***Below is a “toolbox” of HTML elements, attributes, and event hooks that map directly to features in the Paramedic app.***

*> Source reference for all tags/attributes: W3Schools HTML Reference (<https://www.w3schools.com/tags/default.asp>).*

*> All explanations are paraphrased; feel free to click any tag on W3Schools for deeper docs.*

**## 1. Layout & Page Structure**

**<header>**  `html **<header>**<h1>Paramedic Guide</h1>**</header>** ` Fixed title bar at the top of the main contents page.

**<nav>**  `html **<nav** id="mainNav"**></nav>** ` Holds back/forward/home buttons; semantic for accessibility.

**<main>**  `html **<main** id="contentArea"**></main>** ` Primary container for topic lists and focused sections.

**<section>**  `html **<section** class="topic-group"**></section>** ` Groups each major topic or sub‑topic list.

**<article>**  `html **<article** class="topic-detail"**></article>** ` Dedicated area for a single focused subtopic (text, images, etc.).

**<aside>** `html **<aside** id="patientInput" hidden**></aside>** ` Slide‑out panel for user‑entered data (auto‑collapses when hidden).

**<footer>**  `html **<footer>**&copy; 2025 EMS Toolkit**</footer>** ` Optional bottom bar or legal info.

**## 2. Lists, Buttons & Collapsible Items**

**<ul>** / **<li>** `html **<ul><li** class="topic"**>**Airway**</li></ul>** ` Base structure for the expandable topic tree.

**<button>** `html **<button** class="arrow-btn"**>**&#x25B6;**</button>`** Arrow icons (▶ / ▼) that rotate via CSS for expand/collapse.

**<a>** `html **<a** href="#shock"**>**Shock Mgmt**</a>**  Clickable topic titles that can also expand/collapse.

**<details>**  / **<summary>** `html **<details><summary>**IV Access**</summary>**<p>Equipment list…</p>**</details>** Native expand/collapse without extra JS; arrow auto‑rotates.

**<span>** `html **<span** class="badge warning"**>**!**</span>**  Inline icons, warnings, strikethrough markers, etc.

**<svg>**  `html **<svg** class="arrow" viewBox="0 0 24 24"**>**<path d="M8 5l8 7-8 7"/>**</svg>`**  Scalable arrow that you can rotate with CSS.

**## 3. Search & Autocomplete**

**<input type="search">** `html **<input type="search"** id="searchBox" placeholder="Search…"**>**` Built‑in search semantics; triggers `oninput` for live filtering.

**<datalist>** `html **<datalist** id="topicHints"**>**<option value="Airway">**</datalist>** Provides dropdown suggestions as the user types.

**## 4. User Data Entry & Dialogs**

**<form>** `html <form id="patientForm"></form>` Wraps user‑input fields for vitals, meds, etc.

**<input>** `html <input type="number" id="age">` Text, number, checkbox, etc.—capturing user data.

**<textarea>** `html <textarea id="notes"></textarea>` Multi‑line free‑text notes.

**<dialog>** `html <dialog id="confirmDlg">Saved!</dialog>` Optional modal for confirmations without external libraries.

**## 5. Media & Rich Content**

**<img>** `html <img src="airway.png" alt="Airway diagram">` Inline images in topic details.

**<video>** `html <video controls src="intubation.mp4"></video>` Training videos inside subtopic pages.

**<figure>** / **<figcaption>** `html <figure><img src="ecg.png"><figcaption>12‑lead ECG</figcaption></figure>` Keeps images + captions grouped semantically.

**## 6. Text Formatting & Visibility**

**<strong>** /  **<em>**  `html **<strong>**Critical**</strong>**` Emphasize vital info.

**<s>** (strikethrough) `html **<s>**Expired Drug**</s>**` Show text that is no longer appropriate (based on user data).

**hidden** attribute `html <div **hidden**>Secret</div>` Quick hide/show without removing from DOM.

**data-** attributes `html <li **data-**topic-id="shock">Shock</li>` Store IDs or state flags directly on elements for JS.

