Advanced JavaScript – Course Review (9 Sessions)

> A concise reference and recap for everyone who attended the live sessions **and** anyone catching up afterward. Each section lists the big ideas first, then shows a tiny, self-contained code snippet to jog your memory.

Session 1 – Variables & Basics

Key ideas

1. `var` vs `let` vs `const` (scope, redeclare, reassign)
2. Primitive types + `null`
3. Basic string / number operations
4. Intro to arrays and indexing

```
// var: redeclare + reassign
var x = 10;
var x = 20; // ☑ no error
x = 30;

// let: reassign only
let y = 10;
y = 20;      // ☑

// const: read-only
const z = 10; // ❌ cannot redeclare / reassign

// Array access starts at index 0
const arr = [1, 2, 3];
console.log(arr[2]); // 3
```

Session 2 – Conditionals & Loops

Key ideas

1. `switch` for cleaner multi-branch logic
2. `while`, `for`, `break`, `continue`
3. Loop exercises (multiplication table)
4. Variable scope recap

```
switch (day) {
  case 'Friday':
    console.log('Weekend!');
    break;
  default:
```

```
    console.log('Just another day');
}

for (let i = 1; i <= 10; i++) {
  if (i === 5) continue; // skip 5
  if (i === 8) break;    // stop at 8
  console.log(i);
}
```

Session 3 – DOM Selectors, Functions & Objects

Key ideas

1. `getElementById`, `querySelector`
2. Real-world function: form validation
3. Object syntax + `this`
4. Saving / reading objects with `localStorage`

```
// Select and read text
const box = document.querySelector('.box');

// Simple validator
function isValid(username) {
  return username.trim() !== '';
}

// Persist data
localStorage.setItem('user', JSON.stringify({ name: 'Marwan' }));
```

Session 4 – DOM Queries & Dynamic Elements

Key ideas

1. `querySelectorAll` + `forEach`
2. `document.createElement`, `appendChild`, `insertBefore`
3. Preview of event handling

```
// Style every product > $100
const cards = document.querySelectorAll('.product-card');
cards.forEach(c => {
  const price = +c.querySelector('.price').textContent.slice(1);
  if (price > 100) c.classList.add('premium');
});
```

Session 5 – Building a Header via JavaScript

Key ideas

1. Translate static HTML/CSS into JS-generated DOM
2. Parent ↔ child relationships, hierarchy diagram
3. Re-usable pattern for dynamic page sections

```
const header = document.createElement('header');
const h1 = document.createElement('h1');
h1.textContent = 'Essential Science';
header.appendChild(h1);
document.body.prepend(header);
```

Session 6 – Images, `classList` & Attributes

Key ideas

1. Create/insert `<img>` on demand
2. `classList.add / remove / toggle / contains`
3. `getAttribute` / `setAttribute` / `removeAttribute`

```
document.getElementById('showImgBtn').onclick = () => {
  const img = document.createElement('img');
  img.src = 'https://via.placeholder.com/150';
  document.getElementById('imgContainer').appendChild(img);
};
```

Session 7 – Functions & Event Essentials

Key ideas

1. Function declaration vs expression
2. Mastering `classList` operations
3. Attaching events with `addEventListener`
4. Quick reference of common DOM events

```
// Toggle highlight class on click
btn.addEventListener('click', () => {
  box.classList.toggle('highlight');
});
```

Session 8 – Using CSS CDNs (Bootstrap / Font Awesome)

Key ideas

1. What a CDN is & pros/cons
2. Adding CDN links statically *or* via JS
3. Toggling Bootstrap / FA classes with `classList`

```
<!-- Bootstrap via CDN -->
<link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"
rel="stylesheet" />
```

```
// Dynamically inject Font Awesome if absent
if (!document.querySelector('link[href*="font-awesome"]')) {
  const fa = document.createElement('link');
  fa.rel = 'stylesheet';
  fa.href = 'https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/5.15.4/css/all.min.css';
  document.head.appendChild(fa);
}
```

Session 9 – Bootstrap Utilities & Components

Key ideas

1. Understanding Bootstrap naming (`btn`, `bg-*`, `m-*`, responsive prefixes)
2. Manipulating Bootstrap classes via JS
3. Creating components (alerts, modals) in code

```
// Create success alert dynamically
alertContainer.innerHTML = `
  <div class="alert alert-success alert-dismissible fade show" role="alert">
    Saved!
    <button type="button" class="btn-close" data-bs-dismiss="alert"></button>
  </div>`;
```

Big-Picture Takeaways

• **DOM is code** – build, style, and wire elements with JavaScript alone. • `classList` **is your Swiss-army knife** for visual state changes. • **CDNs & Bootstrap** provide instant UI building blocks. • **Break work into small, reusable functions** to stay organized.

Happy coding! Revisit the session folders for full demos and practice tasks.