

¡Javascript!

guayerd 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?

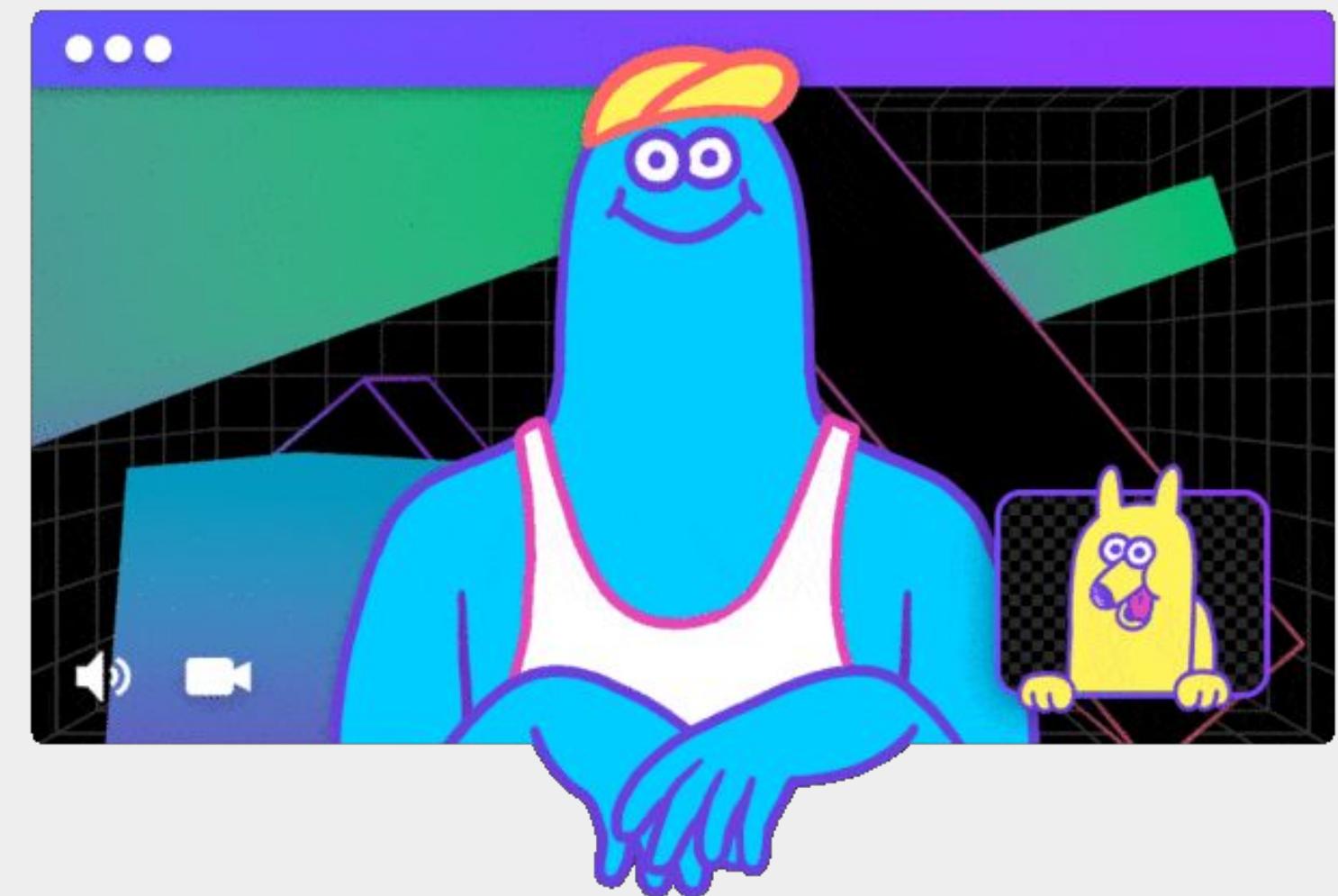
¿Algo nos bloquea?

¿Cómo seguimos?



