m3.js

m3.js

May 25, 2026

CONTENTS:

CHAPTER

ONE

INSTALLATION

m3.js and m3.css are intentionally distributed as single flat files. There is no package manager, no bundler, and no build pipeline involved. Just two files — thats all you need.

On this page

1.1 Option 1 — Download & Self-host (Recommended)

Download both files and place them in your project's static assets folder.

```
your-project/
|- index.html
|- m3.css
‘- m3.js
```

Then reference them in your HTML <head>:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>My App</title>

  <!-- 1. Material Design 3 styles -->
```

```
  <link rel="stylesheet" href="m3.css">
</head>
<body>

  <!-- your page content here -->

  <!-- 2. Material Design 3 behaviour - load last -->
  <script src="m3.js"></script>
</body>
</html>
```

Important:

Always load `m3.js` **after** the rest of your HTML body so that the DOM is fully parsed before the script tries to wire up components. Loading it at the bottom of `<body>` (not in `<head>`) is the safest approach.

1.2 Option 2 — CDN (Quick Prototyping)

If you just want to experiment, you can load the files directly from a CDN such as jsDelivr or unpkg by pointing the `src` / `href` at the raw GitHub URL.

```
<!-- Replace 'main' with a specific commit SHA for stability -->
<link rel="stylesheet"
      href="https://cdn.jsdelivr.net/gh/your-org/m3@main/m3.css">
<script
      src="https://cdn.jsdelivr.net/gh/your-org/m3@main/m3.js"></script>
```

Warning:

CDN links pinned to `main` will always fetch the latest version, which may include breaking changes. Pin to a specific tag or commit SHA in production enviroments.

1.3 Browser Support

m3.css and m3.js use modern web platform features. The table below shows the minimum browser version that supports each major feature.

Feature	Chrome / Edge	Firefox	Safari
CSS custom properties (tokens)	49+	31+	9.1+
Cache API (offline)	40+	39+	11.1+
Web App Manifest	39+	85+	15.4+
View Transitions API (page transitions)	111+	*(partial)*	18+
`color-mix()`	111+	113+	16.2+

All features degrade gracefully in older browsers — the page still renders correctly, it just wont have the enhanced behaviour (e.g. page transitions are skipped silently on browsers that don't support `document.startViewTransition`).

1.4 File Sizes

Both files are designed to be small enough that you don't need to worry about bundling.

File	Uncompressed	Gzip
`m3.css`	~18 KB	~4 KB
`m3.js`	~12 KB	~3.5 KB

Tip:

Even self-hosted, modern servers will automatically serve these files with gzip or Brotli compression, so real-world download sizes will be around those *Gzip* figures.

1.5 Next Steps

Head over to the Quick-Start Guide guide to build your first page.

CHAPTER

TWO

QUICK-START GUIDE

This page walks you through building a complete, Material You–styled page from scratch. By the end you'll have a working app with a top bar, navigation drawer, cards, buttons, a dialog, and a snackbar — all with zero JavaScript written by you.

On this page

2.1 Step 1 — Boilerplate

Start with this minimal template. The `<meta>` tags let m3.js know your app's name and accent colour (see ../api/configuration for the full list).

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="m3-app-name"   content="My First App">
  <meta name="m3-app-short"  content="MyApp">
  <meta name="m3-theme-color" content="#6750A4">
```

```
  <title>My First App</title>
  <link rel="stylesheet" href="m3.css">
</head>
<body>
  <!-- content goes here -->
  <script src="m3.js"></script>
</body>
</html>
```

Save this as `index.html` and open it in a browser. You'll see a plain white (or dark, if your OS uses dark mode) page — the canvas is ready.

2.2 Step 2 — Add a Top App Bar

The `<header>` element maps directly to the M3 *Top App Bar* component.

```
<header>
  <!-- Hamburger menu button (opens the nav drawer) -->
  <button class="icon" data-m3-drawer-toggle aria-label="Open navigation">
    [menu]
  </button>

  <!-- App title -->
  <span class="title-large">My First App</span>

  <!-- Spacer pushes action buttons to the right -->
  <span class="flex-1"></span>

  <!-- Theme toggle (light ↔ dark) -->
  <button class="icon" data-m3-theme-toggle aria-label="Toggle theme">
    (theme)
  </button>
</header>
```

The `data-m3-drawer-toggle` and `data-m3-theme-toggle` attributes are magic hooks that m3.js listens for. No additional JavaScript is needed.

