

¡Javascript!

guayverd beta hub

Agenda del día



01

Introducción

Repaso métodos
String, Number y
Object.

02

DOM

Respuesta servidor
DOM
Consola y DOM
Nodos y Elementos
Métodos de acceso.
querySelector
querySelectorAll
Otros métodos de acceso
innerHTML
innerText
createElement
remove()
append()

03

Ejercitación

Mediante un prompt que el
usuario vaya construyendo
una página con elementos.



daily

¿Cómo venimos?

¿Algo nos bloquea?

¿Cómo seguimos?



DOM

Document Model Object



Respuesta del servidor a una petición

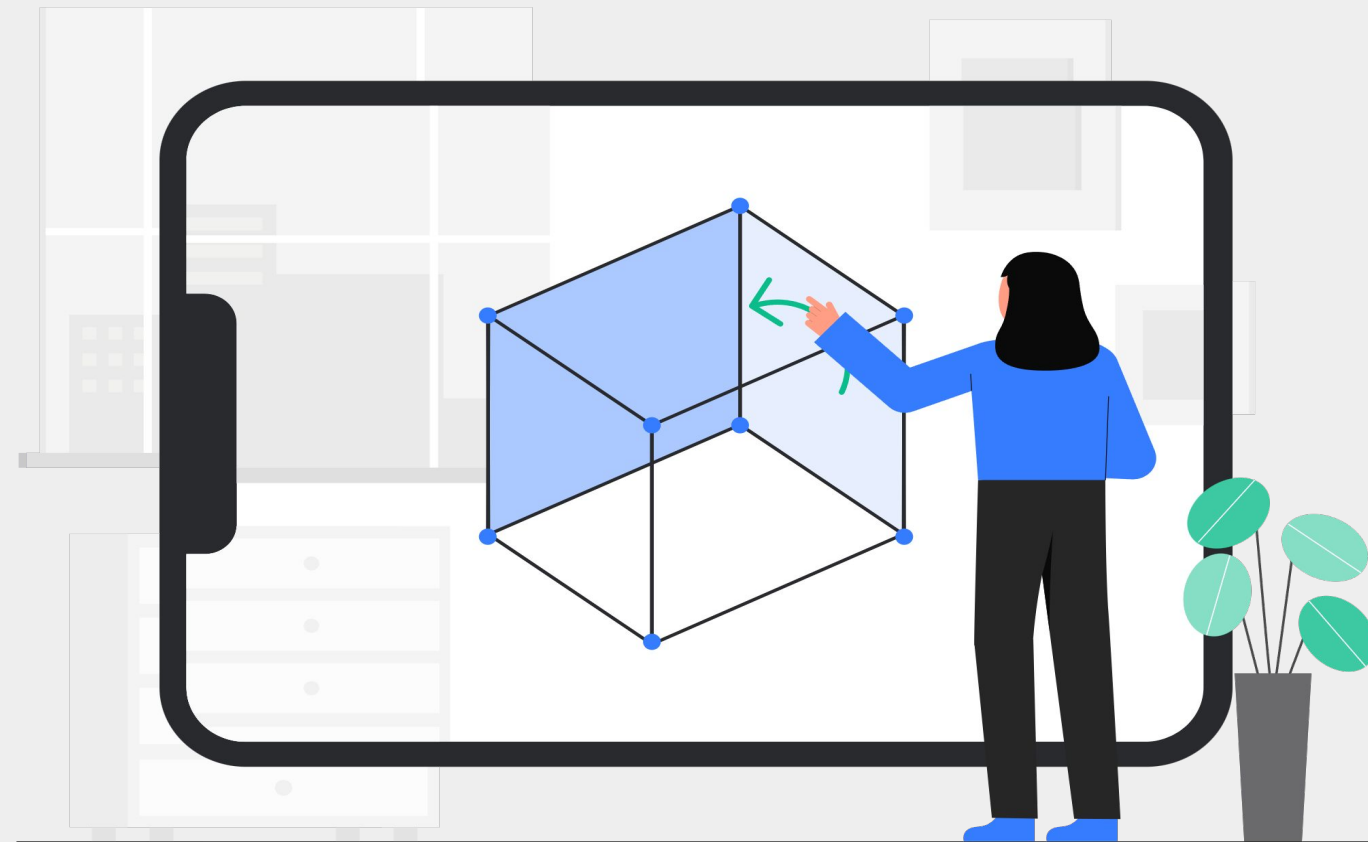
Cuando queremos entrar a una app web, hacemos una petición al servidor que nos devuelve un HTML. El navegador convierte esta respuesta en un objeto en forma de árbol, llamado DOM.