**## 7. Accessibility & State**

**aria-expanded**  `html <button **aria-expanded**="false"></button>` Screen‑reader cue for collapsible arrows.

**aria-controls** `html <button **aria-controls**="sublist1"></button>` Links arrow button to the list it toggles.

**Tabindex** `html <li **tabindex**="0">Topic</li>` Keyboard navigation across lists.

**## 8. Navigation & History (JS Events)**

**onclick**  `arrowBtn.**onclick** = rotateArrow;` Expand/collapse, open topics, back/home buttons.

**oninput** `searchBox.**oninput** = filterList;` Live search suggestions.

**onchange** `form.**onchange** = recalcFormulas;` Re‑calculate meds/doses when user edits data.

**onpopstate** `window.**onpopstate** = restoreView;` Handle browser‑style back/forward navigation.

**onstorage** `window.**onstorage** = syncAcrossTabs;` Sync localStorage data if app is opened twice.

**## 9. Persistent Data (JS, not HTML but crucial)**

**localStorage** /  **sessionStorage** store user‑entered vitals, medication lists, and history so they persist between sessions.

```javascript

**localStorage**.setItem('patientAge', ageInput.value);

```

**history.pushState()** /  **history.back()** / **history.forward()** drive custom back/forward buttons while keeping highlight of the last clicked item.

**## 10. Putting It Together – Mini‑Template**

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Paramedic Guide</title>

<link rel="stylesheet" href="css/styles.css">

</head>

<body>

<header>

<button id="menuToggle" aria-expanded="false">&#9776;</button>

<h1>Paramedic Guide</h1>

<nav>

<button id="backBtn" disabled>&larr;</button>

<button id="fwdBtn" disabled>&rarr;</button>

<button id="homeBtn">&#8962;</button>

</nav>

</header>

<aside id="patientInput" hidden>

<form id="patientForm">

<label>Age <input type="number" id="age"></label>

<label>Weight (kg) <input type="number" id="weight"></label>

<button type="submit">Save</button>

</form>

</aside>

<main>

<input type="search" id="searchBox" placeholder="Search topics…" list="topicHints">

