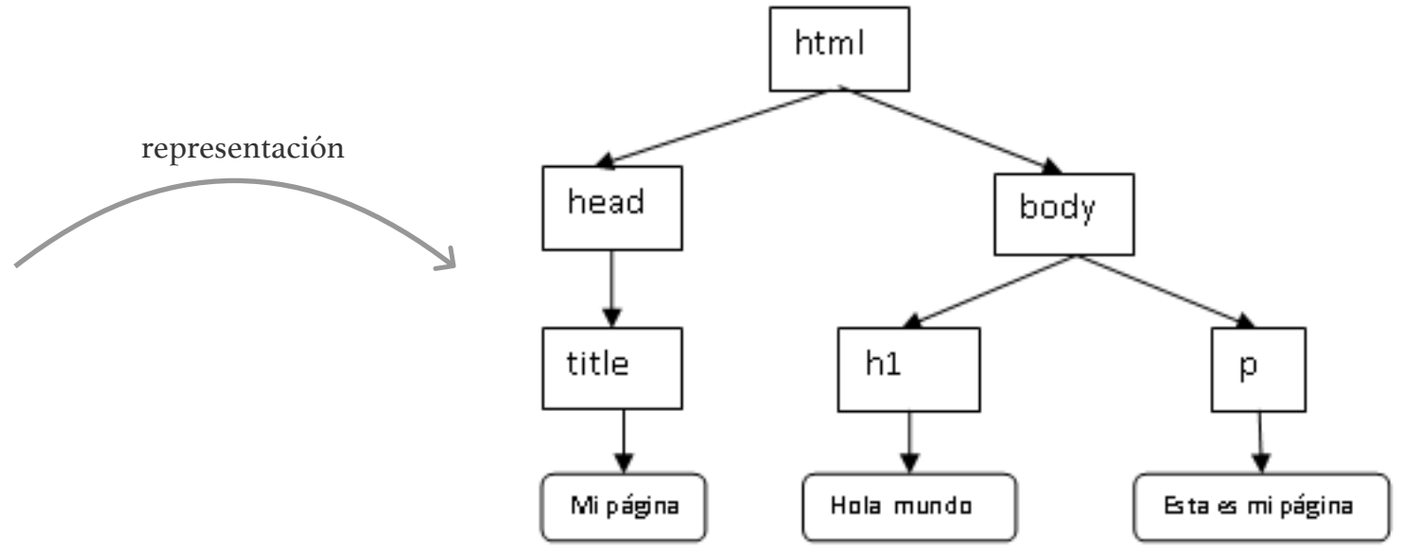
representando la representación...

El **DOM** es la representación de nuestra estructura (HTML) en Javascript, mediante el cual *no sólo podemos acceder a los bloques* (elementos) *sino que también podemos modificarlos*. Si pudiesemos graficar el DOM sería algo así:

```
<html>
  <head>
    <title>Mi página</title>
  </head>

  <body>
    <h1>Hola mundo</h1>

    <p>Esta es mi página</p>
  </body>
</html>
```



¿cómo identificar elementos?

Nuestro objetivo será entonces identificar elementos. ¿Por qué? Todo lo que querramos hacer en nuestra aplicación va a suceder en la estructura. La estructura está compuesta por elementos. Por lo cual para hacer cualquier cosa en nuestra aplicación vamos a necesitar identificarlos. **Todo ocurre sobre y en los *elementos***.

Ya utulizamos la función ***querySelector***, que recibiendo un parámetro (un selector) retornará un elemento que coincida con este selector.

Introducimos ahora a ***querySelectorAll***, que en vez de devolver un elemento devuelve una lista de elementos que coincidan con el selector.

¿cómo identificar elementos?