2.3 Step 3 — Add a Navigation Drawer

Place a `<nav>` element immediately after the `<header>`. m3.js will wire up the open/close behaviour and the overlay backdrop automatically.

```
<nav>
  <a href="/" class="active">[home] Home</a>
  <a href="/about">i  About</a>
  <a href="/settings">[gear]  Settings</a>
</nav>
```

Links whose `href` matches the current URL are given the `active` class automatically at page load (via `initActiveLinks` in m3.js).

2.4 Step 4 — Page Content with Cards

Wrap your main content in a `<main>` block. Use `<article>` for filled cards and `<aside>` for outlined cards.

```
<main>
  <h1>Welcome</h1>
  <p>This is your first Material You page.</p>

  <!-- Filled card -->
  <article>
    <h3>Getting Started</h3>
    <p>Drop m3.css and m3.js into any HTML page and you're done.</p>
    <div style="display:flex; gap:8px; margin-top:16px;">
      <button>Primary Action</button>
      <button class="outlined">Secondary</button>
      <button class="text">Learn More</button>
    </div>
  </article>

  <!-- Outlined card -->
  <aside>
    <h4>Pro Tip</h4>
    <p>Use <code>&lt;aside&gt;</code> for less prominent card content.</p>
  </aside>
</main>
```

2.5 Step 5 — Dialogs

Dialogs use the native HTML `<dialog>` element. m3.js handles opening, closing, and the backdrop overlay for you.

```
<!-- Trigger button -->
<button data-m3-dialog-open="my-dialog">Open Dialog</button>

<!-- The dialog itself (place anywhere in <body>) -->
<dialog id="my-dialog">
  <h3>Confirm action</h3>
  <p>Are you sure you want to proceed?</p>
  <div style="display:flex; gap:8px; justify-content:flex-end; margin-top:16px;">
    <button class="text" data-m3-dialog-close>Cancel</button>
    <button>Confirm</button>
  </div>
</dialog>
```

- `data-m3-dialog-open="<id>"` — clicking this element opens the dialog with that ID.
- `data-m3-dialog-close` — clicking this element (inside a dialog) closes it.
- Clicking the backdrop also closes the dialog.

2.6 Step 6 — Snackbars (Toasts)

Trigger a snackbar from JavaScript using the global `window.M3.snack()` API.

```
<button onclick="window.M3.snack('Settings saved!', 'Undo', () => console.log('undone'), 4000)">
  Save Settings
</button>
```

The four parameters are:

```
M3.snack(message, actionLabel, actionCallback, durationMs)
```

All parameters after `message` are optional. Passing `null` for `actionLabel` produces a snackbar with no action button.

2.7 Putting It All Together

Here is the complete `index.html` from the steps above:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="m3-app-name"    content="My First App">
  <meta name="m3-app-short"   content="MyApp">
  <meta name="m3-theme-color" content="#6750A4">
  <title>My First App</title>
  <link rel="stylesheet" href="m3.css">
</head>
<body>

  <header>
    <button class="icon" data-m3-drawer-toggle aria-label="Menu">[menu]</button>
    <span class="title-large">My First App</span>
    <span class="flex-1"></span>
    <button class="icon" data-m3-theme-toggle aria-label="Toggle theme">(theme)</button>
  </header>

  <nav>
    <a href="/" class="active">[home] Home</a>
    <a href="/about">i  About</a>
```

```
  </nav>

  <main>
    <article>
      <h3>Hello, Material You</h3>
      <p>Edit this file and refresh to see changes.</p>
      <button onclick="window.M3.snack('It works! [party]')">Show Snackbar</button>
    </article>
  </main>

  <dialog id="my-dialog">
    <h3>Dialog Title</h3>
    <p>Dialog content goes here.</p>
    <button class="text" data-m3-dialog-close>Close</button>
  </dialog>

