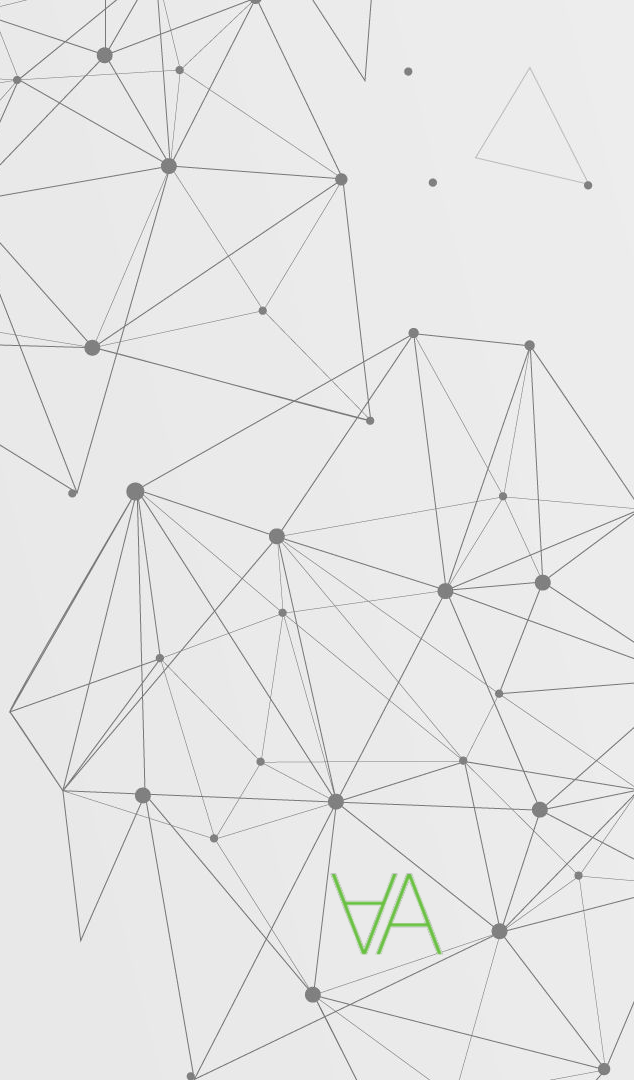
ALICIA VÁZQUEZ

JavaScript: Document Object Model

Alicia Vázquez
@aliciaFPInf

01

Incluir JS en HMTL



Incluir JavaScript en nuestra web

HTML nos ofrece la etiqueta **<script></script>** que será el que nos abra un espacio en el que incluir nuestro código JS.

De la misma manera que hemos hecho con CSS, podemos escribir directamente el código JavaScript en nuestro archivo HTML o bien hacerlo en un fichero ***.js** que luego importamos a nuestra página.

Además podemos poner el script dónde queramos del código y cuantas veces queramos.

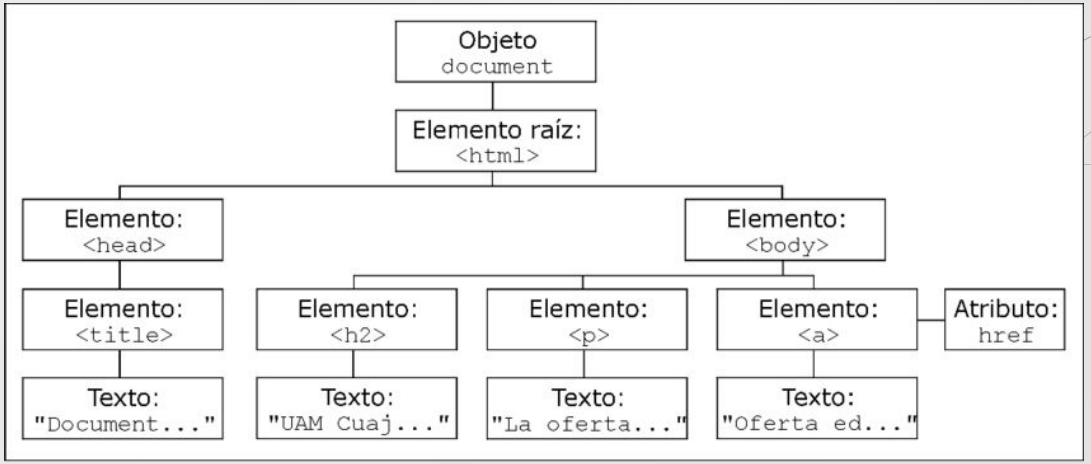
¿Cuál crees que es la mejor opción? ¿Por qué?

