# DOM and Event Revision

## 1. DOM Basics

The Document Object Model (DOM) is a tree representation of HTML/XML documents. JavaScript interacts with the DOM through browser-provided APIs to read and modify content, structure, and styles.

Core Concepts:

* • Nodes (base interface) and Element nodes (tags)
* • document.createElement, querySelector, append, remove

## 2. Selectors

Methods to find elements:

* • getElementById(): fast, single element
* • getElementsByClassName()/TagName(): live HTMLCollection, multiple elements
* • querySelector()/All(): CSS selectors, static NodeList, multiple elements

## 3. Content Manipulation

* • innerHTML / outerHTML: parse and set HTML markup (XSS risk, performance)
* • textContent: raw text, includes hidden, fast
* • innerText: visible text only, respects CSS, reflow cost

## 4. Traversal & Relationships

Access related elements:

* • parentNode / parentElement
* • children (HTMLCollection) / childNodes (NodeList)
* • firstElementChild / nextElementSibling

## 5. Creating & Removing Elements

* • document.createElement, createTextNode
* • insert: append, prepend, insertBefore, insertAdjacentHTML
* • remove(): modern, parent.removeChild() for compatibility
* • DocumentFragment: batch inserts for performance

## 6. Attributes & Properties

* • getAttribute, setAttribute, removeAttribute
* • dataset for data-\* attributes
* • element.property (src, id) for standard props

## 7. Styles

* • element.style.property for inline styles
* • classList.add/remove/toggle for CSS classes
* • CSS variables via style.setProperty('--var', value)

## 8. Events Fundamentals

* Event object properties:
* • type, target, currentTarget, bubbles, cancelable
* • preventDefault(), stopPropagation()
* Event phases: capture, target, bubble

## 9. Common Event Types

* • Mouse: click, dblclick, mousedown, mouseup, mousemove
* • Keyboard: keydown, keyup
* • Pointer: pointerdown, pointerup, pointermove
* • Form: submit, reset, input, change
* • Others: scroll, resize, touch events, custom events

## 10. Event Handling Patterns

* • addEventListener with options: capture, once, passive
* • removeEventListener requires same handler and options
* • Event delegation: attach listener to parent, use event.target
* • simulate events: element.click(), dispatchEvent(new Event())

## 11. FormData

* • Create with new FormData(form) or manually append
* • Methods: append, get, getAll, set, delete, entries
* • Use with fetch(): fetch(url, { method: 'POST', body: formData })