  <script src="m3.js"></script>
</body>
</html>
```

Open the file locally or serve it with any static file server (e.g. `python -m http.server`) and you should see a fully-functional Material You page.

2.8 Next Steps

- Theming — Customise colours, shapes, and typography.
- Progressive Web App (PWA) — Turn your page into an installable PWA.
- ../api/css-components — Full list of CSS components and modifier classes.

CHAPTER

THREE

THEMING

m3.css is built entirely on CSS custom properties (variables). Overriding the colour scheme, typography, shape scale, or motion tokens is as simple as re-declaring the relevant variable in your own stylesheet — no source edits required.

On this page

3.1 How Theming Works

All design tokens are declared on `:root` inside `m3.css`. When you override them, the cascade ensures your values win everywhere in the document:

```
/* my-theme.css - load this *after* m3.css */
:root {
  --md-primary:           #00639B;   /* teal-ish blue */
  --md-on-primary:        #FFFFFF;
  --md-primary-container: #CEE5FF;
```

```
  --md-on-primary-container: #001D32;
}
```

Load the override file after `m3.css` in your HTML:

```
<link rel="stylesheet" href="m3.css">
<link rel="stylesheet" href="my-theme.css">   <!-- overrides -->
```

3.2 Colour Tokens

The full set of colour tokens and their roles:

Token	Role	Default (light)
`--md-primary`	Main brand colour (buttons, active states)	`#6750A4`
`--md-on-primary`	Text/icons on primary colour	`#FFFFFF`
`--md-primary-container`	Softer primary fill (chips, selected tabs)	`#EADDFF`
`--md-on-primary-container`	Text on primary container	`#21005D`
`--md-secondary`	Supporting brand colour	`#625B71`
`--md-tertiary`	Accent for highlights	`#7D5260`
`--md-error`	Error / destructive state	`#B3261E`
`--md-surface`	Page background	`#FEF7FF`
`--md-on-surface`	Default text colour	`#1C1B1F`
`--md-outline`	Borders, input outlines	`#79747E`
`--md-outline-variant`	Subtle dividers	`#CAC4D0`

Note:

Each colour role has a `--md-*-container` and `--md-on-*` counterpart following the Material You tonal-surface pattern. See the ../api/design-tokens reference for the complete list.

3.3 Dark Theme

m3.css ships with a full dark-theme palette. It is activated in two ways:

1. **System preference** — via `@media (prefers-color-scheme: dark)` (always active).
2. **Manual toggle** — adding the class `dark` (or `data-theme="dark"`) to <html> or <body>.

m3.js wires up any element with `data-m3-theme-toggle` to switch between light and dark and remembers the user's choice in `localStorage`.

```
<!-- This button toggles theme on click -->
<button class="icon" data-m3-theme-toggle aria-label="Toggle theme">(theme)</button>
```

To force dark mode regardless of system preference, add the class to `<html>` directly:

```
<html lang="en" class="dark">
```

3.3.1 Customising Dark Tokens

Override dark-mode tokens inside the `.dark` selector:

```
.dark, [data-theme="dark"] {
  --md-primary: #A78BFA;   /* lighter purple for dark bg */
}
```

3.4 Shape Scale

Corners use five named sizes, all overrideable:

```
:root {
  --md-shape-xs:   4px;   /* chips, snackbars */
  --md-shape-sm:   8px;   /* cards (small) */
  --md-shape-md:   12px;  /* menus */
  --md-shape-lg:   16px;  /* cards, drawers */
  --md-shape-xl:   28px;  /* dialogs, bottom sheets */
  --md-shape-full: 9999px; /* pills, FABs */
}
```

For a squarer look set all values to `4px`; for a fully rounded "pill" style increase `--md-shape-lg` and `--md-shape-xl`.

3.5 Typography

The default font stack is:

```
--md-font:      'Google Sans', 'Segoe UI', Roboto, sans-serif;
--md-font-mono: 'Roboto Mono', 'Fira Code', monospace;
```

Swap in any font family (load the font first via `<link>` or `@import`):

```css
@import url('https://fonts.googleapis.com/css2?family=Inter:wght@400;500;700');

:root {
  --md-font: 'Inter', sans-serif;
}
```

3.6 Motion Tokens

All animations in m3.css reference these shared tokens so you can slow down or speed up the entire UI from one place:

```css
:root {
  --md-dur-short: 150ms;
  --md-dur-med:   250ms;
  --md-dur-long:  400ms;

  --md-ease-std:   cubic-bezier(.2,0,0,1);
  --md-ease-decel: cubic-bezier(0,0,0,1);
  --md-ease-accel: cubic-bezier(.3,0,1,1);
}
```

To honour the user's `prefers-reduced-motion` preference, add this to your stylesheet:

```css
@media (prefers-reduced-motion: reduce) {
  :root {
    --md-dur-short: 0ms;
    --md-dur-med:   0ms;
    --md-dur-long:  0ms;
  }
}
```

3.7 Full Custom Palette Example

Below is an example of a warm-amber theme override:

```css
/* amber-theme.css */
:root {
  --md-primary:            #7C5800;
  --md-on-primary:         #FFFFFF;
  --md-primary-container:  #FFDEA0;
  --md-on-primary-container:#271900;

  --md-secondary:          #6B5D3F;
```

```
  --md-on-secondary:        #FFFFFF;
  --md-secondary-container:#F5DEBC;
  --md-on-secondary-container:#241A04;

  --md-tertiary:            #4D6544;
  --md-on-tertiary:         #FFFFFF;
  --md-tertiary-container: #CFECC3;
  --md-on-tertiary-container:#0B2007;

  --md-surface:             #FFFBFF;
  --md-on-surface:          #1E1B16;
}

.dark, [data-theme="dark"] {
  --md-primary:             #EDBD48;
  --md-on-primary:          #412D00;
  --md-surface:             #1E1B16;
  --md-on-surface:          #E9E2D9;
}
```

3.8 Next Steps

- Progressive Web App (PWA) — Configure the PWA and offline behaviour.
- ../api/design-tokens — Complete token reference.

CHAPTER
FOUR

PROGRESSIVE WEB APP (PWA)

m3.js turns any page into an installable, offline-capable Progressive Web App with zero server configuration. This page explains how the PWA features work and how to customise them.

On this page

4.1 Overview

When m3.js loads, it automatically:

1. Injects PWA-related <meta> tags (viewport, apple-mobile-web-app-capable, etc.).
2. Generates and injects a Web App Manifest via a `data:` URI (no server file needed).
3. Caches the current page and all its linked assets using the **Cache API**.
4. Listens for the `beforeinstallprompt` event and offers an *Install* snackbar after 10 seconds.
5. Shows an *"You are offline"* snackbar when the device loses connectivity.

All these behaviours can be tuned or disabled via <meta> tags in your <head>.

4.2 Configuration

Set any of the following meta tags before `m3.js` loads:

```
<!-- App display name (shown in manifest and install prompt) -->
<meta name="m3-app-name"     content="My Application">

<!-- Short name (≤12 chars recommended, used as icon label) -->
<meta name="m3-app-short"    content="MyApp">

<!-- Primary/accent colour (used for theme-color + manifest) -->
<meta name="m3-theme-color" content="#6750A4">

<!-- Cache bucket name - change this to bust the cache -->
<meta name="m3-cache-name"  content="myapp-v2">

<!-- Set to "false" to disable offline caching entirely -->
<meta name="m3-offline"      content="true">

<!-- Set to "false" to suppress the install-prompt snackbar -->
<meta name="m3-install-prompt" content="true">
```

See ../api/configuration for the full reference.

4.3 How Offline Caching Works

m3.js uses the browser's **Cache API** directly from the main thread — no service worker is required. This is simpler to deploy but has one limitation: the cache is populated *after* the page loads, so the very first visit must be online. Subsequent visits (even offline) will be served from the cache.

4.3.1 What gets cached

On every page load m3.js caches the following same-origin resources:

- The current page URL (`window.location.href`)
- All <link rel="stylesheet"> stylesheets

- All <script src="..."> scripts
- All <img src="..."> images
- All <link rel="icon"> favicons

Cross-origin URLs (e.g. Google Fonts CDN) are excluded to avoid CORS issues.

4.3.2 Cache Naming

The default cache bucket is `m3-cache-v1`. To invalidate the cache after a deployment, change the `m3-cache-name` meta value:

```
<meta name="m3-cache-name" content="myapp-v2">
```

Users who had the old cache will automatically receive fresh files on their next online visit.

4.3.3 Manual Cache Control

Two utilities are exposed on `window.M3.cache`:

```
// Manually clear the entire cache bucket
await window.M3.cache.clear();