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <script src="scripts/01-variables.js" defer></script>

    <link rel="stylesheet" text="text/css"
href="css/style.css">
    <title>Aprendiendo JavaScript Moderno</title>
</head>
<body>
    <h1>Aprendiendo JavaScript Moderno</h1>
    <div id="contenedor"></div>
</body>
</html>
```

Document Object Model

La creación del **Document Object Model** o **DOM** es una de las innovaciones que más ha influido en el desarrollo de las páginas web dinámicas y de las aplicaciones web más complejas.

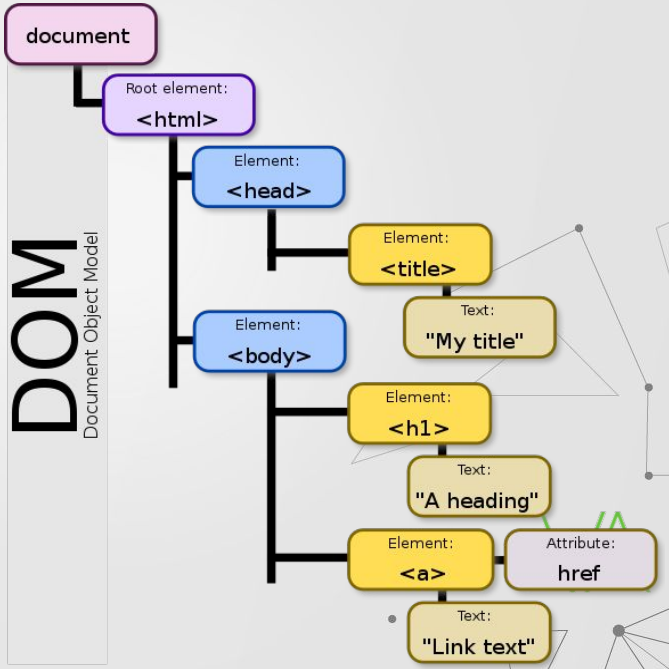
DOM permite a los programadores web acceder y manipular las páginas **HTML** como si fueran documentos XML.
El **DOM** es una representación en árbol del documento HTML. Gracias a los lenguajes de **JavaScript**, PHP, etc. podemos acceder al DOM para modificarlo desde la programación nuestra página HTML.



Document Object Model

La Interfaz **document** representa la página HTML cargada en el navegador. Es a través de esta interfaz que accederemos a todos los elementos del DOM siguiendo la jerarquía de estos.

```
<!DOCTYPE html>
<html>
  <head>
    <title>My
title</title>
  </head>
  <body>
    <h1>A heading</h1>
    <a href="">Link
text</a>
  </body>
</html>
```



02

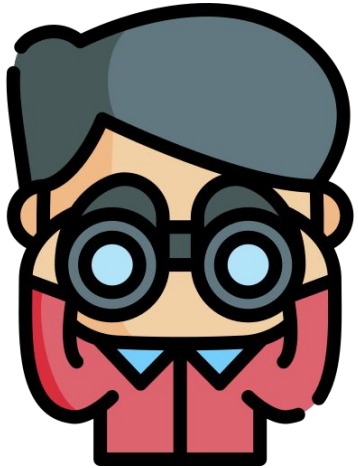
ElementHTML

Document Object Model-Element

Para acceder a los diferentes elementos del DOM tenemos varios **métodos** que varían según el dato que usemos para identificarlos. Empezamos por el más sencillo.

ElementHTML `document.getElementById(id)`	Busca el elemento HTML con el **id**. Si lo encuentra devuelve el **elementoHTML**, si no devuelve **null**.
ElementHTML `document.querySelector(selectorCSS)`	Busca el elemento HTML que cumpla ese selector. Si lo encuentra devuelve el **elementoHTML o el primero si hay varios**, si no devuelve **null**.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejemplo 1 - Hello World JS</title>
</head>
<body>
    <h1 id="titulo">Aprendiendo JavaScript</h1>
    <p>Hola Mundo!</p>
    <p id="info">Esta página es <strong>muy simple</strong></p>
    <span class="importante">Esto es muy importante</span>
    <script>
        let titulo = document.getElementById("titulo");
        console.log("Elemento con Id=titulo: "+ titulo );
    </script>
 </body>
</html>
```



Element: Propiedades

Los **elementos** del **document** tienen las siguientes propiedades principales.

