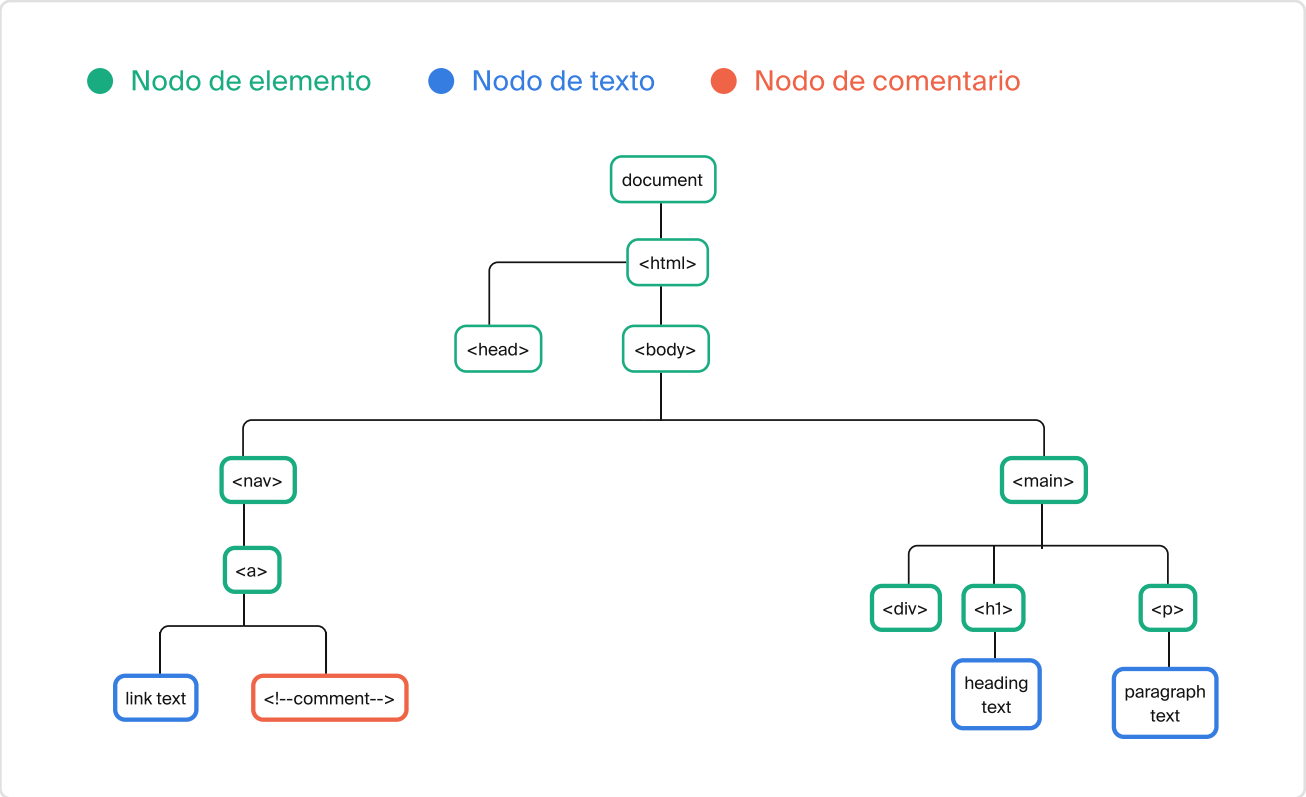
Conceptos básicos del DOM

Sprint 8

El Modelo de objetos del documento (DOM)

Una representación abstracta de un documento HTML. Consta de tres tipos de nodos: **elementos**, **texto** y **comentarios**.



Seleccionar elementos HTML

`document.querySelector` selecciona el primer elemento con un string del selector CSS coincidente:

```
// llamarlo en el documento devuelve el primer elemento coincidente en la página
const element = document.querySelector(selectorString);

// llamarlo en otro elemento devuelve el primer elemento coincidente dentro de ese elemento
const element = someOtherElement.querySelector(selectorString);
```

`document.querySelectorAll` selecciona todos los elementos con un string selector CSS coincidente:

```
// selecciona todos los elementos coincidentes en la página
const elements = document.querySelectorAll(selectorString);
```

Estos métodos pueden usarse en cualquier elemento HTML. Por ejemplo:

```
// selecciona el primer elemento con la clase `parent`
const parent = document.querySelector(".parent");

// selecciona el primer elemento contenido en un `parent` que tenga una clase `child`
const child = container.querySelector(".child");
```

En general, debes usar selectores `".class"` con `querySelector` y `querySelectorAll`

Manipular elementos del DOM

Sprint 8

Manipular los atributos de un elemento

Utiliza `element.setAttribute()` para establecer el atributo de un elemento:

```
// establece el atributo del elemento al valor proporcionado
element.setAttribute(attribute, value);

// ejemplo: establecer el atributo `src` al primer elemento con la clase `image`
document.querySelector(".image").setAttribute("src", "path/to/image");

// ejemplo: establecer el atributo booleano `disabled`
document.querySelector(".button").setAttribute("disabled", "");

// ejemplo: establecer estilos CSS en línea a través del atributo `style`
document.querySelector(".button").setAttribute("style", "background-color: #000");
```

Utiliza `element.removeAttribute()` para eliminar el atributo de un elemento:

```
// ejemplo: eliminar el atributo de un elemento al proporcinado
element.removeAttribute("disabled");
```