Si estamos en la búsqueda de no un elemento, sino de una lista de elementos, cobrará sentido usar ***querySelectorAll***. Al igual que ***querySelector***, recibe un parámetro, pero en este caso retornará una lista de elementos de la estructura que coincidan con el selector del parámetro.

representación de los elementos

```
<h1>Título primero</h1>

<h1>El segundo de los títulos</h1>
```

```
window.onload = function() {
  const titulos = document.querySelectorAll('h1')
}
```

array con dos elementos

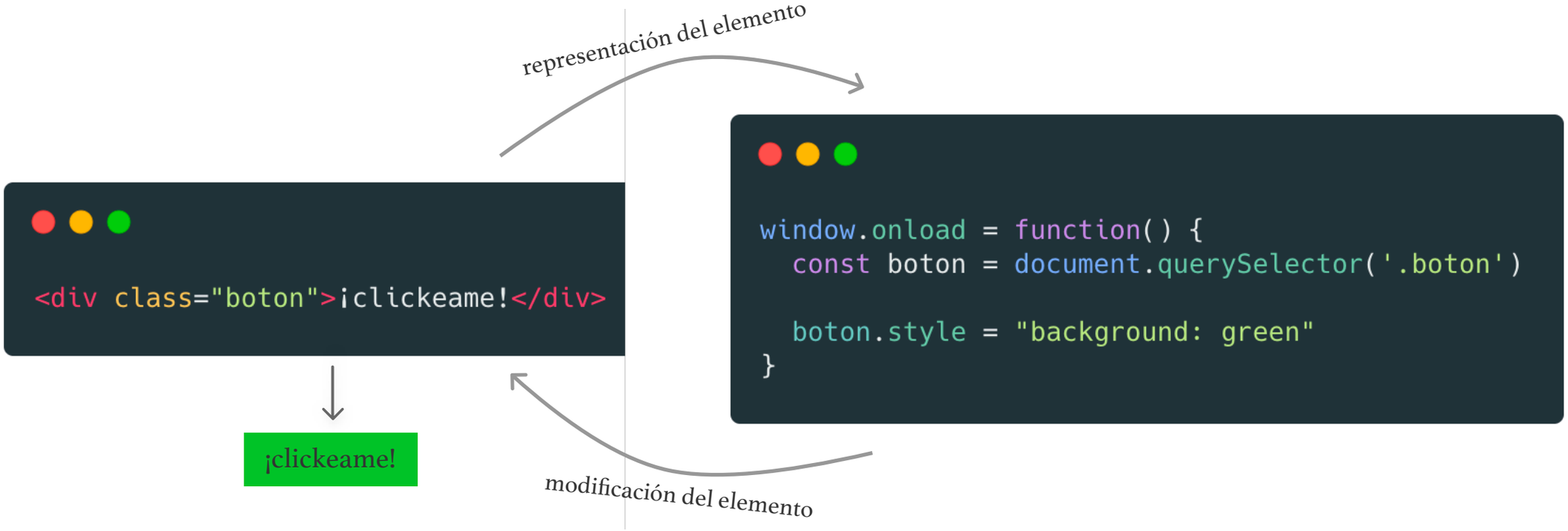
¿qué hacer con los elementos?

Dijimos que los elementos, mediante el *DOM*, los representamos como objetos. Como todo objeto, un elemento del *DOM* tiene propiedades. Propiedades que al modificar su valor, modificaremos al elemento de la estructura.

¿Qué propiedades podemos modificar?