Por ejemplo: *** Esta página es muy simple ***

`element.nodeName`	Devuelve el nombre del nodo (etiqueta si es un elemento HTML). Sólo lectura. Por ejemplo: *P*
`element.textContent` (mantiene los espacios y tabulaciones)	Igual que lo anterior, pero sin tener en cuenta las etiquetas que puedan haber dentro. También podemos asignar Por ejemplo: *Esta página es muy simple*
`element.innerHTML`	Accedemos al texto HTML del elemento. Todo lo que hay entre la etiqueta que abre elemento y la que lo cierra, incluyendo otras etiquetas HTML. Por ejemplo: *Esta página es muy simple*

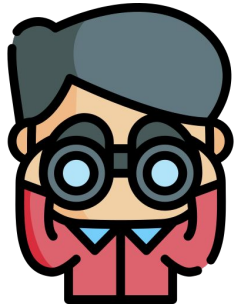
Element: Propiedades

Los **elementos** del **document** tienen las siguientes propiedades principales.

*Por ejemplo: Esta página es muy simple *

`element.outerHTML`	Igual que *.innerHTML* pero incluyendo el HTML del propio elemento HTML. Por ejemplo: *Esta página es muy simple *
`element.value`	Devuelve la propiedad 'value' de un <input>. Como los <inputs> no tienen etiqueta de cierre (</input>) no podemos usar .innerHTML ni .textContent.

```
<body>
 <!-- Añadimos este código -->
    <p id="info">Esta página es <strong>muy simple</strong></p>
    <script>
        let elemento = document.getElementById("info");
        console.log("nodeName: " + elemento.nodeName);
        console.log("textContent: " + elemento.textContent);
        console.log("innerHTML: " + elemento.innerHTML);
        console.log("outerHTML: " + elemento.outerHTML);
    </script>
 </body>
```



Aprendiendo JavaScript

Hola Mundo!

Adiós mundo *cruel*

Esto es muy importante

Esta página es **muy simple**

DevTools is now available in Spanish! Always match Chrome's language Switch DevTools to Spanish

Elements Console Sources Network Performance Memory Application

top Filter

```
Elemento con Id=titulo: [object HTMLHeadingElement]
nodeName: P
textContent: Esta página es muy simple
innerHTML: Esta página es <strong>muy simple</strong>
outerHTML: <p id="elemento">Esta página es <strong>muy simple</strong></p>
```

Element: Propiedades. Modificar

De la misma manera que puedo ver las propiedades, las puedo modificar y la actualización es automática!!

Al ser una propiedad de un objeto no tengo más que igualarla al nuevo valor.

```
elemento.outerHTML="<h3 id='info'>AHORA SOY UN <strong>TITULO</strong></h3>" ;
elemento.innerHTML="AHORA SOY UN <em>TITULO</em>" ;
elemento.textContent="AHORA SOY UN <em>TITULO</em>" ;
elemento.textContent="Solo texto!!!";
```

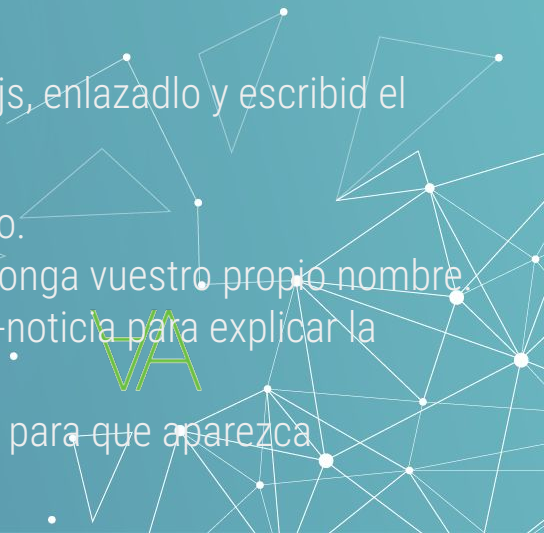
Ejercicio 1: Hacker.