The image shows a web browser window with the URL `guayerd.com/es/`. The page content includes the Guayerd logo, a navigation menu with 'EN' and 'PT', and a main heading 'Inclusión social y laboral a través de Educación en Tecnología'. Below this is a 'Conoce más' button and a section titled 'Qué hacemos' with a bullet point. The text below the bullet point reads: 'Somos una Edtech de impacto que ofrece experiencias educativas en Tecnología a través de una metodología de triple soporte: formamos, acompañamos e insertamos laboralmente talento con potencial proveniente de contextos de vulnerabilidad socio...'. The browser's developer tools are open, showing the 'Red' (Network) tab. The 'Resposta' (Response) sub-tab is selected, displaying the HTML response for the 'es/' resource. The HTML code is as follows:

```
1 <!DOCTYPE html>
2 <html lang="es-ES">
3   <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale=1">
6     <link rel="profile" href="https://gmpg.org/xfn/11">
7     <title>Guayerd &#8211; Social and labor inclusion through Education
8     <meta name='robots' content='max-image-preview:large' />
9     <link rel="alternate" href="https://www.guayerd.com/" hreflang="en"
10    <link rel="alternate" href="https://www.guayerd.com/es/" hreflang="
```

SPRINT 2

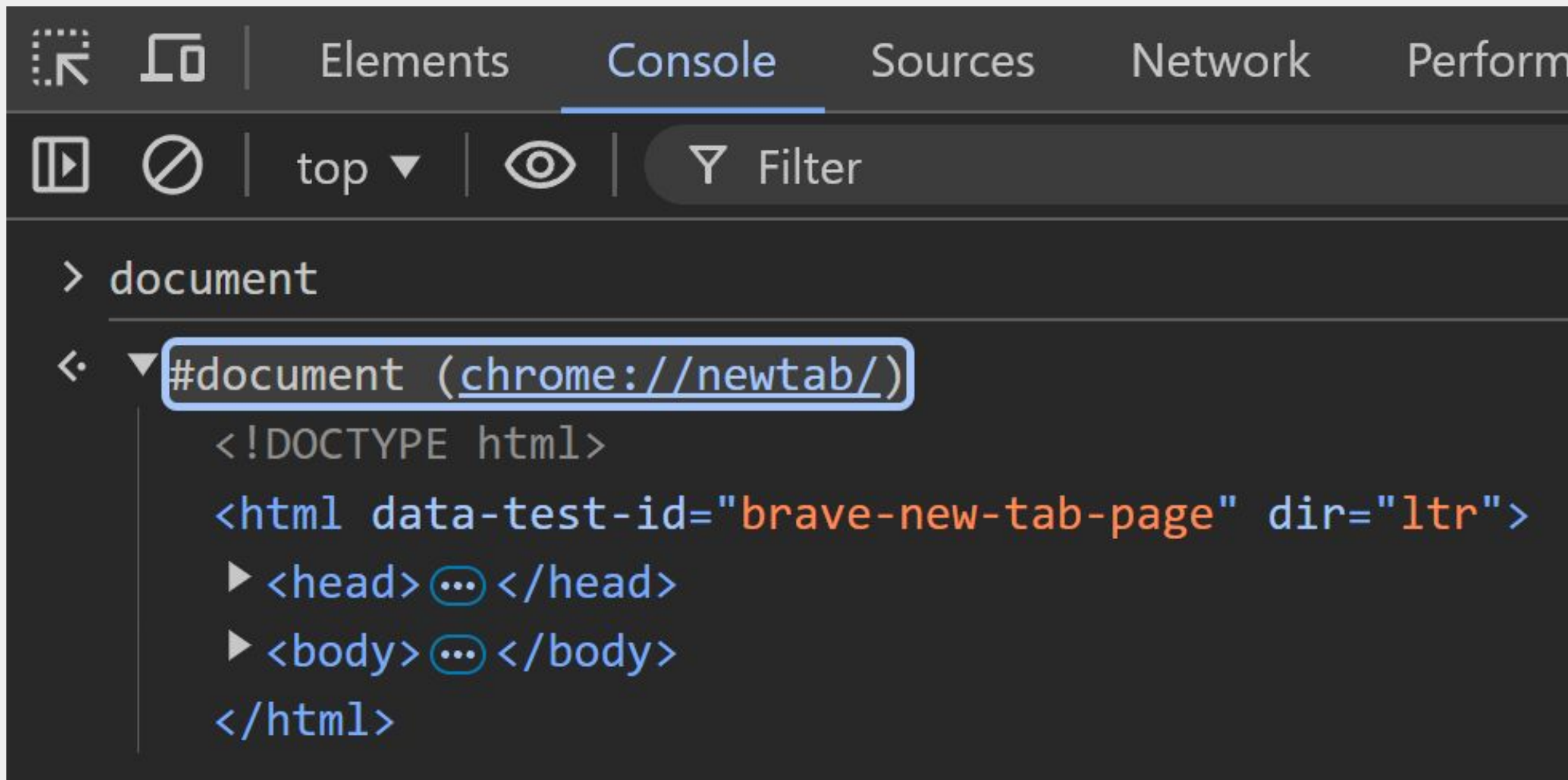
DOM



DOM es el acrónimo de **Document Object Model**. Es la representación en memoria de lo que vemos en la interfaz de usuario (UI).

Es un objeto anidado que puede ser accedido y modificado por **Javascript**. Por lo cual, al editarlo, cambia lo que vemos en la **UI**.