<datalist id="topicHints"></datalist>

<section id="topics">

<!-- dynamically built UL/LIs with arrow buttons -->

</section>

<article id="detailView" hidden>

<!-- focused subtopic content -->

</article>

</main>

<script src="js/app.js"></script>

</body>

</html>

```

**### How to Use This List**

**1.** \*\*Start with the template\*\* and break it into `index.html`, `css/`, and `js/` folders as you planned.

**2.** \*\*Copy‑paste\*\* any of the snippets above where they fit (lists, details, search bar, etc.).

**3.** \*\*Wire up JavaScript\*\* to rotate arrows, toggle `hidden`, update `aria-expanded`, and manage history/localStorage.

**4.** \*\*Remove duplicates\*\* – all items above are unique and directly map to your requirements.

**CSS**, grouped by category.

All definitions were cross‑checked in the W3Schools CSS Reference (https://www.w3schools.com/cssref/index.php).

**### 1. Layout & Box‑Model**

**display:flex** `css .toolbar{**display:flex**;justify-content:space-between;} `

Puts navigation buttons (Back / Forward / Home) on one line and spaces them automatically.

**flex-direction** `css .topics{**flex-direction**:column;} ` Stacks topic buttons vertically.

**flex-wrap** `css .history{**flex-wrap**:wrap;} ` Allows long history lists to wrap onto new lines.

**gap** `css .toolbar{**gap**:0.5rem;} ` Clean spacing between icons without extra margins.

**order** `css .homeBtn{**order**:3;} ` Re‑orders flex items (e.g., keep Home under Back/Forward).

**display:grid** `css .subtopics{**display:grid**;grid-template-columns:1fr 1fr;} ` Two‑column subtopic panels.

**grid-template-columns** | see above | Defines the column structure.

**grid-gap** `css .subtopics{**grid-gap**:1rem;} ` Space between grid cells.

**position:fixed** `css .toolbar{**position:fixed**;top:0;right:0;} ` Keeps navigation bar visible at all times.

**position:sticky** `css h1{**position:sticky**;top:0;} ` Keeps the page title stuck under the toolbar when scrolling.

**z-index** `css .overlay{**z-index**:999;} ` Ensures the data‑entry overlay sits above all other content.

**overflow-y:auto** `css .history{max-height:60vh;**overflow-y:auto**;} ` Scrollable history panel.

**box-sizing:border-box** `css \*,\*::before,\*::after{**box-sizing:border-box**;} `

Padding & border stay inside declared width/height—simpler sizing.

**margin** /  **padding** Standard spacing around or inside elements.

**min-height** /  **max-height** Controls collapsible panel heights during animation.

**### 2. Visual Styling**

**color** & **background-color** `css .warning{**color**:#fff;**background**:#e53935;} ` Red warning banners.

**background-image** `css .arrow{**background:url(arrow.svg)** center/contain no-repeat;} ` Blue arrow icon without extra `<img>`.

**border-radius** `css .button{border-radius:6px;} ` Rounded topic buttons.

**box-shadow** `css .panel{box-shadow:0 2px 6px rgba(0,0,0,.15);} ` Subtle card effect for panels.

**opacity** `css .faded{opacity:.4;pointer-events:none;} ` Fades topics that no longer apply.

**filter** `css .blurred{filter:blur(2px);} ` Optional visual emphasis/de‑emphasis.

**text-decoration:line-through** `css .struck{**text-decoration:line-through**;} ` Strike‑through outdated treatments.

**cursor:pointer** `css button{**cursor:pointer**;} ` Shows clickable areas.

**accent-color** `css input[type="checkbox"]{**accent-color**:#1976d2;} ` Makes form controls match the app’s blue.

**### 3. Typography**

**font-family** `css body{font-family:"Segoe UI",Arial,sans-serif;} ` Clean, legible default font.

**font-size** `css h1{font-size:1.5rem;} ` Responsive heading sizes.

**font-weight** `css .bold{font-weight:700;} ` Emphasize critical values.

**font-style**  \***italic, normal**\* for emphasis or notes.

**line-height** `css p{**line-height**:1.4;} ` Improves readability.

**letter-spacing** `css .code{**letter-spacing**:0.05em;} ` Better monospace readability for formulas.

**text-transform:uppercase** `css .tag{**text-transform:uppercase**;} ` Consistent tag appearance.

**### 4. Selectors & Combinators**

**.class** /  **#id** Standard element targeting.

**\***  Global resets (`**\***{margin:0}`)

**[data-state="open"]** `css **[data-state="open"]**>.arrow{transform:rotate(90deg);} ` Rotates arrow when parent is open.

**element element** (descendant) Style nested lists.

**element>element`** (child) Prevents deep list styling leaks.

**element+element`** (adjacent) Add margin only between consecutive items.

**:hover** Highlight last‑clicked item on mouseover.

**:focus**  / **:focus-visible** | Keyboard accessibility.

**:active** Immediate press feedback on buttons.

**:checked**` | ```css input:checked + .arrow{transform:rotate(90deg);} ``` | Pure‑CSS collapsible lists with hidden checkboxes.

**:disabled**` | Gray out Forward button until usable.

**:not()`** | ```css .topic:not(.faded){cursor:pointer;} ``` | Target everything except faded items.

**:nth-child(n)` / `:nth-of-type()`** | Zebra striping or custom ordering.

**:has()`\*** | ```css .topic**:has(**.warning**)**{border-left:4px solid #e53935;} ` Parent styling when it contains a warning (supported in modern browsers).

**### 5. Pseudo‑elements**

**::before** /  **::after** `css .topic**::before**{content:"►";color:#1976d2;} ``` | Creates arrow bullets without extra HTML.

**::placeholder** `css input**::placeholder**{color:#888;} ``` | Styles search‑bar hint text.

**::selection** `css **::selection**{background:#1976d2;color:#fff;} ``` | Branded text selection color.

**### 6. Animation & Transition**

**transition`** | ```css .arrow{**transition**:transform .25s ease;} ``` | Smooth arrow rotation.

**transition-delay**` | Adds staggered list reveals.

**transform:rotate()`** | Rotates arrow 90° down.

**transform:translateY()`** | Slide‑down panels.