Fase 1: Reconocimiento (Usando la Consola del Navegador) Antes de escribir el fichero .js, probad esto en la consola de Chrome/Firefox (F12):

a. **Investiga al autor**: Selecciona el nodo con el id autor e imprime por consola quién escribió la noticia original.
b. **Captura el titular**: Guarda el elemento h1 en una variable llamada titular y muestra por consola su texto actual.
c. **Espía el input**: Selecciona el input de comentarios y muestra por consola qué texto tiene escrito por defecto.

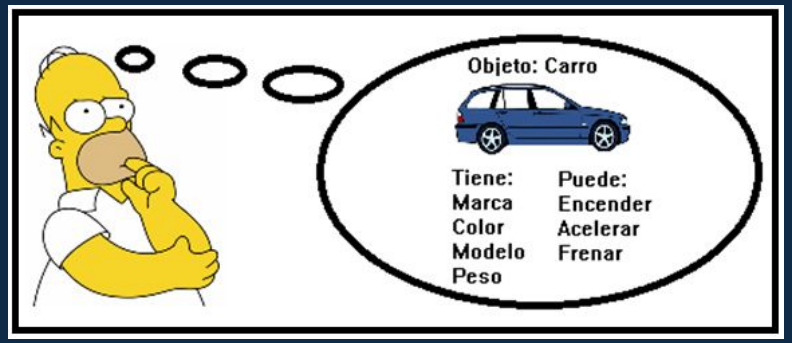
Fase 2: El Ataque (Creando el fichero hacker.js) Ahora cread el fichero hacker.js, enlazadlo y escribid el código para alterar la realidad:

d. **Cambio de Titular:** Selecciona el h1 y cámbialo por algo dramático.
e. **Suplantación de Identidad:** Cambia el texto del #autor para que ponga vuestro propio nombre.
f. **Manipulación del contenido**: Cambia el texto del párrafo #cuerpo-noticia para explicar la nueva noticia loca.
g. **Troleo del Input:** Cambia el valor (value) del input de comentarios para que aparezca pre-escrito: "¡No me lo puedo creer, es increíble!".



NOTA

Programación Orientada a Objetos.



Carro

Atributos
- Color
- Marca
- Modelo

Métodos
- arrancar()
- acelerar()
- detenerse()

Persona

Atributos
- Nombre
- Edad
- Trabajo

Métodos
- caminar()
- comer()
- trabajar()

03

Atributos de los Elementos

Atributos del ElementHTML

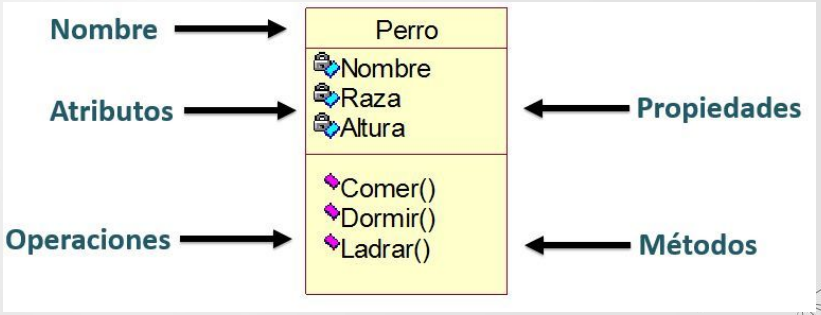
Una vez seleccionado el elemento podemos acceder a sus atributos, añadir más, cambiar su valor y modificar sus estilos.

`element.attributes`	Obtenemos un array con todos los atributos.
`element.hasAttribute("nameAttribute")`	Indica si el atributo está o no definido.
`element.getAttribute("nameAttribute")`	Devuelve el valor del atributo.
`element.setAttribute("nameAttribute ", value)`	Usando el nombre del atributo del elemento podemos acceder a él y modificarlo .
`element.removeAttribute("nameAttribute")`	Elimina el atributo.

Atributos

Para atributos del tipo *id*, *class (className)*, o *title* existen atajos y podemos acceder a ellos directamente como de una propiedad se tratara:

```
parrafo.id="identificadorNuevo"
```



Estilo

Los estilos están accesibles a través del atributo **style**. Cualquier estilo es una propiedad de dicho atributo pero con la sintaxis *camelCase* en vez de *kebab-case*.