En Javascript DOM se escribe así



```
> document
< #document (chrome://newtab/)
  <!DOCTYPE html>
  <html data-test-id="brave-new-tab-page" dir="ltr">
    > <head> ... </head>
    > <body> ... </body>
  </html>
```


Acceder a las propiedades del DOM

Debemos usar console.dir.

```
> console.dir(document)
VM186:1
▼ #document ⓘ
  ▶ location: Location {ancestorOrigins: DOMStringList, href: 'chrome://newtab/', origin: 'chrome://newtab'
    _reactListening6709p8qjjmt: true
    URL: "chrome://newtab/"
  ▶ activeElement: body
  ▶ adoptedStyleSheets: Proxy(Array) {}
    alinkColor: ""
  ▶ all: HTMLAllCollection(290) [html, head, meta, meta, title, link, link, link, link, link, link, script,
  ▶ anchors: HTMLCollection []
  ▶ applets: HTMLCollection []
    baseURI: "chrome://newtab/"
    bgColor: ""
  ▶ body: body
    characterSet: "UTF-8"
    charset: "UTF-8"
    childElementCount: 1
  ▶ childNodes: NodeList(2) [<!DOCTYPE html>, html]
  ▶ children: HTMLCollection [html]
    compatMode: "CSS1Compat"
    contentType: "text/html"
    cookie: ""
    currentScript: null
  ▶ defaultView: Window {window: Window, self: Window, document: document, name: '', location: Location, ...}
    designMode: "off"
    dir: "ltr"
  ▶ doctype: <!DOCTYPE html>
```


Elementos

El DOM está conformado por una gran cantidad de **tipos de nodos**. Entre ellos hay unos determinantes para nosotros, los elementos. Los elementos son el equivalente de las etiquetas HTML. También hay jerarquías de elementos, HTML es padre de body y los divs son elementos descendientes.

```
> document.body
< <body>
  <div id="root" style="--ntp-extra-content-effect-multiplier: 0; --ntp-scroll-percent: 0; --ntp-fixed-content-height: 2775px;">
    <div class="App--suoy3r gvmVNR"> flex
      ::before
      <div class="StyledPage--h5bui fGBygr"> ... </div> grid
      <leo-dialog showclos class="SettingsDialog--1ffucv0 jDuJrT"> ... </leo-dialog>
    </div>
  </div>
</body>
```