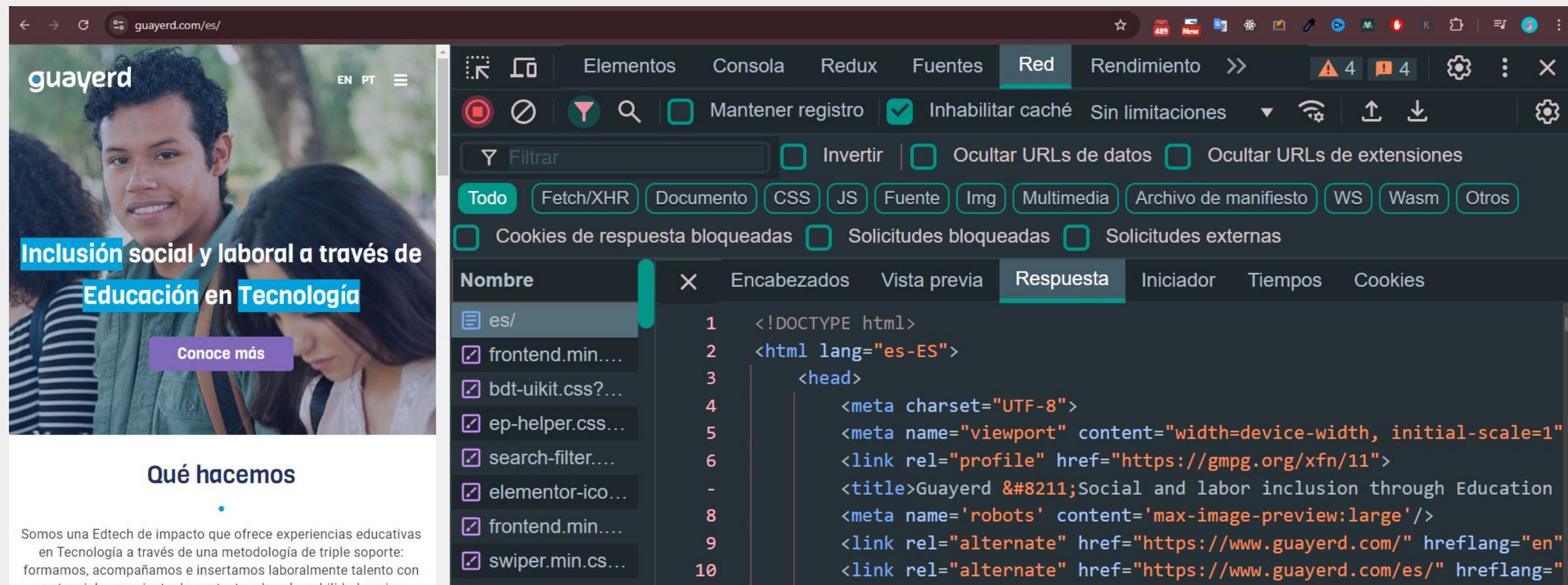
DOM

Document Model Object



Respuesta del servidor a una petición

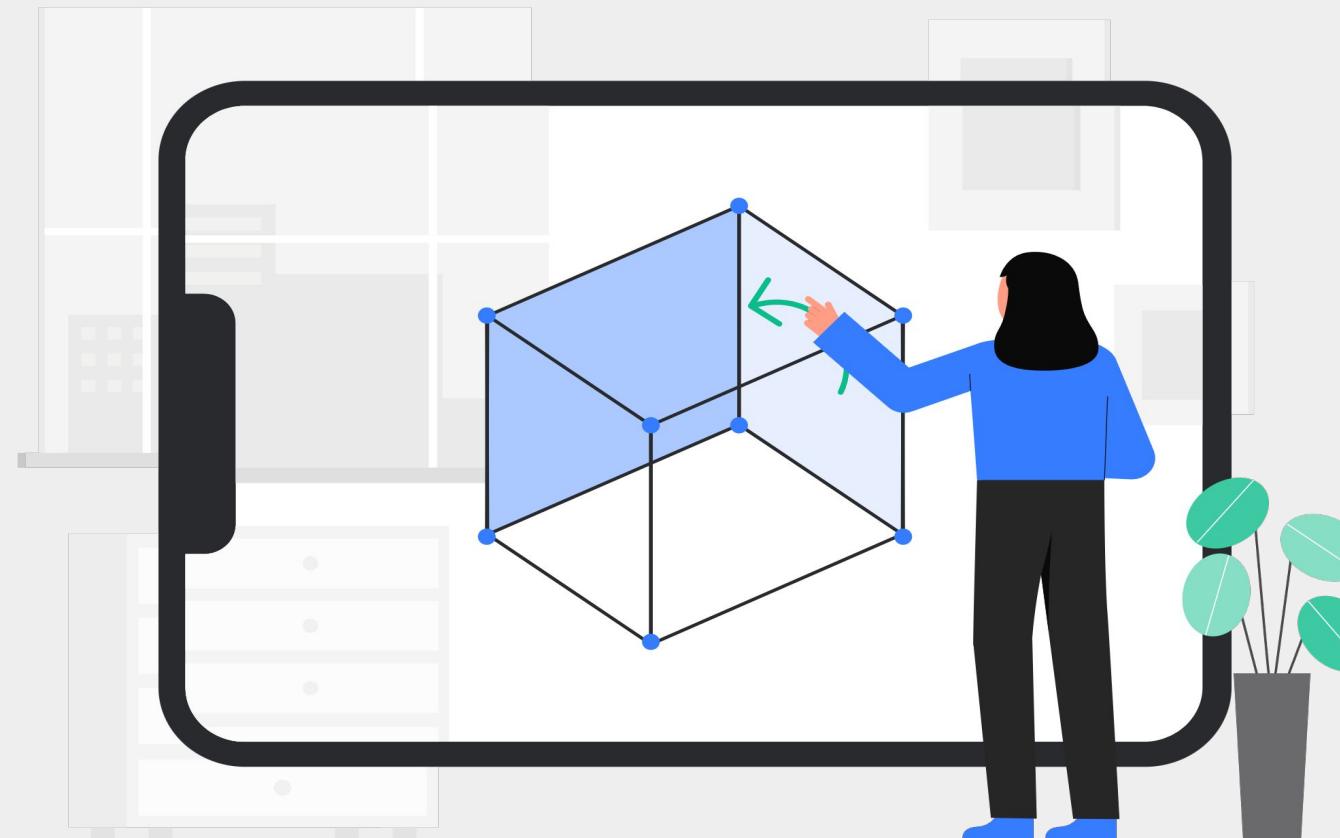
Cuando queremos entrar a una app web, hacemos una petición al servidor que nos devuelve un HTML. El navegador convierte esta respuesta en un objeto en forma de árbol, llamado DOM.



