1. Selecting Elements

- **Name:** querySelector
    - **Work:** Selects the first element matching a CSS selector.
    - **Where:** Used on document, e.g., document.querySelector(".myClass").
    - **Why:** To target a specific element for manipulation.
    - **Example:** let para = document.querySelector("p");
    - **Explanation:** This grabs the first <p> tag in your HTML. You can use any CSS selector like #id, .class, or even div p.
- **Name:** querySelectorAll
    - **Work:** Selects all elements matching a CSS selector, returns a NodeList.
    - **Where:** Used like document.querySelectorAll(".item").
    - **Why:** To work with multiple elements at once.
    - **Example:** let items = document.querySelectorAll(".item"); items.forEach(item => item.style.color = "blue");
    - **Explanation:** This selects all elements with class item and turns their text blue. NodeList needs a loop (like forEach) to act on all items.
- **Name:** getElementById
    - **Work:** Selects an element by its unique ID.
    - **Where:** Used as document.getElementById("header").
    - **Why:** Quick way to grab one element with an ID.
    - **Example:** let header = document.getElementById("header"); header.textContent = "New Title";
    - **Explanation:** This finds the element with id="header" and changes its text to "New Title".

---

2. Creating Events

- **Name:** addEventListener
    - **Work:** Adds an event handler (like click, hover) to an element.
    - **Where:** Used on an element, e.g., element.addEventListener("click", fn).
    - **Why:** Makes your page interactive.
    - **Example:** button.addEventListener("click", () => alert("Clicked!"));
    - **Explanation:** When the button is clicked, an alert pops up. You can add multiple listeners to the same element.
- **Name:** onclick
    - **Work:** Assigns a function to the click event directly.
    - **Where:** Used like element.onclick = function() {}.
    - **Why:** Simpler but limited to one event handler.
    - **Example:** button.onclick = () => console.log("Button clicked");
    - **Explanation:** Clicking the button logs a message to the console. If you add another onclick, it overwrites this one.

---

3. Updating or Creating Classes

- **Name:** classList.add
  - **Work:** Adds a class to an element.
  - **Where:** Used like element.classList.add("active").
  - **Why:** To apply styles or mark states dynamically.
  - **Example:** div.classList.add("highlight");
  - **Explanation:** Adds the highlight class to div, which could trigger CSS like background: yellow.
- **Name:** classList.remove
  - **Work:** Removes a class from an element.
  - **Where:** Used like element.classList.remove("active").
  - **Why:** To remove styles or reset an element.
  - **Example:** div.classList.remove("highlight");
  - **Explanation:** Removes the highlight class, reverting any associated styles.
- **Name:** classList.toggle
  - **Work:** Adds a class if absent, removes it if present.
  - **Where:** Used like element.classList.toggle("visible").
  - **Why:** Perfect for on/off states (e.g., show/hide).
  - **Example:** div.classList.toggle("visible");
  - **Explanation:** If visible isn't on the div, it's added; if it's there, it's removed—great for toggling visibility with CSS.

    ```
    const btn = document.getElementById("toggleBtn");

    btn.addEventListener("click", function() {

        document.body.classList.toggle("dark-mode"); // Toggles dark mode

    });
    ```

    This will add/remove the `"dark-mode"` class every time you click the button.

---

4. Adding Elements, Tags, and Attributes

- **Name:** createElement: **`document.createElement()` only allows creating a single element at a time**, so you **cannot** do this:
- document.createElement(" Hey Sid, how are you? "); // ❌

Instead do this:

const div = document.createElement("div"); // **Create **

const p = document.createElement("p"); // **Create **

p.textContent = "Hey Sid, how are you?"; // **Add text to **

div.appendChild(p); // **Add inside **

document.body.appendChild(div); // **Add to the page**

    - **Work:** Creates a new HTML element.
    - **Where:** Used like document.createElement("span").
    - **Why:** To dynamically build page content.
    - **Example:** let span = document.createElement("span");
    - **Explanation:** Creates a element in memory (not yet on the page) that you can customize.you have to use append after it.
- **Name:** appendChild
    - **Work:** Adds an element as a child of another.
    - **Where:** Used like parent.appendChild(child).
    - **Why:** To insert elements into the DOM.
    - **Example:** document.body.appendChild(span);
    - **Explanation:** Adds the span from above to the <body> so it appears on the page.
- **Name:** innerHTML: The `.innerHTML` property **sets or gets** the HTML content **inside** an element, but it does **not replace the element itself**.

    div.innerHTML = " Hello ";

    It **removes** all existing content inside the `<div>`.

    Then, it **inserts** `<p>Hello</p>` inside the `<div>`.

    The `<div>` itself **remains** in the DOM.

    - **Work:** Sets or gets HTML content inside an element.
    - **Where:** Used like element.innerHTML = "Bold".
    - **Why:** To add or replace content with tags.
    - **Example:** div.innerHTML = " Hello ";
    - **Explanation:** Replaces div's content with a tag saying "Hello". Be cautious—it can overwrite existing content.
- **Name:** textContent
    - **Work:** Sets or gets plain text content.
    - **Where:** Used like element.textContent = "Hi".

    - **Why:** Safer for text-only updates.
    - **Example:** span.textContent = "World";
    - **Explanation:** Sets the span's text to "World" without interpreting it as HTML.
- **Name:** setAttribute
    - **Work:** Adds or updates an attribute on an element.
    - **Where:** Used like element.setAttribute("data-info", "123").
    - **Why:** To add IDs, classes, or custom attributes.
    - **Example:** button.setAttribute("id", "btn1");
    - **Explanation:** Gives the button an id of "btn1" so you can select it later.

---

5. Adding CSS Through JS

- **Name:** style
    - **Work:** Sets inline CSS properties.
    - **Where:** Used like element.style.property = "value".
    - **Why:** For quick style changes.
    - **Example:** div.style.backgroundColor = "lightblue";
    - **Explanation:** Changes the div's background to light blue. Use camelCase for properties (e.g., fontSize).
- **Name:** style.cssText
    - **Work:** Sets multiple styles as a string.
    - **Where:** Used like element.style.cssText = "color: red; font-size: 16px;".
    - **Why:** To apply several styles at once.
    - **Example:** div.style.cssText = "border: 1px solid black; padding: 10px;".
    - **Explanation:** Adds a border and padding to div in one line, like inline CSS.