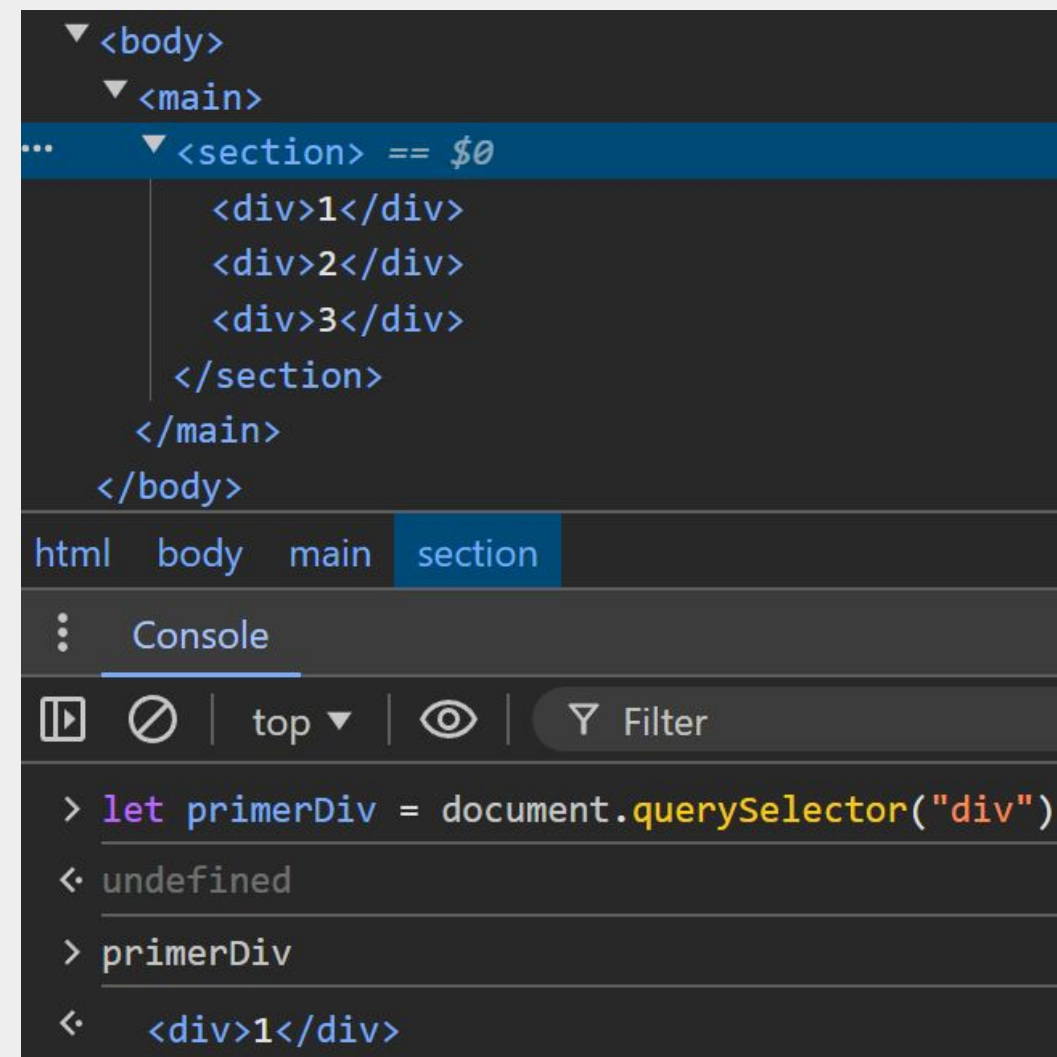
Métodos de acceso

DOM



querySelector

La sintaxis del argumento es como la de los selectores CSS.
Devuelve el primer elemento que coincide con el argumento pasado al método.



```
<body>
  <main>
    <section> == $0
      <div>1</div>
      <div>2</div>
      <div>3</div>
    </section>
  </main>
</body>
```

html body main section

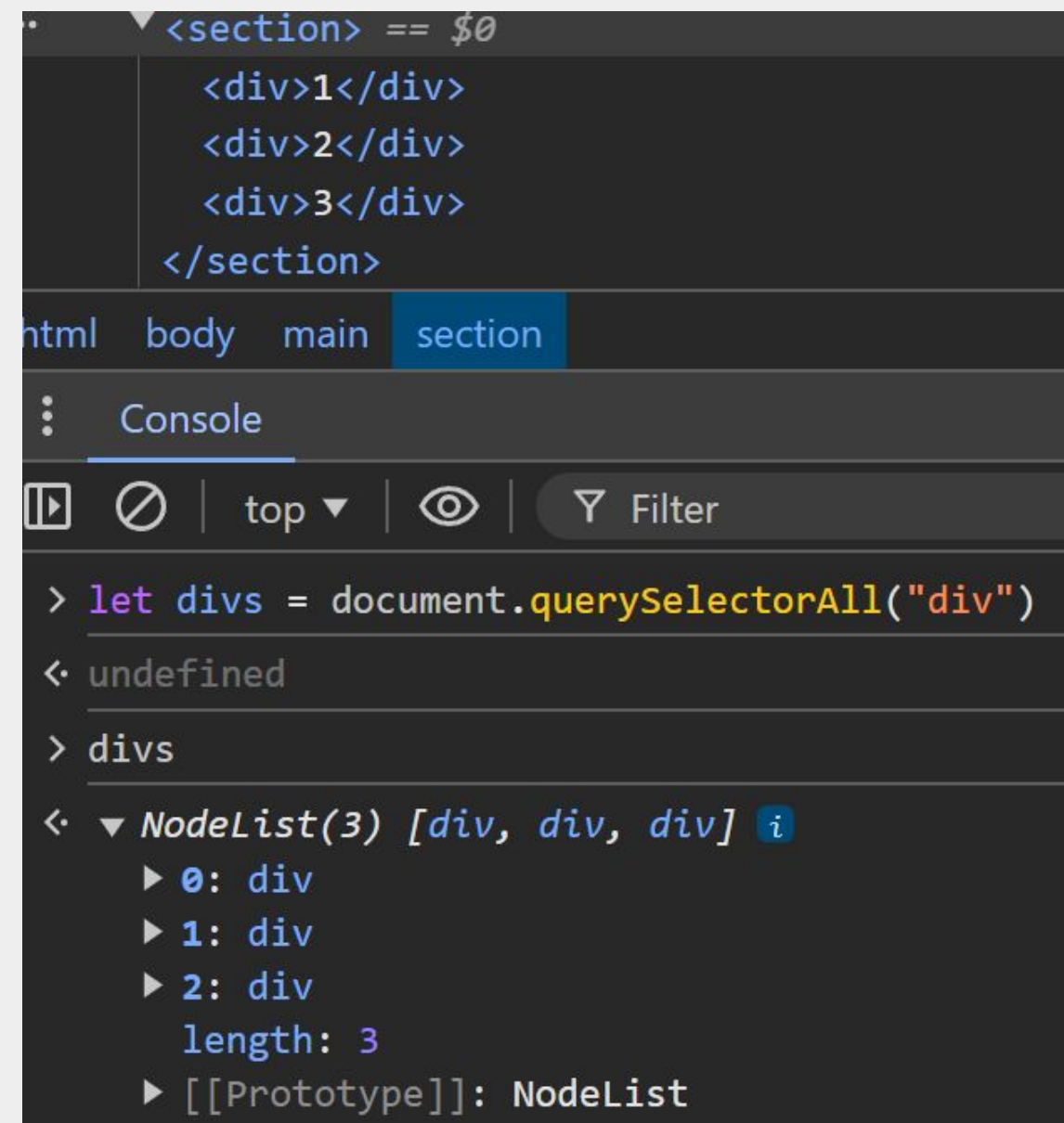
⋮ Console

⏏ | 🔍 | top ▼ | 👁 | 🔍 Filter

```
> let primerDiv = document.querySelector("div")
< undefined
> primerDiv
< <div>1</div>
```

querySelectorAll

Devuelve una estructura parecida a un array que contiene todas las coincidencias.