Propiedades/Metodos	Descripción
`element.style.property`	Podemos modificar el estilo de cualquier propiedad CSS del elemento.

Por ejemplo para cambiar el color de fondo (propiedad **background-color**) y ponerle el color rojo al elemento miPrimeraLista haremos:

```
miPrimeraLista.style.backgroundColor = 'red';
```

Nota: Es mejor no modificar los estilos directamente a los elementos sino asignarles o quitarles clases que tendremos definidos en el CSS(ej. Bootstrap)

```
<body>
    <h1 id="titulo" atributoInventado="valorAtributo">Aprendiendo JavaScript Moderno</h1>
    <p>Hola Mundo!</p>
    <p class="importante">Adiós mundo <em>cruel</em></p>
    <span class="importante">Esto es muy importante</span>
    <script>
        let titulo = document.getElementById("titulo");
        console.log("El titulo de la página es: " + titulo.innerHTML);
//Aprendiendo JavaScript Moderno
        titulo.innerHTML = "Le cambio el titulo";
        titulo.className="upper";
        titulo.setAttribute("name","pepe");
    </script>
</body>
```

Ejercicios: Examen

Ejercicio 1: Cambiar el Contenido (1 punto)

1. Crea un archivo HTML con un <h1> que diga "Título original" y un párrafo con el texto "Texto original".
2. Escribe un archivo JavaScript para:
    - Cambiar el texto del <h1> a "Título actualizado".
    - Cambiar el texto del párrafo a "Texto actualizado".

Ejercicio 2: Modificar Atributos (2 puntos)

1. Crea un archivo HTML con una imagen con el atributo `src` apuntando a una foto y un párrafo con un atributo `id="descripcion"`.
2. Usa JavaScript para:
    - Cambiar el atributo `src` de la imagen para que apunte a una foto diferente.
    - Cambiar el contenido del párrafo a "Imagen actualizada".

Ejercicio 3: Modificar Estilos (2 puntos)

1. Crea un archivo HTML con un `<h1 id="titulo">Hola, mundo</h1>`.
2. Usa JavaScript para cambiar:
    - El color del texto a rojo.
    - El tamaño de la fuente a 32px.

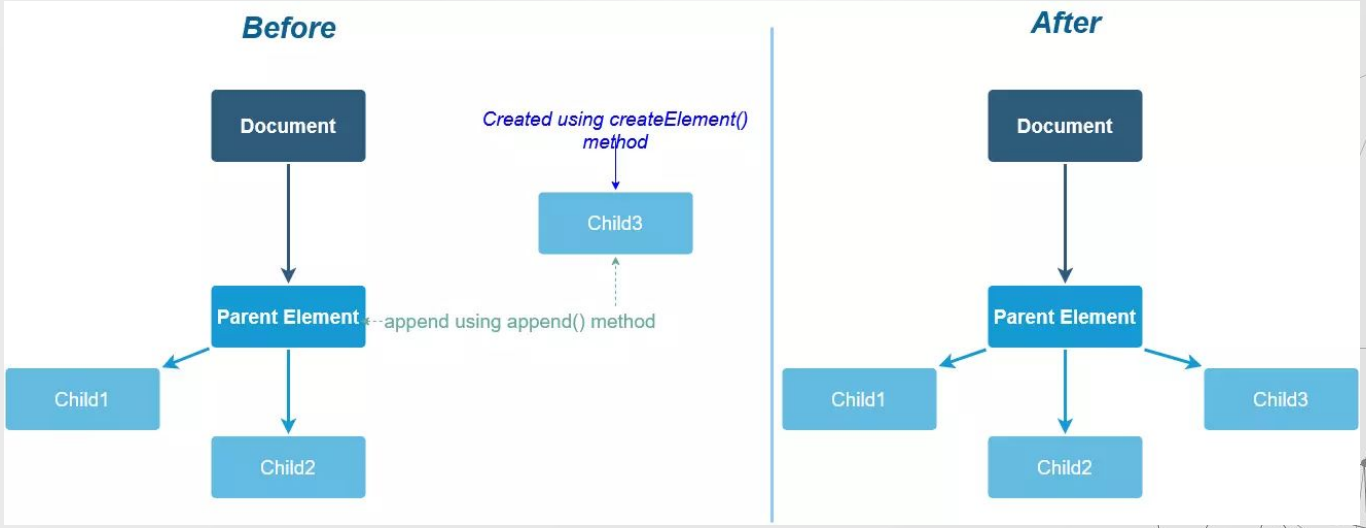
04

Manipular el DOM

Manipular el DOM

La creación de nuevos elementos (nodosHTML) se hace en dos acciones.