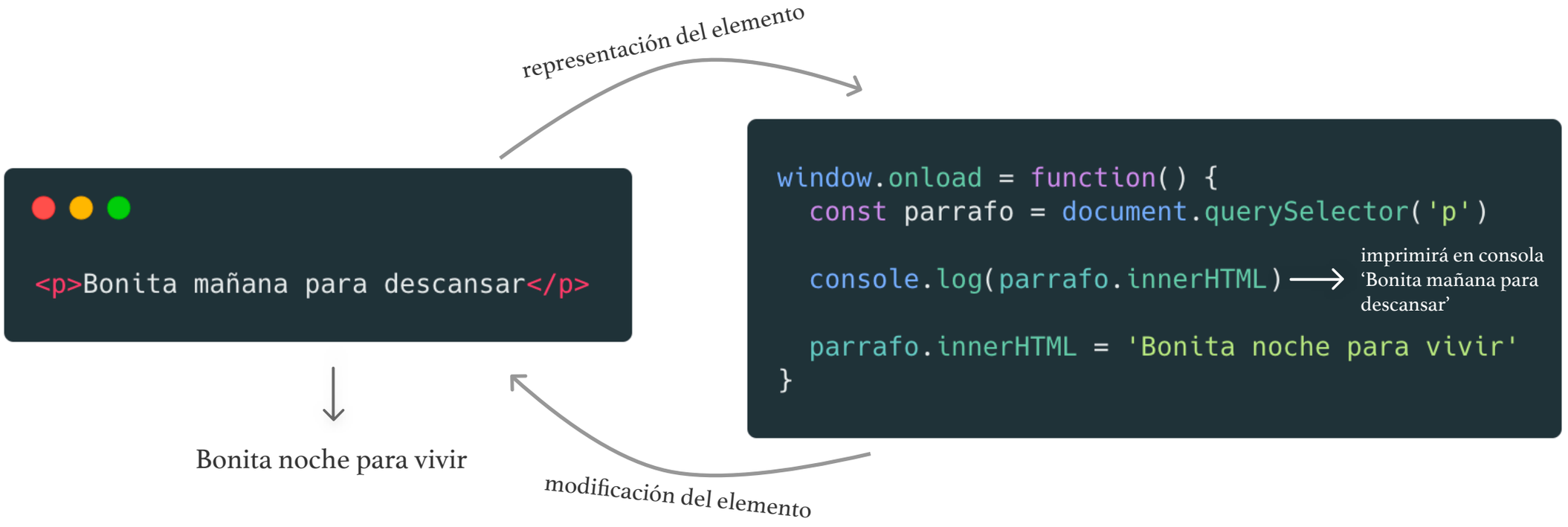
¿qué hacer con los elementos?

Mediante la propiedad ***style*** tendremos la posibilidad de asignarle como valor un string con *código CSS* para darle estilo al elemento en cuestión.



¿qué hacer con los elementos?

innerHTML será la propiedad del contenido del elemento. Si nuestro elemento es un párrafo (un elemento cuyo tag es p), el contenido será el párrafo que presente. Podemos tanto acceder a él (consultarlo) como modificarlo.