**@keyframes**` | ```css **@keyframes** fadeIn{from{opacity:0}to{opacity:1}} ``` | Custom fade or slide animations.

**animation`** | ```css .panel{**animation**:fadeIn .3s;} ``` | Runs the keyframes.

**animation-fill-mode:forwards**` | Keeps final animation state (e.g., open panel stays visible).

**### 7. Responsive & Conditional Rules**

**@media`** | ```css @media (max-width:600px){.toolbar{flex-direction:column;}} ``` | Mobile‑first layout tweaks.

**@supports**` | ```css @supports (backdrop-filter:blur(4px)){.overlay{backdrop-filter:blur(4px);}} ``` | Progressive enhancement.

**@import**` | Pulls in shared component styles.

**@font-face**` | Self‑hosted fonts for offline use in the field.

**### 8. CSS Functions & Variables**

**calc()`** | ```css .panel{max-height:**calc(**100vh - 4rem**)**;} ``` | Sizes panels minus toolbar height.

**clamp()`** | ```css h1{font-size:**clamp(**1.2rem,2vw+1rem,2rem**)**;} ``` | Fluid typography.

**var()`** | ```css .button{background:**var(**--primary**)**;} ``` | Central theme colors.

**url()** Loads SVGs or images in `background-image`.

**min()** / **max()** Advanced responsive math.

**### 9. Units & Measurements**

**px**  Pixel‑perfect icons.

**rem** Scales with root font for accessibility.

**em**  Relative sizing inside components.

**%**  Flexible widths/heights.

**vh** /  **vw** Full‑screen panels (e.g., history list `height:60**vh**`).

**### 10. Color Values**

**Hex (`#1976d2`)** Brand blue arrow / buttons.

**rgb()**  / **rgba()**  `css background:rgba(25,118,210,.1); ` translucent overlays.

**hsl()** / **hsla()** Easier hue adjustments for warning shades.

**## One‑sentence source summary**

**CSS** Reference pages catalog every property, selector, pseudo‑class, pseudo‑element, at‑rule, function, unit, and color value with syntax, description, and browser support, making it a handy checklist while you build the app.

(Source: https://www.w3schools.com/cssref/index.php)

List of **JavaScript** properties, methods, and objects from the W3Schools Alphabetical JavaScript Reference.

(plain HTML + JS, no modules, ready to drop into separate files)

**## 1. DOM Selection & Creation**

**document.getElementById()** Fast way to grab one element (e.g., main title, search bar). ```js const searchBar = document.getElementById('search'); ```

**document.querySelector()**  Select the first match with CSS‑style selectors (e.g., “.arrow.right”). ```js const firstArrow = document.querySelector('.arrow.right'); ```

**document.querySelectorAll()**  Select \*all\* matches (e.g., all arrows that must rotate). ```js const arrows = document.querySelectorAll('.arrow'); ```

**document.createElement()** Build buttons, list items, warning banners, etc., on the fly. ```js const btn = document.createElement('button'); btn.textContent = 'New Topic'; ```

**Element.append()**  Append nodes or strings (handy for building subtopic lists). ```js topicItem.append(subList); ```

**Element.appendChild()**  Classic way to add a node as last child. ```js list.appendChild(li); ```

**Element.insertBefore()**  Insert nodes at specific positions (e.g., highlight insertion). ```js list.insertBefore(li, list.firstChild); ```

**Element.remove()**  Remove an element (e.g., collapse a sublist). ```js subList.remove(); ```

**Node.cloneNode()**  Duplicate templates for repeated UI pieces. ```js const copy = template.cloneNode(true); ```

**## 2. Class & Style Manipulation**

**`Element.classList.add()`** / **`remove()`** / **`toggle()`** / **`contains()`** Show/hide, rotate arrows, strikethrough, fade, highlight. `js **arrow.classList.**toggle**(**'rotate-down'**)**; `

**`Element.style`**  Inline style tweaks (e.g., fade opacity). `js btn**.style**.opacity = 0.4; `

**`getComputedStyle()`** Read current style (e.g., to decide if list is open). `js const isOpen = **getComputedStyle(**list**)**.display !== 'none'; `

**## 3. Event Handling**

**`addEventListener()`\*\*** | Attach click, input, keydown, etc. ```js arrow.addEventListener('click', toggleSection); ```

**`removeEventListener()`**  Detach when no longer needed (memory). ```js arrow.removeEventListener('click', toggleSection); ```

**`Event.target`**  Know which element was clicked (for delegation). ```js list.addEventListener('click', e => { if (e.target.matches('.arrow')) { … } }); ```

**`Event.preventDefault()`**  Stop default link scroll when using custom nav. ```js link.addEventListener('click', e => { e.preventDefault(); … }); ```

**`Event.stopPropagation()`**  Avoid bubbling when nested arrows. ```js e.stopPropagation(); ```

**`keydown`** /  **`keyup` events**  Detect typing in search bar. `js search.addEventListener(**'keyup'**, filterTopics); `

**`input` event** Real‑time search suggestions. ```js search.addEventListener('input', showSuggestions); ```

**`MouseEvent.altKey/ctrlKey/shiftKey`**  Optional keyboard shortcuts for medics. ```js if (e.ctrlKey && e.key === 'F') openSearch(); ```

**## 4. Storage & Persistence**

**`localStorage.setItem()`** / **`getItem()`** / **`removeItem()`** / **`clear()`**  Persist user‑entered data, history, preferences across sessions.

`js **localStorage.setItem(**'patientData', JSON.stringify(data)**)**; const stored = JSON.parse(**localStorage.getItem(**'patientData')**)**; `

**`sessionStorage`**  Short‑lived data (one app session). ```js sessionStorage.setItem('lastView', 'topic-abc'); ```

**`JSON.stringify()`** / **`JSON.parse()`**  Convert objects to strings for storage and back. ```js const json = JSON.stringify(obj); const obj2 = JSON.parse(json); ```

**## 5. Navigation & History**

**`history.back()`** / **`history.forward()`** Implement the always‑visible back/forward buttons. ```js backBtn.addEventListener('click', () => history.back()); ```

**`history.pushState()`** / **`popstate` event** Maintain highlight of last clicked item, deep‑link subtopics.