SPRINT 2

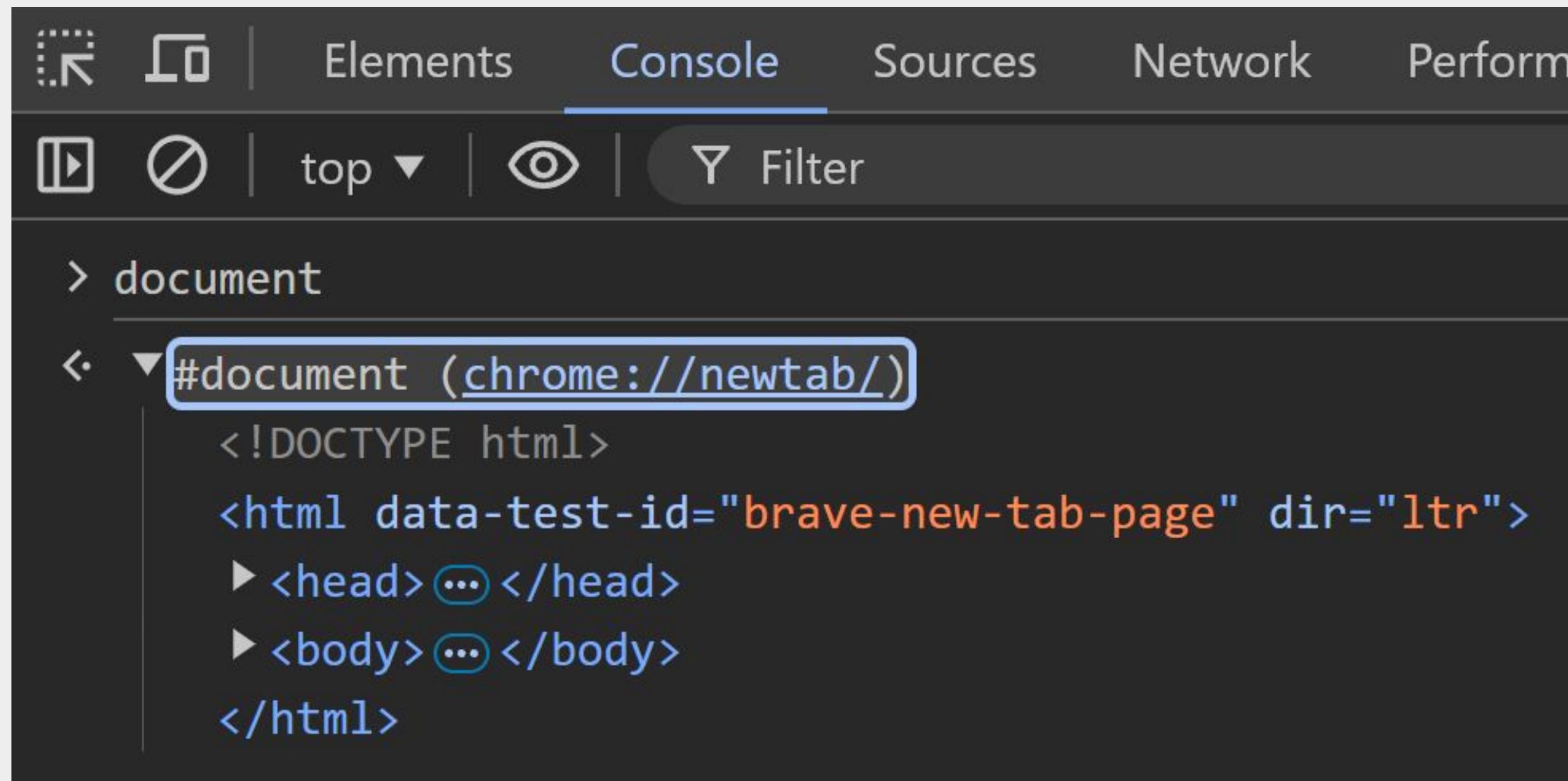
DOM

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



The screenshot shows the Chrome DevTools interface with the "Elements" tab selected. The main area displays the DOM tree for the current page. At the top, there are icons for play, stop, and refresh, followed by dropdown menus for "top" and "Filter". The DOM tree starts with the root element "document" (highlighted with a blue border), which contains the following structure:

```
> document
<-- ▼ #document (chrome://newtab/)

  <!DOCTYPE html>
  <html data-test-id="brave-new-tab-page" dir="ltr">
    ▶ <head> ...
    ▶ <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, 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-con
tent-height: 2775px;">
    ▼<div class="App--suoy3r gvmVNR"> flex
      ::before
      ▶ <div class="StyledPage--h5buhi fGBYgr">...</div> grid
      ▶ <leo-dialog showclose class="SettingsDialog--1ffucv0 jDuJrT">...</leo-dialog>
    </div>
  </div>
</body>
```