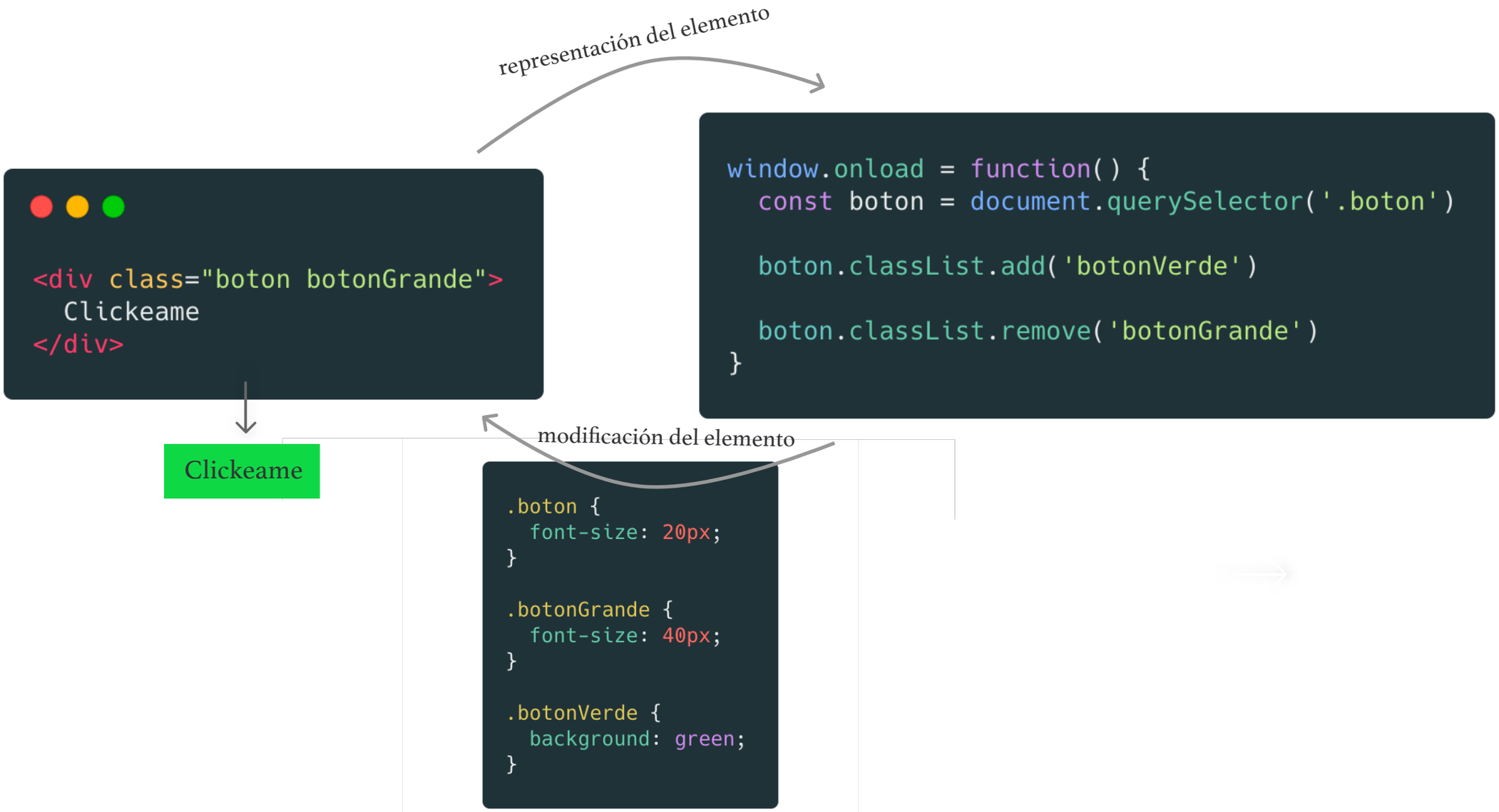


¿qué hacer con los elementos?

Existe una propiedad llamada ***classList***. Es un objeto que tienen los elementos del DOM para interactuar con las clases que estos tienen. Mediante esta propiedad podemos *agregar y borrar clases* a los elementos para así darles estilo usando reglas de CSS. Lo haremos mediante las propiedades que este objeto tiene:

- **add**: recibe una clase y la agrega a la lista de clases del elemento.
- **remove**: recibe una clase y la elimina de las clases del elemento.
- **contains**: recibe una clase y devuelve ***true*** o ***false*** si el elemento la contiene.
- **toggle**: recibe una clase, y *si el elemento no la tiene* la agrega. *Si la tiene*, la borra.

¿qué hacer con los elementos?



¿como construir elementos?

Ya identificamos elementos. Ya consultamos sus propiedades. Ya modificamos sus propiedades. Ahora, ¿qué sucede si queremos construir elementos?

Puede ocurrir que no queramos consultar elementos existenes sino **crear elementos nuevos**, lo cual, desde claro es posible.

Pero tenemos que diferenciar construir un elemento de agregarlo a la estructura, *al igual que diferenciamos entre construir un ladrillo de agregarlo a una construcción*.
Entendiendo esto, resulta claro que la acción de poner un bloque en nuestra estructura se divide en dos: **construirlo** y **agregarlo**.

¿como construir elementos?

El objeto que representa a la estructura, ***document***, tiene una propiedad para conseguir esto. ***createElement*** es una función que recibe un tag y **construye un elemento con este tag**.

Este elemento sólamente está creado en Javascript y no así en la estructura. Es un elemento que aún no fue depositado en HTML, pero de todos modos tiene todas las propiedades que tienen los objetos que representan elementos de *DOM*.

¿como construir elementos?

```
window.onload = function() {
  const nuevoBoton = document.createElement('div')

  nuevoBoton.innerHTML = 'Clickea en este nuevo botón'

  nuevoBoton.classList.add('boton')
}
```

El código de arriba creará un elemento cuyo **tag** es ***div***, su **contenido**, *'Clickea en este nuevo botón'* y tendrá una sola **clase**: ***boton***.

Como dijimos, este elemento no aparecerá en la estructura (nuestro código HTML). ¿Por qué? Porque *construimos el ladrillo, no lo pusimos en la construcción*.

