DOM: Selecting & Changing Elements

Reach into the page: the DOM tree, getElementById, querySelector, textContent, styles and classList — and innerHTML's danger warning.

Introduction

Until now, everything you built lived in the console. The **DOM** (Document Object Model) is how JavaScript steps onto the page itself: the browser turns your HTML into a live **tree** of elements, and JavaScript holds the root of that tree in a built-in object called `document`. Select any branch of the tree, and you can change its text, its colours and its classes — live, while the user watches. *(Roman Urdu: DOM browser ke paas bana hua aap ke page ka naksha hai — JavaScript is nakshay se har cheez badal sakta hai.)*

Key points

- **`document` is your handle:** every DOM job starts from it — `document.getElementById(...)`, `document.querySelector(...)` and so on.
- **`getElementById` catches ONE element** by its `id`, and ids are unique, so you always get exactly one element back (or `null` if the id does not exist).
- **`textContent` rewrites text:** read it to get an element's text, set it to replace the text — the page updates instantly, no reload.
- **`querySelector` uses CSS rules:** it returns the **first** match for any CSS selector you already know — `#title` for an id, `.card` for a class, `p` for a tag. **`querySelectorAll`** returns **all** matches as a list you can loop over.

  (Roman Urdu: querySelector pehla match deta hai, querySelectorAll saaray matches ki list deta hai.)
- **Styles from JS:** `element.style.color = "red"` changes a style live. CSS names with a dash turn **camelCase** in JavaScript: `background-color` becomes `backgroundColor`, `font-size` becomes `fontSize`.
- **`classList` is the clean way:** `add("dark")`, `remove("dark")` and `toggle("dark")` (add if missing, remove if present) — like a light switch for a whole set of styles at once.
- **`innerHTML` is dangerous with user text:** `innerHTML` **parses** whatever you give it as real HTML, while `textContent` treats everything as plain text. Never put user-typed text into a page with `innerHTML` — a user can type real tags and scripts and inject them into your page.

  (Roman Urdu: User ki likhi cheez kabhi innerHTML se page mein na daalo — woh apna code ghusa sakta hai.)

Code examples

Example 1 — Catch a heading and rewrite it live

```
<!DOCTYPE html>
<html>
<head>
  <title>DOM Example 1</title>
</head>
<body>
  <h1 id="title">Old Title</h1>
  <p id="msg">This text will change too.</p>

  <script>
    const heading = document.getElementById("title");
    heading.textContent = "Welcome to Code Scope";

    const para = document.getElementById("msg");
    para.textContent = "JavaScript changed this line live.";
  </script>
</body>
</html>
```

What it does: `getElementById("title")` catches the heading by its id — the heading is caught. Setting `textContent` replaces its text right away. The same two steps catch the paragraph and rewrite it. The script sits at the end of the body, so the elements already exist when JavaScript looks for them.

What you will see: the page shows the heading `Welcome to Code Scope` and the line `JavaScript changed this line live.` — the words `Old Title` never appear, because the change happens before you can notice them.

Example 2 — querySelector, styles and the classList light switch

```
<!DOCTYPE html>
<html>
<head>
  <title>DOM Example 2</title>
  <style>
    .dark { background-color: #0b0f19; color: #faf9f6; }
  </style>
</head>
<body>
  <h1 class="heading">Code Scope</h1>
  <p class="note">First note.</p>
  <p class="note">Second note.</p>

  <script>
    const heading = document.querySelector(".heading");
    heading.style.color = "green";
    heading.style.fontSize = "40px";

    const notes = document.querySelectorAll(".note");
    notes[1].style.color = "blue";

    document.body.classList.toggle("dark");
  </script>
</body>
</html>
```

What it does: `querySelector(".heading")` finds the first element with that class, and its styles are changed with camelCase names — `fontSize` is the JavaScript form of `font-size`. `querySelectorAll(".note")` collects **both** notes into a list, and only the second one (index `1`) is turned blue. Finally, `classList.toggle("dark")` flips the light switch once: the `dark` class is added to the body, so its CSS takes over the whole page.

What you will see: a dark page with a big green heading `Code Scope`, the first note in the page's light text colour, and the second note in blue.

Example 3 — The colour-changer page

```
<!DOCTYPE html>
<html>
<head>
  <title>Colour Changer</title>
</head>
<body>
  <h1 id="title">Pick a colour</h1>
  <button onclick="paint('red')">Red</button>
  <button onclick="paint('green')">Green</button>
  <button onclick="paint('blue')">Blue</button>

  <script>
    function paint(colour) {
      document.body.style.backgroundColor = colour;
      const heading = document.getElementById("title");
      heading.textContent = "Background is now " + colour;
    }
  </script>
</body>
</html>
```

What it does: One function does all the work: it sets the body's `backgroundColor` (the camelCase form of `background-color`) and rewrites the heading with the colour's name. Each button calls the same function with a different colour word, so three buttons repaint the page in three colours without a single reload.

What you will see: a white page with three buttons. Click `Red` and the whole background turns red and the heading reads `Background is now red`. Click `Green`, then `Blue` — the page and the heading change instantly each time.

Quick recap

- The DOM is the browser's live tree of your HTML, and `document` is how JavaScript walks it.
- Select with `getElementById` (one, by id) or `querySelector` / `querySelectorAll` (CSS selectors, first / all).
- Change text with `textContent`, styles with camelCase `style` names, and whole looks with `classList` — and keep user text away from `innerHTML`.

Common mistakes

1. **Script runs before the element exists** — a script in the head looks for elements that are not built yet, gets `null`, and the next line throws an error. The fix: put your script at the end of the body, after the HTML it needs.

2. **Writing CSS names with dashes in JavaScript** — `element.style.background-color` is invalid and breaks the line. The fix: camelCase every dashed name — `backgroundColor`, `fontSize`, `marginTop`.
3. **Putting user-typed text in with innerHTML** — the browser parses it as HTML, so a user can inject tags, fake buttons or scripts into your page. The fix: show user text with `textContent` only; use `innerHTML` only for markup you wrote yourself.

Practice

1. Build a page with a heading that has an id and two paragraphs that share a class. Select the heading with `getElementById` and change its text, then select both paragraphs with `querySelectorAll` and give only the second one a new colour.
2. Extend the colour-changer page: add two more buttons (orange and purple), and make the `paint` function also change the heading's text colour to white whenever a colour is picked.

Practice quiz

1. What is the difference between querySelector and querySelectorAll?

- There is none
- querySelector returns the first match, querySelectorAll returns all matches as a list you can loop over
- querySelector works only on ids, querySelectorAll only on classes
- querySelector returns a list, querySelectorAll returns one element

Answer: querySelector returns the first match, querySelectorAll returns all matches as a list you can loop over

2. How do you write the CSS property background-color in JavaScript?

- element.style.background-color
- element.style.backgroundColor, dashed names become camelCase
- element.style.BackgroundColor
- element.style.bg-color

Answer: element.style.backgroundColor, dashed names become camelCase

3. How must user-typed text be put on the page?

- With innerHTML, always
- With textContent, which treats it as plain text; innerHTML parses it as real HTML and lets a user inject tags and scripts
- With console.log
- User text can never be shown

Answer: With textContent, which treats it as plain text; innerHTML parses it as real HTML and lets a user inject tags and scripts

Watch the video lesson: https://youtu.be/9Cvyt7ZnMII

Read online: https://scopetreasure.com/notes/js-dom-selecting-changing · More free tools and notes at https://scopetreasure.com