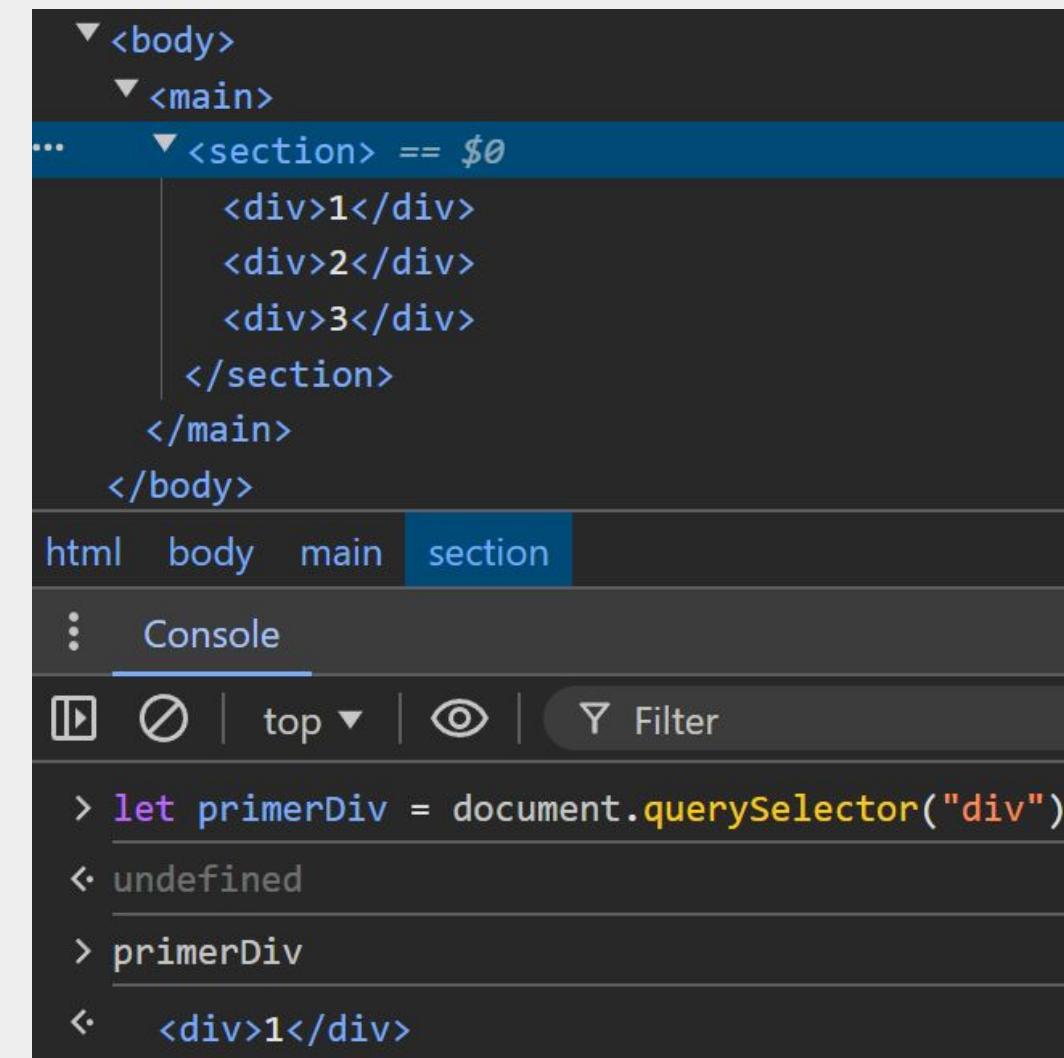
Métodos de acceso

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.



The screenshot shows a browser's developer tools console. At the top, there is a tree view of the DOM structure:

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