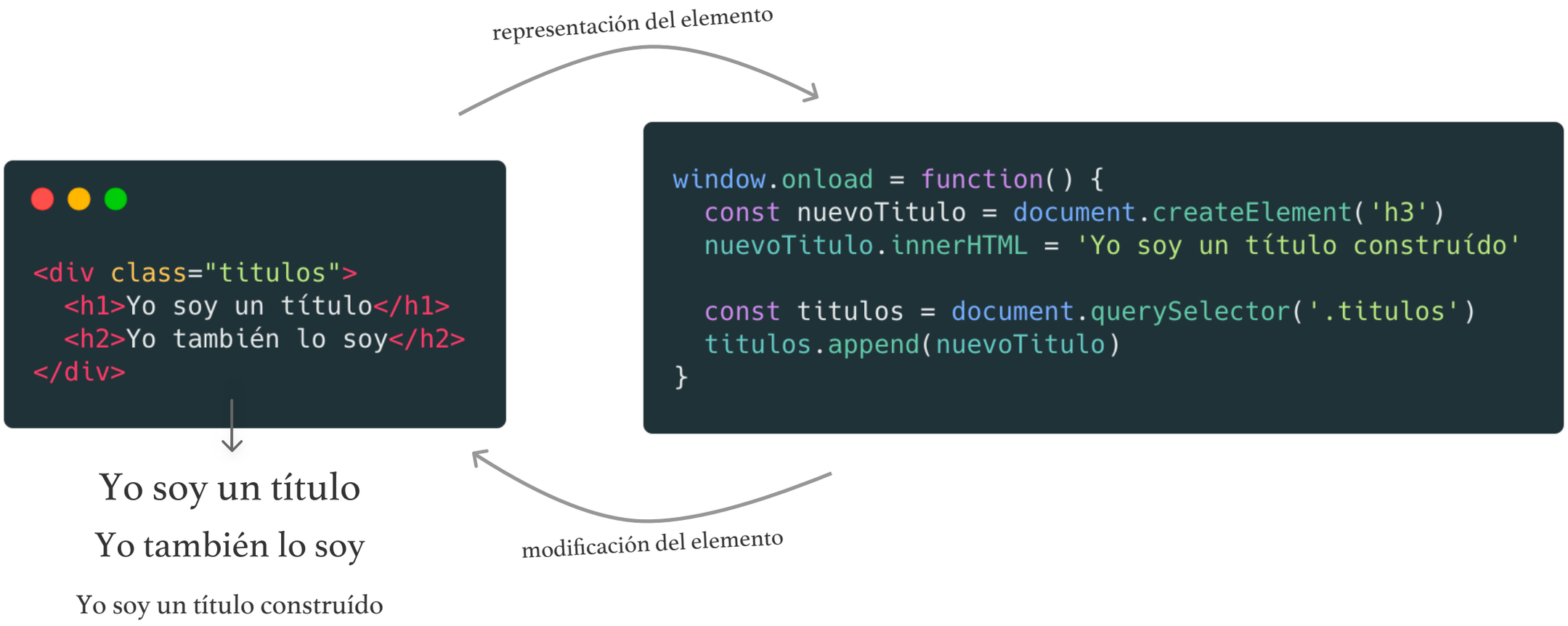
¿cómo agregar elementos a la estructura?

Una vez creado nuestro ladrillo podremos depositarlo en la construcción.

¿Pero dónde? Es una decisión que debemos tomar nosotros, como programadores. Lo cierto es que, como dijimos, **todo lo que hagamos en la estructura de nuestra aplicación ocurrirá en un elemento**. Y esta no es la excepción.

Siempre que agreguemos un elemento que hemos construido *programaticamente*, lo haremos dentro de un elemento que exista previamente en la estructura. Luego de identificarlo, podremos usar su propiedad ***append***, que recibe un elemento y lo **agrega al final de su contenido**.

¿cómo agregar elementos a la estructura?



trabajo práctico de dom ✍️

Crear una *aplicación web* que inicialmente muestre una mesa vacía. Utilizando el ***tablero de control*** (un *botón de agregar* y uno de *borrar*) se podrán agregar y borrar, de a una, las sillas que rodean la mesa. La mesa soporta hasta 8 mesas.