// Re-run the cache population (called automatically on boot)
await window.M3.cache.init();
```

4.4 Web App Manifest

m3.js generates and injects a Web App Manifest at runtime using a `Blob` URL, so no `manifest.json` file is needed on the server.

The generated manifest includes:

Field	Value
`name`	`m3-app-name` meta (or `document.title`)
`short_name`	`m3-app-short` meta (or first 12 chars of title)
`start_url`	Current pathname + query string
`display`	`standalone`
`theme_color`	`m3-theme-color` meta
`icons`	Auto-generated letter-icon PNG at 192×192 and 512×512

Tip:

If your project already has a <link rel="manifest"> in the <head>, m3.js will skip manifest injection and respect your existing file.

4.4.1 Using a Custom Manifest

Simply add your own manifest link before m3.js loads:

```
<head>
  <link rel="manifest" href="/manifest.webmanifest">
  <!-- ... -->
</head>
<body>
  <script src="m3.js"></script>  <!-- will skip auto-manifest -->
</body>
```

4.5 Install Prompt

Browsers fire a `beforeinstallprompt` event when the page meets the PWA installability criteria (HTTPS, manifest, served page). m3.js captures this event and, after **10 seconds**, shows a snackbar:

```
Install MyApp on your device    [Install]
```

Clicking *Install* triggers the native browser install dialog. If the user dismisses the snackbar, they can still install the app from the browser menu.

4.5.1 Triggering the Prompt Manually

You can trigger the install prompt from your own UI at any time:

```
await window.M3.installApp();
```

If the install prompt has already been used or is not available (e.g. already installed, or desktop Chrome hasn't fired the event yet), this function is a no-op.

4.5.2 Disabling the Prompt

```
<meta name="m3-install-prompt" content="false">
```

4.6 Offline Indicator

m3.js automatically listens to the browser's `online` / `offline` events:

- When the device goes offline → snackbar: *"You are offline"* (5 s)
- When connectivity returns → snackbar: *"Back online ✓"* (3 s)
- If the page loads while already offline → the offline snackbar is shown immediately.

This behaviour requires no configuration and cannot currently be disabled short of patching the source file. A configuration option is planned for a future release.

4.7 Limitations

Because m3.js does not use a service worker, there are some things it cannot do:

- **Background sync** — queuing network requests for later is not supported.
- **Push notifications** — push requires a service worker registration.
- **Intercepting fetch** — only same-origin assets listed in the DOM at load time are cached; dynamic assets fetched later by JavaScript are not cached.
- **Offline navigation** — if the user navigates to a URL that has not been cached, the browser will show its default offline page. Pre-cache additional URLs manually if needed (open the cache with `caches.open(cacheName)` and call `cache.add()`).

For full service-worker capabilities, pair m3.js with a dedicated service worker library such as Workbox.

4.8 Next Steps

- ../api/js-api — Full JavaScript API reference.
- ../api/configuration — All `<meta>` configuration options.

CHAPTER

FIVE

JAVASCRIPT API REFERENCE

m3.js exposes a small global `window.M3` object with utilities you can call from your own scripts. Everything else is automatic and does not require any JavaScript from you.

On this page

5.1 Global Object — `window.M3`

After m3.js initialises (on `DOMContentLoaded` or immediately if the DOM is already ready), the following properties are available on `window.M3`:

Property	Description
`M3.snack()`	Show a snackbar / toast notification
`M3.badge()`	Update a badge counter on an element
`M3.installApp()`	Trigger the native PWA install prompt
`M3.cache`	Cache utility object (`init`, `clear`)
`M3.theme`	Theme controller object (`apply`, `toggle`, `current`)

5.2 M3.snack(message, actionLabel, actionCallback, duration)

Displays a Material You snackbar at the bottom of the screen.

Parameters

Parameter	Type	Description
`message`	`string`	The text to display in the snackbar. Required.
`actionLabel`	`string|null`	Label for the action button. Pass `null` for no button.
`actionCallback`	`function|null`	Called when the action button is clicked. Pass `null` if unused.
`duration`	`number`	Time in milliseconds before the snackbar auto-dismisses. Default: `4000`.

Examples

```
// Simple message (4 s auto-dismiss)
window.M3.snack('File saved.');

// With action button
window.M3.snack('Item deleted', 'Undo', () => restoreItem(), 6000);