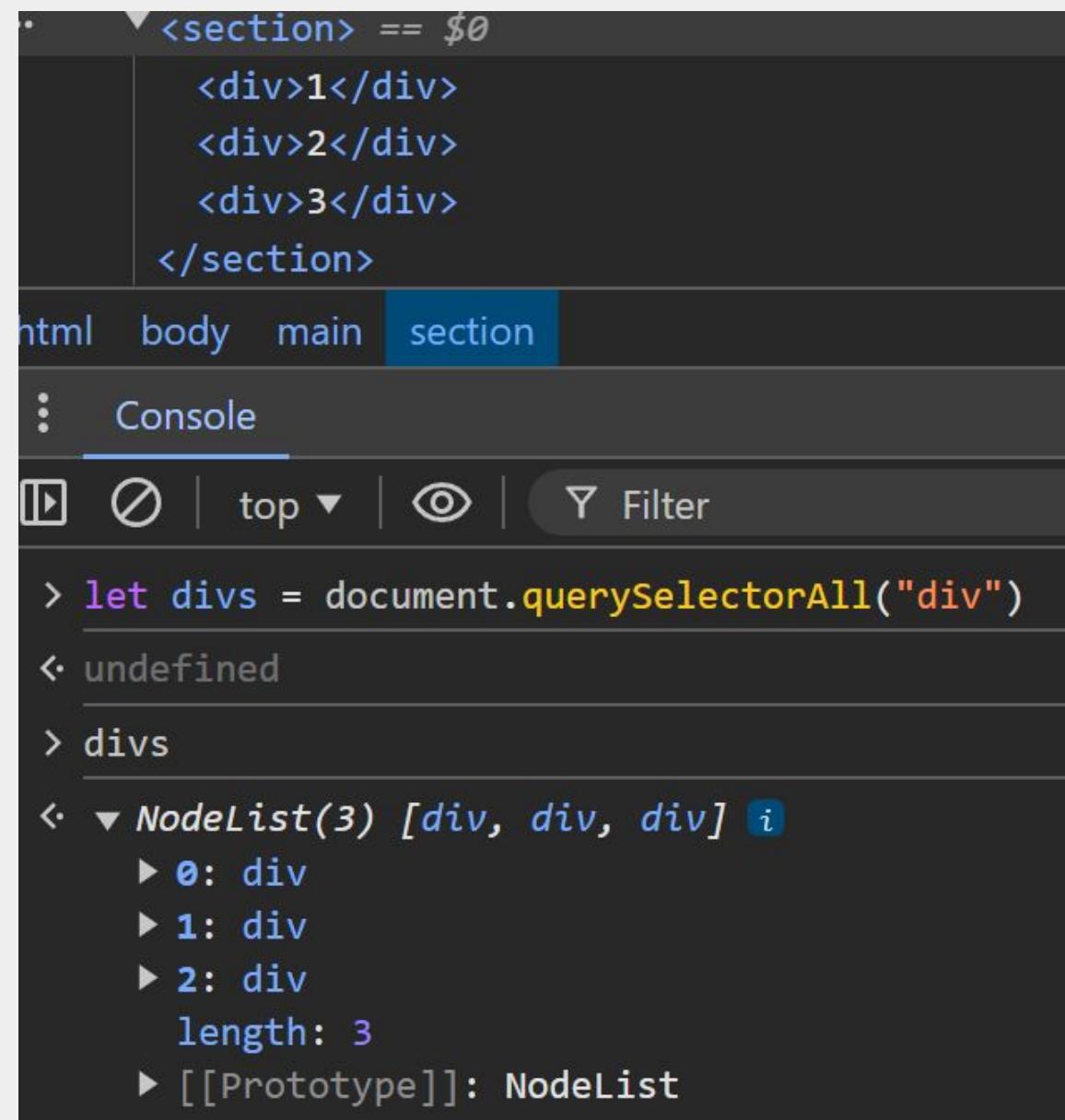
Below the DOM tree, the console tab is selected, showing the following interaction:

```
html body main section
⋮ Console
▶ ⏺ | top ▾ | ⏻ | Filter
> let primerDiv = document.querySelector("div")
< undefined
> primerDiv
< <div>1</div>
```

