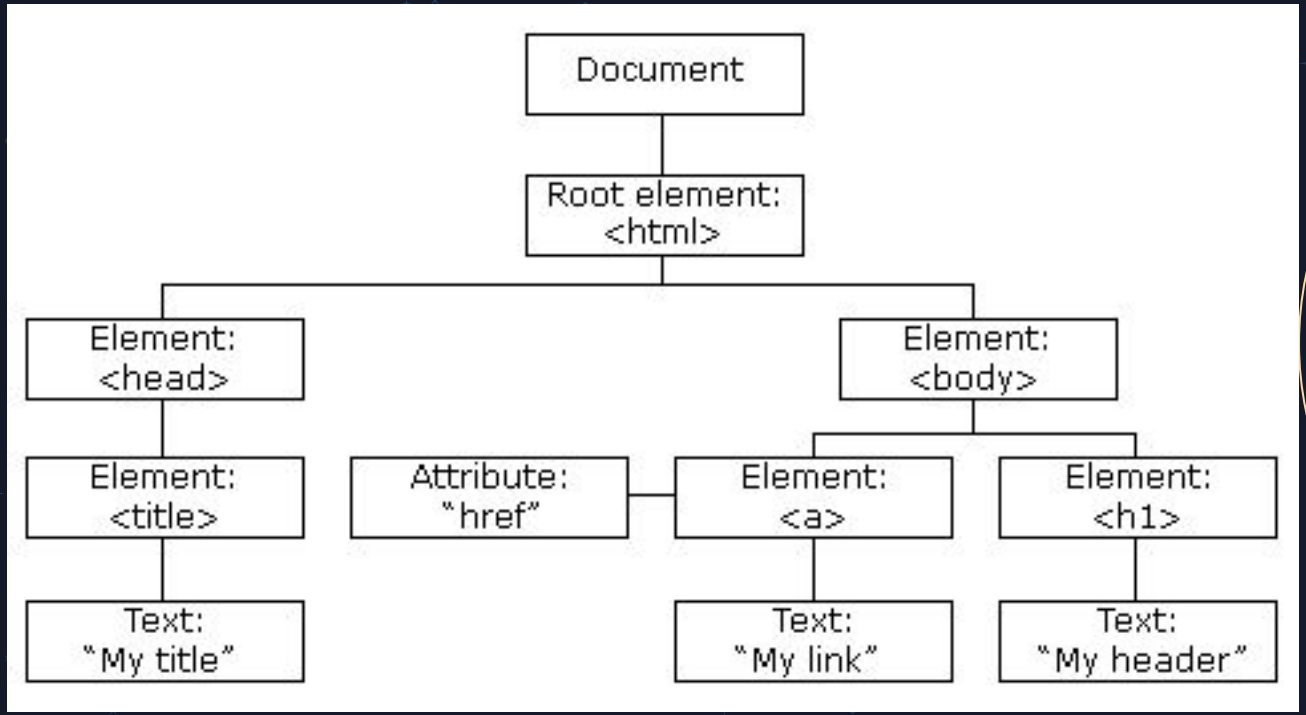
Navegación

Usando *elementos* (solo HTML)

- **`parentElement`**
  - Regresa el elemento contenedor del actual
- **`previousElementSibling`**
  - Regresa el elemento adyacente previo
- **`nextElementSibling`**
  - Regresa el elemento adyacente siguiente
- **`firstElementChild`**
  - Regresa el primer hijo del elemento
- **`lastElementChild`**
  - Regresa el último hijo del elemento



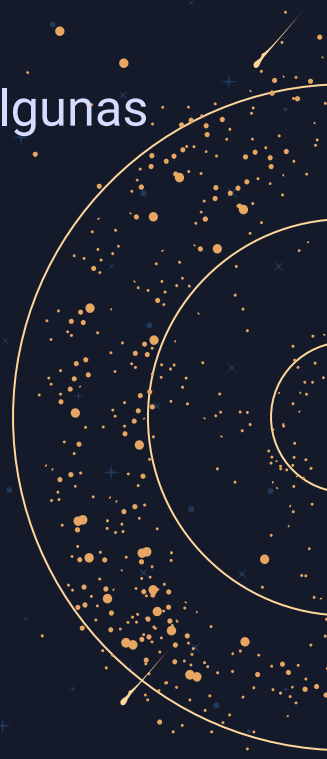
Navegación


Document
Root element:
<html>
Element:
<head>
Element:
<body>
Element:
<title>
Attribute:
"href"
Element:
<a>
Element:
<h1>
Text:
"My title"
Text:
"My link"
Text:
"My header"

Navegación de tablas

Adicional a lo revisado, las tablas (**<table>**) tienen algunas propiedades adicionales:

- **`elementoDeTabla.rows`**
  - *Colección* con los elementos **tr**
- **`elementoDeTabla.caption/tHead/tFoot`**
  - Referencias a los elementos **caption, thead** y **tfoot**
- **`elementoDeTabla.tBodies`**
  - *Colección* de elementos **tbody**