// Long-running notification (10 s)
window.M3.snack('Uploading...', null, null, 10000);
```

Only one snackbar is shown at a time. Calling `snack()` while one is visible replaces the existing snackbar immediately.

5.3 M3.badge(selector, count)

Programmatically set a badge counter on any element.

Parameters

Parameter	Type	Description
`selector`	`string`	A CSS selector identifying the parent element (e.g. `'#nav-messages'`).
`count`	`number`	Badge value. Counts > 99 display as `99+`. `0` hides the badge.

Examples

```
// Show a badge with value 5 on the messages nav item
window.M3.badge('#nav-messages', 5);

// Hide the badge
window.M3.badge('#nav-messages', 0);

// Show "99+" for large counts
window.M3.badge('#notifications', 120);
```

The badge is created automatically the first time and re-used on subsequent calls. The parent element will have `position: relative` applied automatically.

5.4 M3.installApp()

Triggers the native browser PWA install prompt (the `beforeinstallprompt` dialog).

```
// e.g. wire to your own "Install" button
document.getElementById('install-btn').addEventListener('click', async () => {
  await window.M3.installApp();
});
```

This function is a no-op if:

- The browser has not fired `beforeinstallprompt` yet (criteria not met).
- The app has already been installed.
- The user previously dismissed the prompt in this session.

Note:

`beforeinstallprompt` is only fired on HTTPS origins (or `localhost` during development).

5.5 M3.cache

The cache utility object, exposing two async methods.

5.5.1 M3.cache.init()

Re-runs the asset-caching logic. Called automatically at startup; call manually if you add new resources after page load.

```
await window.M3.cache.init();
```

5.5.2 M3.cache.clear()

Deletes the entire cache bucket identified by the `m3-cache-name` meta value (default `m3-cache-v1`).

```
await window.M3.cache.clear();
// Users will re-download all assets on the next visit
```

5.6 M3.theme

The theme controller used internally by the `data-m3-theme-toggle` mechanism.

Properties

Property	Description
`M3.theme.current`	Current theme string: `'light'`, `'dark'`, or `'auto'`.

Methods

5.6.1 M3.theme.apply(theme)

Set the theme explicitly.

```
window.M3.theme.apply('dark');   // force dark
window.M3.theme.apply('light');  // force light
window.M3.theme.apply('auto');   // follow OS preference
```

The choice is persisted to `localStorage` under the key `m3-theme`.

5.6.2 M3.theme.toggle()

Toggle between `'light'` and `'dark'`. If the current theme is `'auto'` it switches to `'dark'`.

```
window.M3.theme.toggle();
```

5.7 Data-Attribute Hooks

m3.js listens for clicks on elements with these `data-*` attributes. No JavaScript is required on your part — just add the attribute to the HTML element.

Attribute	Behaviour
`data-m3-theme-toggle`	Toggles light/dark theme on click.
`data-m3-drawer-toggle`	Opens / closes the <nav> drawer.
`data-m3-dialog-open="<id>"`	Opens the <dialog id="..."> with the given ID.
`data-m3-dialog-close`	Closes the nearest ancestor <dialog>.
`data-m3-sheet-open="<id>"`	Opens the bottom sheet with the given ID.
`data-m3-sheet-close`	Closes the nearest ancestor bottom sheet.
`data-m3-scroll-top`	FAB / button that scrolls to the top of the page. Appears automatically after scrolling 200 px

Example

```
<!-- No JavaScript needed for any of these -->
<button data-m3-theme-toggle>Toggle Theme</button>
<button data-m3-drawer-toggle>Open Nav</button>
<button data-m3-dialog-open="settings-dialog">Settings</button>

<dialog id="settings-dialog">
  ...
  <button data-m3-dialog-close>Close</button>
</dialog>
```

5.8 Lifecycle & Boot Order

m3.js runs its `boot()` function in the following order when the DOM is ready:

1. `injectMeta()` — adds PWA meta tags
2. `injectManifest()` — creates and injects the Web App Manifest
3. `M3Cache.init()` — caches page assets
4. `M3Theme.init()` — restores saved theme preference
5. `initRipples()` — attaches ripple effect listeners
6. `initScrollHeader()` — scroll-aware top bar elevation
7. `initDialogs()` — wires dialog open/close data attributes
8. `initTabs()` — tab selection and indicator
9. `initFloatingLabels()` — floating label inputs
10. `initDrawer()` — nav drawer open/close
11. `initBottomSheets()` — bottom sheet open/close

12. `initInstallPrompt()` — PWA install prompt
13. `initScrollTop()` — scroll-to-top FAB visibility
14. `initActiveLinks()` — marks matching `<nav>` links as active
15. `initFocusRings()` — hides focus outlines for mouse users
16. `initDetails()` — animated `<details>` expand/collapse
17. `syncThemeColorMeta()` — keeps `<meta name="theme-color">` in sync
18. `initOfflineIndicator()` — online/offline snackbar
19. `initFormValidation()` — error styling for invalid inputs
20. `initPageTransitions()` — View Transitions API for navigation

The `window.M3` object (`snack`, `badge`) is available immediately on the global scope when the script is parsed, before `DOMContentLoaded`.

CHAPTER

SIX

CSS COMPONENTS

m3.css maps semantic HTML elements directly to Material Design 3 components. In most cases you don't need any special class — the element itself carries the styling. Additional classes act as modifiers to select a different variant.

On this page

6.1 Layout Structure

6.1.1 Top App Bar — <header>