```js history.pushState({id}, '', '#'+id); window.addEventListener('popstate', e => highlight(e.state.id)); ```

**`location.hash`**  Simple anchor‑style navigation to subtopics. ```js location.hash = 'topic-123'; ```

**## 6. Timers & Animation Helpers**

**`setTimeout()`** / **`clearTimeout()`**  Delay warnings, auto‑collapse input box after X ms. ```js const t = setTimeout(closeBox, 5000); ```

**`setInterval()`** / **`clearInterval()`** Re‑check data, blink alert, or refresh timers. ```js const id = setInterval(updateClock, 1000); ```

**`requestAnimationFrame()`** Smooth arrow rotation or fade animations. ```js function step(ts){ … requestAnimationFrame(step);} requestAnimationFrame(step); ```

**## 7. Array & Collection Utilities**

**`Array.forEach()`**  Loop through topics to build DOM. ```js topics.forEach(t => makeButton(t)); ```

**`Array.map()`**  Transform arrays (e.g., to suggestion strings). ```js const names = topics.map(t => t.title); ```

**`Array.filter()`** Live search filtering. ```js const matches = topics.filter(t => t.title.includes(q)); ```

**`Array.find()`** Get first match (e.g., current topic). ```js const topic = topics.find(t => t.id === id); ```

**`Array.some()`** / **`every()`** Check if any / all subtopics done. ```js if (subs.some(s => s.done)) … ```

**`Array.push()`** / **`pop()`** / **`splice()`** / **`slice()`**  Manage history stack manually if desired. ```js historyStack.push(id); ```