The code `document.querySelector("div")` is run, resulting in the first `div` element from the DOM tree being returned.

querySelectorAll

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

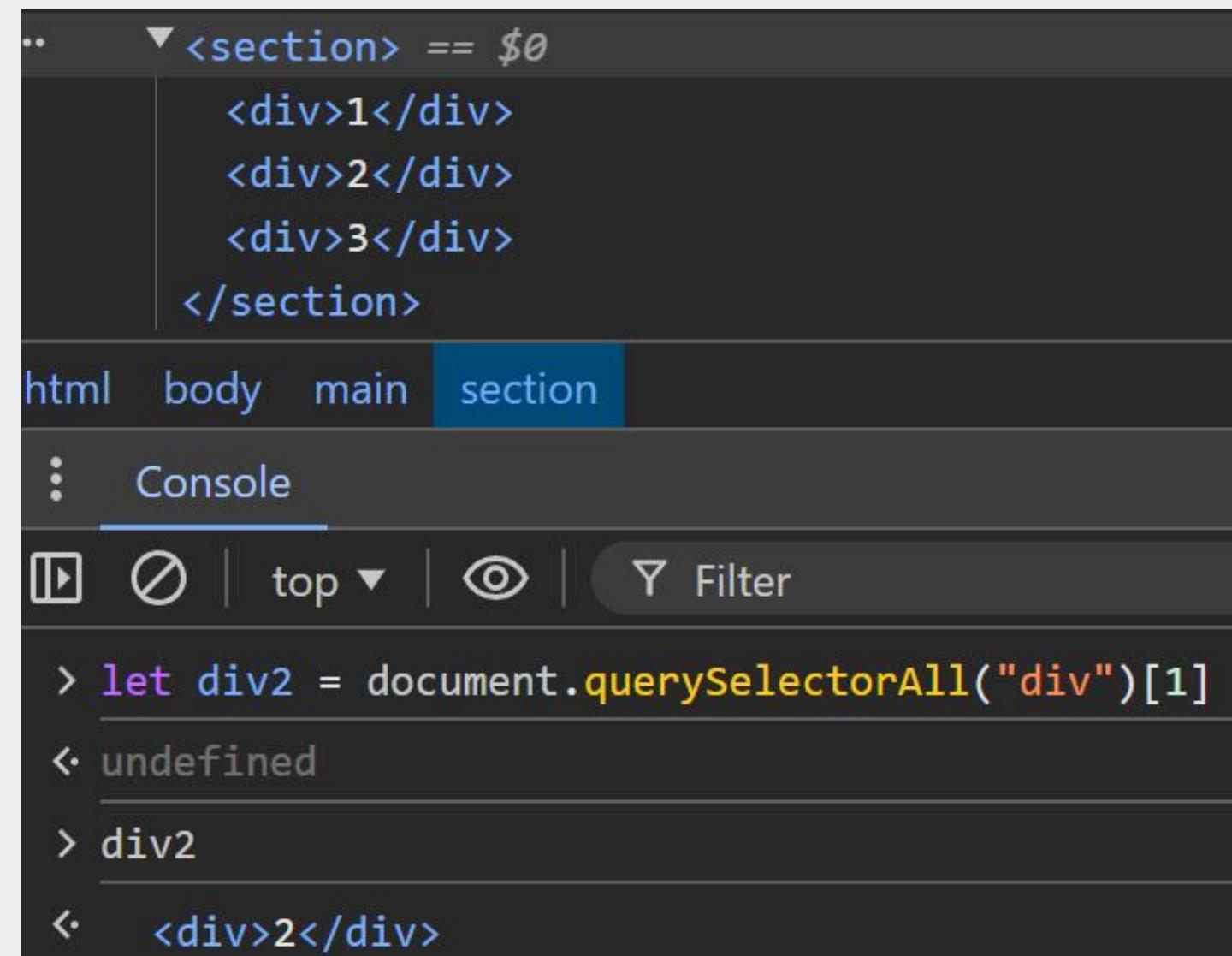


The screenshot shows a browser developer tools console. At the top, there's a tree view of the DOM structure, specifically focusing on a section element containing three div elements. Below the tree, tabs for 'html', 'body', 'main', and 'section' are visible, with 'section' being the active tab. The main console area has a title 'Console'. Below the title are buttons for 'Run', 'Clear', 'Top', 'Scope', and 'Filter'. The console output shows the following code and its execution 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.