```
<header>
  <button class="icon" data-m3-drawer-toggle>[menu]</button>
  <span class="title-large">App Title</span>
  <span class="flex-1"></span>
  <button class="icon">⋮</button>
</header>
```

The header is fixed to the top of the viewport and gains elevation shadow when the page is scrolled (handled by m3.js's `initScrollHeader`).

6.1.2 Navigation Drawer — <nav>

```
<nav>
  <a href="/">Home</a>
  <a href="/reports">Reports</a>
  <hr>                          <!-- section divider -->
  <a href="/settings">Settings</a>
</nav>
```

On mobile the drawer is hidden off-screen and opened by toggling `data-m3-drawer-toggle`. On wider viewports it is rendered inline as a Navigation Rail.

6.1.3 Page Content — <main>

```
<main>
  <!-- your page content -->
</main>
```

`<main>` fills the remaining space next to the nav drawer and adds comfortable padding.

6.2 Cards

6.2.1 Filled Card — <article>

The default card style with a tinted surface fill.

```
<article>
  <h3>Card Title</h3>
  <p>Card supporting text.</p>
</article>
```

6.2.2 Outlined Card — <aside>

A card with a visible border and no fill — used for less prominent content.

```
<aside>
  <h4>Note</h4>
  <p>Outlined card content.</p>
</aside>
```

6.3 Buttons

Four button variants are available:

Variant	HTML	Use case
Filled (default)	<button>	Primary action
Tonal	<button class="tonal">	Secondary action with colour
Outlined	<button class="outlined">	Secondary action, no fill
Text	<button class="text">	Low-emphasis action
Icon	<button class="icon">	Toolbar / compact icon actions
FAB	<button class="fab">	Floating action button

```
<button>Filled</button>
<button class="tonal">Tonal</button>
<button class="outlined">Outlined</button>
<button class="text">Text</button>
<button class="icon" aria-label="Menu">[menu]</button>
<button class="fab">+ Create</button>
```

All buttons receive the ripple effect from m3.js automatically.

6.4 Form Controls

6.4.1 Text Field — <input type="text">

Wrap an <input> in a <label> with class field to get a floating-label outlined text field:

```
<label class="field">
  <input type="text" placeholder=" ">
  <span>Email address</span>
</label>
```

The label floats above the input when it gains focus or has a value. This is handled by m3.js's initFloatingLabels.

6.4.2 Other Input Types

```
<!-- Checkbox -->
<input type="checkbox" id="cb1">
<label for="cb1">Remember me</label>

<!-- Radio -->
<input type="radio" name="opt" id="r1"> <label for="r1">Option A</label>
<input type="radio" name="opt" id="r2"> <label for="r2">Option B</label>

<!-- Switch (range-style toggle) -->
<input type="checkbox" class="switch" role="switch" id="sw1">
<label for="sw1">Enable notifications</label>

<!-- Slider -->
<input type="range" min="0" max="100" value="50">

<!-- Select -->
<select>
  <option>Option 1</option>
  <option>Option 2</option>
</select>
```

6.4.3 Textarea — <textarea>

```
<label class="field">
  <textarea placeholder=" " rows="4"></textarea>
  <span>Message</span>
</label>
```

6.5 Progress Indicators

6.5.1 Linear Progress — <progress>

```
<!-- Determinate (known value) -->
<progress value="60" max="100"></progress>

<!-- Indeterminate (unknown duration) -->
<progress></progress>
```

6.6 Dialog

Use the native <dialog> element. m3.js manages `showModal()` / `close()` for you.