**`Set`\*\* (`add`, `has`, `delete`)** Track unique visited topics. ```js const visited = new Set(); visited.add(id); ```

**`Map`\*\*** (**`set`**, **`get`)**  Key/value storage for topic meta. ```js const meta = new **Map**(); meta.**set(**id, {seen:true}**)**; ```

**## 8. String & RegExp Helpers**

**`String.includes()`** / **`startsWith()`** / **`toLowerCase()`** Case‑insensitive search in the search bar. | ```js if (title.toLowerCase().includes(query)) … ```

**`String.replace()`** Insert highlighted spans in search results. ```js result.innerHTML = title.replace(re, '<mark>$&</mark>'); ```

**`RegExp`** Build dynamic search patterns. ```js const re = new RegExp(query, 'i'); ```

**## 9. Fetching Resources**

**`fetch()`** Load external images/videos lists, update drug databases. ```js fetch('data/topics.json').then(r => r.json()).then(data => build(data)); ```

**`Promise.then()`** / **`catch()`** / **`finally()`**  Handle async results from `fetch` or custom calculations. ```js fetch(url).then(handle).catch(showError); ```

**`async`** / **`await`**  Cleaner syntax for async flows. ```js async function load(){ const res = await fetch(url); return res.json(); } ```

**## 10. Math & Number Helpers**

**`Math.round()`** / **`floor()`** / **`ceil()`** / **`random()`** Drug dose rounding, random case drills. ```js const dose = Math.round(weight \* mgPerKg); ```

**`Number.parseFloat()`** / **`parseInt()`** / **`toFixed()`**  Convert & format numeric inputs. ```js const kg = parseFloat(input.value).toFixed(1); ```

**## 11. Date & Time**

**`Date()` constructor** / **`Date.now()`**  Timestamp user actions, show last‑updated. ```js const ts = Date.now(); ```

**`Date.toLocaleString()`**  Display dates in user’s locale (en‑US). ```js date.toLocaleString('en-US'); ```

**## 12. Miscellaneous Window & Navigator**

**`window.scrollTo()`** / **`Element.scrollIntoView()`** Auto‑scroll to highlighted item after back nav. ```js activeItem.scrollIntoView({behavior:'smooth'}); ```

**`navigator.onLine`**  Warn if offline (important in field). ```js if(!navigator.onLine) showOfflineBanner(); ```

**### Putting It Together (Mini Demo Snippet)**

```html

<!-- index.html -->

<ul id="topics"></ul>

<input id="search" placeholder="Search…" />

<script src="app.js"></script>

```

```js

// app.js

const topics = [

{id:'airway', title:'Airway Management', subs:['Intubation','Suction']},

{id:'cpr', title:'CPR', subs:['Adult','Pediatric']}

];

const ul = document.getElementById('topics');

const search = document.getElementById('search');

function buildTopics(list){

ul.innerHTML = ''; // clear

list.forEach(t => {

const li = document.createElement('li');

li.innerHTML = `<span class="arrow right"></span> ${t.title}`;

li.dataset.id = t.id;

li.classList.add('topic');

ul.append(li);

});

}

function toggleSection(e){

const li = e.target.closest('.topic');

if(!li) return;

li.classList.toggle('open'); // rotates arrow via CSS

history.pushState({id:li.dataset.id}, '', '#'+li.dataset.id);

}

function filterTopics(){

const q = search.value.toLowerCase();

const filtered = topics.filter(t => t.title.toLowerCase().includes(q));

buildTopics(filtered);

}

ul.addEventListener('click', toggleSection);

search.addEventListener('input', filterTopics);

buildTopics(topics); // initial render

```

**### How to Organize Files**

**/css**

**styles.css**

**/js**

**app.js**

**storage.js**

**nav.js**

**index.html**

\*Each file can import functions by adding `**<script src="**…**"></script>**`in HTML (no ES modules).\*

---

These references should cover: collapsible arrows, dynamic search, user‑entered data persistence, back/forward navigation, strikethrough/fade, auto‑calculations, and more.