The screenshot shows a browser's developer tools console. The DOM tree on the left shows a section element containing three div elements, indexed from 1 to 3. The 'section' tab is selected in the tabs bar. The console below shows the following interaction:

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

Métodos antiguos

Pero que aún se usan.

| | |
|----------------------------------|--|
| getElementById("id") | Recupera un elemento con el id pasado. |
| getElementsByClassName("clase") | Recupera la lista de elementos que tengan la clase. |
| getElementsByTagName("etiqueta") | Recupera la lista de elementos que tengan la etiqueta. |

Modificación de elementos

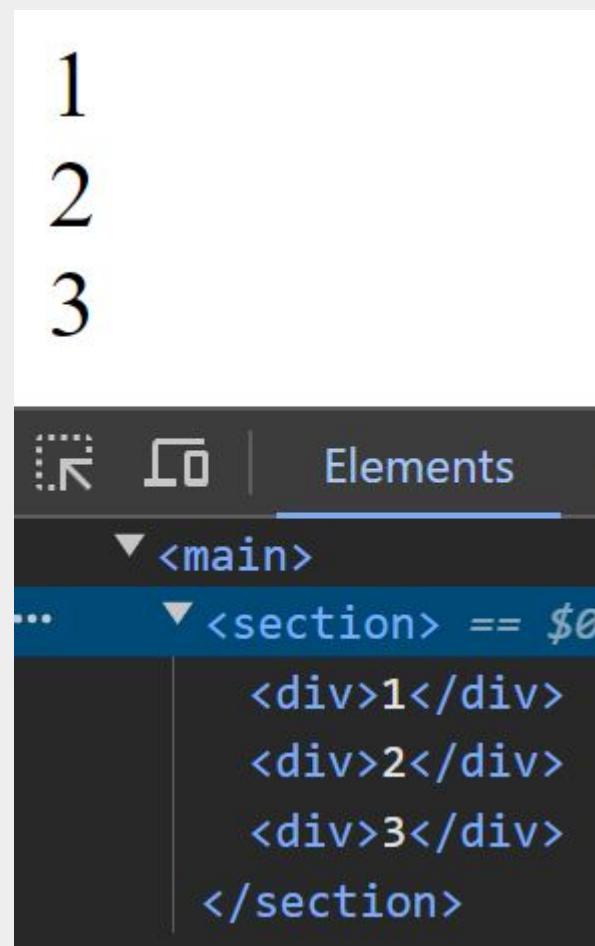
DOM



lolgif.ru

innerHTML

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



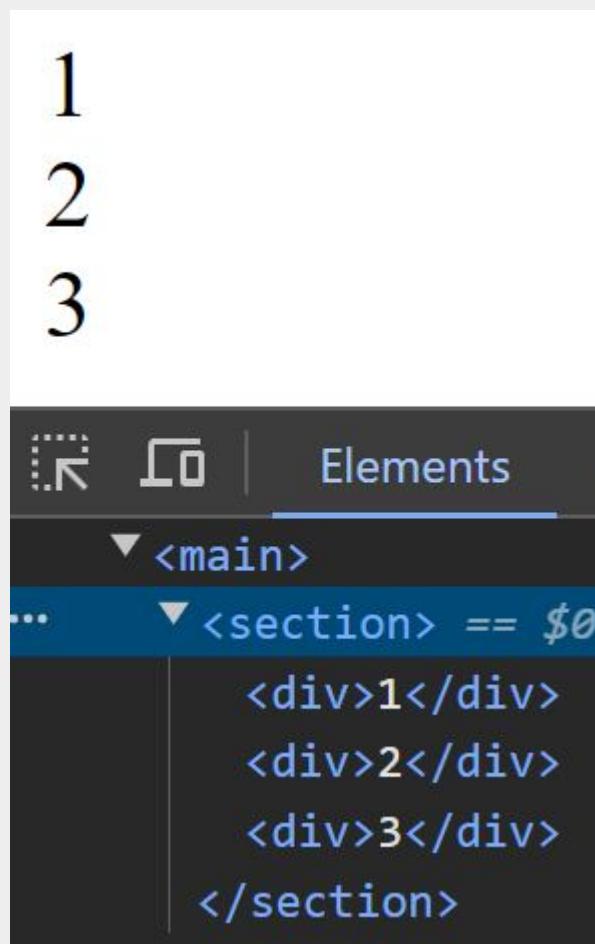
```
1
NO
3

Elements      Console      Sources      Network
<main>
  <section> == $0
    <div>1</div>
    > <div>...</div>
    <div>3</div>
  </section>
</main>

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 browser's developer tools with the 'Elements' tab selected. Inside, a 'main' element contains a 'section' element. The 'section' element has three child 'div' elements. The first 'div' has the text '1', the second has '5', and the third has '3'. Below the developer tools, the JavaScript console is open. It shows the following code being run:

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

This demonstrates changing the text content of the second 'div' from '2' to '5' using the 'innerText' property.

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();
```