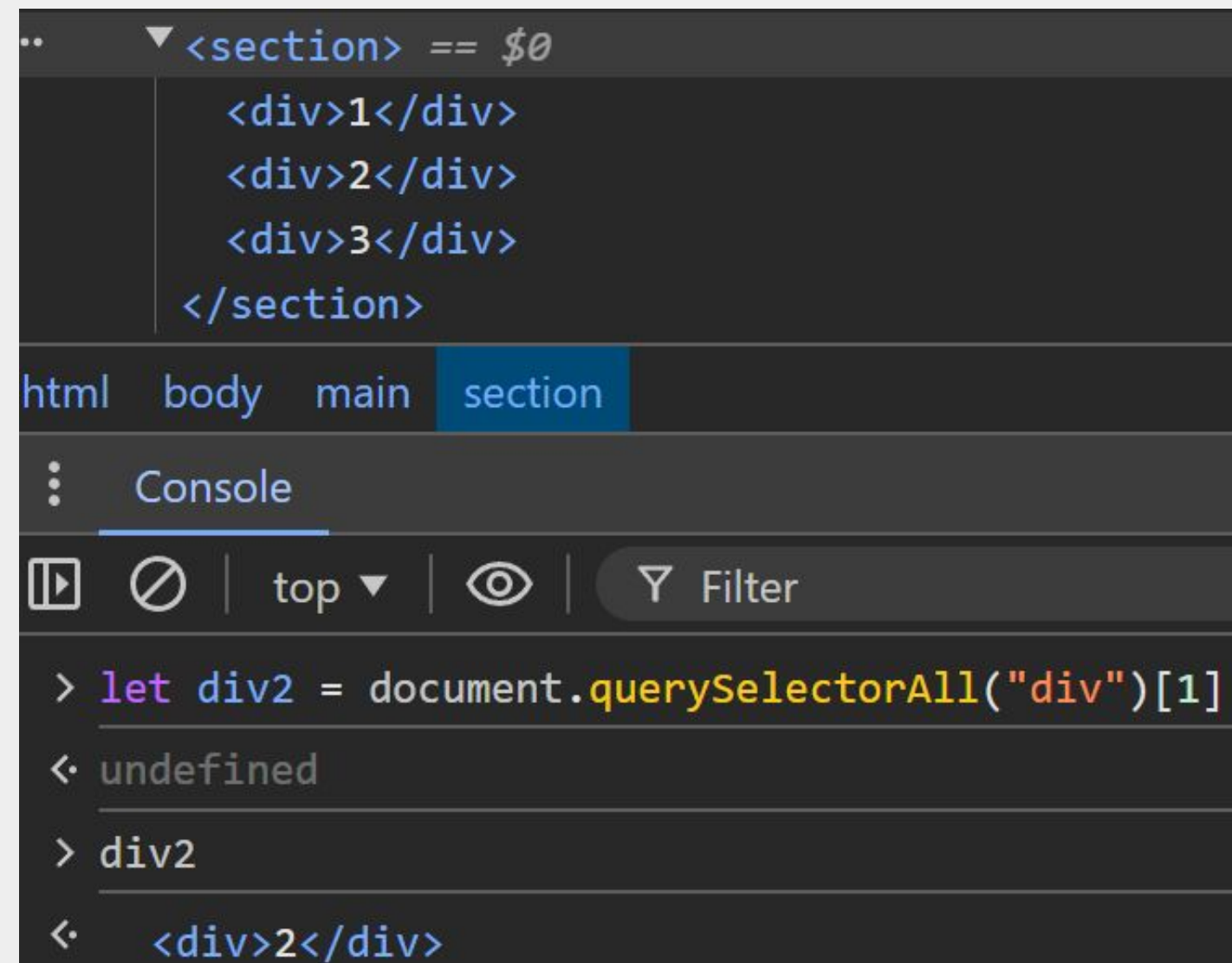


The screenshot shows a web browser's developer console. At the top, a snippet of HTML is displayed: `<section> == $0`, `<div>1</div>`, `<div>2</div>`, `<div>3</div>`, and `</section>`. Below this, the breadcrumb navigation shows the path: `html` > `body` > `main` > `section`. The console tab is active, showing the following commands and results:

```
> let divs = document.querySelectorAll("div")
< undefined
> divs
< ▼ NodeList(3) [div, div, div] ⓘ
  ▶ 0: div
  ▶ 1: div
  ▶ 2: div
  length: 3
  ▶ [[Prototype]]: NodeList
```


querySelectorAll por índice

Recuperar el segundo elemento de la lista.



```
..  ▾ <section> == $0
    <div>1</div>
    <div>2</div>
    <div>3</div>
    </section>

html  body  main  section
⋮
Console
[ ] [ ] | top ▾ | [ ] | [ ] Filter

> let div2 = document.querySelectorAll("div")[1]
< undefined
> div2
< <div>2</div>
```

Métodos antiguos

Pero que aún se usan.