Muchos atributos pueden establecerse cambiando directamente la propiedad correspondiente de un elemento:

```
// ejemplo: establecer el atributo `disabled` de un elemento.
element.disabled = true;

// ejemplo: eliminar el atributo `disabled` de un elemento.
element.disabled = false;

// ejemplo: establecer los estilos en línea de un elemento, asegúrate de usar camelCase en lugar de kebab-case.
element.style.backgroundImage = "url(path/to/image)";
```

Manipular las clases de un elemento a través de `classList` .

```
// acceder a la lista de clases de un elemento
element.classList;

// agregar una clase a la lista de clase de un elemento
element.classList.add("some-class");
```

```
// eliminar una clase de la lista de clase de un elemento
element.classList.remove("some-class");

// eliminar una clase si existe, agregarla si no
element.classList.toggle("new-class");
```

Remplazar el contenido de un elemento

Remplaza por completo contenido de texto con `element.textContent` :

```
element.textContent = "remplazo del contenido de texto";
```

Remplaza por completo contenido de texto con `element.innerHTML` :

```
element.innerHTML = "<span>remplazo del contenido HTML</span>";
```

Adición al contenido de un elemento

`element.insertAdjacentText()` **inserta nuevo contenido de texto:**

```
element.insertAdjacentText(where, "inserta este texto");
```

`element.insertAdjacentHTML()` **inserta nuevo contenido HTML:**

```
element.insertAdjacentHTML(where, "<span>inserta este contenido</span>");
```

El primer parámetro de estas funciones describe dónde debe insertarse el nuevo contenido:

- `"beforebegin"`: antes del propio elemento.
- `"afterbegin"` : antes del primer hijo del elemento.
- `"beforeend"` : después del último hijo del elemento.
- `"afterend"` : después del propio elemento.

Eventos y el objeto evento

Sprint 8

Eventos

Cosas que pasan en una página web.

Los tipos comunes de eventos incluyen:

- `"click"` : se activa cuando un usuario hace clic en un elemento.
- `"mouseover"` : se activa cuando se pasa el cursor sobre un elemento.
- `"mouseout"` : se activa cuando el cursor deja de pasar sobre un elemento y se aleja.
- `"scroll"` : se activa cuando el usuario desplaza un elemento.
- `"submit"` : se activa cuando el usuario hace clic en un elemento de envío `<button>` o presiona Enter mientras edita un campo de entrada en un formulario.

Detecta eventos y reacciona ante ellos con `element.addEventListener()`:

```
// ejemplo: eliminar el atributo "disabled" de un elemento
element.removeAttribute("disabled");
```

Detecta eventos y reacciona ante ellos con `element.addEventListener()`:

```
// ejemplo: registrar un string en la consola cuando se hace clic en un botón
document.querySelector(".button").addEventListener("click", function() {
  console.log("Se ha hecho clic en el botón");
});
```

El objeto `event`

Contiene información útil sobre el evento y el elemento que lo activó.

Puedes acceder al objeto evento como el primer argumento de `addEventListener()`:

```
// por convención, lo llamamos `event`, `evt` o `e`
element.addEventListener("click", function (event) {
  console.log(event); // registra el objeto evento en la consola
});
```

Las propiedades y métodos incluyen:

- La propiedad `event.target` almacena al elemento en el que ocurrió el evento.
- `event.preventDefault()` evita que ocurra el comportamiento predeterminado del navegador cuando se activa el evento.
  - Suele usarse para evitar que la página vuelva a cargarse en eventos submit.

Elementos de plantilla

Sprint 8

`<template>`

Los elementos que contienen códigos reutilizables de marcado HTML:

```
<template id="user">    <!-- utiliza un id para seleccionar la plantilla -->
  <div class="user">
    <img class="user__avatar" alt="avatar">   <!-- el atributo src se establecerá con JavaScript -->
    <p class="user__name"></p>   <!-- textContent se establecerá con JavaScript -->
  </div>
</template>
```

Nota: Una `<template>` y su contenido no se renderizarán en la página.

Renderizar elementos `<template>` con JavaScript.

```
// parent es el contenedor donde se renderizarán los usuarios
const userList = document.querySelector(".users");

// selecciona la plantilla
const userTemplate = document.querySelector("#user")
                            .content              // accede al contenido de las etiquetas de plantilla
                            .cloneNode(true);     // clona el nodo y todo su contenido

// inserta el contenido necesario
userElement.querySelector(".user__avatar").src = "tinyurl.com/v4pfzwy";
userElement.querySelector(".user__name").textContent = "Duke, alcalde de Cormorant";

// haz que aparezca en la página
userList.append(userElement);
```

Agregar elementos al DOM

Cada una de estas funciones acepta una lista de parámetros separados por comas.

Los parámetros pueden ser nodos DOM o strings:

- `node.append(firstParam, secondParam, ...)` agrega los parámetros después del último hijo del `node`.
- `node.prepend(firstParam, secondParam, ...)` agrega los parámetros antes del primer hijo del `node`.
- `node.after(firstParam, secondParam, ...)` inserta los parámetros después del `node`.
- `node.before(firstParam, secondParam, ...)` inserta los parámetros antes del `node`.
- `node.replaceWith(firstParam, secondParam, ...)` remplaza el `node` con los parámetros.