guayerd



¡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

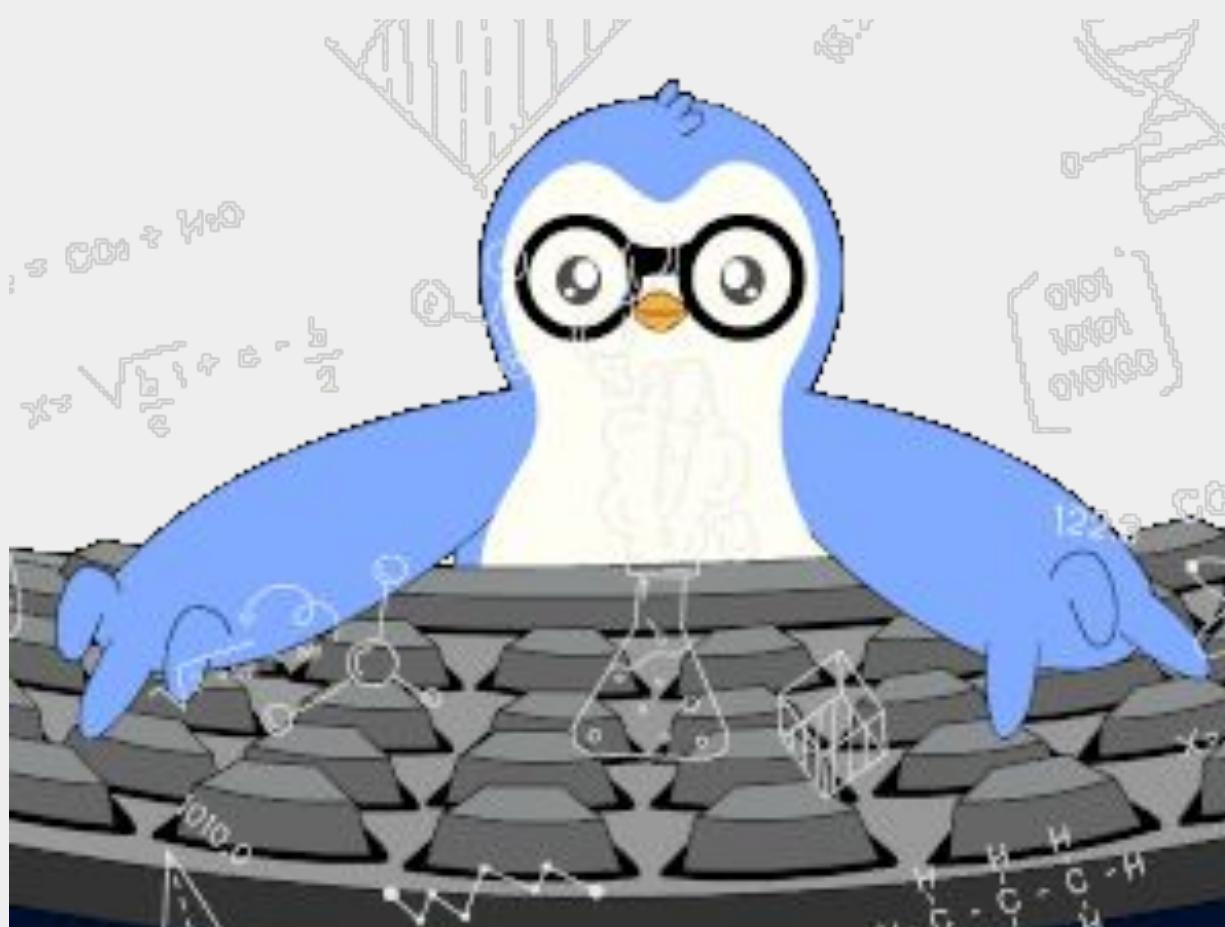
De clase 15. Entregable..

- 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?