```
<button data-m3-dialog-open="confirm-dialog">Delete item</button>

<dialog id="confirm-dialog">
  <h3>Delete item?</h3>
  <p>This action cannot be undone.</p>
  <div class="flex justify-end gap-2" style="margin-top:16px;">
    <button class="text" data-m3-dialog-close>Cancel</button>
    <button style="background:var(--md-error);color:var(--md-on-error)">Delete</button>
  </div>
</dialog>
```

6.7 Bottom Sheet

A bottom sheet slides up from the bottom of the screen. It is useful for contextual actions on mobile.

```
<button data-m3-sheet-open="my-sheet">Open sheet</button>

<div class="bottom-sheet" id="my-sheet">
  <div class="handle"></div>
  <h3>Sheet title</h3>
  <button class="text" data-m3-sheet-close>Close</button>
</div>
```

6.8 Tabs

```
<div class="tabs" data-m3-tabs>
  <button class="active" data-tab="tab1">Tab One</button>
  <button data-tab="tab2">Tab Two</button>
  <button data-tab="tab3">Tab Three</button>
</div>

<div id="tab1">Content for tab one.</div>
<div id="tab2" hidden>Content for tab two.</div>
<div id="tab3" hidden>Content for tab three.</div>
```

m3.js handles tab selection, indicator animation, and panel visibility.

6.9 Data Table

```
<table>
  <thead>
    <tr>
```

```
      <th>Name</th>
      <th>Status</th>
      <th>Date</th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td>Alice</td>
      <td>Active</td>
      <td>2024-01-15</td>
    </tr>
  </tbody>
</table>
```

6.10 List

```
<ul class="list">
  <li>Item one</li>
  <li>Item two</li>
  <li>Item three</li>
</ul>
```

6.11 Chips

```
<!-- Assist chip (default) -->
<div class="chip">Label</div>

<!-- Filter chip (toggleable) -->
<div class="chip filter selected">Filtered</div>

<!-- Input chip -->
<div class="chip input">Tag <span>×</span></div>
```

6.12 Badge

```
<!-- Small dot badge -->
<span class="badge"></span>

<!-- Large numbered badge -->
<span class="badge large">12</span>
```

For programmatic badge updates see `window.M3.badge()` in JavaScript API Reference.

6.13 Snackbar

Snackbars are generated dynamically by m3.js. See JavaScript API Reference for the API. You can also create static snackbars in HTML (they won't auto-dismiss):

```
<div class="snackbar">
  Message text
  <button>Action</button>
</div>
```

6.14 Divider

```
<div class="divider"></div>          <!-- full-width -->
<div class="divider inset"></div>    <!-- left-inset variant -->
<hr>                                 <!-- same as .divider -->
```

6.15 Tooltip

Add `data-tooltip` to any element:

```
<button data-tooltip="Opens the settings panel">[gear]</button>
```

The tooltip appears above the element on hover using a pure-CSS `::before` pseudo-element.

6.16 Expansion Panel — `<details>`

```
<details>
  <summary>Click to expand</summary>
  <p>This content is revealed with a smooth height animation, managed by m3.js.</p>
</details>
```

6.17 Utility Classes

m3.css ships a small set of utility classes to avoid inline styles for common patterns.

Colour utilities

```
<div class="primary">Primary surface</div>
<div class="primary-container">Primary container</div>
<div class="surface-container">Surface container</div>
<span class="text-primary">Primary text</span>
<span class="text-error">Error text</span>
```

Elevation

```
<div class="elev-1">Slight shadow</div>
<div class="elev-3">Medium shadow</div>
```

Shape

```
<div class="rounded-sm">Small corners</div>
<div class="rounded-full">Pill shape</div>
```

Flex layout

```
<div class="flex items-center justify-between gap-4">
  <span>Left</span>
  <span>Right</span>
</div>
```

Spacing (`p-*` = padding, `m-*` = margin, `gap-*` = gap)

```
<div class="p-4 m-2 gap-3 flex">...</div>
```

Responsive visibility

```
<div class="hide-mobile">Hidden on mobile (< 600px)</div>
<div class="hide-tablet">Hidden on tablet</div>
<div class="hide-desktop">Hidden on desktop</div>
```