1. Se crea el nodo y se asignan los atributos, valores, etc.
2. Se añade ese nuevo nodo como hijo del nodo seleccionado.



Crear el NODO

Es posible crear nuevos elementos de HTML desde JS y modificar así el **DOM**

Sea un **node**, cualquier nodo del documento (texto, comentario, elemento, script). Del **node**, podemos saber su nombre, su tipo y su valor. (nodeName, nodeType y nodeValue)

Existen métodos que nos permiten ir creando nuevos nodos.

ElementHTML document.createElement(tagName, *options*)	Crea y devuelve el tipo de elemento HTML especificado en **tagName**. NO ESTÁ INSERTADO EN EL HTML.
node document.createTextNode(text)	Crea y devuelve el tipo node de tipo texto, luego tendremos que añadirlo a un nodo HTML. NO ESTÁ INSERTADO EN NINGÚN ELEMENTO.
node document.createTextComment(text)	Crea y devuelve el tipo **node** de tipo comentario. NO ESTÁ INSERTADO EN EL HTML.

Añadir el NODO (al árbol)

Una vez tenemos creado un nodo deberemos incluirlo en el documento y especificar en qué rama lo vamos a “colgar”

`document.appendChild(` **`element`**`)` `element.appendChild(` **`node`**`)`	Permite añadir al elemento otro **elemento/node**, modificamos la estructura del árbol, la jerarquia.
`element.remove()`	Elimina el propio elemento.
`document.removeChild(` **`element`**`)` `element.removeChild(` **`node`**`)`	Elimina en node especificado, sea cual sea.

Opción 1: Create y append

```
<script>
    //Creamos y añadimos nuevos elementos usando CREATE y APPEND
    let parrafoPrimero = document.createElement('p');
    let parrafoPrimeroTexto = document.createTextNode('Párrafo que creamos al principio pero que añadimos al final');
    parrafoPrimero.appendChild(parrafoPrimeroTexto);

    let destacado = document.createElement('em');
    destacadoTexto = document.createTextNode('texto que destacamos');
    destacado.appendChild(destacadoTexto);

    let parrafoSegundo = document.createElement('p');
    let parrafoSegundoTexto1 = document.createTextNode('Este es el ');
    let parrafoSegundoTexto2 = document.createTextNode(' del segundo párrafo. Tenemos que ir añadiendo en el orden en que queremos que salga.');

    //Añadimos los nosdos texto y em que hemos en el orden.
    parrafoSegundo.appendChild(parrafoSegundoTexto1);
    parrafoSegundo.appendChild(destacado);
    parrafoSegundo.appendChild(parrafoSegundoTexto2);

    //Tenemos creados todos los parrafos y ahora hay que añadirlos a un elemetos que está en el DOM.
    let miDiv = document.getElementById('articulos');
    miDiv.style.backgroundColor='green';

    //Insertamos el P que hemos definido en segunda posición, ANTES que el que hay ya dentro de la DIV.
    let parrafoDOM = miDiv.children[0];
    miDiv.insertBefore(parrafoSegundo, parrafoDOM);

    //O bien lo pondriamos a continuación
    // miDiv.appendChild(parrafoSegundo);
    //Lo añadimos al final, se colocará automáticamente en la cola.
    miDiv.appendChild(parrafoPrimero);
</script>
```

Opción 2: Combinar create/append e innerHTML

```
<div id="articulosInner">
      <p>Parrafo que está en el HTML</p>
  </div>

  <script>

      let miDivInner = document.getElementById('articulosInner');
      miDivInner.style.backgroundColor = 'yellow';

      miDivInner.innerHTML += '<p>Párrafo añadido al final, por defecto.</p>';

      //Creamos un parrafo y le especificamos que texto HTML tiene (incluye tags)
      let parrafoSegundoInner = document.createElement('p');
      parrafoSegundoInner.innerHTML = 'Este es el texto que <em>destacamos del segundo párrafo</em>.
Tenemos que ir añadiendo en el orden en que queremos que salga.';

      //Añadimos este nuevo parrafo antes que el primero que hay.
      let parrafoDOMInner = miDivInner.children[0];
      miDivInner.insertBefore(parrafoSegundoInner, parrafoDOMInner);
  </script>
```

Ejercicio 1: Hacker.