<code>getElementById("id")</code>	Recupera un elemento con el id pasado.
<code>getElementsByClassName("clase")</code>	Recupera la lista de elementos que tengan la clase.
<code>getElementsByTagName("etiqueta")</code>	Recupera la lista de elementos que tengan la etiqueta.

Modificación de elementos

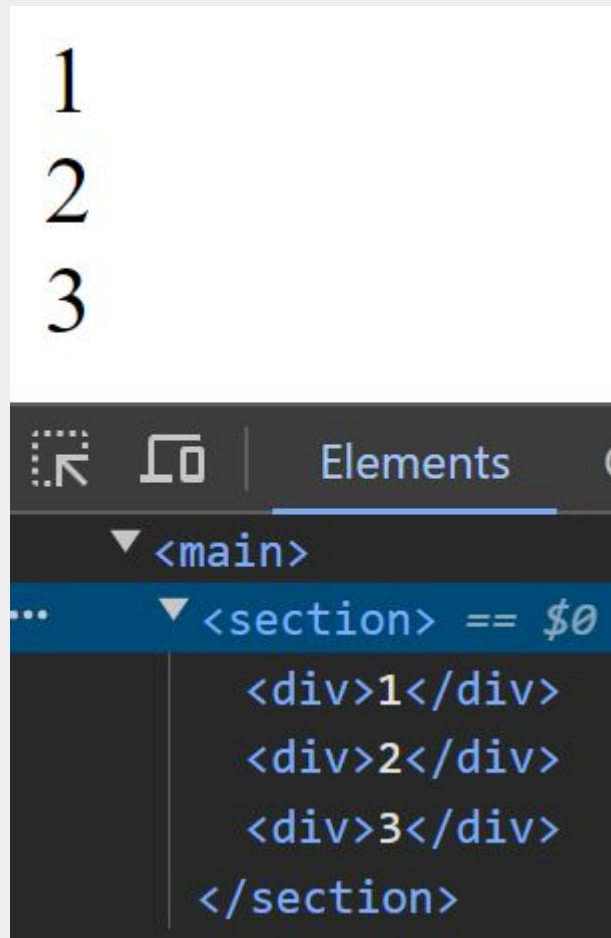
DOM



InnerHTML

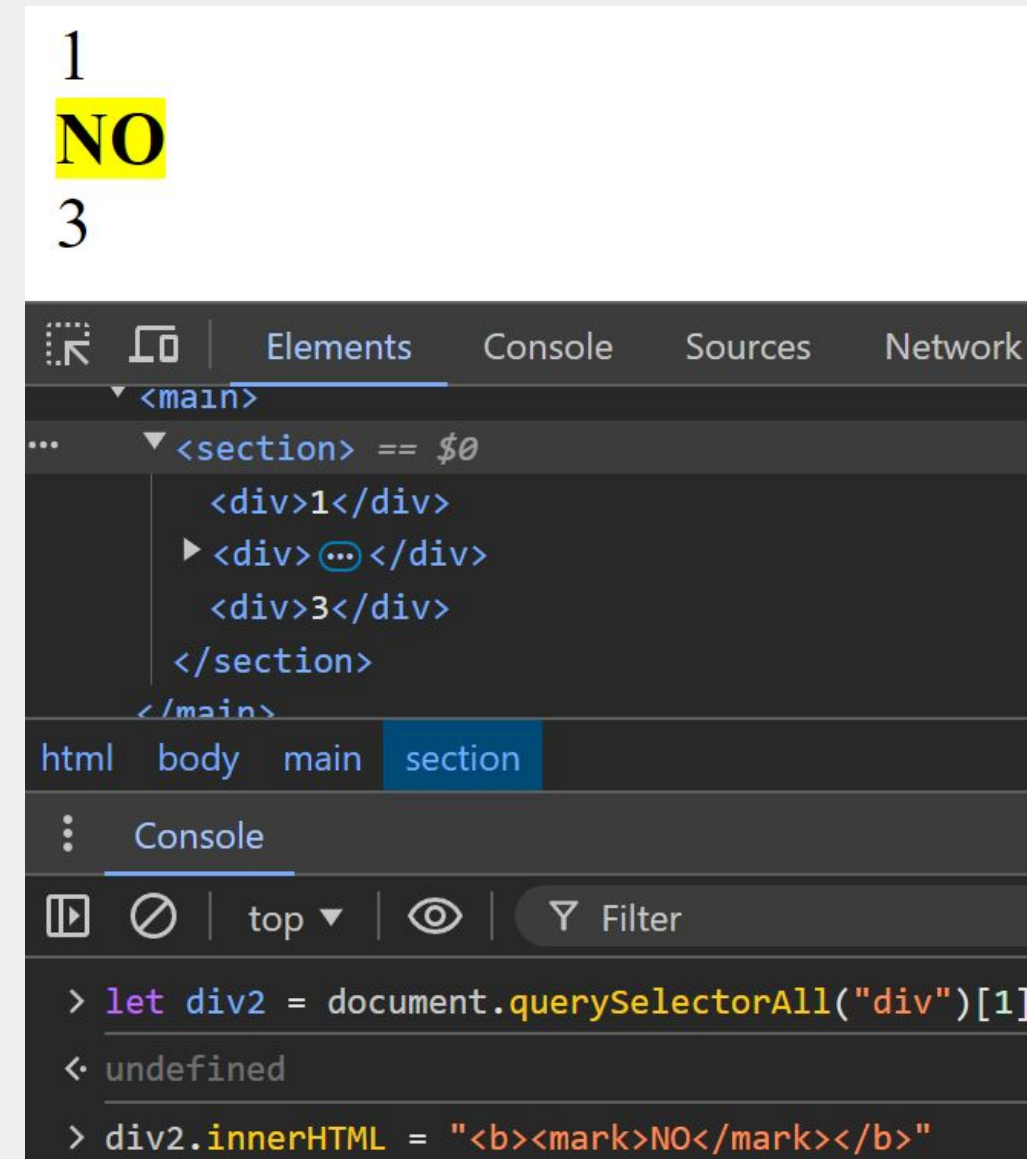
Permite ver o recrear el contenido de un elemento o etiqueta. Recibe un string.

```
1  
2  
3
```



```
<main>  
  <section> == $0  
    <div>1</div>  
    <div>2</div>  
    <div>3</div>  
  </section>  
</main>
```