Navegación de tablas

Tabla de HTML

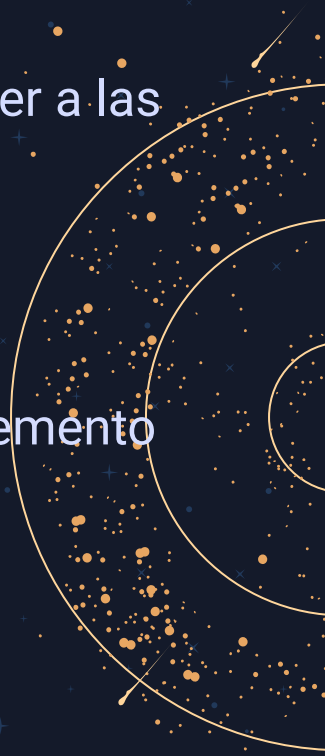
Encabezado 1	Encabezado 2
Col 1,1	Colo 1,2
Col 2,1	Col 2,2
Pie de tabla	

Navegación de tablas

- **thead**, **tfoot** y **tbody** incluyen la propiedad ***.rows*** para acceder a las filas de cada uno

- **tr**
  - ***.cells*** - *Colección* de **td** y **th**, dentro de **tr**
  - ***.sectionRowIndex*** - Índice del **tr** dentro de su elemento **thead**, **tbody** o **tfoot**
  - ***.rowIndex*** - Índice del **tr** en la tabla entera

- **td**
  - ***.cellIndex*** - Índice de una celda dentro de su **tr**



03

BÚSQUEDA DE ELEMENTOS

Métodos de búsqueda

- **`document.getElementById(id)`**
  - Obtiene un elemento por su ID (el cual debería ser único)
  - ***También se puede usar el id directamente como si fuera una variable***
  - Solo se puede usar con ***document***
- **`elem.getElementsByTagName(tag)`**
  - *Colección* de elementos por tipo (etiqueta de HTML)
- **`elem.getElementsByClassName(class)`**
  - *Colección* de elementos por clase (la cual puede ser reusada)
- **`elem.getElementsByName(name)`**
  - *Colección* de elementos por nombre (usualmente inputs)

- **`elem.querySelectorAll(css)`**
  - *Elementos dados por un selector de CSS (sumamente útil)*
- **`elem.querySelector(css)`**
  - Regresa el primer elemento de la selección

04

COLECCIONES

Concepto

Dentro de **JS** existen múltiples formas de almacenar conjuntos de datos, las más conocidas siendo arreglos y objetos, sin embargo, también existen las ***colecciones***.

Estos elementos no pueden invocar los métodos de arreglos como *filter, find, findIndex,* etc.

Aún con lo anterior, las *colecciones sí son iterables*, por lo cual pueden ser recorridas usando ***for-each*** de bloque y ***of***.

Es posible crear listas a partir de estas usando **Array.from(colection)** para posteriormente usar los métodos descritos anteriormente

Colecciones del DOM

Cuando se obtiene una colección del **DOM** mediante los métodos de búsqueda previamente revisados, no es posible modificarla, es decir, *solo son de lectura*

Las colecciones del **DOM** están “vivas”, es decir, se actualizan automáticamente cuando un cambio ocurre en la estructura general.

NodeList VS HTMLCollection

- **NodeList:** Más detallada, incluye todos los nodos internos (texto y comentarios
- **HTMLCollection:** Incluye solo elementos de HTML

Ejercicio 1

Crea un archivo **html** y realiza lo siguiente:

- Añade 3 elementos arbitrarios como h1, p, hr, ...
- Al final, añade otro elemento de tipo párrafo con un id “p1”
- Agrega texto a cada etiqueta creada (puede ser *lorem*)

Desde un archivo **JS** realiza lo siguiente:

- Dentro del último elemento agregado, modifica su texto para poner el contenido de todos los demás nodos (usa childNodes, innerText y textContent)
- Después, dentro del mismo elemento, pon solo el texto de los elementos (usa children e innerText

Opcional: Cambia el color de todos los h1 a amarillo (usa tagName y style.color)

Ejercicio 2

Utiliza los archivos base dentro de *ej2_base*. Usando **Bootstrap**, crea la estructura básica de un contenedor que muestre una imagen a la derecha, texto en el centro, y botones a la izquierda.

Realiza el diseño a tu gusto y después corta y pégalo como cadena interpolada (comillas simples invertidas) en el archivo de **JS** correspondiente. Se usará como plantilla.

Desarrolla una función **userToHTML** que recibe un *objeto* de tipo Usuario. La función regresará una cadena de HTML con los datos del usuario (aquí dentro va la plantilla).

Ejercicio 2

Desarrolla otra función **userListToHTML** que reciba un arreglo de usuarios y los muestre en el div principal de ***ej2.html*** (utiliza **innerHTML**).

Comprueba que **userListToHTML** y **userToHTML** funcionan mandando a llamar un arreglo generado con los datos encontrados en ***users.js*** o genera al menos 3 con los datos que quieras.

Utiliza el DOM para agregar la clase **“rounded-circle”** a cada imagen de usuario.

Ejercicio 2

Juan Perez

Correo: juan.perez@correo.mx

Fecha de nacimiento: 1980-10-10

Sexo: H

Diego Lopez

Correo: diego.lopez@correo.mx

Fecha de nacimiento: 1993-02-06

Sexo: H

Diana Gomez

Correo: diana.gomez@correo.mx

Fecha de nacimiento: 1991-12-08

Sexo: M