Fase 3: Añade una nueva noticia.

Crea una nueva noticia siguiendo la misma estructura que la anterior. Inventa lo que tu creas conveniente.

¿Te atreves a poner una imagen?

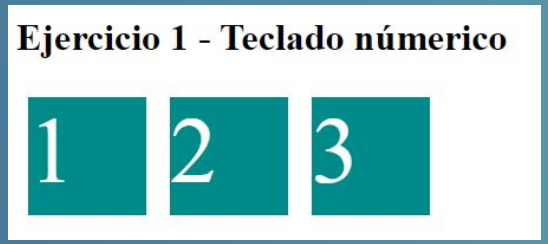
Ejercicios: Examen

Ejercicio 4: Crear y Añadir Elementos (2 puntos)

1. Crea un archivo HTML con un `<div id="contenedor"></div>`.
2. Usa JavaScript para:
    - Crear un nuevo elemento `<p>` con el texto "Este es un párrafo añadido".
    - Añadir este párrafo como hijo del `<div>` con id `contenedor`.

Ejercicio 2: Teclado numérico

Partiendo del código facilitado y únicamente usando código js (create, append y modificando los atributos de los elementos), crea la siguiente pantalla.

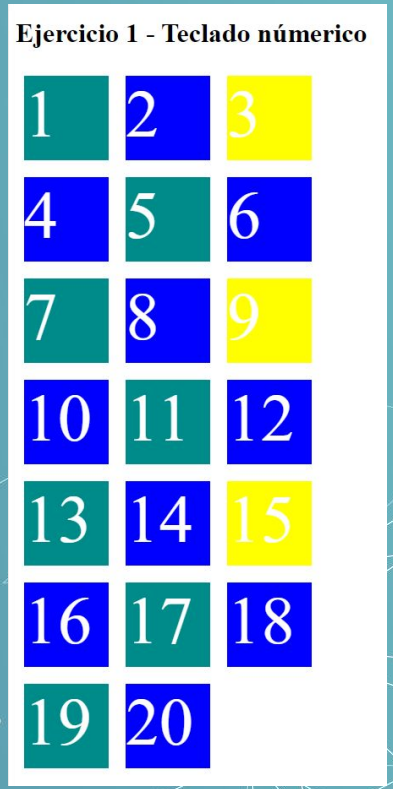


Mejoras:

1. Crea un clase para darle estilo a la tecla del número.
2. Trata de hacer un teclado numérico…
3. Mejora el código, ¿se te ocurre cómo?

Ejercicio 2: Teclado numérico

- Añade las estructuras de control necesarias para:
  a. Cambiar fondo teclas de los números pares,
  b. Cambiar fondo teclas de los números múltiplos de 3.
  c. El usuario no pueda poner más de 30.
  d. Añade algo más que tu creas para mejorar el teclado, cualquier cosa que se te ocurra.



Ejercicio 3: Teclado alfabético

- Partiendo del teclado numérico; modifica o añade nuevas funciones que te permitan crear un teclado alfabético.
- Haz que las vocales tengan un color diferente.

Ayuda:

- Código ASCII - [65,90].
- Existe el método `let letra=String.fromCharCode(ascii_code)`

Ejercicios: Examen

Ejercicio 5: Tablero de ajedrez (3 puntos)

1. Crea un archivo HTML con un `<div id="tablero"></div>`.
2. Usa JavaScript para:
   - Generar un tablero de 8x8 casillas (64 celdas) dentro del `<div>`.
   - Cada celda será un cuadrado de 50x50 píxeles con un borde.
   - Usa un bucle para crear las casillas dinámicamente.
3. Aplica los siguientes estilos condicionales.
   - Las casillas son negras y blancas, empezando por negro.
4. Coloca las fichas de ajedrez donde toquen, poniendo una X si son blancas o una O si son ne