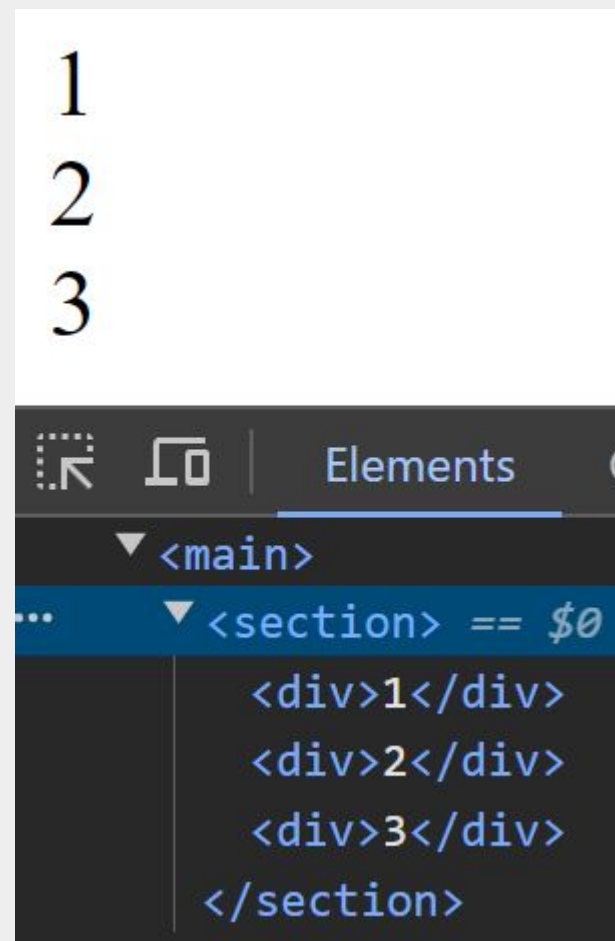
```
1  
NO  
3
```



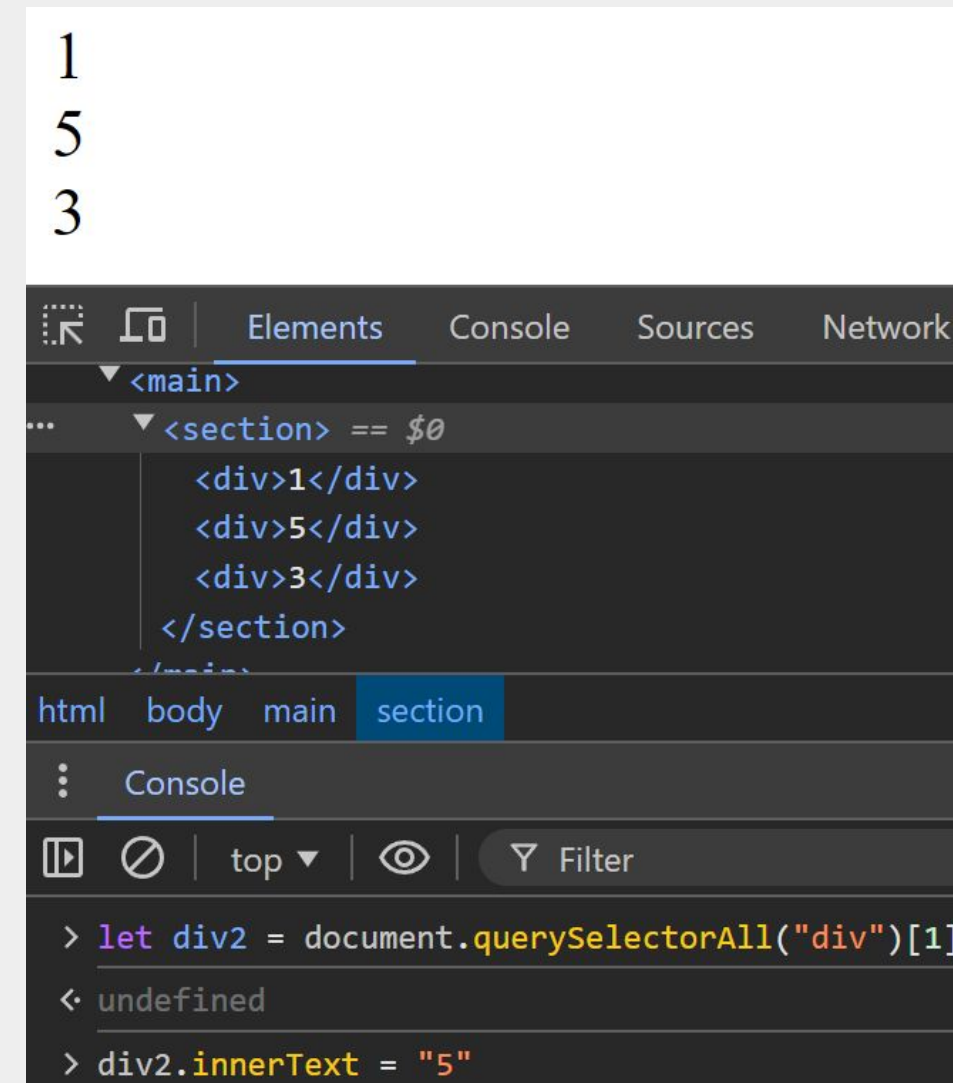
```
html body main section  
Console  
top  
Filter  
> let div2 = document.querySelectorAll("div")[1]  
< undefined  
> div2.innerHTML = "<b><mark>NO</mark></b>"
```


InnerText

Permite ver o recrear el texto de un elemento o etiqueta. Recibe un string.



The screenshot shows a web browser with a white background containing the numbers 1, 2, and 3 stacked vertically. Below the browser window, the Chrome DevTools 'Elements' panel is open. It shows a tree structure where the root is `<main>`, which contains a `<section>` element. The `<section>` element is selected and expanded, showing its children: `<div>1</div>`, `<div>2</div>`, and `<div>3</div>`.



The screenshot shows a web browser with a white background containing the numbers 1, 5, and 3 stacked vertically. Below the browser window, the Chrome DevTools 'Elements' and 'Console' panels are open. The 'Elements' panel shows the same tree structure as the previous image, but the `<div>5</div>` element is selected. The 'Console' panel shows the following code and output:

```
> let div2 = document.querySelectorAll("div")[1]
< undefined
> div2.innerText = "5"
```

Crear, agregar y eliminar elementos

DOM



CreateElement

Crear un elemento, pero no lo agrega al DOM.



```
const h2 = document.createElement("h2")
```

```
h2.innerText = "Hello!"
```

Append

Inserta el elemento creado en el DOM en la parte inferior del elemento de destino.
Usamos prepend para insertar al inicio de la etiqueta.



```
const h2 = document.createElement("h2");  
  
h2.innerText = "Hello!";  
  
document.body.append(h2);
```


Remove

Elimina elementos del DOM.



```
let h2 = document.querySelector("h2");  
h2.remove();
```



¡DEMO SPRINT 2!



15/10/2025



Tienen que presentar:

- Navbar desde el array recorrido con for of (Ejercicio de la clase 13).
- Array de productos y mostrar datos en la card (Ejercicio de la clase 16).
- **Ejercicio de la clase 18.**



Que el css sea decente.



¡Manos a la obra!

SPRINT 2

Ejercicio 1

No entregable.

*Subir en una nueva rama
llamada ejercicios (o el
nombre que eligieron)*

- Crear un HTML de práctica vinculado a un Js.
- Mediante un prompt que el usuario vaya construyendo una página con elementos.
- Los elementos serán:
 - button
 - input
 - textarea
 - h1
 - p
- Si los elementos contienen texto agregarlos por prompt.

SPRINT 2

Ejercicio 1

De clase 15. Entregable.

- Crear un array js de productos con un prompt AI.
- Introducirlo en main.js asignándolo a la variable data.
- Asignar data.map a la variable que habíamos asignado antes un array vacío, implementando las propiedades de los objetos del array en el marcado de una card.
- Crear la lista de productos e insertarlos en el elemento <main />.

SPRINT 2

Ejercicio 2

De clase 15. Entregable.

- En los ver más de la Home, en el `<a href,` luego de `.html` poner `?prod=${producto.id}`
- Usar el array de productos.
- Filtrarlo por el número de producto obtenido con `window.location`.
- Lograr que el producto mostrado cambie dependiendo del número de producto en la URL.

¡DEMO SPRINT 2!



15/10/2025



Tienen que presentar:

- Navbar desde el array recorrido con for of (Ejercicio de la clase 13).
- Array de productos y mostrar datos en la card (Ejercicio de la clase 16).
- **Ejercicio de la clase 18.**



Que el css sea decente.



retro

¿Cómo nos fué?

¿Qué cosas no quedaron claras y
necesitamos repasar la próxima?

